Universidad de Valladolid

escuela de ingeniería informática

grado en ingeniería informática

Aplicación web para Casos clínicos de Óptica y Optometría

Alumno: Alejandro Sanz Pérez
Tutor: Manuel Ángel González Delgado

Universidad de Valladolid

Resumen

El objetivo de este trabajo de fin de grado es desarrollar una Aplicación Web para Casos clínicos de Óptica y Optometría. Esta aplicación está pensada para su uso con un fin didáctico para alumnos de Óptica y Óptometría.

El proyecto se ha desarrollado con el stack MEAN. Para su desarrollo hemos seguido una metodología ágil como SCRUM.

Agradecimientos

En primer lugar, quiero dar las gracias a mi familia y amigos los cuales han sido un apoyo incondicional sin los que habría sido imposible llegar hasta aquí.

También me gustaría mencionar lo agradecido que estoy a la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Valladolid por la oportunidad de estudiar aquí y por todos los conocimientos aprendidos.

Abstract

The objective of this final degree project is to develop a web application for Optical and Optometric Clinical Cases. This application is intended for use for educational purposes for Optics and Optometry students.

The project has been developed with the MEAN stack. For its development we have followed an agile methodology like SCRUM.

Índice General

Lista de figuras

Lista de tablas

1 Introducción y objetivos

1.1 Introducción

Existen diversas aplicaciones para gestión de formularios con distintas cuestiones con los cuales se puede llevar a cabo ejercicios con un fin didáctico para los alumnos, como por ejemplo, Google Forms, siendo esta la herramienta que estaba siendo utilizada anteriormente para desempeñar esta labor. El tipo de aplicación escogida para el desarrollo es Web, debido a su portabilidad, escalabilidad y accesibilidad.

En cuanto a la parte del desarrollo se ha optado por usar metodologías ágiles desarrollo de software, la escogida es SCRUM debido a que ya tenia un conocimiento previo acerca de esta metodología.

Las metodologías ágiles se usan para tratar de aumentar el rendimiento de las personas que están en el proyecto, tratando de minimizar los costes lo máximo posible. Otro de los motivos por los que usamos dicha metodología, es porque nos permite una gran flexibilidad a la hora de participar con el cliente para así poder decidir por donde va a ir el desarrollo.

1.2 Motivación

La idea de desarrollar esta AppWeb surgió en el momento en el que la herramienta Google Forms no permitía un almacenamiento tan alto de datos, por lo tanto, no servía para realizar dicha labor.

1.3 Objetivos

1.3.1 Objetivos de desarrollo

Desarrollar un aplicación web de Casos clínicos Óptica y Optometría con un fin educativo para los alumnos de la Universidad de Valladolid, esta va a estar dividida en dos partes:

- Usuario: Para acceder a esta parte no requiere ningún tipo de identificación, en este caso puede acceder a la vista principal donde se encuentran todos los formularios. Y desde ahí, a cada formulario para responder las correspondientes preguntas de ese caso al que ha accedido.

También puede acceder a un formulario de contacto donde puede enviar cualquier tipo de duda rellenándolo. Estas dudas serán enviadas al correo electrónico del administrador.

- Administrador: Una vez identificado correctamente el usuario con su correo electrónico y contraseña puede realizar:
  - Registro y borrado de administradores
  - Creación, edición, y borrado de formularios (casos), preguntas y respuestas.
  - Acceso a estadísticas totalmente anónimas.

El objetivo principal se divide en distintos subobjetivos, algunos son:

- Desarrollar una aplicación con la que cualquier usuario normal que disponga de la aplicación podría llevar a cabo la creación, borrado o edición de formularios sin la necesidad de tener ningún conocimiento de programación. Y también, la resolución de dichos casos respondiendo a las preguntas.
- Hacer uso de la flexibilidad de SCRUM para poder llevar a cabo un mejor desarrollo.

1.3.2 Objetivos personales

Los objetivos personales que he alcanzado con este proyecto son:

- Aprender nuevos conocimientos en el desarrollo de aplicaciones web tanto en el frontend como en el backend haciendo uso de tecnologías actuales.
- Poder seguir aprendiendo cómo se lleva a cabo la planificación un proyecto desde su inicio hasta su fin con la metodología SCRUM.
- Conocer cómo se llevan a cabo todos los pasos para el desarrollo completo de un proyecto, desde la especificación de requisitos al despliegue, pasando por otros pasos como desarrollo y testing.

2 Estructura de la memoria

La memoria del trabajo de fin de grado se va dividir en distintas partes, estas son:

- Introducción y objetivos: En esta sección describimos una introducción al proyecto, la motivación para hacerlo y por último, los objetivos, tanto de desarrollo como personales.

- Proceso de desarrollo y planificación: Se hace una descripción de la metodología que seguiremos en el proyecto y su aplicación en el mismo, conteniendo esto tanto una planificación inicial del proyecto como las historias de usuario y tareas del proyecto.

- Estado del arte: Se pone en contexto tecnológico, acerca de las herramientas que se van a usar para el desarrollo del proyecto y las diferentes opciones disponibles y su aplicación en este proyecto.

- Plan de riesgos: Se explicará qué son los riesgos y cómo elaborar un informe de riesgos. También enunciaremos los riesgos principales que va a tener nuestro proyecto a lo largo del desarrollo.

- Diseño e implementación del sistema: En esta sección trataremos la arquitectura y el patrón que vamos a usar para el proyecto. Además, también se mostrará el seguimiento del proyecto sprint a sprint.

- Presupuesto: Se han descrito los tipos de costes del proyecto, así como el coste final del proyecto.

- Testing: Se ha descrito distintas pruebas a las que someter a la aplicación, para así comprobar su correcto funcionamiento.

- Conclusiones: En esta sección, se hace un un breve resumen de cómo ha ido el desarrollo del proyecto, es decir, si se han cumplido los objetivos tanto de desarrollo como personal. Además, también posibles mejoras futuras.

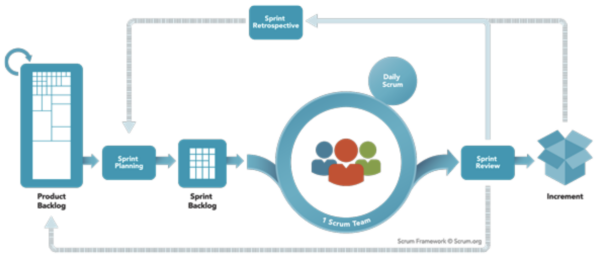


Figure 1: Scrum

3 Proceso de desarrollo y planificación

3.1 Metodología ágil : Scrum

- **¿Qué es Scrum?**: Scrum [1] es un proceso que se usa para llevar a cabo de forma más sencilla un desarrollo, consiguiendo cumplir de esta forma los requisitos establecidos por el cliente. Para la consecución de estos requisitos se fomenta la colaboración de los miembros del equipo, lo cuales se tiene que adaptar constantemente a los cambios que puedan suceder en el proyecto. Esta metodología funciona de manera incremental usando el empirismo. Figura 1.

- **Los eventos de Scrum**: Estos se usan para poder organizar de mejor manera el desarrollo, evitando tanto reuniones demasiado extensas como el añadir reuniones no establecidas en Scrum.
  La duración de un sprint es fija, esta nunca puede variar una vez este haya comenzado.Los eventos de Scrum son [2]:

  - Sprint: El Sprint es el evento más importante de Scrum. Se tratara de que la duración de esta reunión sea siempre la misma. La frecuencia con la que se celebran es de 3 o 2 semanas, siendo el limite máximo de frecuencia de 1 mes y de esta forma poder garantizar que se están haciendo unos buenos progresos para llegar al objetivo .
  - Sprint planning : Este evento se usa para determinar cuales serán los elementos que estarán en el siguiente Sprint Backlog. Para una sprint de un mes la duración máxima se suele estimar en 8 horas.

  - Daily Scrum : Es una reunión corta de entre 10-15 minutos, con frecuencia diaria, y en la que se comentan lo que se esta llevando a cabo ahora y planificación para las próximas 24-48 horas.
  - Sprint review : Consiste en la comprobación del incremento al final de cada sprint.
  - Sprint Retrospective : Se realiza después del Sprint review y en este caso se comentan las cosas positivas y negativas en el último sprint y tratar de mejorarlo para el futuro.

- **Artefactos**: Los artefactos [3] de Scrum se usan para así obtener mejores adaptaciones e inspecciones.
  - Product Backlog : Este artefacto es la mejor opción para obtener la máxima información acerca del producto, este contiene una lista con todos los requisitos que se necesitan para implementar el producto.
  - Sprint Backlog : Este artefacto se compone por una lista de elementos lo cuales vamos a tener que desarrollar en el periodo de ese sprint.
  - Increment: Parte desarrollada en un sprint que esta lista para ser usada.

- **Roles**: Los roles [4] que podemos encontrar dentro de un equipo son los siguientes:
  - Scrum master : Esta persona es el líder del equipo, se encarga de guiar al resto de integrantes para llegar a los objetivos.
  - Product Owner : Es responsable de comprobar la rentabilidad del proyecto desde un punto de vista económico.
  - Team member : Son los componentes del grupo que se encargar del desarrollo del proyecto.

3.2 Aplicación de Scrum en el proyecto

Para la aplicación de Scrum en este proyecto, el reparto de los roles citados anteriormente ha sido el siguiente:

- Alumno: Scrum master y Team member.
- Tutores: Product Owner.

A continuación, describiremos la planificación realizada para el proyecto. Las tecnologías usadas para el desarrollo serán descritas en el próximo capitulo.

3.2.1 Planificación

Según una estimación en el trabajo necesario para realizar la aplicación web y el peso de créditos que tiene el trabajo de fin de grado de Ingeniería Informática, el numero de horas totales del proyecto debería oscilar en unas 300 horas.

Teniendo en cuenta la disponibilidad semanal aproximada que tengo, he decidido dividir en proyecto en 8 sprints, haciendo uso de la metodología ágil descrita anteriormente. La duración total del proyecto seria aproximadamente de 4 meses o 16 semanas, es decir, un sprint cada dos semanas, teniendo una carga de trabajo de 32,5 horas cada dos semanas, es decir, por cada sprint que se realice. La planificación inicial esta en la tabla 1.

Sprint	Descripción	Fecha de inicio	Fecha de fin
1	Documentación y preparación del entorno de desarrollo	19/01/2020	02/02/2020
2	Inicio del desarrollo del API Rest y diseño de la Base de Datos	03/02/2020	17/02/2020
3	Continuación del desarrollo del API Rest, routers y autenticación del usuario	18/02/2020	03/03/2020
4	Desarrollo de los controladores del API Rest	04/03/2020	18/03/2020
5	Comienzo del desarrollo del frontend con angular	19/03/2020	02/04/2020
6	Desarrollo de algunos servicios y componentes	03/04/2020	17/04/2020
7	Desarrollo de los componentes y servicios que se encargan del tratamiento de las preguntas	18/04/2020	02/05/2020
8	Finalización del desarrollo y la documentación del proyecto	03/05/2020	17/05/2020

Table 1: Tabla de planificación inicial

3.2.2 Product Backlog o historias de usuario

En este apartado describimos el conjuntos de historias de usuario que se han especificado para desarrollar el proyecto en la tabla 2.

3.2.3 Tareas

Las tareas que se han realizado en el proyecto en cada sprint, se pueden observar en el apartado de seguimiento del proyecto sprint a sprint (Apartado 6.2).

3.3 Ventajas de aplicar Scrum en el proyecto

El motivo por el que elegimos esta metodología es por lo bien que se podía ajustar a este tipo de desarrollos, desarrollos con constante comunicación con el cliente, aportándonos así una gran cantidad de beneficios en este proyecto. Los principales beneficios obtenidos son:

- Entrega quincenal o mensual de resultados.
- Gran flexibilidad y adaptación a cambios en los requisitos.
- Mitigación de posibles riesgos del proyecto.
- Alineamiento entre el cliente y el equipo. Esto se consigue trabajando el cliente y el equipo en conjunto.
- Productividad. Obtenida por ejemplo, como consecuencia de tener una buen alineamiento entre el cliente y el equipo.

Historia	Titulo	Descripción
1	Vistas	Desarrollo de todas las interfaces con la que interaccionara el usuario
2	Login	Posibilidad de iniciar sesión para entrar como administrador
3	Formulario de contacto	Vista donde el usuario puede enviar sugerencias al administrador
4	Realizar la encuestas	Cualquier usuario puede acceder a cualquier caso y resolverlo
5	Crear un caso	Un administrador puede crear un caso
6	Editar un caso	Un administrador puede editar un caso
7	Borrar un caso	Un administrador puede borrar un caso
8	Crear una pregunta	Un administrador puede crear una pregunta
9	Editar una pregunta	Un administrador puede editar una pregunta
10	Borrar una pregunta	Un administrador puede eliminar una pregunta
11	Crear una pregunta y sus respuestas	Un administrador puede crear una pregunta y sus respuestas
12	Editar una pregunta y sus respuestas	Un administrador puede editar una pregunta y sus respuestas
13	Borrar una pregunta y sus respuestas	Un administrador puede eliminar una pregunta y sus respuestas
14	Registrar administrador	Un administrador puede añadir a otro administrador
15	Eliminar un administrador	Un administrador puede borrar a un administrador
16	Acceso a estadísticas	Un administrador puede ver las estadísticas de cada pregunta

Table 2: Tabla de historias de usuario

4 Estado del arte

En esta sección se pone en contexto tecnológico al lector para que así pueda comprender de mejor forma el proyecto al completo.

El proyecto es un sistema de gestión de Casos clínicos de Óptica y Optometría con stack MEAN. En este capitulo se hablará de los componentes del Stack MEAN, distintos productos existentes en el mercado, así como un explicación de qué es la metodología ágil Scrum, cómo funciona y una planificación inicial.

4.1 Aplicaciones de la gestión de casos

Las aplicaciones de gestión de casos se encargan de dar la posibilidad de que un administrador pueda crear distintos casos cada uno de ello con sus distintas preguntas y a su vez cada pregunta con sus cuatro respuestas correspondientes. Podemos distinguir tres tipos de casos según su dificultad. Cada usuario que no sea administrador va a poder resolver cada uno de estos tests o sugerir dudas al administrador, para la corrección de algún error en alguna de las preguntas.

Todo esto simplificado con una interfaz intuitiva para que cualquier usuario normal puede llevar a cabo la labor tanto de administrador como de usuario, sin necesidad de tener algún conocimiento de programación.

4.2 Productos existentes

Hoy en día, podemos encontrar bastantes aplicaciones que pueden hacer algo similar, como Google Forms o Qualtrihacer alras.

Aunque, como en el caso de Google Forms, muchas no son capaces de almacenar tanta cantidad de casos, con sus preguntas y respuestas, por lo que no son válidas para esta labor. Este factor y el de que no son hechas a medida, es decir, puede que no cubran todas nuestras necesidades son factores determinantes para decantarse por desarrollar una propia.

4.3 API Rest

REST [5] esto se puede definir como una interfaz entre sistemas que usa distintos formatos posibles como XML y JSON para o bien generar operaciones u obtener

datos. Para esas transacciones usa HTTP.

4.3.1 Características

- Protocolo cliente/servidor sin estado: Cada petición HTTP contiene la información necesaria para su ejecución. Por lo tanto, ni el servidor ni el cliente necesitan tener ninguna información de estados anteriores.
- Los objetos REST siempre se manipulan a partir de la URI.
- Las operaciones más importantes son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- Sistema de capas: El sistema REST se divide en distintas capas y cada una de ellas se encarga de una funcionalidad.
- Uso de hipermedios: Es la capacidad de proporcionar enlaces al cliente para ejecutar operaciones concretas.

  El principio de HATEOS es de imprescindible disponibilidad en un API REST. Esto define que por cada petición efectuada al servidor, recibimos una respuesta.

  En la figura 2 podemos ver una respuesta de ejemplo de este principio.

4.3.2 Ventajas y alternativas en el desarrollo

A continuación, describiremos las distintas ventajas que nos aporta mostrando así las razones por las que se ha escogido esta opción y distintas alternativas. Principalmente estas son las ventajas que nos proporciona en el desarrollo, mantenimiento y rendimiento:

- Flexibilidad y portabilidad: Se pueden realizar migraciones de un servidor a otro. Debido a esto se pueden almacenar el frontend y el backend en distintos servidores, siendo esto una enorme ventaja en cuanto al manejo.
- Escalabilidad: Es una de sus características más destacables. Tiene esta cualidad gracias a la independencia entre el cliente y el servidor.

Figure 2: Ejemplo de respuesta de API REST

- Independencia: Gracias a la independencia entre el servidor y el cliente ya citada anteriormente, tenemos la posibilidad de desarrollador el proyecto dividiéndolo en partes independientes.

Como alternativa destacamos SOAP. Sobre SOAP podemos destacar distintas ventajas que tiene el API REST, entre otras citamos:

- Mucho menor uso de recursos.
- Mayor facilidad a la hora de elaborar.
- Mayor escalabilidad y rendimiento a gran escala.

4.4 Entorno y herramientas usadas para el proyecto

En esta sección describiremos el entorno, y todas y cada una de las herramientas que hemos usado tanto para el desarrollo como para la documentación del proyecto.

4.4.1 Visual Studio Code

Visual Studio Code [7] es un editor de código fuente. Entre otras cosas nos aportar muchas opciones dentro de el mismo que nos pueden facilitar el desarrollo.

Figure 3: Visual Studio Code

Características	Lenguajes
Resaltado de sintaxis	Archivo batch · C · C# · C++ · CSS · Clojure · CoffeeScript · Diff · Dockerfile · F# · Git-commit · Git-rebase · Go · Groovy · HLSL · HTML · Handlebars · archivo INI · JSON · Java · JavaScript · JavaScript React · Less · Lua · Makefile · Markdown · Objective-C · Objective-C++ · PHP · Perl · Perl 6 · PowerShell · Properties · Pug template language,[7] [8] · Python · R · Razor · Ruby · Rust · SQL · Sass · ShaderLab · Shell script (Bash) · Swift · TypeScript · TypeScript React · Visual Basic · XML · XQuery · XSL · YAML
Snippets	Groovy · Markdown · Nim[9] · PHP · Swift
Autocompletado de código	CSS · HTML · JavaScript · JSON · Less · Sass · TypeScript
Refactorización	C# · TypeScript
Depuración	• JavaScript and TypeScript for Node.js projects • C# and F# for Mono projects on Linux and macOS • C and C++ on Windows, Linux and macOS • Python with Python plug-in installed • PHP with XDebug and PHP Debug plug-in installed

Figure 4: Visual Studio Code Caracteristicas

Algunas de las opciones pueden ser finalización inteligente de código, depuración, Git, uso de una terminal... Otra ventaja muy importante es que es gratuito lo cual provoca que sea mucho más accesible para todo el mundo, convirtiéndose así en uno de los editores más usados del mundo. Además de esto, tenia familiarización previa con el editor. Podemos ver su logo en la figura 3.

Otro aspecto crítico es el que podemos ver en la figura 4, en ella podemos observar que lenguajes son compatibles y con que características su compatibilidad. En nuestro caso los lenguajes que vamos a usar son TypeScript, HTML, CSS, JSON y JavaScript. Usaremos estos para hacer el desarrollo completo de la aplicación web, tanto la parte del frontend como la parte de backend.

Por todo esto principalmente, nos hemos decantado por usar este editor en vez de otros como Atom o Sublime text.

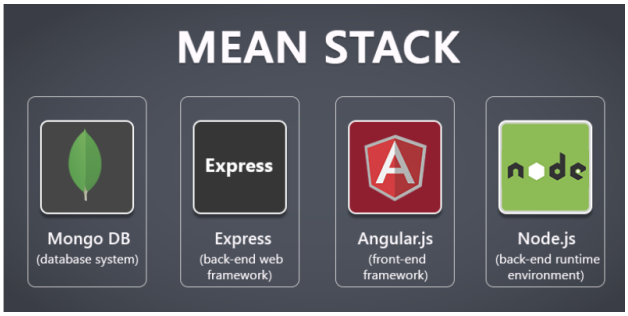


Figure 5: MEAN Stack

Figure 6: Logo Mongo DB

4.4.2 MEAN

MEAN Stack [8] es un framework que se usa para el desarrollo de aplicaciones web, cuyo lenguaje es JavaScript, figura 5. Este está compuesto por MongoDB, ExpressJS, AngularJS y NodeJS, los cuales describiremos a continuación.

4.4.3 MongoDB

MongoDB [9] es un sistema de base datos NoSQL. A diferencia de de las base de datos SQL no guarda los datos en tablas sino que lo guarda en una estructura de datos BSON con un esquema dinámico, estos se denominan documentos, podemos observar un ejemplo en la figura 7.

De forma breve describiremos las principales características y motivos por los que es una buena opción para el stack de NodeJS, que hicieron que nos inclinásemos por esta base de datos, estas son:

- Consultas ad hoc: Se pueden llevar a cabo muchos tipos de consultas con MongoDB ya sea por campos, rangos...Pudiendo devolver estas o bien un documento completo o algún campo específico del mismo, esto resulta muy útil para el tratamiento de datos.

- Indexación: De forma general los índices se usan de la misma forma que en las bases de datos SQL. Aunque en este caso también se puede añadir más índices de carácter secundario.

- Replicación: Esta es una característica de un transcendencia muy importante para prevenir futuras posibles perdidas de datos. MongoDB soporta una replicación primario-secundario. Actuando el secundario como copia de seguridad del primario para no perder información en el que caso de que este falle.

- Balanceo de carga: Esta es una característica que nos permite poder continuar usando el sistema, es decir, que este siga funcionando en el caso de que un servidor caiga. Esto se debe a capacidad de ejecutarse en distintos servidores de forma simultanea. Gracias a esto la aplicación web poseerá una alta disponibilidad, lo cual es muy importante para poder usarlo en cualquier momento.

- Almacenamiento de archivos: Nos permite la manipulación de archivos y contenido como imágenes, lo cual era necesario usar en el proyecto.

- Ejecución de JavaScript del lado del servidor: Este es un motivo muy importante por el que se usa esta base de datos. Como se ha citado anteriormente JavaScript es el lenguaje escogido para desarrollar el proyecto por lo tanto, la posibilidad de poder usarlo en el lado del servidor para hacer consultas en la base de datos es una enorme ventaja. En cuanto al desarrollo nos permite hacerlo de forma mucho más rápida y eficaz que con otras opciones.

- Alto desempeño: Esto se define como la capacidad de soportar de forma exitosa una gran carga de trabajo, también con trafico pesado, siendo esto

```
{
    "_id":
  ObjectId("4efa8d2b7d284dad101e4bc7"),
    "Last Name": "PELLERIN",
    "First Name": "Franck",
    "Age": 29,
    "Address": {
        "Street": "1 chemin des Loges",
        "City": "VERSAILLES"
    }
}
```


Figure 7: Ejemplo Documento MongoDB

de carácter critico. Ya que en este caso si la aplicación web no soporta el uso concurrente de muchos usuarios en un momento determinado no nos serviría.

- Alta escalabilidad: Gracias a su estructura se puede llevar a cabo una escalabilidad horizontal, esto es muy interesante ya que si la aplicación crece de forma importante no habría ningún problema en asimilar este crecimiento.

Estas son las principales razones que otras bases de datos no pueden proporcionarnos de forma conjunta por las que se ha optado por MongoDB, la base de datos NoSQL lider del mercado para aplicaciones modernas, en vez de otras como Cassandra o PostgreSQL.

4.4.4 ExpressJS

ExpressJS [10] es una framework usado en NodeJS para la elaboración de aplicaciones web. Este entre otras características es rápido, sencillo y flexible, además como unas de las propiedades más importantes que ofrece son el uso de routers que nos permiten conectar y realizar la peticiones solicitadas por el frontend en la parte del servidor mediante URL, así como la creación de forma accesible de APIs.

Figure 8: Logo de Angular

4.4.5 Angular

Angular [11] es una framework que se usa para la parte de front-end en el desarrollo de aplicaciones web elaboradas en TypeScript. Gracias a este framework podremos conseguir un desarrollo y pruebas menos complejas. Este es usado para el desarrollo y mantenimiento de aplicaciones web comúnmente llamadas de una sola página.

A continuación, los elementos más usados de angular como los componentes, servicios y módulos.

- Componentes:

  Son aquellos elementos los cuales están compuestos por un archivo de lógica .ts, un HTML y una hoja de estilos asociados. Estos van precedidos del decorador @Component. Dentro de los corchetes del decorador @Component vamos a encontrar distintos elementos, figura 9:

  - Selector: Escribir el nombre de la etiqueta que se usará cuando se desee representar.

```
@Component({
  selector: 'app-show-forms',
  templateUrl: './show-forms.component.html',
  styleUrls: ['./show-forms.component.css'],
  providers: [UserService, FormService]
})
```

Figure 9: Elementos del componente

- TemplateUrl: Ruta de la template html.
- StyleUrl: Ruta de la hoja de estilos que aplicaremos a la template html escrita anteriormente.
- Providers: Escribimos los servicios necesarios para el funcionamiento.

Una vez tenemos el HTML y el archivo de lógica, debemos conectar ambos, esto se realiza mediante el databinding [12]. Esta propiedad llamada databinding es la que nos permite que las paginas web en Angular sean de carácter dinámico, es decir, que el contenido de la página web pueda variar. De forma más sencilla esto se podría decir que es la forma con la que nosotros podemos llevar a cabo un intercambio de datos entre el archivo de lógica (archivo .ts) y la vista (archivo .html).
Podemos encontrar distintos tipos para el intercambio de datos, los cuales son:

- String interpolation: Podemos hacer que muestre en nuestra template HTML el valor que tiene en un momento determinado esa variable en el archivo de lógica. Figura 10.

- Property binding : Se usa para tener un atributo dinámico en angular, esto también se podría llevar a cabo con String interpolation pero podría proporcionar un rendimiento peor. En el siguiente ejemplo se le asignaría el valor de persona.photoURL a la propiedad src.

$$\langle\ img[src] = "persona.photoURL"\ \rangle \tag{1}$$

- Event binding: Este tipo de comunicación se realiza cuando el usuario activa un evento. Un buen ejemplo puede ser el siguiente, en el cual

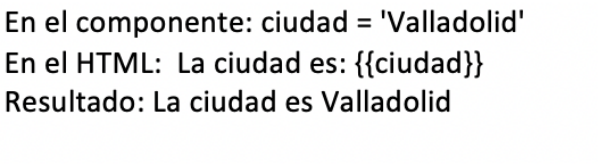


Figure 10: String interpolation

el usuario activa el evento pulsando un botón.

$$\langle button(click) = "AccionAlPulsar()" > Pulsar\langle /button > \quad (2)$$

– Two way data binding: Esta es sin duda la forma más simple de poder usar los dos tipos de data binding mencionados anteriormente. En el siguiente ejemplo mostraríamos por la vista el valor de variable que este puesta, y podríamos actualizar su valor si esta fuese modificada por el usuario en el vista.

$$\langle input\ type = "text"\ class = "form - control" = "name"\rangle \quad (3)$$

Esto también podría ser usado para la comunicación entre padres e hijos de la misma manera.

- Módulos:

  Estos son clases precedidas de del decorador @ngModule. Este esta compuesto tipos de metadatos, figura 11:

  – imports: se encuentran los modulos que vamos a necesitar importar.
  – declarations: las tuberías, directivas y componentes que pertenecen a este módulo.
  – providers : servicios que usa este modulo.
  – exports : son las declaraciones que nosotros queremos dejar visibles y usables para cuando este módulo sea importado en otro módulo.

```
@NgModule({
imports: [ BrowserModule, HttpModule, FormsModule ],
declarations: [ PersonComponent, ContactComponent, ContactListComponent ],
providers: [ PersonService, ContactService ],
exports: [ ContactListComponent, ContactComponent ]
})
export class ContactModule {}
```

Figure 11: Module

- Directivas:

  Las directivas [13] se usan directamente en la template del HTML, estas se pueden usar para distintas funciones. Destacamos dos tipos de directivas:

  - De atributo: Podemos cambiar la apariencia de un texto del template, como por ejemplo cambiar nuestro párrafo de color o de tamaño. Para estos casos se puede usar ngModel (descrito anteriormente) o ngClass, la cual permite aplicar clases de forma dinámica.
  - Estructurales: Son usadas para mostrar, ocultar o cambiar elementos en la vista del usuario. Principalmente destacamos tres tipos:
    - *ngIf: Mediante esta directiva podemos ocultar o mostrar cosas dependiendo de si la variable que pongamos dentro cumpla o no la condición.
    - *ngFor:Esta directiva nos permite mostrar repetidas veces un array de "cosas" o mostrar un elemento tantas veces como queramos.
    - *ngSwitch: Podemos mostrar u ocultar algo dependiendo del valor que tenga la variable que pongamos en la condición.

- Tuberías:

  En angular las tuberías o pipes [14] son usadas para mostrar los resultado en la vista, es decir, en el HTML. Estas tuberías pueden ser personalizadas o predefinidas. A continuación, podemos encontrar distintos ejemplos:

  - Pasar el valor de un objeto a un valor de tipo fecha:

$$\{\{dateObj|date\}\} \tag{4}$$

La salida seria una cadena de tipo fecha como : 'Jun 2, 2019'

- Transformar un cadena en mayúsculas o en minúsculas.

$$\{\{value|uppercase\}\} \tag{5}$$

La salida seria una cadena en mayúsculas como : 'HOLA'

$$\{\{value|lowercase\}\} \tag{6}$$

La salida seria una cadena en minúsculas como : 'hola'

- Mostrar el valor en porcentaje.

$$\{\{value|percent\}\} \tag{7}$$

La salida seria un porcentaje de un valor : 10

- Servicios: Un Servicio en Angular es el mecanismo para compartir funcionalidad entre componentes, usar un servicio para compartir información entre componentes es una buena practica y es la recomendación debido a su mantenibilidad. Son el intermediario entre los componentes y el api rest, estos se conectan con el api rest mediante conexiones http para así poder hacer peticiones de distintos tipos como post, get, delete, put... y de esta poder recoger, comprobar, actualizar o borrar datos de base de datos.

- Inyección de dependencias La inyección de dependencias [15] es la responsable del control del acceso a un servicio, sin este control si dos componentes acceden de forma simultanea se podrían crear dos instancias distintas, mediante esto se encarga de implementar el patrón Singleton. Un servicio singleton es un servicio para el que solo existe una instancia en una aplicación. Destacamos dos opciones:

  - ProvidedIn: Estos son aquellos que se usan en los servicios, un servicio es aquella clase que esta precedida por el decorador @Injectable.Figura 12.
  - Providers:Estos dan la posibilidad de que un componente (o @NgModule) pueda usar los servicios que se encuentren dentro de los corchetes de providers.Figura 13.

```
@Injectable({
 providedIn: 'root'
})
```

Figure 12: ProvidedIn

```
@Component({
 ...
 ...
 providers: [UserService, QuestionService, AnswerService]
})
```

Figure 13: Providers

- Ventajas de angular: En este apartado citaremos las principales ventajas que nos aporta usar angular en el front-end del desarrollo las cuales han motivado que sea el escogido para el desarrollo. Estas son las siguientes:

  - Desarrollo más rápido: Esto se nota sobre todo al comienzo del proyecto. Angular ya crea por ti tanto la estructura como la organización de ficheros, para que así no pierdas tiempo en ello y puedas dedicarte de lleno directamente en las funcionalidades.

  - Facilidad de mantenimiento: Esto se debe al uso de TypeScript como lenguaje en el desarrollo. Este es bastante legible lo cual nos permite llevar a cabo cambios de forma rápida.

  - Reutilización: Esto principalmente se logra gracias al uso de componentes, ya que estos pueden ser usados repetidas veces.

  - Velocidad y rendimiento de la aplicación web: Proporciona la posibilidad de crear una aplicación de una página, lo cual hace que las webs hagan cambios de forma instantánea, provocando esto que el tiempo

Figure 14: Logo de NodeJS

de carga de las paginas sea mucho menor. Esto es un motivo muy importante, debido que si la aplicación va rápido el usuario tendrá una interacción mejor con ella.

– Presente y futuro: Es una framework muy demandado en la actualidad, experimentando grandes crecimientos en su uso cada mes. Así como su uso no deja de crecer en el presente también posee un gran futuro estable, lo cual habla muy bien de él, corroborando que es una buena elección.

4.4.6 NodeJS

NodeJS [16] es una tecnología que puede ser usada para el desarrollo de aplicaciones de carácter general. En este caso caso a ver usada para el desarrollo una aplicación web. A grandes rasgos es una plataforma de ejecución de JavaScript, el lenguaje usado para el desarrollo. Aunque esta tecnología puede ser usada para muchos tipos de desarrollos comúnmente se suele usar para API REST, es decir, servicios web que devuelven datos en formato JSON.

A continuación, mencionaremos las principales características por la que se a optado por NodeJS, asi como por los principales usos del mismo en el proyecto, estos son:

- Muy buena capacidad para realizar una gran cantidad de tareas con un

Figure 15: Logotipo de NPM

pequeño consumo de recursos, provocando que sea una muy buena opción para proyectos que tengan una alta concurrencia.

- La posibilidad de poder usarlo en la mayoría de los servidores.
- Usado por aplicaciones tan importantes como PayPal o Linkendin. Esto nos muestra que es una gran opción para el desarrollo de nuestro proyecto.
- Proporciona transferencias de datos de forma inmediata.
- Lenguaje backend: Como se ha citado anteriormente es usado para el desarrollo de servicios web basados en API REST. Gracias a NodeJS podemos desarrollar la parte del servidor de forma muy cómoda.
- Programas de consola: Este es un uso importante, ya que la muchas herramientas de frontend están desarrolladas con dicha plataforma. Esta consiste en la creación de programas que se ejecutan haciendo uso de comandos en una terminal o consola.

Con esto terminaríamos con los subsistemas de software que componen el MEAN, continuamos con el resto de tecnologías usadas.

4.4.7 Paquetes NPM

NPM [17] (Node Package Manager) como su propio nombre indica es un gestor de paquetes desarrollado en JavaScript. Es un gestor de gran utilidad en el desarrollo ya que con tan solo una linea podemos instalar librerías almacenándose en 'node modules' por defecto, agregar dependías en el proyecto o administrar todo el proyecto en general.

Todas estas dependencias se encuentran el fichero packaje.json, el cual podemos

Figure 16: Fichero package.json

observar en la figura 16. A continuación, procederemos a explicar brevemente cada una de las dependencias que aparecen el fichero.

- **bcrypt-nodejs**: Bcrypt [18] es un paquete que hemos instalado y usado en el proyecto para guardar encriptadas las contraseñas en la base de datos, aumentando de esta forma la seguridad, ya que si las guardamos en plano todo aquel que tenga acceso a la base de datos podrá obtener todas las contraseñas registradas. Las contraseñas se guardan encriptadas mediante una función de hashing, siendo imposible que dos contraseñas generen el mismo hash.

- **body-parser**: Es un paquete que nos permite analizar los cuerpos de una solicitud de HTTP. Esta admite dos tipos de formatos: JSON y urlencoded.

- **connet-multiparty**: Dependencia usada para el tratamiento de archivos.

- **cors**: Cors [19] (Intercambio de recursos de origen cruzado) es un procedimiento que nos permite obtener permisos para acceder a ciertos recursos específicos desde un origen distinto al que pertenece. Esto lo consigue usando las cabeceras HTTP adicionales.

- **jsonwebtoken**: Jsonwebtoken [20] es una dependencia del proyecto que se usa para permitir a cualquier usuario sea autenticado en el servidor

Figure 17: Logotipo de Github

mediante el uso de Tokens, realizando esto de forma sencilla y segura. Gracias al uso de tokens evitamos tener que enviar los credenciales en cada invocación.

- **moment**: Paquete que se encarga de datos como fechas, horas...
- **mongoose**: Mongoose [21] es una biblioteca de JavaScript que nos permite crear modelos específicos. En MongoDB se almacenaran los datos siguiendo las definiciones del esquema de cada modelo.
- **mongoose-pagination**: Nos permite hacer una consulta que nos devuelve un resultado dividido por paginas.
- **nodemailer**: Este paquete se usa para poder enviar mensajes mediante el correo electrónico.

4.4.8 Github

La tecnología usada para el control de versiones ha sido elegida Github, esta es una herramienta que nos permite crear repositorios privados donde podemos ir almacenando las versiones que se vayan implementando en las distintas ramas que nosotros podemos ir creando.
Ramas usadas principalmente son:

- origin/master: En esta rama se encuentra el código que esta listo para producción. Esta es una rama muy usada comúnmente.

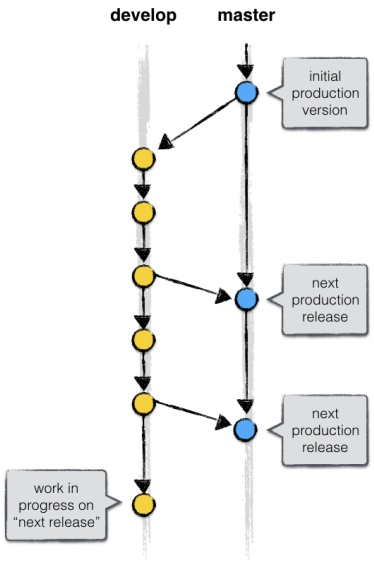


Figure 18: Ramas [23]

- origin/develop: En esta rama se encuentra el código que se esta desarrollando y que todavía no esta listo para estar en producción. Una vez una parte del desarrollo por ejemplo, un funcionalidad especifica, está acabada se pasa a origin/master.

Gracias a la figura 18 podemos observar una descripción de cómo funcionaria el control de versiones con las dos ramas descritas anteriormente.

4.4.9 HTML5

HTML 5 [22] es un lenguaje de programación que se usa para determinar la estructura de la vista de la pagina web. Esta estructura va a poder ser modificada con el uso de CSS (maquetándolo) y con JavaScript para hacerlo de carácter dinámico. Este lenguaje es escrito mediante etiquetas rodeadas de corchetes como por ejemplo:

$$< h1 > Texto < h1/ > o < p > Texto2 < p/ > \tag{8}$$

- **Aplicación en la web**: Esta tecnología es utilizada para nuestra página web para realizar la estructura del contenido en cada una de nuestras vistas, con las cuales va a interactuar el usuario.

4.4.10 CSS

CSS [24] u hojas de estilo en cascada. Es un lenguaje que se usa para crear un diseño más visual, es decir, más atractivo de la vista con la que va a interaccionar el usuario, aportándole por ejemplo, estilos, posicionamiento de los elementos, temas... Este lenguaje se suele aplicar para interfaces de usuario escritas en HTML o XHTML.

A continuación, describiremos en que consisten el conjunto de reglas de una hoja de estilos:

- **Selectores**: Con los selectores determinamos a que etiquetas concretas o atributos se le aplican esos estilos. Estos se pueden aplicar :

  - A todos los elementos que sean de un tipo determinado como h1, h2, p...
  - A elementos con un atributo especifico como la clase o el identificador único (id).

  Cabe destacar que estos son sensibles a las mayúsculas. Y que en el caso de aplicarlo a un clase se lo podremos estar aplicando a uno o más elementos pero si se lo aplicamos al identificador esto solo afectará a un elemento.

- **Orden de importancia**: En una hoja de estilos CSS hay un orden de importancia el cual se puede observar en la figura 19. Para un mismo documento HTML se le pueden asignar múltiples hojas de estilos al mismo tiempo. También cabe otra posibilidad de aplicarlo que es incrustarlo dentro del documento HTML.

- **Especificidad**: La especificidad se usa para determinar que cambio o estilo se le aplica a un elemento concreto del HTML cuando a este se le están

Esquemas de prioridad CSS (de mayor a menor importancia)

Prioridad	Tipo de origen de CSS	Descripción
1	Importancia	La anotación `!important` sobreescribe la prioridad anterior
2	Inline	Un estilo aplicado a un elemento HTML por medio del atributo `style`
3	Media Type	Una propiedad aplica a todas las media types, a menos que un media type específico CSS esté definido
4	Definido por el usuario	La mayoría de los navegadores tienen esta característica de accesibilidad: un estilo CSS definido por el usuario
5	Especificidad del selector	Un selector contextual específico (`#heading p`) sobreescribe una definicón general (`p`)
6	Orden de las reglas	La última regla especificada tiene una mayor prioridad
7	Herencia	Si una propiedad no está especificada, es heredada del elemento padre
8	Definición de propiedad CSS en el documento HTML	Una regla CSS común sobreescribe el valor del navegador
9	Predeterminado del navegador	La prioridad más baja: estos valores son determinados por las especificaciones iniciales de la W3C

Figure 19: Orden de importancia [24]

aplicando varios al mismo tiempo, lo cual sin esto podría dar incongruencias. Podemos observar unos ejemplos de especificidad en la figura 20.

- **Aplicación en la web**: CSS es una tecnología que se usa para poder darle una mejor apariencia a las vistas de nuestra AppWeb siguiendo la estructura que previamente hemos creado con HTML. En ocasiones también hemos usado Bootstrap, el cual es un framework que simplemente nos facilita el uso de CSS dándonos algunos estilos ya predefinidos.

4.4.11 JavaScript

JavaScript [25] es un lenguaje de programación que principalmente está basado en prototipos, orientado a objetos, de tipado débil y dinámico.

Este lenguaje se usa de forma principal en la parte de cliente, lo cual se refiere a la parte con la que va a interaccionar directamente el usuario haciendo que las páginas web sean de carácter dinámico, también es usado en el lado del servidor lo cual es lo que se centra en la parte que no se ve en la aplicación web, es decir, se refiere a la parte que interacciona con la base de datos.

Para poder interaccionar con una pagina web el lenguaje JavaScript debe usar una implementación del DOM (Document Object Model). Hoy en día la mayoría de los navegadores modernos hacen uso de ello. Actualmente una de las mejoras

Selectores	Especificidad
H1 {color: white;}	0, 0, 0, 1
P EM {color: green;}	0, 0, 0, 2
.grape {color: red;}	0, 0, 1, 0
P.bright {color: blue;}	0, 0, 1, 1
P.bright EM.dark {color: yellow;}	0, 0, 2, 2
#id218 {color: brown;}	0, 1, 0, 0
style=" "	1, 0, 0, 0

Figure 20: Especificidad [24]

que se han introducido es el uso de peticiones AJAX con JavaScript, esto nos proporciona la posibilidad de enviar y recibir información del servidor desde la parte del cliente.

- **Características**: Las características más comunes en la mayoría de desarrollos son las siguientes.
  - Imperativo y Estructurado.
  - Dinámico:
    - Tipado dinámico: Esto quiere decir que el tipo no esta asociado a la variable sino al valor de la misma. El tipo lo podemos comprobar con typeof.
    - Objetual: Esta compuesto por objetos.
    - Evaluación en tiempo de ejecución.
  - Funcional: Es un paradigma de programación. Esto nos describe como se debe organizar un programa según unos principios.
  - Prototípico: Para la herencia se usan prototipos en vez de clases.
- **Aplicación en la web**: JavaScript permite que la página web sea dinámica, es decir, que pueda mostrar cosas distintas en función del contexto o lo

que el usuario elija mientras interacciona con ella.
En nuestro caso no vamos a usar JavaScript de forma directa, usaremos un framework que nos facilitara la laboral aportándonos más funcionalidades.

4.4.12 TypeScript

TypeScript [26] es un lenguaje de programación que amplia la sintaxis de JavaScript, añadiendo alguna que otra funcionalidad más, por eso cabe destacar que un código de JavaScript casi con toda probabilidad funcionará.

- **Sistema de tipos**: Como en cualquier lenguaje podemos encontrar distintos tipos de datos, los más usados serian String y Number pero tambien soporta algunos más:
  - Boolean: Tipo de dato que puede valer true o false.
  - Array: Es vector donde se pueden almacenar muchos elementos de un mismo tipo.
  - Tuple: Es similar al array, solo que a diferencia del array este tiene un número fijo de elementos escritos.
  - Enum: Tipo de dato que expresa una enumeración.
  - Any: Este tipo de dato nos permite declarar que el tipo puede ser cualquiera.
- **Aplicación en la web**: TypeScript es usado en nuestro proyecto de forma muy amplia ya que incluye numerosos beneficios a la hora de desarrollar. Se ha usado en angular, es decir, para la parte del frontend.

4.4.13 Postman

Postman [27] es una herramienta que para este proyecto en concreto hemos usado para el testeo del API REST.

Ademas de otras funcionalidades, nos permite comprobar que el API REST funciona correctamente sin la necesidad de tener desarrollado aun el frontend. Esto es una buena ayuda ya que podemos encontrar y solucionar errores de manera rápida que si tuviésemos que desarrollar también el frontend para hacerlo.

El proceso de instalación de POSTMAN [28] es muy sencillo, basta con entrar

Figure 21: Logotipo Postman

Figure 22: Logotipo de LaTeX

en su pagina oficial y descargarlo, después se instala de forma similar a cualquier aplicación común.

4.4.14 LaTeX

Latex [29] es la herramienta escogida para la realización de la memoria del trabajo de fin de grado. Es sistema de composición tipográfica basado en TeX. Es ampliamente usando en documentos de tipo científicos, elaboración de tesis, producción de libros de texto...

Los principales motivos por los que se ha escogido esta opción frente a otras opciones como Word es por su calidad profesional y por una excelente calidad de imprenta, con lo que se conseguirá una memoria mucho más profesional y mejor que usando otras alternativas más tradicionales.

Figure 23: Logotipo de Visual Paradigm

4.4.15 Visual Paradigm

Visual Paradigm [30] es una herramienta UML que se usa para el modelado del mismo. Con esta herramienta podemos llevar a cabo diagramas de casos de uso, o diagramas de clases, de entidad relación... Además de esto, también pueden llevar a cabo tareas de generación de informes o generación de código. Una de los contras del software es que no es de carácter gratuito.

En nuestro proyecto lo hemos usado para la elaboración de diagramas de clases en el que mostraremos cada una de las clases creadas para el proyecto con los atributos correspondientes de cada uno.

5 Plan de riesgos

La gestión de riesgos [31] es el procedimiento de identificar, analizar y tomar decisiones frente a factores de riesgo que puedan surgir mientras se lleva a cabo el proyecto y para poder alcanzar lo objetivos. Esto implica contemplar posibles riesgos que puedan suceder en el futuro.

Para elaborar un informe debemos tener en cuenta los siguientes aspectos:

- Identificar los riesgos.
- Evaluar los riesgos: El equipo de proyectos debe identificar los orígenes de tales riesgos.
- Desarrollar respuestas frente al riesgo: Una vez se llevan a cabo los dos anteriores aspectos lleva el momento de valorar los posibles remedios para gestionar el riesgo en el caso de que suceda o evitar incluso que este suceda. Lo que nosotros llamaremos el plan de contingencia y medidas preventivas para reducir el impacto del mismo si este sucede.

5.1 Posibles riesgos en el proyecto

En la siguientes tablas se va a describir distintos posibles riesgos que podemos encontrar en un proyecto, la mayorías se podrían aplicar para cualquier proyecto de software de carácter general.

De la tabla 3 a la tabla 9 vamos a enunciar el nombre del riesgo, una pequeña descripción, probabilidad de que se suceda, impacto del mismo en el caso de que suceda, las medidas preventivas y el plan de contingencia frente al riesgo. Todos estos riesgos se han tenido en cuenta a la hora de planificar el proyecto.

Nombre	Falta de conocimientos
Descripción	El equipo no esta lo suficientemente formado con las herramientas
Probabilidad	Baja
Impacto	Medio
Medidas preventivas	Reserva de horas en el proyecto para aprender
Plan de contingencia	Solicitar ayuda a alguna fuente con más conocimientos

Table 3: Riesgo de falta de conocimientos con las herramientas

Nombre	Problemas con el hardware
Descripción	Rotura del equipo usado en el proyecto
Probabilidad	Baja
Impacto	Alto
Medidas preventivas	Hacer copias de seguridad de todo lo realizado
Plan de contingencia	Reparación del hardware y ampliación del proyecto

Table 4: Riesgo de hardware

Nombre	Enfermedad
Descripción	La persona encargada del proyecto cae enferma
Probabilidad	Media
Impacto	Alto
Medidas preventivas	Tener margen con la fecha finalización del proyecto
Plan de contingencia	Posponer las tareas a otro sprint

Table 5: Riesgo de enfermedad

Nombre	Cambio de requisitos
Descripción	El cliente quiere que la aplicación tenga otra funcionalidad
Probabilidad	Baja
Impacto	Alto
Medidas preventivas	Usar SCRUM para poder acometer mejor estos cambios.
Plan de contingencia	Realizar los cambios en la aplicación

Table 6: Riesgo de requisitos dinámicos

Nombre	Mala planificación
Descripción	Mala estimación en la duración del proyecto
Probabilidad	Media
Impacto	Medio
Medidas preventivas	Tener margen con la fecha finalización del proyecto
Plan de contingencia	Prolongar el desarrollo y añadir un sprint más si fuera necesario

Table 7: Riesgo de una mala planificación

Nombre	Falta de disponibilidad del personal
Descripción	El personal no tiene el tiempo necesario para dedicárselo al proyecto
Probabilidad	Media
Impacto	Alto
Medidas preventivas	Tener margen con la fecha finalización del proyecto
Plan de contingencia	Prolongar el desarrollo del proyecto añadiendo un sprint si fuera necesario

Table 8: Riesgo de falta de disponibilidad del personal

Nombre	Requisitos poco concretos
Descripción	No se conoce bien cierta funcionalidad que tiene la aplicación
Probabilidad	Bajo
Impacto	Medio
Medidas preventivas	Definir bien las tareas a realizar
Plan de contingencia	Repasar las tareas para comprobar que estas sean correctas

Table 9: Riesgo de requisitos poco concretos

6 Diseño e implementación del sistema

En este punto describiremos como se ha ido desarrollando el proyecto, desde que se definieron las historias previamente descritas(tabla 2) hasta completarlo todo.

En primer lugar describiremos la arquitectura usada y cómo se ha llevado a cabo el seguimiento del proyecto en cada uno de los sprints.

6.1 Arquitectura

Podemos observar la arquitectura usada en el proyecto en la figura 24. En ella podemos observa como aparecen todas la herramientas descritas anteriormente, destacando principalmente MEAN (MongoDB, ExpressJS, AngularJS, NodeJS).

6.1.1 Patrón MVC

En esta sección veremos cómo un framework normalmente implementa un patrón arquitectónico MVC (modelo-vista-controlador). Además, haremos una descripción de lo que es un framework, cómo los hemos usado en el proyecto, y sus ventajas.

Un framework [32] es una marco de trabajo o entorno de trabajo que nos da unas pautas sobre como desarrollar un proyecto.

En el proyecto hemos desarrollado todo haciendo uso de frameworks para de esta forma poder tener una guía que nos ayude a construir el proyecto de forma

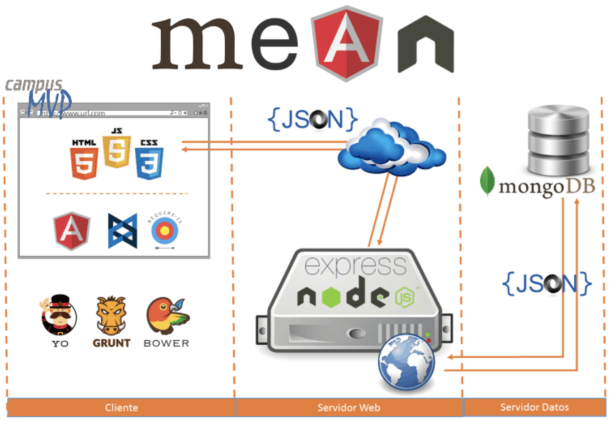


Figure 24: Arquitectura MEAN stack

más rápida y simple. Entre otras cosas proporciona librerías y/o funcionalidades predefinidas que pueden ayudarnos mucho en determinados momentos que nos sea conveniente usarlo.

A continuación, citamos la ventajas que nos proporciona el uso de un framework en el proyecto:

- Estructura y organización predefinida. El uso de un framework nos proporciona la estructura del proyecto, así como la forma con la que debemos trabajar. Esto es de gran ayuda en el desarrollo de la aplicación web porque nos permite preocuparnos más en lo que a las funcionalidades se refiere y no tanto en la estructura.

- Reutilización del código. Evitaremos que distintos fragmentos de código estén repetidos muchas veces en el proyecto, como por ejemplo la conexión a la base de datos de la aplicación web. Esta ventaja es de gran utilidad porque nos ahorra tiempo debido a que no tenemos que volver a escribir repetidas veces lo mismo.

- Agilidad y rapidez en el desarrollo. Al tener algunas cosas menos en las que preocuparnos y algunas funcionalidades ya predefinidas nos permite llevar a cabo el desarrollo de forma más rápida.

- Menor coste en el desarrollo. Cuanto más rápido se desarrolle el proyecto el coste será menor.
- Buenas prácticas de desarrollo con el uso de patrones. Esto es de gran ayuda en el proyecto porque gracias al uso de un framework tenemos unas pautas ya marcadas de cómo se debe realizar el desarrollo.
- Facilidad a la hora de encontrar una librería o código que ya cubra funcionalidades de tu desarrollo. El uso de framework bastante conocido y con una gran soporte como son los usados en el proyecto nos beneficia ya que encontraremos de forma más simple una librería o funcionalidad que necesitemos en el desarrollo de la aplicación web.
- Facilita el mantenimiento. Al usar un determinado framework de la forma predefinida su mantenimiento sera más sencilla ya que si otra persona tiene que mantenerlo va a entenderlo de forma más rápida.

Desventajas de la utilización de un framework:

- Tiempo de aprendizaje. Usar un framework requiere un tiempo de aprendizaje para entender y familiarizarnos con él. En nuestro caso esto si era una desventaja real ya que previamente no tenia ningún conocimiento acerca de ningún framework usado en el proyecto por lo que requirió un tiempo aprender el mismo.
- Elección del framework. Esta es otra desventaja real que ha sucedido en el trabajo de fin de grado ya que antes de comenzar el desarrollo se dedico algún día para decidir que framework escoger.

6.2 Seguimiento del proyecto sprint a sprint

En esta sección mostraremos cómo se han ido realizando todas y cada una de las tareas, tanto de desarrollo como de documentación. Cada una de estas tareas que usaremos tendrán un identificador, una pequeña descripción, horas por persona (H-P) y el estado de la misma. De esta forma iremos sabiendo el número de horas invertidas en cada tarea y el estado de la misma en ese momento.

En cuanto a la división de las tareas se ha llevado a cabo siguiendo estas pautas:

- Se ha comenzado con el desarrollo del API REST antes del frontend debido que gracias a la herramienta Postman mencionada anteriormente podemos llevar a cabo el API REST de forma total sin necesidad del frontend y por contra para hacer completamente el frontend necesitaríamos el API REST.

- Dentro del desarrollo del API REST o del frontend se ha llevado un orden muy similar en ambos, en primer lugar se han llevado a cabo funcionalidades de carácter más necesario como autenticación del usuario, o registro de usuario... antes que otras que necesitan de esas anteriores para poder comprobar bien su funcionamiento, es decir, debemos desarrollar mejor primero la creación de un caso y una vez esté esto, procedemos a desarrollar la eliminación y edición del mismo.

Esta visión nos aporta distintas ventajas como tener conocimiento de las horas invertidas hasta el momento y las que creemos que nos quedan para finalizar el proyecto, saber que tareas están realizadas y cuales no, así como el estado de cada tarea en un momento determinado. Teniendo un mayor control sobre como se esta desarrollando el proyecto.

6.2.1 Sprint 1

En este sprint principalmente se dedico a documentar y organizar todo lo que se iba a realizar en el proyecto. También ha servido para comenzar con la preparación del entorno de desarrollo.

En la tabla 10 se describen cada una de la tareas realizadas en este sprint. El trabajo realizado en todos y cada uno de los sprints es aproximadamente de 32,5 horas. En este sprint se ha optado por llevar a cabo tareas como saber los requisitos del usuario o preparación del entorno, ya que son estas de carácter necesario para poder iniciar el desarrollo del proyecto.

En este sprint cumplimos los objetivos establecidos y pudimos completar todas las tareas. Como la instalación de todas las herramientas necesarias para el proyecto como: Angular, NodeJS, Express, MongoDB, visual studio code, npm y correspondientes paquetes que se iban a necesitar para el desarrollo del proyecto. También, se creó un repositorio en github donde se hará posteriormente el control de versiones del proyecto.

Id	Descripción	H-P	Estado
T-001	Documentar introducción y objetivos	3h	Completado
T-002	Requisitos del usuario	2h	Completado
T-003	Preparación del entorno de desarrollo	9h	Completado
T-004	Documentación de metodologías y patrones	9h	Completado
T-005	Documentación del entorno y herramientas usadas	6h	Completado
T-006	Documentar la planificación del proyecto	2h	Completado

Table 10: Tabla sprint 1

6.2.2 Sprint 2

En este sprint tras llevar a cabo toda la preparación del entorno procedemos a llevar a cabo los primeros pasos en el desarrollo. Las tareas realizadas se describen en la tabla 11.

Como se ha explicado antes se ha comenzado con el desarrollo del API REST antes que el frontend. Las tareas iniciales para comenzar el desarrollo son el diseño de la base de datos, y la conexión a la misma, ya que es donde se va a almacenar toda la información del API REST, también es importante llevar a cabo la estructura de carpetas para saber donde crear cada cosa, y por último, la creación de los modelos que determinarán cómo se van a almacenar los datos de la base de datos, esto es necesario antes que los controladores, ya que estos necesitarán de su existencia para funcionar.

Id	Descripción	H-P	Estado
T-007	Diseño de la base de datos	3h	Completado
T-008	Creación de la conexión de la base de datos y el servidor web	8h	Completado
T-009	Estructura del API REST	8h	Completado
T-010	Creación de los modelos	9h	Completado
T-011	Documentar la implementación	4h	Completado

Table 11: Tabla sprint 2

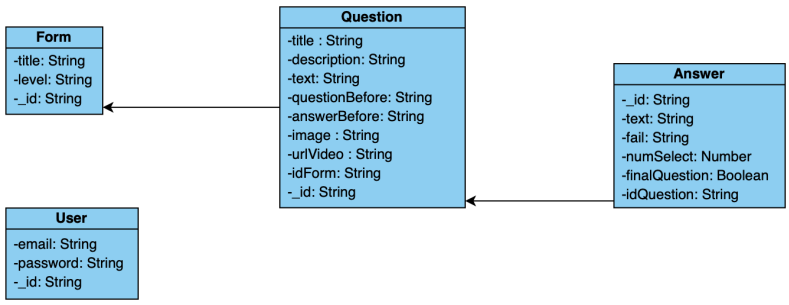


Figure 25: Diseño de la base de datos

- En primer lugar creamos el diseño de la base de datos, la cual puedes observar en la figura 25.
- A continuación, elaboramos la estructura del API REST, esta va estar compuesto por una carpeta en la que en su interior se van a encontrar todos los ficheros del proyecto correspondientes al backend, como podemos observar en la figura 26, se han creado distintas carpetas como models que contendrá todos los modelos del proyecto, otra con todos los controladores, routers...
- En la tarea 008, creamos la conexión de la base de datos en el fichero index.js y el servidor web en el app.js, podemos observar el resultado en las figuras 27 y 28.
- Por último, en lo que al desarrollo respecta realizamos todos los modelos necesarios para el API Rest. Para este proyecto hemos realizado 4 modelos: answer.js, question.js, form.js, user.js. Como ejemplo se mostrara el modelo de question.js en la figura 29.

6.2.3 Sprint 3

Para este Sprint como tareas principales tenemos la elaboración de algún controlador, los cuales se encargan de conectar directamente con la base de datos para hacer todo tipo de consultas, la autenticación mediante JWT y la elaboración de los routers los cuales se encargan de enlazar los controladores con el frontend mediante peticiones post, get, put o delete. Las tareas se encuentran en el tabla 12.

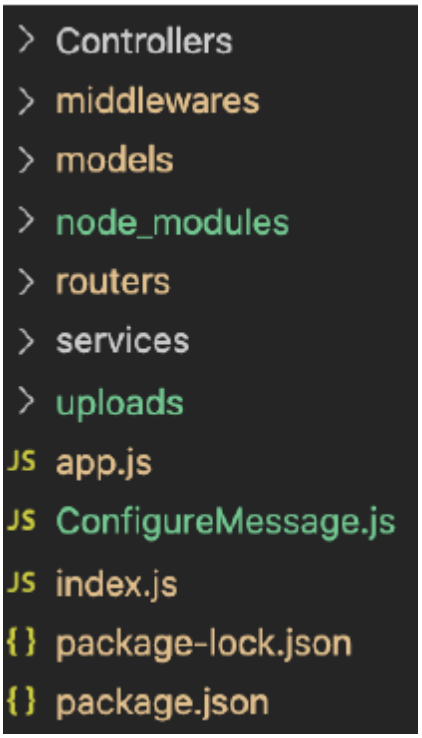


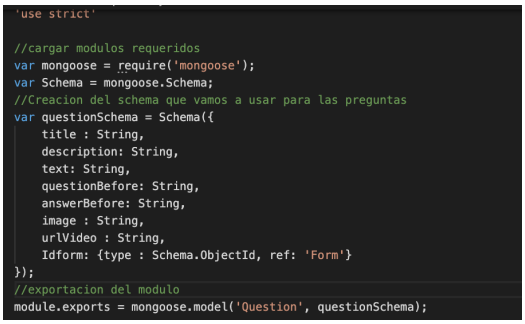
Figure 26: Estructura del API Rest(Backend)

```
'use strict'

//cargar las dependencias del modulo
var mongoose = require('mongoose');
var app = require('./app');
//Determinar el pue (property) NodeJS.Process.env: NodeJS.ProcessEnv
var port = process.env.PORT || 3977;

//Conexion con la base de datos de mongodb
mongoose.Promise = global.Promise;
mongoose.connect('mongodb://localhost:27017/tfgDB',{ useNewUrlParser: true , useUnifiedTopology: true},(err, res) => {

    if(err){
        throw err;
    }else{
        console.log("La conexion a la base de datos esta funcionando correctamente");
        app.listen(port, function(){
            console.log("Server del api rest escuchando");
        });
    }
})
```

Figure 27: Index.js

```
'use strict'

//cargar las dependencias del modulo
var express = require('express');
var bodyParser = require('body-parser');

var app = express();

//cargar rutas
var user_routes = require ('./routers/user');
var form_routes = require('./routers/form');
var question_routes = require('./routers/question');
var answer_routes = require('./routers/answer');
var faq_routes = require('./routers/faq');

app.use(bodyParser.urlencoded({extended:false}));
app.use(bodyParser.json());

//configurar cabeceras http
app.use((req, res, next)=>{
    res.header('Access-Control-Allow-Origin','*');
    res.header('Access-Control-Allow-Headers','Authorization, X-API-KEY, Origin, X-Requested-With, Content-Type, Accept, Access-Control-Allow-Request-Method');
    res.header('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, DELETE');
    res.header('Allow','GET, POST, OPTIONS, PUT, DELETE');

    next();
});

//rutas base
app.use('/api', user_routes);
app.use('/api', form_routes);
app.use('/api', question_routes);
app.use('/api', answer_routes);
app.use('/api', faq_routes);

module.exports = app;
```

Figure 28: App.js



```
'use strict'

//cargar modulos requeridos
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
//Creacion del schema que vamos a usar para las preguntas
var questionSchema = Schema({
    title : String,
    description: String,
    text: String,
    questionBefore: String,
    answerBefore: String,
    image : String,
    urlVideo : String,
    Idform: {type : Schema.ObjectId, ref: 'Form'}
});
//exportacion del modulo
module.exports = mongoose.model('Question', questionSchema);
```

Figure 29: Modelo de una pregunta

Después de lo realizado en el Sprint anterior, en este caso lo siguiente necesario sería desarrollar la autenticación del usuario ya que va a ser necesario para los routers, después hicimos estos y continuamos con la siguiente tarea, la cual fue el desarrollo del controlador del usuario, siendo los controladores el último elemento a desarrollar en la parte del API REST.

Id	Descripción	H-P	Estado
T-012	Creación del router de user	2,5h	Completado
T-013	Creación del router de form	2,5h	Completado
T-014	Creación del router de question	2,5h	Completado
T-015	Creación del router de answer	2,5h	Completado
T-016	Desarrollo de la autenticación del usuario	9h	Completado
T-017	Desarrollo del controlador de usuario	9h	En progreso
T-018	Documentación de lo implementado	4h	Completado

Table 12: Tabla sprint 3

- En primer lugar desarrollamos la autenticación, ya que es usada por todos y cada uno de los routers, en los casos que sea requerido para realizar dicha función, ya que en algunos casos es requerido ser administrador para realizar una función. Para la autenticación se desarrollaron dos ficheros, una es un middleware, y se llama authenticated.js, el otro es una servicio llamado jwt.js.
- Después, se desarrollaron los router, como ejemplo, vamos a mostrar el router de answer en la figura 30.
- Por último, comenzamos con el desarrollo del controlador de usuario, en este sprint no se pudo completar, por lo tanto se pasará al siguiente sprint, para así finalizarlo.

6.2.4 Sprint 4

Para este sprint se han realizado todos y cada uno de los controladores necesarios, acabando también el de user que no pudo ser finalizado en el anterior sprint, así como la documentación necesaria de la implementación realizada. Las tareas

```
'use strict'

//cargar las dependencias del modulo
var express = require('express');
//cargar el modelo de answer
var AnswerController = require('../Controllers/answer');
//cargar express.router para poder hacer la peticiones get, post...
var api = express.Router();
//cargar le middleware para la autenticacion
var md_auth = require('../middlewares/authenticated');

//conjunto de peticiones que cargaran una funcion del servicio de answer
api.get('/answer/:id', AnswerController.getAnswer);
api.post('/answer',md_auth.ensureAuth, AnswerController.saveAnswer);
api.get('/answers/:Idquestion?', AnswerController.getAnswers);
api.put('/answer/:id', AnswerController.updateAnswer);
api.delete('/answer/:id',md_auth.ensureAuth, AnswerController.deleteAnswer);

module.exports = api;
```

Figure 30: Ejemplo del router answer

realizadas se detallan en la tabla 13.

En primer lugar se acabó el desarrollo correspondiente al controlador de usuario. Después, se lleva a cabo el resto de controladores que se conectan directamente con la base de datos. Se llevo a cabo el desarrollo de estos ya que son el último elemento que queda por desarrollar en la parte del API REST. Cada uno de los controladores sigue una estructura similar:

- Al comienzo del código contiene unas lineas en las que importa las dependencias del modulo, así como los modelos necesarios para su funcionamiento, muy similar a las primeras lineas de la figura 30.

- Una vez se ha importado todo lo necesario se crean las funciones necesarias. En nuestro caso se han creado principalmente entre otras, funciones para la creación, edición, borrado y acceso de los datos de formularios o casos, preguntas, usuarios y respuestas.

  A continuación, describiremos algunos ejemplos de funciones en las figuras 31, 33, 34 y 32, con esas figuras podemos encontrar una función que cumpliría cada tipo de petición, ya sea post, delete, put y get respectivamente.

- En la última linea siempre se exportan todas y cada una de las funciones que han sido creadas en el controlador, para que estas puedan ser usadas

```
//guardar una pregunta nueva
function saveQuestion(req, res){
    var question = new Question();

    var params = req.body;
    question.title = params.title;
    question.text = params.text;
    question.questionBefore = params.questionBefore;
    question.answerBefore = params.answerBefore;
    question.Idform = params.Idform;
    question.finalQuestion = params.finalQuestion;
    question.description = params.description;
    question.urlVideo = params.urlVideo;

    question.save((err, questionStored) =>{
        if(err){
            res.status(500).send({message: "Error guardar la pregunta"});
        }else{
            if(!questionStored){
                res.status(404).send({message: "No se ha guardado la pregunta"});
            }else{
                res.status(200).send({question: questionStored});
            }
        }
    });
}
```

Figure 31: Ejemplo función SaveQuestion

```
//obtener todas las preguntas de un formulario, en este caso se obtenien tambien con paginacion, 4 preguntas por pagina
function getQuestions(req, res){

    var parameters = req.params.Idform.split(' ');
    var formID = parameters[0];

    if(parameters[1]){
        var page = parameters[1];
    }else{
        var page = 1;
    }
    var itemsPerPage = 4;//numero de items por pagina

    if(!formID){ //Get all questions
        var find = Question.find({});


    }else{ //Get questions with formID
        var find = Question.find({Idform : formID});

    }
    find.populate({path:'Idform'}).paginate(page, itemsPerPage, function(err, questions, totalItems){
        if(err){
            res.status(500).send({message: 'Error en la petición'});
        }else{
            if(!questions){
                res.status(404).send({message: 'No hay preguntas'});
            }else{
                res.status(200).send({
                    totalItems: totalItems,
                    questions : questions
                });
            }
        }

    });
}
```

Figure 32: Ejemplo función GetQuestions

```
//borrar una pregunta
function deleteQuestion(req, res){
    var questionId = req.params.id;


    Question.findByIdAndRemove(questionId,(err, questionRemoved)=>{
        if(err){
            res.status(500).send({message: 'Error al eliminar la pregunta'});
        }else{
            if(!questionRemoved){
                res.status(404).send({message: 'La pregunta no ha sido eliminado'});
            }else{

                Answer.find({question: questionRemoved._id}).remove((err, answerRemoved)=>{
                    if(err){
                        res.status(500).send({message: 'Error al eliminar la respuesta'});
                    }else{
                        if(!answerRemoved){
                            res.status(404).send({message: 'La respuesta no ha sido eliminado'});
                        }else{
                            res.status(200).send({question: questionRemoved});
                            fs.unlink('./uploads/'+questionRemoved.image+'', (err) => {
                                if (err) throw err;
                                console.log('Imagen eliminada correctamente');
                              })

                        }
                    }
                });
            }
        }
    });

}
```

Figure 33: Ejemplo función DeleteQuestion

```
//actualizar una pregunta
function updateQuestion(req, res){
    var questionId = req.params.id;
    var update = req.body;

    Question.findByIdAndUpdate(questionId, update, (err,questionUpdated )=>{
        if(err){
            res.status(500).send({message: 'Error en el servidor'});
        }else{
            if(!questionUpdated){
                res.status(404).send({message: 'No se ha actualizado la pregunta'});
            }else{
                res.status(200).send({question : questionUpdated});
            }
        }

    });
}
```

Figure 34: Ejemplo función UpdateQuestion

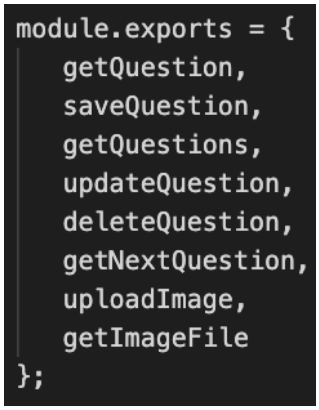


Figure 35: Ejemplo de la exportación de las funciones de un controlador

en otro ficheros. Como en el ejemplo de la figura 35.

Id	Descripción	H-P	Estado
T-017	Desarrollo del controlador de usuario	2h	Completado
T-019	Desarrollo del controlador de formulario	7h	Completado
T-020	Desarrollo del controlador de pregunta	9h	Completado
T-021	Desarrollo del controlador de respuesta	8h	Completado
T-022	Documentar lo implementado	5h	Completado

Table 13: Tabla sprint 4

6.2.5 Sprint 5

Una vez acabado ya totalmente la parte del API Rest, procedimos a iniciar la parte del frontend con angular. Las tareas están descritas en la tabla 14.

Para iniciar la parte del frontend con angular al igual que hicimos con el API REST lo primero que haremos es la creación del proyecto y su estructura, los modelo de datos, componente principal (cabecera) y comenzar con el componente de login, se ha elegido este componente porque esta sería la primera acción

que tendría que hacer un usuario para convertirse en administrador siendo así el componente de prioritario en el desarrollo.

Id	Descripción	H-P	Estado
T-023	Creación del proyecto y su estructura	5h	Completado
T-024	Creación de los modelos de datos	9h	Completado
T-025	Creación del componente principal	5h	Completado
T-026	Implementación del componente login	9h	Completado
T-027	Documentación de lo implementado	4h	Completado

Table 14: Tabla sprint 5

- En primer lugar creamos el proyecto de forma rápido con angular cli [33] con este comando podremos crear un proyecto de forma muy rápida.

$$ng\ new\ mi - nuevo - proyecto - angular \tag{9}$$

  Una vez que con este comando hemos iniciado un nuevo proyecto de angular creamos nuestra estructura de carpetas que vamos a seguir en el proyecto (figura 36). Esta principalmente se divide en tres carpetas, la de componentes, modelos y servicios.

- Se han creado para este sprint los modelos de datos, los cuales son similares a los del API Rest. En la figura 37 podemos encontrar el modelo de datos de answer como ejemplo.

- Tras distintas dudas acerca de cómo plantear la solución más acertada, finalmente en la vista del componente principal 'app.component.html' se ha optado por la opción de desarrollar ahí la cabecera de la aplicación que es el elemento que va a estar en todas las vistas de la aplicación. Podemos observar el resultado en la figura 38.

- A continuación, se procede al desarrollo del componente login, este va a estar compuesto por una vista, una archivo de lógica .ts y un css para dar mejor visibilidad a nuestra vista. Podemos observar dicha estructura en la figura 40. Todos y cada uno de los componentes de este proyecto va a

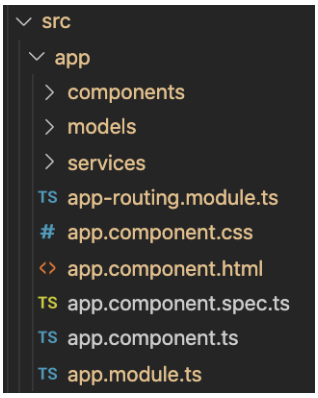


Figure 36: Estructura del frontend

```
export class Answer{
    constructor(
     public _id: string,
     public text: string,
     public fail: string,
     public numselect: number,
     public finalQuestion: boolean,
     public Idquestion: string
    ){}
}
```

Figure 37: Ejemplo del modelo de datos de Answer

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<script  src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>


<nav class="navbar navbar-default">
  <div class= "container">
    <div class="navbar-header" style="margin-right: 20%; float: left">
        <a class="navbar-brand " href=""> Casos clínicos Optica y Optometría – Universidad de Valladolid</a>
    </div>
    <div *ngIf="!identity" class="nav navbar-nav  ml-auto" style="margin-right: -20%;">
      <a class="navbar-item nav-link" [routerLink]="['contact-form']" style="font-size: 150%;"> FAQ</a>
    </div>
    <div *ngIf="!identity" class=" nav navbar-nav navbar-right ml-auto">
      <a class="navbar-brand " routerLink="login" style="font-size: 150%;"> Login</a>
    </div>
    <div *ngIf="identity" class="nav navbar-nav  ml-auto" style="margin-right: -20%;">
      <a class="navbar-item nav-link" [routerLink]="['register']" style="font-size: 150%;"> Registro</a>
    </div>
    <div *ngIf="identity" class=" nav navbar-nav navbar-right ml-auto" style="margin-right: 5%;">
      <a class="navbar-item nav-link active" (click)="logout()" style="font-size: 150%;"> Logout</a>
    </div>
  </div>

</nav>
<body>
<router-outlet></router-outlet>
</body>
```

Figure 38: Vista del componente principal

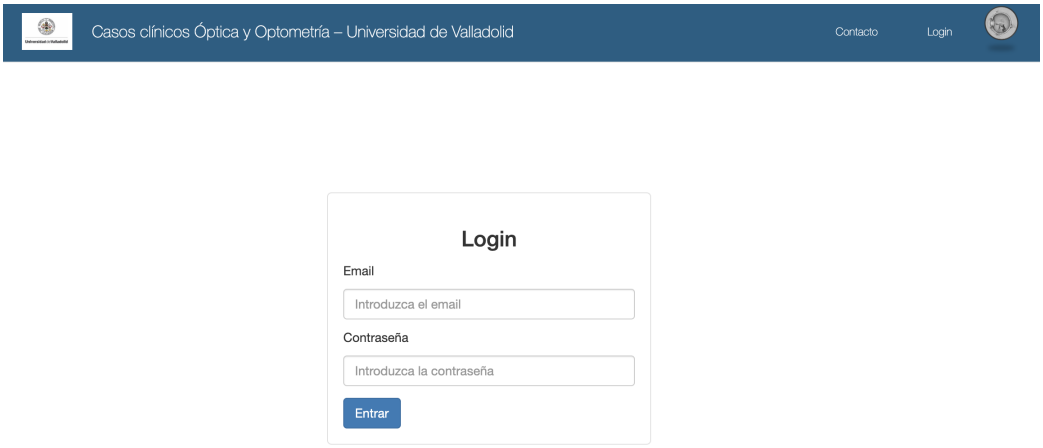


Figure 39: Vista del componente login

seguir la misma estructura.

Este componente se encarga de realizar el inicio de sesión de un usuario que quiera pasar a ser administrador. En la figura 39 podemos ver la vista del componente descrito anteriormente.

6.2.6 Sprint 6

En este sprint se continuará con el desarrollo de los servicios y componentes. Al haber finalizado el componente de login en el sprint anterior, en este se procederá a crear su servicio para así poder conectarse con el API Rest. Todas tareas se encuentran en la tabla 15.

En este sprint debido a que era la primera toma de contacto con lo componentes y servicios de angular, su desarrollo nos ha llevado más tiempo de lo esperado, es decir, hemos tenido que invertir alguna hora más para terminarlo, por lo tanto, no hemos podido acabar todos objetivos. Por orden de prioridad decidimos post-poner la documentación ya que es la tarea menos crítica del sprint.

Figure 40: Estructura de un componente

```
singup(user_to_login, gethash = null){
    if(gethash != null){
      user_to_login.getHash = gethash;
    }

    return this._http.post(this.url+'login',user_to_login);
}
```

Figure 41: Servicio user ejemplo del inicio de sesión

En este sprint se han ido desarrollando por orden los componentes y servicios correspondientes a las funcionalidades que tendría disponible el administrador una vez se haya identificado. Estas son la creación de un caso con el componente create-form y con el servicio form. Y de esta misma forma los componentes show-forms y edit-forms realizando las modificaciones necesarias en el servicio de form que se encarga de todo lo que respecta al tratamiento de los casos.

- En primer lugar procedimos al desarrollo del servicio user, este actúa cómo conector mediante peticiones http con el API Rest. Podemos observar un ejemplo la figura 41 , en esta figura podemos observar como se implementaría el servicio para el inicio de sesión de un usuario.

- A continuación, desarrollamos el componente create-form. Este se encarga de proporcionar una vista donde el usuario pueda crear un caso, este caso esta compuesto por su titulo y nivel de dificultad. Se puede observar un ejemplo de la vista en la figura 42.

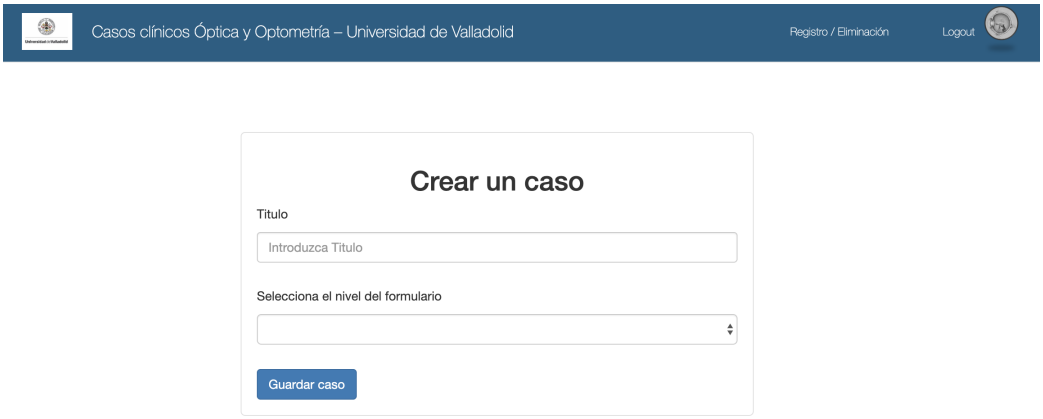


Figure 42: Vista para crear un caso

- Desarrollamos el servicio form para que se conecte con el API Rest para la creación del formulario.

- Una vez terminado esto, procedemos a desarrollar el componente show-forms, el cual se encarga de mostrar el listado de todos los formularios o casos que haya en la base de datos, estos se mostraran separados por niveles y mediante paginación, es decir, existen varias paginas en cada una de ellas puede haber hasta 4 casos.
  Podemos observar unos ejemplos de la vista de dicho componente en las figura 43 y 44.

- Por último, desarrollamos el componente edit-forms, que va a proporcionar una vista donde el usuario va a poder editar un caso, ya sea modificando su nivel de dificultad o su titulo. A continuación, podemos ver una vista de ejemplo del componente en la figura 45.

- También, haremos los ajustes necesarios en el servicio form para que los

Figure 43: Venta principal de la aplicación web

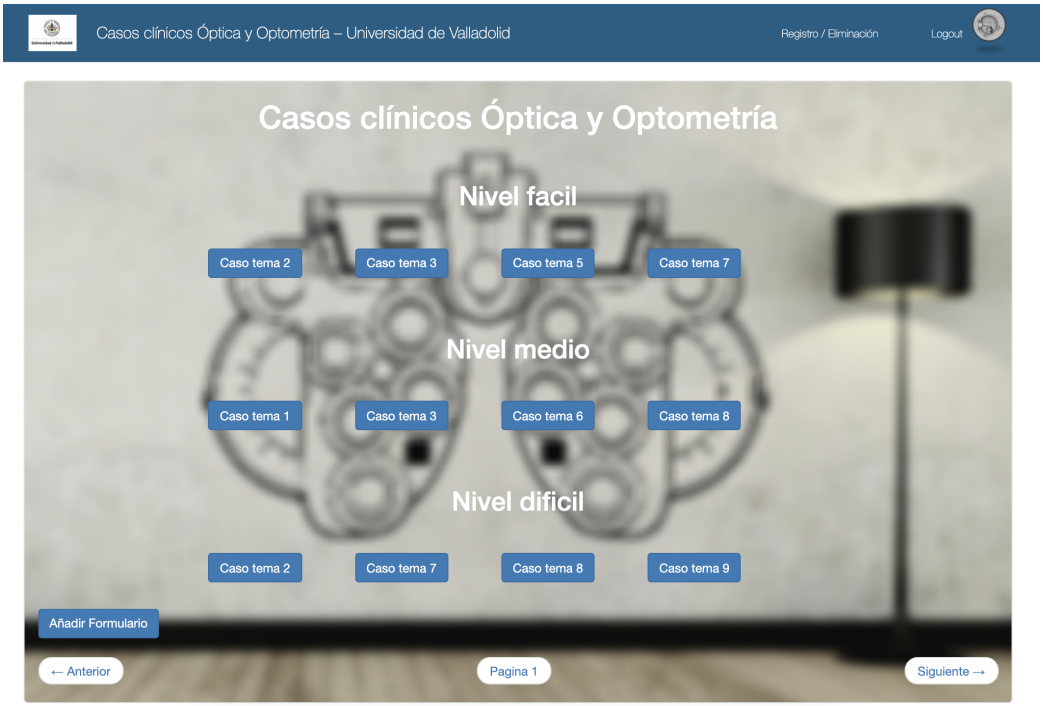


Figure 44: Venta principal de la aplicación web para el administrador

Figure 45: Vista para edición de un caso

componentes edit-forms y show-forms se conecten con el backend.

- Los opciones de edición, borrado y creación solo van a estar disponibles para el administrador.

6.2.7 Sprint 7

En este Sprint se debe terminar la documentación que no pudimos acabar en el sprint anterior y una vez acabada continuar con el desarrollo de componentes y servicios. Las tareas desarrolladas se encuentran en la tabla 16.

En este sprint como en los anteriores procedimos a continuar con los desarrollos de los componentes y creación o modificaciones pertinentes en los servicios necesarios siguiendo los criterios explicados anteriormente en cuanto al orden de elección. Agregando nuevas funcionalidades a la web como creación de preguntas, registro de administradores y mostrar tanto una pregunta de forma individual

Id	Descripción	H-P	Estado
T-028	Desarrollo del servicio user para el login	3h	Completado
T-029	Desarrollo del componente create-form	7h	Completado
T-030	Desarrollo del servicio form	3h	Completado
T-031	Desarrollo de componente de show-forms	7h	Completado
T-032	Desarrollo de componente de edit-forms	7h	Completado
T-033	Ajustes en el service form	3h	Completado
T-034	Documentar lo desarrollado	2h	En progreso

Table 15: Tabla sprint 6

y con más detalle, cómo mostrar las preguntas de un caso concreto.

A continuación, describiremos de forma un poco más detallada cada uno de los desarrollos realizados:

- Tras finalizar la documentación del anterior sprint comenzamos con el desarrollo del componente register. Este se encarga de proporcionar la opción a un administrador de poder registrar nuevos administradores en el sistema, así como también eliminar alguno de los existentes.
  Podemos observar una vista como ejemplo de este componente en la figura 46.

- Se han hecho las modificaciones necesarias en el servicio user para que se conecte con el API Rest y funcione correctamente el registro y eliminación de administrados en la base de datos.

- Gracias al componente create-question, el usuario administrador va a poder crear una pregunta que corresponda a un caso. Simplemente rellenado los campos de las vista del componente.
  Podemos observar una vista como ejemplo de este componente en la figura 47.

- Desarrollo del componente show-questions, que nos mostrará todas las preguntas de un caso concreto, que previamente se ha seleccionado, en esta vista podremos acceder a cada de las preguntas, o borrar o editar el

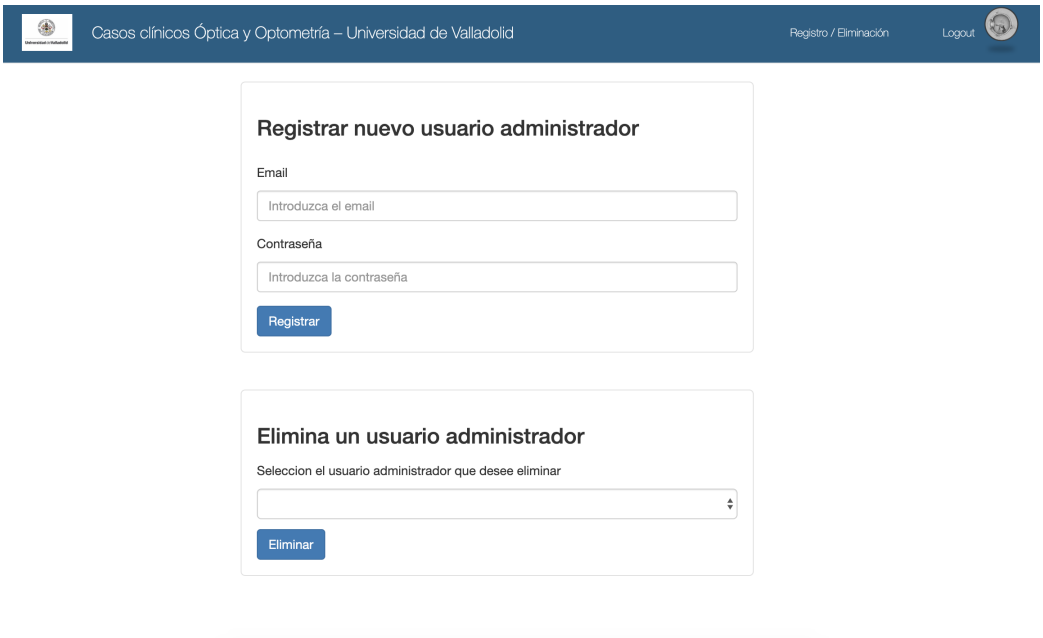


Figure 46: Vista para el registro o eliminación de usuarios administradores

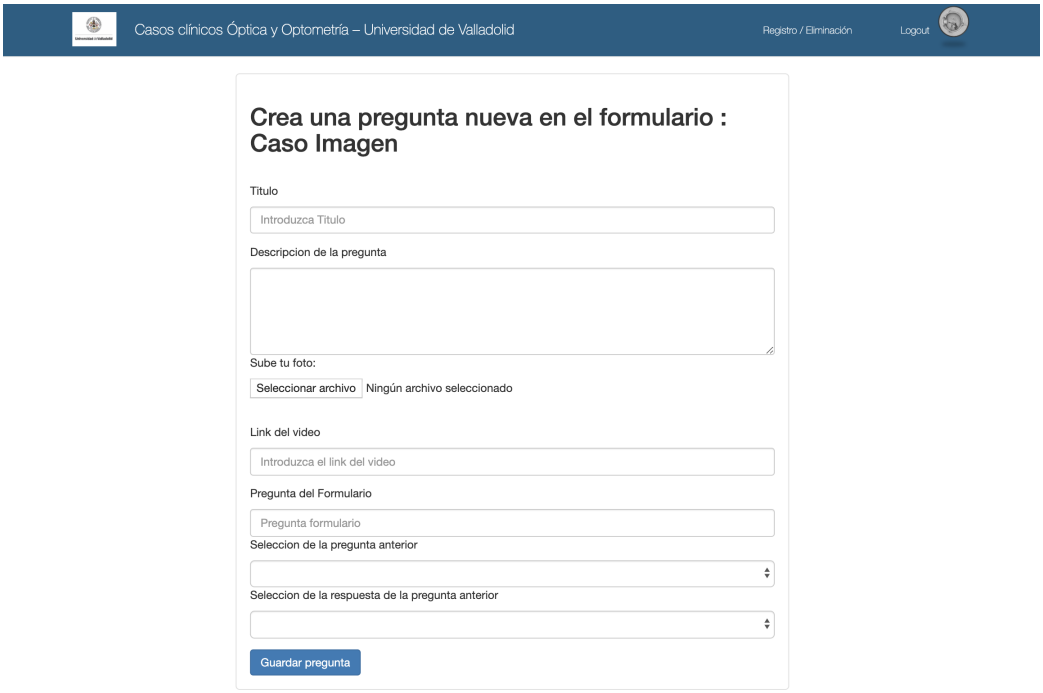


Figure 47: Vista para la creación de preguntas de un caso

Figure 48: Preguntas de un caso concreto

caso seleccionado previamente.
En la figura 48 podemos observar un ejemplo real de la vista de dicho componente.

- En el componente show-question podemos observar los campos de esa pregunta, así como borrarla, o añadir respuestas a esa pregunta (sino contiene ninguna) o editar la pregunta y sus respuestas (si contiene alguna respuesta).
  Se puede observar en la figura 49 un ejemplo de la vista del componente show-question.

- Por último, desarrollamos el servicio question para que todos los componentes anteriores se conecten con el API Rest. Además de documentar de lo implementado.

6.2.8 Sprint 8

Para el Sprint número 8, el cual es el último previsto inicialmente, lo esperado es terminar de desarrollar los últimos componentes y servicios así como la documentación que falta por realizar. Las tareas están descritas en la tabla 17.

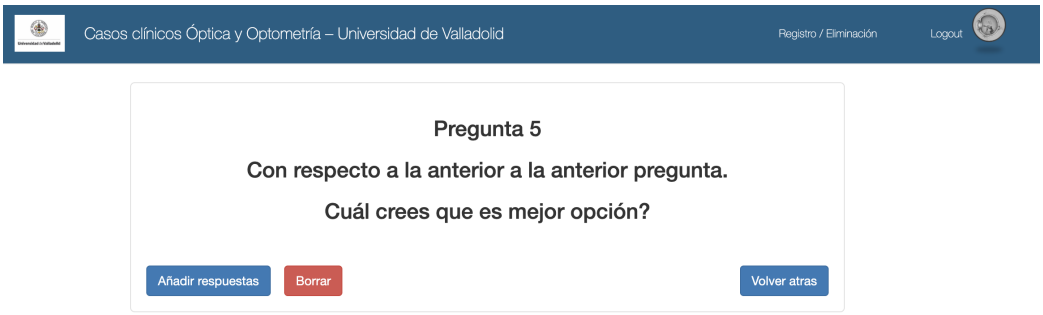


Figure 49: Titulo, descripción y pregunta de una cuestión

Id	Descripción	H-P	Estado
T-034	Documentar lo desarrollado en el sprint 6	2h	Completado
T-035	Desarrollo del componente register	5h	Completado
T-036	Modificar el servicio user	3h	Completado
T-037	Desarrollo del componente create-question	5h	Completado
T-038	Desarrollo del componente show-questions	5h	Completado
T-039	Desarrollo del componente show-question	5h	Completado
T-040	Desarrollo del servicio question	4h	Completado
T-041	Documentar lo implementado	4h	Completado

Table 16: Tabla sprint 7

Debido a una serie de ampliaciones propuestas por el cliente se ha tenido que prorrogar en 2 semanas la duración del proyecto sobre la planificación inicial, y por lo tanto, crear otro sprint más.

En este sprint al igual que en los anteriores continuamos añadiendo funcionalidades, en este caso añadimos las funcionalidades de creación de respuesta, edición de preguntas con sus respuestas, una vista para el momento en el que un usuario (sin privilegios de administrador) finalizase un caso respondiendo a las preguntas de forma correcta, y la vista que mostraría las preguntas donde tiene que responder dicho usuario, el criterio de elección ha sido el mismo que en anteriores sprints.

- Se ha desarrollado el componente de create-answer, este proporciona una vista al administrador (el usuario normal no tiene acceso) para poder crear las distintas respuestas que pertenecen a una pregunta rellenando los campos necesarios.
  En la figura 50 hay una figura de ejemplo de este componente.

- Se ha desarrollado edit-question-answers, en este componente se da la posibilidad de poder editar tanto los campos de una pregunta y los de las respuestas dentro de una vista, siempre y cuando ya existan respuestas.
  En la figura 51 se puede ver un ejemplo de la vista de este componente.

- Se ha desarrollado end-form, este componente es una vista que se mostrará cuando un usuario haya acabado el formulario de forma correcta, o cuando haya seleccionado una respuesta que no lleva a ninguna salida mostrandole un texto especifico de porque está mal esa repuesta escogida, dándole la posibilidad de volver a la anterior.
  En este componente podemos encontrar dos tipos de vistas, como en las figuras 52 y 53 de ejemplo.

- Desarrollo de show-question-user, esta nos proporciona una vista en la que vamos a ir viendo cada una de las preguntas de un caso concreto con sus respuestas, según las opciones que vayamos seleccionando nos ira mostrando una u otra pregunta.

- Por último, se desarrollo el servicio answer para conectar los componentes con el API Rest, así como las modificaciones necesarias en el servicio question para añadirle nuevas funcionalidades.

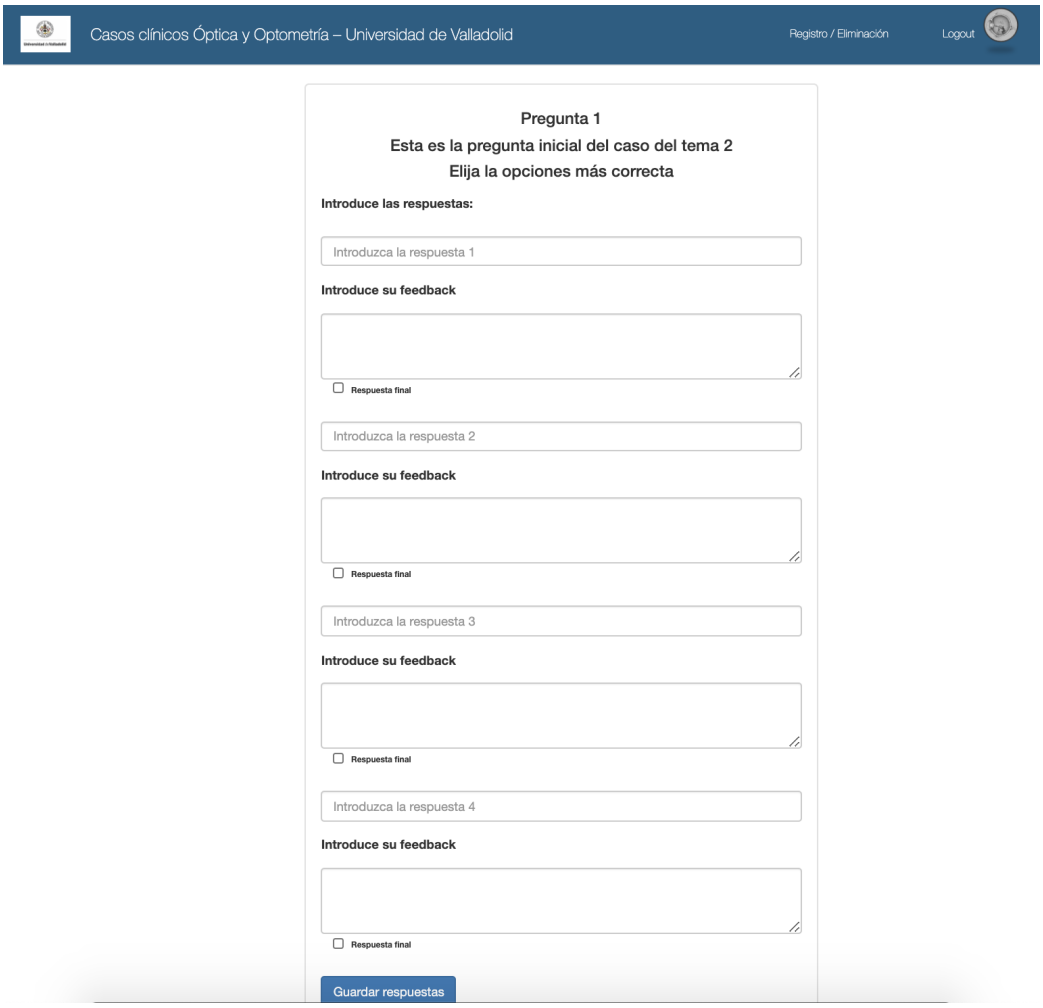


Figure 50: Añadir respuestas

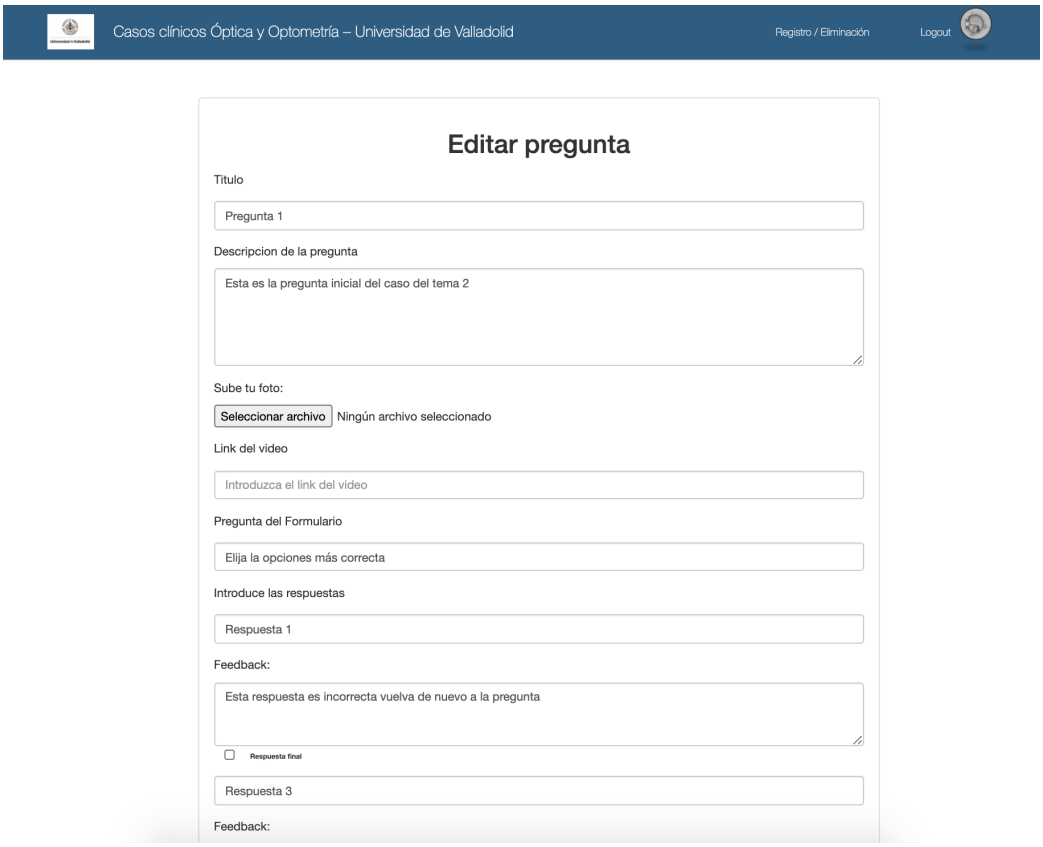


Figure 51: Edición de una pregunta

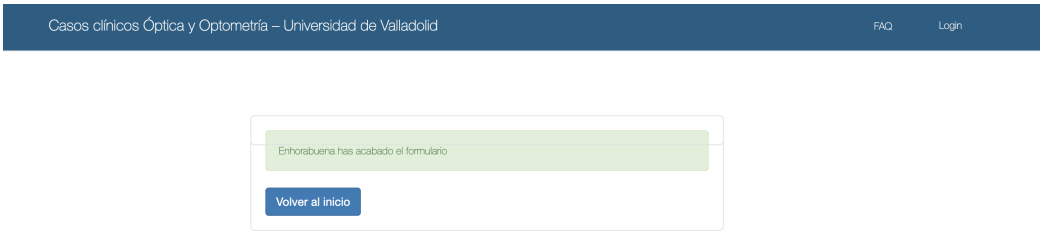


Figure 52: Vista del fin del caso con éxito

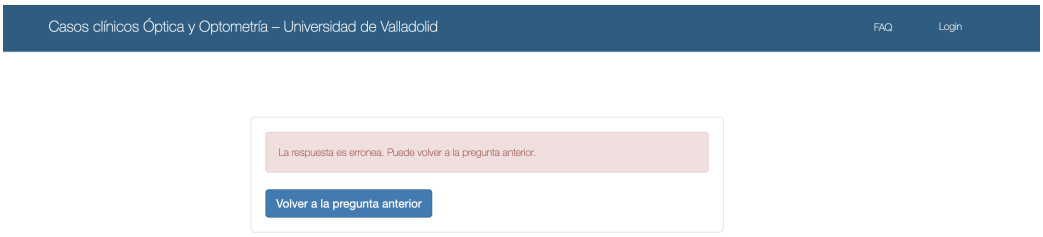


Figure 53: Vista al elegir una respuesta errónea sin salida

Id	Descripción	H-P	Estado
T-042	Desarrollo del componente create-answer	5h	Completado
T-043	Desarrollo del componente edit-question-answers	5h	Completado
T-044	Desarrollo del componente end-form	5h	Completado
T-045	Desarrollo del componente show-question-user	5h	Completado
T-046	Desarrollo el servicio answer	5h	Completado
T-047	Modificaciones en el servicio question	3h	Completado
T-048	Documentación de de lo implementado	4h	Completado

Table 17: Tabla sprint 8

6.2.9 Sprint 9 (Adicional)

En este sprint ha tenido que ser añadido de forma adicional a pesar de que no estaba en la planificación inicial. Esto se debe a algunas ampliaciones y ajustes que se han tenido que ir haciendo a lo largo del desarrollo, lo cual ha provocado que el desarrollo del proyecto haya tenido que ser aumentado en 2 semanas. Las tareas que llevamos a cabo en este último sprint para finalizar el proyecto se pueden observar en la tabla 18.

Las tres principales mejoras que se han solicitado y las cuales han provocado este sprint adicional son:

- El desarrollo de una vista donde cualquier usuario del sistema pueda completar los campos de un formulario donde envie cuestiones, sugerencias o errores al administrador, dichas cuestiones, sugerencias o errores se le enviarán vía email.

- Se ha tenido que hacer una serie de modificaciones y ampliaciones en el desarrollo para que las preguntas pudieran tener adicionalmente tanto una imagen como un vídeo si así se deseaba.

- Se ha añadido también la posibilidad de ver una gráfica de estadísticas con el número de veces que han sido seleccionadas las respuestas de cada pregunta en cada caso.

A continuación, mostramos de forma un poco más detallada las tareas realizadas en este sprint.

- Para el desarrollo de la primera mejora citada anteriormente, comenzamos con el desarrollo de una vista donde el usuario debía rellenar lo campos:
  - Nombre
  - Asunto
  - E-mail
  - Mensaje

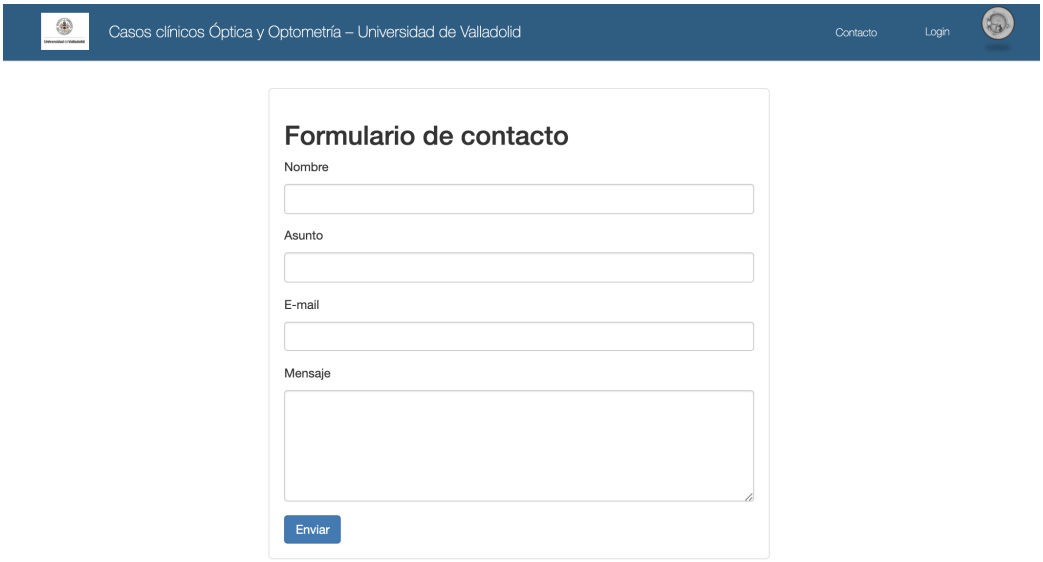


Figure 54: Vista del formulario de contacto

Una vez rellenados estos campos dispondrá de un botón enviar para así hacérselo llegar al administrador mediante el correo electrónico.
En la figura 54 se puede observar una vista de ejemplo del formulario de contacto.

- A continuación, se desarrolla el servicio message que va a conectar con el router del API Rest. Este se va a desarrollar también en este sprint y a su vez este conectara con el fichero Configuremessage.js del API Rest. Una vez acabado esto, esta mejora se habrá llevado a cabo de forma completa.

- Ahora vamos a por la segunda mejora, en este caso ha habido que realizar distintas modificaciones en distintos lugares:

  – Se han modificado los modelos de question, tanto del frontend como del API Rest, añadiendo los dos campos necesarios para poder tener un vídeo y una imagen en una pregunta.

  - Modificaciones en las vistas, tanto de edición como creación en la parte del administrador, y modificación en la vista que muestra las preguntas al usuario mientras resuelve un caso en la parte del usuario normal.

  - Modificación en los ficheros de lógica de los componentes en los que estén involucradas las preguntas, como por ejemplo create-question.ts.

  - Los servicios del frontend no han requerido modificaciones.

  - En la parte del API Rest en primer lugar hemos tenido que realizar modificaciones en el router de question, así como en el controlador de question, añadiendo las correspondientes funciones para añadir dicha funcionalidad.

- En tercer lugar se desarrolló la gráfica de estadísticas, las preguntas con sus estadísticas se mostrarán de cuatro en cuatro. Para hacer esto de forma resumida se han realizado distintos pasos [36]:

  - Instalación de las librerías necesarias mediante los siguientes comandos e importarlos en el archivo de lógico del componente elegido:

$$npm\ install\ --save\ ng2-charts \tag{10}$$

$$npm\ install\ --save\ chart.js \tag{11}$$

  - Después, recoger todos los datos necesarios para crear las gráficas y mostrar las mismas como se puede observer en la figura 55.

- Para finalizar documentamos todo lo que faltaba por documentar para de esta forma acabar el proyecto al completo. Algunas cosas que documentamos fueron conclusiones, testing y el plan de riesgos y presupuestos.

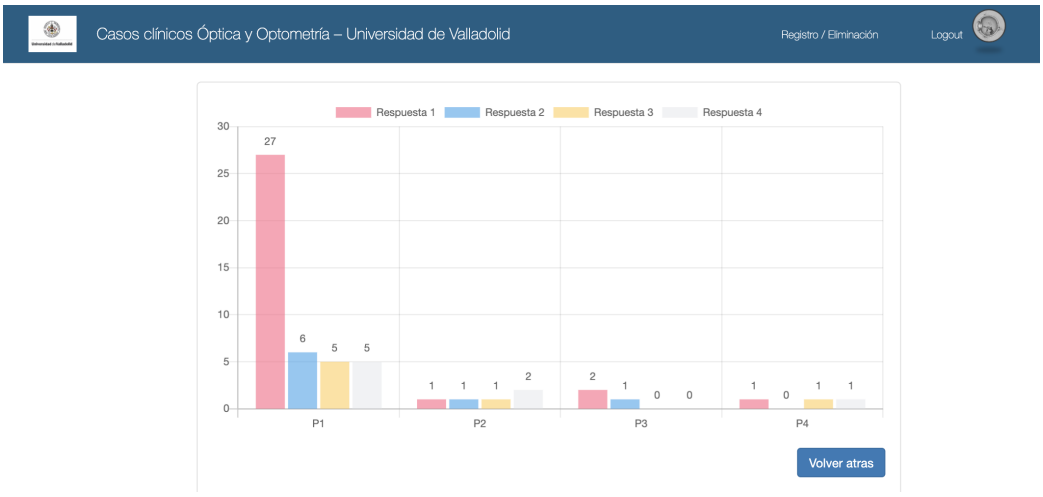


Figure 55: Estadísticas de las preguntas

Id	Descripción	H-P	Estado
T-049	Desarrollo del componente contact-form	5h	Completado
T-050	Desarrollo del servicio message	3h	Completado
T-051	Desarrollo del fichero ConfigureMessage en el API Rest	3h	Completado
T-052	Desarrollo del router para lo desarrollado antes	2h	Completado
T-053	Ajustes para que una pregunta tenga imagen y video	8h	Completado
T-054	Añadir funcionalidad de gráfica de estadísticas	8h	Completado
T-055	Documentación de la memoria	8h	Completado

Table 18: Tabla sprint 9

Empleado	Horas invertidas	Coste/hora	Coste final
Team Member	208 horas	30	6240 euros
Scrum master	70 horas	30	2100 euros
Product Owner	59 horas	40	2360 euros

Table 19: Tabla coste final del personal

7 Presupuesto

7.1 Tipos de costes

El plan de presupuesto se ha dividido en distintos apartados según el tipo de coste que nos podemos encontrar en un proyecto.

7.1.1 Coste del personal

Después de informarme acerca de salarios en distintas fuentes de internet [34] [35]. Observamos que el sueldo aproximado de un desarrollador web en unos 20.000 euros al año, este se asociaría al team member. Por lo que el sueldo es 10.41 euros/hora. El scrum master va a tener un sueldo similar al mencionado por el team member. Y por último, el product owner tiene un sueldo más elevado, este es de 16 euros la hora. Aunque a estos sueldos hay que añadirle otros gastos aparte, como el seguro médico. Por lo tanto, el coste por hora va a ser más elevado.

En la tabla 19 podemos observar la cantidad de horas invertidas por cada uno y de esta forma poder determinar el coste del personal.

El coste final de proyecto que ha tomado 337 horas es de 10700 euros en lo que al personal respecta.

7.1.2 Coste del hardware

Para el proyecto hemos necesitado un ordenador portátil. En este caso el equipo usado es un Macbook pro de 13", cuyo coste es de 1500 euros.

El uso de dicho activo durante los meses que ha durado el proyecto, cuya duración es 337 horas, nos deja como coste final por su uso 300 euros.

Tipo de coste	Coste
Personal	10700 euros
Hardware	300 euros
Software	0 euros
Indirectos (15%)	1614 euros
Coste final	12,614 euros

Table 20: Tabla coste final del proyecto

7.1.3 Coste final

El coste de software ha sido nulo debido a que no se ha hecho uso de ninguna herramienta de pago para llevar a cabo el desarrollo.

El resto de costes ya han sido mencionados anteriormente por lo que el coste final sera la suma de todos estos, más un pequeño porcentaje de costes indirectos. El resumen de todos los costes aparece en la tabla 20.

8 Testing

En cuanto a las pruebas de testeo diferenciaremos dos tipos de pruebas:

- Caja blanca: Estas pruebas son aquellas que en las que se prueba el código de la aplicación de forma interna, para la realización de estas pruebas se requiere conocimientos acerca de los frameworks usados en el desarrollo.

  Estas pruebas se han ido realizando a medida que se iban desarrollando las funcionalidades, probando todas y cada una de las posibilidades de fallo que pudiese haber.

- Caja negra: Son las pruebas que se llevan a cabo mediante el uso de la interfaz gráfica, sin tener en cuenta nada del funcionamiento interno.

  En este caso se va a describir todos y cada uno de los test o pruebas en la aplicación con el fin de poder comprobar que funciona bien, es intuitiva y fácil de usar por cualquier usuario que previamente no la ha usado nunca. Las pruebas las vamos a encontrar de la tabla 21 a la tabla 30. Cada una de estas tablas va estar compuesta por un nombre, identificador, importancia, observaciones , descripción de la prueba, resultado esperado,pre-condición y resultado obtenido.

Nombre	Logueo en la aplicación
Identificador	P001
Importancia	Alta
Observaciones	Ha tardado unos segundos en encontrar la opción.
Descripción de la prueba	Un usuario debe ir a ventana de login e iniciar sesión
Resultado esperado	El usuario se loguea sin problema
Resultado obtenido	El usuario lo ha hecho bien.
Precondición	Ninguna

Table 21: Test: Test de logueo

Nombre	Enviar sugerencia o duda
Identificador	P002
Importancia	Alta
Observaciones	Hemos cambiado el nombre a contacto en la vista para que sea más visible.
Descripción de la prueba	El usuario debe encontrar el formulario de contacto para enviar una duda o sugerencia
Resultado esperado	El usuario lo encuentra y lo envía
Resultado obtenido	El usuario lo ha llevado a cabo sin problemas.
Precondición	Ninguna

Table 22: Test: Enviar una sugerencia o duda

Nombre	Acceder a un caso
Identificador	P003
Importancia	Alta
Observaciones	Ninguna
Descripción de la prueba	Acceder a un caso para resolverlo
Resultado esperado	El usuario accede a uno
Resultado obtenido	Ha accedido de forma correcta y rápida.
Precondición	Ninguna

Table 23: Test: Acceder a un caso en la ventana inicial

Nombre	Añadir a nuevo administrador
Identificador	P004
Importancia	Media
Observaciones	Hemos cambiado el nombre de la opción a Registro / Eliminación para que sea más intuitivo
Descripción de la prueba	El administrador debe encontrar dónde se añaden los nuevos administrador y añadir uno
Resultado esperado	El administrador añade a uno nuevo
Resultado obtenido	El usuario ha añadido el nuevo administrador.
Precondición	Debe estar identificado previamente

Table 24: Test: Añadir un nuevo administrador

Nombre	Eliminar a un administrador
Identificador	P005
Importancia	Media
Observaciones	Ninguna
Descripción de la prueba	El administrador debe encontrar dónde se eliminan los administradores y eliminar uno
Resultado esperado	El administrador elimina a uno
Resultado obtenido	Se ha eliminado correctamente.
Precondición	Debe estar identificado previamente

Table 25: Test: Eliminar a un administrador

Nombre	Crear un nuevo caso
Identificador	P006
Importancia	Alta
Observaciones	Ninguna
Descripción de la prueba	El administrador debe encontrar dónde se crean los casos y crear uno
Resultado esperado	El administrador crea un caso
Resultado obtenido	El usuario crea un caso
Precondición	Debe estar identificado previamente

Table 26: Test: Crear un caso

Nombre	Editar y eliminar caso
Identificador	P007
Importancia	Alta
Observaciones	Ninguna
Descripción de la prueba	El administrador debe saber cómo editar y luego borrar un caso
Resultado esperado	El administrador edita y borra un caso
Resultado obtenido	Edita y elimina un caso de forma correcta.
Precondición	Debe estar identificado previamente

Table 27: Test: Editar y eliminar un caso

Nombre	Crear una nueva pregunta con sus respuestas
Identificador	P008
Importancia	Alta
Observaciones	Ninguna
Descripción de la prueba	El administrador debe ser capaz crear una pregunta con sus respuestas
Resultado esperado	El administrador crea una pregunta con sus respuestas
Resultado obtenido	Crea una pregunta correctamente
Precondición	Debe estar identificado previamente

Table 28: Test: Creación de una nueva pregunta

Nombre	Edición y borrado de una pregunta
Identificador	P009
Importancia	Alta
Observaciones	Ninguna
Descripción de la prueba	El administrador debe ser capaz editar un pregunta y luego borrarla
Resultado esperado	El administrador edita y borra una pregunta
Resultado obtenido	El administrador edita y borra la pregunta correctamente.
Precondición	Debe estar identificado previamente

Table 29: Test: Edición y borrado de una pregunta

Nombre	Estadísticas de una pregunta
Identificador	P010
Importancia	Media
Observaciones	Ninguna
Descripción de la prueba	El administrador debe encontrar donde ver las estadísticas de una pregunta
Resultado esperado	El administrador observa las estadísticas de una pregunta
Resultado obtenido	El administrador encuentra y observa las estadísticas de una pregunta.
Precondición	Debe estar identificado previamente

Table 30: Test: Estadísticas de una pregunta

9 Conclusiones

En esta sección se va a tratar de estudiar todos los resultados obtenidos en el desarrollo de este proyecto.

En primer lugar, decir que el desarrollo se ha llevado a cabo de manera satisfactoria. Por consiguiente se han conseguido llevar a cabo todos lo objetivos del proyecto aunque para esto se tuviera que alargar dos semanas más el desarrollo. Esto no se debe a una mala planificación, sino a un ligero aumento de requisitos, siendo la planificación bastante correcta.

Ahora vamos a compararlo con los objetivos iniciales del proyecto:

- Desarrollo de una aplicación que pueda usar un usuario normal. Este ha sido un objetivo que se ha logrado mediante el diseño de interfaces intuitivas y fáciles de usar para cualquier persona.

- Aplicación de una metodología ágil en el proyecto. Esto se ha aplicado con existo para así poder abordar de mejor forma el proyecto.

- Se han podido cubrir todas las necesidades que se requerían en la aplicación, en la que podemos distinguir dos tipos de usuarios: el usuario

normal y el administrador. Cada uno de estos puede llevar a cabo todas sus funciones requeridas.

En cuanto a los objetivos personales, todos ellos han sido realizados con creces, estos son:

- He adquirido muchos conocimientos para el desarrollo de aplicaciones web, en concreto con el MEAN Stack (que previamente no había usado), ya descrito en el apartado de el Estado del arte, con el que podemos hacer un desarrollo full stack.
- Mejorar mi entendimiento de una metodología ágil como SCRUM.
- Me he acercado mucho más a cómo es un desarrollo completo de un proyecto, pudiendo ver todas y cada de sus fases.

En conclusión, en cuanto a lo personal ha sido también un TFG muy satisfactorio.

9.1 Lineas futuras

En este apartado citaremos y explicaremos de forma breve alguna de las mejoras futuras que se podrían realizar con el fin de mejor la aplicación web añadiendo nuevas funcionalidades.

Una de las posibles mejoras es "gamificar" [37] la aplicación web. Esto consiste en usar de técnicas u opciones que normalmente encontramos en un juego y añadirlas a la aplicación web con el fin de fomentar el uso de la misma, motivando más a los alumnos en este caso a conseguir unos objetivos.

Para conseguir esto se proponen distintos cambios en la aplicación web:

- Necesidad de que cada usuario tenga una cuenta propia.
- Guardar en cada cuenta sus registros individuales, como pueden ser estadísticas en cuanto a preguntas acertadas, casos resueltos, tiempo empleado en la resolución...

- Con el fin de fomentar el uso de la aplicación se añadiría una opción de recompensas, estas recompensas serían determinadas por los administradores. Obteniéndolas los usuarios que completasen los requisitos de la recompensa. Esto podría dar lugar a un sistema de puntuaciones en el que conseguir un recompensa te de además de la recompensa una puntuación que se sumará a tu puntuación individual total.

- Añadir retos, con el fin de que los alumnos compitan de forma directa entre ellos, para así poder obtener una puntuación más alta. Estos retos podrían ser: Resolver el caso en un tiempo determinado, sin ninguna respuesta errónea...

- "Gamificar" la interfaz, es decir, hacer cambios en las vistas con las que interacciona el usuario con el fin de que estas se se asemejen a un videojuego.

- Por último, se pondría una tabla de rankings donde todos los usuarios podrían ver su posición en la tabla en función de la puntuación total individual de cada uno. También cabe la posibilidad de hacer otros tipos de rankings a parte de la puntuación total, todo esto dependería de los datos que se recolecten.

10 ANEXO I: Manual de Usuario

En este manual de usuario se dará una breve explicación de cada una de las partes de la web. Siendo una guía de uso para un usuario sin experiencia con la web.

Ventana principal

En esta ventana podremos observar los distintos casos que ya hay creados a los cuales podremos acceder. Esta vista tiene dos casos posibles, los cuales son:

- Identificado previamente: Si el usuario esta identificado previamente con esta podrá, en primer lugar desde la cabecera podrá acceder al registro o eliminación de cuentas de administradores y cierre de sesión.

  Por otra parte se podrá acceder a cada caso que nosotros escojamos para su edición, borrado... así como con el botón "Añadir formulario" podemos añadir algún caso más, todo esto lo podemos observar en la figura 56.

- Sin identificar: En este caso el usuario no se ha identificado previamente, en este caso podrá acceder en la cabecera al formulario de contacto donde podrá enviar alguna sugerencia o fallo encontrado, o la ventada de login donde puede identificarse para convertirse en administrador.
  También, podrá acceder a cada caso de la ventana para proceder a la resolución del mismo, esto se puede observar en la figura 57.

Login

Ventana donde un usuario que previamente no está identificado se puede identificar y adquirir así los privilegios de administrador, esto se realiza mediante un e-mail y una contraseña. Se puede acceder a ella en la cabecera, figura 58.

Formulario de contacto

En este caso el usuario no está previamente identificado. Se accede mediante la cabecera. Gracias a esta opción se le puede mandar una sugerencia, duda o error al administrador vía correo rellenando (figura 59):

- Nombre
- Asunto

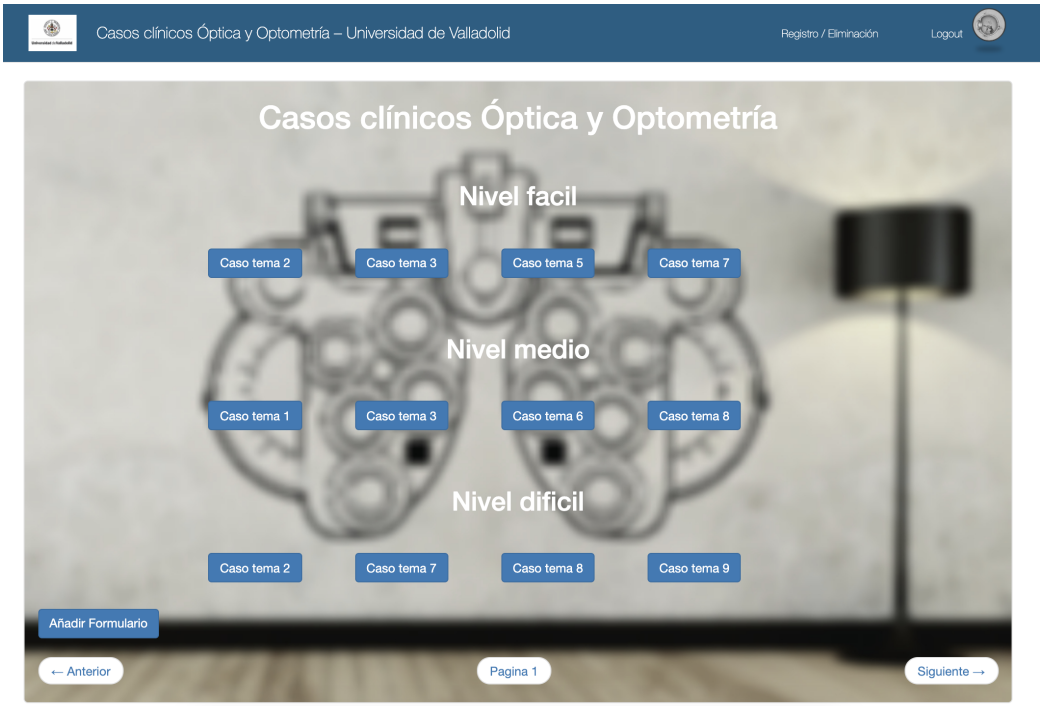


Figure 56: Ventana principal de administrador

Figure 57: Ventana principal sin identificación

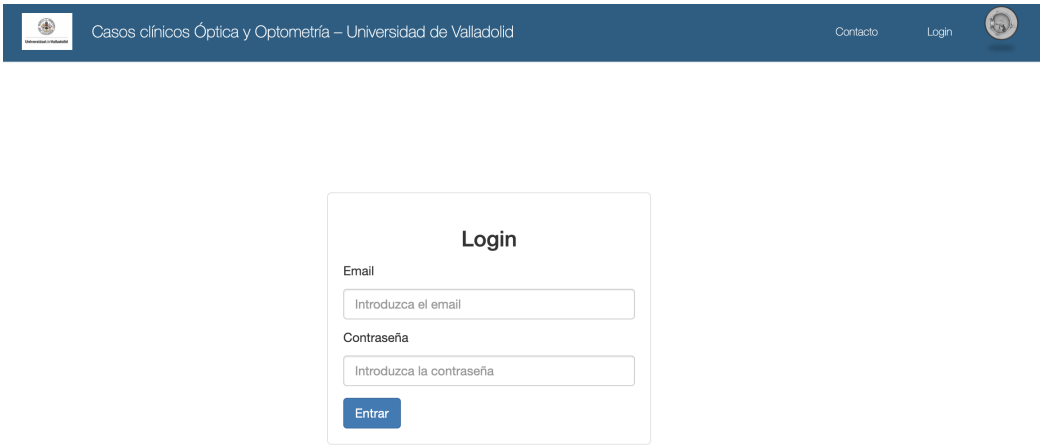


Figure 58: Identificación del usuario

- E-mail
- Mensaje

Registro/eliminación de administradores

A esta opción se accede en la cabecera, para poder tener visible esta opción es necesario que el usuario sea administrador. En ella se podrá tanto añadir administradores rellenando los campos e-mail y contraseña como eliminar alguno de los ya existentes, todo lo mencionado se puede observar en la figura 60.

Acceso a un caso concreto

Tras elegir como administrador un caso concreto accederemos a esta ventana. En ella podremos editar o borrar el caso al que hemos accedido, acceder a las estadísticas de las preguntas de la página en la que nos encontramos, así como crear una nueva pregunta o acceder a una pregunta concreta de ese caso (figura 61).

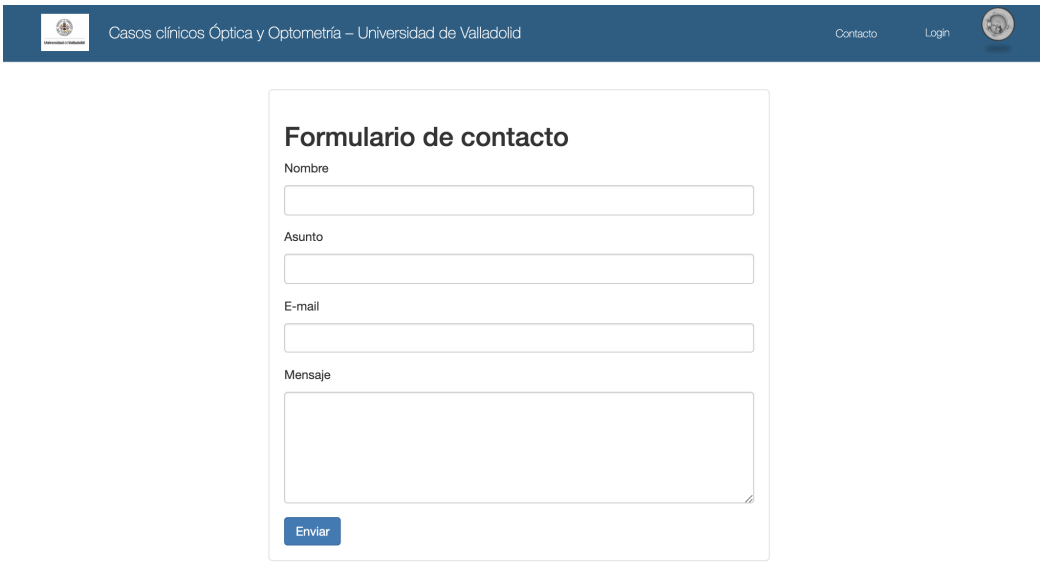


Figure 59: Formulario de contacto

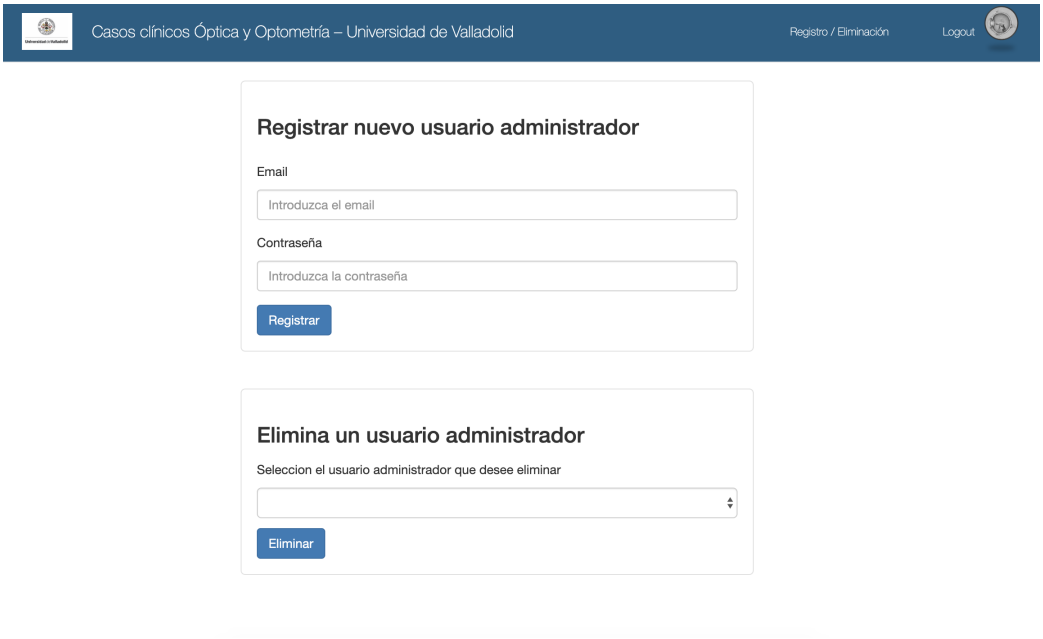


Figure 60: Registro/Eliminación de usuario

Figure 61: Acceso a un caso concreto

Figure 62: Editar un caso

Edición de un caso

Para acceder a esta opción hay que ser administrador, en ella podemos cambiar los valores del caso, el titulo y el nivel del mismo (figura 62).

Estadísticas

Al acceder a esta opción podemos observar las estadísticas de las preguntas de la pagina en la que nos encontramos, se puede observar en la figura 63. Solo puede acceder un administrador.

Crear una pregunta

Al escoger esta opción mencionada anteriormente, podemos crear una pregunta para un formulario concreto, es necesario ser administrador para hacerlo,la crearemos rellenando los campos (figura 64):

- Titulo

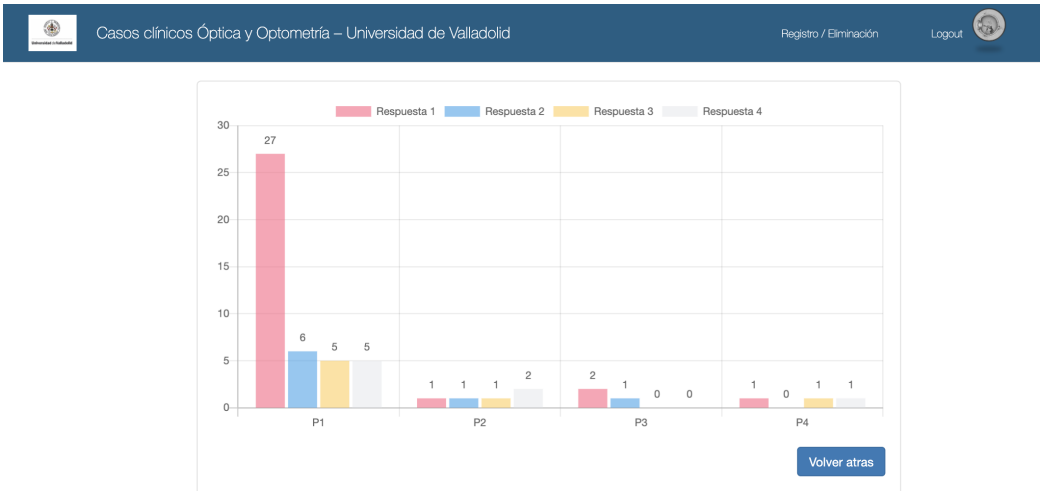


Figure 63: Estadísticas

- Descripción de la pregunta
- Imagen
- Enlace del vídeo
- Pregunta
- Pregunta y respuesta con la que se enlaza

Ver una pregunta

Una vez una pregunta existe podemos acceder a ella viendo sus datos más relevantes, así como la posibilidad de añadir las respuestas (si estas no existen), editar la pregunta o borrar la misma (figura 65).

Añadir respuestas

En esta opción se podrán añadir las respuestas correspondientes a una pregunta de un caso, es necesario ser administrador para poder hacerlo. Para ello será necesario escribir todas las respuestas de esa pregunta, en el caso de ser una respuesta errónea se puede añadir un feedback, también se puede marcar como

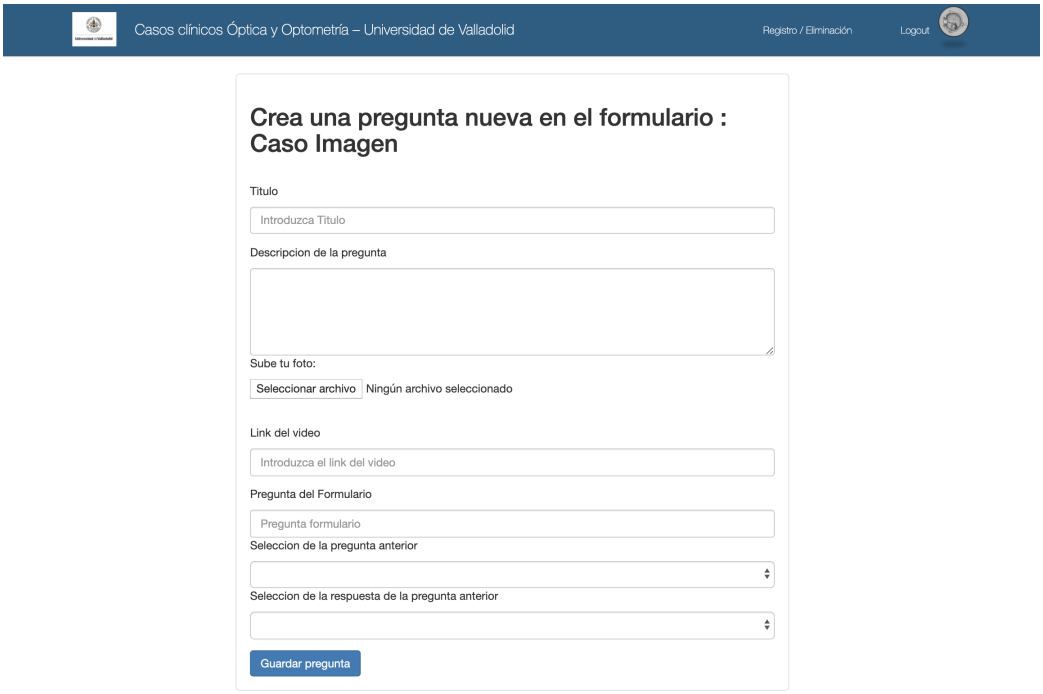


Figure 64: Creación de una pregunta

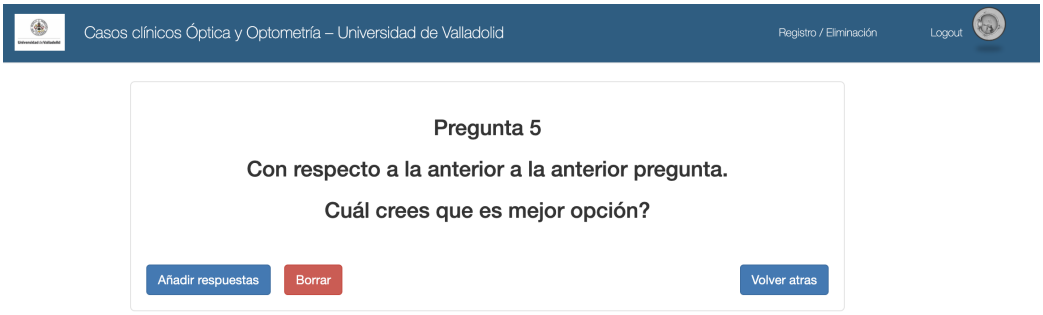


Figure 65: Ver una pregunta

respuesta final si, es decir, esa respuesta va a ser la última en la resolución de un caso (figura 66).

Editar una pregunta

Una vez ya ha sido creada la pregunta y sus respuestas podemos editarla, pudiendo modificar cada uno de los campos introducidos anteriormente, se puede observar una figura de ejemplo en la figura 67(es necesario ser administrador).

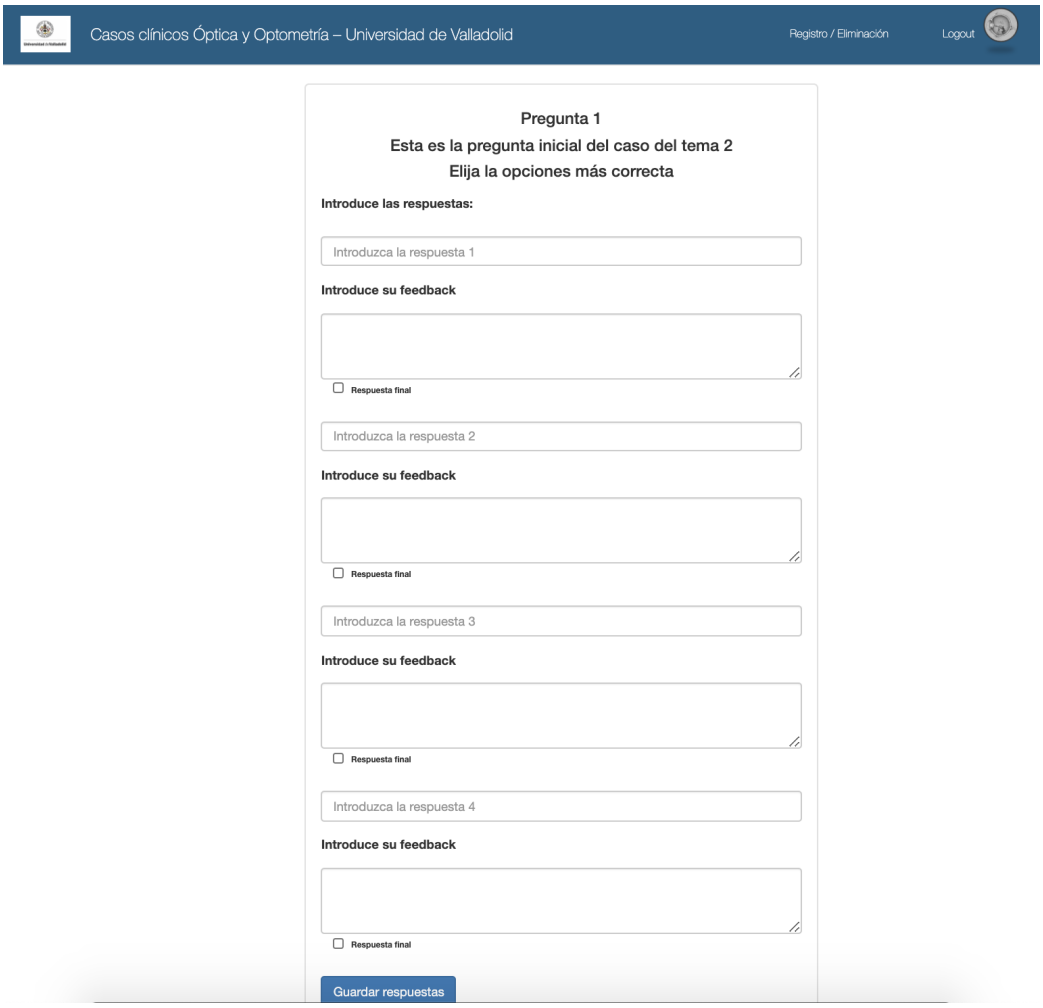


Figure 66: Añadir respuestas

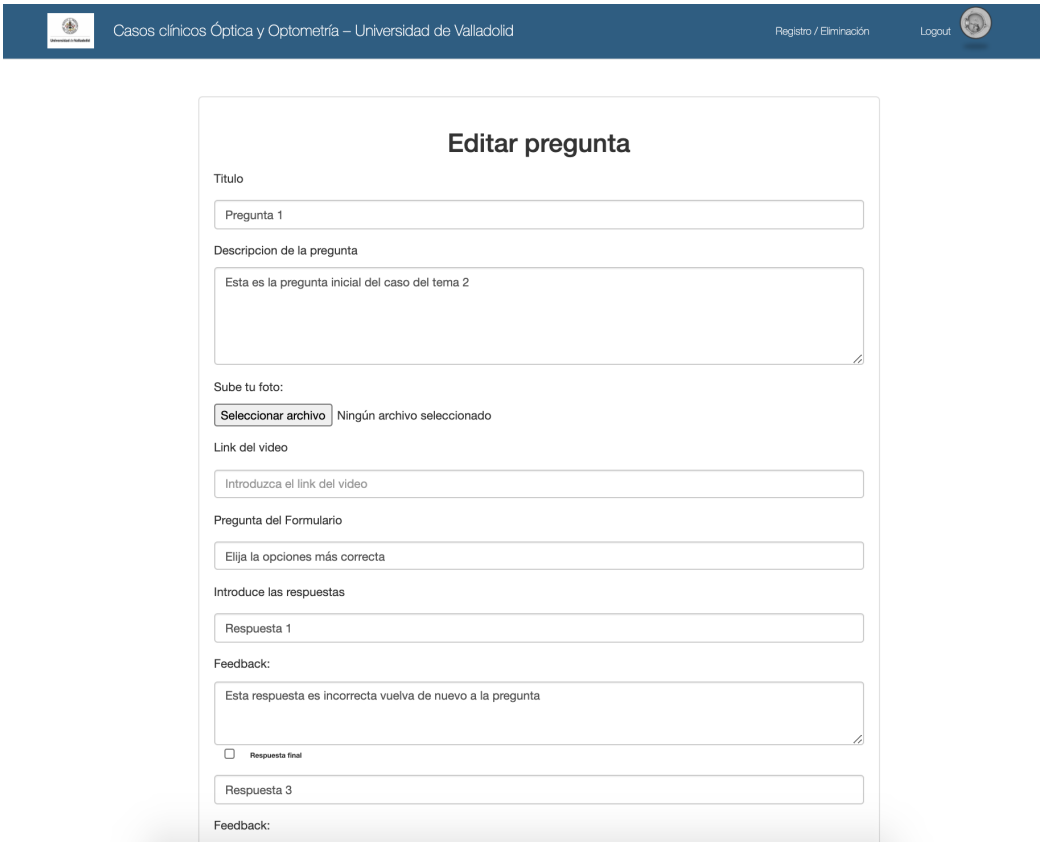


Figure 67: Editar una pregunta

11 ANEXO II: Contenido del CD-ROM

El contenido que se encuentra en el CD-ROM es:

- Es el PDF con la memoria del proyecto.
- Y una carpeta denominada proyecto-tfg: En esta carpeta se va a encontrar todo lo relacionado con el código fuente del desarrollo del proyecto, en su interior habrá dos carpetas una con el código correspondiente al frontend y otra con el backend.

12 ANEXO III: Manual de instalación para desarrolladores

En esta sección se detallará de forma breve el software requerido, dependencias necesarias para el proyecto y comandos útiles.

Software requerido

El software requerido es:

- Npm (versión 6.14+)
- Node JS (versión 13.7+)
- MongoDB (versión 4.2+)

Instalación de dependencias

Para la instalación de dependencias primero copiamos el código del CD, y a continuación usamos el comando:

```
npm install
```

Después de usar este comando tendríamos todas la dependencias necesarias para el funcionamiento del proyecto.

Comandos útiles

- npm start: Gracias a esto comando podemos iniciar el API REST del proyecto.
- ng serve: Con este comando levantamos la parte de frontend del proyecto, de forma normal en el puerto 4200 (http://localhost:4200/).

Inicialización de la base de datos

Esto se requiere realizar en una sola ocasión para crear el primer usuario administrador del sistema. Para ello debemos usar los siguientes comandos según el orden el que están escritos.

- Entrar el interprete de mongo con el comando:

```
mongo
```

- Elegir la database del proyecto:

```
use CasosUva
```

- Por último, insertamos al usuario en el sistema con:

```
db.users.save({"email" : "correo","password" : "Contraseña"})
```

13 ANEXO IV: Manual de despliegue

A continuación, se describen todos y cada uno de los pasos a seguir para poner la aplicación web en el servidor. Antes de todo esto se debe preparar el proyecto en local, realizando las modificaciones pertinentes así como generar un artefacto en la parte de angular (front-end) mediante el comando:

```
npm run build –prod
```

Una vez hecho esto, nos conectaremos con el servidor mediante ssh.

```
ssh usuario@ip
```

A continuación, ejecutaremos una serie de comandos para descargar el gestor de paquetes, instalar git para así poder clonar el repositorio del proyecto y de esta forma trabajar de forma más rápida y cómoda. Los comandos usados son los siguientes:

```
apt-get update
apt-get install aptitude
aptitude update
aptitude install git
```

Después, para poder gestionar de mejor forma la versión de node que estamos usando en el servidor instalaremos nvm así como sus dependencias necesarias para su correcto funcionamiento. Para esto ejecutamos los comandos:

```
aptitude install build-essential libssl-dev
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh
| bash
source ~/.profile
```

Por último, en esta fase de configuración del servidor instaláremos nginx(un servidor web), node mediante nvm y mongodb, nuestra base de datos.

```
aptitude install nginx
nvm install version
aptitude Install mongodb
```

Una vez terminada la fase de configuración del servidor creamos una carpeta para el proyecto en el directorio home. Para esto, primero nos desplazamos al directorio home con:

```
cd /home
```

Y después creamos la carpeta con:

```
mkdir appweb
```

Clonamos el repositorio del proyecto en producción.

```
git clone url-repositorio
```

Dentro de la carpeta appweb, crearemos la carpeta uploads, en esta carpeta es donde se almacenaran todas la imágenes del proyecto.

```
mkdir -p uploads
```

A continuación, instalamos las dependencias del proyecto mediante:

```
npm update
```

Una vez hallamos hecho esto podemos arrancar la aplicación, aunque todavía quedan unos pasos para completar el proceso de despliegue. Lo ejecutamos con:

```
npm start
```

Para completar el proceso nos queda en primer lugar, instalar un software que nos permite dejar la aplicación web ejecutando en segundo plano, esto lo hacemos mediante:

```
npm install pm2 -g
```

Para ejecutarlo en segundo plan lo ejecutamos con:

```
pm2 start index.js
```

Podemos observar los procesos que están corriendo en segundo plan con:

```
pm2 list
```

Si queremos cortar la ejecución del mismo usaremos:

```
pm2 kill
```

Al realizar esto, solo nos quedaría un último paso. Ahora mismo la aplicación funcionario en el puerto 3977, sin embargo es recomendable redirigirlo en el puerto 80 siendo este ya el último paso a realizar. Esto se lleva a cabo con el siguiente comando:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp –dport 80 -j REDIRECT
–to-port 3977
```

Una vez completado este proceso ya habríamos finalizado el despliegue, lo único que nos quedaría seria inicializar la base de datos tal y como esta descrito en la sección 12.

14 Bibliografia

References

[1] Último acceso el 20/03/2020. ¿Qué es SCRUM?. Disponible en: https://www.scrum.org/resources/blog/que-es-scrum

[2] Último acceso el 20/03/2020. Eventos en Scrum. Disponible en: https://www.saraclip.com/eventos-en-scrum/

[3] Último acceso el 22/03/2020. Artefactos Scrum. Disponible en: https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html

[4] Último acceso el 22/03/2020. Proceso y roles Scrum. Disponible en: https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum/proceso-roles-de-scrum.html

[5] Último acceso el 21/03/2020. Api rest: que es y cuales son sus ventajas en el desarrollo de proyectos. Disponible en: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[6] . Último acceso el 23/03/2020. MacOsCatalina. Disponible en: https://es.wikipedia.org/wiki/MacOS_Catalina

[7] Último acceso el 23/03/2020. Visual Studio Code. Disponible en : https://code.visualstudio.com

[8] Último acceso el 23/03/2020. MEAN. Disponible en: https://es.wikipedia.org/wiki/MEAN

[9] Último acceso el 25/03/2020. Que es mongodb. Disponible en: https://openwebinars.net/blog/que-es-mongodb/

[10] Último acceso el 25/03/2020. EspressJS un framework para nodeJS. Disponible en: https://www.solucionex.com/blog/expressjs-un-framework-para-nodejs

[11] Último acceso el 25/03/2020. Angular. Disponible en: https://es.wikipedia.org/wiki/Angular_(framework)

[12] Último acceso el 25/03/2020. Guia practica del databinding en angular. Disponible en: https://www.acontracorrientech.com/guia-practica-del-databinding-en-angular/

[13] Último acceso el 28/03/2020. Angular custom Directive y sus tipos. Disponible en: https://www.arquitecturajava.com/angular-custom-directive-y-sus-tipos/

[14] Último acceso el 28/03/2020. Tubos incorporados. Disponible en: https://riptutorial.com/es/angular2/example/3757/tubos-incorporados

[15] Último acceso el 28/03/2020. Singlenton services. Disponible en: https://angular.io/guide/singleton-servicesusing-providedin

[16] Último acceso el 26/03/2020. NodeJS. Disponible en: https://desarrolloweb.com/home/nodejs

[17] Último acceso el 27/03/2020. Que es node package manager. Disponible en: https://openwebinars.net/blog/que-es-node-package-manager/

[18] Último acceso el 27/03/2020. Encriptación de password en nodejs y mongodb: bcrypt. Disponible en: https://solidgeargroup.com/password-nodejs-mongodb-bcrypt/?lang=es

[19] Último acceso el 27/03/2020. Access control CORS. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Access_control_CORS

[20] Último acceso el 27/03/2020. Implementar json web tokens nodejs. Disponible en: https://www.oscarblancarteblog.com/2018/01/16/implementar-json-web-tokens-nodejs/

[21] Último acceso el 27/03/2020. Introducción a mongoose para mongodb y nodejs. Disponible en: https://code.tutsplus.com/es/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs–cms-29527

[22] Último acceso el 26/03/2020. HTML5. Disponible en: https://es.wikipedia.org/wiki/HTML5.

[23] Último acceso el 26/03/2020. Gitflow mejora la gestion de tu repositorio git. Disponible en: http://bemobile.es/blog/2016/11/gitflow-mejora-la-gestion-de-tu-repositorio-git/

[24] Último acceso el 26/03/2020. Hoja de estilos en cascada. Disponible en: https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada.

[25] Último acceso el 26/03/2020. JavaScript. Disponible en: https://es.wikipedia.org/wiki/JavaScript

[26] Último acceso el 26/03/2020. TypeScript. Disponible en:https://es.wikipedia.org/wiki/TypeScript

[27] Último acceso el 26/03/2020. Que es postman. Disponible en: https://openwebinars.net/blog/que-es-postman/

[28] Último acceso el 27/03/2020. Postman. Disponible en: https://www.postman.com

[29] último acceso el 27/03/2020. Manual de Latex. Disponible en: https://es.wikibooks.org/wiki/Manual_de_LaTeX

[30] Último acceso el 27/03/2020. Visual Paradigm. Disponible en: https://en.wikipedia.org/wiki/Visual_Paradigm

[31] Último acceso el 28/03/2020. Gestión de riesgos: ¿Qué es? ¿Para qué y cómo emplearla?. Disponible en: https://gerens.pe/blog/gestion-riesgo-que-por-que-como/

[32] Último acceso el 29/03/2020. Framework o librerias. Disponible en: https://www.tithink.com/es/2018/08/29/framework-o-librerias-ventajas-y-desventajas/

[33] Último acceso el 27/03/2020. AngularCLI. Disponible en: https://desarrolloweb.com/articulos/angular-cli.html

[34] Último acceso el 28/03/2020. Salarios de programador web. Disponible en: https://www.indeed.es/salaries/programador-web-Salaries

[35] Último acceso el 28/03/2020. Sueldo de un desarrollador web. Disponible en: https://www.campustraining.es/noticias/sueldo-desarrollador-web/

[36] Último acceso el 10/04/2020. Angular 2 Charts Demo. Disponible en: https://valor-software.com/ng2-charts/

[37] Último acceso el 15/04/2020. Gamificación: el aprendizaje divertido. Disponible en: https://www.educativa.com/blog-articulos/gamificacion-el-aprendizaje-divertido/

15 Glosario

- MEAN stack: (acrónimo para: MongoDB, ExpressJS, AngularJS, NodeJS), es un framework que se usa para el desarrollo de aplicaciones web dinámicas, basado en el lenguaje de programación JavaScript.
- HTTP: protocolo de comunicación que permite las transferencias de información en la World Wide Web.
- XML: Es una metalenguaje que se usa para almacenar datos de forma legible.
- JSON: Es un formato de texto sencillo para el intercambio de datos.
- API: es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- REST: es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.
- BSON: representación binaria de JSON.
- Data-binding: Es la comunicación entre el archivo de lógica y la vista para poder mostrar o modificar información de la vista.
- Dependencias en angular: son servicios u objetos que alguna de nuestras clases necesita para su funcionamiento, ya sean componentes, directivas o servicios.
- NPM: Es un sistema de gestión de paquetes para NodeJS.
- DOM (modelo de objetos del documento): Esa una API para documentos de HTML y XML en la cual se define la estructura lógica del documento, como se accede y trata el mismo.
- Middleware: Es un software que permite a una aplicación comunicarse con otra aplicación, red, o hardware...
- Sprint en Scrum: Es una periodo de tiempo donde se lleva a cabo el desarollo de una parte de un producto utilizable. El desarrollo de un proyecto se divide en muchos sprints.

- URI (Uniform Resource Identifier): Es un identificador que nos permite poder localizar, acceder o compartir un recurso de forma única.
- Token: Es una cadena de caracteres, que se usa entre otras cosas para la autenticación de usuarios de forma más segura, para evitar tener que pasar cada vez los credenciales.