TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Aplicando técnicas de Ingeniería del Software al desarrollo de Videojuegos: Ogame, un caso práctico.

Curso 2017/2018

Alumno/a:

Juan Alberto Llopis Expósito

Director/es:

Luis Fernando Iribarne Martínez

Resumen

El objetivo de este proyecto es el desarrollo de una metodología enfocada en la creación de videojuegos por parte de equipos pequeños y probada en un caso práctico, un juego de navegador basado en el videojuego online Ogame. Dicha metodología ayudará a desarrolladores Indies a crear productos de calidad, puesto que siendo ellos la base de la industria de los videojuegos en los últimos años, son los que más fracasan debido a la falta de metodologías en su desarrollo, causados principalmente por su falta de experiencia.

La metodología estará centrada en la división del trabajo en módulos, buscando mantener la documentación utilizando una metodología ágil e intentando que la carga de trabajo que genera sea la menor posible, al dividir el proyecto en módulos la documentación se haría mucho más rápido.

Palabras clave: Videojuegos, Patrones de diseño, Modelado y diseño de software, Pruebas de software.

Abstract

The main purpose of this project is the development of a methodology focused on the game development by small teams and tested in a practice case, a videogame like the online game Ogame. This methodology will help Indie developers to create quality software, since they are the videogame industry core in the last years, they are the ones that fails the most because of the lack of methodologies in the development, mainly due to the lack of experience.

The methodology will be focused on the division of work into modules, trying to keep the documentation using an agile methodology and the workload as low as possible, by dividing the project into modules the documentation would be much faster.

Keywords: Videogames, Design patterns, Software modeling and design, Software testing.

Índice

Índice de figuras

Índice de tablas

1. Introducción

La industria del videojuego no deja de crecer desde 2006, recibiendo un impulso del sector móvil cada año debido a su constante crecimiento [1]. En 2017 la industria de los videojuegos generó 116.000 millones de dólares a nivel mundial, y se prevé que esta cifra vaya en aumento hasta 2020 donde generaría 143.000 millones (véase Figura 1). De esta cifra el 44% constituye el mercado móvil, el cual se prevé que para 2020 suponga más del 50% de los ingresos totales de la industria de los videojuegos [2].

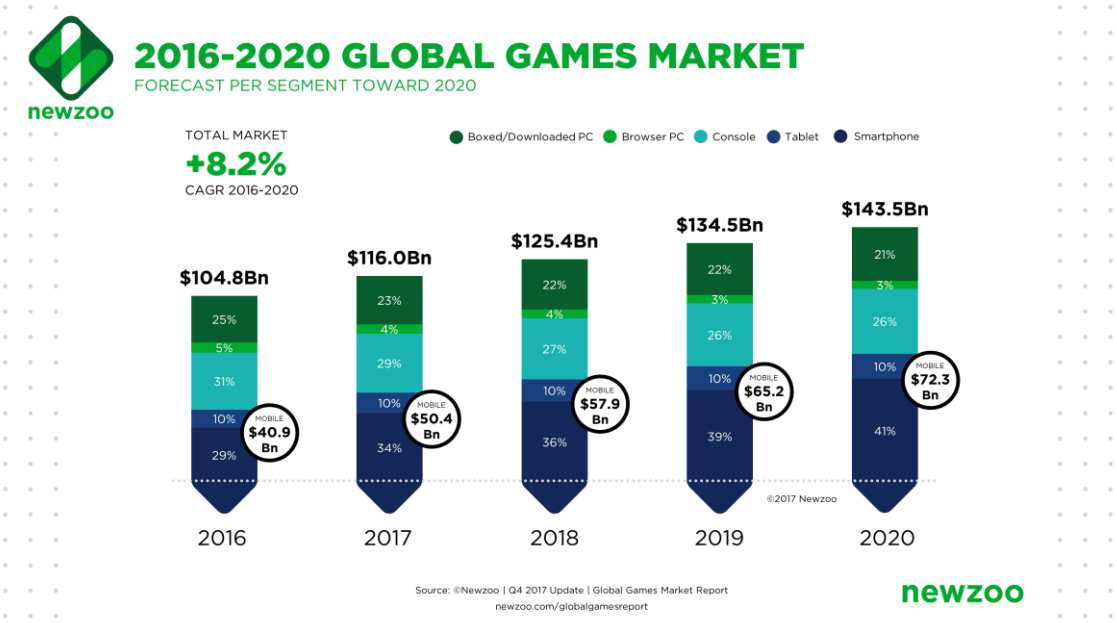

Figura 1. Ingresos de la industria de videojuegos por año. [2]

En España, la venta de videojuegos alcanzó en 2017 los 1.900 millones de dólares, habiendo alcanzado en 2016 los 1.177 millones de dólares y en 2015 1.083 millones, a pesar de esta notable subida aún siguen siendo cifras muy alejadas del resto de países europeos [3].

Como muestra la Figura 2, los principales países son China seguida de Estados Unidos, siendo China el país que lidera esta industria gracias principalmente al sector móvil, el cual conforma más del 50% de los beneficios que obtienen [2].

IMAGE	RANK	COUNTRY	POPULATION	INTERNET POPULATION	TOTAL REVENUES IN US DOLLARS
	1	China	1,410 M	814 M	32,536 M
	2	United States of America	324 M	260 M	25,426 M
	3	Japan	127 M	121 M	14,048 M
	4	Germany	82 M	74 M	4,430 M
	5	United Kingdom	66 M	62 M	4,238 M
	6	Republic of Korea	51 M	47 M	4,203 M
	7	France	65 M	57 M	2,977 M
	8	Canada	37 M	33 M	1,968 M
	9	Spain	46 M	39 M	1,918 M
	10	Italy	59 M	43 M	1,881 M

Figura 2. Ingresos de la industria de videojuegos por país. [2]

Mientras tanto España se encuentra en mitad del crecimiento de esta industria, en los últimos años el número de empresas ha aumentado. En 2014 eran 330 empresas las que formaban el sector, mientras que en 2017 ya había 450 empresas. A pesar de esta aparente subida, en 2016 en España había 480 empresas, por lo que de 2016 a 2017 desaparecieron 30 empresas y se espera que desaparezcan más por su inactividad [4].

Esto es debido a la aparición de gran cantidad de microempresas y la falta de editores españoles, actualmente un 47% de las empresas la conforman menos de 5 empleados (véase Figura 3) y un 68% son microempresas, siendo un 52% de las empresas jóvenes (llevan menos de 5 años formadas) [4].

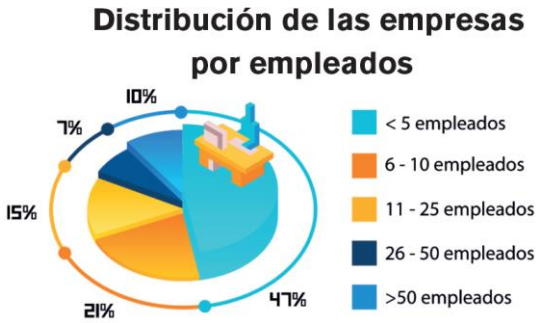

Figura 3. Estadística del número de empleados por empresa de videojuegos. [4]

Tras el gran crecimiento y la falta de posibilidades en España es normal la desaparición de empresas, ya que actualmente no hay posibilidad de mercado para todas ellas, muchos de los profesionales se tienen que ir fuera de España debido a esto [4].

El empleo también ha presentado una subida, de 4.660 profesionales que conformaban la industria en 2016 pasaron a formarla 6.778 profesionales en 2017, cada vez hay más profesionales en el sector, pero no más puestos de trabajo disponibles [4].

En resumen, desde hace varios años España está sufriendo un gran crecimiento de esta industria, eclipsada por la falta de posibilidades para aquellos que quieren iniciarse en el sector, obligándoles a irse fuera del país o formar su propia microempresa y arriesgar su poder económico debido a la falta de editores que les apoyen.

Como hemos visto, el número de empresas Indies está creciendo continuamente intentando hacerse un hueco en el mercado dominado por las grandes empresas [5]. Pero ¿qué significa realmente empresa Indie?

Distintos autores le dan definiciones diferentes, unos optan por la definición de independiente económicamente y otros por la definición de innovadores [6].

Según Robert Boyd, cofundador de Zeboyd Games: “Un desarrollador Indie es una persona o un grupo pequeño no perteneciente a ninguna compañía que hace juegos, un juego Indie es un juego hecho por un desarrollador Indie” [7].

Mientras que según Kellee Santiago, cofundadora de TGC: “Indie es cuando innovas en la manera de hacer juegos” [7].

Según la RAE, independiente es alguien que no depende de otro, autónomo, alguien cuya opinión no está intervenida por otra persona [8].

El editor o Publisher es el cliente, quien dirige el proyecto, el que aporta el dinero y dice qué se hace, por lo que una empresa independiente sería aquella que no posee esta figura.

Esto no significa que definiciones como la de Kellee Santiago no sean verdad, los desarrolladores independientes suelen innovar cuando desarrollan, pero no por eso son independientes, innovan por la simple razón de que no pueden competir con las grandes empresas, no tienen el mismo presupuesto debido a la falta de editor, por lo que tienen que intentar innovar para hacerse un hueco en el mercado.

También algunos desarrolladores utilizan sus recursos para desarrollar como ellos quieren sin que nadie les retenga [9], sí, están innovando, pero en este caso también serían independientes por no estar controlados por un editor, no por innovar.

Como dijo Marco Massarutto, productor en Kunos Simulazioni: “Ser Indie significa que el desarrollador decide qué hace, cómo lo hace y cuándo lo hace sin la influencia de un editor o de inversor externo.” [10]

El gran crecimiento de empresas independientes en España según esta definición sería debido a la falta de editores españoles. Al no haber suficientes editores, muchas empresas carecen de esta figura, no porque no quieren tenerlo, sino porque directamente no pueden, pasando a ser empresas independientes.

Como se ve en la Figura 4, una de las razones más importantes del fracaso de muchas empresas Indies es debido a la falta de una buena gestión del proyecto. Debido a la falta de experiencia afrontan el desarrollo sin una metodología bien estructurada, confiando en que no es necesaria al ser una empresa pequeña [11].

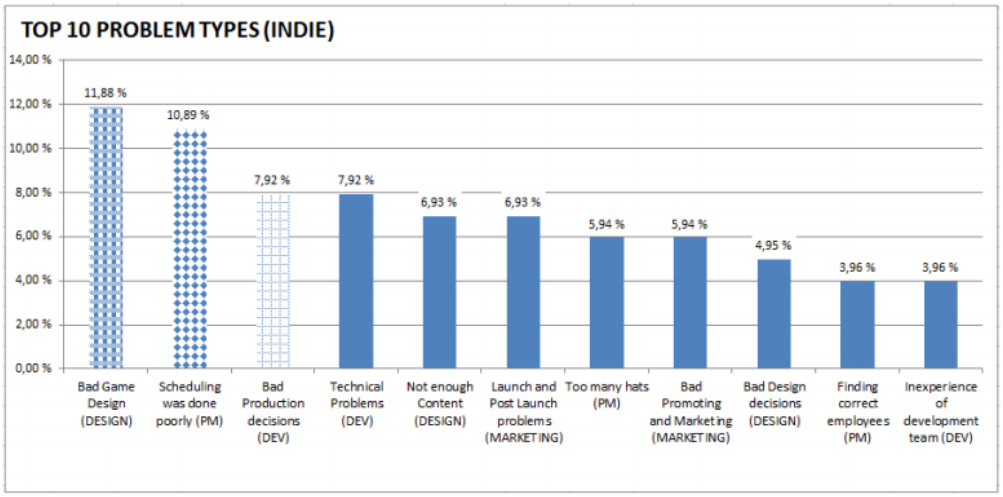

Figura 4. Estadística acerca de los problemas que surgen en empresas indies. [11]

A pesar de ser proyectos más pequeños que los de las grandes empresas es necesaria una buena metodología de desarrollo debido a lo volátil que es el mercado, continuamente van cambiando los requisitos, añadiéndose nuevos según se va avanzando en el proceso de desarrollo [12].

Las grandes empresas utilizan metodologías ágiles adaptadas a sus necesidades, esto adicionalmente a la experiencia que poseen del sector provoca que sus proyectos no fracasen completamente como en el caso de las empresas Indies [13].

A vista del estado actual de las empresas Indies se propone este proyecto, el desarrollo de una metodología enfocada en la creación de videojuegos por parte de equipos pequeños y probada en un caso práctico, un juego de navegador basado en el videojuego online Ogame [14], buscando no cometer el error de estudiar una metodología en casos no ideales [15] y ayudar a desarrolladores Indies a crear productos de calidad, puesto que como se ha comprobado, siendo ellos la base de la industria de los videojuegos en los últimos años [16], son los que más fracasan debido a la falta de metodologías en su desarrollo [17], causados principalmente por su falta de experiencia.

1.1. Metodologías existentes

Durante el desarrollo de un videojuego los cambios en los requisitos son ineludibles, ya sea por cambios en la industria, porque algo no salió como se esperaba o simplemente porque al ver el juego en acción surgen nuevas ideas. Los métodos ágiles permiten gestionar estos cambios con un mínimo de esfuerzo, respondiendo a los cambios de manera mucho más rápida que los métodos tradicionales [18].

Según se va avanzando en el desarrollo, los requisitos del producto son más claros, disminuyendo el número de cambios que se deben de realizar, pero al principio todo es muy difuso, por ejemplo, vas a desarrollar un shooter, tienes claro que el usuario dispara, la bala golpea al rival y éste se muere. Cuando estás implementando la tarea de disparar te das cuenta de que la bala podría tener gravedad, haces un cambio y lo implementas, pero entonces descubres que sin gravedad o con menos gravedad, aunque no es tan realista es más divertido. Este ejemplo ocurre en numerosas ocasiones durante el desarrollo de un videojuego, aunque también es cierto que ocurre porque no se hace un análisis de requisitos bien detallado, aun así, las personas cometemos errores y se nos puede llegar a olvidar algo [19].

La metodología ágil al contrario que la tradicional no penaliza los errores, te permite enmendarte, buscar una solución, arreglarlo y aprender de ello. En el caso de la metodología en cascada esos cambios o errores habría que arrastrarlos hasta el final del proyecto, como una bola de nieve rodando por una montaña, o hacer un esfuerzo titánico e intentar modificar el requisito junto a toda su documentación [20].

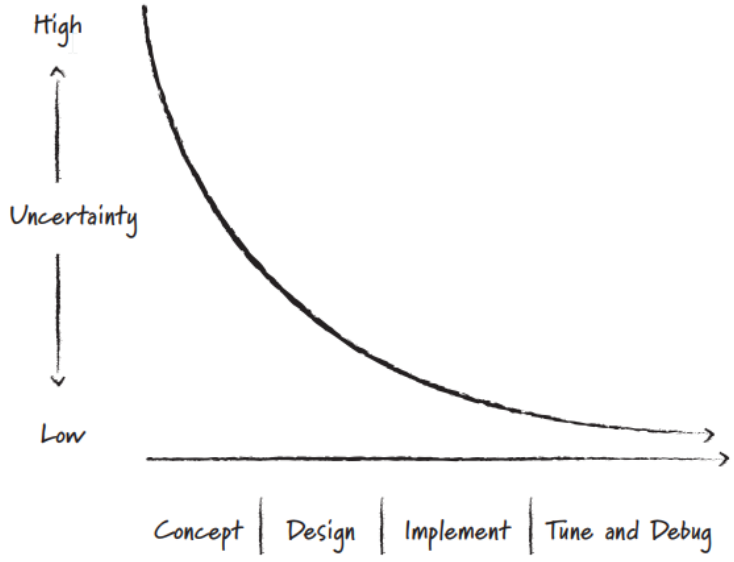

Figura 5. Gráfica acerca de la incertidumbre durante el desarrollo. [19]

Por esta razón es que las empresas han elegido utilizar la metodología ágil, aunque adaptándola a sus necesidades, ya que el equipo de desarrollo está formado además de por programadores por personal que nada tiene que ver con el desarrollo de software (véase Figura 6) [21].

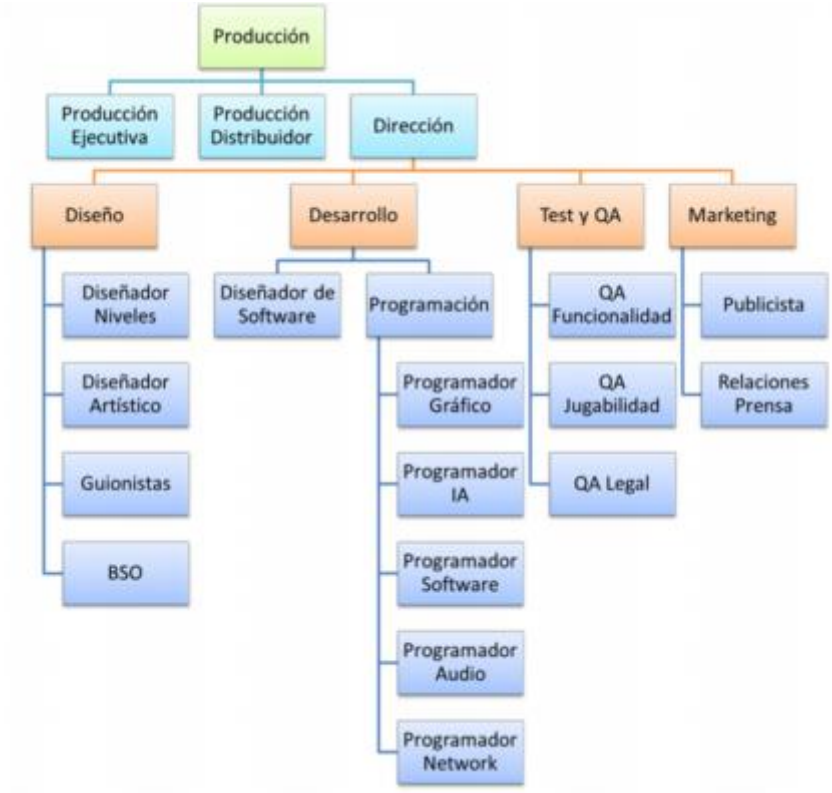

Figura 6. Organigrama desarrollo de videojuegos. [21]

1.1.1. Extreme Programming

Extreme Programming (XP) es una metodología ágil, iterativa e incremental para el desarrollo de software.

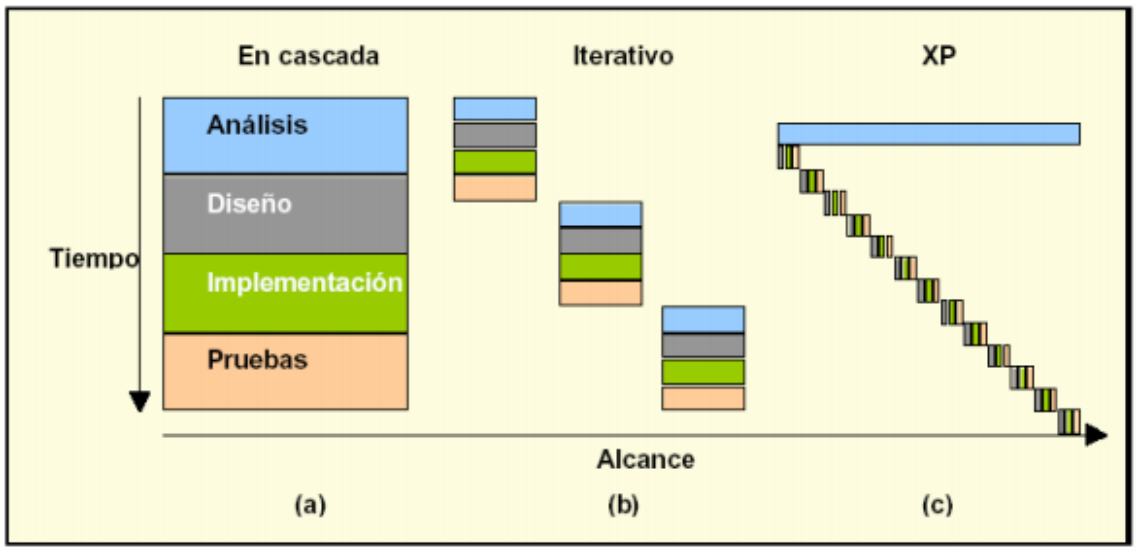

Figura 7. Comparativa entre metodología en cascada, iterativa y Extreme Programming. [22]

Está basada en la interacción continua con el cliente, el cual, durante todo el desarrollo del producto trabaja junto al equipo, identificando las tareas, resolviendo dudas y realizando las pruebas de aceptación [23].

XP posee 5 valores que deben seguirse durante el desarrollo del producto además de varios principios [24], estos valores son:

- **Comunicación**: La comunicación es importante para la cooperación y resolución de problemas, puede surgir un problema que alguien con más experiencia sepa resolver, o con el simple hecho de discutirlo con un compañero puedes encontrar la solución.
- **Simplicidad**: No hacer más de lo que se tiene que hacer intentando hacer todo lo más simple posible, esto además de reducir el esfuerzo facilita la comunicación entre el equipo, puesto que cuanto más simple, más fácil es de entender.
- **Feedback**: Durante todo el proceso de desarrollo se obtiene *feedback* acerca del producto, ya sea por parte del cliente o por parte de los propios desarrolladores.
- **Coraje**: Si surge un problema y no se puede identificar la causa, tener el coraje de esperar a identificarlo, coraje a trabajar en equipo, a realizar cambios, a todo eso se refiere el valor de coraje.
- **Respeto**: El valor más fundamental, se debe de respetar tanto el proyecto como los compañeros de equipo y el cliente. Si no hay respeto entonces esta metodología no funcionará.

XP recomienda añadir más valores a los que propone, según la empresa donde se implante.

Además de estos valores XP propone una serie de principios [25], los más importantes son:

- **Posesión del código**: XP establece que todos son dueños del código, el código no es de quien lo escribe, sino del equipo de desarrollo.
- **Integración**: La integración es continua durante todo el proyecto, cada vez que se termina algo se integra con el resto del proyecto.
- **Espacio de trabajo**: El espacio de trabajo debe de ser abierto, pero que cada trabajador posea una pequeña zona privada.
- **Lanzamiento del producto**: El lanzamiento debe de ser a lo largo de todo el proyecto en pequeñas *releases*.
- **Estándar del código**: Todo el equipo de desarrollo comparte el mismo estilo de programación y se ciñe a él.
- **Semanas de 40 horas**: Este principio no significa trabajar 40 horas semanales, el significado es poner un límite de horas semanales, de manera que si se va mal de tiempo nunca se puedan superar esas horas fijadas.

XP está formado por tres fases principales [25], estas fases son:

- **Fase de planificación**: El usuario escribe historias de usuario con título, descripción y las pruebas que tendrá asociadas. Después los programadores estiman tiempo y coste teniendo en cuenta que una historia de usuario no puede tener más de 3 puntos (cada punto es una semana). El usuario ordena las historias de usuario por prioridad, los programadores estiman los puntos que harán esa iteración y el cliente selecciona las historias que se realizarán en la iteración. Por último, las historias de usuario se dividen en tareas que son escogidas por los desarrolladores
- **Fase de iteración**: Se programa por parejas, uno codifica y el otro va aportando ideas y ayudando, en cualquier momento se pueden cambiar los puestos. Primero se realizan las pruebas, luego la implementación asociada a ellas y por último se refactoriza si fuera necesario. Mientras tanto el cliente va desarrollando pruebas de aceptación, generalmente mediante el uso de hojas de cálculo. Hay que tener en cuenta que es incremental, no se construye una clase de golpe, sino que se va construyendo poco a poco de manera iterativa.
- **Fase de lanzamiento**: Una vez se ha terminado de iterar y no hay más cambios que realizar el producto se lanza, los desarrolladores lo instalan y el cliente lo prueba.

Por último, en el equipo de desarrollo hay siete roles presentes [26]:

- **Programador**: Se encarga de la implementación y las pruebas unitarias.
- **Cliente**: Establece las historias de usuario a hacer y en qué orden se hace, además realiza las pruebas de aceptación.
- **Encargado de las pruebas**: Ayuda al cliente a realizar las pruebas de aceptación, además proporciona el resultado de las pruebas a los miembros del equipo.
- **Gestor**: Se encarga de los problemas externos, de formar el equipo y gestionarlo, así como tratar los problemas que surjan.
- **Encargado del seguimiento**: Su función es seguir el plan de lanzamiento, las iteraciones y las pruebas de aceptación.
- **Entrenador**: Monitoriza los procesos realizando cambios cuando sea necesario, además sirve como mentor del grupo.
- **Consultor**: Es un miembro externo al equipo, encargado de un tema específico necesario para el proyecto.

1.1.2. SCRUM

SCRUM al igual que XP es una metodología ágil, iterativa e incremental, en cada iteración se va desarrollando un poco más del producto.

Hay tres roles fundamentales [27]:

- **Product Owner**: Es la persona encargada de establecer la comunicación entre el equipo de desarrollo y los stakeholders, representándolos. Él decide cómo será el resultado final, así como la prioridad de las distintas historias de usuario, si el proyecto se debe cancelar o no lo decide él.
- **Scrum Máster**: Su trabajo es aislar al equipo de desarrollo de interferencias externas y asegurarse de que se siga la metodología SCRUM correctamente. Es un líder servil, sirve y ayuda al equipo, no da órdenes.
- **Equipo de desarrollo**: Son los encargados de realizar los incrementos de cada sprint, en todos ellos recae la responsabilidad. Todos deben conocer SCRUM y tener clara la visión del producto, son un equipo, el proyecto, así como sus tareas, son responsabilidad de todos, no solo de quien la realiza.

Antes de comenzar con las fases de SCRUM se debe de establecer la visión del producto y redactar las historias de usuario. El Product Owner es el encargado de realizar estas tareas.

Una vez se tienen las historias de usuario y una visión clara del producto a desarrollar se comienzan con las fases de SCRUM [28].

- **Sprint planning**: Su duración máxima es de un día de trabajo, asisten todos los miembros del equipo y aquellos que puedan aportar información.
  El Product Owner presenta las historias de usuario ordenadas por prioridad y el equipo de desarrollo le plantea preguntas según las dudas que tengan. El equipo establece un objetivo para el sprint y desglosa las historias de usuario en tareas para formar la pila del sprint. Por último, cada desarrollador se asigna tareas de la pila teniendo en cuenta la duración establecida para el sprint y la estimación de tareas que pueden realizar.
- **Reunión diaria**: Cada día el equipo se reúne no más de 15 minutos de manera informal y cada desarrollador comenta acerca de lo que hizo el día anterior, lo que va a hacer el día de hoy y que problemas tuvo.
- **Revisión del sprint**: Se realiza al finalizar el sprint, se presenta el producto en su estado actual al Product Owner, que decide qué historias de usuario están hechas y cuáles no, además de proporcionar *feedback* al equipo de desarrollo.
- **Retrospectiva del sprint**: Esta fase es externa al sprint y va después de la revisión, el equipo habla acerca de lo que ha ido mal y bien y sobre lo que se puede cambiar para hacerlo mejor.

Según se van realizando Sprints el equipo de desarrollo va ajustándose a su velocidad de trabajo gracias a las gráficas burndown, si se realizan menos tareas de las que se planificaron en el siguiente sprint se cogen menos, mientras que si durante el sprint se ve falta de trabajo (sobra tiempo) entonces en la siguiente iteración se cogerán más tareas [29].

Esto además del *feedback* obtenido sobre el producto permite al equipo adaptarse y mejorar a lo largo del desarrollo.

1.1.3. Rational Unified Process

Rational Unified Process también llamado RUP es una metodología iterativa e incremental dirigida por casos de uso, en cada iteración se realiza un poco del producto. Las primeras iteraciones se centran en la documentación y la arquitectura del software intentando establecer la base sobre la que se sostendrá el producto, mientras que iteraciones posteriores se centran más en implementación y pruebas [30].

La arquitectura conforma las partes más importantes del software, así como sus interfaces. Forma entre un 10% y un 20% del porcentaje final de la aplicación, es su esqueleto, por eso se realiza en las primeras etapas del proyecto [30].

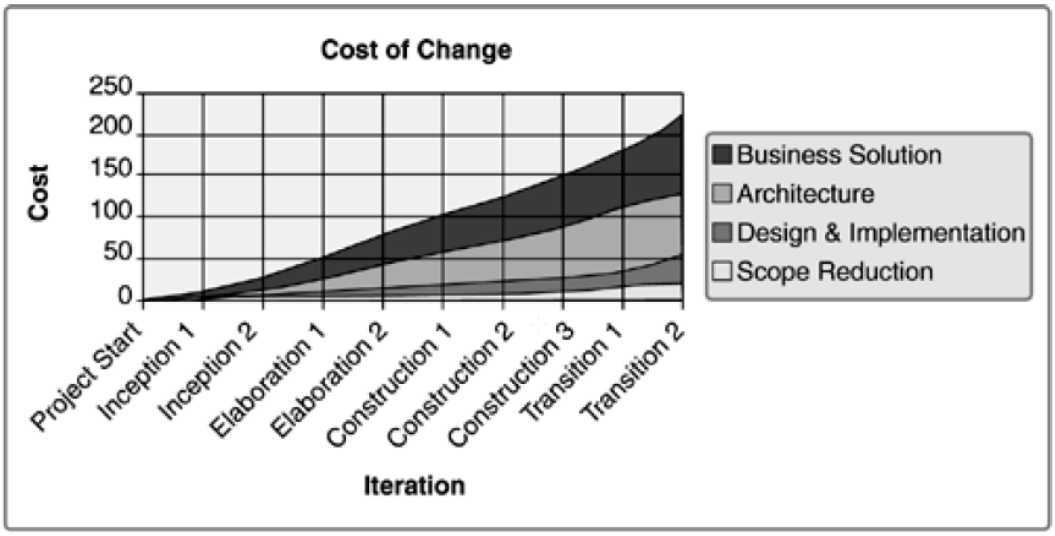

Figura 8. Gráfica del coste de cambios a lo largo del proceso de desarrollo RUP. [30]

Al igual que XP posee una serie de principios [30], estos son:

- Encargarse de los riesgos lo más pronto posible para evitar que luego ocasionen problemas.
- Entregables con valor al cliente, la documentación tiene que estar realizada de manera que no solo el cliente la entienda, sino que también sirva para el desarrollo.
- Lo que de verdad mide el progreso son los entregables software, no la documentación.
- Facilitar en la medida de lo posible la modificación de requisitos.
- Establecer al comienzo del proyecto una arquitectura estable con el objetivo de mitigar riesgos.
- Construir la aplicación mediante el uso de componentes para reducir el coste de mantenimiento y facilitar la reutilización.
- Trabajar como un equipo, no como un grupo, los problemas que surjan son responsabilidad de todos y no solo de individuos.
- Realizar software de calidad.

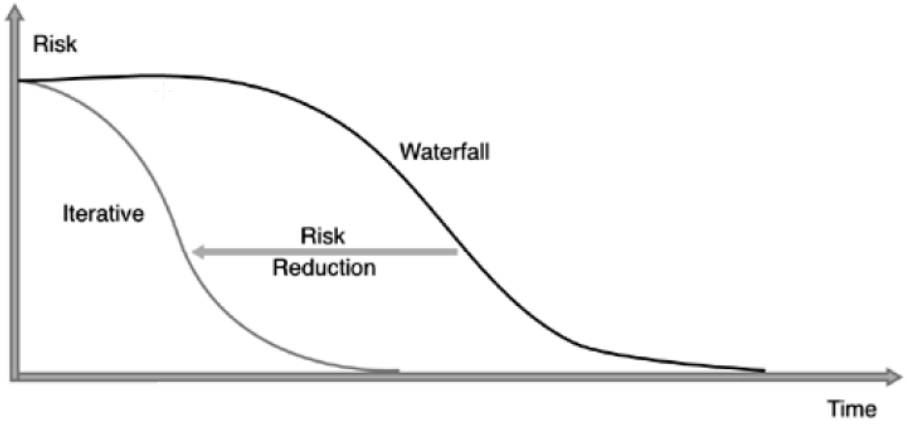

Figura 9. Gráfica comparativa del riesgo en función del tiempo entre las metodologías iterativa y en cascada. [30]

RUP está formado por una estructura dinámica donde se indican las iteraciones y una estructura estática donde se establecen los procesos [31].

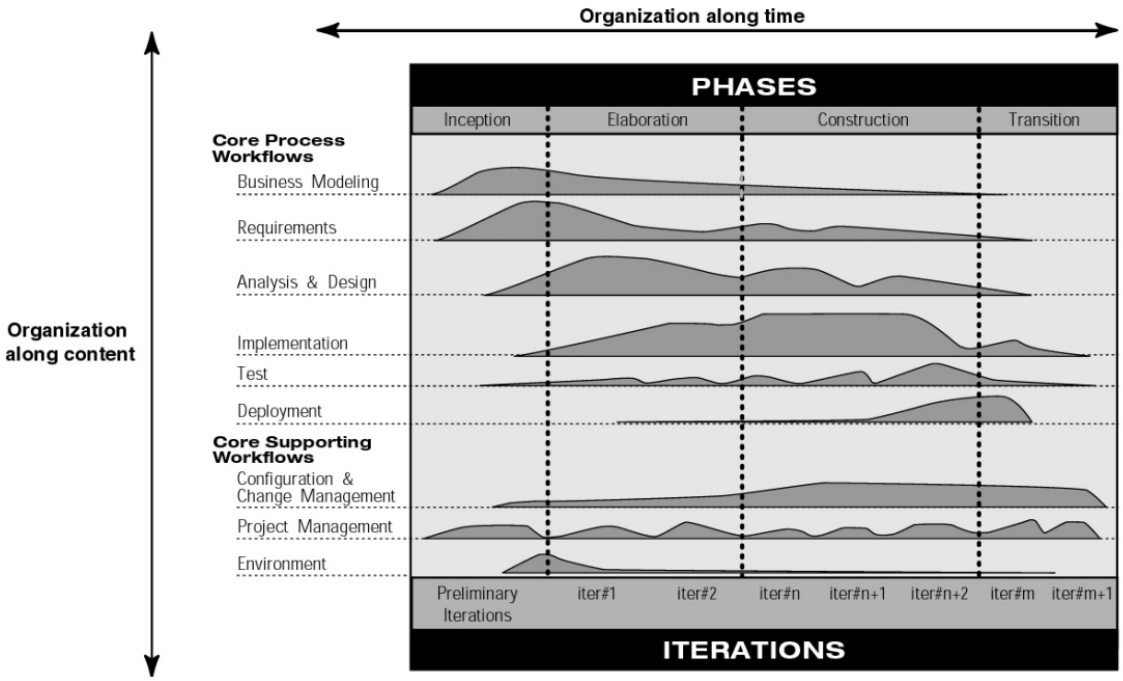

Figura 10. Fases que conforman la metodología RUP. [31]

La estructura dinámica está formada por tantas iteraciones como sean necesarias para alcanzar el objetivo propuesto, pero no más de esas para evitar que el proyecto se vaya retrasando.

Las fases de la estructura dinámica son:

- **Comienzo**: Entender el problema y establecer un objetivo. Mitigar riesgos, producir los casos de negocio y obtener el permiso de los stakeholders.
- **Elaboración**: Engloba muchas tareas difíciles como diseño, implementación y pruebas de la arquitectura del sistema; así como los riesgos asociados.
- **Construcción**: Fase en la que se pasa de una arquitectura funcional a la primera versión del producto, al final de la fase se despliega una versión del sistema completamente funcional.
- **Transición**: Confirmar que el software tiene todo lo que necesita el usuario. Se realizan las pruebas con el objetivo de preparar el producto para su entrega, así como ajustes obtenidos por el *feedback*.

La estructura estática la forman los artefactos que se realizan en cada una de las fases de la estructura dinámica: diagramas de casos de uso o de diseño, documentación como la visión del producto, código fuente, ejecutable, etc. Un artefacto puede estar documentado formalmente mediante herramientas o informalmente, por ejemplo, mediante correos [32].

Además de estas metodologías existen muchas otras, pero son demasiadas como para explicarlas todas, algunas de ellas son: Lean software development, las metodologías Crystal y el desarrollo basado en funcionalidades [33].

1.1.4. Otras metodologías

Además de las metodologías ágiles ya explicadas existen otras, en este apartado se explicará brevemente la metodología en cascada y varias metodologías creadas expresamente para el desarrollo de videojuegos a partir de modificaciones de otras metodologías.

1.1.4.1. Modelo en cascada

Se trata de un modelo basado en la documentación, debe de hacerse gran cantidad de documentación al comienzo del proyecto, ya que para su desarrollo es necesario tener los requisitos bien especificados por su incapacidad de volver a fases anteriores sin realizar un gran esfuerzo y coste [34].

Es recomendado para realizar proyectos cortos, mantener software o portarlo debido a la necesidad de requisitos bien especificados, ya que los problemas que surjan deben arrastrarse hasta fases finales donde ya se pueden resolver, en el caso de proyectos grandes, por ejemplo, dos años, entonces al final del proyecto habría acumulados dos años de problemas [35].

1.1.4.2. Extreme Game Development

Se trata de una adaptación de XP al desarrollo de videojuegos, una metodología ágil [36].

En la primera iteración ya se entrega una versión funcional que se va mejorando en cada iteración por lo que es iterativo e incremental.

Las diferencias con XP son la figura del Publisher, la adaptación a la gran cantidad de roles que presenta el desarrollo de un videojuego y el uso de UML para el diseño del juego.

La figura del Publisher sería similar a la del cliente y realizaría la misma labor, indicar al equipo qué es lo que quiere y realizar cambios según se vaya desarrollando el proyecto.

No se aporta mucha más información acerca de las diferencias con XP.

1.1.4.3. SUM

Se trata de una metodología basada en SCRUM y XP, y adaptada al desarrollo de videojuegos [13].

SUM está formada por cuatro roles:

- **Productor interno**: Similar al Scrum Máster.
- **Cliente**: Correspondería con el Product Owner.
- **Equipo de desarrollo**: Sería el equipo de SCRUM, pero formado por distintos subroles dentro del equipo para adaptarlo a las características del desarrollo de videojuegos.
- **Verificador beta**: Responsable de realizar la verificación funcional del videojuego.

SUM está formado por 6 fases (véase Figura 11) donde 5 de ellas se ejecutan de manera secuencial y una de ellas, la gestión de riesgos se ejecuta durante todo el proyecto. Estas fases son:

- **Concepto**: Fase en la que se elabora la especificación del proyecto, cuál es su objetivo, a que público va dirigido, que herramientas se van a utilizar, etc.
  Si la idea es válida y aceptada por todos se pasa a la siguiente fase.
- **Planificación**: En esta fase se realizaría la planificación del proyecto, aunque flexible ya que es una metodología ágil, y el análisis del proyecto, identificando los requisitos funcionales y no funcionales.
- **Elaboración**: Esta fase es iterativa e incremental, se identifican las tareas a realizar, se desarrollan las tareas mientras se comprueba que se mantiene el objetivo del producto y por último se recibe retroalimentación acerca de cómo ha ido el proceso de implementación, no acerca del producto desarrollado.
- **Beta**: Esta fase también sería iterativa, se libera una versión beta del juego para obtener *feedback* de cómo está desarrollado, según ese *feedback* obtenido se realizan los ajustes necesarios y se vuelve a lanzar otra versión beta. Cuando se alcanza el criterio de finalización establecido esta fase acaba y se pasa a la siguiente.
- **Cierre**: Se entrega el producto software y se analiza que salió bien y mal a lo largo del proyecto para hacerlo mejor la próxima vez.
- **Gestión de riesgos**: En esta fase ejecutada a lo largo del proyecto se identifican los riesgos y se establece la probabilidad de ocurrencia y el impacto que supondría, cuando los riesgos están identificados se procede a monitorizarlos para actuar rápidamente en caso de que ocurran.

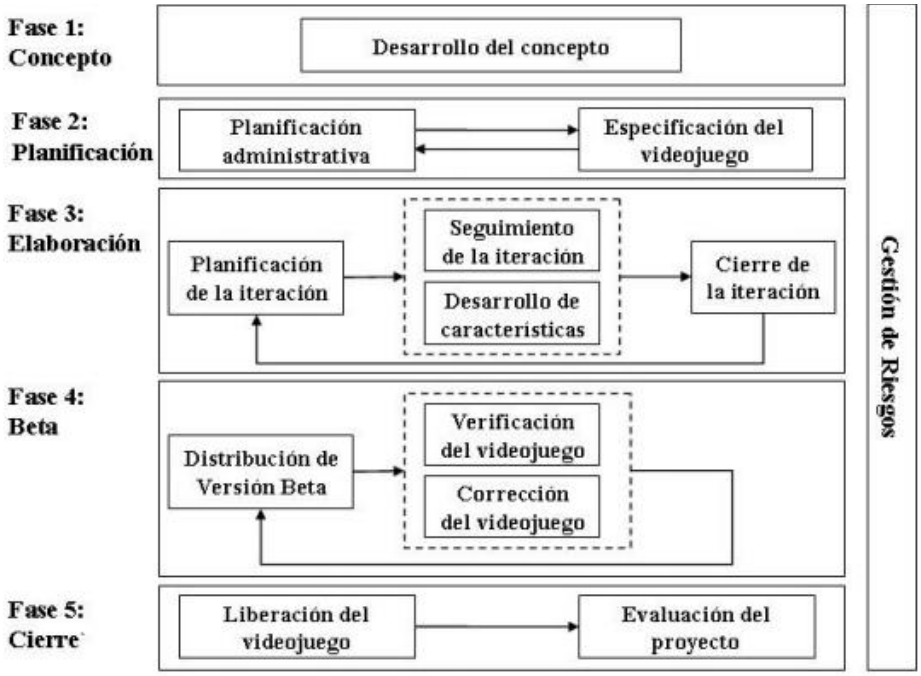

Figura 11. Fases de la metodología SUM. [13]

1.1.4.4. Game-Scrum

Game-Scrum se trata de una metodología basada en SCRUM y XP, unifica la gestión de proyectos de SCRUM con la gestión técnica de XP [37].

Esta metodología se divide en tres fases:

- **Pre-producción**: En esta fase se identifican los objetivos del videojuego y su factor de diversión (qué es lo que lo hace divertido), intentando reducir la búsqueda de estos elementos en la fase de producción. Para ayudar a identificarlos se realizan tormentas de ideas y prototipos.
  Por último, se documenta el diseño del juego, los elementos que lo conforman y cómo interactúan entre ellos. Este documento dará lugar al product backlog en la fase de producción.
- **Producción**: El documento del diseño del juego se pasa al product backlog, dividiéndose en cada iteración los elementos más importantes en tareas más pequeñas. La implementación se realiza siguiendo las prácticas de SCRUM y de XP.
- **Post-producción**: Se realiza una retroalimentación mediante el desarrollo de un postmortem, con el objetivo de identificar las fortalezas y debilidades del proceso de producción.

De esta metodología se destaca la falta de información acerca de la fase de producción, una fase que debería estar explicada al detalle.

Otras metodologías desarrolladas para facilitar el desarrollo de los videojuegos son: CASCRUM [15], DAV [17] y Game Unified Process [38].

2. Propuesta metodológica

A pesar de la gran cantidad de metodologías existentes como ya se ha visto, en el caso de que se utilice alguna es una metodología ágil. Esto es debido a dos razones, por la falta de conocimientos sobre la ingeniería del software y por la forma de trabajar de los equipos para desarrollar este software.

No son solo programadores o analistas los encargados de desarrollar el software, también hay personal enfocado al diseño, historia, sonido, etc., haciendo difícil el implementar las metodologías existentes puesto que están centradas en equipos de programadores. Además, se opta por las metodologías ágiles ya que es necesario un *feedback* constante por la naturaleza cambiante de los videojuegos, hoy pueden ser populares los juegos de disparos, pero mañana pueden pasar a ser los juegos de puzles, es algo que sin una retroalimentación continua puede llevar el proyecto al fracaso.

Por esta razón, las grandes empresas no utilizan solo metodologías ágiles, sino que las utilizan adaptándolas a ellos para suplir la carencia que estas presentan, aunque sigue habiendo una falta de metodologías para la primera razón, la falta de conocimiento.

Por ello, presento una metodología adaptada al desarrollo de videojuegos especializada en empresas Indies, basada en distintas metodologías ágiles (Extreme Programming, Rational Unified Process, SCRUM, etc.) y adaptada a las necesidades de una empresa de videojuegos pequeña (menos de 5 personas) que quiere abrirse paso en la industria.

Para el desarrollo de esta metodología se han utilizado las buenas prácticas y principios de XP, la ventaja de la documentación, los diagramas UML y la división de RUP, las fases y también la documentación del modelo en cascada y la retroalimentación continua de SCRUM (también presente en XP).

La diferencia fundamental con estas metodologías es que en este caso el único rol es el desarrollador, puesto que el cliente es un rol externo (por ejemplo, niños) siendo entonces el propio desarrollador quien pone las fechas en función del presupuesto disponible.

Durante el desarrollo de la metodología se pensó como mantener la documentación (análisis, diseño y estudio del problema) pero siendo ágil, pudiendo realizar cambios con un mínimo esfuerzo. Por ello se optó por dividir la implementación en módulos, si no se hubiera hecho de esta manera habría llevado a terminar realizando un ciclo de vida del salmón [34], siendo muy costoso el realizar cambios en fases previas y provocando la creación de una metodología ágil falsa ya que no sería fácil realizar cambios en los requisitos.

Figura 12. Ciclo de vida del salmón. [34]

Esta metodología se llamaría Modular Split (MS) y consistiría en dividir el proyecto en módulos de entre 1 y 2 semanas sobre los que se iteraría. Se ha hecho de esta forma por una simple razón, se ha buscado mantener la documentación utilizando una metodología ágil intentando que la carga de trabajo que genera sea la menor posible, al dividir el proyecto en módulos la documentación se haría mucho más rápidamente.

Cada módulo estaría formado por un análisis, diseño, implementación y por último la obtención de *feedback*.

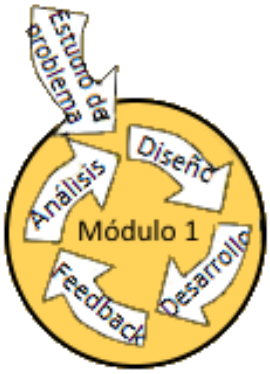

Figura 13. Ciclo de vida de un módulo la metodología propuesta. (Elaboración propia)

Tras terminar todos los módulos habría una fase de integración donde al final se recibiría un *feedback* y en función de este *feedback* se iteraría otra vez o se pasaría a la fase final de lanzamiento del producto.

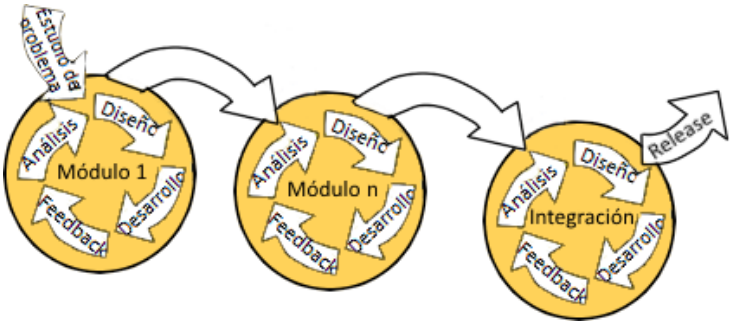

Figura 14. Ciclo de vida de la metodología propuesta. (Elaboración propia)

Inspirado en Extreme Programming, esta metodología presenta una serie de principios que se deben de seguir durante todo el desarrollo:

Refactorización: Todos, cuando éramos pequeños hemos jugado con tierra y nos hemos ensuciado, llenándonos de tierra por todos lados obligándonos después a ducharnos. Es más divertido jugar con la tierra que ducharse, pero cuando lo haces te das cuenta de que valió la pena y te sientes mejor. Lo mismo pasa con la refactorización, es más divertido programar las cosas directamente, sin preocuparte por la legibilidad ni la optimización, pero cuando te preocupas y le dedicas tiempo te das cuenta de que hacerlo valió la pena, se disfruta más leyendo el código y facilita que tus compañeros entiendan lo que pusiste y no tengan que dedicar horas perdidas a comprender qué hace un método.

Pruebas: Todo el desarrollo será basado en pruebas, se realizarán las pruebas y posteriormente la implementación asociada a dichas pruebas. De esta manera el código estará ya refactorizado, disminuyendo la carga de trabajo posterior. Las pruebas se realizarán siguiendo el esquema de XP.

The Test/Code Cycle in XP

- Write one test.
- *Compile the test.* It should fail to compile, as you haven't yet implemented the code that the test calls.
- Implement just enough to compile. (Refactor first if necessary.)
- *Run the test and see it fail.*
- Implement just enough to make the test pass.
- *Run the test and see it pass.*
- Refactor for clarity and to remove duplication.
- Repeat from the top.

Figura 15. Ciclo de vida de la codificación/pruebas de Exteme programming. [25]

Al hacerlo de esta manera se están haciendo tres cosas de una vez, implementar, probar y refactorizar. Aumenta la calidad del código y disminuye el trabajo, todos ganan con esta fórmula.

Trabajo en equipo: El desarrollo de software es un trabajo en equipo, es muy importante la cooperación, que todos los miembros tengan un objetivo común y sepan claramente qué es lo que se va a hacer y cómo. Una falta de comunicación o coordinación puede llevar al fracaso del proyecto. Para evitar esto se insta a que los miembros del equipo trabajen en una misma mesa o si es a distancia que sea conectados a una aplicación VoIP.

Simplicidad: Muchas veces pasa que nos dejamos llevar añadiendo requisitos sin pensar en las consecuencias, sobre todo si tenemos poca experiencia en el sector. Este principio busca solucionar eso, está bien ser creativo y aportar nuevas ideas, pero hay que ser realista, es un equipo pequeño y no hay presupuesto ni personal para estar tanto tiempo desarrollando. Se buscará siempre hacer todo lo más sencillo y simple posible, a veces es bueno ser vago, tardar menos en hacer algo bien (no perfecto) y que al cliente le guste reduciendo costes y aumentando beneficios.

El proyecto constaría de 4 fases principales:

- **Estudio del problema:** En esta fase se realiza una especificación explicando el videojuego a desarrollar, tras esto se realiza un análisis preliminar seguido de los prototipos a papel o bocetos.
- **Desarrollo de los módulos:** Se comienza dividiendo el proyecto en módulos, se ordenan de mayor a menor importancia y se comienza con el más importante. Para cada módulo se realiza de manera iterativa un análisis, diseño e implementación. Tras realizar la implementación se obtiene *feedback* de los posibles clientes y según el estado del módulo y el *feedback* obtenido se realiza una segunda iteración o se considera el módulo como terminado y se pasa al siguiente.
- **Integración:** Al finalizar la implementación de todos los módulos se integran en un solo producto siguiendo el mismo patrón que los módulos (análisis, diseño, implementación y *feedback*) con el objetivo de refinar el software.
- **Release:** El producto ya finalizado se lanza al público y se identifican las fortalezas y debilidades del proceso de desarrollo, con el objetivo de hacerlo mejor la próxima vez.

Esta división de fases se ha realizado con el objetivo de mantener un mínimo de documentación, pero sin afectar demasiado al desarrollo preservando las características de los métodos ágiles: poder realizar cambios rápidamente con un mínimo coste y esfuerzo.

La importancia de la documentación en este caso no es simplemente para facilitar el mantenimiento del producto, sino que su objetivo principal, por lo que se ha mantenido en esta metodología, es el facilitar el desarrollo de software de calidad. Los casos de uso y los diagramas de clase (análisis y diseño) ayudan a identificar los requisitos y las clases, así como la reutilización, provocando que la etapa de implementación sea más rápida y de más calidad. Además, los casos de uso facilitan ponerse en la perspectiva del usuario de manera que se tienen en cuenta en mayor medida sus necesidades.

Otro objetivo a buscar con esta división es la de obtener continuo *feedback* a causa de lo volátil que es el mercado de los videojuegos, el cambio de un requisito puede provocar que el videojuego sea un éxito o un fracaso.

2.1. Estudio del problema

Esta fase consiste en formalizar la idea para dividir en módulos el proyecto.

La idea se plasma en una especificación, un documento donde se explique la idea, es decir, qué debe tener el videojuego a realizar. Este documento tiene que ser lo más objetivo posible puesto que a partir de él se comenzarán a identificar los requisitos y construir la aplicación.

El objetivo de este documento es el de evitar engordar el proyecto demasiado según se va avanzando, intentar formalizar una idea que sea posible realizar y establecer un objetivo común para todos los desarrolladores.

Tras realizar la especificación se hace un análisis preliminar, en este análisis se identifican los actores y la funcionalidad que realiza cada uno. Para cada funcionalidad y actor se pone una breve descripción. Hay que tener en cuenta que esta fase está pensada para hacerse rápidamente, el objetivo no es realizar mucha documentación, solo establecer los objetivos, la visión del producto e identificar los módulos.

Por último, se realizan los prototipos a papel o bocetos de la aplicación con el objetivo de tener una idea aproximadamente de como estarán distribuidos los elementos y los diseños necesarios que habrá que realizar.

En el caso práctico se mostrarán ejemplos con plantillas de esta documentación a realizar, siempre es más fácil rellenar una plantilla que improvisar, además establecer un estándar facilita la comunicación entre desarrolladores.

2.2. Desarrollo de los módulos

Esta fase se comienza dividiendo el proyecto en módulos a partir del análisis preliminar y los prototipos. Los módulos deben de ser pequeños, de 2 semanas como máximo.

Tras tener los módulos identificados se ordenan por prioridad o importancia para comenzar a trabajar con el módulo más importante.

El número de iteraciones de los módulos es algo muy importante, cada iteración es de una semana ya que esta metodología está enfocada en proyectos pequeños. La cantidad de iteraciones mínima y máxima de cada módulo es algo que se debe establecer al comienzo de esta fase y que se tiene que mantener durante todo el desarrollo, se recomienda una iteración como mínimo y dos como máximo.

La diferencia entre el número de iteraciones máximas y mínimas no debe de ser superior a dos, es decir, dos iteraciones como mínimo y cuatro como máximo es posible, dos iteraciones como mínimo y cinco como máximo no es válido. Es importante no hacer más iteraciones que las máximas establecidas ni menos que la mínima ya que podría acabar trayendo caos al proyecto.

Se empieza a trabajar con el módulo de mayor prioridad, primero se realiza el diagrama de casos de uso de ese módulo.

Los diagramas de casos de uso no se realizarán de manera habitual, se harán enfocados en la interfaz donde cada caso de uso es un elemento de ella.

Por ejemplo, en la siguiente imagen pueden apreciarse tres zonas: arriba una cabecera, a la izquierda un menú y en el centro la página principal.

Figura 16. Ventana de la pantalla principal (jugador). (Elaboración propia)

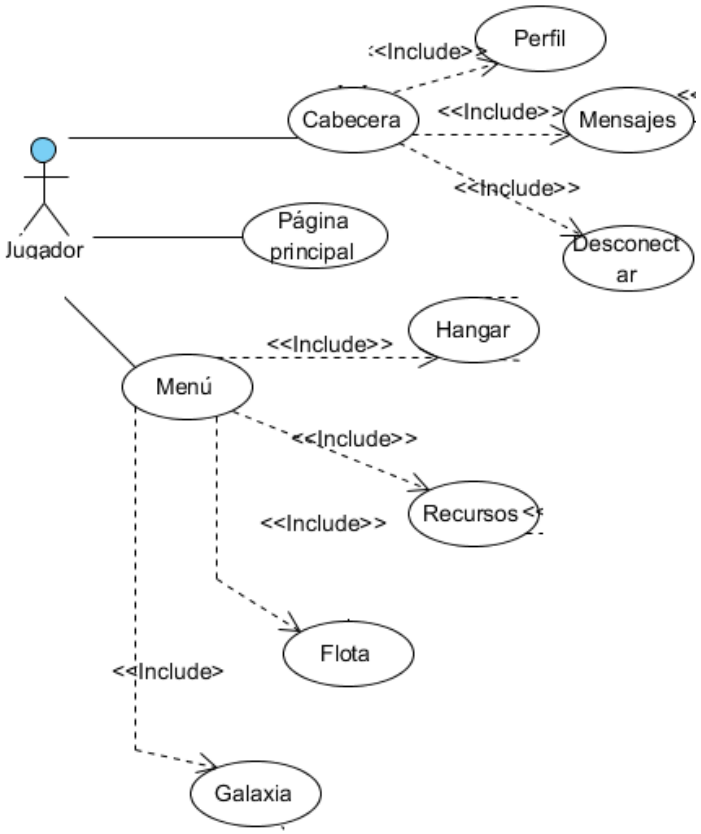

Figura 17. Diagrama casos de uso pantalla principal jugador. (Elaboración propia)

De esta manera se iría completando el diagrama, en el caso de que aparezcan listas de elementos se crea un caso de uso para la lista y otro para el elemento, un ejemplo sería una bandeja de mensajes.

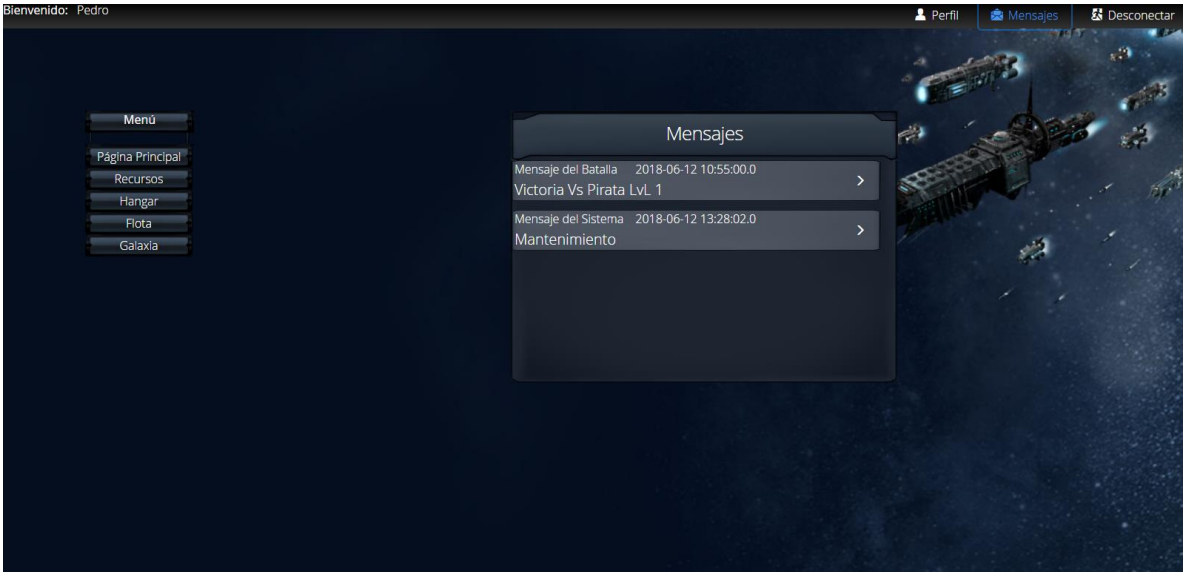

Figura 18. Ventana de mensajes. (Elaboración propia)

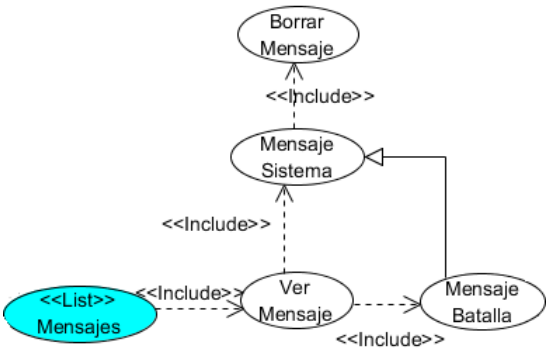

Figura 19. Diagrama casos de uso mensajes. (Elaboración propia)

Tras tenerlo hecho se procede a realizar cada uno de los casos de uso asociados a dicho diagrama. Como se ha comentado anteriormente en el caso práctico se mostrarán ejemplos con plantillas.

Al terminar los casos de uso se realiza un diagrama de clases de la interfaz y en el caso de haber base de datos el diagrama entidad relación.

Con el diagrama de clases pasa lo mismo que con el de casos de uso, está enfocado en la interfaz.

El diagrama de clases de la interfaz se realiza a partir del diagrama de casos de uso, cada caso de uso se traduce en una clase, siguiendo la idea de que cada clase (excepto en algunos casos) es una ventana o parte de esta, el diagrama de clases del primer ejemplo sería el que se muestra a continuación.

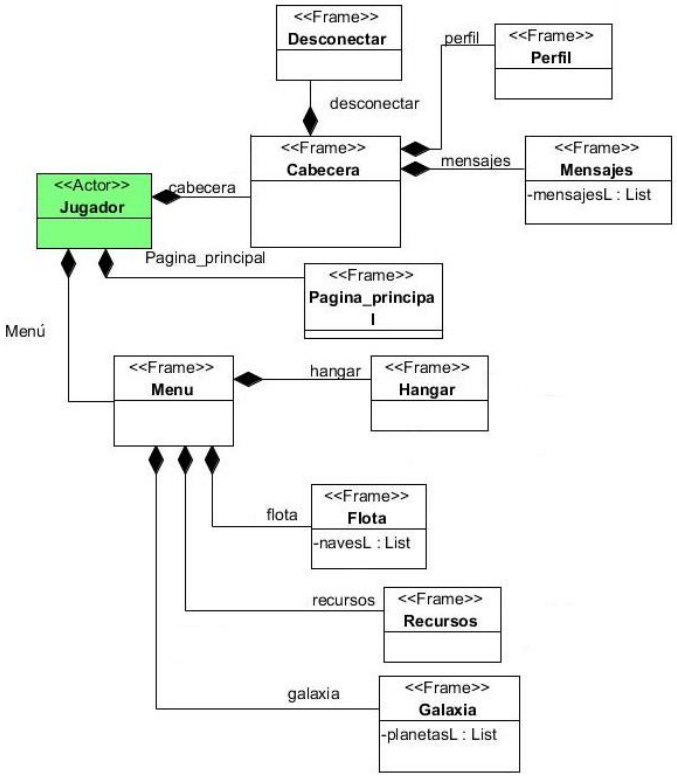

Figura 20. Diagrama de clases pantalla principal jugador. (Elaboración propia)

Con todo esto ya se puede comenzar a implementar el módulo.

Se identifican las distintas tareas que conforman el módulo, se estima su duración y se reparten entre los miembros para realizar la primera iteración. No se pueden coger una cantidad de tareas tal que la suma de su duración sea mayor de 1 semana.

Como se ha comentado anteriormente se comienza realizando las pruebas asociadas a la tarea a realizar siguiendo el esquema de la Figura 15, primero las pruebas, luego la implementación de esas pruebas, y por último la refactorización.

Al finalizar la iteración se obtiene *feedback* de lo desarrollado para ver que cambiar o mejorar, este *feedback* se obtendrá en función del público al que vaya dirigido el videojuego, puede ser por amigos, familiares o foros de internet.

Después de obtener el *feedback* se identifican los problemas que ha habido durante la iteración y qué se podría mejorar con el objetivo de hacerlo mejor la próxima vez.

Por último, si se ha terminado el módulo y no hay nada que cambiar se pasa al siguiente módulo, si no, se realiza otra iteración, se hacen los cambios necesarios en el análisis y diseño, se vuelve a dividir el módulo en tareas y se reparten entre los desarrolladores.

Esto se realiza para cada módulo hasta que todos los módulos estén terminados, cuando ya no queden más módulos se pasa a la siguiente fase.

2.3. Integración

Tras terminar de desarrollar todos los módulos se comienza a integrarlos, las etapas son las mismas que las de los módulos: análisis, diseño, implementación y *feedback*.

Las dos primeras fases consisten en unir la documentación realizada, mientras que en la fase de implementación se integran todos los módulos para formar un solo producto software.

Por último, se obtiene *feedback* del producto final y se realiza una segunda iteración si es necesario.

A pesar de no ser recomendable realizar una integración al final, se ha optado por hacerlo de esta manera para permitir la modificación de los módulos en cualquier momento; si se integraba al terminar cada módulo, entonces se impedía realizar cualquier cambio en etapas posteriores. De esta manera se pueden realizar modificaciones en cualquier momento a cambio de tener que realizar una gran integración al final del proyecto.

2.4. Release

Ya estando el producto terminado con todos los módulos integrados en un solo software se lanza al mercado, se identifica qué salió mal y qué se puede mejorar para futuros proyectos y se comienza a preparar el mantenimiento del producto.

3. Desarrollo

En esta sección se tratarán las distintas fases que componen la metodología propuesta.

3.1. Estudio del problema

Esta fase es la primera que se realiza, consiste en formalizar la idea para dividir en módulos el proyecto.

3.1.1. Especificación

La aplicación a desarrollar solo permitirá visualizar la ventana de inicio de sesión y registro al cibernauta que entre en la web.

El jugador al registrarse se le asignará un planeta, el primero disponible en la lista. Los planetas se dividirán en sectores y sistemas, en cada sistema habrá 10 sectores, y en cada sector habrá 10 planetas.

El jugador podrá ver sus recursos, habrá tres tipos de recursos: petróleo, metal y oro. Podrá aumentar la cantidad de recursos que genera cada segundo, esto lo hará mejorando las instalaciones de cada uno mediante el uso de más recursos, por ejemplo, oro y metal para mejorar la generación de petróleo. Además, podrá construir naves utilizando recursos, las naves tardarán un tiempo en construirse y no se podrán construir naves distintas mientras haya alguna nave construyéndose. Habrá tres tipos de naves: caza, corveta y nave de transporte.

El jugador podrá visualizar una lista de naves donde para cada nave que posea el jugador se indicará el número de naves en espera, las naves en movimiento y el total de naves.

El jugador tendrá acceso a una lista de movimientos que haya hecho él hacia otros planetas o que se dirijan hacia su planeta, si el movimiento es hacia otro planeta tendrá la opción de hacer regresar las naves. Cada movimiento le indicará el planeta de origen, el planeta de destino y el tiempo de llegada, además de poder visualizar las naves y la cantidad de ellas que estén efectuando dicho movimiento. Para atacar otros planetas el jugador podrá ver un listado de todos los planetas en un sector determinado, pudiendo cambiar de sector y de sistema. Al lado de cada planeta habrá un botón de atacar, al pulsarlo el jugador elegirá que naves y que cantidad de cada una envía, cambiando el tiempo de llegada de las naves en función de lo que elija. Si confirma el ataque las naves se enviarán al destino seleccionado. Un jugador no podrá atacarse a sí mismo y si un planeta es atacado con éxito 5 veces seguidas pasará a no poder ser atacado durante 8 horas a no ser que el propietario realice un ataque.

Habrá una sección de mensajes donde aparecerán mensajes del sistema enviados por el administrador o el técnico y mensajes de resultados de ataques o defensas, indicando el planeta de destino, el de origen, la fecha en la que ocurrió, las naves atacantes y defensoras, las naves que quedaron con vida tras la pelea y los recursos que obtuvo el atacante. Por último, el jugador podrá modificar su correo y contraseña.

El administrador además de poder hacer lo mismo que el jugador dispondrá de una lista de usuarios donde podrá cambiar su correo, su contraseña, el tipo de usuario y bloquearlo o borrarlo. También tendrá la posibilidad de crear un nuevo usuario, así como enviar mensajes a todos los usuarios del juego.

El técnico será el encargado de modificar y personalizar las características del juego, tendrá acceso a las mismas opciones que un jugador además de las secciones que le permitan ejercer su labor. El técnico podrá crear nuevas naves, así como modificar las ya existentes, las naves no se podrán eliminar. Al crear una nueva nave podrá introducirle un nombre, ponerle una imagen, modificar las características de la nave (salud, escudo, agilidad, velocidad, daño y capacidad de carga), modificar el coste de recursos de la nave, modificar el tiempo de construcción de cada nave (en segundos) e indicar si la nave está desbloqueada por defecto o no, en el caso de estar bloqueada qué piratas la pueden desbloquear y que probabilidad hay de que la desbloqueen. En el caso de modificar una nave podrá cambiar todos los campos de crear excepto el de bloquear la nave, si la nave está desbloqueada no se podrá bloquear su obtención. Habrá una sección donde el técnico podrá modificar la cantidad de recursos iniciales que produce cada instalación y ver de qué manera aumenta la producción según se mejoran. El técnico podrá modificar los planetas piratas, los planetas piratas son todos aquellos planetas no ocupados por usuarios, reservándose el quinto planeta de cada sector para que sea un planeta pirata. Cada planeta pirata tiene un nivel del 1 al 10 que determina su dificultad, este nivel es aleatorio. La finalidad de los planetas piratas es la obtención de gran cantidad de recursos, así como el desbloqueo de nuevas naves. El técnico podrá modificar de dichos planetas, para cada nivel, la cantidad de cada nave que disponen, así como el nivel de sus instalaciones de recursos. Al cabo de 5 horas de ser atacado, el planeta restaurará sus naves. El técnico al igual que el administrador podrá enviar mensajes a todos los usuarios del juego.

3.1.2. Análisis preliminar

En esta subsección se realizará un análisis preliminar del problema a tratar para entenderlo mejor y facilitar posteriormente el análisis que se realizará en cada módulo.

3.1.1.1. Usuarios del sistema

Hemos identificado en el sistema los siguientes usuarios:

- **Cibernauta**: Es el usuario que accede a la web sin haber iniciado sesión en ella.
- **Jugador**: Es la persona que se registra en la web.
- **Administrador**: Persona encargada de la gestión de usuarios de la web.
- **Técnico**: Persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego tales como las características de las naves y la generación de recursos.

3.1.1.2. Sistemas Informáticos externos al Sistema

Algunas de las gestiones del sistema están llevadas por sistemas externos, a continuación, se describen estos sistemas:

- **Gestión de notificaciones**: Las notificaciones referentes a la cuenta del usuario se tramitarán mediante un gestor de correo externo a la aplicación, como es el caso del registro y del cambio de contraseña.

3.1.1.3. Funciones del Sistema

A continuación, se explicarán todas las funciones llevadas a cabo en el sistema, este es el modelo que se propone como plantilla, indicando qué usuarios y sistemas externos participan en ellas y una breve descripción de ellas.

- **Funcionalidad:** Identificarse
  - **Actores:** Cibernauta
  - **Descripción:** El Cibernauta podrá identificarse en la web introduciendo el usuario y la contraseña.

- **Funcionalidad:** Registrarse
  - **Actores:** Cibernauta
  - **Descripción:** El Cibernauta podrá registrarse en la web introduciendo su correo, un nombre de usuario y una contraseña que deberá escribir dos veces.

- **Funcionalidad:** Consultar datos del perfil.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Tanto el Jugador, como el Administrador y el Técnico podrán ver los datos de su perfil, el Administrador además podrá ver los datos de todos los demás usuarios.

- **Funcionalidad:** Modificar datos personales.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Se podrán modificar los datos personales excepto el correo, además el Administrador podrá cambiar el tipo de usuario.

- **Funcionalidad:** Bloquear/Desbloquear usuario
  - **Actores:** Administrador.
  - **Descripción:** El Administrador podrá bloquear y desbloquear usuarios, impidiendo o permitiendo que el usuario acceda a su cuenta.

- **Funcionalidad:** Dar de alta a un usuario.
  - **Actores:** Administrador.
  - **Descripción:** El Administrador podrá dar de alta a un usuario en la base de datos.

- **Funcionalidad:** Dar de baja a un usuario.
  - **Actores:** Administrador.
  - **Descripción:** El Administrador podrá dar de baja a un usuario, eliminándolo así de la base de datos.

- **Funcionalidad:** Consultar lista de usuarios.
  - **Actores:** Administrador.
  - **Descripción:** El Administrador podrá ver una lista de los usuarios registrados en el sistema.

- **Funcionalidad:** Consultar recursos disponibles.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán consultar la cantidad de recursos que tienen.

- **Funcionalidad:** Consultar nivel y producción de las instalaciones.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán consultar el nivel de sus instalaciones de recursos, así como la cantidad de recursos que producen.
- **Funcionalidad:** Aumentar el nivel de las instalaciones.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán aumentar el nivel de las instalaciones, a cambio de recursos, para que generen más recursos.

- **Funcionalidad:** Consultar naves existentes.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán ver las naves disponibles, así como las naves que no tienen desbloqueadas.

- **Funcionalidad:** Construir naves.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** A cambio de recursos, los usuarios podrán construir naves.

- **Funcionalidad:** Cancelar construcción de naves.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán cancelar las construcciones de naves no terminadas, recuperando así los recursos invertidos.

- **Funcionalidad:** Consultar naves poseídas.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Se podrá consultar las naves que se poseen, así como su localización, en movimiento o en espera.

- **Funcionalidad:** Consultar lista de movimientos.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Se podrá ver una lista de los movimientos de tus flotas y de otras flotas hacia tu planeta.

- **Funcionalidad:** Cancelar ataque.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán cancelar un ataque si las naves aun no llegaron al planeta de destino.

- **Funcionalidad:** Consultar datos de movimiento.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán ver los datos de cada movimiento.

- **Funcionalidad:** Consultar lista de planetas.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Se podrá consultar la lista de planetas, en qué sector y en qué sistema se encuentran.

- **Funcionalidad:** Atacar planeta.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán atacar planetas que no sean suyos y no tengan una protección activada.

- **Funcionalidad:** Consultar lista de mensajes.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Se podrá ver una lista de mensajes del sistema.

- **Funcionalidad:** Consultar mensaje.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán ver los detalles de cada mensaje.

- **Funcionalidad:** Borrar mensaje.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán descartar mensajes de su lista.

- **Funcionalidad:** Desconectarse.
  - **Actores:** Jugador, Administrador, Técnico.
  - **Descripción:** Los usuarios podrán desconectarse del sistema.

- **Funcionalidad:** Crear una nave.
  - **Actores:** Técnico.
  - **Descripción:** El Técnico podrá crear nuevas naves pudiendo personalizar sus características.

- **Funcionalidad:** Consultar lista de naves.
  - **Actores:** Técnico.
  - **Descripción:** El Técnico podrá ver una lista de las naves registradas en el sistema.

- **Funcionalidad:** Modificar una nave.
  - **Actores:** Técnico.
  - **Descripción:** El Técnico podrá modificar las naves ya existentes, pero si una nave está desbloqueada no podrá bloquearla.

- **Funcionalidad:** Modificar tasa de generación de recursos.
  - **Actores:** Técnico.
  - **Descripción:** El Técnico podrá modificar la tasa de generación de recursos por nivel de las instalaciones.

- **Funcionalidad:** Consultar lista de piratas.
  - **Actores:** Técnico.
  - **Descripción:** El Técnico podrá ver una lista de los piratas existentes.

- **Funcionalidad:** Modificar planetas piratas.
  - **Actores:** Técnico.
  - **Descripción:** El Técnico podrá modificar la configuración de los planetas piratas.

- **Funcionalidad:** Enviar mensaje.
  - **Actores:** Administrador, Técnico.
  - **Descripción:** El Administrador y el Técnico podrán enviar mensajes a todos los usuarios.

3.1.3. Tecnologías y herramientas

A continuación se comentarán las distintas tecnologías y herramientas utilizadas para tanto el desarrollo como la documentación.

3.1.3.1. Herramientas

Para las herramientas se proponen tres maneras de documentarlas: realizar una lista con los nombres de las herramientas a utilizar, método simple en el caso de que no haya demasiado tiempo para documentar; indicar nombre de la herramienta, de que tipo es (documentación, desarrollo, etc.) y una breve descripción, esto sería lo ideal; por último, en párrafos explicando cada una, esta es la manera en la que lo he realizado, principalmente por el tipo de documento que estoy realizando, aun así recomiendo la segunda opción.

El uso de herramientas como las herramientas CASE es muy importante en el desarrollo de software, reducen el tiempo de ciertas tareas y pueden ayudar a automatizar tareas repetitivas. Una consideración importante es el saber elegir que herramientas usar, ya que una mala elección puede llevar a aumentar el tiempo de desarrollo en lugar de reducirlo [39].

Las herramientas a utilizar durante el desarrollo del proyecto pueden dividirse en dos tipos: las herramientas de uso diario, que se utilizarán durante prácticamente todo el proyecto y las herramientas de desarrollo, que se utilizarán solamente en momentos específicos.

3.1.3.2. Desarrollo

Para el desarrollo del producto se ha optado por el IDE Eclipse [40] debido a su coste (es gratis) y a que se va a utilizar java como lenguaje de programación, además de permitir la integración con todas las herramientas necesarias para el desarrollo de manera fácil y rápida.

Dentro de Eclipse se van a usar varios frameworks, el principal Vaadin [41] junto a Vaadin designer, con este framework se realizará el entorno web ya que traduce el código java a html permitiendo así realizar páginas web sabiendo solo programar en java. Vaadin designer se usará para facilitar el desarrollo de las ventanas, ya que permite realizarlas arrastrando componentes.

El framework Spring [42] se utilizará para facilitar el acceso y gestión de los datos de la base de datos por parte de la aplicación, así como el desarrollo de las pruebas unitarias junto a JUnit [43].

NetBeans [44] a pesar de ser un IDE no se utilizará como entorno de desarrollo, este IDE tiene una peculiaridad, permite realizar la ingeniería inversa de una base de datos generando sus clases java con consultas predefinidas, estas clases java se reutilizarán en el proyecto lo cual facilitará en gran medida el proceso de desarrollo debido a su estructura.

Por último, respecto al IDE se utilizará EGit [45] para la conexión con el repositorio de Github [46], se ha elegido Github frente a otras herramientas debido a su fácil configuración, experiencia en su uso y beneficios por ser alumno.

En cuanto a la base de datos se utilizará Mysql como lenguaje utilizando la herramienta Mysql Workbench [47] para el diseño y control de la base de datos, junto a Xampp [48] para habilitar la conexión. La principal razón de que se utilice Mysql es el uso de cron jobs, ya que se utilizarán para los movimientos de las flotas de naves.

Al final no se realizará integración con Jenkins [49] debido a la metodología que se ha propuesto, además no será posible utilizar un hosting ya que son compartidos y no permiten el uso de cron jobs. Se optará por un servidor VPS, posiblemente una máquina Ubuntu alojada en Fiware Lab [50].

Por último, para el diseño de la aplicación se usará la herramienta PaintTool SAI [51], un programa para la realización de ilustraciones y procesamiento de imágenes, se ha elegido este programa frente a muchas otras alternativas debido a la cantidad de opciones y personalización que ofrece, facilitando en gran medida la realización de ilustraciones.

3.1.3.3. Documentación

Para la parte de documentación se utilizará para los documentos Microsoft Word debido a su gran cantidad de opciones y personalización de documentos, estos documentos se subirán a Google Drive para un fácil acceso por parte del director.

La documentación de la planificación se realizará utilizando la herramienta Microsoft Project, mientras que la referente a los requisitos se hará con la herramienta REM [52] que proporciona plantillas para facilitar su documentación.

La estimación del esfuerzo con su documentación se hará siguiendo el modelo COCOMO [53] mediante el uso de una herramienta desarrollada por la Universidad del Sur de California.

Para los diagramas se utilizará Visual Paradigm [54], una de las mejores herramientas para la realización de diagramas debido a su gran abanico de características.

Por último, la comunicación con el director se hará utilizando el servicio de correo Gmail.

3.1.3. Estimación

Esta fase no es necesario hacerla, aunque siempre es recomendable para comprobar si un proyecto con las funcionalidades que se han especificado es viable o no, sobre todo teniendo en cuenta que estos proyectos son realizados a partir de ideas propias.

3.1.3.1. Puntos de función

A continuación se muestra para cada entrada, salida y consulta del sistema su dificultad en función de los atributos que le afecten.

Tabla 1. Puntos de función – Entradas. (Elaboración propia)

Entradas	Atributos	Dificultad
Registrarse	4	Baja
Subir nivel instalación de recursos	5	Media
Elegir un tipo de nave	1	Baja
Construir nave	5	Media
Cancelar construcción	3	Baja
Cancelar movimiento	1	Baja
Atacar planeta	2	Baja
Editar Perfil	3	Baja
Crear usuario	6	Baja
Borrar usuario	1	Baja
Editar usuario	5	Baja
Crear nave	13	Alta
Editar nave	13	Alta
Modificar información recursos	2	Baja
Modificar pirata	2	Baja
Crear mensaje	2	Baja

Tabla 2. Puntos de función – Salidas. (Elaboración propia)

Salidas	Atributos	Dificultad
Mensaje error registro	1	Baja
Mensaje éxito registro	1	Baja
Nombre del usuario	1	Baja
Recursos disponibles	3	Baja
Información instalaciones de recursos	7	Baja
Cargar el hangar	2	Baja
Cargar naves del tipo elegido	10	Baja
Mostrar naves en construcción	3	Baja
Mensaje error construcción nave	1	Baja
Mostrar naves de la flota	3	Baja
Cargar movimientos	3	Baja
Cargar información de movimiento	5	Baja
Cargar información de los planetas	6	Media
Cargar sistemas	1	Baja
Cargar naves disponibles para el ataque	2	Baja
Cargar información preliminar del movimiento	3	Baja
Mensaje error ataque	1	Baja
Mensaje éxito ataque	1	Baja
Cargar información del perfil	3	Baja
Mensaje error editar perfil	1	Baja
Mensaje éxito editar perfil	1	Baja

Cargar mensajes	2	Baja
Cargar mensaje de batalla	17	Media
Cargar mensaje del sistema	2	Baja
Mensaje error crear usuario	1	Baja
Mensaje éxito crear usuario	1	Baja
Cargar lista de usuarios	3	Baja
Cargar información eliminar usuario	4	Baja
Mensaje error borrar usuario	1	Baja
Mensaje éxito borrar usuario	1	Baja
Cargar información editar usuario	6	Baja
Mensaje error editar usuario	1	Baja
Mensaje éxito editar usuario	1	Baja
Cargar información necesaria nave	1	Baja
Mensaje error crear nave	1	Baja
Mensaje éxito crear nave	1	Baja
Cargar lista de naves	4	Baja
Cargar información nave	13	Media
Mensaje error editar nave	1	Baja
Mensaje éxito editar nave	1	Baja
Cargar información de los recursos	4	Baja
Cargar lista de piratas	5	Baja
Cargar información de los piratas	7	Media
Mensaje error crear mensaje	1	Baja
Mensaje éxito crear mensaje	1	Baja

Tabla 3. Puntos de función – Consultas. (Elaboración propia)

Consultas	Atributos	Dificultad
Iniciar sesión	2	Baja

3.1.3.2. COCOMO

Para calcular el esfuerzo, costo y duración del proyecto se utilizó el modelo COCOMO intermedio. Se calcularon los puntos de función del proyecto (apartado anterior) y posteriormente se estableció un valor para cada uno de los atributos de este modelo.

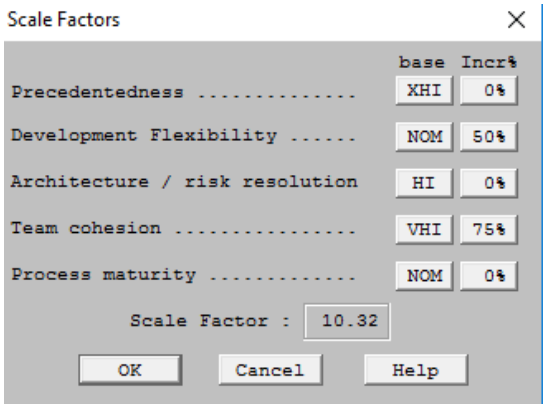

Figura 21. Scale Factors. (Elaboración propia)

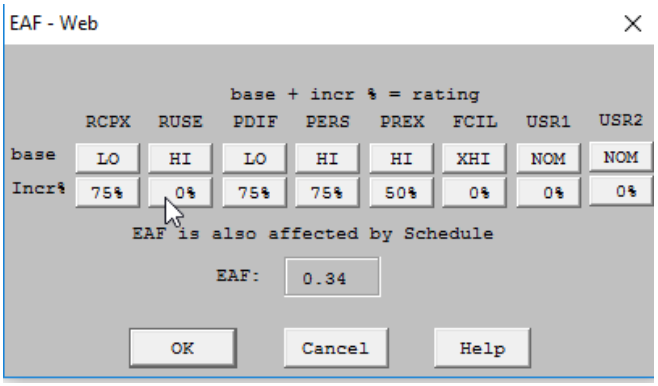

Figura 22. EAF. (Elaboración propia)

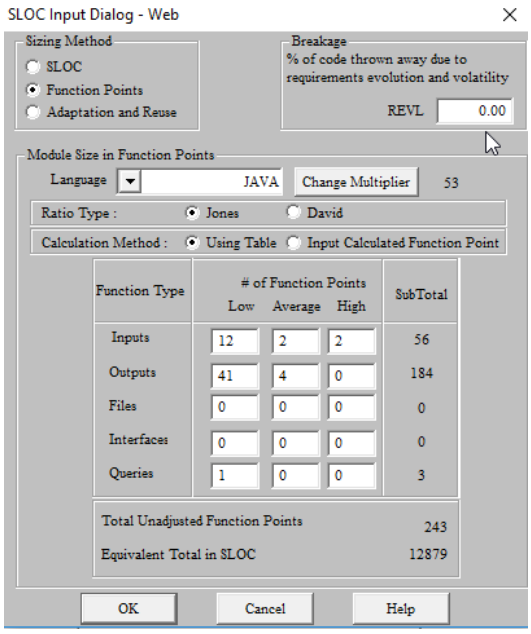

Figura 23. Puntos de función. (Elaboración propia)

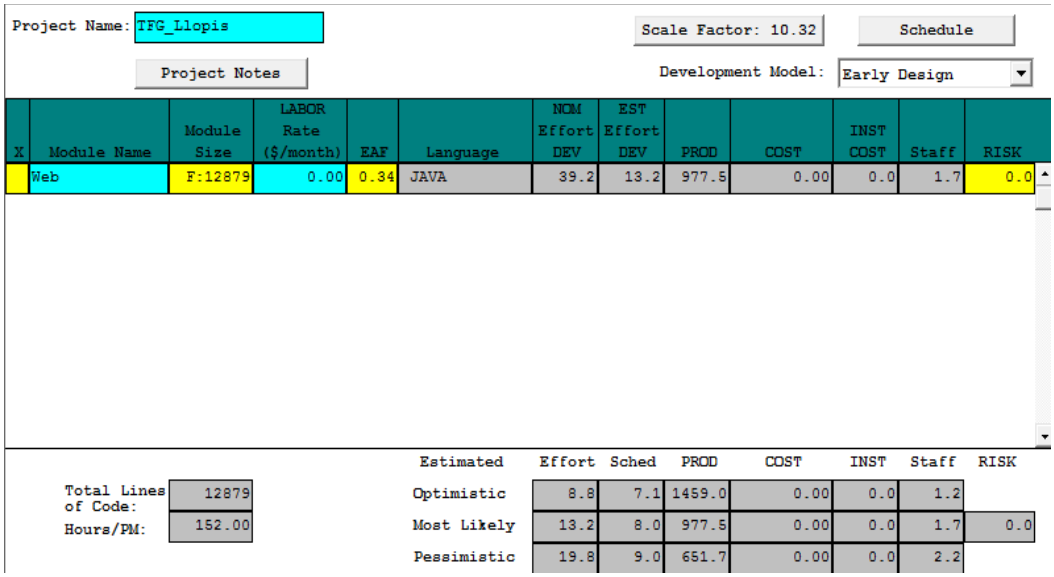

Figura 24. Resultado COCOMO. (Elaboración propia)

El proyecto se realizaría en Java y su tamaño sería de 12879 líneas de código, donde se necesitarían dos personas trabajando durante 8 meses y una de ellas trabajando al 70%.

3.2. Identificación de los módulos

Tras realizar el estudio del problema se identifican los distintos módulos y se ordenan por prioridad. Esta es la parte más difícil de esta metodología, sin experiencia es muy complicado hacer una identificación correcta de los módulos con una duración máxima limitada.

A partir del estudio del problema los módulos obtenidos son los siguientes:

- **Módulo de inicio de sesión**: Este módulo engloba el inicio de sesión y el registro por parte del cibernauta, es la base de toda aplicación web y se estima una duración de como máximo una semana (una iteración).
- **Modulo usuarios**: El módulo usuarios es básicamente las funcionalidades del administrador excepto los mensajes, se estima una duración de 1 semana.
- **Módulo técnico**: Al igual que el módulo anterior este se centra en un rol, en este caso el de técnico, este módulo posee muchas funcionalidades y ventanas, se estima una duración de 2 semanas.
- **Módulo naves**: Este módulo incluiría la construcción de naves y los recursos de un jugador (subir nivel, consultar, generar, etc.), es un módulo muy pequeño en cuanto a ventanas, aunque podría presentar problemas en algunas funcionalidades y diseño, se estima una duración de 1 semana.
- **Módulo información**: En este módulo se realizaría lo referente a la información que recibe el jugador, las naves que posee y los mensajes del sistema,

tanto la recepción como el envío por parte del técnico y el administrador. Se estima una duración de 1 semana.

- **Módulo ataque**: Este módulo contiene todo lo relacionado con el ataque, el poder atacar otros planetas, los mensajes de batalla y la lista de movimientos que se están realizando con sus datos. Se estima una duración de 2 semanas, es cierto que no tiene prácticamente funcionalidades o ventanas, pero detrás tiene muchos scripts de la base de datos.

Las estimaciones se han realizado teniendo en cuenta el tiempo restante, en algunos casos sería recomendable o dividir el módulo en otros más pequeños o darle alguna semana más, por ejemplo, para cada módulo de una semana sería recomendable dejarlos en dos semanas para obtener *feedback* y pulirlo, sobre todo el diseño.

Los módulos ya están ordenados por prioridad, se ha ordenado en función de la funcionalidad, la más básica primero y la más especifica la última.

La implementación se comenzaría el día 11 de abril y se finalizaría el 22 de junio, el calendario asociado sería el siguiente:

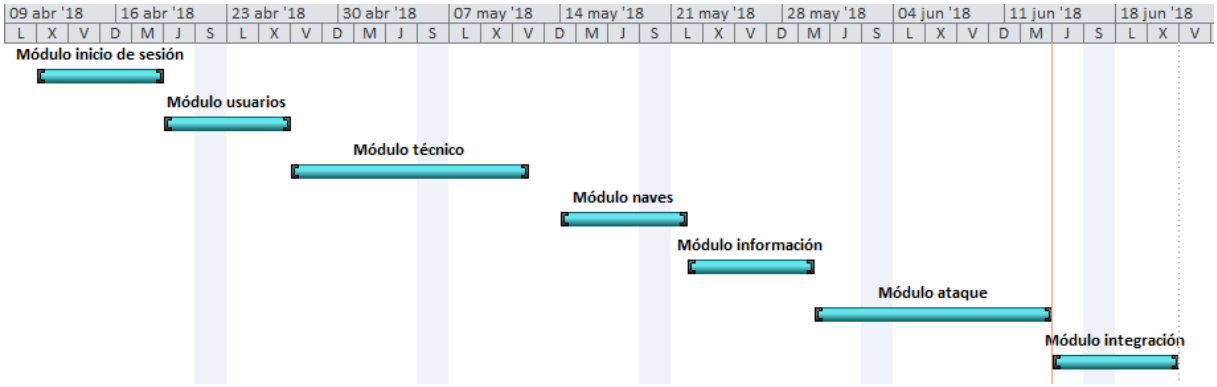

Figura 25. Planificación en el tiempo del proyecto. (Elaboración propia)

Como se puede apreciar el tiempo va muy justo ya que se estimó que la finalización del proyecto sería el día 22 de junio y la integración se va a terminar el día 21 en el mejor caso comenzando el primer módulo el día 11 de abril, esto probablemente ocasione que algunas iteraciones duren menos del tiempo establecido, algo no recomendable en esta metodología.

3.3. Implementación

En esta subsección se tratarán las dos siguientes fases, la del desarrollo de los módulos y la fase de integración.

3.3.1. Módulo Inicio de Sesión

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.1.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.1.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

3.3.1.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo.

Figura 26. Diagrama de casos de uso del módulo Inicio de Sesión (Cibernauta). (Elaboración propia)

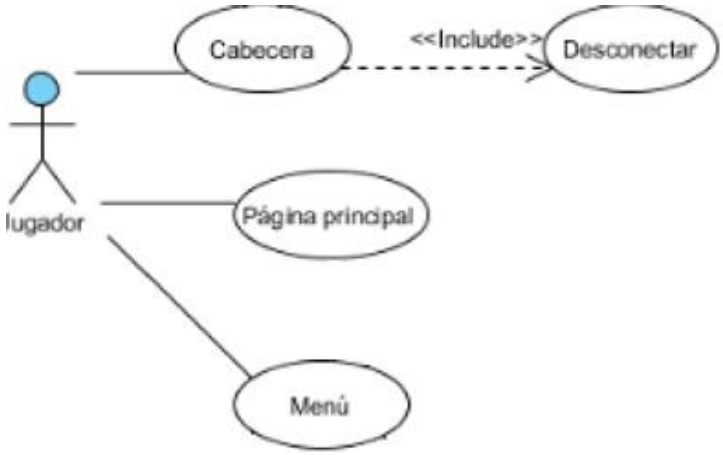

Figura 27. Diagrama de casos de uso del módulo Inicio de Sesión (Jugador). (Elaboración propia)

3.3.1.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 4. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 5. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 6. Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 7. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.1.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 8. Caso de uso identificarse. (Elaboración propia)

<table>
<tr><td>CDU-0001</td><td>Identificarse</td></tr>
<tr><td>[Versión]</td><td>1.0 (11/04/2018)</td></tr>
<tr><td>[Dependencias]</td><td></td></tr>
<tr><td>Descripción</td><td>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el cibernauta quiera identificarse en el sistema.</td></tr>
<tr><td>Precondición</td><td>No estar identificado.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>Si el cibernauta quiere registrarse, se realiza el caso de uso Registrarse (CDU-0002)</td></tr>
<tr><td>2</td><td>El actor Cibernauta (CDU-ACT-0001) introduce su email y contraseña.</td></tr>
<tr><td>3</td><td>El sistema autentifica la información e identifica al usuario en el sistema.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El usuario está identificado.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>3</td><td>Si la información no es correcta, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto</td></tr>
<tr><td>3</td><td>Si el usuario está bloqueado, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto</td></tr>
<tr><td>3</td><td>Si el cibernauta no está registrado, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto</td></tr>
</table>
</td></tr>
</table>

Tabla 9. Caso de uso registrarse. (Elaboración propia)

CDU-0002	Registrarse
[Versión]	1.0 (11/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el cibernauta quiera registrarse en el sistema o durante la realización de los siguientes casos de uso: [CDU-0001] Identificarse
Precondición	No está registrado.
Secuencia normal	**Paso** / **Acción** 1 / El actor Cibernauta (CDU-ACT-0001) introduce un email, nombre de usuario y dos veces la contraseña. 2 / El sistema registra los datos en la base de datos y confirma al usuario el registro.
Postcondición	Está registrado en el sistema
Excepciones	**Paso** / **Acción** 2 / Si el correo utilizado ya existe en la base de datos, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto 2 / Si el usuario utilizado ya existe en la base de datos, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto 2 / Si las contraseñas no coinciden, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto 2 / Si algún dato es inválido, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto

Tabla 10. Caso de uso página principal. (Elaboración propia)

CDU-0003	Página principal
[Versión]	1.0 (11/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario se identifique
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los recursos que posee el usuario
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 11. Caso de uso desconectarse. (Elaboración propia)

CDU-0004	Desconectarse
[Versión]	1.0 (11/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera desconectarse del sistema.
Precondición	Estar identificado en el sistema.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de desconexión. 2 / El sistema desconecta al usuario del sistema
Postcondición	No estar identificado en el sistema.
Excepciones	**Paso** / **Acción** - / -

3.3.1.1.2. Relación Diagrama – Casos de Uso

A continuación se muestran las relaciones entre los diagramas y los casos de uso de los actores.

Tabla 12. Relación diagrama – caso de uso. Módulo inicio de sesión. (Elaboración propia)

Especificación/ Diagrama	CDU-0001	CDU-0002	CDU-0003	CDU-0004
Iniciar Sesión	↑			
Registrarse		↑		
Cabecera				↑
Desconectar				↑
Menú			↑	
Página principal			↑	

3.3.1.2. Diseño

A continuación se muestran los diagramas de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

Figura 28. Diagrama de clases del módulo de Inicio de Sesión (Cibernauta). (Elaboración propia)

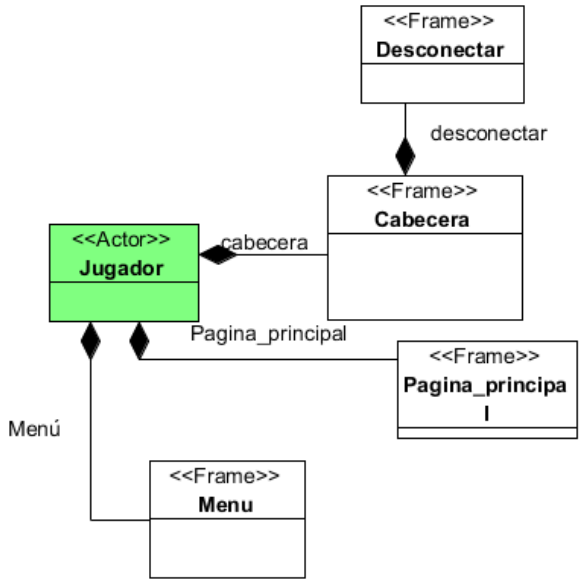

Figura 29. Diagrama de clases del módulo de Inicio de Sesión (Jugador). (Elaboración propia)

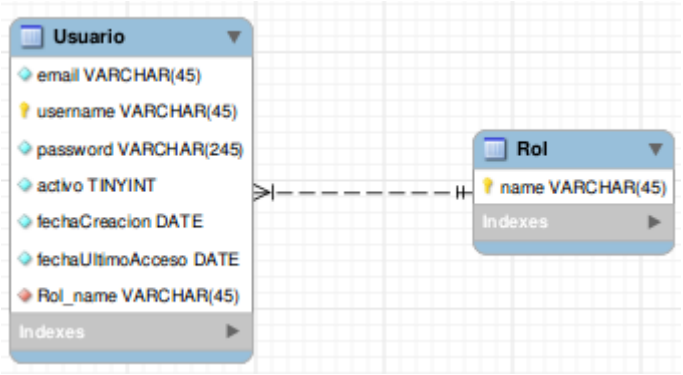

Figura 30. Diagrama de la base de datos del módulo de Inicio de Sesión. (Elaboración propia)

3.3.1.3. Implementación del módulo

Este módulo implica solo una semana de duración y una persona en el desarrollo, por lo que para el reparto de tareas no hay ningún problema.

Cabe destacar que al ser el primer módulo se debió de configurar todo: Spring Tool Suite (eclipse con los plugins de Spring), Vaadin, NetBeans, los plugins necesarios para eclipse y la conexión entre eclipse y la base de datos.

3.3.1.3.1. Probar las clases de la base de datos

Primero se comenzó realizando las pruebas y su código asociado de las clases de la base de datos, en este caso la clase Usuario y la clase Rol.

```
@Test
public void testUsuarioFull()
{
    Date fechaAhora = new Date();
    Usuario user = new Usuario("juan@gmail.com", "Pepe", "1234", true, fechaAhora, fechaAhora);

    assertEquals("juan@gmail.com", user.getEmail());
    assertEquals("Pepe", user.getUsername());
    assertEquals(true, user.getActivo());
    assertEquals(fechaAhora, user.getFechaCreacion());
    assertEquals(fechaAhora, user.getFechaUltimoAcceso());
    assertTrue(CheckPassword.verifyHash("1234", user.getPassword()));
}

@Test
public void testUsuarioEmail()
{
    Date fechaAhora = new Date();
    Usuario user = new Usuario("juan@gmail.com");

    assertEquals("juan@gmail.com", user.getEmail());
}

@Test
public void testUsuarioEspacios()
{
    Date fechaAhora = new Date();
    Usuario user = new Usuario("juan@ gmail. com", "P ep e", "1 23 4", true, fechaAhora, fechaAhora);

    assertEquals("juan@gmail.com", user.getEmail());
    assertEquals("Pepe", user.getUsername());
    assertEquals(true, user.getActivo());
    assertEquals(fechaAhora, user.getFechaCreacion());
    assertEquals(fechaAhora, user.getFechaUltimoAcceso());
    assertTrue(CheckPassword.verifyHash("1234", user.getPassword()));
```

Figura 31. Prueba de la clase de la base de datos de usuario. (Elaboración propia)

3.3.1.3.2. Probar el inicio de sesión

Siguiendo la metodología establecida se realizaron las pruebas asociadas a la funcionalidad de inicio de sesión y al mismo tiempo se realizaba la implementación.

```
@Test
public void testLoginCorrecto()
{
    String username = "juan";
    String password = "damn";

    assertTrue(CrudUsuario.login(username, password, repo).isEmpty());
}

@Test
public void testLoginUsuarioIncorrecto()
{
    String username = "";
    String password = "damn";

    assertFalse(CrudUsuario.login(username, password, repo).isEmpty());
}

@Test
public void testLoginPasswordIncorrecto()
{
    String username = "juan";
    String password = "";

    assertFalse(CrudUsuario.login(username, password, repo).isEmpty());
}

@Test
public void testLoginJugador()
{
    String username = "juan";
    String password = "damn";

    assertTrue(CrudUsuario.login(username, password, repo).isEmpty());
    assertEquals(repo.findByUsername(username).get(0).getRolName().getName(), "Jugador");
```

Figura 32. Pruebas de inicio de sesión. (Elaboración propia)

El acceso a la base de datos para insertar, eliminar y actualizar se realiza mediante interfaces intermedias haciendo uso de las órdenes SQL creadas en las propias clases de la base de datos, de esta manera el acceso a la base de datos es sencillo, controlado y limpio.

```
package com.tfgllopis.moduloInicioSesion;

import java.io.Serializable;


@Entity
@Table(name = "Usuario", catalog = "mydb", schema = "")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u")
    , @NamedQuery(name = "Usuario.findByEmail", query = "SELECT u FROM Usuario u WHERE u.email = :email")
    , @NamedQuery(name = "Usuario.findByUsername", query = "SELECT u FROM Usuario u WHERE u.username = :username")
    , @NamedQuery(name = "Usuario.findByPassword", query = "SELECT u FROM Usuario u WHERE u.password = :password")
    , @NamedQuery(name = "Usuario.findByActivo", query = "SELECT u FROM Usuario u WHERE u.activo = :activo")
    , @NamedQuery(name = "Usuario.findByFechaCreacion", query = "SELECT u FROM Usuario u WHERE u.fechaCreacion = :fe
    , @NamedQuery(name = "Usuario.findByFechaUltimoAcceso", query = "SELECT u FROM Usuario u WHERE u.fechaUltimoAcce
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    // @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-
    @Id
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 50)
    @Column(name = "email")
```

Figura 33. Sentencias SQL empotradas en la clase de usuario. (Elaboración propia)

```
package com.tfgllopis.moduloInicioSesion;

import java.util.List;

public interface UsuarioRepository extends JpaRepository<Usuario, Integer> {

    public List<Usuario> findByEmailLikeIgnoreCase(@Param("email") String string);

    public List<Usuario> findByUsername(@Param("username") String string);

}
```

Figura 34. Interfaz de acceso a las sentencias SQL empotradas en la clase de usuario. (Elaboración propia)

Al realizarlo de esta manera, se identificó una nueva clase destinada exclusivamente a funciones CRUD, en ella se comprueba la validez de los datos y según el resultado se informa del error o se realiza la operación correspondiente.

```java
public class CrudUsuario
{
    public static String login(String username, String password, UsuarioRepository repo)
    {

        if(!UserDataValidator.comprobarUsuarioBD(username, repo)) return "Usuario no registrado";
        if(!UserDataValidator.comprobarPassword(username, password, repo)) return "Datos incorrectos";
        if(!UserDataValidator.comprobarActivo(username, repo)) return "Usuario bloqueado";

        actualizarFechaAcceso(username, repo);
        return "";
    }

    private static void actualizarFechaAcceso(String username, UsuarioRepository repo)
    {
        Usuario user = Usuario.cargarUsuario(username, repo);
        user.setFechaUltimoAcceso(new Date());
        user.guardarUsuario(repo);
    }
```

Figura 35. Clase CRUD de usuario. (Elaboración propia)

```java
public class UserDataValidator
{
    public static boolean comprobarEmail(String email)
    {
        return EmailValidator.getInstance().isValid(email);
    }

    public static boolean comprobarUser(String user)
    {
        //Detecta todo lo que no sean letras ni números
        Pattern p = Pattern.compile("[^a-zA-Z0-9]+");
        return !p.matcher(user).find();
    }

    public static boolean comprobarPassword(String pass1, String pass2)
    {
        pass1 = pass1.replaceAll("\\s+","");
        pass2 = pass2.replaceAll("\\s+","");

        if(pass1.isEmpty()) return false;

        return pass1.equals(pass2);
    }

    public static boolean comprobarEmailBD(String email, UsuarioRepository repo)
    {
        email = email.replaceAll("\\s+","");

        return !repo.findByEmailLikeIgnoreCase(email).isEmpty();
    }
```

Figura 36. Clase de validación de datos del usuario. (Elaboración propia)

En esta etapa se procedió a buscar una manera de encriptar las contraseñas, para ello se utilizó la clase BCrypt.

BCrypt encripta la contraseña generando un salt aleatorio que combina con la contraseña antes de crearle un hash, de esta manera dos contraseñas iguales no poseen el mismo hash.

Una de las características de BCrypt es la posibilidad de establecer el número de iteraciones que la función hash realiza para encriptar las contraseñas. El número de iteraciones se ha establecido en 2^{11}, es decir, 11 rondas, lo máximo que se podía poner sin que el rendimiento se viera afectado y las contraseñas tuvieran la suficiente protección.

```
public class CheckPassword
{
    private static int LOGROUNDS = 11;

     public static String hash(String password) {
            return BCrypt.hashpw(password, BCrypt.gensalt(LOGROUNDS));
        }

        public static boolean verifyHash(String password, String hash)
        {
            return BCrypt.checkpw(password, hash);
        }
}
```

Figura 37. Clase de comprobación de contraseña del usuario. (Elaboración propia)

3.3.1.3.3. Probar el registro

Al igual que con el inicio de sesión se realizan sus pruebas y en función de ellas se implementa la funcionalidad correspondiente en la clase CRUD.

```
public static String registro(String username, String email, String password1, String password2, UsuarioRep
{
    Usuario user;
    Date fechaRegistro = new Date();
    //Gestor_Correos correo = new Gestor_Correos();
    if(!UserDataValidator.comprobarUser(username)) return "Usuario inválido";
    if(!UserDataValidator.comprobarEmail(email)) return "Email inválido";
    if(!UserDataValidator.comprobarPassword(password1, password2)) return "Las contraseñas no coinciden";
    if(UserDataValidator.comprobarUsuarioBD(username, repo)) return "El usuario ya existe";
    if(UserDataValidator.comprobarEmailBD(email, repo)) return "El correo ya está en uso";
    //if(!correo.correo_registro(email)) return "Imposible acceder al correo";

    user = new Usuario(email, username, password1, true, fechaRegistro, fechaRegistro);
    user.setRolName(rolRepo.findByName("Jugador").get(0));
    user.guardarUsuario(repo);

    return "";
}
```

Figura 38. Método de registro del usuario. (Elaboración propia)

Por último para terminar la funcionalidad se implementó el gestor de correo mediante la librería de mail de javax.

```
public class Gestor_Correos implements Gestor_correo {

    private final Properties properties = new Properties();

    private Session session;

    private void init() {

        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.stmp.user", "username");
```

Figura 39. Clase del gestor de correo (parte 1). (Elaboración propia)

```
        // To use SSL
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");

        session = Session.getDefaultInstance(props, new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication("teamburton96@gmail.com", "password");// Specify
            }
        });
```

Figura 40. Clase del gestor de correo (parte 2). (Elaboración propia)

3.3.1.3.4. Realizar la ventana de inicio de sesión

Al ser la primera ventana además de realizar el diseño con el Vaadin Designer se tuvo que configurar todo el esquema de navegabilidad.

Primero se preparó la clase VaadinUI, clase principal de la navegabilidad para establecer que ventana se abre al entrar en la web, en este caso se guardó una variable de sesión y en función de dicha variable te redirigía a la ventana de inicio de sesión, a la de administrador o a la de usuario.

```
@PreserveOnRefresh
@SpringUI
@Theme("valo")
public class VaadinUI extends UI {

    @Autowired
    private UsuarioRepository userRepo;

    @Autowired
    private RolesRepository rolRepo;

    private String usuario;

    @Override
    protected void init(VaadinRequest vaadinRequest)
    {
        String session = (String) VaadinService.getCurrentRequest().getWrappedSession().getAttribute("currentUser");

        if(session == null || session.isEmpty())
        {
            usuario = "";
            setContent(new Cibernauta());
        }else
        {
            setContent(new Jugador());
        }

    }
```

Figura 41. Clase VaadinUI. (Elaboración propia)

Luego se crearon las clases de la interfaz siguiendo el diagrama de clases, siendo las clases de los actores las encargadas de indicar las ventanas a las que dicho usuario tiene acceso.

```
public class Cibernauta extends Cibernauta_Ventana {
    VerticalLayout vLayout;

    public Cibernauta() {
        vLayout = new VerticalLayout();
        this.addComponent(vLayout);


        Navigator navigator = new Navigator(UI.getCurrent(), vLayout);

        navigator.addView(Iniciar_sesion.VIEW_NAME, Iniciar_sesion.class);
        navigator.addView(Registrarse.VIEW_NAME, Registrarse.class);

        navigator.navigateTo(Iniciar_sesion.VIEW_NAME);

        if (navigator.getState().isEmpty())
        {
            navigator.navigateTo(Iniciar_sesion.VIEW_NAME);
        }

    }
```

Figura 42. Clase del cibernauta. (Elaboración propia)

Por último se creó y diseñó la ventana de inicio de sesión, inicializando en dicha clase todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

```
public Iniciar_sesion()
{

    userRepo = ((VaadinUI) UI.getCurrent()).getInterfazUsuario();

    loginB.addClickListener(new Button.ClickListener()
    {

        @Override
        public void buttonClick(ClickEvent event)
        {
            String value = CrudUsuario.login(usuarioF.getValue(), passwordF.getValue(), userRepo);
            errorL.setValue(value);

            if(value.isEmpty())
            {
                errorL.setVisible(false);

                UI.getCurrent().getNavigator().destroy();
                ((VaadinUI) UI.getCurrent()).setUsuario(usuarioF.getValue());
                VaadinService.getCurrentRequest().getWrappedSession().setAttribute("currentUser",
                VaadinUI.getCurrent().setContent(new Jugador());
            }else
            {
                correctoL.setVisible(false);
                errorL.setVisible(true);
            }
        }
    });
```

Figura 43. Constructor de la clase de inicio de sesión. (Elaboración propia)

Figura 44. Ventana antigua de inicio de sesión. (Elaboración propia)

3.3.1.3.5. Realizar la ventana de registro

Se realizó de la misma manera que el inicio de sesión, se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

```
public Registrarse()
{
    userRepo = ((VaadinUI) UI.getCurrent()).getInterfazUsuario();
    rolRepo = ((VaadinUI) UI.getCurrent()).getInterfazRol();

    registrarseB.addClickListener(new Button.ClickListener()
    {

        @Override
        public void buttonClick(ClickEvent event)
        {
            String value = CrudUsuario.registro(usuarioF.getValue(), emailF
            errorL.setValue(value);

            if(value.isEmpty())
            {
                errorL.setVisible(false);
                doNavigate(Iniciar_sesion.VIEW_NAME + "/" + "registrado");
            }else
            {
                errorL.setVisible(true);
            }

        }
    });
```

Figura 45. Constructor de la clase de registro. (Elaboración propia)

Figura 46. Ventana antigua de registro. (Elaboración propia)

3.3.1.3.6. Realizar la página principal (cabecera y menú incluido)

Se siguieron los mismos pasos que con inicio de sesión y registro, en este caso no hay llamadas CRUD, solo inicializaciones de los componentes.

```
public Jugador()
{
    cabeceraLayout.addComponent(new Cabecera());
    menuLayout.addComponent(new Menu());

    Navigator navigator = new Navigator(VaadinUI.getCurrent(), principalLayout);

    navigator.addView(Pagina_principal.VIEW_NAME, Pagina_principal.class);

    navigator.navigateTo(Pagina_principal.VIEW_NAME);

    if (navigator.getState().isEmpty())
    {
        navigator.navigateTo(Pagina_principal.VIEW_NAME);
    }

}
```

Figura 47.Constructor de la clase del jugador. (Elaboración propia)

3.3.1.4. Feedback

No se ha obtenido ningún consejo de mejora tras probarlo el usuario.

3.3.2. Módulo Usuarios

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.2.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.2.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo.

3.3.2.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

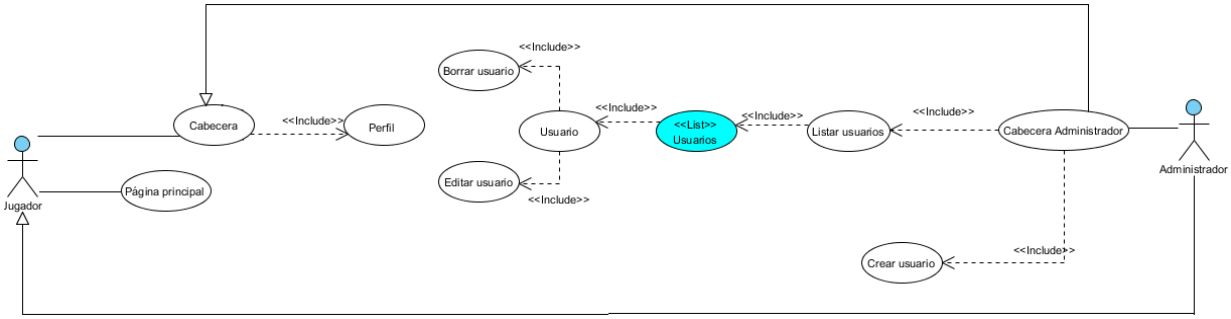

Figura 48. Diagrama de casos de uso del módulo de usuarios. (Elaboración propia)

3.3.2.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 13. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 14. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 15 Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 16. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.2.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 17. Caso de uso perfil. (Elaboración propia)

<table>
<tr><td>CDU-0005</td><td>Perfil</td></tr>
<tr><td>[Versión]</td><td>1.0 (19/04/2018)</td></tr>
<tr><td>[Dependencias]</td><td></td></tr>
<tr><td>Descripción</td><td>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar sus datos.</td></tr>
<tr><td>Precondición</td><td>Estar identificado en el sistema</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El actor Jugador (CDU-ACT-0002) solicita al sistema iniciar el procedimiento de consultar sus datos personales.</td></tr>
<tr><td>2</td><td>El sistema muestra su correo, su nombre de usuario y su contraseña.</td></tr>
<tr><td>3</td><td>Si quiere modificar su correo, el actor Jugador (CDU-ACT-0002) introduce un nuevo correo y confirma los cambios.</td></tr>
<tr><td>4</td><td>Si quiere modificar su contraseña, el actor Jugador (CDU-ACT-0002) introduce su nueva contraseña dos veces y confirma los cambios.</td></tr>
<tr><td>5</td><td>El actor Jugador (CDU-ACT-0002) guarda los cambios realizados</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>-</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>4</td><td>Si las contraseñas no coinciden, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto</td></tr>
<tr><td>5</td><td>Si el correo utilizado ya existe en la base de datos, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto</td></tr>
<tr><td>5</td><td>Si el correo utilizado es inválido, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto</td></tr>
</table>
</td></tr>
</table>

Tabla 18. Caso de uso crear usuario. (Elaboración propia)

CDU-0006	Crear usuario
[Versión]	1.0 (19/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera crear un usuario
Precondición	Estar identificado como administrador
Secuencia normal	**Paso** / **Acción** 1 / El actor Administrador (CDU-ACT-0003) introduce un email, nombre de usuario, dos veces la contraseña, si está bloqueado su acceso y que tipo de usuario es. 2 / El sistema registra los datos en la base de datos y confirma al administrador el registro.
Postcondición	Haya un nuevo usuario en la base de datos
Excepciones	**Paso** / **Acción** 2 / Si el correo utilizado ya existe en la base de datos, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 2 / Si el usuario utilizado ya existe en la base de datos, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 2 / Si las contraseñas no coinciden, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 2 / Si algún dato es inválido, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto

Tabla 19. Caso de uso consultar usuarios. (Elaboración propia)

CDU-0007	Consultar usuarios
[Versión]	1.0 (19/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando administrador quiera ver los usuarios registrados en el sistema.
Precondición	Estar identificado como administrador.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los usuarios existentes, su usuario, su email y si está bloqueado su acceso. 2 / Si el administrador quiere editar un usuario, se realiza el caso de uso Editar usuario (CDU-UC-0008) 3 / Si el administrador quiere borrar un usuario, se realiza el caso de uso Borrar usuario (CDU-UC-0009)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 20. Caso de uso editar usuario. (Elaboración propia)

CDU-0008	Editar usuario
[Versión]	1.0 (19/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera editar los datos de un usuario o durante la realización de los siguientes casos de uso: [CDU-0007] Consultar usuarios
Precondición	Estar identificado como administrador.
Secuencia normal	**Paso** / **Acción** 1 — Si quiere modificar el correo, el actor Administrador (CDU-ACT-0003) introduce un nuevo correo 2 — Si quiere modificar su contraseña, el actor Administrador (CDU-ACT-0003) introduce su nueva contraseña dos veces 3 — Si quiere bloquearle el acceso, el actor Administrador (CDU-ACT-0003) indica que el usuario no está activo (Si se está modificando a sí mismo no puede editar este campo) 4 — Si quiere cambiar el tipo de usuario, el actor Administrador (CDU-ACT-0003) elige el nuevo tipo de usuario que quiere que sea (Si se está modificando a sí mismo no puede editar este campo) 5 — El actor Administrador (CDU-ACT-0003) confirma los cambios
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** 5 — Si el correo utilizado ya existe en la base de datos, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 5 — Si el correo utilizado es inválido, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 5 — Si las contraseñas no coinciden, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto

Tabla 21. Caso de uso borrar usuario. (Elaboración propia)

CDU-0009	Borrar usuario
[Versión]	1.0 (19/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera borrar un usuario de la base de datos o durante la realización de los siguientes casos de uso: [CDU-0007] Consultar usuarios
Precondición	Estar identificado como administrador.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los datos del usuario a eliminar 2 / El actor Administrador (CDU-ACT-0003) elimina al usuario (No puede eliminarse a sí mismo) 3 / El sistema confirma la eliminación exitosa de la base de datos
Postcondición	El usuario ya no se encuentre en la base de datos
Excepciones	**Paso** / **Acción** - / -

3.3.2.1.2. Relación Diagrama – Casos de Uso

A continuación se muestran las relaciones entre los diagramas y los casos de uso de los actores.

Tabla 22. Relación diagrama – caso de uso. Módulo usuarios. (Elaboración propia)

Especificación/ Diagrama	CDU-0005	CDU-0006	CDU-0007	CDU-0008	CDU-0009
Cabecera	↑				
Perfil	↑				
Cabecera Administrador		↑	↑		
Crear usuario		↑			
Listar usuarios			↑		
Usuarios			↑		
Usuario			↑		
Editar usuario				↑	
Borrar usuario					↑

3.3.2.2. Diseño

A continuación se muestra el diagrama de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

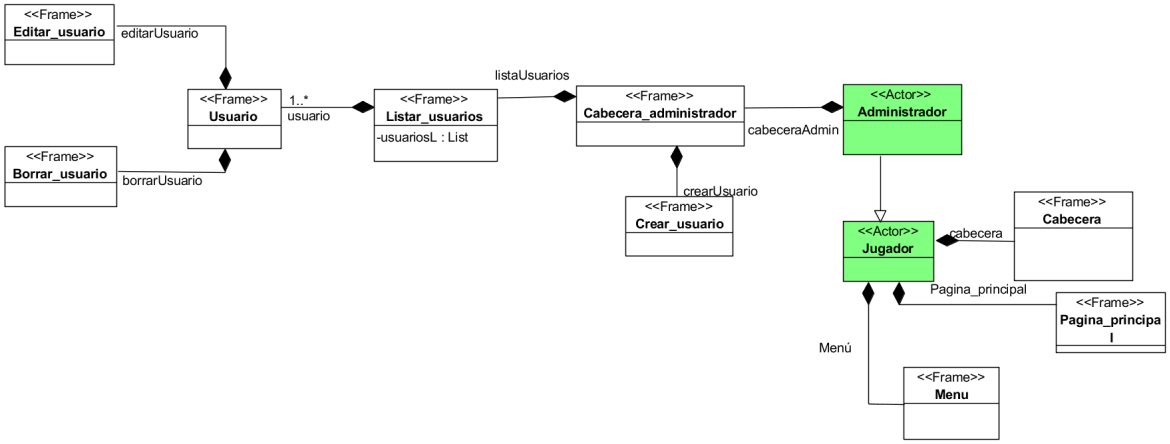

Figura 49. Diagrama de clases del módulo de usuarios. (Elaboración propia)

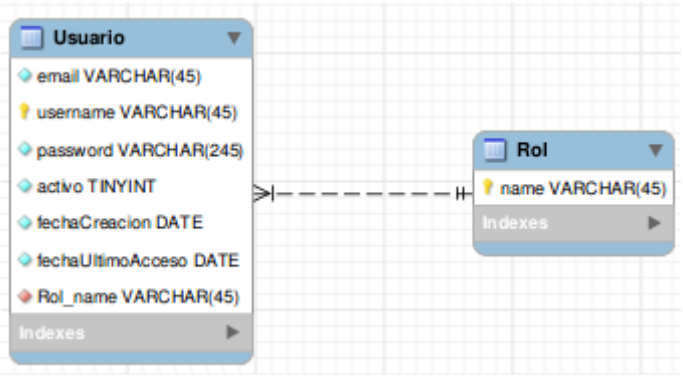

Figura 50. Diagrama de la base de datos del módulo de usuarios. (Elaboración propia)

3.3.2.3. Implementación del módulo

Solo es una semana de duración y una persona en el desarrollo, por lo que para el reparto de tareas no hay ningún problema.

La forma de resolver las distintas tareas es la misma que el anterior módulo, por lo que se resumirán aquellas que no tengan contenido adicional.

3.3.2.3.1. Probar las clases de la base de datos

Primero se comenzó realizando las pruebas y su código asociado de las clases de la base de datos, en este caso la clase Usuario y la clase Rol.

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class ModuloUsuarioApplicationTests {

    @Test
    public void testUsuarioFull()
    {
        Date fechaAhora = new Date();
        Usuario user = new Usuario("juan@gmail.com", "Pepe", "1234", true, fechaAh

        assertEquals("juan@gmail.com", user.getEmail());
        assertEquals("Pepe", user.getUsername());
        assertEquals(true, user.getActivo());
        assertEquals(fechaAhora, user.getFechaCreacion());
        assertEquals(fechaAhora, user.getFechaUltimoAcceso());
        assertTrue(CheckPassword.verifyHash("1234", user.getPassword()));
    }

    @Test
    public void testUsuarioEmail()
    {
        Date fechaAhora = new Date();
        Usuario user = new Usuario("juan@gmail.com");

        assertEquals("juan@gmail.com", user.getEmail());
    }

    @Test
    public void testUsuarioEspacios()
    {
        Date fechaAhora = new Date();
        Usuario user = new Usuario("juan@ gmail. com", "P ep e", "1 23 4", true, f
```

Figura 51. Pruebas de la clase de la base de datos de usuario. (Elaboración propia)

3.3.2.3.2. Probar el registro de un usuario por parte del administrador

El registro del administrador prácticamente el mismo que el usuario, por lo que simplemente se creó a partir de la del usuario añadiendo los atributos adicionales.

```
public static String registroAdmin(String username, String email, String password1, String password2,
{
    Usuario user;
    Date fechaRegistro = new Date();
    //Gestor_Correos correo = new Gestor_Correos();
    if(!UserDataValidator.comprobarUser(username)) return "Usuario inválido";
    if(!UserDataValidator.comprobarEmail(email)) return "Email inválido";
    if(!UserDataValidator.comprobarPassword(password1, password2)) return "Las contraseñas no coincid
    if(UserDataValidator.comprobarUsuarioBD(username, repo)) return "El usuario ya existe";
    if(UserDataValidator.comprobarEmailBD(email, repo)) return "El correo ya está en uso";
    //if(!correo.correo_registro(email)) return "Imposible acceder al correo";

    user = new Usuario(email, username, password1, activo, fechaRegistro, fechaRegistro);
    user.setRolName(rolRepo.findByName(rol).get(0));
    user.guardarUsuario(repo);

    return "";
}
```

Figura 52. Método del registro del administrador. (Elaboración propia)

3.3.2.3.3. Probar la modificación de un usuario y la modificación del propio perfil

La modificación de los usuarios era similar al registro, por lo que se diseñaron a partir de las clases de registro.

```
public static String modificarPerfil(String username, String emailOriginal, String emailNuevo, Stri
{
    Usuario user = Usuario.cargarUsuario(username, repo);
    Date fechaRegistro = new Date();
    if(!UserDataValidator.comprobarEmail(emailNuevo)) return "Email inválido";

    if(!emailOriginal.equals(emailNuevo))
    {
        if(UserDataValidator.comprobarEmailBD(emailNuevo, repo)) return "El correo ya está en uso";
        user.setEmail(emailNuevo);
    }

    if(!password1.isEmpty())
    {
        if(!UserDataValidator.comprobarPassword(password1, password2)) return "Las contraseñas no c
        user.setPassword(password1);
    }

    user.guardarUsuario(repo);

    return "";
}
```

Figura 53. Método de modificar perfil del usuario. (Elaboración propia)

```
@Test
public void testPasswordVacio()
{
    String pass1  = "  ";
    String pass2 = "    ";
    assertFalse(UserDataValidator.comprobarPassword(pass1, pass2));
}

@Test
public void testEmailExistente()
{
    String email = "juanito@ual.es";
    assertTrue(UserDataValidator.comprobarEmailBD(email, repo));
}

@Test
public void testEmailNoExistente()
{
    String email = "paco@ual.es";
    assertFalse(UserDataValidator.comprobarEmailBD(email, repo));
}

@Transactional
@Test
public void testModificarPerfilCorrecto()
{
    Usuario aux;

    assertTrue(CrudUsuario.modificarPerfil("juan", "juanito@ual.es", "juan

    aux = Usuario.cargarUsuario("juan", repo);

    assertEquals(aux.getEmail(), "juan2@gmail.com");
    assertTrue(CheckPassword.verifyHash("12345", aux.getPassword()));
}
```

Figura 54. Pruebas del método de modificar perfil del usuario. (Elaboración propia)

3.3.2.3.4. Realizar la ventana del perfil

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

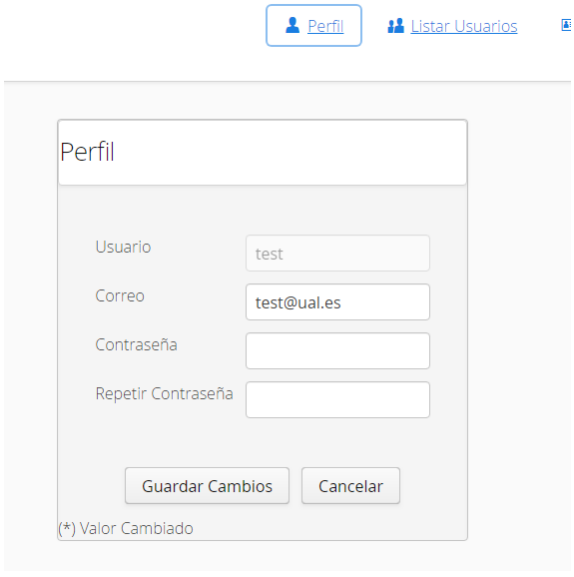

Figura 55. Ventana antigua del perfil. (Elaboración propia)

3.3.2.3.5. Realizar la ventana consulta de usuarios

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

En este caso se realizaron modificaciones en el CSS de VAADIN para el centrado de los elementos de la tabla.

Figura 56. Ventana antigua de listar usuarios. (Elaboración propia)

3.3.2.3.6. Realizar la ventana de editar usuario

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

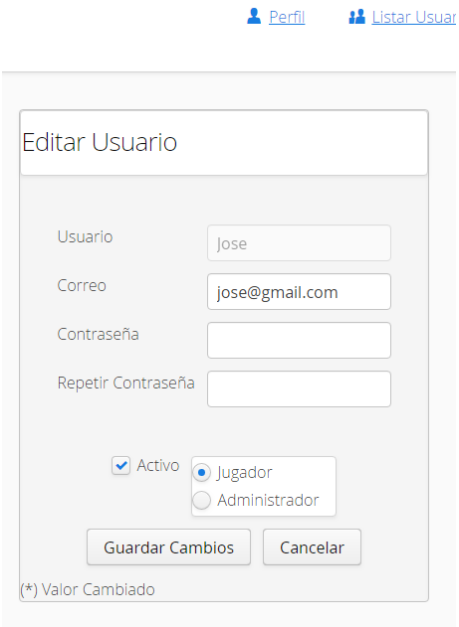

Figura 57. Ventana antigua de editar usuario del administrador. (Elaboración propia)

3.3.2.3.7. Realizar la ventana de crear usuario

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Además al ser prácticamente igual a la de editar usuario se reutilizó dicha ventana realizando varias modificaciones.

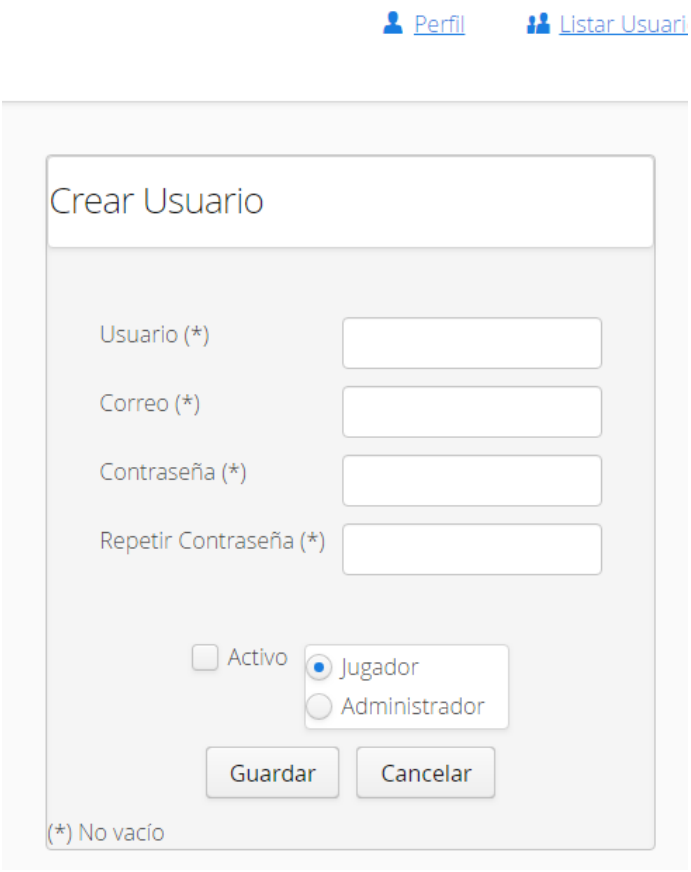

Figura 58. Ventana antigua de crear usuario. (Elaboración propia)

3.3.2.3.8. Realizar la ventana de borrar usuario.

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

No se creó una clase CRUD ni pruebas para esta funcionalidad debido a que en este momento era muy simple, solo había dos tablas por lo que nada más se veía afectado.

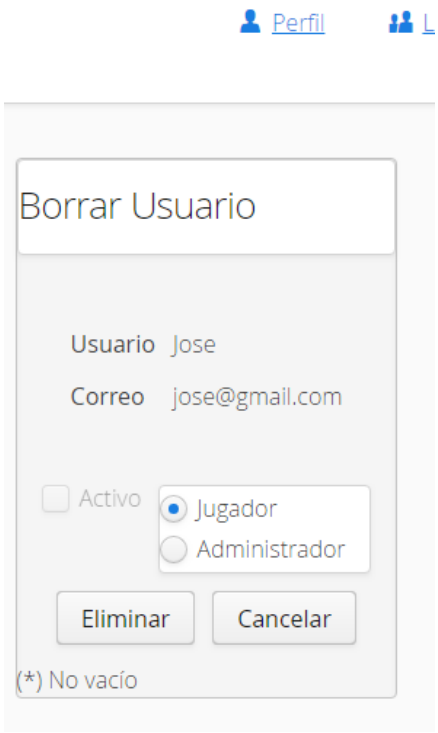

Figura 59. Ventana antigua de borrar usuario. (Elaboración propia)

3.3.2.4. Feedback

No se ha obtenido ningún consejo de mejora tras probarlo el usuario.

3.3.3. Módulo Técnico

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.3.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.3.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo.

3.3.3.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

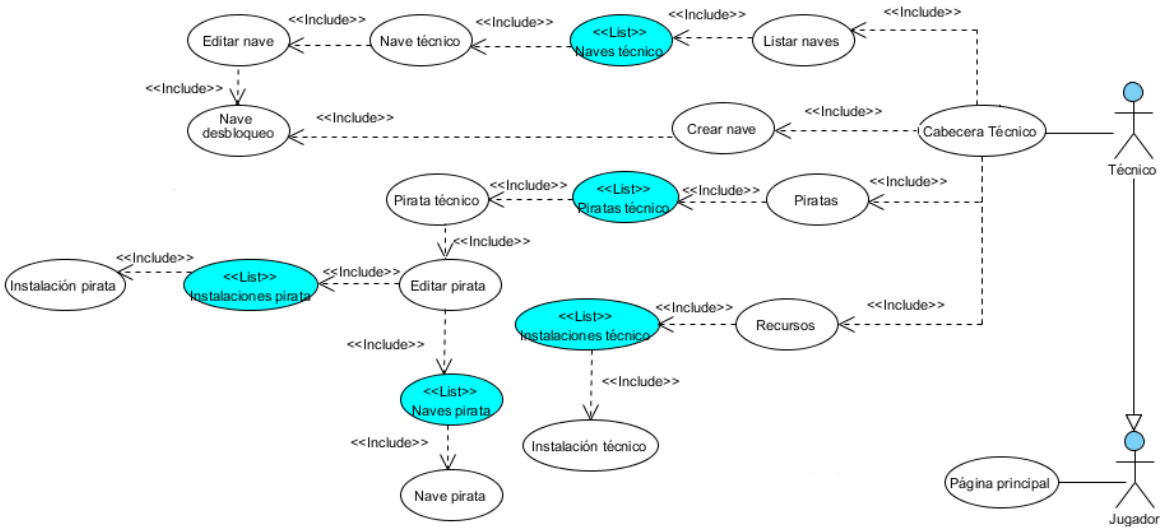

Figura 60. Diagrama de casos de uso del módulo técnico. (Elaboración propia)

3.3.3.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 23. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 24. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 25. Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 26. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 27. Caso de uso consultar lista de naves. (Elaboración propia)

<table>
<tr><td>CDU-0010</td><td>Consultar lista de naves</td></tr>
<tr><td>[Versión]</td><td>1.0 (27/04/2018)</td></tr>
<tr><td>[Dependencias]</td><td></td></tr>
<tr><td>Descripción</td><td>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera ver las naves existentes</td></tr>
<tr><td>Precondición</td><td>Estar identificado como técnico.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><th>Paso</th><th>Acción</th></tr><tr><td>1</td><td>El sistema muestra las naves existentes, su imagen, su nombre, su tipo y si está bloqueada su obtención.</td></tr><tr><td>2</td><td>Si el técnico quiere editar una nave, se realiza el caso de uso Editar nave (CDU-0011)</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>-</td></tr>
<tr><td>Excepciones</td><td><table><tr><th>Paso</th><th>Acción</th></tr><tr><td>-</td><td>-</td></tr></table></td></tr>
</table>

Tabla 28. Caso de uso editar nave. (Elaboración propia)

CDU-0011	Editar nave
[Versión]	1.0 (27/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiere editar los datos de una nave o durante la realización de los siguientes casos de uso: [CDU-0010] Consultar lista de naves
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 – Si quiere modificar la imagen, el actor Técnico (CDU-ACT-0004) sube una nueva imagen. 2 – Si quiere modificar el tipo de nave, el actor Técnico (CDU-ACT-0004) selecciona un nuevo tipo de nave. 3 – Si quiere modificar las características, el actor Técnico (CDU-ACT-0004) introduce nuevos valores para ellas. (salud, escudo, agilidad, velocidad, daño y capacidad de carga) 4 – Si quiere modificar los costes, el actor Técnico (CDU-ACT-0004) introduce nuevos valores para ellos (metal, oro y petróleo) 5 – Si quiere modificar el tiempo de construcción, el actor Técnico (CDU-ACT-0004) introduce un nuevo valor. 6 – Si quiere modificar la probabilidad de ser desbloqueada en los distintos piratas, el actor Técnico (CDU-ACT-0004) introduce nuevos valores de probabilidad de desbloqueo. 7 – Si quiere desbloquear la nave para todos los usuarios, el actor Técnico (CDU-ACT-0004) desactiva el bloqueo. 8 – El actor Técnico (CDU-ACT-0004) confirma los cambios.
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** 8 – Si algún dato es inválido, el sistema informa al técnico del problema, a continuación este caso de uso queda sin efecto

Tabla 29. Caso de uso consultar lista de piratas. (Elaboración propia)

CDU-0012	Consultar lista de piratas
[Versión]	1.0 (27/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando técnico quiera ver los piratas existentes.
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 – El sistema muestra los piratas existentes, su imagen, su nivel y el nivel de sus instalaciones. 2 – Si el técnico quiere editar un pirata, se realiza el caso de uso Editar pirata (CDU-0013)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 30. Caso de uso editar pirata. (Elaboración propia)

CDU-0013	Editar pirata
[Versión]	1.0 (27/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera editar las instalaciones o naves de un pirata o durante la realización de los siguientes casos de uso: [CDU-0012] Consultar lista de piratas
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / Si quiere modificar el nivel de las instalaciones, el actor Técnico (CDU-ACT-0004) introduce un nuevo nivel. 2 / Si quiere modificar sus naves, el actor Técnico (CDU-ACT-0004) introduce para cada nave la nueva cantidad. 3 / El actor Técnico (CDU-ACT-0004) confirma los cambios
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** - / -

Tabla 31. Caso de uso crear nave. (Elaboración propia)

CDU-0014	Crear nave
[Versión]	1.0 (27/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera crear una nave
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / El actor Técnico (CDU-ACT-0004) introduce una imagen, un nombre, el tipo de nave, la salud, el escudo, la agilidad, la velocidad, el daño, los costes de metal, oro y petróleo, el tiempo de construcción, si está bloqueada por defecto y en caso de estar bloqueada la probabilidad de ser desbloqueada en los distintos piratas. 2 / El sistema registra los datos en la base de datos y confirma al administrador el registro.
Postcondición	Haya una nueva nave en la base de datos.
Excepciones	**Paso** / **Acción** 2 / Si el nombre utilizado ya existe en la base de datos, el sistema informa al técnico del problema, a continuación este caso de uso queda sin efecto 2 / Si algún dato es inválido, el sistema informa al técnico del problema, a continuación este caso de uso queda sin efecto

Tabla 32. Caso de uso editar instalaciones. (Elaboración propia)

CDU-0015	Editar instalaciones
[Versión]	1.0 (27/04/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera editar la tasa de generación de recursos de las instalaciones
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra la generación de recursos por nivel. 2 / Si quiere modificar la producción inicial, el actor Técnico (CDU-ACT-0004) introduce para cada instalación la nueva producción inicial. 3 / Si ha modificado la producción inicial, el sistema actualiza la generación de recursos por nivel mostrada. 4 / El actor Técnico (CDU-ACT-0004) confirma los cambios.
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** - / -

3.3.3.1.2. Relación Diagrama – Casos de Uso

A continuación se muestran las relaciones entre los diagramas y los casos de uso de los actores.

Tabla 33. Relación diagrama – caso de uso, técnico. Módulo técnico. (Elaboración propia)

Especificación/ Diagrama	CDU-0010	CDU-0011	CDU-0012	CDU-0013	CDU-0014	CDU-0015
Cabecera Técnico	↑		↑		↑	↑
Listar naves	↑					
Naves técnico	↑					
Nave técnico	↑					
Editar nave		↑				
Piratas			↑			
Piratas técnico			↑			
Pirata técnico			↑			
Editar pirata				↑		
Instalaciones pirata				↑		
Instalación pirata				↑		
Naves pirata				↑		
Nave pirata				↑		
Crear nave					↑	
Recursos						↑
Instalaciones técnico						↑
Instalación técnico						↑

3.3.3.2. Diseño

A continuación se muestra el diagrama de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

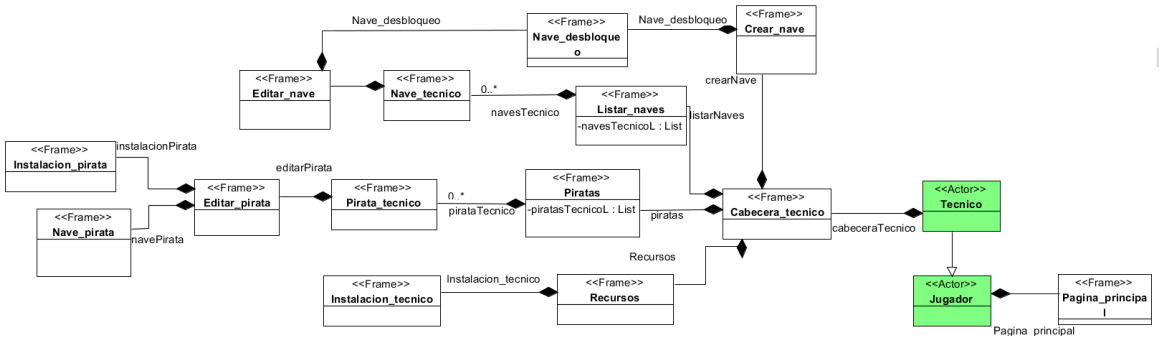

Figura 61. Diagrama de clases del módulo técnico. (Elaboración propia)

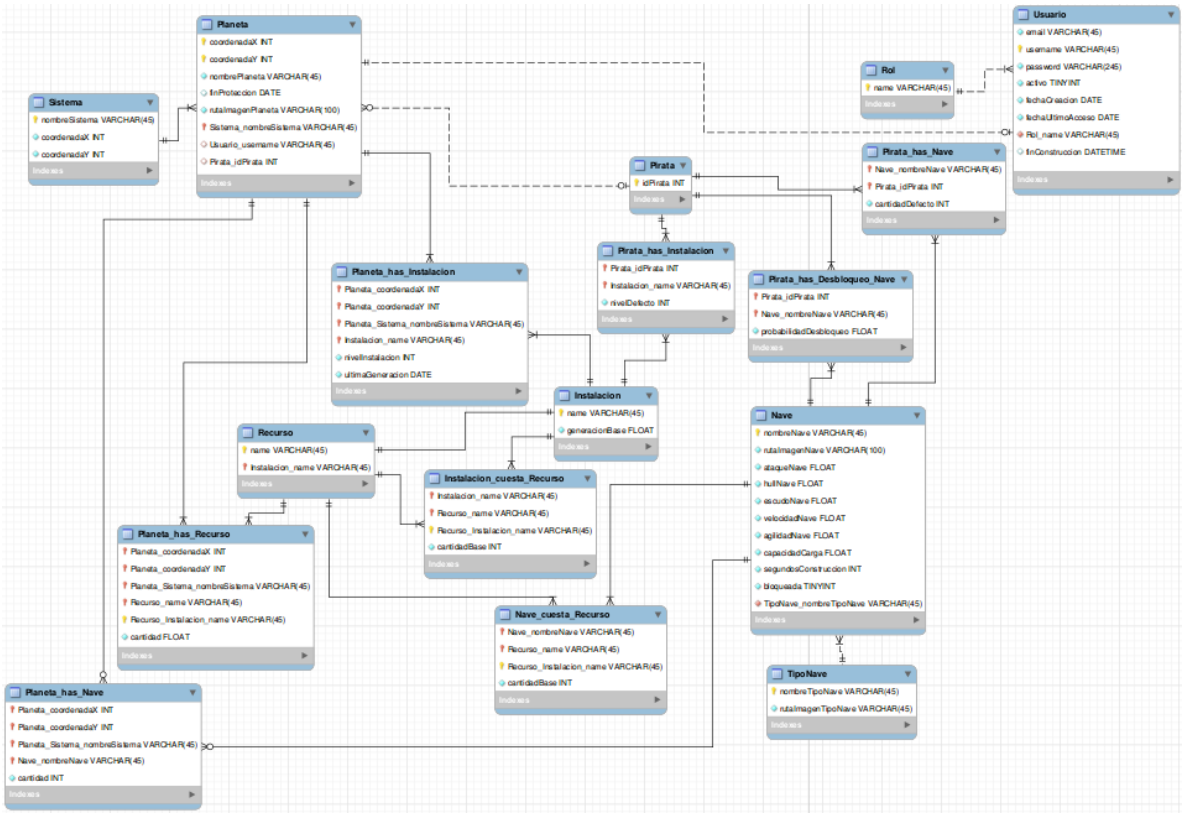

Figura 62. Diagrama de la base de datos del módulo técnico. (Elaboración propia)

3.3.3.3. Implementación del módulo

Al ser dos semanas de duración las tareas se dividieron en dos iteraciones, en la primera no se obtuvo *feedback* ya que se centró en las pruebas y su funcionalidad, por lo que no había ventanas hechas.

3.3.3.3.1. Primera iteración

Esta iteración se centró en las pruebas y su funcionalidad.

3.3.3.3.1.1. Probar las clases de la base de datos

A pesar de aumentar la complejidad de la base de datos se realizó de la misma manera que los módulos anteriores, esta complejidad no afectaba a sus pruebas asociadas.

```
@Test
public void testNaveFull()
{
    Nave nave = new Nave("Vulture", "/WEB-INF/images/image.png", 120, 200, 105, 20
    TipoNave tipoNave = new TipoNave("Caza", "/WEB-INF/images/image.png");

    nave.setTipoNavenombreTipoNave(tipoNave);

    assertEquals("Vulture", nave.getNombreNave());
    assertEquals("/WEB-INF/images/image.png", nave.getRutaImagenNave());
    assertEquals(120, nave.getAtaqueNave(), 0.0);
    assertEquals(200, nave.getHullNave(), 0.0);
    assertEquals(105, nave.getEscudoNave(), 0.0);
    assertEquals(20, nave.getVelocidadNave(), 0.0);
    assertEquals(52, nave.getAgilidadNave(), 0.0);
    assertEquals(10, nave.getCapacidadCarga(), 0.0);
    assertEquals(11, nave.getSegundosConstruccion(), 0.0);
    assertEquals((short) 1, nave.getBloqueada());
    assertEquals(tipoNave, nave.getTipoNavenombreTipoNave());
}

@Test
public void testTipoNave()
{
    TipoNave tipoNave = new TipoNave("Caza", "/WEB-INF/images/image.png");

    assertEquals("Caza", tipoNave.getNombreTipoNave());
    assertEquals("/WEB-INF/images/image.png", tipoNave.getRutaImagenTipoNave());
}

@Test
public void testRecursoFull()
{
    Recurso recurso = new Recurso("Metal", "Mina");
    Instalacion instalacion = new Instalacion("Mina", 20);
```

Figura 63. Pruebas de las clases nave, tipo nave y recursos de la base de datos. (Elaboración propia)

3.3.3.3.1.2. Probar la creación de naves

Se realizó de la misma manera que con el usuario, en este caso se identificó una nueva clase CRUD y una nueva clase de validación.

También aumentó la complejidad ya que las naves tienen mucho más campos que los usuarios, además de relaciones con muchas tablas (piratas, recursos, usuarios, etc.).

```
@Transactional
@Test
public void testCrearNaveCorrectoDesbloqueada()
{
    Nave aux;
    ArrayList<PiratahasDesbloqueoNave> desbloqueos;
    String[] probabilidadesDesbloqueo = new String[]{};

    assertTrue(CrudNave.crearNave("Type-6", "imagenUrl", "Caza", "15", "20.3", "

    aux = Nave.cargarNave("Type-6", repo);
    desbloqueos = new ArrayList<>(PiratahasDesbloqueoNave.cargarDesbloqueosNave('

    assertEquals(aux.getNombreNave(), "Type-6");
    assertEquals(aux.getRutaImagenNave(), "imagenUrl");
    assertEquals(aux.getTipoNavenombreTipoNave().getNombreTipoNave(), "Caza");
    assertEquals(aux.getSegundosConstruccion(), 15, 0.0);
    assertEquals(aux.getAtaqueNave(), 20.3, 0.0001);
    assertEquals(aux.getHullNave(), 10.2, 0.0001);
    assertEquals(aux.getEscudoNave(), 50.4, 0.0001);
    assertEquals(aux.getVelocidadNave(), 22.4, 0.0001);
    assertEquals(aux.getAgilidadNave(), 3.1, 0.0001);
    assertEquals(aux.getCapacidadCarga(), 80.9, 0.0001);
    assertTrue(desbloqueos.isEmpty());
}
```

Figura 64. Prueba de crear una nave de manera correcta y desbloqueada. (Elaboración propia)

```
public static String crearNave(String nombre, String imagen, String tipoNave, String segundosConstrucci
{
    Nave nave;
    NavecuestaRecurso naveCuesta;
    Short bloqueo = (short) 0;
    if(!NaveDataValidator.comprobarNombre(nombre)) return "Nombre inválido";
    if(!NaveDataValidator.comprobarImagen(imagen)) return "Imagen inválida";
    if(!NaveDataValidator.comprobarValorInteger(segundosConstruccion)) return "Tiempo de construcción r
    if(!NaveDataValidator.comprobarValorFloat(ataqueNave)) return "Valor de ataque no válido";
    if(!NaveDataValidator.comprobarValorFloat(hullNave)) return "Valor de salud no válido";
    if(!NaveDataValidator.comprobarValorFloat(escudoNave)) return "Valor de escudo no válido";
    if(!NaveDataValidator.comprobarValorFloat(velocidadNave)) return "Valor de velocidad no válido";
    if(!NaveDataValidator.comprobarValorFloat(agilidadNave)) return "Valor de agilidad no válido";
    if(!NaveDataValidator.comprobarValorFloat(capacidadCarga)) return "Valor de carga no válido";
    if(!NaveDataValidator.comprobarValorInteger(costeOro)) return "La cantidad de oro es incorrecta";
    if(!NaveDataValidator.comprobarValorInteger(costeMetal)) return "La cantidad de metal es incorrecta
    if(!NaveDataValidator.comprobarValorInteger(costePetroleo)) return "La cantidad de petróleo es incc
    if(!NaveDataValidator.comprobarNaveBD(nombre, repo)) return "La nave ya existe";

    for(int i = 0; i < probabilidadesDesbloqueo.length; i++)
    {
        if(!NaveDataValidator.comprobarValorFloat(probabilidadesDesbloqueo[i])) return "La " + (i+1) +"
    }

    if(bloqueada) bloqueo = (short) 1;

    nave = new Nave(nombre, imagen, Float.parseFloat(ataqueNave.replaceAll(",",".")), Float.parseFloat(
    nave.setTipoNavenombreTipoNave(tipoNaveRepo.findByNombreTipoNave(tipoNave).get(0));
    nave.guardarNave(repo);
```

Figura 65. Método de crear naves. (Elaboración propia)

```
public class NaveDataValidator
{
    public static boolean comprobarNombre(String nombre)
    {
        nombre = nombre.replaceAll("\\s+","");

        return !nombre.isEmpty();
    }

    public static boolean comprobarImagen(String imagen)
    {
        imagen = imagen.replaceAll("\\s+","");

        return !imagen.isEmpty();
    }

    public static boolean comprobarValorFloat(String numero)
    {
        numero = numero.replaceAll("\\s+","");
        numero = numero.replaceAll(",",".");

        if(FloatValidator.getInstance().isValid(numero, Locale.US))
        {

            return Float.parseFloat(numero) >= 0;
        }

        return false;

    }
```

Figura 66. Clase validadora de los datos de las naves.

3.3.3.3.1.3. Probar la modificación de naves

Muy similar a la creación de naves ya que los campos son los mismos.

```
@Transactional
@Test
public void testModificarNaveCorrectoDesbloqueada()
{
    Nave aux;
    ArrayList<PiratahasDesbloqueoNave> desbloqueos;
    String[] probabilidadesDesbloqueo = new String[]{"2","8","1","9"};

    assertTrue(CrudNave.modificarNave("Type-6", "imagenUrl", "Caza", "15",

    aux = Nave.cargarNave("Type-6", repo);
    desbloqueos = new ArrayList<>(PiratahasDesbloqueoNave.cargarDesbloqueo

    assertEquals(aux.getNombreNave(), "Type-6");
    assertEquals(aux.getRutaImagenNave(), "imagenUrl");
    assertEquals(aux.getTipoNavenombreTipoNave().getNombreTipoNave(), "Caz
    assertEquals(aux.getSegundosConstruccion(), 15, 0.0);
    assertEquals(aux.getAtaqueNave(), 20.3, 0.0001);
    assertEquals(aux.getHullNave(), 10.2, 0.0001);
    assertEquals(aux.getEscudoNave(), 50.4, 0.0001);
    assertEquals(aux.getVelocidadNave(), 22.4, 0.0001);
    assertEquals(aux.getAgilidadNave(), 3.1, 0.0001);
    assertEquals(aux.getCapacidadCarga(), 80.9, 0.0001);
    assertEquals(aux.getBloqueada(), 0);
}
```

Figura 67. Prueba de la modificación de las naves. (Elaboración propia)

3.3.3.3.1.4. Probar la modificación de las instalaciones

De las instalaciones solo se puede modificar la generación inicial, por lo que las pruebas son muy simples.

```
@Test
public void testEditarGeneracionInstalacion()
{
    Instalacion auxMetal, auxOro, auxPetroleo;

    Instalacion instalacionMetal = Instalacion.cargarInstalacion("Mina de Met
    instalacionMetal.setGeneracionBase(120);

    Instalacion instalacionOro = Instalacion.cargarInstalacion("Mina de Oro",
    instalacionOro.setGeneracionBase(90);

    Instalacion instalacionPetroleo = Instalacion.cargarInstalacion("Platafor
    instalacionPetroleo.setGeneracionBase(160);

    Recursos.updateInstalacion(instalacionMetal, instalacionOro, instalacionF

    auxMetal = Instalacion.cargarInstalacion("Mina de Metal", instalacionRepo
    auxOro = Instalacion.cargarInstalacion("Mina de Oro", instalacionRepo);
    auxPetroleo = Instalacion.cargarInstalacion("Plataforma Petrolifera", ins

    assertEquals(auxMetal.getGeneracionBase(), 120, 0.0);
    assertEquals(auxOro.getGeneracionBase(), 90, 0.0);
    assertEquals(auxPetroleo.getGeneracionBase(), 160, 0.0);

}
```

Figura 68. Prueba de la modificación de las instalaciones. (Elaboración propia)

3.3.3.3.1.5. Probar la modificación de los piratas

Estas pruebas, al igual que las pruebas de las instalaciones, son muy simples, aunque un poco más complejas, solamente se puede modificar la cantidad de naves y el nivel de las instalaciones.

```
@Test
public void testNumeroVacio()
{
    String numeroVacio = "    ";

    assertFalse(NaveDataValidator.comprobarValorInteger(numeroVacio));
}

@Test
public void testEditarPirataInstalacion()
{
    PiratahasInstalacion auxMetal, auxOro, auxPetroleo;
    PiratahasInstalacion instalacionMetal = pirataInstalacionRepo.findByP
    instalacionMetal.setNivelDefecto(1);

    PiratahasInstalacion instalacionOro = pirataInstalacionRepo.findByPir
    instalacionOro.setNivelDefecto(3);

    PiratahasInstalacion instalacionPetroleo = pirataInstalacionRepo.find
    instalacionPetroleo.setNivelDefecto(5);

    Editar_pirata.updatePirataInstalacion(instalacionMetal, instalacionOr

    auxMetal = pirataInstalacionRepo.findByPirataidPirata_Instalacionname
    auxOro = pirataInstalacionRepo.findByPirataidPirata_Instalacionname(1
    auxPetroleo = pirataInstalacionRepo.findByPirataidPirata_Instalacionn

    assertEquals(auxMetal.getNivelDefecto(), 1);
    assertEquals(auxOro.getNivelDefecto(), 3);
    assertEquals(auxPetroleo.getNivelDefecto(), 5);

}
```

Figura 69. Pruebas de la modificación de las instalaciones de los piratas. (Elaboración propia)

3.3.3.3.2. Segunda iteración

En esta iteración se realizaron las ventanas del módulo con su código asociado.

3.3.3.3.2.1. Realizar las ventanas de crear nave

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Crear nave y editar nave al ser más complejas tuvieron más trabajo en la inicialización de los componentes.

```
public Crear_nave()
{
    String filePath = VaadinService.getCurrent().getBaseDirectory().getAbsolutePath() + "/VAADIN/
    ArrayList<Pirata> piratas;
    ArrayList<TipoNave> tiposNave;
    configuracionBotones();
    naveRepo = ((VaadinUI) UI.getCurrent()).getInterfazNave();
    tipoRepo = ((VaadinUI) UI.getCurrent()).getInterfazTipo();
    naveCuestaRepo = ((VaadinUI) UI.getCurrent()).getInterfazNaveCuesta();
    pirataDesbloqueoRepo = ((VaadinUI) UI.getCurrent()).getInterfazPirataDesbloqueo();
    pirataRepo = ((VaadinUI) UI.getCurrent()).getInterfazPirata();

    tiposNave = new ArrayList<>(tipoRepo.findAll());
    tipoNaveCombo.setItems(tiposNave);
    tipoNaveCombo.setSelectedItem(tiposNave.get(0));

    final ImageUploader uploader = new ImageUploader(imagenNave, subirImagen, errorL, "", "");
    subirImagen.setReceiver(uploader);
    subirImagen.addSucceededListener(uploader);
    subirImagen.addStartedListener(uploader);

    piratas = new ArrayList<>(pirataRepo.findAll());

    for(int i = 0; i < piratas.size(); i++)
    {
        layoutDesbloqueo.addComponent(new Nave_desbloqueo(piratas.get(i).getIdPirata(), "100"));
    }
```

Figura 70. Constructor de la clase de crear nave. (Elaboración propia)

Además, al tener la posibilidad de modificar la imagen se tuvo que crear una clase (ImageUploader) que gestionara dicha subida para controlar los archivos que se suben.

```
public OutputStream receiveUpload(String filename, String mimeType)
{
 // Create and return a file output stream
     FileOutputStream fos = null; // Output stream to write to
     try
     {
         // Open the file for writing.
         filePath = VaadinService.getCurrent().getBaseDirectory().getAbsolutePath(
         file = new File(filePath + filename);
         this.imageName = filename;
         fos = new FileOutputStream(file);
     } catch (final java.io.FileNotFoundException e)
     {
         return null;
     }
     return fos; // Return the output stream to write to
}

public void uploadSucceeded(SucceededEvent event)
{
 // Show the uploaded file in the image viewer
    imagen.setSource(new FileResource(file));
    imagen.setWidthUndefined();
    imagen.setHeightUndefined();
}

public void uploadStarted(StartedEvent event)
{
    String contentType = event.getMIMEType();
```

Figura 71. Métodos de la clase ImageUpload. (Elaboración propia)

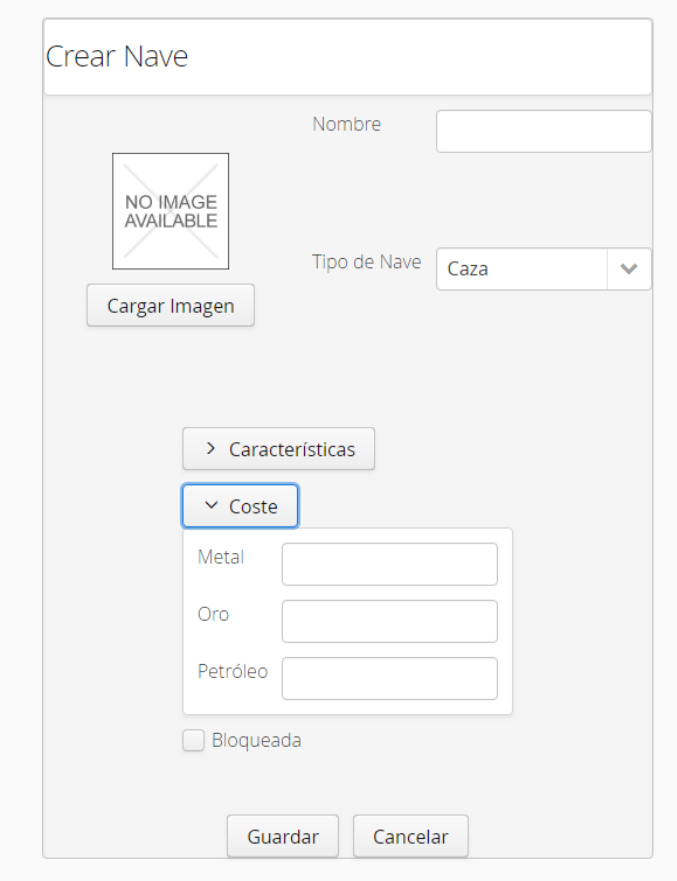

Figura 72. Ventana antigua de crear nave. (Elaboración propia)

3.3.3.3.2.2. Realizar la ventana de listar naves

Ventana prácticamente igual que listar usuarios, se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

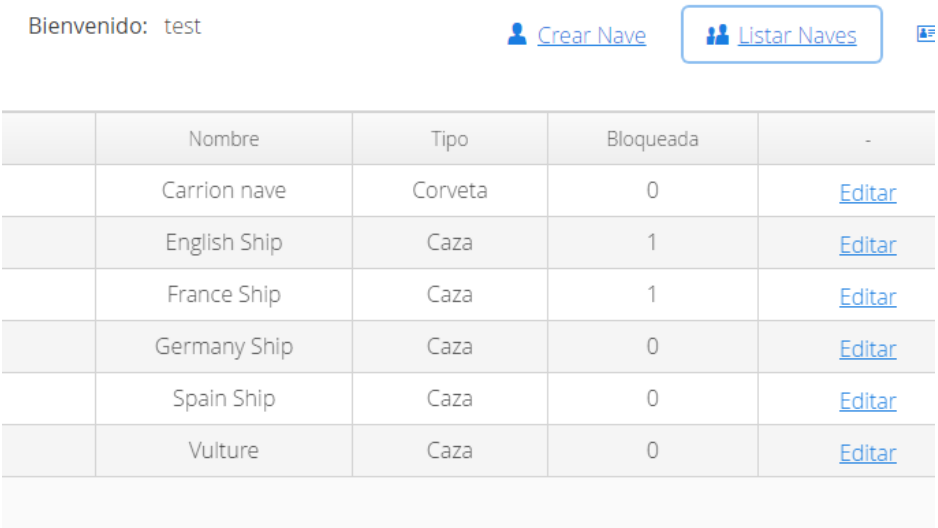

Figura 73. Ventana antigua de listar naves. (Elaboración propia)

3.3.3.3.2.3. Realizar la ventana de editar nave

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Al ser prácticamente igual que crear nave se reutilizó, por lo que las diferencias entre ambas son mínimas.

Figura 74. Ventana antigua de modificar nave. (Elaboración propia)

3.3.3.3.2.4. Realizar las ventanas de recursos

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

En esta ventana se añadieron ejemplos de la generación de recursos por nivel que se actualizan en función de la generación de recursos por defecto. Esto se hace colocando un listener los campos de texto.

```
tasaInstalacionF.addValueChangeListener(new ValueChangeListener()
{
    @Override
    public void valueChange(ValueChangeEvent event)
    {
        if(NaveDataValidator.comprobarValorFloat(tasaInstalacionF.getValue()))
        {
            recursos.setRecursosInfo(tipoInstalacion, Float.parseFloat(tasaInstalacionF.getValue()
        }
```

Figura 75. Listener de la actualización de recursos generados por nivel. (Elaboración propia)

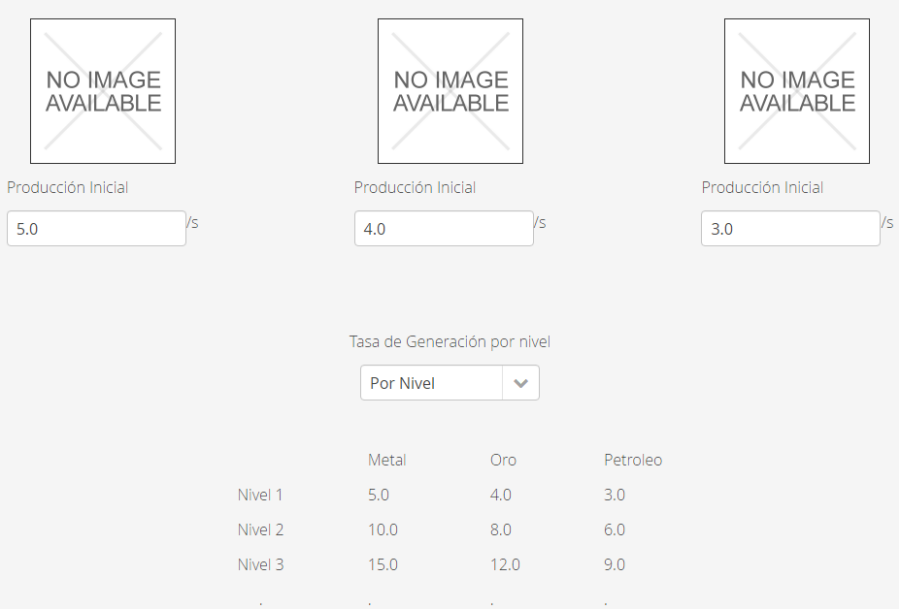

Figura 76. Ventana antigua de modificar instalaciones. (Elaboración propia)

3.3.3.3.2.5. Realizar la ventana de listar piratas

Ventana prácticamente igual que listar usuarios, se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

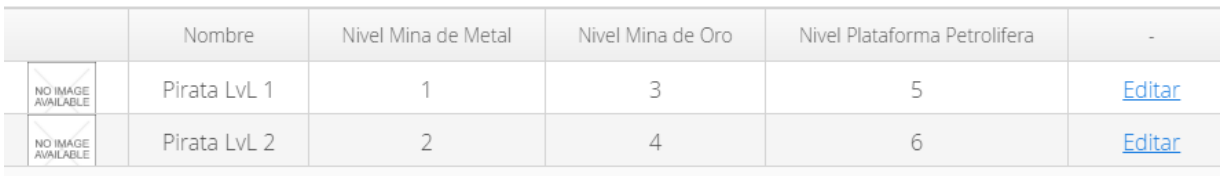

Figura 77. Ventana antigua de listar piratas. (Elaboración propia)

3.3.3.3.2.6. Realizar las ventanas de editar piratas

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

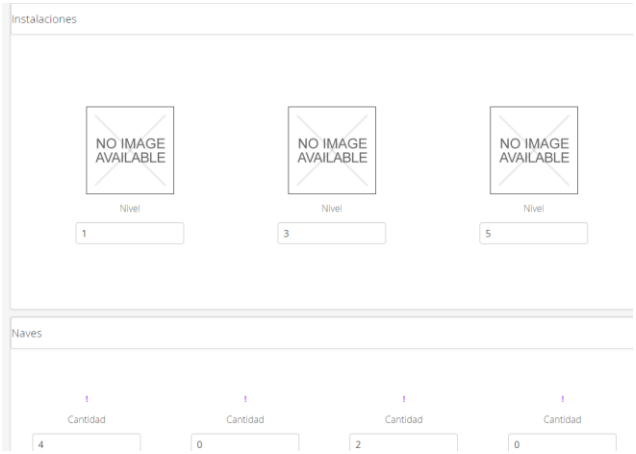

Figura 78. Ventana antigua de modificar piratas. (Elaboración propia)

Esta captura fue hecha tras cambiar las rutas de las imágenes por lo que las imágenes de las naves no aparecen.

3.3.3.4. Feedback

No se ha obtenido ningún consejo de mejora tras probarlo el usuario.

3.3.4. Módulo Naves

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.4.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.4.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo.

3.3.4.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

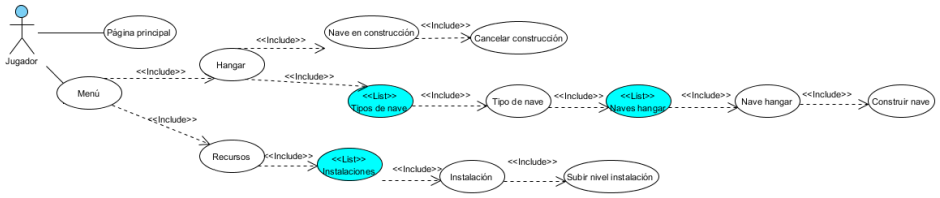

Figura 79. Diagrama de casos de uso del módulo naves. (Elaboración propia)

3.3.4.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 34. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 35. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 36. Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 37. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (11/04/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.4.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 38. Caso de uso construcción de naves. (Elaboración propia)

CDU-0016	Construcción de naves
[Versión]	1.0 (14/05/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera gestionar la construcción de naves.
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los tipos de naves existentes. 2 / Si hay naves en construcción, el sistema muestra las naves en construcción y el tiempo restante 3 / Si el usuario desea cancelar la construcción de naves en curso, se realiza el caso de uso Cancelar construcción (CDU-0018) 4 / Si el usuario desea construir naves, se realiza el caso de uso Construir naves (CDU-0017)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 39. Caso de uso construir naves. (Elaboración propia)

CDU-0017	Construir naves
[Versión]	1.0 (14/05/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera construir naves. o durante la realización de los siguientes casos de uso: [CDU-0016] Construcción de naves
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) selecciona el tipo de nave que le interesa construir. 2 / El sistema muestra las naves existentes para el tipo de nave seleccionada. 3 / El actor Jugador (CDU-ACT-0002) selecciona la nave que desea construir. (No puede seleccionar las naves que no tiene desbloqueadas) 4 / El actor Jugador (CDU-ACT-0002) indica el número de naves que desea construir 5 / El actor Jugador (CDU-ACT-0002) confirma la construcción 6 / El sistema le descuenta los recursos necesarios y comienza la construcción de las naves
Postcondición	Haya naves en construcción
Excepciones	**Paso** / **Acción** 6 / Si hay naves en construcción, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto

Tabla 40. Caso de uso cancelar construcción. (Elaboración propia)

CDU-0018	Cancelar construcción
[Versión]	1.0 (14/05/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera cancelar la construcción de las naves o durante la realización de los siguientes casos de uso: [CDU-0016] Construcción de naves
Precondición	Estar identificado en el sistema y haya naves construyéndose.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) indica que desea cancelar la producción de naves actual 2 / El sistema cancela la construcción de naves y le devuelve los recursos invertidos.
Postcondición	Las naves no estén en construcción
Excepciones	**Paso** / **Acción** - / -

Tabla 41. Caso de uso subir nivel instalación. (Elaboración propia)

CDU-0019	Subir nivel instalación
[Versión]	1.0 (14/05/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiere subir de nivel alguna instalación
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestras las instalaciones existentes 2 / Si posee suficientes recursos, el actor Jugador (CDU-ACT-0002) selecciona la instalación que desea subir de nivel 3 / El sistema descuenta los recursos necesarios y sube de nivel la instalación
Postcondición	Generar más recursos
Excepciones	**Paso** / **Acción** - / -

3.3.4.1.2. Relación Diagrama – Casos de Uso

A continuación se muestran las relaciones entre los diagramas y los casos de uso de los actores.

Tabla 42. Relación diagrama – caso de uso. Módulo naves. (Elaboración propia)

Especificación/ Diagrama	CDU-0016	CDU-0017	CDU-0018	CDU-0019
Menú	↑			↑
Hangar	↑			
Nave en construcción	↑			
Cancelar construcción			↑	
Tipos de nave		↑		
Tipo de nave		↑		
Naves hangar		↑		
Nave hangar		↑		
Construir nave		↑		
Recursos				↑
Instalaciones				↑
Instalación				↑
Subir nivel instalación				↑

3.3.4.2. Diseño

A continuación se muestra el diagrama de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

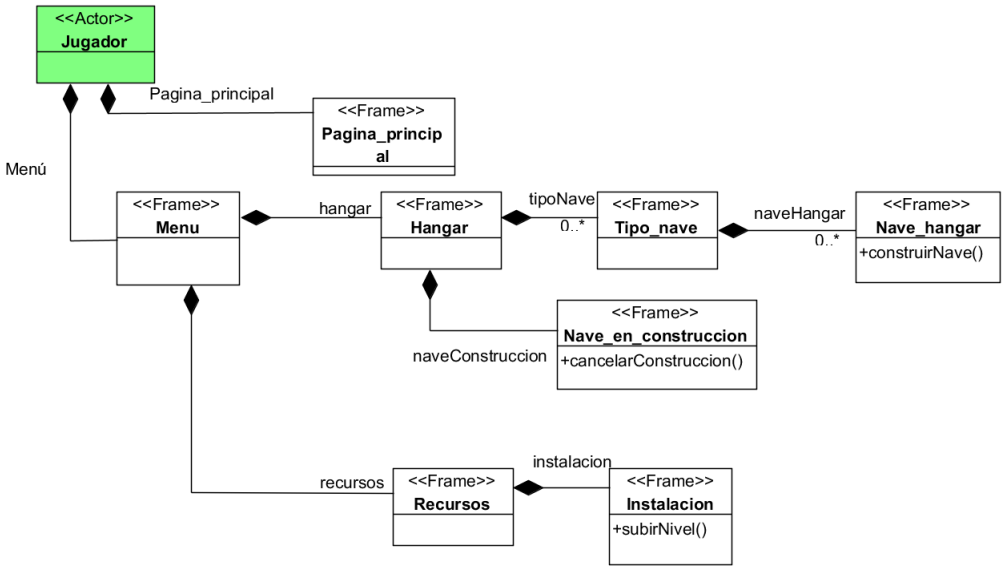

Figura 80. Diagrama de clases del módulo naves. (Elaboración propia)

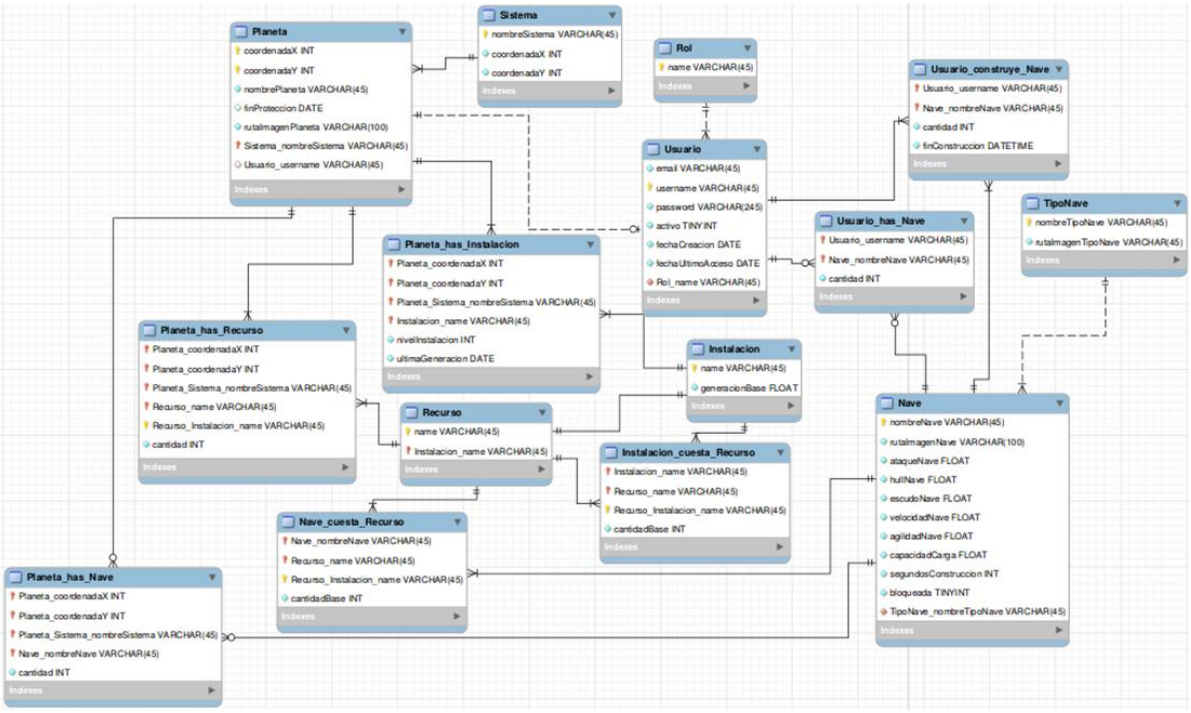

Figura 81. Diagrama de la base de datos del módulo naves. (Elaboración propia)

3.3.4.3. Implementación del módulo

Solo es una semana de duración y una persona en el desarrollo, por lo que para el reparto de tareas no hay ningún problema, las tareas identificadas son:

3.3.4.3.1. Probar las clases de la base de datos

Se realizó de la misma manera que los módulos anteriores, además la mayoría de las pruebas ya se hicieron en el módulo anterior.

```java
@Test
public void testNaveFull()
{
    Nave nave = new Nave("Vulture", "/WEB-INF/images/image.png", 120, 200, 105,
    TipoNave tipoNave = new TipoNave("Caza", "/WEB-INF/images/image.png");

    nave.setTipoNavenombreTipoNave(tipoNave);

    assertEquals("Vulture", nave.getNombreNave());
    assertEquals("/WEB-INF/images/image.png", nave.getRutaImagenNave());
    assertEquals(120, nave.getAtaqueNave(), 0.0);
    assertEquals(200, nave.getHullNave(), 0.0);
    assertEquals(105, nave.getEscudoNave(), 0.0);
    assertEquals(20, nave.getVelocidadNave(), 0.0);
    assertEquals(52, nave.getAgilidadNave(), 0.0);
    assertEquals(10, nave.getCapacidadCarga(), 0.0);
    assertEquals(11, nave.getSegundosConstruccion(), 0.0);
    assertEquals((short) 1, nave.getBloqueada());
    assertEquals(tipoNave, nave.getTipoNavenombreTipoNave());
}

@Test
public void testTipoNave()
{
    TipoNave tipoNave = new TipoNave("Caza", "/WEB-INF/images/image.png");

    assertEquals("Caza", tipoNave.getNombreTipoNave());
    assertEquals("/WEB-INF/images/image.png", tipoNave.getRutaImagenTipoNave());
}

@Test
public void testUsuarioHasNave()
{
    Date fechaAhora = new Date();
    UsuarioHasNave usuarioNave = new UsuarioHasNave("Pepe", "Vulture", 20);
```

Figura 82. Pruebas de las clases nave, tipo nave y usuario de la base de datos. (Elaboración propia)

3.3.4.3.2. Probar la construcción de naves

Pruebas no muy complejas, lo más complejo fue crear el método en la clase CRUD debido a la gran cantidad de tablas que se veían afectadas por esta funcionalidad.

```
public static String construirNave(Nave nave, Usuario usuario, String cantidad, Usuarioconstruye
{
    Date fechaFinConstruccion;
    Calendar calendar = Calendar.getInstance();
    Planeta planeta = planetaRepo.findByUsuarioUsername(usuario);
    PlanetahasRecurso cantidadMetal = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoorden
    PlanetahasRecurso cantidadOro = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoordenad
    PlanetahasRecurso cantidadPetroleo = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoor
    int costeMetal = naveCuestaRepo.findByRecursoname_NavenombreNave(nave.getNombreNave(), "Meta
    int costeOro = naveCuestaRepo.findByRecursoname_NavenombreNave(nave.getNombreNave(), "Oro").
    int costePetroleo = naveCuestaRepo.findByRecursoname_NavenombreNave(nave.getNombreNave(), "P
    UsuarioconstruyeNave naveConstruida;

    if(!construyeRepo.findByUsuariousername(usuario.getUsername()).isEmpty()) return "Ya hay nav
    if(nave.getBloqueada() == 1 && usuarioNaveRepo.findByNavenombreNaveUsuarioUsername(nave.getN
    if(!NaveDataValidator.comprobarValorInteger(cantidad)) return "La cantidad de naves introduc

    costeMetal *= Integer.parseInt(cantidad);
    costeOro *= Integer.parseInt(cantidad);
    costePetroleo *= Integer.parseInt(cantidad);

    if(costeMetal > cantidadMetal.getCantidad()) return "No posees suficiente metal";
    if(costeOro > cantidadOro.getCantidad()) return "No posees suficiente oro";
    if(costePetroleo > cantidadPetroleo.getCantidad()) return "No posees suficiente petroleo";

    calendar.add(Calendar.SECOND, Integer.parseInt(cantidad) * nave.getSegundosConstruccion());
    fechaFinConstruccion = calendar.getTime();

    cantidadMetal = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoordenadaY(), planeta.ge
    cantidadMetal.setCantidad(cantidadMetal.getCantidad()-costeMetal);
```

Figura 83. Método de construir naves. (Elaboración propia)

3.3.4.3.3. Probar la cancelación de naves e identificación de naves ya construidas

Estas pruebas aumentaron la complejidad debido a la gestión de los tiempos, cuando una construcción se cancela se debe de comprobar el tiempo restante para ver cuantas naves se construyeron, cuantas no se construyeron y cuantos recursos se le deben de devolver al usuario.

Para ello dentro de la clase CRUD se creó un método para comprobar qué naves ya se han construido, otro método para añadir las naves construidas a la cuenta del usuario y otro método para cancelar la construcción de las naves.

Para comprobar tanto las naves ya construidas así como si el tiempo en el que terminará la construcción es posterior al tiempo actual, se le resta el tiempo actual ($T_{const} - T_{act}$) > 0. El valor obtenido ahí se divide entre 1000 para pasarlo a segundos ($s_{restantes}$).

Por último, como sabemos los segundos que tarda una nave en construirse y cuantas faltan por ser construidas, obtenemos los segundos totales que tardarán en ser construidas las naves. Ese valor se lo restamos a $s_{restantes}$ y obtenemos los segundos que han pasado, así que al dividirlo entre los segundos que tarda una nave en ser construida obtenemos las naves construidas.

La fórmula quedaría:

$$s_{restantes} = ((T_{const} - T_{act})/1000)$$

$$n_{construidas=} ((s_{const} * n_{restantes}) - s_{restantes})/n_{rest}$$

Ya con estos datos podemos calcular todo lo necesario para completar la cancelación de la construcción de naves de manera correcta.

```
public static String cancelarConstruccion(Usuario usuario, UsuarioconstruyeNaveRepository construyeRepo,
{
    Date fechaAhora = new Date();
    int seconds, segundosConstruccion, navesConstruidas, beneficiosMetal, beneficiosOro, beneficiosPetro
    ArrayList<UsuarioconstruyeNave> naveConstruyendo = new ArrayList<>(construyeRepo.findByUsuariouserna
    Planeta planeta = planetaRepo.findByUsuarioUsername(usuario);
    PlanetahasRecurso recursoMetal, recursoOro, recursoPetroleo;

    if(naveConstruyendo.isEmpty()) return "No hay naves construyendose";
    if(naveConstruyendo.get(0).getFinConstruccion().before(fechaAhora))
    {
        añadirNaves(usuario, naveConstruyendo.get(0).getNavenombreNave(), naveConstruyendo.get(0).getCan
        construyeRepo.delete(naveConstruyendo.get(0));
        return "Las naves ya terminaron de construirse";
    }

    seconds = (int) (naveConstruyendo.get(0).getFinConstruccion().getTime()-fechaAhora.getTime())/1000;
    segundosConstruccion = naveConstruyendo.get(0).getNave().getSegundosConstruccion();
    navesConstruidas = ((segundosConstruccion * naveConstruyendo.get(0).getCantidad()) - seconds)/segund
    añadirNaves(usuario, naveConstruyendo.get(0).getNavenombreNave(), navesConstruidas, planeta, planeta

    recursoMetal = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoordenadaX(), planeta.getCoordena
    recursoOro = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoordenadaX(), planeta.getCoordenada
    recursoPetroleo = planetaRecursoRepo.findByPlanetaRecurso(planeta.getCoordenadaX(), planeta.getCoord

    costeMetal = naveCuestaRepo.findByRecursoname_NavenombreNave(naveConstruyendo.get(0).getNavenombreNa
    costeOro = naveCuestaRepo.findByRecursoname_NavenombreNave(naveConstruyendo.get(0).getNavenombreNave
    costePetroleo = naveCuestaRepo.findByRecursoname_NavenombreNave(naveConstruyendo.get(0).getNavenombr

    beneficiosMetal = (naveConstruyendo.get(0).getCantidad() - navesConstruidas) * costeMetal;
```

Figura 84. Método de cancelar la construcción de naves. (Elaboración propia)

3.3.4.3.4. Probar la generación de recursos

Las instalaciones producen recursos cada segundo, el problema es que no se puede tener a la base de datos calculando valores cada segundo por cada usuario, por lo que la solución a la que se ha llegado es que al entrar en ventanas concretas se llama a un método que en función de la última vez que se calcularon te calcula los recursos generados.

```
public static String generarRecursos(Planeta planeta, PlanetahasRecursoRepository planeta
{
    Timestamp fechaAhora = new Timestamp(System.currentTimeMillis());
    int generacionSegundoMetal, generacionSegundoOro, generacionSegundoPetroleo, metalGen
    PlanetahasRecurso metalPlaneta, oroPlaneta, petroleoPlaneta;
    PlanetahasInstalacion instalacionMetal = planetaInstalacionRepo.findByInstalacionname
    PlanetahasInstalacion instalacionOro = planetaInstalacionRepo.findByInstalacionnamePl
    PlanetahasInstalacion instalacionPetroleo = planetaInstalacionRepo.findByInstalacionn

    generacionSegundoMetal = instalacionMetal.getNivelInstalacion() * (int) instalacionMe
    generacionSegundoOro = instalacionOro.getNivelInstalacion() * (int) instalacionOro.ge
    generacionSegundoPetroleo = instalacionPetroleo.getNivelInstalacion() * (int) instala

    segundos =  (int) (fechaAhora.getTime()-instalacionMetal.getUltimaGeneracion().getTim

    instalacionMetal.setUltimaGeneracion(fechaAhora);
    instalacionOro.setUltimaGeneracion(fechaAhora);
    instalacionPetroleo.setUltimaGeneracion(fechaAhora);

    metalGenerado = generacionSegundoMetal * segundos;
    oroGenerado = generacionSegundoOro * segundos;
    petroleoGenerado = generacionSegundoPetroleo * segundos;

    metalPlaneta = planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Metal");
    oroPlaneta = planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Oro");
    petroleoPlaneta = planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Petroleo");

    metalPlaneta.setCantidad(metalPlaneta.getCantidad() + metalGenerado);
    oroPlaneta.setCantidad(oroPlaneta.getCantidad() + oroGenerado);
```

Figura 85. Método de generar recursos. (Elaboración propia)

Para las pruebas simplemente se establece una fecha de generación anterior y se llama al método sabiendo con antelación el tiempo que ha pasado.

```
@Transactional
@Test
public void testGenerarRecursos()
{
    Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.SECOND, -280);
    Date fechaPosterior = calendar.getTime();
    int recursosMetal = (int) instalacion1.getGeneracionBase() * planetaInstalacion1.getNi
    int recursosOro = (int) instalacion2.getGeneracionBase() * planetaInstalacion2.getNive
    int recursosPetroleo = (int) instalacion3.getGeneracionBase() * planetaInstalacion3.ge

    planetaInstalacion1.setUltimaGeneracion(fechaPosterior);
    planetaInstalacion2.setUltimaGeneracion(fechaPosterior);
    planetaInstalacion3.setUltimaGeneracion(fechaPosterior);

    planetaInstalacionRepo.save(planetaInstalacion1);
    planetaInstalacionRepo.save(planetaInstalacion2);
    planetaInstalacionRepo.save(planetaInstalacion3);

    assertTrue(CrudRecurso.generarRecursos(planeta, planetaRecursoRepo, planetaInstalacionI

    assertEquals(recursosMetal, planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Me
    assertEquals(recursosOro, planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Oro"
    assertEquals(recursosPetroleo, planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas",
}
```

Figura 86. Pruebas de generar recursos. (Elaboración propia)

3.3.4.3.5. Probar la subida de nivel de las instalaciones

Pruebas sencillas que simplemente cambian el nivel de la instalación de un planeta concreto descontando los recursos necesarios si se dispone de ellos.

```
@Transactional
@Test
public void testSubirNivel4Correcto()
{
    PlanetahasRecurso auxMetal = planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Metal");
    PlanetahasRecurso auxOro = planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Oro");
    PlanetahasRecurso auxPetroleo = planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Petro
    auxMetal.setCantidad(480);
    auxOro.setCantidad(720);
    auxPetroleo.setCantidad(880);
    planetaRecursoRepo.save(auxMetal);
    planetaRecursoRepo.save(auxOro);
    planetaRecursoRepo.save(auxPetroleo);

    planetaInstalacion1.subirNivelInstalacion();
    planetaInstalacion1.subirNivelInstalacion();
    planetaInstalacion1.subirNivelInstalacion();

    planetaInstalacionRepo.save(planetaInstalacion1);

    assertTrue(CrudRecurso.subirNivel(usuario, "Mina de Metal", planetaRepo, planetaInstalacionRe

    assertEquals(5, planetaInstalacionRepo.findByInstalacionnamePlaneta("Mina de Metal", 0, 0, "A
    assertEquals(0, planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Metal").getCantidad()
    assertEquals(0, planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Oro").getCantidad());
    assertEquals(0, planetaRecursoRepo.findByPlanetaRecurso(0, 0, "Atlas", "Petroleo").getCantida
}
```

Figura 87. Prueba de subir de nivel una instalación. (Elaboración propia)

3.3.4.3.6. Realizar la ventana de construcción de naves

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Esta ventana es de las más complejas de la aplicación, en función del tipo de nave seleccionado, carga la lista de naves de ese tipo mostrando la primera. Pulsando las flechas laterales se va navegando por esa lista controlando no salirse al pulsar rápidamente.

Además, al cargar una nave muestra sus datos y en función de las naves indicadas para construir actualiza los costes de esa nave.

```
for(int i = 0; i < tipoNaves.size(); i++)
{
    aux = new Tipo_nave(tipoNaves.get(i), i);

    tipoNaveLayout.addComponent(aux);

    aux.getImage().addClickListener(new ClickListener()
    {
        @Override
        public void click(ClickEvent event)
        {
            construirLayout.setVisible(true);
            nombreTipoL.setValue(tipoNaves.get(Integer.parseInt(event.getComponent().
            posicion = 0;
            naves = new ArrayList<>(naveRepo.findByNombreTipoNave(tipoNaves.get(Integ
            comprobarFlechaDer();
            comprobarFlechaIzq();
            naveLayout.removeAllComponents();
            naveLayout.addComponent(new Nave_hangar(naves.get(posicion)));
            actualizarDatosNave(naves.get(0));
            construirF.setValue("");
        }
    });
}

flechaIzq.addClickListener(new ClickListener()
```

Figura 88. Carga de las naves de un tipo al seleccionar el tipo de nave. (Elaboración propia)

```
flechaIzq.addClickListener(new ClickListener()
{

    @Override
    public void click(ClickEvent event)
    {
        posicion -= 1;
        if(posicion < 0)posicion = 0;
        naveLayout.removeAllComponents();
        naveLayout.addComponent(new Nave_hangar(naves.get(posicion)));
        comprobarFlechaIzq();
        comprobarFlechaDer();
        actualizarDatosNave(naves.get(posicion));
        construirF.setValue("");
    }

});

flechaDer.addClickListener(new ClickListener()
{

    @Override
    public void click(ClickEvent event)
    {
        posicion += 1;
        if(posicion > naves.size())posicion = naves.size();
        naveLayout.removeAllComponents();
        naveLayout.addComponent(new Nave_hangar(naves.get(posicion)));
        comprobarFlechaIzq();
        comprobarFlechaDer();
        actualizarDatosNave(naves.get(posicion));
        construirF.setValue("");
    }
```

Figura 89. Navegabilidad entre las naves existentes. (Elaboración propia)

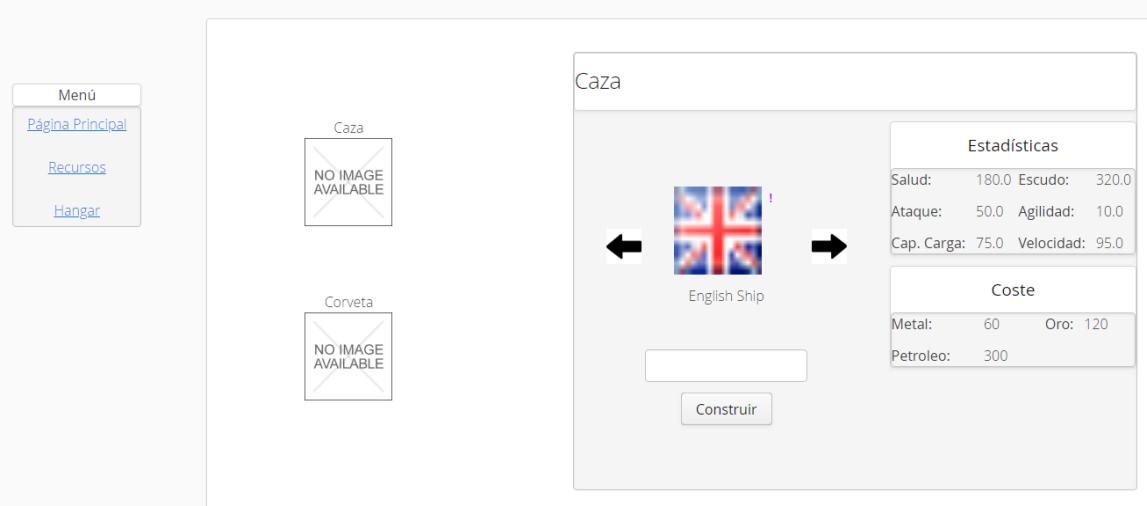

Figura 90. Ventana antigua de hangar. (Elaboración propia)

3.3.4.3.7. Realizar la ventana cancelación de naves

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Se trata de una subventana de hangar que muestra la nave en construcción y el tiempo restante.

```
public Nave_en_construccion(VerticalLayout layoutPadre, Label errorL,UsuarioconstruyeNaveReposi
{
    Date fechaAhora = new Date();
    this.layoutPadre = layoutPadre;
    this.errorL = errorL;
    UsuarioconstruyeNave naveConstruida = construyeRepo.findByUsuariousername(((VaadinUI) UI.ge
    long tiempoRestante = (naveConstruida.getFinConstruccion().getTime()-fechaAhora.getTime());
    int hours   = (int)((tiempoRestante % DAY) / HOUR);
    int minutes = (int)((tiempoRestante % HOUR) / MINUTE);
    int seconds = (int)((tiempoRestante % MINUTE) / SECOND);
    navesConstruccionL.setValue(naveConstruida.getCantidad() + "");
    tiempoL.setValue(hours + ":" + minutes + ":" + seconds);
    imagenNave.setSource(naveConstruida.getNave().getImage().getSource());

    cancelarConstruccionB.addClickListener(new Button.ClickListener()
    {

        @Override
        public void buttonClick(ClickEvent event)
        {
            String value = CrudNave.cancelarConstruccion(((VaadinUI) UI.getCurrent()).getUsuari

            errorL.setValue(value);

            if(value.isEmpty())
            {
                layoutPadre.removeAllComponents();
```

Figura 91. Constructor de la clase de naves en construcción. (Elaboración propia)

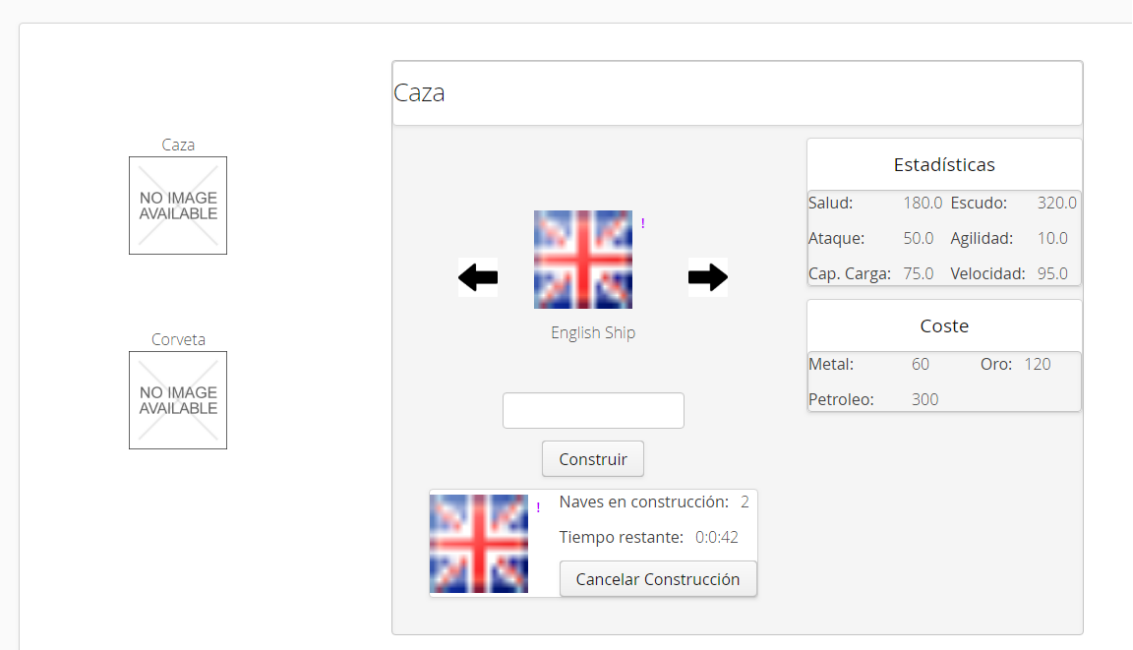

Figura 92. Ventana antigua de hangar con una nave en construcción. (Elaboración propia)

3.3.4.3.8. Realizar la ventana de las instalaciones

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

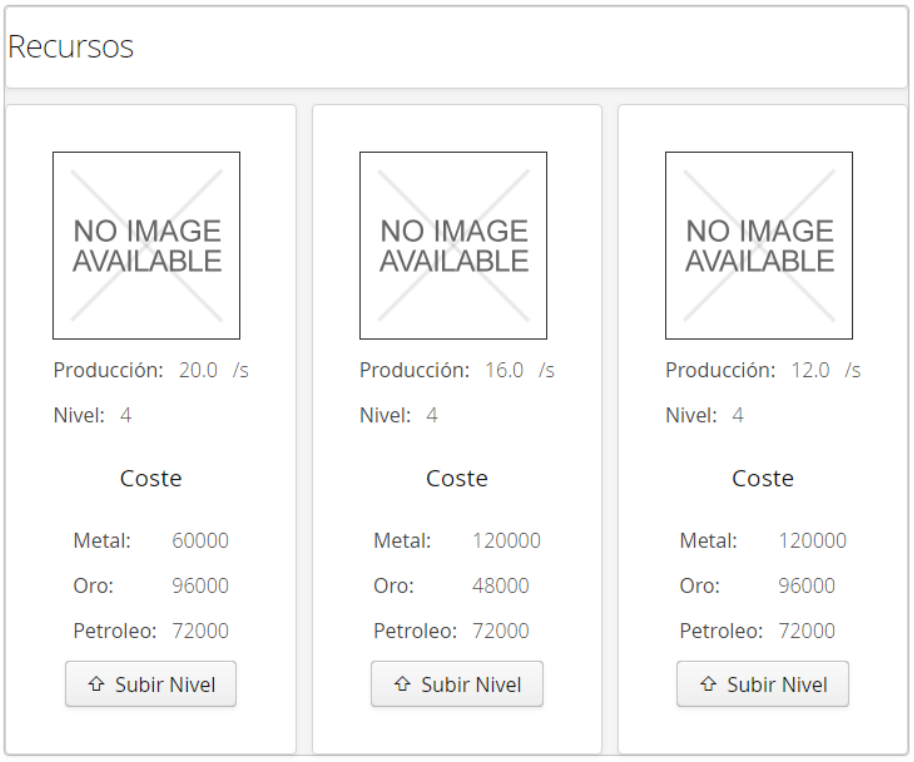

Figura 93. Ventana antigua de recursos. (Elaboración propia)

3.3.4.4. Feedback

Tras probarlo el usuario ha aconsejado que se coloquen placeholders en los campos de texto ya que no sabe qué información debería colocar en ellos.

3.3.5. Módulo Información

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.5.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.5.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo.

3.3.5.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

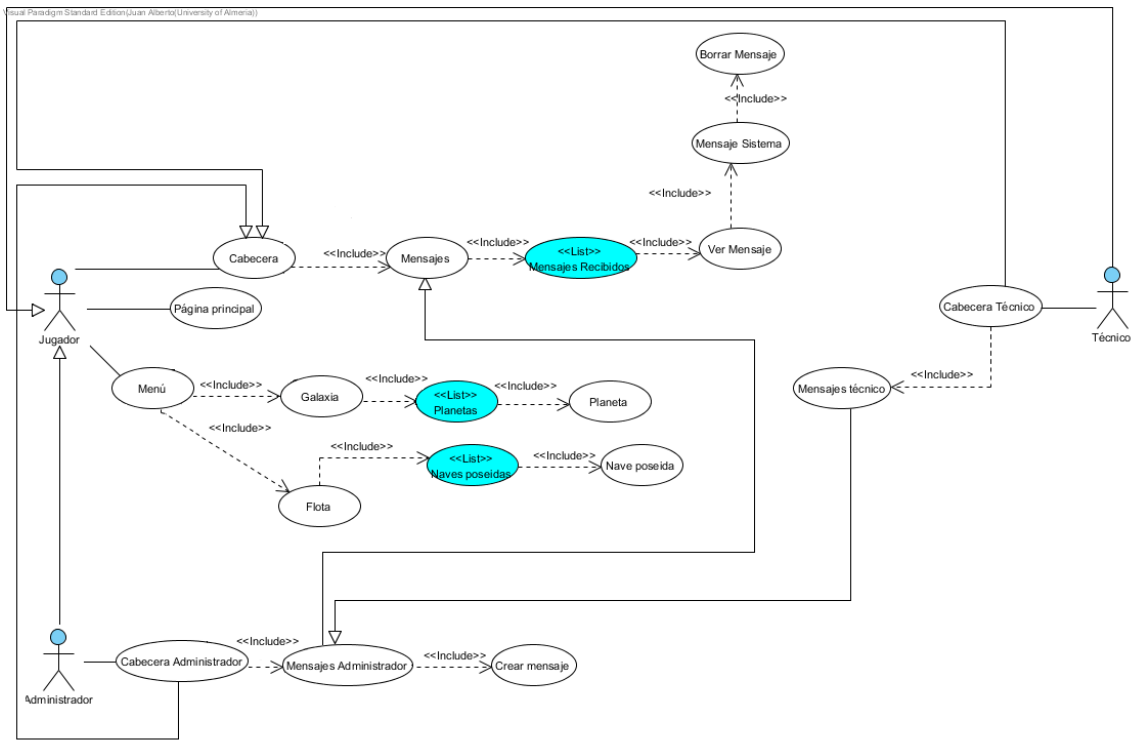

Figura 94. Diagrama de casos de uso del módulo información. (Elaboración propia)

3.3.5.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 43. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 44. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 45. Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 46. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.5.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 47. Caso de uso ver mensaje. (Elaboración propia)

CDU-0020	Ver mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar sus datos.
Precondición	Estar identificado en el sistema.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) solicita al sistema iniciar el procedimiento de consultar sus mensajes 2 / El sistema muestra una lista de mensajes con su tipo de mensaje, fecha de envío, asunto y planeta de destino si es un mensaje de batalla. 3 / Si quiere ver más información de un mensaje del sistema, se realiza el caso de uso Mensaje sistema (CDU-0021) 4 / Si es administrador o técnico y quiere crear un mensaje, se realiza el caso de uso Crear mensaje (CDU-0023)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 48. Caso de uso mensaje sistema. (Elaboración propia)

CDU-0021	Mensaje sistema
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar un mensaje del tipo sistema. o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Haber seleccionado un mensaje del sistema de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de ver el mensaje seleccionado 2 / El sistema muestra los datos del mensaje, el asunto y la descripción. 3 / Si el usuario quiere borrar el mensaje, se realiza el caso de uso Borrar mensaje (CDU-0022)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 49. Caso de uso borrar mensaje. (Elaboración propia)

CDU-0022	Borrar mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que desea borrar un mensaje. o durante la realización de los siguientes casos de uso: [CDU-0021] Mensaje sistema
Precondición	Haber seleccionado un mensaje de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) confirma que desea eliminar el mensaje 2 / El sistema marca el mensaje como descartado
Postcondición	El usuario es redireccionado a [CDU-0020] Ver mensajes y el mensaje no se le mostrará a ese usuario.
Excepciones	**Paso** / **Acción** - / -

Tabla 50. Caso de uso crear mensaje. (Elaboración propia)

CDU-0023	Crear mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador/técnico quiere enviar un mensaje a todos los usuarios. o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Estar identificado como administrador o técnico.
Secuencia normal	**Paso** / **Acción** 1 / El actor Administrador (CDU-ACT-0003) escribe el asunto y el cuerpo del mensaje 2 / El actor Administrador (CDU-ACT-0003) envía el mensaje 3 / El sistema confirma el envío del mensaje
Postcondición	Todos los usuarios reciben el mensaje
Excepciones	**Paso** / **Acción** - / -

Tabla 51. Caso de uso consultar naves poseídas. (Elaboración propia)

CDU-0024	Consultar naves poseídas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver las naves que posee
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestras las naves que tiene el usuario, indicando las naves en espera y las naves en movimiento.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 52. Caso de uso consultar lista de planetas. (Elaboración propia)

CDU-0025	Consultar lista de planetas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver los planetas existentes
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los planetas existentes dividiéndolos por sistemas y sectores y mostrando de cada uno una imagen, un nombre y sus coordenadas dentro del sistema.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

3.3.5.1.2. Relación Diagrama – Casos de Uso

A continuación se muestran las relaciones entre los diagramas y los casos de uso de los actores.

Tabla 53. Relación diagrama – caso de uso. Módulo información. (Elaboración propia)

Especificación/ Diagrama	CDU-0020	CDU-0020	CDU-0021	CDU-0022	CDU-0023	CDU-0024	CDU-0025
Cabecera	↑						
Mensajes	↑						
Mensajes Recibidos	↑						
Ver Mensaje	↑						
Mensaje Sistema			↑				
Borrar Mensaje				↑			
Menú						↑	↑
Flota						↑	
Naves poseídas						↑	
Nave poseída						↑	
Galaxia							↑
Planetas							↑
Planeta							↑
Cabecera Administrador		↑					
Mensajes Administrador		↑					
Crear mensaje					↑		
Cabecera Técnico		↑					
Mensajes técnico		↑					

3.3.5.2. Diseño

A continuación se muestra el diagrama de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

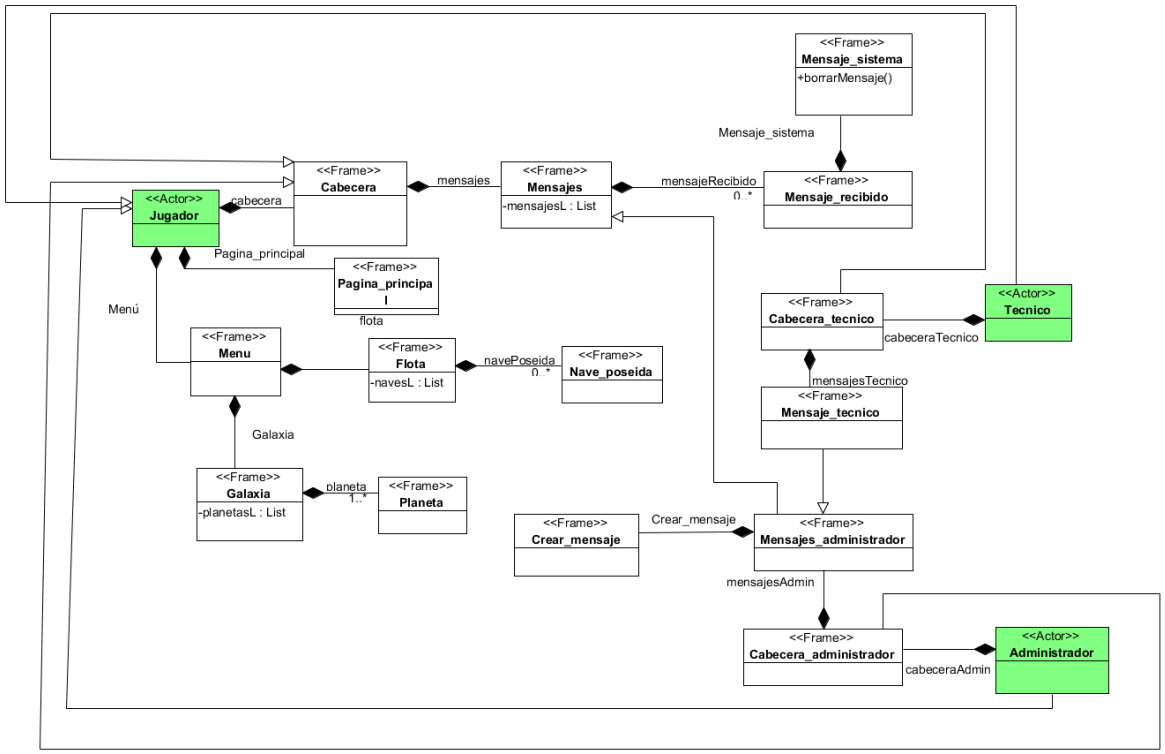

Figura 95. Diagrama de clases del módulo información. (Elaboración propia)

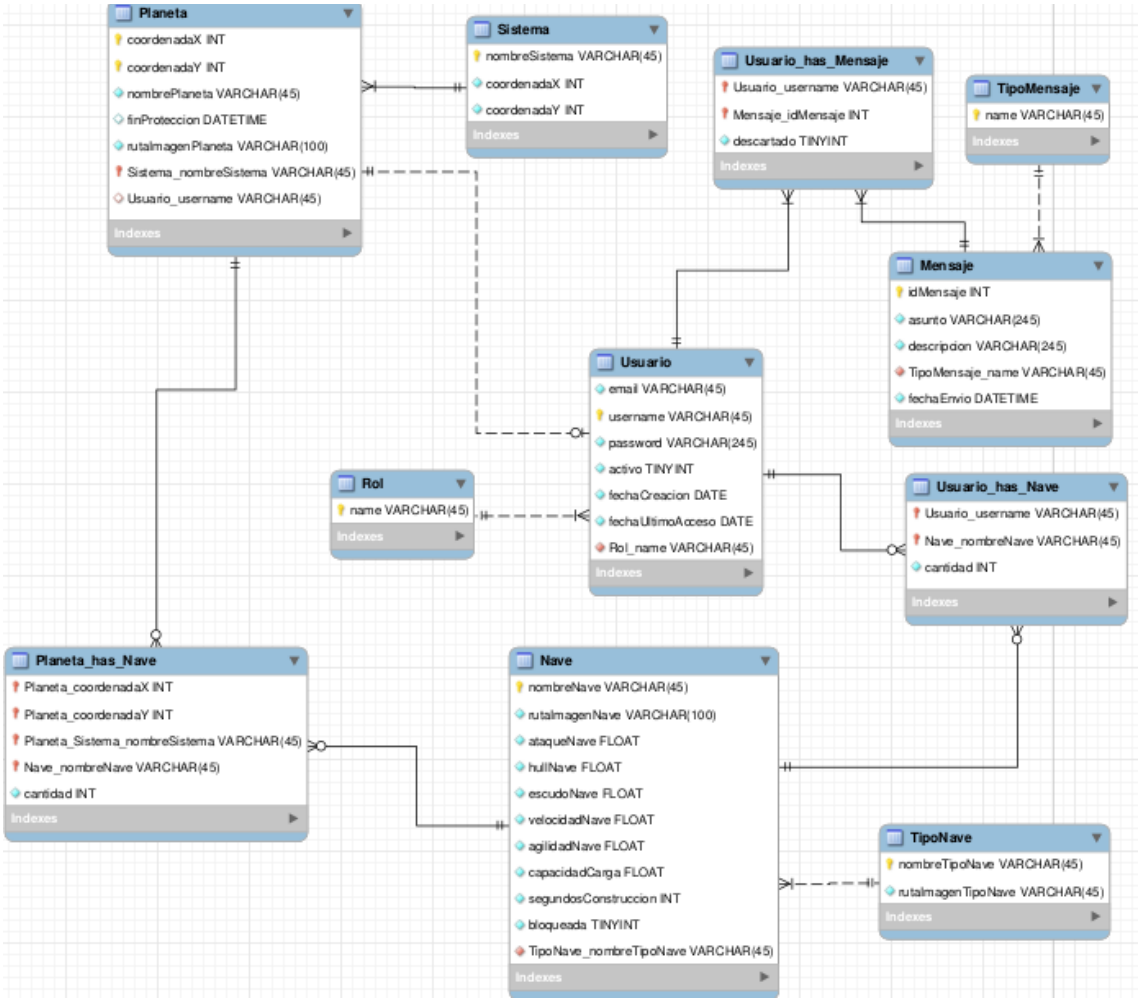

Figura 96. Diagrama de la base de datos del módulo información. (Elaboración propia)

3.3.5.3. Implementación del módulo

Solo es una semana de duración y una persona en el desarrollo, por lo que para el reparto de tareas no hay ningún problema.

3.3.5.3.1. Probar las clases de la base de datos

Se realizó de la misma manera que los módulos anteriores, además la mayoría de las pruebas ya se hicieron en módulos anteriores.

```
public void testPlanetaHasNavePK()
{
    PlanetaHasNavePK planetaNavePK = new PlanetaHasNavePK(0, 1, "Atlas", "Vulture");
    PlanetaHasNave planetaNave = new PlanetaHasNave(planetaNavePK, 32);

    assertEquals(0, planetaNave.getPlanetaHasNavePK().getPlanetacoordenadaX());
    assertEquals(1, planetaNave.getPlanetaHasNavePK().getPlanetacoordenadaY());
    assertEquals("Atlas", planetaNave.getPlanetaHasNavePK().getPlanetaSistemanombreSistem
    assertEquals("Vulture", planetaNave.getPlanetaHasNavePK().getNavenombreNave());
    assertEquals(32, planetaNave.getCantidad(), 0.0);
}

@Test
public void testTipoMensaje()
{
    TipoMensaje tipoM = new TipoMensaje("Sistema");

    assertEquals("Sistema", tipoM.getName());
}

@Test
public void testMensaje()
{
    Date fechaAhora = new Date();
    TipoMensaje tipoM = new TipoMensaje("Sistema");
    Mensaje mensaje = new Mensaje("Mantenimiento", "Mantenimiento el dia 15", fechaAhora)

    mensaje.setTipoMensajename(tipoM);

    assertEquals("Mantenimiento", mensaje.getAsunto());
    assertEquals("Mantenimiento el dia 15", mensaje.getDescripcion());
```

Figura 97. Pruebas de las clases de la base de datos de los planetas, los tipos de mensaje y los mensajes. (Elaboración propia)

3.3.5.3.2. Probar la creación de mensajes

Pruebas sencillas en las que simplemente se crea un mensaje y se asigna a todos los usuarios.

No se hicieron muchas pruebas ya que la mayoría de la funcionalidad dependía de elementos de la interfaz, lo que no permitía realizar pruebas unitarias.

```
public static String crearMensaje(String asunto, String descripcion, UsuarioRepository usuarioRe
{
    Mensaje mensaje, mensajeEnviado;

    if(!MensajeDataValidator.comprobarMensajeVacio(asunto)) return "Usuario vacío";
    if(!MensajeDataValidator.comprobarMensajeVacio(descripcion)) return "Descripción vacía";

    mensaje = new Mensaje(asunto, descripcion, new Date());
    mensaje.setTipoMensajename(tipoMensajeRepo.findByName("Sistema"));
    mensajeEnviado = mensajeRepo.save(mensaje);

    return broadcastMensaje(mensajeEnviado, usuarioRepo, usuarioMensajeRepo);
}

private static String broadcastMensaje(Mensaje mensaje, UsuarioRepository usuarioRepo, Usuarioha
{
    List<UsuariohasMensaje> usuariosMensajeL = new ArrayList<>();
    List<Usuario> usuariosL = usuarioRepo.findAll();

    for(int i = 0; i < usuariosL.size(); i++)
    {
        if(!usuariosMensajeL.add(new UsuariohasMensaje(usuariosL.get(i).getUsername(), mensaje.g
    }

    usuarioMensajeRepo.saveAll(usuariosMensajeL);

    return "";
}
```

Figura 98. Métodos de creación y entrega de mensajes. (Elaboración propia)

3.3.5.3.3. Realizar la ventana de flota

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Era similar al resto de listas, es decir, una tabla que muestra las naves del usuario.

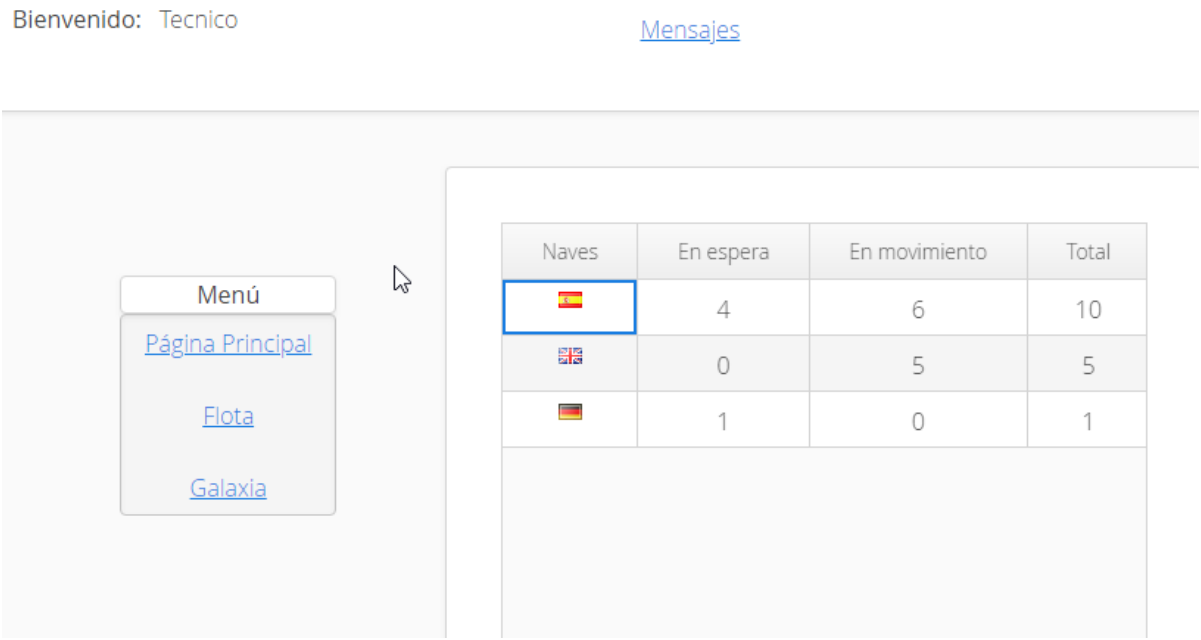

Figura 99. Ventana antigua de flota. (Elaboración propia)

3.3.5.3.4. Realizar la ventana de galaxia

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Esta ventana dio muchos problemas, las Tabs de Vaadin están hechas de tal manera que cada Tab posee un elemento, pero en este caso se buscaba que según el Tab que se eligiera, la tabla cargara unos valores.

Además, debían de aparecer tantos botones como número de “páginas” para navegar entre los planetas de un sistema.

Para solucionar el tema de cambios de Tab se almacenó el layout que contiene la tabla que nos interesa y en el listener de cambio de Tab se cambia el layout actual por un nuevo layout, se añade el layout que almacenamos, se selecciona y se borra la anterior Tab.

```
sistemasTab.addSelectedTabChangeListener(new SelectedTabChangeListener()
{

    @Override
    public void selectedTabChange(SelectedTabChangeEvent event)
    {
        if(event.getTabSheet().getSelectedTab() != currentComponent)
        {
            int idSeleccionada = Integer.parseInt(event.getTabSheet().getSelectedTab().getId());
            String nombreSeleccionada = event.getTabSheet().getTab(idSeleccionada).getCaption();
            Tab nuevaTab;
            VerticalLayout aux = new VerticalLayout();

            aux.setId(currentComponent.getId());
            event.getTabSheet().replaceComponent(currentComponent, aux);
            currentComponent.setId(idSeleccionada + "");

            nuevaTab = event.getTabSheet().addTab(currentComponent, nombreSeleccionada);
            event.getTabSheet().setSelectedTab(nuevaTab);

            event.getTabSheet().removeTab(event.getTabSheet().getTab(idSeleccionada));
            event.getTabSheet().setTabPosition(nuevaTab, idSeleccionada);
            currentComponent = event.getTabSheet().getSelectedTab();

            initTab(event.getTabSheet().getTab(currentComponent).getCaption());
        }
    }
});
```

Figura 100. Listener de la gestión de las pestañas de la ventana galaxia. (Elaboración propia)

Por último, los botones se solucionaron realizando un bucle que recorre la lista de planetas de 10 en 10 poniendo un botón por cada iteración y asignándole como id un número empezando por el 1 y que va incrementando en cada iteración.

Al pulsar en cada botón se obtiene el rango de datos del botón pulsado mediante la siguiente fórmula:

$$inicio = 10 * (idBoton - 1)$$

$$fin = 10 + inicio$$

Si por ejemplo, se pulsase en el botón 5, el inicio sería 40 y el final sería 50. Hay que tener en cuenta que el método subList no coge el último elemento, por lo que obtendríamos 10 elementos.

```
private void initTab(String nombreSistema)
{
    int contadorPlanetas = 1;
    Button coordenadaYB;

    planetasL = planetaRepo.findBySistemanombreSistema(nombreSistema);

    for(int i = 0; i < planetasL.size(); i += 10)
    {
        coordenadaYB = new Button(contadorPlanetas + "");
        coordenadaYB.setId(contadorPlanetas + "");
        hLayoutBotones.addComponent(coordenadaYB);

        if(contadorPlanetas == 1) seleccionadoB = coordenadaYB;
        contadorPlanetas++;

        coordenadaYB.addClickListener(new Button.ClickListener()
        {

            @Override
            public void buttonClick(ClickEvent event)
            {
                int subListInicio = 10*(Integer.parseInt(event.getButton().getI
                int subListFinal = 10 +  subListInicio;

                seleccionadoB.removeStyleName("friendly");
                seleccionadoB = event.getButton();
                seleccionadoB.addStyleName("friendly");

                if(subListFinal > planetasL.size()-1) subListFinal = planetasL.
```

Figura 101. Método de la inicialización de las pestañas de la ventana galaxia. (Elaboración propia)

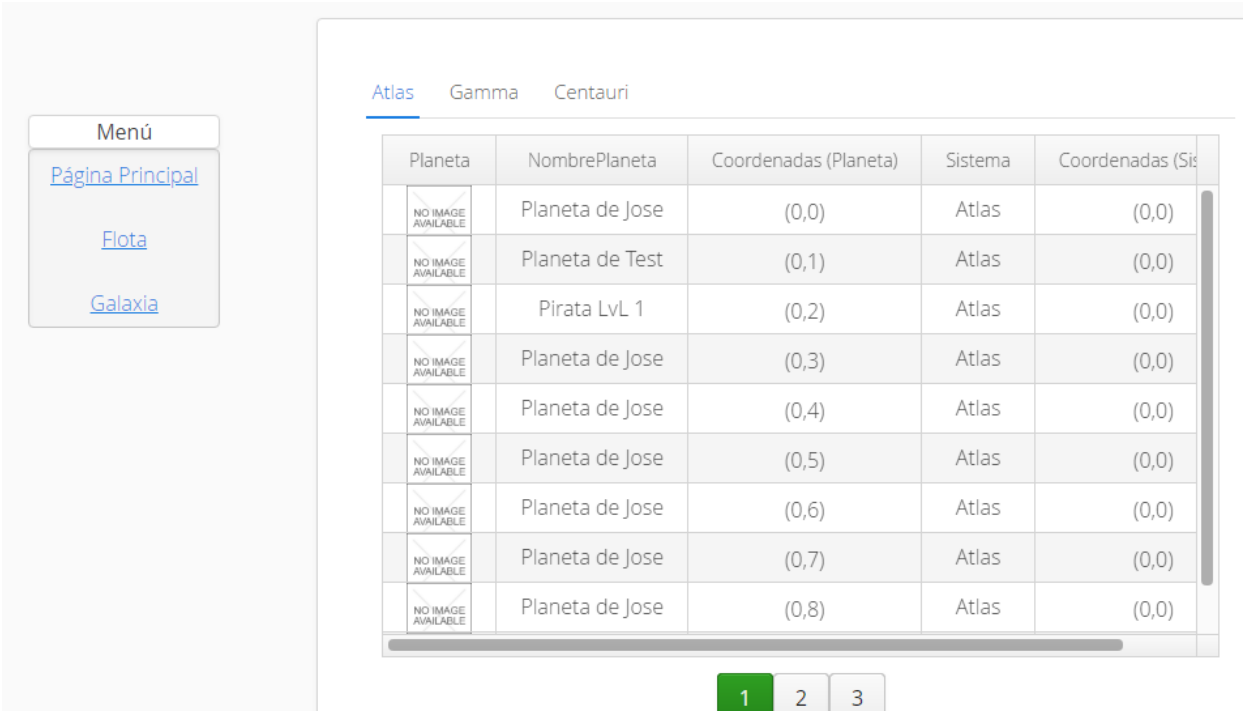

Figura 102. Ventana antigua de galaxia. (Elaboración propia)

3.3.5.3.5. Realizar la ventana ver mensajes

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

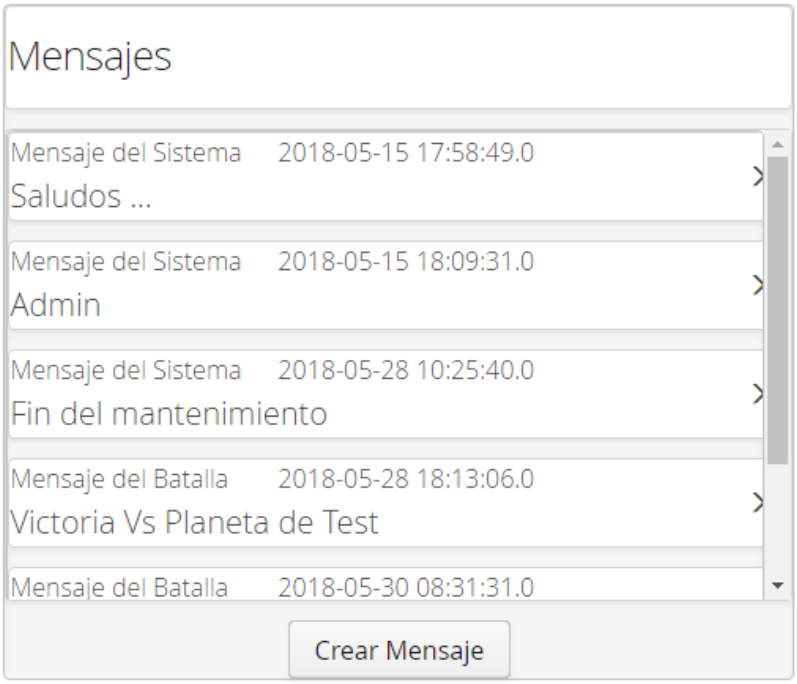

Figura 103. Ventana antigua de mensajes. (Elaboración propia)

3.3.5.3.6. Realizar la ventana de ver un mensaje

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Este apartado dio problemas al controlar el acceso de los usuarios para evitar que vean mensajes que no deberían de ver.

```
@Override
public void enter(ViewChangeEvent event)
{
    if (!event.getParameters().isEmpty() && MensajeDataValidator.comprobarValorInteger(e
    {
        usuarioMensajeRepo = ((VaadinUI) UI.getCurrent()).getInterfazUsuarioMensaje();
        mensaje = usuarioMensajeRepo.findByMensajeidMensajeUsuariousernameNoDescartado(I
        if(mensaje == null) return;

        tipoL.setValue("Mensaje del " + mensaje.getMensaje().getTipoMensajename().getNam
        asuntoF.setValue(mensaje.getMensaje().getAsunto());
        descripcionF.setValue(mensaje.getMensaje().getDescripcion());
    }
}
```

Figura 104. Método de inicialización de la ventana de ver un mensaje. (Elaboración propia)

Figura 105. Ventana antigua de ver un mensaje del tipo sistema. (Elaboración propia)

3.3.5.3.7. Realizar la ventana de crear mensajes

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

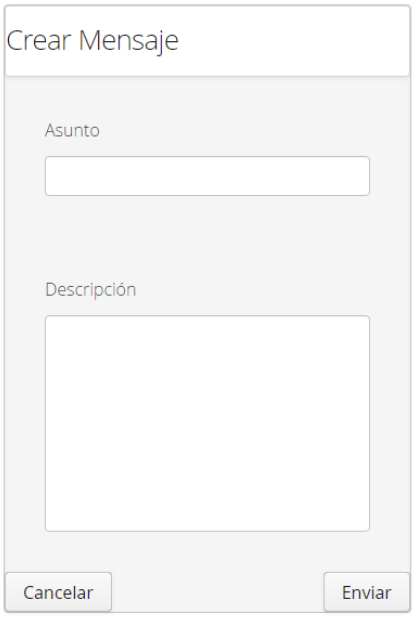

Figura 106. Ventana antigua de crear un mensaje. (Elaboración propia)

3.3.5.4.Feedback

No se ha obtenido ningún consejo de mejora tras probarlo el usuario.

3.3.6. Módulo Ataque

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.6.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.6.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo.

3.3.6.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

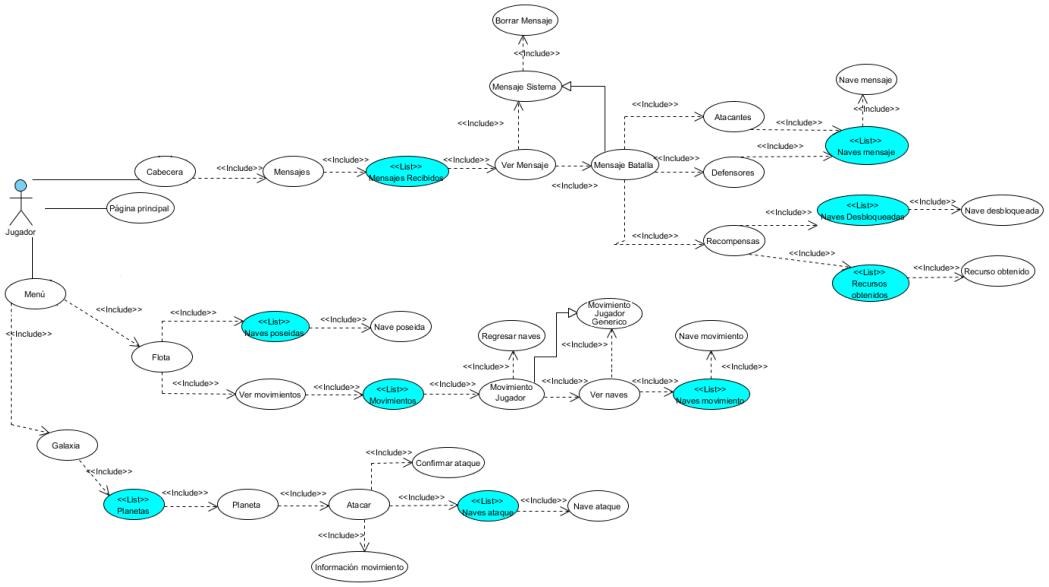

Figura 107. Diagrama de casos de uso del módulo ataque. (Elaboración propia)

3.3.6.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 54. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 55. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 56. Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 57. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.6.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 58. Caso de uso ver mensaje. (Elaboración propia)

CDU-0020	Ver mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar sus datos.
Precondición	Estar identificado en el sistema.
Secuencia normal	**Paso** / **Acción** 1 — El actor Jugador (CDU-ACT-0002) solicita al sistema iniciar el procedimiento de consultar sus mensajes 2 — El sistema muestra una lista de mensajes con su tipo de mensaje, fecha de envío, asunto y planeta de destino si es un mensaje de batalla. 3 — Si quiere ver más información de un mensaje de una batalla, se realiza el caso de uso Mensaje batalla (CDU-0005) 4 — Si quiere ver más información de un mensaje del sistema, se realiza el caso de uso Mensaje sistema (CDU-0021) 5 — Si es administrador o técnico y quiere crear un mensaje, se realiza el caso de uso Crear mensaje (CDU-0023)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 59. Caso de uso ver mensaje. (Elaboración propia)

CDU-0021	Mensaje sistema
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar un mensaje del tipo sistema. o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Haber seleccionado un mensaje del sistema de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de ver el mensaje seleccionado 2 / El sistema muestra los datos del mensaje, el asunto y la descripción. 3 / Si el usuario quiere borrar el mensaje, se realiza el caso de uso Borrar mensaje (CDU-0022)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 60. Caso de uso borrar mensaje. (Elaboración propia)

CDU-0022	Borrar mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que desea borrar un mensaje. o durante la realización de los siguientes casos de uso: [CDU-0026] Mensaje batalla, [CDU-0021] Mensaje sistema
Precondición	Haber seleccionado un mensaje de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) confirma que desea eliminar el mensaje 2 / El sistema marca el mensaje como descartado
Postcondición	El usuario es redireccionado a [CDU-0020] Ver mensajes y el mensaje no se le mostrará a ese usuario.
Excepciones	**Paso** / **Acción** - / -

Tabla 61. Caso de uso consultar naves poseídas. (Elaboración propia)

CDU-0024	Consultar naves poseídas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver las naves que posee
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestras las naves que tiene el usuario, indicando las naves en espera y las naves en movimiento.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 62. Caso de uso consultar lista de planetas. (Elaboración propia)

CDU-0025	Consultar lista de planetas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver los planetas existentes
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los planetas existentes dividiéndolos por sistemas y sectores y mostrando de cada uno una imagen, un nombre y sus coordenadas dentro del sistema.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 63. Caso de uso mensaje batalla. (Elaboración propia)

CDU-0026	Mensaje batalla
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar un mensaje del tipo batalla o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Haber seleccionado un mensaje de batalla de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de ver el mensaje seleccionado 2 / El sistema muestra los datos del mensaje, las naves atacantes que se enviaron y fueron destruidas, las naves defensoras que se enviaron y fueron destruidas y las recompensas obtenidas (recursos y naves obtenidas). 3 / Si el usuario quiere borrar el mensaje, se realiza el caso de uso Borrar mensaje (CDU-0022)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 64. Caso de uso ver movimientos. (Elaboración propia)

CDU-0027	Ver movimientos
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver los movimientos hacia o desde su planeta o durante la realización de los siguientes casos de uso: [CDU-0024] Consultar naves poseídas
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los movimientos hacia o desde el planeta del usuario mostrando el planeta de origen y destino y el tiempo de llegada. 2 / Si el movimiento es hacia otro planeta y el usuario quiere cancelarlo, se realiza el caso de uso Regresar naves (CDU-0029) 3 / Si quiere ver más información acerca de un movimiento, se realiza el caso de uso Consultar información de movimiento (CDU-0028)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 65. Caso de uso consultar información de movimiento. (Elaboración propia)

CDU-0028	Consultar información de movimiento
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver más información de un movimiento o durante la realización de los siguientes casos de uso: [CDU-0027] Ver movimientos
Precondición	El sistema muestra la información del movimiento, el planeta de origen y destino, el tiempo estimado de llegada y las naves que se dirigen al planeta.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra la información del movimiento, el planeta de origen y destino, el tiempo estimado de llegada y las naves que se dirigen al planeta.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 66. Caso de uso regresar naves. (Elaboración propia)

CDU-0029	Regresar naves
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera cancelar un movimiento realizado por sus naves o durante la realización de los siguientes casos de uso: [CDU-0027] Ver movimientos
Precondición	El sistema muestra la información del movimiento, el planeta de origen y destino, el tiempo estimado de llegada y las naves que se dirigen al planeta.
Secuencia normal	**Paso** / **Acción** 1 / El sistema cancela el movimiento actual 2 / El sistema crea un nuevo movimiento con las naves regresando al planeta del usuario
Postcondición	Hay un nuevo movimiento creado
Excepciones	**Paso** / **Acción** - / -

Tabla 67. Caso de uso atacar planeta. (Elaboración propia)

<table>
<tr><td>CDU-0030</td><td>Atacar planeta</td></tr>
<tr><td>[Versión]</td><td>1.0 (06/03/2018)</td></tr>
<tr><td>[Dependencias]</td><td></td></tr>
<tr><td>Descripción</td><td>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera atacar un planeta o durante la realización de los siguientes casos de uso: [CDU-0025] Consultar lista de planetas</td></tr>
<tr><td>Precondición</td><td>Estar identificado</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El sistema muestra todas las naves existentes</td></tr>
<tr><td>2</td><td>El actor Jugador (CDU-ACT-0002) indica la cantidad de cada nave que desea enviar en el ataque</td></tr>
<tr><td>3</td><td>El sistema actualiza el tiempo de llegada en función de las naves y cantidad de naves escogidas.</td></tr>
<tr><td>4</td><td>El actor Jugador (CDU-ACT-0002) realiza el ataque</td></tr>
<tr><td>5</td><td>El sistema registra el ataque en la base de datos y se lo confirma al usuario</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Se ha creado un movimiento para ese ataque</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>5</td><td>Si el planeta de destino tiene protección activa, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto</td></tr>
</table>
</td></tr>
</table>

3.3.6.1.2. Relación Diagrama – Casos de Uso

A continuación se muestran las relaciones entre los diagramas y los casos de uso de los actores.

Tabla 68. Relación diagrama – caso de uso, jugador. Módulo ataque. (Elaboración propia)

Especificación/ Diagrama	CDU-0020	CDU-0021	CDU-0022	CDU-0024	CDU-0025	CDU-0026	CDU-0027	CDU-0028	CDU-0029	CDU-0030
Cabecera	↑									
Mensajes	↑									
Mensajes Recibidos	↑									
Ver Mensaje	↑									
Mensaje Sistema		↑								
Borrar Mensaje			↑							
Mensaje Batalla						↑				
Atacantes						↑				
Defensores						↑				
Naves mensaje						↑				
Nave mensaje						↑				
Naves Desbloqueadas						↑				
Nave desbloqueada						↑				
Recursos obtenidos						↑				
Recurso obtenido						↑				
Menú				↑	↑					
Flota				↑						
Naves poseídas				↑						
Nave poseída				↑						
Ver movimientos							↑			
Movimientos							↑			
Movimiento							↑			
Regresar naves									↑	
Ver naves								↑		
Información movimiento								↑		
Naves movimiento								↑		
Nave movimiento								↑		
Galaxia					↑					
Planetas					↑					
Planeta					↑					
Atacar										↑
Confirmar ataque										↑
Naves ataque										↑
Nave ataque										↑

3.3.6.2. Diseño

A continuación se muestra el diagrama de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

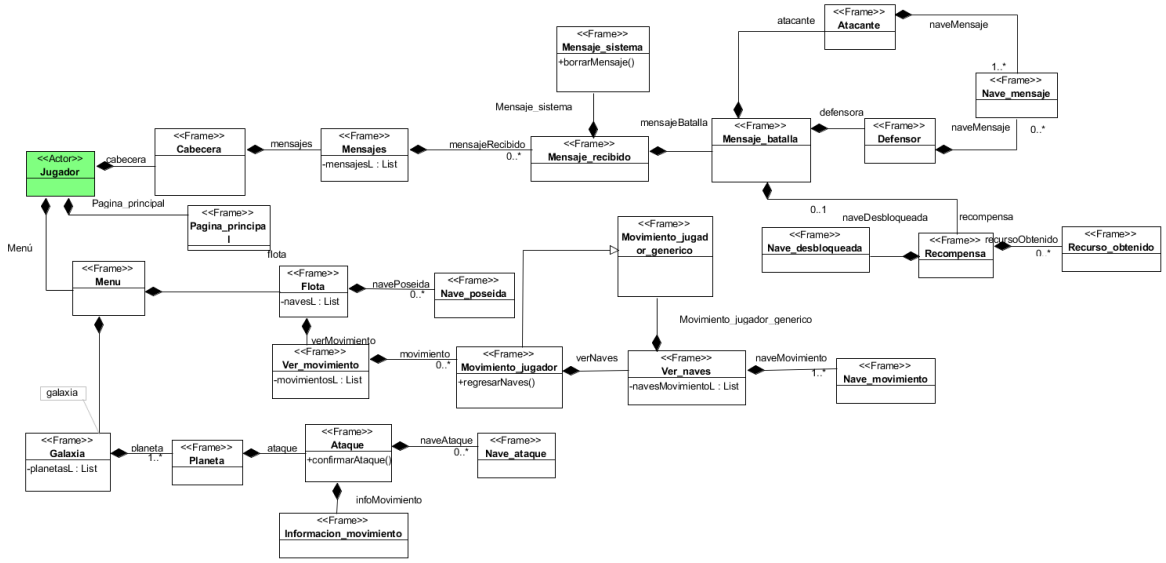

Figura 108. Diagrama de clases del módulo ataque. (Elaboración propia)

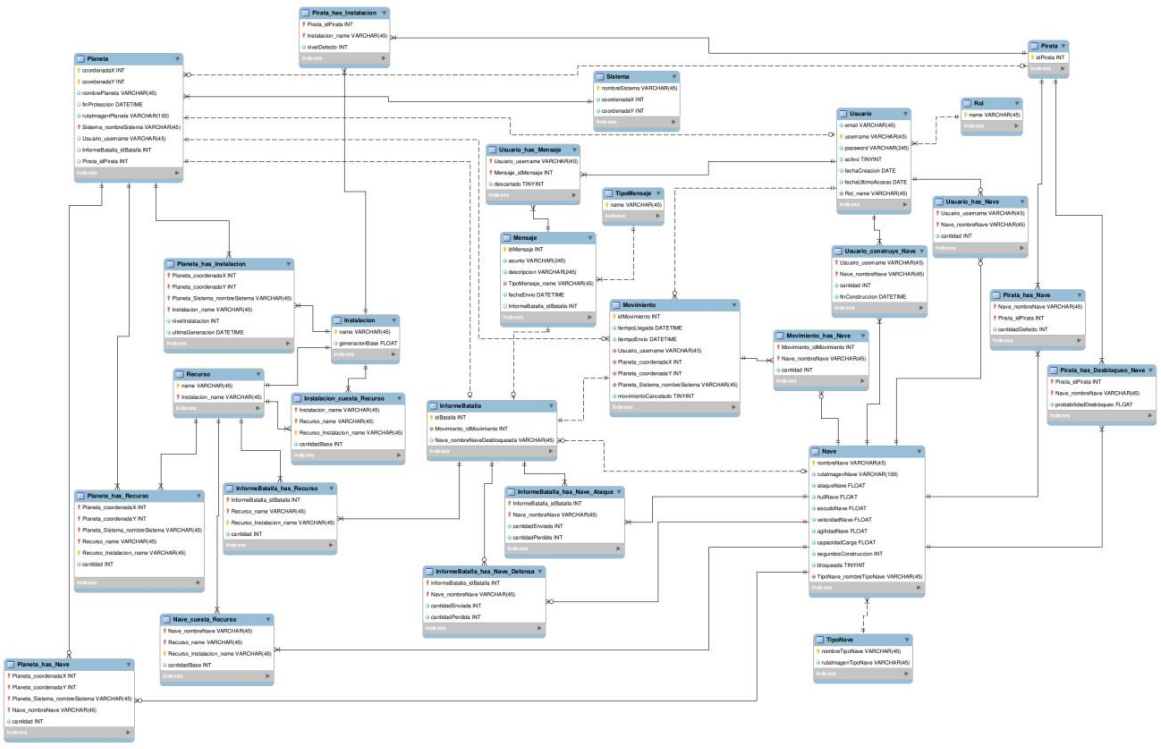

Figura 109. Diagrama de la base de datos del módulo ataque. (Elaboración propia)

3.3.6.3.Implementación del módulo

Al ser dos semanas de duración las tareas se dividieron en dos iteraciones, en la primera no se obtuvo *feedback* ya que se centró en las pruebas y su funcionalidad, por lo que no había ventanas hechas.

No hay tantas pruebas debido a que la mayoría dependen de elementos de la interfaz, impidiendo realizar pruebas unitarias.

3.3.6.3.1. Primera iteración

Esta iteración se centró en las pruebas y su funcionalidad.

3.3.6.3.1.1. Probar las clases de la base de datos

Este módulo incluía todas las tablas de la base de datos por lo que las pruebas eran muy extensas, por suerte casi todas las tablas habían sido probadas en módulos anteriores por lo que se pudo reutilizar de ellos y disminuir en gran medida la carga de trabajo.

```
@Test
public void testInformeBatallaFull()
{
    Date fechaAhora, fechaLuego;
    Calendar aux = Calendar.getInstance();

    aux.add(Calendar.HOUR_OF_DAY, 2);
    fechaAhora = new Date();
    fechaLuego = aux.getTime();

    InformeBatalla informe = new InformeBatalla();

    Movimiento movimiento = new Movimiento(fechaLuego, fechaAhora, (sho
    Nave nave = new Nave("Vulture", "/WEB-INF/images/image.png", 120, 2

    informe.setMovimientoidMovimiento(movimiento);
    informe.setNavenombreNaveDesbloqueada(nave);

    assertEquals(movimiento, informe.getMovimientoidMovimiento());
    assertEquals(nave, informe.getNavenombreNaveDesbloqueada());
}

@Test
public void testInformeListaPlaneta()
{
    InformeBatalla informe = new InformeBatalla();
```

Figura 110. Pruebas de los informes de batalla. (Elaboración propia)

3.3.6.3.1.2. Probar el sistema de protección de los planetas

La prueba de esta funcionalidad era compleja debido a la gran cantidad de datos que debían de inicializarse.

```
@Test
public void comprobarPlanetaPropio()
{
    assertFalse(Galaxia.comprobarPlanetaDisponible(usuario, planet
}

@Transactional
@Test
public void comprobarNoAtacado()
{
    assertTrue(Galaxia.comprobarPlanetaDisponible(usuario, planeta
}

@Transactional
@Test
public void comprobarUltimoAtaqueMenos8Horas()
{
    assertTrue(Galaxia.comprobarPlanetaDisponible(usuario, planeta
}

@Transactional
@Test
public void comprobarSinProteccionHoras()
{
    assertTrue(Galaxia.comprobarPlanetaDisponible(usuario, planeta
}

@Transactional
@Test
```

Figura 111. Pruebas de comprobación acerca de la protección de un planeta. (Elaboración propia)

La comprobación de un planeta dependía de muchas variables, primero que ese planeta hubiera sido atacado, es decir, que tuviera en la base de datos un informe de batalla asociado al planeta, si no entonces no había sido atacado y no tenía protección.

Luego debía de comprobarse que el ataque hubiera sido hace menos de 8 horas, si no significaba que no tenía protección

Una vez eso estaba comprobado para cada uno de los 5 últimos movimientos se comprobaba primero que no hubieran pasado 8 horas o más, si las habían pasado entonces no tenía protección.

Por último se comprobaba que esos movimientos fueran de derrota, si lo eran entonces el planeta tenía protección.

```
public static String comprobarPlanetaDisponible(Usuario usuario, Planeta planeta, Mov
{
    List<InformeBatallahasNaveDefensa> aux;
    InformeBatalla informeBatalla;
    int seconds;
    Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.HOUR_OF_DAY, -8);
    Date fechaAhora = calendar.getTime();
    int contadorDerrotas = 0;
    InformeBatalla ultimoInforme = planeta.getInformeBatallaidBatalla();
    List<Movimiento> movimientos = movimientoRepo.findByPlanetaDestino(planeta);

    if(usuario.equals(planeta.getUsuariousername())) return "No puedes atacarte a ti
    if(ultimoInforme == null) return "";

    seconds = (int) (fechaAhora.getTime() - ultimoInforme.getMovimientoidMovimiento()
    if(seconds >= 0) return "";

    for(int i = movimientos.size()-1; i > 0; i--)
    {
        calendar = Calendar.getInstance();
        calendar.setTime(movimientos.get(i).getTiempoLlegada());
        calendar.add(Calendar.HOUR_OF_DAY, -8);

        seconds = (int) (calendar.getTime().getTime() - movimientos.get(i-1).getTiemp
        if(seconds >= 0) return "";

        informeBatalla = informeRepo.findByIdMovimiento(movimientos.get(i));

        if(informeBatalla == null) continue;
```

Figura 112. Método de comprobar si un planeta está disponible para ser atacado. (Elaboración propia)

3.3.6.3.1.3. Realizar los triggers necesarios

Esta es la parte más compleja y el núcleo de la aplicación, los triggers de la base de datos para los viajes entre planetas y las batallas.

Hay tres triggers en la aplicación: el trigger de ataque, el trigger de recuperación de las naves tras la batalla y el trigger de recuperación de las naves en caso de que se cancele el ataque.

Estos tres triggers se ejecutan mediante cron jobs en el tiempo que se le indica al crearlos.

El trigger de recuperación de las naves tras la batalla coge las naves del último movimiento a tu planeta (generado un instante antes por el trigger de ataque) y las añade a tu planeta. Este es el trigger que más problemas puede dar por eso mismo, debería de buscarse una mejor manera de implementarlo ya que dará problemas en el futuro.

```
"CREATE EVENT movimiento_" +movimiento.getIdMovimiento() + "_battleShips ON SCHEDULE AT '2037-12-12 23:59
    + "DECLARE nom Varchar(45);\n"
    + "DECLARE cantidad_aux INT;\n"
    + "DECLARE done INT DEFAULT FALSE;\n"
    + "DECLARE atacanteNom CURSOR FOR SELECT Nave_nombreNave "
        + "FROM Movimiento_has_Nave, (SELECT MAX(idMovimiento) AS maximo "
                                    + "FROM Movimiento "
                                    + "WHERE Planeta_coordenadaX = " +planetaOrigen.getCoordenadaX()
                                    + " AND Planeta_coordenadaY = " +planetaOrigen.getCoordenadaY()
                                    + " AND Planeta_Sistema_nombreSistema = '" + planetaOrigen.getSi
        + "WHERE Movimiento_idMovimiento = max_table.maximo;\n"
    + "DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;\n"
    + "OPEN atacanteNom;\n"
    + "atacantes_loopResultado: LOOP\n"
        + "Fetch atacanteNom INTO nom;\n"
        + "IF done THEN\n"
            + "LEAVE atacantes_loopResultado;\n"
        + "END IF;\n"
        +"SELECT cantidad INTO cantidad_aux "
            + "FROM Movimiento_has_Nave,  (SELECT MAX(idMovimiento) AS maximo "
                                    + "FROM Movimiento "
                                    + "WHERE Planeta_coordenadaX = " +planetaOrigen.getCoordenadaX()
                                    + " AND Planeta_coordenadaY = " +planetaOrigen.getCoordenadaY()
                                    + " AND Planeta_Sistema_nombreSistema ='" +planetaOrigen.getSiste
            + "WHERE Movimiento_idMovimiento = max_table.maximo"
```

Figura 113. Trigger de recuperación de naves tras la batalla. (Elaboración propia)

El trigger de recuperación de naves tras la cancelación simplemente añade las naves del movimiento al planeta, dichas naves se almacenaron antes de realizar el movimiento para facilitar su obtención.

```
String recuperarNaves = "CREATE EVENT IF NOT EXISTS movimiento_" +movimiento.getIdMovimiento()
+"_ships" +"\nON SCHEDULE AT '2037-12-12 23:59:59'\nDO BEGIN\n" +setNaves +"END;";
```

Figura 114. Trigger de recuperación de naves tras la cancelación del movimiento. (Elaboración propia)

Por último el trigger de ataque, este trigger se divide en tres fases, la fase pre-batalla, la fase de batalla y la fase post-batalla.

En la fase pre-batalla primero se declaran todas las variables y si el planeta destino es un pirata se comprueba si ya pasaron más de 5 horas, en caso positivo se regeneran todas sus naves.

```
+ "IF " + idPirata  +" > 0 AND " +idBatalla + " > 0 THEN\n"
    +"SELECT tiempoLlegada INTO tiempoUltimoAtaque FROM Movimiento\n"
        +"WHERE idMovimiento = (SELECT Movimiento_idMovimiento \n"
        +"FROM InformeBatalla\n"
        +"WHERE idBatalla = (SELECT InformeBatalla_idBatalla\n"
                    +"FROM Planeta \n"
                    + " WHERE coordenadaX = " +planetaDestino.getCoordenadaX()
                    + " AND coordenadaY = " +planetaDestino.getCoordenadaY()
                    + " AND Sistema_nombreSistema= '" +planetaDestino.getSistemanombreSistema() +
        + "SELECT TIMESTAMPDIFF(HOUR, tiempoUltimoAtaque,NOW()) INTO horasUltimoAtaque;\n"
        + "IF horasUltimoAtaque >= 5 THEN\n"
            + "UPDATE Planeta_has_Nave planeta\n"
                + "LEFT JOIN Pirata_has_Nave pirata ON planeta.Nave_nombreNave = pirata.Nave_nombreNa
                + "SET planeta.cantidad = pirata.cantidadDefecto\n"
                + "WHERE pirata.Pirata_idPirata =" + idPirata + "\n"
                + " AND Planeta_coordenadaX = " +planetaDestino.getCoordenadaX()
                + " AND Planeta_coordenadaY = " +planetaDestino.getCoordenadaY()
                + " AND Planeta_Sistema_nombreSistema= '" +planetaDestino.getSistemanombreSistema() +
        + "END IF;\n"
+ "END IF;\n"
```

Figura 115. Trigger de la fase de pre-batalla (parte 1). (Elaboración propia)

Después se generan los recursos del planeta en función de la última vez que se generaron y del nivel de sus instalaciones.

```
+ "SET @ultimaGeneracion= '2018-05-23 18:45:20';"
+ "SET @nivelInstalacion= 0;"
+ "SET @generacionBase= 2.4;"
+ "SET @segundosGenRecursos= 0;"
+ "SELECT nivelInstalacion, ultimaGeneracion INTO @nivelInstalacion, @ultimaGeneracion "
    + "FROM Planeta_has_Instalacion "
    + "WHERE Planeta_coordenadaX =" +planetaDestino.getCoordenadaX()
    + " AND Planeta_coordenadaY =" +planetaDestino.getCoordenadaY()
    + " AND Planeta_Sistema_nombreSistema ='" +planetaDestino.getSistemanombreSistema()
    + "' AND Instalacion_name='Mina de Metal';\n"
+ "IF " + idPirata  +" > 0 THEN\n"
    + "SELECT nivelDefecto INTO @nivelInstalacion "
        + "FROM Pirata_has_Instalacion "
        + "WHERE Pirata_idPirata =" +idPirata
        + " AND Instalacion_name='Mina de Metal';\n"
+ "END IF;\n"
+ "SELECT generacionBase INTO @generacionBase "
    + "FROM Instalacion "
    + "WHERE name = 'Mina de Metal';\n"
+ "SELECT TIMESTAMPDIFF(SECOND, @ultimaGeneracion,NOW()) INTO @segundosGenRecursos;\n"
+ "UPDATE Planeta_has_Instalacion "
    + "SET ultimaGeneracion = NOW() "
    + "WHERE Planeta_coordenadaX =" +planetaDestino.getCoordenadaX()
    + " AND Planeta_coordenadaY =" +planetaDestino.getCoordenadaY()
    + " AND Planeta_Sistema_nombreSistema ='" +planetaDestino.getSistemanombreSistema()
    + "' AND Instalacion_name='Mina de Metal';\n"
+"UPDATE  Planeta_has_Recurso "
            + "SET cantidad = cantidad + ((@nivelInstalacion * @generacionBase) * @segunc
            + "WHERE Recurso_name = 'Metal' "
            + "AND Recurso_instalacion_name = 'Mina de Metal' "
            + "AND Planeta_coordenadaX = " + planetaDestino.getCoordenadaX()
            + " AND Planeta_coordenadaY = " + planetaDestino.getCoordenadaY()
            + " AND Planeta_Sistema_nombreSistema = '" + planetaDestino.getSistemanombreS
+ "SELECT nivelInstalacion, ultimaGeneracion INTO @nivelInstalacion, @ultimaGeneracion "
            + "FROM Planeta_has_Instalacion "
            + "WHERE Planeta_coordenadaX =" +planetaDestino.getCoordenadaX()
            + " AND Planeta_coordenadaY =" +planetaDestino.getCoordenadaY()
            + " AND Planeta_Sistema_nombreSistema ='" +planetaDestino.getSistemanombreSis
            + "' AND Instalacion_name='Mina de Oro';\n"
+ "IF " + idPirata  +" > 0 THEN\n"
```

Figura 116. Trigger de la fase de pre-batalla (parte 2). (Elaboración propia)

Luego se inicializan las naves que participarán en la batalla, las naves atacantes y las naves defensoras.

```
+ "OPEN atacanteNom;\n
+ "atacantes_loopInicio: LOOP\n"
    + "Fetch atacanteNom INTO nom;\n"
    + "IF done THEN\n"
        + "LEAVE atacantes_loopInicio;\n"
    + "END IF;\n"
    + "INSERT INTO NaveCombate (nombreNave, ataqueNave, hullNave, escudoNave, velocidadNave, ag
    + "SELECT nombreNave, ataqueNave, hullNave, escudoNave, velocidadNave, agilidadNave, capaci
        + "FROM Nave WHERE nombreNave = nom;\n"
    + "SELECT cantidad INTO cantidad_aux "
        + "FROM Movimiento_has_Nave "
        + "WHERE Movimiento_idMovimiento =" +movimiento.getIdMovimiento()
        + " AND Nave_nombreNave = nom;\n"
    +"UPDATE NaveCombate "
        +"SET "
            + "ataqueNave = ataqueNave * cantidad_aux,"
            + "hullNave = hullNave * cantidad_aux,"
            + "escudoNave = escudoNave * cantidad_aux,"
            + "agilidadNave = agilidadNave * cantidad_aux,"
            + "cantidad = cantidad_aux "
        + "WHERE nombreNave = nom;\n"
+ "END LOOP;\n"
+ "SET done = FALSE;\n"
+ "CLOSE atacanteNom;\n"
+ "OPEN defensorNom;\n"
+ "defensores_loopInicio: LOOP\n"
    + "Fetch defensorNom INTO nom;\n"
    + "IF done THEN\n"
        + "LEAVE defensores_loopInicio;\n"
    + "END IF;\n"
    + "INSERT INTO NaveDefensa (nombreNave, ataqueNave, hullNave, escudoNave, velocidadNave, ag
    + "SELECT nombreNave, ataqueNave, hullNave, escudoNave, velocidadNave, agilidadNave, capaci
        + "FROM Nave WHERE nombreNave = nom;\n"
    + "SELECT cantidad INTO cantidad_aux "
        + "FROM Planeta_has_Nave "
        + "WHERE Planeta_coordenadaX =" +planetaDestino.getCoordenadaX()
        + " AND Planeta_coordenadaY =" +planetaDestino.getCoordenadaY()
        + " AND Nave_nombreNave = nom;\n"
    + "UPDATE NaveDefensa "
```

Figura 117. Trigger de la fase de pre-batalla (parte 2). (Elaboración propia)

Tras la inicialización comienza la fase de batalla, esta fase transcurre en 6 rondas, si en esas 6 rondas ninguno de los dos bandos acaba con el contrario entonces se considera victoria del defensor.

La fase de batalla se divide en dos partes, en la primera el atacante realiza su ataque mediante un bucle donde cada una de sus naves ataca a una nave rival.

```
+ "Fetch atacanteNom INTO nom;\n"
+ "IF done THEN\n"
    + "LEAVE atacantes_loop;\n"
    + "CLOSE atacanteNom;\n"
+ "END IF;\n"
+ "SELECT COUNT(*) INTO @rank "
    + "FROM NaveDefensa "
    + "WHERE cantidad > 0;\n"
+ "SET randomNumber = 0;\n"
+ "SELECT FLOOR(RAND() * (@rank - 1 + 1)) + 1 INTO randomNumber;\n"
+ "SET @rank = 0;\n"
+ "SELECT "
    + "nombreNave,"
    + "ataqueNave,"
    + "hullNave,"
    + "escudoNave,"
    + "agilidadNave,"
    + "cantidad "
    + "INTO nomEnem , atEnem , hullEnem , shieldEnem , agEnem , cantEnem FROM "
    + "("
        + "SELECT @rank := @rank + 1 AS posicion,"
            + "nombreNave,"
            + "ataqueNave,"
            + "hullNave,"
            + "escudoNave,"
            + "agilidadNave,"
            + "cantidad "
            + "FROM NaveDefensa "
            + "WHERE cantidad > 0"
    + ") AS RankTable "
    + "WHERE posicion = randomNumber;\n"
+ "SELECT ataqueNave INTO atAl "
    + "FROM NaveCombate "
    + "WHERE nombreNave = nom;\n"
```

Figura 118. Trigger de la fase de batalla (parte 1). (Elaboración propia)

Si el ataque de la nave atacante es mayor que el escudo del defensor el daño restante pasa a la salud. En caso de que la salud baje de 0 la nave rival es destruida, si no entonces se calculan las naves defensoras restantes dividiendo su salud total restante entre la salud de una sola nave y quedándonos con la parte entera.

```
    + "IF atAl > shieldEnem THEN\n"
        + "SET hullEnem = hullEnem - (atAl - shieldEnem);\n"
        + "SET shieldEnem = 0;\n"
    + "ELSE\n"
        + "SET shieldEnem = shieldEnem - atAl;\n"
    + "END IF;\n"
    + "IF hullEnem <= 0 THEN\n"
        + "UPDATE NaveDefensa SET cantidad = 0 WHERE nombreNave = nomEnem;\n"
    + "ELSE\n"
        + "SELECT hullNave, escudoNave, ataqueNave, agilidadNave INTO hullOne,
        + "FROM Nave "
        + "WHERE nombreNave = nomEnem;\n"
        + "SET cantidad_aux  = FLOOR(hullEnem/hullOne);\n"
        + "UPDATE NaveDefensa "
            + "SET "
                + "cantidad = cantidad_aux, "
                + "hullNave = hullEnem,"
                + "escudoNave = shieldEnem,"
                + "ataqueNave = atOne * cantidad_aux,"
                + "agilidadNave = agOne * cantidad_aux "
            + "WHERE nombreNave = nomEnem;\n"
    + "END IF;\n"
+ "END LOOP;\n"
+ "SET done = FALSE;\n"
+ "CLOSE atacanteNom;\n"
```

Figura 119. Trigger de la fase de batalla (parte 2). (Elaboración propia)

Las naves defensoras comenzarían con el mismo proceso, así hasta que en algún bando no queden naves o transcurran las 6 rondas.

Cabe destacar la ausencia de la agilidad, en un futuro se plantea añadirla de manera que provoque aleatoriedad en las batallas.

Por último comienza la fase de post-batalla, se crea un informe de batalla y se le asigna al planeta atacado.

Posteriormente se actualizan las naves defensoras en función del resultado y se elimina el trigger que permitía cancelar el ataque.

```
+ "INSERT INTO InformeBatalla (Movimiento_idMovimiento) VALUES(" +movimiento.getIdMovimie
+ "SELECT idBatalla INTO idBat FROM InformeBatalla WHERE Movimiento_idMovimiento = " +mov
+ "UPDATE Planeta"
    + " SET InformeBatalla_idBatalla = idBat"
    + " WHERE coordenadaX = " +planetaDestino.getCoordenadaX()
    + " AND coordenadaY = " +planetaDestino.getCoordenadaY()
    + " AND Sistema_nombreSistema= '" +planetaDestino.getSistemanombreSistema() +"';\n"
+ "OPEN defensorNom;\n"
+ "defensores_loopResultado: LOOP\n"
    + "Fetch defensorNom INTO nom;\n"
    + "IF done THEN\n"
        + "LEAVE defensores_loopResultado;\n"
    + "END IF;\n"
    + "SELECT cantidad INTO cantidad_aux "
        + "FROM NaveDefensa "
        + "WHERE nombreNave = nom;\n"
    + "SELECT cantidad INTO cantidad_enviada "
        + "FROM Planeta_has_Nave "
        + "WHERE Nave_nombreNave = nom "
        + "AND Planeta_coordenadaX =" +planetaDestino.getCoordenadaX()
        + " AND Planeta_coordenadaY =" +planetaDestino.getCoordenadaY()
        + " AND Planeta_Sistema_nombreSistema ='" +planetaDestino.getSistemanombreSistema
    + "UPDATE Planeta_has_Nave "
        + "SET cantidad = cantidad_aux  "
        + "WHERE Nave_nombreNave = nom "
        + "AND Planeta_coordenadaX =" +planetaDestino.getCoordenadaX()
        + " AND Planeta_coordenadaY =" +planetaDestino.getCoordenadaY()
        + " AND Planeta_Sistema_nombreSistema ='" +planetaDestino.getSistemanombreSistema
    + "IF " + usuarioDestino + " is not null THEN\n"
        + "UPDATE Usuario_has_Nave\n"
        + "SET cantidad = cantidad - (cantidad_enviada - cantidad_aux)\n"
        +"WHERE Usuario_username = " + usuarioDestino
        + " AND Nave_nombreNave = nom;\n"
    + "END IF;\n"
    + "INSERT INTO InformeBatalla_has_Nave_Defensa VALUES(idBat, nom, cantidad_enviada, c
+ "END LOOP;\n"
+ "SET done = FALSE;\n"
+ "CLOSE defensorNom;\n"
+ "DROP EVENT movimiento_" +movimiento.getIdMovimiento() + "_ships;\n"
```

Figura 120. Trigger de la fase de post-batalla (parte 1). (Elaboración propia)

En caso de victoria se crea un movimiento de vuelta de las naves, se restan las naves perdidas del total y se cogen del defensor tantos recursos como carga disponible y recursos tenga el rival (el 30% de sus recursos).

```
+ "INSERT into Movimiento (tiempoLlegada, tiempoEnvio,Usuario_username, "
    + "Planeta_coordenadaX,Planeta_coordenadaY,Planeta_Sistema_nombreSistema,movi
+ "Select MAX(idMovimiento) INTO idMov "
    + "FROM Movimiento"
    +" WHERE Planeta_coordenadaX=" + planetaOrigen.getCoordenadaX()
    +" AND Planeta_coordenadaY=" + planetaOrigen.getCoordenadaY()
    +" AND Planeta_Sistema_nombreSistema='" + planetaOrigen.getSistemanombreSiste
+ "OPEN atacanteNom;\n"
+ "atacantes_loopResultado: LOOP\n"
    + "Fetch atacanteNom INTO nom;\n"
    + "IF done THEN\n"
        + "LEAVE atacantes_loopResultado;\n"
    + "END IF;\n"
    + "SELECT cantidad INTO cantidad_aux "
        + "FROM NaveCombate  "
        + "WHERE nombreNave = nom;\n"
    +" SET totalCargo = totalCargo + (SELECT capacidadCarga*cantidad FROM NaveCom
    +"SELECT FLOOR(cantidad * 0.3) INTO metalObtenido "
        + "FROM Planeta_has_Recurso "
        + "WHERE Recurso_name = 'Metal' "
        + "AND Recurso_instalacion_name = 'Mina de Metal' "
        + "AND Planeta_coordenadaX = " + planetaDestino.getCoordenadaX()
        + " AND Planeta_coordenadaY = " + planetaDestino.getCoordenadaY()
        + " AND Planeta_Sistema_nombreSistema = '" + planetaDestino.getSistemanom
    +"SELECT FLOOR(cantidad * 0.3) INTO oroObtenido "
        + "FROM Planeta_has_Recurso "
        + "WHERE Recurso_name = 'Oro' "
        + "AND Recurso_instalacion_name = 'Mina de Oro' "
        + "AND Planeta_coordenadaX = " + planetaDestino.getCoordenadaX()
        + " AND Planeta_coordenadaY = " + planetaDestino.getCoordenadaY()
        + " AND Planeta_Sistema_nombreSistema = '" + planetaDestino.getSistemanom
    +"SELECT FLOOR(cantidad * 0.3) INTO petroleoObtenido "
        + "FROM Planeta_has_Recurso "
        + "WHERE Recurso_name = 'Petroleo' "
        + "AND Recurso_instalacion_name = 'Plataforma Petrolifera' "
        + "AND Planeta_coordenadaX = " + planetaDestino.getCoordenadaX()
        + " AND Planeta_coordenadaY = " + planetaDestino.getCoordenadaY()
        + " AND Planeta_Sistema_nombreSistema = '" + planetaDestino.getSistemanom
    + "SET offsetRecursos = 0;\n"
    + "IF metalObtenido < totalCargo/3 THEN\n"
```

Figura 121. Trigger de la fase de post-batalla (parte 2). (Elaboración propia)

Por último se crean los informes de batalla restantes, se modifica el tiempo del trigger de vuelta de naves, en el caso de que el planeta defensor fuera un pirata se comprueba si se desbloqueó alguna de sus naves disponibles y se envía un mensaje al atacante y al defensor con el resultado de la batalla.

```
+ "ALTER EVENT movimiento_" +movimiento.getIdMovimiento() + "_battleShips ON SCHEDULE
+ "INSERT INTO InformeBatalla_has_Recurso VALUES(idBat, 'Metal', 'Mina de Metal', met
+ "INSERT INTO InformeBatalla_has_Recurso VALUES(idBat, 'Oro', 'Mina de Oro', oroObte
+ "INSERT INTO InformeBatalla_has_Recurso VALUES(idBat, 'Petroleo', 'Plataforma petro
+ "INSERT INTO Mensaje (asunto, descripcion, TipoMensaje_name, fechaEnvio, InformeBat
+ "IF " + idPirata  +" > 0 THEN\n"
    + "SET randomNumberFloat = 0;\n"
    + "SELECT RAND() * (100 - 1 + 1) + 1 INTO randomNumberFloat;\n"
    + "SELECT Nave_nombreNave INTO naveDesbloqueada"
        + " FROM Pirata_has_Desbloqueo_Nave\n"
        + "WHERE Pirata_idPirata = " + idPirata +"\n"
        + "AND probabilidadDesbloqueo > randomNumberFloat\n"
        + "AND Nave_nombreNave not IN (SELECT Nave_nombreNave as nombreNave\n"
                                    + "FROM Usuario_has_Nave\n"
                                    + "WHERE Usuario_username = '" +planetaOrigen
        + "ORDER BY probabilidadDesbloqueo\n"
        + "LIMIT 1;\n"
    + "IF naveDesbloqueada is not null THEN\n"
        + "UPDATE InformeBatalla "
            + "SET Nave_nombreNaveDesbloqueada = naveDesbloqueada "
            + "WHERE idBatalla = idBat;\n"
        + "INSERT INTO Usuario_has_Nave (Usuario_username, Nave_nombreNave, cantidad)
        + "INSERT INTO Planeta_has_Nave (Planeta_coordenadaX, Planeta_coordenadaY, Pl
    + "END IF;\n"
+ "END IF;\n"
+ "SELECT LAST_INSERT_ID() INTO idMen;\n"
+ "INSERT INTO Usuario_has_Mensaje (Usuario_username, Mensaje_idMensaje, descartado)
+ "IF " + usuarioDestino + " is not null THEN\n"
    + "INSERT INTO Mensaje (asunto, descripcion, TipoMensaje_name, fechaEnvio, Inform
    + "SELECT LAST_INSERT_ID() INTO idMen;\n"
    + "INSERT INTO Usuario_has_Mensaje (Usuario_username, Mensaje_idMensaje, descarta
+ "END IF;\n"
```

Figura 122. Trigger de la fase de post-batalla (parte 3). (Elaboración propia)

Si la batalla acabó en derrota por parte del atacante se elimina el trigger de vuelta de naves, se restan del total las naves que se enviaron y se envía un mensaje al atacante y al defensor con el resultado de la batalla.

```
+ "ELSE\n"
    + "DROP EVENT movimiento_" +movimiento.getIdMovimiento() + "_battleShips;\n"
    + "OPEN atacanteNom;\n"
    + "atacantes_loopActualizarNaves: LOOP\n"
        + "Fetch atacanteNom INTO nom;\n"
        + "IF done THEN\n"
            + "LEAVE atacantes_loopActualizarNaves;\n"
        + "END IF;\n"
        + "SELECT cantidad INTO cantidad_enviada "
            + "FROM Movimiento_has_Nave "
            + "WHERE Movimiento_idMovimiento = " +movimiento.getIdMovimiento()
            + " AND Nave_nombreNave = nom;\n"
        + "UPDATE Usuario_has_Nave\n"
            + "SET cantidad = cantidad - cantidad_enviada\n"
            + "WHERE Usuario_username = '" + movimiento.getUsuariousername().getUserna
            + "' AND Nave_nombreNave = nom;\n"
        + "INSERT INTO InformeBatalla_has_Nave_Ataque VALUES(idBat, nom, cantidad_envi
    + "END LOOP;\n"
    + "SET done = FALSE;\n"
    + "CLOSE atacanteNom;\n"
    + "INSERT INTO Mensaje (asunto, descripcion, TipoMensaje_name, fechaEnvio, Informe
    + "SELECT idMensaje INTO idMen FROM Mensaje WHERE InformeBatalla_idBatalla = idBat
    + "INSERT INTO Usuario_has_Mensaje (Usuario_username, Mensaje_idMensaje, descartac
    + "IF " + usuarioDestino + " is not null THEN\n"
        + "INSERT INTO Mensaje (asunto, descripcion, TipoMensaje_name, fechaEnvio, Inf
        + "SELECT LAST_INSERT_ID() INTO idMen;\n"
        + "INSERT INTO Usuario_has_Mensaje (Usuario_username, Mensaje_idMensaje, desca
    + "END IF;\n"
+ "END IF;\n"
```


Figura 123. Trigger de la fase de post-batalla (parte 4). (Elaboración propia)

3.3.6.3.2. Segunda iteración

En esta iteración se realizaron las ventanas del módulo con su código asociado.

3.3.6.3.2.1. Realizar la ventana de ataque

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

De esta ventana destaca el cálculo de distancias y tiempo en función de las naves enviadas y la posición de los planetas.

La distancia se obtiene calculando la distancia entre dos puntos en un plano ya que cada planeta tiene una coordenada X y una coordenada Y.

$$u = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Esto nos da un valor llamado unidad que establece la distancia real entre dos puntos, esta distancia es en minutos, por ejemplo la distancia entre (5,3) y (2,1) es 3,6 minutos.

Después se coge la nave con la menor velocidad (V_{min}) y se divide entre 100, se hace de esta manera ya que una nave con velocidad de 100 (velocidad máxima) tardaría en llegar la velocidad real entre los dos planetas, y una nave con menor velocidad tardaría más tiempo del real.

Por último el tiempo real se divide entre este valor obtenido, de esta manera obtendremos el tiempo en minutos que tardará la flota en llegar al planeta destino.

La fórmula quedaría:

$$t_{viaje} = u/(\frac{V_{min}}{100})$$

Para ataques entre distintos sistemas se calcula la distancia entre los dos sistemas y se le suma 10 a la distancia para añadir realismo. Se le suma 2 a la distancia de los sistemas y se multiplica con la distancia entre los dos planetas.

```
public static double calcularDistancia(Planeta planetaOrigen, Planeta planetaDestino, It
{
    Nave_ataque aux;
    int cantidadTotal = 0;
    double distanciaSistemaX = Math.pow(planetaDestino.getSistema().getCoordenadaX() - p
    double distanciaSistemaY = Math.pow(planetaDestino.getSistema().getCoordenadaY() - p
    double distanciaSistemas = Math.sqrt(distanciaSistemaX + distanciaSistemaY);
    double distanciaPlanetaX = Math.pow(planetaDestino.getCoordenadaX() - planetaOrigen.
    double distanciaPlanetaY = Math.pow(planetaDestino.getCoordenadaY() - planetaOrigen.
    double distanciaPlanetas = Math.sqrt(distanciaPlanetaX + distanciaPlanetaY);
    if(distanciaSistemas > 0) distanciaPlanetas += 10;
    distanciaPlanetas *= distanciaSistemas + 2;
    float velocidadMinima = 100;
    double tiempoLlegada;

    while(iterator.hasNext())
    {
        aux = iterator.next();
        if(!NaveDataValidator.comprobarValorInteger(aux.cantidadF.getValue()) || Integer
        if(aux.getNave().getNave().getVelocidadNave() < velocidadMinima) velocidadMinima
        cantidadTotal += Integer.parseInt(aux.cantidadF.getValue());
    }

    if(cantidadTotal <= 0) return 0;

    tiempoLlegada = distanciaPlanetas/(velocidadMinima/100);

    return tiempoLlegada;
}
```

Figura 124. Método de calcular la distancia entre planetas. (Elaboración propia)

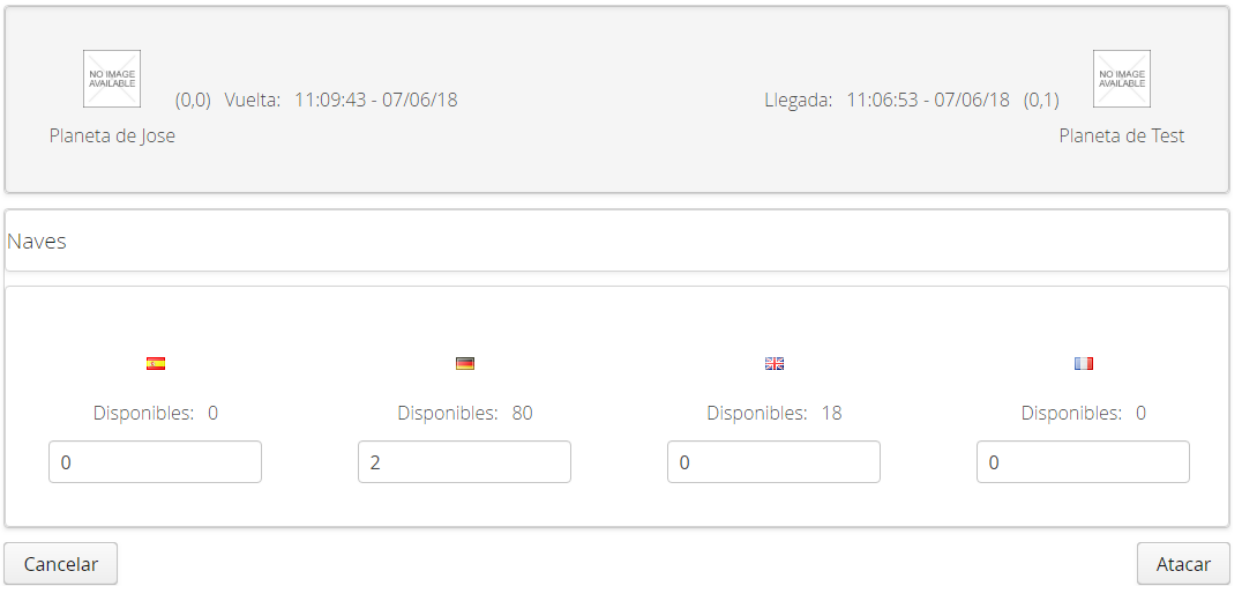

Figura 125. Ventana antigua de ataque. (Elaboración propia)

3.3.6.3.2.2. Realizar las ventanas de los movimientos

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

En este grupo de ventanas solo cabe destacar que la imagen entre los planetas se desplaza en función del tiempo restante, para ello se le resta al tiempo total el tiempo restante y el resultado de eso se divide entre el tiempo total, con esto ya tenemos el tiempo que ha pasado, pero nos interesa el restante.

$$t = (t_{total} - t_{restante})/t_{total}$$

Ese valor se multiplica por 100 para obtener un valor entre 0 y 100, se le resta a 100 para obtener el tiempo restante y se divide entre 10 para pasarlo a un formato válido que nos permita moverlo entre los dos planetas (expand ratio).

$$e_{ratio} = [100 - (t * 100)]/10$$

```
private void setExpandRatio(Movimiento movimiento)
{
    float expandRatio;
    Date fechaAhora = new Date();
    double secondsTotal = (movimiento.getTiempoLlegada().getTime() - movimiento.getT
    double secondsRestante = (movimiento.getTiempoLlegada().getTime() - fechaAhora.g
    double seconds = ((secondsTotal - secondsRestante) / secondsTotal);
    expandRatio = (float) (100 - (seconds * 100)) / 10;

    if(expandRatio < 0.5) expandRatio = (float) 0.5;
    hLayout.setExpandRatio(layoutTiempoRestante, expandRatio);
}
```

Figura 126. Método de desplazamiento de la imagen de distancia entre planetas. (Elaboración propia)

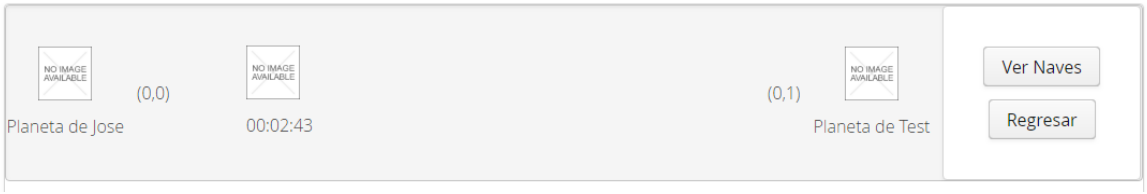

Figura 127. Ventana antigua de un movimiento. (Elaboración propia)

3.3.6.3.2.3. Realizar la ventana de los mensajes de tipo batalla

Se creó y diseñó la ventana inicializando todos los componentes gráficos, haciendo las llamadas a las clases CRUD y redirigiendo según el resultado obtenido.

Simplemente muestra los datos de los informes de batalla, el informe de ataque, el de defensa y el de recursos con las naves desbloqueadas.

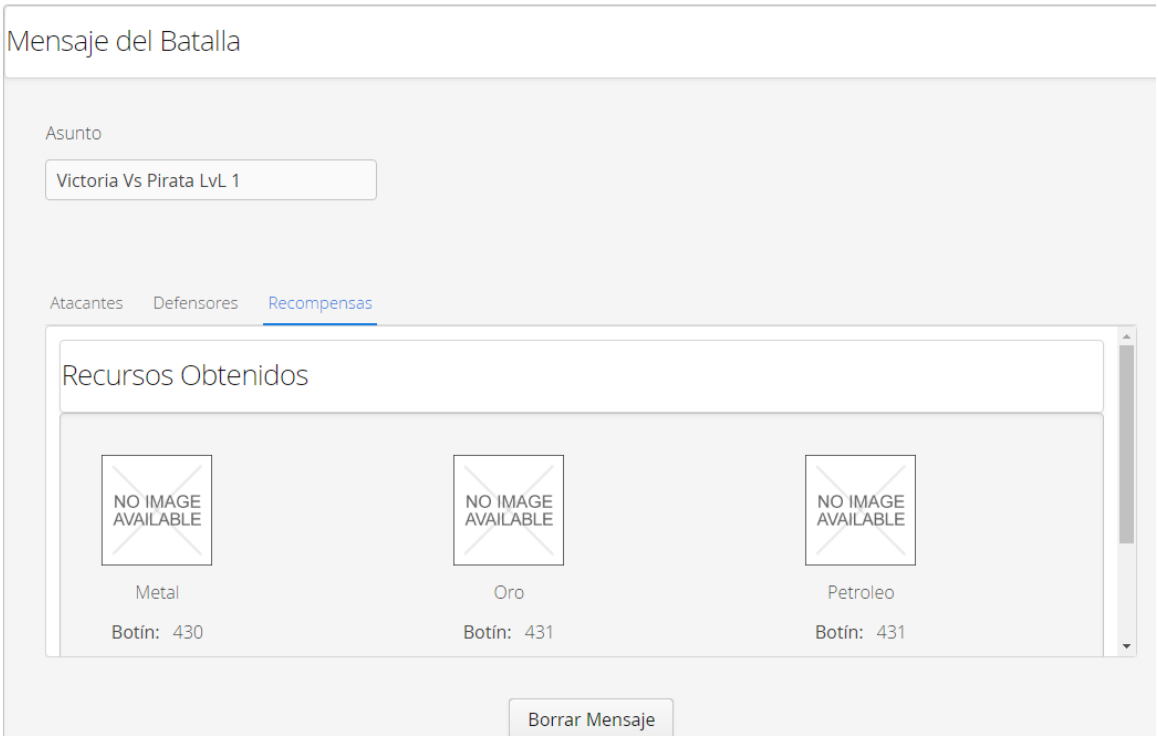

Figura 128. Ventana antigua de un mensaje de tipo batalla. (Elaboración propia)

3.3.6.3.2.4. Ajustar los triggers para que funcionen correctamente

Debido a la complejidad de los triggers surgieron muchos errores en ellos que se fueron solucionando, por ejemplo algunos cursores no se cerraban correctamente, probablemente surjan más errores sobre todo por el id máximo.

3.3.6.4. Feedback

No se ha obtenido ningún consejo de mejora tras probarlo el usuario.

3.3.7. Módulo Integración

Se comienza elaborando los casos de uso y los diagramas de clases, una vez hechos se estiman, se reparten las tareas y se comienza a implementar.

3.3.7.1. Análisis de requisitos

A continuación se realizará un análisis de requisitos detallado de este módulo a partir de la información obtenida en el estudio del problema y del análisis preliminar realizado.

3.3.7.1.1. Casos de Uso del módulo

En esta subsección se mostrarán los diagramas de casos de uso y los casos de uso del módulo.

3.3.7.1.1.1. Diagrama de Casos de Uso del módulo

A continuación se muestran los diagramas de casos de uso del módulo, los diagramas se encuentran con mejor resolución en el anexo.

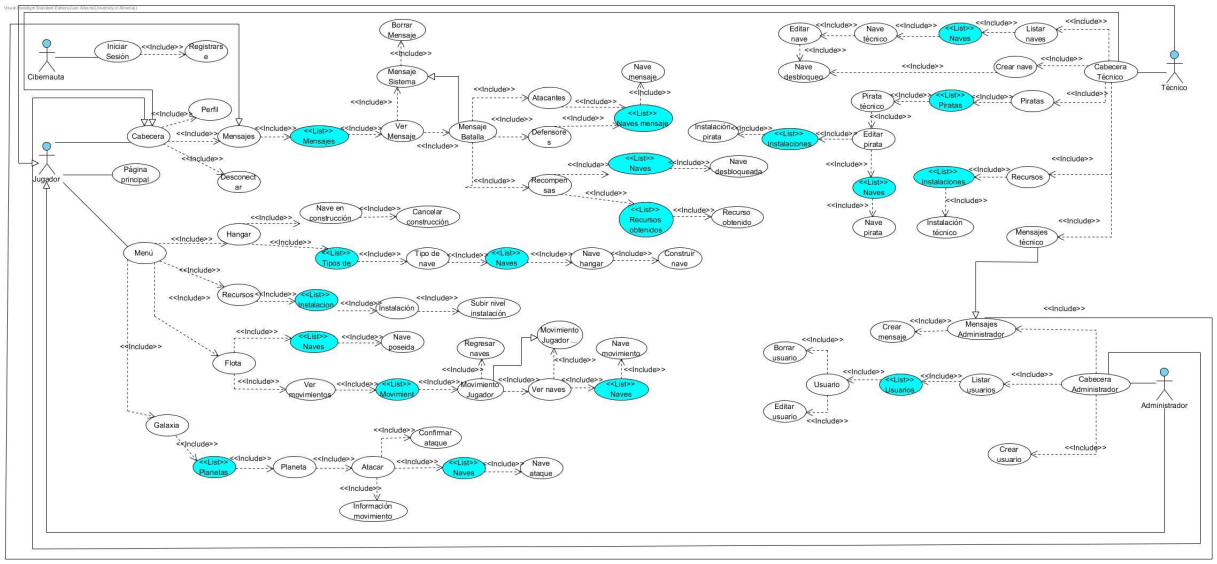

Figura 129. Diagrama de casos de uso del módulo integración. (Elaboración propia)

3.3.7.1.1.2. Especificación de Actores del Sistema

A continuación se muestra la información de los actores del sistema.

Tabla 69. Especificación del cibernauta. (Elaboración propia)

CDU-ACT-0001	Cibernauta
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa al usuario que accede a la web sin haber iniciado sesión en ella.
Comentarios	Ninguno

Tabla 70. Especificación del jugador. (Elaboración propia)

CDU-ACT-0002	Jugador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona que se registra en la web.
Comentarios	Ninguno

Tabla 71. Especificación del administrador. (Elaboración propia)

CDU-ACT-0003	Administrador
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona encargada de la gestión de usuarios de la web.
Comentarios	Ninguno

Tabla 72. Especificación del técnico. (Elaboración propia)

CDU-ACT-0004	Técnico
[Versión]	1.0 (06/03/2018)
Descripción	Este actor representa a la persona cuya finalidad es la de controlar la jugabilidad mediante la modificación de varios parámetros del juego como las características de las naves y la generación de recursos.
Comentarios	Ninguno

3.3.7.1.1.3. Especificación de Casos de Uso del módulo

En esta subsección se tratarán los casos de uso del módulo.

Tabla 73. Caso de uso identificarse. (Elaboración propia)

<table>
<tr><td>CDU-0001</td><td>Identificarse</td></tr>
<tr><td>[Versión]</td><td>1.0 (06/03/2018)</td></tr>
<tr><td>[Dependencias]</td><td></td></tr>
<tr><td>Descripción</td><td>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el cibernauta quiera identificarse en el sistema.</td></tr>
<tr><td>Precondición</td><td>No estar identificado.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>Si el cibernauta quiere registrarse, se realiza el caso de uso Registrarse (CDU-0002)</td></tr>
<tr><td>2</td><td>El actor Cibernauta (CDU-ACT-0001) introduce su email y contraseña.</td></tr>
<tr><td>3</td><td>El sistema autentifica la información e identifica al usuario en el sistema.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El usuario está identificado.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>3</td><td>Si la información no es correcta, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto</td></tr>
<tr><td>3</td><td>Si el usuario está bloqueado, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto</td></tr>
<tr><td>3</td><td>Si el cibernauta no está registrado, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto</td></tr>
</table>
</td></tr>
</table>

Tabla 74. Caso de uso registrarse. (Elaboración propia)

CDU-0002	Registrarse
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el cibernauta quiera registrarse en el sistema o durante la realización de los siguientes casos de uso: [CDU-0001] Identificarse
Precondición	No está registrado.
Secuencia normal	**Paso** / **Acción** 1 / El actor Cibernauta (CDU-ACT-0001) introduce un email, nombre de usuario y dos veces la contraseña. 2 / El sistema registra los datos en la base de datos y confirma al usuario el registro.
Postcondición	Está registrado en el sistema
Excepciones	**Paso** / **Acción** 2 / Si el correo utilizado ya existe en la base de datos, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto 2 / Si el usuario utilizado ya existe en la base de datos, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto 2 / Si las contraseñas no coinciden, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto 2 / Si algún dato es inválido, el sistema informa al cibernauta del problema, a continuación este caso de uso queda sin efecto

Tabla 75. Caso de uso página principal. (Elaboración propia)

CDU-0003	Página principal
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario se identifique
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los recursos que posee el usuario
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 76. Caso de uso desconectarse. (Elaboración propia)

CDU-0004	Desconectarse
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera desconectarse del sistema.
Precondición	Estar identificado en el sistema.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de desconexión. 2 / El sistema desconecta al usuario del sistema
Postcondición	No estar identificado en el sistema.
Excepciones	**Paso** / **Acción** - / -

Tabla 77. Caso de uso perfil. (Elaboración propia)

CDU-0005	Perfil
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar sus datos.
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) solicita al sistema iniciar el procedimiento de consultar sus datos personales. 2 / El sistema muestra su correo, su nombre de usuario y su contraseña. 3 / Si quiere modificar su correo, el actor Jugador (CDU-ACT-0002) introduce un nuevo correo y confirma los cambios. 4 / Si quiere modificar su contraseña, el actor Jugador (CDU-ACT-0002) introduce su nueva contraseña dos veces y confirma los cambios. 5 / El actor Jugador (CDU-ACT-0002) guarda los cambios realizados
Postcondición	-
Excepciones	**Paso** / **Acción** 4 / Si las contraseñas no coinciden, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto 5 / Si el correo utilizado ya existe en la base de datos, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto 5 / Si el correo utilizado es inválido, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto

Tabla 78. Caso de uso crear usuario. (Elaboración propia)

CDU-0006	Crear usuario
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera crear un usuario
Precondición	Estar identificado como administrador
Secuencia normal	**Paso** / **Acción** 1 / El actor Administrador (CDU-ACT-0003) introduce un email, nombre de usuario, dos veces la contraseña, si está bloqueado su acceso y que tipo de usuario es. 2 / El sistema registra los datos en la base de datos y confirma al administrador el registro.
Postcondición	Haya un nuevo usuario en la base de datos
Excepciones	**Paso** / **Acción** 2 / Si el correo utilizado ya existe en la base de datos, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 2 / Si el usuario utilizado ya existe en la base de datos, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 2 / Si las contraseñas no coinciden, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 2 / Si algún dato es inválido, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto

Tabla 79. Caso de uso consultar usuarios. (Elaboración propia)

CDU-0007	Consultar usuarios
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando administrador quiera ver los usuarios registrados en el sistema.
Precondición	Estar identificado como administrador.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los usuarios existentes, su usuario, su email y si está bloqueado su acceso. 2 / Si el administrador quiere editar un usuario, se realiza el caso de uso Editar usuario (CDU-UC-0008) 3 / Si el administrador quiere borrar un usuario, se realiza el caso de uso Borrar usuario (CDU-UC-0009)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 80. Caso de uso editar usuario. (Elaboración propia)

CDU-0008	Editar usuario
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera editar los datos de un usuario o durante la realización de los siguientes casos de uso: [CDU-0007] Consultar usuarios
Precondición	Estar identificado como administrador.
Secuencia normal	**Paso** / **Acción** 1 — Si quiere modificar el correo, el actor Administrador (CDU-ACT-0003) introduce un nuevo correo 2 — Si quiere modificar su contraseña, el actor Administrador (CDU-ACT-0003) introduce su nueva contraseña dos veces 3 — Si quiere bloquearle el acceso, el actor Administrador (CDU-ACT-0003) indica que el usuario no está activo (Si se está modificando a si mismo no puede editar este campo) 4 — Si quiere cambiar el tipo de usuario, el actor Administrador (CDU-ACT-0003) elige el nuevo tipo de usuario que quiere que sea (Si se está modificando a si mismo no puede editar este campo) 5 — El actor Administrador (CDU-ACT-0003) confirma los cambios
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** 5 — Si el correo utilizado ya existe en la base de datos, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 5 — Si el correo utilizado es inválido, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto 5 — Si las contraseñas no coinciden, el sistema informa al administrador del problema, a continuación este caso de uso queda sin efecto

Tabla 81. Caso de uso borrar usuario. (Elaboración propia)

CDU-0009	Borrar usuario
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador quiera borrar un usuario de la base de datos o durante la realización de los siguientes casos de uso: [CDU-0007] Consultar usuarios
Precondición	Estar identificado como administrador.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los datos del usuario a eliminar 2 / El actor Administrador (CDU-ACT-0003) elimina al usuario (No puede eliminarse a si mismo) 3 / El sistema confirma la eliminación exitosa de la base de datos
Postcondición	El usuario ya no se encuentre en la base de datos
Excepciones	**Paso** / **Acción** - / -

Tabla 82. Caso de uso consultar lista de naves. (Elaboración propia)

CDU-0010	Consultar lista de naves
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera ver las naves existentes
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra las naves existentes, su imagen, su nombre, su tipo y si está bloqueada su obtención. 2 / Si el técnico quiere editar una nave, se realiza el caso de uso Editar nave (CDU-0011)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 83. Caso de uso editar nave. (Elaboración propia)

CDU-0011	Editar nave
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiere editar los datos de una nave o durante la realización de los siguientes casos de uso: [CDU-0010] Consultar lista de naves
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 – Si quiere modificar la imagen, el actor Técnico (CDU-ACT-0004) sube una nueva imagen. 2 – Si quiere modificar el tipo de nave, el actor Técnico (CDU-ACT-0004) selecciona un nuevo tipo de nave. 3 – Si quiere modificar las características, el actor Técnico (CDU-ACT-0004) introduce nuevos valores para ellas. (salud, escudo, agilidad, velocidad, daño y capacidad de carga) 4 – Si quiere modificar los costes, el actor Técnico (CDU-ACT-0004) introduce nuevos valores para ellos (metal, oro y petróleo) 5 – Si quiere modificar el tiempo de construcción, el actor Técnico (CDU-ACT-0004) introduce un nuevo valor. 6 – Si quiere modificar la probabilidad de ser desbloqueada en los distintos piratas, el actor Técnico (CDU-ACT-0004) introduce nuevos valores de probabilidad de desbloqueo. 7 – Si quiere desbloquear la nave para todos los usuarios, el actor Técnico (CDU-ACT-0004) desactiva el bloqueo. 8 – El actor Técnico (CDU-ACT-0004) confirma los cambios.
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** 8 – Si algún dato es inválido, el sistema informa al técnico del problema, a continuación este caso de uso queda sin efecto

Tabla 84. Caso de uso consultar lista de piratas. (Elaboración propia)

CDU-0012	Consultar lista de piratas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando técnico quiera ver los piratas existentes.
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 – El sistema muestra los piratas existentes, su imagen, su nivel y el nivel de sus instalaciones. 2 – Si el técnico quiere editar un pirata, se realiza el caso de uso Editar pirata (CDU-0013)
Postcondición	-
Excepciones	**Paso** / **Acción** - – -

Tabla 85. Caso de uso editar pirata. (Elaboración propia)

CDU-0013	Editar pirata
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera editar las instalaciones o naves de un pirata o durante la realización de los siguientes casos de uso: [CDU-0012] Consultar lista de piratas
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / Si quiere modificar el nivel de las instalaciones, el actor Técnico (CDU-ACT-0004) introduce un nuevo nivel. 2 / Si quiere modificar sus naves, el actor Técnico (CDU-ACT-0004) introduce para cada nave la nueva cantidad. 3 / El actor Técnico (CDU-ACT-0004) confirma los cambios
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** - / -

Tabla 86. Caso de uso crear nave. (Elaboración propia)

CDU-0014	Crear nave
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera crear una nave
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / El actor Técnico (CDU-ACT-0004) introduce una imagen, un nombre, el tipo de nave, la salud, el escudo, la agilidad, la velocidad, el daño, los costes de metal, oro y petróleo, el tiempo de construcción, si está bloqueada por defecto y en caso de estar bloqueada la probabilidad de ser desbloqueada en los distintos piratas. 2 / El sistema registra los datos en la base de datos y confirma al administrador el registro.
Postcondición	Haya una nueva nave en la base de datos.
Excepciones	**Paso** / **Acción** 2 / Si el nombre utilizado ya existe en la base de datos, el sistema informa al técnico del problema, a continuación este caso de uso queda sin efecto 2 / Si algún dato es inválido, el sistema informa al técnico del problema, a continuación este caso de uso queda sin efecto

Tabla 87. Caso de uso editar instalaciones. (Elaboración propia)

CDU-0015	Editar instalaciones
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el técnico quiera editar la tasa de generación de recursos de las instalaciones
Precondición	Estar identificado como técnico.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra la generación de recursos por nivel. 2 / Si quiere modificar la producción inicial, el actor Técnico (CDU-ACT-0004) introduce para cada instalación la nueva producción inicial. 3 / Si ha modificado la producción inicial, el sistema actualiza la generación de recursos por nivel mostrada. 4 / El actor Técnico (CDU-ACT-0004) confirma los cambios.
Postcondición	Los cambios realizados son reflejados en la base de datos.
Excepciones	**Paso** / **Acción** - / -

Tabla 88. Caso de uso construcción de naves. (Elaboración propia)

CDU-0016	Construcción de naves
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera gestionar la construcción de naves.
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los tipos de naves existentes. 2 / Si hay naves en construcción, el sistema muestra las naves en construcción y el tiempo restante 3 / Si el usuario desea cancelar la construcción de naves en curso, se realiza el caso de uso Cancelar construcción (CDU-0018) 4 / Si el usuario desea construir naves, se realiza el caso de uso Construir naves (CDU-0017)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 89. Caso de uso construir naves. (Elaboración propia)

CDU-0017	Construir naves
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera construir naves. o durante la realización de los siguientes casos de uso: [CDU-0016] Construcción de naves
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) selecciona el tipo de nave que le interesa construir. 2 / El sistema muestra las naves existentes para el tipo de nave seleccionada. 3 / El actor Jugador (CDU-ACT-0002) selecciona la nave que desea construir. (No puede seleccionar las naves que no tiene desbloqueadas) 4 / El actor Jugador (CDU-ACT-0002) indica el número de naves que desea construir 5 / Si introduce una cantidad de naves mayor a la que puede pagar, el sistema actualiza el número introducido con el máximo válido. 6 / El actor Jugador (CDU-ACT-0002) confirma la construcción 7 / El sistema le descuenta los recursos necesarios y comienza la construcción de las naves
Postcondición	Haya naves en construcción
Excepciones	**Paso** / **Acción** 7 / Si hay naves en construcción, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto

Tabla 90. Caso de uso cancelar construcción. (Elaboración propia)

CDU-0018	Cancelar construcción
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera cancelar la construcción de las naves o durante la realización de los siguientes casos de uso: [CDU-0016] Construcción de naves
Precondición	Estar identificado en el sistema y haya naves construyéndose.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) indica que desea cancelar la producción de naves actual 2 / El sistema cancela la construcción de naves y le devuelve los recursos invertidos.
Postcondición	Las naves no estén en construcción
Excepciones	**Paso** / **Acción** - / -

Tabla 91. Caso de uso subir nivel instalación. (Elaboración propia)

CDU-0019	Subir nivel instalación
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiere subir de nivel alguna instalación
Precondición	Estar identificado en el sistema
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestras las instalaciones existentes 2 / Si posee suficientes recursos, el actor Jugador (CDU-ACT-0002) selecciona la instalación que desea subir de nivel 3 / El sistema descuenta los recursos necesarios y sube de nivel la instalación
Postcondición	Generar más recursos
Excepciones	**Paso** / **Acción** - / -

Tabla 92. Caso de uso ver mensaje. (Elaboración propia)

CDU-0020	Ver mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar sus datos.
Precondición	Estar identificado en el sistema.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) solicita al sistema iniciar el procedimiento de consultar sus mensajes 2 / El sistema muestra una lista de mensajes con su tipo de mensaje, fecha de envío, asunto y planeta de destino si es un mensaje de batalla. 3 / Si quiere ver más información de un mensaje de una batalla, se realiza el caso de uso Mensaje batalla (CDU-0005) 4 / Si quiere ver más información de un mensaje del sistema, se realiza el caso de uso Mensaje sistema (CDU-0021) 5 / Si es administrador o técnico y quiere crear un mensaje, se realiza el caso de uso Crear mensaje (CDU-0023)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 93. Caso de uso mensaje sistema. (Elaboración propia)

CDU-0021	Mensaje sistema
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar un mensaje del tipo sistema. o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Haber seleccionado un mensaje del sistema de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de ver el mensaje seleccionado 2 / El sistema muestra los datos del mensaje, el asunto y la descripción. 3 / Si el usuario quiere borrar el mensaje, se realiza el caso de uso Borrar mensaje (CDU-0022)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 94. Caso de uso borrar mensaje. (Elaboración propia)

CDU-0022	Borrar mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario indique que desea borrar un mensaje. o durante la realización de los siguientes casos de uso: [CDU-0026] Mensaje batalla, [CDU-0021] Mensaje sistema
Precondición	Haber seleccionado un mensaje de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) confirma que desea eliminar el mensaje 2 / El sistema marca el mensaje como descartado
Postcondición	El usuario es redireccionado a [CDU-0020] Ver mensajes y el mensaje no se le mostrará a ese usuario.
Excepciones	**Paso** / **Acción** - / -

Tabla 95. Caso de uso crear mensaje. (Elaboración propia)

CDU-0023	Crear mensaje
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el administrador/técnico quiere enviar un mensaje a todos los usuarios. o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Estar identificado como administrador o técnico.
Secuencia normal	**Paso** / **Acción** 1 / El actor Administrador (CDU-ACT-0003) escribe el asunto y el cuerpo del mensaje 2 / El actor Administrador (CDU-ACT-0003) envía el mensaje 3 / El sistema confirma el envío del mensaje
Postcondición	Todos los usuarios reciben el mensaje
Excepciones	**Paso** / **Acción** - / -

Tabla 96. Caso de uso consultar naves poseídas. (Elaboración propia)

CDU-0024	Consultar naves poseídas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver las naves que posee
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestras las naves que tiene el usuario, indicando las naves en espera y las naves en movimiento.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 97. Caso de uso consultar lista de planetas. (Elaboración propia)

CDU-0025	Consultar lista de planetas
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver los planetas existentes
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los planetas existentes dividiéndolos por sistemas y sectores y mostrando de cada uno una imagen, un nombre y sus coordenadas dentro del sistema.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 98. Caso de uso mensaje batalla. (Elaboración propia)

CDU-0026	Mensaje batalla
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera consultar un mensaje del tipo batalla o durante la realización de los siguientes casos de uso: [CDU-0020] Ver mensaje
Precondición	Haber seleccionado un mensaje de batalla de su lista de mensajes.
Secuencia normal	**Paso** / **Acción** 1 / El actor Jugador (CDU-ACT-0002) inicia el procedimiento de ver el mensaje seleccionado 2 / El sistema muestra los datos del mensaje, las naves atacantes que se enviaron y fueron destruidas, las naves defensoras que se enviaron y fueron destruidas y las recompensas obtenidas (recursos y naves obtenidas). 3 / Si el usuario quiere borrar el mensaje, se realiza el caso de uso Borrar mensaje (CDU-0022)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 99. Caso de uso ver movimientos. (Elaboración propia)

CDU-0027	Ver movimientos
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver los movimientos hacia o desde su planeta o durante la realización de los siguientes casos de uso: [CDU-0024] Consultar naves poseídas
Precondición	Estar identificado
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra los movimientos hacia o desde el planeta del usuario mostrando el planeta de origen y destino y el tiempo de llegada. 2 / Si el movimiento es hacia otro planeta y el usuario quiere cancelarlo, se realiza el caso de uso Regresar naves (CDU-0029) 3 / Si quiere ver más información acerca de un movimiento, se realiza el caso de uso Consultar información de movimiento (CDU-0028)
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 100. Caso de uso consultar información de movimiento. (Elaboración propia)

CDU-0028	Consultar información de movimiento
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver más información de un movimiento o durante la realización de los siguientes casos de uso: [CDU-0027] Ver movimientos
Precondición	El sistema muestra la información del movimiento, el planeta de origen y destino, el tiempo estimado de llegada y las naves que se dirigen al planeta.
Secuencia normal	**Paso** / **Acción** 1 / El sistema muestra la información del movimiento, el planeta de origen y destino, el tiempo estimado de llegada y las naves que se dirigen al planeta.
Postcondición	-
Excepciones	**Paso** / **Acción** - / -

Tabla 101. Caso de uso regresar naves. (Elaboración propia)

CDU-0029	Regresar naves
[Versión]	1.0 (06/03/2018)
[Dependencias]	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera cancelar un movimiento realizado por sus naves o durante la realización de los siguientes casos de uso: [CDU-0027] Ver movimientos
Precondición	El sistema muestra la información del movimiento, el planeta de origen y destino, el tiempo estimado de llegada y las naves que se dirigen al planeta.
Secuencia normal	**Paso** / **Acción** 1 / El sistema cancela el movimiento actual 2 / El sistema crea un nuevo movimiento con las naves regresando al planeta del usuario
Postcondición	Hay un nuevo movimiento creado
Excepciones	**Paso** / **Acción** - / -

Tabla 102. Caso de uso atacar planetas. (Elaboración propia)

<table>
<tr><td>CDU-0030</td><td>Atacar planeta</td></tr>
<tr><td>[Versión]</td><td>1.0 (06/03/2018)</td></tr>
<tr><td>[Dependencias]</td><td></td></tr>
<tr><td>Descripción</td><td>El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera atacar un planeta o durante la realización de los siguientes casos de uso: [CDU-0025] Consultar lista de planetas</td></tr>
<tr><td>Precondición</td><td>Estar identificado</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El sistema muestra todas las naves existentes</td></tr>
<tr><td>2</td><td>El actor Jugador (CDU-ACT-0002) indica la cantidad de cada nave que desea enviar en el ataque</td></tr>
<tr><td>3</td><td>El sistema actualiza el tiempo de llegada en función de las naves y cantidad de naves escogidas.</td></tr>
<tr><td>4</td><td>El actor Jugador (CDU-ACT-0002) realiza el ataque</td></tr>
<tr><td>5</td><td>El sistema registra el ataque en la base de datos y se lo confirma al usuario</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Se ha creado un movimiento para ese ataque</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>5</td><td>Si el planeta de destino tiene protección activa, el sistema informa al usuario del problema, a continuación este caso de uso queda sin efecto</td></tr>
</table>
</td></tr>
</table>

3.3.7.1.2. Ternas de trazabilidad

En esta subsección se mostrará la relación entre las ventanas de la interfaz y sus casos de uso y clases asociadas de los diagramas.

3.3.7.1.2.1. Cibernauta

A continuación se mostrarán las ternas de trazabilidad para el actor cibernauta.

Inicio de sesión

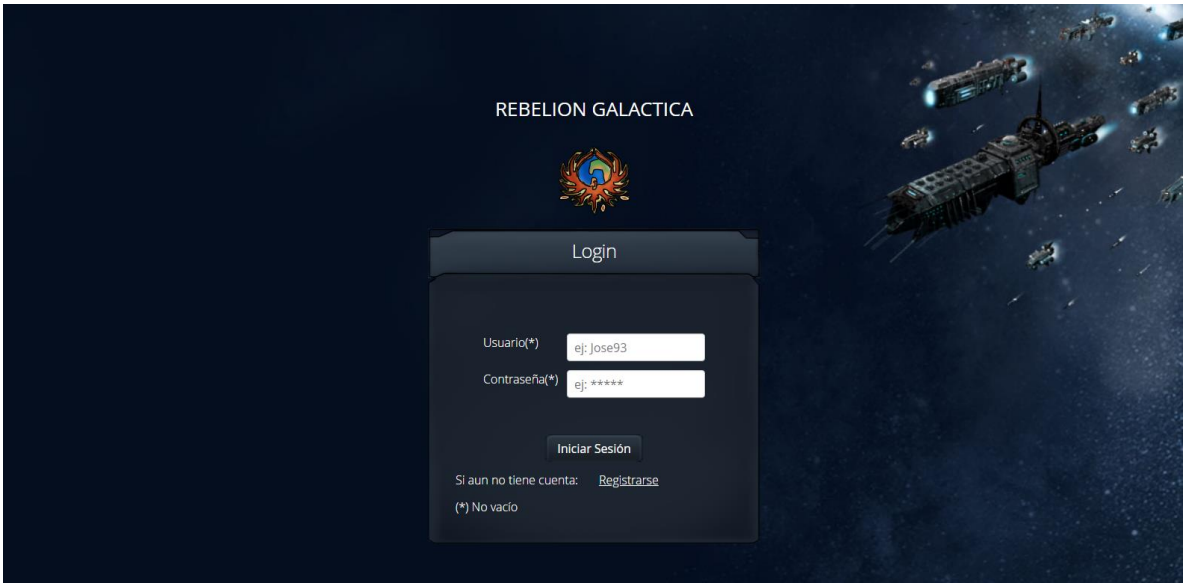

Figura 130. Ventana de inicio de sesión. (Elaboración propia)

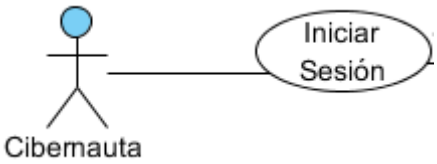

Figura 131. Diagrama casos de uso inicio de sesión. (Elaboración propia)

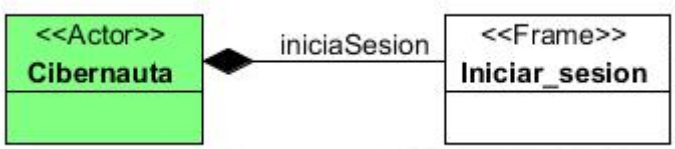

Figura 132. Diagrama de clases de inicio de sesión. (Elaboración propia)

Registro

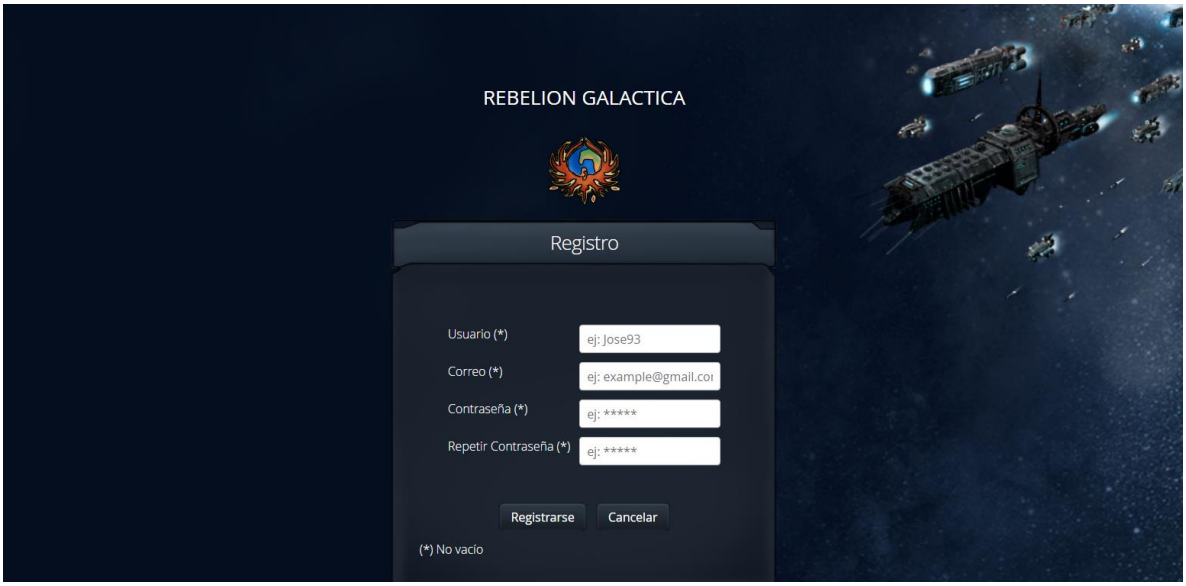

Figura 133. Ventana de registro. (Elaboración propia)

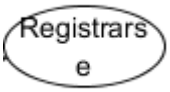

Figura 134. Diagrama casos de uso registrarse. (Elaboración propia)

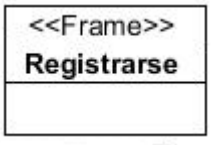

Figura 135. Diagrama de clases registrarse. (Elaboración propia)

3.3.7.1.2.2. Jugador

A continuación se mostrarán las ternas de trazabilidad para el actor jugador.

Pantalla principal (Jugador)

Figura 136. Ventana de la pantalla principal (jugador). (Elaboración propia)

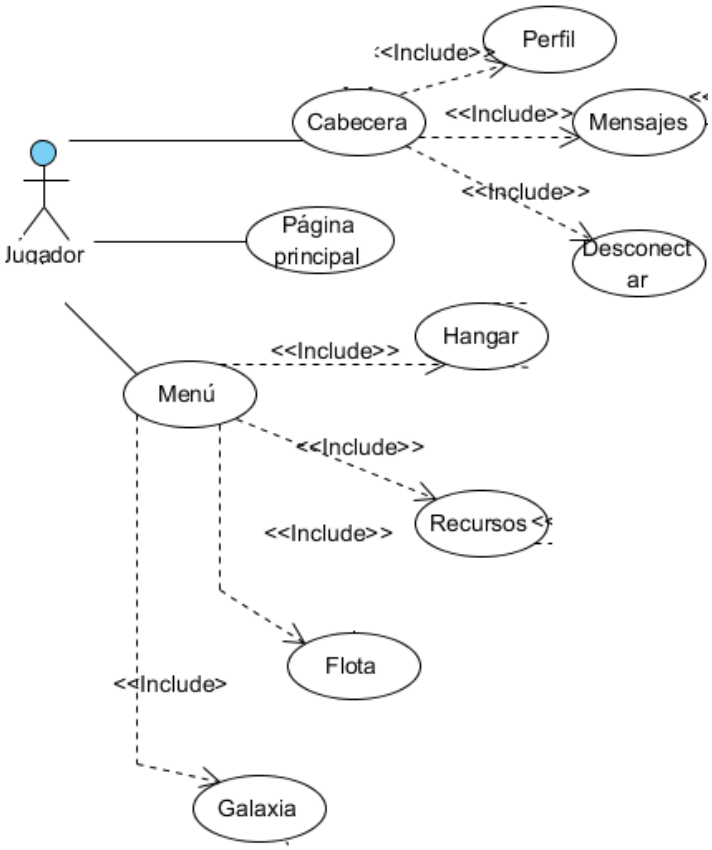

Figura 137. Diagrama casos de uso pantalla principal (Jugador). (Elaboración propia)

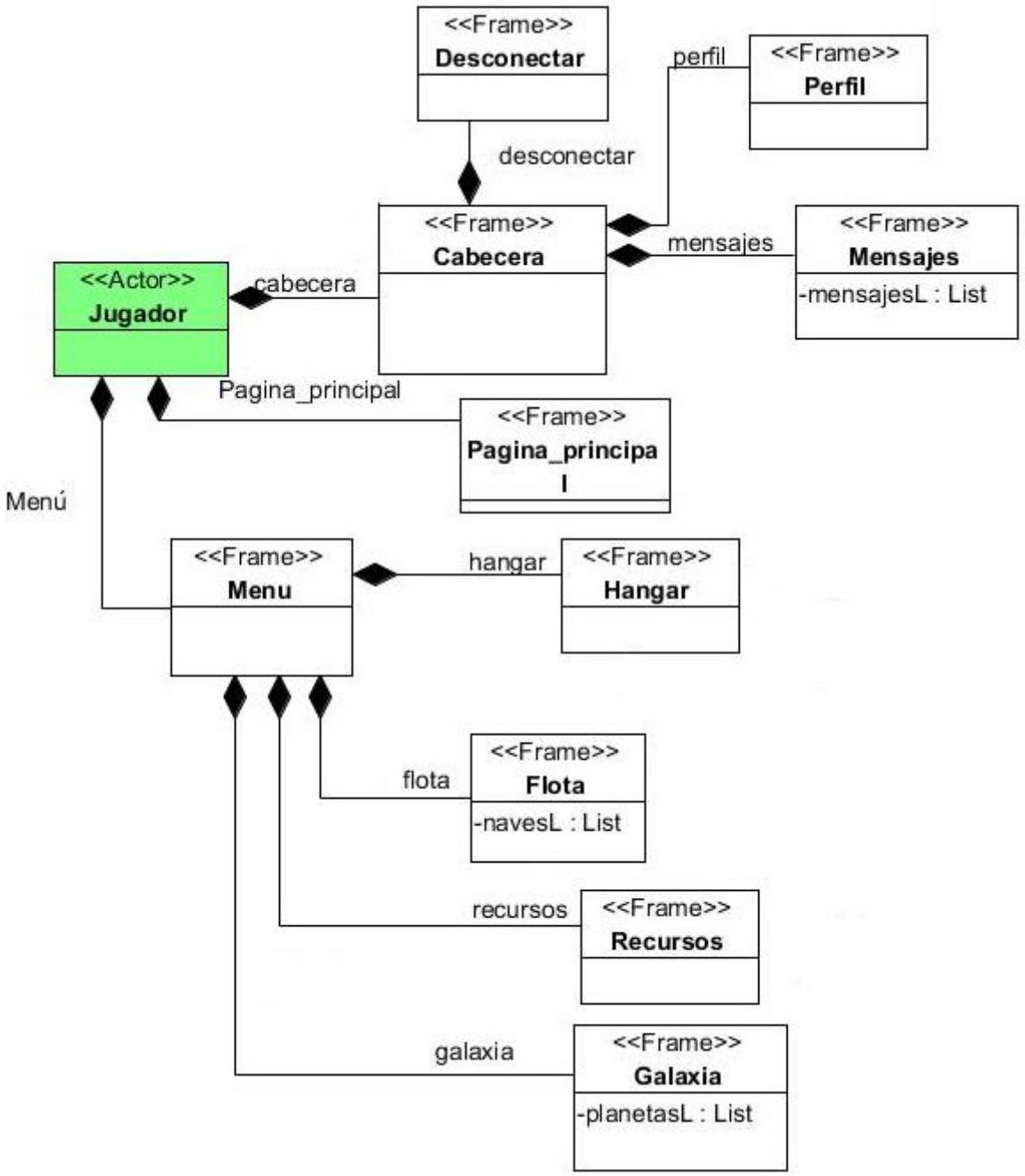

Figura 138. Diagrama de clases pantalla principal (Jugador). (Elaboración propia)

Página principal

Figura 139. Página principal de la ventana pantalla principal. (Elaboración propia)

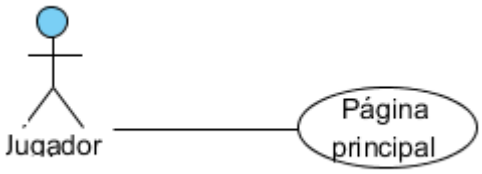

Figura 140. Diagrama casos de uso página principal (Jugador). (Elaboración propia)

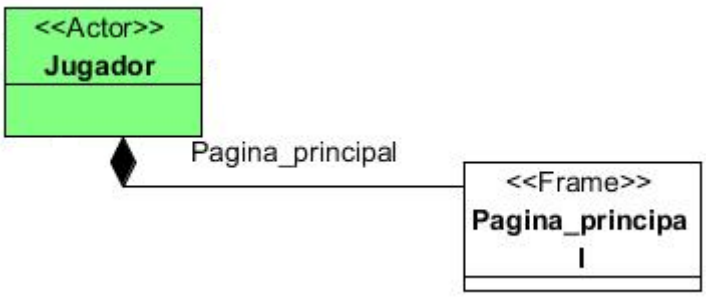

Figura 141. Diagrama de clases página principal (Jugador). (Elaboración propia)

Cabecera (Jugador)

Figura 142. Cabecera del jugador. (Elaboración propia)

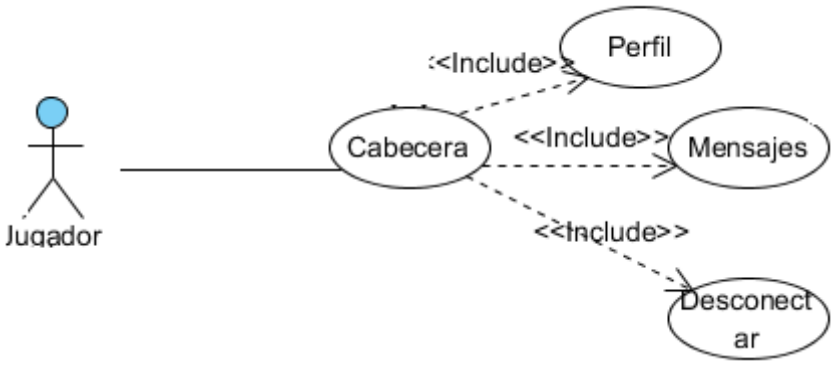

Figura 143. Diagrama casos de uso cabecera (Jugador). (Elaboración propia)

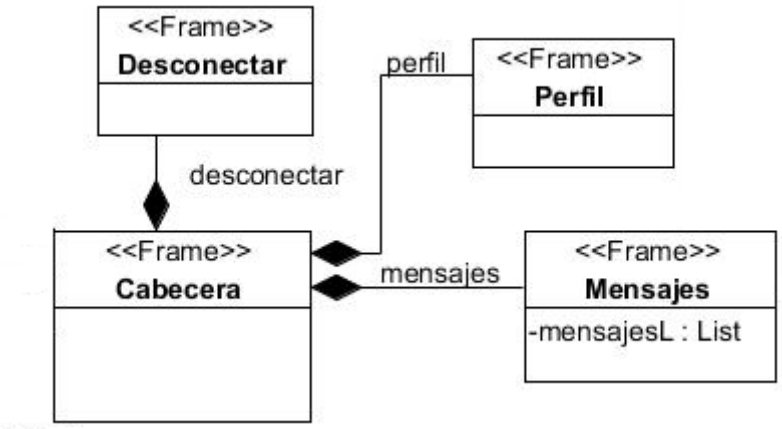

Figura 144. Diagrama de clases cabecera (Jugador). (Elaboración propia)

Menú

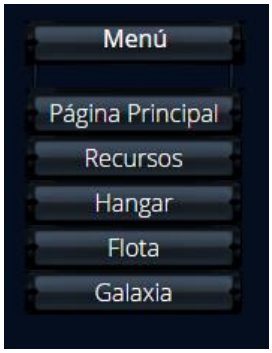

Figura 145. Ventana del menú. (Elaboración propia)

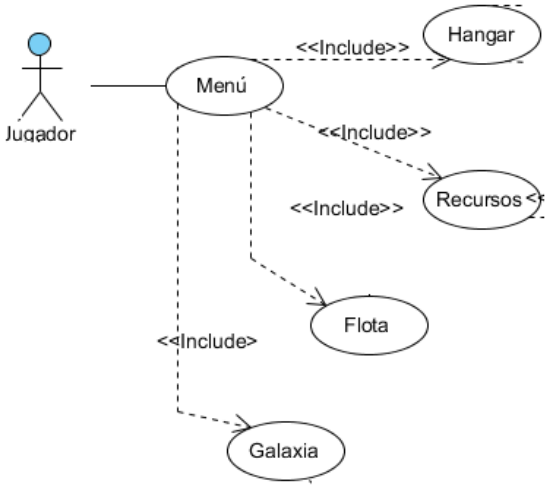

Figura 146. Diagrama casos de uso menú (Jugador). (Elaboración propia)

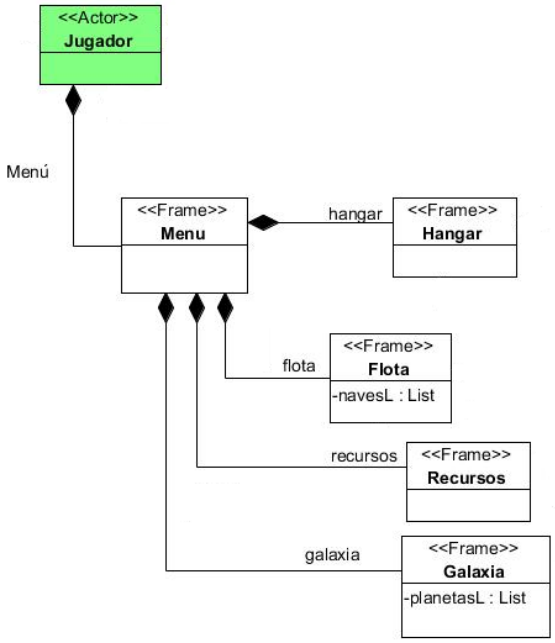

Figura 147. Diagrama de clases menú (Jugador). (Elaboración propia)

Ventana de instalaciones (Recursos)

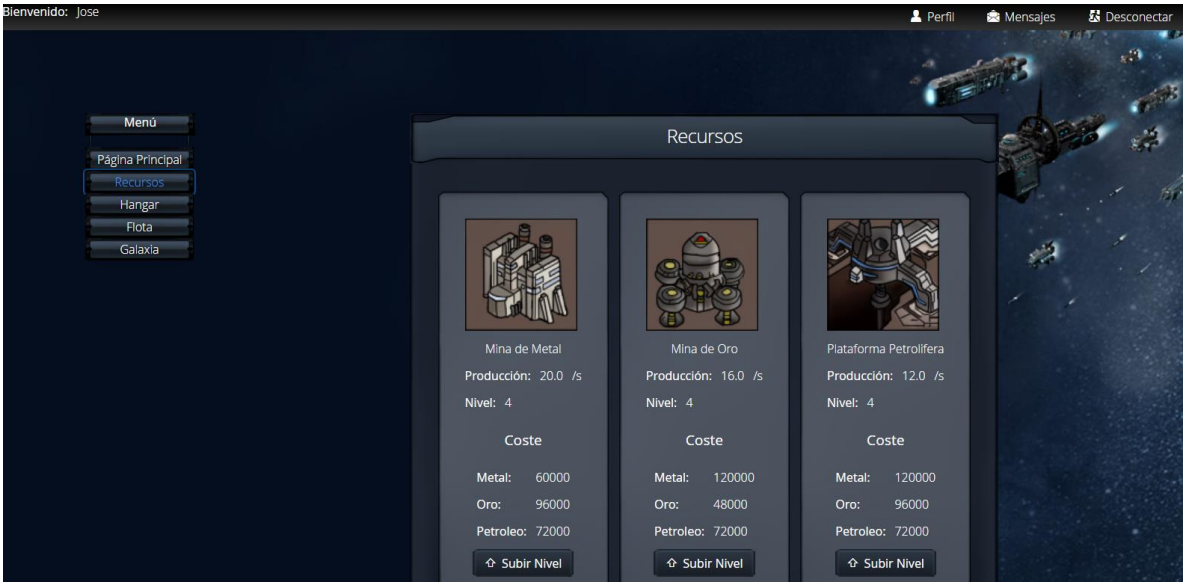

Figura 148. Ventana de recursos. (Elaboración propia)

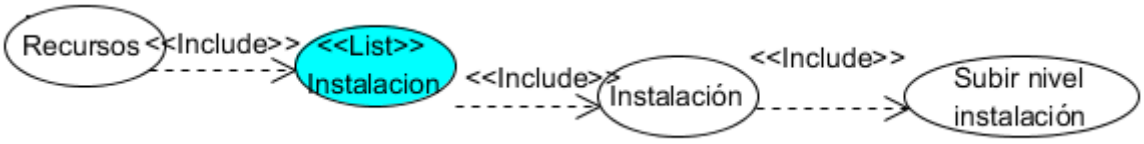

Figura 149. Diagrama casos de uso recursos. (Elaboración propia)

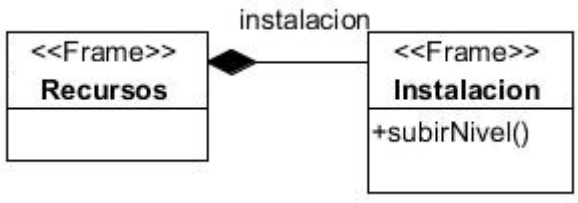

Figura 150. Diagrama de clases recursos. (Elaboración propia)

Hangar

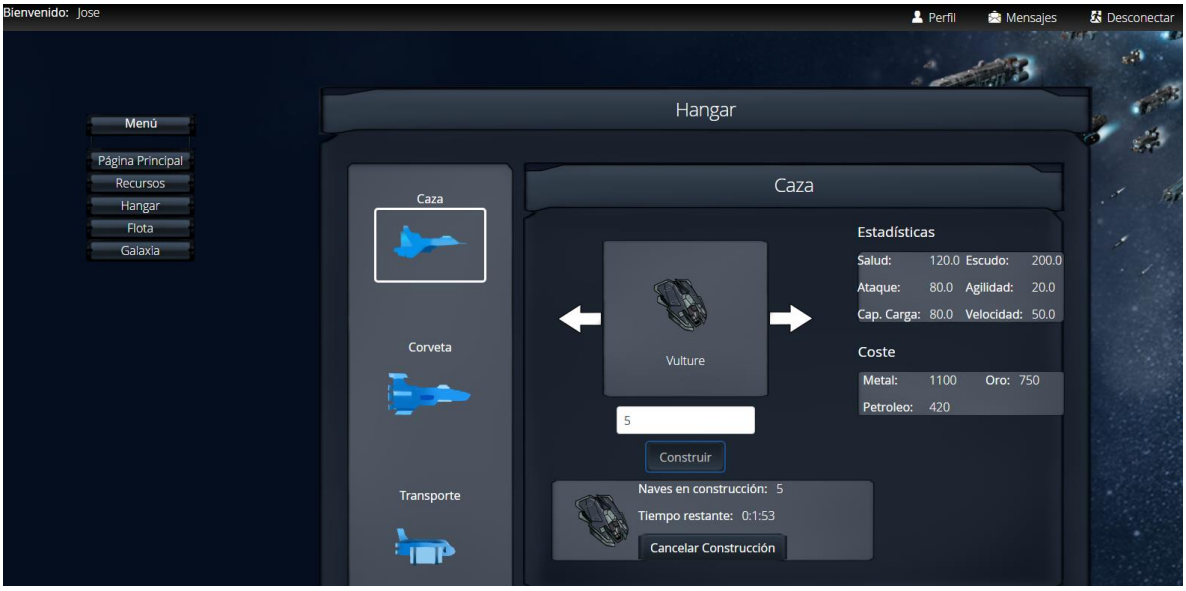

Figura 151. Ventana de hangar. (Elaboración propia)

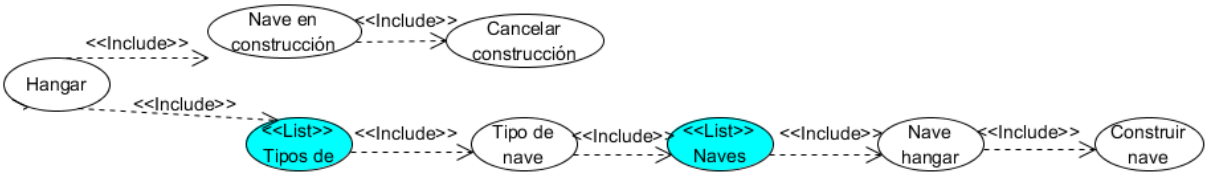

Figura 152. Diagrama casos de uso hangar. (Elaboración propia)

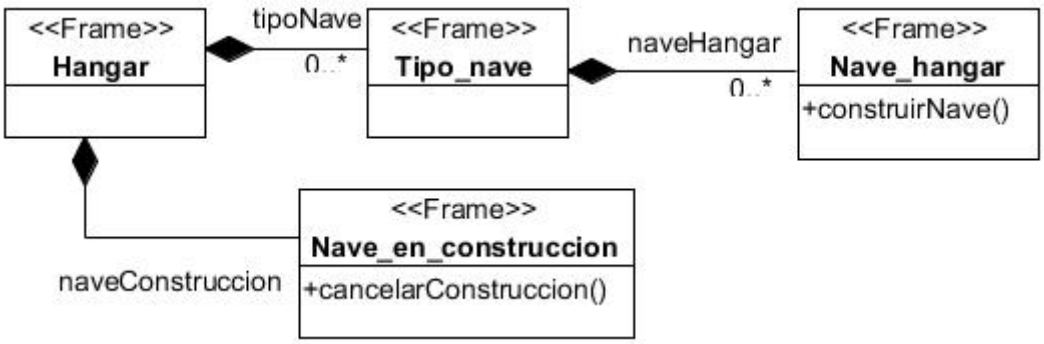

Figura 153. Diagrama de clases hangar. (Elaboración propia)

Tipos de nave

Figura 154. Tipos de nave de la ventana de hangar. (Elaboración propia)

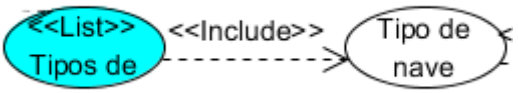

Figura 155. Diagrama casos de uso tipos de nave. (Elaboración propia)

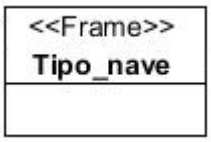

Figura 156. Diagrama de clases tipos de nave. (Elaboración propia)

Naves

Figura 157. Naves de la ventana hangar. (Elaboración propia)

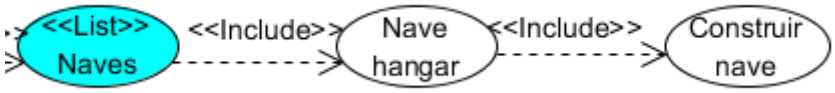

Figura 158. Diagrama casos de uso naves del hangar. (Elaboración propia)

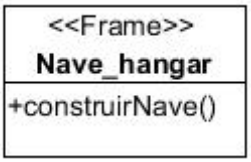

Figura 159. Diagrama de clases naves del hangar. (Elaboración propia)

Cancelar construcción

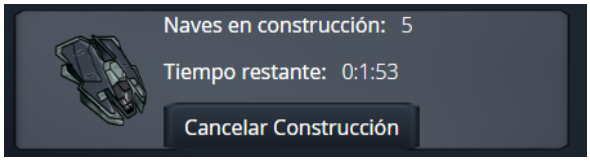

Figura 160. Cancelación de la construcción de la ventana hangar. (Elaboración propia)

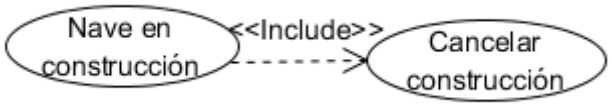

Figura 161. Diagrama casos de uso cancelar construcción. (Elaboración propia)

Figura 162. Diagrama de clases cancelar construcción. (Elaboración propia)

Ventana de flota

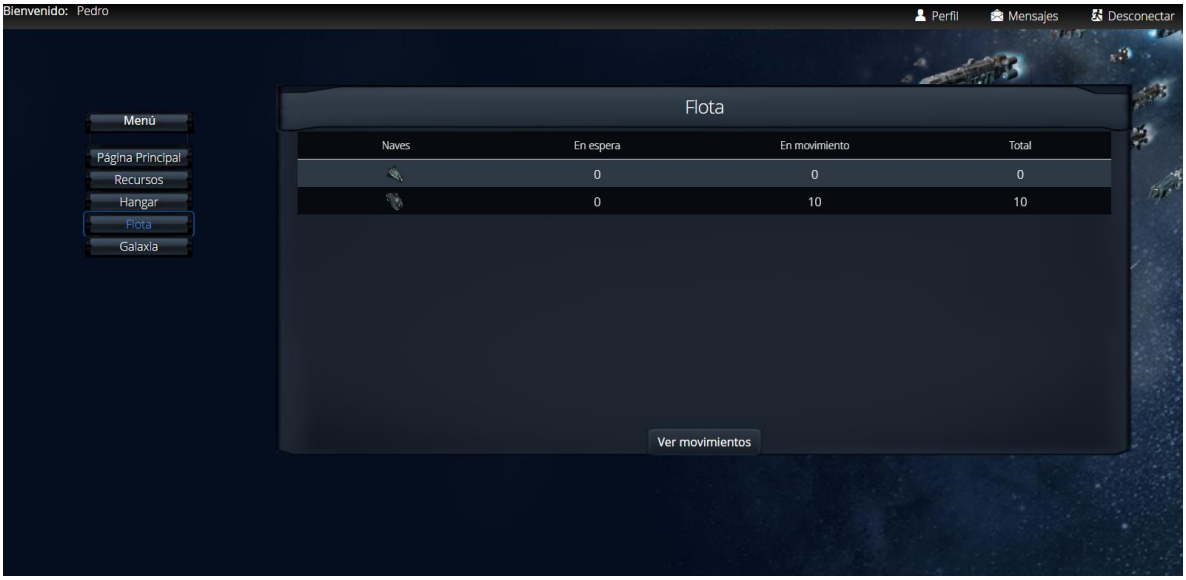

Figura 163. Ventana de flota. (Elaboración propia)

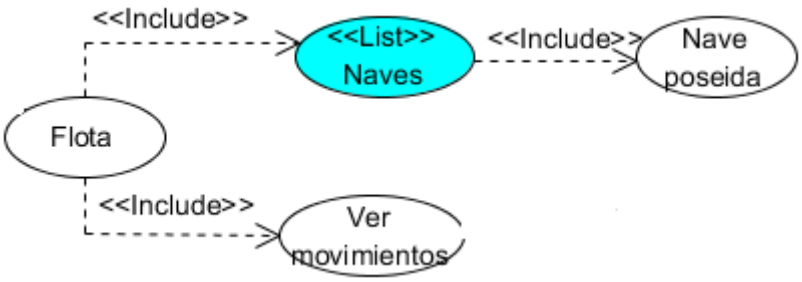

Figura 164. Diagrama casos de uso flota. (Elaboración propia)

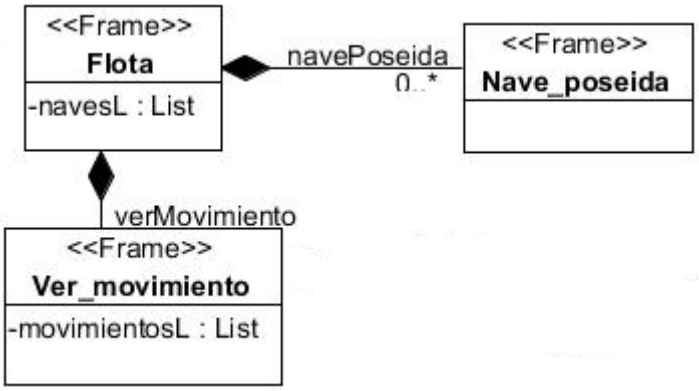

Figura 165. Diagrama de clases flota. (Elaboración propia)

Ventana de movimientos

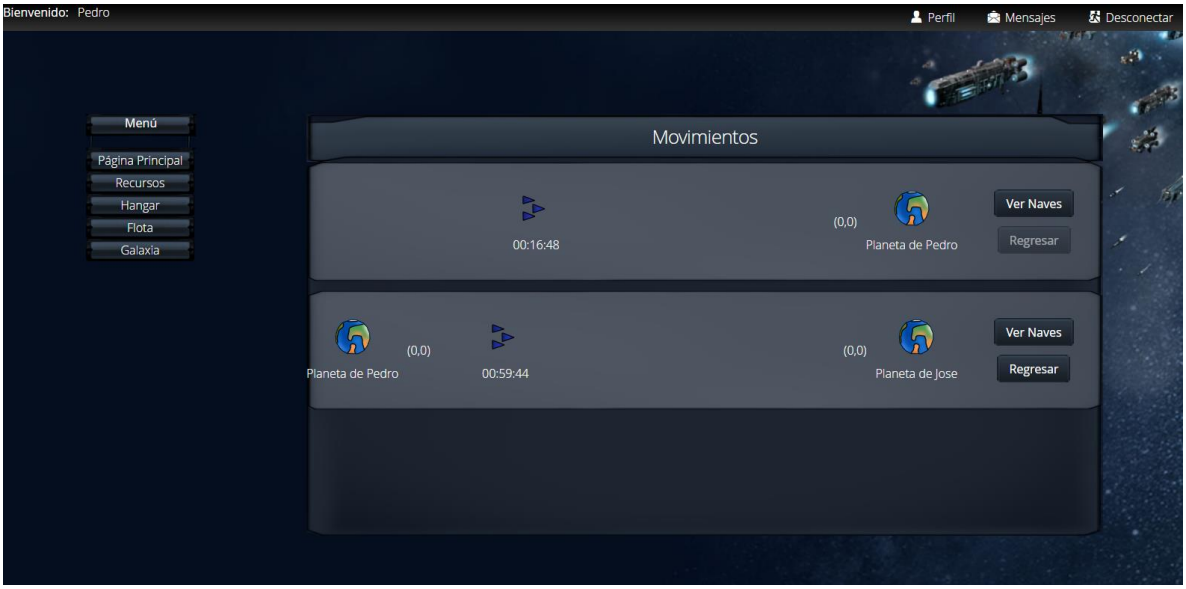

Figura 166. Ventana de movimientos. (Elaboración propia)

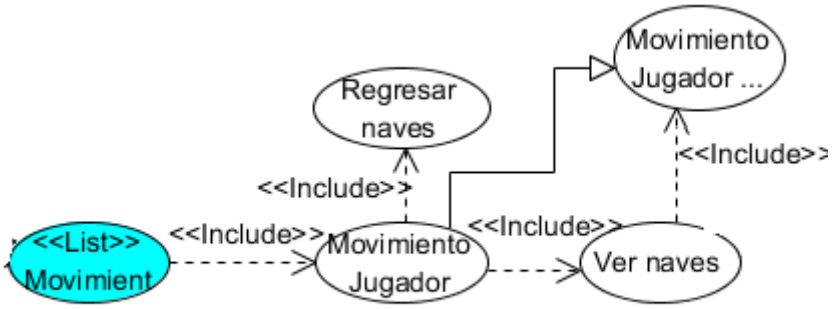

Figura 167. Diagrama casos de uso movimientos. (Elaboración propia)

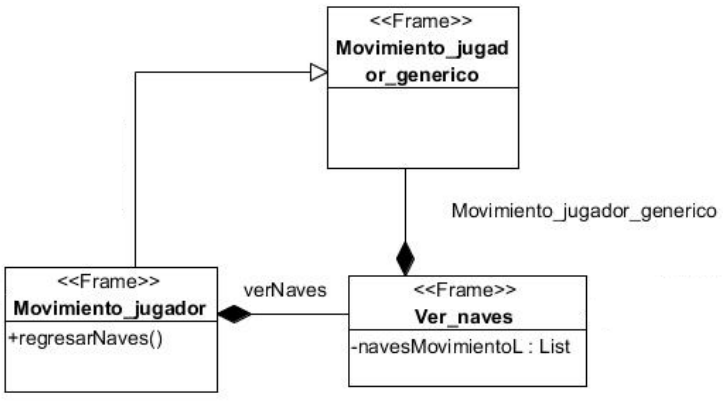

Figura 168. Diagrama de clases movimientos. (Elaboración propia)

Ventana de movimiento

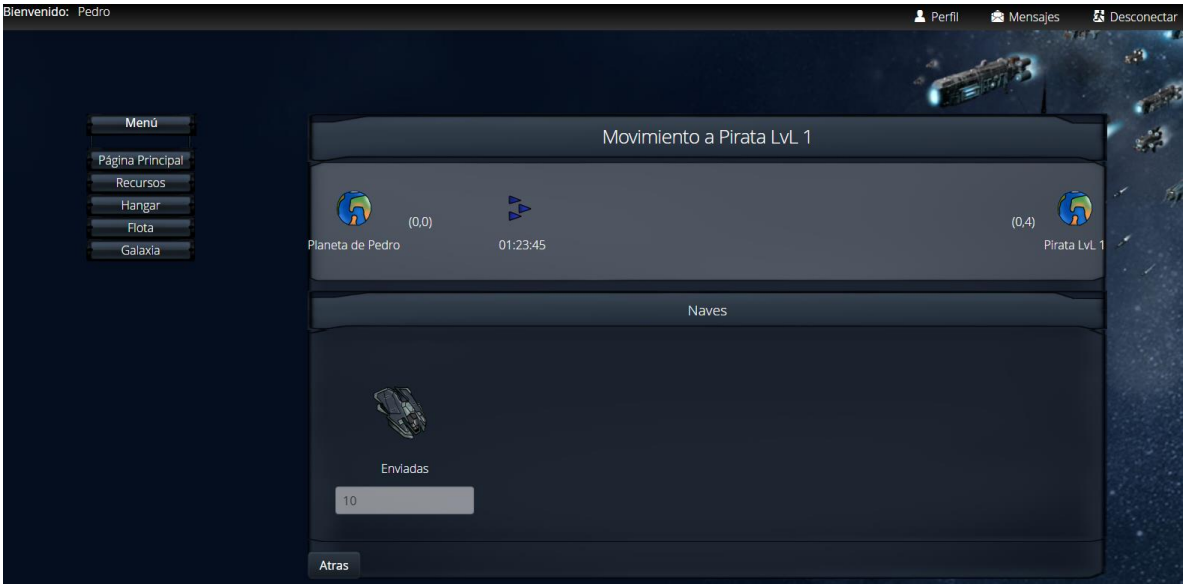

Figura 169. Ventana de movimiento. (Elaboración propia)

Figura 170. Diagrama casos de uso movimiento. (Elaboración propia)

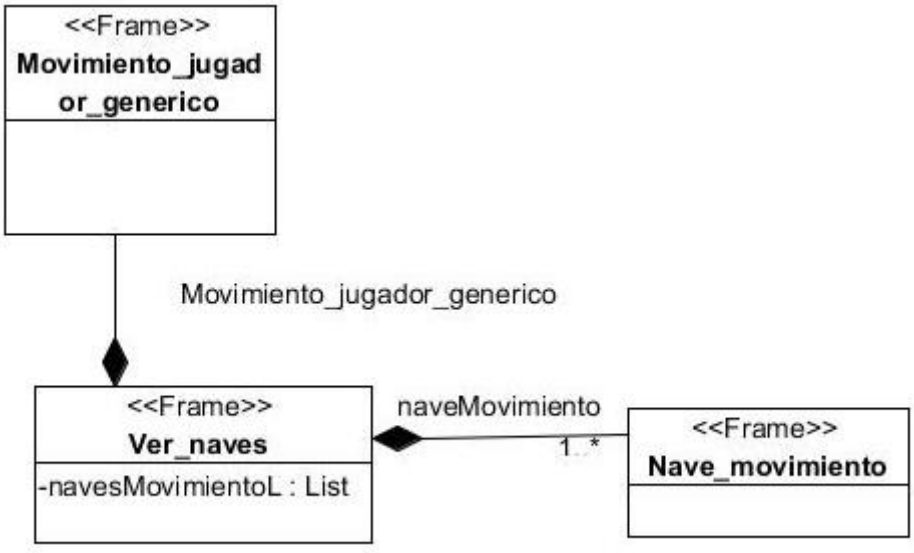

Figura 171. Diagrama de clases movimiento. (Elaboración propia)

Listado de planetas (Galaxia)

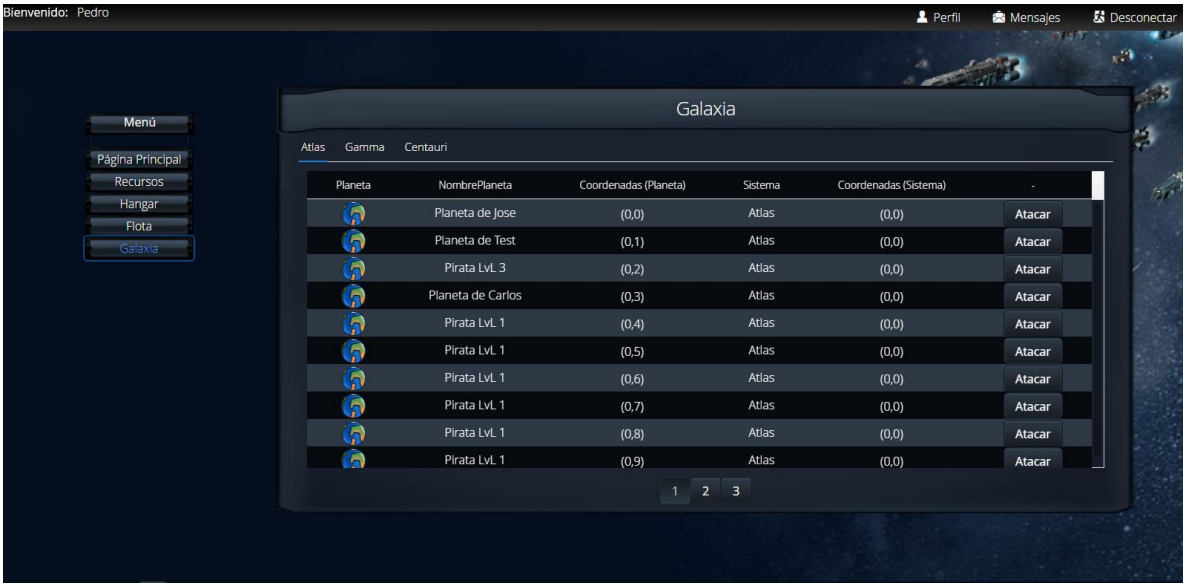

Figura 172. Ventana de galaxia. (Elaboración propia)

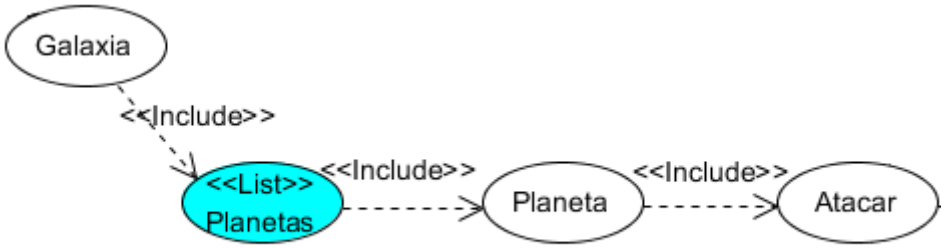

Figura 173. Diagrama casos de uso galaxia. (Elaboración propia)

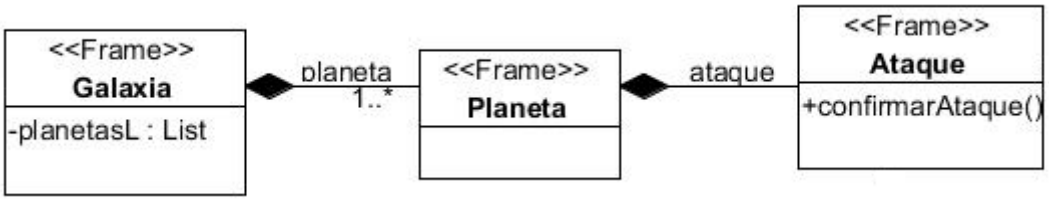

Figura 174. Diagrama de clases galaxia. (Elaboración propia)

Ventana de ataque

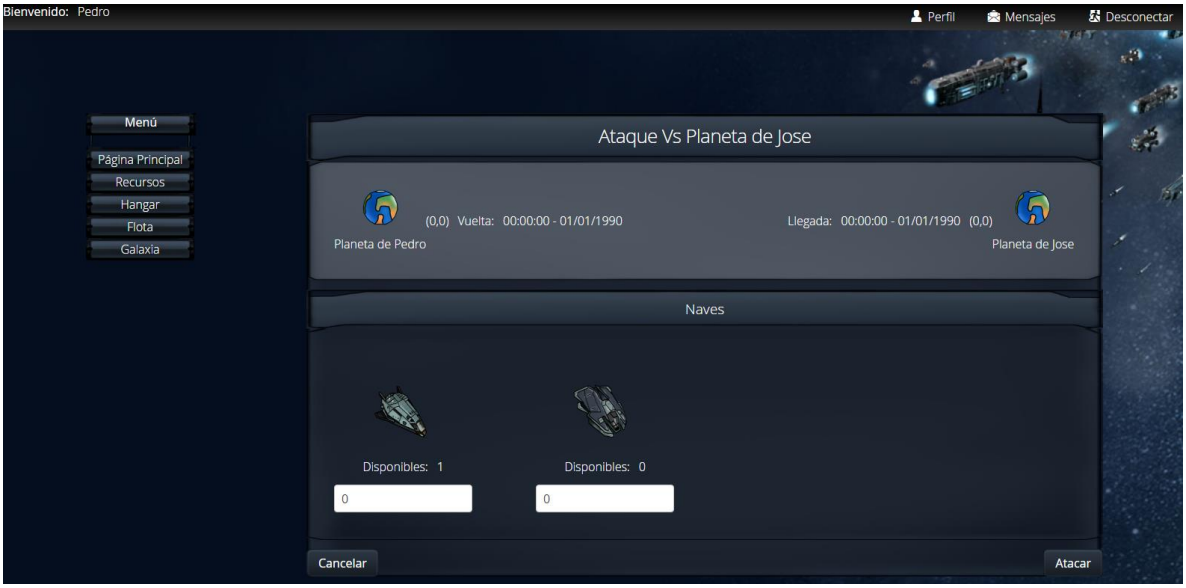

Figura 175. Ventana de ataque. (Elaboración propia)

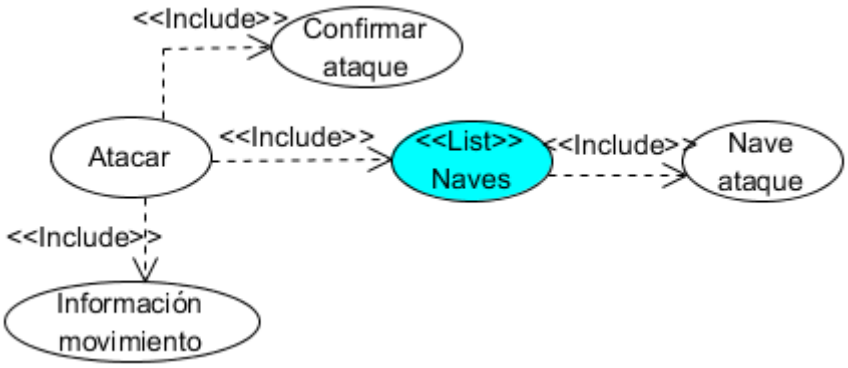

Figura 176. Diagrama casos de uso ataque. (Elaboración propia)

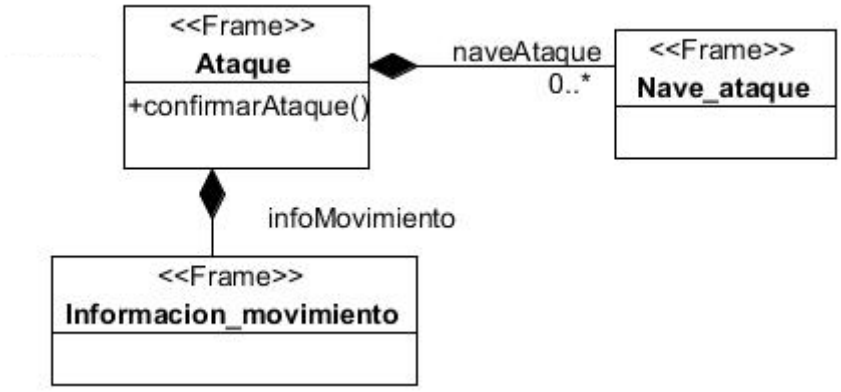

Figura 177. Diagrama de clases ataque. (Elaboración propia)

Perfil

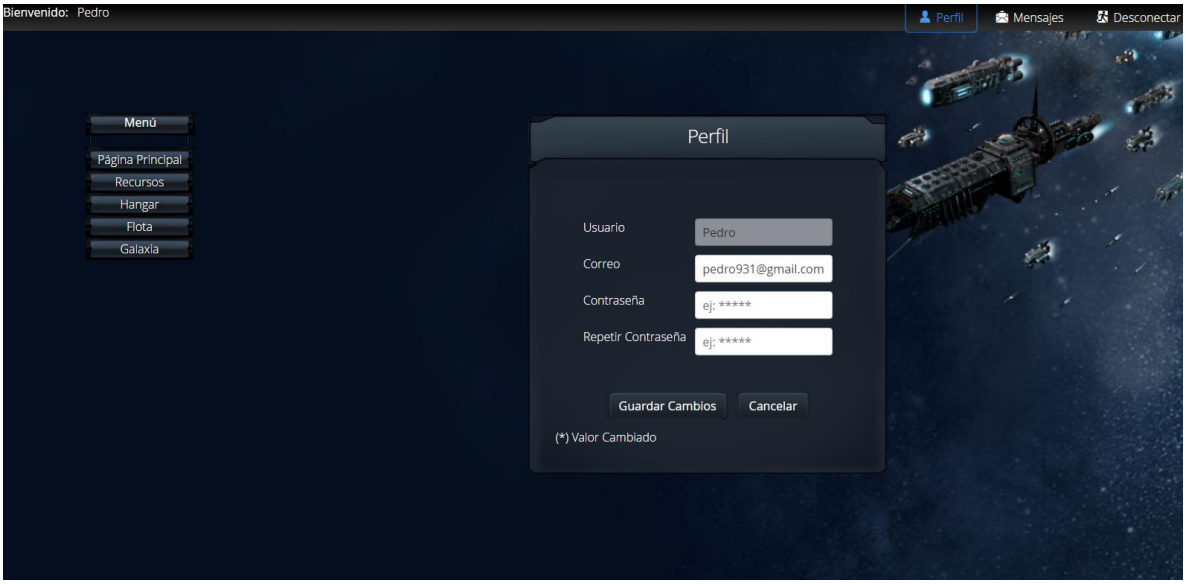

Figura 178. Ventana de perfil. (Elaboración propia)

Figura 179. Diagrama casos de uso perfil. (Elaboración propia)

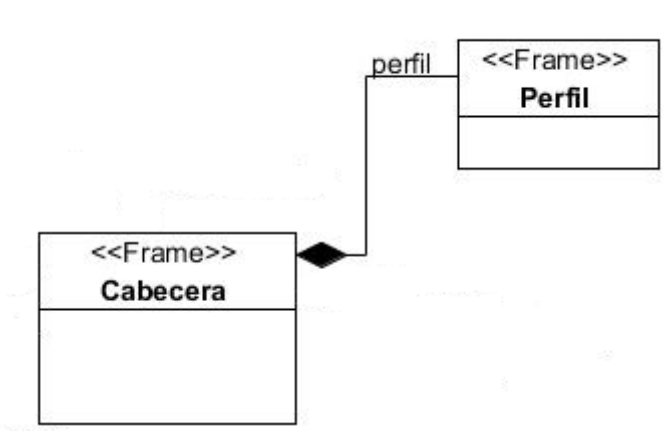

Figura 180. Diagrama de clases perfil. (Elaboración propia)

Listado de mensajes

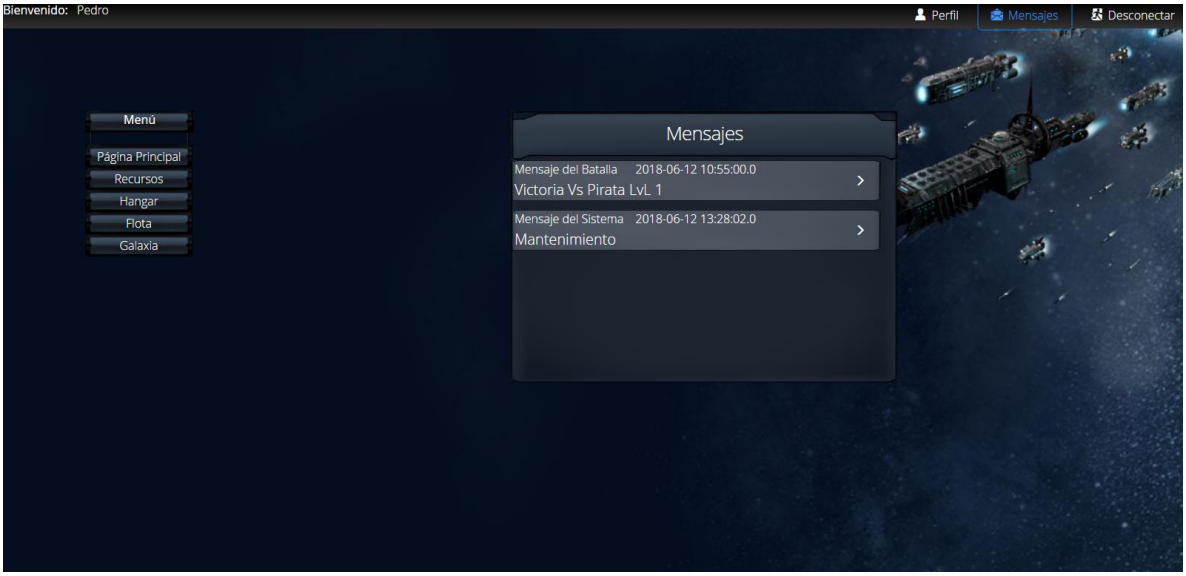

Figura 181. Ventana de mensajes. (Elaboración propia)

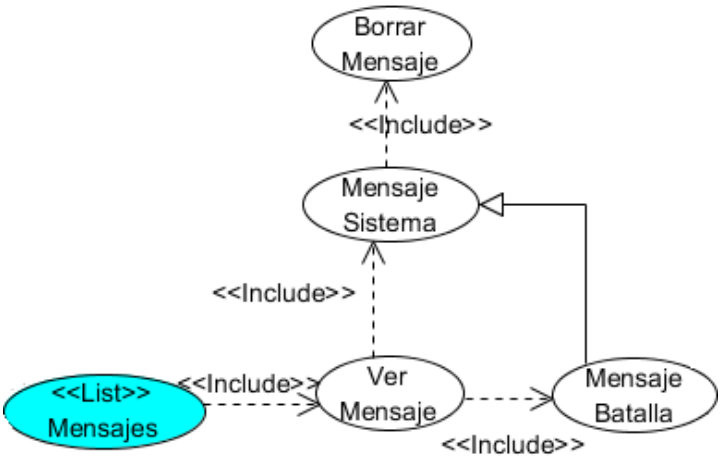

Figura 182. Diagrama casos de uso mensajes. (Elaboración propia)

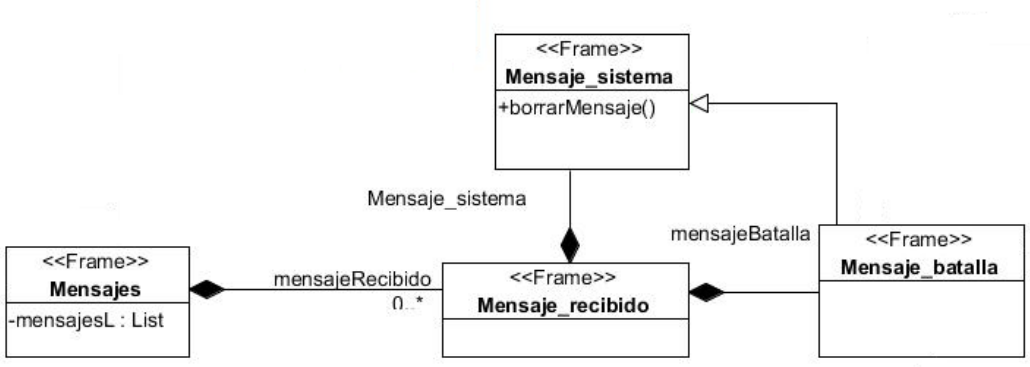

Figura 183. Diagrama de clases mensajes. (Elaboración propia)

Naves atacantes de un mensaje de tipo batalla

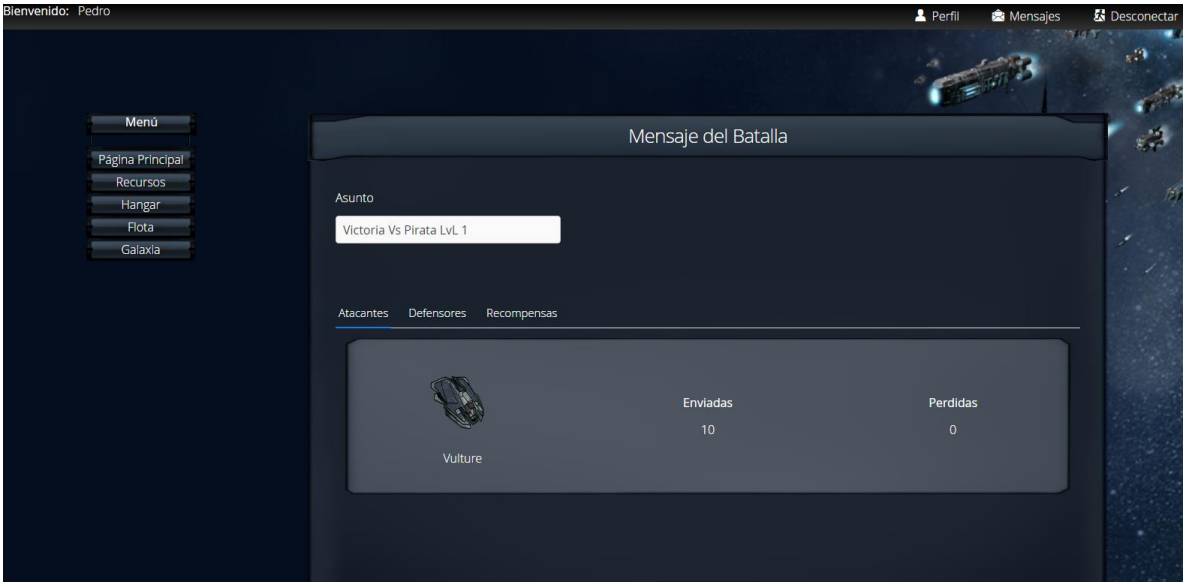

Figura 184. Ventana de mensaje de tipo batalla (atacantes). (Elaboración propia)

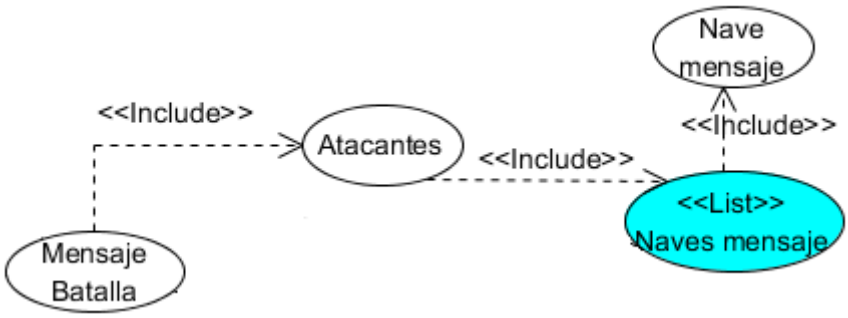

Figura 185. Diagrama casos de uso atacantes. (Elaboración propia)

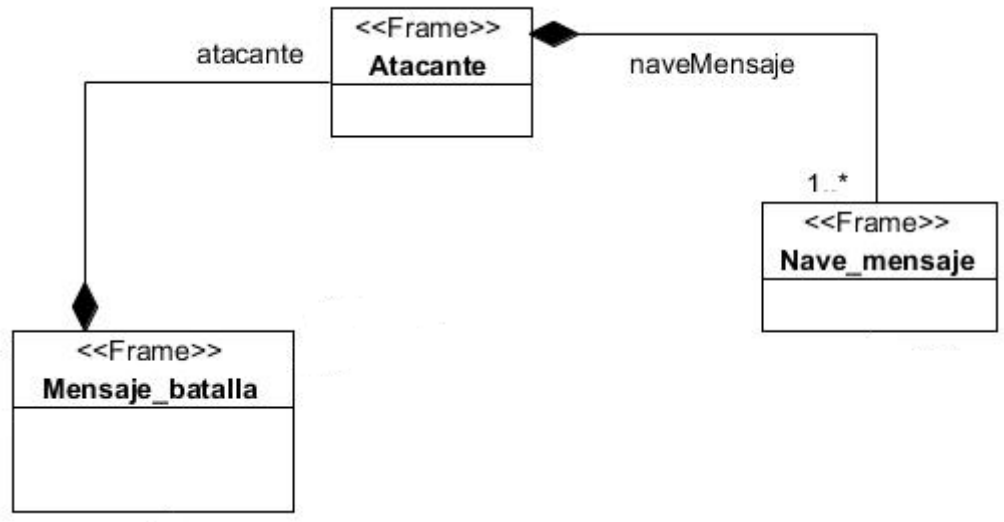

Figura 186. Diagrama de clases atacantes. (Elaboración propia)

Naves defensoras de un mensaje de tipo batalla

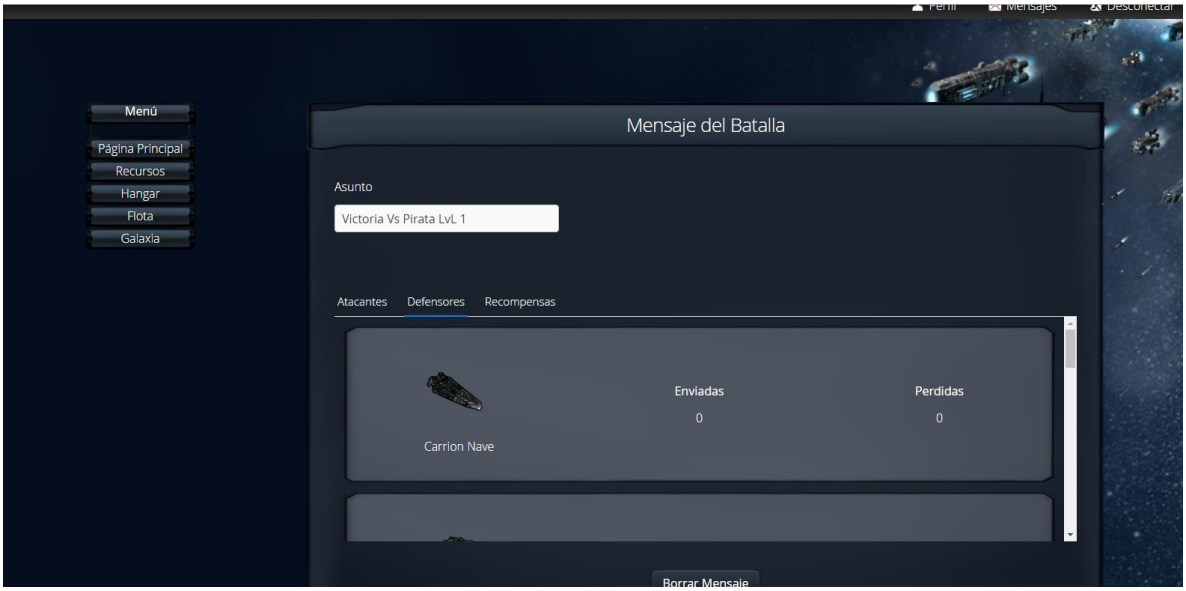

Figura 187. Ventana de mensaje de tipo batalla (defensores). (Elaboración propia)

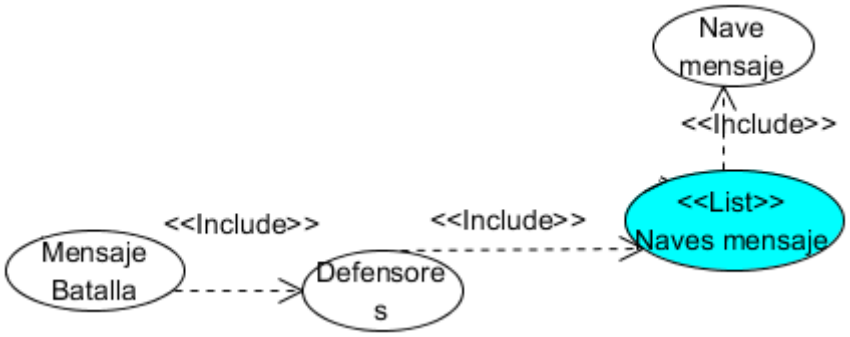

Figura 188. Diagrama casos de uso defensores. (Elaboración propia)

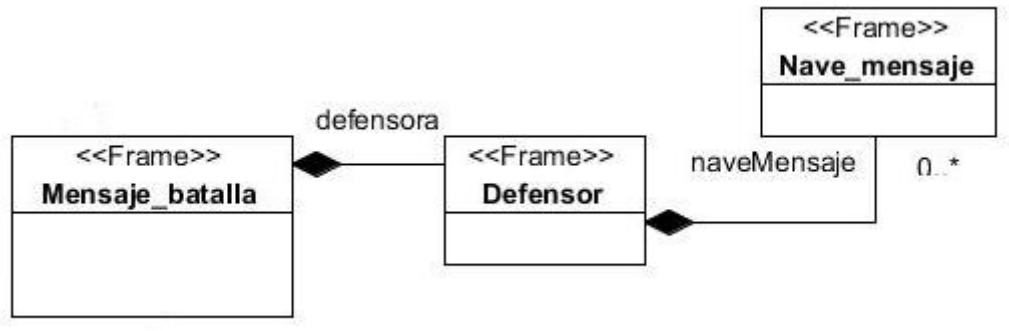

Figura 189. Diagrama de clases defensores. (Elaboración propia)

Recursos obtenidos de un mensaje de tipo batalla

Figura 190. Ventana de mensaje de tipo batalla (recursos). (Elaboración propia)

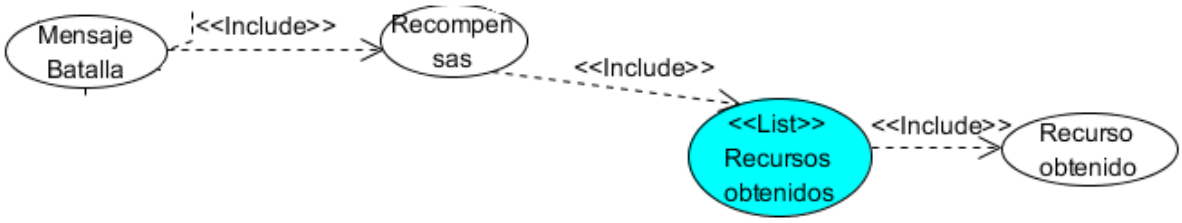

Figura 191. Diagrama casos de uso recursos obtenidos. (Elaboración propia)

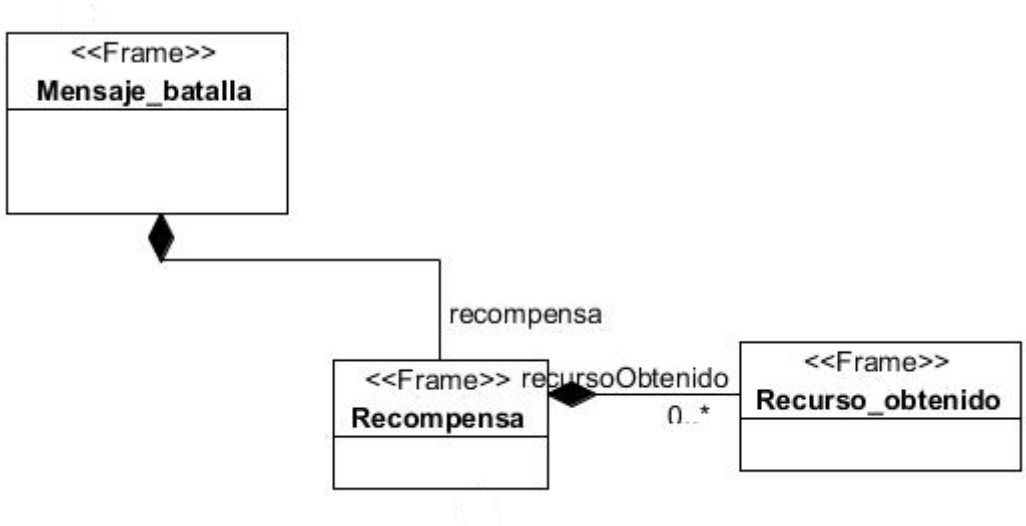

Figura 192. Diagrama de clases recursos obtenidos. (Elaboración propia)

Naves desbloqueadas de un mensaje de tipo batalla

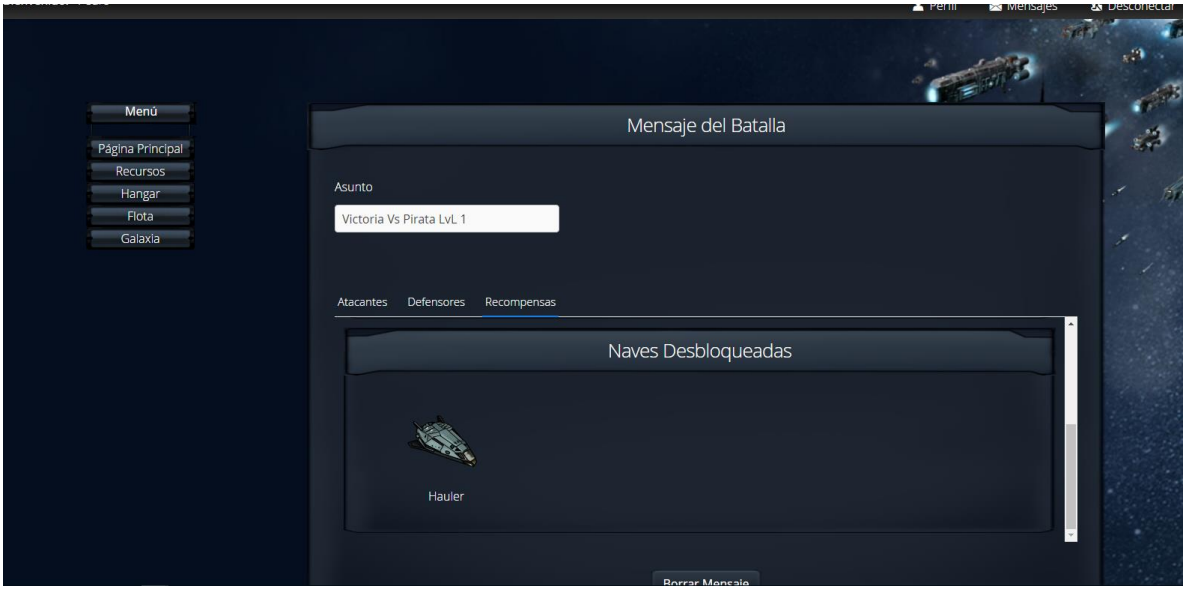

Figura 193. Ventana de mensaje de tipo batalla (naves desbloqueadas). (Elaboración propia)

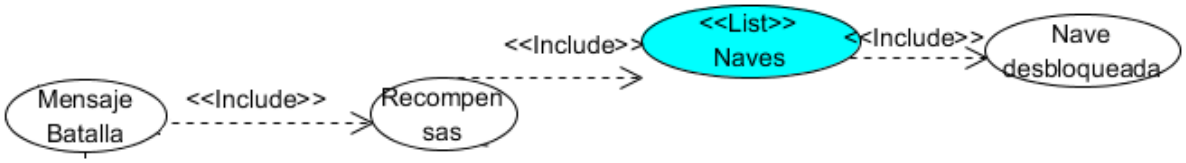

Figura 194. Diagrama casos de uso naves desbloqueadas. (Elaboración propia)

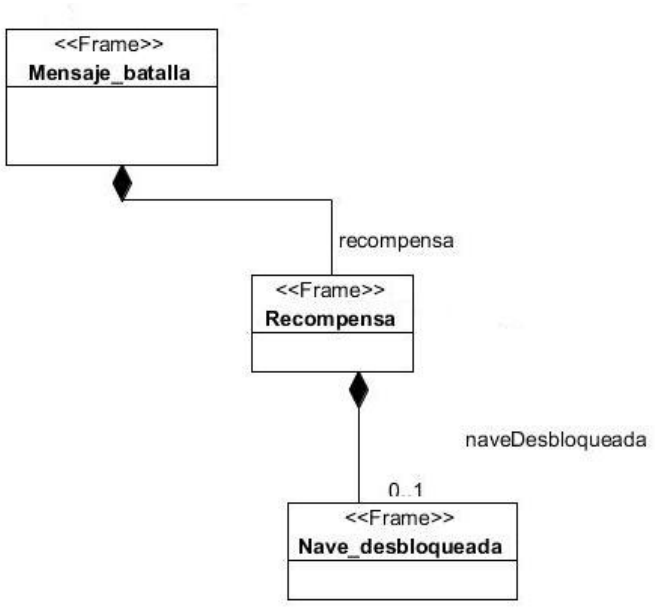

Figura 195. Diagrama de clases naves desbloqueadas. (Elaboración propia)

Mensaje de tipo sistema

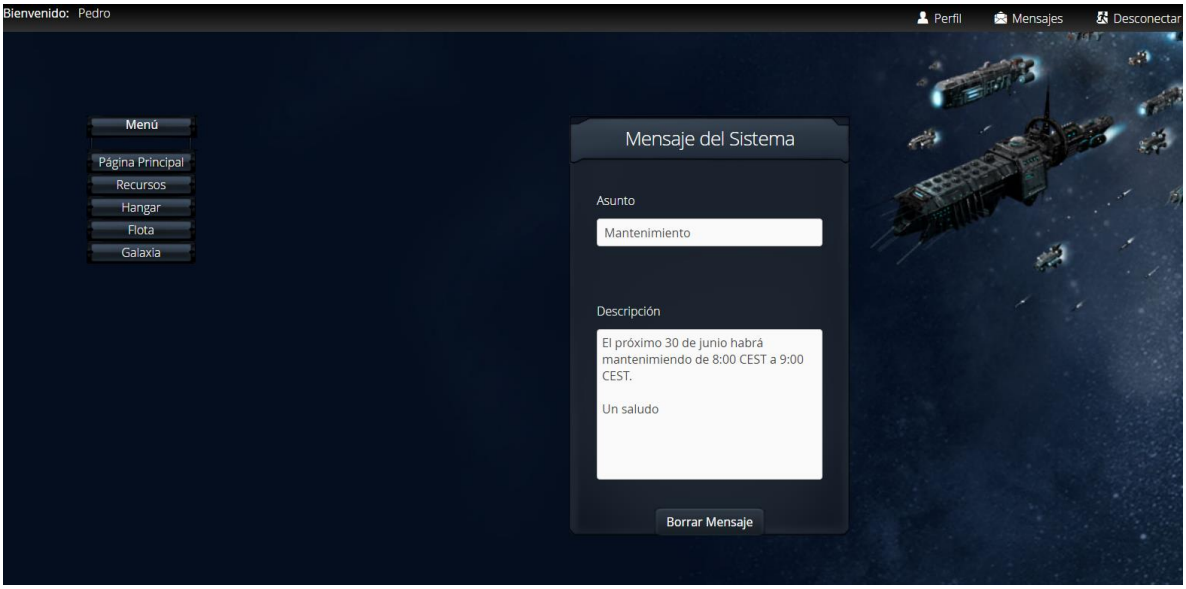

Figura 196. Ventana de mensaje de tipo sistema. (Elaboración propia)

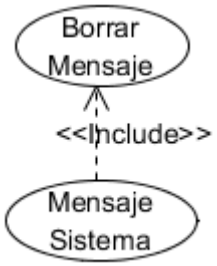

Figura 197. Diagrama casos de uso borrar mensaje. (Elaboración propia)

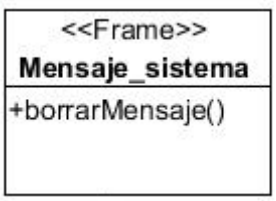

Figura 198. Diagrama de clases borrar mensaje. (Elaboración propia)

3.3.7.1.2.3. Administrador

A continuación se mostrarán las ternas de trazabilidad para el actor administrador.

Pantalla principal (Administrador)

Figura 199. Ventana de la pantalla principal (administrador). (Elaboración propia)

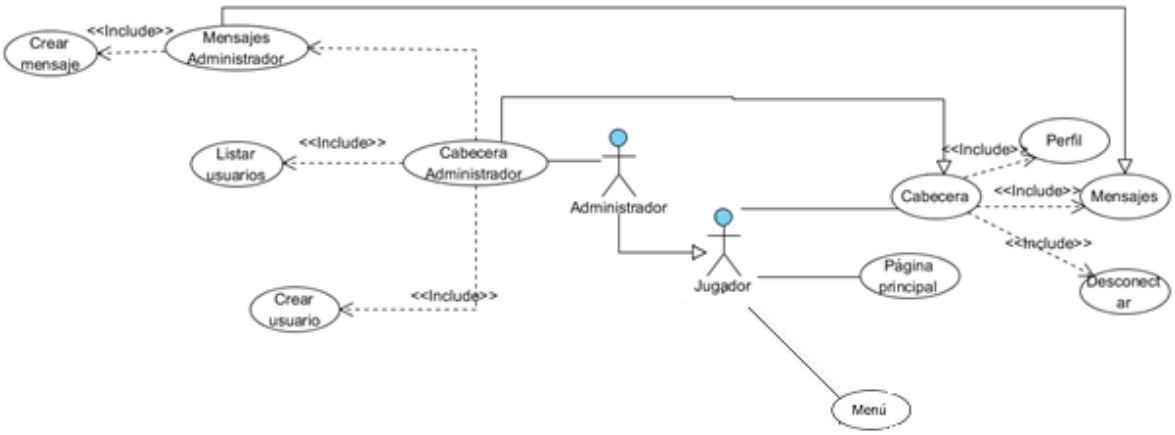

Figura 200. Diagrama casos de uso pantalla principal (Administrador). (Elaboración propia)

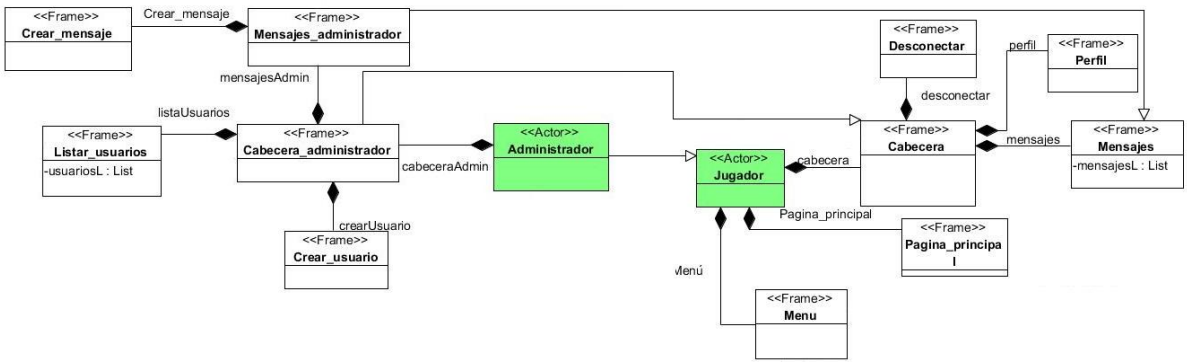

Figura 201. Diagrama de clases pantalla principal (Administrador). (Elaboración propia)

Página principal

Figura 202. Página principal de la ventana pantalla principal. (Elaboración propia)

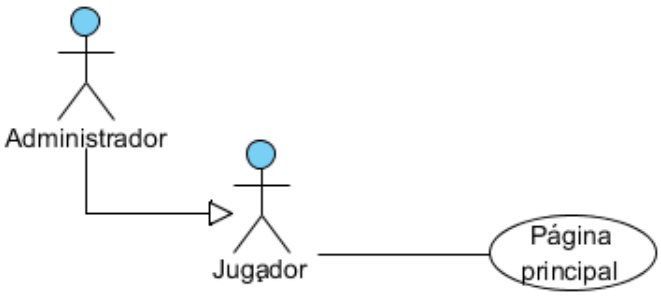

Figura 203. Diagrama casos de uso página principal (Administrador). (Elaboración propia)

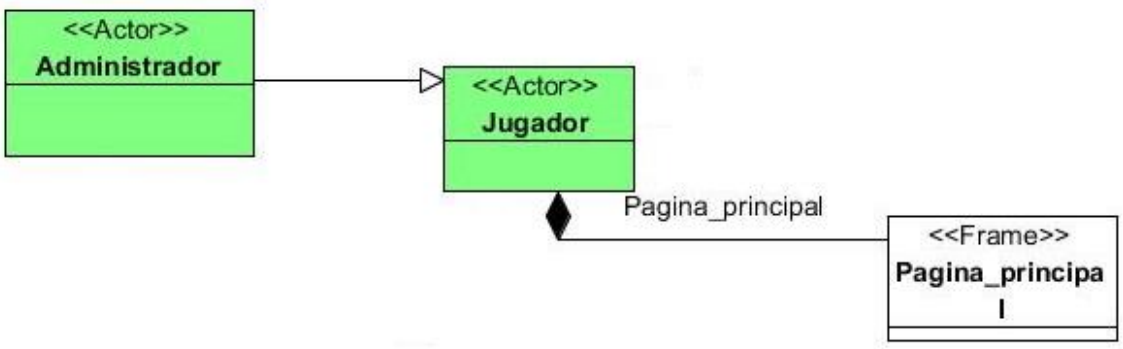

Figura 204. Diagrama de clases página principal (Administrador). (Elaboración propia)

Cabecera (Administrador)

Figura 205. Cabecera del administrador. (Elaboración propia)

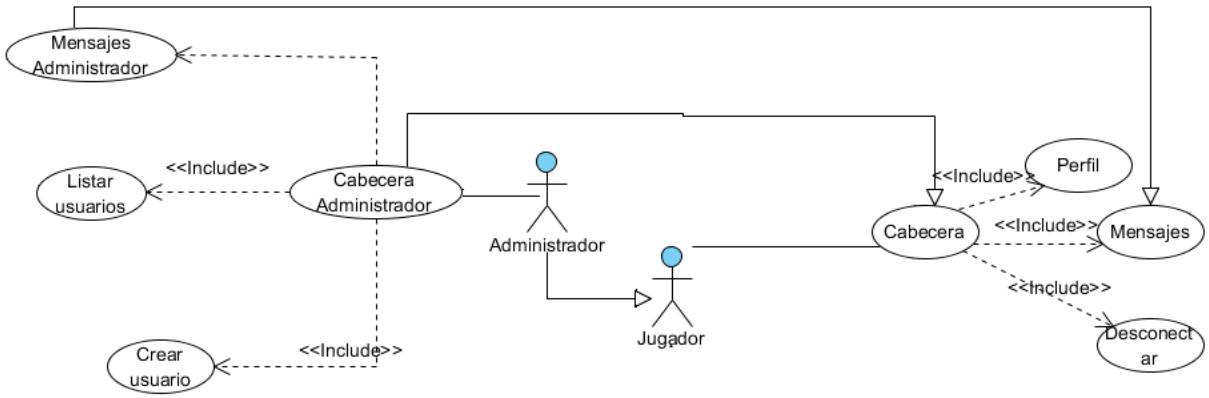

Figura 206. Diagrama casos de uso cabecera (Administrador). (Elaboración propia)

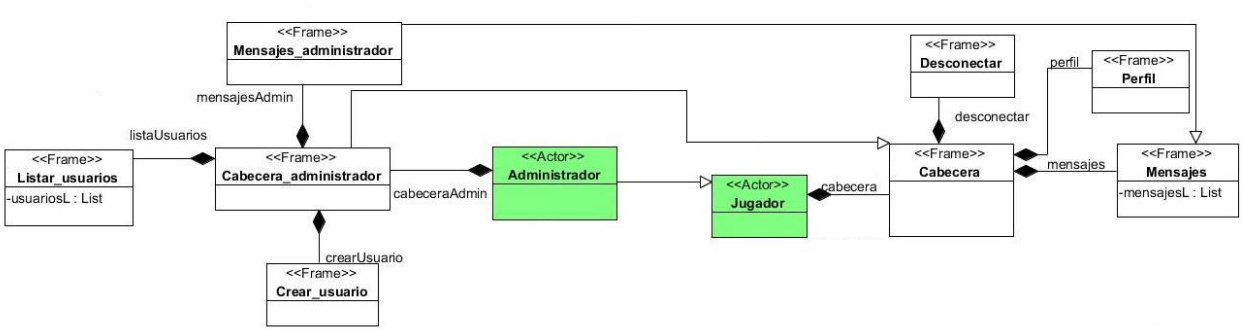

Figura 207. Diagrama de clases cabecera (Administrador). (Elaboración propia)

Menú

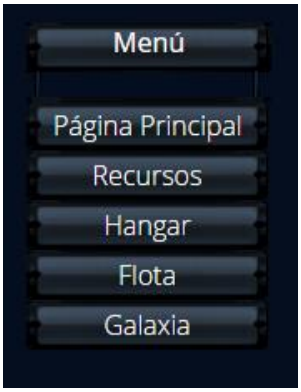

Figura 208. Ventana del menú. (Elaboración propia)

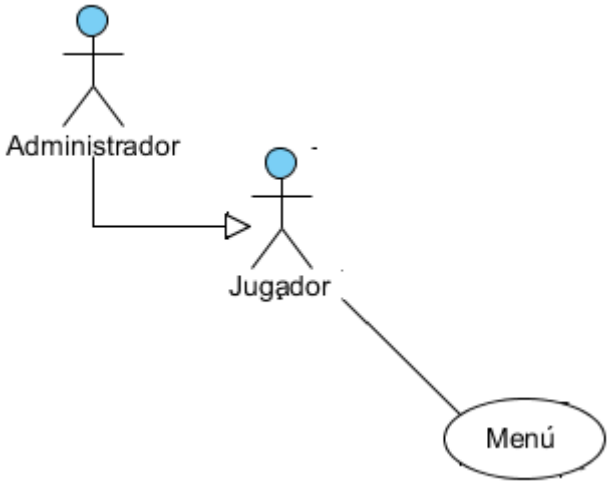

Figura 209. Diagrama casos de uso menú (Administrador). (Elaboración propia)

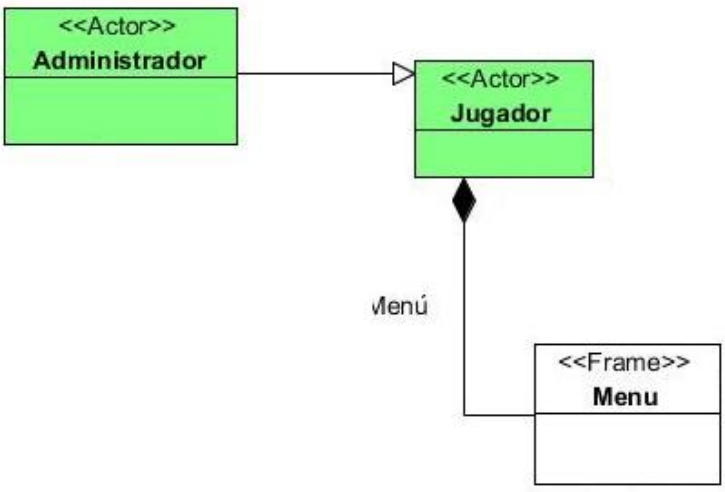

Figura 210. Diagrama de clases menú (Administrador). (Elaboración propia)

Listado de usuarios

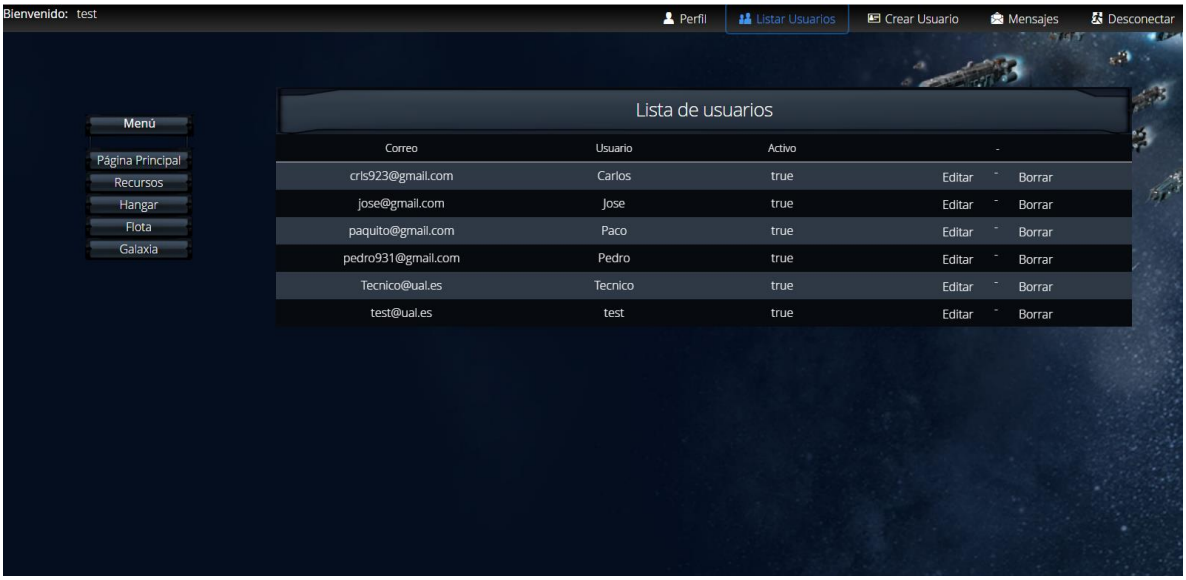

Figura 211. Ventana de listado de usuarios. (Elaboración propia)

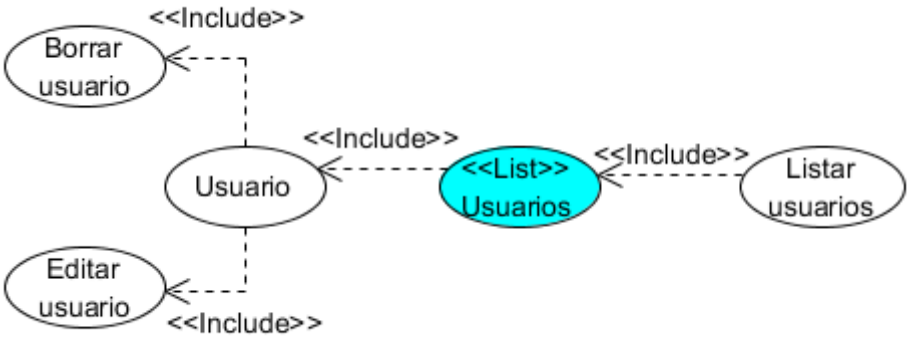

Figura 212. Diagrama casos de uso listar usuarios. (Elaboración propia)

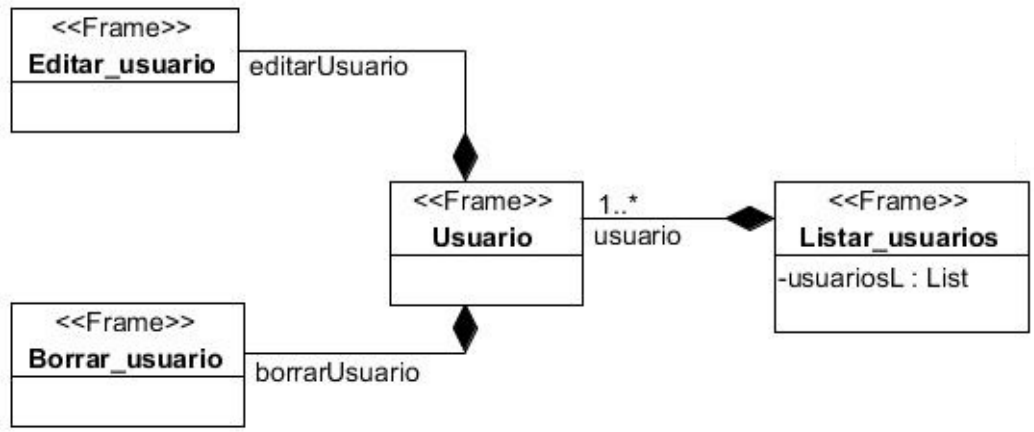

Figura 213. Diagrama de clases listar usuarios. (Elaboración propia)

Editar usuario

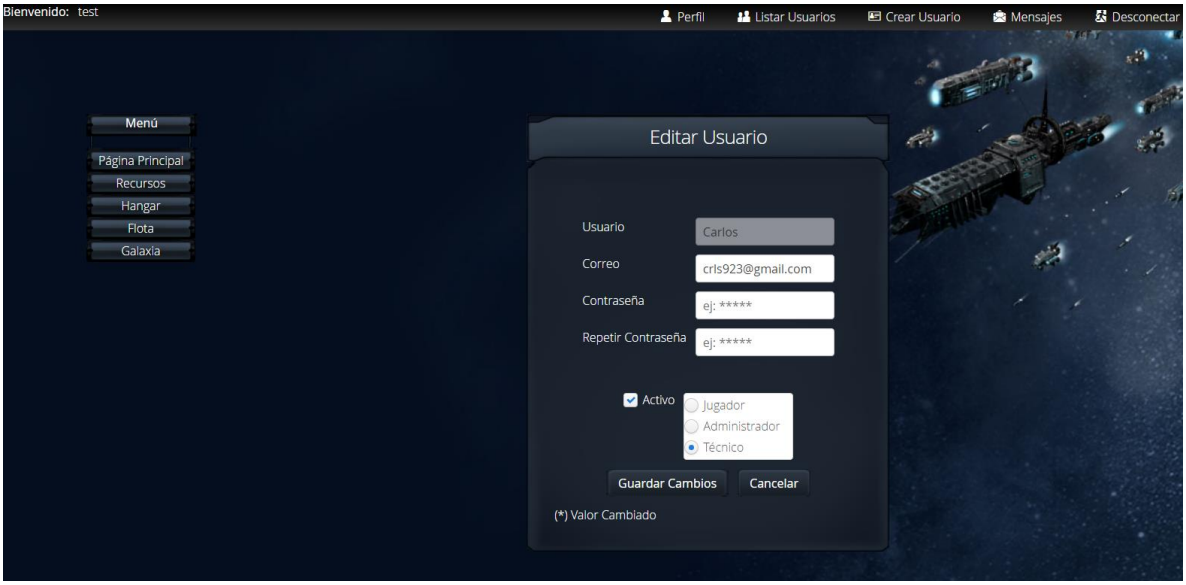

Figura 214. Ventana de editar usuario. (Elaboración propia)

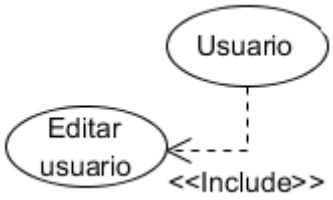

Figura 215. Diagrama casos de uso editar usuario. (Elaboración propia)

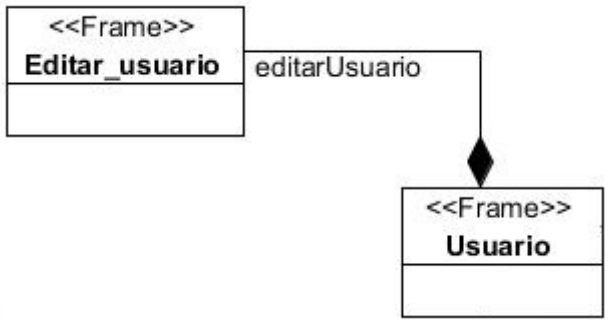

Figura 216. Diagrama de clases editar usuario. (Elaboración propia)

Borrar usuario

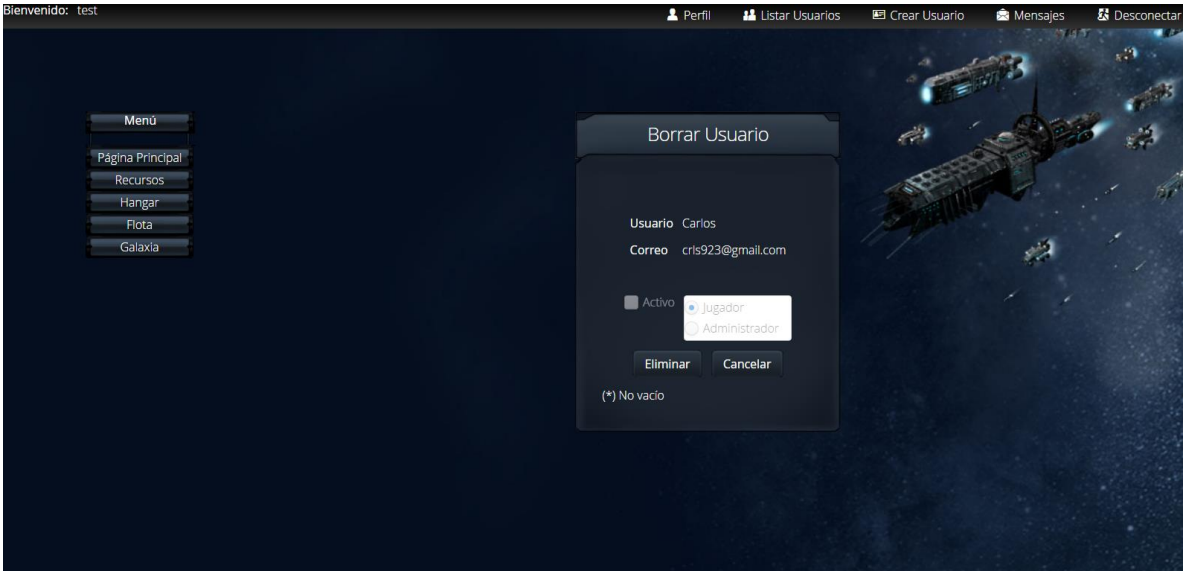

Figura 217. Ventana de borrar usuario. (Elaboración propia)

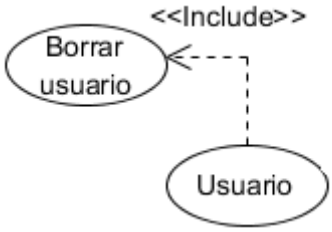

Figura 218. Diagrama casos de uso borrar usuario. (Elaboración propia)

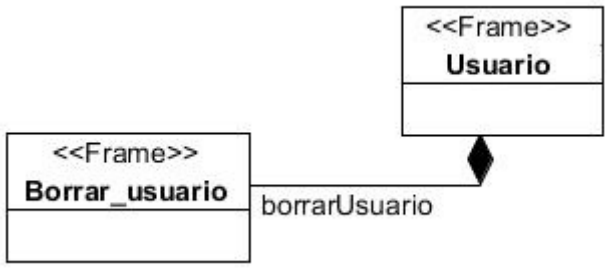

Figura 219. Diagrama de clases borrar usuario. (Elaboración propia)

Crear usuario

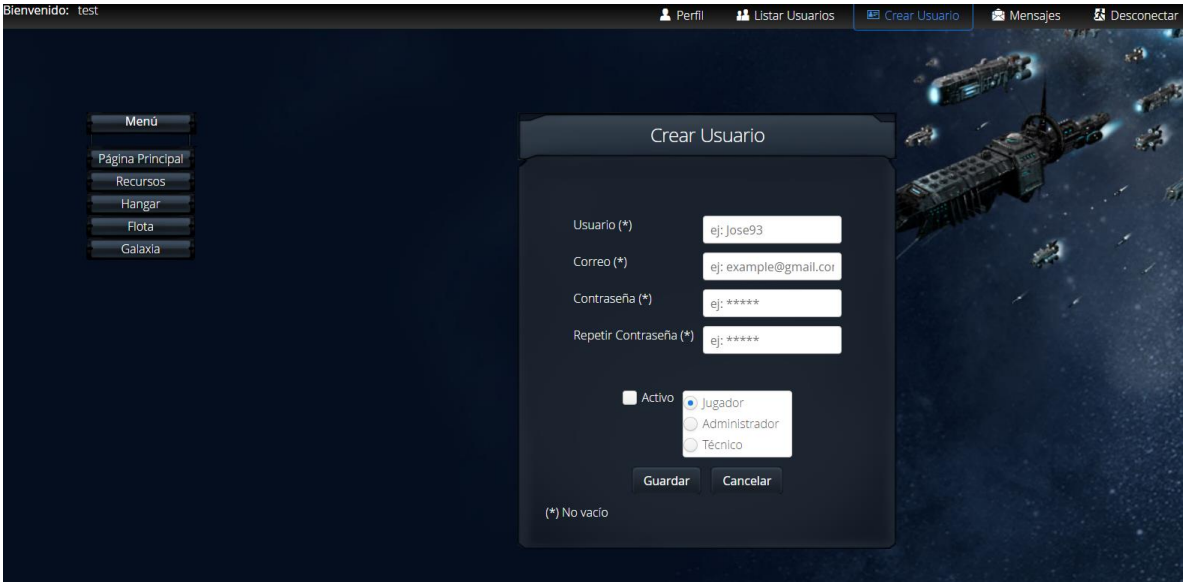

Figura 220. Ventana de crear usuario. (Elaboración propia)

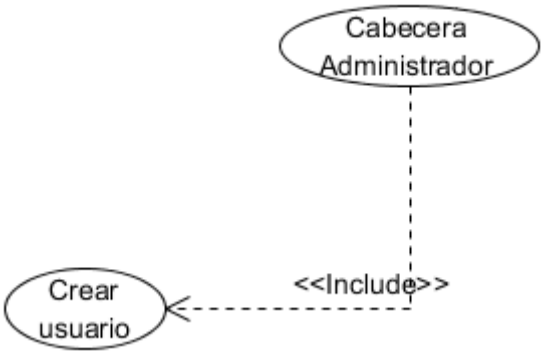

Figura 221. Diagrama casos de uso crear usuario. (Elaboración propia)

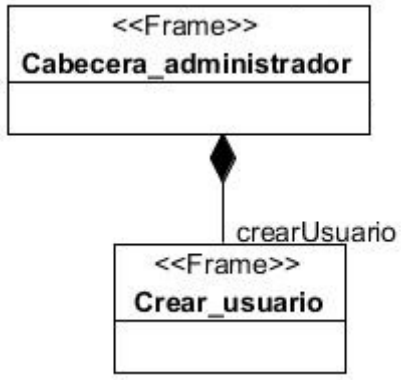

Figura 222. Diagrama de clases crear usuario. (Elaboración propia)

Crear mensaje de tipo sistema

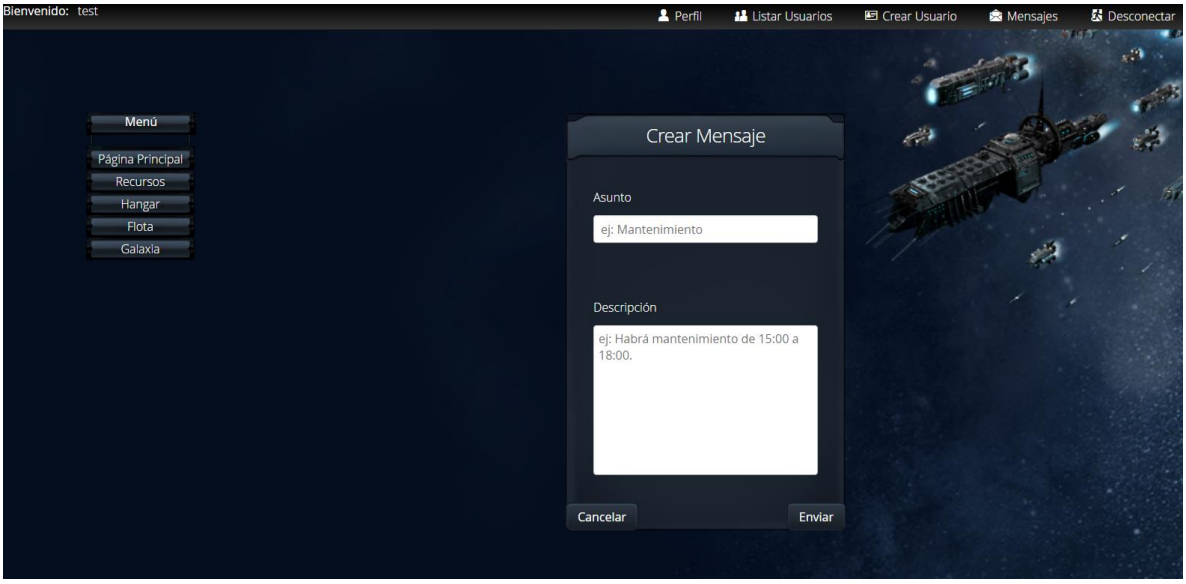

Figura 223. Ventana de crear mensaje de tipo sistema. (Elaboración propia)

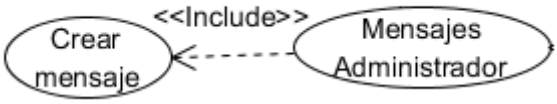

Figura 224. Diagrama casos de crear mensaje. (Elaboración propia)

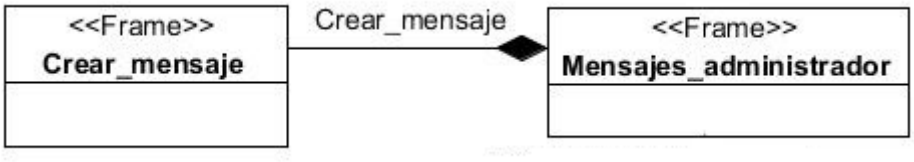

Figura 225. Diagrama de clases crear mensaje. (Elaboración propia)

3.3.7.1.2.4. Técnico

A continuación se mostrarán las ternas de trazabilidad para el actor técnico.

Pantalla principal (Técnico)

Figura 226. Ventana de la pantalla principal (técnico). (Elaboración propia)

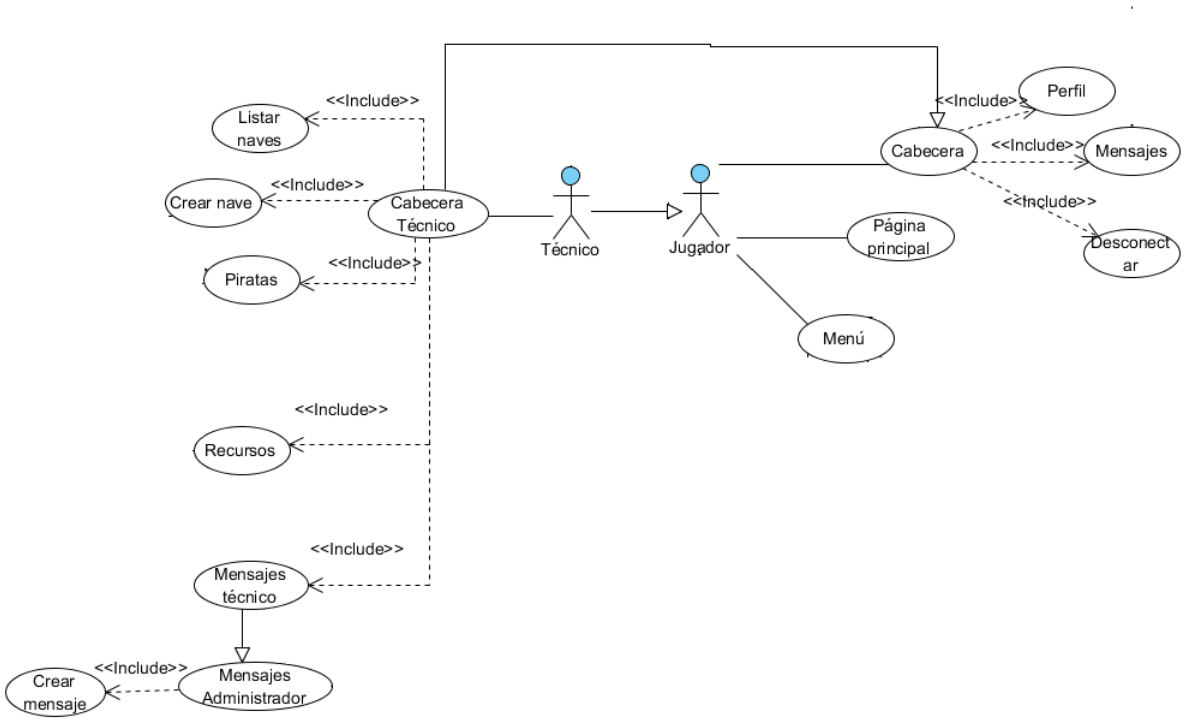

Figura 227. Diagrama casos de uso pantalla principal (Técnico). (Elaboración propia)

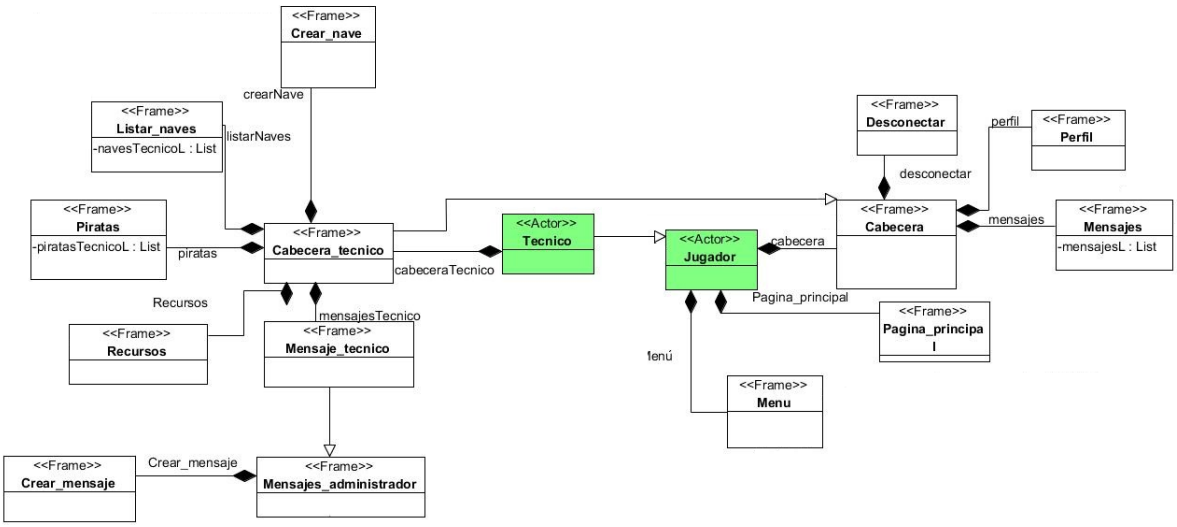

Figura 228. Diagrama de clases pantalla principal (Técnico). (Elaboración propia)

Página principal

Figura 229. Página principal de la ventana pantalla principal. (Elaboración propia

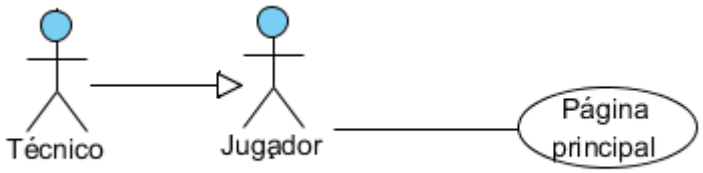

Figura 230. Diagrama casos de uso página principal (Técnico). (Elaboración propia)

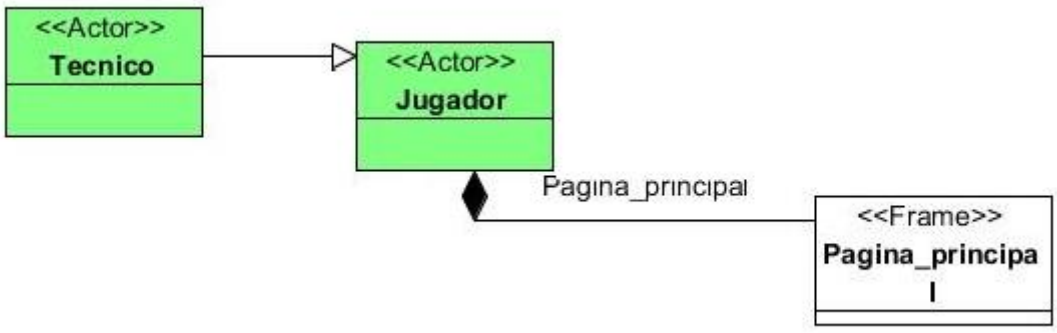

Figura 231. Diagrama de clases página principal (Técnico). (Elaboración propia)

Cabecera (Técnico)

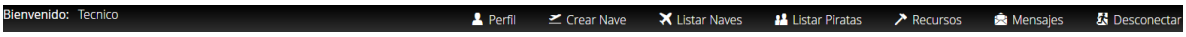

Figura 157. Cabecera del técnico. (Elaboración propia)

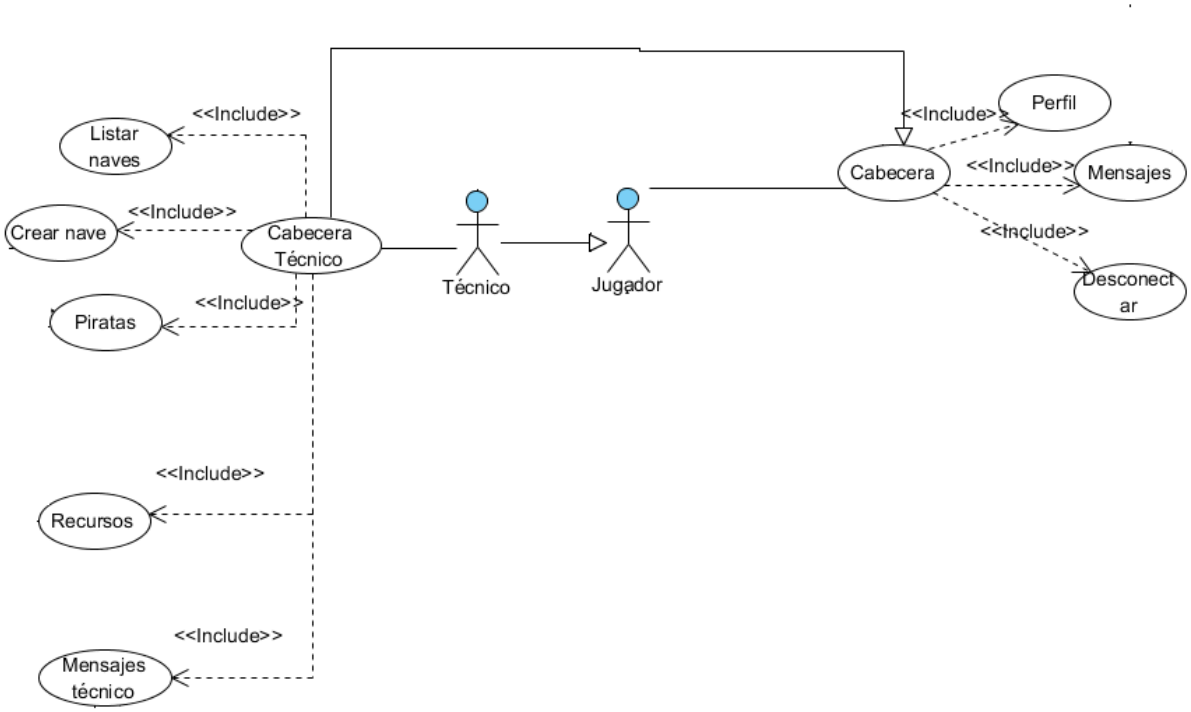

Figura 232. Diagrama casos de uso cabecera (Técnico). (Elaboración propia)

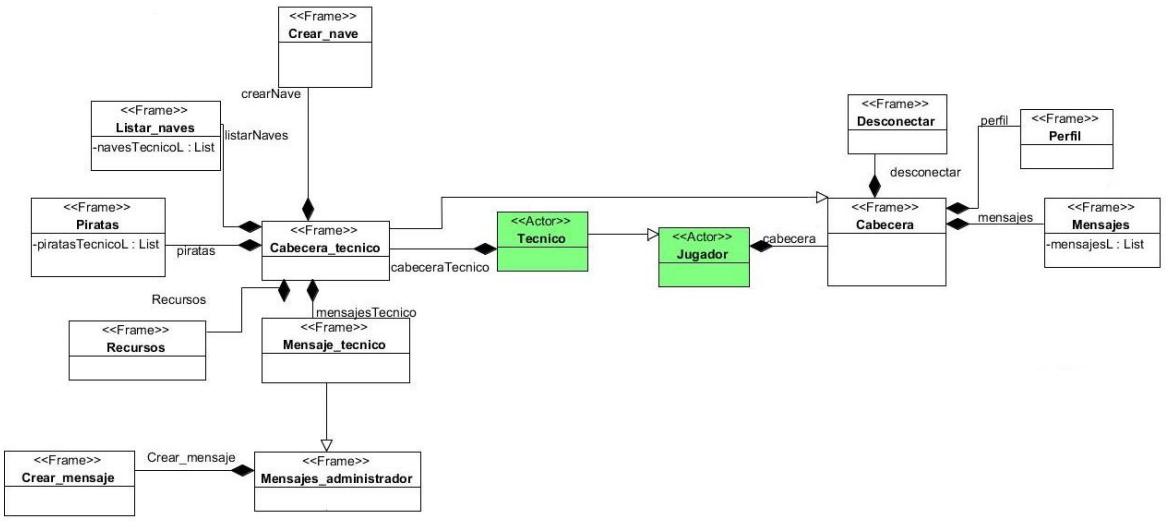

Figura 233. Diagrama de clases cabecera (Técnico). (Elaboración propia)

Menú

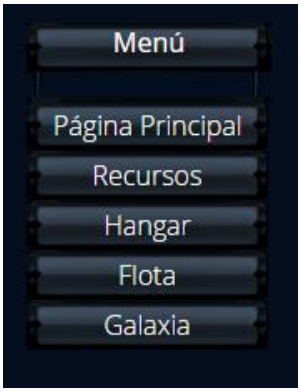

Figura 234. Ventana del menú. (Elaboración propia)

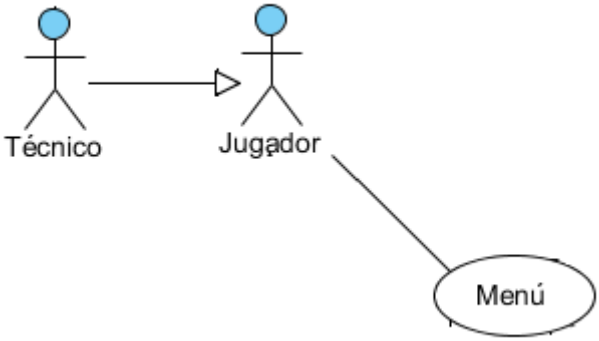

Figura 235. Diagrama casos de uso menú (Técnico). (Elaboración propia)

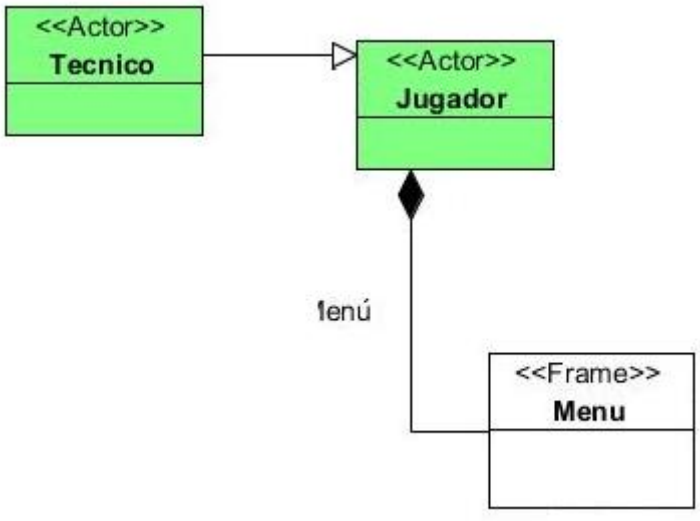

Figura 236. Diagrama de clases menú (Técnico). (Elaboración propia)

Crear nave

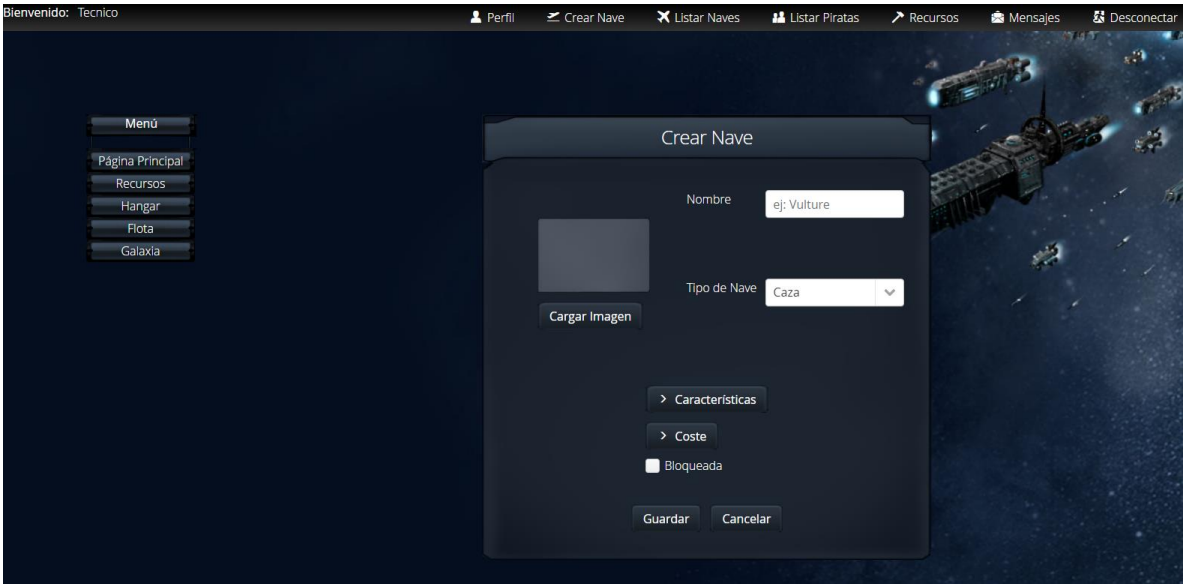

Figura 237. Ventana de crear nave. (Elaboración propia)

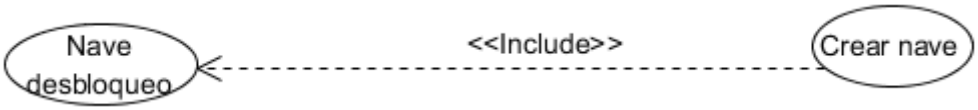

Figura 238. Diagrama casos de uso crear nave. (Elaboración propia)

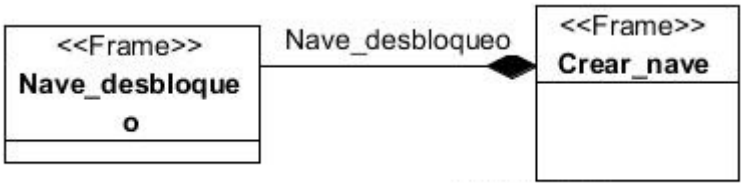

Figura 239. Diagrama de clases crear nave. (Elaboración propia)

Listado de naves

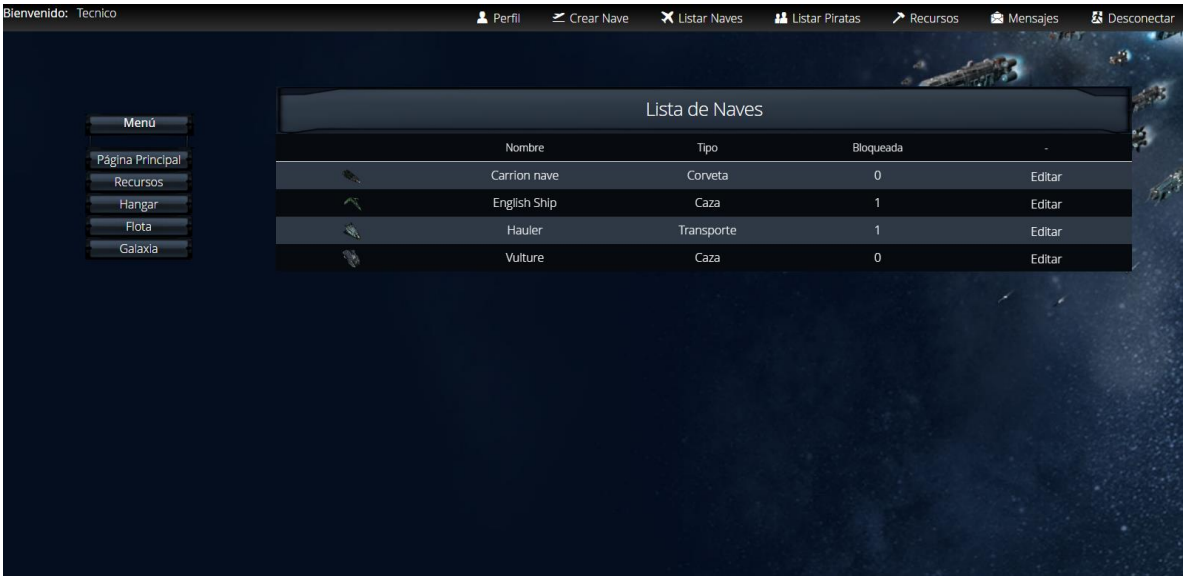

Figura 240. Ventana de listado de naves. (Elaboración propia)

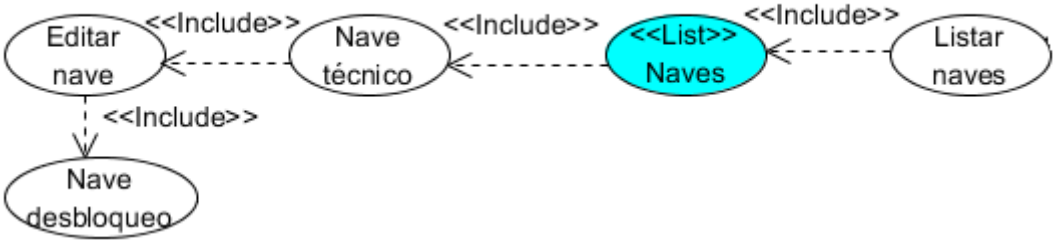

Figura 241. Diagrama casos de uso listar naves. (Elaboración propia)

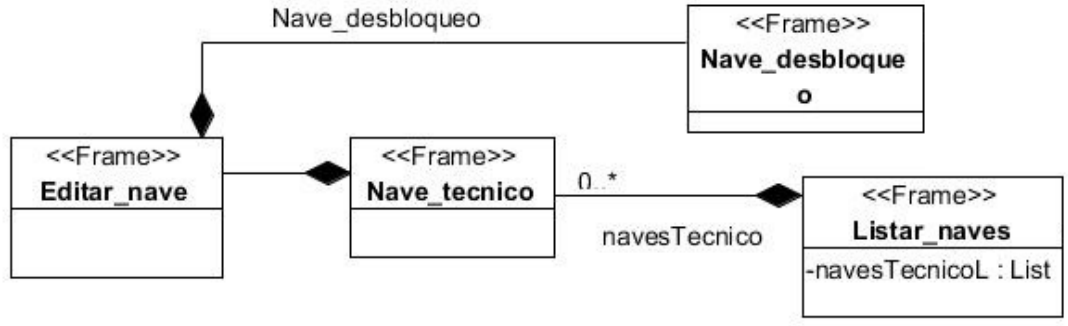

Figura 242. Diagrama de clases listar naves. (Elaboración propia)

Editar nave

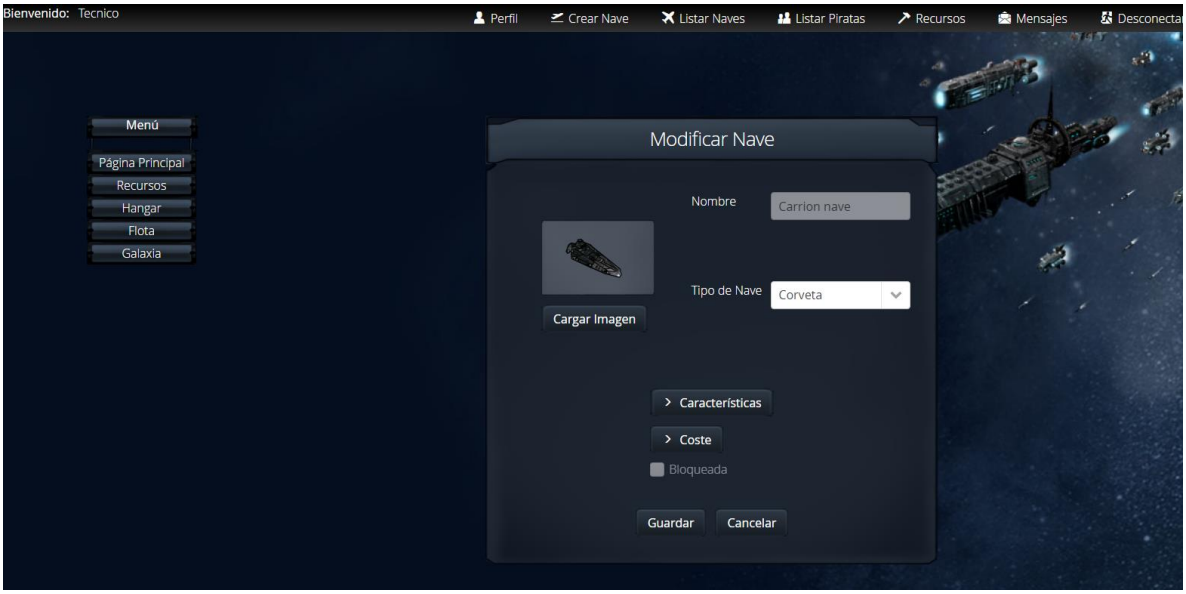

Figura 243. Ventana de editar nave. (Elaboración propia)

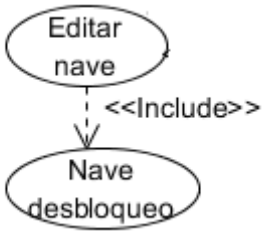

Figura 244. Diagrama casos de uso editar nave. (Elaboración propia)

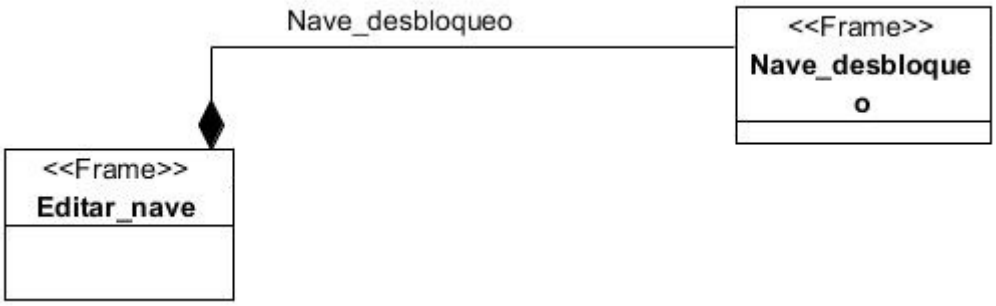

Figura 245. Diagrama de clases editar nave. (Elaboración propia)

Listado de piratas

Figura 246. Ventana de listado de piratas. (Elaboración propia)

Figura 247. Diagrama casos de uso piratas. (Elaboración propia)

Figura 248. Diagrama de clases piratas. (Elaboración propia)

Editar pirata (Niveles de sus instalaciones)

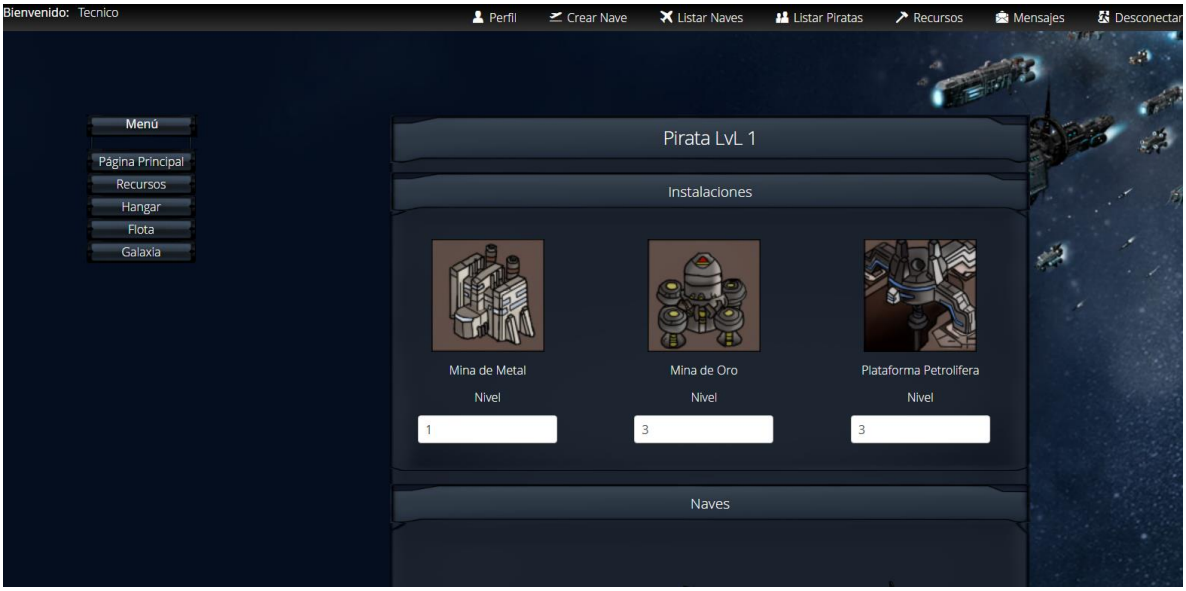

Figura 249. Ventana de editar pirata (instalaciones). (Elaboración propia)

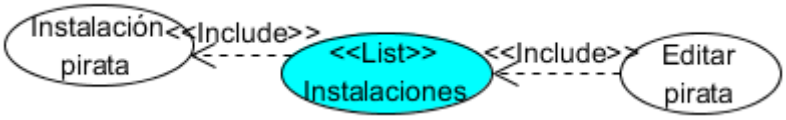

Figura 250. Diagrama casos de uso instalaciones pirata. (Elaboración propia)

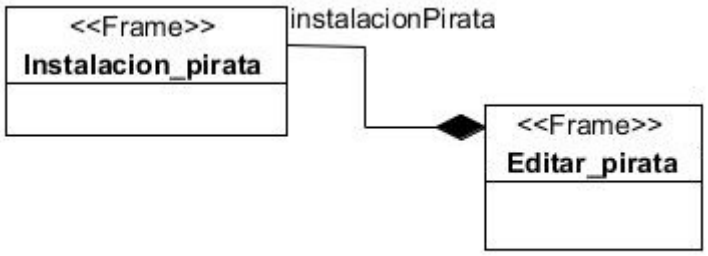

Figura 251. Diagrama de clases instalaciones pirata. (Elaboración propia)

Editar pirata (Cantidad de naves disponibles)

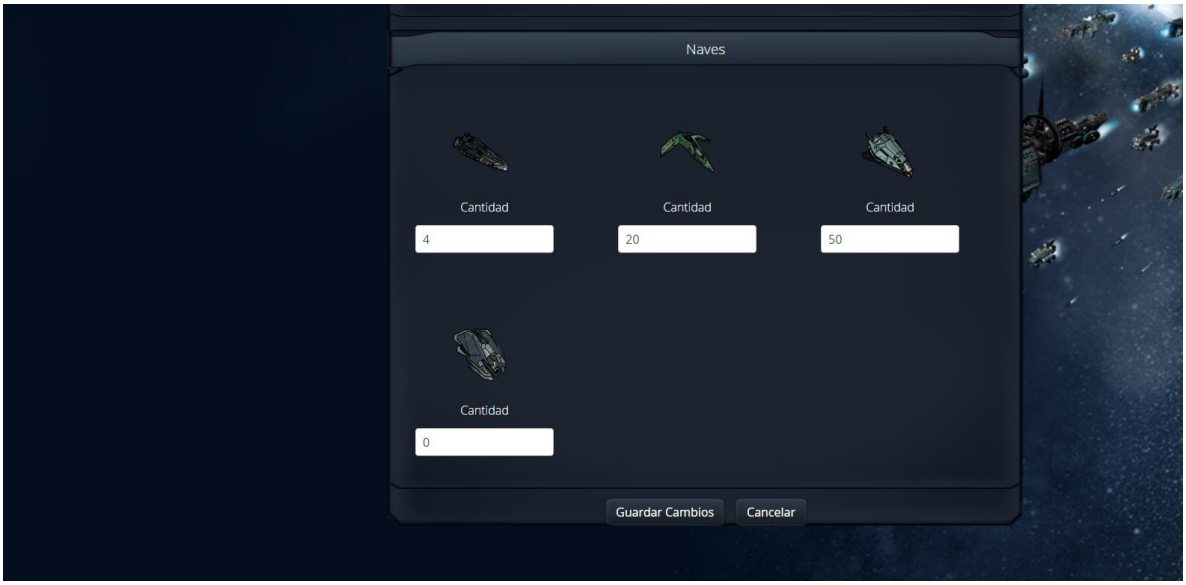

Figura 252. Ventana de editar pirata (naves). (Elaboración propia)

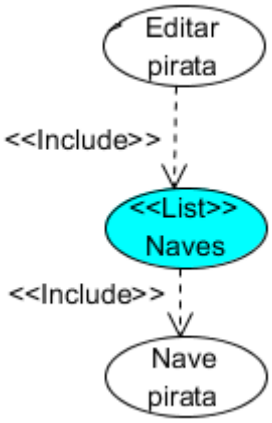

Figura 253. Diagrama casos de uso naves pirata. (Elaboración propia)

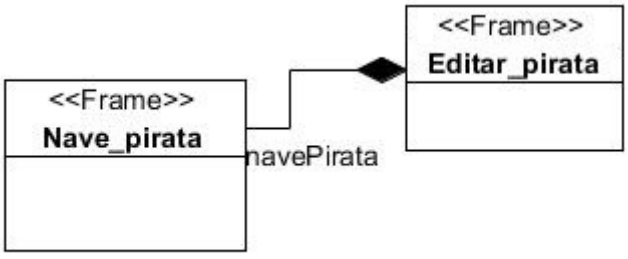

Figura 254. Diagrama de clases naves pirata. (Elaboración propia)

Editar instalaciones (Recursos)

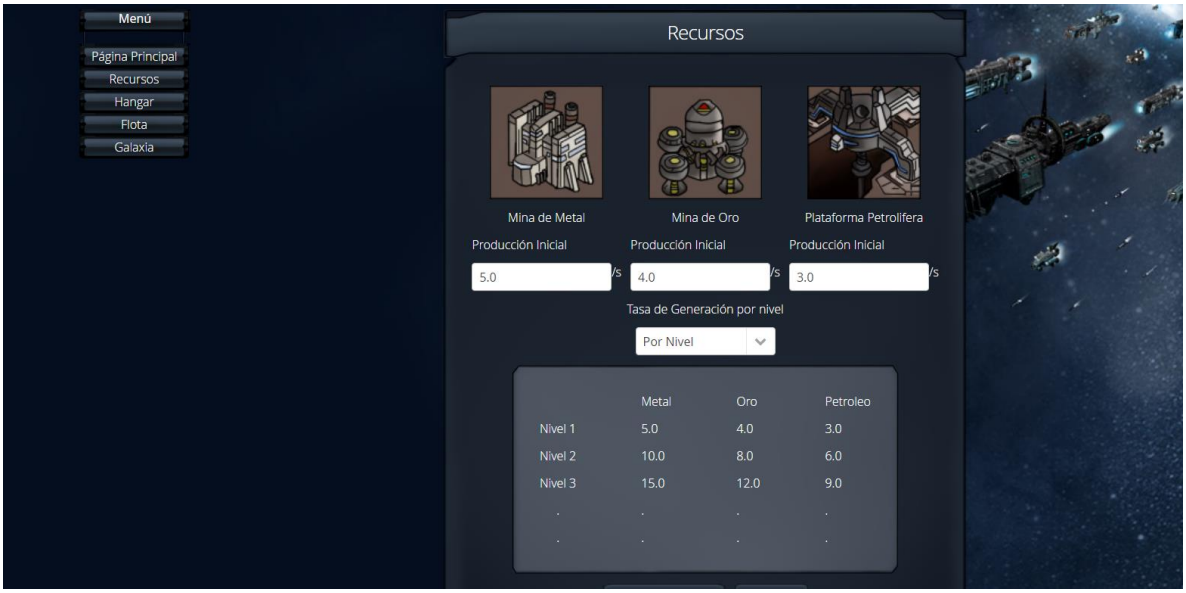

Figura 255. Ventana de recursos (técnico). (Elaboración propia)

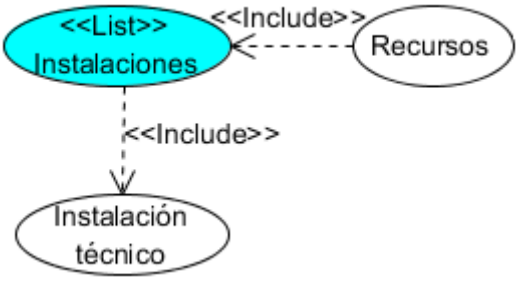

Figura 256. Diagrama casos de uso recursos técnico. (Elaboración propia)

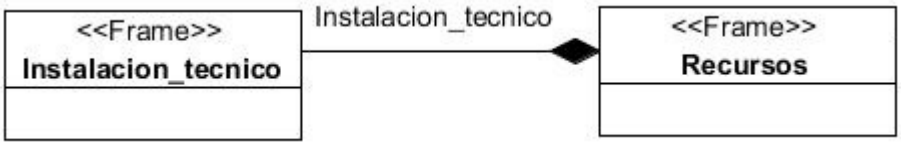

Figura 257. Diagrama de clases recursos técnico. (Elaboración propia)

3.3.7.2. Diseño

A continuación se muestra el diagrama de clases de la interfaz del módulo y el diagrama de la base de datos necesaria para la realización de este módulo. Los diagramas se encuentran con mejor resolución en el anexo.

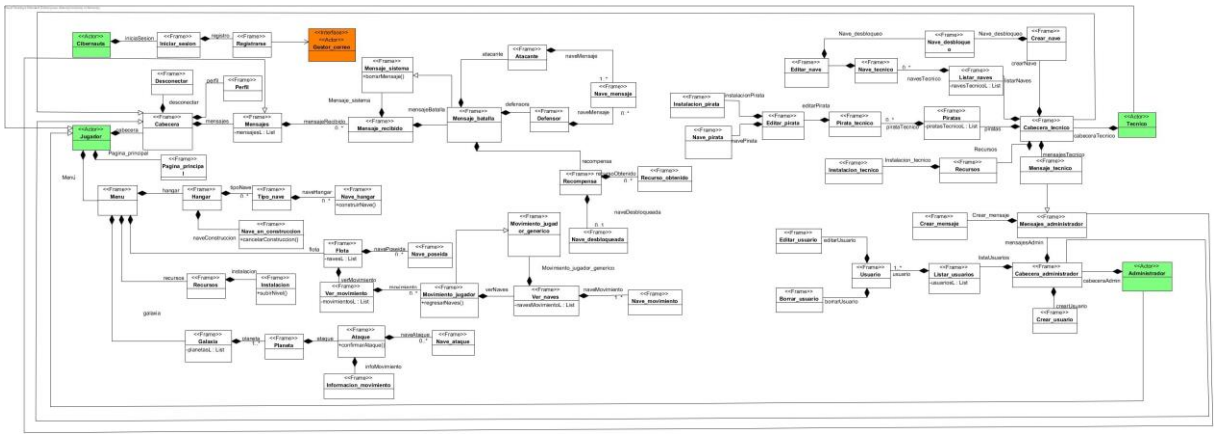

Figura 258. Diagrama de clases del módulo integración. (Elaboración propia)

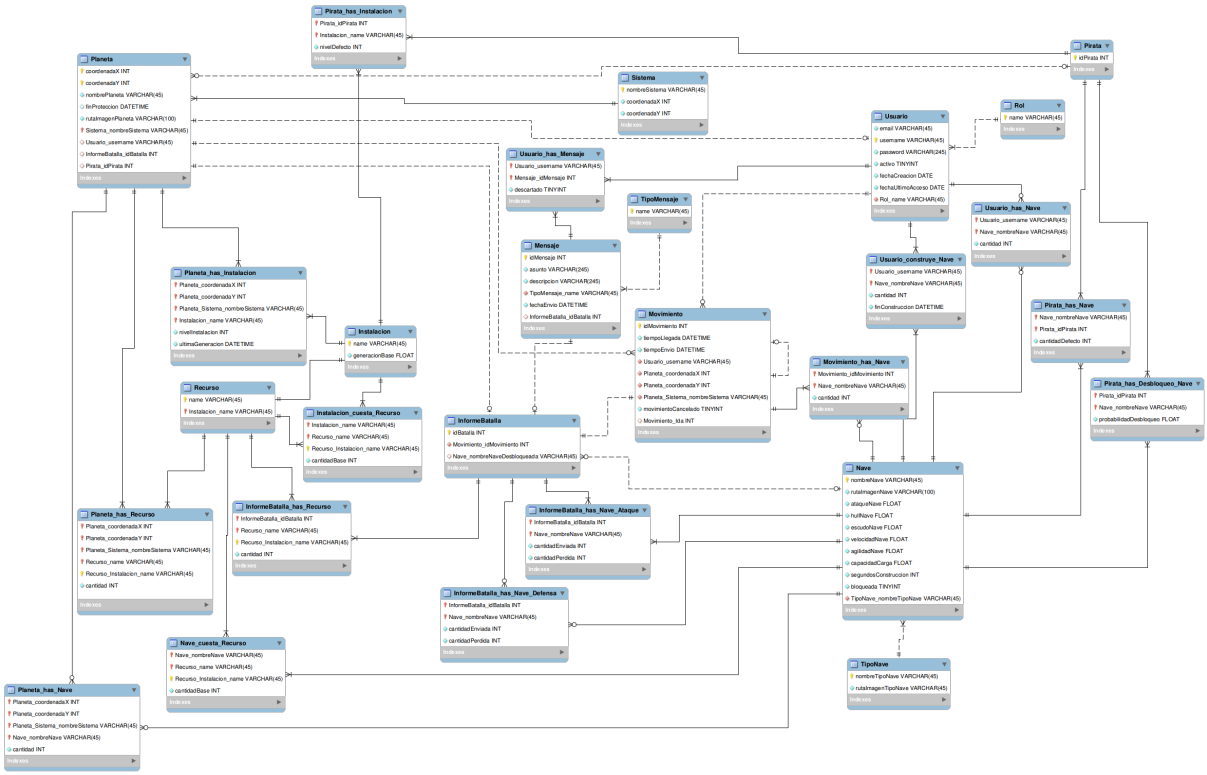

Figura 259. Diagrama de la base de datos del módulo integración. (Elaboración propia)

3.3.7.3.Implementación del módulo

Solo es una semana de duración y una persona en el desarrollo, por lo que para el reparto de tareas no hay ningún problema. Tras la integración quedaron un total de 187 clases y 265 casos de prueba.

3.3.7.3.1. Unión de los módulos

Gracias a que se hizo un desarrollo basado en pruebas la integración fue sencilla, se fueron importando todas las clases y ejecutando las pruebas, en el caso de que una prueba no funcionase se veía la razón y se solucionaba.

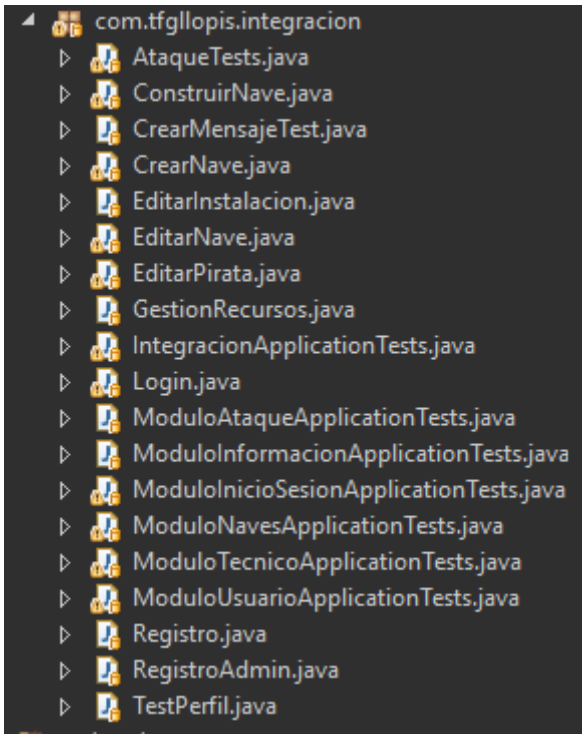

Figura 260. Listado de las clases de pruebas. (Elaboración propia)

Después se establecía la relación entre las distintas ventanas, por ejemplo en el menú incluir todos los botones a las distintas ventanas (la navegabilidad).

```
public Menu()
{
    paginaPrincipalB.addClickListener(new Button.ClickListener()
    {

        @Override
        public void buttonClick(ClickEvent event)
        {
            doNavigate(Pagina_principal.VIEW_NAME);
        }
    });

    recursoB.addClickListener(new Button.ClickListener()
    {

        @Override
        public void buttonClick(ClickEvent event)
        {
            doNavigate(Recursos.VIEW_NAME);

        }
    });

    hangarB.addClickListener(new Button.ClickListener()
    {

        @Override
        public void buttonClick(ClickEvent event)
        {
            doNavigate(Hangar.VIEW_NAME);
```

Figura 261. Constructor de la clase menú. (Elaboración propia)

Por último se hacían modificaciones en la funcionalidad que se había extendido por causa de otros módulos, por ejemplo ahora al crear un usuario también se le tenía que asignar un planeta libre y se eliminaban las clases CRUD, colocando de esta manera cada funcionalidad en su clase correspondiente en lugar de en una clase general.

```
public static String inicializarUsuario(Usuario usuario, PlanetaHasNaveRepository planetaNa
{
    Planeta planeta = planetaRepo.findByPlanetaLibre().get(0);
    PlanetahasInstalacion planetaInstalacionMetal = planetaInstalacionRepo.findByInstalacio
    PlanetahasInstalacion planetaInstalacionOro = planetaInstalacionRepo.findByInstalacionn
    PlanetahasInstalacion planetaInstalacionPetroleo = planetaInstalacionRepo.findByInstala
    PlanetahasRecurso planetaRecursoMetal = planetaRecursoRepo.findByPlanetaRecurso(planeta
    PlanetahasRecurso planetaRecursoOro = planetaRecursoRepo.findByPlanetaRecurso(planeta.g
    PlanetahasRecurso planetaRecursoPetroleo = planetaRecursoRepo.findByPlanetaRecurso(plan

    planeta.setUsuariousername(usuario);
    planeta.setPirataidPirata(null);
    planeta.setInformeBatallaidBatalla(null);
    planeta.setNombrePlaneta("Planeta de " + usuario.getUsername());

    planetaInstalacionMetal.setUltimaGeneracion(new Date());
    planetaInstalacionMetal.resetNivelInstalacion();
    planetaInstalacionOro.setUltimaGeneracion(new Date());
    planetaInstalacionOro.resetNivelInstalacion();
    planetaInstalacionPetroleo.setUltimaGeneracion(new Date());
    planetaInstalacionPetroleo.resetNivelInstalacion();

    planetaRecursoMetal.setCantidad(0);
    planetaRecursoOro.setCantidad(0);
    planetaRecursoPetroleo.setCantidad(0);

    planetaNaveRepo.deleteAll(planetaNaveRepo.findByPlaneta(planeta.getCoordenadaX(), plane

    planetaRepo.save(planeta);

    planetaInstalacionRepo.save(planetaInstalacionMetal);
    planetaInstalacionRepo.save(planetaInstalacionOro);
    planetaInstalacionRepo.save(planetaInstalacionPetroleo);
```

Figura 262. Método de inicializar usuario. (Elaboración propia)

3.3.7.3.2. Resolución de errores

Tras integrar todo se comenzaron a resolver problemas que se llevaban arrastrando desde módulos anteriores, y que surgieron al integrar o que fueron identificados en las pruebas posteriores, por ejemplo, la incorrecta ruta de las imágenes o un error en la navegabilidad de algunas ventanas.

También se solucionó el problema del id máximo de un movimiento, se añadió un nuevo campo a la tabla movimiento que almacena en caso de ser un movimiento de vuelta el id del movimiento de ida.

3.3.7.3.3. Guía de usuario

En la página principal se colocó un pequeño tutorial para explicar brevemente a los usuarios cómo funciona la aplicación y cuál es el objetivo del juego.

3.3.7.3.4. Mejora en el diseño

Por último se realizaron mejoras en el diseño de la aplicación incorporando un CSS propio, introduciendo explicaciones al usuario y textos de ayuda y mejorando dentro de lo posible la capacidad responsive de la aplicación puesto que aunque Vaadin te indica que es responsive no lo es ni posee herramientas para ello.

```
.fondo_cabecera_cuerpo
{
  background: rgba(0,0,0,0);
  border: none;
}

 .recursos_fondo
{
  background: url(Recursos_Fondo.png) no-repeat !important;
  background-size: 100% 100% !important;
}

 .recursos_fondo_horizontal
{
  background: url(Recursos_Fondo_Horizontal.png) no-repeat;
  background-size: 100% 100%;
}

.v-grid-cell
{
  background-color: #2f3946 !important;
  color: white !important;
}

.v-grid-row-stripe > td
{
  background-color: #070a0e !important;
  color: white !important;
}

.v-grid-row > td
{
  border: none !important;
```

Figura 263. Captura de parte del archivo del CSS. (Elaboración propia)

3.3.7.4. Feedback

Tras probarlo, el usuario ha aconsejado que los recursos se vean en todas las ventanas, así al ir a construir naves se sabe cuantos recursos faltan. También ha aconsejado que se disminuya el ancho de la ventana de hangar.

3.4. Release

El producto se ha desarrollado de manera exitosa, se han cumplido todos los requisitos y se ha realizado un buen diseño de la interfaz como se esperaba.

Se han implementado 256 casos de prueba, es decir, el código se ha probado en gran medida. La aplicación se ha subido a un servidor y está accesible para cualquiera.

Aun así hay varias cosas que han salido mal durante el desarrollo: solo se han realizado pruebas unitarias, no se han realizado pruebas funcionales ni pruebas de rendimiento o estrés, esto es debido a dos razones, la primera y la principal, el tiempo. El proyecto ha ido muy justo de tiempo, era más grande de lo esperado y había muchas cosas que pulir, sobre todo en el módulo del ataque, si el trigger de ataque fallaba la aplicación era un fracaso, por esta razón se han simplificado cosas como la guía de usuario, que no se identificó como un requisito pero que era necesaria debido a la complejidad de la aplicación.

La segunda razón alimenta la primera, es el propio Vaadin, las pruebas funcionales con Vaadin son muy lentas, se debe de codificar todo, no es como Selenium que te las realiza automáticamente, por lo que se tarda demasiado tiempo en realizarlas.

Otra cosa que no ha salido como se esperaba es el responsive, Vaadin se vende como un framework de desarrollo web responsive, que te da herramientas que facilitan la adaptación de la interfaz a distintas resoluciones, pero esto es falso, todo el responsive tiene que hacerse a mano en el css, además no permite el uso de bootstrap, por lo que la complejidad aumenta.

Por último, cabe destacar la ausencia de dos cosas que considero necesarias para este proyecto y no se identificaron en los requisitos ni se han realizado, la primera es la normalización de la base de datos. La base de datos es el núcleo de la aplicación, si es ineficiente, el juego es injugable, es decir, termina en fracaso, en el momento en el que se identificó este requisito era demasiado costoso en tiempo solucionarlo.

La segunda ausencia es en cuanto a normativa, en este caso, la ley de protección de datos europea que entró en vigor el 25 de mayo de este año. La aplicación carece de la adaptación e información con respecto a esta nueva normativa, por lo que solamente estará accesible para su defensa.

A pesar de todas estas debilidades, el proyecto no ha sido un fracaso, al contrario, ha salido mejor de lo esperado, la razón de que haya más debilidades que fortalezas es que de las debilidades hay que aprender para que no vuelvan a ocurrir, por lo que centrarse en ellas facilita que no se repitan.

4. Casos prácticos

A continuación vamos a ver una serie de casos reales de uso de la aplicación.

Primero nos pondremos en el papel de un cibernauta que se registra por primera vez y realiza sus primeros pasos en el juego.

Luego veremos la figura de un administrador, cómo bloquea a un usuario que está causando problemas, crea un nuevo técnico y envía un aviso de parada del sistema por mantenimiento.

Por último trataremos el rol de un técnico que se identifica para añadir una nueva nave al juego, modificar los piratas y avisar de los cambios de balance realizados.

4.1. Nuevo usuario

Esta sería la primera pantalla que veríamos como cibernauta al entrar a la aplicación, la ventana de inicio de sesión.

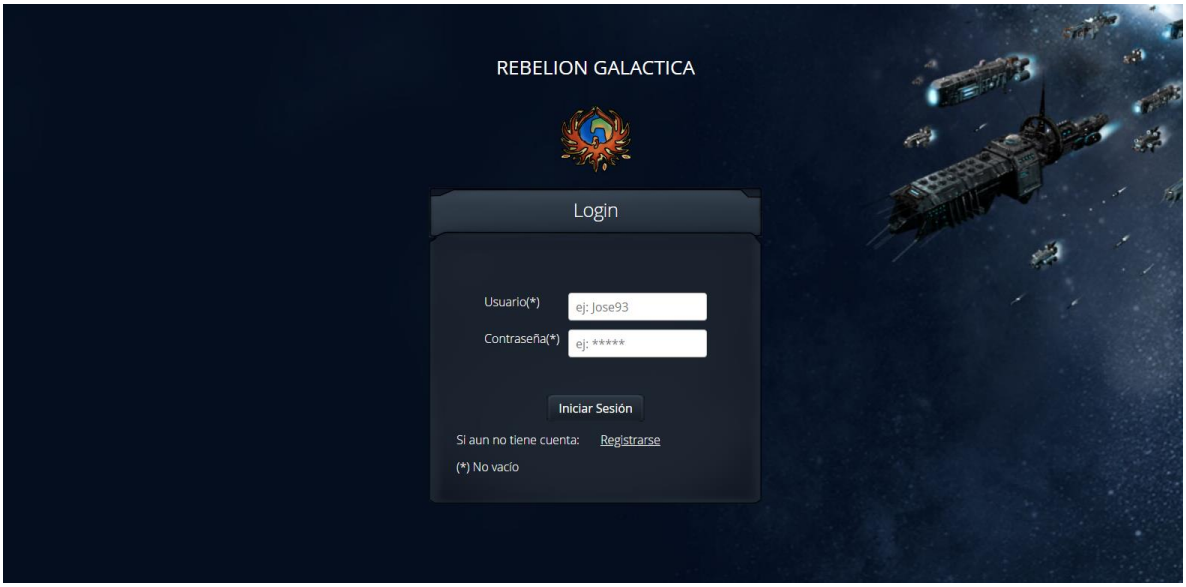

Figura 264. Ventana de inicio de sesión (casos prácticos). (Elaboración propia)

A continuación rellenamos nuestros datos personales para registrarnos.

Figura 265. Ventana de registro (casos prácticos). (Elaboración propia)

En el caso de equivocarnos en algún campo el sistema nos avisaría del error para que lo solucionáramos.

Figura 266. Ventana de registro error (casos prácticos). (Elaboración propia)

Tras arreglar el error ya podemos registrarnos correctamente.

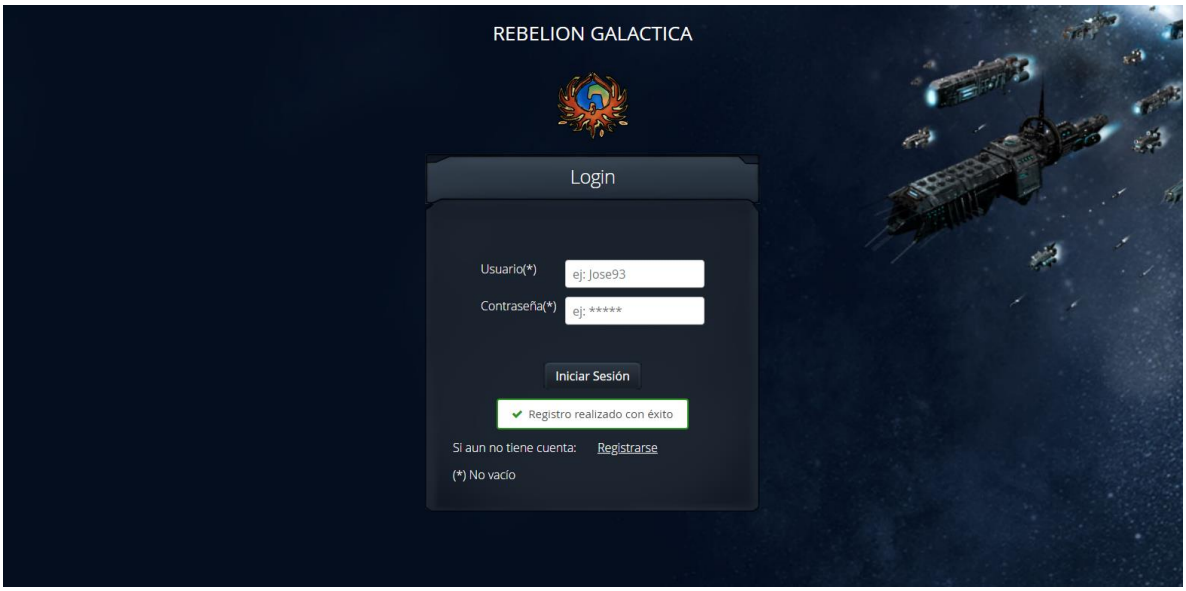

Figura 267. Ventana de registro correcto (casos prácticos). (Elaboración propia)

Tras iniciar sesión lo primero que vemos son nuestros recursos y un breve tutorial de los pasos a seguir en el juego.

Figura 268. Pantalla principal jugador (casos prácticos). (Elaboración propia)

Seguimos el tutorial y nos construimos la nave disponible y más barata que encontramos.

Figura 269. Ventana de hangar (casos prácticos). (Elaboración propia)

El sistema nos avisa de que la construcción ha comenzado correctamente y que en unos 4 minutos tendremos las naves en nuestra flota.

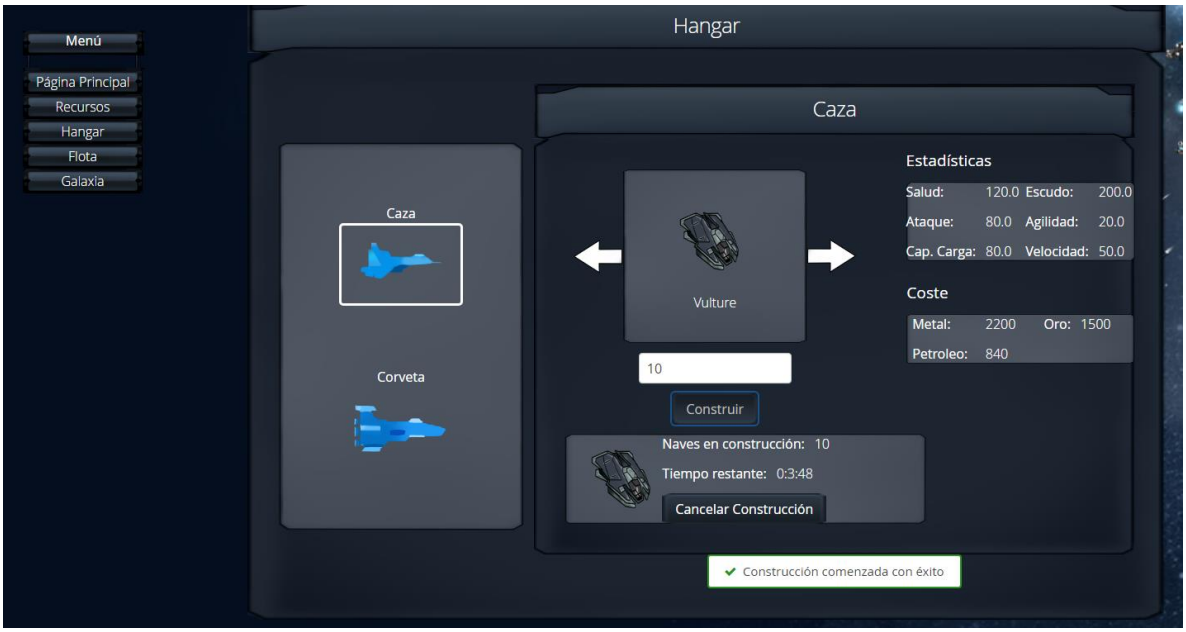

Figura 270. Ventana de hangar construyendo naves (casos prácticos). (Elaboración propia)

Navegamos por las naves existentes y encontramos una interesante, aunque tiene un candado y no deja construirla.

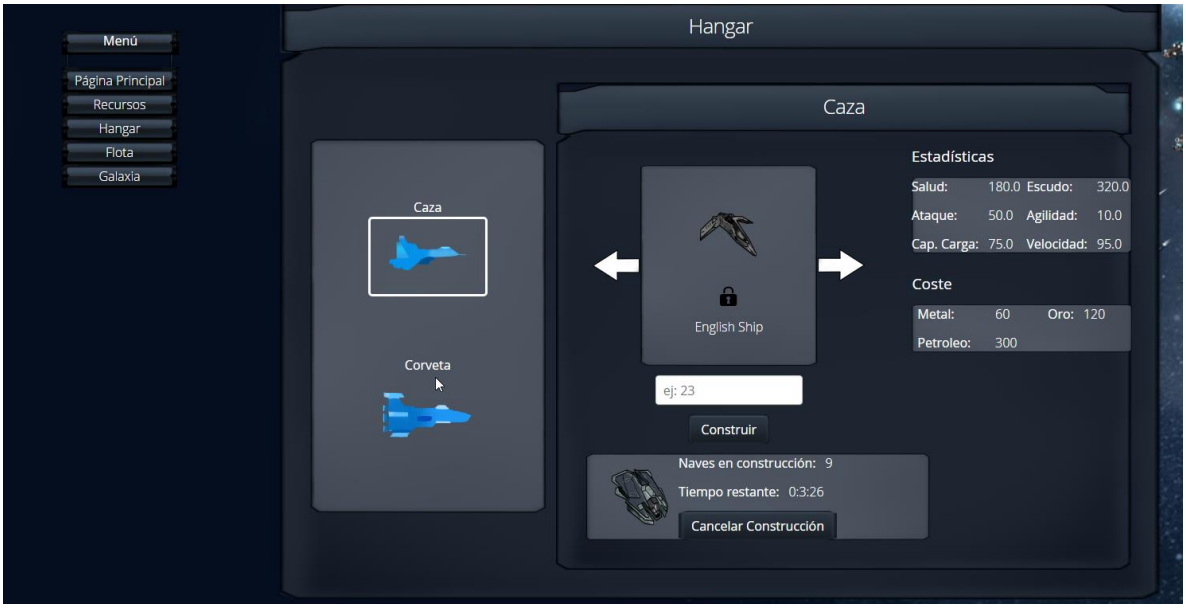

Figura 271. Ventana de nave bloqueada (casos prácticos). (Elaboración propia)

Volvemos a la página principal para seguir con el tutorial, ahora recomienda subir de nivel las instalaciones, así que vamos a ello.

Figura 272. Pantalla principal jugador (casos prácticos). (Elaboración propia)

Vamos a intentar subir de nivel la mina de metal ya que es la primera que aparece.

Figura 273. Ventana de recursos (casos prácticos). (Elaboración propia)

No tenemos suficientes recursos, vamos a volver al tutorial a ver que nos recomienda ahora.

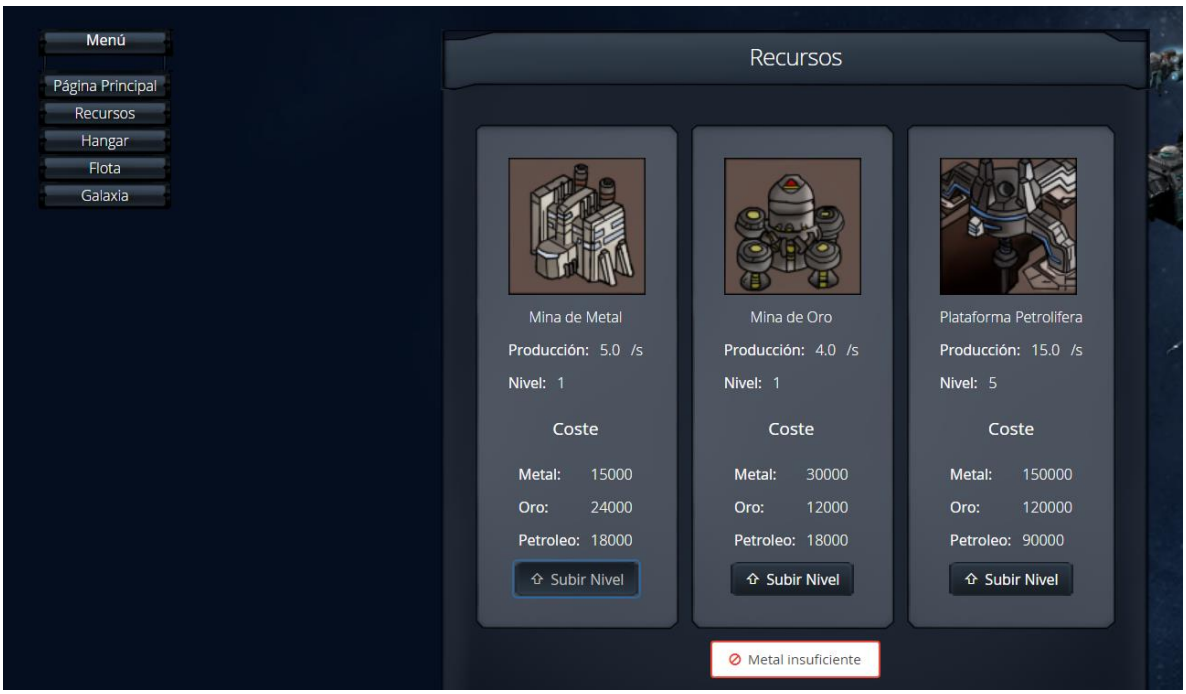

Figura 274. Ventana de recursos error (casos prácticos). (Elaboración propia)

Nos dice que derrotemos piratas para desbloquear naves y conseguir recursos, pues vamos a ello.

Figura 275. Pantalla principal jugador (casos prácticos). (Elaboración propia)

Hay jugadores y piratas, probemos a atacar al pirata que tiene menos nivel.

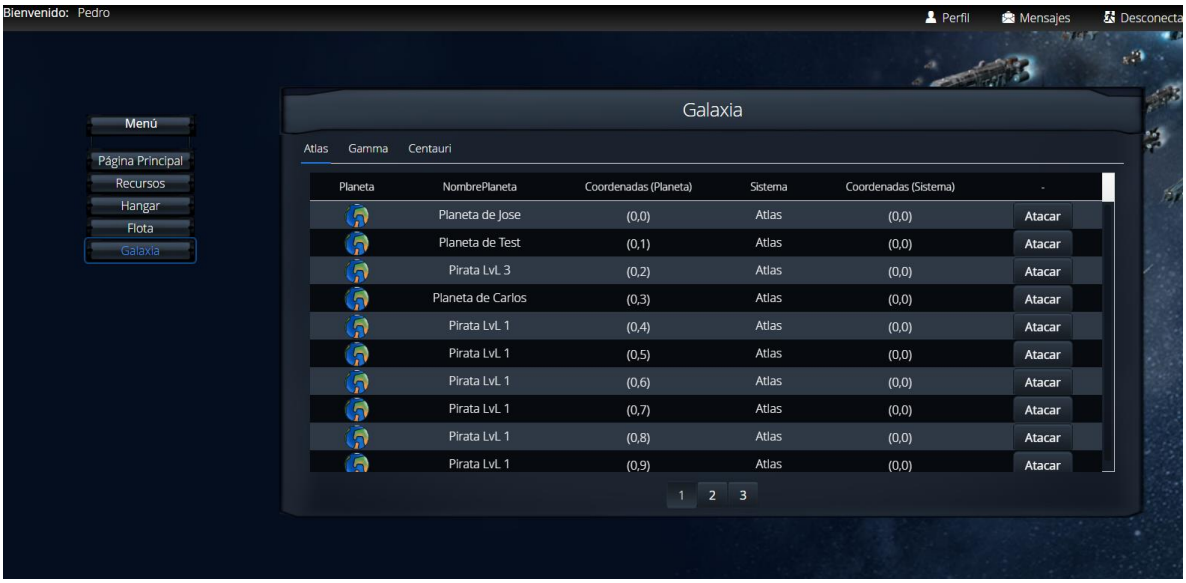

Figura 276. Ventana de galaxia (casos prácticos). (Elaboración propia)

Ahora hay que elegir las naves que queremos enviar de las que tenemos disponibles.

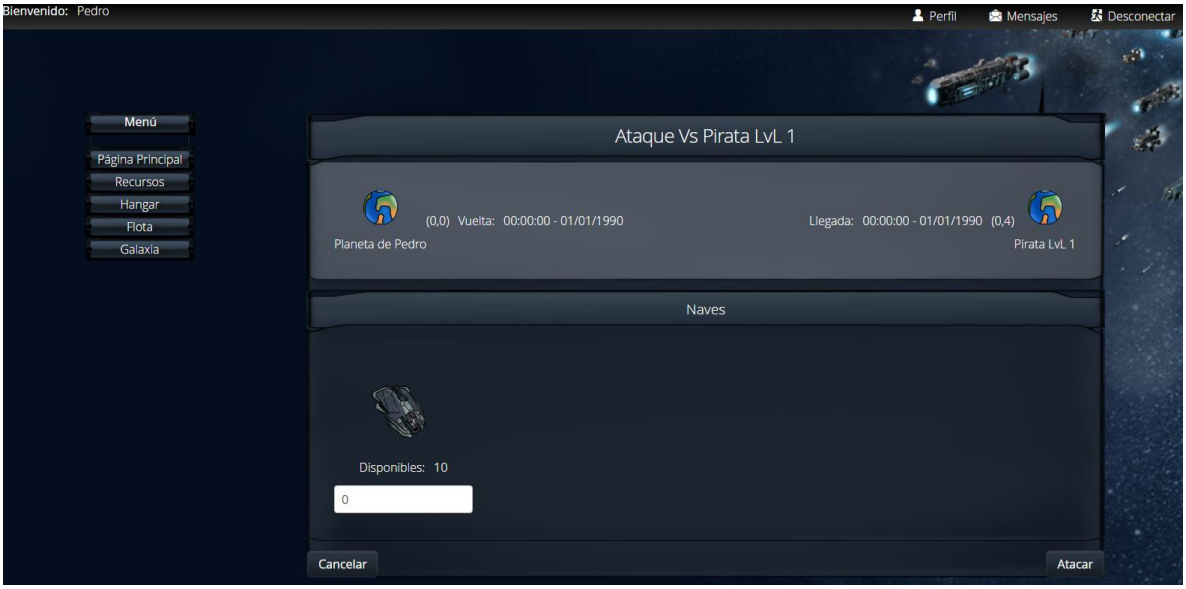

Figura 277. Ventana de ataque (casos prácticos). (Elaboración propia)

Enviamos todas, parece que tardará 1 hora en llegar al planeta de los piratas.

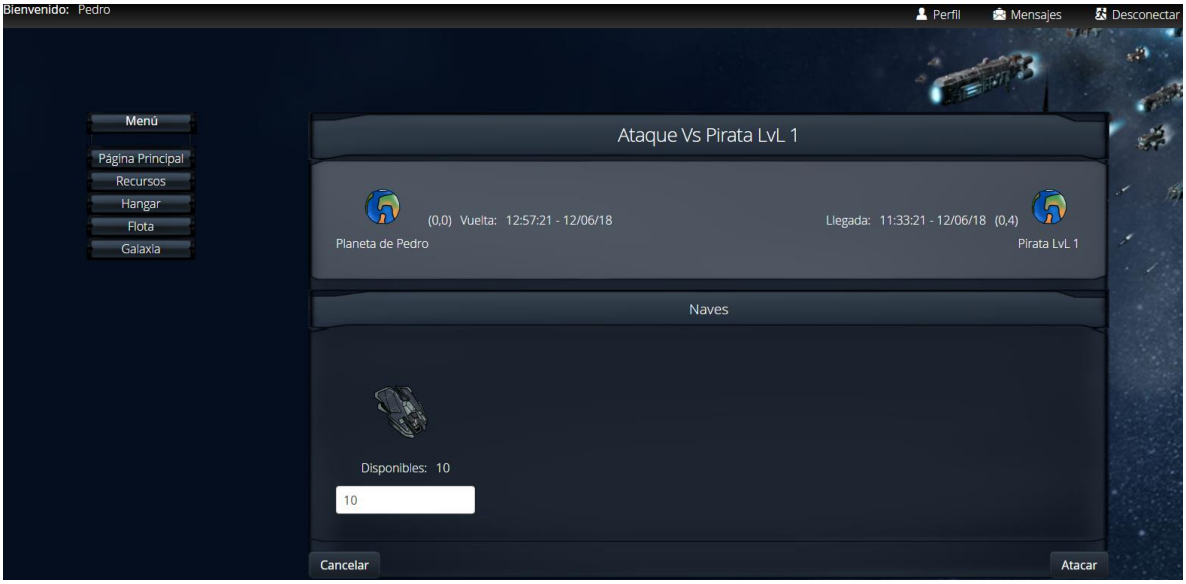

Figura 278. Ventana de ataque con naves (casos prácticos). (Elaboración propia)

Tras confirmar el ataque nos sale un resumen de nuestro ataque y el tiempo restante.

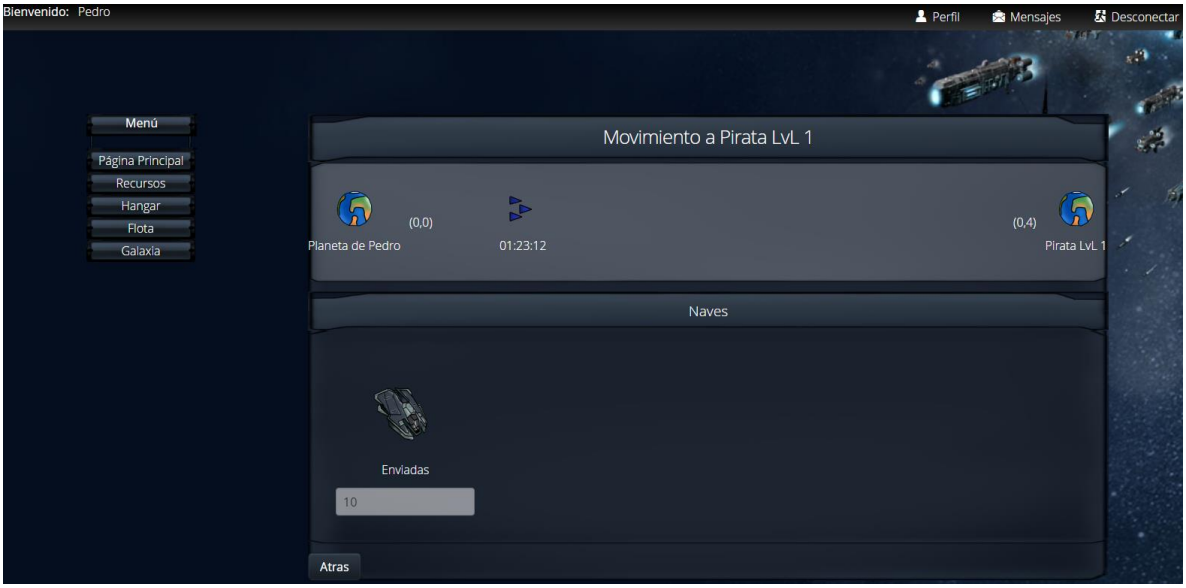

Figura 279. Ventana de movimiento (casos prácticos). (Elaboración propia)

Tras esperar el tiempo estimado (acelerar el tiempo) vemos que el ataque ya acabó y que increíblemente hemos ganado, vamos a ver qué es lo que sucedió.

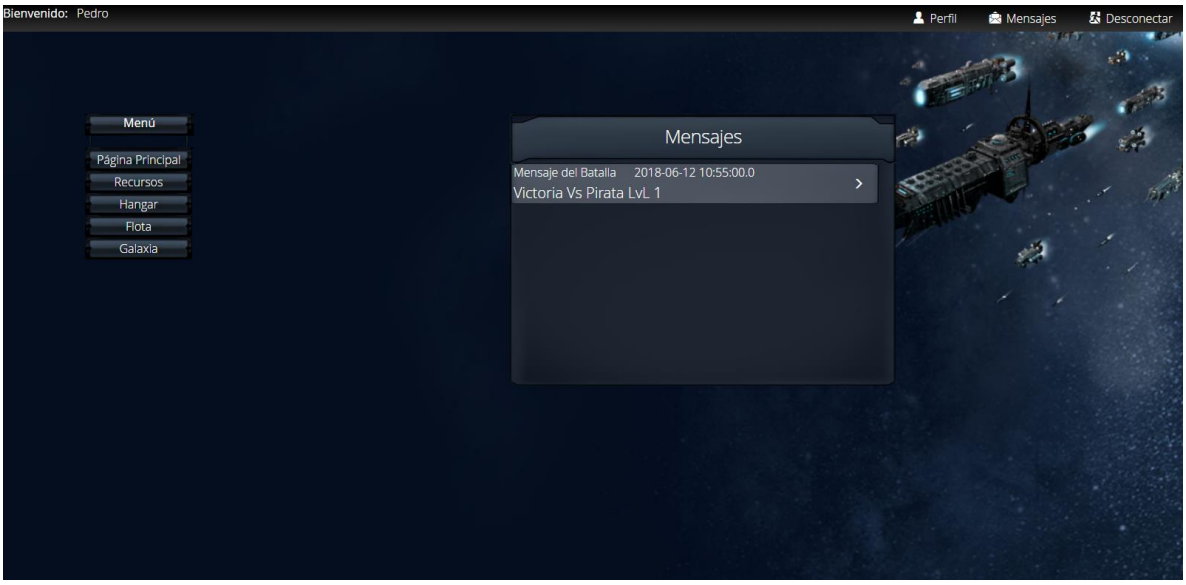

Figura 280. Ventana de mensajes (casos prácticos). (Elaboración propia)

No perdimos ninguna nave.

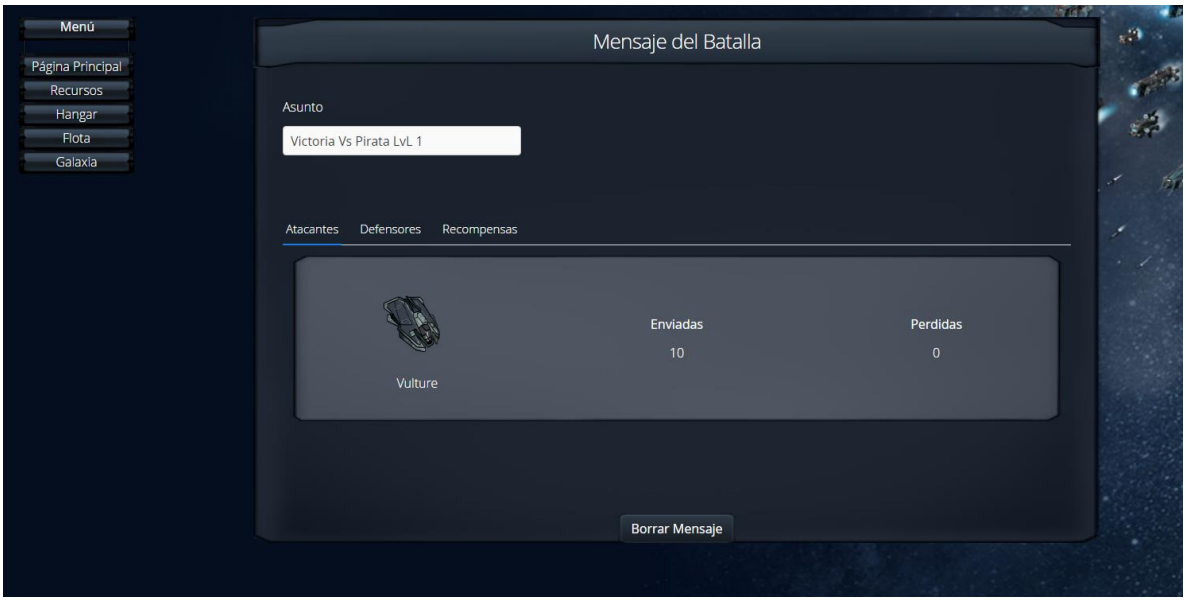

Figura 281. Ventana de atacantes de mensaje de tipo batalla (casos prácticos). (Elaboración propia)

No tenía ninguna nave, posiblemente alguien le atacó antes.

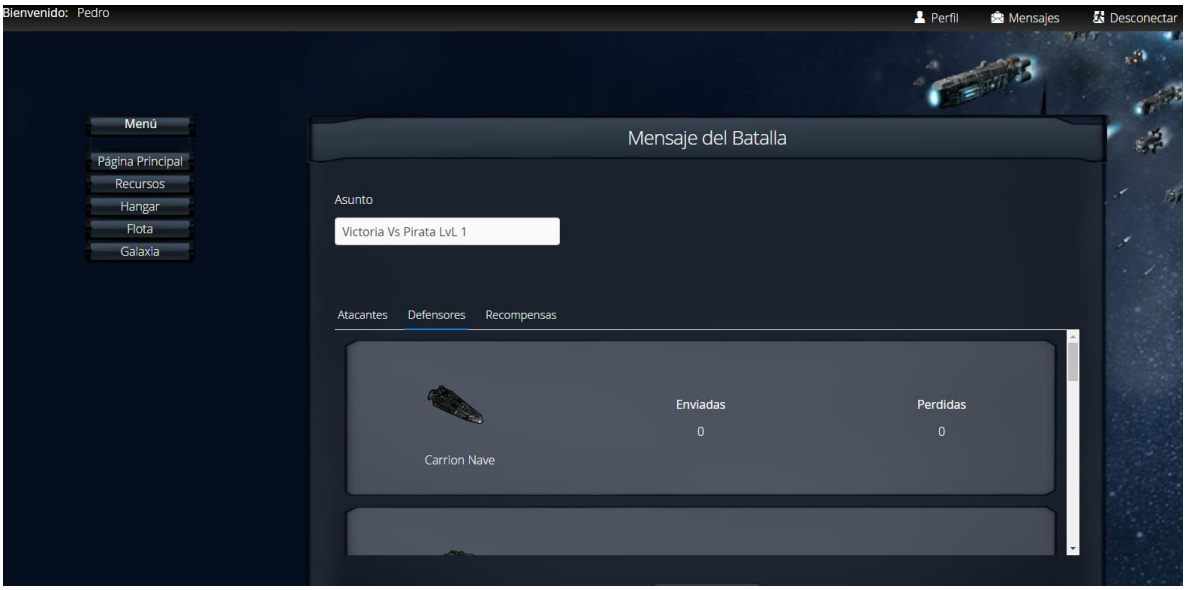

Figura 282. Ventana de defensores de mensaje de tipo batalla (casos prácticos). (Elaboración propia)

Hemos desbloqueado una nueva nave y conseguido recursos.

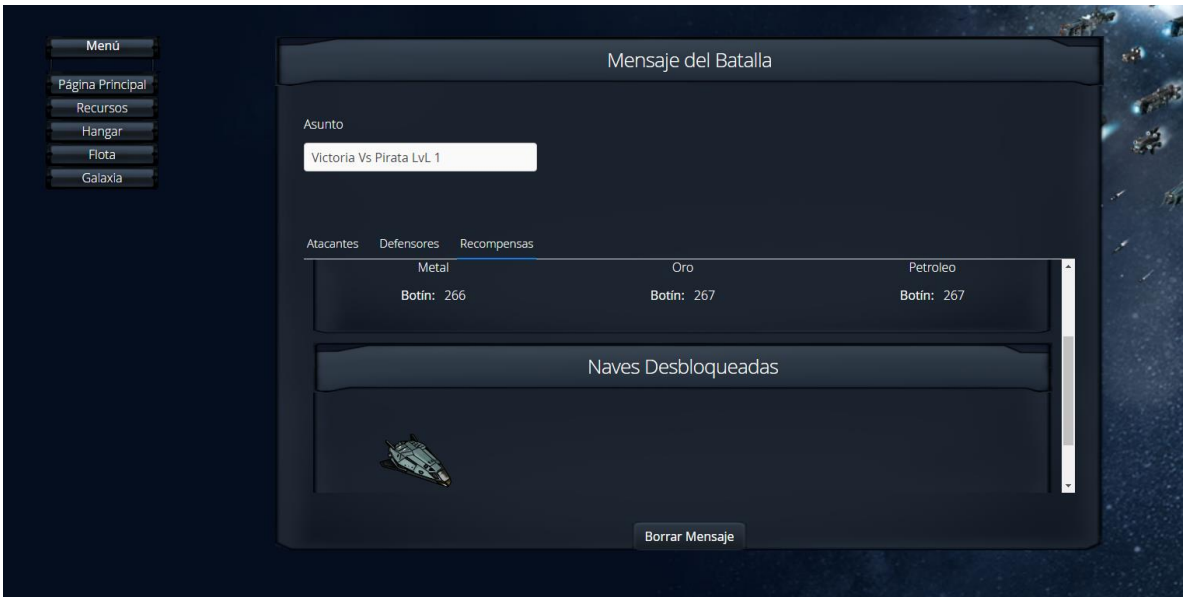

Figura 283. Ventana de recompensas de mensaje de tipo batalla (casos prácticos). (Elaboración propia)

4.2. Gestión de administrador

Ahora toca tomar el rol de un administrador, nuestro primer objetivo es dar de alta un técnico para que gestione el juego.

Figura 284. Pantalla principal administrador (casos prácticos). (Elaboración propia)

Introducimos los datos que nos han proporcionado y lo creamos.

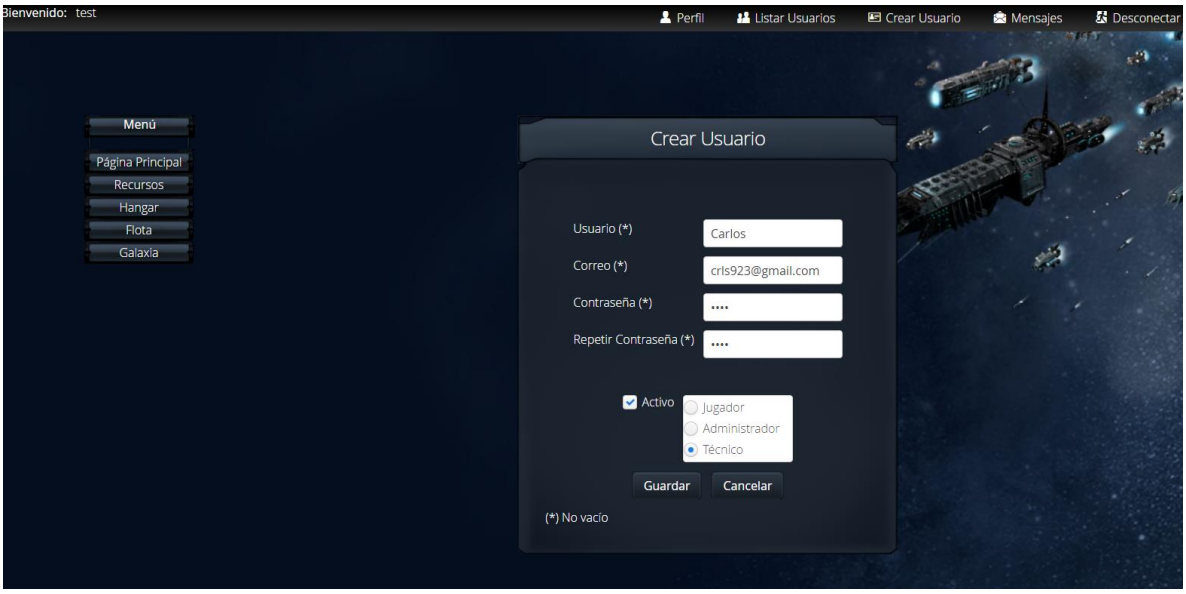

Figura 285. Ventana de crear usuario (casos prácticos). (Elaboración propia)

El técnico se ha creado con éxito, ahora vamos a bloquear a un usuario del que nos han reportado comportamientos extraños.

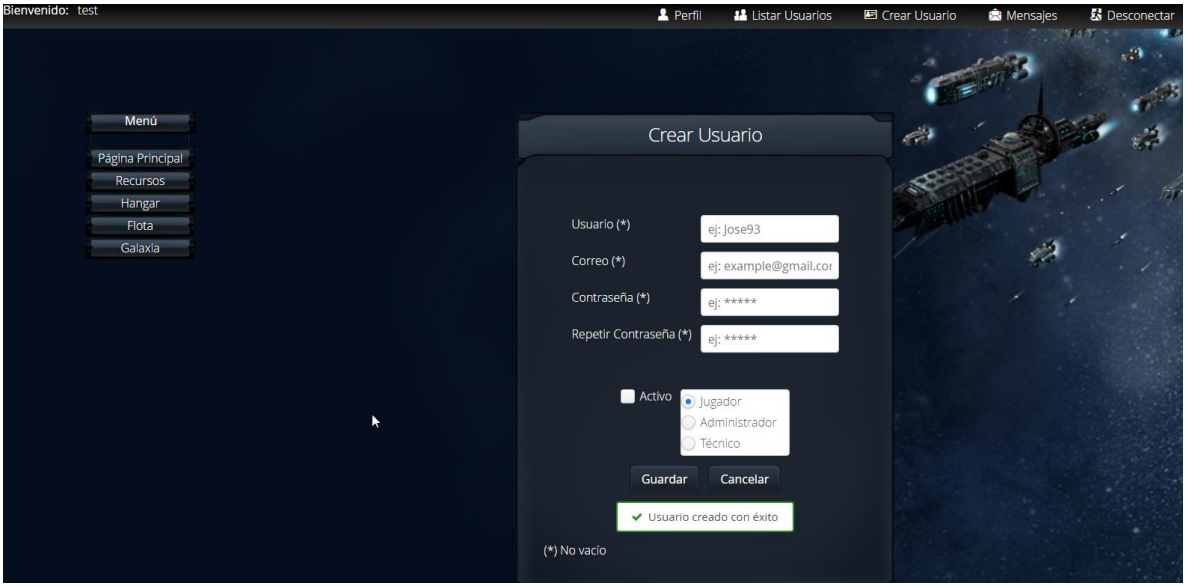

Figura 286. Ventana de crear usuario correcto (casos prácticos). (Elaboración propia)

El usuario era jj, vamos a bloquearle el acceso.

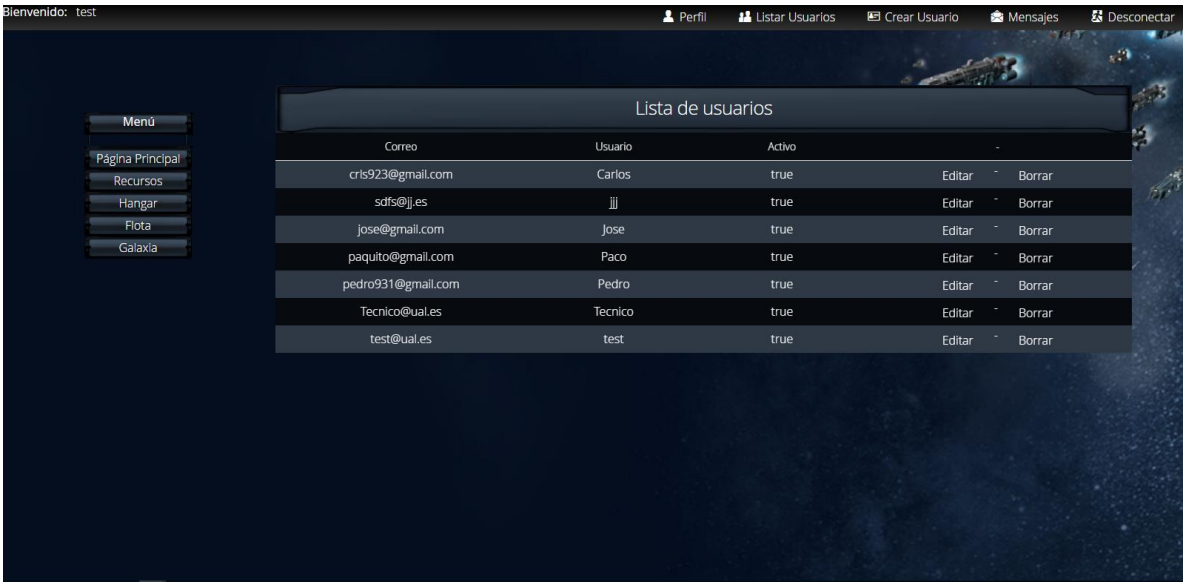

Figura 287. Ventana de listado de usuarios (casos prácticos). (Elaboración propia)

Estos son sus datos.

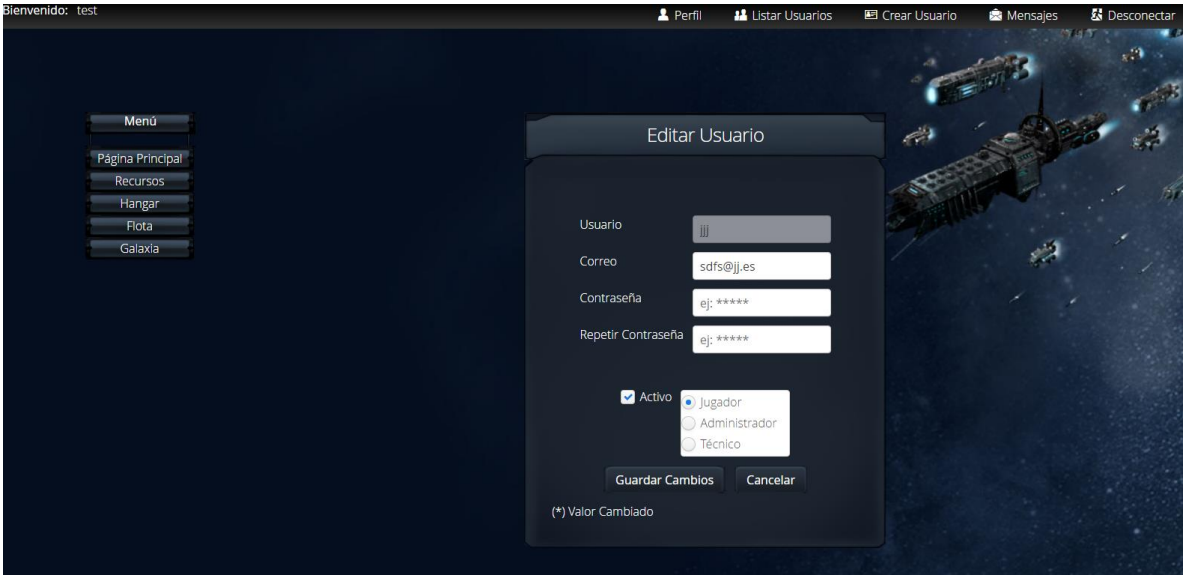

Figura 288. Ventana de editar usuario (casos prácticos). (Elaboración propia)

Le bloqueamos el acceso y confirmamos.

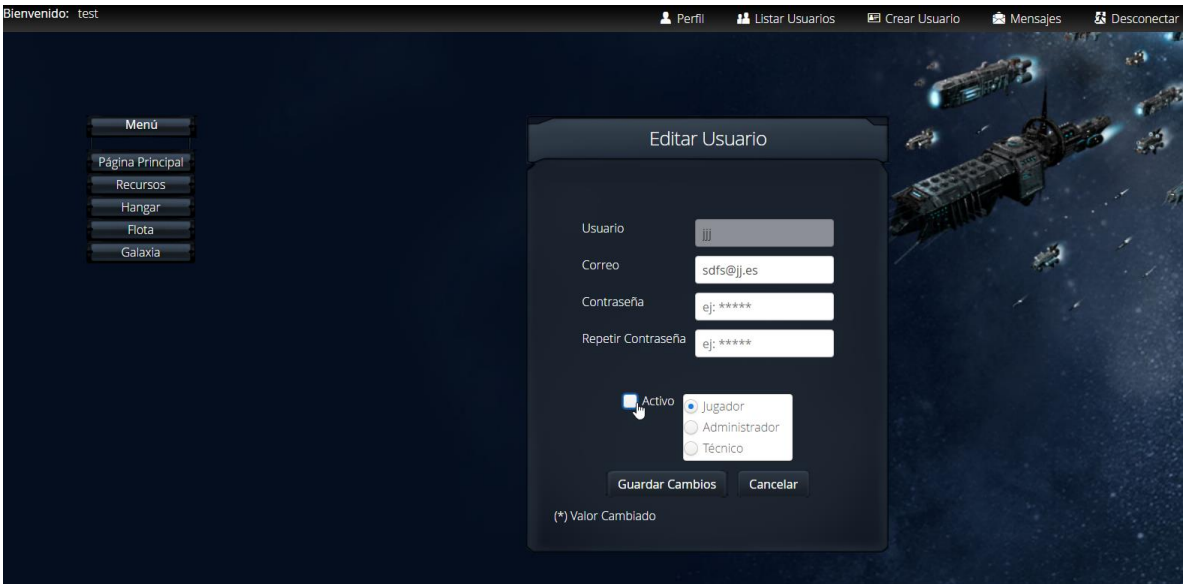

Figura 289. Ventana de editar usuario bloqueado (casos prácticos). (Elaboración propia)

El usuario ha sido bloqueado con éxito, ahora vamos a avisar a los usuarios de un mantenimiento que habrá al final de mes.

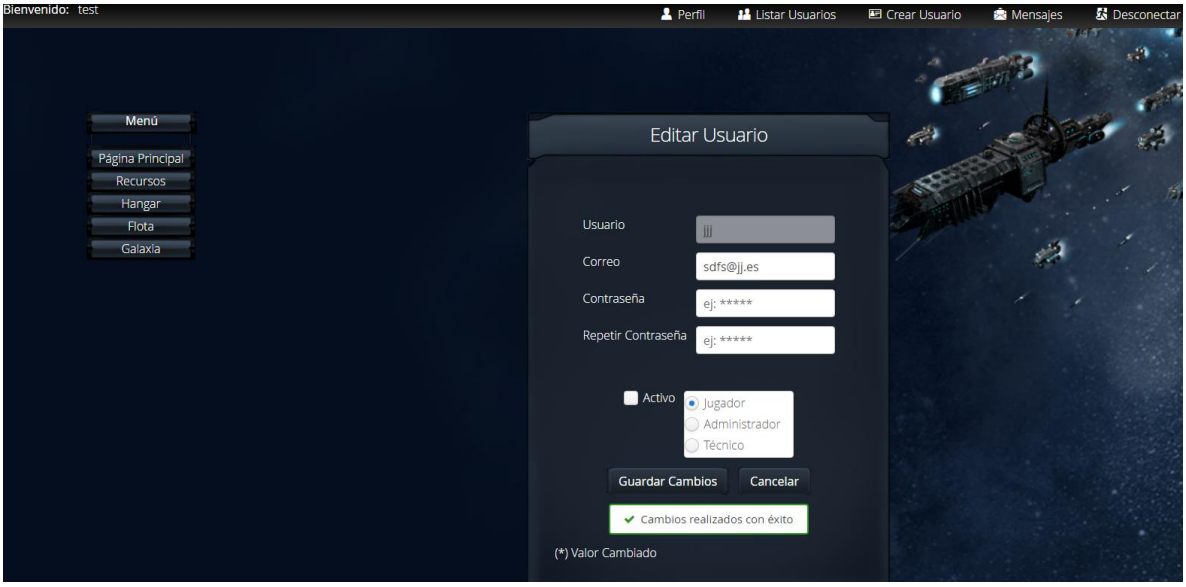

Figura 290. Ventana de editar usuario correcto (casos prácticos). (Elaboración propia)

Nos vamos a mensajes para crear uno nuevo.

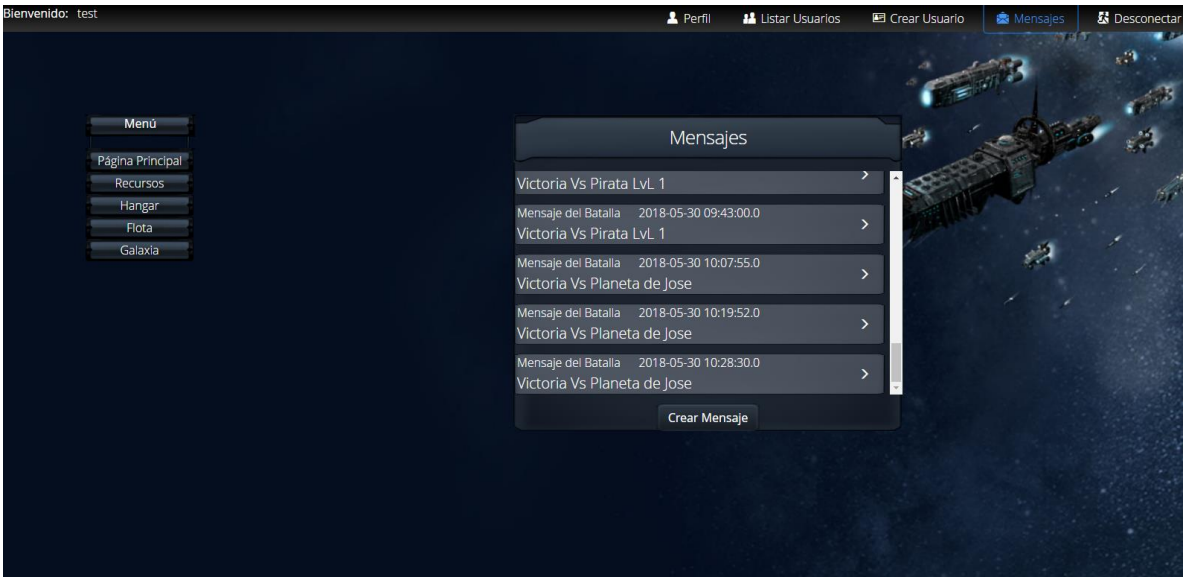

Figura 291. Ventana de mensajes (administrador) (casos prácticos). (Elaboración propia)

Redactamos el mensaje y lo enviamos.

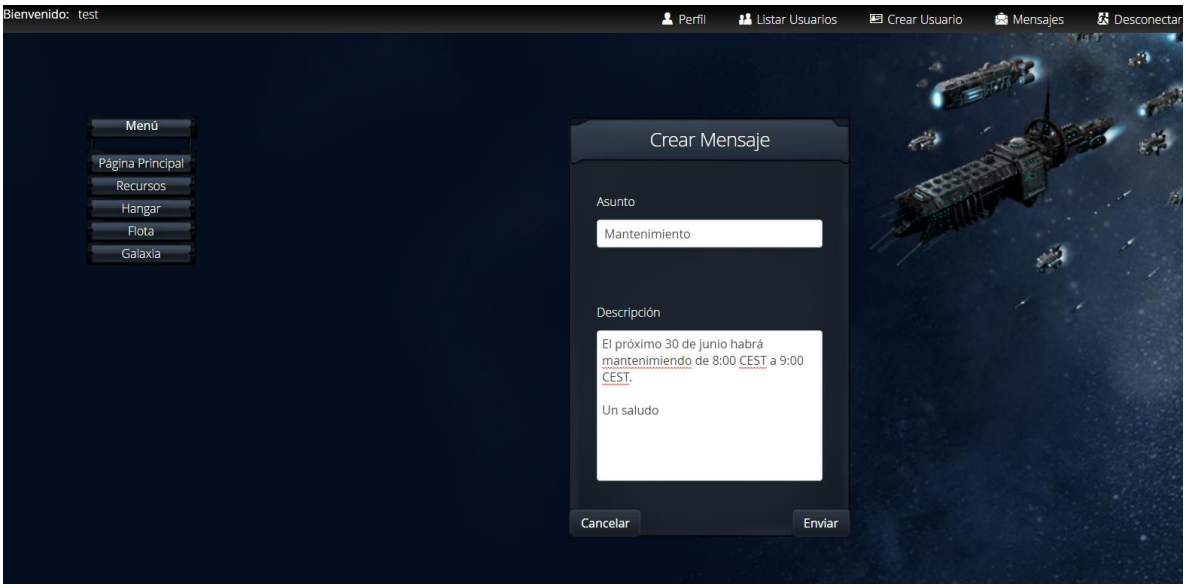

Figura 292. Ventana de crear mensaje (casos prácticos). (Elaboración propia)

Mensaje enviado con éxito y nos aparece en nuestro buzón.

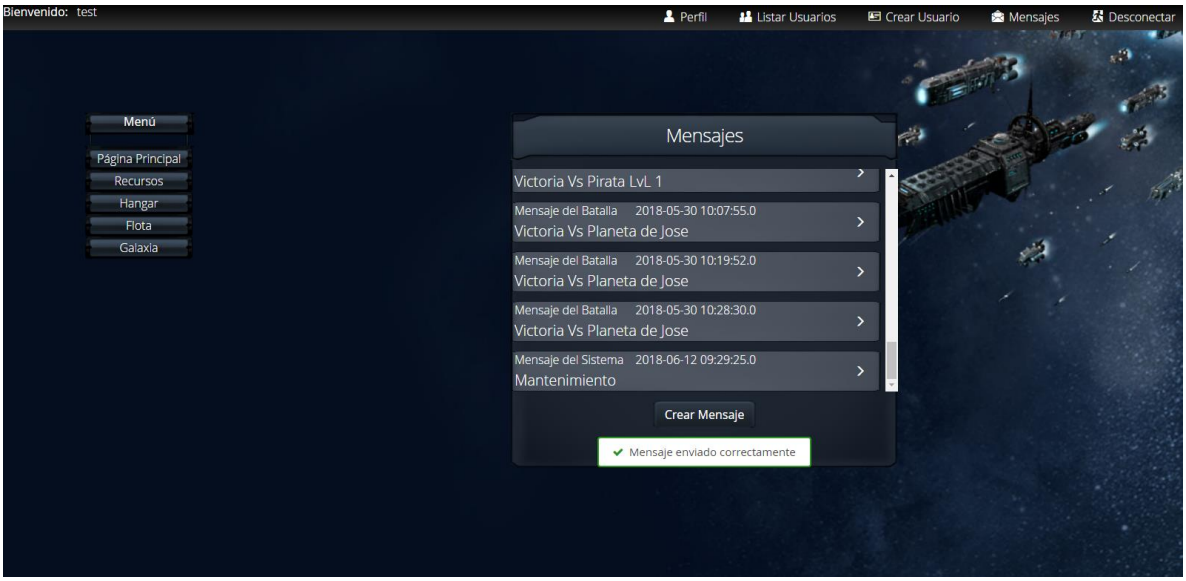

Figura 293. Ventana de crear mensaje correcto (casos prácticos). (Elaboración propia)

4.3. Cambios en la jugabilidad

A continuación tomaremos el rol de un técnico, primero realizaremos la creación de una nueva nave.

Figura 294. Pantalla principal técnico (casos prácticos). (Elaboración propia)

Nos vamos a la sección de crear nave y comenzamos a rellenar los datos.

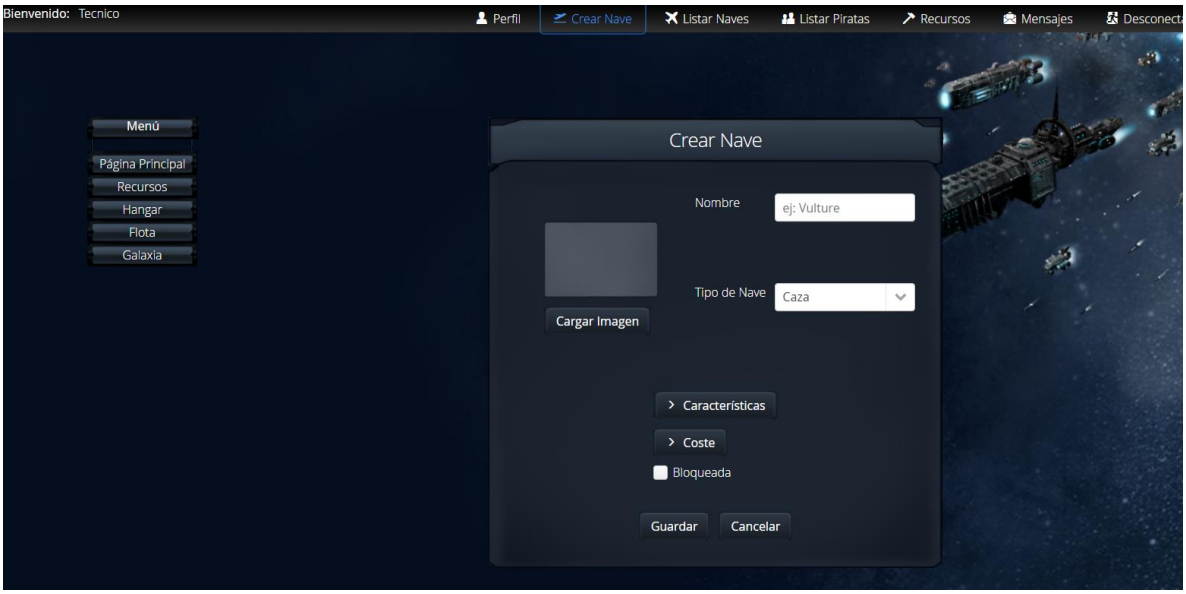

Figura 295. Ventana de crear nave (casos prácticos). (Elaboración propia)

Esta será la primera nave de tipo transporte de la aplicación.

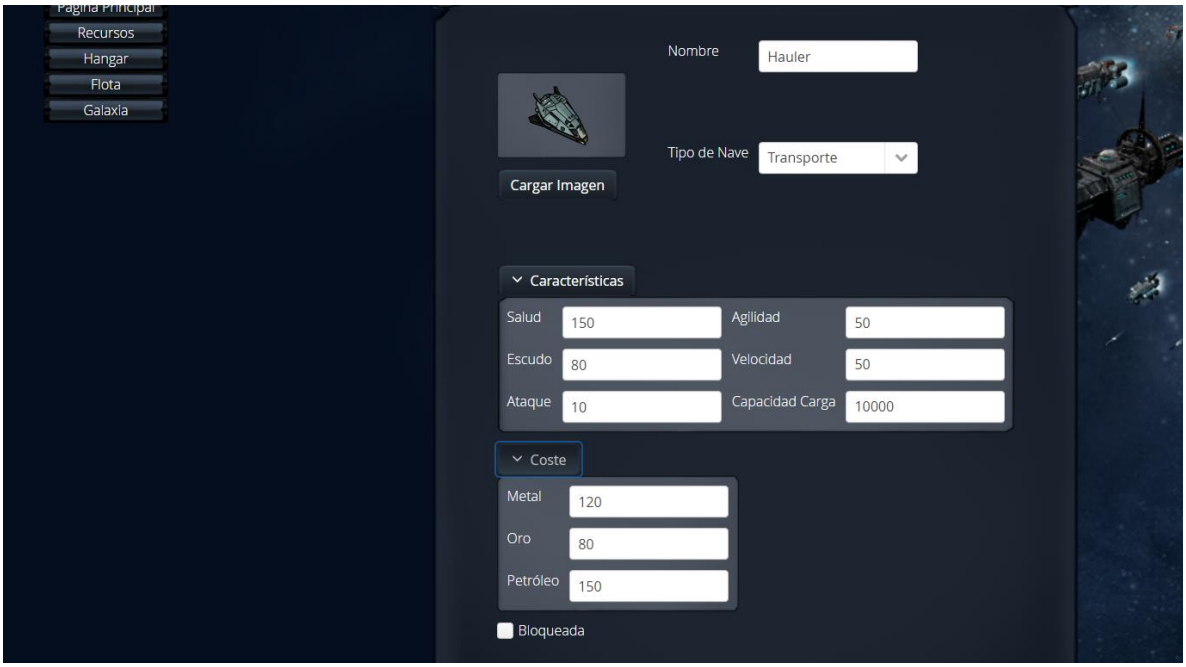

Figura 296. Ventana de crear nave características y coste (casos prácticos). (Elaboración propia)

Una vez establecidas las características decidimos que sea una nave que se deba de desbloquear, como no es muy poderosa será muy fácil de conseguir.

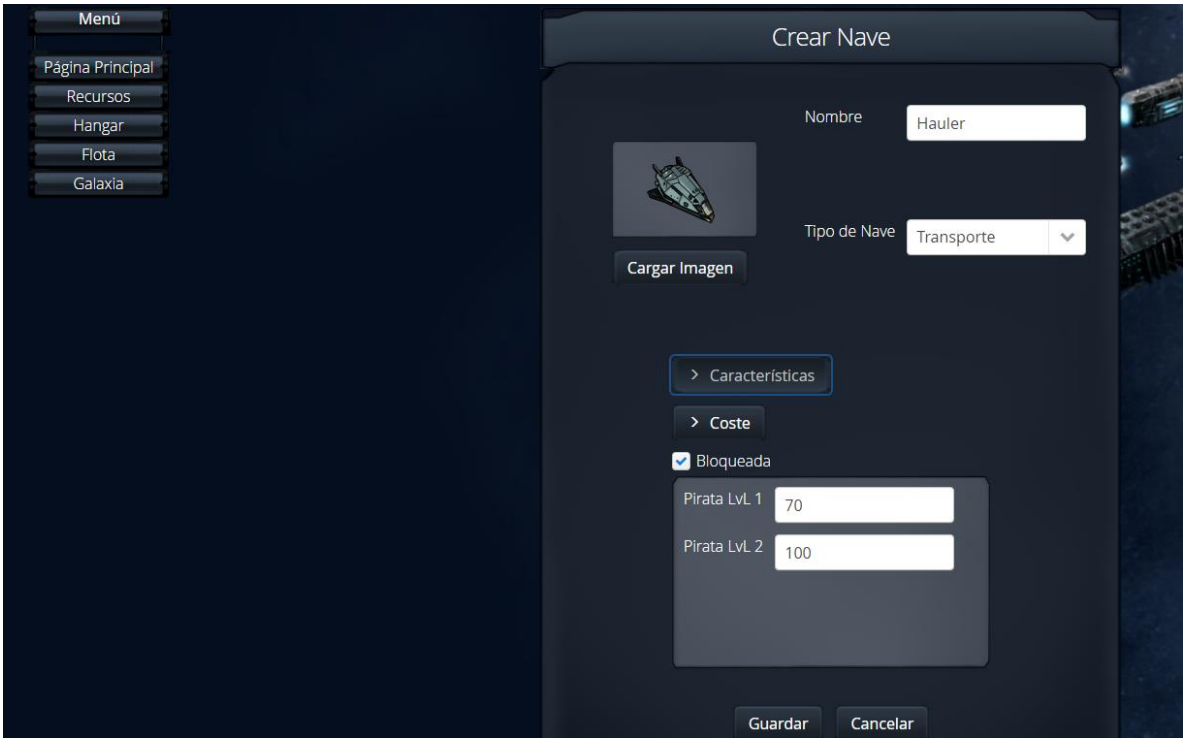

Figura 297. Ventana de crear nave bloqueada (casos prácticos). (Elaboración propia)

Guardamos y comprobamos que se ha creado correctamente.

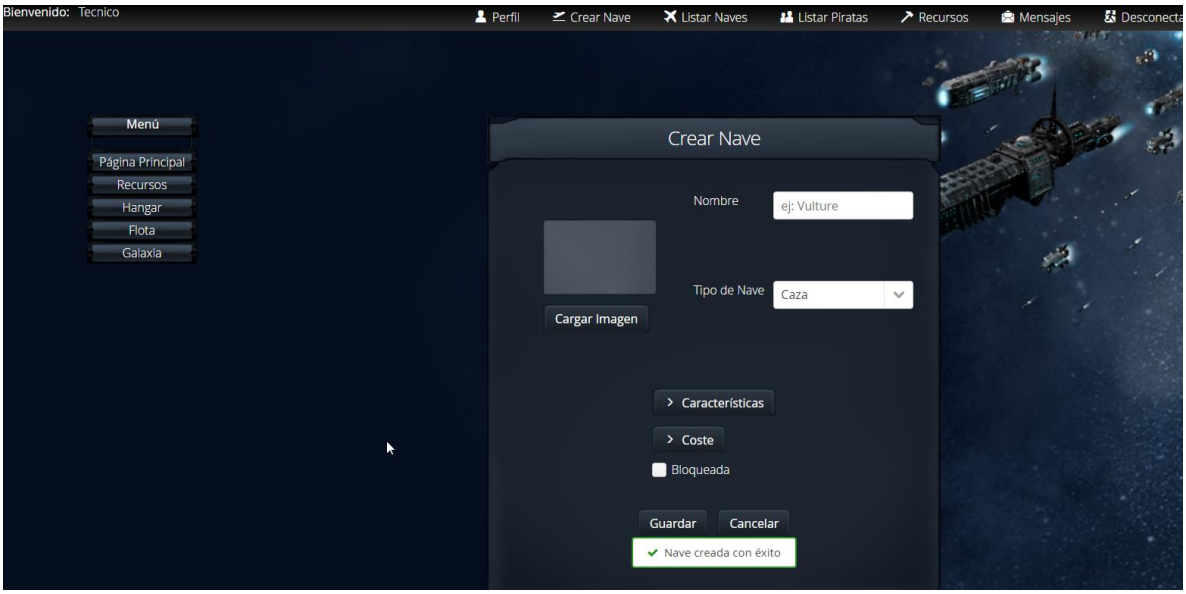

Figura 298. Ventana de crear nave correcta (casos prácticos). (Elaboración propia)

Ahora vamos a modificar el pirata de nivel 1, da demasiados recursos y tiene una defensa muy débil.

Figura 299. Ventana de pistado de piratas (casos prácticos). (Elaboración propia)

Estas son las instalaciones y naves del pirata.

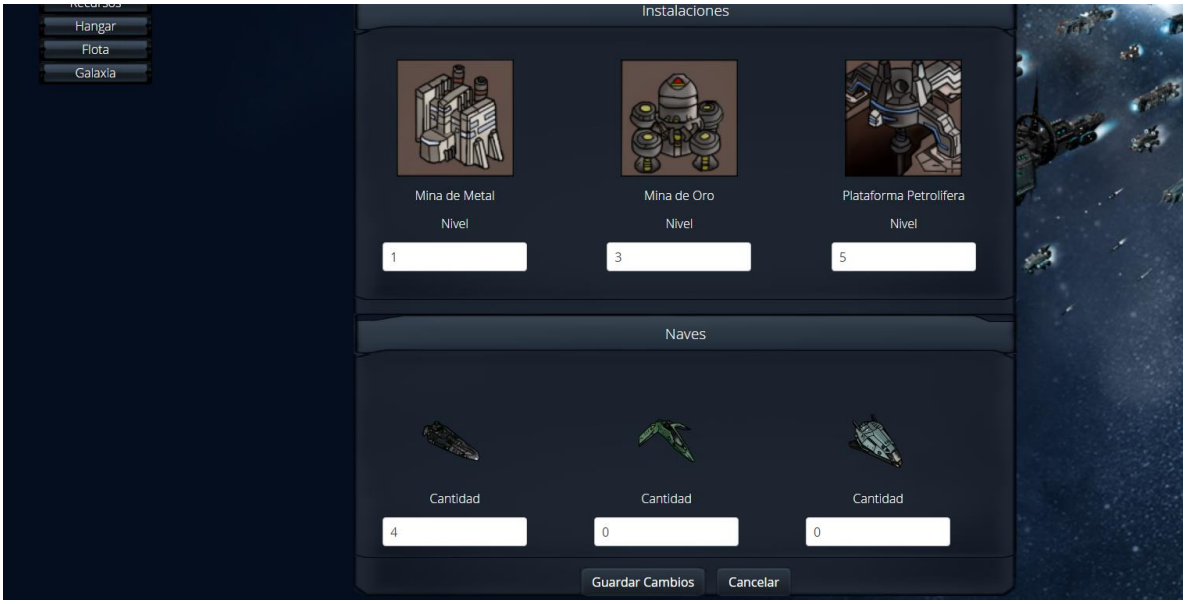

Figura 300. Ventana de editar pirata (casos prácticos). (Elaboración propia)

Le bajamos el nivel de la plataforma petrolífera y le ponemos más naves.

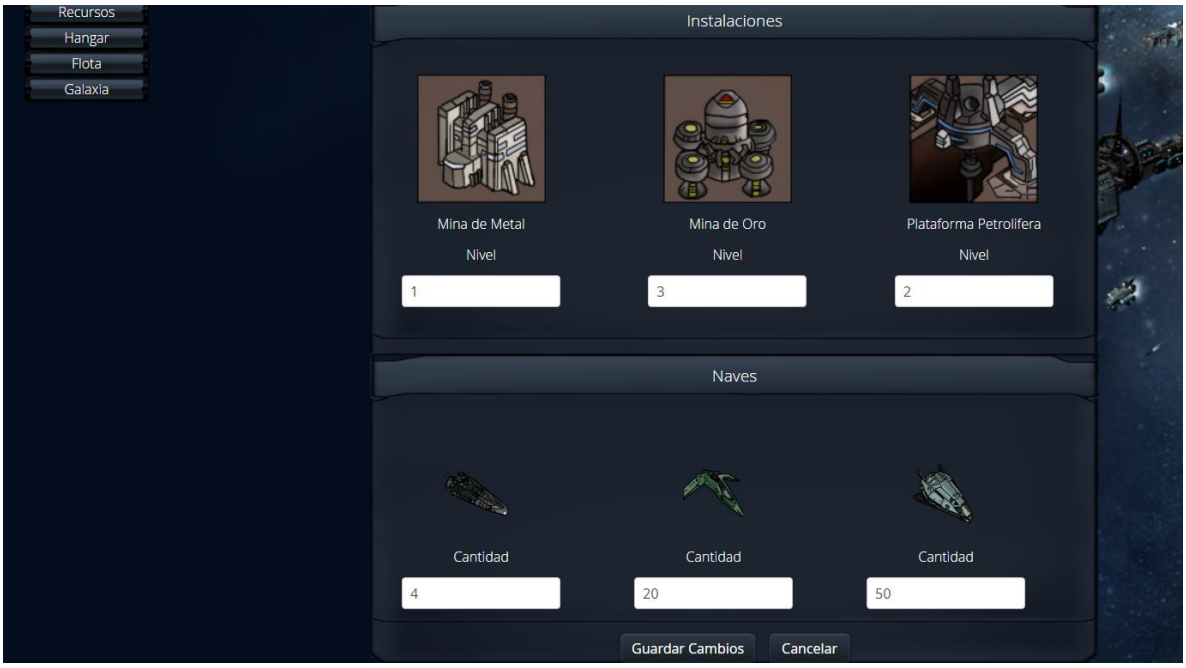

Figura 301. Ventana de editar pirata cambiado (casos prácticos). (Elaboración propia)

Guardamos y comprobamos que se han modificado sus datos.

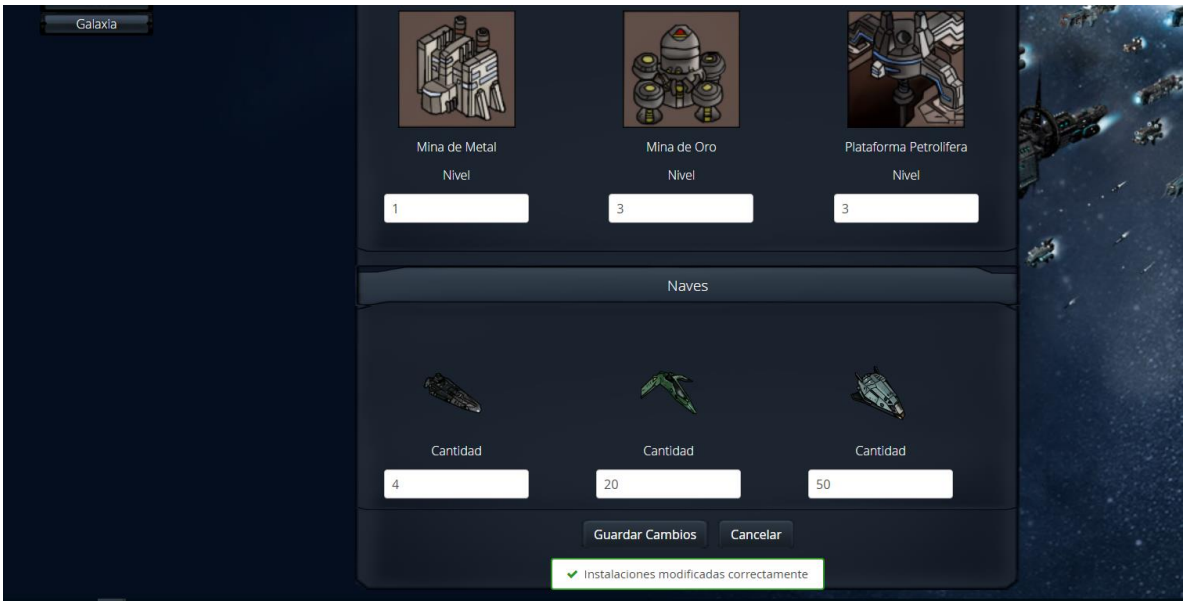

Figura 302. Ventana de editar pirata correcto (casos prácticos). (Elaboración propia)

Ya realizados los cambios vamos a avisar a los usuarios de cuales han sido.

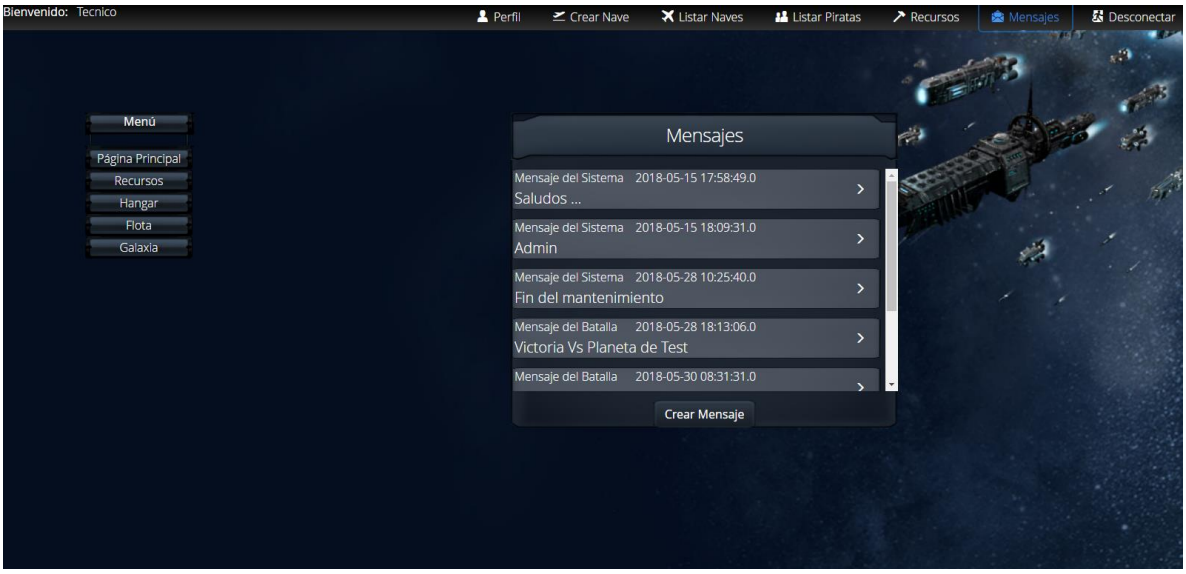

Figura 303. Ventana de mensajes (técnico) (casos prácticos). (Elaboración propia)

Redactamos el mensaje con los cambios y lo enviamos.

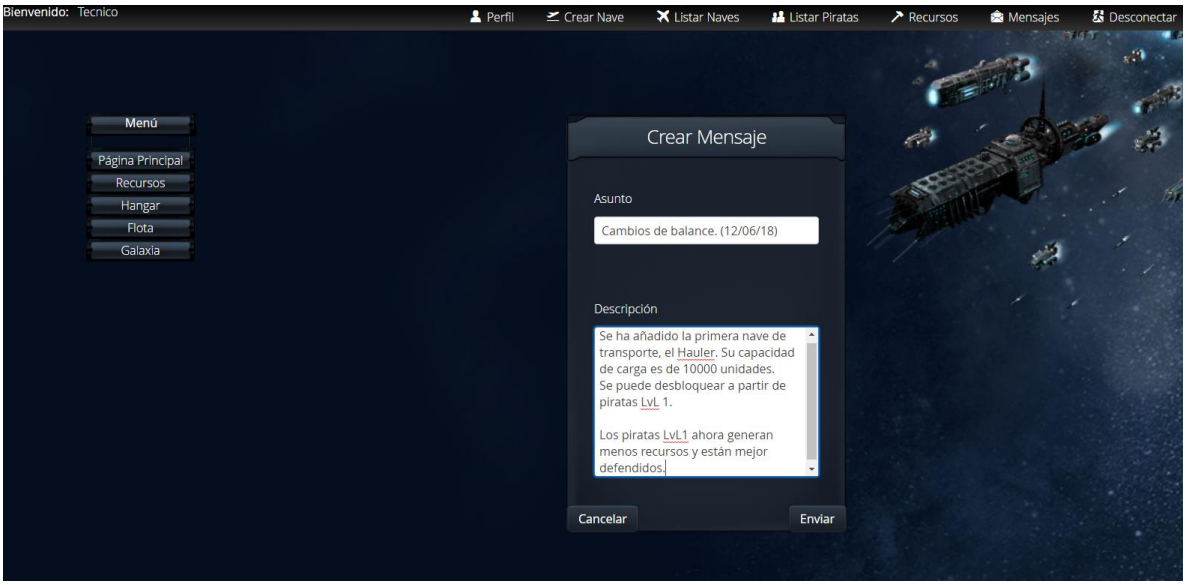

Figura 304. Ventana de crear mensaje (técnico) (casos prácticos). (Elaboración propia)

El mensaje ya nos sale en nuestro buzón por lo que se envió correctamente.

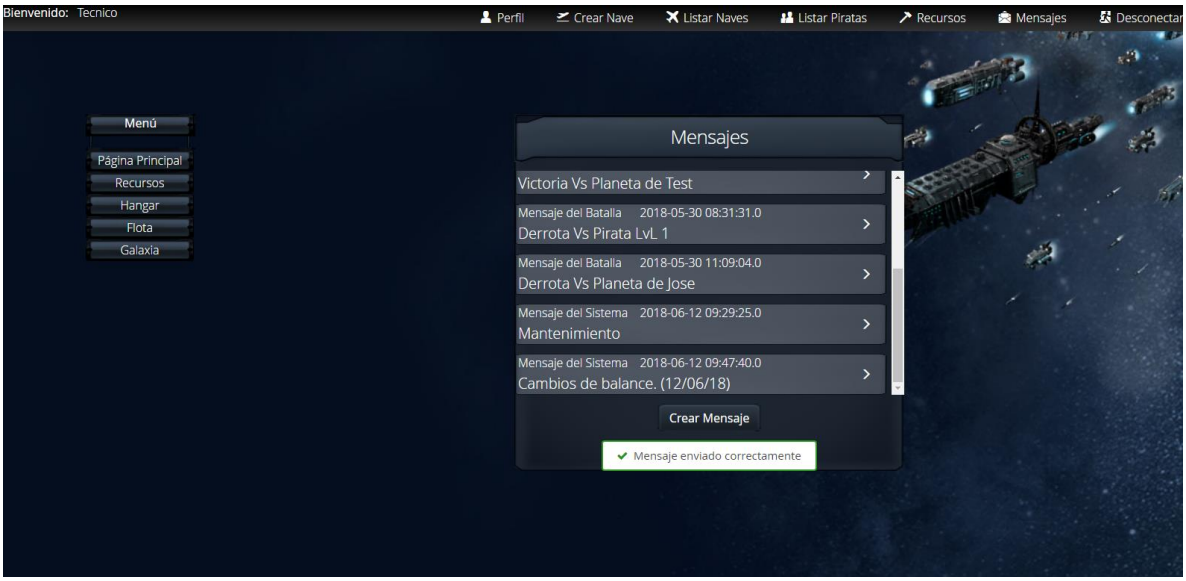

Figura 305. Ventana de crear mensaje correcto (técnico) (casos prácticos). (Elaboración propia)

4. Conclusiones

Se ha creado una metodología ágil basada en la modularización o división de la funcionalidad y esta metodología se ha utilizado para desarrollar un videojuego online basado en Ogame.

Tras utilizar esta metodología para el proyecto propuesto he identificado sus fortalezas y sus debilidades.

Sus puntos fuertes son el desarrollo basado en pruebas y lo que se buscaba, la modularización. Durante el desarrollo el hecho de realizar la aplicación a partir de las pruebas además de facilitar la refactorización ha ayudado en la identificación de errores y en la integración de todos sus módulos, gracias a esto en la integración simplemente había que ejecutar las pruebas para comprobar que todo el back-end había sido integrado con éxito.

La modularización ha permitido dividir un proyecto muy grande en proyectos pequeños, toda la funcionalidad se controlaba mucho mejor de esta manera, abrumaba menos y no resultaba tan monótono al ser dividido en partes con funcionalidades muy distintas.

También simplificó la documentación, no era necesario estar varios días documentando, al estar la documentación dividida solo era necesario dedicarle un poco de tiempo.

En cuanto a sus debilidades cabe destacar la identificación de los módulos y la estimación de sus tiempos. Es una metodología para equipos con poca experiencia y estos dos aspectos requieren de mucha experiencia, yo mismo lo he vivido, tras terminar el proyecto he identificado que el módulo de inicio de sesión debería de haber sido el último ya que al integrar faltaban funcionalidades en el registro.

También sucedió que terminaba un módulo y a lo mejor me sobraba media semana, si, esos días restantes pueden utilizarse para pulirlo, pero si el tiempo va justo lo normal es aprovecharlos y empezar el siguiente módulo.

Por último el *feedback*, hay veces que es difícil encontrar un grupo de personas que te lo proporcionen, sobre todo en fases muy tempranas o en módulos demasiado simples, en mi caso opté por gente cercana a mí para que me proporcionasen ese *feedback*, aunque lo normal es que pruebe la aplicación el público al que va dirigido.

Con esto cabe decir que la metodología funciona, una metodología siempre va a tener debilidades, aunque habría que pulirlas para que afecten lo menos posible. Dejar un poco más de flexibilidad en los tiempos, dar consejos para obtener *feedback* y encontrar una manera de facilitar dentro de lo posible la estimación y división por módulos.

5. Bibliografía

1. PwC. [Online].; 2017 [visitado 2018 Febrero 5]. URL: https://www.pwc.es/es/sala-prensa/notas-prensa/2017/mercado-de-videojuegos-crecera-hasta-2021.html.
2. Newzoo. [Online]. [visitado 2018 Abril 5]. URL: https://newzoo.com/insights/articles/new-gaming-boom-newzoo-ups-its-2017-global-games-market-estimate-to-116-0bn-growing-to-143-5bn-in-2020/.
3. LLORENTE & CUENCA. El sector de los videojuegos en España: impacto económico y escenarios fiscales. Análisis Económico. ; 2018.
4. DEV. Libro Blanco del desarrollo español de videojuegos. ; 2017.
5. Rufí JPP. El Imperio Indie del Videojuego Español: Debilidades, Amenazas, Fortalezas y Oportunidades de la Industria Española Del Software de Ocio. Razón y Palabra. 2016 Mayo; 20(2_93): p. 839-852.
6. Zaytsev O. IGN. [Online].; 2014 [visitado 2018 Abril 5]. URL: http://es.ign.com/unrest-mac/82154/feature/que-es-un-juego-independiente.
7. Dutton F. Eurogamer. [Online].; 2012 [visitado 2018 Abril 5]. URL: https://www.eurogamer.net/articles/2012-04-16-what-is-indie.
8. RAE. [Online]. [visitado 2018 Abril 5]. URL: http://dle.rae.es/srv/search?m=30&w=independiente.
9. McMillen E. Gamasutra. [Online].; 2012 [visitado 2018 Abril 6]. URL: https://www.gamasutra.com/view/feature/182380/postmortem_mcmillen_and_himsls_.php.
10. Zaytsev O. IGN. [Online].; 2014 [visitado 2018 Abril 6]. URL: http://es.ign.com/indie/85414/feature/que-es-un-juego-independiente-y-2.
11. Kovanto A. The Improvements for Indie Game Development. Bachelor's Thesis. Joensuu: Karelia University of Applied Sciences; 2013.
12. Bendilas D. Game Software Lifecycle for Mobile Devices A Case Study on iOS. Trabajo Fin de Master. Thessaloniki: Universidad de Macedonia, Departamento de informática aplicada; 2012.
13. Acerenza N, Coppes A, Mesa G, Viera A, Fernández Albano E, Laurenzo T, et al. Una Metodología para desarrollo de videojuegos. Simposio Argentino de Ingeniería de Software. 2009 Agosto;: p. 171-176.

14. GmbH G. Es.ogame.gameforge.com. [Online]. [visitado 2018 Febrero 5]. URL: https://es.ogame.gameforge.com/.

15. Gonzalez Gill JA, Perez E. Una propuesta metodológica para la construcción de videojuegos. Memorias de Congresos UTP. 2016 Diciembre; 1(1): p. 64-70.

16. Stuart K. The digital apocalypse: how the games industry is rising again. The Guardian. 2016 Mayo: p. 1.

17. Armendáriz Moreno GA, Saltos Guaraca MG. Barreno, A., Alexander, G., Guaraca, S., & Gonzalo, M. (2013). Adaptación de las metodologías ágiles Scrum y Extreme Game Development en una metodología para el desarrollo de videojuegos en Android. Caso práctico: Desarrollo de un videojuego. Tesis de grado. Riobamba: Escuela Superior Politécnica de Chimborazo, Facultad de informática y electrónica; 2013.

18. Cohen D, Lindvall M, Costa P. Agile software development. Nueva York: Fraunhofer Center Maryland and University of Maryland; 2003.

19. Keith C. Agile game development with Scrum: Pearson Education; 2010.

20. McGuire R. Gamasutra. [Online].; 2006 [visitado 2018 Febrero 9]. URL: https://www.gamasutra.com/view/feature/131151/paper_burns_game_design_with_.php?page=2.

21. Pereira AMM. El proceso productivo del videojuego: fases de producción. Historia y Comunicación Social. 2014 Marzo; 19: p. 791-805.

22. Murillo O. Sistema de Seguimiento Academico Escola "Colegio Ave Maria". [Online].; 2012 [visitado 2018 Abril 6]. URL: https://sistemacademicoescolaravemaria.wordpress.com/about/pagina-3/.

23. Schofield B. Gamasutra. [Online].; 2007 [visitado 2018 Abril 6]. URL: https://www.gamasutra.com/view/feature/130120/embracing_fun_why_extreme_.php?page=2.

24. Beck K. Extreme Programming Explained: Embrace Change. Segunda ed. Addison-Wesley , editor.; 2004.

25. Wake WC. Extreme Programming Explored Xp , editor.; 2001.

26. Penadés MC, Letelier Torres PO. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Técnica administrativa. 2006; 5(26).

27. Schwaber K, Sutherland J. La Guía de Scrum. ; 2016.

28. Menzinsky A, López G, Palacio J. Guía de Scrum Manager; 2006.

29. Kniberg H. Scrum y XP desde las trincheras C4Media , editor.; 2007.

30. Kroll P, Kruchten P, Booch G. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP Professional AW, editor.; 2003.

31. staff I. IBM. [Online].; 2005 [visitado 2018 Abril 6]. URL: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

32. Procesos de Software. [Online].; 2018 [visitado 2018 Abril 6]. URL: https://procesosdesoftware.wikispaces.com/METODOLOGIA+RUP.

33. Dingsøyr T, Nerur S, Balijepally V, Moe NB. A decade of agile methodologies: Towards explaining agile software development. The Journal of Systems and Software. 2012; 85: p. 1213-1221.

34. McConnell S. Rapid Development: Taming Wild Software Schedules Press M, editor.; 1996.

35. Petersen K, Wohlin C, Baca D. The waterfall model in large-scale development. International Conference on Product-Focused Software Process Improvement. 2009 Junio; 32: p. 386-400.

36. Demachy T. Gamasutra. [Online].; 2003 [visitado 2018 Abril 6]. URL: https://www.gamasutra.com/view/feature/131236/extreme_game_development_right_on_.php?page=1.

37. Godoy A, Barbosa E. Game-Scrum: An Approach to Agile Game Development. Proceedings of SBGames. 2010;: p. 292-295.

38. Flood K. Gamedev. [Online].; 2003 [visitado 2018 Abril 6]. URL: https://www.gamedev.net/articles/programming/general-and-gameplay-programming/game-unified-process-r1940/.

39. Jones W. Case Tools - Aids for Systems Development. [Online].; 2002 [visitado 2018 Febrero 16]. URL: http://www.umsl.edu/~sauterv/analysis/488_f02_papers/CASE.html.

40. Eclipse. [Online]. [visitado 2018 Febrero 17]. URL: https://www.eclipse.org/.

41. Vaadin.com. [Online]. [visitado 2018 Febrero 5]. URL: https://vaadin.com/.

42. Spring. [Online]. [visitado 2018 Febrero 17]. URL: https://spring.io/.

43. Junit.org. [Online]. [visitado 2018 Febrero 17]. URL: https://junit.org/junit5/.

44. NetBeans. [Online]. [visitado 2018 Abril 10]. URL: https://netbeans.org/.

45. EGit. [Online]. [visitado 2018 Abril 4]. URL: https://projects.eclipse.org/projects/technology.egit.

46. Github. [Online]. [visitado 2018 Febrero 17]. URL: https://github.com/.

47. Mysql.com. [Online]. [visitado 2018 Febrero 17]. URL: https://www.mysql.com/products/workbench/.

48. Apachefriends. [Online]. [visitado 2018 Febrero 17]. URL: https://www.apachefriends.org/es/index.html.

49. Jenkins. [Online]. [visitado 2018 Febrero 17]. URL: https://jenkins.io/.

50. Fiware Lab. [Online]. [visitado 2018 Abril 10]. URL: https://account.lab.fiware.org/.

51. Systemax.jp. [Online]. [visitado 2018 Febrero 17]. URL: https://www.systemax.jp/en/sai/.

52. Lsi.us.es. [Online]. [visitado 2018 Febrero 17]. URL: http://www.lsi.us.es/descargas/descarga_programas.php?id=3.

53. CSSE Website. [Online]. [visitado 2018 Febrero 17]. URL: http://csse.usc.edu/csse/research/cocomoii/cocomo_downloads.htm.

54. Visual-paradigm.com. [Online]. [visitado 2018 Febrero 17]. URL: https://www.visual-paradigm.com/.

55. Icons8. [Online]. [visitado 2018 Mayo 28]. URL: https://icons8.com.

6. Anexo

En esta sección se mostrarán los diagramas de caso de uso, de clase de la interfaz y de la base de datos de los módulos a una mejor resolución. También se han incluido todas las ventanas de la interfaz y las imágenes utilizadas en la aplicación.

6.3. Diagramas de Casos de Uso

A continuación se muestran los diagramas de casos de uso de todos los módulos realizados.

6.3.4. Módulo Inicio de Sesión

Figura 306. Diagrama de casos de uso del módulo Inicio de Sesión (Cibernauta). (Elaboración propia)

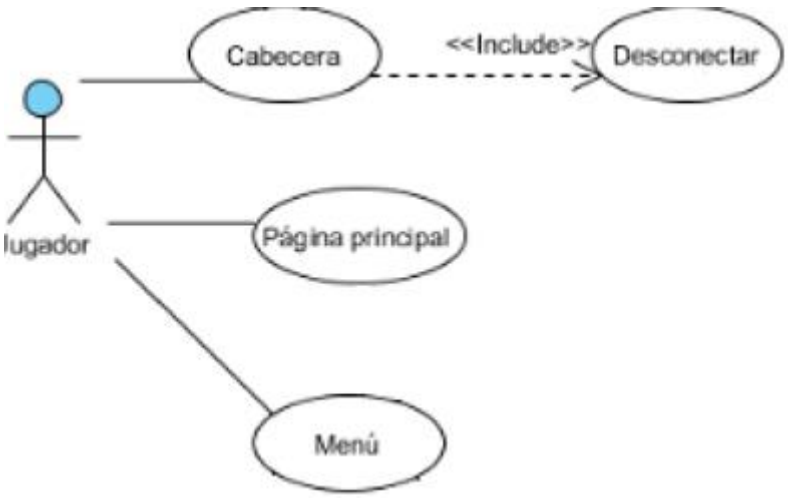

Figura 307. Diagrama de casos de uso del módulo Inicio de Sesión (Jugador). (Elaboración propia)

6.3.5. Módulo Usuarios

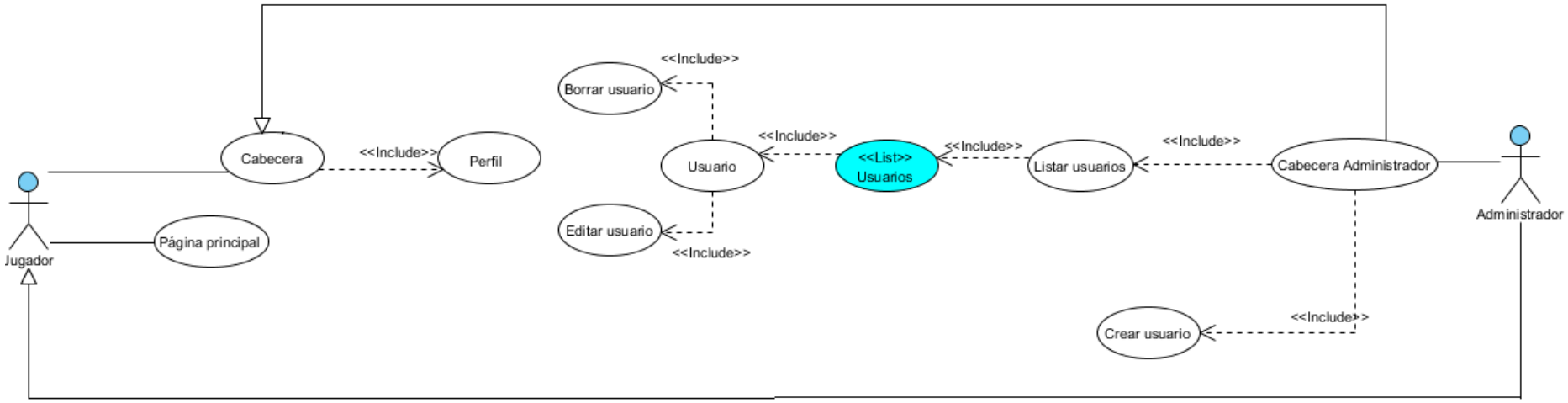

Figura 308. Diagrama de casos de uso del módulo de usuarios. (Elaboración propia)

6.3.6. Módulo Técnico

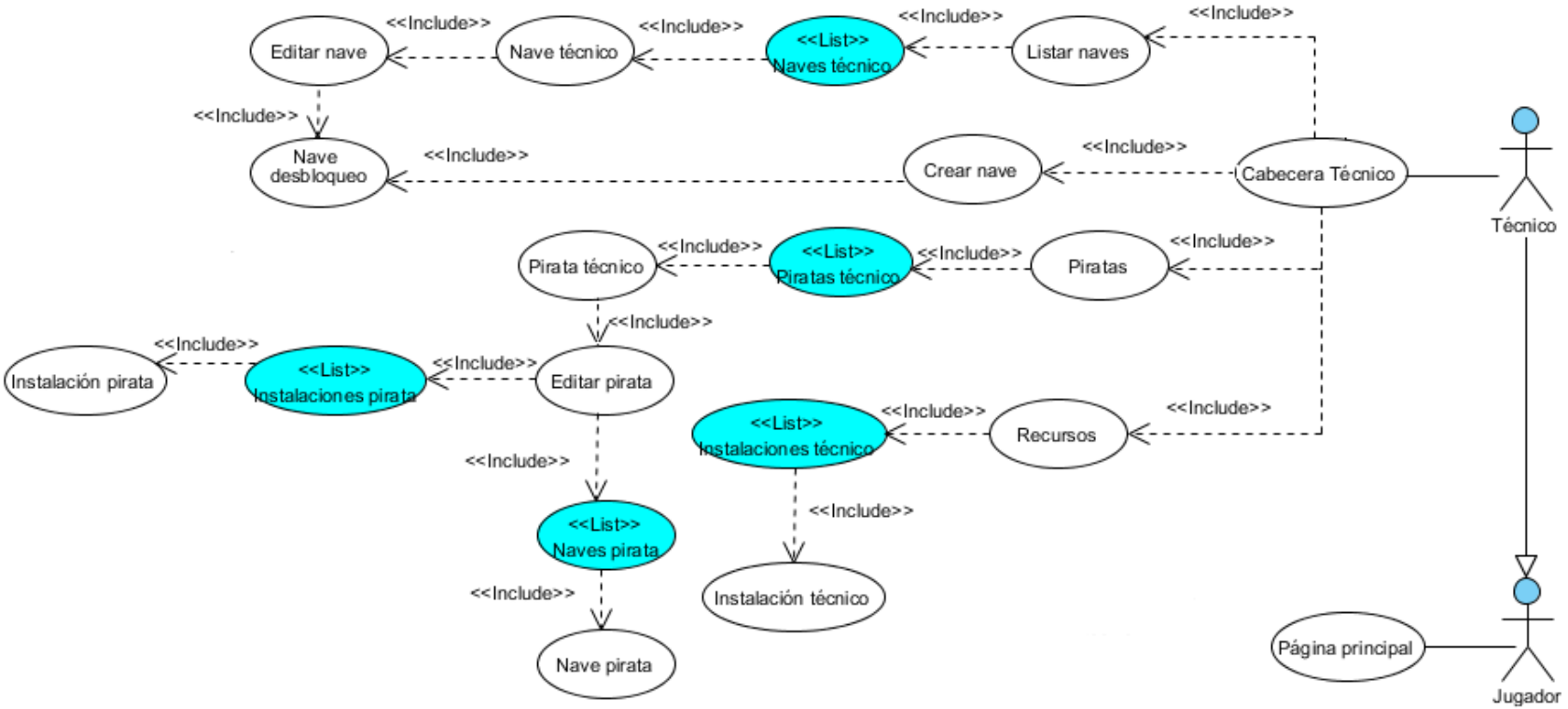

Figura 309. Diagrama de casos de uso del módulo técnico. (Elaboración propia)

6.3.7. Módulo Naves

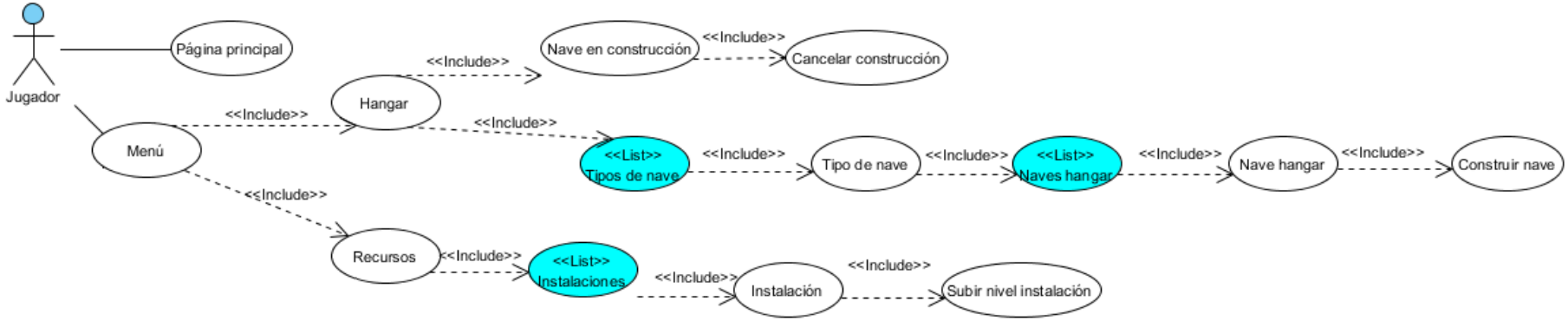

Figura 310. Diagrama de casos de uso del módulo naves. (Elaboración propia)

6.3.8. Módulo Información

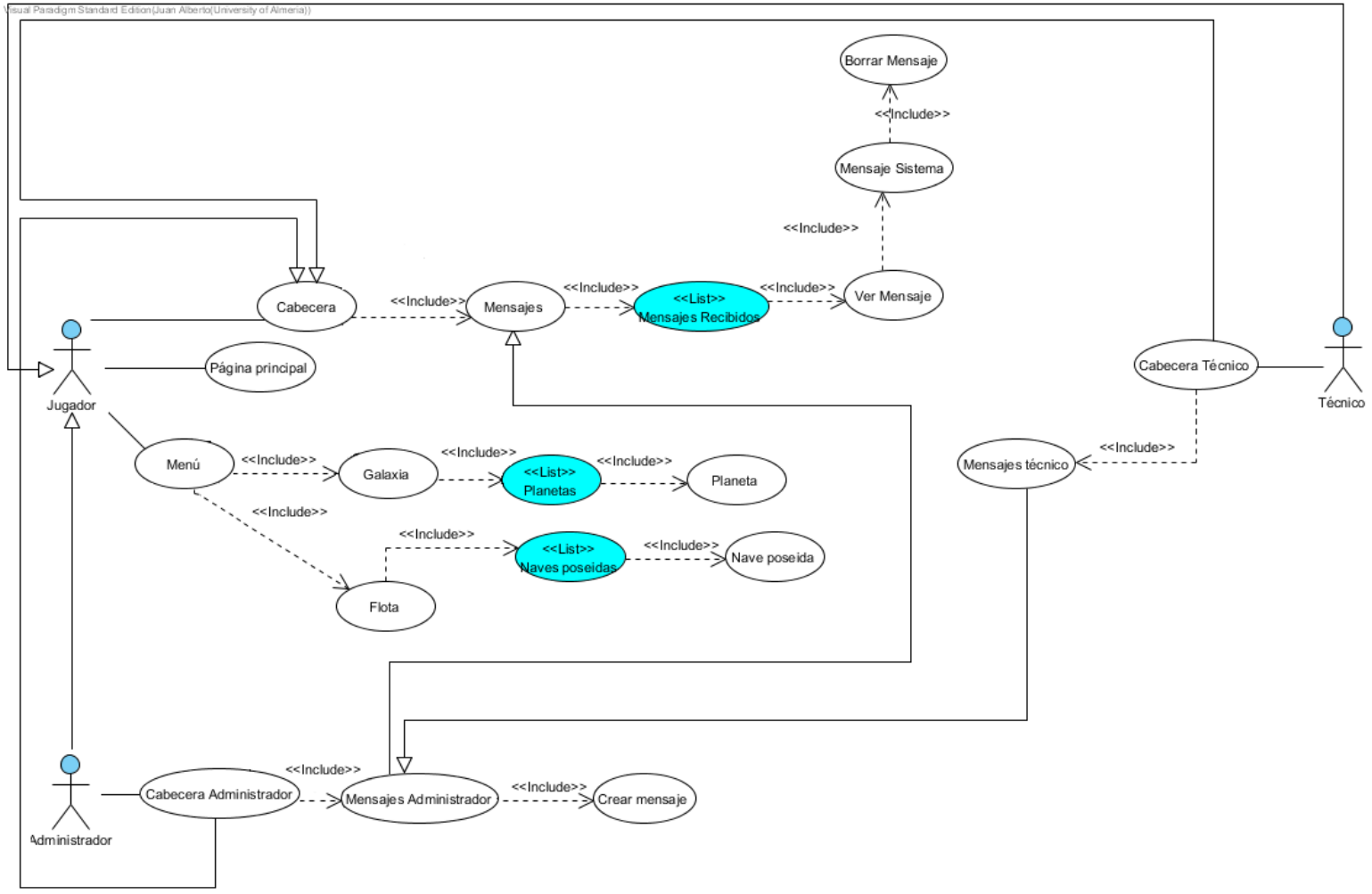

Figura 311. Diagrama de casos de uso del módulo información. (Elaboración propia)

6.3.9. Módulo Ataque

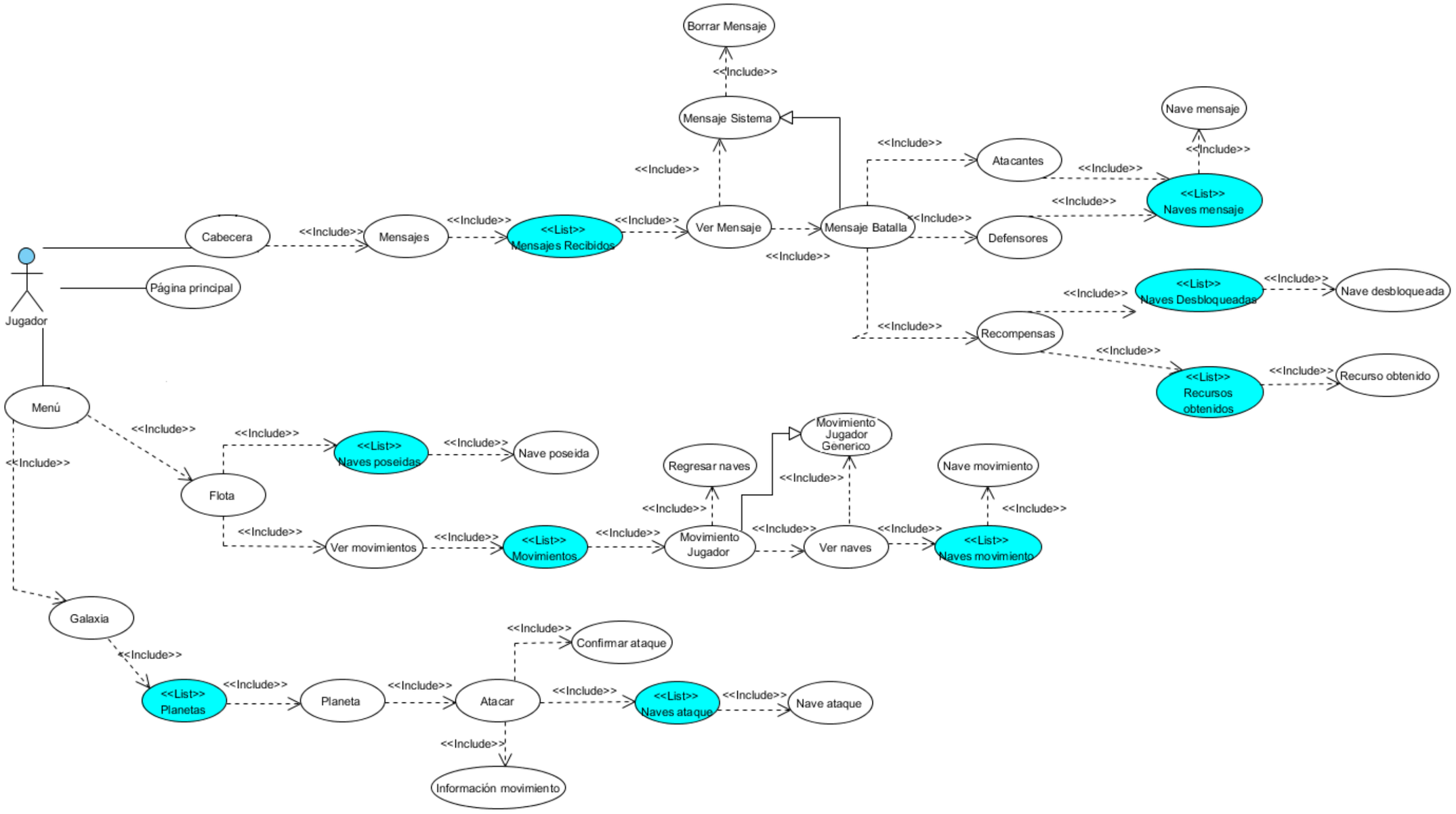

Figura 312. Diagrama de casos de uso del módulo ataque. (Elaboración propia)

6.3.10. Módulo Integración

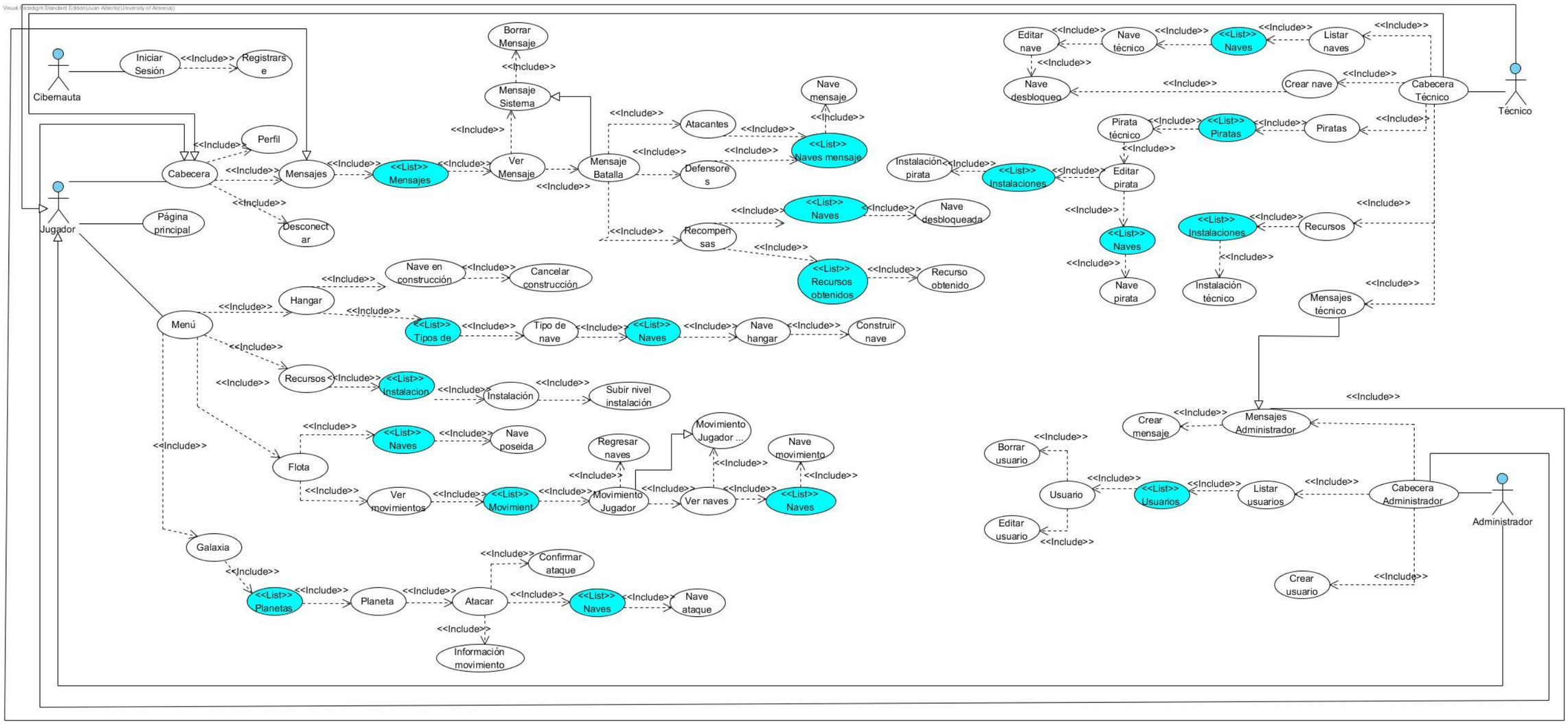

Figura 313. Diagrama de casos de uso del módulo integración. (Elaboración propia)

6.4. Diagramas de Clases

A continuación se muestran los diagramas de clases de la interfaz de todos los módulos realizados.

6.4.4. Módulo Inicio de Sesión

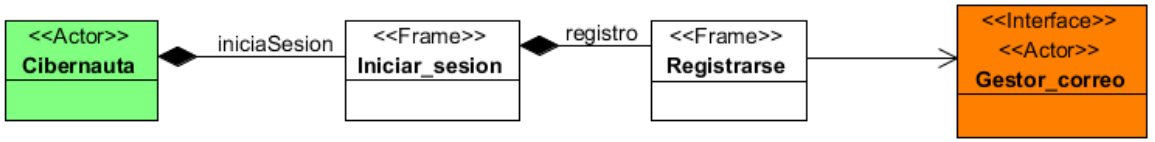

Figura 314. Diagrama de clases del módulo de Inicio de Sesión (Cibernauta). (Elaboración propia)

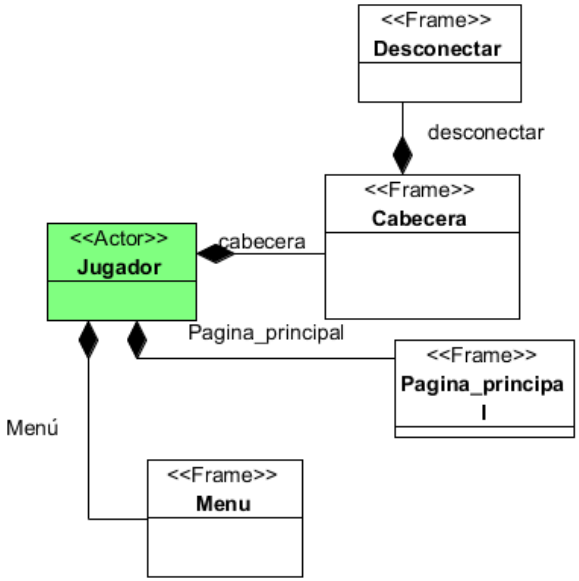

Figura 315. Diagrama de clases del módulo de Inicio de Sesión (Jugador). (Elaboración propia)

6.4.5. Módulo Usuarios

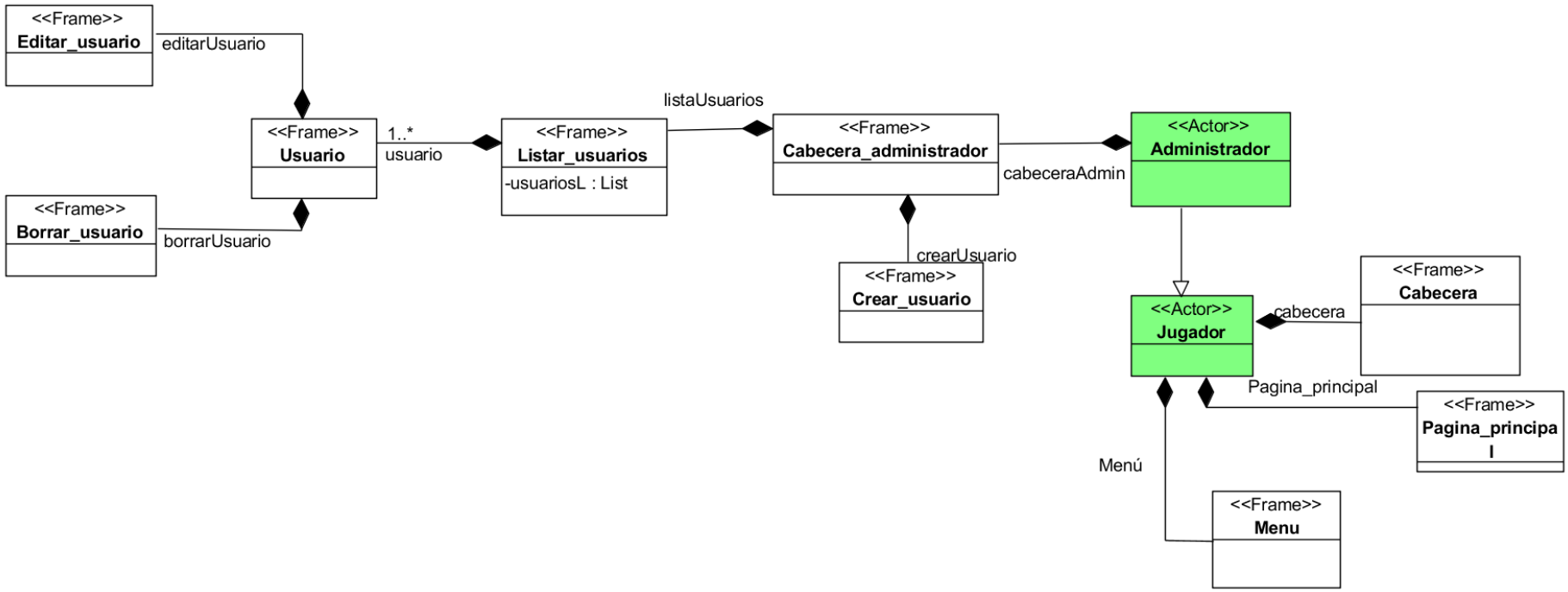

Figura 316. Diagrama de clases del módulo de usuarios. (Elaboración propia)

6.4.6. Módulo Técnico

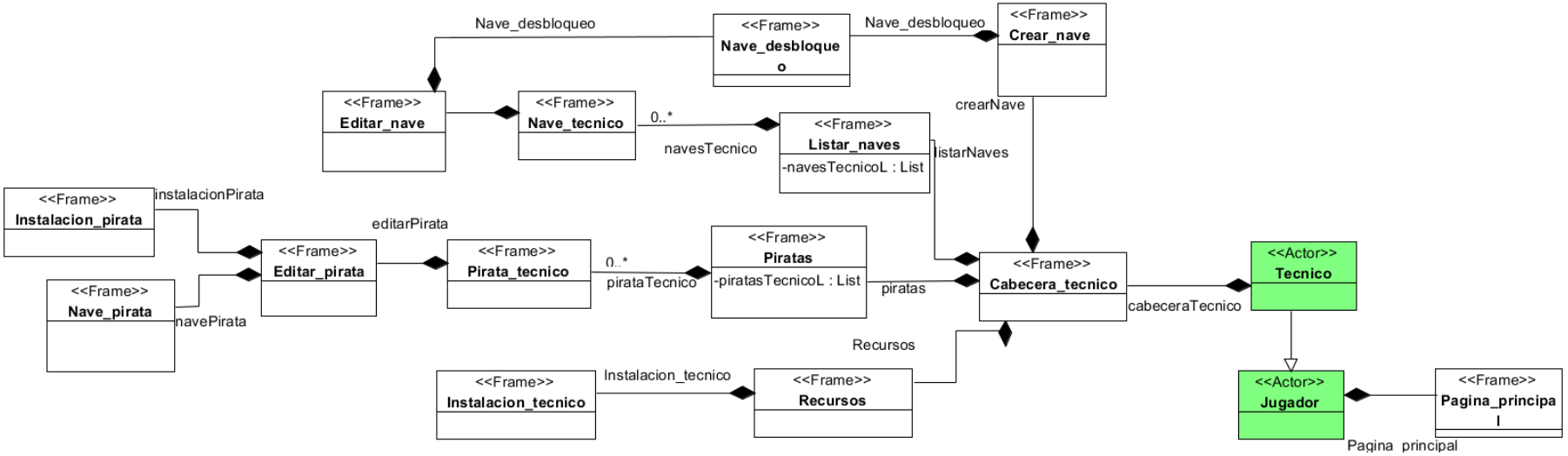

Figura 317. Diagrama de clases del módulo técnico. (Elaboración propia)

6.4.7. Módulo Naves

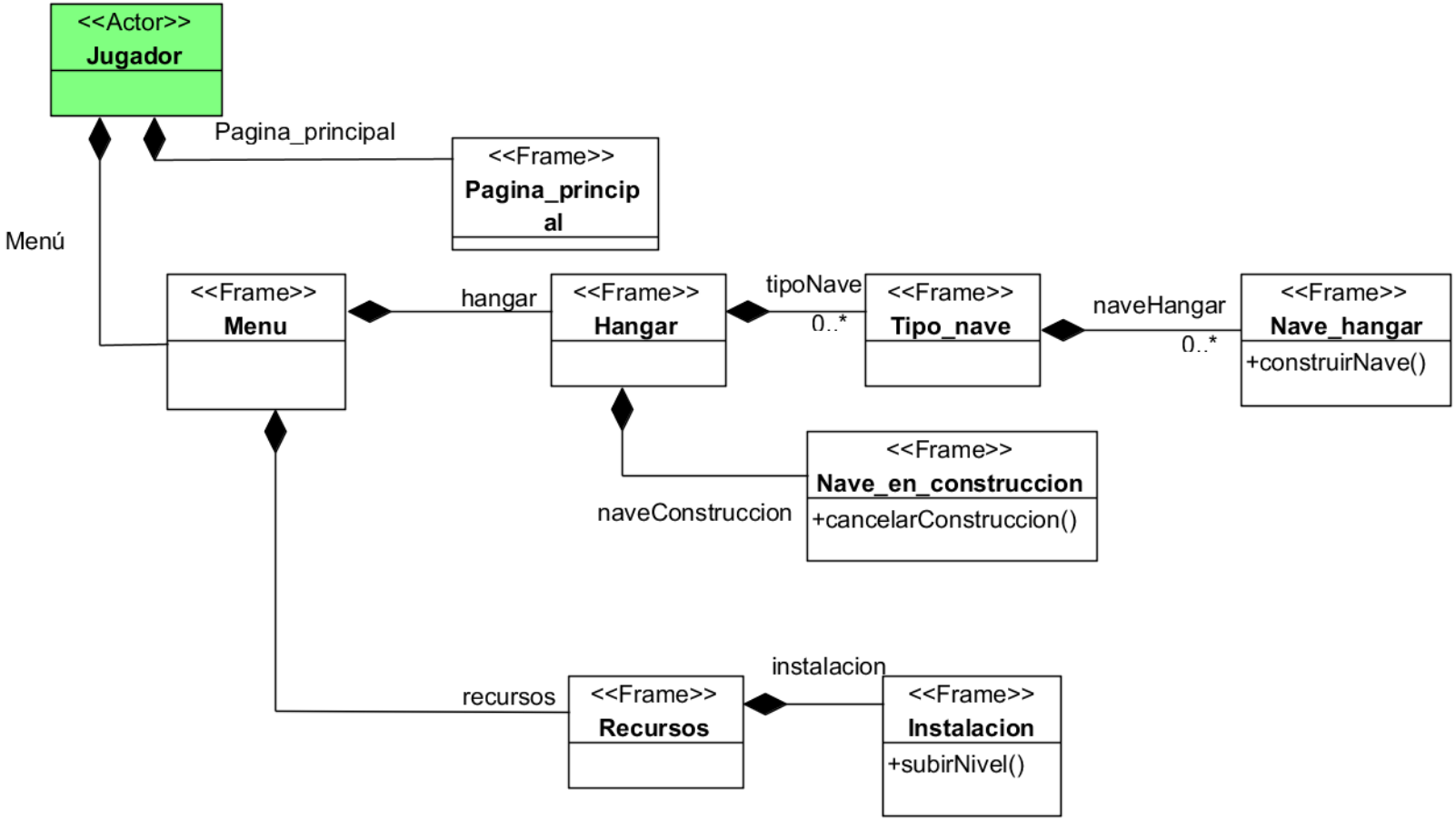

Figura 318. Diagrama de clases del módulo naves. (Elaboración propia)

6.4.8. Módulo Información

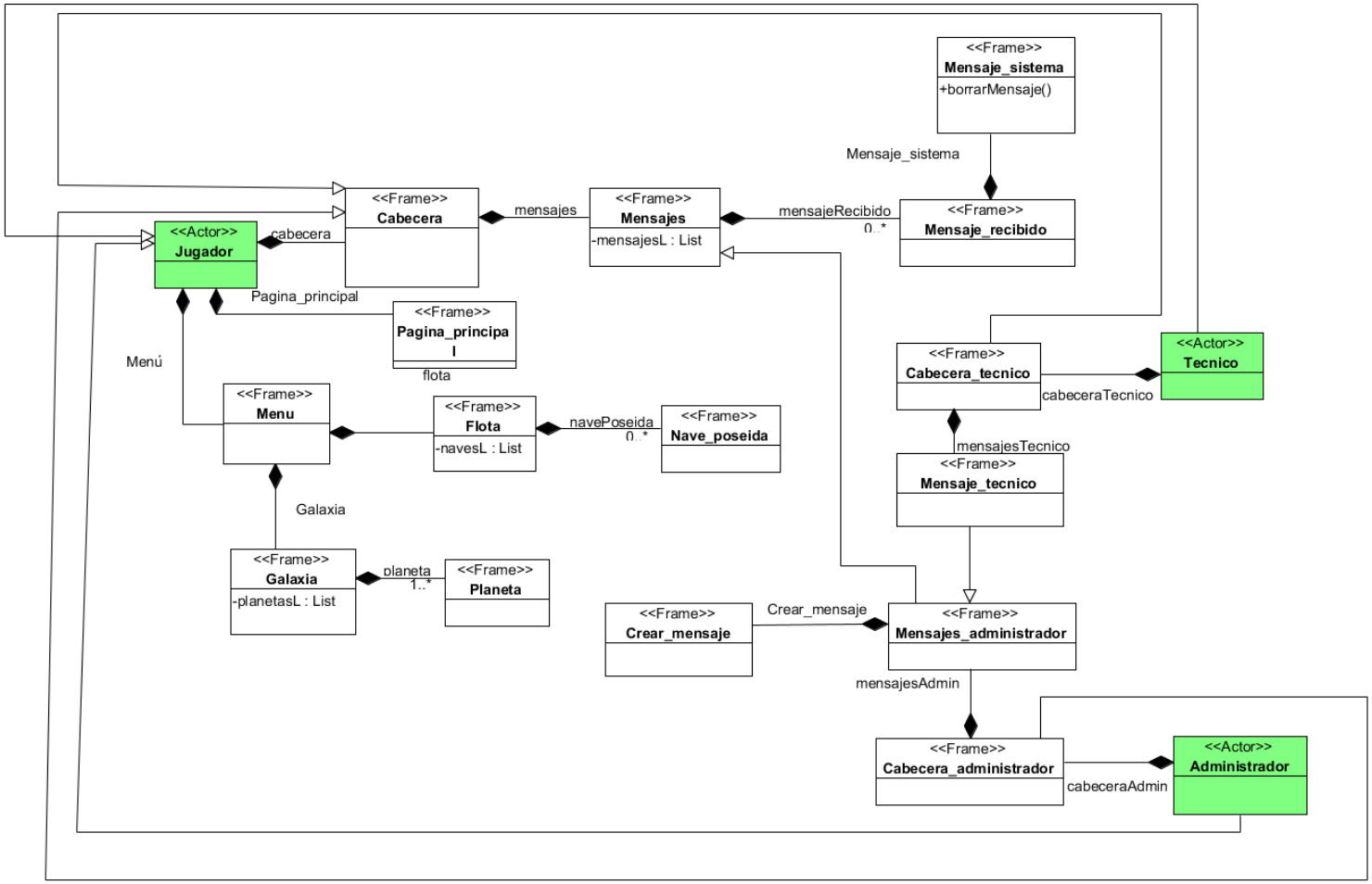

Figura 319. Diagrama de clases del módulo información. (Elaboración propia)

6.4.9. Módulo Ataque

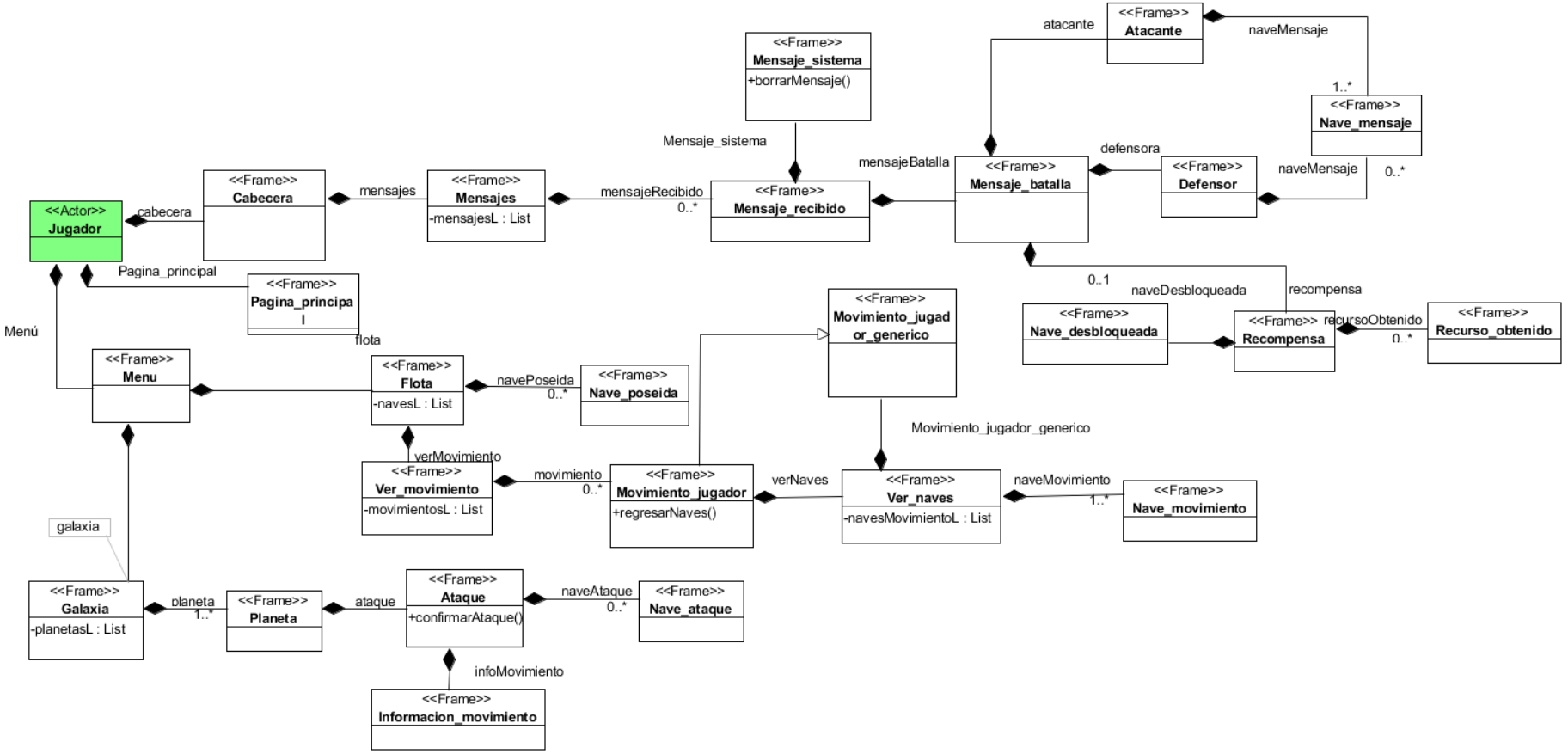

Figura 320. Diagrama de clases del módulo ataque. (Elaboración propia)

6.4.10. Módulo Integración

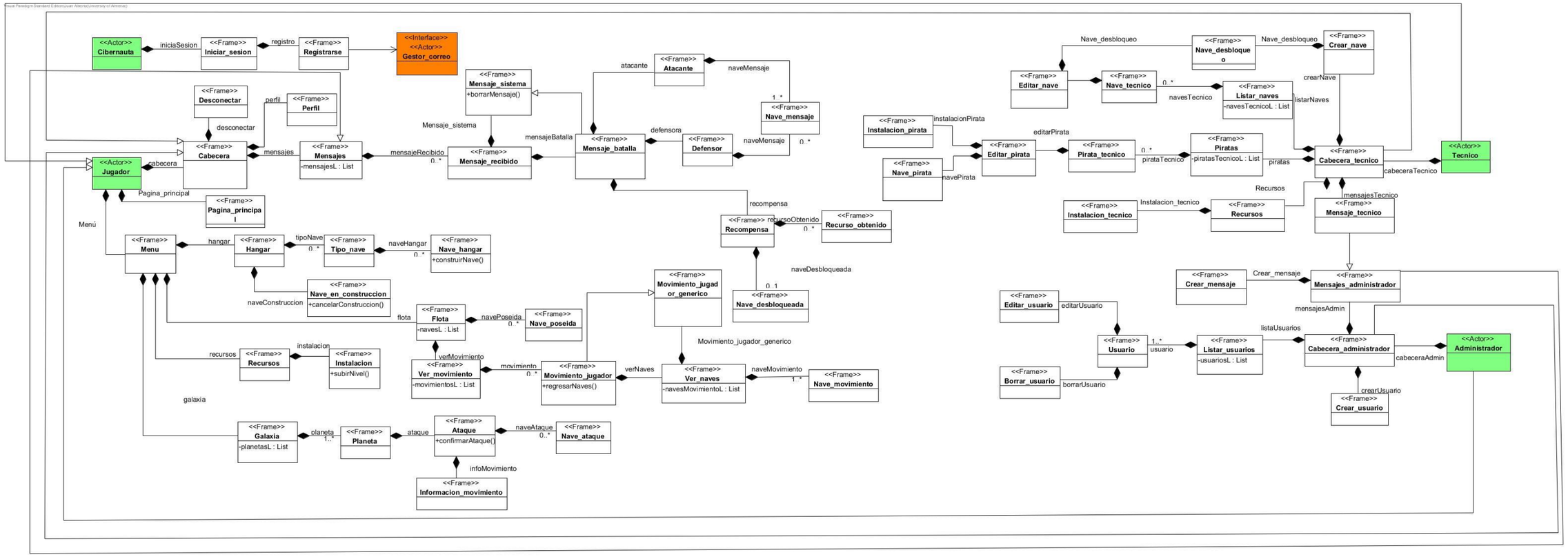

Figura 321. Diagrama de clases del módulo integración. (Elaboración propia)

6.5. Diagramas de la Base de Datos

A continuación se muestran los diagramas de la base de datos de todos los módulos realizados.

6.5.4. Módulo Inicio de Sesión

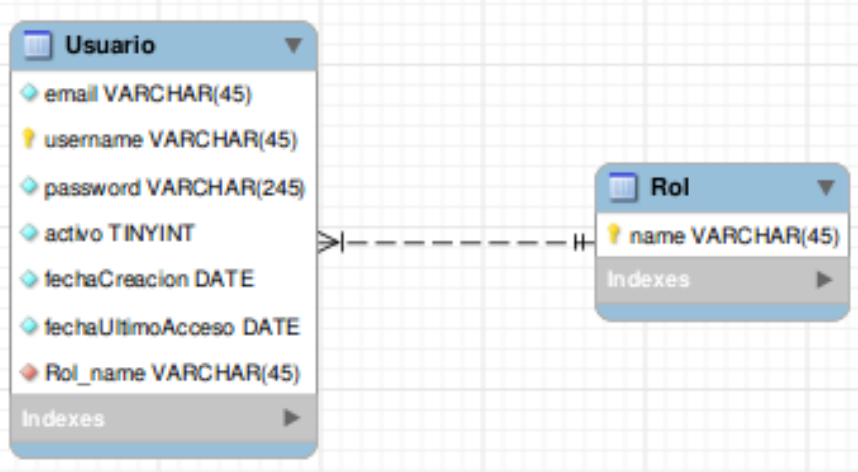

Figura 322. Diagrama de la base de datos del módulo de Inicio de Sesión. (Elaboración propia)

6.5.5. Módulo Usuarios

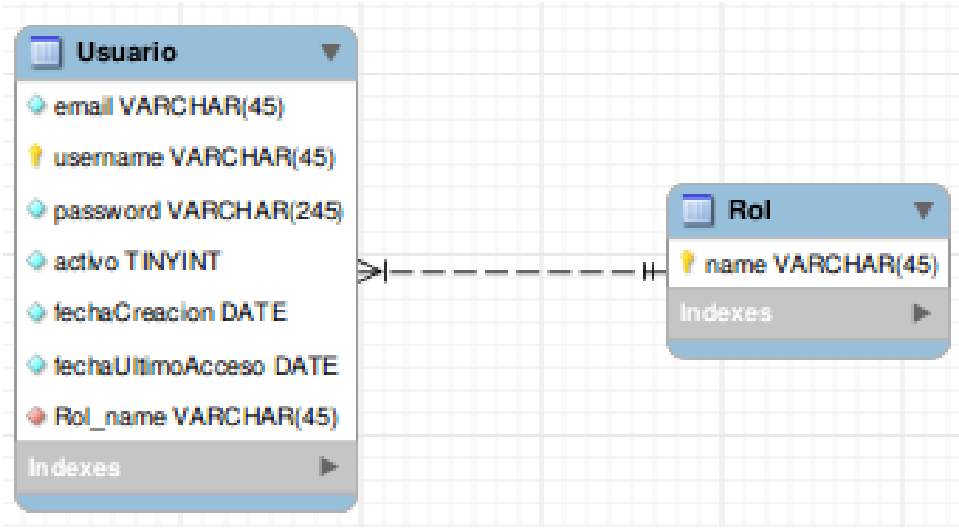

Figura 323. Diagrama de la base de datos del módulo de usuarios. (Elaboración propia)

6.5.6. Módulo Técnico

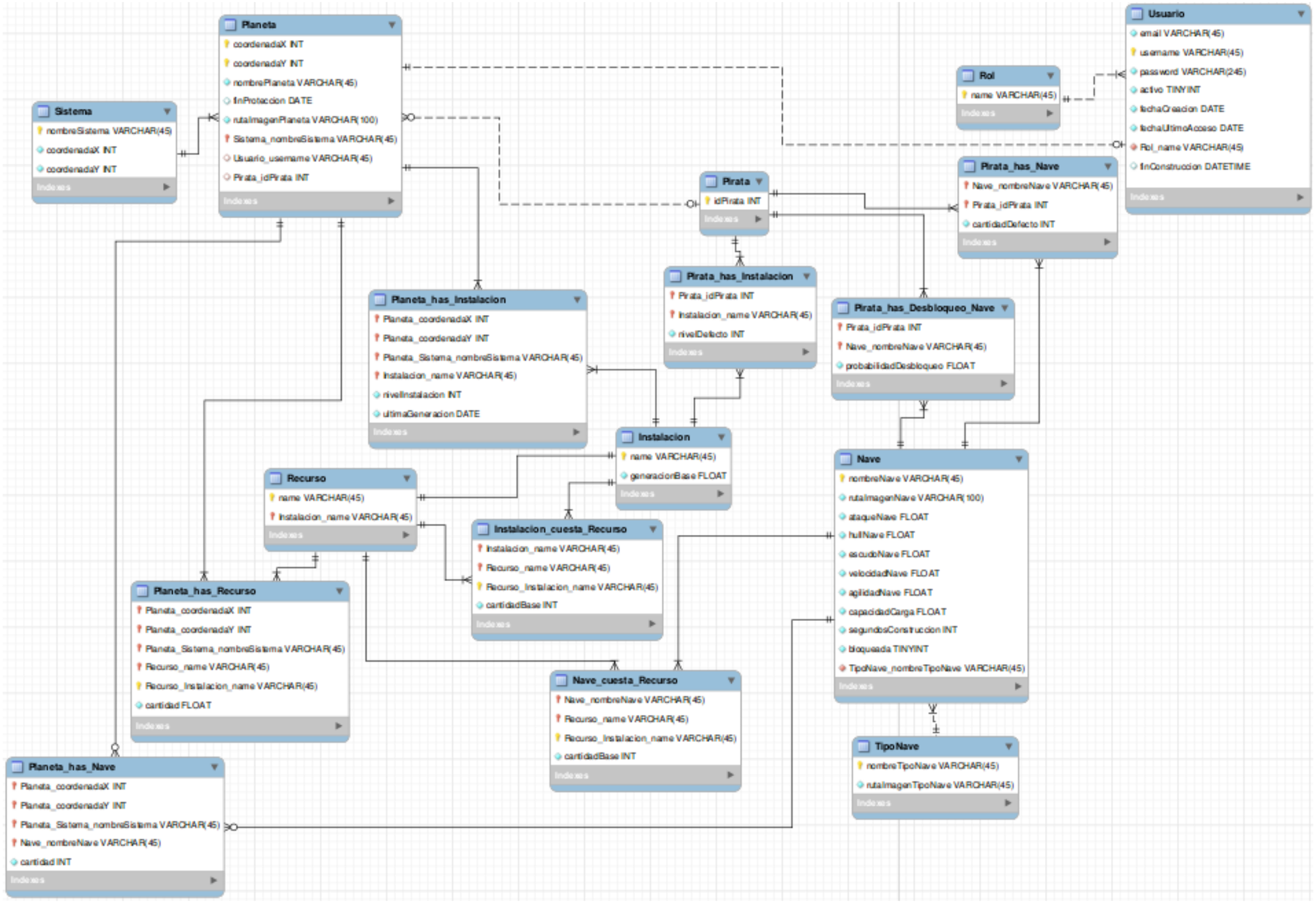

Figura 324. Diagrama de la base de datos del módulo técnico. (Elaboración propia)

6.5.7. Módulo Naves

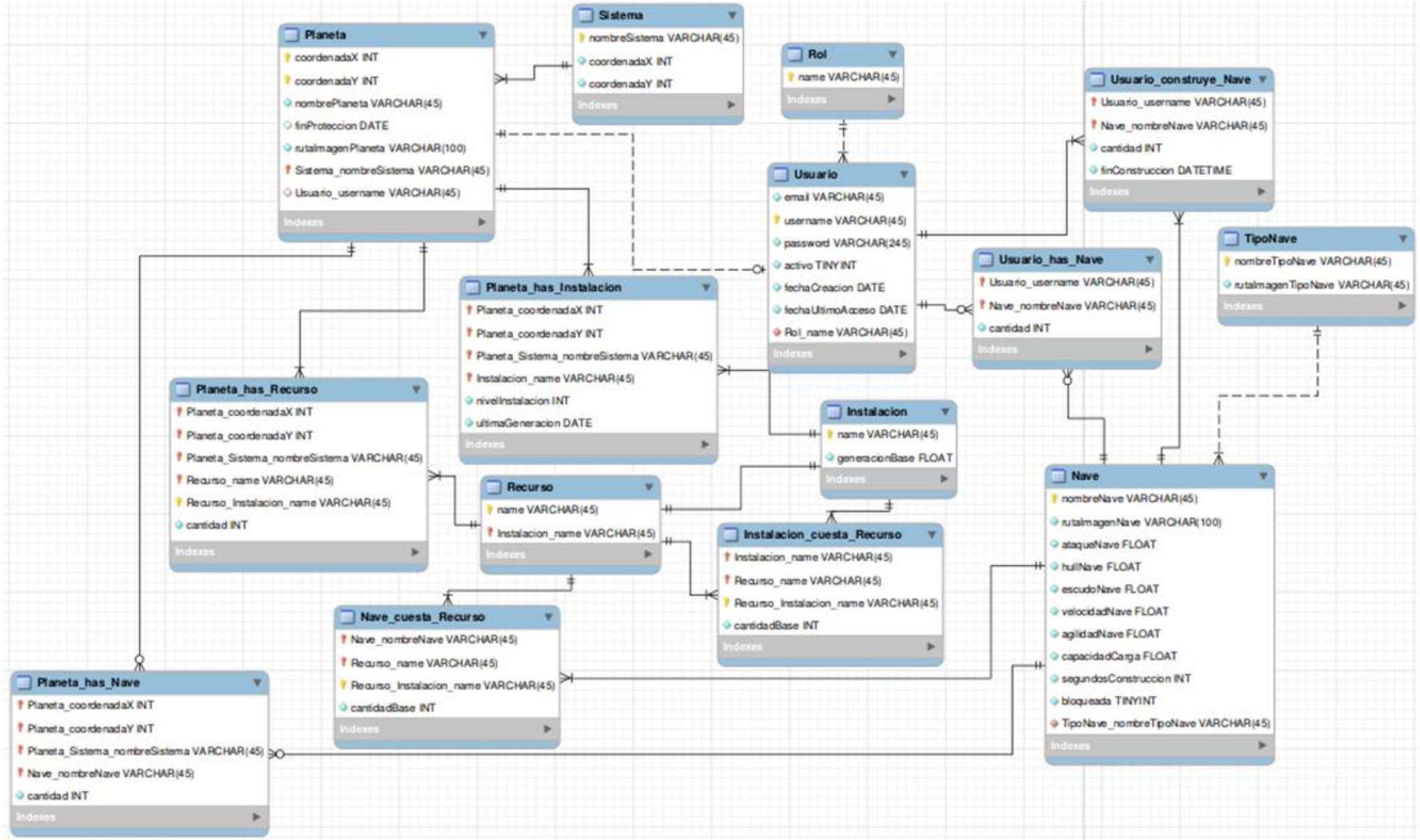

Figura 325. Diagrama de la base de datos del módulo naves. (Elaboración propia)

6.5.8. Módulo Información

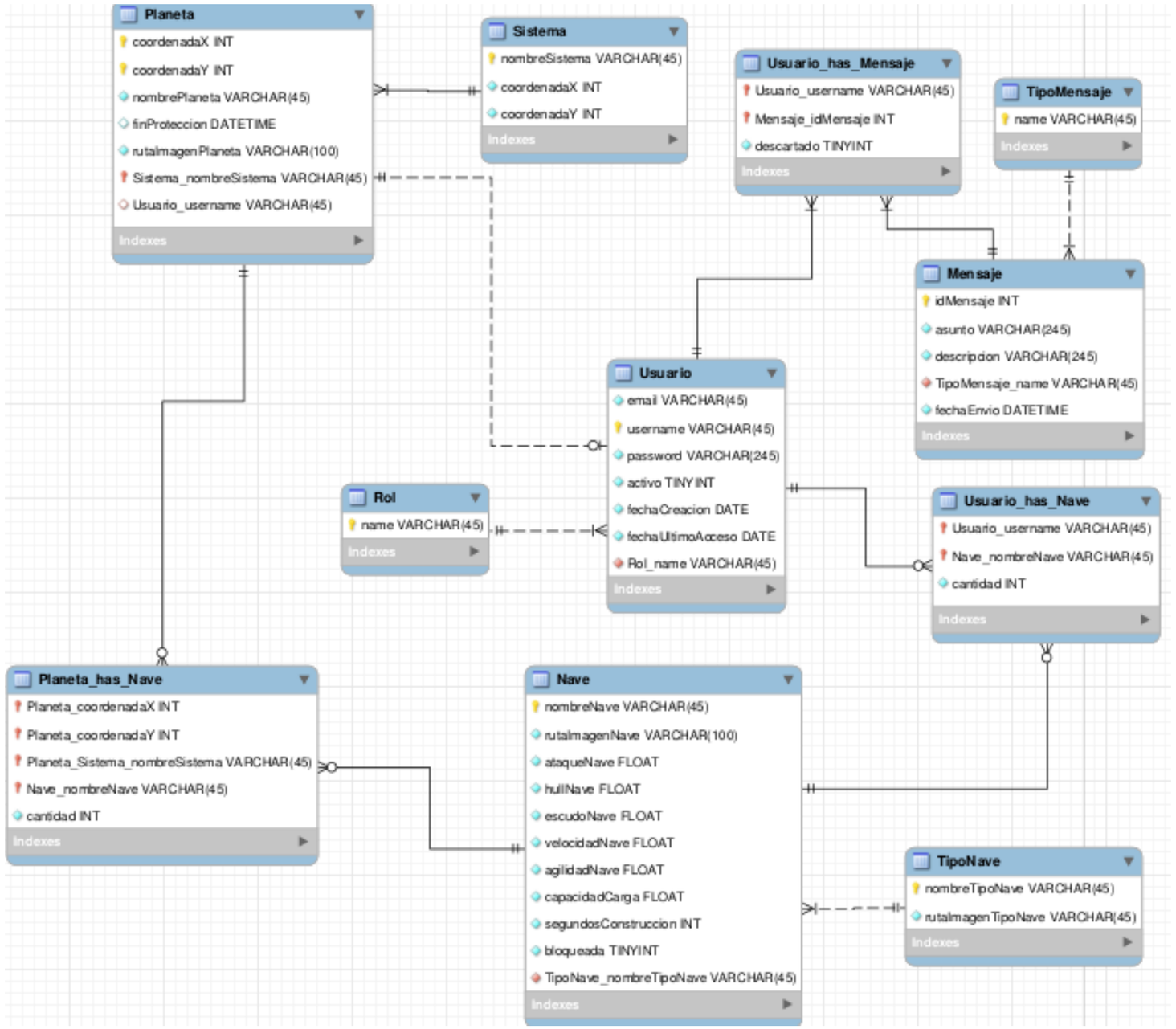

Figura 326. Diagrama de la base de datos del módulo información. (Elaboración propia)

6.5.9. Módulo Ataque

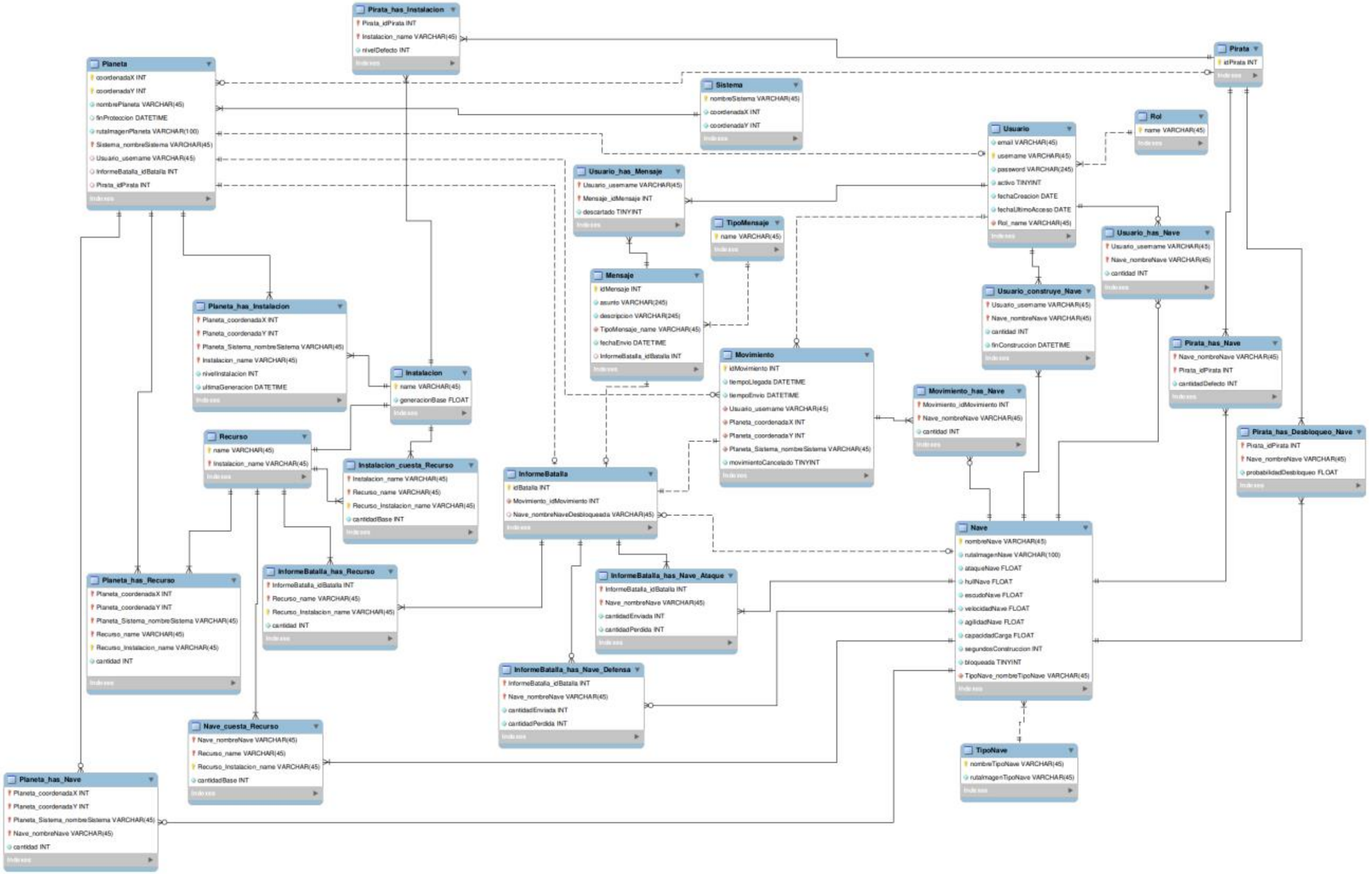

Figura 327. Diagrama de la base de datos del módulo ataque. (Elaboración propia)

6.5.10. Módulo Integración

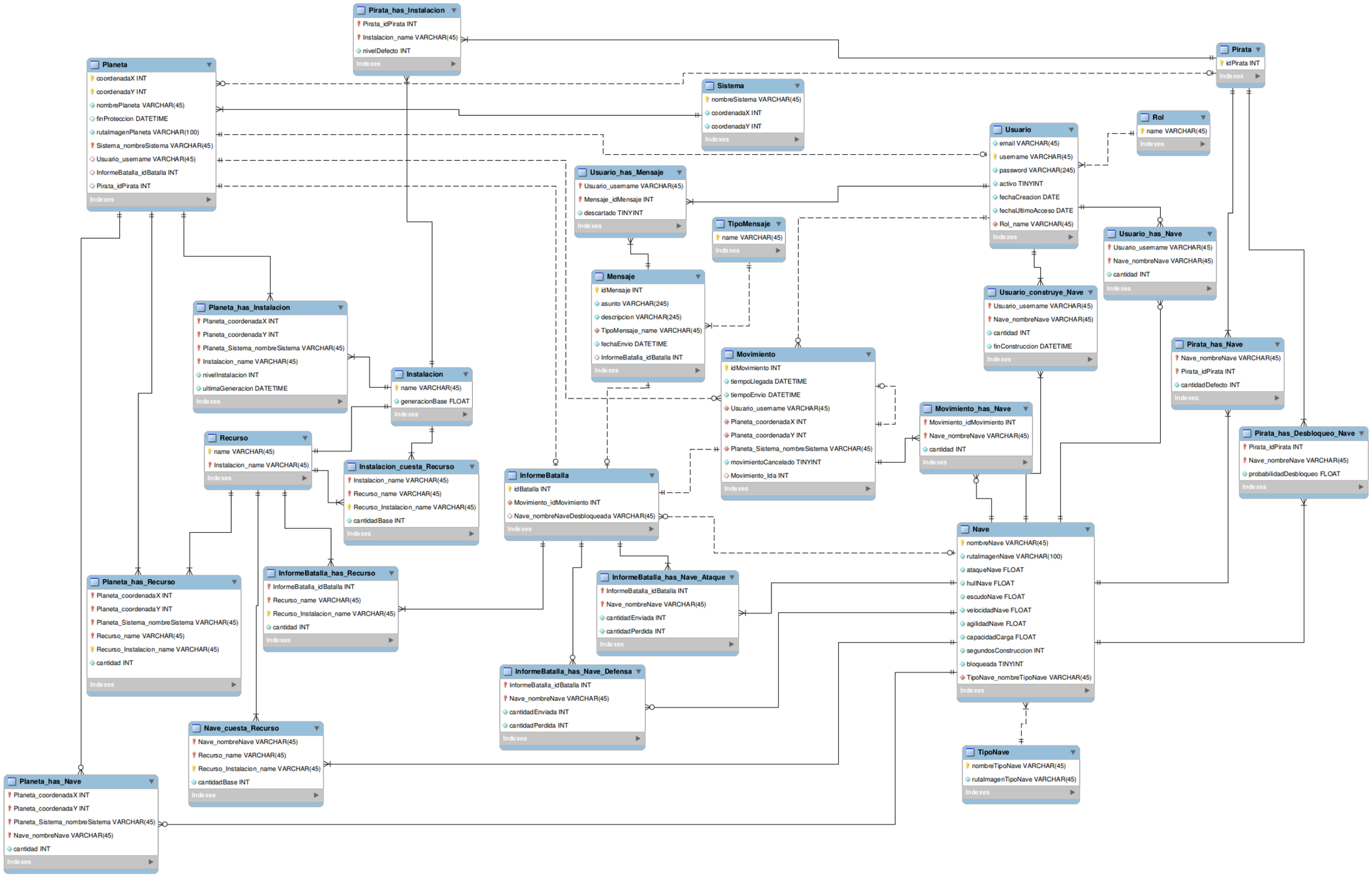

Figura 328. Diagrama de la base de datos del módulo integración. (Elaboración propia)

6.6. Capturas de la interfaz

En esta subsección se muestran todas las capturas de la interfaz divididas en función del actor que accede a ellas.

6.6.4. Cibernauta

Inicio de sesión

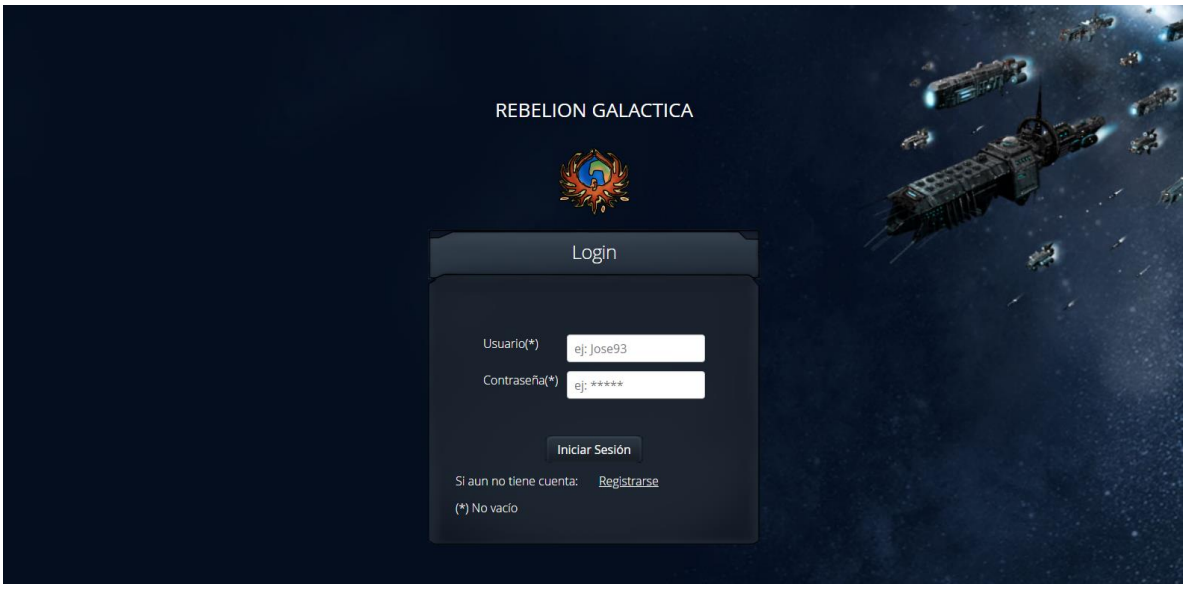

Figura 329. Ventana de inicio de sesión. (Elaboración propia)

Registro

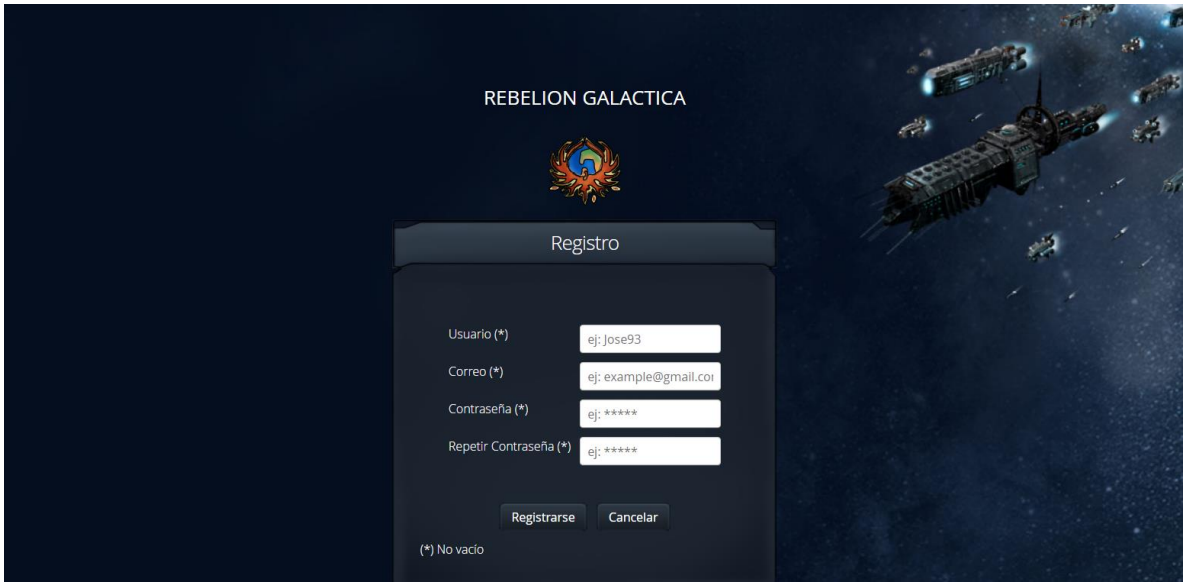

Figura 330. Ventana de registro. (Elaboración propia)

6.6.5. Jugador

Página principal (Jugador)

Figura 331. Ventana de la pantalla principal (jugador). (Elaboración propia)

Ventana de instalaciones (Recursos)

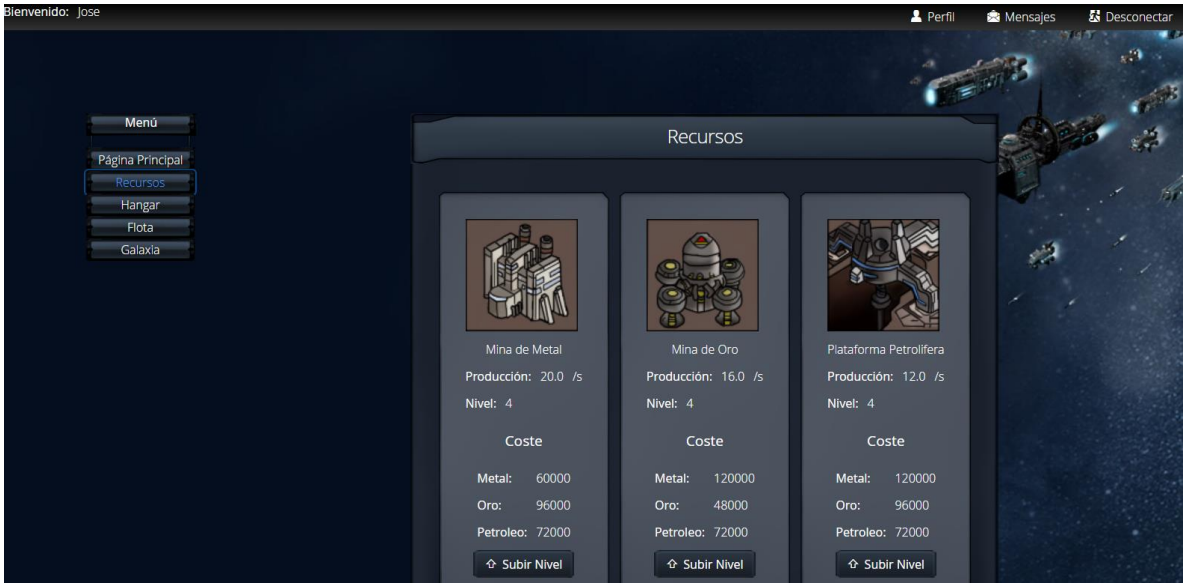

Figura 332. Ventana de recursos. (Elaboración propia)

Hangar

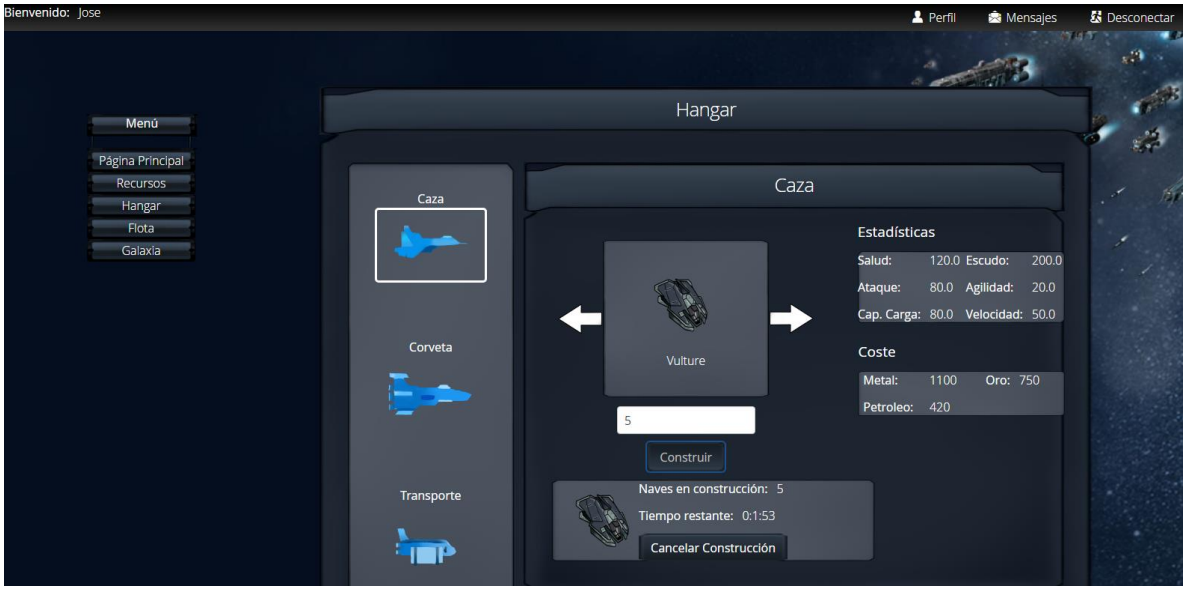

Figura 333. Ventana de hangar. (Elaboración propia)

Ventana de flota

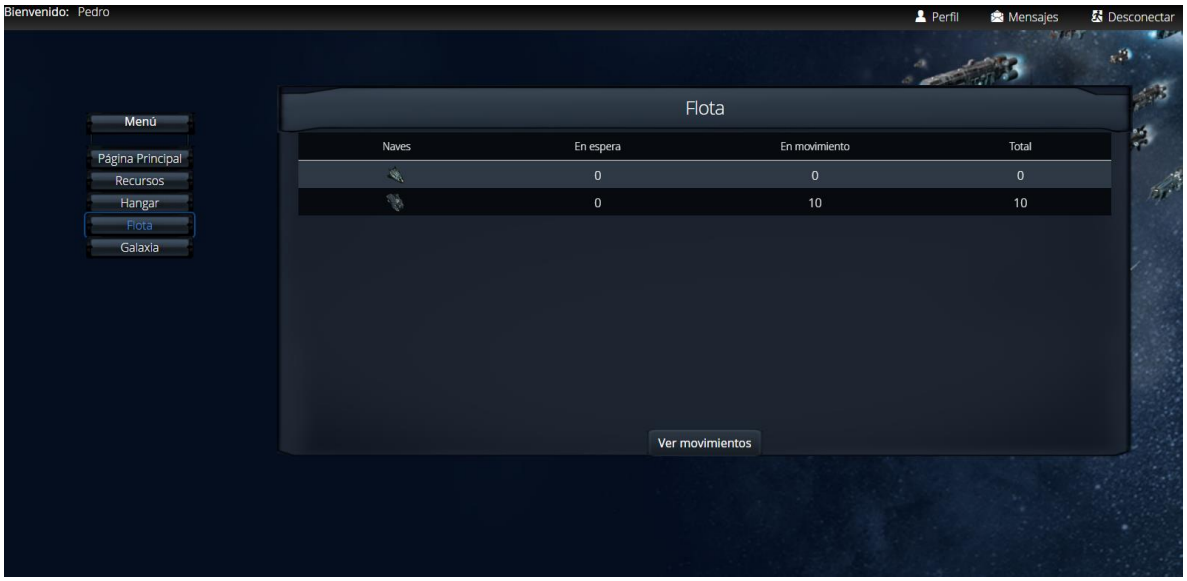

Figura 334. Ventana de flota. (Elaboración propia)

Ventana de movimientos

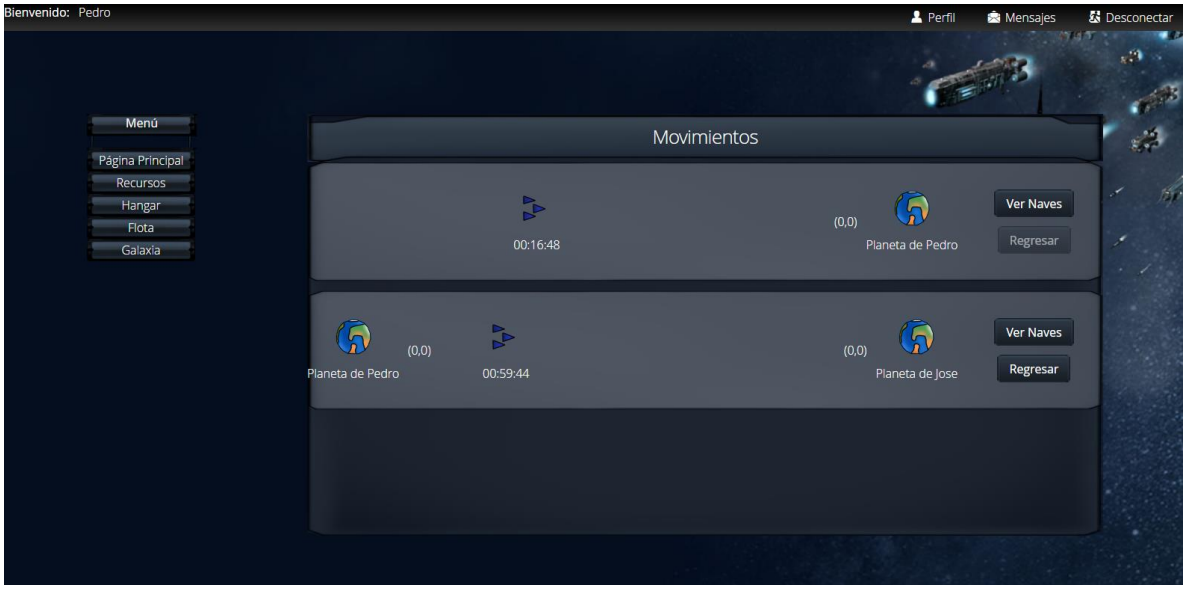

Figura 335. Ventana de movimientos. (Elaboración propia)

Ventana de movimiento

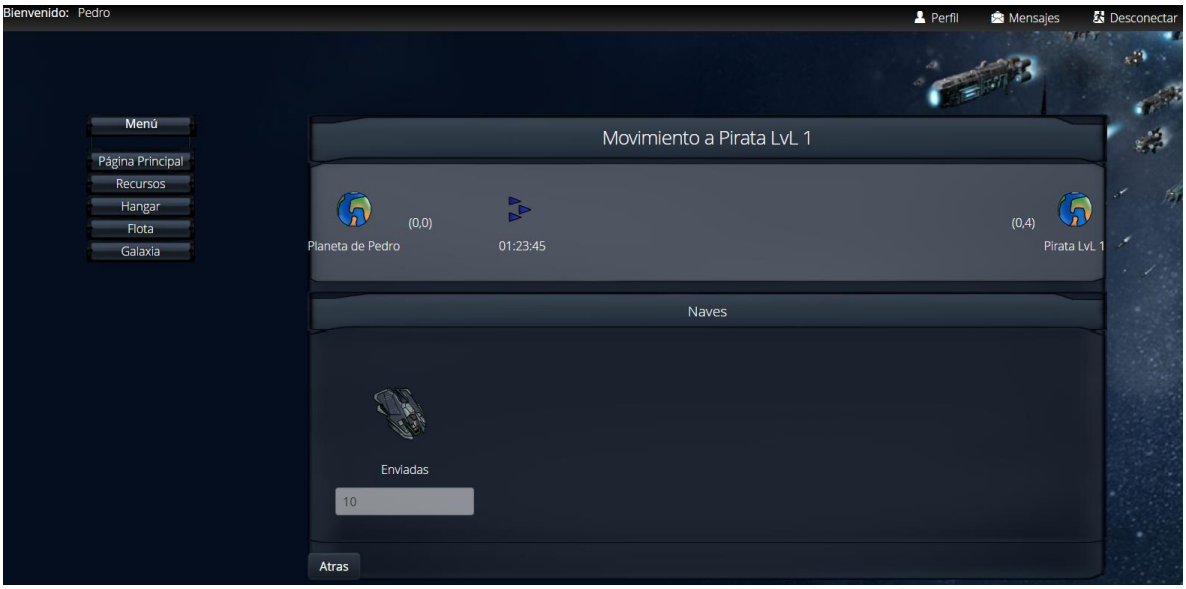

Figura 336. Ventana de movimiento. (Elaboración propia)

Listado de planetas (Galaxia)

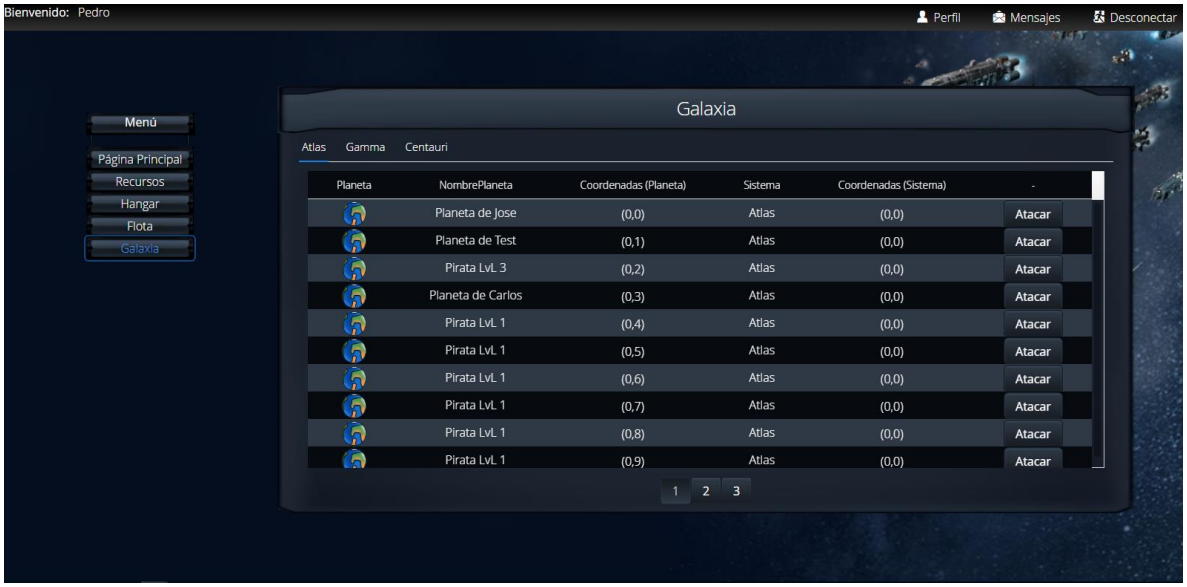

Figura 337. Ventana de galaxia. (Elaboración propia)

Ventana de ataque

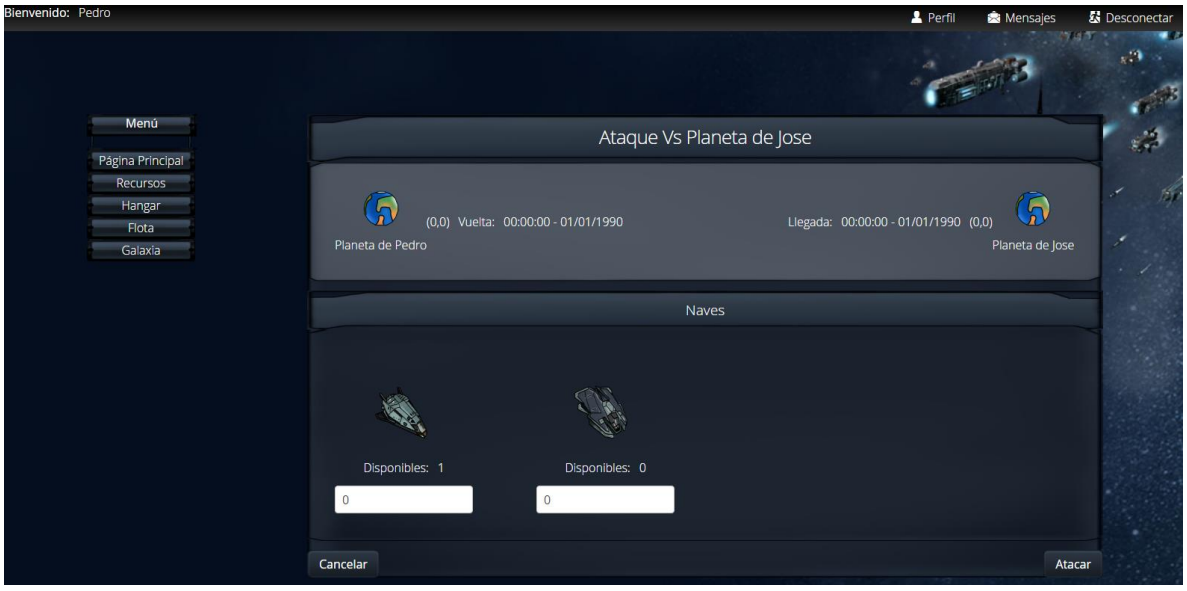

Figura 338. Ventana de ataque. (Elaboración propia)

Perfil

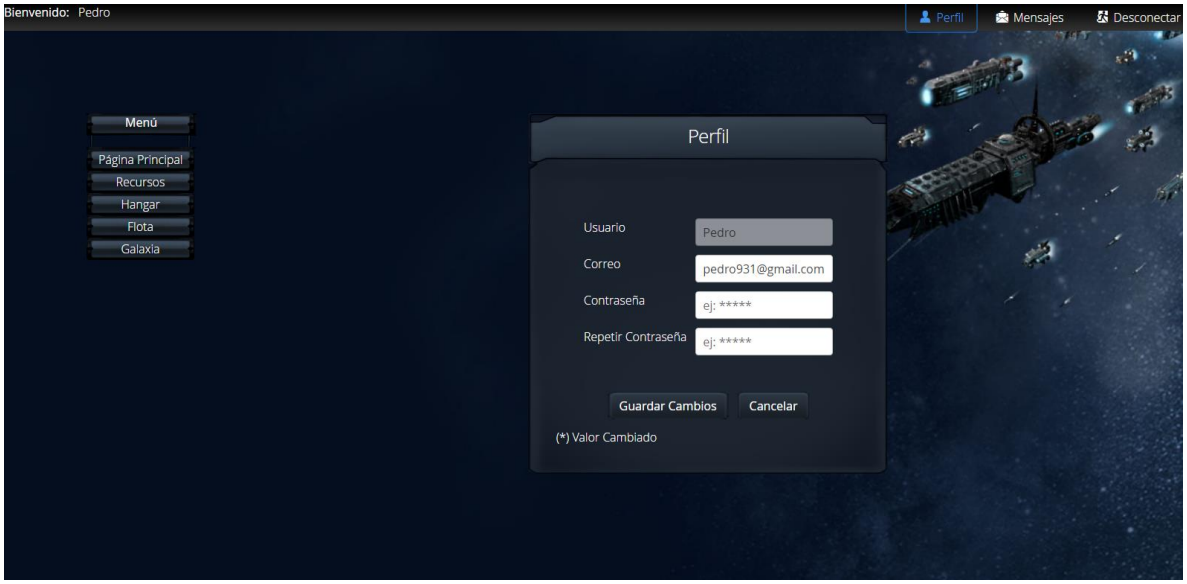

Figura 339. Ventana de perfil. (Elaboración propia)

Listado de mensajes

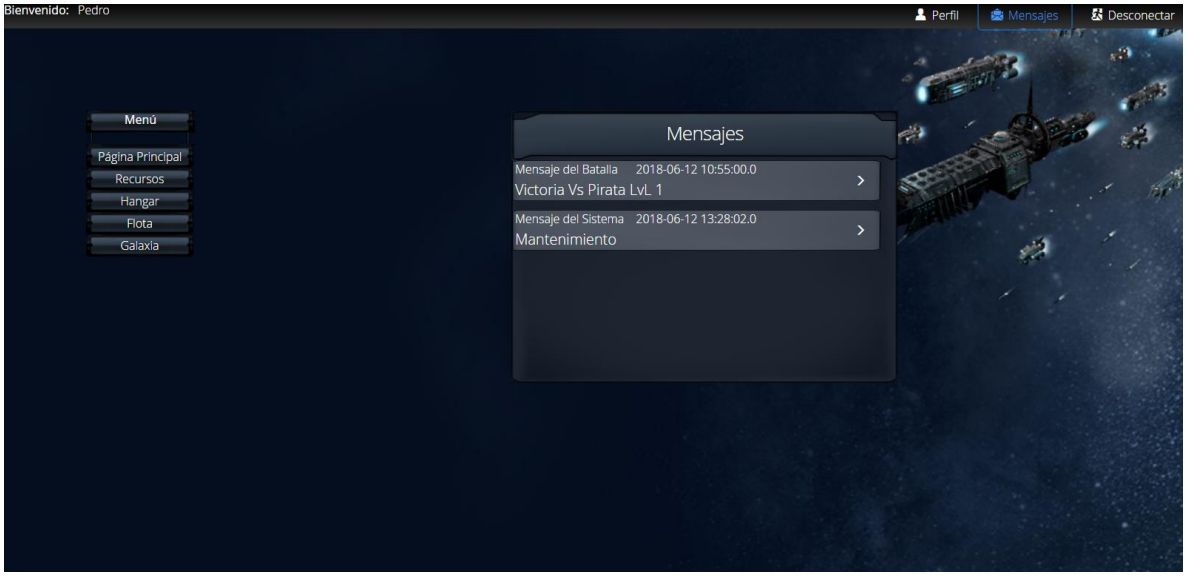

Figura 340. Ventana de mensajes. (Elaboración propia)

Naves atacantes de un mensaje de tipo batalla

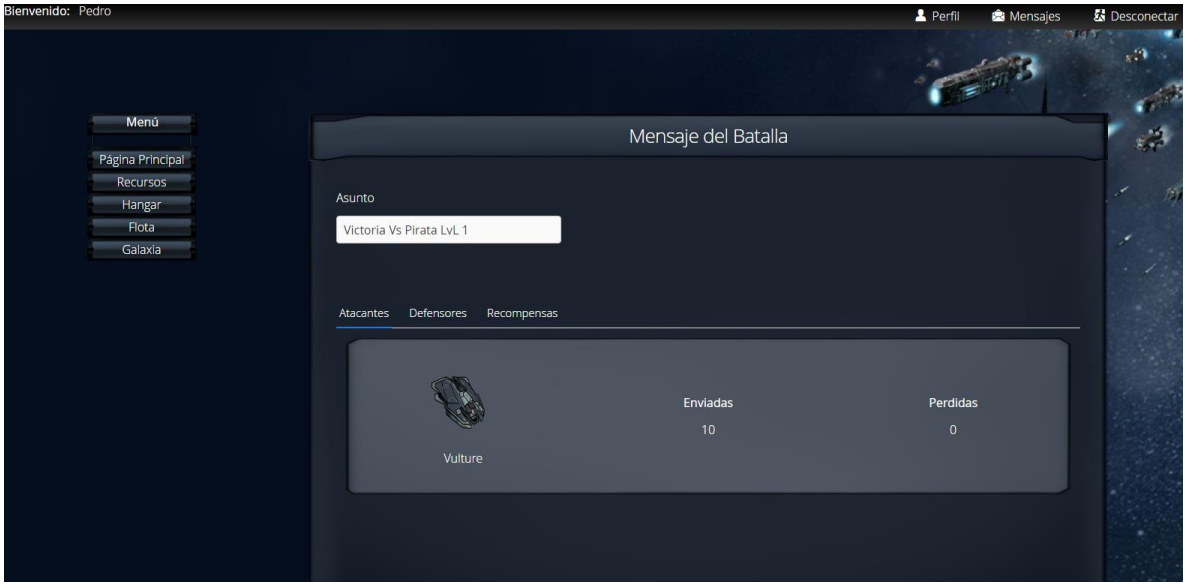

Figura 341. Ventana de mensaje de tipo batalla (atacantes). (Elaboración propia)

Naves defensoras de un mensaje de tipo batalla

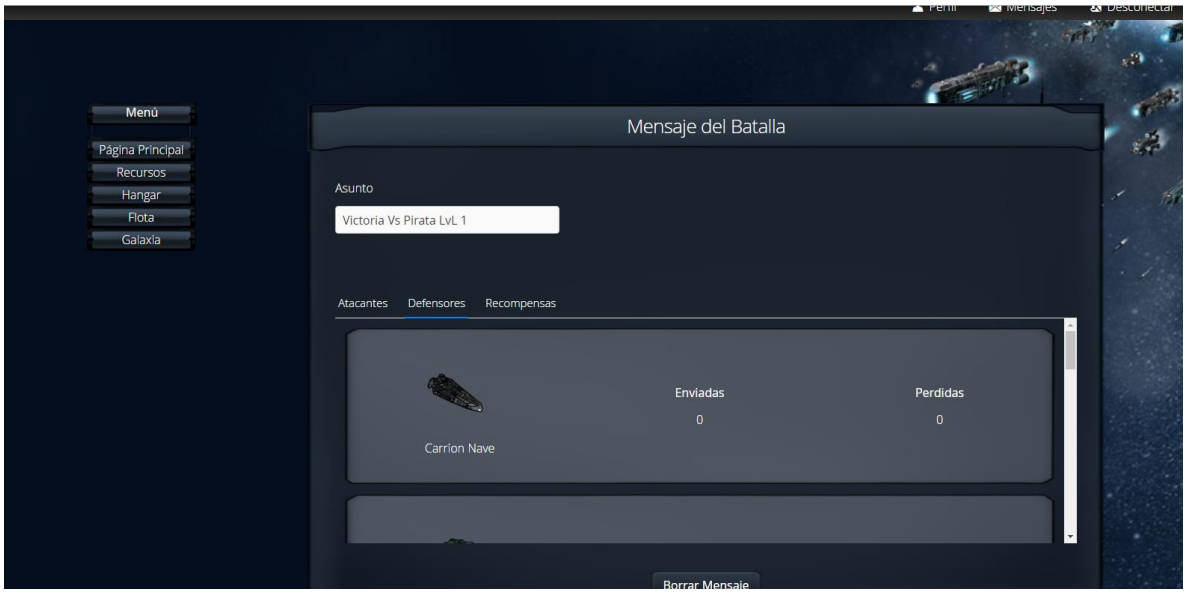

Figura 342. Ventana de mensaje de tipo batalla (defensores). (Elaboración propia)

Recursos obtenidos de un mensaje de tipo batalla

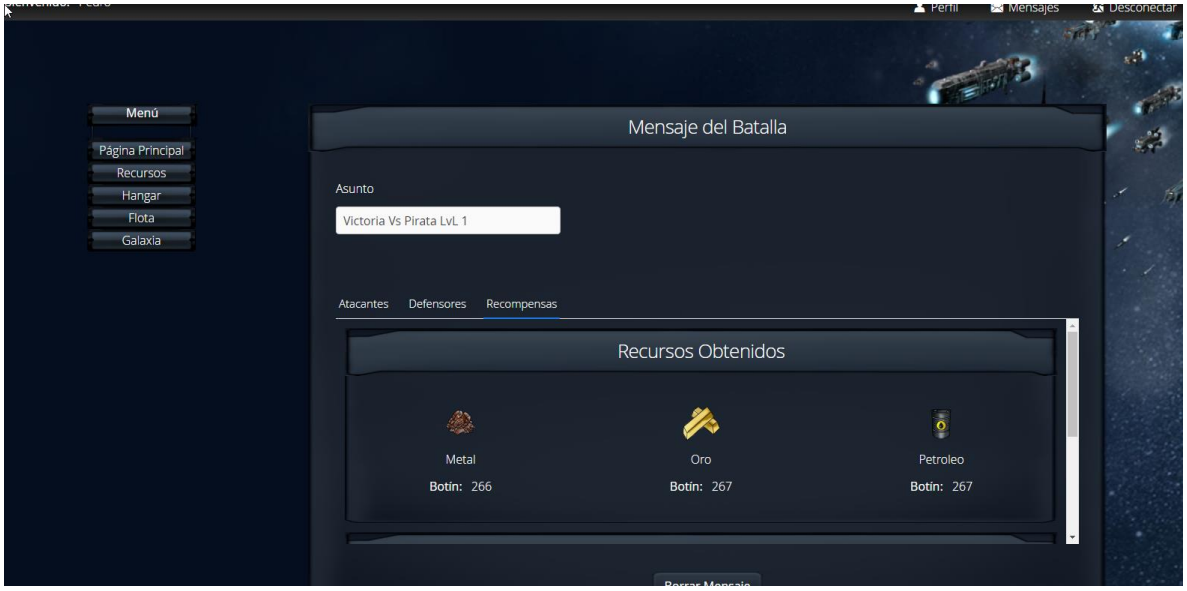

Figura 343. Ventana de mensaje de tipo batalla (recursos). (Elaboración propia)

Naves desbloqueadas de un mensaje de tipo batalla

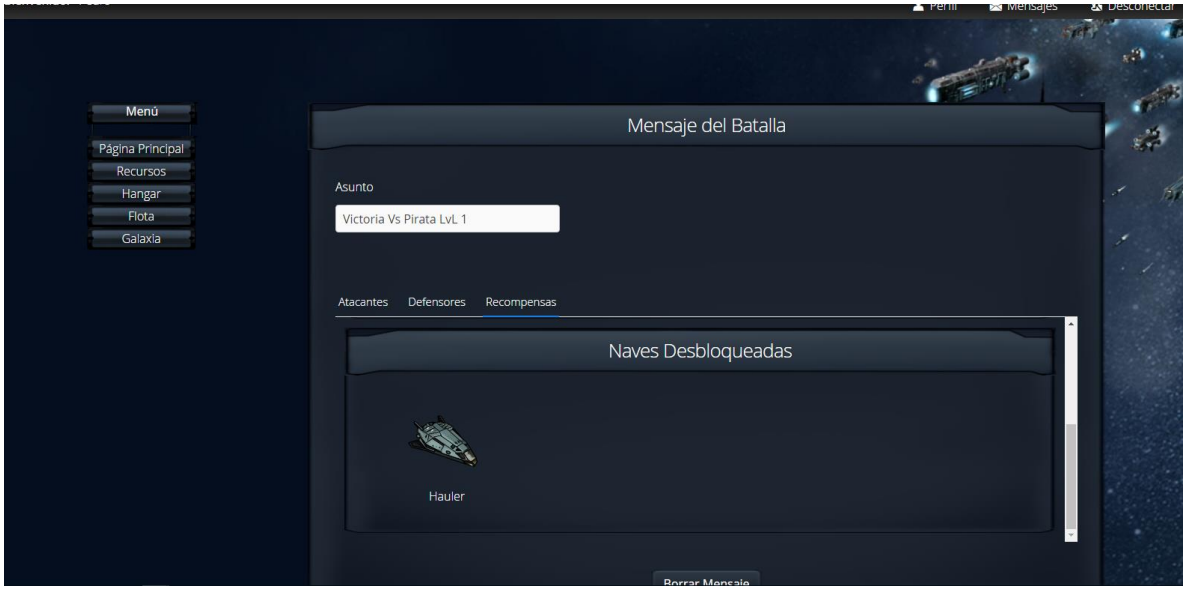

Figura 344. Ventana de mensaje de tipo batalla (naves desbloqueadas). (Elaboración propia)

Mensaje de tipo sistema

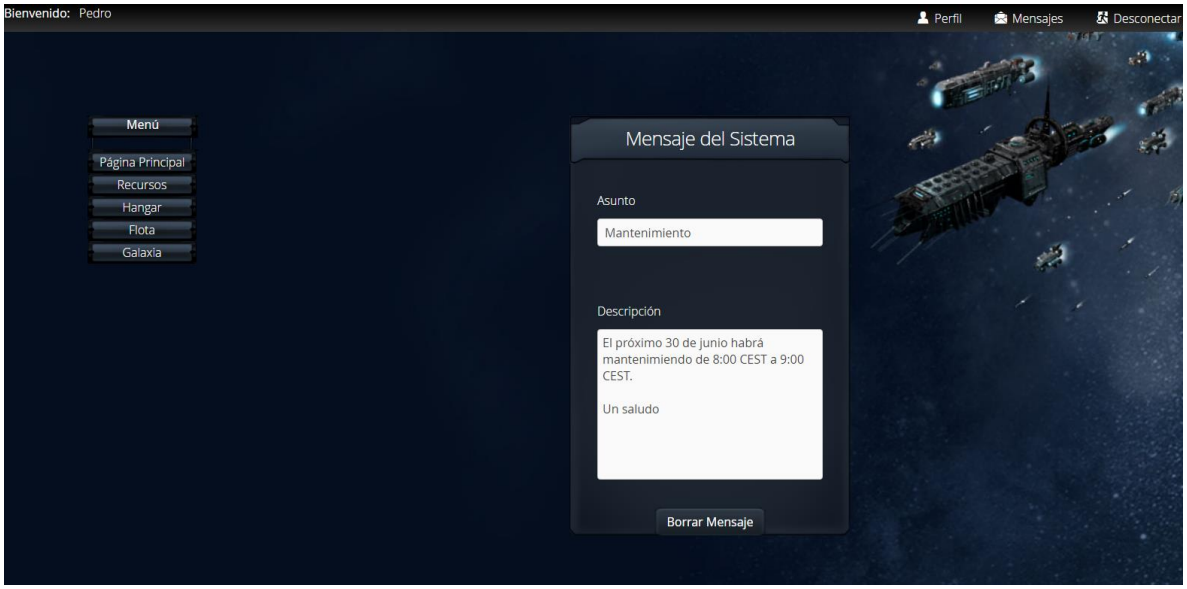

Figura 345. Ventana de mensaje de tipo sistema. (Elaboración propia)

6.6.6. Administrador

Página principal (Administrador)

Figura 346. Ventana de la pantalla principal (administrador). (Elaboración propia)

Listado de usuarios

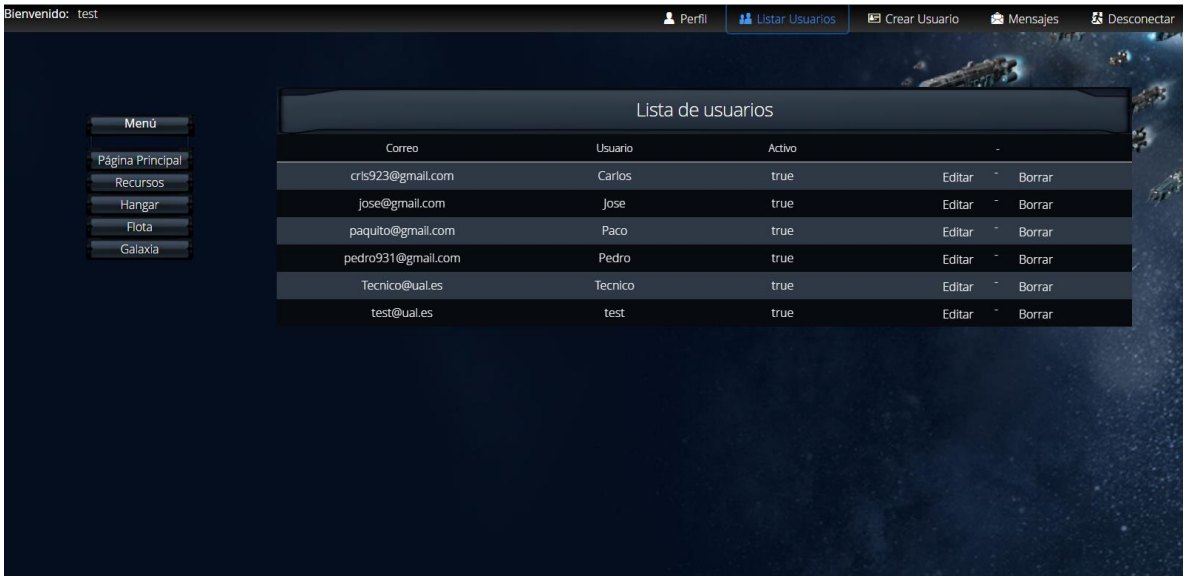

Figura 347. Ventana de listado de usuarios. (Elaboración propia)

Editar usuario

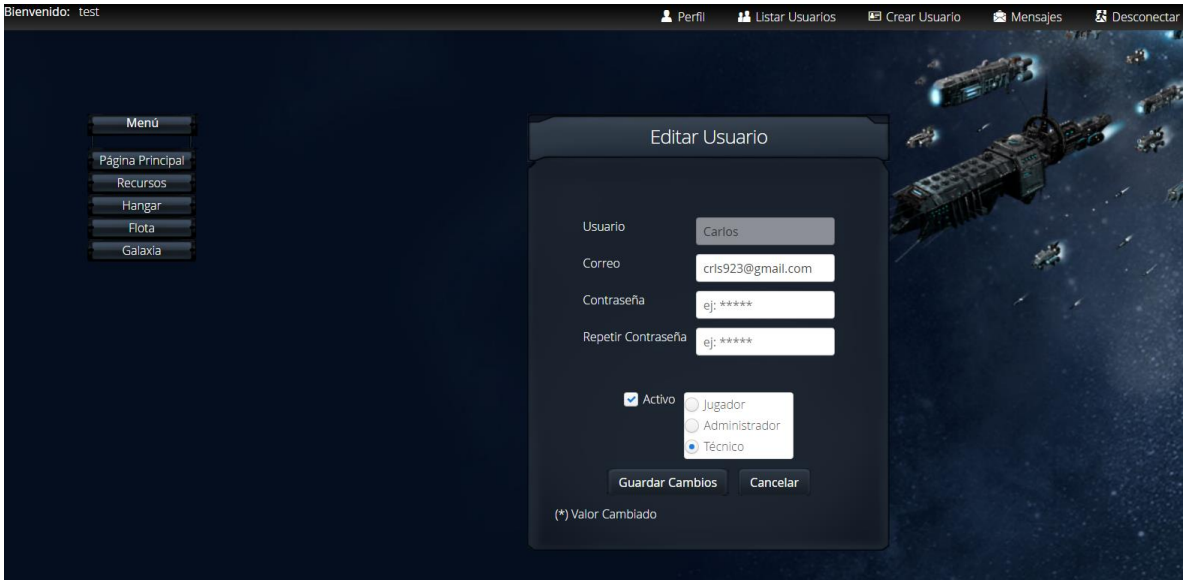

Figura 348. Ventana de editar usuario. (Elaboración propia)

Borrar usuario

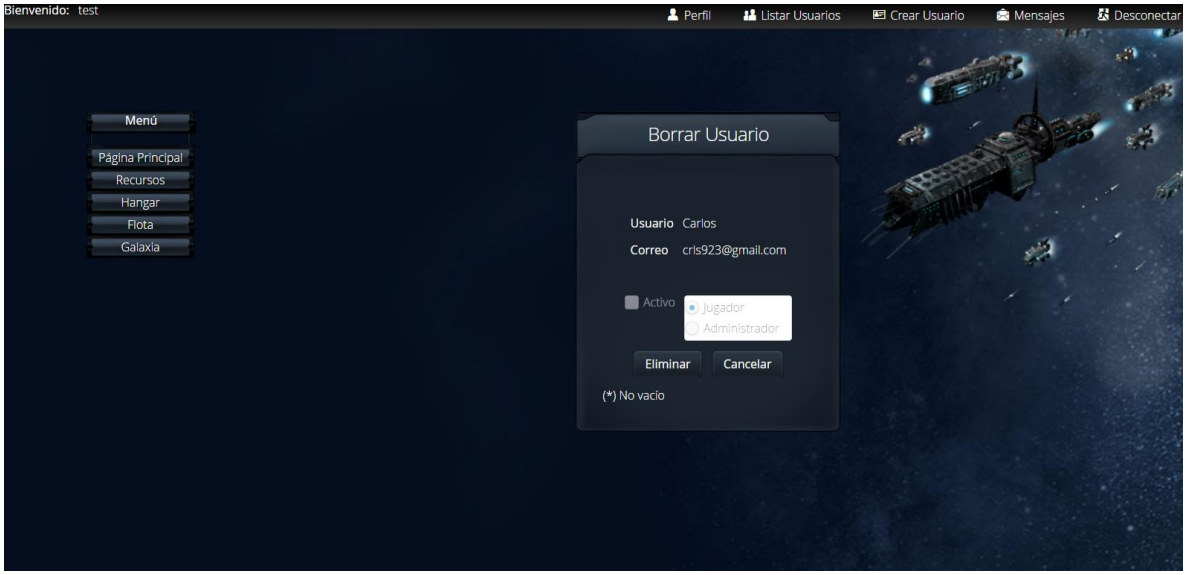

Figura 349. Ventana de borrar usuario. (Elaboración propia)

Crear usuario

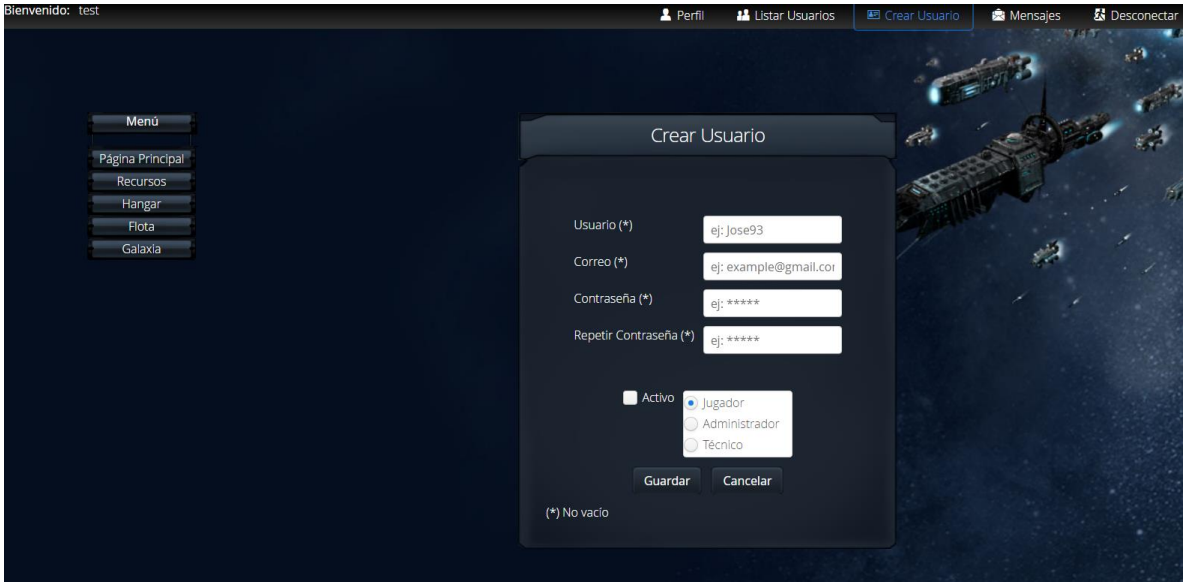

Figura 350. Ventana de crear usuario. (Elaboración propia)

Crear mensaje de tipo sistema

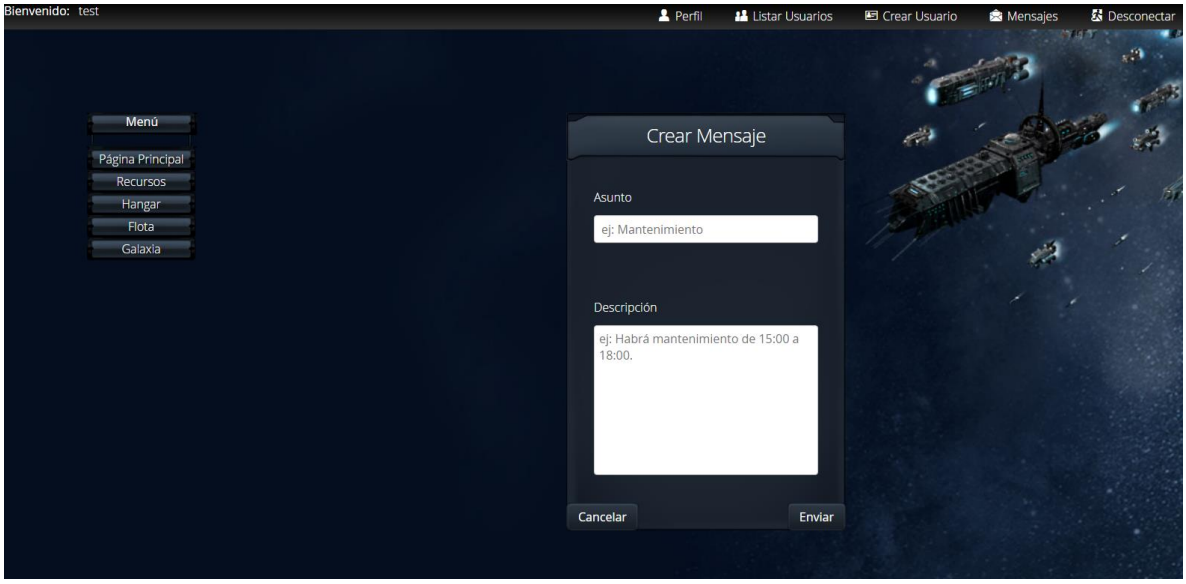

Figura 351. Ventana de crear mensaje de tipo sistema. (Elaboración propia)

6.6.7. Técnico

Página principal (Técnico)

Figura 352. Ventana de la pantalla principal (técnico). (Elaboración propia)

Crear nave

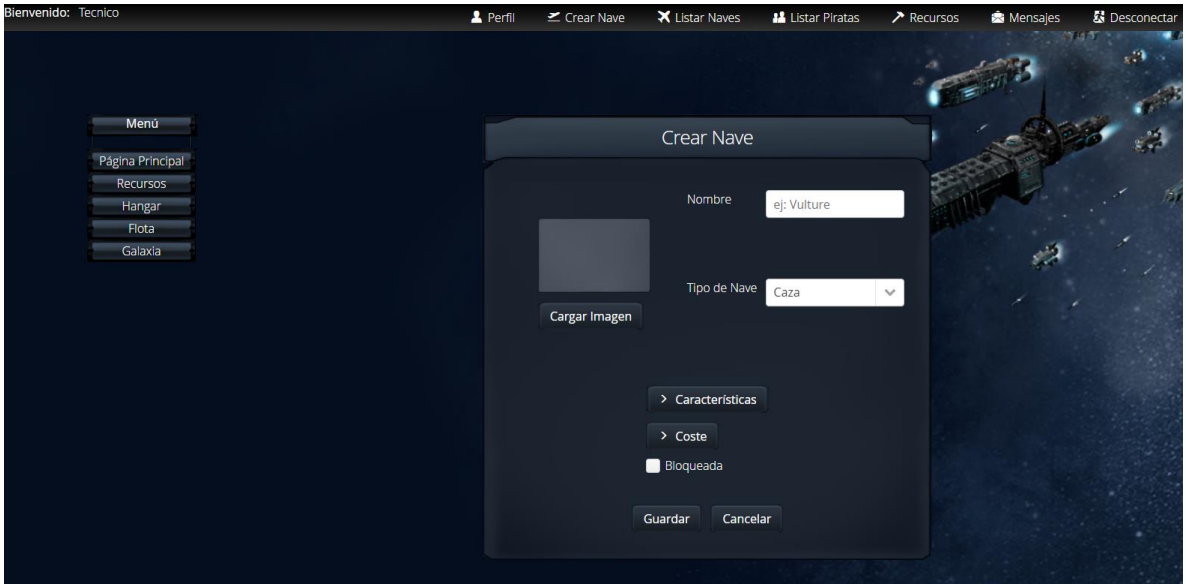

Figura 353. Ventana de crear nave. (Elaboración propia)

Listado de naves

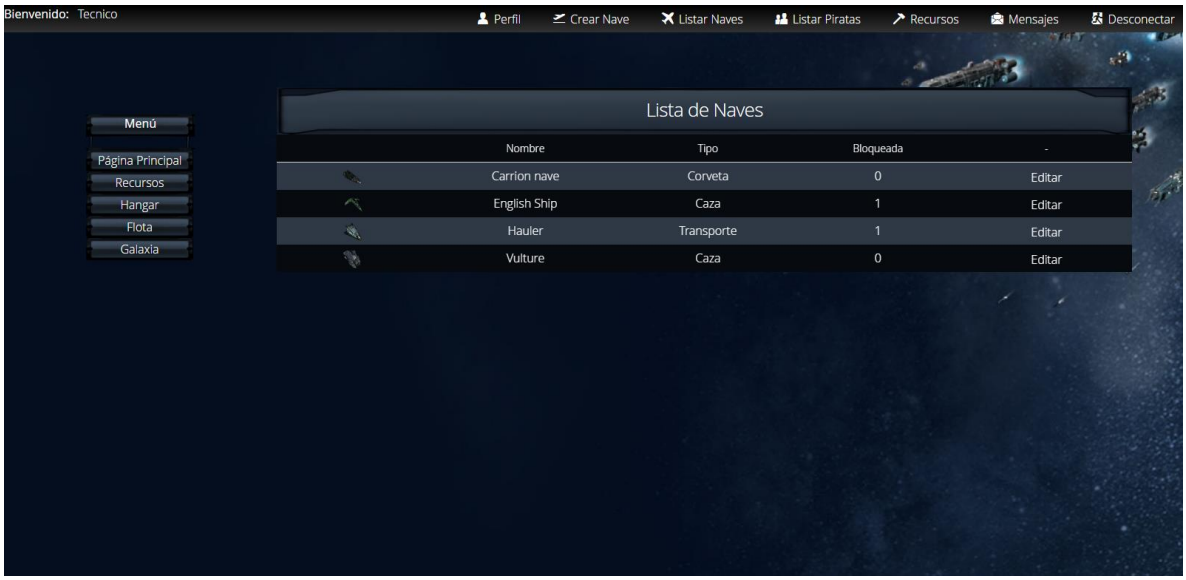

Figura 354. Ventana de listado de naves. (Elaboración propia)

Editar nave

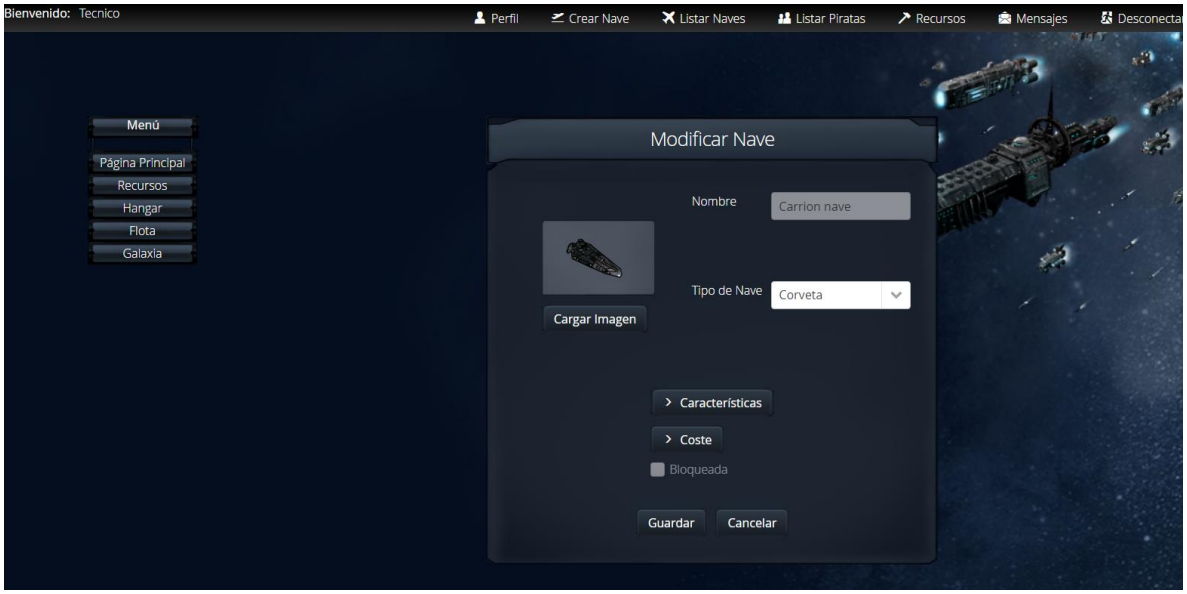

Figura 355. Ventana de editar nave. (Elaboración propia)

Listado de piratas

Figura 356. Ventana de listado de piratas. (Elaboración propia)

Editar pirata (Niveles de sus instalaciones)

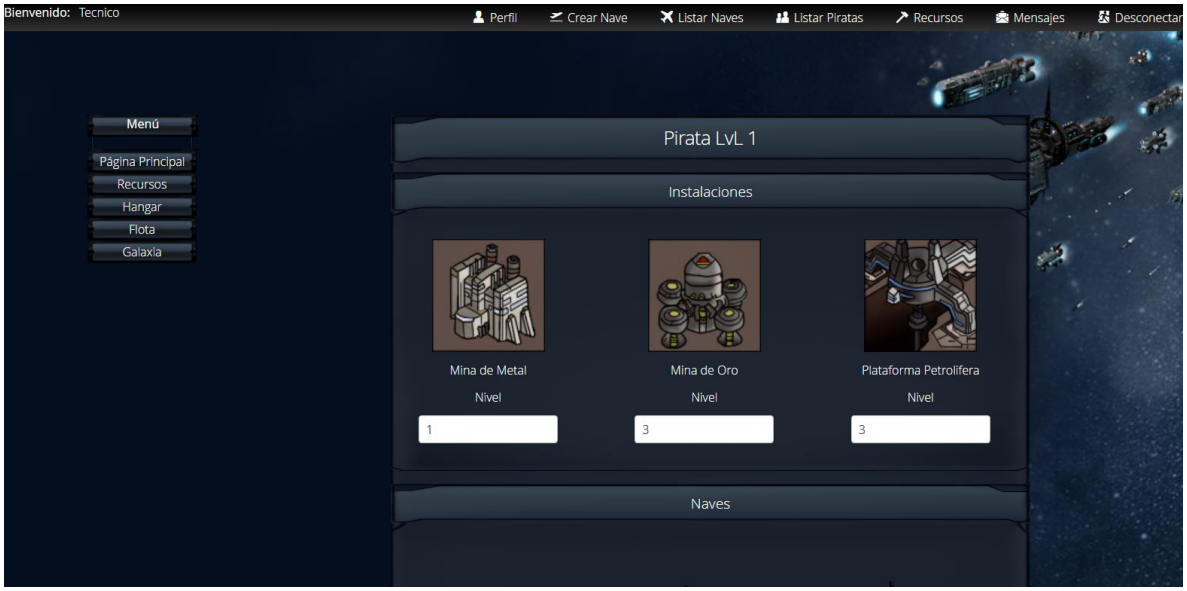

Figura 357. Ventana de editar pirata (instalaciones). (Elaboración propia)

Editar pirata (Cantidad de naves disponibles)

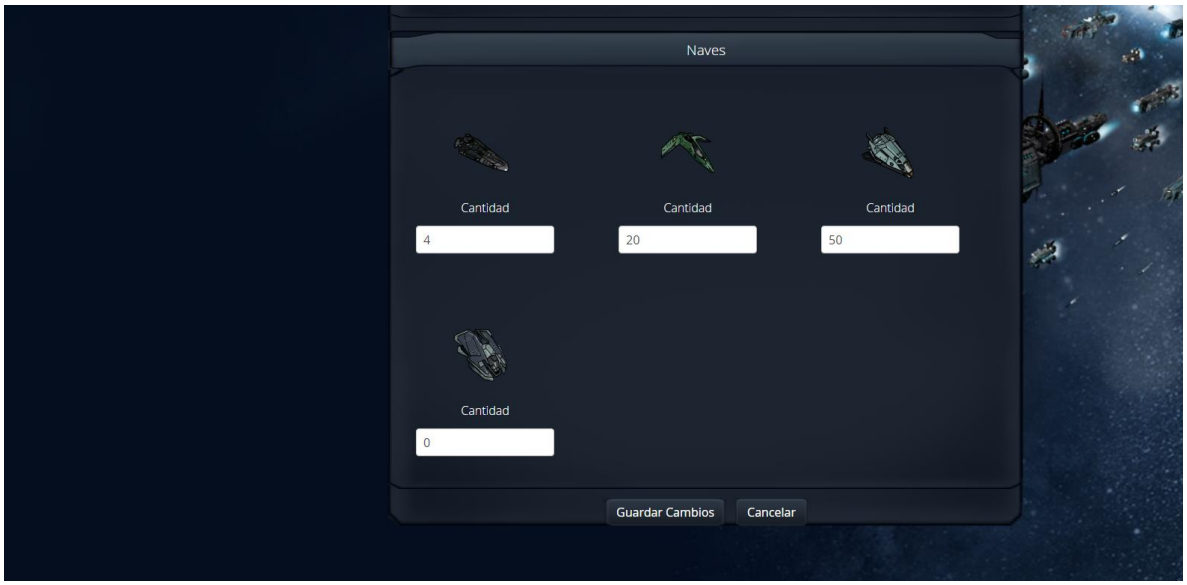

Figura 358. Ventana de editar pirata (naves). (Elaboración propia)

Editar instalaciones (Recursos)

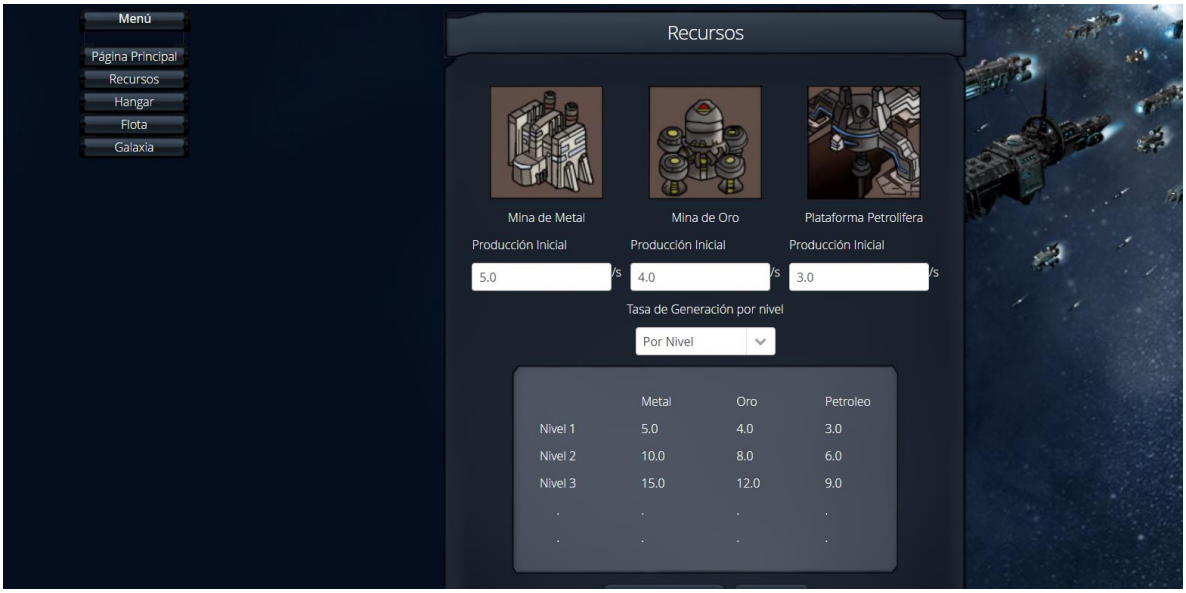

Figura 359. Ventana de recursos (técnico). (Elaboración propia)

6.7. Imágenes utilizadas

En esta subsección se muestran todas las imágenes utilizadas en la aplicación realizada.

Fondo de pantalla

Figura 360. Fondo de pantalla de la aplicación. [14]

Tipos de nave

Figura 361. Tipo de nave corveta. (Elaboración propia)

Figura 362. Tipo de nave caza. (Elaboración propia)

Figura 363. Tipo de nave transporte. (Elaboración propia)

Icono de nave bloqueada

Figura 364. Icono de nave bloqueada. [55]

Naves

Figura 365. Nave Anaconda. (Elaboración propia)

Figura 366. Nave Eagle. (Elaboración propia)

Figura 367. Nave Vulture. (Elaboración propia)

Figura 368. Nave Hauler. (Elaboración propia)

Flechas navegabilidad

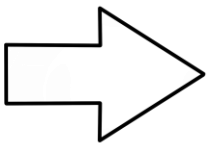

Figura 369. Flecha derecha de navegabilidad. (Elaboración propia)

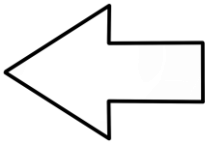

Figura 370. Flecha izquierda de navegabilidad. (Elaboración propia)

Logo de la aplicación

Figura 371. Logo de la aplicación. (Elaboración propia)

Recursos

Figura 372. Recurso metal. (Elaboración propia)

Figura 373. Recurso oro. (Elaboración propia)

Figura 374. Recurso petróleo. (Elaboración propia)

Instalaciones

Figura 375. Mina de metal. (Elaboración propia)

Figura 376. Mina de oro. (Elaboración propia)

Figura 377. Plataforma petrolífera. (Elaboración propia)

Planeta

Figura 378. Icono de los planetas. (Elaboración propia)

Icono movimiento entre planetas

Figura 379. Icono de movimiento entre planetas. (Elaboración propia)

Fondo título del menú

Figura 380. Fondo del título del menú. (Elaboración propia)

Fondo botón del menú

Figura 381. Fondo de los botones del menú. [14]

Fondo botón del menú seleccionado

Figura 382. Fondo de los botones del menú seleccionados. [14]

Fondo botón

Figura 383. Fondo de los botones. [14]

Fondo botón pulsado

Figura 384. Fondo de los botones pulsados. [14]

Cabecera ventanas

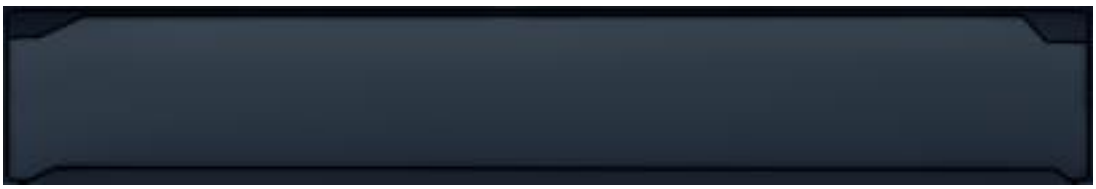

Figura 385. Cabecera de las ventanas. (Elaboración propia)

Fondo vertical ventanas

Figura 386. Fondo vertical de las ventanas. (Elaboración propia)

Fondo horizontal ventanas

Figura 387. Fondo horizontal de las ventanas. (Elaboración propia)

Fondo ventanas

Figura 388. Fondo de las ventanas. (Elaboración propia)

Imágenes del tutorial

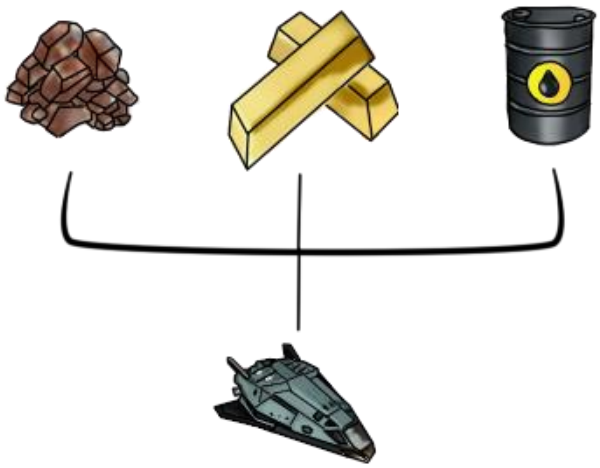

Figura 389. Imagen construcción tutorial. (Elaboración propia)

Figura 390. Imagen desbloqueo de naves tutorial. (Elaboración propia)

Figura 391. Número uno tutorial. (Elaboración propia)

Figura 392. Número dos tutorial. (Elaboración propia)

Figura 393. Número tres tutorial. (Elaboración propia)

Figura 394. Número cuatro tutorial. (Elaboración propia)

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

El objetivo de este proyecto es el desarrollo de una metodología enfocada en la creación de videojuegos por parte de equipos pequeños y probada en un caso práctico, un juego de navegador basado en el videojuego online Ogame. Dicha metodología ayudará a desarrolladores Indies a crear productos de calidad, puesto que siendo ellos la base de la industria de los videojuegos en los últimos años, son los que más fracasan debido a la falta de metodologías en su desarrollo, causados principalmente por su falta de experiencia. La metodología estará centrada en la división del trabajo en módulos, buscando mantener la documentación utilizando una metodología ágil e intentando que la carga de trabajo que genera sea la menor posible, al dividir el proyecto en módulos la documentación se haría mucho más rápido.

The main purpose of this project is the development of a methodology focused on the game development by small teams and tested in a practice case, a videogame like the online game Ogame. This methodology will help Indie developers to create quality software, since they are the videogame industry core in the last years, they are the ones that fails the most because of the lack of methodologies in the development, mainly due to the lack of experience. The methodology will be focused on the division of work into modules, trying to keep the documentation using an agile methodology and the workload as low as possible, by dividing the project into modules the documentation would be much faster.

GRADO EN INGENIERÍA INFORMÁTICA, 2017/2018