TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Animáster - Automatización de "Ánima: Beyond Fantasy" mediante el uso de UWP y Azure DevOps para la gestión del proyecto

Curso 2020/2021

Alumno/a:

Felipe Antonio Cano Galera

Director/es:

Javier Criado Rodríguez

Manel Mena Vicente

Agradecimientos

En primer lugar, me gustaría agradecer a Javi y a Manel, mis tutores, por todas esas reuniones de una hora que al final acababan siendo mucho más, por la ayuda prestada, por su interés, por el trato recibido y por todo lo que he aprendido con ellos a lo largo de este proyecto y de sus asignaturas en la carrera.

A Inés, por ser tan gran apoyo emocional y una compañera de viaje excepcional a lo largo de estos 4 años.

A Ismael, porque cuando tienes un problema, un amigo te intenta animar, pero un buen amigo se une a ti para intentar solucionarlo, y a lo largo de la carrera lo has hecho más veces de las que puedo contar. Estoy feliz de poder decir que eres un buen amigo.

Y por supuesto a mi madre, por no haberse rendido nunca, porque lo que has conseguido tú sola es increíble. Eres mi modelo a seguir y mi orgullo. Te quiero.

Gracias a todos.

Índice

1.Introducción 8
1.1 "Anima: Beyond Fantasy" y los juegos de rol 8
1.1.1 Evolución de la popularidad de los juegos de rol 8
1.1.2 Fundamentos 8
1.1.3 El manual 9
1.1.4 Ánima: Beyond Fantasy 10
1.2 Motivación 10
1.3 Objetivos 12
1.4 Requisitos 13
1.4.1 Recopilación de opiniones y necesidades 13
1.4.1.1 Resultado de las encuestas 14
1.4.1.2 Conclusiones 14
1.4.2 Requisitos funcionales 16
1.4.3 Requisitos no funcionales 17
1.4.4 Requisitos de información 18
1.5 Estado del arte 22
1.6 Planificación temporal 24
1.6.1 Riesgos 25
1.6.1.1 Identificación y análisis 26
1.6.1.2 Planificación 27
1.6.1.3 Supervisión 27
1.6.3 Comparativa de tiempos finales 27
2. Azure DevOps 29
2.1 Qué es Azure DevOps 29
2.2 Utilización de Azure DevOps en el proyecto 30
2.2.1 Boards 31
2.2.2 Repos 33
2.2.3 Artifacts 34
3. API Rest 35
3.1 Que es API Rest 35
3.2 Utilización de API Rest en el proyecto 36
4. MongoDB 37
4.1 Qué es MongoDB 37

4.2 Base de datos relacional vs no relacional 38
4.3 Utilización de MongoDB en el proyecto 38
5. GraphQL 40
5.1 Qué es GraphQL 40
5.2 Utilización de GraphQL en el proyecto 40
6. Docker 43
6.1 Qué es Docker 43
6.2 Utilización de Docker en el proyecto 44
7. Animáster 48
7.1 Diagrama de casos de uso 48
7.2 Guía de estilo 49
7.3 Maquetado 50
7.4 Entorno de trabajo 60
7.5 Implementación 61
7.6 Persistencia de datos 63
7.7 Despliegue 64
8. Conclusiones 66
8.1 Resultados 66
8.2 Futuras mejoras 66
Anexo 1 – Ficha de personaje 67
Anexo 2 – Encuesta para directores de juego 71
Anexo 3 – Encuesta para Jugadores 72
Anexo 4 – Resultado de las encuestas 73
Bibliografía 79

Índice de Figuras

Fig. 1. Set de dados de rol. Incluye dados de 4, 6, 8, 12, 20 y 100 caras..... 9
Fig. 2. Diagrama de flujo de un ataque. En Morado, las declaraciones, en verde los lanzamientos de dados, en amarillo las cuentas matemáticas y en rojo las consultas al manual. 11
Fig. 3. Ejemplos de tablas de modificadores a tiradas. Se contemplan situaciones que se pueden dar a la hora de calcular ataques y el modificador que ello conlleva en la tirada. 12
Fig. 4. Ejemplo de ficha en Excel 23
Fig. 5. Ejemplo de partida en roll20. A la izquierda, el lanzador de dados, a la derecha, el historial de tiradas, en el centro, un encuentro. 24
Fig. 6. Planificación temporal del proyecto 24
Fig. 7. Ciclo de vida de los riesgos según Sommeville 26
Fig. 8. Creación de un proyecto Agile 30
Fig. 9. Esquema de la plantilla Agile 30
Fig. 10. Panel Kanban de Azure Boards 31
Fig. 11. Backlog 32
Fig. 12. Vista detalle de un ítem 32
Fig. 13. Planificación y gestión del Epic "Recopilación de opiniones" 33
Fig. 14. Opciones disponibles para usar con Repos 34
Fig. 15. Origen establecido en Azure en lugar de Nuget.org 34
Fig. 16. Ejemplo de un feed en funcionamiento 35
Fig. 17. Esquema de un sistema con API [8] 35
Fig. 18. Top de tecnologías/lenguajes que más quieren aprender los desarrolladores 37
Fig. 19. Fragmento del modelo de personaje 39
Fig. 20. Ejemplo de intercambio de datos GraphQL 40
Fig. 21. Consulta de ejemplo GraphQL 41
Fig. 22. Comparativa de tamaño de datos con una consulta completa. Endpoint creado solo para este ejemplo 41
Fig. 23. Query de la consulta para el almacén de personajes 42
Fig. 24. Fragmento de la respuesta a la consulta de la Figura 35 42
Fig. 25. Esquema de estructura de Docker [10] 43
Fig. 26. Flujo de creación de un contenedor 44
Fig. 27. Estructura de la composición del proyecto 44
Fig. 28. Docker-compose del proyecto 45
Fig. 29. Localización del Dockerfile en el proyecto 45
Fig. 30. Dockerfile con la estructura de multi stage building 46
Fig. 31. Librería de imagenes de docker 46
Fig. 32. El stack de docker en funcionamiento 47
Fig. 33. API en funcionamiento desde docker 47
Fig. 34. Mongo Express funcionando desde docker 47
Fig. 35. Diagrama de casos de uso 48
Fig. 36. Guía de estilos 49
Fig. 37. Ejemplo de iconos Segoe MDL2 49
Fig. 38. Página de inicio 50
Fig. 39. Página de inicio con menú de hamburguesa desplegado 51
Fig. 40. Almacén de personajes 51

Fig. 41. Página resumen de la ficha 52
Fig. 42. Página principal de creación de personaje 53
Fig. 43. Creación de ficha, pestaña PDs - Combate 53
Fig. 44. Creación de ficha, pestaña PDs - Místicas 54
Fig. 45. Creación de ficha, pestaña PDs - Psíquicas 54
Fig. 46. Creación de ficha, pestaña PDs – Secundarias 55
Fig. 47. Vista de combate 56
Fig. 48. Vista de cálculo de ataque 57
Fig. 49. Vista de cálculo de ataque. Principio de simetría 57
Fig. 50. Vista de cálculo de ataque. Principios de similitud, agrupación y región común 58
Fig. 51. Vista de cálculo de ataque expandida 59
Fig. 52. Vista de cálculo de ataque a distancia 59
Fig. 53. Vista de cálculo de ataque a distancia expandida 60
Fig. 54. Estructura de .NET 60
Fig. 55. Vista de Windows Template Studio 61
Fig. 56. Organización de un proyecto alimentado por .Net Core 62
Fig. 57. Estructura Model View view-Model 62
Fig. 58. Persistencia 63
Fig. 59. Configuración de la conexión FTP 64
Fig. 60. Publicación mediante FTP 65

1.Introducción

1.1 “Anima: Beyond Fantasy” y los juegos de rol

1.1.1 Evolución de la popularidad de los juegos de rol

Desde su creación en el año 1974 con el juego “Dungeons & Dragons”, el género de juegos de rol ha ido ganando cada vez más popularidad hasta el día de hoy, y no es de extrañar, ya que es un tipo de juego muy peculiar en el que cada grupo de jugadores amolda las reglas de la partida para conseguir algo a gusto de todos. Ya sea al modificar la longitud de partida (que puede llevar desde una sola partida de 4 horas hasta una campaña completa de varios años) hasta el número de jugadores que pueden participar, la mayor ventaja de los juegos de rol es la flexibilidad de las reglas.

Según un artículo de la editorial “Wizards of the Coast” [1], en junio de 2019 se registraron que al menos 40 millones de personas habían jugado alguna vez a “Dungeons & Dragons” y que, además, 9 millones de personas suelen ver retransmisiones de partidas.

Debido a la popularidad de “Dungeons & Dragons” otros han intentado emular su éxito mediante la creación de otros juegos a través de la publicación de sus propios manuales de normas. Pero antes de entrar en detalle con los manuales, veamos los fundamentos de un juego de rol.

1.1.2 Fundamentos

El rol es un juego de mesa sin tablero donde el elemento fundamental es la imaginación. Es una inmensa obra de teatro en la que cada uno de los participantes toma el papel de un personaje que interpreta sin un guion predefinido.

Una partida de rol estándar se compone por un director de juego y los jugadores. Se puede considerar al director de juego como el árbitro de la partida y está encargado de crear una historia para ella. Normalmente la historia gira en torno a un tema que se haya elegido entre todos, aunque también puede ser propuesto solo por el director.

Cada jugador crea entonces un personaje, para el cual se decidirán prácticamente todos los aspectos de este, como personalidad, aspiraciones, historia y características físicas y mentales, todo ello siguiendo la normativa del manual que se esté empleando. Toda esta información viene recogida en su ficha de personaje. Es imprescindible que el personaje sea acorde a la historia que ha planteado del director de juego.

Una vez listo todo se puede iniciar la partida de la cual el director de juego decidirá su transcurso de la forma más imparcial y objetiva que pueda, recordando, por supuesto, que la finalidad principal es que todo el mundo se divierta. Entre sus responsabilidades están la de describir todo lo que hay alrededor de los personajes de los jugadores haciendo de narrador y a la vez interpretar cualquier personaje que no pertenezca a los jugadores, es decir, crear una historia interactiva.

Cada jugador, por su parte, solo puede controlar a su personaje creado. Para controlarlo debe interpretarlo, hablando desde la primera persona cuando su personaje interviene y describiendo las acciones que este haga.

Adicionalmente, para determinar ciertas situaciones, como un personaje siendo capaz de recordar cierto acontecimiento histórico o de asestar un golpe en un combate, es necesario recurrir al azar para mantener la intriga y evitar que el director acarree con todas las decisiones. Por ello lo más común es la utilización de un set de dados (Figura 1). Para esto se lanza un dado (que varía dependiendo del juego y la acción a realizar) y se le suma la habilidad de ese personaje al realizar esa acción. En caso de superar determinado valor, la acción se toma como superada y el director de juego describe la situación. En caso contrario, describe como el personaje ha fallado en el intento.

Fig. 1. Set de dados de rol. Incluye dados de 4, 6, 8, 12, 20 y 100 caras.

1.1.3 El manual

“Juegos de rol” es el género que engloba a todos los juegos basados en el fundamento descrito en el apartado anterior. Dentro de los juegos de rol hay una gran cantidad de juegos distintos como “Dungeons & Dragons”, “Aquelarre”, “Pathfinder” y, por supuesto, el objetivo de este proyecto: “Anima: Beyond Fantasy” [2].

La diferencia que hay entre ellos es el sistema de reglas seguidas, la ambientación, las herramientas que tiene el director a su disposición y la estructura de la ficha del personaje. Toda esta información viene recogida en el manual de juego, el cual puede ser realmente extenso, llegando en algunos casos hasta cientos de páginas. Todas estas páginas debe haberlas leído y comprendido el director de juego para poder crear una ambientación correcta y poder tomar el papel de árbitro. Los jugadores, por su parte, solo deben leer una pequeña parte de las páginas del manual en comparación, que normalmente hacen referencia a la creación de su personaje y les sirve para aprender a jugar su tipo de personaje, dado que si, por ejemplo, su personaje no utiliza armas a distancia, no es necesario que sepa cómo funciona todo el sistema de combate a distancia.

1.1.4 Ánima: Beyond Fantasy

Ánima basa su ambientación en una la épica medieval y da a los jugadores herramientas para crear un sinfín de personajes con completa libertad sobre las reglas del juego. De todos los manuales más populares actualmente en el mercado, es uno de los que más ámbitos contempla con mayor precisión y por ello, de los más complejos de utilizar. Adicionalmente, el manual de Ánima incluye la ficha a papel que se debe utilizar para la creación de personajes la cual se compone de 2 folios por delante y por detrás (Se puede comprobar en el Anexo 1).

El manual "básico" está compuesto por 320 páginas, acompañado de diferentes libros que se centran en todos los aspectos de forma más profunda. Actualmente, los libros que son más populares a la hora de utilizarse en partida son *"Anima: Beyond Fantasy Core Exxet"*, *"Pantalla del director"*, *"Dominus Exxet: Dominios del ki"*, *"Arcana Exxet: Secretos de lo sobrenatural"* y *"Prometeum Exxet"* (aunque este último en menor medida) [3] para un total de 789 páginas. Por supuesto, se puede jugar con menos libros, e incluso solo con el base (Core Exxet), dado que los demás se consideran expansiones, aunque están muy recomendadas.

1.2 Motivación

Este proyecto pretende minimizar los llamados "tiempos muertos" que se producen a la hora de jugar. Un tiempo muerto se da cuando en determinada situación se produce un parón en la partida para determinar el resultado de una tirada excesivamente compleja. Un ejemplo de una tirada muy compleja es el ataque. Normalmente el ataque se realiza con una tirada enfrentada, es decir, un jugador declara que ataca a un personaje (normalmente un enemigo puesto por el director de juego) y debe tirar un dado de 100 caras y sumarle su habilidad de ataque, mientras que el dueño del personaje atacado debe tirar un dado de 100 caras y sumarle su habilidad defensiva. Se debe restar al resultado de la tirada del atacante el resultado de la tirada del defensor para determinar si el ataque impacta o no y con qué efectividad. Inicialmente parece simple, pero se deben de tener en cuenta diversos factores que pueden aumentar el tiempo dedicado a la tirada, como por ejemplo una tirada abierta, en la que si un jugador saca más de 90 (91,92,93… en tiradas abiertas consecutivas) suma ese valor y vuelve a tirar para conseguir una tirada mucho mayor. También puede suceder que el defensor se defienda tan bien que pueda contraatacar, o que se defienda tan mal que reciba un golpe crítico.

La forma de comprobar todo eso no es relevante en esta sección, pero si hay que tenerla en cuenta, puesto que una sucesión de tiradas concreta en un ataque puede elevar el tiempo de cálculo de 10 o 15 segundos a 5 minutos, cosa que en una partida muy centrada en los combates puede llegar a hacerse tedioso. Se puede comprobar en la Figura 2 el diagrama de flujo completo de un ataque cuerpo a cuerpo. El diagrama sigue un código de color para determinar los puntos donde más tiempo se consume. En morado están las declaraciones, que se utilizan para indicar las acciones de un personaje (Por ejemplo, un jugador puede declarar que ataca a un enemigo). Son meros segundos sin importancia. En verde se encuentran las tiradas, las cuales no consumen mucho tiempo, pero si salen tiradas abiertas si se puede extender un poco más. En amarillo cuentas matemáticas, las cuales a veces requieren de comprobar un valor concreto en un campo de la ficha de personaje. Por último, en rojo las consultas al manual. Es muy común necesitar consultar una tabla o una norma, dado que muchas veces son muy extensas como para llegar a aprendérselas de memoria. Un ejemplo de una tabla relacionada con este diagrama se puede comprobar en la Figura 3.

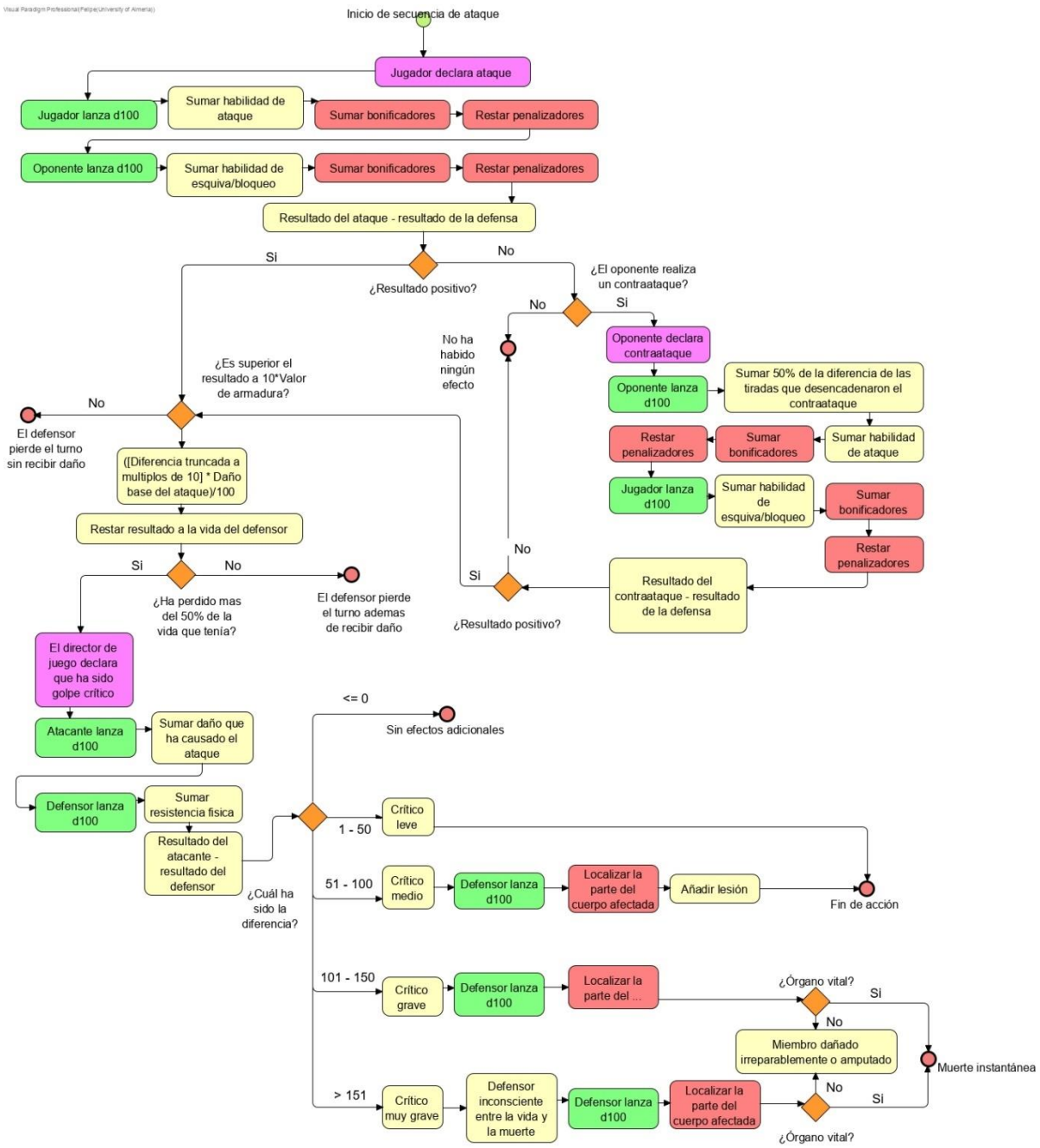


Fig. 2. Diagrama de flujo de un ataque. En Morado, las declaraciones, en verde los lanzamientos de dados, en amarillo las cuentas matemáticas y en rojo las consultas al manual.

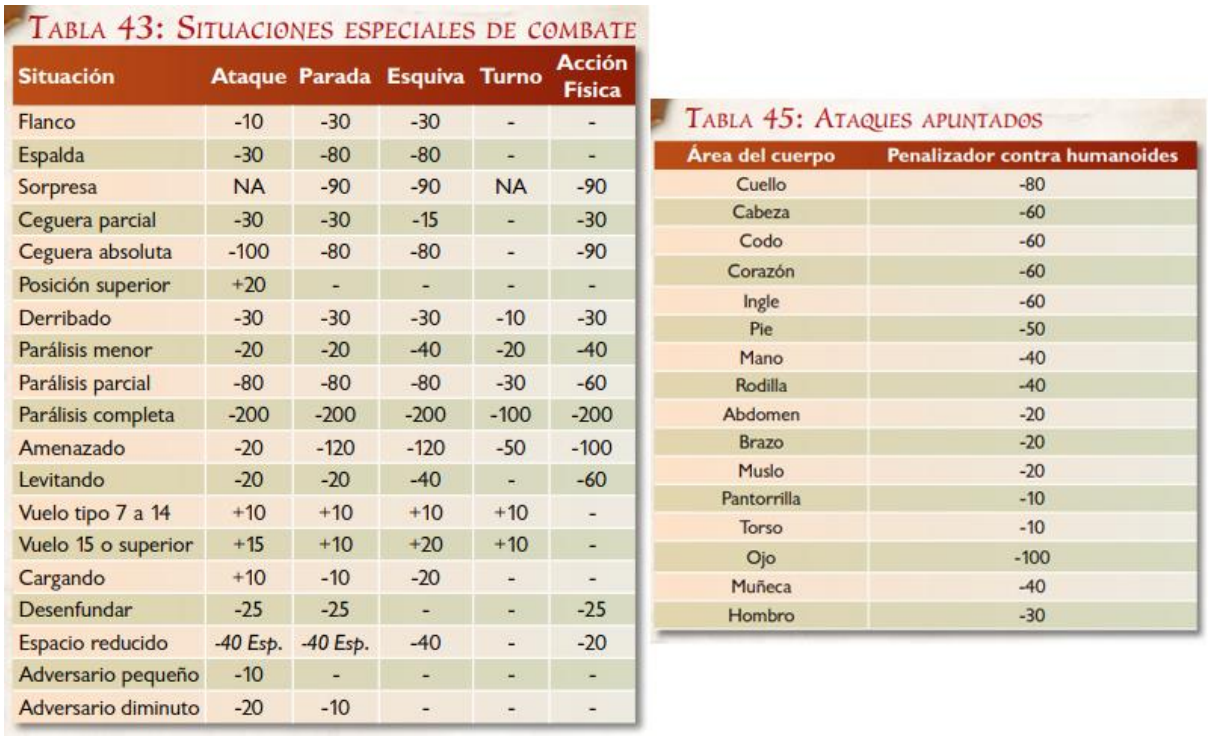


TABLA 43: SITUACIONES ESPECIALES DE COMBATE

Situación	Ataque	Parada	Esquiva	Turno	Acción Física
Flanco	-10	-30	-30	-	-
Espalda	-30	-80	-80	-	-
Sorpresa	NA	-90	-90	NA	-90
Ceguera parcial	-30	-30	-15	-	-30
Ceguera absoluta	-100	-80	-80	-	-90
Posición superior	+20	-	-	-	-
Derribado	-30	-30	-30	-10	-30
Parálisis menor	-20	-20	-40	-20	-40
Parálisis parcial	-80	-80	-80	-30	-60
Parálisis completa	-200	-200	-200	-100	-200
Amenazado	-20	-120	-120	-50	-100
Levitando	-20	-20	-40	-	-60
Vuelo tipo 7 a 14	+10	+10	+10	+10	-
Vuelo 15 o superior	+15	+10	+20	+10	-
Cargando	+10	-10	-20	-	-
Desenfundar	-25	-25	-	-	-25
Espacio reducido	-40 Esp.	-40 Esp.	-40	-	-20
Adversario pequeño	-10	-	-	-	-
Adversario diminuto	-20	-10	-	-	-

TABLA 45: ATAQUES APUNTADOS

Área del cuerpo	Penalizador contra humanoides
Cuello	-80
Cabeza	-60
Codo	-60
Corazón	-60
Ingle	-60
Pie	-50
Mano	-40
Rodilla	-40
Abdomen	-20
Brazo	-20
Muslo	-20
Pantorrilla	-10
Torso	-10
Ojo	-100
Muñeca	-40
Hombro	-30

Fig. 3. Ejemplos de tablas de modificadores a tiradas. Se contemplan situaciones que se pueden dar a la hora de calcular ataques y el modificador que ello conlleva en la tirada.

La principal motivación de este proyecto recae en que Ánima supone para mí un pasatiempo con el que disfrutar algunas tardes con mis amigos desde hace más de 3 años, pero que a veces puede volverse terriblemente tedioso bajo determinadas circunstancias. La complejidad de Ánima hace que se puedan crear partidas muy ricas y detalladas, pero ese detalle cuesta tiempo de calcular correctamente. Por ello, considero que una aplicación que automatice algunos cálculos y les dé a los jugadores algunas comodidades podría ser muy beneficiosa, sobre todo el hecho de tener la ficha en versión digital, dado que en la versión a papel (la intencionada por el manual) es necesario borrar los campos y reescribirlos cada vez que cambian, acto que a la larga puede deteriorar notablemente el papel.

1.3 Objetivos

El objetivo principal es crear una aplicación que sea capaz de automatizar lo máximo posible las principales tareas llevadas a cabo en una partida para reducir los tiempos muertos lo máximo posible con el fin de que resulte atractiva para los jugadores de Ánima. Entre las principales tareas que se pretende que incluya la automatización o ayudas a su uso, están:

- Permitir la creación y modificación de fichas de personaje dentro de la propia app para no necesitar hacer uso de las fichas a papel como se hace normalmente.

- Un almacén de personajes que permita ver las fichas propias o las de la comunidad

- Automatización del sistema de tiradas de dados en ataque y los cálculos que estas causan

- Una interfaz para el director del estado de sus personajes y el de los jugadores.

- Automatización del sistema de turnos.

Además de la creación de la propia aplicación, se considera un objetivo fundamental controlar el desarrollo del proyecto mediante las herramientas proporcionadas por Azure DevOps. Además de esto, se ha desarrollado apoyándose en el uso de una API para controlar el flujo de datos de la comunidad.

La naturaleza modular de la aplicación permitirá que se vayan añadiendo extensiones en caso de que se vaya a seguir ampliando en un futuro, permitiendo que sea fácilmente escalable para futuros manuales o necesidades.

1.4 Requisitos

1.4.1 Recopilación de opiniones y necesidades

Dado que no hay un cliente definido en el proyecto del que recopilar los requisitos, se ha sustituido esta Figura por un estudio sobre el público objetivo (sobre el cual me incluyo) mediante la creación de encuestas, las cuales se pueden encontrar en los anexos 2 y 3. Las encuestas tienen la finalidad de encontrar aquellas partes de una partida donde los jugadores tienen más problemas para crear una solución en la aplicación. Ambas encuestas son parecidas, pero una es para los jugadores y otra para los directores de juego, debido a que tienen un estilo de juego tan diferenciado.

Las encuestas se han dirigido a grupos reducidos de personas de confianza para, de ser necesario, poder entablar con ellos una entrevista basándonos en las respuestas dadas.

Las encuestas se componen de:

- Una descripción inicial de la encuesta y de la importancia de hacerla con tranquilidad, así como el agradecimiento.

- Preguntas de protocolo como nombre, mail, experiencia jugando, frecuencia de las partidas y partidas en curso.

- Una pregunta para determinar la presencia de dispositivos electrónicos en las partidas, algo de gran importancia para tener en cuenta, dado que una partida con dispositivos electrónicos, aunque no sean con ordenador ya hace que los jugadores tengan cierta inclinación inicial a usar una aplicación de escritorio.

- Una serie de preguntas que se han considerado cruciales debido a que son acciones que se dan con cierta normalidad y que algunas puedes consumir bastante tiempo.

- Una pregunta abierta, principalmente para contemplar posibles situaciones que no se habían tenido en cuenta inicialmente.

1.4.1.1 Resultado de las encuestas

Inicialmente, unas valoraciones respecto a los resultados:

- Hay muchos menos directores de juego (4) que jugadores (11), principalmente por el trabajo extra que conlleva preparar las partidas, gestionarlas y tener la responsabilidad de confrontar a los jugadores como árbitros de la partida. Esto lleva a que el número de directores de confianza que se han podido ofrecer a realizar la encuesta sea tan bajo.
- La serie de preguntas relevantes se han hecho en formato de tabla, dándole a los jugadores la posibilidad de responderlas en función de 5 valores, por lo que, para una representación más clara, se ha decidido no utilizar el resultado que proporcionan los formularios de Google y hacerlos manualmente.
- Debido a la extensión ocupada por todas las figuras, se han introducido en el "Anexo 4 – Resultado de las encuestas"

1.4.1.2 Conclusiones

Tras analizar los resultados se pueden sacar algunas conclusiones respecto a Ánima en función a cada pregunta.

- Partidas activas (Anexos 4.3 y 4.4) y tiempo de frecuencia de juego (Anexos 4.5 y 4.6)

Comparando los resultados se puede comprobar que ningún director de juego tiene más de una partida simultáneamente, mientras que hay jugadores que tienen 3 o más. Esto deja ver la carga de trabajo que supone ser director en una partida frente a la que supone ser jugador. Ya sea por dificultad logística de llevar varias partidas o por el tiempo que ello consume, los directores son propensos a tener solo una a la vez. Además, se obtiene un dato interesante. El 50% de los directores encuestados juegan al menos una vez a la semana, frente al 63.7% de los jugadores. Esto podría demostrar que no es falta de ganas de jugar, sino problemas de logística para mantener varias partidas.

- Presencia de dispositivos electrónicos (Anexos 4.7 y 4.8)

Respecto a la presencia de electrónica en las partidas, comprobando los resultados se puede comprobar como en total solo el 6% de las partidas no es común que tengan dispositivos electrónicos. Y no es de extrañar dado que, con el avance de la tecnología, los juegos de rol han empezado a apoyarse cada vez más en el uso de ordenadores para acceder a tableros virtuales, música para dar ambiente a la habitación donde se juega o incluso videollamadas para el desarrollo de la partida, ya sea con amigos, debido a que a veces el desplazamiento puede suponer un problema o con gente al azar, algo que es una práctica muy común mediante el uso de foros. Esto es una buena noticia, dado que, si ya es normal la presencia de ordenadores, intentar introducir una aplicación en las partidas se debería sentir poco intrusivo.

- Serie de preguntas 1 (Anexos 4.9 y 4.10)

Para la serie de preguntas se ha hecho hincapié en las posibilidades que se tienen a la hora de tirar dados. Se puede comprobar cómo, en general, tanto jugadores como directores odian que otra persona tire los dados por ellos. Esto es completamente lógico, puesto que ¿A que no sería lógico estar jugando una partida de parchís y que otra persona tirara los dados en tu lugar? Sucede lo mismo en juegos de rol. A los jugadores les gusta tener esa sensación de control a la hora de entregarse al azar, el saber que "la mala suerte del otro" no influye en sus tiradas. Y es que incluso los jugadores parecen reticentes a que un programa les dé los resultados directamente, ahorrándose todo el tiempo perdido de cuentas entre medias. En cambio, los directores también muestran cierto interés en tirar ellos

mismo los dados, pero a ninguno de los encuestados le disgusta la idea de que un programa les dé el resultado directamente. Esto se debe a las importancias de las jugadas. La única herramienta que tiene para interactuar con el mundo cada jugador es su personaje. Todo lo que influya en él se vuelve automáticamente más importante, todo lo que lo ponga en peligro lo vuelve más tenso, ya que, como en cualquier historia, la muerte es permanente, y la pérdida de su personaje incurre en que un jugador ya no pueda interactuar con la historia. Por ello cada tirada que tiene efecto sobre él es importante.

Por otro lado, están los directores. Normalmente tienen muchos personajes, tantos como personas haya en la historia que está contando, y como en cualquier historia, están los personajes principales, los secundarios, los de relleno… Es normal que no les importen tanto las tiradas que tengan que hacer personajes secundarios, pues están mucho más desapegados de ellos. Y hablando de personajes, todos los personajes deben tener unas estadísticas, por lo que crear los mapas donde se combaten (como se muestra en la Figura 9) se puede volver molesto en ese aspecto, así que una forma de incorporar personajes ya creados podría ser una adición interesante al proyecto.

Otra situación donde la automatización de tiradas se hace más patente es en los combates. Supongamos un combate de 4 vs. 4. El director de juego tiene 4 personajes que controlar con los que se enfrentara a los jugadores. Ellos, por su parte, tienen uno cada uno. Contando de que cada personaje ataca y se defiende una vez, cada jugador hará dos tiradas por ronda, las cuales tienen enorme importancia en su personaje, que puede salir malherido. En cambio, el director tendrá que hacer 8 tiradas cada ronda. Recordando la Figura 2 se puede entender por qué los directores prefieren un sistema que les dé los resultados automáticos. Tienen muchas más tiradas cada turno y son tiradas que afectan a personajes que solo están ahí para pelear contra los jugadores, fácilmente reemplazables y sin ningún cariño por ellos.

Está claro que se necesita un sistema de automatización de tiradas, pero que permita tener en cuenta las tiradas de los jugadores, es decir, que automatice tan solo parcialmente las tiradas para acelerar el lado del director de juego, pero sin afectar a la diversión de los jugadores.

- Serie de preguntas 2 (Anexos 4.11 y 4.12)

De la serie de preguntas 2 también se sacan algunas conclusiones. Hay una diferencia enorme entre el tedio que sufren los jugadores y el que sufren los directores. Se hace palpable tan solo con echarle un vistazo a las Figuras. Mientras que la de los jugadores están bastante balanceadas y predomina el verde (entretenimiento), la Figura de los directores de juego está predominantemente roja/anaranjada, y es por los mismos motivos que los arriba mencionados. La cantidad de cosas que debes controlar. Cuando eres un jugador, controlas tu personaje, cuando eres un director, controlas muchos, incluso tienes que mantener un ojo sobre los personajes de tus jugadores para asegurarte de que nada se sale de lo normal. Esto hace que, si ya para un jugador hay ciertas partes que se vuelven tediosas, para un director hay muchas cosas que podrían mejorarse. Y es ahí donde se pone el punto de mira en este proyecto. Si el director de juego está ocupado mirando fichas, gestionándolas, controlando a todo el mundo y haciendo cuentas matemáticas con las tiradas, la partida no puede avanzar, dado que él es el eje fundamental de la historia y es el narrador. Es necesario aliviar sobre todo la carga de trabajo de los directores de juego, ya que esto conlleva aumentar el dinamismo de la partida para todo el mundo. Para ello hay que fijarse en las partes más tediosas de la serie de preguntas 2. Tener una visión general del estado de todos los personajes en combate, saber el contenido del inventario de los jugadores, el sistema de turnos de combate (explicado más adelante), gestionar las tiradas complejas, reducir el tiempo de los turnos de combate… son cosas absolutamente fundamentales para las partidas que a la vez aburren a los directores. Es necesario mejorarlas.

1.4.2 Requisitos funcionales

Los requisitos funcionales que se han extraído para este proyecto son los siguientes:

RF – 01	Tiradas automáticas
Fuente	Encuestas
Descripción	La aplicación deberá automatizar las tiradas más comunes en un combate y añadir una ayuda de posible interpretación de la tirada.
Consideraciones	Es necesario añadir opciones para determinar como de automatizada se desea la tirada, con la posibilidad de añadir valores manualmente (como las tiradas de los jugadores, ya que suelen gustarles hacerlas a mano). Para la ayuda a la interpretación se seguirán las directrices marcadas por el manual

RF – 02	Creación y modificación de fichas de personaje
Fuente	Encuestas
Descripción	La aplicación deberá incluir un sistema de creación de fichas de personajes regido por las normas del manual. Dicha ficha deberá ser visualizable, modificable, compartible y eliminable.
Consideraciones	Ninguna

RF – 03	Almacén de personajes
Fuente	Encuestas
Descripción	La aplicación deberá incluir un sistema para almacenar y ordenar las fichas creadas. También deberá incluir una opción para ver las fichas creadas por la comunidad y hacer una copia personal.
Consideraciones	El visionado del almacén debe ser rápido y por tanto cada ficha en él será un resumen que, al seleccionarlo, llevará a la ficha completa.

RF – 04	Estados de los personajes
Fuente	Encuestas
Descripción	La aplicación deberá tener un sistema de control de los estados de todos los personajes de los jugadores y de sus oponentes, en caso de que los haya.
Consideraciones	Se considera oponente a todo aquel personaje creado por el director de juego cuyas características necesiten ser tenidas en cuenta. Inicialmente es para la simulación de combates, pero se puede utilizar para tenerlos a mano.

RF – 05	Turnos de combate automatizados
Fuente	Encuestas
Descripción	La aplicación deberá calcular automáticamente el orden de juego en un combate de todos los participantes.
Consideraciones	Un participante es cualquier personaje que haya sido añadido al sistema de control de estados (RF - 04)

1.4.3 Requisitos no funcionales

RNF – 01	Usabilidad
Fuente	Encuestas
Descripción	La aplicación debe ser intuitiva y su uso debe resultar más fácil y rápido que la forma de jugar convencional
Consideraciones	Ninguna

RNF – 02	Eficiencia de datos
Fuente	Valoración personal
Descripción	La aplicación debe extraer los datos de las fichas de la comunidad de forma eficiente, ya que se puede acumular un gran número.
Consideraciones	Ninguna

RNF – 03	Reusabilidad
Fuente	Valoración personal
Descripción	Este proyecto se debe llevar a cabo teniendo en cuenta que se realizará una aplicación similar en el futuro
Consideraciones	Se ha previsto una posible versión solo para jugadores que esté conectada con Animáster, por lo que todo lo que se pueda deberia ser planteado para ser reutilizable, a fin de mantener una consistencia en ambas aplicaciones

RNF – 04	Disponibilidad
Fuente	Valoración personal
Descripción	La api debe estar disponible en todo momento
Consideraciones	Una funcionalidad de la aplicación es conseguir fichas hechas por algún miembro de la comunidad en cualquier momento, dado que debido a la alta flexibilidad de las partidas esa necesidad se puede generar cuando menos se espera. La api de donde se extraen las fichas debe estar siempre disponible para estar a la altura de los improvistos

1.4.4 Requisitos de información

A continuación, se listan los requisitos de información

RI - 1	Personaje
Requisitos funcionales asociados	RF-01, RF-02, RF-03, RF-04, RF-05
Requisitos no funcionales asociados	RNF-02, RNF-03
Descripción	Un personaje contendrá toda la información referente a él según estipula la ficha oficial de Ánima.
Datos	❏ ID ❏ autor: string ❏ nombre: string ❏ clase: string ❏ nivel: long ❏ tamaño: long ❏ lore: string ❏ inventario: List<string> ❏ dinero: List<long> ❏ principales: Principales ❏ estado: Estado ❏ combate: Combate ❏ nivelDeMagia: long ❏ nivelDeMagiaUsado: long ❏ metamagia: long ❏ PDs: PDs
Comentario	Es el eje central de la aplicación. El campo “autor” es el UUID mencionado al final de esta sección

RI - 2	Estado
Requisitos funcionales asociados	RF-01, RF-02, RF-03, RF-04, RF-05
Requisitos no funcionales asociados	RNF-02, RNF-03
Descripción	El estado actual de un personaje
Datos	❏ pdVMax: long ❏ pdV: long ❏ cansancioMax: long ❏ cansancio: long ❏ cansancioMod: long ❏ zeonMax: long ❏ zeon: long ❏ zeonAcumulado: long

	❏ kiMax: long ❏ ki: long ❏ CVLibresMax: long ❏ CVLibres: long
Comentario	Son, en parte, valores finales obtenidos a partir del “RI-3 PDs” y aplicandole diversas modificaciones

RI – 3	Combate
Requisitos funcionales asociados	RF-01, RF-02, RF-03, RF-04, RF-05
Requisitos no funcionales asociados	RNF-02, RNF-03
Descripción	Los atributos de combate de un personaje determinado
Datos	❏ habAtaque: long ❏ habEsquiva: long ❏ habParada: long ❏ proyMagicaOf: long ❏ proyMagicaDef: long ❏ desequilibrioOfensivo": long ❏ proyeccionPsiquica: long ❏ llevarArmadura: long ❏ acciones: long ❏ turno: List<long> ❏ movimiento: long ❏ movimientoMod: long ❏ regeneracion: long ❏ regeneracionMod: long ❏ armaDesarrollada: String ❏ TA: long
Comentario	Son, en parte, valores finales obtenidos a partir del “RI-4 PDs” y aplicándole diversas modificaciones

RI – 4	PDs
Requisitos funcionales asociados	RF-01, RF-02, RF-03, RF-04, RF-05
Requisitos no funcionales asociados	RNF-02, RNF-03
Descripción	La distribución de puntos de desarrollo de un determinado personaje
Datos	❏ ataque: List<long> ❏ parada: List<long> ❏ esquiva: List<long> ❏ llevarArmadura: List<long>

	❏ AGIPtKi: List<long> ❏ CONPtKi: List<long> ❏ DESPtKi: List<long> ❏ FUEPtKi: List<long> ❏ PODPtKi: List<long> ❏ VOLPtKi: List<long> ❏ AGIAcuKi: List<long> ❏ CONAcuKi: List<long> ❏ DESAcuKi: List<long> ❏ FUEAcuKi: List<long> ❏ PODAcuKi: List<long> ❏ VOLAcuKi: List<long> ❏ CM: List<long> ❏ zeon: List<long> ❏ ACT: List<long> ❏ multiploRegen: List<long> ❏ proyeccionMagica: List<long> ❏ nivelDeMagia: List<long> ❏ convocar: List<long> ❏ controlar: List<long> ❏ atar: List<long> ❏ desconvocar: List<long> ❏ CV: List<long> ❏ proyeccionPsi: List<long> ❏ Secundarias: Secundarias
Comentario	Ninguno

RI – 5	Principales
Requisitos funcionales asociados	RF-01, RF-02, RF-03, RF-04, RF-05
Requisitos no funcionales asociados	RNF-02, RNF-03
Descripción	Las caracteristicas principales de un personaje determinado
Datos	❏ AGI: List<long> ❏ CON: List<long> ❏ DES: List<long> ❏ FUE: List<long> ❏ INT: List<long> ❏ PER: List<long> ❏ POD: List<long> ❏ VOL: List<long>
Comentario	Ninguno

RI - 6	Secundarias
Requisitos funcionales asociados	RF-01, RF-02, RF-03, RF-04, RF-05
Requisitos no funcionales asociados	RNF-02, RNF-03
Descripción	Las características secundarias de un personaje determinado
Datos	❏ acrobacias: List<Long> ❏ atletismo: List<Long> ❏ montar: List<Long> ❏ nadar: List<Long> ❏ trepar: List<Long> ❏ saltar: List<Long> ❏ pilotar: List<Long> ❏ estilo: List<Long> ❏ intimidar: List<Long> ❏ liderazgo: List<Long> ❏ persuasion: List<Long> ❏ comercio: List<Long> ❏ callejeo: List<Long> ❏ etiqueta: List<Long> ❏ advertir: List<Long> ❏ buscar: List<Long> ❏ rastrear: List<Long> ❏ animales: List<Long> ❏ ciencia: List<Long> ❏ ley: List<Long> ❏ herbolaria: List<Long> ❏ historia: List<Long> ❏ tactica: List<Long> ❏ medicina: List<Long> ❏ memorizar: List<Long> ❏ navegacion: List<Long> ❏ ocultismo: List<Long> ❏ tasacion: List<Long> ❏ valoracionMagica: List<Long> ❏ frialdad: List<Long> ❏ proezaDeFuerza: List<Long> ❏ resDolor: List<Long> ❏ cerrajeria: List<Long> ❏ disfraz: List<Long> ❏ ocultarse: List<Long> ❏ robo: List<Long> ❏ sigilo: List<Long> ❏ tramperia: List<Long>

	❏ venenos: List<Long> ❏ arte: List<Long> ❏ baile: List<Long> ❏ forja: List<Long> ❏ runas: List<Long> ❏ alquimia: List<Long> ❏ animismo: List<Long> ❏ musica: List<Long> ❏ trucoDeManos: List<Long> ❏ caligrafiaRitual: List<Long> ❏ orfebreria: List<Long> ❏ confeccion: List<Long> ❏ conMarionetas: List<Long> ❏ multVida: List<Long> ❏ especial: List<Long>
Comentario	Ninguno

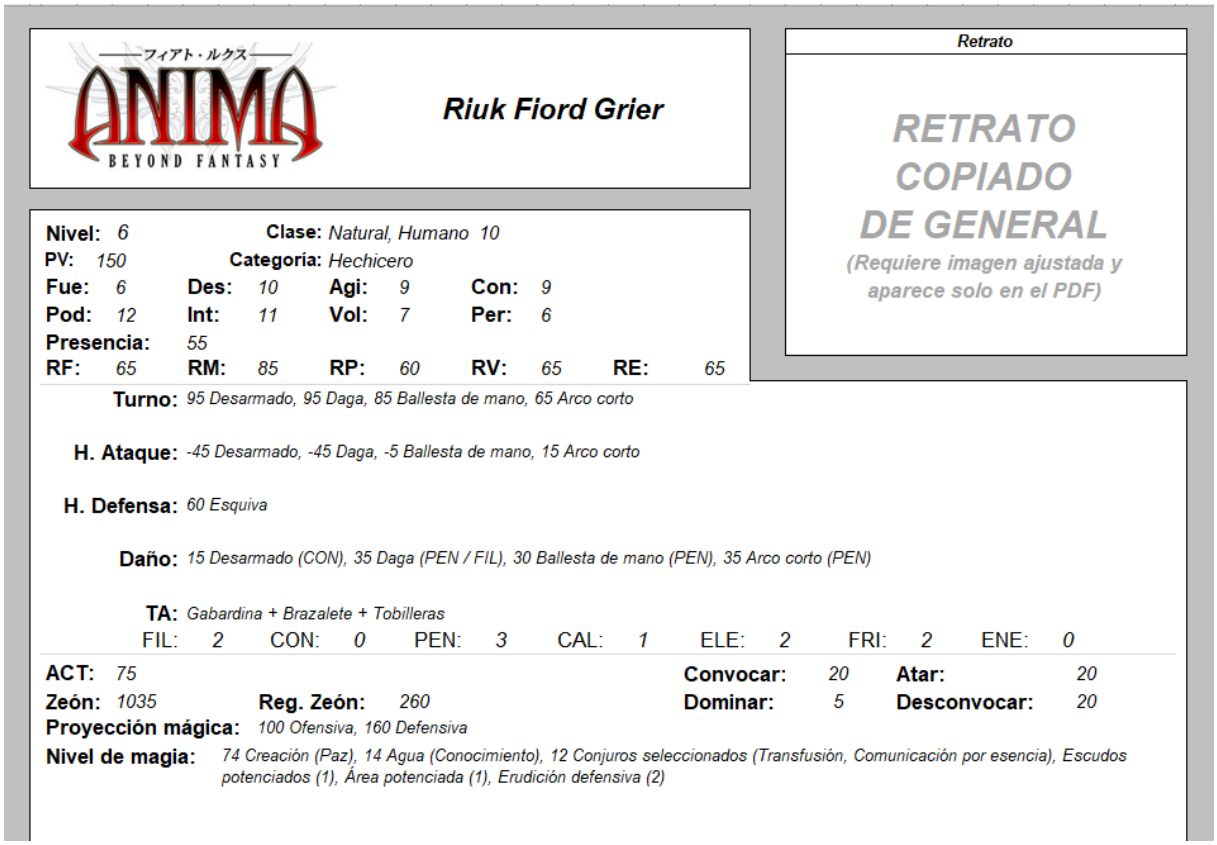
Se puede observar que no hay usuarios en la aplicación. Esto se debe a que a cada usuario de la aplicación se le asignará automáticamente un UUID. Un UUID (identificador único universal o *universally unique identifier* en inglés) es un número de 16 bytes representados en forma canónica mediante 32 dígitos hexadecimales separados por guiones en un grupo de ocho, tres de cuatro y uno de doce, lo cual lleva a un total de $3{,}4 \times 10^{38}$ posibilidades distintas. En promedio, sería necesario generar 2.7 trillones de UUIDs para que se diera una sola colisión. Para ponerlo en perspectiva: Si cada persona de la tierra generara un UUID al día, se tardarían, en promedio, 382 millones de años para generar una colisión. No va a haber tantos usuarios en la aplicación ni la civilización moderna va a durar tanto tiempo, por lo que se puede decir que el riesgo de colisión es prácticamente 0.

1.5 Estado del arte

Actualmente existen muy pocas alternativas a la hora de jugar Ánima.

La primera y la que recomienda el manual es utilizar la ficha a papel. Este método plantea un serio problema: La cantidad de variables para tener en cuenta al crearla. Es tal la cantidad de variables y lo que se tarda en crear la ficha como tal que normalmente los jugadores tienen que prepararla antes de quedar para jugar, porque pueden tardar varias horas en completarla. Todas las características están conectadas entre ellas de una u otra forma y cambiar una en cierto momento te obliga a revisar todos los lugares donde afecta el cambio, borrarlo y reescribirlo. Además, cosas como la salud del personaje, que suele variar constantemente, hace que su casilla se emborrone al poco tiempo de uso.

Para remediar esto, un grupo de jugadores creó y compartió un Excel que facilitaba en cierta medida la creación de la ficha, pero presenta numerosos bugs que siguen sin solucionar y no permite llevar una visión general del personaje, ya que toda la información relevante está desperdigada por varios libros del Excel sobre los que hay que buscar. Adicionalmente, esta ficha no soluciona los problemas que tienen los directores de juego, ya que no permite automatizar tiradas, no puede mirar los personajes de sus jugadores y para tener varios personajes propios a la vez se requiere de tener una ficha abierta por cada personaje.



フィアト・ルクス
ANIMA
BEYOND FANTASY

Riuk Fiord Grier

Retrato

RETRATO COPIADO DE GENERAL
(Requiere imagen ajustada y aparece solo en el PDF)

Nivel: 6 **Clase:** Natural, Humano 10
PV: 150 **Categoría:** Hechicero
Fue: 6 **Des:** 10 **Agi:** 9 **Con:** 9
Pod: 12 **Int:** 11 **Vol:** 7 **Per:** 6
Presencia: 55
RF: 65 **RM:** 85 **RP:** 60 **RV:** 65 **RE:** 65

Turno: 95 Desarmado, 95 Daga, 85 Ballesta de mano, 65 Arco corto

H. Ataque: -45 Desarmado, -45 Daga, -5 Ballesta de mano, 15 Arco corto

H. Defensa: 60 Esquiva

Daño: 15 Desarmado (CON), 35 Daga (PEN / FIL), 30 Ballesta de mano (PEN), 35 Arco corto (PEN)

TA: Gabardina + Brazalete + Tobilleras
FIL: 2 CON: 0 PEN: 3 CAL: 1 ELE: 2 FRI: 2 ENE: 0

ACT: 75 **Convocar:** 20 **Atar:** 20
Zeón: 1035 **Reg. Zeón:** 260 **Dominar:** 5 **Desconvocar:** 20
Proyección mágica: 100 Ofensiva, 160 Defensiva
Nivel de magia: 74 Creación (Paz), 14 Agua (Conocimiento), 12 Conjuros seleccionados (Transfusión, Comunicación por esencia), Escudos potenciados (1), Área potenciada (1), Erudición defensiva (2)

Fig. 4. Ejemplo de ficha en Excel

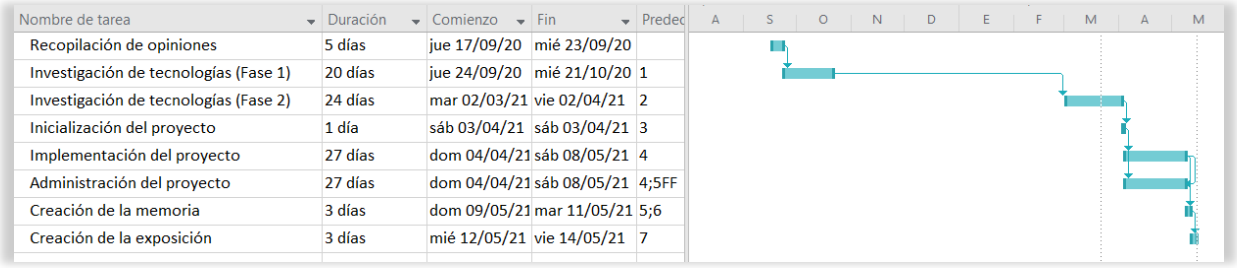
La única herramienta de la que dispone un director de juego para hacer su parte es roll20 [4].

Roll20 es una página web gratuita que permite a los directores organizar partidas. El principal y mayor problema es que está creada para “Dungeons&Dragons”, por lo que la enorme mayoría de sus funcionalidades son incompatibles con Ánima, dejando solamente la posibilidad de poner música, crear mapas para los combates y hacer tiradas realmente simples en las que seleccionas el dado y suma o resta el valor que le indiques, por lo que no ahorra realmente ningún tiempo a la hora de hacer los cálculos. Adicionalmente, utilizar o no roll20 no excluye de la necesidad de tener que elegir la ficha a papel o el Excel para controlar los personajes.

Fig. 5. Ejemplo de partida en roll20. A la izquierda, el lanzador de dados, a la derecha, el historial de tiradas, en el centro, un encuentro.

1.6 Planificación temporal

El proyecto se ha organizado siguiendo el siguiente esquema:



Nombre de tarea	Duración	Comienzo	Fin	Predec
Recopilación de opiniones	5 días	jue 17/09/20	mié 23/09/20	
Investigación de tecnologías (Fase 1)	20 días	jue 24/09/20	mié 21/10/20	1
Investigación de tecnologías (Fase 2)	24 días	mar 02/03/21	vie 02/04/21	2
Inicialización del proyecto	1 día	sáb 03/04/21	sáb 03/04/21	3
Implementación del proyecto	27 días	dom 04/04/21	sáb 08/05/21	4
Administración del proyecto	27 días	dom 04/04/21	sáb 08/05/21	4;5FF
Creación de la memoria	3 días	dom 09/05/21	mar 11/05/21	5;6
Creación de la exposición	3 días	mié 12/05/21	vie 14/05/21	7

Fig. 6. Planificación temporal del proyecto

A continuación, se muestra una breve explicación de cada una de las fases.

- Recopilación de opiniones (5 horas): Mediante el uso de encuestas de Google se realizan una batería de preguntas dirigidas al público objetivo de la aplicación, a fin de obtener opiniones diversas sobre las necesidades reales de los jugadores para perfilar asi el resultado final de la aplicación. Las 5 horas se reparten a lo largo de una semana, para dar tiempo a los encuestados a contestar. Esta etapa abarca desde el 17/09/2020 al 23/09/2020

- Investigación de tecnologías (85 horas): En esta etapa se realizan investigaciones referentes a las distintas tecnologías que se pretenden usar en este proyecto: UWP, GraphQl, Mongo DB y Docker. Debido a la influencia en el horario académico de las prácticas de empresa, hay una pausa que divide esta etapa en periodos de fechas, referenciadas como fase 1 y 2. La primera fase, de aproximadamente 40 horas, abarca desde el 24/09/2020 al 21/10/2020 y la segunda, de las 45 restantes, desde el 2/03/2021 hasta el 2/04/2021.

- Inicialización del proyecto (10 horas): Se comienza la gestión del proyecto iniciándolo en Azure DevOps, preparando los hitos, las issues relacionadas y las definiciones de estas teniendo en cuenta las necesidades reales del público objetivo de la aplicación. Esta etapa abarcará un solo día, el 3/04/2021.

- Implementación del proyecto (150 horas): Esta etapa incluye todo lo relacionado con la implementación de las distintas partes de la aplicación, hasta la finalización de esta. Abarca desde el día 4/04/2021 hasta el 8/05/2021

- Administración del proyecto (15 horas): Esta etapa son horas dedicadas a la gestión de los issues y la documentación para futuras referencias. Se intercalarán estas horas según se vayan necesitando en la etapa “Implementación del proyecto”.

- Creación de la memoria (20 horas): Se genera la memoria a entregar. Abarca desde el día 9/05/2021 hasta el 11/05/2021. Creación de la exposición (15 horas): Etapa final en la que, recogiendo el trabajo realizado, se prepara la exposición a defender ante el tribunal del TFG.

- Creación de la exposición (15 horas): Etapa final en la que recogiendo todo el trabajo realizado se prepara la exposición a defender ante el tribunal del TFG.

1.6.1 Riesgos

Un riesgo es un problema potencial que se puede llegar a hacer realidad en un proyecto. Por ello, en la etapa de inicio del proyecto es necesario plantearlos para poder planear una forma de abordarlos.

Según Ian Sommerville [5] la correcta gestión de riesgos se divide en:

- Identificación: Actividad enfocada a descubrir los posibles riesgos del proyecto al estudiando los aspectos contextuales del mismo.
- Análisis: Se estudia el conjunto para determinar la prioridad por la cual deberían ser monitorizados. En este estudio se incluye el tipo de riesgo, la probabilidad de ocurrencia de este y el impacto.
- Planificación: Se crean los planes para los riesgos que, tras el análisis, se les han sido asignados suficiente prioridad para considerarse relevantes.
- Supervisión: Se establecen mecanismos de evaluación tanto periódica como dirigida por eventos para comprobar el impacto en todo momento.

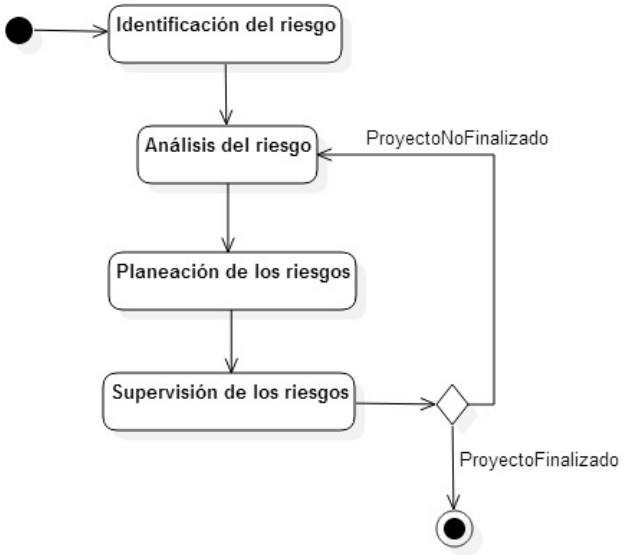


Fig. 7. Ciclo de vida de los riesgos según Sommeville

1.6.1.1 Identificación y análisis

Se han identificado y analizado una serie de riesgos, a los cuales, siguiendo la metodología de Sommerville se han asignado una probabilidad de ocurrencia por franjas de la siguiente forma:

Muy bajo (<10%), bajo (10-25%), moderado (25-50%), alto (50-75%) o muy alto (>75%).

El impacto se ha valorado como catastrófico, serio, tolerable o insignificante.

Riesgo	Categoría	Probabilidad de ocurrencia	Impacto
Estimación de las horas incorrecta debido a la inexperiencia	Riesgo técnico	Alto	Tolerable
Alta complejidad de las tecnologías abordadas	Riesgo del proyecto, riesgo técnico	Moderado	Serio
Inexperiencia en el entorno escogido	Riesgo técnico	Muy alto	Serio
Alto impacto de las asignaturas del curso académico	Riesgo del proyecto	Muy alto	Serio
Oferta de trabajo durante el curso académico	Riesgo personal	Moderado	Serio

1.6.1.2 Planificación

La planificación de los riesgos que se ha determinado ha sido adoptar una estrategia mixta de minimización y contingencia.

Una estrategia de minimización se centra en adoptar las medidas necesarias para bajar todo lo posible el impacto y la probabilidad de ocurrencia. Por su parte, una estrategia de contingencia aprovecha la supervisión de riesgos para aplicar alternativas.

En este proyecto se ha decidido iniciarlo con tiempo de sobra para tener margen y, si es posible, realizar la entrega del TFG en mayo. En caso de que los riesgos se vayan cumpliendo según se van monitorizando, se puede dar el caso de que se active el plan de contingencia, que se basa en aprovechar el tiempo extra con el que se inició el proyecto y adherirle el tiempo que se gana al entregar el TFG en junio/julio.

1.6.1.3 Supervisión

Gracias a las herramientas proporcionadas por Azure DevOps (explicadas más adelante) es posible llevar un seguimiento claro de los riesgos, ya que en este caso en concreto todos afectan a la planificación temporal al no tener presupuesto ni clientes.

Por ello, el plan de supervisión se basará en ir comprobando los riesgos según se vayan actualizando los ítems del backlog y en caso de que las horas estimadas restantes superen al tiempo restante para la entrega inicial, pasarlo a la siguiente.

1.6.3 Comparativa de tiempos finales

A continuación, se mostrarán todas las tareas con la predicción inicial de horas de trabajo que se habían estimado en comparación a las horas que realmente se han necesitado y, además, una pequeña observación donde se ha considerado necesario. Todo con el apoyo de Azure DevOps.

Nombre de la tarea	Tiempo estimado	Tiempo real invertido	Comentarios
Creación de las encuestas	3 h	3 h	
Distribución de las encuestas	1 h	1 h	
Análisis de resultados	1 h	2 h	
Investigación: UWP	30 h	42 h	Ha habido numerosas dificultades, sobre todo en referente a problemas con las versiones de los paquetes NuGet utilizados
Investigación: MongoDB	20 h	35 h	Incluye la puesta en marcha de la base de datos y los numerosos cambios que ha sufrido
Investigación: API	10 h	30 h	Dificultad a la hora de implementarla por el desconocimiento en la materia al inicio del proyecto
Investigación: GraphQL	10 h	30 h	Gran cantidad de problemas a la hora de implementarlo en .NET sobre todo por ser algo relativamente nuevo y con poco soporte encontrado al respecto
Azure: Creación de los Work items	4 h	4 h	

Investigación: Docker	5 h	16 h	He encontrado algunos problemas al hacer funcionar Docker con .Net, debido a que lo hace automáticamente pero concretamente con MongoDB daba problemas
Azure: Actualización de los Work ítems	21 h	16 h	Por suerte, menos trabajo del esperado, gracias a las facilidades que aporta Azure DevOps
Análisis del problema	5 h	14 h	A pesar de mi experiencia en el dominio, he encontrado dificultades para analizar todas las relaciones que contiene la ficha, debido a su enorme cantidad y complejidad
Bocetado	10 h	25 h	Como en el análisis, muchos campos que crear, muchas cosas que tener en cuenta y muchos cambios a lo largo del proyecto
US: Creación/modificación de fichas	60 h	85 h	Mi inexperiencia en el manejo de .Net ha tomado factura en el numero de horas que he tenido que emplear a la hora de crear la aplicación de escritorio, eso sumado a la enorme cantidad de parámetros a tener en cuenta a la hora de crear la interfaz
US: Almacén de personajes	15 h	22 h	
US: Calculo de turnos	5 h	7 h	
US: Control de estado	25 h	39 h	
US: Automatización de tiradas	25 h	15 h	
Creación de la memoria	20 h	58 h	20 horas para la creación de una memoria de semejante magnitud, con todos los diagramas e imágenes que he tenido que crear, era realmente optimista
Horas totales	300 h	444 h	A parte del hecho de que, efectivamente, se han cumplido la mayoría de riesgos que se habían previsto, mi inexperiencia a la hora de estimar las horas de trabajo ha sido el mayor factor a tener en cuenta

2. Azure DevOps

2.1 Qué es Azure DevOps

Azure DevOps [6], nacido de los términos ingleses “Development operations”, es un conjunto de herramientas y servicios que ayuda a administrar los proyectos de desarrollo de software en todas las fases de su ciclo de vida.

Está dividido en 5 herramientas especializadas:

- Boards: Se podría considerar el núcleo del proyecto, pues permite controlar todo el proceso, las tareas, los equipos, los individuos. Gracias a esto se puede monitorizar de forma eficiente el avance del trabajo y el estado general del proyecto en todo momento. Es flexible y altamente personalizable a las necesidades.
- Repos: El sistema de repositorios públicos y privados de Azure que, además, permite enlazar los commits con el backlog de los Boards incluso desde VisualStudio, para tener un control total sobre las versiones.
- Artifacts: Sistema de control de paquetes NuGet (Librerías) y sus versiones que funciona a nivel de proyecto. Desde aquí, permite que todos los miembros del equipo que participan en el proyecto mantengan de forma automática las mismas versiones y librerías para reducir los problemas de incompatibilidad al mínimo.
- Test plans: Proporciona las herramientas para la automatización de las pruebas unitarias, ya sea para aplicaciones de escritorio, web o móvil.
- Pipelines: Un automatizador de compilación e implementación que funciona con cualquier lenguaje y es compatible con GitHub

Además, Azure DevOps proporciona soporte tanto a proyectos ágiles como a predictivos.

Con una cuenta gratuita de Microsoft se puede acceder a la plataforma de Azure DevOps y crear un nuevo proyecto. Para crearlo (Figura 8), solo se necesita el nombre del proyecto, una descripción, seleccionar la privacidad, el control de versiones y una de las 4 plantillas disponibles, que son:

- Basic: Para proyectos sencillos
- Agile: Para proyectos complejos donde se va a utilizar historias de usuarios, bugs y tareas.
- Scrum: Permite trabajar con metodología Scrum, backlog, seguimiento de proyectos, bugs, task board, etc.
- CMMI: Para proyectos con metodología CMMI, que permite la mejora de los procesos proporcionando los mejores elementos y tareas para llevarlos a cabo.

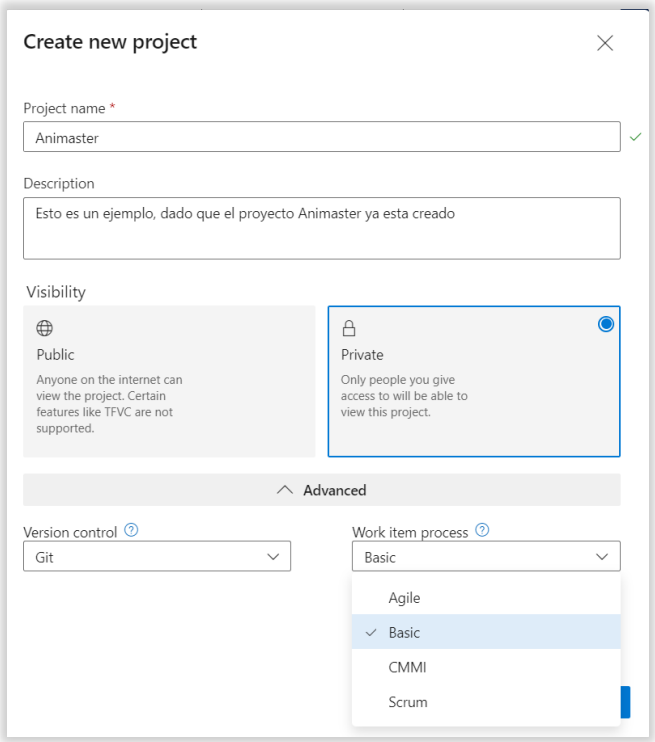


Fig. 8. Creación de un proyecto Agile

2.2 Utilización de Azure DevOps en el proyecto

En el caso de este proyecto, se ha decidido utilizar la plantilla de metodología Agile [7]. Esta plantilla elabora unas preconfiguraciones al crear el proyecto basadas en Epics, Features, User Stories, Task, Issues y Bugs (Figura 9).

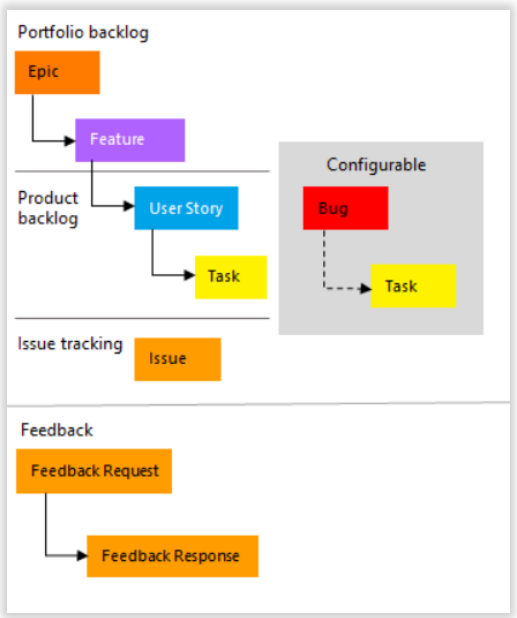


Fig. 9. Esquema de la plantilla Agile

Los tipos de ítems que presentan la plantilla han sido usados en este proyecto de la siguiente forma:

- Epic: Hace referencia a cada fase de desarrollo descrita en la planificación temporal, los cuales están compuestos de Features.

- Features: Cumplen una doble función. Fuera del Epic “Implementación del proyecto” distingue las áreas de las que se compone (como podrían ser las diferentes tecnologías de la fase “Investigación de tecnologías”) y están compuestos de Tasks. Dentro del Epic de implementación definen los distintos módulos en los que se divide la aplicación y estan compuestos de User Stories.

- User Story: Representa un requisito para la aplicación y está compuesta de Tasks.

- Task: Representa la unidad mínima de trabajo, y puede ir desde el diseño de la interfaz para un requisito determinado hasta la creación de las preguntas de una encuesta.

- Issues: Utilizada para los problemas detectados durante la implementación, posibles mejoras o cambios que se planean abordar en algún momento.

2.2.1 Boards

La primera herramienta que se va a explicar es también el eje central de Azure, como se ha explicado en el apartado 2.1

Nada más abrir el proyecto se nos muestra la típica vista de Kanban (Figura 10) en la que se nos muestran todos los ítems de un determinado tipo. La vista se compone del panel en sí, un filtro para buscar ítems en concretos y la barra de configuración, con el selector de panel.

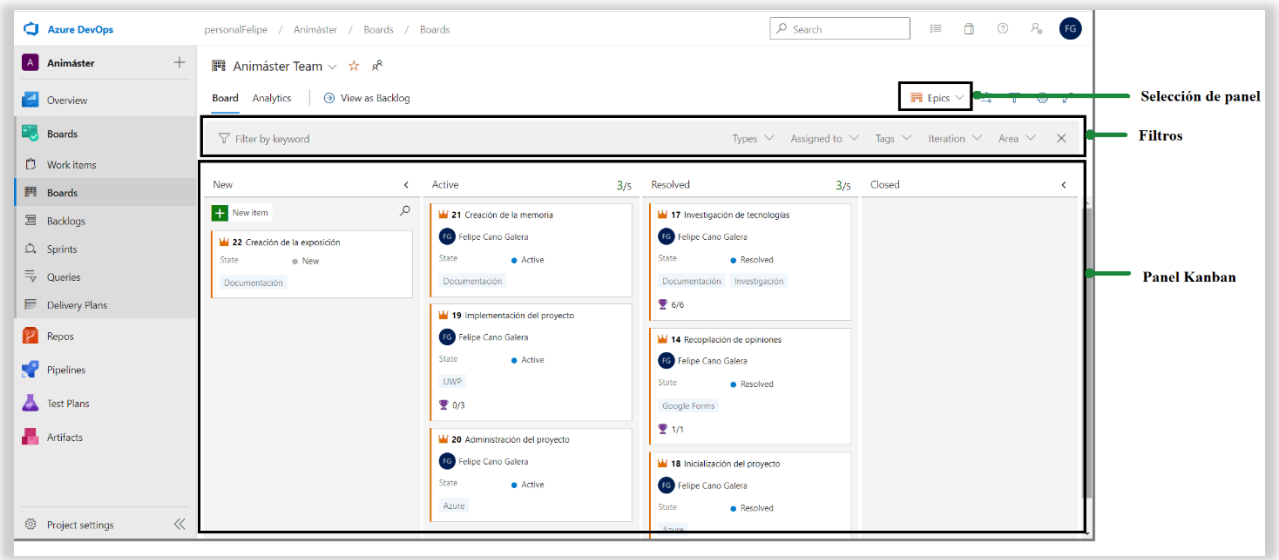


Fig. 10. Panel Kanban de Azure Boards

La vista de los Boards que se ha usado en el proyecto es el Backlog. En el backlog (Figura 11), como se puede intuir por el nombre, se listan todas las tareas que se pretenden realizar en un proyecto, así como la posibilidad de configurarlo al gusto. En la Figura se puede comprobar como he dividido el backlog en las columnas de nombre del ítem, estado, esfuerzo (que en este caso se mide en horas), tags, progreso de los ítems hijos, Value Area (Architectural si aporta valor a la estructura del proyecto o Business si aporta valor al proyecto en sí) y tipo de Ítem.

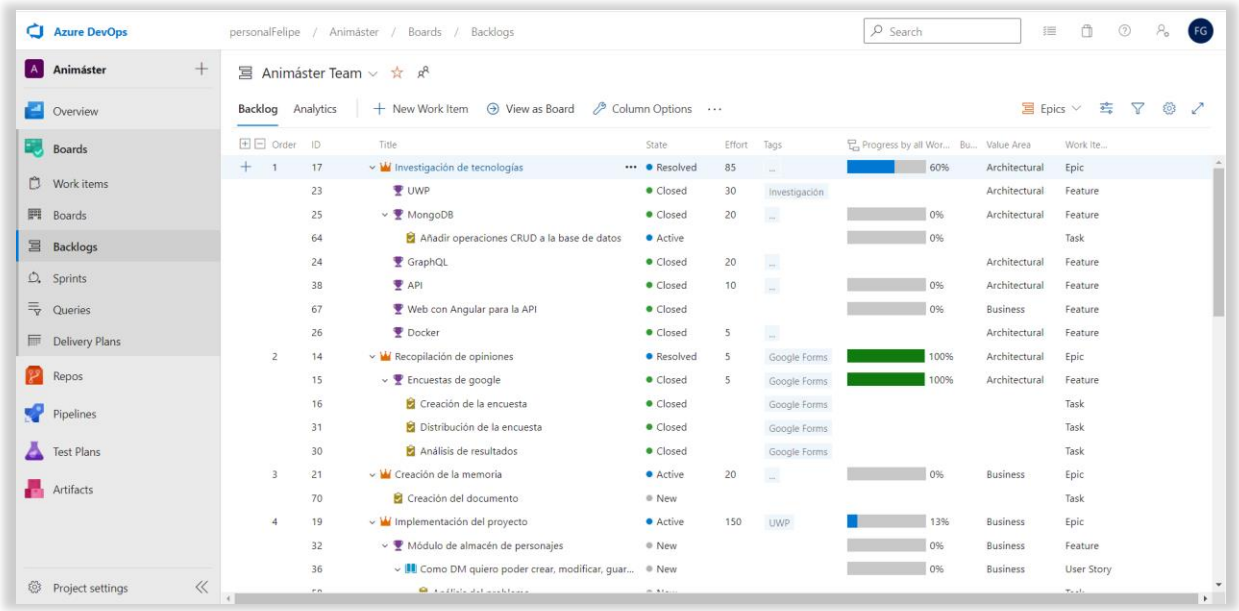


Fig. 11. Backlog

Todos los ítems tienen una ventana de detalle. Un ejemplo de uno de los ítems que esta completado y cerrado en el momento de escribir esto es “Task: Creación de encuestas” (Figura 12)

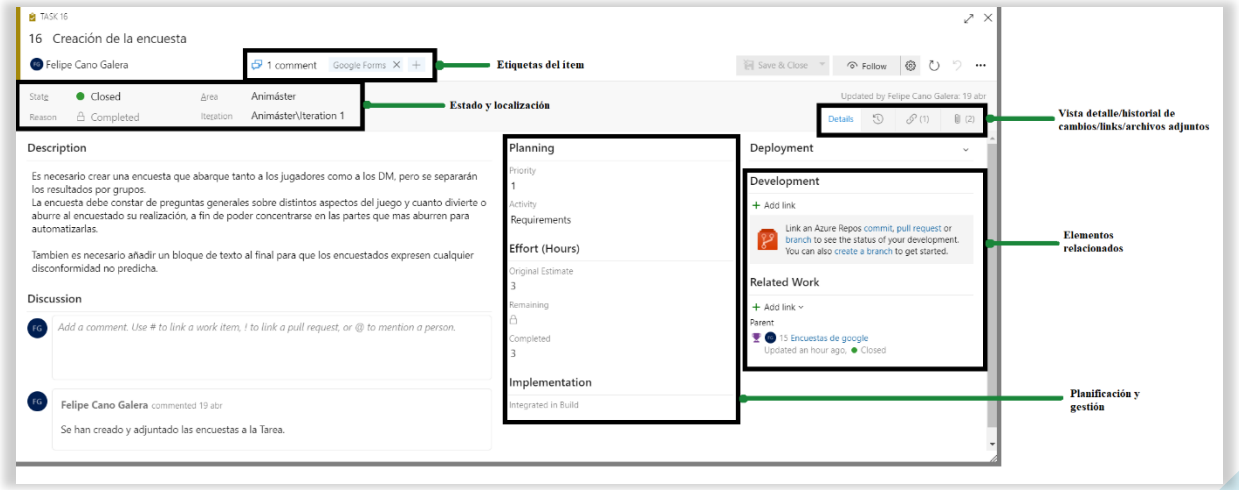


Fig. 12. Vista detalle de un ítem

Caben mencionar varias partes clave de la vista, pues en ella están muchas de las herramientas que hacen a Azure un administrador tan potente.

Desde las etiquetas del ítem podemos adjuntarle estas para una localización más fácil en las búsquedas.

En “Estado y localización” se encuentran las opciones para asignar el área en caso de que haya más de un proyecto, la iteración (equivalente a sprint) y las opciones para cambiar el estado del ítem (En este caso ‘New’ para ítems sin empezar, ‘Active’ para los que están siendo trabajados, ‘Closed’ para los que están terminados y ‘Resolved’ para los que cuentan, además, con la aprobación de mis tutores).

En “Elementos relacionados” se encuentran los commits que mencionan este ítem (En caso de que los haya), así como los elementos padres o hijos. En este caso, al no haber código no hay commits relacionados, aunque si hay un elemento padre, la feature “Encuestas de Google”

Por último, en “Planificación y gestión” se pueden encontrar las medidas que se han estimado y el seguimiento de las horas invertidas. En caso de ser un *Epic* o *Feature*, se pueden encontrar pequeñas variaciones (Figura 13)

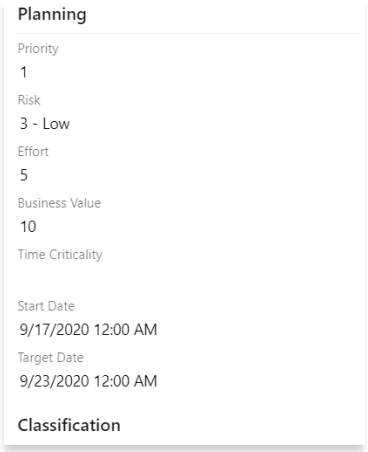


Fig. 13. Planificación y gestión del Epic "Recopilación de opiniones"

2.2.2 Repos

Azure Repos funciona exactamente como cabría esperar de un sistema de repositorios, por lo que seré breve.

Permite ver los archivos subidos, los commits (y sus ítems enlazados), los pushes, branches, tags (Para marcar determinado momento de interés en el historial de commits) y los pull requests. (Figura 14)

Debido a que en el proyecto estoy solo la opción de los tags y los pull requests no ha sido usada.

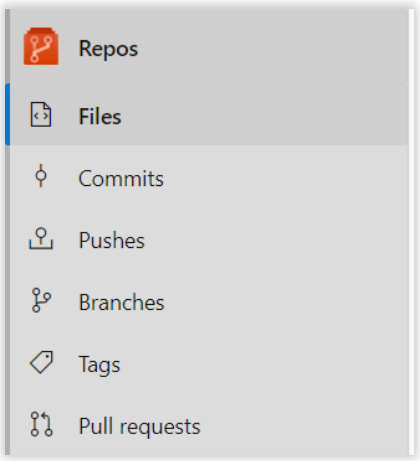


Fig. 14. Opciones disponibles para usar con Repos

2.2.3 Artifacts

Artifacts se compone de dos piezas fundamentales, los “Feeds” y el “Origin”.

A la hora de administrar librerías, Visual Studio extrae los paquetes NuGet de la api dedicada a ello. Esto se considera el origen de los paquetes.

En un proyecto grande, conviene que todos los integrantes tengan los mismos paquetes y las mismas versiones de los paquetes para evitar incompatibilidades, por lo que mediante Artifacts se puede establecer como origen de los paquetes el feed (Figura 15).

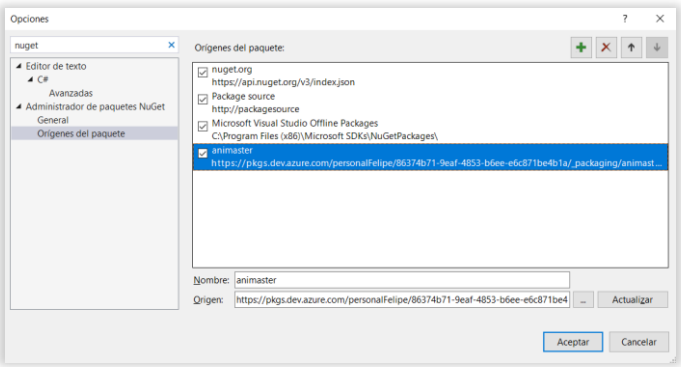


Fig. 15. Origen establecido en Azure en lugar de Nuget.org

El feed es el listado de paquetes con sus respectivas versiones del cual extrae la información los proyectos cuyo origen están en él. Cuando un integrante añade un paquete nuevo este cambio es reflejado en Artifacts para que todos los integrantes del proyecto usen ese paquete. Un ejemplo del feed del proyecto se puede ver en la Figura 16. Como es obvio, Artifacts no aporta ninguna utilidad si estás solo en un proyecto más allá de llevar un listado de los paquetes que utilizas, a no ser que, como en mi caso, programes desde dos ordenadores distintos en determinadas circunstancias.

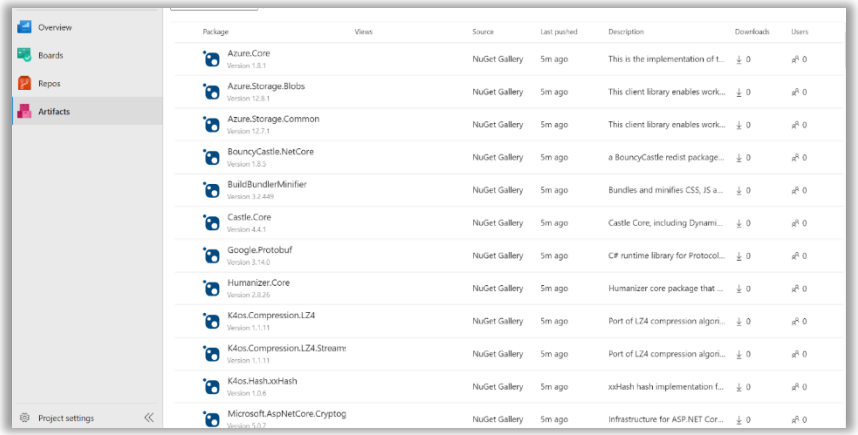


Fig. 16. Ejemplo de un feed en funcionamiento

3. API Rest

3.1 Que es API Rest

Una API de transferencia de estado representacional (REST) es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar software de las aplicaciones creando una estructura cliente-servidor, en la que se sitúa en el medio. Permite que una aplicación de un usuario (cliente) realice una petición concreta y definida por el programador para que la API permita al servidor comprenderla y responder adecuadamente a ella.

El hecho de que la API sea la encargada de la conexión entre el cliente y el servidor hace que aumente enormemente la seguridad y el control que se tiene sobre todo el sistema. La API solo aceptará las peticiones que se hayan definido y, tras acceder al servidor, solo extraerá los datos que sean correspondientes a esas peticiones.

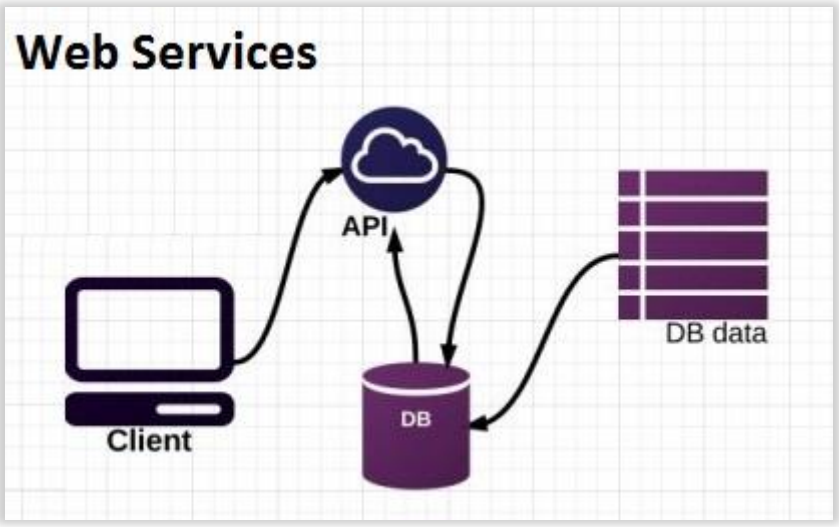


Fig. 17. Esquema de un sistema con API [8]

3.2 Utilización de API Rest en el proyecto

A la hora de trabajar con una API es necesario definir su comportamiento. Para ello se utilizan las rutas, también llamadas endpoints. Cada endpoint puede hacer distintas funciones dependiendo del protocolo http que se utilice para acceder. Así, un endpoint para obtener un personaje también puede ser el mismo que para borrarlo o para modificarlo, simplemente utilizando GET, PUT o DELETE.

En este proyecto se pretende tener endpoints suficientes para tener CRUD.

Los endpoints utilizados son:

Tabla 1. Endpoints

Ruta	Protocolo	Descripción
/personajes	GET	Devuelve todos los personajes del usuario que lo solicita. Es posible acceder mediante una consulta de GraphQL para obtener la versión simplificada.
/personajes/crear	POST	Crea un personaje
/personajes/:id	GET	Devuelve el personaje cuya id coincida con la del endpoint
/personajes/:id	PUT	Modifica el personaje cuya id coincida con la del endpoint
/personajes/:id	DELETE	Borra el personaje cuya id coincida con la del endpoint
/personajes/comunidad	GET	Devuelve todos los personajes de la comunidad

4. MongoDB

4.1 Qué es MongoDB

MongoDB es una base de datos no relacional que cada vez está ganando más popularidad en la comunidad de desarrolladores. Tanto es así, que lleva dos años en el top 1 de “Tecnología que más quieres aprender a usar” según las encuestas de StackOverflow de 2019 y 2020 [9] (Figura 18).

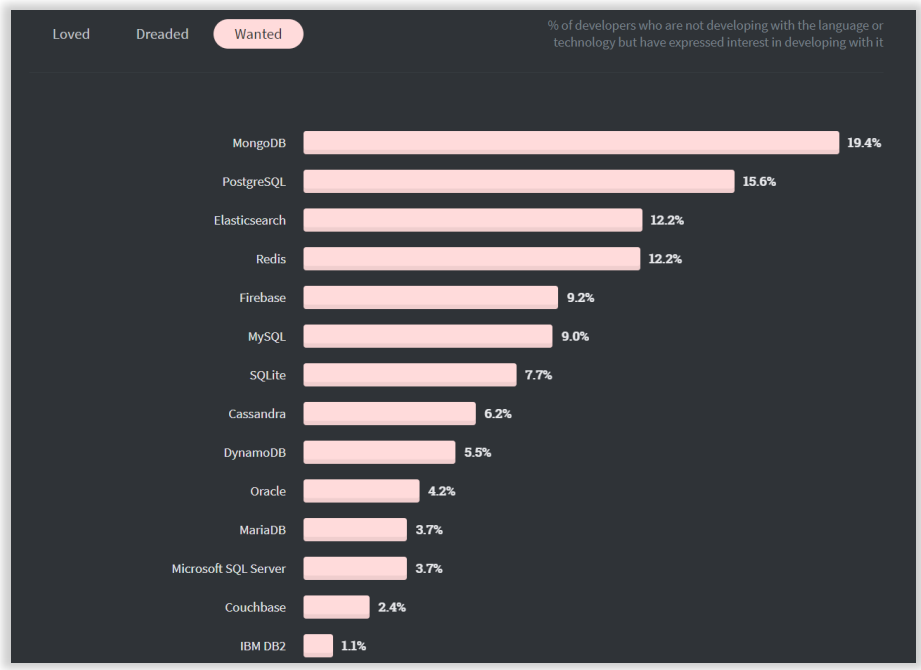


Fig. 18. Top de tecnologías/lenguajes que más quieren aprender los desarrolladores

Que sea una base de datos no relacional significa que no trabaja con estructuras de datos definidas y la información no se organiza en registros ni campos. En su lugar un conjunto de datos se denomina documento, que se pueden agrupar en colecciones que son el equivalente a las tablas en bases de datos convencionales y dentro de los documentos se encuentran los campos, equivalente a las columnas de las tablas. Además, el ser una BD NoSQL, también implica que no usa el lenguaje SQL como herramienta de consulta, sino que lo hace como apoyo.

Para almacenar la información, MongoDB hace uso de representaciones binarias de JSON, llamadas BSON que permiten la búsqueda rápida de datos. A pesar de ello, al trabajar con MongoDB los datos se manejan en formato JSON.

Por último, para la id, MongoDB, utiliza un valor parecido a las UUID en hexadecimal por defecto y la llama “_id”, aunque es fácilmente configurable.

4.2 Base de datos relacional vs no relacional

A la hora de abordar el proyecto, pueden existir dudas sobre qué tipo de base de datos utilizar, por ello es necesario valorar la alternativa. Para la comparativa se puede comprobar la siguiente tabla.

Bases de datos relacionales	Bases de datos no relacionales
Basada en tablas	Basada en documentos
Estructura fija y definida	Estructura dinámica y flexible
Menor rendimiento	Mayor rendimiento
Diseñada para evitar redundancia	Permite redundancia
Pude sufrir con grandes volúmenes de datos	Optimizada para mantener grandes volúmenes de datos
Escalabilidad estándar	Escalabilidad muy alta
Transacciones ACID	Transacciones ACID más complejas y solo a partir de la versión 4.0
Alta fiabilidad	Baja fiabilidad
Compatibles con todas las consultas SQL	No compatible con todas las consultas SQL
Sistema estandarizado, muchos años de uso	Sistema no estandarizado, relativamente nueva

4.3 Utilización de MongoDB en el proyecto

Hay muchas cosas que valorar, pero lo principal y el motivo de la elección de una base de datos no relacional es, sobre todo, la flexibilidad que permite a la hora de almacenar datos, dado que en determinado punto un director de juego podría querer guardar fichas con datos sin rellenar dado que pueden no ser relevantes. Además, también se ha valorado el rendimiento y la escalabilidad. A pesar de que es menos fiable y no la utilizaría nunca para el manejo de datos financieros, en este caso son datos públicos que todo el mundo puede consultar a su antojo.

Dentro del proyecto, es necesario crear un modelo (Figura 19) para poder crear los personajes que irán a la base de datos. Este modelo debe de tener unos Flags para que mongo pueda reconocer los atributos y cuál de ellos actuara como Id. Dentro de la base de datos, el flag BSONElement es el que determina el nombre por el cual se accede a determinado campo, mientras que, en la aplicación, es el atributo el que se utiliza. Es importante tener esto en cuenta, ya que además es case sensitive y puede llevar a confusión.

```
11  namespace Animaster.Web.Models
12  {
13
14
                16 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
15      public class Personaje
16      {
17          [BsonId]
18          [BsonRepresentation(BsonType.ObjectId)]
                3 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
19          public String Id { get; set; }
20
21          [BsonElement("nombre")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
22          public string Nombre { get; set; }
23
24          [BsonElement("clase")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
25          public string Clase { get; set; }
26
27          [BsonElement("nivel")]
                2 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
28          public long Nivel { get; set; }
29
30          [BsonElement("tamaño")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
31          public long Tamanio { get; set; }
32
33          [BsonElement("inventario")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
34          public List<string> Inventario { get; set; }
35
36          [BsonElement("dinero")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
37          public List<long> Dinero { get; set; }
38
39          [BsonElement("principales")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
40          public Principales Principales { get; set; }
41
42          [BsonElement("estatus")]
                0 referencias | Felipe Antonio Cano Galera, Hace 34 días | 1 autor, 1 cambio
43          public Dictionary<string, long> Estatus { get; set; }
```

Fig. 19. Fragmento del modelo de personaje

5. GraphQL

5.1 Qué es GraphQL

GraphQL es un lenguaje Query que surge como alternativa a REST y permite hacer consultas y recibir una respuesta predecible que se utiliza para la comunicación cliente-servidor de una API.

La principal ventaja de GraphQL es que es muy eficiente a la hora de intercambiar información. Envía al cliente solo la información necesaria y hace una sola request por vista. Esto lo realiza introduciendo en el body de la petición http la query.

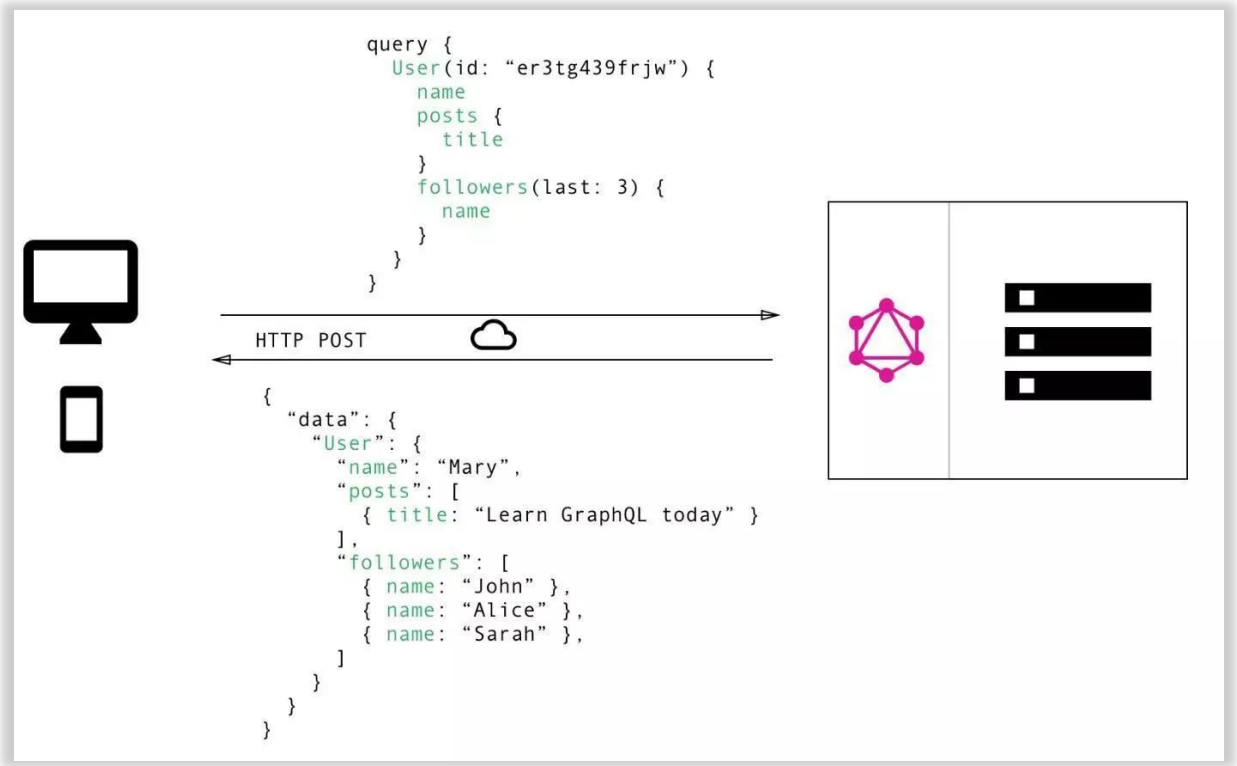


Fig. 20. Ejemplo de intercambio de datos GraphQL

5.2 Utilización de GraphQL en el proyecto

A pesar de que es una forma distinta de abordar un problema que la que ofrece API Rest, se ha decidido usar ambas tecnologías en el proyecto, usando solo graphQL para donde más impacto tiene.

Dado que el proyecto pretende incluir la opción de extraer fichas de la comunidad hay una gran cantidad de datos que pueden llegar a fluir en cada consulta. Es por esto por lo que para solventarlo se ha utilizado GraphQL para la petición al endpoint “/personajes/comunidad”.

Utilizando Postman a efectos de demostración para poder ver los resultados más claros, se muestra un ejemplo en la Figura 21. En él, se ve una consulta al endpoint en la que se introduce por el body una query en la que solo se solicita el nombre de todos los personajes de nivel 6 de la comunidad.

Debajo se ven los datos recuperados.

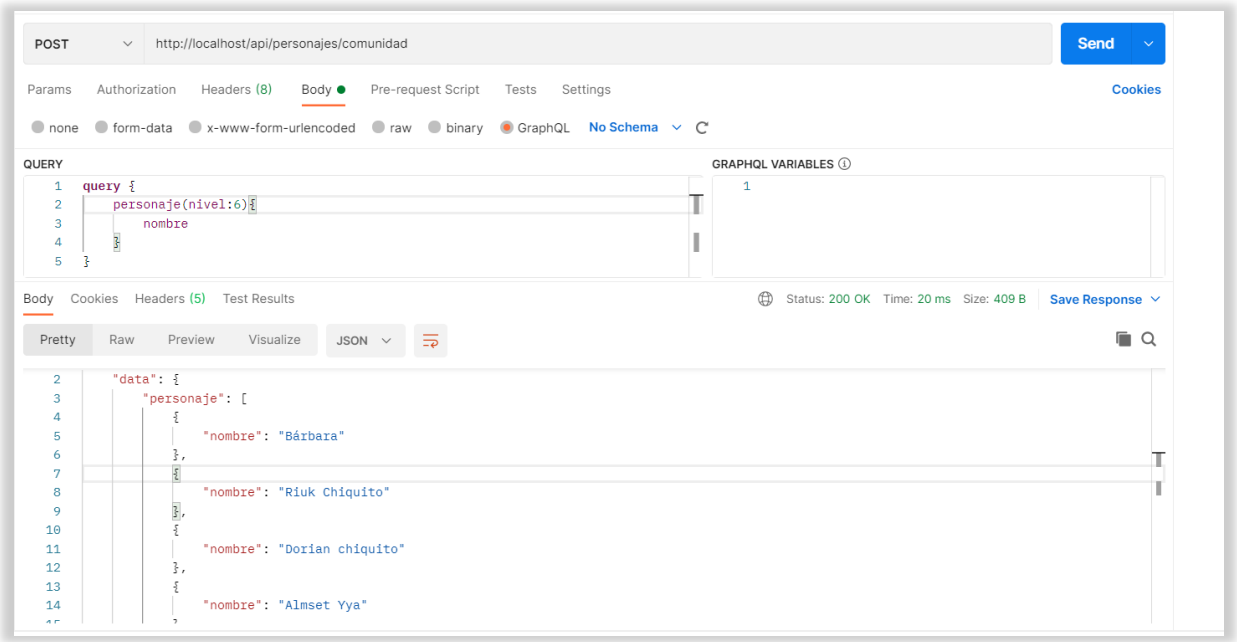


Fig. 21. Consulta de ejemplo GraphQL

Aquí se comprueba la potencia de GraphQL. En lugar de recuperar todo el documento de cada personaje, cada uno con sus más de 50 campos, solo se recupera el campo que se ha solicitado, disminuyendo el tamaño de los datos transferidos a solo 400 Bytes en comparación con los 1.38 KB que supondría la misma consulta completa (Figura 22). Supone una reducción del 72%.

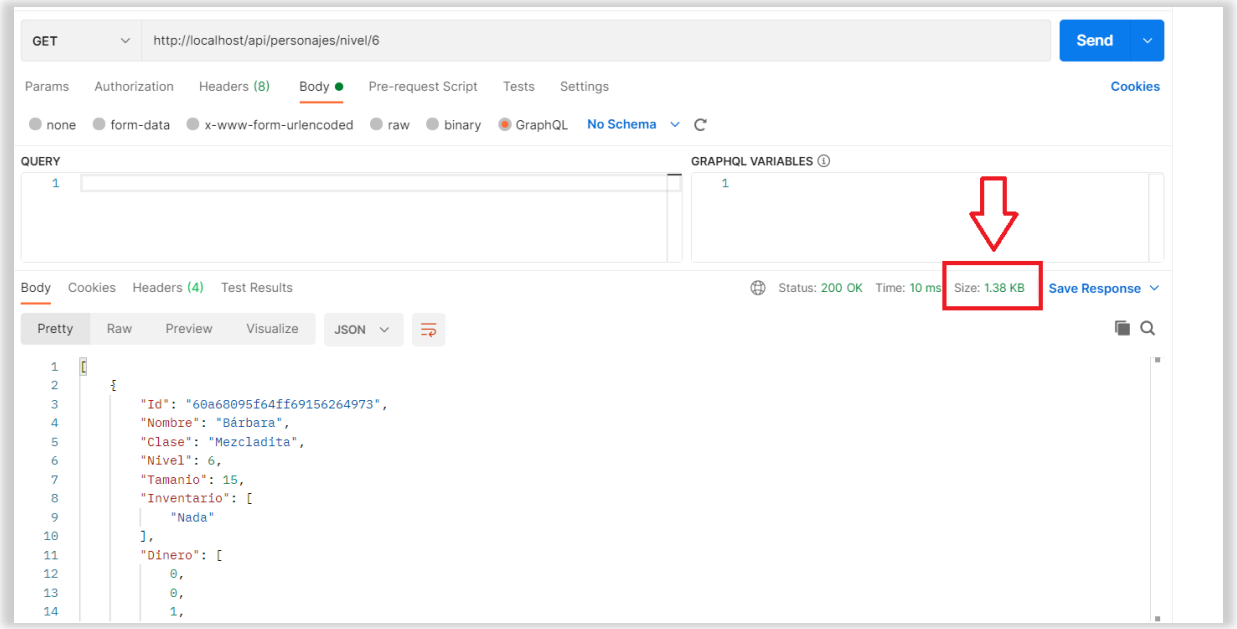


Fig. 22. Comparativa de tamaño de datos con una consulta completa. Endpoint creado solo para este ejemplo.

A continuación, un ejemplo de una petición real del proyecto. En la aplicación, para satisfacer el RF-03 es necesaria la creación de un almacén de personajes. Dicho almacén solo mostrará una versión muy resumida de las fichas, a modo de expositor. Dado que no es necesario el resto de la ficha, se realiza una petición mediante GraphQL para que extraiga solo los datos que son necesarios para exponer, en este caso nombre, clase y principales de los personajes de nivel 6 (Figuras 23 y 24). Esto muestra el gran poder que tiene GraphQL, dado que mientras de normal se extraería todo el documento para unos pocos campos, con esta consulta se puede satisfacer la necesidad de obtener el requisito de información 5 (Principales) y parte del 1 (Personaje) sin necesidad de traer con ellos todos los demás requisitos que no van a ser consumidos por la aplicación en ese momento.

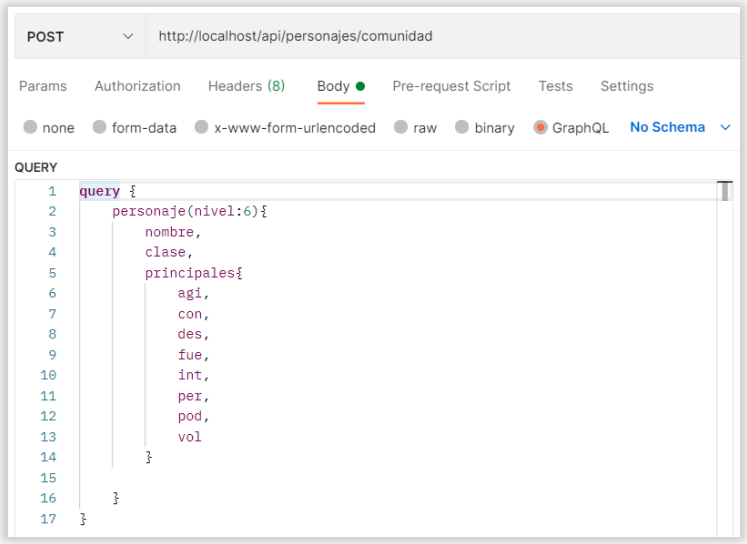


Fig. 23. Query de la consulta para el almacén de personajes

Fig. 24. Fragmento de la respuesta a la consulta de la Figura 35

6. Docker

6.1 Qué es Docker

Docker es una plataforma que permite empaquetar software junto a todo lo necesario para que este funcione, como bibliotecas, herramientas de sistema, código y tiempo de ejecución.

Esto puede resultar muy útil para asegurarnos de que el código se va a ejecutar en cualquier entorno gracias a estar aislado en un contenedor, el cual actúa como si fuera una máquina virtual.

En el entorno de Docker se distinguen 3 conceptos fundamentales: Contenedor, imagen y dockerfile.

Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de una forma rápida y consistente de un entorno de programación a otro. Puede parecer como una máquina virtual, pero en realidad no lo es, puesto que la máquina virtual emula también el hardware, mientras que un contenedor reutiliza el hardware y el Kernel de la máquina que lo está hospedando (Figura 25). Se considera que un contenedor es una instancia en ejecución de una imagen Docker.

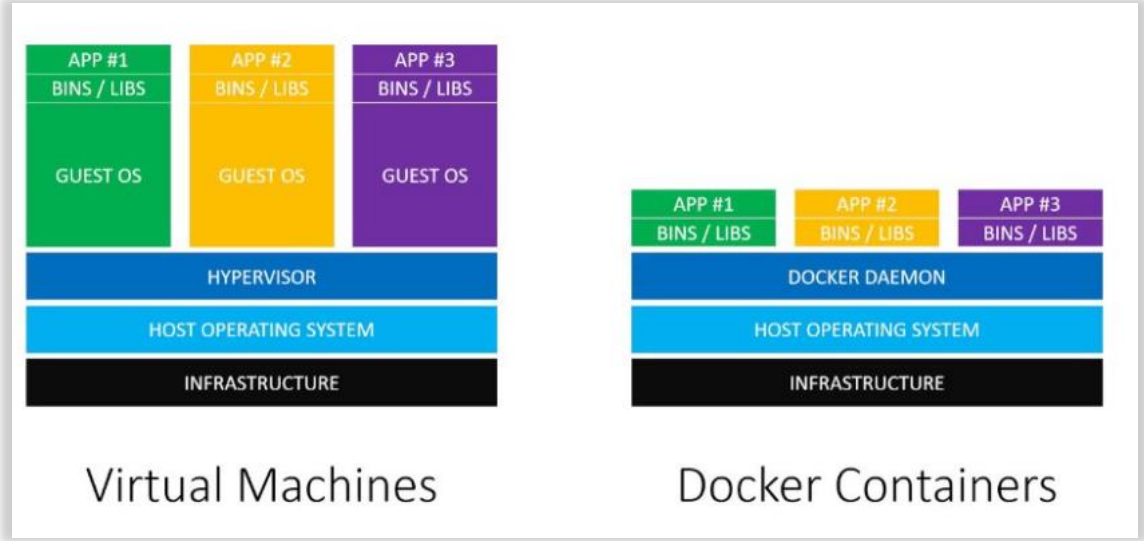


Fig. 25. Esquema de estructura de Docker [10]

Una imagen de Docker es como una plantilla, una captura del estado de un contenedor. Esta captura se puede utilizar más tarde para crear múltiples contenedores para comprobar como funcionarían varias instancias de la aplicación a probar. Normalmente, se parte de una imagen base pública que tiene incorporados elementos básicos como java, Ubuntu o apache entre otros, y se le van añadiendo los elementos propios, como podría ser el código, una base de datos y el contenido. Para lograr esto se utiliza el Dockerfile.

Un Dockerfile es un archivo de configuración utilizado para crear imágenes en el que indicamos que queremos que contenga la imagen y se agregan los distintos comandos para utilizar las herramientas.

El flujo de creación de un contenedor parte del Dockerfile para crear la imagen y esta se ejecuta en tantos contenedores como queramos (Figura 26)

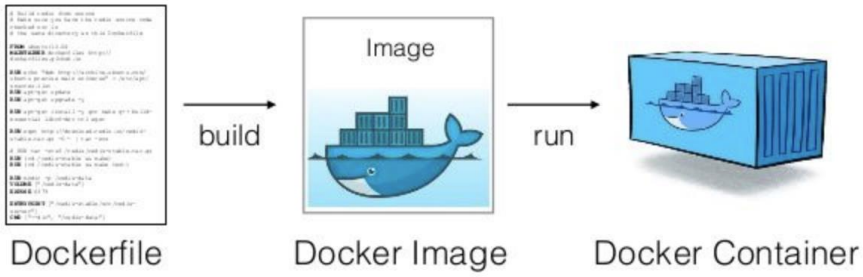


Fig. 26. Flujo de creación de un contenedor

6.2 Utilización de Docker en el proyecto

En nuestro caso el proyecto está claramente diferenciado entre una API y la aplicación de escritorio, por lo que se ha aprovechado esta separación para utilizar Docker.

La idea principal es crear una imagen de la API, junto a la base de datos y algunos datos ya introducidos en ella y ser capaz de ejecutarla en cualquier momento con la comodidad de Docker Desktop. Así, a la hora de hacer pruebas de la aplicación, siempre se harían sobre un contenedor, de tal forma que en caso de que deje de funcionar por algún motivo, se pueda volver al estado de la imagen simplemente reiniciando el contenedor. Además, al hacer pruebas con la aplicación desde Visual Studio, se ahorran muchos recursos y tiempo mediante la inicialización de la API desde el exterior mediante Docker.

Para hacer esto, es necesario hacer uso del Docker Compose, una herramienta que permite agrupar y ejecutar aplicaciones multi contenedor. En este caso, habrá un contenedor para la API, otro para la base de datos y otro para MongoExpress para tener la comodidad de poder acceder a la base de datos de forma directa por si fuera necesario.

El esquema para esto sería como muestra la Figura 27.

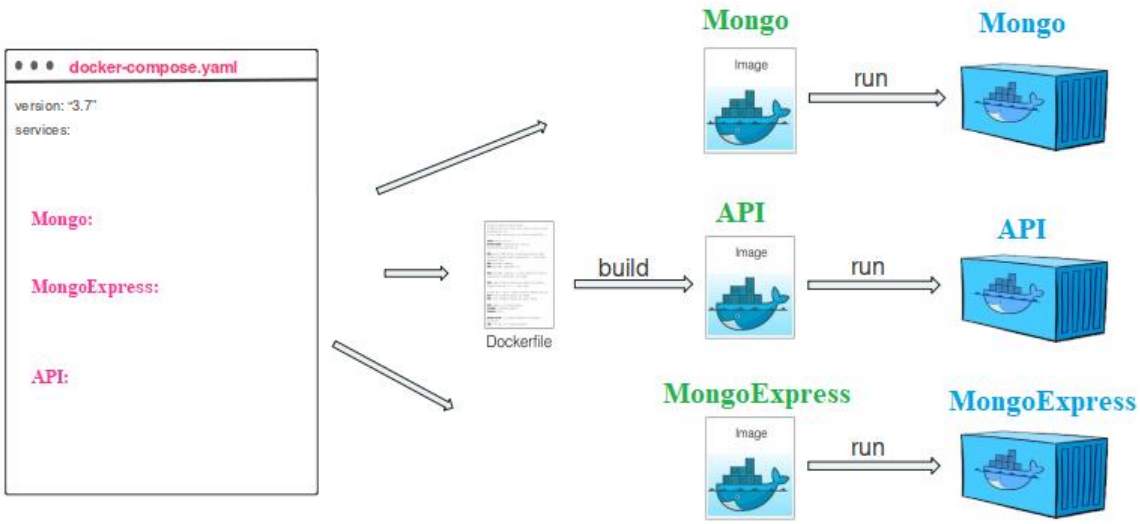


Fig. 27. Estructura de la composición del proyecto

Para la creación de esta estructura se requiere de la creación de un Docker compose donde se especifique los servicios, en este caso API, mongo y mongo-express. En este caso, tanto mongo como mongo-express utilizan imágenes públicas. Para la api, hay que construirla mediante el dockerfile (Figura 28 y 29).

docker-compose.yml

```yaml
version: '3'

services:
  api:
    build: "./Animaster.Web.Api"
    restart: always
    ports:
      - "5001:5001"
      - "80:80"
      - "443:443"

  mongo:
    image: mongo
    restart: always
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example

  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: root
      ME_CONFIG_MONGODB_ADMINPASSWORD: example
    depends_on:
      - mongo
```

Fig. 28. Docker-compose del proyecto

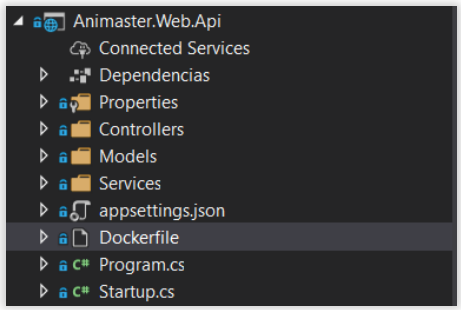


Fig. 29. Localización del Dockerfile en el proyecto

Este dockerfile se ha creado siguiendo una técnica de “Multi stage build” usando multiples “FROM”, en la que el documento se crea por fases a las que se les va dando nombre. Una vez terminada una fase, se continúa con la siguiente utilizando la anterior como base. De esta forma se puede ir dejando atrás cualquier cosa que no quieras en la build final (Figura 30).

```
3  FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base
4  WORKDIR /app
5  EXPOSE 80
6  EXPOSE 443
7
8  FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build
9  WORKDIR /src
10 COPY ["./Animaster.Web.Api.csproj", "./Animaster.Web.Api/"]
11 RUN dotnet restore "Animaster.Web.Api/Animaster.Web.Api.csproj"
12 COPY [".", "./Animaster.Web.Api"]
13 WORKDIR "/src/Animaster.Web.Api"
14 RUN dotnet build "Animaster.Web.Api.csproj" -c Release -o /app/build
15
16 FROM build AS publish
17 RUN dotnet publish "Animaster.Web.Api.csproj" -c Release -o /app/publish
18
19 FROM base AS final
20 WORKDIR /app
21 COPY --from=publish /app/publish .
22 EXPOSE 5001
23 ENTRYPOINT ["dotnet", "Animaster.Web.Api.dll"]
```

Fig. 30. Dockerfile con la estructura de multi stage building

Una vez creado el dockerfile, se ejecuta para tener la imagen. Docker Desktop incorpora una librería de imágenes para tener un visionado claro de las que disponemos en el disco. En este caso se puede comprobar en la Figura 31 como tengo la de la API, mongo y mongo-express listas para su uso (Ignorar las dos últimas, son de uso personal que no tiene que ver con el proyecto).

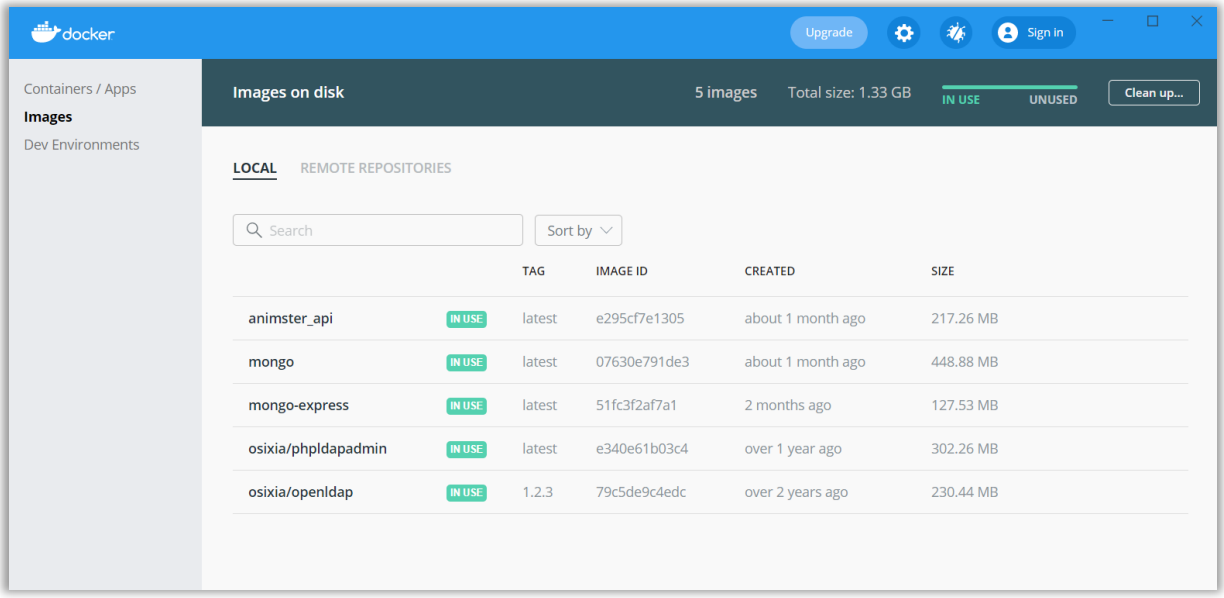


Fig. 31. Librería de imagenes de docker

Con las imágenes disponibles ahora es posible crear la estructura definida por el compose, como se muestra en la Figura 32. Con la facilidad de pulsar un botón, se pueden iniciar todos los servicios definidos (Figuras 33 y 34)

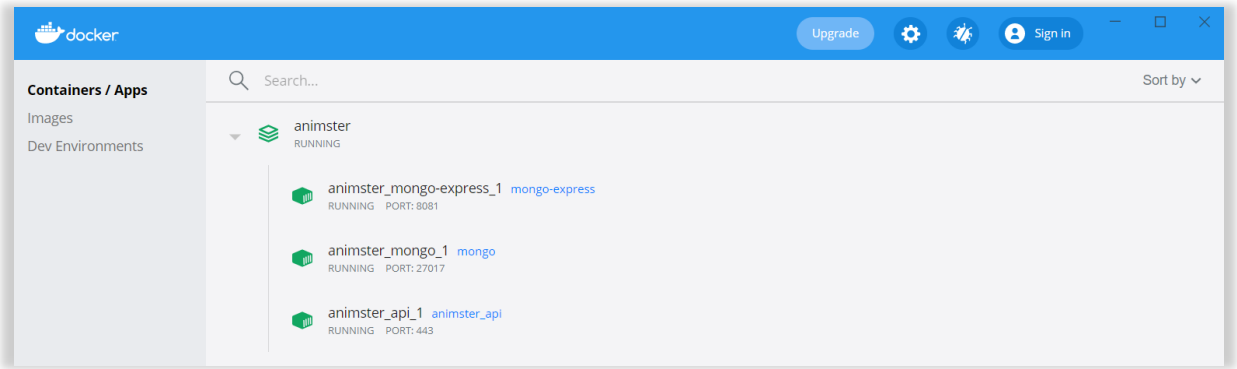


Fig. 32. El stack de docker en funcionamiento

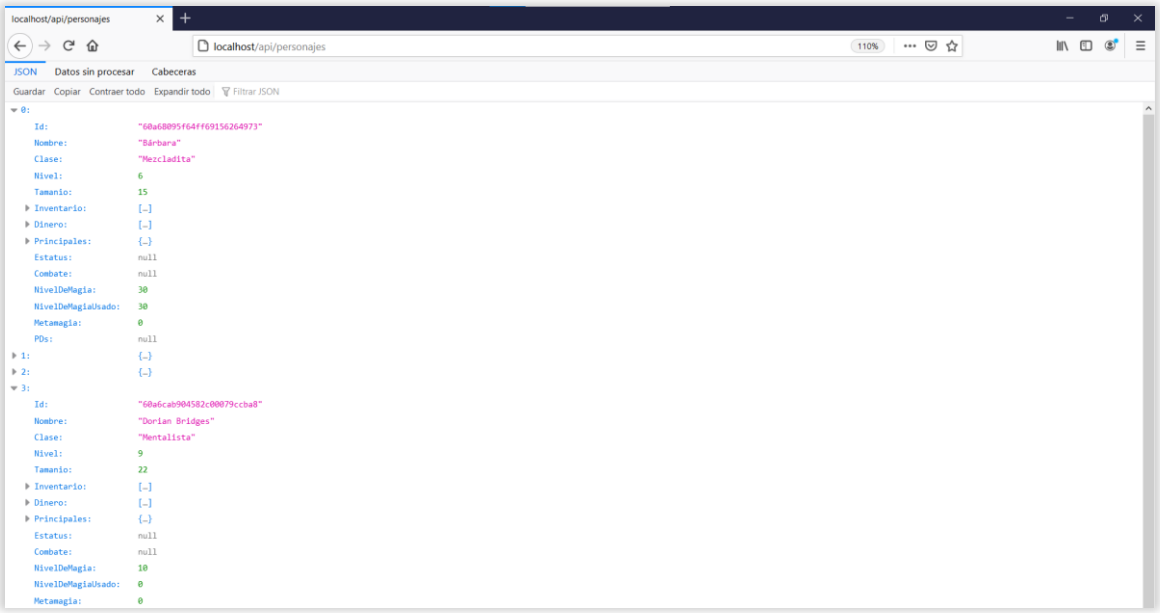


Fig. 33. API en funcionamiento desde docker

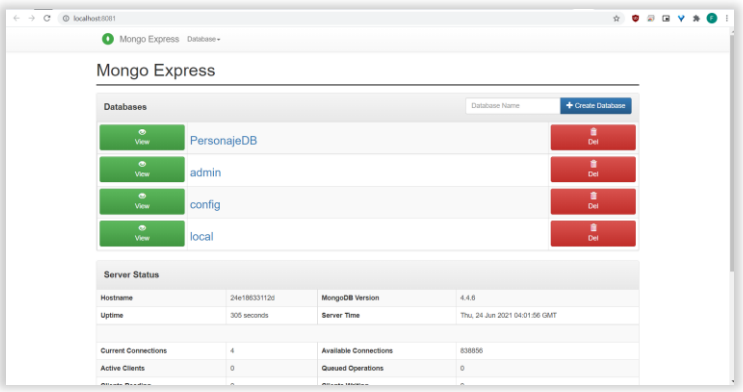


Fig. 34. Mongo Express funcionando desde docker

7. Animáster

7.1 Diagrama de casos de uso

Tras analizar los requisitos se ha modelado el siguiente diagrama de casos de uso para las funcionalidades de la aplicación

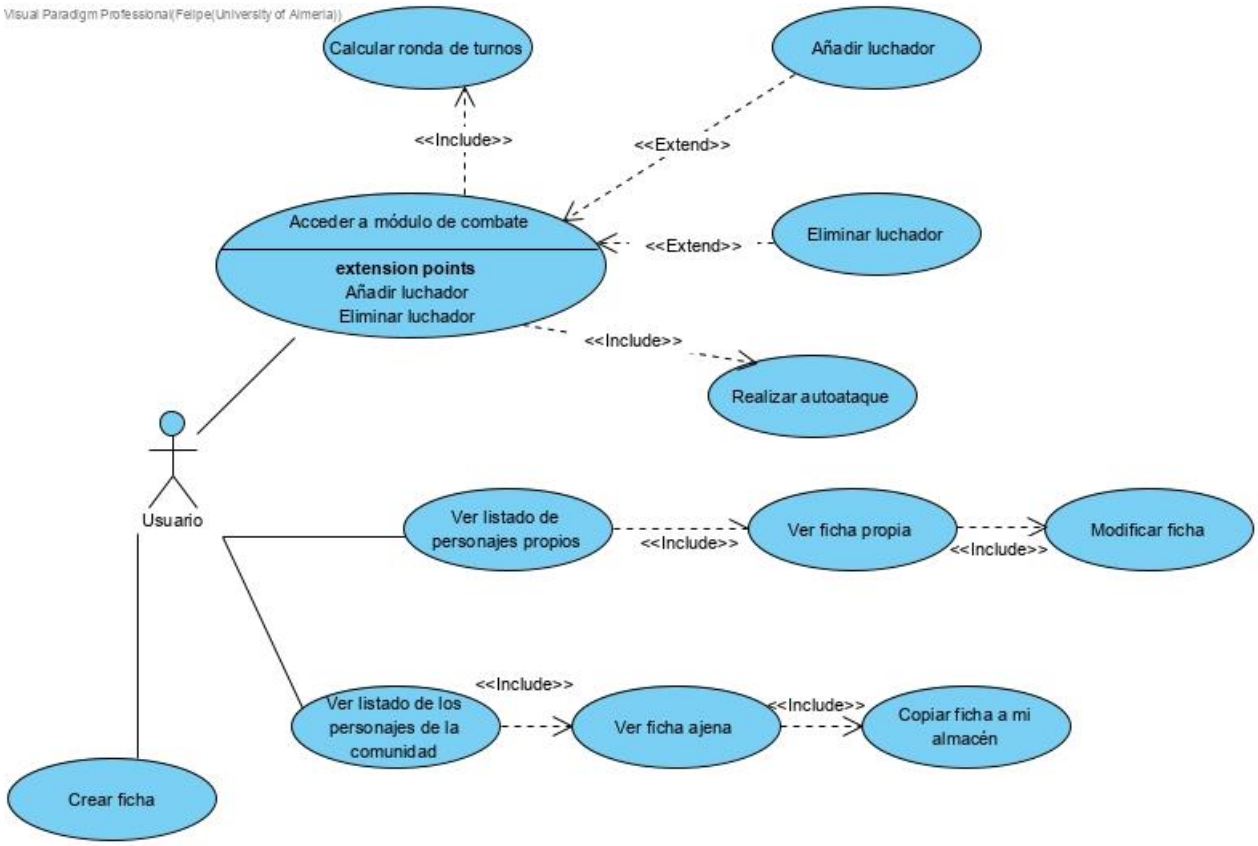


Fig. 35. Diagrama de casos de uso

7.2 Guía de estilo

Se ha optado por una paleta de grises para la interfaz para lograr un tema oscuro, algo que para las largas horas de juego puede suponer una diferencia en el cansancio de la vista. Como color de acento, se ha elegido un tema azul marino, dado que combina muy bien. Para el color del texto, el blanco puro contrasta con la paleta de colores de la interfaz, haciendo una lectura cómoda.

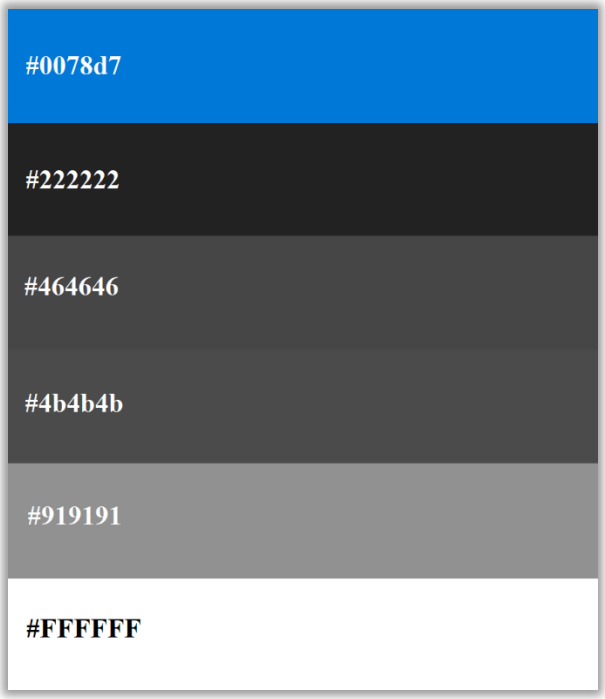


Fig. 36. Guía de estilos

Para la iconografía se ha utilizado Segoe MDL2 Assets, una librería de iconos realmente amplia que cubre numerosos ámbitos. Algunos ejemplos se pueden ver a continuación.

Fig. 37. Ejemplo de iconos Segoe MDL2

7.3 Maquetado

Para el maquetado de la aplicación se ha utilizado AdobeXD A continuación, se listan todas las interfaces diseñadas.

La vista de inicio incorpora un tutorial de uso de la aplicación para que los usuarios que están empezando en ánima puedan utilizar la aplicación correctamente desde el principio. Los usuarios que ya tienen cierta experiencia en el dominio del problema encontrarán la aplicación suficientemente intuitiva para no necesitarlo, aunque por supuesto, tendrían acceso igualmente.

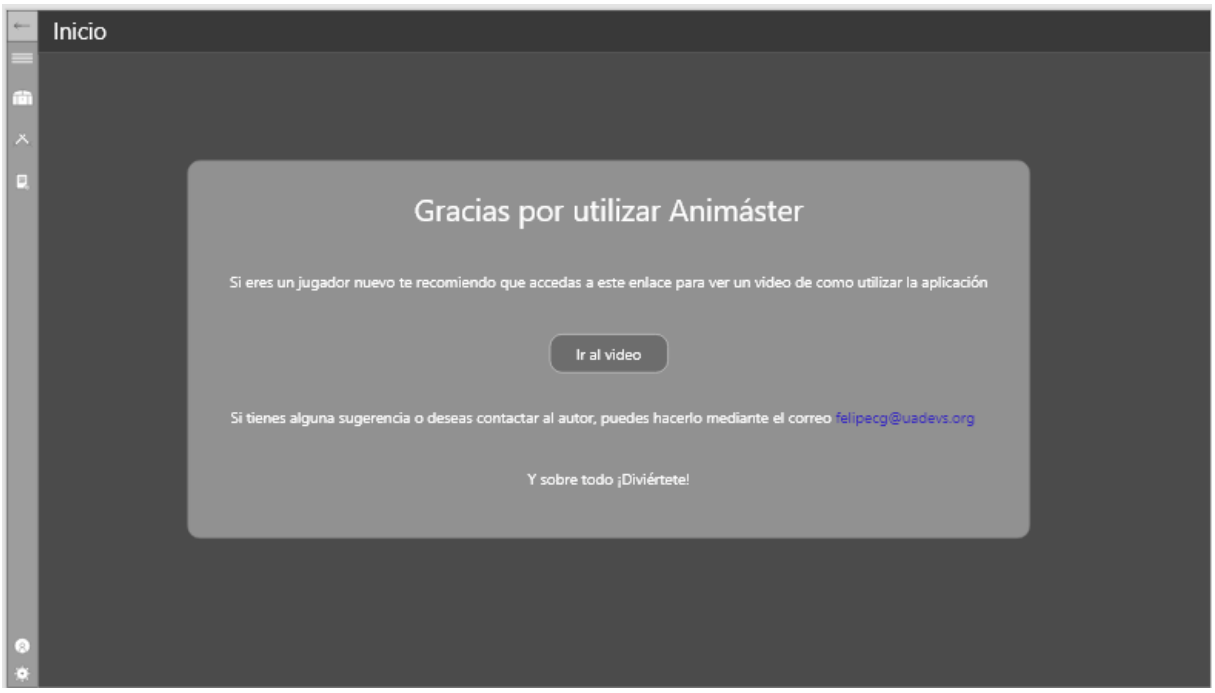


Fig. 38. Página de inicio

La navegabilidad principal de la aplicación se basa en un menú de hamburguesa que lleva a las 3 vistas principales. Además, incluirá una flecha de retroceso al último estado visitado.

Fig. 39. Página de inicio con menú de hamburguesa desplegado

El almacén de personajes, como se ha mencionado antes, serán fichas simplificadas, tendrá la posibilidad de discriminar entre fichas propias o de la comunidad, dispondrá de filtros y, en caso de seleccionar una ficha llevará a la vista detalle.

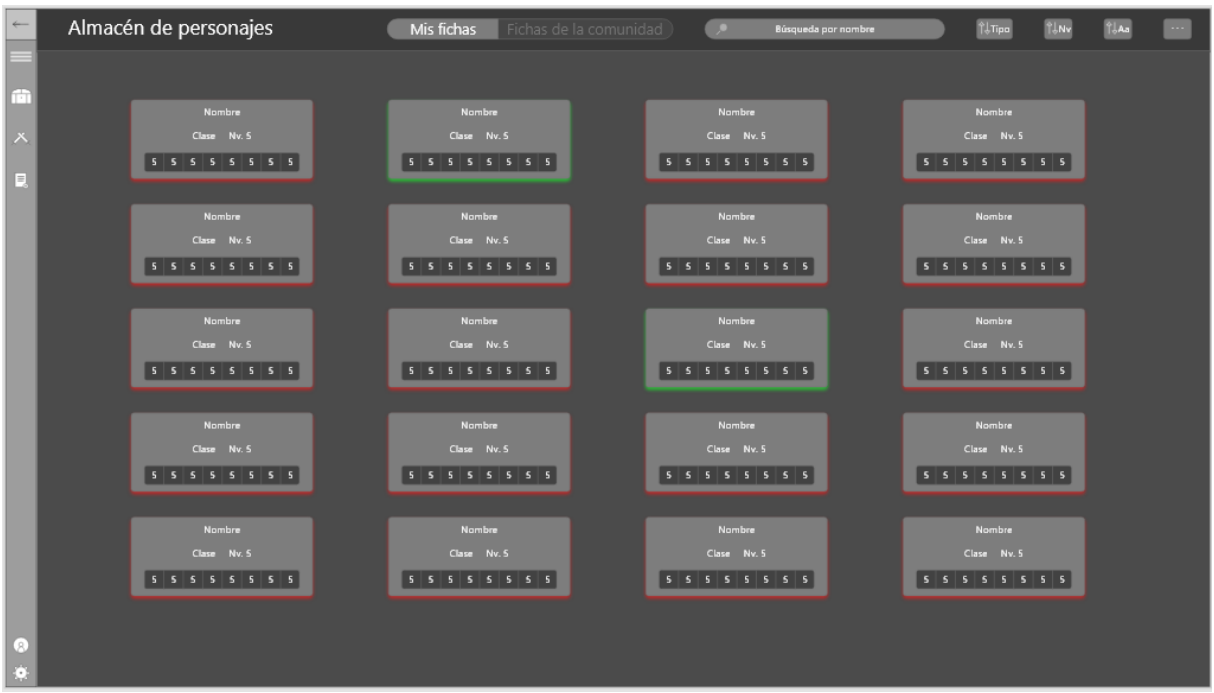


Fig. 40. Almacén de personajes

La vista detalle de una ficha puede ser de dos formas distintas. Si es una ficha propia, se podrá borrar, utilizar y modificar con normalidad. En caso de ser de la comunidad, tan solo se puede hacer una copia para luego modificarla si se desea. La ficha es extensa y compleja, por lo que se ha dividido lo máximo posible en distintos apartados para cada grupo de características que tienen relación entre ellas. La ventana resumen, concretamente, pone a disposición del usuario todas las características que normalmente más falta hace y más veces hay que consultar. Notar que, en caso de ser una ficha ajena, el botón “Borrar ficha” cambia por “Hacer una copia”. Adicionalmente, esta ventana solo se puede comprobar una vez que se ha creado ya la ficha, cosa que va en consonancia con el hecho de que al crearla se sube a la base de datos y se le asigna un ID. Este ID es lo que se le da al usuario para compartir la ficha. Cualquier persona con ese ID podrá buscar una ficha en concreto, cosa que es realmente útil para que un director de juego utilice las fichas de sus jugadores sin tener que copiarlas manualmente.

En todas las interfaces de la ficha se han mantenido los principios de agrupación y balanceo de Gestalt. En las que ha sido posible, se ha aplicado simetría para hacerlas visualmente más atractivas.

Fig. 41. Página resumen de la ficha

A la hora de crear o modificar la ficha se utilizan las pestañas principal y PDs. La pestaña principal dispone de los primeros campos que debería rellenar un jugador a la hora de crear un personaje. En el caso de que la ficha se esté creando, será necesaria una confirmación de creación.

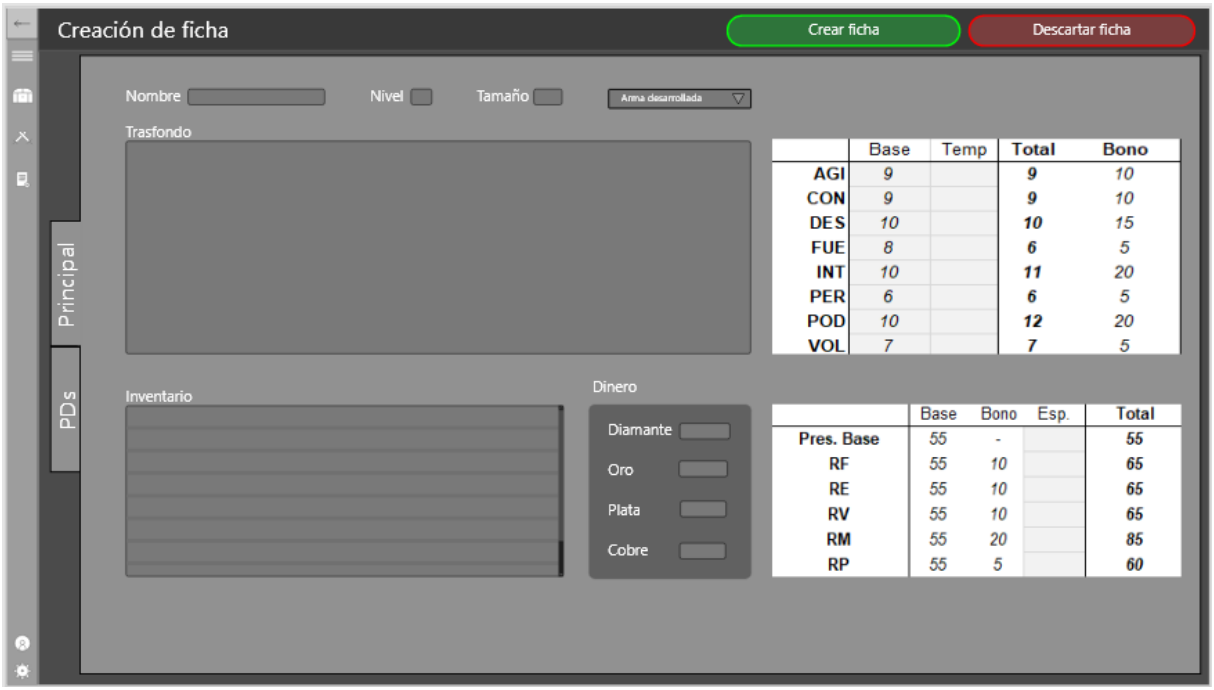


Fig. 42. Página principal de creación de personaje

La pestaña de PDs, por su parte, permite al jugador personalizar su personaje de la forma tan profunda que permite el manual. La mayoría de los campos están automatizados, teniendo que rellenar 2 o 3 en cada característica, dependiendo de cuál sea, en lugar de los 8-10 que normalmente habría que calcular a mano. Las características están agrupadas según el grupo al que pertenecen. Siendo combate (Figura 43), Místicas (Figura 44), Psíquicas (Figura 45) y secundarias (Figura 46).

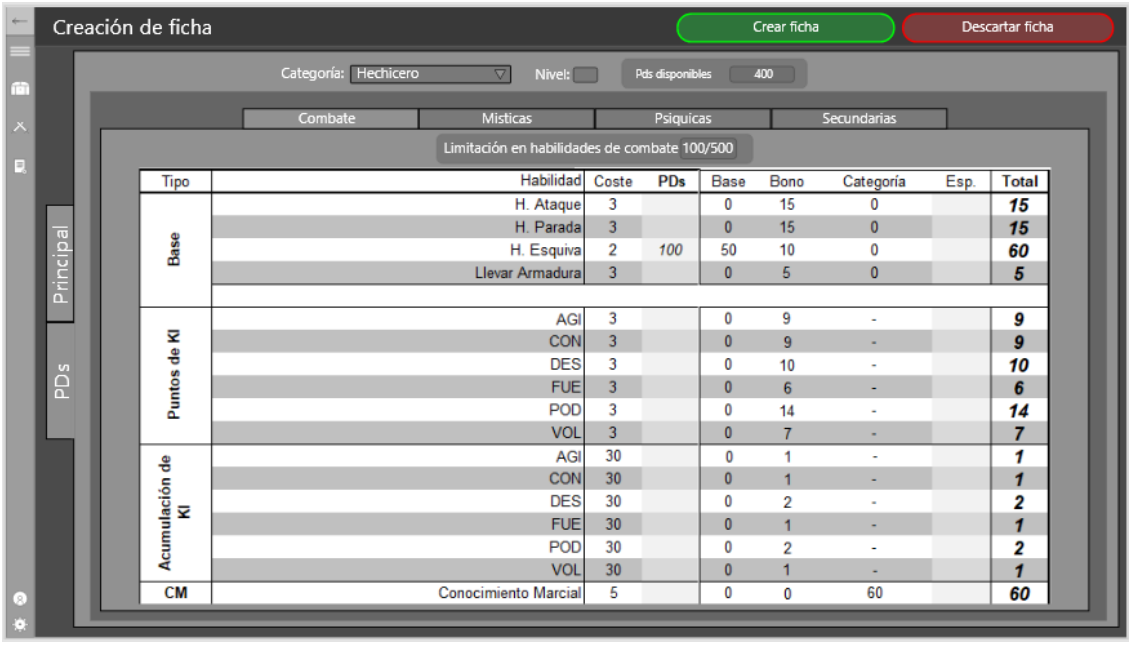
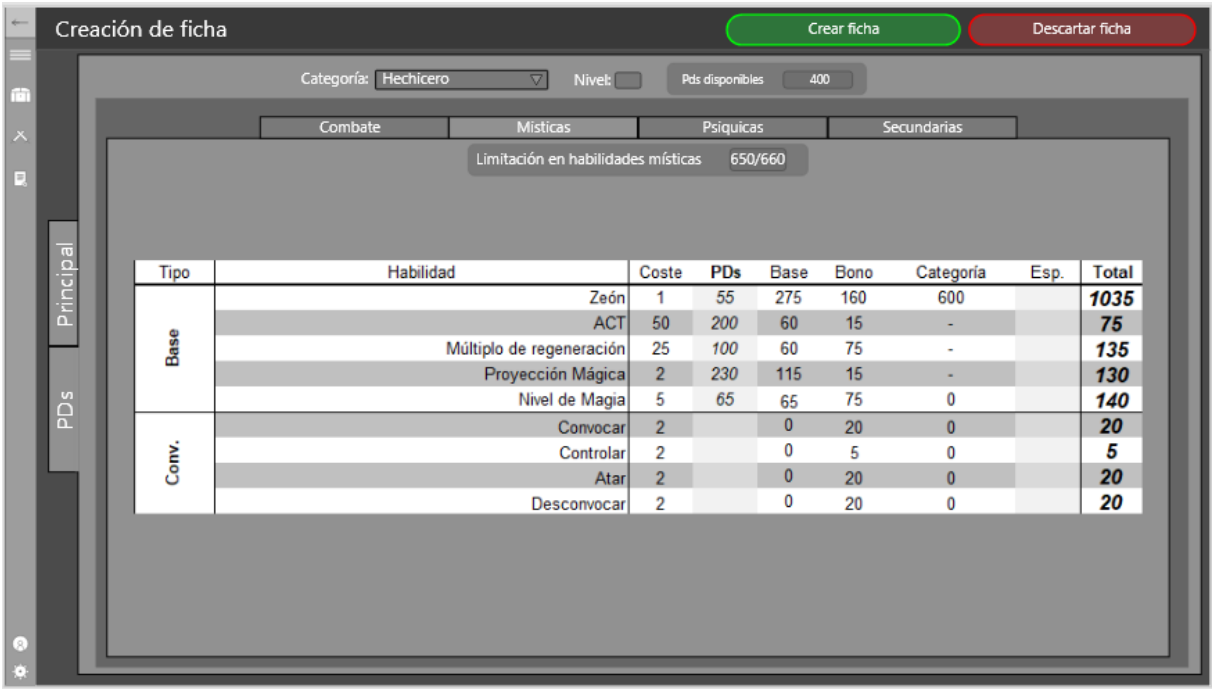


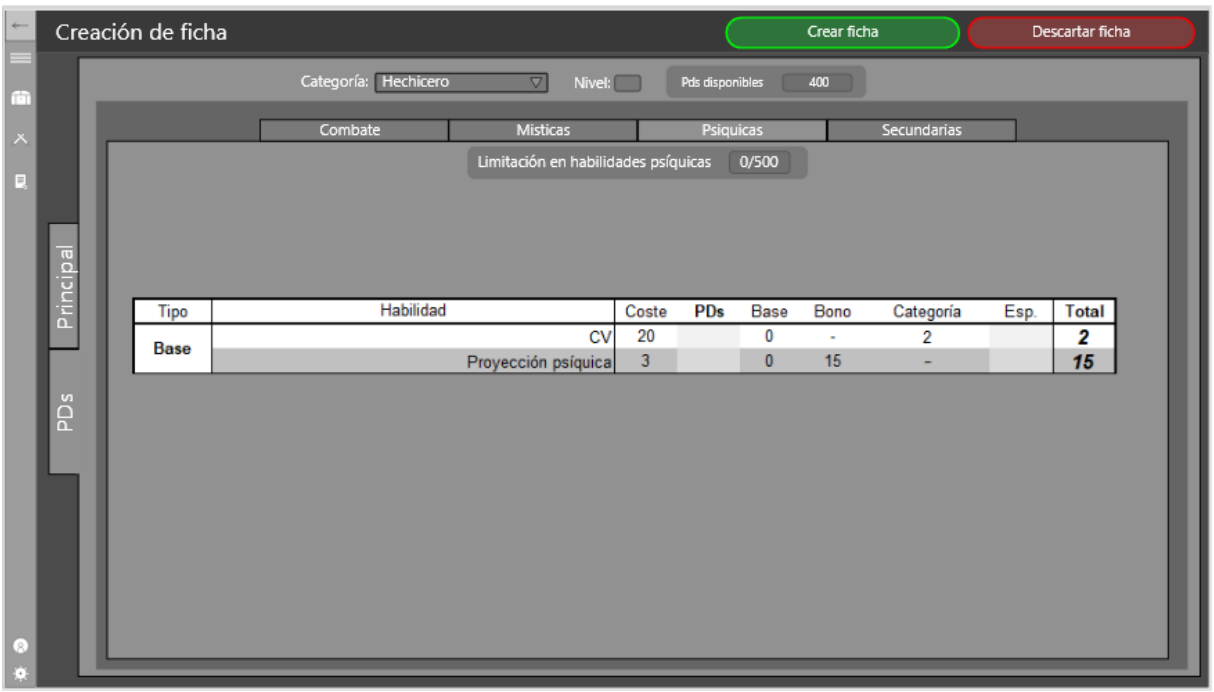
Fig. 43. Creación de ficha, pestaña PDs - Combate



Tipo	Habilidad	Coste	PDs	Base	Bono	Categoría	Esp.	Total
Base	Zeón	1	55	275	160	600		1035
	ACT	50	200	60	15	-		75
	Múltiplo de regeneración	25	100	60	75	-		135
	Proyección Mágica	2	230	115	15	-		130
	Nivel de Magia	5	65	65	75	0		140
Conv.	Convocar	2		0	20	0		20
	Controlar	2		0	5	0		5
	Atar	2		0	20	0		20
	Desconvocar	2		0	20	0		20

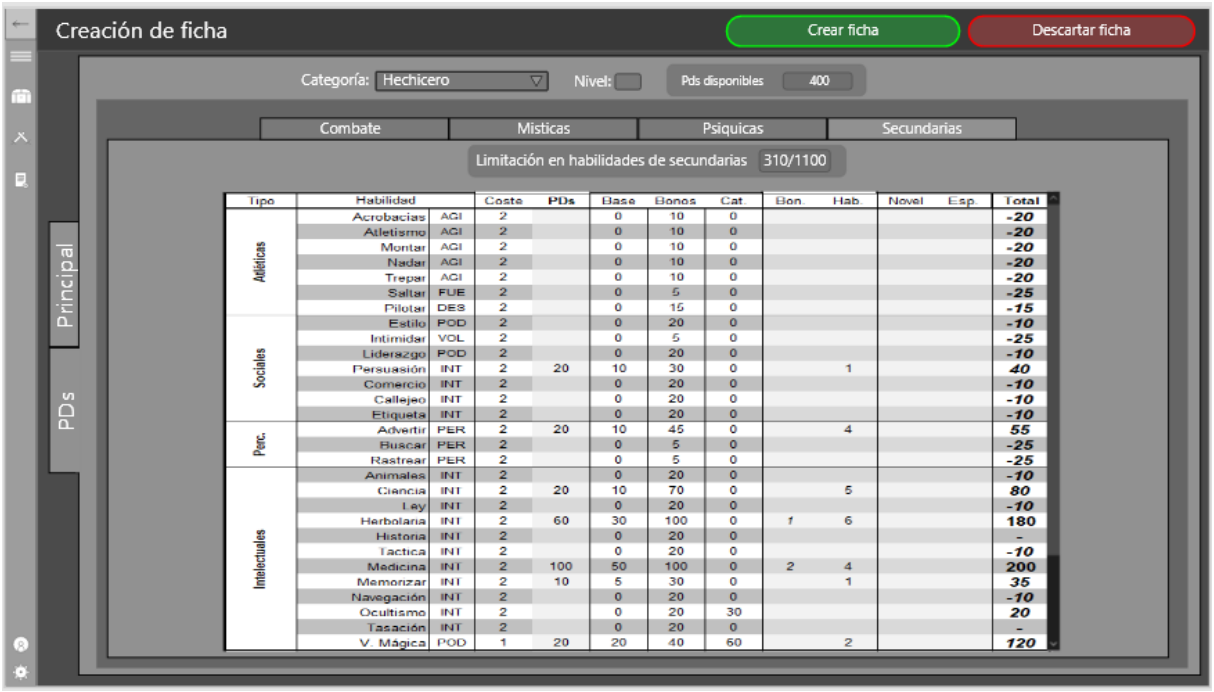
Fig. 44. Creación de ficha, pestaña PDs - Místicas

Se puede comprobar que la pestaña “Psíquicas” está prácticamente vacía. Esto se debe a que el manual base solo incluye dos características para tener en cuenta, mientras que las expansiones aumentan bastante las posibilidades en este aspecto. Por ello, se ha dejado el espacio preparado para futuras actualizaciones de la aplicación.



Tipo	Habilidad	Coste	PDs	Base	Bono	Categoría	Esp.	Total
Base	CV	20		0	-	2		2
	Proyección psíquica	3		0	15	-		15

Fig. 45. Creación de ficha, pestaña PDs - Psíquicas



Tipo	Habilidad		Coste	PDs	Base	Bonos	Cat.	Bon.	Hab.	Nivel	Esp.	Total
Atléticas	Acrobacias	AGI	2		0	10	0					-20
	Atletismo	AGI	2		0	10	0					-20
	Montar	AGI	2		0	10	0					-20
	Nadar	AGI	2		0	10	0					-20
	Trepar	AGI	2		0	10	0					-20
	Saltar	FUE	2		0	5	0					-25
	Pilotar	DES	2		0	15	0					-15
Sociales	Estilo	POD	2		0	20	0					-10
	Intimidar	VOL	2		0	5	0					-25
	Liderazgo	POD	2		0	20	0					-10
	Persuasión	INT	2	20	10	30	0		1			40
	Comercio	INT	2		0	20	0					-10
	Callejeo	INT	2		0	20	0					-10
	Etiqueta	INT	2		0	20	0					-10
Perc.	Advertir	PER	2	20	10	45	0		4			55
	Buscar	PER	2		0	5	0					-25
	Rastrear	PER	2		0	5	0					-25
Intelectuales	Animales	INT	2		0	20	0					-10
	Ciencia	INT	2	20	10	70	0		5			80
	Ley	INT	2		0	20	0					-10
	Herbolaria	INT	2	60	30	100	0	1	6			180
	Historia	INT	2		0	20	0					-
	Táctica	INT	2		0	20	0					-10
	Medicina	INT	2	100	50	100	0	2	4			200
	Memorizar	INT	2	10	5	30	0		1			35
	Navegación	INT	2		0	20	0					-10
	Ocultismo	INT	2		0	20	30					20
	Tasación	INT	2		0	20	0					-
	V. Mágica	POD	1	20	20	40	60		2			120

Fig. 46. Creación de ficha, pestaña PDs – Secundarias

La vista de combate incluye el calculador de turnos con un botón que calcula la siguiente ronda. Este listado de personajes toma los que se han añadido a los equipos participantes, así como la estadística de turno, que sirve para determinar cuan bonificado está en la tirada. Debajo el diseño del cronómetro, en caso de que diera tiempo a su implementación, ya que no está en los requisitos iniciales.

En el centro y en la derecha están los participantes de los dos equipos, con la posibilidad de añadir nuevos desde el almacén personal en la cabecera. Cada tarjeta representa un personaje, con sus atributos principales visibles para una valoración rápida del estado del combate de un vistazo.

Cada tarjeta, además, incluye 3 botones. Uno para ir a la vista de autoataque, otro para verla ficha completa o modificarla (extremadamente útil para directores que quieren comprobar cosas de sus jugadores sin darles la pista de que lo está mirando) y otra para eliminar al personaje del equipo.

Fig. 47. Vista de combate

La vista detalle, que al principio puede parecer muy densa, en realidad es bastante sencilla pues se rellenan todos los datos posibles cogiéndolos del personaje al que pertenecía el botón pulsado. Además de esto, se utilizará un diccionario de datos siendo la clave el personaje y el valor una instancia de la interfaz para guardar las selecciones y ligarlas al personaje, de tal forma de que la próxima vez que se abra la interfaz, se mantengan las selecciones que tenía. Por último, notar que hay dos botones, calcular y aplicar resultado. Cada vez que se selecciona “calcular”, se calcula una nueva tirada distinta con los mismos datos. Estos resultados no se aplicarán a no ser que el usuario seleccione “aplicar resultado”. Esto se ha creado como doble confirmación para evitar errores a la hora de seleccionar los modificadores y para aquellos directores que consideren que determinada tirada arruinaría la diversión de la partida. Efectivamente, se considera que está haciendo trampa, y normalmente no se le dice a los jugadores, pero las opiniones de la comunidad están muy divididas en este aspecto, por lo que puede ser útil, pero no es su principal propósito.

Se puede observar cómo se ha utilizado el contraste para que quede claro que es interactuable en cada momento y para que resalte todas las opciones que tiene el director.

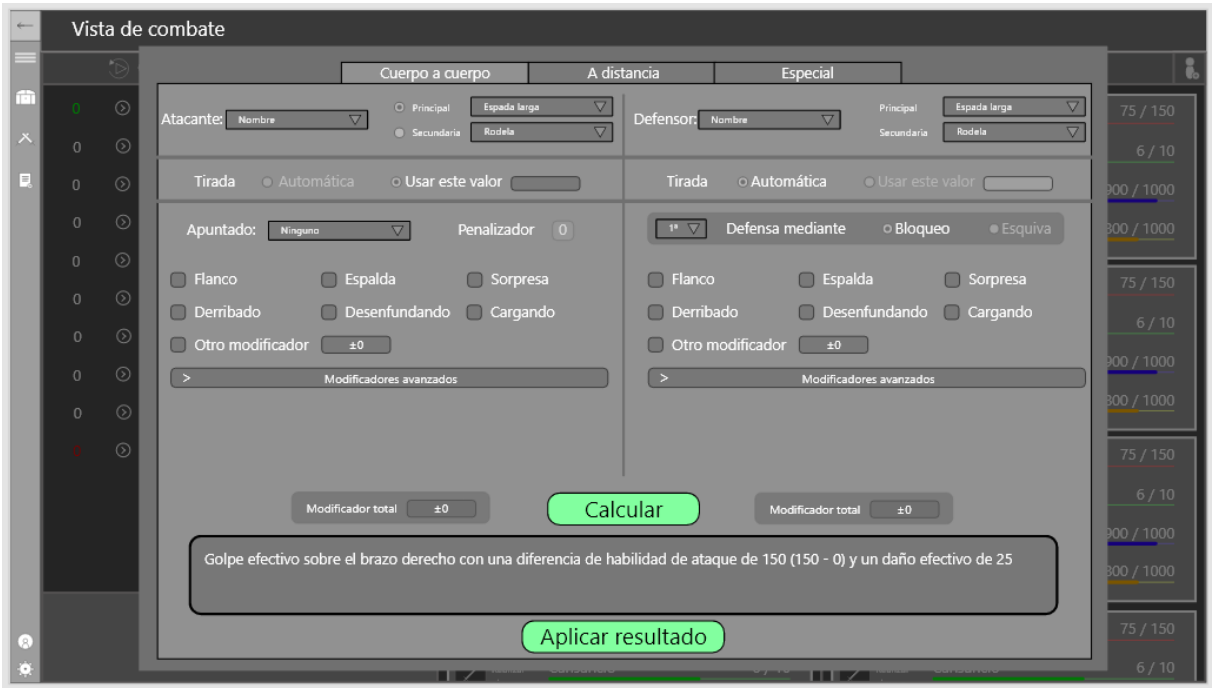


Fig. 48. Vista de cálculo de ataque

Se han utilizado varios principios de Gestalt [11] para facilitar lo máximo posible el entendimiento inmediato de la interfaz.

La vista está dividida horizontalmente entre atacante y defensor utilizando el principio de simetría.

Fig. 49. Vista de cálculo de ataque. Principio de simetría

A su vez, respetando el principio de simetría anteriormente mencionado, hay divisiones verticales utilizando el principio de similitud (de datos, en este caso), de agrupación y de región común para diferenciar claramente entre estado del atacante, forma de calcular la tirada, modificadores y la zona del cálculo del resultado.

Fig. 50. Vista de cálculo de ataque. Principios de similitud, agrupación y región común

Cabe destacar también que, en la sección modificadores, se han agrupado los modificadores que no suelen usarse o que si se usan suele ser rara vez dado que son situaciones que difícilmente se dan. Al agruparlos, se ha logrado reducir el impacto cognitivo que tiene de primeras la gran cantidad de posibilidades que pueden llegar a darse (y más aún en la pestaña de ataques a distancia), ya que, de primeras, solo están los modificadores que son usados normalmente y con los que se están mucho más familiarizados. Se pueden comprobar los modificadores avanzados desplegados a continuación.

Fig. 51. Vista de cálculo de ataque expandida

También hay una vista concreta para los ataques a distancia, en la cual algunos modificadores cambian.

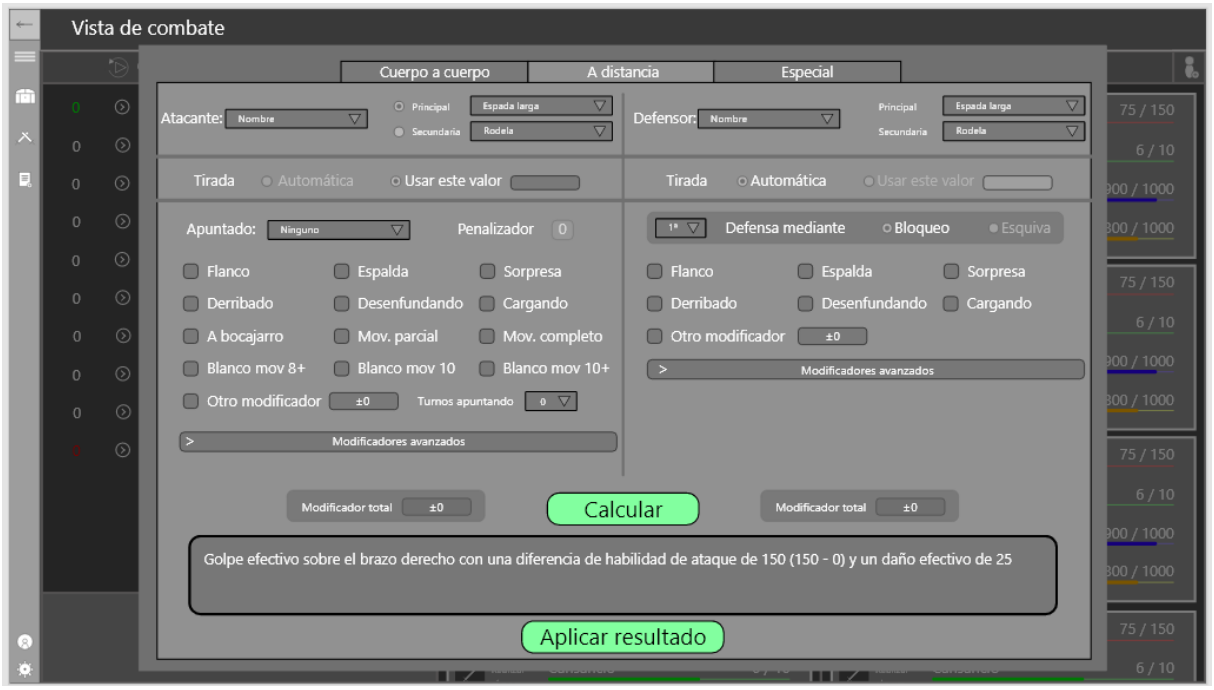


Fig. 52. Vista de cálculo de ataque a distancia

Y, claro está, también utiliza lo mismo que se ha comentado en la vista de cuerpo a cuerpo, la agrupación de situaciones no comunes, y más cuando en combate a distancia hay una cantidad enorme de modificadores.

Fig. 53. Vista de cálculo de ataque a distancia expandida

Claramente, todos los modificadores se encuentran en realidad más separados y con un scrolleable para evitar que estén tan apretados, aunque aquí se han puesto todos para poder visualizarlos.

7.4 Entorno de trabajo

Para el proyecto se han utilizado los lenguajes C# y xaml, debido a que forman parte de UWP el cual está incluido en el estándar .NET. La elección de .NET ha sido puramente personal, dado que tenía alto interés en aprenderlo y esta me ha parecido una buena oportunidad. Como IDE, por supuesto, se ha utilizado Visual Studio 2019, dado que es la mejor elección para trabajar con tecnologías. Esto abre las puertas al trabajo futuro en caso de que se quiera hacer una web o una ampliación a móvil.

Fig. 54. Estructura de .NET

Además, el entorno incluye Windows Template Studio (Fig 55). WTS es una extensión que acelera enormemente la creación de las aplicaciones proporcionando las bases del proyecto o de las distintas vistas que se vayan creando a lo largo de él. Junto con la creación también incluye un ejemplo para poder entender desde el principio cuál es la metodología que se debería seguir, con las conexiones entre clases ya listas. Igualmente es necesario modificar varias cosas, pero esto ayuda mucho.

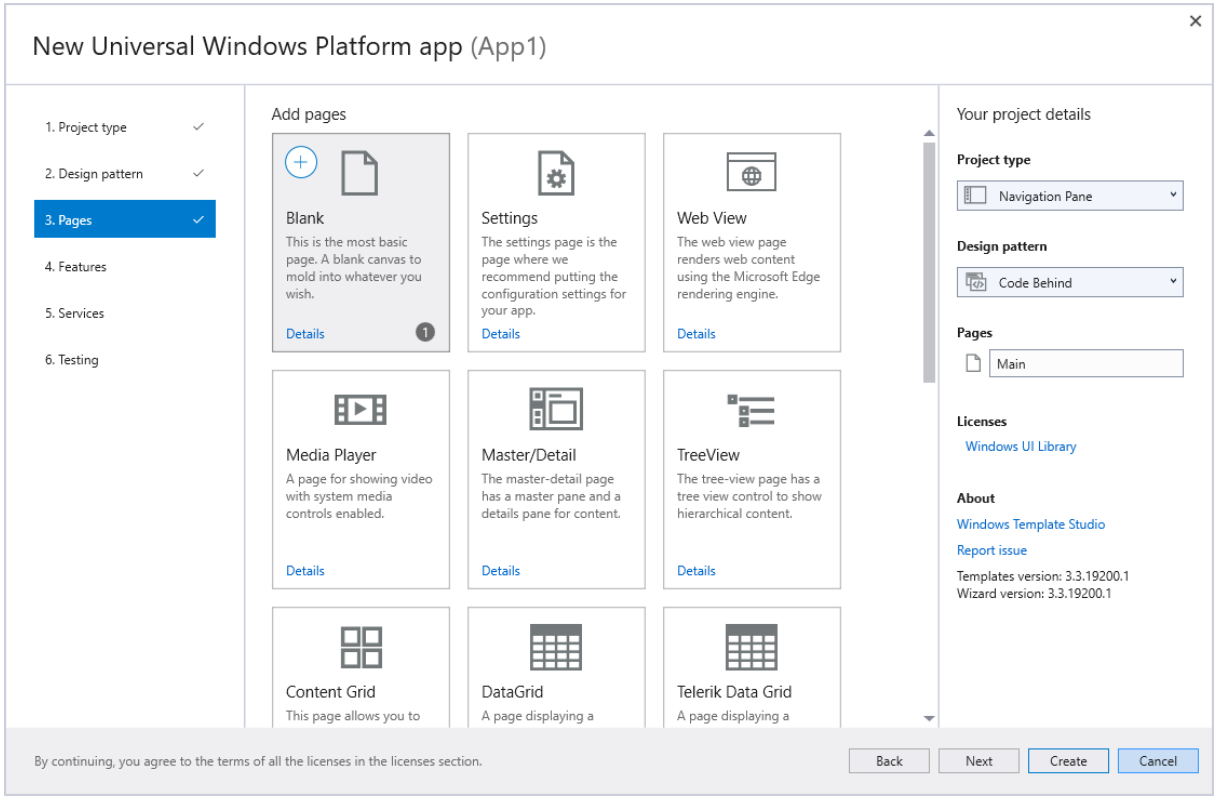


Fig. 55. Vista de Windows Template Studio

7.5 Implementación

La implementación está basada en UWP (Universal Windows Platform) a vista de que se va a realizar un trabajo futuro sobre este proyecto para seguir añadiéndole módulos. Gracias a UWP es posible crear un modelo con las funcionalidades comunes a varios proyectos. Estas funcionalidades comunes se agrupan en el “Core”.

En este proyecto, el Core contiene los Helpers, los Models y los Services.

- Helper: Tipo de clase, generalmente estática, que contiene métodos para implementar o extender una funcionalidad. Solo colecciones, patrones, conversores y gestor(es) de dato(s) podrán ser no estáticos, debido a su naturaleza heredable y de creación de múltiples instancias independientes.

- Model: Clase o estructura no estática que define los objetos básicos del proyecto. No deben existir valores inaccesibles externamente (aunque la implementación puede ocultarlos mediante una variable intermedia). Los únicos métodos que pueden aparecer son: constructores, destructores, ToString y todos aquellos relacionados con la implementación de interfaces del objeto.

- Service: Funciones y clases, se parece mucho a los helpers pero en su lugar gestiona los cambios de la interfaz.

Al contener el Core esos recursos, se pueden reutilizar tantas veces como sea necesario a lo largo de las distintas implementaciones. De esta forma, el proyecto (con el posible trabajo futuro también incluido) quedaría así:

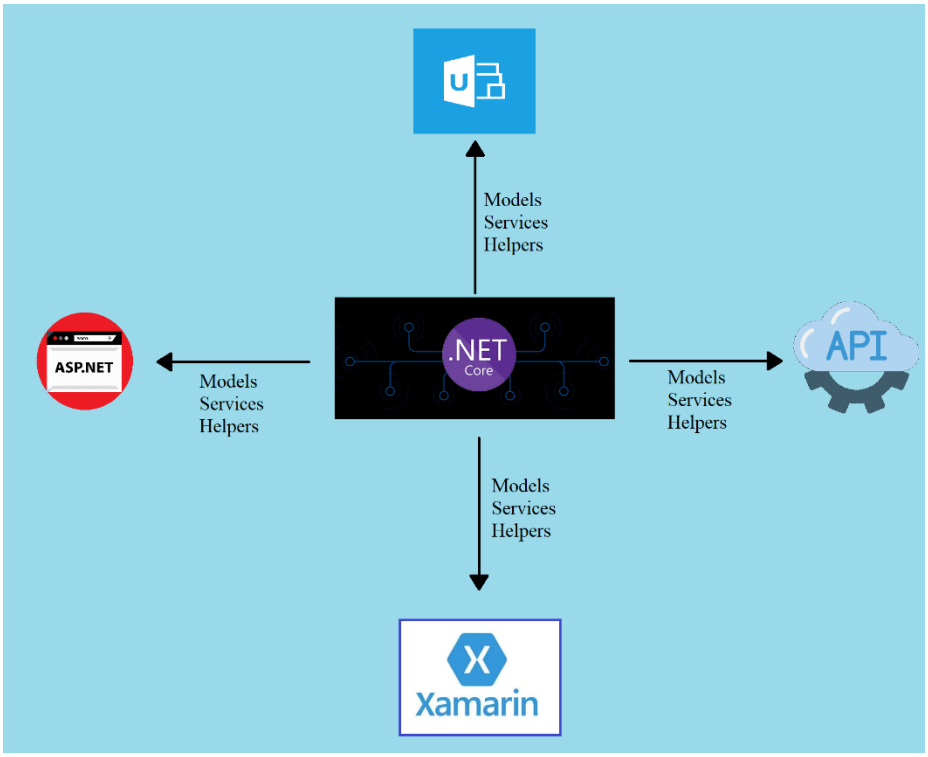


Fig. 56. Organización de un proyecto alimentado por .Net Core

Dentro de UWP (Nuestra aplicación de escritorio), se sigue el patrón Model View Viewmodel. Este patrón permite separar la lógica empresarial y de presentación de una aplicación desde su interfaz de usuario. Mantener una separación clara permite abordar muchos problemas durante el desarrollo y hacer que sea más fácil de probar, mantener y desarrollar. También aumenta la capacidad de reutilización de código y permite a los desarrolladores y diseñadores colaborar con más facilidad.

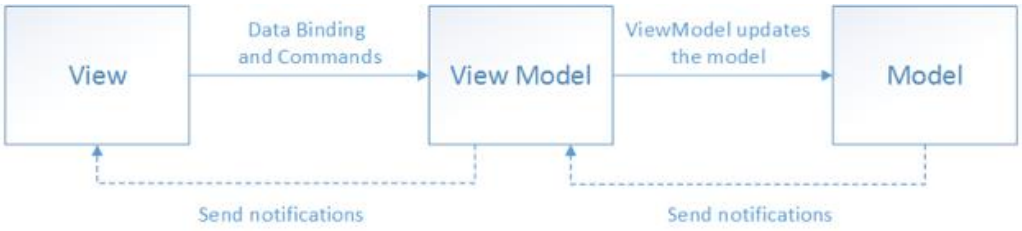


Fig. 57. Estructura Model View view-Model

View: La vista es responsable de definir la estructura, el diseño y la apariencia de lo que el usuario ve en la pantalla.

ViewModel: El modelo de vista implementa las propiedades y los comandos a los que la vista puede enlazarse y notifica a la vista de cualquier cambio de estado a través de los eventos de notificación de

cambios. Las propiedades y los comandos que proporciona el modelo de vista definen la funcionalidad que ofrece la interfaz de usuario, pero la vista determina cómo se mostrará esa funcionalidad.

Model: Las clases de modelo son clases no visuales que encapsulan los datos de la aplicación. Por lo tanto, se puede considerar que el modelo representa el modelo de dominio de la aplicación, que normalmente incluye un modelo de datos junto con la lógica de validación y negocios.

7.6 Persistencia de datos

La persistencia de datos es algo que debe estar claro desde un comienzo para ir desarrollando correctamente la estructura del proyecto.

Podríamos considerar que los datos se almacenan en 3 posibles localizaciones no excluyentes entre ellas, la API, los datos persistentes (almacenamiento local) y las variables (almacenamiento volátil).

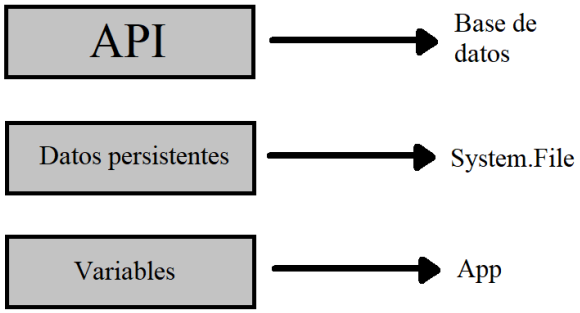


Fig. 58. Persistencia

A la hora de persistir los datos, se toman en cuenta dos posibilidades. La primera en la que la aplicación requiera un determinado campo para mostrárselo al usuario. La segunda en la que un usuario haya creado o modificado una ficha.

En el primer caso, se hacen diversas comprobaciones. Primero, se comprueba si el dato está en las variables en ejecución debido a que ya se ha usado antes esta sesión. En caso de que no esté, se comprueba entonces en los datos persistentes. Si, efectivamente, tampoco está, se recurre entonces a extraer los datos directamente desde la API.

En el segundo caso, al crear una ficha o al cerrarla (si se ha modificado algún cambio que no pertenezca a “estado”) inmediatamente se cambian los datos persistidos en local y se manda un update al endpoint de la API para que actualice esa ficha. En caso de que los cambios incluyan solamente cambios en “estado” se realiza el update solo a nivel local, dado que es irrelevante el estado actual de un personaje en una partida para otra persona que solo le interesa la ficha por cómo está montada.

7.7 Despliegue

Durante el desarrollo del proyecto una empresa mostró interés en actuar como Publisher y tras hablar los términos acepté. Debido a esto, el despliegue utiliza servicios que proporciona la propia empresa.

El despliegue se va a realizar utilizando las comodidades que proporciona Visual Studio en consonancia a las directivas de la empresa. Para ello, en primer lugar, se va a desplegar la API en el servidor de la empresa mediante conexión FTP.

La conexión es necesaria configurarla en el archivo .csproj.

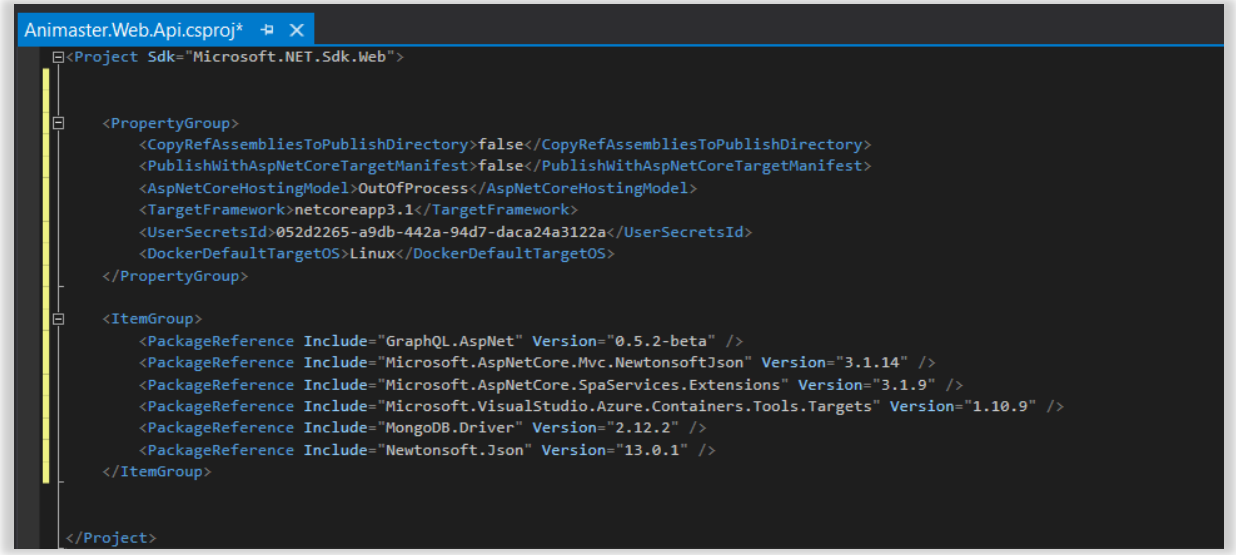


Fig. 59. Configuración de la conexión FTP

Una vez hecho, las opciones de publicación permiten definir la direccion URL que utilizara y diversas credenciales previamente proporcionadas por la empresa.

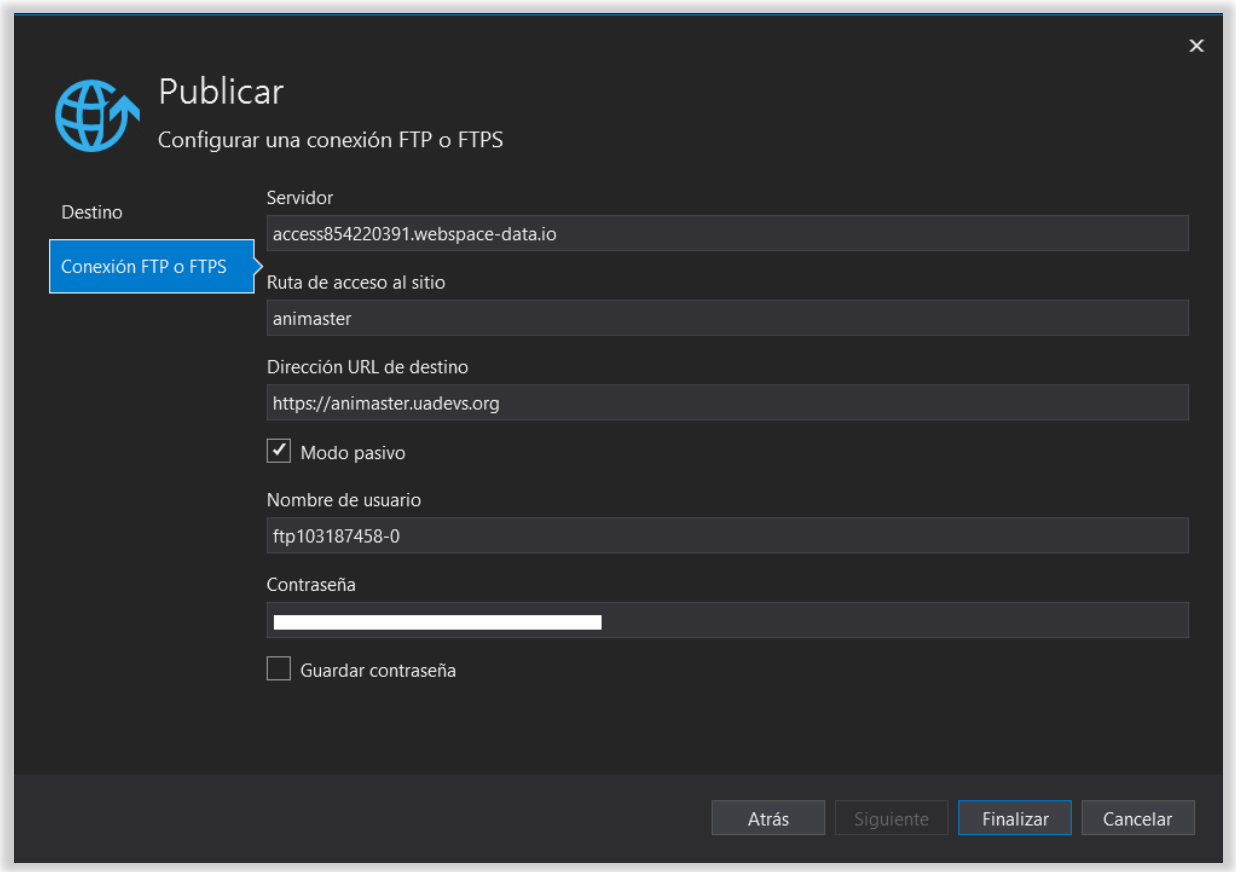


Fig. 60. Publicación mediante FTP

Cabe mencionar que mientras se está publicando la API estará caída, por lo que es necesario tener cuidado de cuando se lanzan nuevas versiones.

Por su parte, la aplicación se empaquetará y se enviará a la empresa para que sea publicada en la Microsoft.

8. Conclusiones

8.1 Resultados

Como director de juego que soy, no podría estar más satisfecho con el resultado. Según iba avanzando en la creación de aplicación, la he ido utilizando en partidas reales y he de decir que, según iba añadiendo módulos, las partidas se han hecho cada vez más disfrutables y ágiles desde mi punto de vista. También he notado en los jugadores cada vez un mayor interés sobre todo por los combates, dado que ahora son muchísimo más dinámicos y rápidos, sin quitarles ese apego por lanzar ellos mismos los dados. De media, cada ronda sin aplicación llevaba entre 20 y 35 minutos. Actualmente se ha reducido a 5 minutos, 10 en los casos más extremos. Me atrevo a decir que ha sido un éxito.

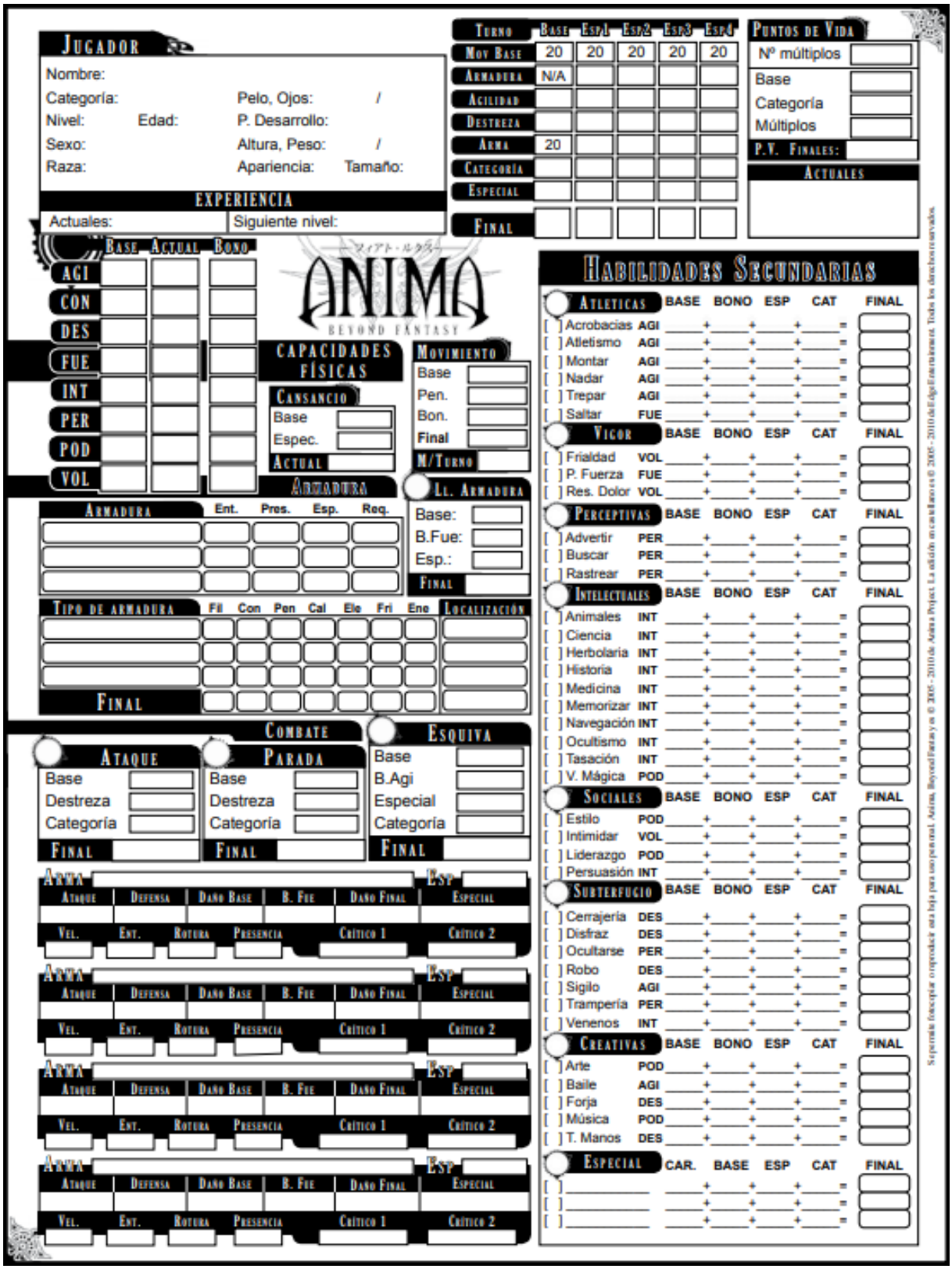
La funcionalidad de que la comunidad haga sus propias fichas y poder navegar por ellas tristemente no ha sido posible utilizarla más allá que para que los jugadores me pasaran la ficha de sus personajes hecha dado que la aplicación, a fecha de hoy, no ha sido lanzada aún para la comunidad de Ánima, aunque se hará en los próximos días en cuanto termine de hablarlo con la empresa publicadora.

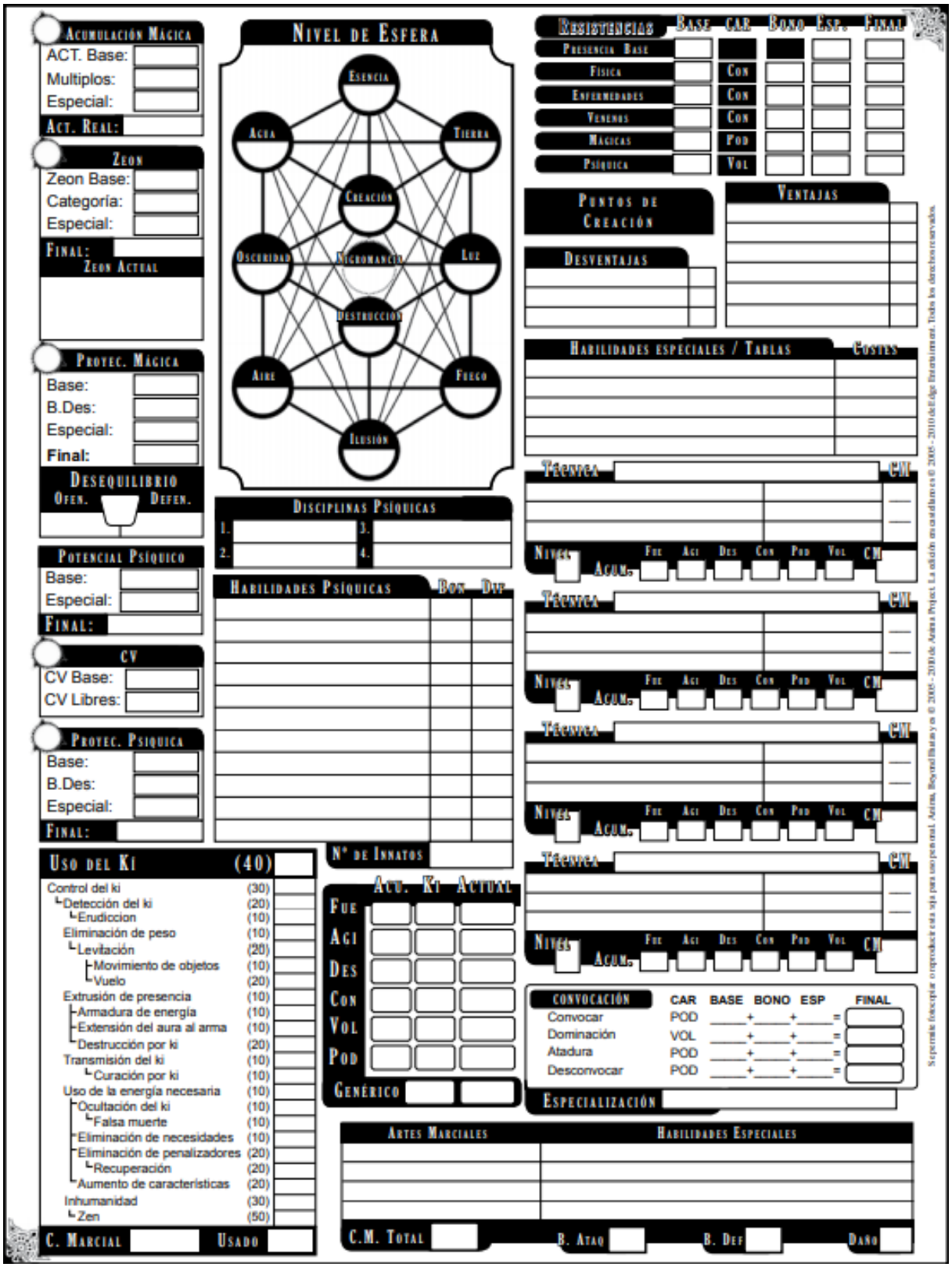
8.2 Futuras mejoras

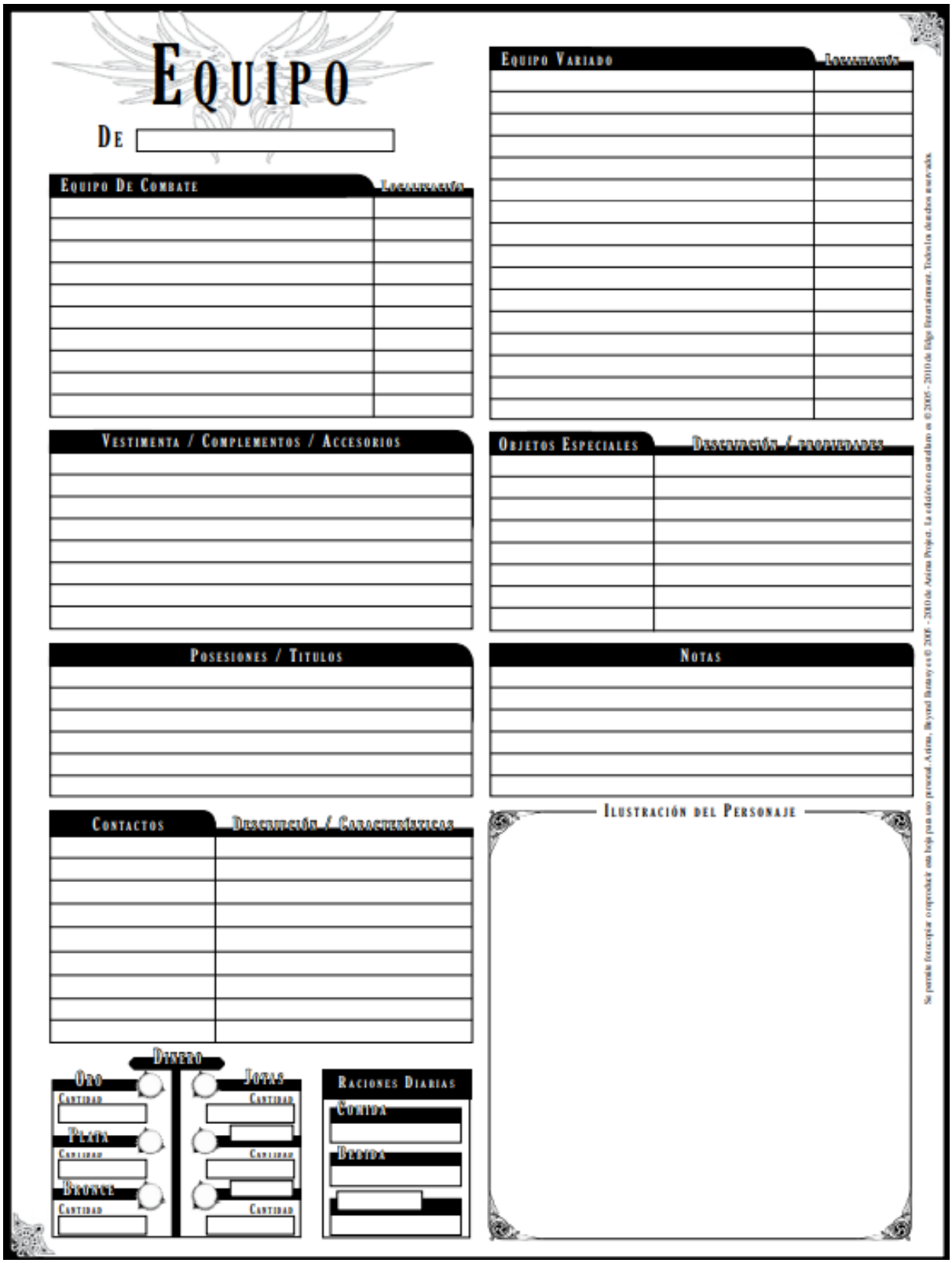
Hay ciertas mejoras que se le podrían realizar al proyecto, entre ellas:

- Un sistema de login: Aunque la aplicación funciona bien con UUID, sería interesante añadir un sistema de login para que los usuarios puedan utilizar sus fichas estén donde estén sin tener que utilizar el ID para traerse las fichas al nuevo dispositivo.
- Añadir más módulos: Ánima tiene muchos manuales, algunos más populares que otros, pero todos igualmente extensos. Sería interesante incluir módulos para poder incluirlos en la aplicación, como la creación de objetos y habilidades personalizadas.
- Una web para poder ver fichas online: Sería interesante (y ya está en proceso) poder comprobar fichas online sin tener que recurrir a la aplicación de escritorio. Ya sea porque no está instalada y solo quieres ver lo que ha compartido algún amigo contigo, o bien porque todavía no sepas de su existencia. Deberá añadir también la posibilidad de copiar la ficha a condición, por supuesto, de que tengas la aplicación instalada. Una buena forma de mostrarle a los jugadores lo que se están perdiendo por no tenerla.
- Sistema de monetización: Siendo realistas, aunque inicialmente la aplicación es gratuita, sigue siendo un producto fruto de muchas horas de trabajo, por lo que sería interesante aplicar un sistema de monetización para diversas funciones dentro de la misma. Por supuesto, estas funciones solo serán complementarias y bajo ningún concepto serán funciones básicas necesarias para desarrollar con normalidad una partida. Por supuesto, la publicidad queda descartada pues aplica un detrimento notable a la experiencia de usuario.
- Una separación de la aplicación del director de juego con la de los jugadores y su conexión en tiempo real: Debido a que los directores y los jugadores tienen papeles tan distintos también deberían tener aplicaciones distintas, conectadas entre ellas a tiempo real para no depender de compartir las fichas para conseguir sus datos y, de paso, que los cambios que produce el director de juego en ellas se reflejen en la pantalla de los jugadores en tiempo real.

Anexo 1 – Ficha de personaje




Acumulación Mágica
ACT. Base:
Multiplos:
Especial:
Act. Real:
Zeon
Zeon Base:
Categoría:
Especial:
Final:
Zeon Actual
Proyec. Mágica
Base:
B.Des:
Especial:
Final:
Desequilibrio
Ofen.
Defen.
Potencial Psíquico
Base:
Especial:
Final:
CV
CV Base:
CV Libres:
Proyec. Psíquica
Base:
B.Des:
Especial:
Final:
Nivel de Esfera
Esencia
Agua
Tierra
Creación
Oscuridad
Nigromancia
Luz
Destrucción
Aire
Fuego
Ilusión
Disciplinas Psíquicas
1.
2.
3.
4.
Habilidades Psíquicas
Bon.
Dif.
Nº de Innatos
Uso del Ki
(40)
Control del ki (30)
Detección del ki (20)
Erudicion (10)
Eliminación de peso (10)
Levitación (20)
Movimiento de objetos (10)
Vuelo (20)
Extrusión de presencia (10)
Armadura de energía (10)
Extensión del aura al arma (10)
Destrucción por ki (20)
Transmisión del ki (10)
Curación por ki (10)
Uso de la energía necesaria (10)
Ocultación del ki (10)
Falsa muerte (10)
Eliminación de necesidades (10)
Eliminación de penalizadores (20)
Recuperación (20)
Aumento de características (20)
Inhumanidad (30)
Zen (50)
C. Marcial
Usado
Acu. Ki Actual
Fue
Agi
Des
Con
Vol
Pod
Genérico
Resistencias
Base
Car
Bono
Esp.
Final
Presencia Base
Física
Con
Enfermedades
Con
Venenos
Con
Mágicas
Pod
Psíquica
Vol
Puntos de Creación
Ventajas
Desventajas
Habilidades especiales / Tablas
Costes
Técnica
CM
Nivel
Acum.
Fue
Agi
Des
Con
Pod
Vol
Convocación
CAR
BASE
BONO
ESP
FINAL
Convocar
Dominación
Atadura
Desconvocar
Especialización
Artes Marciales
Habilidades Especiales
C.M. Total
B. Ataq
B. Def
Daño



EQUIPO

DE

Equipo De Combate	Localización

Equipo Variado	Localización

Vestimenta / Complementos / Accesorios

Objetos Especiales	Descripción / propiedades

Posesiones / Títulos

Notas

Contactos	Descripción / Características

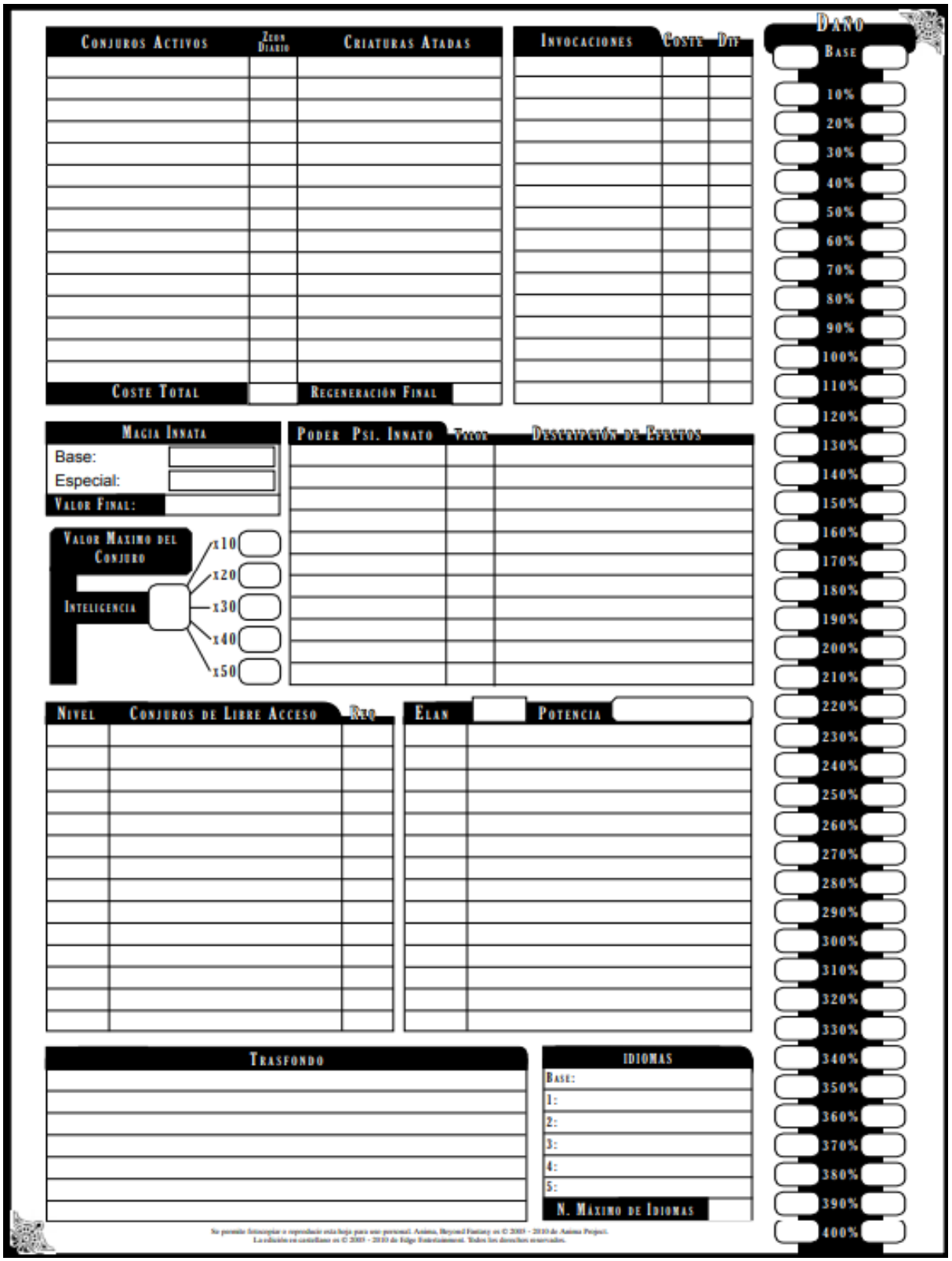
Dinero

Oro	Joyas
Cantidad	Cantidad
Plata	Cantidad
Cantidad	Cantidad
Bronce	Cantidad
Cantidad	Cantidad

Raciones Diarias
Comida
Bebida

Ilustración del Personaje

Se permite fotocopiar o reproducir esta hoja para uso personal. Ánima, Beyond Fantasy es © 2005 - 2010 de Ánima Project. La edición en castellano es © 2005 - 2010 de Edge Entertainment. Todos los derechos reservados.


Conjuros Activos
Zeon Diario
Criaturas Atadas
Coste Total
Regeneración Final
Invocaciones
Coste
Dif
Daño
Base
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
110%
120%
130%
140%
150%
160%
170%
180%
190%
200%
210%
220%
230%
240%
250%
260%
270%
280%
290%
300%
310%
320%
330%
340%
350%
360%
370%
380%
390%
400%
Magia Innata
Base:
Especial:
Valor Final:
Valor Maximo del Conjuro
Inteligencia
x10
x20
x30
x40
x50
Poder Psi. Innato
Coste
Descripción de Efectos
Nivel
Conjuros de Libre Acceso
Req
Elan
Potencia
Trasfondo
Idiomas
Base:
1:
2:
3:
4:
5:
N. Máximo de Idiomas

Anexo 2 – Encuesta para directores de juego

Cuestionario sobre Ánima Beyond Fantasy I (Versión para directores de juego)

Bienvenido y gracias por participar en este cuestionario.
A continuación se le realizaran algunas preguntas respecto a diferentes aspectos relacionados con Ánima.
Recordar que para poder realizar esta encuesta, es necesario haber jugado como Director de juego.
En caso de que seas tanto director como jugador, por favor, al contestar a las preguntas responda desde el punto de vista del director de juego y la experiencia que le ha brindado serlo, ignorando la repercusión que pueda tener en la respuesta su experiencia como jugador.

Cabe mencionar que los resultados de la encuesta y su veracidad es realmente importante, por lo que, por favor, tómese el tiempo necesario para leer las preguntas con calma y dar las respuestas mas precisas posibles. En caso de no saber con exactitud que respuesta dar, déjelo en blanco.

*Obligatorio

Dirección de correo electrónico *

Tu dirección de correo electrónico

Nombre completo *

Tu respuesta

Tiempo jugando Ánima como Director de juego

- ◯ Entre 0 y 6 meses
- ◯ Entre 7 y 12 meses
- ◯ Entre 1 y 2 años
- ◯ Entre 2 y 5 años
- ◯ Más de 5 años

¿Cuantas partidas activas tiene actualmente en curso (Como director de juego)?

- ◯ Ninguna
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ Más de 3

En situaciones personales normales (Con tiempo libre suficiente), ¿Con que frecuencia suele jugar como Director de juego?

- ◯ Prácticamente todos los días
- ◯ Mas de una vez a la semana
- ◯ Al menos una vez a la semana
- ◯ Al menos una vez cada dos semanas
- ◯ Al menos una partida al mes
- ◯ Menos de una partida al mes

¿Es normal la presencia de dispositivos electrónicos en sus partidas?

- ◯ Sí
- ◯ No
- ◯ En algunas partidas si, en otras no

¿Cuanto le gustan las siguientes situaciones?

	Lo odio	Me disgusta	Me es indiferente	Me gusta	Me encanta
Tirar los dados en formato físico yo mismo	◯	◯	◯	◯	◯
Tirar los dados en formato digital (Como roll20)	◯	◯	◯	◯	◯
Pulsar un botón y recibir el resultado, sin importar el numero de variables que haya en la tirada	◯	◯	◯	◯	◯
Otra persona tira los dados en mi lugar	◯	◯	◯	◯	◯

Actualmente, en una partida normal, ¿Como de tediosas (o difíciles) encuentra las siguientes situaciones?

	Tedioso	Ligeramente molesto	Indiferente	Ligeramente interesante	Interesante
Tener una visión general del estado de los personajes de los jugadores	◯	◯	◯	◯	◯
Saber que tienen en el inventario los jugadores	◯	◯	◯	◯	◯
Saber las estadísticas de los jugadores	◯	◯	◯	◯	◯
Gestionar los NPCs que tenga preparados	◯	◯	◯	◯	◯
Gestionar la música	◯	◯	◯	◯	◯
Organizar los turnos del combate según el reglamento (Al inicio de cada ronda)	◯	◯	◯	◯	◯
La duración de las tiradas en combate	◯	◯	◯	◯	◯
El estado de los participantes de un combate	◯	◯	◯	◯	◯
Gestionar las tiradas complejas (Con muchos modificadores y variables)	◯	◯	◯	◯	◯
Preparar un mapa (Si lo usa)	◯	◯	◯	◯	◯

Para finalizar, ¿Hay algún aspecto que no se haya tratado en alguna pregunta anterior que no le guste o que le parezca tedioso a la hora de jugar como director de juego? Puede ser relacionado con cualquier ámbito que afecte a una partida de Rol, desde la organización de las partidas hasta la finalización de estas

Anexo 3 – Encuesta para Jugadores

Cuestionario sobre Ánima Beyond Fantasy I (Versión para jugadores)

Bienvenido y gracias por participar en este cuestionario.
A continuación se le realizaran algunas preguntas respecto a diferentes aspectos relacionados con Ánima.
Recordar que para poder realizar esta encuesta, es necesario haber participado en partida como jugador.
En caso de que seas tanto director como jugador, por favor, al contestar a las preguntas responda desde el punto de vista del jugador y la experiencia que le ha brindado serlo, ignorando la repercusión que pueda tener en la respuesta su experiencia como director de juego.

Cabe mencionar que los resultados de la encuesta y su veracidad es realmente importante, por lo que, por favor, tómese el tiempo necesario para leer las preguntas con calma y dar las respuestas mas precisas posibles. En caso de no saber con exactitud que respuesta dar, déjelo en blanco.

*Obligatorio

Dirección de correo electrónico *

Tu dirección de correo electrónico

Nombre completo

Tu respuesta

Tiempo jugando Ánima como jugador

- ◯ Entre 0 y 6 meses
- ◯ Entre 7 y 12 meses
- ◯ Entre 1 y 2 años
- ◯ Entre 2 y 5 años
- ◯ Más de 5 años

¿Cuantas partidas activas tiene actualmente en curso (Como jugador)?

- ◯ Ninguna
- ◯ 1
- ◯ 2
- ◯ 3
- ◯ Más de 3

En situaciones personales normales (Con tiempo libre suficiente), ¿Con que frecuencia suele jugar como jugador?

- ◯ Prácticamente todos los dias
- ◯ Mas de una vez a la semana
- ◯ Al menos una vez a la semana
- ◯ Al menos una vez cada dos semanas
- ◯ Al menos una partida al mes
- ◯ Menos de una partida al mes

¿Es normal la presencia de dispositivos electrónicos en sus partidas?

- ◯ Sí
- ◯ No
- ◯ En algunas partidas si, en otras no

¿Cuanto le gustan las siguientes situaciones?

	Lo odio	Me disgusta	Me es indiferente	Me gusta	Me encanta
Tirar los dados en formato físico yo mismo	◯	◯	◯	◯	◯
Tirar los dados en formato digital (Como roll20)	◯	◯	◯	◯	◯
Pulsar un botón y recibir el resultado, sin importar el numero de variables que haya en la tirada	◯	◯	◯	◯	◯
Otra persona tira los dados en mi lugar	◯	◯	◯	◯	◯
Crear un personaje nuevo	◯	◯	◯	◯	◯
Gestionar la subida de nivel de mi personaje	◯	◯	◯	◯	◯

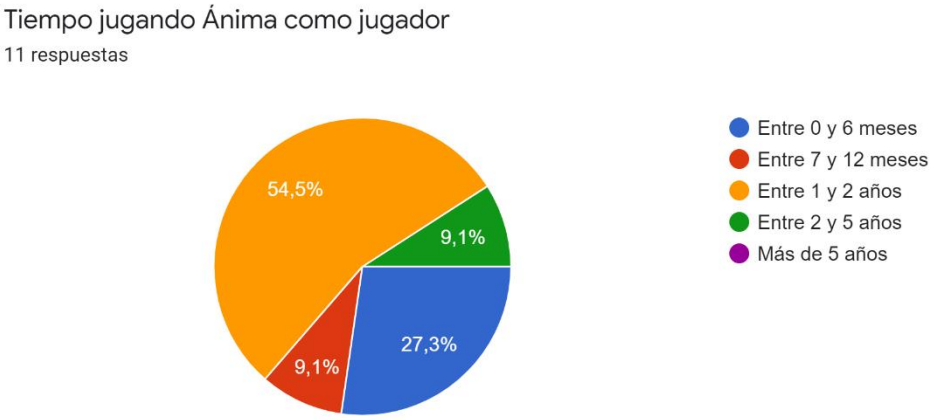
Actualmente, en una partida normal, ¿Como de tediosas (o difíciles) encuentra las siguientes situaciones?

	Tedioso	Ligeramente molesto	Indiferente	Ligeramente interesante	Interesante
Crear un personaje	◯	◯	◯	◯	◯
Gestionar la subida de nivel de un personaje	◯	◯	◯	◯	◯
Mantener organizados y a mano todos mis personajes (En caso de tener varios, por ejemplo, en varias partidas diferentes)	◯	◯	◯	◯	◯
Tener una visión general clara del estado de mi personaje	◯	◯	◯	◯	◯
Llevar un recuento correcto de mi inventario	◯	◯	◯	◯	◯
Buscar algo en mi ficha cuando el director lo requiere	◯	◯	◯	◯	◯
Tener a mano todas mis habilidades/hechizos	◯	◯	◯	◯	◯
Gestionar las invocaciones de mi personaje (Familiares, invocaciones, homúnculos)	◯	◯	◯	◯	◯
Saber todas las diferentes posibilidades que tengo en combate (Según el manual)	◯	◯	◯	◯	◯
Saber las reglas del manual	◯	◯	◯	◯	◯

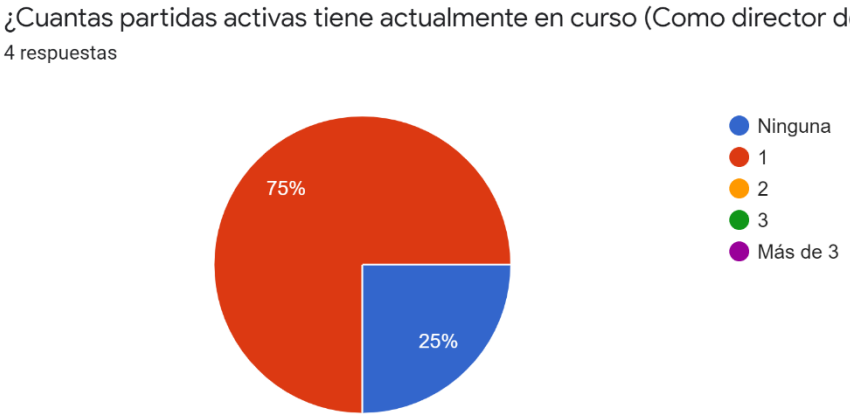
Para finalizar, ¿Hay algún aspecto que no se haya tratado en alguna pregunta anterior que no le guste o que le parezca tedioso a la hora de participar en una partida como jugador? Puede ser relacionado con cualquier ámbito que afecte a una partida de Rol, desde la organización de las partidas hasta la finalización de estas

Anexo 4 – Resultado de las encuestas

Anexo 4.1. Tiempo jugado (Director)

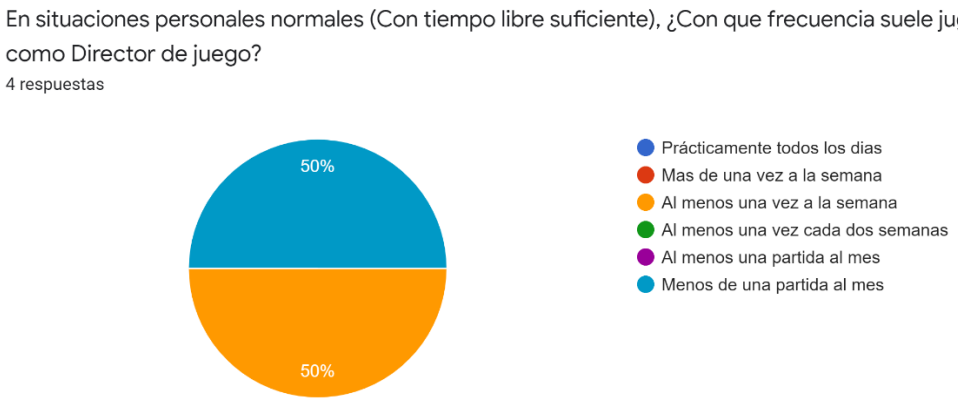


Anexo 4.2. Tiempo jugando (Jugador)

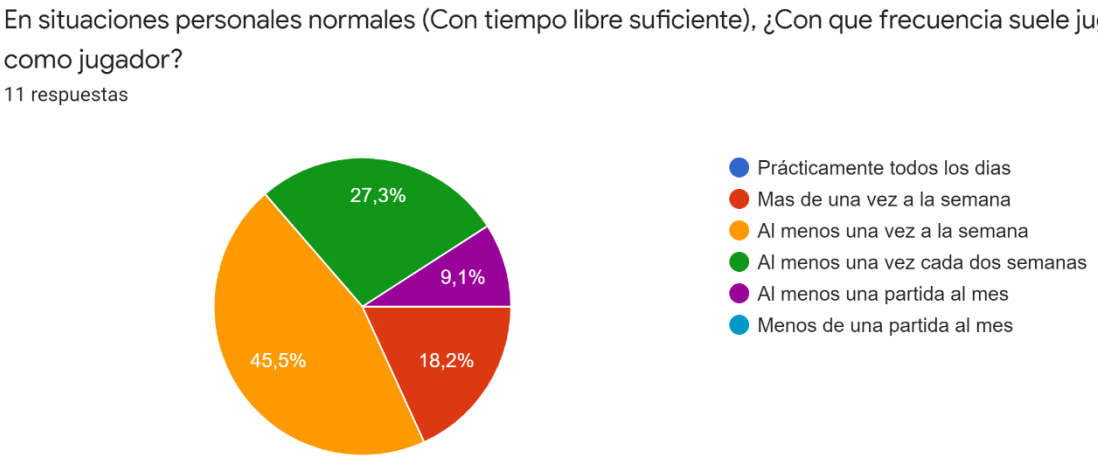


Anexo 4.3. Partidas activas (Director)

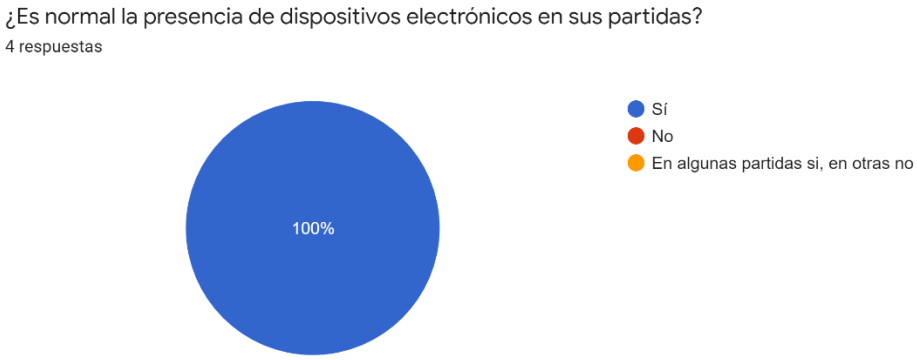
Anexo 4.4. Partidas activas (Jugador)



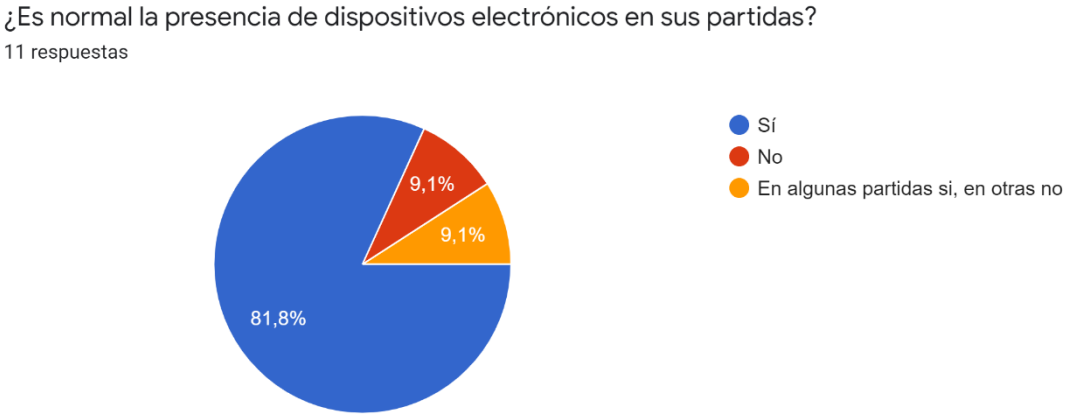
Anexo 4.5. Frecuencia de juego (Director)



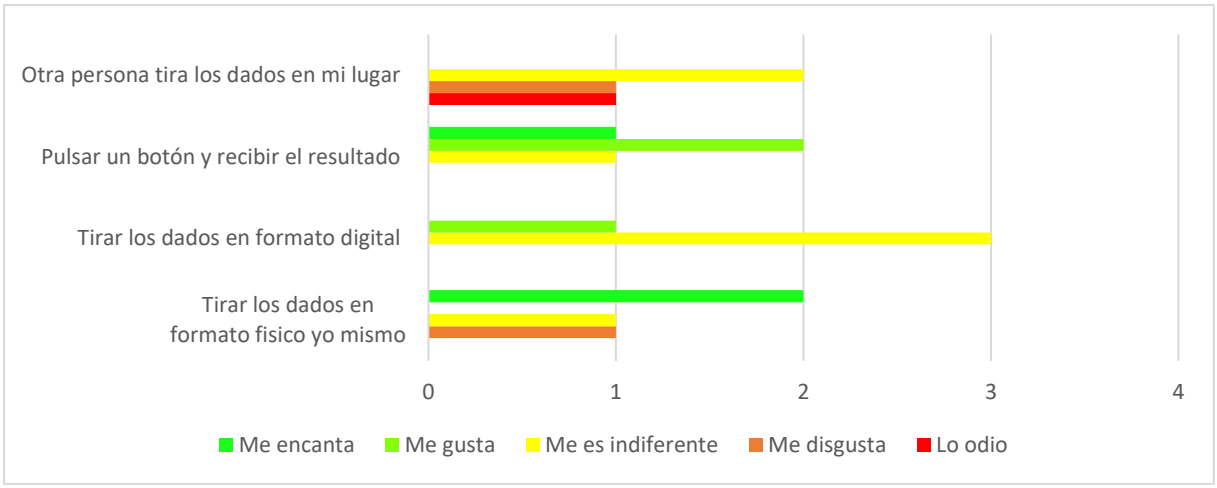
Anexo 4.6. Frecuencia de juego (Jugador)



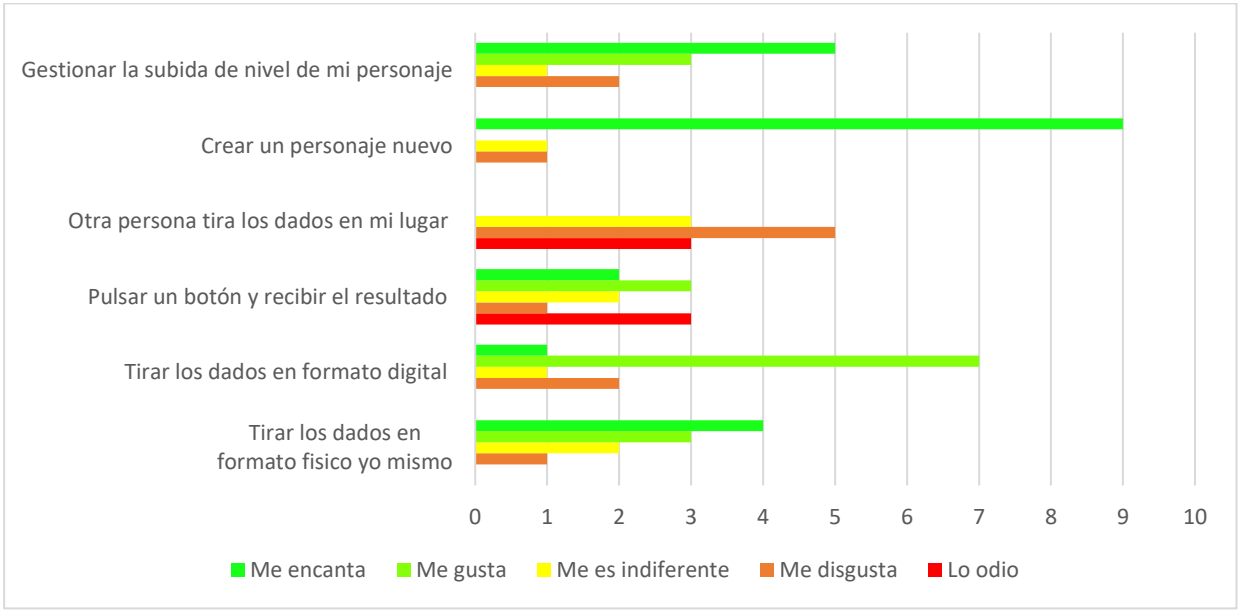
Anexo 4.7. Presencia electrónica (Director)



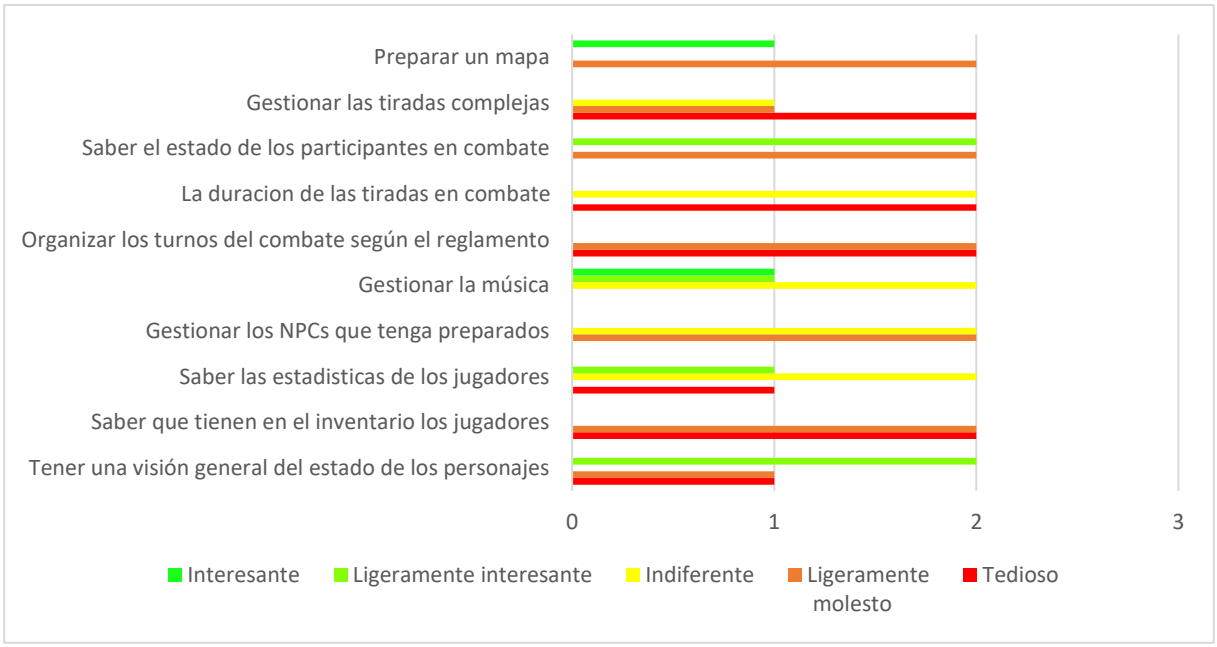
Anexo 4.8. Presencia electrónica



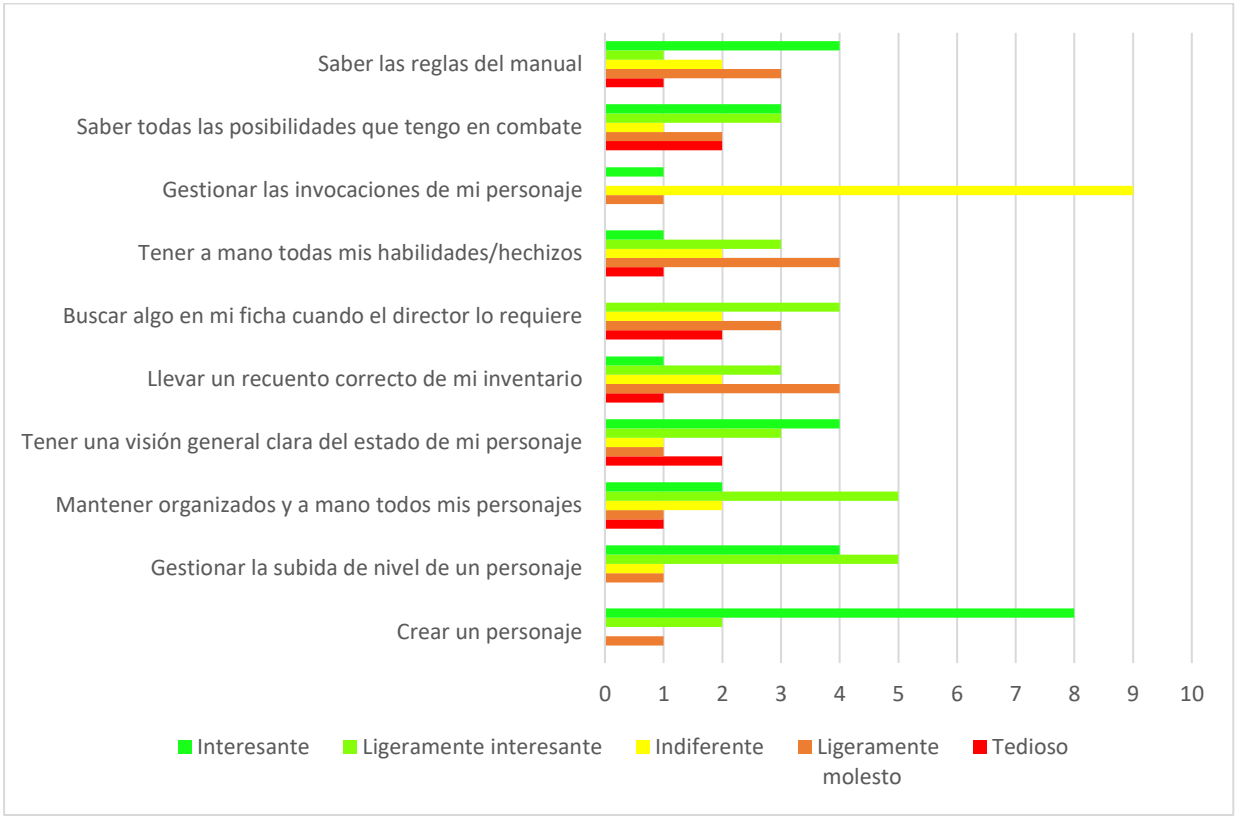
Anexo 4.9. Serie de preguntas 1 (Director)



Anexo 4.10. Serie de preguntas 1 (Jugador)



Anexo 4.11. Serie de preguntas 2 (Director)



Anexo 4.12. Serie de preguntas 2 (Jugador)

Bibliografía

[1] Zambrano, J. R. (2019, 11 julio). 40 Million People Have Played D&D According To WotC. Recuperado 4 de julio de 2021, de https://www.belloflostsouls.net/2019/07/40-million-people-play-dd-according-to-wotc.html

[2] Edge Entertaiment. (2005). Anima: el juego de rol | Wiki Anima: Beyond Fantasy | Fandom. Wiki Anima: Beyond Fantasy. https://anima-beyondfantasy.fandom.com/es/wiki/Anima:_el_juego_de_rol

[3] Entertainment, E. (s. f.). Edge Entertainment - edgeent.com. Recuperado 4 de julio de 2021, de http://www.edgeent.com/

[4] The Orr Group, LLC. (2012, 12 septiembre). Roll20. Recuperado 4 de julio de 2021, de https://roll20.net/

[5] Sommerville, I. (2005). Ingeniería del software. Pearson educación.

[6] Microsoft Azure. (s. f.). ¿Qué es DevOps? Explicación de DevOps | Microsoft Azure. Recuperado 4 de julio de 2021, de https://azure.microsoft.com/es-es/overview/what-is-devops/

[7] Rossberg, J. (2019). Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics. Apress.]

[8] Rizwan, S. (2017, 11 octubre). An Introduction to RESTful APIs. Recuperado 7 de julio de 2021, de https://dzone.com/articles/an-introduction-to-restful-apis

[9] Stack overflow. (s. f.). Stack Overflow Developer Survey 2020. Recuperado 4 de julio de 2021, de https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-databases-wanted4

[10] M. (2019, 5 enero). Introducción práctica a Docker y Docker-Compose - manufosela. Recuperado 7 de julio de 2021, de https://medium.com/@manufosela/introducci%C3%B3n-pr%C3%A1ctica-a-docker-y-docker-compose-f0dba8d7fd28

[11] Sani, S. M., & Shokooh, Y. K. (2016, April). Minimalism in designing user interface of commercial websites based on Gestalt visual perception laws (Case study of three top brands in technology scope). In *2016 Second International Conference on Web Research (ICWR)* (pp. 115-124). IEEE.

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

Este proyecto pretende crear una aplicación de escritoriopara el sistema de rol "Ánima: Beyond Fantasy". Este sistema de rol es de los más complejos disponibles en el mercado, pero esto incurre en que también es el que más situaciones abarca y el que, dentro de la fantasía de su narrativa, se pude considerar más realista. A lo largo del proyecto se utilizan diversas tecnologías recientes que han ido ganando cada vez más popularidad entre los desarrolladores y que pueden realizar aportaciones muy valiosas.

This project aims to create a desktop application for the role-playing system “Anima: Beyond Fantasy”. This system is one of the most complex available on the market, but this means that it is also the one that encompasses the most situations and the one that, within the fantasy of its narrative, can be the most realistic. A variety of recent technologies that have become increasingly popular with developers are used throughout the project due to them being able to provide valuable input