TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Aplicación Web para la Difusión de Convocatorias de Ayudas Universitarias

Curso 2020/2021

Alumno/a:

Eduardo Burruezo Jiménez

Director/es:

Jose Antonio Piedra Fernández

Agradezco este trabajo a toda mi familia, mis padres, mi hermana por toda la ayuda que me han brindado tras tantos años de trabajo y facilitarme las herramientas necesarias para poder lograrlo, sin ellos no estaría donde estoy en la vida.

A mis abuelos, que son las personas que más ilusión les hace que termine de estudiar y a las que más ilusión me hace mostrarles dónde he llegado.

A mi tutor Jose Antonio, por su inestimable ayuda y a Alfonso Bosch, quien aun estando de baja se ofreció a ayudarme en todo lo posible y más.

También agradecer a mis amigos que he conocido a lo largo de estos años Juan José Montoya, Pedro Antonio Galera, Silvia Almodóvar, Teresa Martínez, Almudena Gutiérrez y Francisco José Soto que han hecho los malos momentos menos malos y los buenos momentos mejores si aún cabe y de los que me llevo cosas maravillosas.

Agradecer a todos y cada uno de mis amigos de Murcia, en especial a uno de mis mejores amigos, Sergio Lara que me ha acompañado durante toda mi vida.

A uno de mis mejor amigo Sergio Hernández, que ha sido compañero de penas en muchas ocasiones, pero siempre ha conseguido animarme en los peores momentos para acabar dando lo mejor de mí mismo.

Y en especial a mi pareja Mónica, que sin ella no habría podido conseguirlo, gracias a ella he querido ser mejor y he conseguido ser mejor, la persona que más me ha animado, la que más me ha motivado y más me ha soportado, gracias por estar a mi lado en el fin de esta etapa y por estar en las siguientes que vendrán en el futuro.

Resumen/Abstract

Resumen

En la actualidad en la Universidad de Almería, la oferta de convocatorias de becas y diferentes ayudas se encuentra dispersa en las unidades, servicios y vicerrectorados que se encargan de su gestión. Esta situación dificulta enormemente la búsqueda de información los estudiantes futuros o candidatos a estudiantes que resultan complejos encontrar la información pertinente. La difusión de las becas y ayudas para los estudiantes y para el propio gabinete de comunicaciones es compleja ya que no se dispone de una información centralizada y accesible a las convocatorias abiertas en cada momento. La publicación de las convocatorias difiere de una unidad, servicio o vicerrectorado a otro lo que dificulta aún más la búsqueda de información. Por lo que sería recomendable definir una plantilla de datos comunes para que la información sea accesible. Otro de los principales inconvenientes es que no se guarda un histórico de las convocatorias lo que implica que los alumnos si necesitan consultar convocatorias anteriores no pueden hacerlo y bien recurren a otros organismos para solicitar dicha información o bien directamente al Servicio de Becas y Ayudas. Para ello se ha propuesto desarrollar un sistema de gestión que resuelva todos estos problemas y aporte una ayuda tanto a como a la administración de los estudiantes propia universidad.

Palabras clave: Aplicación CRUD, Gestión de Difusión de Ayudas, Entorno Web.

Abstract

Currently at the University of Almería, the offer of calls for scholarships and different grants is dispersed in the units, services and vice-rectors that are in charge of their management. This situation makes it extremely difficult for future students or candidate students to find information that find it difficult to find relevant information. The dissemination of scholarships and grants for students and for the communications office itself is complex since there is no centralized information accessible to open calls at all times. The publication of the calls differs from one unit, service or vice-rector's office to another, which makes the search for information even more difficult. Therefore, it would be advisable to define a common data template so that the information is accessible. Another of the main drawbacks is that a history of the calls is not kept, which implies that students if they need to consult previous calls cannot do so and either resort to other organizations to request said information or directly to the Scholarships and Aid Service. For this, it has been proposed to develop a management system that solves all these problems and provides help both to and to the administration of the students themselves.

Keywords: CRUD Application, Aid Dissemination Management, Web Environment.

INDICE

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1 INTRODUCCIÓN

1.1 JUSTIFICACIÓN DEL PROBLEMA

En la actualidad en la Universidad de Almería la oferta de convocatorias de becas y ayudas se encuentra dispersa en las diferentes unidades, servicios y vicerrectorados que se encargan de su gestión *[12]*. Esta situación dificulta enormemente la búsqueda de información los estudiantes o futuros candidatos a estudiantes que les resulta complejo encontrar la información pertinente.
La difusión de las becas y ayudas para los estudiantes y para el propio gabinete de comunicaciones es compleja ya que no se dispone de una información centralizada y accesible a las convocatorias abiertas en cada momento.
La publicación de las convocatorias difiere de una unidad, servicio o vicerrectorado a otro lo que dificulta aún más la búsqueda de información. Por lo que sería recomendable definir una plantilla de datos comunes para que la información sea accesible.
Otro de los principales inconvenientes es que no se guarda un histórico de las convocatorias lo que implica que los alumnos si necesitan consultar convocatorias anteriores no pueden hacerlo y bien recurren a otros organismos para solicitar dicha información o bien directamente al Servicio de Becas y Ayudas.

1.1.1 Contribución

La aportación se centra en la creación de un sistema de gestión para la difusión de convocatorias de becas que permite facilitar la publicación de convocatorias, mejorar la consulta y difusión de las convocatorias abiertas. Además, ayudará a centralizar la información para facilitar la creación de un cuadro de mandos que facilite el conocimiento entre unidades, servicios y vicerrectorados sobre el estado actual de todas las convocatorias publicadas en la plataforma.

Además gracias a este sistema las empresas externas siempre con permiso de la universidad dispondrán de una herramienta para ofertas becas ellas mismas, gracias a esto podrán darle una gran oportunidad a muchos de los estudiantes y las empresas podrán formar, o ayudar a que se formen, futuros trabajadores.

1.2 OBJETIVOS DEL PROYECTO

Se pretende desarrollar una Plataforma Web corporativa para la gestión de la difusión de las convocatorias de becas y ayudas ofertadas en la Universidad de Almería. Este sistema se encargará de unificar la información relevante de las convocatorias siguiendo los requerimientos establecidos por la BNDS (Base de Datos Nacional de Subvenciones). Por ello, se creará una sección para la gestión y publicación de convocatorias estableciendo una serie de categorías que faciliten su búsqueda y mejoren la difusión de las mismas (3).

Este sistema deberá contemplar los siguientes objetivos específicos:

- Desarrollar sistema de gestión Web interno que aglutine las Becas y Ayudas que la Universidad de Almería pone a disposición de estudiantes.
- Definir e identificar la información relevante de las convocatorias propiciando una imagen corporativa y centralizando toda la información una única Plataforma Web.
- Implementar el acceso público general a todas las convocatorias a disposición de los estudiantes siguiendo criterios de accesibilidad, usabilidad y adaptabilidad para facilitar las consultas a todas las convocatorias que se ofertan en la actualidad y que se hayan ofertado.
- Destacar las convocatorias abiertas para mejorar su visibilidad y remitir un boletín informativo al gabinete de comunicaciones para optimizar su difusión.
- Integrar las opciones de becas disponibles en función del perfil del estudiante.
- Almacenar un histórico de todas las convocatorias ofertadas por año y la unidad o servicio que la gestiona.
- Implementar un proceso para recopilar y enviar la información requerida a la hora de publicar las convocatorias de subvenciones en la Base de Datos Nacional de Subvenciones.
- Crear un sistema de Gestión de Roles que permita que los responsables de cada área gestionar la información relativa a sus propias convocatorias, estableciendo controles de acceso.

Se plantea un cambio en la gestión de la publicación de convocatorias para integrar todas las becas y ayudas ofertadas en la UAL siguiendo un formato común y adaptándose a los requerimientos impuesto por BNDS. Esto repercutirá en una serie de mejoras:

- Facilitar la publicación de las convocatorias unificando los aspectos relevantes.
- Mejorar la consulta de las convocatorias abiertas y permitir la consulta de convocatorias cerradas.
- Posibilitar el acceso a aquellas convocatorias que por el perfil del estudiante pueda solicitar.
- Potenciar la difusión mediante la generación de boletines semanales con las convocatorias activas.
- Mejorar el conocimiento entre unidades, servicios y vicerrectorados sobre el estado actual de todas las convocatorias publicadas en la plataforma.
- Incorporar las convocatorias públicas en BNDS.

1.3 PLANIFICACIÓN DEL TRABAJO

El trabajo está organizado en varias secciones o capítulos en el que desarrollaremos la solución propuesta. En este primer capítulo hablaremos acerca del problema que se plantea y su posible solución, así como los distintos objetivos que se quieren lograr.

En el segundo capítulo se contextualizará el trabajo con distintas áreas de estudio relacionadas con el proyecto. Este capítulo se mostrarán otras aplicaciones similares a la que hemos desarrollado y analizaremos cada una de ellas para ver tanto sus puntos fuertes como sus puntos más débiles. Tras ese breve análisis haremos un análisis rápido de nuestro sistema para compararlo con los demás. Por último se hará una tabla comparativa con algunos de los puntos que se consideran más importantes y veremos las deficiencias y virtudes de cada uno.

En el tercer capítulo del proyecto vamos a definir las etapas del trabajo en las que se ha dividido el proyecto y ofreceremos una breve explicación de cada etapa. Además, haremos una comparativa del tiempo que se pensó inicialmente que duraría el proyecto y cada fase de él con el tiempo destinado finalmente.

En el cuarto capítulo explicaremos el análisis realizado de la solución y la manera de abarcarlo. Este capítulo estará compuesto de 4 apartados. En el primero de los apartados haremos una descripción simple de la solución. A continuación, explicaremos los casos de uso del sistema, diagramas de actividades donde explicaremos cómo funcionará el servidor con cada una de las acciones y especificación de actores que intervendrán. En los siguientes dos apartados comentamos los requisitos funcionales y no funcionales del sistema.

Para el quinto capítulo vamos a enumerar y explicar todas y cada una de las herramientas utilizadas para el desarrollo de la solución sin contar las herramientas usadas para el desarrollo de la memoria ya que se entiende que la memoria es una herramienta meramente explicativa de la solución principal. Este capítulo constará de 5 apartados donde explicaremos primero de todos los lenguajes usados para el desarrollo. En el segundo apartado comentaremos los entornos de desarrollo, es decir, el programa desde el cual se ha desarrollado el proyecto. En el tercer apartado comentaremos las herramientas que hacen referencia a toda la referencia y los recursos utilizados para el desarrollo del proyecto y su correcta solución. Más tarde hablaremos de la infraestructura, es decir, los medios necesarios para el desarrollo de la solución. Por último, hablaremos de la estructura MEBN Stack y cuáles son sus componentes.

En el sexto capítulo vamos a explicar el desarrollo del proyecto. En este capítulo vamos a explicar la arquitectura del sistema paso a paso a través de la explicación del Backend, después explicaremos sin entrar demasiado en profundidad el proceso de implementación para la página web. Después explicaremos el desarrollo de las interfaces y formularios que se mostrarán de cara a los usuarios.

Finalmente, en el capítulo séptimo valoraremos los resultados que se han obtenido tras completar el proyecto. En este capítulo vamos a comentar los resultados, detallar lo que se ha aprendido durante el desarrollo total del proyecto y por último valoraremos posibles líneas de trabajo con las que se podría continuar desarrollando y mejorando el proyecto.

2 ESTADO DEL ARTE

En este capítulo vamos a hablar sobre la existencia de ciertos sistemas existentes para otras universidades para poder mostrar las semejanzas y diferencias que tienen con el proyecto que vamos a realizar

Vamos a describirlas para mostrar cómo están desarrolladas otras plataformas semejantes con las distintas posibilidades que se encuentran actualmente y haremos una comparativa de las herramientas con el proyecto que vamos a desarrollar.

2.1 OTRAS APLICACIONES SIMILARES

En este apartado vamos a mostrar algunas aplicaciones web similares y hablaremos acerca de las similitudes, diferencias e inspiraciones que han aportado estos sitios web. Hay muchas aplicaciones similares, ya que como es lógico cada universidad tiene su propio apartado de becas.

En primer lugar, vamos a nombrar la página de becas de la universidad de Jaén *[9]* (*Ilustración1)*

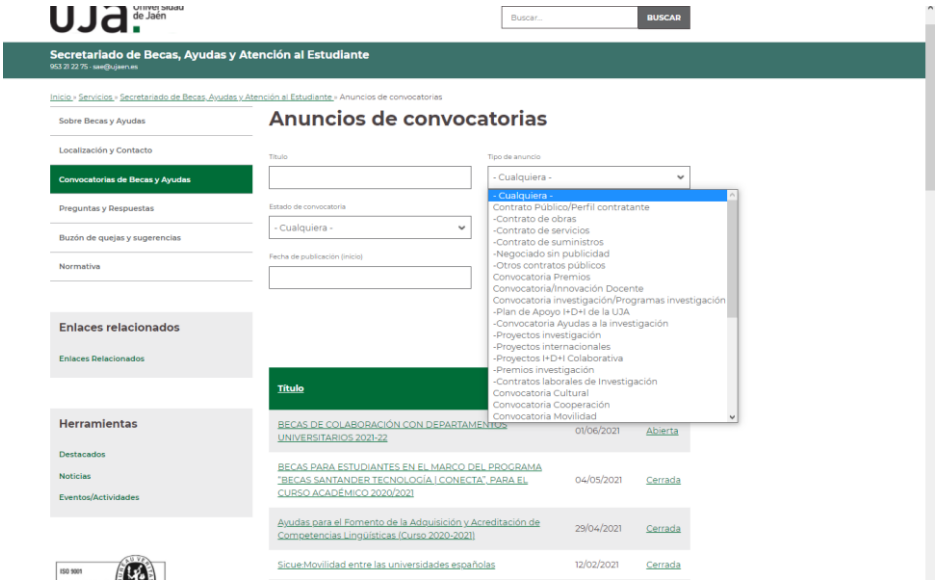


Ilustración 1 Página becas Universidad de Jaén.

Como se puede observar esta aplicación web, aunque dispone de un buscador, resulta un poco confuso ya que aglutina muchos campos haciendo pesada la búsqueda, por el contrario si queremos buscarla mediante la tabla que propone nos dispone de muy poca información y está demasiado resumida, dejando solo el título, fecha de publicación y el estado de la convocatoria, que aunque son datos muy importantes existen ciertos datos que también interesan a los estudiantes como puede ser el número de convocatorias o la cuantía económica.

Otra de las aplicaciones que me han servido de inspiración ha sido la de la universidad de Cádiz *[11]* (*Ilustración 2)*.

Ilustración 2 Página de becas Universidad de Cádiz

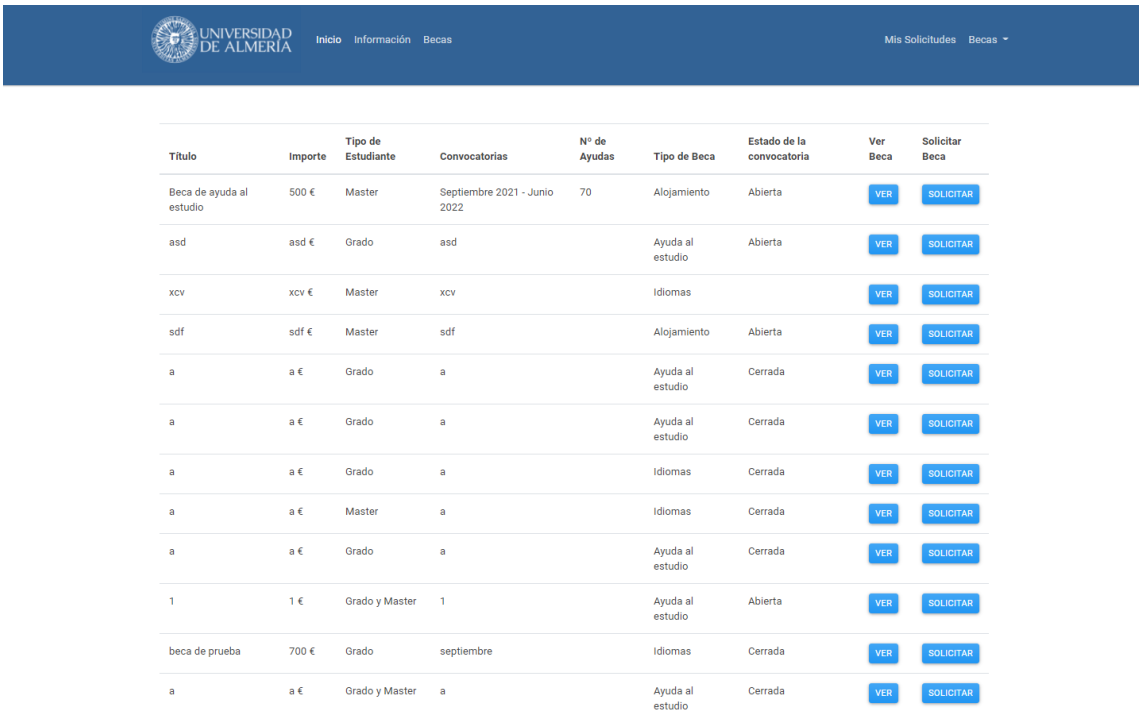
Este sitio web como se puede observar tiene una especie de cuadro donde acceder a cada uno de los apartados, esto no es demasiado práctico ya que hay que navegar entre demasiados menús para poder hacer cualquier gestión lo que puede resultar tedioso a la hora de ver convocatorias o solicitar becas. La parte buena de esta página es que directamente tienes un apartado para solicitar las becas y puedes ver las resoluciones directamente.

Siguiendo con los análisis vamos a explicar la tercera base de la que hemos tomado ciertas ideas para la realización de nuestra aplicación. En este caso será la página de becas de la universidad de Córdoba *[10]* (*Ilustración 3)*.

Ilustración 3 Página de becas Universidad de Córdoba

Como se puede observar, esta sería la interfaz visual más cercana a la nuestra, ya que se compone de una tabla mostrando los valores más significativos de cada convocatoria, sin embargo, en primer lugar, la información aparece en un espacio demasiado reducido, por lo que no se muestra totalmente como observamos en la imagen mostrada lo que obliga al alumno a tener que desplazarse y perdiendo de vista de información relevante, además, no nos permite solicitar las becas desde la misma aplicación web, si no que deberemos descargar un formulario para rellenarlo y volver a subirlo.

Finalmente llegamos a nuestra propia solución (*Ilustración 4)*, la cual analizaremos para poder más tarde compararlo con el resto de sistemas referenciados.



Título	Importe	Tipo de Estudiante	Convocatorias	Nº de Ayudas	Tipo de Beca	Estado de la convocatoria	Ver Beca	Solicitar Beca
Beca de ayuda al estudio	500 €	Master	Septiembre 2021 - Junio 2022	70	Alojamiento	Abierta	VER	SOLICITAR
asd	asd €	Grado	asd		Ayuda al estudio	Abierta	VER	SOLICITAR
xcv	xcv €	Master	xcv		Idiomas		VER	SOLICITAR
sdf	sdf €	Master	sdf		Alojamiento	Abierta	VER	SOLICITAR
a	a €	Grado	a		Ayuda al estudio	Cerrada	VER	SOLICITAR
a	a €	Grado	a		Ayuda al estudio	Cerrada	VER	SOLICITAR
a	a €	Grado	a		Idiomas	Cerrada	VER	SOLICITAR
a	a €	Master	a		Idiomas	Cerrada	VER	SOLICITAR
a	a €	Grado	a		Ayuda al estudio	Cerrada	VER	SOLICITAR
1	1 €	Grado y Master	1		Ayuda al estudio	Abierta	VER	SOLICITAR
beca de prueba	700 €	Grado	septiembre		Idiomas	Cerrada	VER	SOLICITAR
a	a €	Grado y Master	a		Ayuda al estudio	Cerrada	VER	SOLICITAR

Ilustración 4 Página de becas Proyecto fin de grado

Observamos que en este caso la solución se presenta con una interfaz sencilla donde destaca la información que es lo realmente relevante en este tipo de sistemas. Además, dispone de un sistema de solicitud donde podremos rellenar directamente el formulario para solicitarla. Por el contrario, el único problema que podría estar presente es la falta de un buscador, lo cual ayudaría mucho a los estudiantes agilizando las consultas de las becas.

Tras analizar todos estos casos podemos y compararlo con el sistema que se ha desarrollado como solución al problema podemos observar que se han resuelto muchos de los fallos presentes en los casos anteriores tales como problemas de interfaz, funcionamiento o añadiendo soluciones y funcionalidades como la capacidad de realizar solicitudes desde el propio sistema. Además de tener implementado un sistema de roles lo que permite que exista una interfaz única y centralizada, pero limitando las acciones a personas ajenas al sistema. Es por esto que a través de varias ideas se ha realizado una solución óptima que tiene lo mejor de cada caso creando un sistema novedoso.

A continuación, me dispongo a presentar la comparativa donde creo que se resumen los campos más importantes que debe tener un sistema de estas dimensiones para mostrar

cual de todas cumple con los requisitos que creo que son más necesarios. Esta comparativa dispondrá de los siguientes campos (*Tabla 1)*:

- **Información relevante:** Se valorará si la información presente en la aplicación web es la necesaria
- **Interfaz clara:** Se valorará si la interfaz es clara y no distrae al usuario de los campos realmente importantes
- **Buscador:** Se valorará si dispone de un buscador
- **Usabilidad:** Se valorará si la aplicación web es sencilla de usar y de fácil aprendizaje
- **Utilidad:** Se valorará las utilidades de las que disponga la aplicación
- **Responsive:** Se valorará que el sistema sea responsive

La valoración se hará en 3 niveles, donde el signo ✓ indicará que es un campo totalmente cumplido por la aplicación, el signo ✗ indicará que este campo no se cumple en esta aplicación web y el signo ! mostrará si cumple dicho requisito, pero no de la mejor forma posible

Aplicación web	Información relevante	Interfaz clara	Buscador	Usabilidad	Utilidad	Responsive
Universidad de Córdoba	!	✓	✗	✓	✓	✗
Universidad de Cádiz	✗	✗	✗	✓	✓	✗
Universidad de Jaén	✗	!	✓	!	✗	✓
Proyecto fin de grado	✓	✓	✗	✓	✓	✓

Tabla 1 Tabla comparativa de requisitos entre aplicaciones web

3 ETAPAS DE DESARROLLO

El desarrollo del proyecto se divide en siete etapas generales:

1. **Recopilación inicial de Información**: se realizará un estudio para recopilar la información necesaria en el proyecto y el estado del arte.

2. **Análisis del Problema**: se determinará el problema a resolver y la metodología a seguir para el diseño de la aplicación.

3. **Estudio de la tecnología necesaria:** Se realizará un estudio sobre las diferentes tecnologías y recursos necesarios para la realización del sistema.

4. **Implementación del software**: Se realizará la implementación del software a partir de la información recogida mediante fases previas.

5. **Corrección de errores**: Se realizará una revisión de la aplicación para eliminar posibles errores.

6. **Evaluación de los resultados obtenidos**: se estudiarán los resultados obtenidos para corroborar que se han obtenido de manera correcta.

7. **Redacción Memoria**: se realizará la redacción final de la memoria para su posterior presentación.

Cada una de las etapas ha sido analizada y se le ha asignado un tiempo estimado de realización dando como resultado un total de 300 horas. A continuación, mostramos un organigrama de tiempos del proyecto (*Tabla 2)*:

Fases	Enero	Febrero	Marzo	Abril	Mayo	Junio
Recopilación inicial de la información y estado del arte	55 horas					
Análisis del Problema		50 horas				
Estudio de la tecnología necesaria			30 horas			
Implementación del sistema				55 horas		
Corrección de errores					25 horas	
Evaluación de los resultados obtenidos						30 horas
Redacción de Memoria						55 horas

Tabla 2 Distribución de tiempos de las etapas

El tiempo invertido real de horas para completar el proyecto ha sido bastante más sustancial del esperado que la versión presentada inicialmente. Aunque finalmente se ha llegado a concluir el proyecto en la convocatoria elegida ha sido necesario más tiempo. Finalmente han sido necesarias 405 horas en comparación con el plan inicial. Estas horas han aumentado en el apartado del estudio de la tecnología necesaria, donde han sido necesarias

10 horas más de las esperadas por varios problemas con las ideas iniciales y que hubo que cambiar sobre la marcha, llegando a ser un total de 40 horas. 65 horas más en la implementación, esto hace un total de 120 horas para la implementación. En el proceso de corrección de errores se tardaron 40 horas finalmente, 15 horas más de las esperadas ya que hubo que solucionar problemas con el sistema de roles implementado tras el cambio de inicio de sesión, así como problemas de compatibilidad con algunos de los módulos instalados y la base de datos. El último retraso se ha producido en la redacción de la memoria ya que han sido necesarias 70 horas.

La principal causa del retraso fue la imposibilidad de cumplir con el horario establecido debido a mi trabajo y los turnos rotativos lo que ha complicado mucho la dedicación del tiempo necesario. En cuanto a la implementación a la hora de estudiar los métodos más eficaces de servicios de inicio de sesión se observó que el escogido (JsonWebTokens) no era lo suficientemente seguro, por lo que hubo que modificar todo el proceso de inicio de sesión nuevamente para aumentar la seguridad del proyecto. Finalmente, el organigrama de tiempos ha quedado de la siguiente manera (*Tabla 3):*

Fases	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio
Recopilación inicial de la información y estado del arte	55 horas						
Análisis del Problema		50 horas					
Estudio de la tecnología necesaria			40 horas				
Implementación del sistema				120 horas			
Corrección de errores					40 horas		
Evaluación de los resultados obtenidos						30 horas	
Redacción de Memoria						70 horas	

Tabla 3 Distribución de tiempos finales de las etapas

4 ANÁLISIS

Para realizar un análisis correcto necesitaremos destacar todas las funcionalidades del sistema, teniendo en cuenta todos y cada uno de los actores que intervendrán. Se describirá de manera exhaustiva cada uno de las funcionalidades que tiene cada actor.

A través de un buen análisis conseguiremos que el lector pueda entender de forma fácil y sencilla como se ha orientado el proyecto para darle solución a los problemas que se han planteado

4.1 DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN

Se implementará un sistema de difusión de ayudas y becas a través de un portal web donde se centralizarán todas las convocatorias de becas y a través del cual se podrán tanto realizar acciones con las becas desde añadir, editarlas o eliminarlas convocatorias, hasta solicitar becas desde su cuenta de usuario.

Con este objetivo se diseñará una aplicación web que a través de un sistema de roles permita distintas funciones según el tipo de cuentas de cada usuario generando a través de una misma interfaz general distintas vistas y opciones.

Se pretende desarrollar una Apirest *[14]* en el Backend que controle todo el servidor internamente sin necesitar ayuda de programas externos o de terceros a través de un framework de JavaScript para toda la funcionalidad

En el apartado visual se plantea desarrollar una interfaz que resulte simple a la vista pero que disponga de todas las opciones necesarias para poder realizar todas las acciones que se quieran sin llenar de información excesiva e innecesaria al usuario.

Con esta solución no planteamos desarrollar un servicio demasiado amplio o robusto, si no un sistema con una buena base y mucho potencial que disponga de una gran progresión futura en función de las necesidades que vayan surgiendo a lo largo del paso del tiempo y las nuevas herramientas que vayan surgiendo.

4.2 CASOS DE USO DEL SISTEMA

Se realizará una descripción detallada de las funcionalidades, teniendo en cuenta el diagrama de casos de uso en el que vamos a nombrar las acciones que realizarán cada uno de los actores que intervienen. Con la descripción se conseguirá que el lector comprenda fácilmente en que consiste y como se ha orientado la solución a los problemas que se han planteado.

Con el diagrama de casos de uso mostraremos todas las funcionalidades que tiene que realizar cada actor teniendo en cuenta a simple vista que roles tendrá cada actor

4.2.1 Diagrama de casos de uso

En el diagrama de casos de uso (*Ilustración 6)*, la funcionalidad del sistema que se presenta estará representado desde la perspectiva del usuario ("actor" o "rol" en UML). El actor en cuestión no tiene que ser una persona, también puede representar un sistema externo que accede al sistema. De esta manera el diagrama muestra la relación entre los distintos actores que participan y sus necesidades sin mostrar las acciones que llevan a cabo.

A continuación, se muestran los actores que intervendrán, uno será el administrador, el cual dispondrá de todas las funciones posibles, otro será el moderador, el cual heredará algunas funciones del administrador y otro será el estudiante, el cual solo podrá visualizar ciertas funciones.

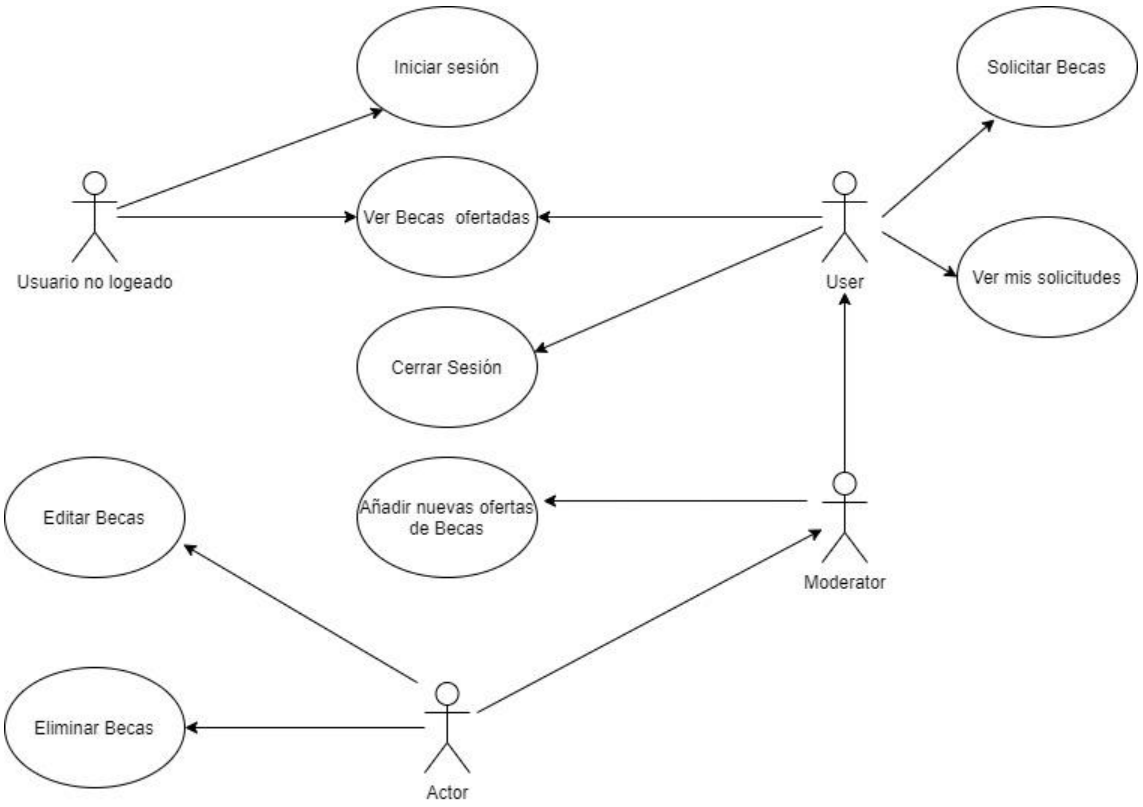


Ilustración 5 Diagrama de casos de uso

Gracias a la correcta descripción de los casos de uso, será más sencilla la creación e implantación de cada una de las funcionalidades de los distintos actores.

4.2.2 Diagrama de actividades

En este momento vamos a mostrar los diagramas de actividades, este diagrama muestra el flujo de las actividades del sistema.

A continuación, se presenta el diagrama de actividades para el Registro (*Ilustración 6):*

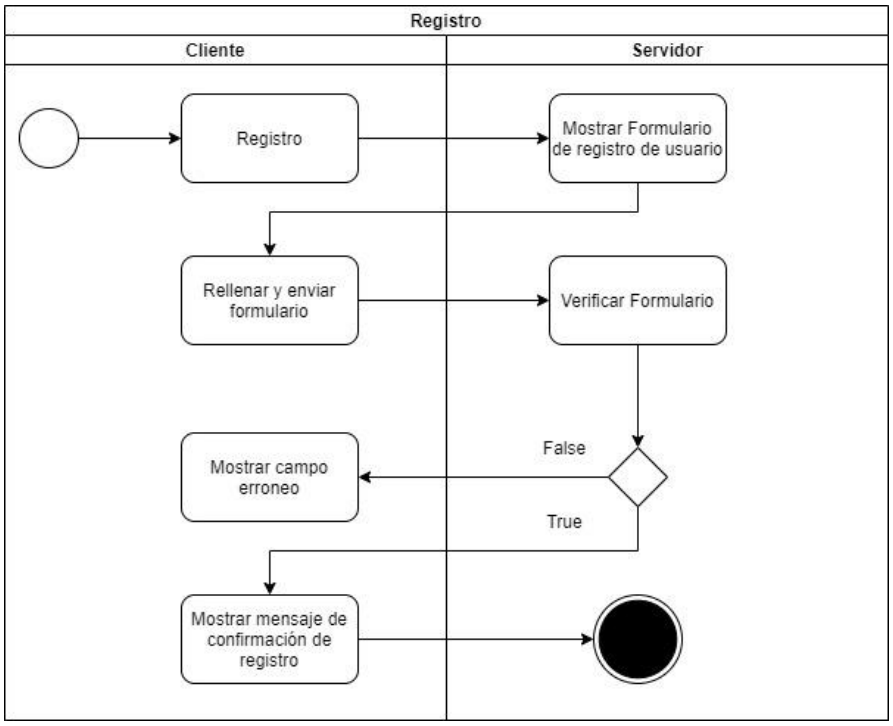


Ilustración 6 Diagrama de Actividades: Registro

A continuación, presento el diagrama de actividades para el Inicio de Sesión (*Ilustración 7):*

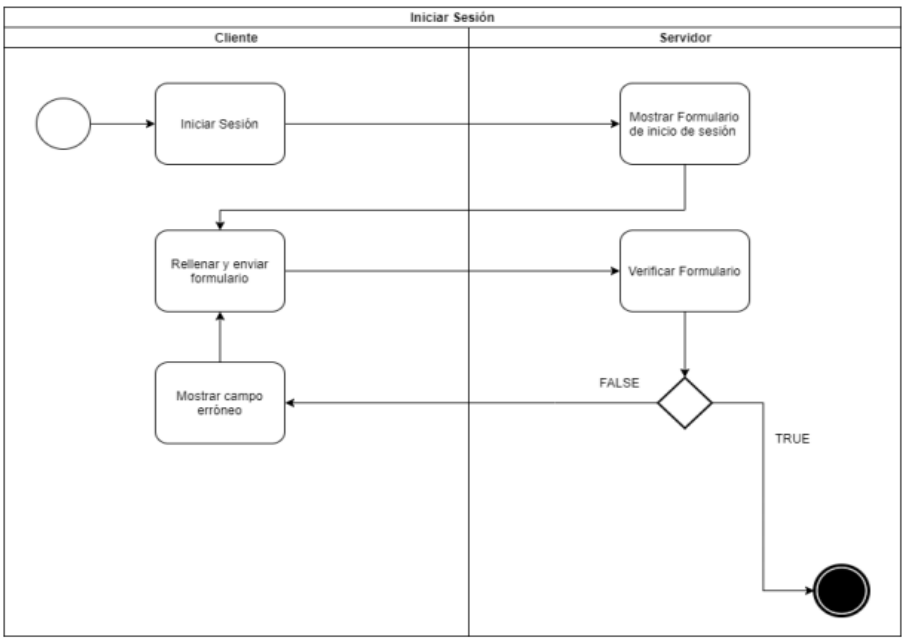


Ilustración 7 Diagrama de Actividades: Inicio de sesión

A continuación, se muestra el diagrama de actividades para el Cierre de Sesión (*Ilustración 8):*

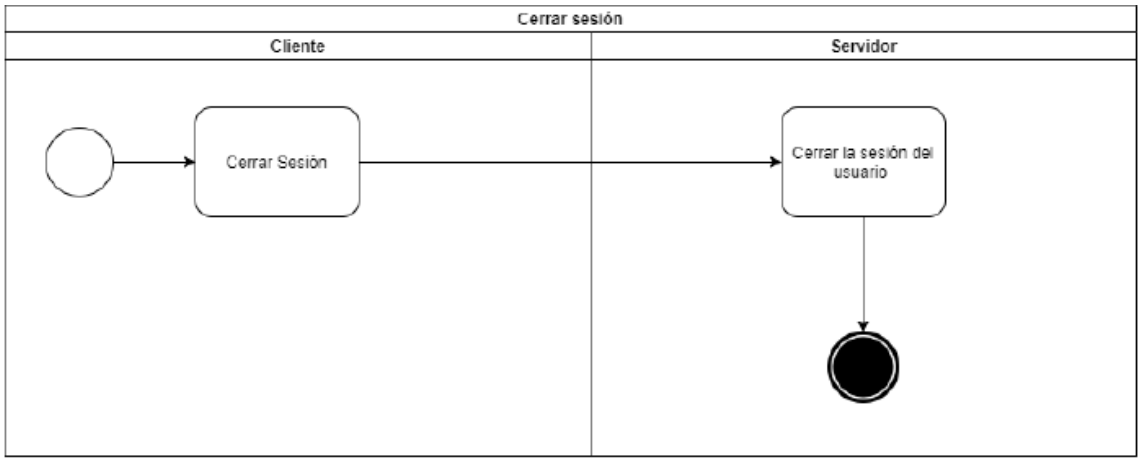


Ilustración 8 Diagrama de Actividades: Cerrar Sesión

A continuación, se presenta el diagrama de actividades para Añadir las Becas (*Ilustración 9):*

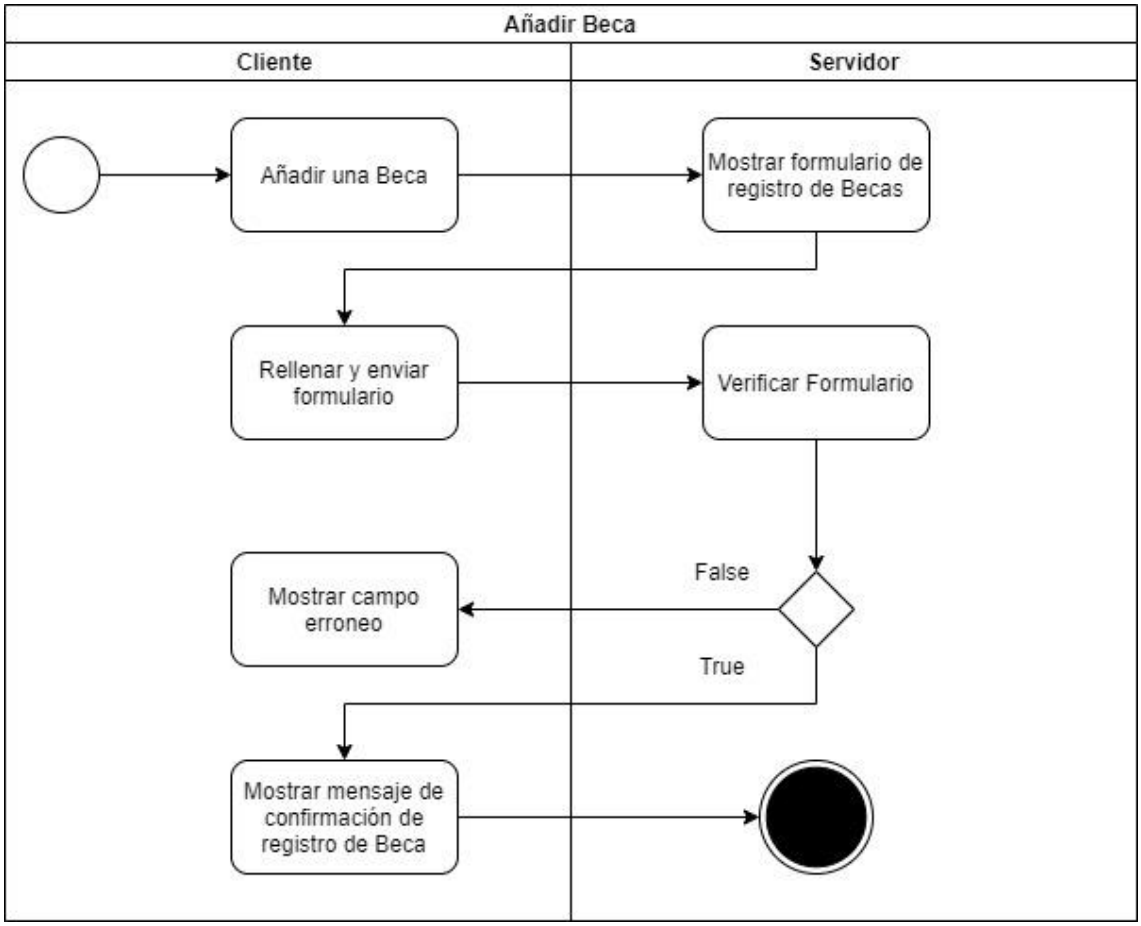


Ilustración 9 Diagrama de Actividades: Añadir una Beca

A continuación, se presenta el diagrama de actividades para la edición de Becas (*Ilustración 10):*

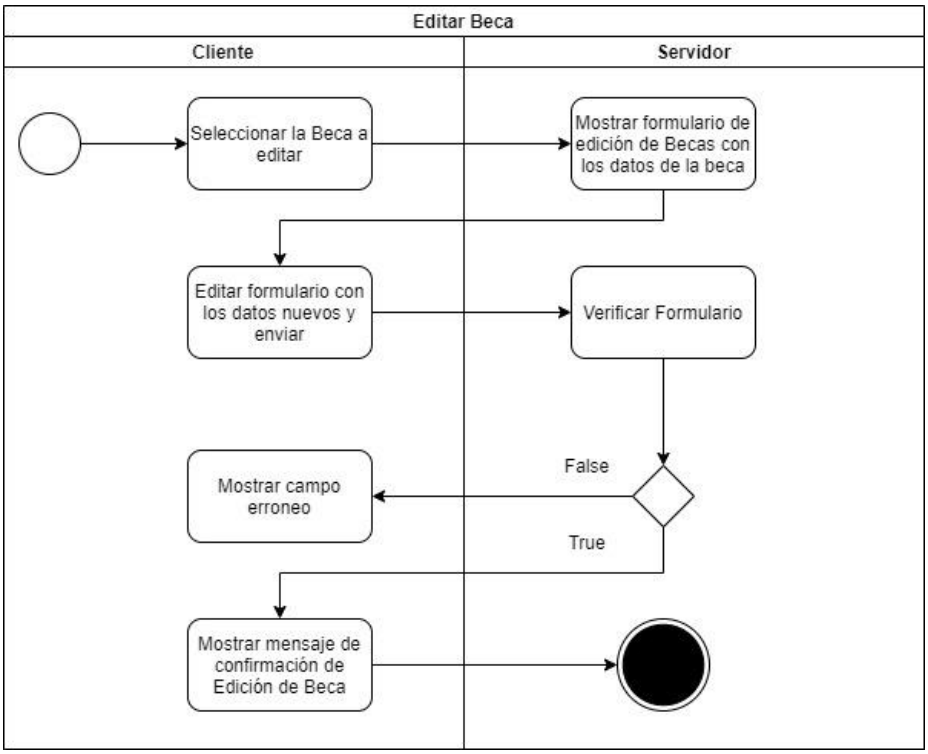


Ilustración 10 Diagrama de Actividades: Edición de una Beca

A continuación, se presenta el diagrama de actividades para la Eliminación de Becas(*Ilustración 11):*

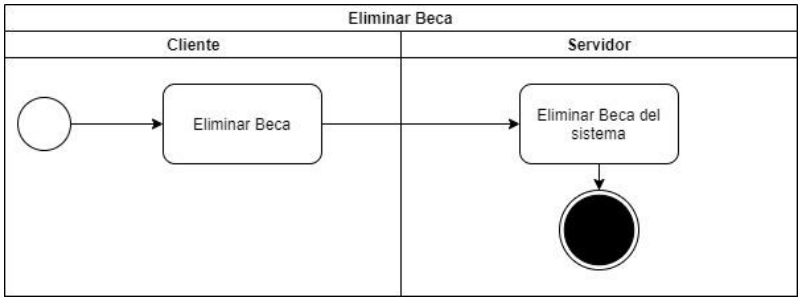


Ilustración 11 Diagrama de Actividades: Eliminar una Beca

A continuación, se presenta el diagrama de actividades para Mostrar las Becas (*Ilustración 12):*

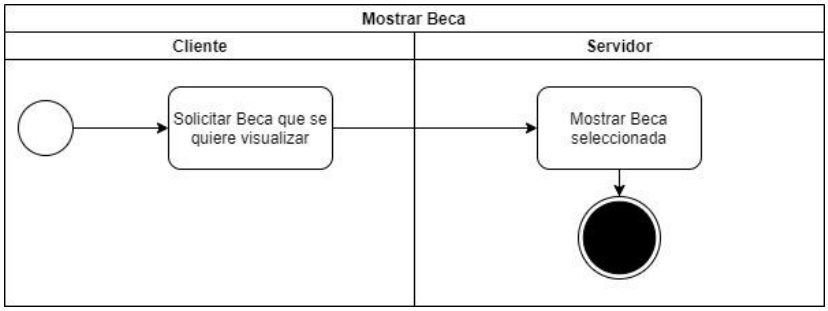


Ilustración 12 Diagrama de Actividades: Mostrar Beca

A continuación, se presenta el diagrama de actividades para el Solicitar Becas (*Ilustración 13):*

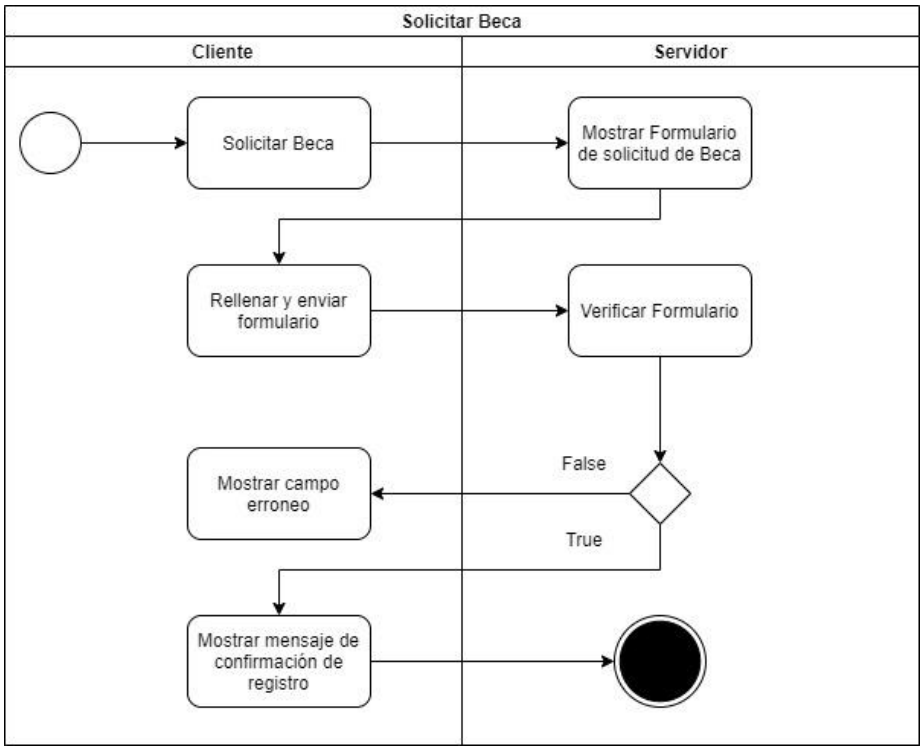


Ilustración 13 Diagrama de Actividades: Solicitar Beca

A continuación, se presenta el diagrama de actividades para Mostrar las solicitudes de Becas (*Ilustración 14):*

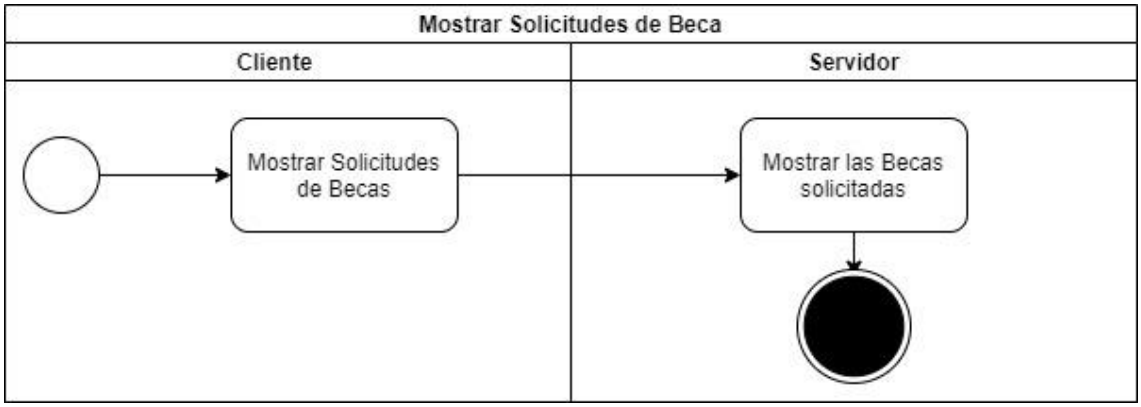


Ilustración 14 Diagrama de Actividades: Mostrar solicitudes de Becas

4.2.3 Descripción de los actores

A continuación, procedemos a describir los actores del sistema que van a actuar (A) *(Tablas 4 – 7)*:

A001	Usuario no logeado
Requisitos asociados	RF-1, RF-2, RF-6
Descripción	Formado por todos los visitantes del sistema
Comentarios	Ninguno

Tabla 4 Descripción de Actores: User no logeado

A002	User
Requisitos asociados	RF-3, RF-4, RF-5, RF-6
Descripción	Formado por todos los usuarios registrados del sistema
Comentarios	Nada

Tabla 5 Descripción de Actores: User

A003	Moderador
Requisitos asociados	RF-3, RF4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	Formado por los usuarios con ciertos privilegios en el sistema
Comentarios	Ninguno

Tabla 6 Descripción de Actores: Moderador

A004	Administrador
Requisitos asociados	RF-3, RF4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	Formado por los responsables del sistema
Comentarios	Ninguno

Tabla 7 Descripción de Actores: Administrador

4.3 REQUISITOS FUNCIONALES

En los diagramas previos se ha mostrado una presentación acerca del planteamiento del sistema de forma general, a continuación, vamos a explicar de forma más detallada el planteamiento:

Como se ha podido observar existen 4 actores que intervendrán en el sistema:

- A001 – Usuario no logeado: Este es un usuario sin identificarse en nuestro sistema. Para registrarse o identificarse tendrá que ir a la pestaña correspondiente y mostrar sus credenciales con sus datos.
- A002 – User: Este es un usuario que se ha registrado y ha iniciado sesión previamente, su información está guardada en el sistema y tiene acceso a ciertas funciones tales como ver las becas ofertadas, solicitar las mismas o ver un registro de sus solicitudes
- A003 – Moderador: Al igual que el "A002" este es un usuario previamente registrado y que ha iniciado sesión, pero a diferencia del anterior caso dispone de más permisos, ya que este usuario podrá ofertar Becas desde el propio sistema, este actor está diseñado para agentes externos que quieran ofertar becas a estudiantes desde la misma plataforma de la UAL.
- A004 – Administrador: Este usuario será el que tenga los permisos totales del sistema, ya que podrá realizar las mismas tareas que los dos actores anteriores ("A002" y "A003") pero además podrá cumplir con otras facetas de administrador, ya que podrá eliminar y editar las propias becas que se encuentren ofertadas.

Partiendo del diagrama de casos de uso se pueden establecer los siguientes requisitos funcionales que se han presentado brevemente a través de los distintos diagramas de actividades que hemos mostrado *(Tablas 8-16)*:

Para facilitar su entendimiento y comprensión se ha decidido presentarlos a través de un modelo de tablas donde los campos serán los siguientes:

- **Nombre:** Nombre del requisito
- **Descripción:** Descripción del requisito
- **Actores:** Actores implicados
- **Sistema:** A qué parte del sistema corresponde
- **Requisitos funcionales asociados:** Requisitos funcionales que se deben cumplir para que este se pueda realizar
- **Datos Específicos:** Datos necesarios
- **Comentarios:** Especificaciones más concretas

RF-1	
Propiedad	Valor
Nombre	Registro
Descripción	El usuario registra sus datos en el sistema
Actores	A001
Sistema	Aplicación Web
Requisitos asociados	
Datos específicos	Nombre, email y contraseña
Comentarios	El email no debe estar registrado en el sistema previamente

Tabla 8 Requisitos funcionales: Registro

RF-2	
Propiedad	Valor
Nombre	Inicio de sesión
Descripción	El usuario Inicia sesión en el sistema
Actores	A001
Sistema	Aplicación Web
Requisitos asociados	RF-1
Datos específicos	Email y contraseña
Comentarios	El usuario debe estar registrado en el sistema previamente

Tabla 9 Requisitos funcionales: Inicio de sesión

RF-3	
Propiedad	Valor
Nombre	Cerrar Sesión
Descripción	El usuario cierra sesión en el sistema
Actores	A002, A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-1 Y RF-2
Datos específicos	
Comentarios	Se cerrará la sesión en la navegación Web.

Tabla 10 Requisitos funcionales: Cerrar Sesión

RF-4	
Propiedad	Valor
Nombre	Solicitar Becas
Descripción	El usuario solicitará una de las Becas ofertadas
Actores	A002, A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-2
Datos específicos	Nombre, Tipo de Alumno, DNI, Estudios matriculados y Curso Académico
Comentarios	El usuario podrá solicitar una de las Becas ofertadas una vez inicie sesión.

Tabla 11 Requisitos funcionales: Solicitar Becas

RF-5	
Propiedad	Valor
Nombre	Mostrar solicitudes de Becas
Descripción	El usuario verá un listado con sus solicitudes
Actores	A002, A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-2, RF-4
Datos específicos	
Comentarios	El usuario podrá visualizar un listado con las solicitudes que haya mandado previamente.

Tabla 12 Requisitos Funcionales: Mostrar solicitudes de Becas

RF-6	
Propiedad	Valor
Nombre	Mostrar Becas
Descripción	El usuario verá un listado con las Becas ofertadas
Actores	A001, A002, A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-7
Datos específicos	
Comentarios	El usuario podrá visualizar un listado con las Becas ofertadas.

Tabla 13 Requisitos funcionales: Mostrar Becas

RF-7	
Propiedad	Valor
Nombre	Añadir Becas
Descripción	El usuario añadirá ofertas de Becas al sistema
Actores	A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-2
Datos específicos	Título, Encargado de la Beca, Requisitos para solicitarla, Documentación requerida, Fecha de Resolución, Importe asignado por beca, Tipo de estudiante al que va dirigida la beca, Convocatoria, N.º de Ayudas ofertadas, Tipo de Beca, Descripción de la Beca, Estado de la convocatoria, Fecha de inicio de la publicación, fecha de fin de la publicación, Presupuesto total, Tipo de financiación.
Comentarios	El usuario podrá registrar ofertas de becas en el sistema.

Tabla 14 Requisitos funcionales: Añadir Becas

RF-8	
Propiedad	Valor
Nombre	Editar Becas
Descripción	El usuario editará las ofertas de Becas del sistema
Actores	A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-2, RF-6
Datos específicos	Título, Encargado de la Beca, Requisitos para solicitarla, Documentación requerida, Fecha de Resolución, Importe asignado por beca, Tipo de estudiante al que va dirigida la beca, Convocatoria, N.º de Ayudas ofertadas, Tipo de Beca, Descripción de la Beca, Estado de la convocatoria, Fecha de inicio de la publicación, fecha de fin de la publicación, Presupuesto total, Tipo de financiación.
Comentarios	El usuario podrá editar las ofertas de becas que ya estén dentro del sistema.

Tabla 15 Requisitos funcionales: Editar Becas

RF-9	
Propiedad	Valor
Nombre	Eliminar Becas
Descripción	El usuario eliminará las ofertas de Becas del sistema
Actores	A003, A004
Sistema	Aplicación Web
Requisitos asociados	RF-2, RF-6
Datos específicos	
Comentarios	El usuario podrá eliminar ofertas de Becas que estuviesen previamente registradas

Tabla 16 Requisitos funcionales: Eliminar Becas

4.4 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales son expuestos a continuación *(Tablas 17-26)*:

Para facilitar su entendimiento y comprensión se ha decidido presentarlos a través de un modelo de tables donde los campos serán los siguientes:

- **Nombre:** Nombre del requisito
- **Requisitos funcionales asociados:** Requisitos funcionales que se deben cumplir para que este se pueda realizar
- **Descripción:** Breve descripción del requisito
- **Comentarios:** Especificaciones más concretas.

RNF-1	
Propiedad	Valor
Nombre	Portabilidad
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	La aplicación debe funcionar tanto en dispositivos móviles como web
Comentarios	El diseño será responsive, lo cual permitirá que escale a tamaños de dispositivo móvil, tablets y Pc´s

Tabla 17 Requisito no funcionales: Portabilidad

RNF-2	
Propiedad	Valor
Nombre	Tiempo de respuesta
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	El tiempo de respuesta debe ser el mínimo para que los usuarios no deban esperar demasiado
Comentarios	Se intentará reducir los tiempos de respuesta para evitar largas esperas. No se contemplarán los problemas por falta de conexión de red.

Tabla 18 Requisitos no funcionales: Tiempo de respuesta

RNF-3	
Propiedad	Valor
Nombre	Interfaz
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	Las interfaces gráficas deben estar bien formadas
Comentarios	Las interfaces deberán ser claras, sencillas y bien formadas

Tabla 19 Requisitos no funcionales: Interfaz

RNF-4	
Propiedad	Valor
Nombre	Usabilidad
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	El tiempo de aprendizaje debe ser reducido
Comentarios	El usuario debe poder adaptarse fácilmente al sistema con la ayuda de una interfaz sencilla y cómoda

Tabla 20 Requisitos no funcionales: Usabilidad

RNF-5	
Propiedad	Valor
Nombre	Seguridad
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	El sistema debe guardar los datos de sus usuarios de forma segura
Comentarios	El sistema hace uso de un módulo de encriptación de contraseñas

Tabla 21 Requisitos no funcionales: Seguridad

RNF-6	
Propiedad	Valor
Nombre	Robustez
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	La base de datos debe estar bien estructurada y contener la información de forma organizada
Comentarios	No se permitirán datos duplicados o inservibles, se guardarán las fechas de registro de usuarios

Tabla 22 Requisitos no funcionales: Robustez

RNF-7	
Propiedad	Valor
Nombre	Escalabilidad
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	Posibilidad de incluir nuevas funcionalidades y contenidos al sistema de forma rápida y sencilla
Comentarios	El uso de Stack MEBN permite incluir nuevas colecciones a la base de datos, añadir nuevas consultas en la API y cambiar el Frontend con facilidad

Tabla 23 Requisitos no funcionales: Escalabilidad

RNF-8	
Propiedad	Valor
Nombre	Mensajes de error
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	El sistema mandará mensajes de error al usuario
Comentarios	Durante cualquier proceso de registro, inicio de sesión, añadir, editar o eliminar el sistema podrá mandar mensajes de error al usuario en caso de que algo no salga correctamente

Tabla 24 Requisitos no funcionales: Mensajes de error

RNF-9	
Propiedad	Valor
Nombre	Mensajes de éxito
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	El sistema mandará mensajes de éxito al usuario
Comentarios	Durante cualquier proceso de registro, inicio de sesión, añadir, editar o eliminar el sistema podrá mandar mensajes de éxito al usuario en caso de que las funciones salgan correctamente

Tabla 25 Requisitos no funcionales: Mensajes de éxito

RNF-10	
Propiedad	Valor
Nombre	Formularios
Requisitos asociados	RF-1, RF-2, RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9
Descripción	Los formularios serán eficientes
Comentarios	No se pedirá información irrelevante al usuario en cualquiera de los formularios presentes en el sistema

Tabla 26 Requisitos no funcionales: Formularios

5 TECNOLOGÍAS Y HERRAMIENTAS

En este apartado nos disponemos a explicar las distintas herramientas utilizadas durante el desarrollo del proyecto

5.1 LENGUAJES USADOS:

5.1.1 HTML5:

Se podría definir como un estándar que establece una estructura y contenido de una página web y es la última versión de la tecnología HTML *[7] (Ilustración 15)*, cuyas siglas son "Hyper Text Markup Language"

- **Hyper Text:** Texto que enlaza con otros contenidos
- **Markup (etiqueta):** Toda página web está diseñada a base de etiquetas, por ejemplo, sería "***<h1> Título <h1>"***
- **Language:** HTML es un lenguaje con normas y estructuras predefinidas que se cumplirán durante el desarrollo de nuestra web y su contenido.

El estándar HTML se encuentra ahora mismo en su versión HTML5

Este no es un lenguaje de programación, ya que no tiene funciones aritméticas, variables o estructuras de control, esto significa que HTML solo genera páginas web estáticas, pero permite que se usen varios lenguajes en conjunto lo que provoca que estas páginas estáticas se conviertan en dinámicas.

Este lenguaje se usa para describir estructuras básicas de una página y organizarla de forma en la que se muestra su contenido. Este es un lenguaje descriptivo, es decir, que se desarrolla a través de etiquetas que definen la estructura y contenido de la web

Ilustración 15 Logo HTML5

5.1.2 CSS

CSS *[6] (Ilustración 16)* es un mecanismo que sirve como complemento del leguaje HTML el cual permite indicarle una serie de estilos que le debe dar al desplegar la información del sitio Web.

Al contrario que los comandos HTML le indican al navegador que partes del texto son títulos, párrafos o enlaces CSS le indica entre otras cosas las fuentes, tamaños, colores… que debe tener una página

CSS son las siglas de Cascade Style Sheets (Hoja de estilos en cascada). Desde el uso de HTML las hojas de estilo.

Este mecanismo sirve para aplicar estilos a cada plantilla HTML. Cuando vinculamos una página HTML a una hoja de estilos CSS tendremos una serie de comandos para modificar los elementos siempre que cumplan una serie de condiciones, es decir, es más fácil vincular una plantilla HTML a un archivo CSS para que el navegador entienda lo que deben mostrar según nuestras necesidades que integrarlo por separado en cada uno de los elementos de la plantilla HTML.

Otra de sus muchas ventajas es que nos permite usar la misma plantilla para formatear varios sitios webs sin necesitar hacer cambios, simplemente deberemos modificar los elementos de la plantilla y actualizar los sitios web que se encuentran vinculados.

CSS no solo se limita al texto. Si se combina con JavaScript podemos aplicar varias funciones tales como cambiar el color de un hipervínculo para que los usuarios sepan que ya han visitado dicho lugar o diseñar menús desplegables más interactivos.

Ilustración 16 Logo CSS

5.1.3 JavaScript

JavaScript *[8] (Ilustración 17)* es un lenguaje de programación que ofrece más capacidades interactivas y dinámicas a un sitio Web.

No requiere compilación cuando se ejecuta ya que el navegador puede leer el código directamente sin necesitar servicios de terceros por lo que se considera uno de los lenguajes de los servicios Web junto con HTML y CSS.

Este lenguaje se usa en el lado del servidor para crear y nos permite crear efectos y animaciones sin tener que tener interacción ni responder a eventos inducidos por el usuario tales como accionar botones.

Por lo que podemos ver no tiene nada que ver con el lenguaje de programación Java, más allá del nombre, ya que su función consiste en la ayuda para crear sitios web dinámicos.

El código de programación JavaScript se ejecuta en el navegador del ordenador o de dispositivos iOS y Android.

Es capaz de detectar errores en formularios, diseñar sliders adaptables a cualquier pantalla, realizar cálculos matemáticos o modificar elementos del sitio web.

JavaScript es actualmente uno de los idiomas más populares. Es por ello que se han creado versiones que se pueden ejecutar del lado del servidor como es Node.js. Como resultado, ahora es ejecuta en navegadores y servidores y se ha creado una gran comunidad de desarrolladores alrededor de este lenguaje.

Ilustración 17 Logo de JavaScript

5.2 ENTORNOS DE DESARROLLO

5.2.1 Visual Studio Code:

Este es un editor de código facilitado y desarrollado por Microsoft *(Ilustración 18)* el cual incluye soporte de depuración, control integrado de Git, finalización inteligente de código, refactorización de código y resaltado de las distintas sintaxis. Además, se le pueden añadir distintas extensiones que facilitarán el trabajo ya sea con autocompletados, temas diversos y más herramientas.

Este desarrollador de código ha sido elegido ya que gracias a su optimización de JavaScript termina favoreciendo un desarrollo fluido y simple.

Ilustración 18 Logo Visual Studio Code

5.3 DOCUMENTACIÓN Y RECURSOS

5.3.1 Google Drive

Google dispone de un cloud *(Ilustración 19)* donde almacenar los archivos de los que disponemos, así como nos ofrece distintas herramientas para editar textos actualizados simultáneamente que se trabaja y un sistema de versiones para poder recuperar ciertos documentos. Esta herramienta se ha usado para almacenar los archivos y la documentación del propio proyecto.

Ilustración 19 Logo Google Drive

5.3.2 Draw.io

Draw.io *(Ilustración 20)* es una herramienta de creación y edición de diagramas libre que te permite la integración con distintas plataformas. El software es una aplicación web realizada mayoritariamente JavaScript y licenciada en Apache v2, la cual funciona en diversos navegadores y te permite crear diagramas contando con diversos modelos para distintos diagramas tales como UML, esquema de red, flujogramas y mapas conceptuales entre muchos otros.

Ilustración 20 Logo Draw.io

5.3.3 Postman

Postman *(Ilustración 21)* es una herramienta orientada a los desarrolladores que resulta muy útil ya que nos permite realizar peticiones de forma sencilla e intuitiva a cualquier Api. En este proyecto hemos hecho uso de esta herramienta para la comprobación del correcto funcionamiento de la API

Ilustración 21 Logo Postman

5.3.4 Power Bi

Power Bi *[13] (Ilustración 22)* es un servicio de análisis de datos de Microsoft orientado a proporcionar visualizaciones interactivas y capacidades de inteligencia empresarial, esta herramienta nos permitirá hacer análisis de datos y mostrar gráficas a partir de dichos datos.

Ilustración 22 Logo Power Bi

5.4 INFRAESTRUCTURA

5.4.1 MongoDB Server

Este sistema de base de datos es un sistema multiplataforma orientados a documentos de libre esquema, es decir, cada entrada o registro contiene un esquema de datos distinto sin tener que tener atributos o columnas repetidos

Lo más interesante de este sistema es su rapidez, su simplicidad a la hora de realizar las consultas y su sencillez de manejo y aprendizaje. *[3] (Ilustración 23)*

Ilustración 23 Logo MongoDB

5.4.2 Modelo Vista Controlador

El patrón de diseño de arquitectura software Modelo-Vista-Controlador (MVC) *(Ilustración 24)* es usado para clasificar la información, lógica del sistema e interfaces facilitadas a los usuarios. En este tipo de arquitecturas existe un sistema central o controlador el cual gestiona la entrada y salida del sistema, uno o varios modelos encargados la información necesaria y la interfaz que muestra los resultados al usuario final

Este tipo de arquitectura se usa en el desarrollo web porque si no se usan pueden llevar a confusión entre los distintos componentes, por lo que se usa para crear cierto orden dentro del sistema. De este modo se permite modificar los distintos componentes sin afectar a otros.

Este modelo está compuesto por:

- **Modelo:** Este componente es el cargado de procesar, administrar y actualizar los datos. Si usamos una base de datos en este apartado se realizarán las consultas, búsquedas y filtros

- **Vista:** Este componente se encarga de mostrar pantallas, ventanas, páginas y formularios al usuario, mostrando al final el resultado de la solicitud. Desde el punto de vista del programador este es el componente responsable de las interfaces de usuario o la visualización del sitio web

- **Controlador:** Este es el componente encargado de gestionar las instrucciones que se reciban, controlarlas y procesarlas. A través de este componente podrán pasar modelos y vistas para que se puedan solicitar los datos necesarios, manipularlos para obtener resultados y entregarlos para poder visualizarlos

Este modelo es actualmente uno de los más usados y lo encontraremos tanto en sistemas de gran envergadura como en los más pequeños. Es recomendable trabajar con este modelo para evitar problemas en sistemas complejos

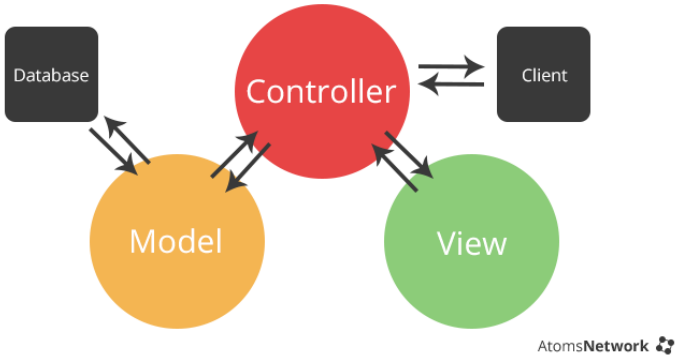


Ilustración 24 Modelo Vista Controlador

5.5 MEBN STACK

El desarrollo full-Stack *[2] (Ilustración 25)* en Javascript, es el conjunto tecnológico necesario para el desarrollo de las capas de una aplicación web con JavaScript. Está compuesto de las cuatro tecnologías más importantes en la industria del desarrollo web actualmente como son: MongoDB, Express, Bootstrap y Node.js

EL desarrollo web full-Stack lleva algunos años en funcionamiento, es cierto que existen otros tales como MEAN (MongoDB, Express, Angular y Node.js) o LAMP (Linux, Apache, MySQL y PHP), entre muchos otros. La popularidad de este sistema de desarrollo reside en que emplea tecnologías interconectadas entre sí.

Ilustración 25 MEBN

5.5.1 MongoDB

Este sistema de base de datos es un sistema multiplataforma orientados a documentos de libre esquema, es decir, cada entrada o registro contiene un esquema de datos distinto sin tener que tener atributos o columnas repetidos

Lo más interesante de este sistema es su rapidez, su simplicidad a la hora de realizar las consultas y su sencillez de manejo y aprendizaje. *[3] (Ilustración 26)*

Ilustración 26 Logo MongoDB

5.5.2 Express

Express *[4] (Ilustración 27)* es un Framework para Node.js, es decir es un framework que sirve para hacer aplicaciones web. Es sencillo y fácil de usar, además se adapta muy bien a la filosofía de Node.js trabajando en el lado del servidor.

Esta herramienta nos permite organizar nuestra web desde la asignación de rutas hasta el manejo de solicitudes y vistas entre otras muchas cosas

Ilustración 27 Logo Express

5.5.3 Bootstrap

Bootstrap [1] *(Ilustración 28)* es un framework CSS y JavaScript que permite dar forma a los sitios web mediante librerías CSS que incluyen muchos de los elementos necesarios tales como botones, tipografías, cuadros, menús…usados en cualquier sitio web a día de hoy.

Se ha tomado la decisión de usar Bootstrap ya que es una tecnología en auge marcada por el mercado y por la gran cantidad de documentación que podemos encontrar.

Ilustración 28 Logo Bootstrap

5.5.4 Node.js

Node.js *[5] (Ilustración 29)*es un run time, esto quiere decir que es un programa que corre JavaScript construido sobre el propio motor de JavaScript que emplea Chrome. Es un entorno de desarrollo usado en la capa del servidor el cual permite el desarrollo de aplicaciones escalables a nivel servidor

Ilustración 29 Logo Node JS

5.6 COMPARATIVA DE TECNOLOGÍAS UTILIZADAS

Procederemos a explicar porque hemos decidido el uso de algunas tecnologías en lugar de otras que ofrecen posibilidades totalmente distintas.

5.6.1 SQL vs NoSQL

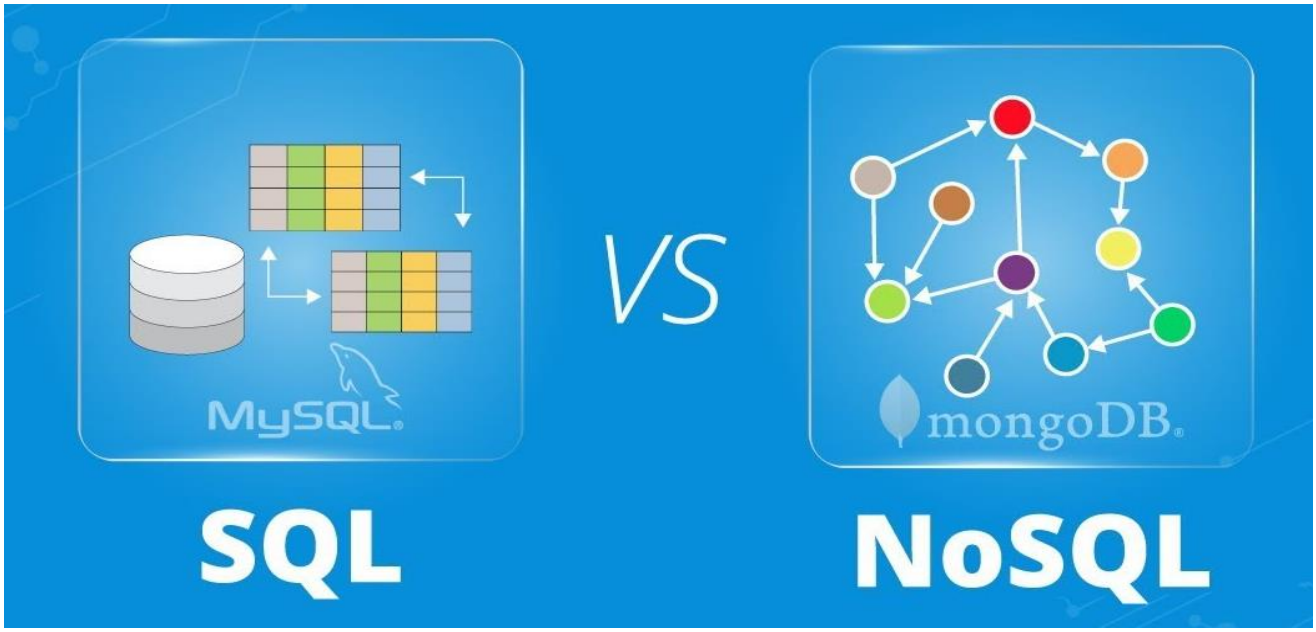


Ilustración 30 SQL vs NoSQL

Se ha tomado la decisión de usar NoSQL, en este caso a través de MongoDB, en vez de SQL como sería PhpMyAdmin por varios motivos *(Ilustración 30)*, el primero de todo es el desarrollar nuevas capacidades y aprender sobre una de las tecnologías con más potencial y popularidad en el desarrollo de bases de datos que existen últimamente, esto se debe a que al contrario que en el caso de las bases de datos SQL, las cuales son mediante tablas y relaciones, es a través de ficheros JSON lo cual permite mayor escalabilidad, libertad de campos ya que siempre se pueden añadir campos nuevos a cualquiera de los documentos. Por el contrario, las tablas de SQL no permiten agregar campos nuevos fácilmente, obligándonos a tener que modificar las tables y todos los registros existentes, esta es una de las razones por lo que se ha usado este tipo de base de datos, ya que para un proyecto así será necesario ir registrando constantemente nuevos campos para poder representar los datos que ayuden más tanto a la institución como al estudiante. Otro de las ventajas de NoSQL en comparativa con SQL es la velocidad, ya que gracias a su capacidad de manejar grandes cantidades de datos no estructurados permitiendo realizar consultas de manera sensible a la carga de trabajo NoSQL nos dará una mayor velocidad en las consultas, más aún en un proyecto como este donde necesitaremos hacer mucho uso de la base de datos.

A continuación, vamos a mostrar un ejemplo de la velocidad de ambos sistemas *[15]* (*Ilustración 31)*:

Los datos de la gráfica fueron tomados en base a 1.000.000 registros con MySQL 5.7.9 y MongoDB 3.2.0 utilizando las configuraciones por defecto en un servidor Ubuntu con 8 CPU's virtuales y 32 GB de RAM en entornos separados.

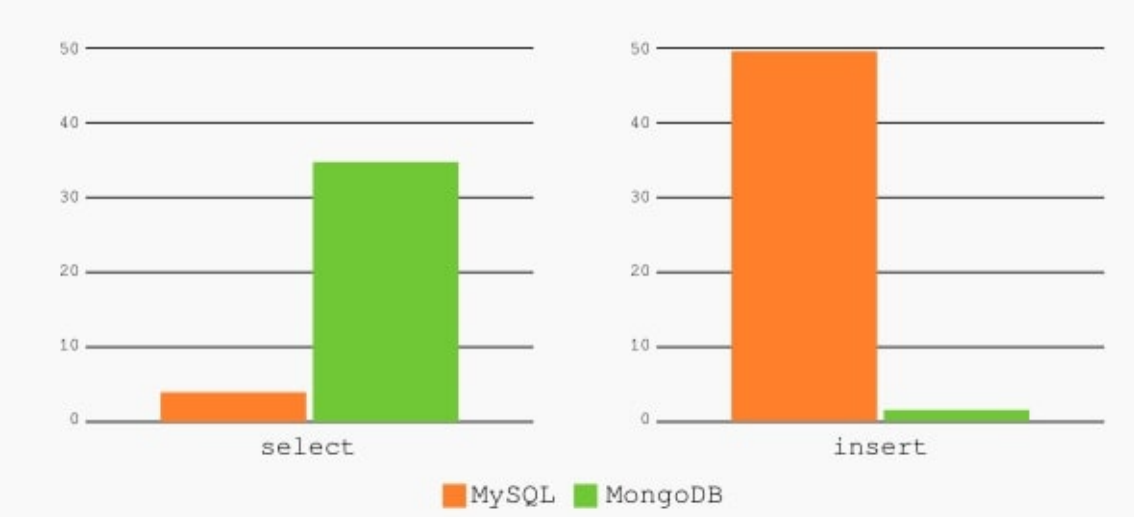


Ilustración 31 Diferencia de velocidades MySQL vs MongoDB

Estas han sido algunas de las muchas ventajas que me han decidido a hacer uso de esta tecnología.

5.6.2 JavaScript vs Python

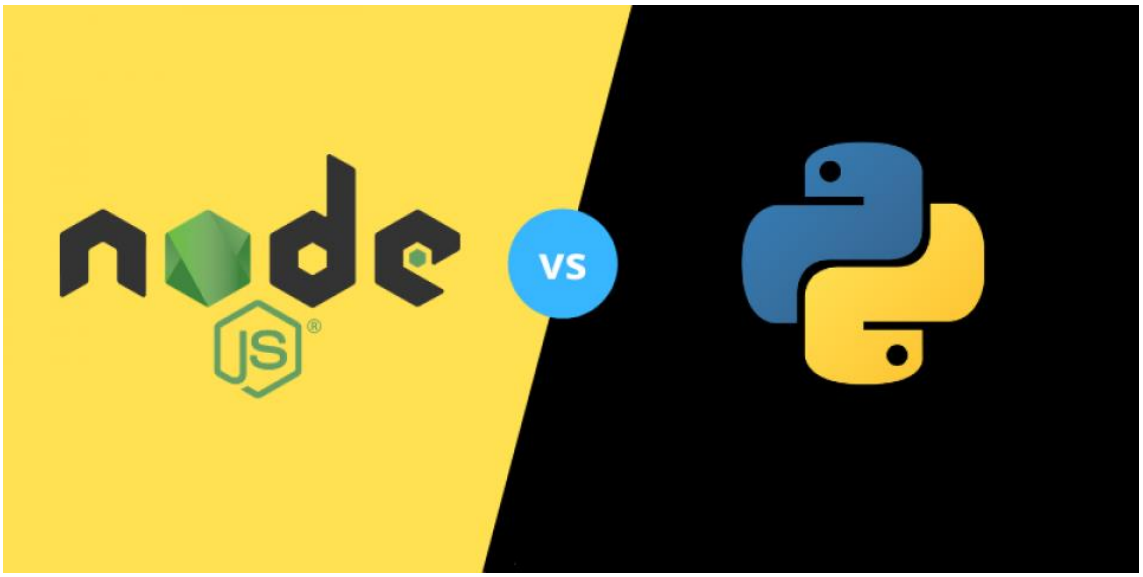


Ilustración 32 Node.js vs Python

En este proyecto se ha hecho uso de Node.js para el desarrollo web, pero sin embargo otra opción totalmente viable sería el uso de Python *(Ilustración 32)* para desarrollar ya que su sintaxis simple y que sea un lenguaje interpretado nos permite desarrollar un sitio web de forma sencilla, sin embargo, me he decantado por el uso de Node.js y a continuación voy a explicar las razones por las que me he decantado por uno.

Si comparamos ambos lenguajes podemos ver que en cuanto a flexibilidad y escalabilidad node.js ofrece a los desarrolladores una escalabilidad sencilla mientras se desarrolla la aplicación web, además presenta un rendimiento sobresaliente y disminuye el tiempo de carga de la aplicación ya que ha habilitado el tiempo de almacenamiento en caché de un solo módulo, sin embargo, Python es un lenguaje más lento ya que es un lenguaje interpretado, además de que no permite una arquitectura asincrónica.

Node.js usa módulos y paquetes administrado en NPM mientras que Python usa empaquetadores que se administran en Python Índex Package que contiene una gran variedad de módulos, en este caso ambos lenguajes son bastantes similares.

Por último, Node.js es la mejor opción en cuanto aplicaciones de alto contenido, mientras que Python está más orientada a aplicaciones de programaciones con cálculos numéricos complejos.

Estas con algunas de las razones por las que he decidido hacer uso de Node.js ya que me ha parecido el lenguaje más indicado para el desarrollo de un proyecto de estas características

5.6.3 Visual Studio Code vs SublimeText

Ilustración 33 VisualStudioCode vs SublimeText

Me dispongo a explicar la razón por la que he elegido VisualStudioCode como entorno de desarrollo en lugar de SublimeText *(Ilustración 33)*.

Ambas opciones son válidas a la hora de realizar cualquier proyecto, la diferencia reside en las capacidades de ayuda que te ofrezcan y la comodidad entre uno u otro, pero finalmente me he decantado por VisualStudioCode, la razón de esto ha sido que es un editor de código totalmente gratuito y abierto, además de que tiene un aspecto bastante más moderno y limpio, lo que facilita el trabajo a la hora de encontrar ayudas entre su interfaz.

6 DESARROLLO

6.1 DISEÑO Y ARQUITECTURA

6.1.1 Diseño de la aplicación Web

El desarrollo de la aplicación está basado en seguir un prototipo realizado como un primer diseño. Este primer diseño estaba formado por una aplicación web la cual contenía una página donde se mostrarán las becas propuestas, otra de inicio de sesión para acceder a las funciones de administrador *(Ilustraciones 34-36)*

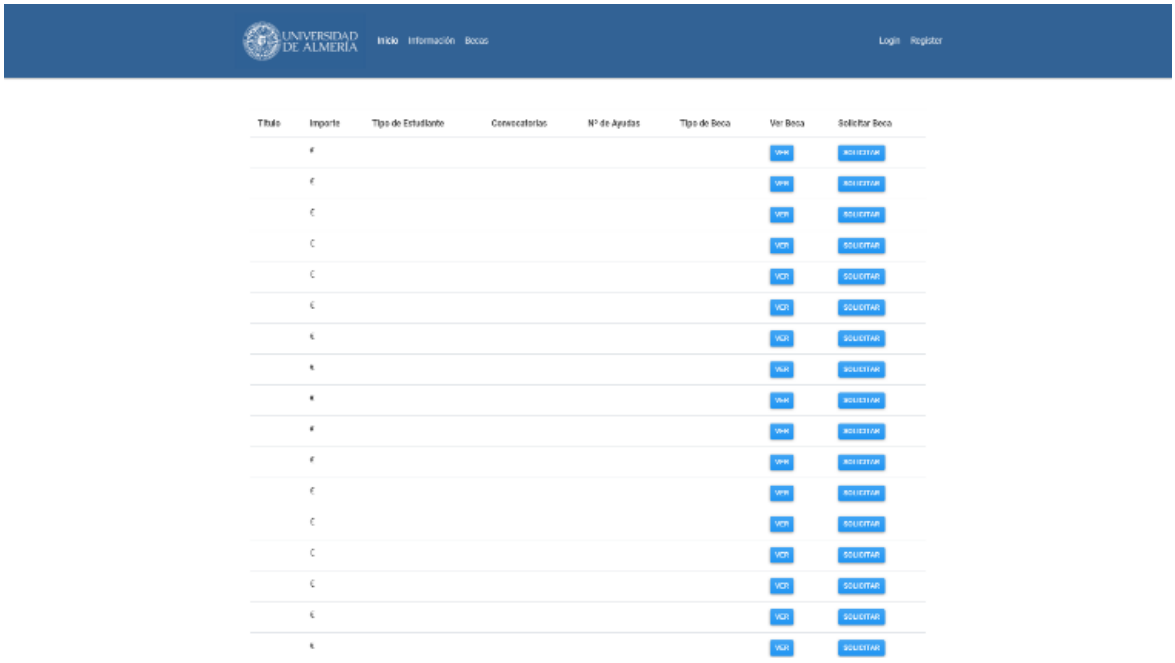


Ilustración 34 Página de Becas

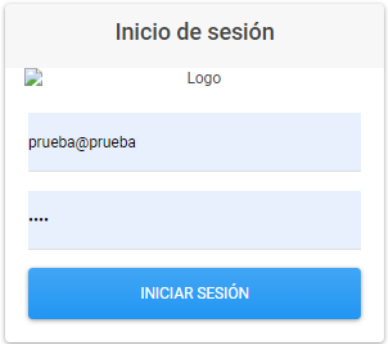


Ilustración 35 Inicio de sesión

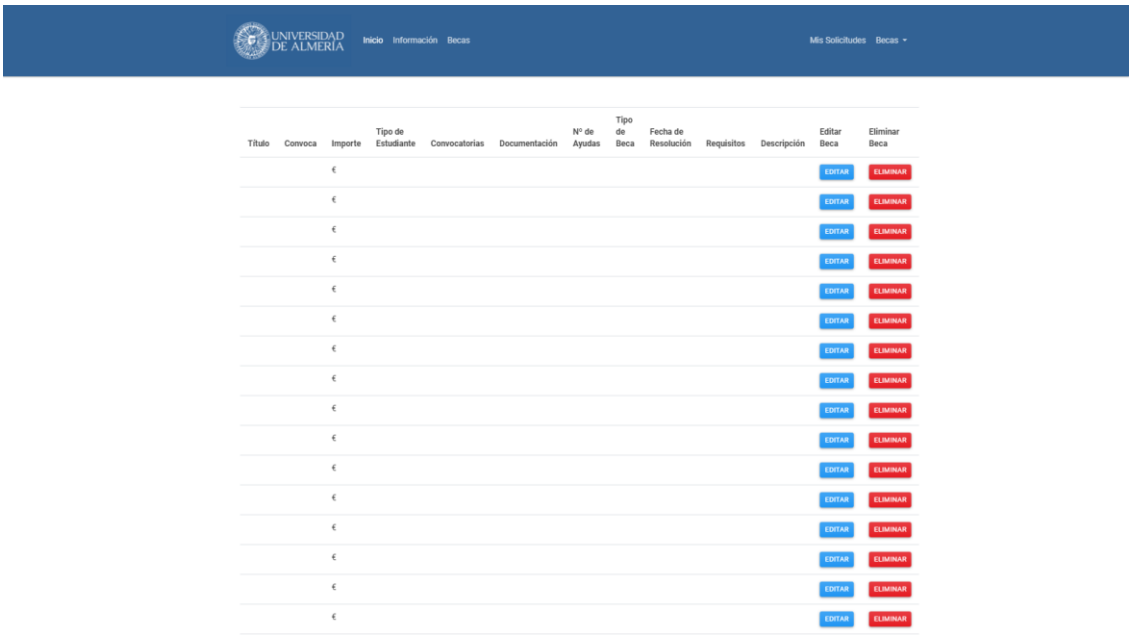


Ilustración 36 Página Becas Administrador

Para poder desarrollar este diseño se llevó a cabo un patrón MVC *(Ilustración 37)*. Gracias a este modelo se tendrá una organización intuitiva y sencilla donde los archivos tendrán nombres claros y separados en carpetas según sus funciones

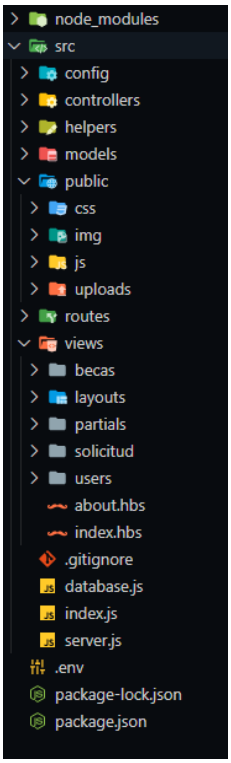


Ilustración 37 Modelo Vista Controlador

Una vez planteado el diseño de la aplicación procedemos a la creación de la base de datos, para ello hemos hecho uso de MongoDB, para mayor facilidad usaremos MongoDB Compass *(Ilustración 38)*, el cual nos permite tener una interfaz para poder manejar la base de datos

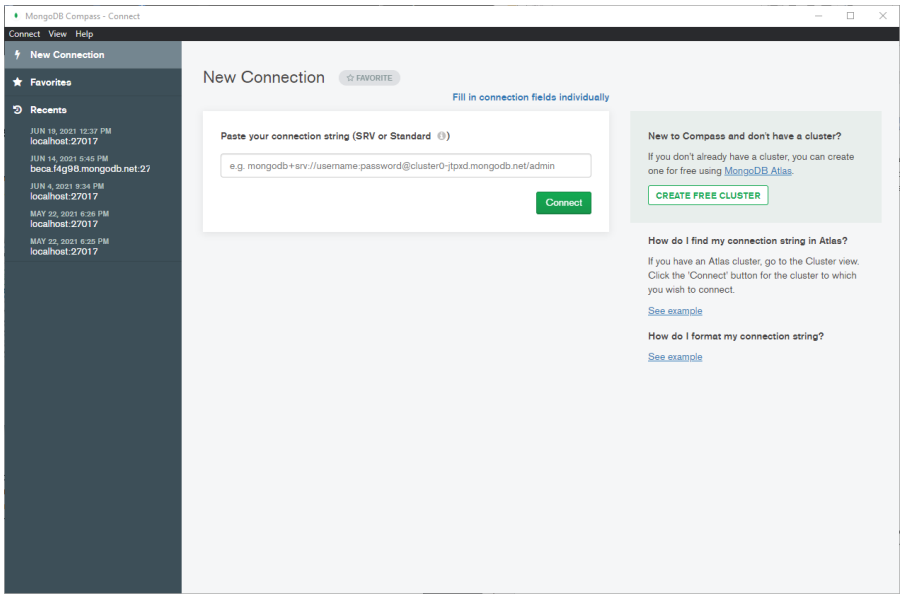


Ilustración 38 MongoDB Compass

Esta herramienta nos permitirá hacer consultas y modificaciones sobre los elementos de las distintas colecciones que tengamos *(Ilustración 39)*.

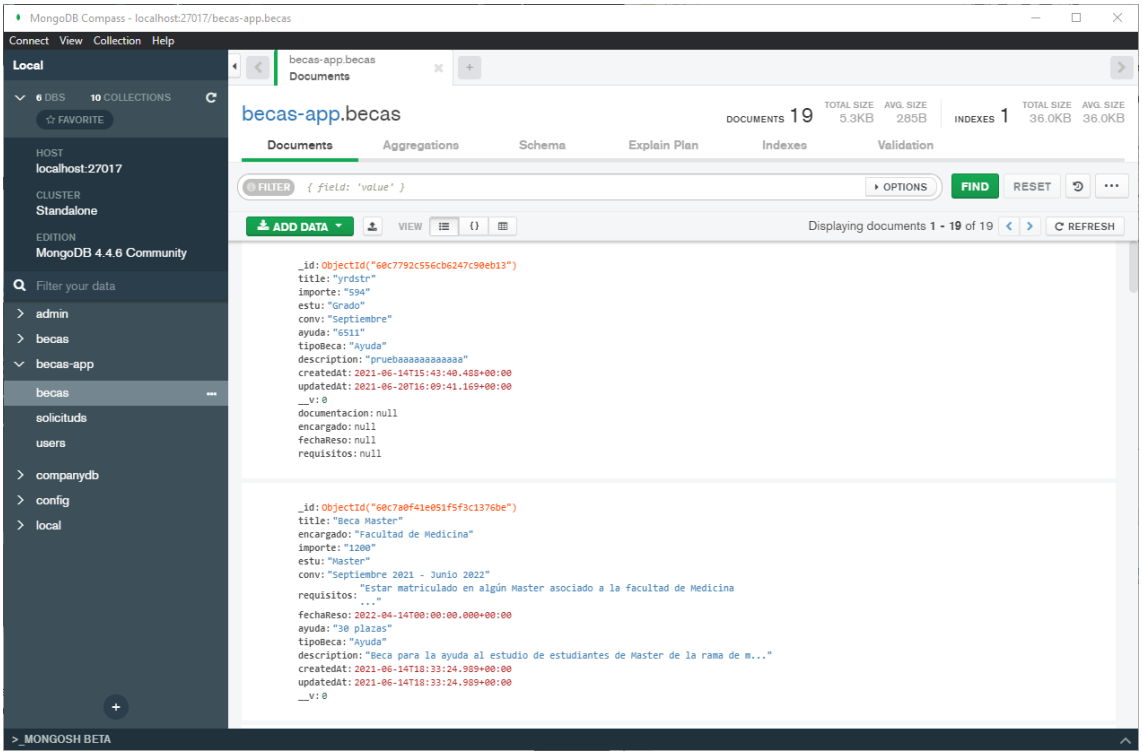


Ilustración 39 Colecciones de la BD

Como comentaremos en futuros apartados para usar la base de datos mediante MongoDB haremos uso de Mongoose, esta es una librería que nos permite la creación de Schemas para MongoDB, gracias a estas estructuras podremos manejarla como una base de datos relacional. A continuación, mostramos un ejemplo de Schema *(Ilustración 40)*:

```
const { Schema, model } = require("mongoose");
const bcrypt = require("bcryptjs");

const UserSchema = new Schema(
  {
    name: { type: String, trim: true },
    email: { type: String, required: true, unique: true, trim: true },
    password: { type: String, required: true },
  },
  {
    timestamps: true,
  }
);

UserSchema.methods.encryptPassword = async (password) => {
  const salt = await bcrypt.genSalt(10);
  return await bcrypt.hash(password, salt);
};

UserSchema.methods.matchPassword = async function (password) {
  return await bcrypt.compare(password, this.password);
};

module.exports = model("User", UserSchema);

```

Ilustración 40 Schema de Users

Como se puede observar, la información queda normalizada, esto quiere decir que usamos Mongoose para definir el Schema de usuario el cual está compuesto de una serie de propiedades y restricciones que se deben cumplir, por ejemplo, todos los datos deben ser de tipo String.

6.1.2 Backend:

El Backend es la parte del desarrollo de aplicaciones web centrada en el funcionamiento lógico de la página, es decir, se encarga del conjunto de acciones que suceden en una página web, como por ejemplo la comunicación con el Servidor.

Algunas de las funciones que gestiona el Backend son las siguientes:

- Conexión con la base de datos
- Funciones lógicas
- Funciones de desarrollo
- Optimización
- Seguridad

Siguiendo esta base el proceso para desarrollar el Backend ha sido el siguiente:

6.1.2.1 Arranque del servidor

Para la creación del proyecto usaremos el comando *"npm init -yes"*, a continuación, instalaremos los módulos *(Ilustración 41)* que necesitaremos tanto para el Frontend como para el Backend:

I. ***Express handlebars***: Motor de plantillas que extiende el HTML.
II. ***Express sessions***: Modulo para crear sesiones dentro del propio servidor tales como la autentificación de usuarios manteniendo sus datos temporalmente en una sesión

III. ***Methods Override***: Módulo para aumentar las funciones de los formularios más allá de las operaciones de Get y Post, tales como Update o Delete
IV. ***Mongoose***: Módulo para la fusión de express con MongoDB
V. ***Passport y Passport local***: Módulo que permite la autentificación de usuarios
VI. ***Bcryptjs***: Módulo que permite la encriptación de contraseñas convirtiendo Strings en Hash
VII. ***Connect flash***: Módulo para el envío de mensajes emergentes a los usuarios
VIII. ***Nodemon:*** Módulo que permite el reinicio automático del servidor, lo que agiliza el desarrollo ya que se reinicia cada vez que se guarda.
IX. ***Mongodb:*** Módulo para el uso y conexión con la base de datos de MongoDB
X. ***Multer:*** Módulo que permite la subida de archivos al servidor
XI. ***Morgan:*** Módulo que permite capturar solicitudes HTTP para Node.js

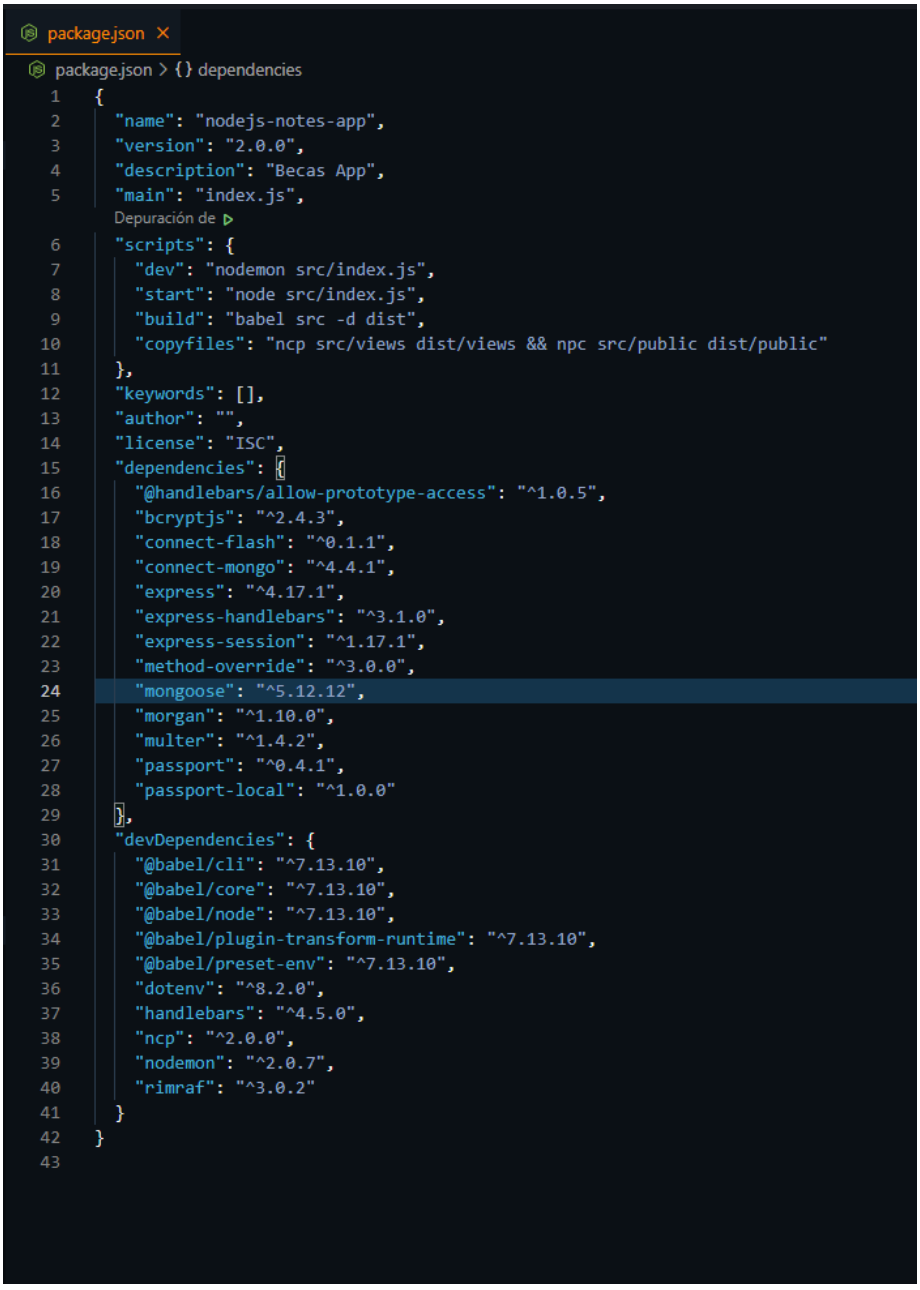


Ilustración 41 Módulos instalados

En el archivo server.js declararemos los módulos que vamos a usar en el proyecto y dividiremos el archivo en varias secciones como son:

- Declaraciones
- Configuraciones
- Middlewares
- Variables globales
- Rutas
- Archivos estáticos
- Inicio del servidor

Primero de todo debemos iniciar el servidor, por defecto lo configuraremos en el puerto 4000, pero en caso de que se nos proporcione otro número de puerto, usaremos la siguiente configuración para que acepte distintos puertos: *"process_env_PORT" (Ilustración 42 - 43)*

```
//Configuracion
app.set("port", process.env.PORT || 4000);
```

Ilustración 42 Configuración del puerto

```
const app = require("./server");
require("./database");

app.listen(app.get("port"), () => {
  console.log("Servidor en el puerto ", app.get("port"));
});
```

Ilustración 43 Inicio del servidor

La configuración de la base de datos se ha hecho en el archivo database.js, es necesario declarar el uso de Mongoose en este archivo ya que será el módulo que necesitaremos *(Ilustración 44)*.

```
const mongoose = require("mongoose");

const { BECAS_APP_MONGODB_HOST, BECAS_APP_MONGODB_DATABASE } = process.env;
const MONGODB_URI =
  "mongodb://" + BECAS_APP_MONGODB_HOST + "/" + BECAS_APP_MONGODB_DATABASE;
mongoose
  .connect(MONGODB_URI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useFindAndModify: false,
    useCreateIndex: true,
  })
  .then((db) => console.log("Base de datos conectada"))
  .catch((err) => console.log(err));
```

Ilustración 44 Conexión a la base de datos

En el apartado de declaraciones necesitaremos el módulo express. Usaremos dicho módulo para crear las variables de nuestra aplicación.

En la sección de configuraciones *(Ilustración 45)*, aparte de establecer el puerto de nuestro server también aplicaremos el módulo handlebars para estructurar nuestro proyecto correctamente.

```
//Configuracion
app.set("port", process.env.PORT || 4000);
app.set("views", path.join(__dirname, "views"));
app.engine(
  ".hbs",
  exphbs({
    defaultLayout: "main",
    layoutsDir: path.join(app.get("views"), "layouts"),
    partialsDir: path.join(app.get("views"), "partials"),
    extname: ".hbs",
    handlebars: allowInsecurePrototypeAccess(Handlebars),
  })
);

app.set("view engine", ".hbs");
```

Ilustración 45 Server.js: Configuración.

A continuación, tenemos el apartado de middlewares *(Ilustración 46)*, donde tendremos los módulos que actuarán entre la base de datos y la aplicación red

```
//Middlewares
app.use(passport.initialize());
app.use(passport.session());
app.use(morgan("dev"));
app.use(express.urlencoded({ extended: false }));
app.use(methodOverride("_method"));
app.use(
  session({
    secret: "secret",
    resave: true,
    saveUninitialized: true,
  })
);
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(flash());
const storage = multer.diskStorage({
  destination: path.join(__dirname, "../src/public/uploads/"),
  filename: function (req, file, cb) {
    cb(null, file.originalname);
  },
});
app.use(
  multer({
    storage,
  }).single("file")
);
```

Ilustración 46 Server.js: Middlewares

En el apartado de variables globales, estableceremos los mensajes del módulo flash como una variable global y guardaremos en *"res.local.user"* los valores del usuario que inicia sesión para que más tarde se puedan usar para autentificarlo *(Ilustración 47)*.

```
//Variables Globales
app.use((req, res, next) => {
  res.locals.success_msg = req.flash("success_msg");
  res.locals.error_msg = req.flash("error_msg");
  res.locals.error = req.flash("error");
  res.locals.user = req.user || null;
  next();
});
```

Ilustración 47 Server.js: Variables globales

El último apartado de nuestro archivo del servidor es el apartado de rutas y archivos estáticos, donde se mostrarán los datos sobre las ubicaciones de los archivos de enrutamiento y una ruta que se puede acceder públicamente *(Ilustración 48)*

```
//Routes

app.use(require("./routes/index.routes"));
app.use(require("./routes/becas.routes"));
app.use(require("./routes/user.routes"));
app.use(require("./routes/solicitud.routes"));
//Archivos estaticos
app.use(express.static(path.join(__dirname, "public")));

module.exports = app;
```

Ilustración 48 Server.js: Routes y Archivos estáticos

6.1.2.2 CRUD

El concepto CRUD (*Ilustración 49)* es un acrónimo cuyo significado es:

- **Create:** Para crear registros.
- **Read and retrieve:** Para leer los registros
- **Update:** Para actualizar los registros
- **Delete:** Para borrar los registros

Estas operaciones están adaptadas a los requisitos del sistema y del usuario, ya sea para la gestión de la Base de datos como para el uso de aplicaciones.

CRUD-Operation	SQL	RESTful HTTP	XQuery
Create	INSERT	POST, PUT	insert
Read	SELECT	GET, HEAD	copy/modify/return
Update	UPDATE	PUT, PATCH	replace, rename
Delete	DELETE	DELETE	delete

Ilustración 49 CRUD

Para la realización del sistema CRUD he hecho uso del lenguaje JavaScript. Para poder simplificar el proceso se ha separado el proceso en dos partes, la parte visual mediante formularios en el Frontend y la parte operacional en el Backend

6.1.2.3 Modelo de datos

A través de un modelado de datos queremos organizar y estructurar los datos de los usuarios, esto facilitará su uso y gestión en la base de datos.

A continuación, vamos a explicar el código en distintos apartados, estos apartados servirán para todos los modelos ya que siguen el mismo esquema.

6.1.2.3.1 Modelo de datos de usuario

Como se ha comentado en apartados anteriormente Mongoose nos permite crear Schemas para que al base de datos se pueda manejar como un modelo relacional.

En este apartado mostraremos el diseño para el registro de usuarios e inicio de sesión *(Ilustración 50)*

```
const { Schema, model } = require("mongoose");
const bcrypt = require("bcryptjs");

const UserSchema = new Schema(
  {
    username: { type: String, trim: true },
    email: { type: String, required: true, unique: true, trim: true },
    password: { type: String, required: true },
    roles: [
      {
        ref: "Role",
        type: Schema.ObjectId,
      },
    ],
  },
  {
    timestamps: true,
  }
);

UserSchema.methods.encryptPassword = async (password) => {
  const salt = await bcrypt.genSalt(10);
  return await bcrypt.hash(password, salt);
};

UserSchema.methods.matchPassword = async function (password) {
  return await bcrypt.compare(password, this.password);
};

module.exports = model("User", UserSchema);

```

Ilustración 50 Modelo de Usuarios

- Primero de todo requeriremos Mongoose para diseñar el Schema (*Ilustración 51)*, además requeriremos Bcryptjs para encriptar las contraseñas.

```
const { Schema, model } = require("mongoose");
const bcrypt = require("bcryptjs");
```

Ilustración 51 UserSchema: Mongoose

- Continuaremos diseñando y creando el Schema denominado "UserSchema" (*Ilustración 52)*. En esta colección se almacenará la información referente a los usuarios que llegarán a nuestro sistema, donde los datos requeridos serán:
  - **Username:** Nombre del usuario
  - **Email:** Email del usuario
  - **Password:** Contraseña del usuario
  - **Roles:** Rol del usuario

```
const UserSchema = new Schema(
  {
    username: { type: String, trim: true },
    email: { type: String, required: true, unique: true, trim: true },
    password: { type: String, required: true },
    roles: [
      {
        ref: "Role",
        type: Schema.ObjectId,
      },
    ],
  },
  {
    timestamps: true,
  }
);
```

Ilustración 52 UserSchema: Datos

- En el siguiente paso vamos a crear unos métodos para cifrar las contraseñas*(Ilustración 53)* para aumentar la seguridad de los usuarios, en el método de encriptación de contraseñas vamos a aplicar una función Hash 10 veces y después asociaremos la contraseña sin encriptar con la encriptada en el método de comparación de contraseñas.

```
UserSchema.methods.encryptPassword = async (password) => {
  const salt = await bcrypt.genSalt(10);
  return await bcrypt.hash(password, salt);
};

UserSchema.methods.matchPassword = async function (password) {
  return await bcrypt.compare(password, this.password);
};

module.exports = model("User", UserSchema);
```

Ilustración 53 UserSchema: Encriptación

Como se puede observar en el modelo de usuarios se hace referencia a otro modelo, en este caso, al modelo de Roles, ya que cuando se generan los usuarios se les asignan roles para que tengan acceso a ciertas funciones según su rol.

6.1.2.3.2 Modelo de Roles

```
const { Schema, model } = require("mongoose");
const roleSchema = new Schema(
  {
    name: String,
  },
  {
    versionKey: false,
  }
);
module.exports = model("Role", roleSchema);
```

Ilustración 54 RolSchema

Como se observa, el Schema (*Ilustración 54)* sigue el mismo modelo que el anterior, pero en esta ocasión los roles se crearán una vez se ejecute el servidor por primera vez, y dichos roles ya permanecerán en el sistema para siempre, para ello crearemos un método que comprobará si existen roles (*Ilustración 55)*, en caso de no existir se crearán. Por último, el método se exportará y lo aplicaremos en la parte de inicio del servidor. Dicho procedimiento se ejecutará en el archivo initialSetup.js

```
const Role = require("../models/Role");

const createRoles = async () => {
  try {
    const count = await Role.estimatedDocumentCount();

    if (count > 0) return;

    const values = await Promise.all([
      new Role({ name: "user" }).save(),
      new Role({ name: "moderator" }).save(),
      new Role({ name: "admin" }).save(),
    ]);

    console.log(values);
  } catch (error) {
    console.error(error);
  }
};
module.exports = createRoles;
```

Ilustración 55 Creación de roles

Por último, debemos usar el método justo al iniciar el servidor, para ello debemos llamar al método a la hora de inicializar el servidor (*Ilustración 56)*.

```
//inicializacion
const app = express();
require("./config/initialSetup");
require("./config/passport");
```

Ilustración 56 Inicialización del sistema de Roles

6.1.2.3.3 Modelo de Becas

Como se puede observar este Schema (*Ilustración 57)*requiere de muchos campos, ya que serán todos los requisitos mínimos para poder crear una nueva Beca

```
const { Schema, model } = require("mongoose");

const BecaSchema = new Schema(
  {    title: {
      type: String,
      required: true,
    },    encargado: {
      type: String,
    },    requisitos: {
      type: String,
    },    documentacion: {
      type: String,
    },    fechaReso: {
      type: Date,
    },    importe: {
      type: String,
      required: true,
    },    estu: {
      type: String,
      required: true,
    },    conv: {
      type: String,
      required: true,
    },    ayuda: {
      type: String,
      required: true,
    },    tipoBeca: {
      type: String,
      required: true,
    },    description: {
      type: String,
      required: true,
    },    estado: {
      type: String,
      required: true,
    },
  },
  {
    timestamps: true,
  }
);
module.exports = model("Beca", BecaSchema);
```

Ilustración 57 BecaSchema

6.1.2.3.4 Modelo de solicitud

En este apartado vamos amostrar el Schema de las solicitudes(*Ilustración 58)*, donde vemos los campos necesarios para solicitar una beca

```
const { Schema, model } = require("mongoose");

const SolicitudSchema = new Schema(
  {
    titleBeca: {
      type: String,
    },
    nombre: {
      type: String,
      required: true,
    },
    tipoAlumno: {
      type: String,
    },
    dni: {
      type: String,
    },
    matriculado: {
      type: String,
    },
    cursoAcademico: {
      type: Date,
    },
  },
  {
    timestamps: true,
  }
);
module.exports = model("Solicitud", SolicitudSchema);
```

Ilustración 58 SolicitudSchema

6.1.2.4 Enrutamiento de los datos

En este apartado comentaremos cómo funciona la lógica interna del sistema web, desde el inicio de sesión hasta el acceso con las diferentes vistas. Estos ficheros sirven como enrutador entre unas funciones y otras dejando el camino claro y despejado. Como todas las rutas siguen el mismo esquema vamos a mostrar Uno de los ficheros de enrutamiento como ejemplo, lo que vamos a contar a continuación se aplica al resto de archivos.

6.1.2.4.1 Enrutamiento de becas

A continuación, mostramos el archivo que contiene las rutas (*Ilustración 59)* de las distintas funciones que se pueden realizar con una beca desde nuestro sistema. Como se puede observar, según la ruta a la que se acceda en cada caso se ejecutarán una serie de funciones, comprobaciones y restricciones.

```
const { Router } = require("express");
const router = Router();

const {
  renderBecaForm,
  renderBecas2,
  renderBecas3,
  createNewBeca,
  renderBecas,
  renderEditForm,
  updateBeca,
  deleteBeca,
} = require("../controllers/becas.controller");
const { createNewSolicitud } = require("../controllers/solicitud.controller");

const { isAuthenticated, isModerator, isAdmin } = require("../helpers/auth");

//Nueva Beca
router.get("/becas/add", [isAuthenticated, isModerator], renderBecaForm);

router.post("/becas/new-beca", isAuthenticated, createNewBeca);

//Obtener todas las Becas
router.get("/becas/becas-admin", isAuthenticated, renderBecas);

router.get("/becas/becas-user", renderBecas2);

//Obtener Beca por Id

router.get("/becas/get-beca/:id", renderBecas3, createNewSolicitud);

// Editar Becas
router.get("/becas/edit/:id", isAuthenticated, renderEditForm);

router.put("/becas/edit-beca/:id", isAuthenticated, updateBeca);

// Eliminar Becas
router.delete("/becas/delete/:id", isAuthenticated, deleteBeca);

module.exports = router;
```

Ilustración 59 becas.routes.js

Como vemos a continuación, dentro de cada ruta podemos ver una serie de comandos, en este caso en concreto el sistema una vez se accede a esta ruta comprobaremos en primer lugar si dicho usuario que intenta acceder está autentificado, una vez comprobado que es un usuario autenticado debemos comprobar si dicho usuario cumple con el rol requerido para poder acceder, en este caso será necesario ser moderador o administrador. Una vez comprobado esto el sistema ya nos lanza a la siguiente función, en este caso será el formulario para añadir una beca *(Ilustración 60)*.

```
//Nueva Beca
router.get("/becas/add", [isAuthenticated, isModerator, isAdmin], renderBecaForm);
```

Ilustración 60 becas.routes.js: Ruta con restricciones

Estas verificaciones se harán en el archivo auth.js y como vamos a mostrar a continuación comprobaremos si dicho usuario está realmente registrado en el sistema a partir del método *"isAuthenticated"* (*Ilustración 61)*, en caso de que no lo esté nos lanzará un mensaje indicando que no es autorizado y nos devolverá a la vista de inicio de sesión para que verifiquemos que somos un usuario registrado

```
helpers.isAuthenticated = (req, res, next) => {
  if (req.isAuthenticated()) {
    return next();
  }
  req.flash("error_msg", "Usuario no autorizado");
  res.redirect("/users/signin");
};
```

Ilustración 61 Auth.js: isAuthenticated

Para la comprobación del rol que posee cada usuario a partir del id de dicho usuario comprobaremos si cada perfil contiene el código concordante con dicho rol, en caso de ser correcto nos dejará acceder a dicho apartado, en este caso mostramos como ejemplo el método isModerator (*Ilustración 62)*, pero en el caso de comprobación de si es administrador el ejemplo será exactamente el mismo, pero en vez de comprobar si es "moderator" comprobara si es "admin"

```
helpers.isModerator = async function (req, res, next) {
  const user = await User.findById(req.user._id);
  console.log(user);
  const roles = await Role.find({ _id: { $in: user.roles } });
  for (let i = 0; i < roles.length; i++) {
    if (roles[i].name == "moderator") {
      next();
      return;
    }
  }
  req.flash("error_msg", "Usuario no Moderador");
};
```

Ilustración 62 Auth.js: isModerator

Regresando al apartado de rutas, en este ejemplo, al contrario que en el caso anterior donde para poder acceder debíamos tener ciertos atributos en nuestro perfil de usuario, no dispone de restricciones ni verificaciones, por lo que este apartado será de libre acceso para todos los usuarios, tanto registrados como gente no registrada *(Ilustración 63)*

```
router.get("/becas/becas-user", renderBecas2);
```

Ilustración 63 becas.routes.js: Ruta sin restricciones

6.1.2.5 Controladores

Vamos a comentar cómo funcionan los controladores. Esto sirve para poder tener de manera ordenada todas las funciones o controladores que llevan a cabo las distintas opciones, es decir, según a la beca que accedamos nos llevará a un sitio que tendrá un controlador y se realizarán unas acciones

6.1.2.5.1 Controladores de becas

Como podemos ver, este controlador (*Ilustración 64)* dispone de distintos métodos a los que el sistema llamará según la ruta a la que accedamos. Vamos a desglosar el código para explicar los distintos métodos. Esta explicación sirve también para los controladores de las solicitudes ya que parten de la misma base y mismos métodos.

```
const becasCtrl = {};

const Beca = require("../models/Beca");

becasCtrl.renderBecaForm = (req, res) => {
  res.render("becas/new-beca");
};

becasCtrl.renderBecas3 = async (req, res) => {
  const beca = await Beca.findById(req.params.id);
  console.log(beca);
  res.render("becas/get-beca", { beca });
};

becasCtrl.createNewBeca = async (req, res) => {
  const {    title,    encargado,    importe,    estu,    conv,    requisitos,    documentacion,    fechaReso,    ayuda,    tipoBeca,    description,    estado,  } = req.body;
  const newBeca = new Beca({
    title,    encargado,    importe,    estu,    conv,    requisitos,    documentacion,    fechaReso,    ayuda,    tipoBeca,    description,    estado,  });
  await newBeca.save();
  req.flash("success_msg", "Beca añadida correctamente");
  res.redirect("/becas/becas-user");
};

becasCtrl.renderBecas = async (req, res) => {
  const becas = await Beca.find();
  res.render("becas/becas-admin", { becas });
};
becasCtrl.renderBecas2 = async (req, res) => {
  const becas = await Beca.find();
  res.render("becas/becas-user", { becas });
};

becasCtrl.renderEditForm = async (req, res) => {
  const beca = await Beca.findById(req.params.id);
  console.log(beca);
  res.render("becas/edit-beca", { beca });
};

becasCtrl.updateBeca = async (req, res) => {
  const {    title,
    encargado,    importe,    estu,    conv,    requisitos,    documentacion,    fechaReso,    ayuda,    tipoBeca,    description,  } = req.body;
  await Beca.findByIdAndUpdate(req.params.id, {
    title,    encargado,    importe,    estu,    conv,    requisitos,    documentacion,    fechaReso,    ayuda,    tipoBeca,    description,  });
  req.flash("success_msg", "Beca modificada correctamente");
  res.redirect("/becas/becas-admin");
};

becasCtrl.deleteBeca = async (req, res) => {
  await Beca.findByIdAndDelete(req.params.id);
  req.flash("success_msg", "Beca eliminada correctamente");
  res.redirect("/becas/becas-admin");
};
becasCtrl.subirArchivo = (req,res) =>{

}

module.exports = becasCtrl;
```

Ilustración 64 Becas.Controller

En el primer método servirá únicamente para redirigirnos al formulario para añadir las becas *(Ilustración 65)*

```
becasCtrl.renderBecaForm = (req, res) => {
  res.render("becas/new-beca");
};
```

Ilustración 65 Becas.Controller: renderBecaForm

En el siguiente método funcionará, a partir del id de una de las becas nos mostrará dicha beca en concreto con una ampliación de los datos *(Ilustración 66)*

```
becasCtrl.renderBecas3 = async (req, res) => {
  const beca = await Beca.findById(req.params.id);
  console.log(beca);
  res.render("becas/get-beca", { beca });
};
```

Ilustración 66 Becas.Controller: renderBecas3

A continuación, podemos ver el método para añadir las becas(*Ilustración 67)* a la base de datos, a partir de los datos que recibamos del formulario se guardará cada dato en su respectivo campo declarado previamente y se guardará en la base de datos, una vez creada nos redirigirá al muestrario de becas para que se pueda comprobar si se ha creado correctamente, también nos aparecerá un mensaje de verificación mostrando que todo ha salido bien

```
becasCtrl.createNewBeca = async (req, res) => {
  const {  title,  encargado,  importe,  estu,  conv,  requisitos,  documentacion,  fechaReso,  ayuda,  tipoBeca,  description,  estado,  } = req.body;
  const newBeca = new Beca({
    title,  encargado,  importe,  estu,  conv,  requisitos,  documentacion,  fechaReso,  ayuda,  tipoBeca,  description,  estado,  });
  await newBeca.save();
  req.flash("success_msg", "Beca añadida correctamente");
  res.redirect("/becas/becas-user");
};
```

Ilustración 67 Becas.Controller: createNewBeca

Para poder editar las becas necesitaremos dos acciones, la primera será mostrar el formulario con los datos de la beca que queremos editar y después editaremos dicha beca modificando los datos recibidos y enviándolos de nuevo a la base de datos *(Ilustración 68)*

```
becasCtrl.renderEditForm = async (req, res) => {
  const beca = await Beca.findById(req.params.id);
  console.log(beca);
  res.render("becas/edit-beca", { beca });
};

becasCtrl.updateBeca = async (req, res) => {
  const {  title,
    encargado,  importe,  estu,  conv,  requisitos,  documentacion,  fechaReso,  ayuda,
    tipoBeca,  description,  } = req.body;
  await Beca.findByIdAndUpdate(req.params.id, {
    title,  encargado,  importe,  estu,  conv,  requisitos,  documentacion,  fechaReso,
    ayuda,  tipoBeca,  description,  });
  req.flash("success_msg", "Beca modificada correctamente");
  res.redirect("/becas/becas-admin");
};
```

Ilustración 68 Becas.Controller: UpdateBecas

Por último, tenemos el método para eliminar las becas, para ello buscaremos la beca a partir de su id y la eliminaremos *(Ilustración 69)*

```
becasCtrl.deleteBeca = async (req, res) => {
  await Beca.findByIdAndDelete(req.params.id);
  req.flash("success_msg", "Beca eliminada correctamente");
  res.redirect("/becas/becas-admin");
};
```

Ilustración 69 Becas.Controller: DeleteBecas

6.1.2.5.2 Controladores de Usuarios

En este apartado vamos a mostrar como realiza internamente el inicio de sesión nuestro sistema (*Ilustración 70)*, para ello haremos uso del modelo creado de usuarios y el modelo de roles ya que nos harán falta a la hora de comprobar los campos que contienen si concuerdan con los recibidos. Lo primero de todo será la parte del registro, donde deberemos realizar varias comprobaciones tales como revisar si el email está en uso, si la contraseña usada cumple con requisitos tales como que debe tener mínimo 4 caracteres o si la contraseña que ha escrito coincide con la confirmación de la contraseña para evitar errores de usuario, además este método llamará al método para poder encriptar la contraseña, lo que permitirá que se cree la función hash que se guardará en el sistema directamente para encriptar la contraseña. En caso de que no se escoja un rol especifico el sistema asignará al usuario el rol de "user"por defecto, lo que permitirá que los alumnos se puedan registrar libremente para solicitar sus becas sin que existan conflictos de asignación de roles erróneos

```
usersCtrl.signup = async (req, res) => {
  const errors = [];
  const { username, email, password, confirm_password, roles }
  = req.body;
  if (password != confirm_password) {
    errors.push({ text: " Las contraseñas no coinciden" });
  }
  if (password.length < 4) {
    errors.push({ text: "La contraseña debe contener al menos 4
    caracteres." });
  }
  if (errors.length > 0) {
    res.render("users/signup", {
      errors,
      username,
      email,
    });
  } else {
    const emailUser = await User.findOne({ email: email });
    if (emailUser) {
      req.flash("error_msg", "El correo ya esta en uso.");
      res.redirect("/users/signup");
    } else {
      const newUser = new User({ username, email, password,
      roles });
      newUser.password = await newUser.encryptPassword(password)
      ;
      if (roles) {
        const foundRoles = await Role.find({ name: { $in: roles
        } });
        user.roles = foundRoles.map((role) => role._id);
      } else {
        const role = await Role.findOne({ name: "user" });
        newUser.roles = [role._id];
      }
      await newUser.save();
      console.log(newUser);
      req.flash("success_msg", "Te has registrado con éxito.");
      res.redirect("/users/signin");
    }
  }
};
```

Ilustración 70 User.Controller: signup

Una vez se ha registrado el usuario podemos acceder a la parte de inicio de sesión (*Ilustración 71)*, donde contaremos con un formulario para introducir los datos, estos datos serán leídos por el sistema y gracias al módulo Passport se comprobará si los datos coinciden con el sistema, en caso de ser correcto nos llevará al apartado de becas, en caso de no coincidir nos volverá a redirigir al inicio de sesión mostrando un mensaje sobre lo que ha fallado, ya sea porque el usuario no está registrado o la contraseña no coincide con la que hay registrada

```
usersCtrl.renderSignInForm = (req, res) => {
  res.render("users/signin");
};

usersCtrl.signin = passport.authenticate("local", {
  successRedirect: "/becas/becas-user",
  failureRedirect: "/users/signin",
  failureFlash: true,
});
```

Ilustración 71 User.Controller: Signin

Para ello hará uso del fichero “Passport.js” donde se configurará la verificación del usuario por pasos, comprobando primero el email y después la contraseña, llamando al método “matchPassword” que realizará la comprobación de las contraseñas

Si el usuario quiere puede cerrar sesión (*Ilustración 72)* a través del botón logout que estará disponible en el menú una vez inicie sesión

```
usersCtrl.logout = (req, res) => {
  req.logout();
  req.flash("success_msg", "Sesión cerrada");
  res.redirect("/users/signin");
};
```

Ilustración 72 User.Controller: Logout

6.1.2.6 Navegación

Una vez declaradas nuestras vistas desarrolladas en “.hbs” y el sistema CRUD establecido con sus controladores y sus rutas estableceremos la navegación del sistema

6.1.2.6.1 Navegación general

Para evitar la repetición de código y para que todo siga un mismo esquema se ha optado por desarrollar un archivo “main.hbs” *(Ilustración 73 - 74)* desde el cual heredarán todo el resto de vistas del sistema:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Becas App</title>
    <!-- BOOTSTRAP -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootswatch/4.5.2/materia/bootstrap.min.css">
    <!-- FONT AWESOME -->
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU" crossorigin="anonymous">
    <!-- CUSTOM CSS -->
    <link rel="stylesheet" href="/css/main.css">
```

Ilustración 73 Navegación: Header

En el body hemos usado la estructura {{ > ejemplo }} para evitar repeticiones de código de forma innecesaria ya que todas las vistas comparten la misma estructura .hbs

- {{>navigation}}: En esta sección irá el menú de navegación, al ser un elemento común y presente en todo momento se ha decidido crear un archivo "navigation.hbs" referenciado en el body.
- {{>messages}} , {{>errors}}: Como hemos nombrado con anterioridad, hay ocasiones en las que el sistema enviará mensajes de error o confirmación a los usuarios. Para ello se han creado los archivos "messages.hbs" y "errors.hbs" donde estará la estructura de cada uno de los mensajes
- {{>body}}: Por último, tenemos la estructura body que permitirá que todas las páginas mantengan la misma estructura

Finalmente encontramos la parte de los scripts, en esta sección irán referenciados los scripts de los que hacemos uso a través de Bootstrap

```
<body>

    {{>navigation}}

    <main class="container p-5">
        {{>messages}}
        {{>errors}}
        {{{body}}}
    </main>


    <!--BOOTSRTRAP 4 -->
 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
 integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH
 +8abtTE1Pi6jizo" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/
    umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2
    +9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
    crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/
    bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3
    +MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
    crossorigin="anonymous"></script>
  </body>
</html>
```

Ilustración 74 Navegación: Body

6.1.2.6.2 Navegación de usuarios autenticados / sin autenticados

Una vez la navegación ha sido establecida debemos asignarla dependiendo de si un usuario está autenticado o no.

Existen casos en los que será que un usuario esté logeado para que pueda acceder a ciertas funciones, como es en el caso de alguien que solicite una beca, ya que se necesitarán sus datos de usuario recogidos previamente y así también se le facilitará un entorno donde pueda revisar todas las becas que ha solicitado, por ello habrá apartados que no podrá visualizar sin iniciar sesión, para ello en el apartado de "navigation.hbs" se han añadido ciertas condiciones:

Usando la variable global "user" donde guardaremos la sesión iniciada haremos una consulta comprobando si dicho usuario está autenticado y en caso de estarlo se mostrará cierta información *(Ilustración 75)*:

```
{{#if user}}
  <li class="nav-item">
    <a class="nav-link" href="/solicitud/solicitud">Mis Solicitudes</a>
  </li>
  <li class="nav-item dropdown">
    <a
      class="nav-link dropdown-toggle"
      href="#"
      id="navbarDropdown"
      role="button"
      data-toggle="dropdown"
      aria-haspopup="true"
      aria-expanded="false"
    >
      Becas
    </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
      <a
        class="dropdown-item"
        href="/becas/becas-admin"
      >Editar/Eliminar</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="/becas/add">Añadir una Beca</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="/users/Logout">Logout</a>
    </div>
  </li>
{{else}}

  <li class="nav-item">
    <a class="nav-link" href="/users/signin">Login</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/users/signup">Register</a>
  </li>
{{/if}}
```

Ilustración 75 Navegación usuario autenticado/no autenticado

Para que un usuario no pueda acceder a las secciones escribiendo las url desde el navegador, como por ejemplo escribiendo ("/becas/becas-admin"), apartado en el cual se lleva la gestión, modificación y eliminación de las becas, se ha usado como comentamos previamente el uso de "Passport" y su método "isAuthenticated()".

6.1.3 Frontend:

El Frontend es la parte de nuestro sistema web a la que los usuarios podrán acceder. Este incluye todas las tecnologías de diseño y desarrollo web que ocurren en los navegadores y que se encargan de la interactividad con los usuarios.

Como nombramos con anterioridad HTML, CSS y JavaScript son los lenguajes usados en este apartado además de librerías como Bootstrap o FontAwesome, para expandir al máximo la posibilidad del sistema

En este apartado vamos a mostrar las distintas vistas del sistema:

Para empezar, vamos a mostrar el apartado de Becas que verá el usuario y sus distintas opciones:

6.1.3.1 Header

En esta apartado vamos amostrar el Header (*Ilustración 76 - 80)* diseñado para el sitio web. Se ha decidido usar un diseño limpio y simple para no cargarlo de información liosa que pueda confundir al usuario. A parte de ser un header responsive, es decir, que se ajusta al tamaño de la página, el header contará con una función en la que según si somos un usuario registrado o no mostrará ciertos campos, en caso de no estar logeado nos mostrará un cuadro de mandos para que podamos iniciar sesión o registrarnos, por el contrario, si estamos logeados nos mostrará un desplegable con las distintas funciones tales como agregar, editar o eliminar

```
<nav
  class="navbar navbar-expand-lg navbar-dark"
  style="background-color: #326295;"
>
  <div class="container">
    <a class="navbar-logo" href="/">
      <img src="/img/logo2.png" class="mx-auto d-block logo m-" alt="Logo" />
    </a>
    <button
      class="navbar-toggler"
      type="button"
      data-toggle="collapse"
      data-target="#navbarNav"
      aria-controls="navbarNav"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item active">
          <a class="nav-link" href="/">Inicio
            <span class="sr-only">(current)</span></a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/about">Información</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/becas/becas-user">Becas</a>
        </li>
      </ul>
      <ul class="navbar-nav ml-auto">
        {{#if user}}
          <li class="nav-item">
            <a class="nav-link" href="/solicitud/solicitud">Mis Solicitudes</a>
          </li>
          <li class="nav-item dropdown">
            <a
              class="nav-link dropdown-toggle"
              href="#"
              id="navbarDropdown"
              role="button"
              data-toggle="dropdown"
              aria-haspopup="true"
              aria-expanded="false"
            >
              Becas
            </a>
            <div class="dropdown-menu" aria-labelledby="navbarDropdown">
              <a
                class="dropdown-item"
                href="/becas/becas-admin"
              >Editar/Eliminar</a>
              <div class="dropdown-divider"></div>
              <a class="dropdown-item" href="/becas/add">Añadir una Beca</a>
              <div class="dropdown-divider"></div>
              <a class="dropdown-item" href="/users/logout">Logout</a>
            </div>
          </li>
        {{else}}

          <li class="nav-item">
            <a class="nav-link" href="/users/signin">Login</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/users/signup">Register</a>
          </li>
        {{/if}}
      </ul>
    </div>
  </div>
</nav>
```

Ilustración 76 Código Header

Ilustración 78 Vista Header sin Iniciar sesión

Ilustración 77 Vista de Header con Sesión Iniciada

Ilustración 79 Vista de Header Responsive

Ilustración 80 Vista de Header Responsive expandido

6.1.3.2 Formulario de Registro de usuarios

Procedemos a mostrar el formulario de registro de usuarios (*Ilustración 81 - 82)*, donde cada usuario deberá rellenar los campos con su nombre, su email y su contraseña, una vez registrado el usuario se le asignará el rol de "user" en el sistema como hemos comentado en secciones anteriores previamente.

```
<div class="row">
  <div class="col-md-4 mx-auto">
    <div class="card">
      <div class="card-header">
        Registro
      </div>
      <div class="card-body">
        <form action="/users/signup" method="POST">
          <div class="form-group">
            <input
              type="text"
              class="form-control"
              name="name"
              placeholder="Name"
              value="{{name}}"
            />
          </div>
          <div class="form-group">
            <input
              type="email"
              class="form-control"
              name="email"
              placeholder="Email"
              value="{{email}}"
            />
          </div>
          <div class="form-group">
            <input
              type="password"
              class="form-control"
              name="password"
              placeholder="Password"
              value="{{password}}"
            />
          </div>
          <div class="form-group">
            <input
              type="password"
              class="form-control"
              name="confirm_password"
              placeholder="Confirm Password"
              value="{{confirm_password}}"
            />
          </div>
          <button class="btn btn-primary btn-block">
            Registrarse
          </button>
        </form>
      </div>
    </div>
  </div>
</div>
```

Ilustración 81 Código Formulario Registro de usuarios

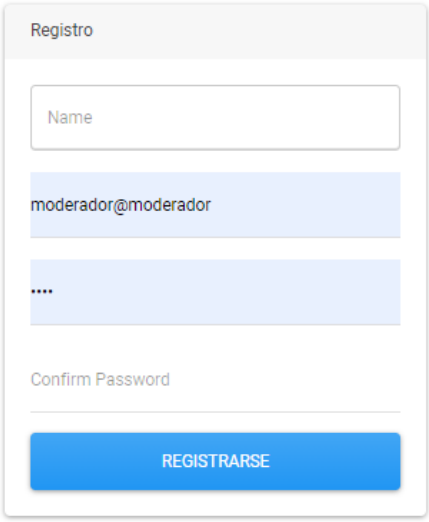


Ilustración 82 Vista formulario registro de usuarios

6.1.3.3 Inicio de sesión

En este apartado vamos a mostrar los formularios de Inicio de sesión (*Ilustración 83 - 84)*, donde el usuario accederá al sistema con sus credenciales

```
<div class="row">
  <div class="col-md-4 mx-auto">
    <div class="card mt-4 text-center">
      <div class="card-header">
        <h1 class="h4">
          Inicio de sesión
        </h1>
      </div>
      <img src="/img/logo.png" class="mx-auto d-block logo m-" alt="Logo" />
      <div class="card-body">
        <form action="/users/signin" method="POST">
          <div class="form-group">
            <input
              type="email"
              class="form-control"
              name="email"
              placeholder="Email"
              autofocus
            />
          </div>
          <div class="form-group">
            <input
              type="password"
              class="form-control"
              name="password"
              placeholder="Password"
            />
          </div>
          <button class="btn btn-primary btn-block">
            Iniciar sesión
          </button>
        </form>
      </div>
    </div>
  </div>
</div>
```

Ilustración 84 Código Formulario Inicio de sesión

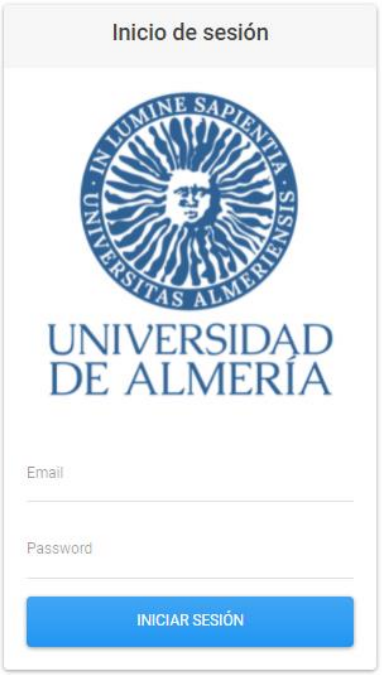


Ilustración 83 Vista Formulario Inicio de Sesión

6.1.3.4 Becas-User

Como se puede observar tenemos una vista simple y clara del contenido resumido de las becas (*Ilustración 85 - 86)*, donde con un simple vistazo podemos ver los distintos campos que nos interesen, también disponemos de 2 botones, uno de ellos nos redirige a la Beca con todos los campos completos. El siguiente botón será el que usaremos si queremos solicitar una Beca, lo que nos llevará a un formulario de solicitud.

```
<table class="table">
  <thead>
    <tr>

      <th scope="col">Título</th>
      <th scope="col">Importe</th>
      <th scope="col">Tipo de Estudiante</th>
      <th scope="col">Convocatorias</th>
      <th scope="col">Nº de Ayudas</th>
      <th scope="col">Tipo de Beca</th>
      <th scope="col">Ver Beca</th>
      <th scope="col">Solicitar Beca</th>

    </tr>
  </thead>

  <tbody>
    {{#each becas}}
      <tr>

        <td>{{title}}</td>
        <td>{{importe}} €</td>
        <td>{{estu}}</td>
        <td>{{conv}}</td>
        <td>{{ayuda}}</td>
        <td>{{tipoBeca}}</td>
        <td>
          <a href="/becas/get-beca/{{_id}}"><button
              type="button"
              class="btn btn-primary btn-sm"
            >Ver</button></a></td>
        <td>
          <a href="/solicitud/add/{{_id}}"><button
              type="button"
              class="btn btn-primary btn-sm"
            >Solicitar</button></a></td>
      </tr>
    {{/each}}
  </tbody>
</table>
```

Ilustración 85 Código Beca-User

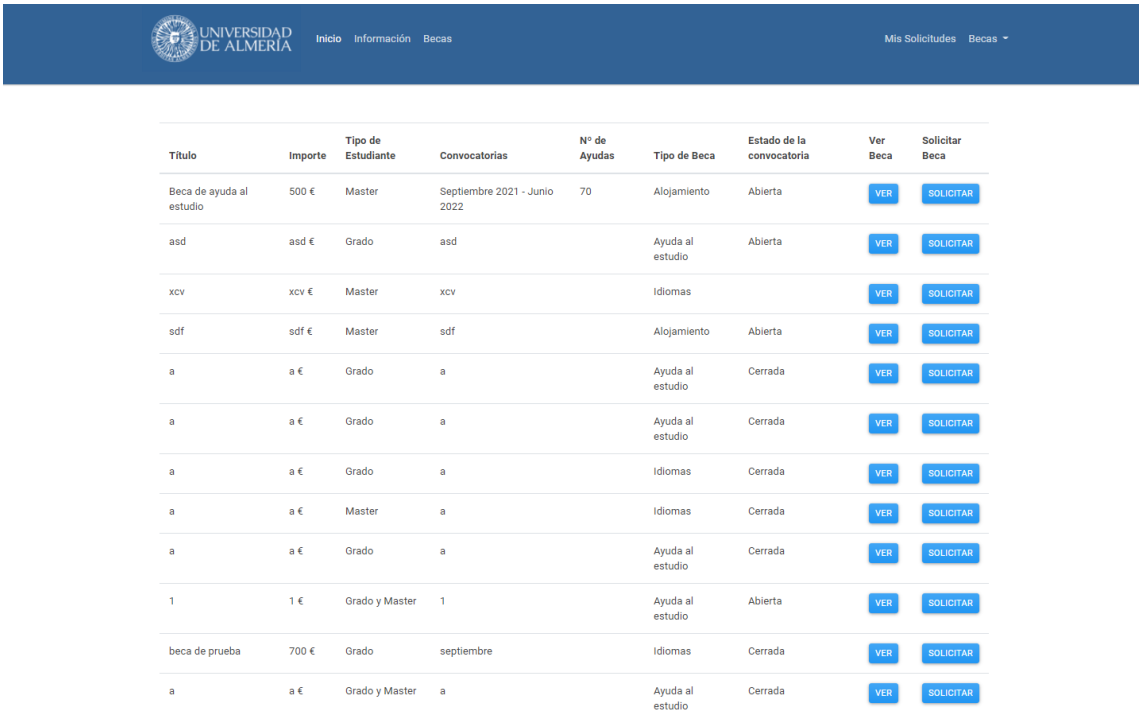


Ilustración 86 Vista Beca-user

6.1.3.5 Ver Beca

Si pulsamos sobre el botón de "Ver "nos redirigirá a la siguiente página (*Ilustración 87 - 88)*:

```
<h3>Ver Becas</h3>
<br />
<div class="row justify-content-md-center">

  <div class="row justify-content-center align-items-center">
    <div class="card">
      <div class="card-body">
        <h4 class="card-title d-flex justify-center align-items-center">
          Título:
          {{beca.title}}
        </h4>
        <p>Convoca: {{beca.convoca}}</p>
        <p>Importe: {{beca.importe}} €</p>
        <p>Tipo de Estudiante: {{beca.estu}} </p>
        <p>Convocatorias: {{beca.conv}} </p>
        <p>Nº de Ayudas ofertadas: {{beca.NumAyudas}} </p>
        <p>Requisitos: {{beca.requisitos}}</p>
        <p>Documentación necesaria: {{beca.documentacion}}</p>
        <p>Fecha de inicio de la publicación: {{beca.fechaIniPub}} </p>
        <p>Fecha de fin de la publicación: {{beca.fechaFinPub}} </p>
        <p>Presupuesto: {{beca.presTotal}} €</p>
        <p>Fecha de Resolución: {{beca.fechaReso}}</p>
        <p>Tipo de Beca: {{beca.tipoBeca}}</p>
        <p>Descripción: {{beca.description}} </p>
        <p>Estado de la convocatoria: {{beca.estado}} </p>
        <p>Financiado por la entidad: {{beca.financiacion}} </p>

      </div>
    </div>
  </div>

</div>
```

Ilustración 87 Código Ver Beca

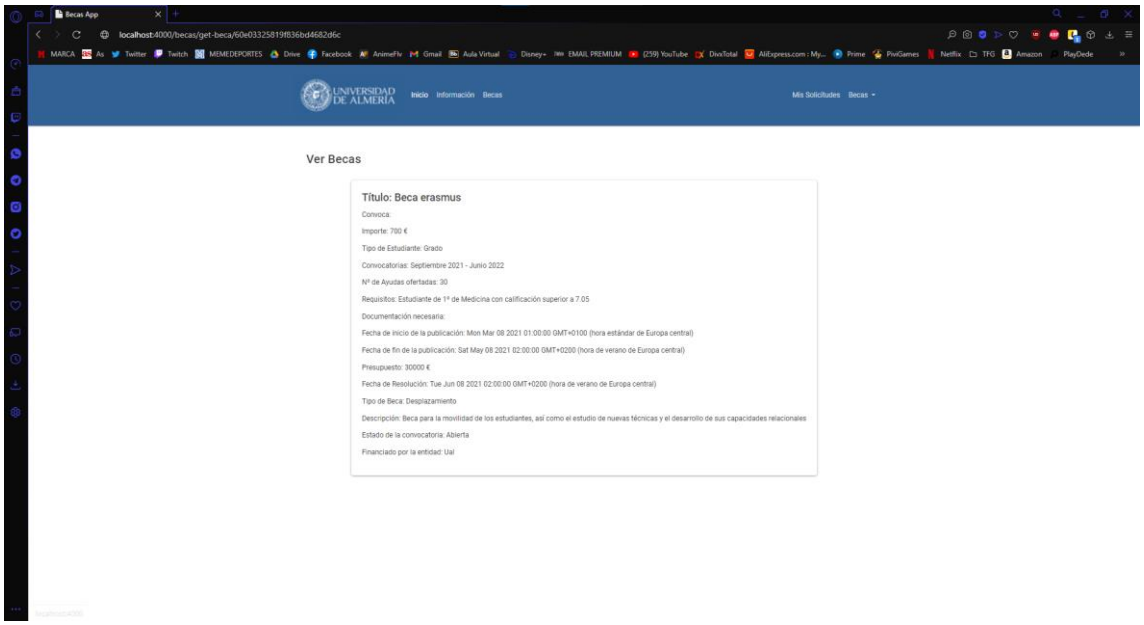


Ilustración 88 Vista Ver Beca

6.1.3.6 Solicitar Beca

Si queremos solicitar la Beca nos llevará al siguiente formulario (*Ilustración 89 - 91)* donde facilitando unos datos podremos solicitarla

```
<h2>Solicitar Beca</h2>

<div class="row justify-content-md-center">

  <div class="row justify-content-center align-items-center">
    <div class="card">
      <div class="card-body">
        <h4 class="card-title d-flex justify-center align-items-center">
          Título:
          {{beca.title}}
        </h4>
        <p>Importe: {{beca.importe}} €</p>
        <p>Tipo de Estudiante: {{beca.estu}} </p>
        <p>Convocatorias: {{beca.conv}} </p>
        <p>Requisitos: {{beca.requisitos}}</p>
        <p>Fecha de Resolución: {{beca.fechaReso}}</p>
        <p>Tipo de Beca: {{beca.tipoBeca}}</p>
        <p>Descripción: {{beca.description}} </p>
      </div>
    </div>
  </div>
</div>
<br />
<div class="container">
  <form action="/solicitud/new-solicitud" method="POST">
    <div class="row">
      <div class="col-25">
        <label for="fname">Titulo de la Beca solicitada</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="titleBeca"
          name="titleBeca"
          value="{{beca.title}}"
        />
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="fname">Nombre del solicitante</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="nombre"
          name="nombre"
          placeholder="Nombre y Apellidos..."
          required="true"
        />
      </div>
    </div>
```

Ilustración 89 Código Solicitar Beca 1

```
    <div class="row">
      <div class="col-25">
        <label for="fname">DNI</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="dni"
          name="dni"
          placeholder="DNI..."
          required="true"
        />
      </div>
    </div>

    <div class="row">
      <div class="col-25">
        <label for="estu">Tipo de estudiante</label>
      </div>
      <div class="col-75">
        <select id="tipoAlumno" name="tipoAlumno">
          <option value="Grado">Grado</option>
          <option value="Master">Master</option>
          <option value="Doctorado">Doctorado</option>
          <option value="Investigacion">Investigación</option>
        </select>
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="fname">Matriculado en</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="matriculado"
          name="matriculado"
          placeholder="Estudios en los que estas matriculado/a..."
          required="true"
        />
      </div>
    </div>
    <br />
    <div class="row">
      <input type="submit" value="Solicitar" />
    </div>
  </form>
</div>
```

Ilustración 90 Código Solicitar Beca 2

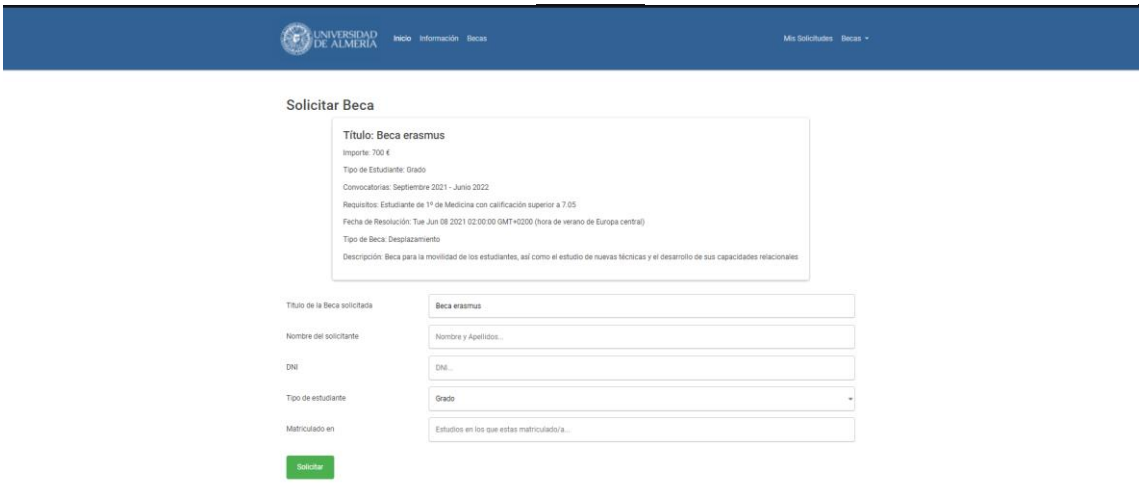


Ilustración 91 Vista Solicitar Beca

6.1.3.7 Ver Mis solicitudes

Dentro de cada usuario podremos ver un apartado donde nos llevará a visualizar todas las becas que hemos solicitado (*Ilustración 92 - 93)*, a continuación, se muestra un breve ejemplo de cómo se visualizarán

```
<h3>Mis Solicitudes</h3>
<br />
{{#each solicitud}}
<div class="row">
<div class="col-md-4" style="margin-bottom: 30px;">
    <div class="card">
      <div class="card-body">
      <h4 class="card-title d-flex justify-center align-items-center">
        Titulo de la Beca:
        {{titleBeca}}
      <h4 class="card-title d-flex justify-center align-items-center">
        Nombre:
        {{nombre}}
      </h4>
      <p>Tipo Alumno: {{tipoAlumno}}</p>
      <p>Dni: {{dni}} </p>
      <p>Matriculado en: {{matriculado}} </p>
    </div>
  </div>
 </div>

</div>
{{/each}}
```

Ilustración 93 Código ver solicitudes

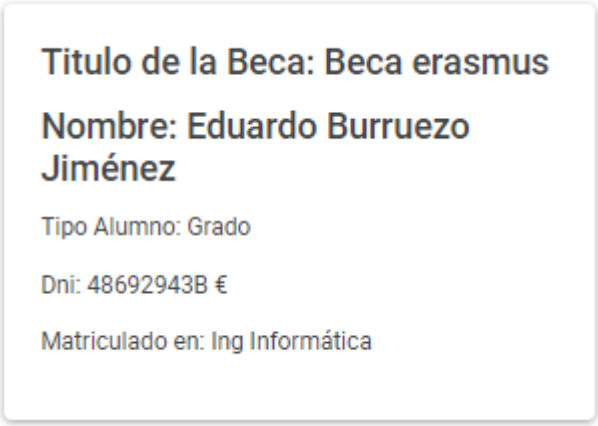


Ilustración 92 Vista Ver solicitudes

6.1.3.8 Becas-admin

Vamos a mostrar el apartado de administrador (*Ilustración 94 - 95)*, donde tendremos las distintas funciones tales como editar, eliminar o agregar becas, así como, gracias a Power Bi *[13]* ver gráficas con distintos datos acerca de las becas y sus componentes.

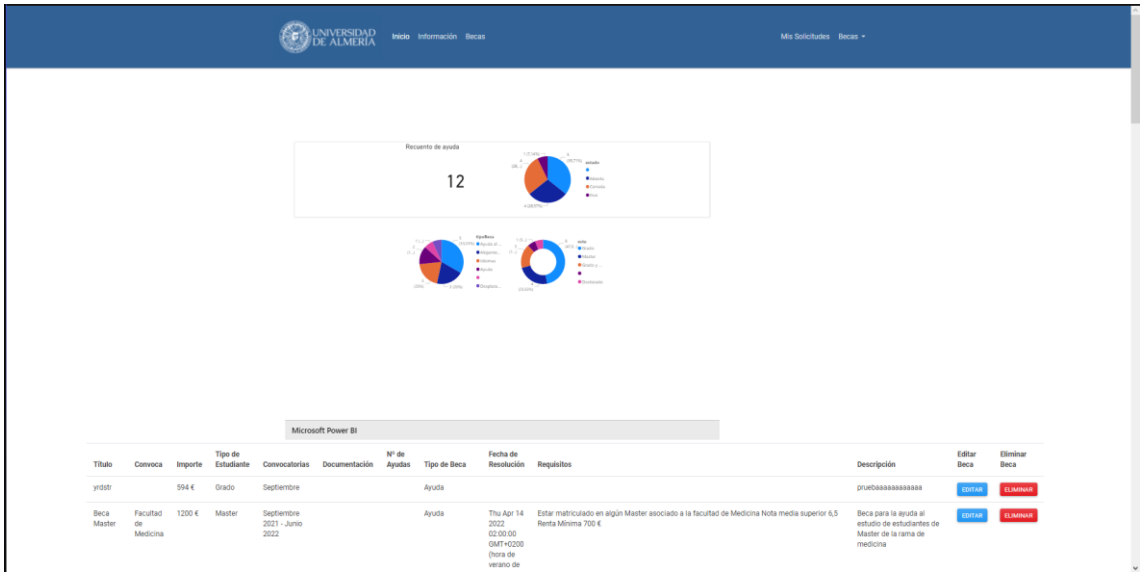


Ilustración 94 Vista Becas-admin

```
<table class="table" style="margin-left: -35%;">
  <thead>
    <tr>

      <th scope="col">Título</th>
      <th scope="col">Convoca</th>
      <th scope="col">Importe</th>
      <th scope="col">Tipo de Estudiante</th>
      <th scope="col">Convocatorias</th>
      <th scope="col">Documentación</th>
      <th scope="col">Nº de Ayudas</th>
      <th scope="col">Tipo de Beca</th>
      <th scope="col">Fecha de Resolución</th>
      <th scope="col">Requisitos</th>
      <th scope="col">Descripción</th>
      <th scope="col">Editar Beca</th>
      <th scope="col">Eliminar Beca</th>

    </tr>
  </thead>

  <tbody>
    {{#each becas}}
      <tr>
        <td>{{title}}</td>
        <td>{{encargado}}</td>
        <td>{{importe}} €</td>
        <td>{{estu}}</td>
        <td>{{conv}}</td>
        <td>{{documentacion}}</td>
        <td>{{ayuda}}</td>
        <td>{{tipoBeca}}</td>
        <td>{{fechaReso}}</td>
        <td>{{requisitos}}</td>
        <td>{{description}}</td>
          <td>
            <a href="/becas/edit/{{_id}}">
            <button type="button" class="btn btn-primary  btn-block btn-sm">Editar</button></form></td>
          <td>

            <form action="/becas/delete/{{_id}}?_method=DELETE" method="POST">
              <input type="hidden" name="_method" value="DELETE" />
              <button
                type="submit"
                class="btn btn-danger btn-block btn-sm"
              >Eliminar</button></form></td>

      </tr>
    {{/each}}
<iframe width="800" height="636" src="https://app.powerbi.com/view?r=eyJrIjoiM2IyYmE3ZWUtNTA1OS00ZjgzLWE5NTMtMWViNGU2OWFlNjY2IiwidCI6IjY2MDBhZTk3LWEwZTktNDdhMC05MTRkLWQ1NWY2ZDFhOGY1YSIsImMiOjh9&pageName=ReportSection" frameborder="0" allowFullScreen="true"></iframe>
    </tbody>
</table>
```

Ilustración 95 Código Becas-admin

6.1.3.9 Editar Becas

Si un administrador desea modificar una beca (*Ilustración 97 - 100)*, ya sea porque debe cambiar algún dato o simplemente cerrar el estado de la convocatoria deberá pasar por el formulario de edición de Becas que se muestra a continuación, donde le aparecerán los campos rellenos con los datos de la beca escogida y desde ahí podrá modificarlos

```
<h3>Editar beca</h3>
<br>
<form action="/becas/edit-beca/{{beca._id}}?_method=PUT" method="POST" enctype="multipart/form-data">
<input type="hidden" name="_method" value="PUT">
  <div class="container">
  <form action="/becas/new-beca" method="POST">
  <div class="row">
    <div class="col-25">
      <label for="fname">Título de la beca</label>
    </div>
    <div class="col-75">
      <input type="text" id="title" name="title" value="{{beca.title}}">
    </div>
  </div>
  <div class="row">
    <div class="col-25">
      <label for="fname">Institución que convoca la Beca</label>
    </div>
    <div class="col-75">
      <input type="text" id="importe" name="importe" value="{{beca.encargado}}">
    </div>
  </div>
  <div class="row">
    <div class="col-25">
      <label for="fname">Importe</label>
    </div>
    <div class="col-75">
      <input type="text" id="importe" name="importe" value="{{beca.importe}}">
    </div>
  </div>
    <div class="row">
    <div class="col-25">
      <label for="estu">Tipo de estudiante</label>
    </div>
    <div class="col-75">
      <select id="estu" name="estu">
        <option value="{{beca.estu}}">{{beca.estu}}</option>
        <option value="Grado">Grado</option>
        <option value="Master">Master</option>
        <option value="Grado y Master">Grado y Master</option>
        <option value="Doctorado">Doctorado</option>
        <option value="Investigacion">Investigación</option>
        <option value="Titulos Própios">Titulos Própios</option>
      </select>
    </div>
  </div>
  <div class="row">
    <div class="col-25">
      <label for="fname">Convocatoria</label>
    </div>
    <div class="col-75">
      <input type="text" id="conv" name="conv" value="{{beca.conv}}">
```

Ilustración 97 Código Formulario Editar Beca 1

```
<div class="row">
  <div class="col-25">
    <label for="lname">Nº de ayudas ofertadas</label>
  </div>
  <div class="col-75">
    <input type="text" id="NumAyudas" name="NumAyudas" value="{{beca.NumAyudas}}">
  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="tipoBeca">Tipo de Beca</label>
  </div>
  <div class="col-75">
    <select id="tipoBeca" name="tipoBeca">
      <option value="{{beca.tipoBeca}}">{{beca.tipoBeca}}</option>
      <option value="Ayuda">Ayuda al estudio</option>
      <option value="Alojamiento">Alojamiento</option>
      <option value="Idiomas">Idiomas</option>
      <option value="Deportes">Deportes</option>
      <option value="Desplazamiento">Desplazamiento</option>
    </select>
  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="lname">Fecha de inicio de la publicación</label>
  </div>
   <div class="col-75" style="font-size:large">
    <datagrid>{{beca.fechaIniPub}}</datagrid>
    <input type="date" id="fechaIniPub" name="fechaIniPub" value="{{beca.fechaIniPub}}">

  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="lname">Fecha de fin de la publicación</label>
  </div>
   <div class="col-75" style="font-size:large">
    <datagrid>{{beca.fechaFinPub}}</datagrid>
    <input type="date" id="fechaFinPub" name="fechaFinPub" value="{{beca.fechaFinPub}}">
  </div>
</div>
  <div class="row">
  <div class="col-25">
    <label for="lname">Presupuesto total</label>
  </div>
  <div class="col-75" style="font-size:large">
    <input type="number" id="presTotal" name="presTotal" value="{{beca.presTotal}}">
  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="subject">Requisitos</label>
  </div>
  <div class="col-75">
```

Ilustración 96 Código Formulario Editar Beca 2

```
<div class="row">
  <div class="col-25">
    <label for="subject">Descripción</label>
  </div>
  <div class="col-75">
    <textarea id="description" name="description" value ="{{beca.description}}" style="height:100px">{{beca.description}}</textarea>
  </div>
</div>
 <div class="row">
  <div class="col-25">
    <label for="lname">Resolución</label>
  </div>
  <div class="col-75" style="font-size:large"> <datagrid>{{beca.fechaReso}}</datagrid>
    <input type="date" id="fechaReso" name="fechaReso" value="{{beca.fechaReso}}">
  </div>
</div>
   <div class="row">
    <div class="col-25">
      <label for="lname">Estado de la convocatoria</label>
    </div>
    <div class="form-check"> <datagrid>{{beca.estado}}                    </datagrid>

      <input
        class="form-check-input"
        type="radio"
        name="estado"
        id="estado"
        value="Abierta"
      />
      <label class="form-check-label" for="flexRadioDefault1">
        Abierta
      </label>
    </div>
    <div class="form-check">
      <input
        class="form-check-input"
        type="radio"
        name="estado"
        id="estado"
        value="Cerrada"
        checked
      />
      <label class="form-check-label" for="flexRadioDefault2">
        Cerrada
      </label>
    </div>
  </div>
      <div class="row">
    <div class="col-25">

      <label for="lname">Financiación externa de la universidad</label>

    </div>
```

Ilustración 98 Código Formulario Editar Beca 3

```
    <div class="form-check">  <datagrid>{{beca.financiacion}}                    </datagrid>
      <input
        class="form-check-input"
        type="radio"
        name="financiacion"
        id="financiacion"
        value="Ual"
      />
      <label class="form-check-label" for="flexRadioDefault1">
        Ual
      </label>
    </div>
    <div class="form-check">
      <input
        class="form-check-input"
        type="radio"
        name="financiacion"
        id="financiacion"
        value="Externa"
        checked
      />
      <label class="form-check-label" for="flexRadioDefault2">
        Externa
      </label>
    </div>
  </div>
  <br>
<div class="row">
    <div class="col-25">
      <label for="lname">Adjuntar documentos</label>
    </div>
    <div class="col-75">
    <div class="form-group ">
      <div class="input-group ">
        <div class="custom-file ">
          <input type="file" name="file" class="custom-file-input" id="inputGroupFile"
            aria-describedby="inputGroupFileAddon"  />
          <label class="custom-file-label border" for="inputGroupFile">Seleccionar archivos</label>
        </div>
      </div>
    </div>
    </div>
  </div>
 <br>
 <div class="row">
   <input type="submit" value="Editar">
 </div>
 </form>
</div>
```

Ilustración 99 Código Formulario Editar Beca 4

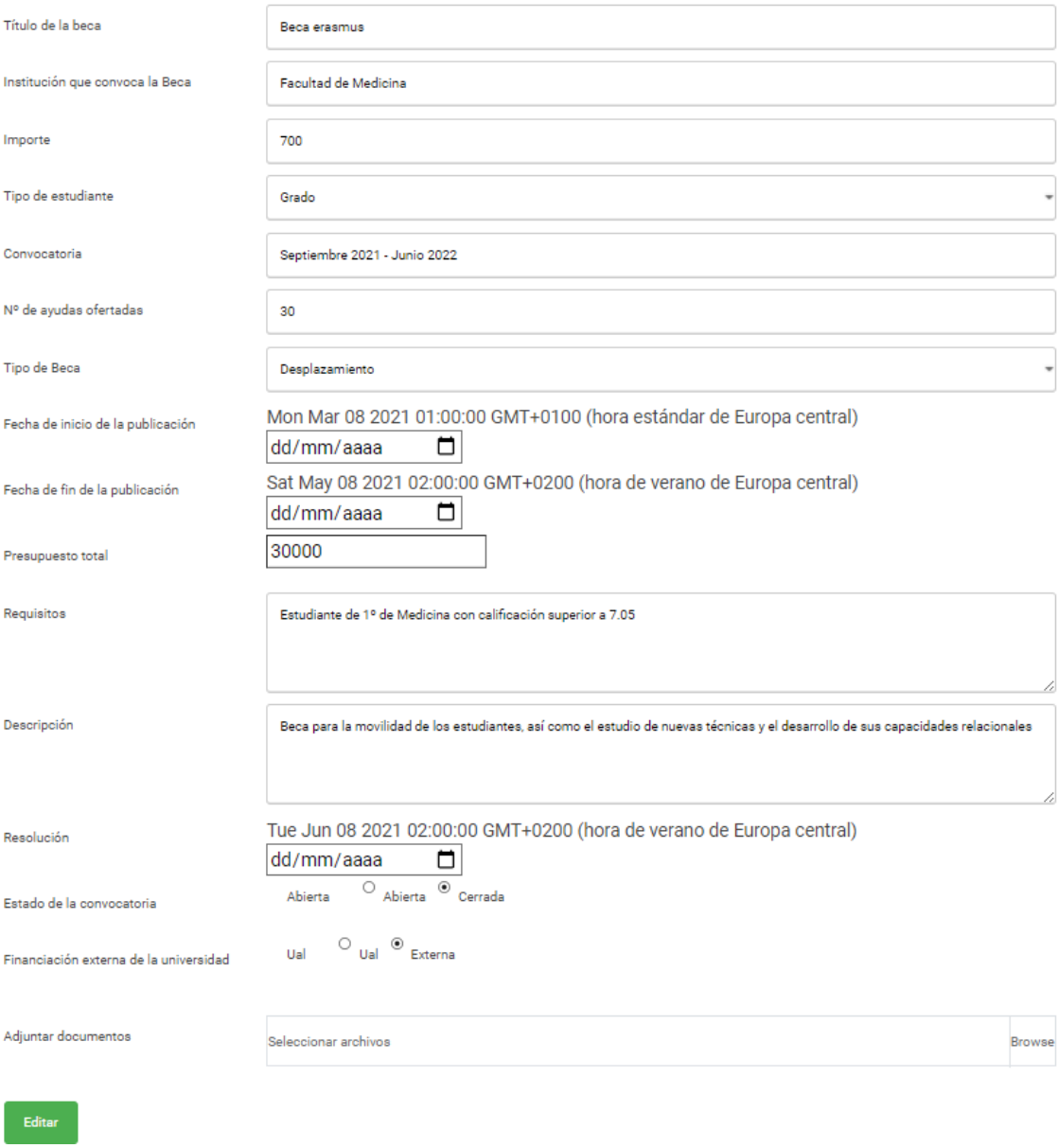


Ilustración 100 Vista Formulario Edición de Becas

6.1.3.10 Añadir Beca

Para añadir Becas (*Ilustración 102 - 106)*, una vez iniciada sesión el usuario deberá desplegar el menú y dar sobre la opción de añadir becas, lo que le llevará al siguiente formulario donde deberá rellenar los campos mostrados y en caso de ser necesario podrá subir documentos al servidor

```
<h2>Añadir Nueva Beca</h2>
<br />

<div class="container">
  <form action="/becas/new-beca" method="POST" enctype="multipart/form-data">
    <div class="row">
      <div class="col-25">
        <label for="fname">Título de la beca</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="title"
          name="title"
          placeholder="Nombre de la beca..."
          required="true"
        />
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="fname">Institución que convoca la Beca</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="encargado"
          name="encargado"
          placeholder="Institución que convoca la beca..."
          required="true"
        />
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="fname">Importe</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="importe"
          name="importe"
          placeholder="Importe de la beca..."
          required="true"
        />
      </div>
    </div>
```

Ilustración 102 Código Formulario Añadir Becas 1

```
    <div class="row">
      <div class="col-25">
        <label for="estu">Tipo de estudiante</label>
      </div>
      <div class="col-75">
        <select id="estu" name="estu">
          <option value="Grado">Grado</option>
          <option value="Master">Master</option>
          <option value="Grado y Master">Grado y Master</option>
          <option value="Doctorado">Doctorado</option>
          <option value="Investigacion">Investigación</option>
          <option value="Titulos Própios">Titulos Própios</option>
        </select>
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="fname">Convocatoria</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="conv"
          name="conv"
          placeholder="Convocatoria de la beca..."
          required="true"
        />
      </div>
    </div>
    <div class="row">
      <div class="col-25">
        <label for="lname">Nº de ayudas ofertadas</label>
      </div>
      <div class="col-75">
        <input
          type="text"
          id="NumAyudas"
          name="NumAyudas"
          placeholder="Nº de ayudas ofertadas..."
          required="true"
        />
      </div>
    </div>

    <div class="row">
      <div class="col-25">
        <label for="tipoBeca">Tipo de Beca</label>
      </div>
      <div class="col-75">
        <select id="tipoBeca" name="tipoBeca">
          <option value="Ayuda al estudio">Ayuda al estudio</option>
          <option value="Alojamiento">Alojamiento</option>
          <option value="Idiomas">Idiomas</option>
          <option value="Deportes">Deportes</option>
          <option value="Desplazamiento">Desplazamiento</option>
        </select>
      </div>
    </div>
```

Ilustración 101 Código Formulario Añadir Becas 2

```
<div class="row">
  <div class="col-25">
    <label for="fname">Fecha de inicio de la publicación</label>
  </div>
  <div class="col-75" style="font-size:large">
    <input
      type="date"
      id="fechaIniPub"
      name="fechaIniPub"
      placeholder="Fecha de inicio de de la publicación..."
      required="true"
    />
  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="fname">Fecha de fin de la publicación</label>
  </div>
  <div class="col-75" style="font-size:large">
    <input
      type="date"
      id="fechaFinPub"
      name="fechaFinPub"
      placeholder="Fecha de fin de de la publicación..."
      required="true"
    />
  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="fname">Presupuesto total</label>
  </div>
  <div class="col-75" style="font-size:large">
    <input
      type="number"
      id="presTotal"
      name="presTotal"
      placeholder="Presupuesto total..."
      required="true"
    />
  </div>
</div>

<div class="row">
  <div class="col-25">
    <label for="subject">Requisitos</label>
  </div>
  <div class="col-75">
    <textarea
      id="requisitos"
      name="requisitos"
      placeholder="Requisitos del solicitante..."
      style="height:100px"
    ></textarea>
  </div>
</div>
```

Ilustración 103 Código Formulario Añadir Becas 3

```
<div class="row">
  <div class="col-25">
    <label for="subject">Descripción</label>
  </div>
  <div class="col-75">
    <textarea
      id="description"
      name="description"
      placeholder="Descripción de la beca..."
      style="height:100px"
      required="true"
    ></textarea>
  </div>
</div>
<div class="row">
  <div class="col-25">
    <label for="fname">Resolución</label>
  </div>
  <div class="col-75" style="font-size:large">
    <input
      type="date"
      id="fechaReso"
      name="fechaReso"
      placeholder="Resolución de la Beca..."
      required="true"
    />
  </div>
</div>

<div class="row">
  <div class="col-25">
    <label for="fname">Estado de la convocatoria</label>
  </div>
  <div class="form-check">
    <input
      class="form-check-input"
      type="radio"
      name="estado"
      id="estado"
      value="Abierta"
    />
    <label class="form-check-label" for="flexRadioDefault1">
      Abierta
    </label>
  </div>
  <div class="form-check">
    <input
      class="form-check-input"
      type="radio"
      name="estado"
      id="estado"
      value="Cerrada"
      checked
    />
    <label class="form-check-label" for="flexRadioDefault2">
      Cerrada
    </label>
  </div>
</div>
```

Ilustración 104 Código Formulario Añadir Becas 4

```
      <div class="row">
      <div class="col-25">
        <label for="fname">Financiación externa de la universidad</label>
      </div>
      <div class="form-check">
        <input
          class="form-check-input"
          type="radio"
          name="financiacion"
          id="financiacion"
          value="Ual"
        />
        <label class="form-check-label" for="flexRadioDefault1">
          Ual
        </label>
      </div>
      <div class="form-check">
        <input
          class="form-check-input"
          type="radio"
          name="financiacion"
          id="financiacion"
          value="Externa"
          checked
        />
        <label class="form-check-label" for="flexRadioDefault2">
          Externa
        </label>
      </div>
    </div>
  <div class="row">
      <div class="col-25">
        <label for="fname">Adjuntar documentos</label>
      </div>
      <div class="col-75">
      <div class="form-group ">
        <div class="input-group ">
          <div class="custom-file ">
            <input type="file" name="file" class="custom-file-input" id="inputGroupFile"
              aria-describedby="inputGroupFileAddon"  />
            <label class="custom-file-label border" for="inputGroupFile">Seleccionar archivos</label>
          </div>
        </div>
      </div>
      </div>
    </div>
    </div>
    <br />
    <div class="row">
      <input type="submit" value="Añadir" />
    </div>

  </form>
</div>
```

Ilustración 105 Código Formulario Añadir Becas 5

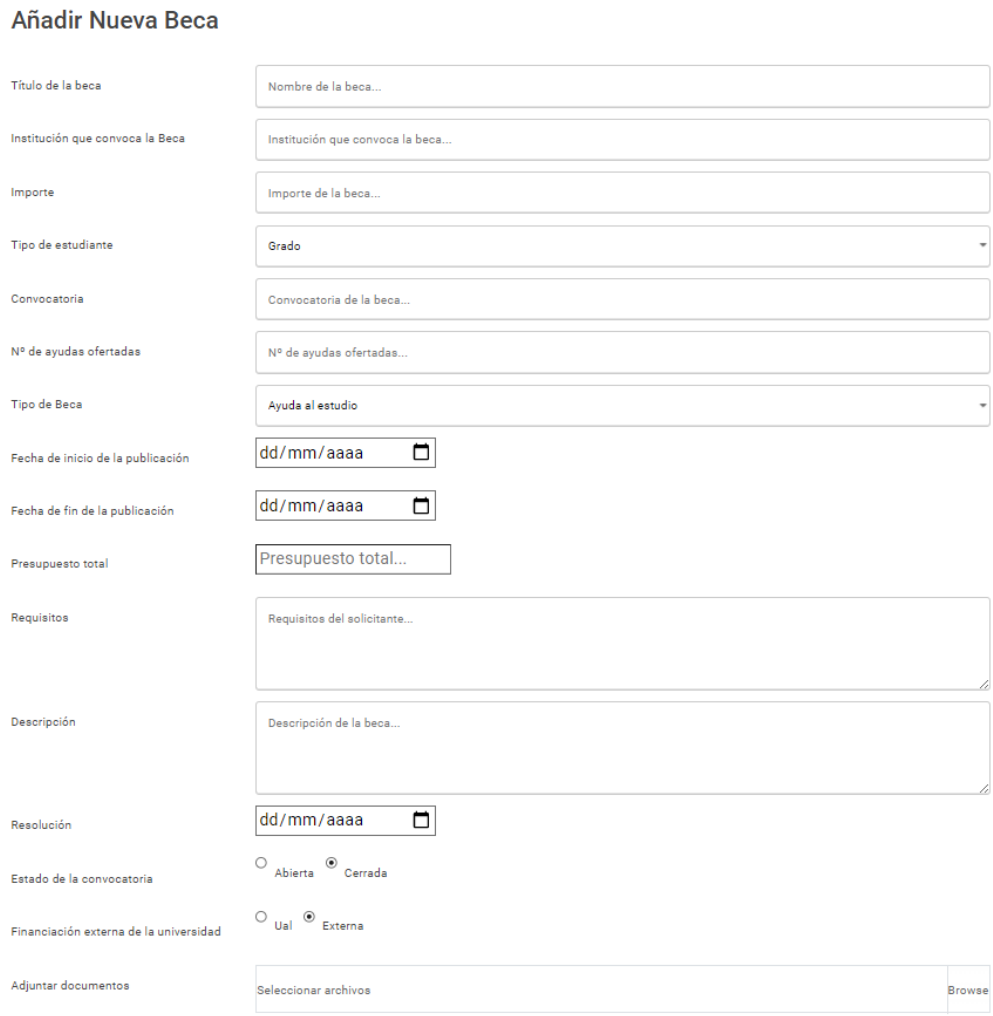


Ilustración 106 Vista Formulario Añadir Becas

6.1.3.11 Alertas y Mensajes a los usuarios

Cuando un usuario realiza una acción el usuario sabrá en todo momento si la acción se ha generado correctamente o por el contrario lanzará un error un mensaje por pantalla de error *(Ilustración 107-109)*.

Ilustración 107 Mensaje de Verificación

Ilustración 108 Mensaje de que hay un campo erróneo

Ilustración 109 Mensaje de Error

7 RESULTADOS

Para sacar resultados reales sobre el sistema se ha decidido crear un formulario con ciertas preguntas relevantes sobre el sistema y pasárselo a un grupo de personas que prueben las distintas funcionalidades. Como podemos observar en el Anexo sobre la usabilidad y la capacidad técnica se preguntan una serie de cuestiones donde los usuarios que testeen la aplicación calificarían varios aspectos de la misma. Una vez resueltos y recibidos nos disponemos a analizar los resultados.

Como conclusión podemos observar que los resultados son bastante buenos, ya que la nota mínima en todos los cuestionarios resueltos es de un 4 y la nota máxima un 5

Si analizamos cada uno de los campos por separados observamos los siguiente:

En cuanto al apartado de presentación observamos que la mayoría de los campos los usuarios que han probado el sistema le han dado una nota de 5 en cada uno de los apartados, por lo que podemos deducir que el sistema posee una buena presentación

El siguiente apartado, el de individualización obtenemos las mismas conclusiones, ya que excepto en un caso en el que uno de los usuarios puso un 4 el resto de los usuarios ha puesto una nota de 5, por lo que llegamos nuevamente a la conclusión de que el apartado de individualización, es correcto.

Para el apartado de interactividad en general como se puede observar tiene unas notas bastante buenas, por lo que la conclusión que podemos sacar a partir de analizar los resultados es que la interactividad de cara al usuario es buena

Si analizamos el apartado del manejo podemos observar que alguno de los usuarios ha calificado el sistema como adecuado, pero en general los resultados han sido muy buenos y parece que el sistema tiene buena acogida, por lo que se entiende que el sistema es de fácil manejo.

Analizando los contenidos se observa que los usuarios han mostrado una buena opinión acerca de los mismos, por lo que calificaremos los contenidos de muy buenos

En cuanto la ayuda los usuarios han dejado claro tras sus opiniones que la ayuda que han recibido es la correcta.

Por la parte del funcionamiento y eficacia los resultados obtenidos son óptimos.

En cuanto al compromiso el programa ha respondido a las expectativas de los usuarios.

Los usuarios en cuanto a los comentarios libres han echado en falta que haya un buscador a parte de un poco más de color en el sistema, estas son cosas que estudiaremos para poder aplicarlas en un futuro

A continuación, procedemos a mostrar una serie de tablas donde nos disponemos a analizar los resultados obtenidos a través de unas gráficas para que se puedan apreciar mejor los resultados obtenidos:

A continuación se presenta una gráfica comparativa de los resultados de las valoraciones *(Tabla 27)*

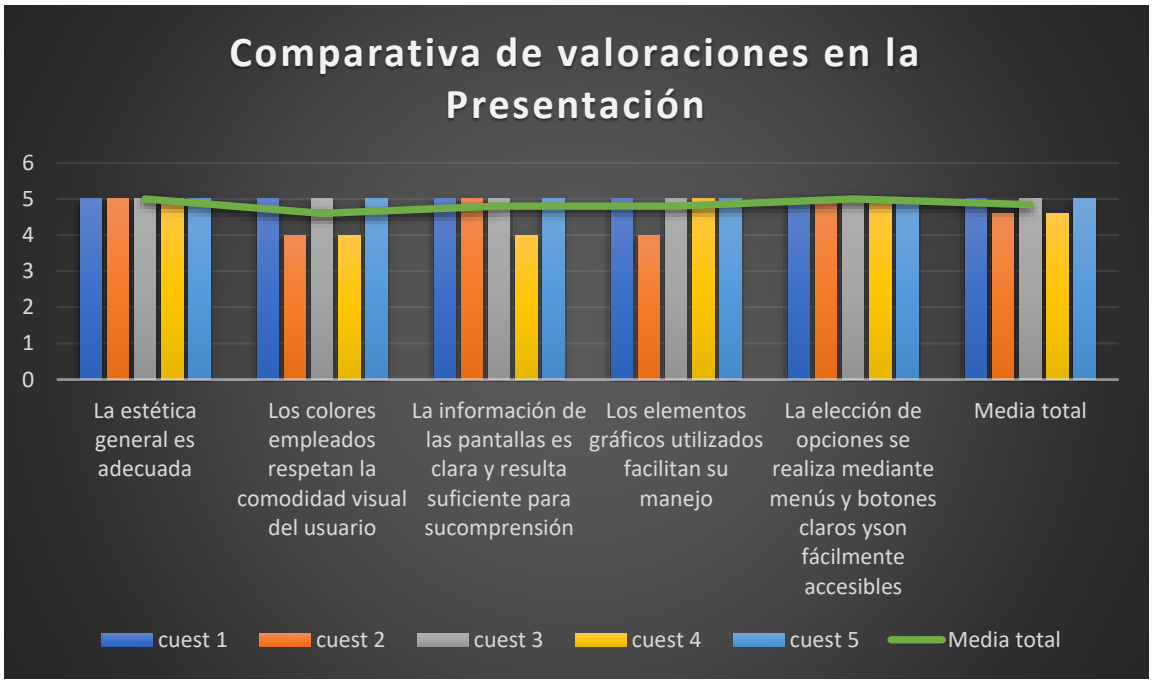


Tabla 27 Valoraciones: Presentación

Procedemos a mostrar una gráfica comparativa de las valoraciones en la individualización *(Tabla 28)*

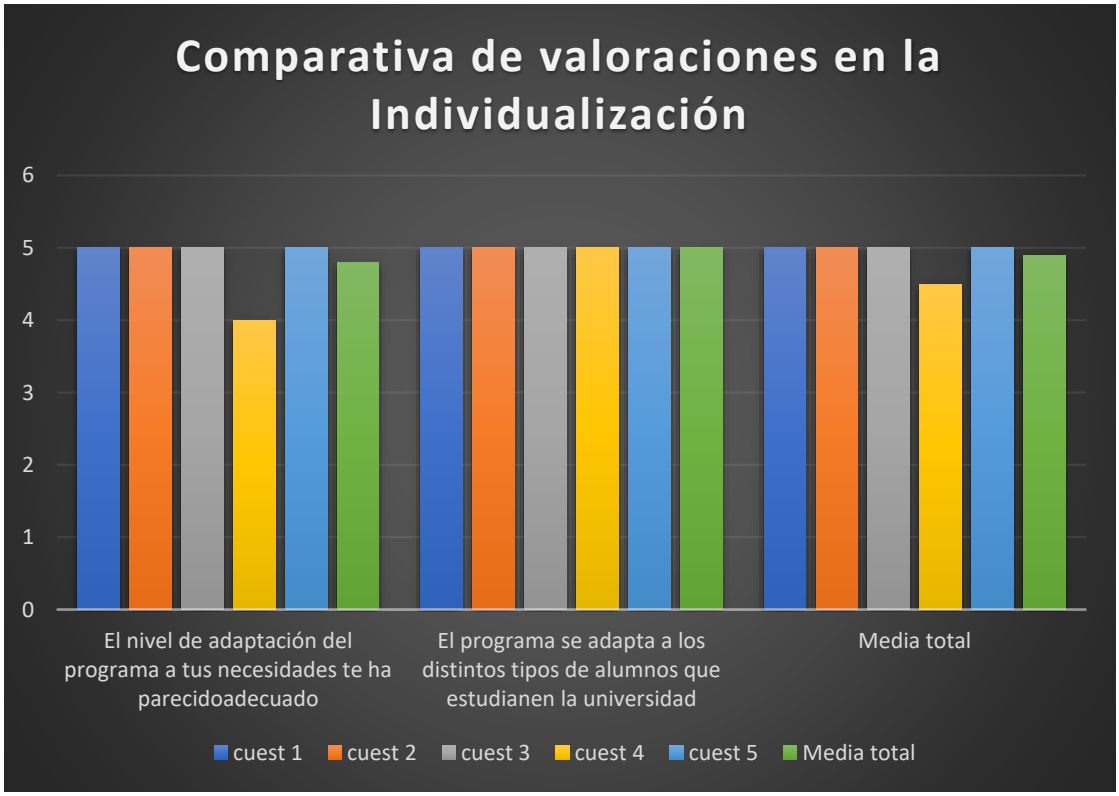


Tabla 28 Valoraciones: Representación

Vamos a mostrar ahora una gráfica comparativa de las valoraciones de la interactividad *(Tabla 29)*

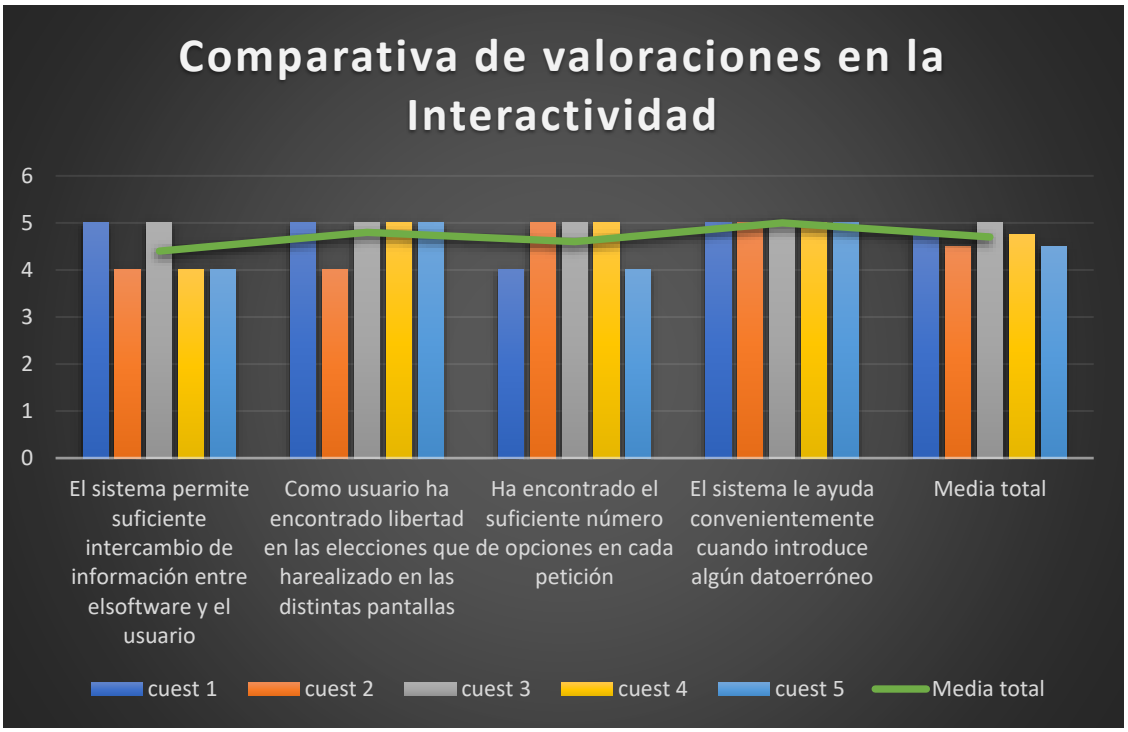


Tabla 29 Valoraciones: Interactividad

Mostramos ahora una gráfica comparativa de las valoraciones en cuanto al manejo *(Tabla 30)*

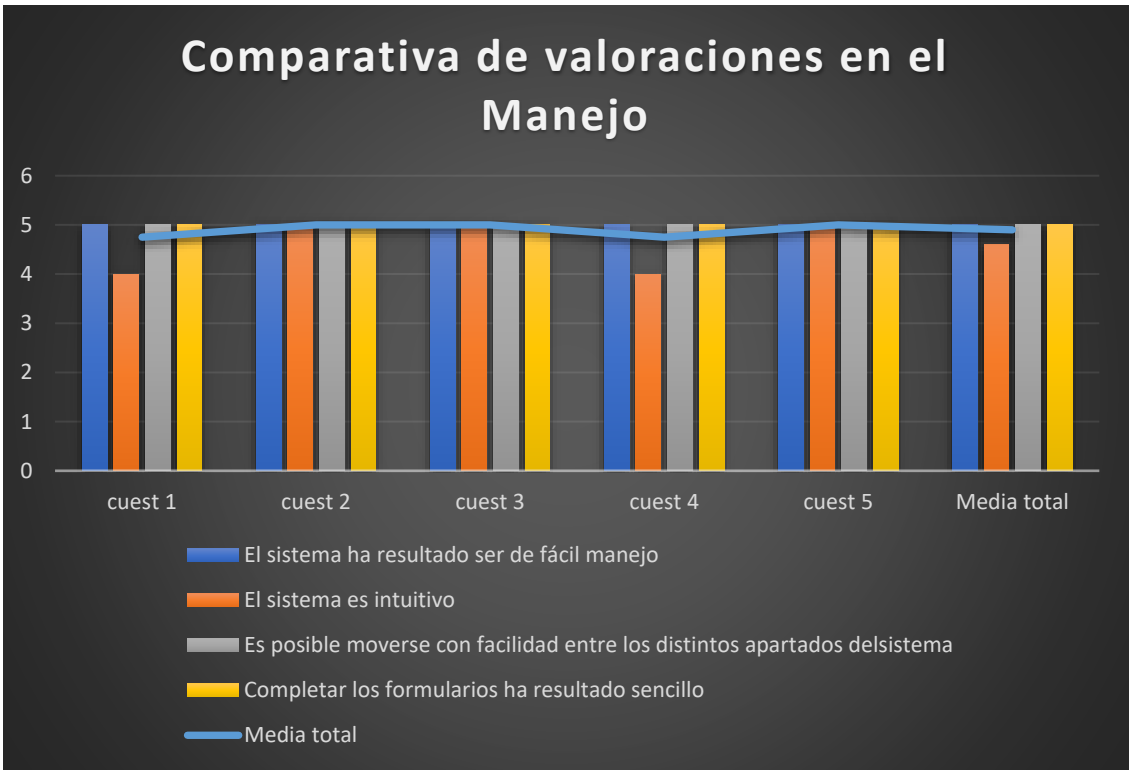


Tabla 30 Valoraciones: Manejo

Se procede a mostrar una gráfica comparativa de las valoraciones de los contenidos *(Tabla 31)*

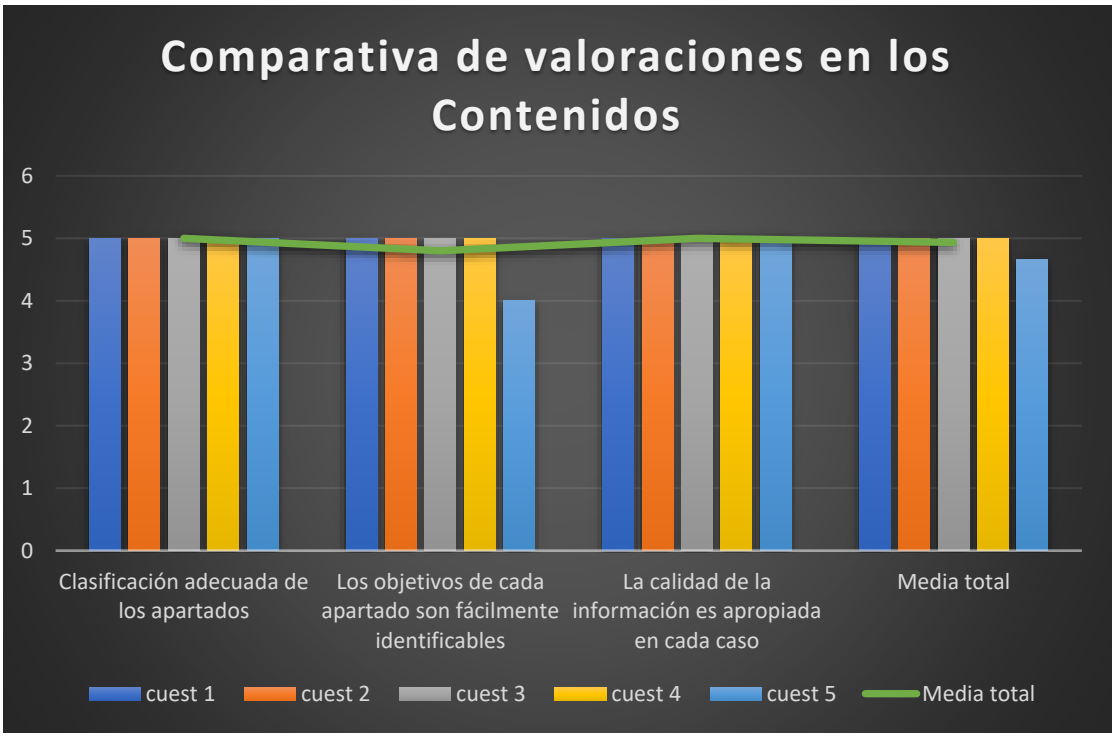


Tabla 31 Valoraciones: Contenidos

Mostramos a continuación una gráfica comparativa de las valoraciones en cuanto a las ayudas *(Tabla 32)*

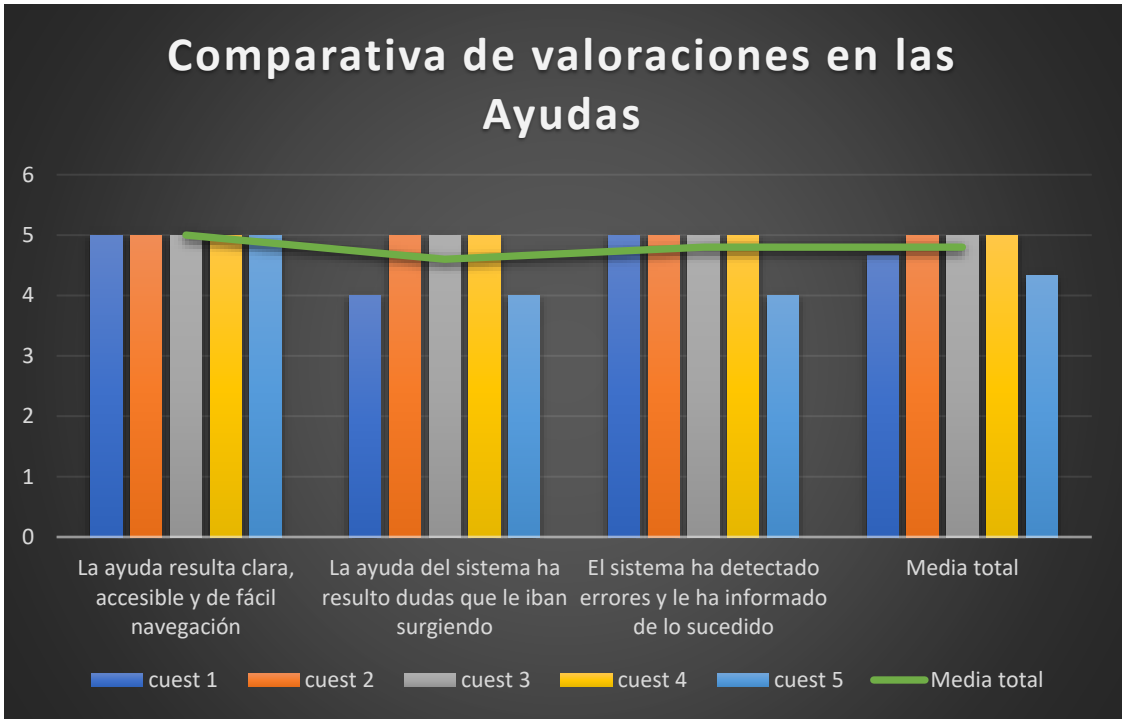


Tabla 32 Valoraciones: Ayudas

Vamos a mostrar una gráfica comparativa de las valoraciones en relación al funcionamiento y eficiencia del sistema *(Tabla 33)*

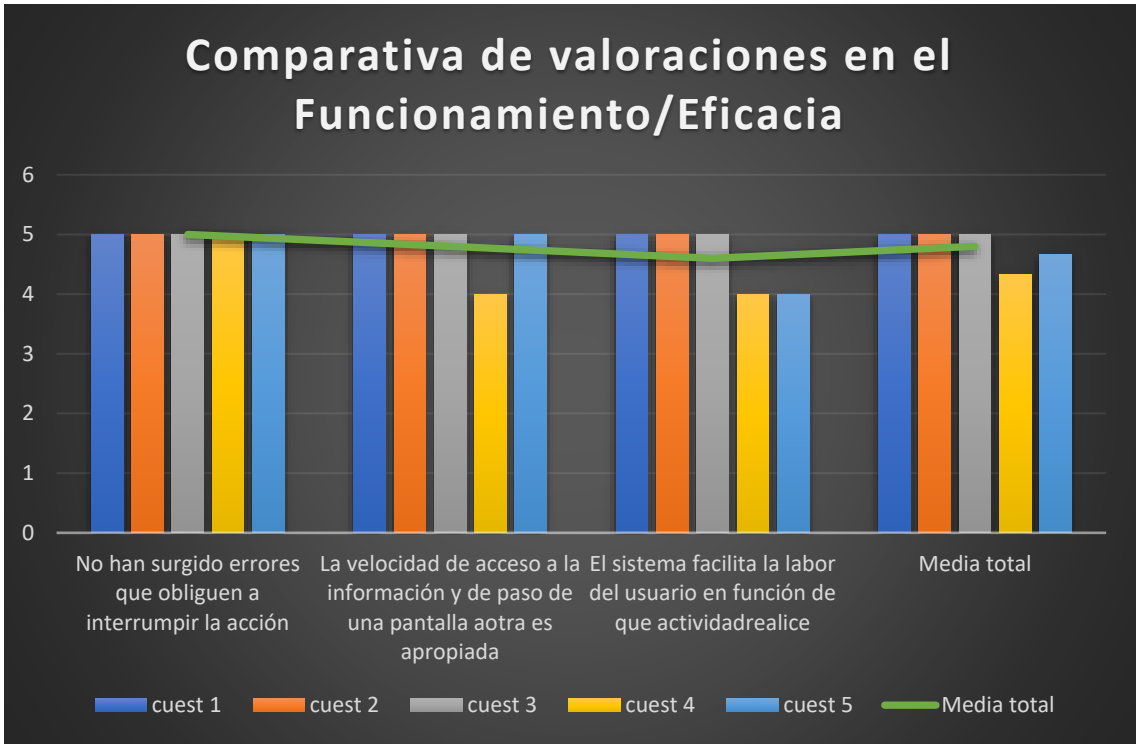


Tabla 33 Valoraciones: Funcionamiento/Eficacia

Por último mostramos una gráfica comparativa de las valoraciones en cuanto al compromiso *(Tabla 34)*

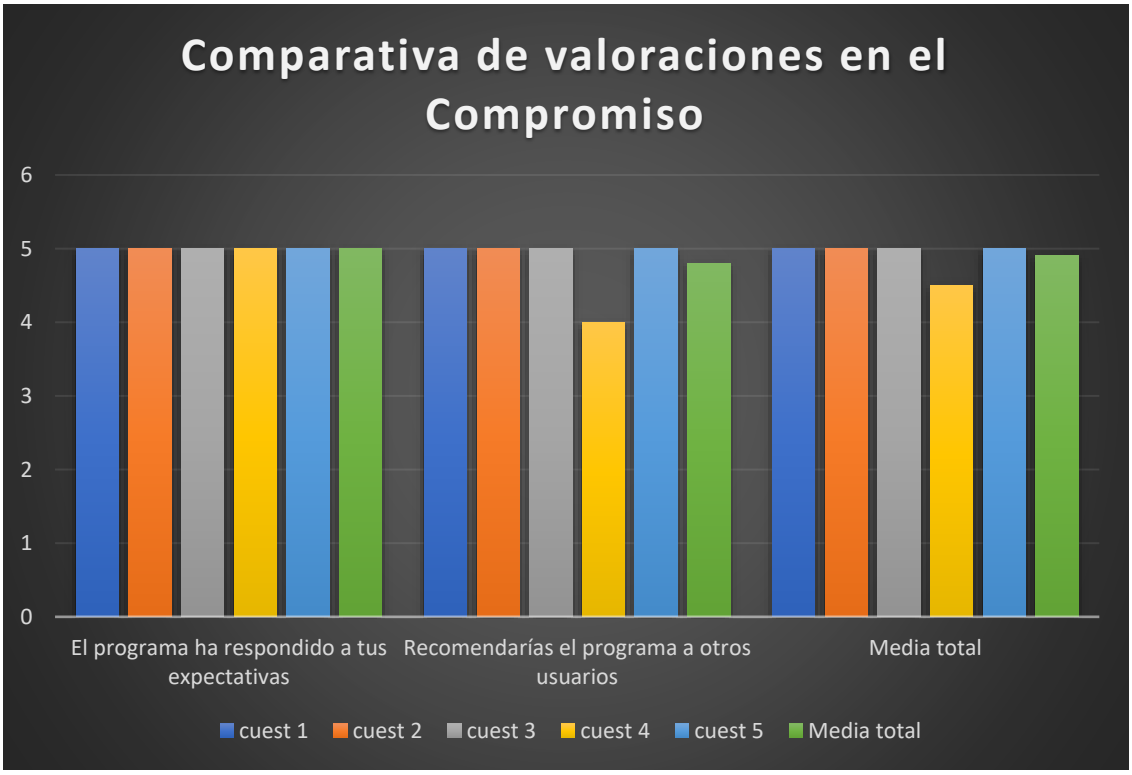


Tabla 34 Valoraciones: Compromiso

8 CONCLUSIONES

Mediante este trabajo se ha desarrollado una Aplicación Web para la Difusión de Convocatorias de Ayudas Universitarias. Esta será una gran herramienta que ayudará a estudiantes, trabajadores de la universidad y servicios externos a promover y solicitar becas de una manera más clara y sencilla gracias a su fácil acceso. Por otro lado, este sistema constará de una interfaz Web accesible a través de internet.

Este proyecto se ha desarrollado dando posibilidad de futuras implementaciones sin necesidad de depender de un software externo.

Como se ha podido observar, a lo largo de este documento se han cumplido los objetivos propuestos, ya que se ha desarrollado un sistema de gestión Web que contiene las Becas y Ayudas de la Universidad de Almería que muestra la información más relevante de las convocatorias a través de una imagen corporativa, centralizando a su vez la información en una sola web con un sistema accesible, usable y adaptable a cualquiera de los posibles formatos que utilicen los estudiantes.

En sintonía con los objetivos se puede comprobar que se ha desarrollado un sistema de gestión interno que aglutina las Becas y Ayudas que la universidad de Almería pone a disposición de los estudiantes.

También se ha creado un sistema donde la información relevante predomina sobre el resto del sistema, centralizándola en la misma aplicación, lo que facilitará a los estudiantes localizar la información que requieran sin perderse entre cientos de menús innecesarios.

Por otro lado, se ha implementado un sistema con un acceso totalmente público, ya que toda la información se puede consultar de forma libre y sin tener que estar inscrito previamente en la universidad, sin embargo, a la hora de realizar una solicitud sí que se deberá estar registrado. Para esto se ha diseñado un sistema de registro simple y fácil de usar para poder almacenar las solicitudes de cada uno de los usuarios.

El sistema destaca cuales de las convocatorias están abiertas gracias a la tabla de la que dispone, lo que ayuda a su fácil identificación y facilita que se puedan solicitar las becas a las que se pueden optar.

Se almacena un histórico con todas las convocatorias ofertadas ya que el sistema las guarda todas en su sistema para que los usuarios puedan consultar becas de otros años, esto les dará una ligera idea de si son aptos para solicitar la beca o cuál es el número de plazas que se acabaron concediendo otros años.

Cabe destacar la creación de una de las funciones más relevantes del sistema, la implementación de un sistema de gestión de roles. Esta permite a los distintos usuarios acceder a las diferentes áreas del sistema para poder gestionar la información relativa de sus propias convocatorias a través de un control de accesos. Permitiendo a su vez que ciertos usuarios llamados en el sistema como moderadores puedan acceder a ciertas funciones. Este tipo de rol está pensado para que empresas externas de la universidad, a la cual le concederá permisos el administrador del sistema puedan ofertar sus propias becas, con lo que se les dará una gran oportunidad a muchos de los estudiantes y las empresas podrán formar, o ayudar a que se formen, futuros trabajadores.

En conclusión, se han desarrollado prácticamente todos los objetivos expuestos. A través de este proyecto no solo se ha creado un sistema que puede ayudar a muchos estudiantes o a

la administración de la universidad, sino que además se han potenciado mis capacidades con algunas de las tecnologías que ya conocía y a su vez he desarrollado nuevas habilidades gracias al aprendizaje que ha supuesto este proyecto en muchas otras tecnologías.

8.1 DETALLE ESPECÍFICO DE LO QUE SE HA APRENDIDO

Este proyecto significa el fin de una etapa que comenzó hace años, antes incluso que el grado en Ing. Informática y que se pretende que refleje lo aprendido, no solo a través de clases si no de compañeros y experiencias como estudiante.

Todo estudiante ha solicitado una beca en algún momento de su etapa en la universidad como ayuda trampolín para poder dar lo mejor de sí mismo. Ver que el sistema ya implementado en la universidad que proponía muchas facilidades a la hora de solicitar ayudas a los estudiantes podía ser mejor, decidí abarcarlo en cuanto se me ofreció para poder ayudar a toda esa gente, ya no solo que viene detrás de mí, sino que se ha esforzado en ofrecerme las posibilidades de seguir adelante con mis estudios.

Este proyecto me ha servido para aprender nuevas tecnologías y herramientas de desarrollo que están en auge estos últimos años, si no para descubrir las necesidades de una entidad tan grande como es una universidad. Nunca antes había tenido ante mí un reto profesional de estas dimensiones como ha sido este proyecto, ya que he tenido que organizar no solo el tiempo de trabajo si no organizar mis pensamientos e ideas.

En cuanto a lo aprendido técnicamente gracias al proyecto lo más importante ha sido

- **API's:** Durante el desarrollo del proyecto se ha aprendido no solo, cómo se implementa un api, si no la función tan importante que desempeña en un sistema y un proyecto de estas dimensiones
- **MongoDB:** En este proyecto se ha decidido usar Mongodb como base de datos, gracias a este proyecto se ha aprendido sobre cómo usar esta base de datos y sobre todo sobre sus ventajas, ya que al contrario que MySQL esta forma de base de datos se basa en ficheros, lo que facilita mucho las relaciones entre los campos y los datos.
- **Stack MEBN:** Gracias a este trabajo he podido desarrollar mis capacidades de programación y entendimiento de los sistemas full – stacks y las herramientas que lo componen.

8.2 TRABAJO FUTURO

En cuanto al sistema de difusión se podrían implementar algunos módulos más para la correcta difusión de las becas como puede ser un sistema de difusión por perfil de estudiantes, pudiendo llegar las propuestas a los alumnos cuyos requisitos sean más cercanos a lo propuesto directamente al teléfono móvil a través de un SMS o correo electrónico.

Otra posible mejora sería la opción de añadir un buscador en tiempo real, donde los estudiantes pudiesen realizar búsquedas dinámicas cambiando las tablas según los distintos campos que seleccionemos o según las palabras que se escriban en el buscador.

En cuanto al uso de Power Bi también sería conveniente diseñar un cuadro de mandos que permitiera al usuario realizar consultas mediante estadísticas y gráficas de los datos recopilados a través de las becas para poder contemplar las posibilidades de por ejemplo,

modificar los importes o las plazas de algunas de las menos solicitadas o donde siempre se quedan becas sin asignar para poder añadirlas a algunas de las más populares donde más gente se quedaría excluido de poder disfrutar de la beca.

Por último, una posible mejora que sería conveniente trabajar en el futuro sería la de ajustar el sistema con el resto del sistema de la universidad, conectándolo con los datos de las bases de datos internas y sus procesos.

9 BIBLIOGRAFÍA

[1] Bootstrap. (s.f.). Obtenido de https://getbootstrap.com/docs/4.1/getting-started/introduction/

[2] Bush, E. (2016). *Full-Stack JavaScript Development.* Red Sky.

[3] Documentación MongoDB. (s.f.). Obtenido de https://www.mongodb.com/blog/channel/quickstart

[4] Express. (s.f.). Obtenido de https://expressjs.com/es/api.html#express

[5] Node.js. (s.f.). Obtenido de https://nodejs.org/api/documentation.html

[6] CSS. (s. f.). W3School. https://www.w3schools.com/css/default.asp

[7] HTML. (s. f.). W3School. https://www.w3schools.com/html/

[8] JavaScript. (s. f.). W3School. https://www.w3schools.com/js/default.asp

[9] Página de Becas de la universidad de Jaén. (s. f.). Universidad de Jaén. https://www.ujaen.es/servicios/sae/anuncios

[10] Página de Becas de la universidad de Córdoba. (s. f.). Universidad de Córdoba. http://www.uco.es/pie/becas-ayudas/convocatorias-propias

[11] Página de Becas de la universidad de Cádiz. (s. f.). Universidad de Cádiz. https://atencionalumnado.uca.es/becas-uca/

[12] Página de Becas Ual. (s. f.). Universidad de Almería. https://www.ual.es/estudios/gestionesacademicas/becas

[13] Power Bi. (s. f.). Microsoft. https://powerbi.microsoft.com/es-es/

[14] Ventajas del uso de Apirest. (s. f.). DesarrolloWeb. https://desarrolloweb.com/articulos/ventajas-inconvenientes-apirest-desarrollo.html

[15] SQL vs NoSQL. (s. f.). SQL vs NoSQL. https://guiadev.com/mysql-vs-mongodb/

ANEXO: SISTEMA WEB: VISTA MÓVIL Y TABLET

A continuación, procedemos a mostrar como quedaría el sistema web responsive *(Ilustraciones 110 - 117)*:

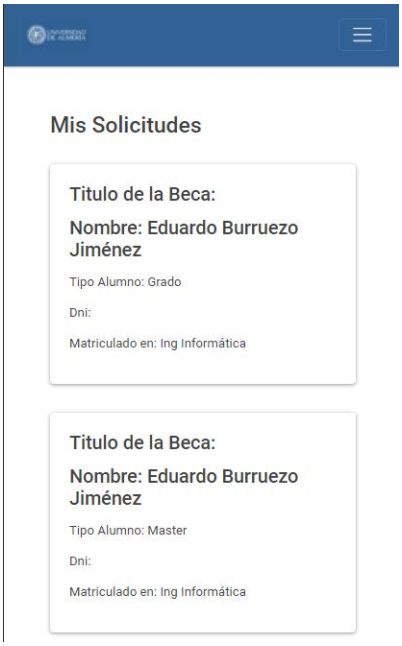


Ilustración 110 Vista Responsive: Mis Solicitudes

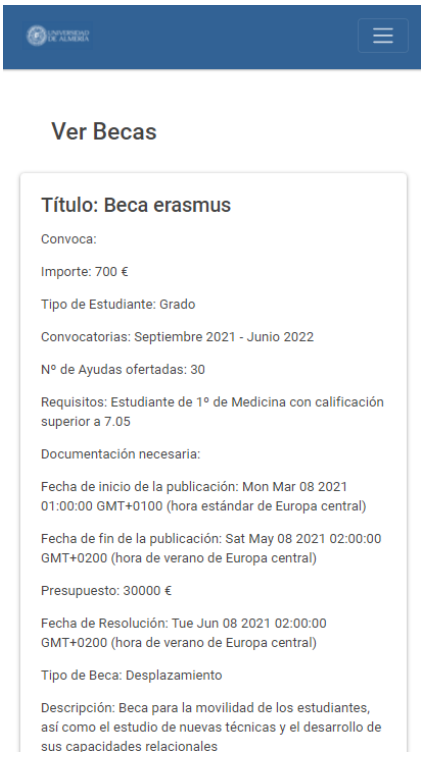


Ilustración 111 Vista Responsive: Ver Becas

Título	Importe	Tipo de Estudiante	Convocatorias	Nº de Ayudas	Tipo de Beca	Ver Beca	Solicitar Beca
yrdstr	594 €	Grado	Septiembre		Ayuda	VER	SOLICITAR
Beca Master	1200 €	Master	Septiembre 2021 - Junio 2022		Ayuda	VER	SOLICITAR

Ilustración 112 Vista Responsive: Becas ofertadas

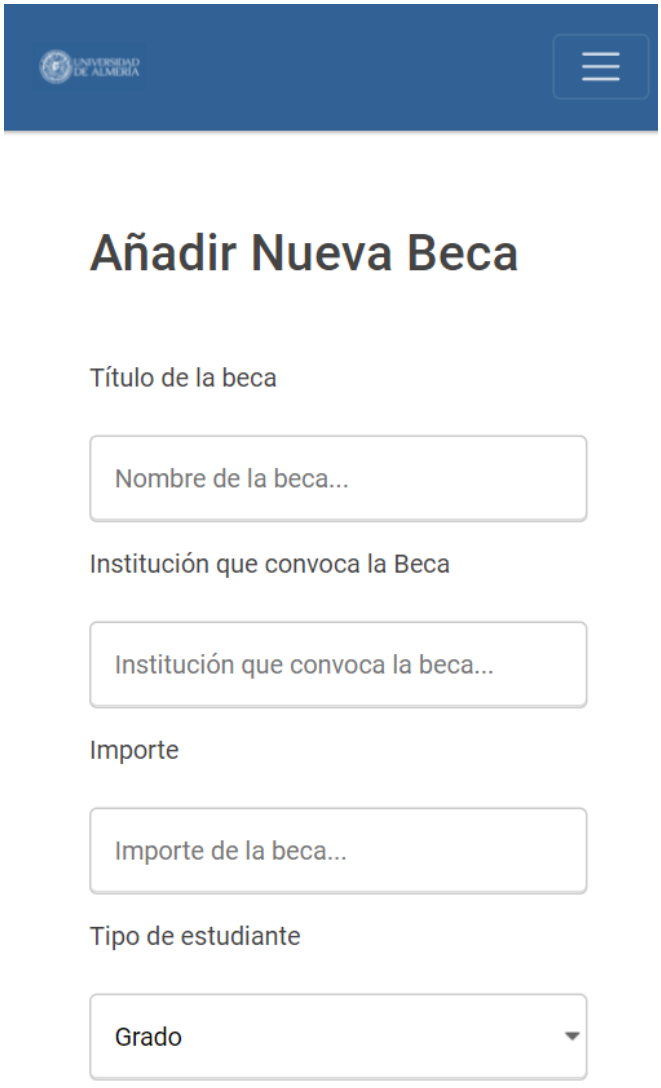


Ilustración 113 Vista Responsive: Añadir Becas

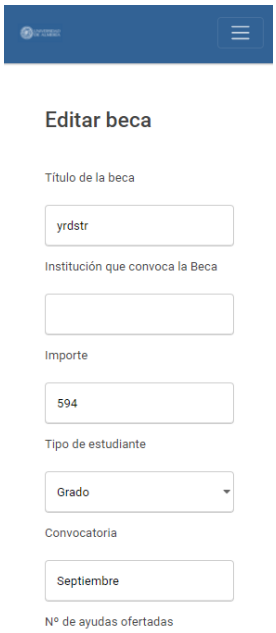


Ilustración 114 Vista Responsive: Editar Becas

Ilustración 115 Vista Responsive: Iniciar Sesión

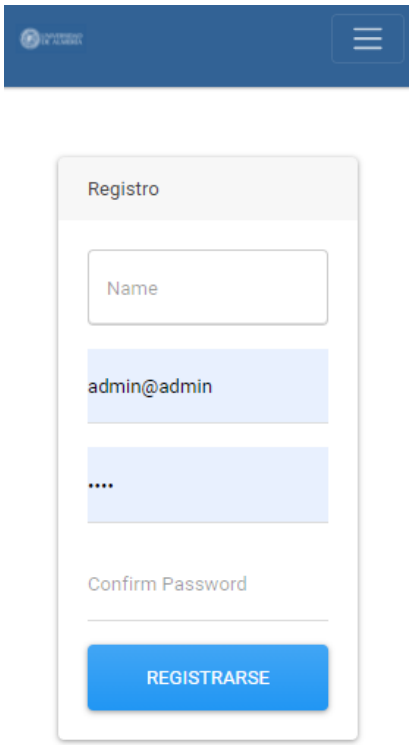


Ilustración 116 Vista Responsive: Registrarse

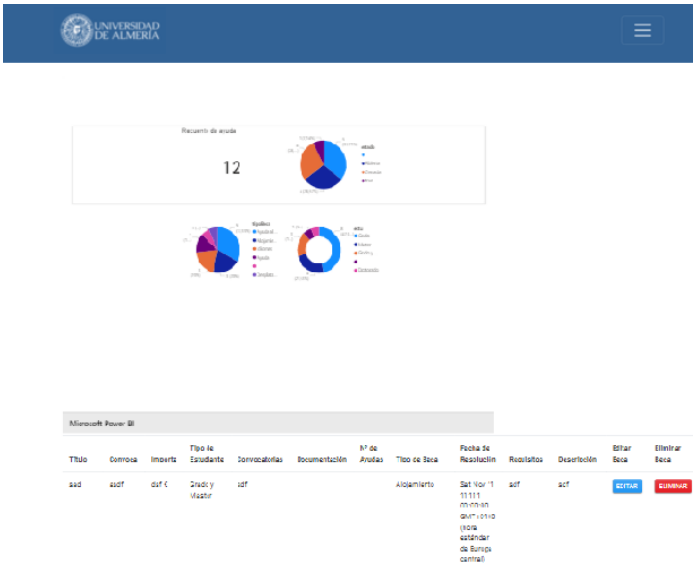


Ilustración 117 Vista Responsive: Becas-admin

ANEXO: CUESTIONARIO SOBRE LA USABILIDAD Y LA CAPACIDAD TÉCNICA

Estimado usuario, con el objetivo de obtener información relevante para comprobar la usabilidad y capacidad técnica de nuestro sistema, se le solicita 15 minutos para poder rellenar este cuestionario con la mayor sinceridad posible. Este cuestionario será de manera totalmente anónima

Por favor, marque con una X la casilla correspondiente.

1. Edad: ☐

2. Titulación académica que posee: ☐

3. Indique su experiencia en el manejo de ordenadores:

Conocimientos nulos	☐	Conocimientos nivel usuario	☐
Avanzado	☐	Experto	☐

A continuación, clasifique las siguientes afirmaciones del 1 al 5 según su grado de acuerdo con las mismas, siendo 1: muy en desacuerdo, 2: algo en desacuerdo, 3: ni en desacuerdo , ni de acuerdo, 4: algo de acuerdo y 5 muy de acuerdo.

4. PRESENTACIÓN	**1**	**2**	**3**	**4**	**5**
La estética general es adecuada					
Los colores empleados respetan la comodidad visual del usuario					
La información de las pantallas es clara y resulta suficiente para su comprensión					
Los elementos gráficos utilizados facilitan su manejo					
La elección de opciones se realiza mediante menús y botones claros y son fácilmente accesibles					

5. INDIVIDUALIZACIÓN	**1**	**2**	**3**	**4**	**5**
El nivel de adaptación del programa a tus necesidades te ha parecido adecuado					
El programa se adapta a los distintos tipos de alumnos que estudian en la universidad					

6. INTERACTIVIDAD.	**1**	**2**	**3**	**4**	**5**
El sistema permite suficiente intercambio de información entre el software y el usuario					
Como usuario ha encontrado libertad en las elecciones que ha realizado en las distintas pantallas					
Ha encontrado el suficiente número de opciones en cada petición					
El sistema le ayuda convenientemente cuando introduce algún dato erróneo					

7. MANEJO	**1**	**2**	**3**	**4**	**5**
El sistema ha resultado ser de fácil manejo					
El sistema es intuitivo					

Es posible moverse con facilidad entre los distintos apartados del sistema					
Completar los formularios ha resultado sencillo					

8. CONTENIDOS	**1**	**2**	**3**	**4**	**5**
Clasificación adecuada de los apartados					
Los objetivos de cada apartado son fácilmente identificables					
La calidad de la información es apropiada en cada caso					

9. AYUDA	**1**	**2**	**3**	**4**	**5**
La ayuda resulta clara, accesible y de fácil navegación					
La ayuda del sistema ha resulto dudas que le iban surgiendo					
El sistema ha detectado errores y le ha informado de lo sucedido					

10. FUNCIONAMIENTO/EFICACIA	**1**	**2**	**3**	**4**	**5**
No han surgido errores que obliguen a interrumpir la acción					
La velocidad de acceso a la información y de paso de una pantalla a otra es apropiada					
El sistema facilita la labor del usuario en función de que actividad realice					

11. COMPROMISO	**1**	**2**	**3**	**4**	**5**
El programa ha respondido a tus expectativas					
Recomendarías el programa a otros usuarios					

12. ¿Qué calificación global le daría a la aplicación?	**1**	**2**	**3**	**4**	**5**	**6**	7	8	9	10

COMENTARIOS LIBRES

13. Indica lo que destacarías como más interesante del sistema

14. Indica lo que destacarías como menos interesante del sistema

15. ¿Qué tipo de contenidos o utilidades has echado en falta en el sistema?

16. Indica alguna sugerencia que consideres que podría mejorar el sistema

17. ¿Seguirías usando el sistema en el futuro?

Sí ☐ No ☐

MUCHAS GRACIAS POR SU COLABORACIÓN

FDO: Eduardo Burruezo Jiménez

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

En la actualidad en la Universidad de Almería la oferta de convocatorias de becas y ayudas se encuentra dispersa en las diferentes unidades, servicios y vicerrectorados que se encargan de su gestión. Esta situación dificulta enormemente la búsqueda de información los estudiantes o futuros candidatos a estudiantes que les resulta complejo encontrar la información pertinente.

La difusión de las becas y ayudas para los estudiantes y para el propio gabinete de comunicaciones es compleja ya que no se dispone de una información centralizada y accesible a las convocatorias abiertas en cada momento.

La publicación de las convocatorias difiere de una unidad, servicio o vicerrectorado a otro lo que dificulta aún más la búsqueda de información. Por lo que sería recomendable definir una plantilla de datos comunes para que la información sea accesible.

Otro de los principales inconvenientes es que no se guarda un histórico de las convocatorias lo que implica que los alumnos si necesitan consultar convocatorias anteriores no pueden hacerlo y bien recurren a otros organismos para solicitar dicha información o bien directamente al Servicio de Becas y Ayudas. Para ello se ha propuesto desarrollar un sistema de gestión que resuelva todos estos problemas y aporte una ayuda tanto a estudiantes como a la administración de la propia universidad.

GRADO EN INGENIERÍA INFORMÁTICA, 2020/2021