UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una web app de carácter universitario orientada a la resolución de retos de programación

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Andrés Ballester Medina

Tutor: Antonio Garrido Tejero

2017-2018

Resumen

En este trabajo se desarrolla una aplicación web adaptativa, más conocidas como *web apps*, para la utilización de una plataforma de retos de programación orientada al ámbito universitario. Esta aplicación se ha realizado utilizando principalmente el framework Angular, pero se han incluido diversas tecnologías para cumplir todos los requisitos que se plantean.

La aplicación hace uso de una API REST que publica los servicios web necesarios para dotar de funcionalidad a la plataforma, la cual incorpora un sistema de seguridad basado en tokens que se ha debido implementar en la aplicación.

El resultado final es una aplicación capaz de adaptarse a ordenadores y dispositivos móviles y que ofrece las funcionalidades de la plataforma de una manera fácil e intuitiva para el usuario.

Palabras clave: aplicación web, Angular, retos de programación, token, aplicación adaptativa.

Abstract

This project consists in the development of an adaptative web application to interact with a university-oriented platform for programming challenges. The application has been developed using Angular framework, however, to achieve all the requirements of this project, other technologies have been included.

This application uses a REST API which provides the web services that make the platform functional. This API integrates a token-based security system that has had to be implemented in the application.

As a result, an adaptative application able to work on desktop computers and mobile devices is developed, and it offers all features of the platform in a user-friendly way.

Keywords: web application, Angular, programming challenges, token, adaptative application.

Índice de contenidos

Índice de figuras

1 INTRODUCCIÓN

A lo largo de nuestra trayectoria en el grado de Ingeniería Informática, nos hemos dado cuenta de que la parte práctica de las asignaturas se centraba exclusivamente en implementar la parte teórica vista en las clases, con una metodología de resolución ya establecida. Este sistema implantado en la carrera es útil porque nos permite entender conceptos aislados y a la vez perfeccionar la programación de ellos. No obstante, hemos echado en falta poder afrontar problemas sin pautas o metodologías a seguir, que nos fuercen a utilizar lo aprendido en las asignaturas, consiguiendo así que los alumnos podamos ser creativos e independientes a la hora de afrontar un problema.

Esa sensación de ir guiados en la resolución de los problemas es lo que nos motiva a desarrollar una aplicación enfocada al ámbito universitario, especialmente a la programación informática, cuya finalidad sea proponer problemas a los estudiantes y que estos tengan la total libertad de poder afrontarlos o no. Esta libertad se extiende a la manera de abordar los problemas, donde cada alumno podrá resolverlos con la solución que considere más adecuada.

La aplicación consiste en una plataforma donde tanto alumnos como profesores puedan publicar problemas que consideren interesantes para que el resto de los estudiantes puedan resolverlos. En esta plataforma los usuarios podrán publicar soluciones haciendo uso del editor de código de la aplicación, a la vez que visualizar y comentar las respuestas de otros usuarios, haciendo una comunidad activa y dinámica donde se fomente el aprendizaje. A su vez, los profesores que publiquen problemas podrán resaltar una respuesta que consideren interesante, ya sea porque es la óptima, porque utilice diferentes conceptos aprendidos en las asignaturas, etc. Además, se pretende enfocar los problemas como retos y crear una sección de desafíos donde se publicarán varios retos durante un corto periodo de tiempo y se puntuará a los usuarios según sus respuestas. Todo esto con el objetivo de gamificar (CORTIZO PÉREZ, 2011) la programación creando una competición donde los usuarios podrán ver sus resultados y los de sus compañeros en un apartado de ranking.

Con esta aplicación se pretende motivar a los alumnos a que apliquen las diferentes técnicas de programación vistas a lo largo de la carrera de una manera más entretenida e interactiva, lo que les ayudará a aprender e interiorizar los conceptos.

1.1 MOTIVACIÓN

Este proyecto va a centrarse en el desarrollo de la web app que utilizarán los usuarios para hacer uso de esta plataforma. Las web apps son aplicaciones web que están optimizadas para diferentes dispositivos desde ordenadores de sobremesa a móviles, y este tipo de aplicaciones están en auge hoy en día, haciendo que profundizar en este tema pueda suponer una ventaja de cara al mundo laboral.

Además, adquirir estos conocimientos es motivador ya que va a permitirme aprender sobre tecnologías punteras a la vez que desarrollo una aplicación que va a poder ser utilizada por otras personas para mejorar sus capacidades como programadores.

1.2 OBJETIVOS

El objetivo principal es desarrollar una aplicación, que haga uso de los servicios de una API REST que ofrece los servicios necesarios para gestionar una aplicación de las características descritas en la introducción, de una forma útil, intuitiva y atractiva para el usuario. Para lograr este objetivo se proponen los siguientes subobjetivos:

- Crear un proyecto base donde desarrollar las diferentes páginas de la aplicación.
- Implementar los servicios que realicen las peticiones REST.
- Realizar la implementación de la seguridad tanto para la protección de la aplicación ante usuarios no autorizados, como la integración con el backend.
- Realizar la internacionalización de la aplicación para que pueda mostrarse en castellano, valenciano e inglés.
- Embeber un editor de código que permita al usuario programar desde la aplicación en varios lenguajes de programación.
- Realizar la aplicación de manera adaptativa, garantizando su funcionamiento tanto en ordenadores como dispositivos móviles.

1.3 IMPACTO ESPERADO

Con esta aplicación se pretende crear un espacio donde los alumnos puedan aprender colaborativamente a modo de juego, gracias al cual desarrollen sus habilidades como programadores y sus conocimientos sobre técnicas vistas durante el grado.

El principal grupo de usuarios afectados, y los que se verían más beneficiados de esta aplicación son los alumnos. Ellos van a poder desarrollar y afrontar problemas más complejos y cercanos al ámbito real, lo cual es muy útil para su futuro como programadores en el ámbito laboral, o incluso aprender a afrontar tipos de problemas que no se ven en las clases.

Respecto a los profesores, esta plataforma les puede ayudar a compartir contenidos extra que pueden ser de interés para los alumnos y que no entran en la materia de la asignatura por su complejidad, porque requieren de otros conocimientos o cualquier otro motivo.

Si logramos hacer que los alumnos apliquen los conocimientos y metodologías aprendidas de una manera interactiva, se puede conseguir incrementar la motivación, el interés por aprender y el rendimiento académico.

1.4 ESTRUCTURA

Este documento se ha estructurado en capítulos profundizando en los diferentes pasos que se han seguido para el desarrollo del proyecto que se plantea. A continuación se realiza una breve descripción de cada capítulo.

En este primer capítulo se ha realizado una introducción al tema del proyecto y se han explicado los objetivos que se pretenden cumplir. También se comenta qué se espera obtener de esta aplicación.

El segundo capítulo contiene el estado del arte. En él se evalúan las diferentes soluciones ya existentes con una finalidad similar a la que se pretende conseguir en este proyecto, destacando los elementos que se pueden aplicar y los que se deben evitar de cada una de ellas durante el desarrollo de este proyecto. Finalmente se proponen las características que debe tener la solución que se desarrolla en este proyecto.

El capítulo tres recoge las principales tecnologías utilizadas y se explica en detalle el propósito de cada una de manera que se tenga una visión global de las diferentes herramientas aplicadas y cómo se relacionan entre sí.

El cuarto capítulo detalla el proceso de análisis que se ha realizado previamente al desarrollo de la aplicación. En él se encuentran los requisitos iniciales que se han planteado para la aplicación.

El quinto capítulo contiene el diseño de la solución, donde se presentan los bocetos que se han seguido para la creación de la interfaz de usuario, se explica la arquitectura que se ha seguido para el desarrollo de la aplicación y se detallan los diferentes elementos que la componen.

El sexto capítulo expone el proceso de desarrollo que se ha seguido para obtener el resultado final de la aplicación. Se comienza describiendo la planificación de las diferentes tareas y se hace un recorrido por cada una de ellas describiendo cómo se han implementado y qué retos se han tenido que afrontar durante su desarrollo.

En el séptimo capítulo se detallan las pruebas que se han realizado para comprobar el correcto funcionamiento de la aplicación. Para este proceso se ha creado y ejecutado un plan de pruebas que ha permitido detectar y corregir los errores que han surgido en la versión inicial de la aplicación.

El octavo capítulo contiene las conclusiones obtenidas tras el desarrollo de la aplicación. Estas conclusiones se centran en comentar los diferentes problemas que se han encontrado a la hora de cumplir con los objetivos establecidos en la introducción. En él también se describen trabajos futuros que se pueden realizar en base a este proyecto, bien porque no se han podido realizar en el plazo establecido para la realización de este trabajo o porque se consideran líneas interesantes de desarrollo para mejorar las funcionalidades de la plataforma.

1.5 Colaboraciones

El desarrollo de la plataforma que se pretende crear es demasiado amplio para poder abordarlo de manera individual. Es por esto que se ha dividido su implementación en dos partes típicas en la mayoría de proyectos de grandes dimensiones.

Por una parte, se desarrollará el backend publicando una API REST que proporcionará los servicios necesarios para dotar de funcionalidad y de persistencia a la plataforma. Esta API podrá ser utilizada por cualquier aplicación que desee hacer uso de ella, no obstante, los servicios dispondrán de seguridad para evitar posibles malos usos. Esta parte del desarrollo se incluye dentro del TFG “Desarrollo de una API REST para una aplicación web de carácter universitario orientada a la resolución de retos de programación” realizado por la alumna Beatriz Benedicto Granell.

Por otro lado, este trabajo se va a centrar en la creación de la aplicación que hará uso de la API mencionada, y proveerá a los usuarios de una manera de interactuar con la plataforma de manera fácil e intuitiva.

2 ESTADO DEL ARTE

El aprendizaje online, también conocido como *e-learning*, es la educación que se imparte y se recibe a través de medios digitales. Según la propuesta de la Dirección General de Telecomunicaciones de Teleeducación, se puede definir este tipo de formación en red como «el desarrollo del proceso de formación a distancia (reglada o no reglada), basado en el uso de las tecnologías de la información y las telecomunicaciones, que posibilitan un aprendizaje interactivo, flexible y accesible, a cualquier receptor potencial» (CABERO, 2006).

El *e-learning* se ha extendido sobre todas las ramas académicas, pero es en el ámbito de la tecnología y la informática donde más se ha asentado este tipo de aprendizaje. Según Class Central, una plataforma de búsqueda y valoración de cursos, de los 50 cursos más populares de todos los tiempos 20 son del ámbito de las tecnologías, lo que supone un 40% (Class Central, 2016).

Visto el interés que tiene la gente en adquirir conocimientos de informática, no es de extrañar que existan una gran cantidad de plataformas de aprendizaje sobre programación que facilitan el acceso a este tipo de conocimientos. Estas plataformas normalmente se centran en la impartición de cursos, gracias a los cuales los alumnos adquieren los conocimientos del tema que se imparte y tienen a su disposición una serie de ejercicios para practicar los conocimientos aprendidos. Esta es la metodología de plataformas como Coursera, Udacity o Codecademy[1].

Otro sistema de aprendizaje popular es focalizarse en ejercicios prácticos guiados donde los alumnos aprenden implementando lo que se les está enseñando mientras avanzan en el curso. Esta opción suele ser más interesante entre los niños o para cursos de muy bajo nivel ya que consigue captar la atención del alumno al mostrar resultados inmediatos de lo que se está aprendiendo. Un ejemplo de este sistema es la plataforma code.org que dispone de un gran abanico de cursos para aprender a programar desarrollando simples videojuegos. Esta plataforma organiza los cursos según la edad de los usuarios, empezando desde la programación por bloques para los más pequeños hasta lenguajes como JavaScript o Python para los más mayores.

También cabe destacar la plataforma SoloLearn[2], que combina ambos sistemas creando cursos no tan completos como las primeras plataformas mencionadas pero que incluyen una mayor cantidad de prácticas para conseguir que los alumnos puedan notar su progreso y así motivarlos para seguir aprendiendo.

No obstante, este tipo de plataformas son de carácter universal e intentan acceder al mayor número posible de alumnos. Centrándose en el ámbito universitario, cada vez es más frecuente que las universidades ofrezcan grados y másteres online. Sin embargo, no hay plataformas populares orientadas a estos perfiles de alumnos más especializados que aborden el aprendizaje de manera práctica, sin poner en primer plano la realización de cursos.

[1] Páginas oficiales de las plataformas e-learning:
- Coursera (Acceso Julio 2018) https://www.coursera.org/
- Udacity (Acceso Julio 2018) https://eu.udacity.com/
- Codecademy (Acceso Julio 2018) https://www.codecademy.com/

[2] Página oficial de SoloLearn (Acceso Julio 2018) https://www.sololearn.com/

También existen otro tipo de plataformas que se alejan del aprendizaje como tal y están orientadas a publicar retos de programación abiertamente para que los usuarios los resuelvan. En ellas los usuarios contestan públicamente a los retos propuestos y otras personas pueden verlo y comentar los errores o mejoras que tienen las soluciones. Esto fomenta el compartir conocimiento entre la comunidad, además de que crea competencia entre usuarios por lo que se *gamifican* este tipo de ejercicios.

Las plataformas orientadas a retos suelen estar dirigidas a un público con una base de conocimiento previo ya que no ofrecen formación de por sí y es la propia comunidad la que se ayuda mutuamente para aprender. Sin embargo, algunas plataformas como Coderbyte o CodeChef proporcionan tutoriales que pueden utilizar los usuarios para adquirir conocimientos sobre algoritmos o técnicas de programación que más tarde puedan aplicar en los retos que resuelvan.

2.1 Situación actual de la tecnología

Un aspecto fundamental de este trabajo es el desarrollo de una aplicación web. Por esto entender el panorama actual de la tecnología es un tema interesante en el que profundizar.

Desde sus orígenes, Internet se ha convertido en el pilar central de difusión de conocimiento de manera global. Pero no solo es útil para este propósito, gracias a él también se pueden realizar tareas específicas. Aquí radica la diferencia entre páginas web, que son una herramienta estática para la difusión de información, y las aplicaciones web, las cuales permiten al usuario interactuar con la herramienta para la realización de esas tareas específicas.

Los orígenes del desarrollo de aplicaciones web se remontan a antes de que internet fuese accesible a nivel global gracias a Perl, pero fue con PHP con el que este tipo de aplicaciones comenzaron su auge. El siguiente gran paso fue dado por Netscape incluyendo la posibilidad de ejecutar scripts por el navegador, definiendo un lenguaje que más tarde evolucionaría y se convertiría en el actual JavaScript (BARZANALLANA, 2012).

Conforme iba creciendo el desarrollo de aplicaciones web gracias a JavaScript, se empezaron a crear las primeras librerías que facilitaban la creación de estas páginas web dinámicas. Una de las librerías que ha sido el pilar fundamental de gran parte de las aplicaciones web desarrolladas en la última década es jQuery.

En la última década, se ha dado un paso más con la aparición de frameworks completos que permiten crear aplicaciones web sin la necesidad de utilizar múltiples librerías y configurarlas para que trabajen conjuntamente. Hoy en día los dos frameworks más populares son Angular y React (StackOverflow, 2018).

Otra tendencia que también hay que destacar es la evolución de la manera en que se sirven las aplicaciones web. Desde los orígenes las aplicaciones web estaban fuertemente ligadas al backend, ya que la lógica necesaria para crear páginas dinámicamente requería de un servidor capaz de ejecutarla y crear el contenido que finalmente llegaría al cliente. Este paradigma comenzó a no ser tan estricto con la llegada de jQuery, una librería que era capaz de modificar elementos de la pantalla sin necesidad de recargarla.

Con el gran avance de la tecnología, los ordenadores (incluidos los móviles) tienen la suficiente potencia para generar todo el contenido de la página dinámicamente y esto ha promovido un paradigma completamente contrario al mencionado anteriormente,

las SPA (*Single Page Applications* o aplicaciones de una página). Este tipo de aplicaciones ya no dependen de un backend que genere las páginas dado que la propia aplicación es capaz de generar todo el contenido sin la necesidad de hacer ninguna llamada a la parte servidora. Esta separación ha permitido la especialización de los programadores ya que, en aplicaciones web desarrolladas de esta manera, hay una clara separación entre frontend y backend haciendo que puedan ser desarrollados independiente por equipos o incluso empresas distintas.

Pero en las SPA no son todo ventajas, su principal problema es el tamaño de las aplicaciones. Como su nombre indica, toda la aplicación es cargada sobre una única página por lo que en un primer acceso se debe cargar toda la aplicación. Esto hace que la navegación por la aplicación sea mucho más rápida pero el primer acceso requiere descargar una cantidad mucho más elevada que la de una aplicación web tradicional (MELNIK, 2018).

El problema mencionado de las SPA junto con otros que no se han comentado debido a que no se encuentran dentro del alcance de este documento, ha sido solucionado parcialmente por una nueva tendencia en el desarrollo web, las PWA (*Progressive Web App* o aplicaciones web progresivas). Este tipo de aplicaciones aprovechan las características de HTML5 y de los nuevos navegadores para crear aplicaciones que son capaces de comportarse de una manera muy similar a como lo harían las aplicaciones nativas de los dispositivos. Estas aplicaciones se instalan en el navegador en el primer acceso por lo que en accesos posteriores la aplicación ya se encuentra en el dispositivo del usuario y no requiere de descargas adicionales.

En este panorama tecnológico es donde nos encontramos actualmente. Las aplicaciones web progresivas y las de una sola página no son la norma debido a su corto periodo de tiempo en el mercado, pero es la tendencia que está siguiendo el desarrollo de aplicaciones web.

Las PWA contrarrestan la tendencia que seguía el desarrollo de aplicaciones móviles hasta ahora, la creación de aplicaciones nativas. Estas aplicaciones son las que podemos encontrar en las tiendas de aplicaciones de los diferentes sistemas operativos y ese es su principal punto débil. Al desarrollar aplicaciones nativas tienes que duplicar el código ajustándolo para las distintas plataformas y una vez finalizado se deben publicar en las tiendas de aplicaciones. Esto puede ser costoso tanto en tiempo como económicamente además de que la necesidad de acceder a la tienda de aplicaciones suele hacer que los usuarios sean reacios a instalarlas (GUSTAFSON, 2017).

Las PWA no necesitan de tiendas de aplicaciones. Desde la propia aplicación web se puede crear un acceso directo en el dispositivo que, a efectos del usuario, no difiere de una aplicación nativa. Esto es una ventaja a tener en cuenta porque reduce los costes de desarrollo y de publicación de las aplicaciones consiguiendo un resultado muy similar al que obtendríamos con una aplicación nativa.

Sin embargo, hay que recordar que esta tecnología aún es joven y tiene muchas carencias e incompatibilidades y aunque los principales navegadores estén preparados sigue sin tener una compatibilidad total, sobre todo con navegadores antiguos como Internet Explorer[3].

Las principales ventajas de las PWA son:

[3] Can I Use? Service Workers (Acceso Julio 2018): https://caniuse.com/#feat=serviceworkers

- **Funcionalidad offline:** Son accesibles sin conexión a internet a pesar de estar instaladas en el navegador.
- **Compatibilidad multiplataforma:** Funcionan en cualquier dispositivo con un navegador web.
- **Reduce los costes:** Al evitar realizar un desarrollo para cada plataforma se requiere una inversión menor.
- **No necesitan una instalación:** Para ser más exactos, no necesitan una instalación tradicional. Las PWA se instalan de manera transparente para el usuario al acceder a ellas desde la web y pulsar el botón de añadir a la pantalla de inicio, lo que evita la necesidad de publicarlas en una tienda de aplicaciones.

Por el contrario, las principales desventajas son:

- **Menor rendimiento:** Al tener que ser utilizadas a través del navegador el rendimiento es menor que las aplicaciones que están programadas en el lenguaje nativo del sistema operativo.
- **Acceso a recursos del dispositivo:** Estas aplicaciones no tienen acceso a todos los recursos del dispositivo pese a que se está avanzando mucho para crear diferentes API de uso de recursos[4]. Por tanto, si la aplicación requiere de la utilización de características como el pago por NFC no se podría desarrollar como PWA actualmente.
- **Interacción con otras aplicaciones:** A menudo las aplicaciones interactúan entre sí pudiendo acceder a características de otras aplicaciones si están instaladas en el dispositivo. Este comportamiento no es posible con las PWA ya que se ejecutan en un entorno cerrado del navegador por motivos de seguridad.

2.2 Crítica al estado del arte

Volviendo al tema de las aplicaciones de *e-learning*, el aprendizaje práctico está relacionado a perfiles con menor conocimiento de programación, teniendo que acudir a plataformas de aprendizaje más teóricas para conseguir conocimientos avanzados.

También cabe destacar que las plataformas que se han mencionado al principio del capítulo están muy polarizadas en cuanto a la independencia del alumno. Por una parte, nos encontramos con plataformas que guían a los alumnos totalmente en su aprendizaje, ya sea porque por su inexperiencia o edad necesiten más ayuda o porque son cursos teóricos que únicamente tienen ejercicios para practicar el temario impartido. Por otra parte, están las plataformas de retos que no ofrecen ninguna guía o soporte a los usuarios siendo éstos los que deciden cuáles hacer y de qué manera, obteniendo como única retroalimentación los comentarios del resto de la comunidad la cual puede ser inexperta.

Analizando lo anteriormente mencionado se llega a la conclusión de que cuanta más libertad tiene el usuario menos posibilidades de ser corregido tiene. Esta conclusión es bastante lógica teniendo en cuenta que todas las plataformas mencionadas tienen un enfoque masivo y, por lo tanto, la revisión de los ejercicios realizados por los usuarios se tiene que hacer de forma automática. En estas plataformas los ejercicios evaluables son aquellos con un alcance limitado, siendo muy difícil analizar automáticamente los que no siguen unas directrices marcadas.

[4] What web can do today (Acceso Julio 2018): https://whatwebcando.today/

En el aspecto tecnológico se puede observar una tendencia hacia las PWA dado que ofrecen una experiencia similar a las aplicaciones nativas disminuyendo el coste de desarrollo a las empresas. Sin embargo, esta tecnología que pone como punto fuerte su capacidad multiplataforma todavía no tiene una compatibilidad total con todos los navegadores y características, con lo que al desarrollar una aplicación de este tipo no se puede tener una certeza absoluta de que todas las características funcionarán en todos los dispositivos.

2.3 PROPUESTA

Como se ha especificado en la introducción, en este trabajo **se pretende desarrollar una plataforma de ámbito universitario donde se propongan retos de programación** a los usuarios, quienes tendrán total libertad para resolverlos de la manera que crean oportuna. En esta plataforma los profesores tienen un papel importante ya que gracias a ellos se obtendrá una figura con conocimiento experto que permitirá a los usuarios obtener correcciones más fiables de las que aprender.

De esta manera se pretende crear una plataforma de retos, como las mencionadas al comienzo de este capítulo, que den una mayor libertad a los usuarios para proponer sus soluciones sin seguir un patrón predefinido. Aunque, al tener un alcance más focalizado como lo es una facultad de informática y disponer de usuarios expertos que corrijan y validen las soluciones de los usuarios, esta plataforma ofrece un lugar donde se pueda adquirir conocimientos sin privar a los usuarios de ser creativos a la hora de implementar sus soluciones. También fomentan la competencia entre usuarios consiguiendo una gamificación que motive a los alumnos a aprender a través de esta plataforma.

En esta plataforma tanto alumnos como profesores podrán publicar retos de programación sobre cualquier tema que les parezca interesante. Una vez publicados, los alumnos podrán verlo e implementar la solución como consideren más adecuado. Tras haber desarrollado su solución, podrán publicar una respuesta al reto con una breve descripción y el código implementado de tal modo que cualquier usuario, tanto alumno como profesor, pueda verla y compartir sus opiniones.

Un aspecto a tener en cuenta en el desarrollo de esta plataforma es la necesidad de que se pueda utilizar en diferentes dispositivos. Hoy en día la mayoría de los alumnos de la universidad utilizan ordenadores portátiles, pero no es raro verlos también utilizando dispositivos como tabletas o móviles. Por esto se propone hacer uso de las aplicaciones web progresivas que permitirán hacer un desarrollo único y multiplataforma, además de tener un aspecto que imita a las aplicaciones nativas en los dispositivos móviles.

3 TECNOLOGÍA UTILIZADA

Para el desarrollo de esta aplicación se va a hacer uso de un conjunto de tecnologías, herramientas y frameworks que van a facilitar la realización del proyecto agilizando la ejecución de ciertas tareas y ayudando a que la implementación implique un menor coste para el programador.

A continuación, se explican en detalle las tecnologías más importantes utilizadas en este proyecto.

3.1 ANGULAR FRAMEWORK

Angular es un framework open source para la creación de aplicaciones web que está entre los más utilizados por los desarrolladores de aplicaciones web desde su aparición en 2010. Su mantenimiento actualmente es llevado por varios equipos, muchos de los cuales son parte de la compañía que lo creó, Google.

La elección de Angular para este proyecto se debe a que es un framework orientado a la creación de aplicaciones web con una curva de aprendizaje bastante corta comparado con otros. Esto permite centrarse en el desarrollo de la aplicación más que en el aprendizaje de la herramienta.

La primera versión de Angular, conocida ahora como AngularJS, combinaba ideas de otros frameworks como los bindings automáticos o el concepto de *Single page application*, pero la idea que lo hizo despuntar frente a los demás fue la utilización de componentes. Estos componentes están centrados en la organización y la reutilización de código de manera que solamente haya que desarrollar el código una única vez. Además, se estructuran de manera arbórea donde los componentes más básicos son utilizados por componentes más complejos de manera que cualquier cambio se propaga por toda esa jerarquía.

En 2016 se lanzó Angular 2, pero esta no fue una actualización del anterior framework. El equipo de Google encargado de esta herramienta utilizó todo lo aprendido durante el desarrollo de AngularJS para crear un framework nuevo que pudiera hacerles frente a sus competidores, y así surgió Angular 2. A partir de ese momento se ha ido actualizando hasta llegar a su versión 6[5] que es la que se va a utilizar para la realización de este proyecto.

A partir de su versión 2, Angular se convirtió en un framework completo a diferencia de las populares librerías para JavaScript que existían y siguen existiendo. Esto es una ventaja ya que evita que se deba incluir en el proyecto una gran cantidad de librerías para realizar funciones como el enrutamiento, las llamadas AJAX, los test unitarios, etc. En su lugar, Angular ofrece todas estas opciones de una manera homogénea para ahorrar al programador la configuración y conexión de todas esas librerías.

Otra característica que destacar de este framework es su filosofía de añadir código JavaScript a ficheros HTML, a diferencia de otros frameworks y librerías que hacen justamente lo contrario. Esta filosofía no es nueva, de hecho es la que seguían las primeras librerías que se desarrollaron para JavaScript. Pero trabajar de esta forma hace que el resultado final sean ficheros HTML y CSS separados de los scripts, lo que

[5] Lanzamiento de la versión 6 de Angular (Acceso Julio 2018): https://blog.angular.io/version-6-of-angular-now-available-cc56b0efa7a4

hace que varias personas puedan trabajar en paralelo y que el código sea más mantenible.

3.2 ANGULAR CLI

Angular CLI[6] (*command line interface*) es una herramienta proporcionada por Angular para facilitar la creación y el desarrollo de proyectos en este framework. Al crear un nuevo proyecto crea la estructura necesaria para el funcionamiento de la aplicación y provee de multitud de herramientas para acelerar la creación de código automatizando tareas como la creación de módulos, servicios o componentes para que el programador no tenga que escribir todo el código repetitivo que se requiere para definir y configurar nuevos elementos. Además, proporciona herramientas para facilitarnos tareas como la depuración o el testeo.

3.3 NODE.JS

Node.js[7] es un entorno de programación que se caracteriza por basarse en el lenguaje JavaScript, un lenguaje de scripts que originalmente estaba orientado a ser ejecutado por el navegador en cliente. Con Node.js se pueden ejecutar scripts en el servidor, lo que hace que las aplicaciones creadas sean altamente escalables.

3.4 NPM

NPM[8] (*Node Package Manager*) es un gestor de dependencias para proyectos basados en JavaScript. Es el gestor por defecto del entorno Node.js y también es el mayor registro de software que existe. Entre los paquetes que incluye, se encuentra el framework completo de Angular y todos los módulos que se van a utilizar para la realización de este proyecto.

3.5 MÓDULOS EXTERNOS

Anteriormente se ha explicado que Angular se basa en la utilización de componentes, pero no necesariamente tienen que ser implementados en el propio proyecto. Hay muchos desarrolladores tanto de Google como terceros que se dedican a crear componentes, que se empaquetan en módulos, y se pueden importar y utilizar en los proyectos creados con este framework.

Estos módulos se gestionan como dependencias gracias al gestor de paquetes NPM y un archivo de definición de dependencias del proyecto (*packaje.json*). Para entender como estas tecnologías interactúan entre sí se ha creado un esquema que representa las relaciones:

[6] Página oficial de Angular CLI (Acceso Julio 2018): https://cli.angular.io/
[7] Página oficial de node.js (Acceso Julio 2018): https://nodejs.org/es/about/
[8] Página oficial de NPM (Acceso Julio 2018): https://www.npmjs.com/

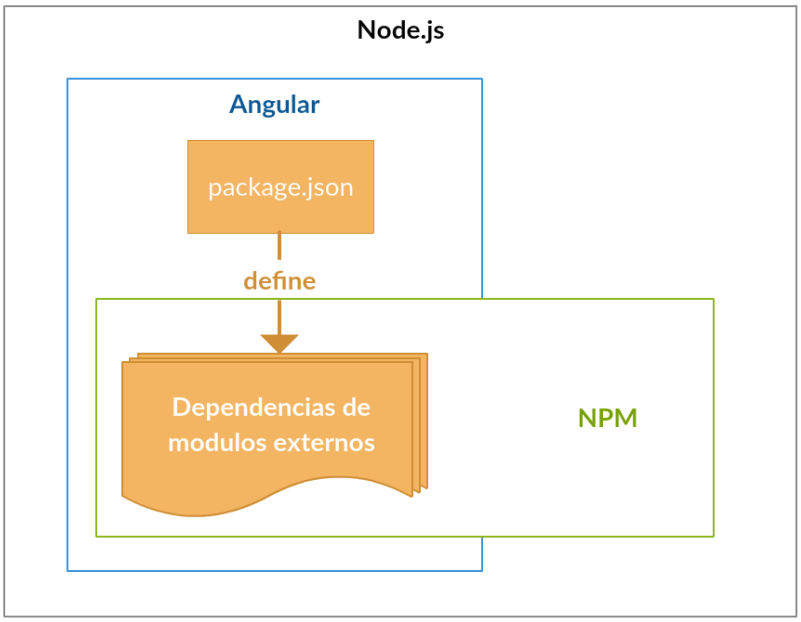

Figura 1: Esquema de relaciones entre tecnologías del proyecto

Como se observa en la Figura 1, las funcionalidades que ofrecen tanto Angular como NPM se pueden ejecutar gracias al entorno node.js. Además, las dependencias definidas en Angular son leídas por NPM que se encarga de traerlas de su repositorio y ponerlas a disposición del proyecto.

Para la creación de esta aplicación se va a hacer uso de varios de estos módulos. A continuación se mencionan los más relevantes.

3.5.1 Angular Material

Angular material[9] es una colección de componentes creados por equipos de Google que siguen el estilo Material Design, una guía de estilos orientada a aplicaciones tanto móviles como web creado por esta misma empresa.

Esta colección destaca por la gran cantidad de componentes que ofrece para la creación de aplicaciones, además de que incluye la colección de iconos y las tipografías de Material Design. Otro punto a favor es que hace uso de los *Angular schematics,* una herramienta introducida en la versión 6 del framework que permite crear esquemas que al aplicarlos al proyecto pueden generar nuevo contenido utilizando varios componentes para crear una estructura más compleja.

3.5.2 Flex-layout

Este módulo, también mantenido por un equipo de Google, provee a la aplicación Angular donde se importe una manera fácil e intuitiva de aplicar la disposición Flexbox[10] a los componentes. Flexbox es un sistema de disposición de elementos en una página web que soluciona muchos problemas que había anteriormente para alinear y posicionar elementos de manera que se adapten a diferentes tamaños de pantalla (Angular Blogs, 2018).

[9] Página oficial de Angular Material (Acceso Julio 2018): https://material.angular.io/
[10] Mozilla Developers Network web docs. CSS flexible Box Layout (Acceso Julio 2018): https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout

Gracias a este módulo y al sistema Flexbox, se va a poder crear una aplicación que se adapte a todos los dispositivos. Esto es importante debido a que se pretende que la aplicación sea utilizada tanto desde ordenadores convencionales como dispositivos móviles.

3.5.3 Ngx-translate

Para garantizar la internacionalización de la aplicación, Angular ofrece su propio sistema. No obstante, para este proyecto se ha decidido utilizar ngx-translate, una librería de internacionalización para Angular que permite definir contenido en diferentes idiomas de una manera más simple, así como gestionar la carga de ficheros con las traducciones. Estos ficheros se crean en formato json y definen tuplas código-traducción para que los códigos introducidos en el desarrollo de la aplicación sean sustituidos por las traducciones correspondientes.

3.5.4 Angular-in-memory-web-api

Este complemento está enfocado a ofrecer a los programadores una manera de falsear una API de backend para poder desarrollar la aplicación de la manera más completa posible.

Este módulo permite crear respuestas predefinidas a las llamadas HTTP que se realicen en la aplicación. Esto lo consigue interceptando todas las llamadas que se realicen a un destino externo a la aplicación y, si el usuario ha definido una respuesta, devuelve los datos que se le haya definido. Gracias a este servicio se puede hacer un desarrollo completo de la aplicación sin necesidad de tener en funcionamiento la API que provee de datos a la aplicación, y la manera en la que intercepta las peticiones hace que simplemente deshabilitándolo la aplicación pase a utilizar la API real sin necesidad de realizar ningún otro cambio.

3.6 Visual Studio Code

Para la realización de la aplicación se ha optado por utilizar el IDE de Microsoft Visual Studio Code[11]. La elección de esta herramienta se debe a que desde sus inicios, Angular ha estado muy ligado a este IDE por lo que se ha adoptado como el preferido por la comunidad a la hora de desarrollar aplicaciones de este framework. Las características más importantes de este programa es la adopción de la tecnología de autocompletado *IntelliSense* propias de las versiones completas de Visual Studio, la ligereza con respecto a otros competidores y el hecho de que, a diferencia de las demás versiones de Visual Studio, es gratuito.

3.7 Monaco editor

Monaco editor[12] es un editor de código desarrollado por Microsoft que puede ser embebido en una página web o aplicación para la edición de código. Este editor es utilizado por el IDE Visual Studio Code descrito anteriormente y provee de características como las sugerencias o el resaltado de sintaxis para más de 20 lenguajes, incluidos los que se pretende incluir en esta aplicación.

Para este trabajo se ha utilizado este editor ya que es fácilmente integrable en Angular por un componente de código libre, gracias al cual con una mínima configuración se

[11] Página oficial de Visual Studio Code (Acceso Julio 2018): https://code.visualstudio.com/
[12] Páfina oficial de Monaco Editor (Acceso Julio 2018): https://microsoft.github.io/monaco-editor/

puede embeber este editor en la aplicación permitiendo a los usuarios tener una experiencia de editor de código muy completa sin necesidad de tener que implementarlo.

3.8 TinyMCE

TinyMCE[13] consiste en un editor de texto embebible el cual provee a la aplicación de una manera de crear contenido complejo con un editor WYSIWYG que mejora la experiencia de usuario al poder crear contenido con una interfaz de usuario simple y no tener que preocuparse por utilizar códigos de marcado u otros sistemas.

3.9 Typescript

Typescript [14] es un lenguaje desarrollado por Microsoft que aplica conceptos de lenguajes orientados a objetos y basados en clases, lo que proporciona una forma estándar de código, comprobación de tipos y otras ventajas que se traducen en un código más limpio, legible y fácil de mantener. Además, este lenguaje se puede compilar en JavaScript por lo que la compatibilidad con los navegadores actuales es total. Como apunte, a partir de la versión 2 de Angular se introdujo Typescript como lenguaje para el desarrollo de la lógica de la aplicación (TURNER, 2015).

3.10 Git

Durante la implementación de la aplicación va a hacerse uso de un sistema de control de versiones, concretamente Git [15]. La utilización de este tipo de sistemas está ampliamente adoptada por las empresas de desarrollo software ya que ofrece ventajas a la hora de trabajar colaborativamente, almacenar versiones, crear copias de seguridad de los proyectos, etc. Para este proyecto sobre todo destaca la capacidad de crear copias de seguridad ya que al no desarrollarse por un equipo no se hace uso de las capacidades para trabajar colaborativamente sobre el mismo código.

El servicio de hosting utilizado para almacenar el repositorio que contendrá la aplicación es GitHub[16], un servicio gratuito que te permite almacenar proyectos de una manera sencilla y visual.

3.11 Jasmine

Jasmine[17] es el framework de testeo más popular en las aplicaciones creadas con Angular. Esta tecnología se utiliza para la creación de tests unitarios de una manera limpia y legible para que cualquier usuario pueda entender qué se está probando en cada momento.

[13] Página oficial de TinyMCE (Acceso Julio 2018): https://www.tiny.cloud/
[14] Página oficial de Typescript (Acceso Julio 2018): https://www.typescriptlang.org/
[15] Página oficial de Git (Acceso Julio 2018): https://git-scm.com/
[16] Página oficial de GitHub (Acceso Julio 2018): https://github.com/
[17] Página oficial de Jasmine (Acceso Julio 2018): https://jasmine.github.io/

4 ANÁLISIS

Tras crear una propuesta de aplicación y habiendo descrito las tecnologías más relevantes que se van a utilizar, se procede a realizar un análisis del problema. En este capítulo se van a recoger las necesidades de la aplicación, así como definir las acciones que los usuarios pueden realizar. Con esto se van a especificar las funcionalidades y el comportamiento del sistema analizando las necesidades del usuario.

Una de las etapas más importantes a la hora de desarrollar un proyecto es el proceso de especificación de requisitos. En éste se reúnen las necesidades del usuario que la solución pretende cubrir para así tener una base sobre la que comenzar a construir la aplicación.

Para este proceso se han seguido las prácticas que define el estándar IEEE830 (IEEE, 1998) adaptándolas a este trabajo, ya que varios de los apartados que el estándar recomienda tienen correspondencia con secciones anteriores de este documento.

4.1 CARACTERÍSTICAS DEL USUARIO

Para estas especificaciones se ha tenido en cuenta que existen tres tipos de usuario: los alumnos, los profesores y usuarios no autentificados.

Los alumnos y profesores tienen características muy diferentes. Por un lado, los alumnos por norma general será gente joven, habituada al uso de dispositivos móviles, con interés de ampliar sus conocimientos y dados a competir entre sí. Por otro lado, los profesores tienen un perfil de más edad, en este caso habituados al uso de tecnologías debido al área académica a la que pertenecen, pero no tan dependientes del uso de dispositivos móviles, con motivación para enseñar y con conocimiento experto sobre los contenidos de la plataforma. Estas diferencias en las características se deben de tener en cuenta para obtener los requisitos de la aplicación de manera que se ajusten a los usuarios que la van a utilizar.

Finalmente, aquellos usuarios que no están autentificados en el sistema únicamente pueden realizar dos acciones que son registrarse o iniciar sesión.

4.2 REQUISITOS FUNCIONALES

Estos requisitos definen las acciones fundamentales que debe poder realizar la plataforma. Para las entradas y salidas de la aplicación se ha tenido en cuenta el diagrama de clases creado en el TFG complementario, en el cual se desarrolla el backend de la aplicación. En el diagrama se especifican los modelos de datos de los principales objetos que manejará la aplicación así como las relaciones entre ellos.

Para este proyecto se han definido los siguientes requisitos funcionales:

Registrar usuario

La plataforma debe ser capaz de registrar un nuevo usuario.

Entrada: login, password y usuario.

Salida:

- Si se crea correctamente el usuario accede a la aplicación.
- Si no se rellena algún campo se muestra un error de validación.

- Si ya existe el usuario se muestra un error informando de que el login no es único.

Iniciar sesión

Un usuario registrado debe poder acceder a la plataforma utilizando sus credenciales.

Entrada: login y password.

Salida:

- Si las credenciales son correctas accede a la aplicación.
- En caso contrario muestra un error de autenticación.

Cerrar sesión

El usuario debe ser capaz de cerrar la sesión activa.

Entrada: -

Salida: Se cierra la sesión del usuario en la aplicación.

Filtrar retos:

El usuario debe ser capaz de filtrar retos obteniendo un listado con aquellos retos que contengan dicha cadena.

Entrada: Cadena de texto con el filtro a aplicar.

Salida: Listado de retos de retos que contienen la cadena de texto en el título, la asignatura, las etiquetas o el usuario creador.

Ver reto:

El usuario debe poder acceder al detalle de un reto para visualizar su contenido.

Entrada: id del reto.

Salida: Información detallada del reto.

Crear reto:

El usuario debe poder crear y publicar un reto.

Entrada:

- Obligatorios: título, descripción y login de usuario.
- Opcionales: asignatura y etiquetas del reto.

Salida:

- Si los campos obligatorios están rellenos se publica el reto en la plataforma.

- En caso contrario se muestra un error con los campos requeridos.

Crear competición

El usuario con rol de profesor debe ser capaz de crear y publicar un reto de tipo competición.

Entrada:

- Obligatorios: título, descripción, login de usuario y fecha límite.
- Opcionales: asignatura y etiquetas del reto.

Salida:

- Si los campos obligatorios están rellenos se publica la competición en la plataforma.
- En caso contrario se muestra un error con los campos requeridos.

Editar reto

El usuario que haya creado un reto debe poder realizar modificaciones sobre él.

Entrada: titulo, descripción, asignatura, etiquetas y login de usuario.

Salida:

- Si el login de usuario no es el mismo que el del creador del reto muestra un error de permisos.
- Si los campos título o descripción se han eliminado se muestra un error con los campos requeridos.
- En caso contrario se modifica y se publica el reto actualizado en la plataforma.

Crear código fuente

El usuario debe ser capaz de crear un código fuente y guardarlo en su espacio personal.

Entrada: título, lenguaje, código fuente y login de usuario.

Salida: Se almacena el código fuente en el espacio personal del usuario.

Ver código fuente

El usuario debe poder visualizar un código fuente publicado por otro usuario.

Entrada: id del código fuente.

Salida: Información detallada del código fuente.

Ejecutar código

El usuario debe ser capaz de ejecutar un código fuente tanto propio como de otro usuario y obtener el resultado de su ejecución.

Entrada: código fuente y lenguaje de programación.

Salida: Resultado de la ejecución del código fuente.

Editar código fuente

El usuario debe ser capaz de editar un código fuente propio y persistir los cambios realizados.

Entrada: título, lenguaje, código fuente y login de usuario.

Salida:

- Si el login de usuario no es el mismo que el del usuario creador del código fuente muestra un error de permisos.
- En otro caso se actualiza el código fuente en el espacio personal del usuario.

Publicar comentario

El usuario debe ser capaz de redactar y publicar un comentario en la vista de un reto.

Entrada: id del reto, texto del comentario y login del usuario.

Salida: Se publica el comentario en el listado de comentarios del reto.

Responder comentario

El usuario debe poder responder a un comentario ya existente sobre un reto, quedando registrada la respuesta al comentario como un comentario más del reto.

Entrada: id del reto, id del comentario padre, texto del comentario y login de usuario.

Salida: Se publica el comentario en el listado de respuestas del comentario padre.

Resaltar comentario:

El usuario con rol de profesor debe ser capaz de resaltar un único comentario por cada uno de los retos que haya publicado. El comentario resaltado quedará registrado en el reto.

Entrada: id del reto, id del comentario y login de usuario.

Salida:

- Si el login de usuario no coincide con el usuario creador del reto muestra un error de permisos.
- En caso contrario se registra el comentario en la información del reto como comentario destacado.

Adjuntar código fuente a comentario

El usuario debe ser capaz de adjuntar uno o varios de sus códigos fuente al comentario que esté redactando.

Entrada: id del comentario, lista de id de códigos fuente y login de usuario.

Salida:

- Si el login de usuario no coincide con el usuario publicador del comentario muestra un error de permisos.
- En caso contrario muestra los códigos adjuntos en la descripción del comentario.

Votar positivamente

El usuario debe tener la posibilidad de votar positivamente un comentario de otro usuario quedando registrado en el comentario la cantidad de votos positivos obtenidos.

Entrada: id del comentario y login de usuario.

Salida: Se incrementa en uno el contador de votos positivos del comentario objetivo y se almacena el login para evitar votos duplicados.

Ver perfil de usuario

El usuario debe ser capaz de visualizar su perfil de usuario.

Entrada: login de usuario.

Salida: Información detallada del usuario.

Cambiar idioma de la aplicación

El usuario debe poder cambiar el idioma de la aplicación a los idiomas castellano, valenciano e inglés.

Entrada: código de idioma.

Salida: Los textos de la aplicación se muestran en el idioma seleccionado.

4.3 REQUISITOS DE DISEÑO

Los requisitos de diseño definen la visualización y el comportamiento que debe tener la aplicación.

- **Interfaz de usuario:** La aplicación debe presentar una interfaz de uso intuitiva y sencilla.
- **Sistema adaptativo:** La interfaz se debe adaptar a los diferentes dispositivos que hagan uso de la plataforma.

5 DISEÑO DE LA SOLUCIÓN

Tras analizar el problema y plantear una solución, el siguiente paso es diseñar cómo se va a llevar a cabo su implementación. En este capítulo se profundiza sobre los diferentes elementos que va a contener el proyecto y la estructura que van a seguir para garantizar una organización que ayude en su desarrollo.

5.1 INTERFAZ DE LA APLICACIÓN

Un aspecto importante en la realización de aplicaciones es la interfaz de usuario. Gracias a ella los usuarios son capaces de interactuar con la aplicación, por lo que tener especial cuidado al realizarla puede suponer un gran aliciente para que los usuarios la utilicen.

El diseño de una interfaz de usuario es un ámbito muy estudiado en el que las grandes empresas dedican una importante cantidad de recursos. Un ejemplo de esto es la multinacional Google que ha creado una guía de diseño para las aplicaciones de su plataforma consiguiendo un estilo unificado. Este estilo llamado *Material Design*[18] es el que se va a utilizar en el desarrollo de esta aplicación debido a que define múltiples aspectos de visualización evitando así la necesidad de realizar el trabajo de diseño de interfaz específicamente para esta aplicación.

Aunque *Material Design* contiene recomendaciones para un gran abanico de componentes que habitualmente contienen las aplicaciones, el diseño general de las páginas de la aplicación se debe de crear para cada una de ellas. Por este motivo se van a realizar una serie de bocetos donde se mostrará la disposición de los elementos y la relación entre ellos.

5.1.1 Inicio de sesión y registro en la aplicación

Figura 2: Página de inicio de sesión

Como se puede apreciar en la Figura 2, en la parte superior se pretende crear una barra de aplicación. Éste es un elemento muy común que contiene el título de la aplicación y algunos menús de acceso rápido. En esta pantalla únicamente incluye el título ya que el usuario todavía no ha iniciado sesión por lo que no debe poder realizar ninguna acción.

Para el inicio de sesión se ha diseñado una página simple donde el usuario tendrá que introducir el usuario y la contraseña para acceder a la aplicación. Se incluye un botón para realizar el acceso y un enlace a la página de registro para que todo aquel que no disponga de un usuario pueda registrarse en la plataforma.

[18] Material Design (Acceso Agosto 2018): https://material.io/

La pantalla de registro de usuario será muy similar a la que se observa en la Figura 2, añadiendo los campos de información extra necesarios como pueda ser el nombre a mostrar en la aplicación. El botón principal cambiará de texto a "Registrar" y el enlace inferior a "Inicia sesión aquí" para indicar al usuario las acciones que se realizan al pulsar en cualquiera de ellos.

En ambas pantallas, si ocurre algún error al comunicar con el servidor como por ejemplo que el usuario y contraseña no sean correctos o que el usuario que se pretende registrar ya existe, se mostrará una alerta informando del error y pidiendo al usuario que vuelva a introducir los datos.

Una vez el usuario pulse en el botón de "Acceder" o "Registrar" y se validen los datos introducidos, se le redirigirá a la página principal donde se muestran los últimos retos.

5.1.2 Página principal

Figura 3: Página principal

La página principal de la aplicación es la primera que aparece tras iniciar sesión. En ella se muestran los retos más recientes subidos a la plataforma en orden cronológico descendente. El mostrar los retos más recientes en la parte superior hace que cada vez que se inicie sesión se muestre contenido nuevo, haciendo la aplicación más dinámica.

Para facilitar a los usuarios la búsqueda de retos se incluye un campo de búsqueda donde pueden introducir un texto y así buscar los retos que contengan este término. Esta búsqueda se realizará por título, descripción, autor, etiquetas y asignatura, de esta manera se simplifica el proceso de búsqueda al usuario evitando tener múltiples campos para cada filtro.

Los retos se dispondrán en un scroll infinito, esto quiere decir que el usuario podrá ir haciendo scroll para ver retos más antiguos y conforme se vaya acercando al final se cargarán más retos. Este sistema da una sensación de fluidez sin perder prestaciones ya que el efecto para el usuario es que puede ver todo los retos, pero realmente se cargan únicamente aquellos que se van a visualizar.

Como se observa en la Figura 3, la barra superior de la aplicación contiene dos acciones: en la parte izquierda se encuentra el botón menú que despliega el menú lateral con los accesos a las diferentes páginas de la aplicación, en la parte derecha se incluye un botón para crear un nuevo reto.

5.1.3 Listado de competiciones

La plataforma se centra en la realización de retos y existen un tipo específico de retos llamados competiciones. Estos retos son especiales y por eso se pretende crear una página que los agrupe para diferenciarlos todavía más de los retos convencionales. Para ello se va a diseñar una página que será similar a la página principal mostrada en la Figura 3 pero mostrando únicamente las competiciones. Además, en cada una de ellas mostrará información adicional como el tiempo restante o la posición del usuario.

5.1.4 Menú de la aplicación

Figura 4: Menú de la aplicación

Al pulsar el botón de menú se despliega un panel lateral que se superpone a la página en la que se encuentre el usuario en ese momento. Si el panel de menú está desplegado, el botón superior izquierdo se transforma en una flecha que al pulsarla vuelve a ocultar el menú.

Como se puede apreciar en la Figura 4, en esta sección se incluyen enlaces directos a las distintas páginas que forman la aplicación que son la página principal, el listado de competiciones, los códigos fuente del usuario y el perfil personal.

5.1.5 Creación y edición de reto

Figura 5: Editor de retos

La pantalla de creación o edición de reto es necesaria para poder añadir nuevos retos a la aplicación.

Esta pantalla, como muestra la Figura 5, cuenta con un campo editable donde se introduce el título del reto que será el que aparecerá en el listado de retos. En la parte inferior se encuentra un editor de texto enriquecido que permitirá a los usuarios introducir la descripción del reto y ajustarlo para que tenga una apariencia más atractiva, evitando un gran bloque de texto plano que pueda intimidar a los usuarios.

El botón de acción superior derecho se transforma en un botón guardar que al pulsarlo guarda y publica el reto para que sea visible para el resto de los usuarios.

Esta pantalla sirve tanto para la creación de nuevos retos donde todos los campos estarán vacíos, como para la edición de retos donde los campos vendrán con la información del reto precargada.

5.1.6 Vista de reto

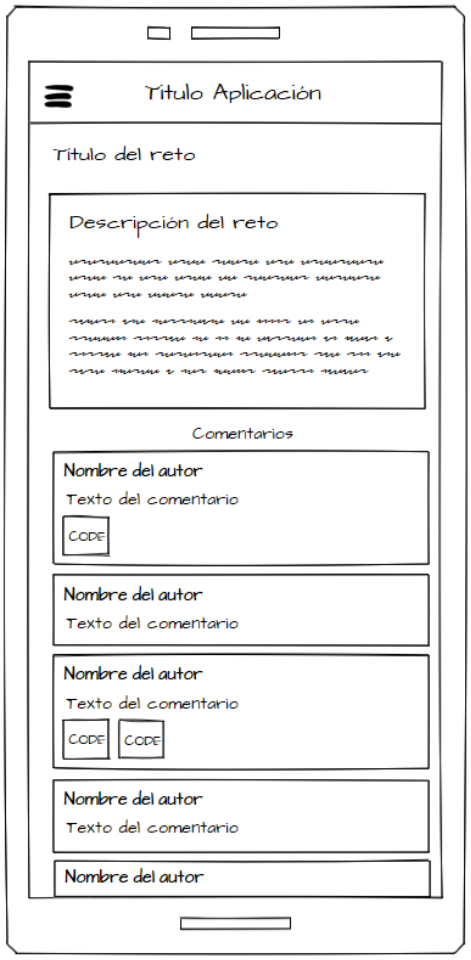

Figura 6: Vista de reto

Una vez un reto sea publicado, los usuarios podrán acceder a él para ver su contenido y proponer sus respuestas. Para ello se ha diseñado una pantalla que muestra la descripción del reto donde se explica el propósito y requisitos del reto que deben resolver los alumnos.

La descripción del reto se contendrá en un panel adaptable que se expandirá dependiendo de la longitud de la descripción. Como se ha comentado en la pantalla de creación de retos, esta descripción está escrita en formato de texto enriquecido por lo que el panel que muestre la información debe ser capaz de mostrar el texto con el formato correcto.

Debajo del reto se encuentran los comentarios que se cargarán con un scroll infinito como el que se explica en la página principal. Cada comentario contendrá el usuario que lo ha creado, el texto del comentario y accesos a los códigos que se hayan adjuntado en caso de haber alguno.

Para añadir un nuevo comentario se desplegará un campo de texto en la parte inferior junto con un selector de códigos que permitirá al usuario adjuntar los códigos que haya creado.

5.1.7 Vista y edición de código fuente

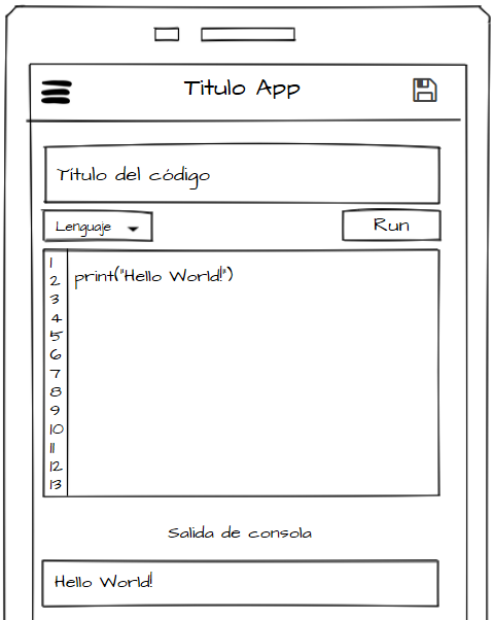

Figura 7: Vista y edición de código fuente

Una de las características principales de la aplicación es que los usuarios puedan programar desde ella. Para ello se ha diseñado la página de vista y edición de código fuente. Esta pantalla servirá tanto para crear códigos fuente propios como para ver y ejecutar códigos fuente de otros usuarios.

La pantalla contiene en la parte superior un campo donde insertar el título del código fuente, de esta manera los usuarios podrán localizarlos fácilmente en su lista de códigos.

Como la aplicación permitirá la creación de códigos en distintos lenguajes, se dispondrá un selector de código donde los usuarios podrán elegir entre los diferentes lenguajes soportados por la plataforma.

El bloque principal de esta pantalla es el editor de código fuente. Este editor permitirá al usuario programar las soluciones de los retos en los que desee participar. Además, según el lenguaje seleccionado se coloreará el código para ayudar al usuario a diferenciar mejor los diferentes elementos como variables, métodos, etc.

Para probar el código implementado se dispone de un botón "Run" que ejecutará el código y la respuesta obtenida será mostrada en un panel inferior que simula lo que se vería por consola si se ejecutase el código desde un ordenador.

5.1.8 Códigos fuente del usuario

Los usuarios podrán crear tantos códigos fuente como deseen, y estos serán almacenados en un espacio personal. Para permitir a los usuarios acceder a este espacio personal se creará una pantalla a modo de listado donde se presentan todos los códigos que ha creado un usuario mostrando el título de cada uno para que sean reconocibles.

También se incluirá un botón para crear un nuevo código fuente. Al pulsar el botón de creación de un nuevo código o seleccionar uno de los códigos mostrados en el listado, se navegará a la pantalla de edición de códigos fuente descrita en el apartado anterior.

5.1.9 Perfil de usuario

Figura 8: Perfil de usuario

Por último, los usuarios dispondrán de una página de información personal. Ésta se organizará por bloques que mostrarán diferente información relativa al usuario.

El primer bloque mostrará información sobre el progreso del usuario a modo de diagrama o experiencia, en la Figura 8 se representa este bloque con una cruz. A su lado se muestra un panel de preferencias donde el usuario podrá ajustar ciertos aspectos de la aplicación como el idioma.

Los siguientes bloques contendrán información sobre actividad reciente del usuario, en ellos se mostrarán los últimos retos y competiciones participados, códigos editados, etc.

5.2 Arquitectura del sistema

Para el diseño de la solución, en un primer nivel más global, se va a dividir en dos partes independientes que interaccionan entre sí. Estas partes son un backend que publica una API REST y un frontend que consume los servicios publicados por la API para crear una interfaz de usuario. Estos dos grandes bloques se han repartido en dos proyectos diferentes como se ha comentado anteriormente en el punto *1.5 Colaboraciones*. Este trabajo se va a centrar concretamente en la implementación del frontend.

La arquitectura de la aplicación web que se va a desarrollar en este trabajo se plantea en tres bloques principales.

- Un conjunto de servicios que sirvan de enlace con la API a la que se conectará la aplicación. Esto va a conseguir abstraer la obtención de la información publicada por el backend haciendo más sencillo el acceso a ellos. Además, esta sección va a ser transversal por lo que estos servicios van a ser accesibles desde cualquier parte de la aplicación.
- Una serie de componentes que implementen pequeñas funcionalidades de la aplicación. Estos componentes van a agruparse en un módulo, que actuará como librería, para poder acceder a ellos e incluirlos en las diferentes pantallas que se pretenden realizar.
- Un bloque que agrupa las pantallas. En el diseño de la aplicación se pretende crear cada pantalla como un módulo independiente que incorporará tanto los componentes que se requieran en esa pantalla, como conexiones con los servicios para obtener la información específica y necesaria que se vaya a mostrar.

En el esquema mostrado en la *Figura 9* se describe cómo están relacionados estos bloques entre sí:

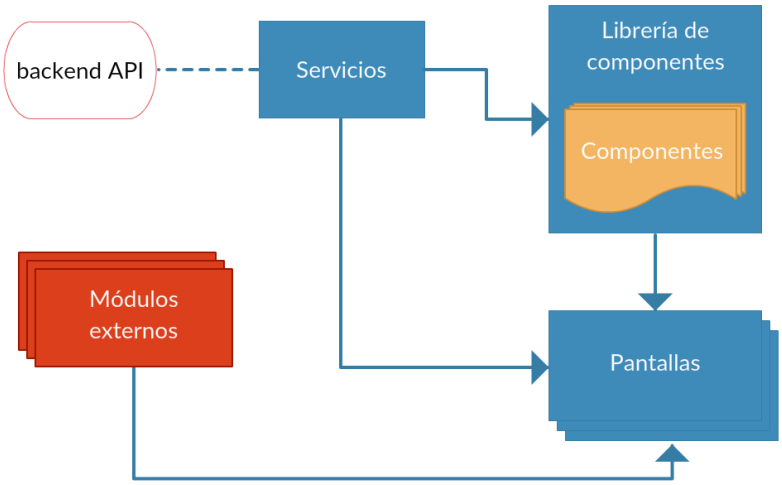

Figura 9: Esquema de la arquitectura de la aplicación

Como se aprecia en la *Figura 9* y se ha comentado anteriormente, los servicios sirven de puente, tanto para los componentes como para las pantallas, para acceder a la información publicada por el backend. Además, los componentes sirven para que las pantallas puedan reutilizar elementos sin repetir código. Finalmente, como se ha comentado en el apartado 3 se va a hacer uso de módulos externos que añadirán funcionalidades más complejas a la aplicación como puede ser el editor de código, la internacionalización o el estilo adaptativo.

5.3 Diseño detallado

En esta sección se va a profundizar en cómo se va a estructurar el proyecto para contener los diferentes elementos descritos en el punto anterior. La realización del diseño de la aplicación va fuertemente ligada al framework que se ha utilizado, ya que la manera que tiene de gestionar los diferentes elementos influye a la hora de estructurar el contenido. Además, hay que destacar que el diseño propuesto para la aplicación se basa en técnicas y recomendaciones ya existentes y conocidas para el desarrollo de aplicaciones con Angular, ajustándolas a las necesidades particulares de este proyecto.

En el nivel superior de la estructura del proyecto se van a crear dos directorios, uno destinado a contener todos los servicios que se van a usar en la aplicación, y otro que va a contener los módulos que formen la aplicación. Esta manera de organizar el código separa los elementos con una orientación más lógica (los servicios) de los elementos orientados a crear las vistas. Se ha optado por esta separación debido a que ayuda al mantenimiento y a la escalabilidad de la aplicación.

El directorio que contiene los servicios, por convención y debido a su alcance, se va a llamar *shared*. El directorio que incluye el grueso de la aplicación se va a llamar *app*, nombre que se le asigna automáticamente al crear el proyecto utilizando la herramienta Angular-CLI. Aquí se van a incluir tanto la librería de componentes comunes como las pantallas que se van a realizar.

Con el esquema de organización descrito se cubren los tres grandes bloques que se mencionaban en el apartado 5.1. A continuación, se describe el diseño más en detalle de estos bloques.

5.3.1 Servicios

El bloque de servicios, el cual se va a crear en un directorio llamado *shared,* va a agrupar los diferentes servicios que serán utilizados durante el desarrollo de la aplicación. Este tipo de clases es muy común en proyectos de backend, pero la introducción de un inyector de dependencias por parte de Angular permite obtener todas las ventajas de este modelo de arquitectura en el frontend.

Como estas clases pueden contener cualquier tipo de lógica, se han agrupado en diferentes subdirectorios según su funcionalidad principal de la siguiente manera:

- **Guards**: Incluye las guardas de la aplicación, esto son unos servicios que se encargan de incluir cierta seguridad en la aplicación permitiendo o denegando el acceso a determinadas páginas o recursos según el criterio implementado en estos servicios. Hay que tener en cuenta que el código de la aplicación se ejecuta en cliente por lo que no se puede delegar toda la seguridad de la aplicación a estos servicios, sino que se debe implementar también en la parte servidora (backend).
- **Interceptors**: Aquí se agrupan los interceptores. Estos servicios implementan una interfaz de Angular que les permite interceptar las peticiones HTTP que se realicen a rutas externas a la aplicación. Estos servicios son de utilidad para el manejo de excepciones, la implementación de seguridad en las peticiones, etc.
- **Utils**: Durante el desarrollo del proyecto habrá cierta lógica que se repita en varias localizaciones del proyecto y que, en aras de reducir la cantidad de código, se agrupan en este bloque de servicios.
- **API-services**: Este subdirectorio contiene todos los servicios encargados de realizar las peticiones a los servicios web publicados por el backend creado en el TFG complementario a este. Estas clases contendrán la información necesaria para realizar las conexiones como puede ser la URL o el tipo de método HTTP que se utiliza. De esta manera se consigue tener en un único lugar toda esta información para poder hacer frente a modificaciones de estos parámetros de una manera ágil.
- **Model**: Adicionalmente se añade este quinto bloque que no contiene servicios. Aquí se encuentran las clases que definen el modelo de los objetos que serán obtenidos al hacer las peticiones HTTP. Se incluye en este bloque porque su contenido está fuertemente ligado al de api-services y porque es una ruta accesible desde cualquier punto de la aplicación. Esto es importante porque la

mayor parte de la lógica de la aplicación consiste en la manipulación de estos objetos.

Para entender la relación entre todos estos elementos y cómo se hace uso de ellos se ha creado la siguiente figura:

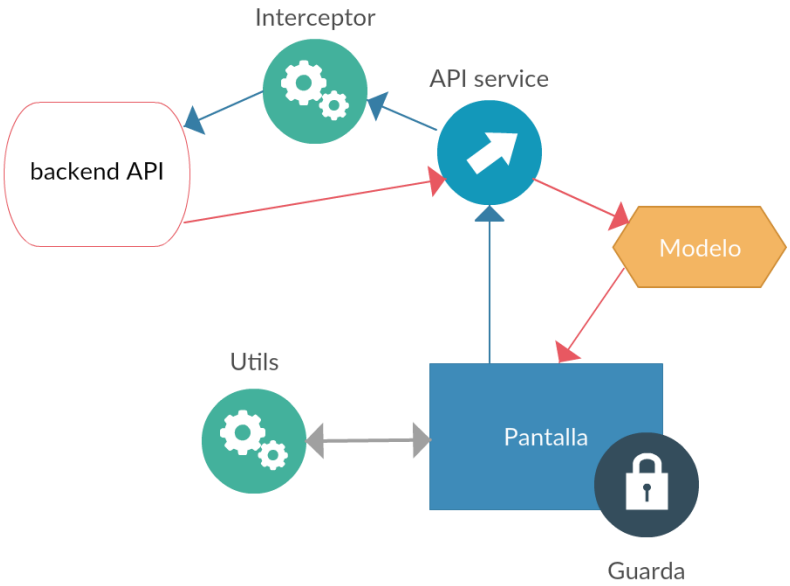

Figura 10: Esquema de funcionamiento de los servicios

Como se puede apreciar en la *Figura 10*, estos servicios proporcionan funcionalidades que son usadas por la lógica de la pantalla. Para facilitar su entendimiento se ha coloreado de azul el flujo de las peticiones al backend y en rojo el flujo de la respuesta que éste proporciona.

- La guarda se encarga de proteger el acceso a la página, bloqueando y redirigiendo las peticiones de usuarios que no han iniciado sesión en la plataforma.
- El servicio *Utils* provee a la pantalla de métodos comunes que se utilizan en la lógica de la pantalla.
- El API service se encarga de hacer las peticiones al backend.
- El interceptor captura estas peticiones y las modifica, añadiéndole las cabeceras que requiere la seguridad del backend.
- Al recibir la respuesta, se mapea a un objeto según el modelo que se le haya indicado en API service. Este objeto es el que le llega finalmente a la lógica de la pantalla.

5.3.2 Librería de componentes

Teniendo en mente el concepto de componentes que Angular define, se plantea crear un módulo a modo de librería que incluya los componentes más simples y básicos de la aplicación para que más tarde puedan ser utilizados desde las diferentes pantallas que se vayan a crear. Esta técnica de diseño es muy común verla en proyectos de este framework, debido a que permite reducir la cantidad de código duplicado y mejora la mantenibilidad al conseguir que las modificaciones realizadas en estos componentes sean propagadas por toda la aplicación. El módulo que contiene a todos estos componentes se llama comúnmente *SharedModule* ya que es un módulo transversal.

Pese a que la motivación principal de crear este módulo es el albergar los componentes compartidos, al estar importado por el resto de los módulos del proyecto, se puede aprovechar también para incluir otros elementos que sean accesibles para cualquier pantalla o componente que se desarrolle.

A nivel estructural, por cada elemento distinto que se cree dentro de este paquete se creará un subdirectorio. Este sistema aísla los diferentes componentes entre sí, algo lógico teniendo en cuenta que deben ser independientes. Así se consigue que sean fácilmente localizables y permite exportarlos individualmente para ser utilizados en otros proyectos en caso de que se quieran reutilizar.

5.3.3 Interfaz de usuario

La aplicación final va a consistir en una serie de pantallas, y así se va a reflejar en el diseño de la aplicación. Por cada pantalla se va a crear un módulo que contendrá un componente principal encargado de mostrar y gestionar el contenido, y una serie de componentes hijos encargados de las diferentes secciones de la página.

El crear cada pantalla en un módulo independiente no es un requisito de Angular. De hecho, se puede crear una aplicación completa en un único módulo. Sin embargo, separarlo de este modo trae consigo una serie de ventajas.

1. Teniendo el proyecto tan estructurado te permite identificar de un vistazo dónde está cada contenido de la aplicación.
2. Las dependencias se incluyen únicamente en el módulo de las pantallas que lo necesitan, evitando cargas de dependencias innecesarias.
3. Con este diseño se puede utilizar la carga de módulos bajo demanda, también llamada *lazy loading*. Esta técnica acelera la carga inicial al no tener que cargar el contenido de todas las pantallas en el primer acceso a la aplicación.

Además, se pretende que la aplicación tenga una barra superior y un menú accesible desde cualquier punto de la aplicación. Para conseguir esto se estructuran los diferentes módulos en dos niveles.

En el primer nivel se ubicarán las pantallas de acceso y de registro debido a que estas dos pantallas son accesibles sin haber iniciado sesión y desde ellas no se tiene acceso a los elementos comunes. En este nivel también se incluirá un módulo que será el encargado de mostrar la barra superior, el menú de la aplicación y un contenedor donde se dispondrá el contenido de las demás páginas de la aplicación. Esto evita replicar el código de estos elementos en todas las pantallas que los necesiten.

El segundo nivel contendrá el resto de las pantallas que componen la aplicación. Todas las pantallas de este segundo nivel serán cargadas dentro del contenedor que se ha mencionado en el párrafo anterior. Siguiendo este diseño se consigue un aspecto unificado de la aplicación además de cargas más fluidas, ya que parte del contenido mostrado va a mantenerse estático al navegar de una página a otra.

Con la organización de los módulos de las pantallas ya definida, se debe establecer qué contendrá cada uno de estos módulos. El contenido de cada uno va a ser muy variable pero principalmente encontraremos 3 elementos básicos. En la siguiente imagen se puede observar la estructura del módulo de una pantalla:

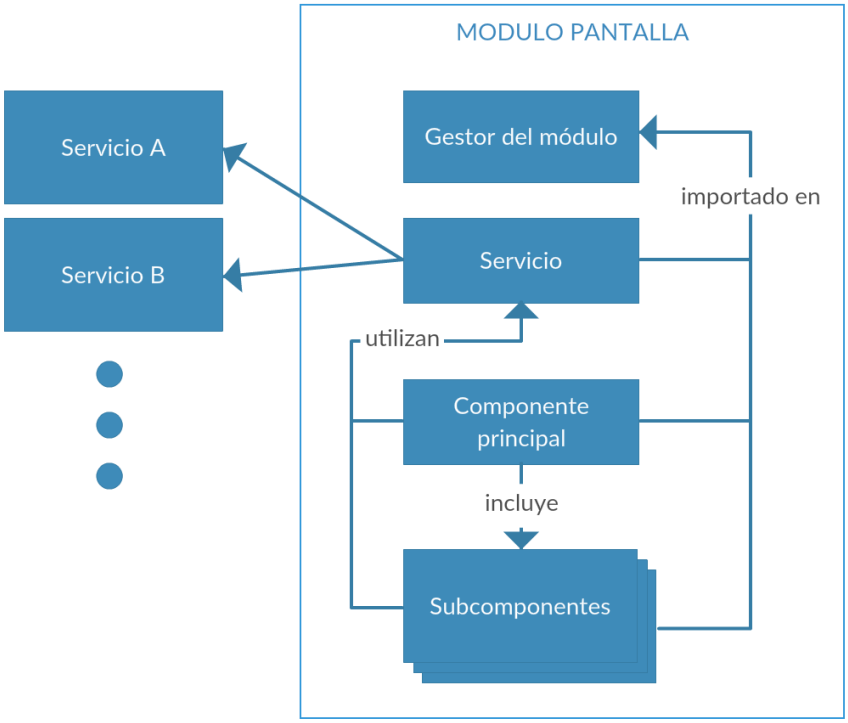

Figura 11: Estructura del módulo de una pantalla

A continuación se definen los elementos básicos que contiene un módulo como el indicado en la *Figura 11*:

- Un archivo que haga de empaquetador del módulo, incluyendo las dependencias con otros módulos y los diferentes elementos que se incluyan dentro de éste.
- Una clase, de tipo servicio, que sirva para agrupar todas las llamadas a otros servicios de la pantalla. Siguiendo este patrón, cada vez que se desee hacer uso de una funcionalidad desde la pantalla se llamará a esta clase que actuará de proxy y llamará al servicio que provea el método requerido. Así se consigue generar un código más limpio en la lógica de la pantalla.
- Un componente principal que incluya la lógica de la pantalla. En este componente se realizarán dos acciones principales, gestionar el contenido de la pantalla y reaccionar ante los diferentes eventos que puedan ocurrir durante el uso de la aplicación. Este componente también incluye un archivo HTML que contiene la representación visual de la pantalla. Estos archivos son potenciados gracias al uso de Angular por lo que pueden contener algo de lógica básica que facilita la generación de contenido.

Siguiendo la filosofía de componentes de Angular, las pantallas pueden incluir un conjunto de componentes hijos que gestionen secciones de la pantalla. La manera de estructurarlos es incluyéndolos en subdirectorios dentro del módulo de la pantalla. Estos componentes deben ser incluidos en el empaquetador del módulo, y harán las llamadas a través del servicio definido en el padre. No obstante, tendrán sus propios archivos de lógica y HTML.

Esta estructura es recursiva en tantos niveles como se desee, es decir, los componentes pueden tener componentes hijos que a su vez pueden tener componentes hijos y así sucesivamente un número ilimitado de veces.

6 DESARROLLO

Tras haber analizado el problema y diseñado la solución el siguiente paso es la implementación. En este capítulo se detalla el proceso de desarrollo de la solución explicando cómo se ha implementado cada una de las diferentes partes que forman la aplicación.

6.1 PLANIFICACIÓN

Para realizar el desarrollo de una manera estructurada y poder ajustarse al tiempo requerido para la realización de un Trabajo de Final de Grado, se ha realizado una distribución de las tareas mediante un diagrama de Gantt.

La planificación de tareas es un recurso muy importante para un ingeniero informático ya que estimar de una manera razonable el coste temporal permite gestionar mejor los plazos y el coste económico del proyecto.

Se ha dividido el desarrollo del proyecto en 5 bloques principales:

- **Inicio del proyecto:** Tareas iniciales de diseño y creación de la estructura básica del proyecto.
- **Desarrollo de las páginas:** Implementación de las diferentes páginas, incluyendo la creación de los componentes que las forman y la lógica de navegación.
- **Integración de la seguridad:** Creación de un sistema para proteger la aplicación de manera conjunta con el servidor.
- **Servicios de obtención de datos:** Implementación de los servicios que proveen de datos a la aplicación.
- **Pruebas:** Desarrollo de tests unitarios y ejecución de pruebas funcionales.

Sin detallar más las tareas se puede ver que existen dependencias entre bloques. El primer bloque, que incluye el inicio del proyecto, debe ser el primero en realizarse ya que consiste en crear la estructura base sobre la que se va a desarrollar todo el proyecto. Una vez creada la base, se desarrollan las páginas y la navegación entre ellas para conseguir una primera versión que contenga la funcionalidad de la aplicación. La integración de la seguridad y los servicios de obtención de datos pueden realizarse de forma paralela, realizando la integración de ambos en las páginas desarrolladas para dotarlas de estas funcionalidades más avanzadas. Finalmente se encuentra el proceso de pruebas que comprende los tests unitarios, los cuales deben realizarse paralelamente junto con la implementación del código, y las pruebas funcionales que se realizan una vez se haya finalizado el desarrollo del proyecto.

A continuación, en la Figura 12, se muestra el diagrama de Gantt generado.

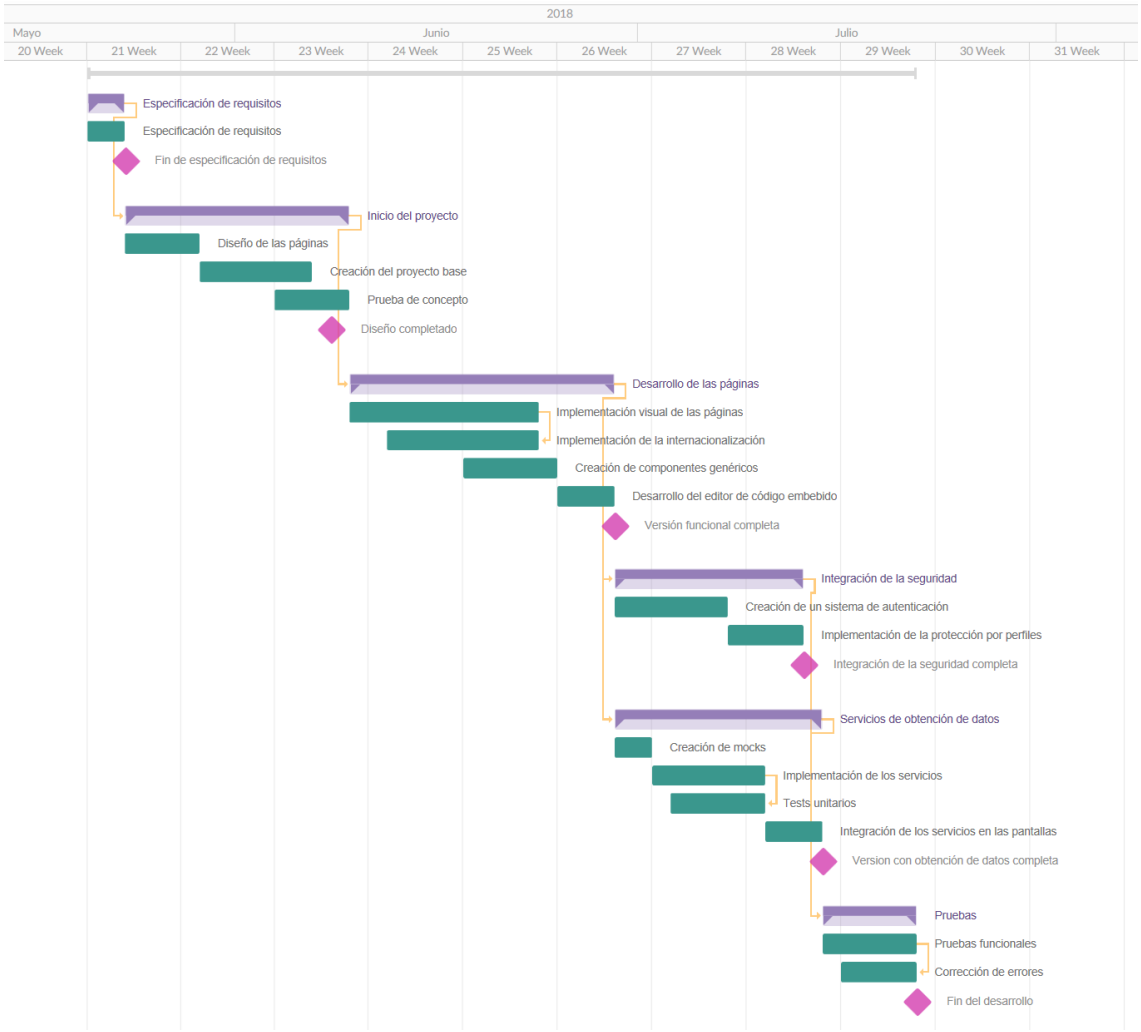

Figura 12: Diagrama de Gantt del Proyecto

En la Figura 12 se puede observar la distribución que se ha hecho de las tareas anteriormente mencionadas reflejadas en un diagrama de Gantt. En el diagrama se presentan en morado los bloques de tareas, en verde las subtareas de cada uno de los bloques y en rosa los hitos del proyecto.

6.2 Inicio del proyecto

El primer paso para iniciar el proyecto es instalar las herramientas necesarias. Para crear un proyecto con Angular se requiere el entorno de desarrollo Node y la herramienta Angular CLI, ambos explicados en la sección 3.

Con Angular CLI se puede crear un proyecto Angular en blanco que contiene la estructura mínima necesaria para poder compilarlo. De esta manera se genera un proyecto con la siguiente estructura:

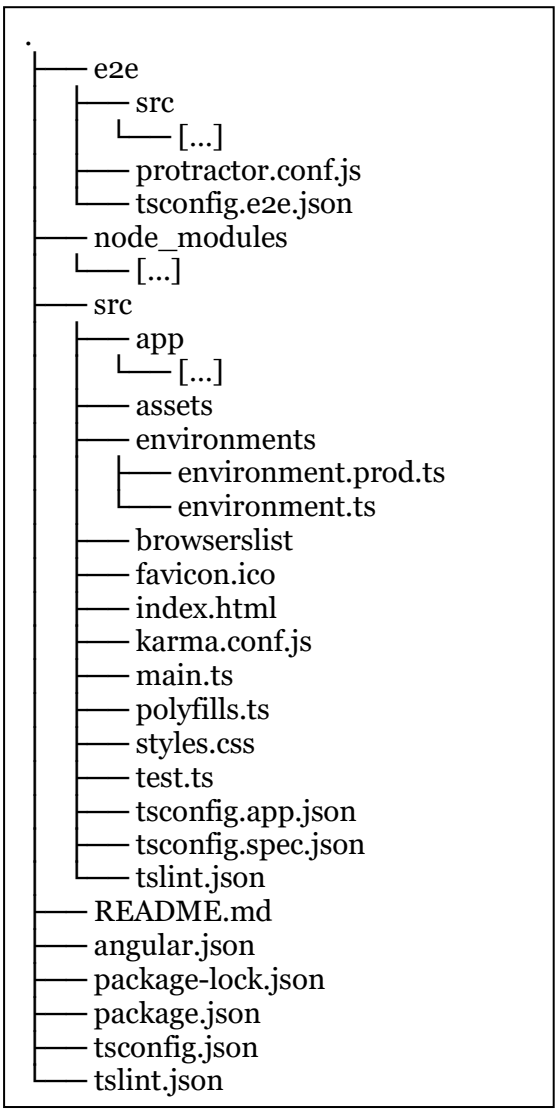

Figura 13: Estructura inicial del proyecto

En la Figura 13 se puede apreciar la estructura base que se genera al iniciar un proyecto con Angular CLI.

En la raíz del proyecto se encuentran varios archivos de configuración que son:

- **angular.json:** Configuración general del proyecto Angular.
- **package.json:** Contiene un listado de las dependencias del proyecto.
- **tsconfig.json:** Configuración del compilador de Typescript.
- **tslint.json:** Reglas para el analizador de código.

Estos archivos vienen configurados automáticamente por Angular CLI por lo que generalmente no se deben realizar grandes modificaciones.

Además de estos archivos, se encuentran tres directorios:

- **e2e:** Contiene un proyecto donde incluir los tests unitarios de protector con los que se comprueba el funcionamiento de los elementos de las páginas.
- **node_modules:** Contiene las dependencias que se hayan indicado en el package.json
- **src:** Carpeta padre del proyecto donde se ubican todos los archivos que formarán la aplicación.

Como Angular a partir de su versión 6 permite crear varias aplicaciones dentro de un mismo proyecto podemos ver que la carpeta src contiene también archivos de configuración que contienen la configuración especifica de la aplicación. En este caso, al crear una única aplicación, no es necesario tener configuraciones específicas.

Dentro del directorio *src* cabe destacar los siguientes elementos:

- **index.html y favicon.ico:** Angular genera aplicaciones de una sola página y esa página será el archivo *index.html*. Angular utilizará este archivo para cargar los diferentes elementos de la aplicación de manera dinámica sin recargar la página. El favicon.ico es el icono que esta página mostrará en la barra superior del navegador.
- **styles.css:** Estilos globales que se aplicarán a la aplicación. Normalmente se utiliza para definir el tema.
- **assets:** En esta carpeta se incluirán los recursos de la aplicación como pueden ser imágenes o archivos de internacionalización.
- **browserlist y polyfills.ts:** Archivos para mejorar la compatibilidad con navegadores de la aplicación. Vienen configurados por defecto para soportar las versiones actuales de los navegadores más populares.

- **karma.conf.json y test.ts:** Configuración e inicialización de los tests unitarios para la lógica de la aplicación con la herramienta Karma.
- **main.ts:** Contiene el script inicial de la aplicación mediante el cual Angular comienza a cargar toda la aplicación una vez se accede a ella.
- **environments:** En este directorio se incluyen archivos de configuración de entorno.
- **app:** Directorio donde se incluyen las diferentes páginas que conforman la aplicación.

6.2.1 Prueba de concepto

Para la realización de este proyecto se ha optado por utilizar la guía de estilos *Material Design* ofrecida por Google y para ello se va a hacer uso de la librería *Angular Material* que incluye múltiples componentes que siguen este diseño, así como facilidades para incluir temas en la aplicación. Sin embargo, se va a realizar un pequeño prototipo a modo de prueba de concepto para ver el aspecto que tendrá una página de la aplicación.

Para esta prueba se ha nombrado a la aplicación Purp UPV, siendo Purp un acrónimo de *Plataforma Universitaria de Retos de Programación*. Además, se ha incluido la dependencia de *Angular Material* y se ha definido un tema para la aplicación utilizando el morado como color principal debido a la similitud del nombre "Purp" con la traducción al inglés del color morado "purple".

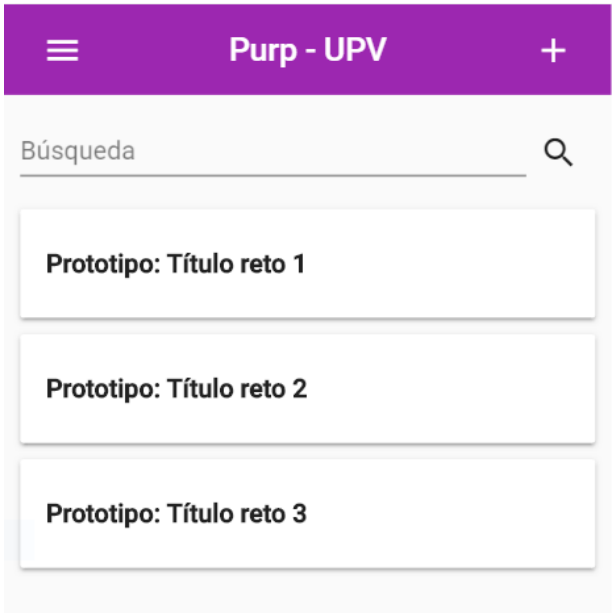

Figura 14: Implementación del prototipo

El resultado final del prototipo se puede observar en la Figura 14. Como se puede apreciar, se ha implementado el diseño que se describe en la sección 5.1.2 para la Página Principal. El resultado obtenido gracias a *Angular Material* es un aspecto muy común en las aplicaciones móviles de hoy en día.

Una vez revisado el diseño del prototipo y viendo que la apariencia es la que se deseaba para la aplicación, se puede continuar con el proceso de desarrollo.

6.3 DESARROLLO DE LAS PÁGINAS

Antes de comenzar a implementar las páginas se debe tener en cuenta que, siguiendo el diseño propuesto, la carga de páginas se debe hacer bajo demanda. Por este motivo se debe realizar una configuración previa y también definir la estructura de páginas que va a tener la aplicación.

6.3.1 Carga de páginas

La navegación y la carga de las páginas en Angular se realiza gracias al módulo *RouterModule*. En una aplicación básica de Angular, este módulo se encarga únicamente de la navegación, detectando los cambios en la URL que se produzcan y modificando la pantalla con el contenido requerido, pero en este proyecto se pretende hacer uso de la carga de páginas bajo demanda que este ofrece.

La carga de las páginas bajo demanda (también llamada *lazy*) quiere decir que el módulo que contiene cada página se compila de manera independiente y se carga únicamente cuando el usuario accede a ella. En caso de no utilizarse este sistema, Angular carga todas las páginas que componen la aplicación en el primer acceso a la página. Es por esto que utilizar la carga de páginas bajo demanda reduce drásticamente el tiempo de carga inicial de la aplicación.

No obstante, cargar los módulos bajo demanda hace que al pasar de una página a otra hayan tiempos de espera que de otra manera no ocurrirían. Esto se debe a lo comentado anteriormente, el no cargar los módulos al entrar en la aplicación provoca que deban ser cargados cuando el usuario acceda a la página, y eso genera un pequeño retraso.

Para implementar la navegación en Angular se necesita un módulo de enrutamiento que indica a Angular qué debe cargar cuando se acceda a determinadas rutas.

```
const routes: Routes = [
  { path: '', pathMatch: 'full', redirectTo: 'login' },
  { path: 'login', loadChildren: './login/login.module#LoginModule' },
  { path: 'paginas', loadChildren: './app-frame/app-frame.module#AppFrameModule' },
  { path: '**', pathMatch: 'full', redirectTo: 'login' }
];
@NgModule({
  imports: [ RouterModule.forRoot(routes) ],
  exports: [ RouterModule ]
})
export class AppRoutingModule { }
```

Figura 15: Módulo de enrutamiento principal

En la Figura 15 podemos observar el módulo de enrutamiento principal de la aplicación. Como se puede apreciar, la estructura de estos módulos es muy simple, siendo el principal elemento el array de rutas llamado “routes”. En este array se define qué módulo cargar para cada ruta que se introduzca en el navegador.

Este módulo de enrutamiento es el principal por lo que gestiona la carga de las rutas a partir del contexto de la aplicación, es decir, la URL donde está alojada la aplicación. Teniendo esto en cuenta, el primer elemento del array “routes” indica que, si accedemos a la URL de la aplicación sin especificar nada más, se redirige al usuario a la página de acceso añadiendo “/login” al final de la URL. El último elemento tiene un comportamiento similar salvo que indica a Angular que, si no reconoce la URL introducida, redirija al usuario a la página de acceso. Esto evita mostrar las típicas páginas de “Error 404 – Not Found”.

En la Figura 15 también se puede observar que si se accede a las rutas “URL/login” o “URL/paginas” se carga el módulo indicado, y es en este punto donde se hace uso de la carga de páginas bajo demanda. En una aplicación simple de Angular, en lugar de cargar otros módulos se cargarían directamente las páginas que se deben mostrar. De esa manera, este módulo de enrutamiento contendría la definición de todas las rutas de la aplicación, lo que provoca el problema anteriormente mencionado por el que todas las páginas se cargarían a la vez aumentando los tiempos de espera.

6.3.2 Jerarquía de páginas

El sistema que se ha utilizado en este proyecto consiste en indicar qué módulo se debe cargar para una ruta dada y este módulo tendrá que contener su propio módulo de enrutamiento donde definir qué página o módulo hijo se debe cargar. De esta manera se puede crear una estructura en forma de árbol donde únicamente se cargarán los módulos necesarios para la pantalla final que se desea visualizar. En la Figura 16 se puede observar el mapa de módulos de la aplicación donde se ha indicado las rutas de cada uno.

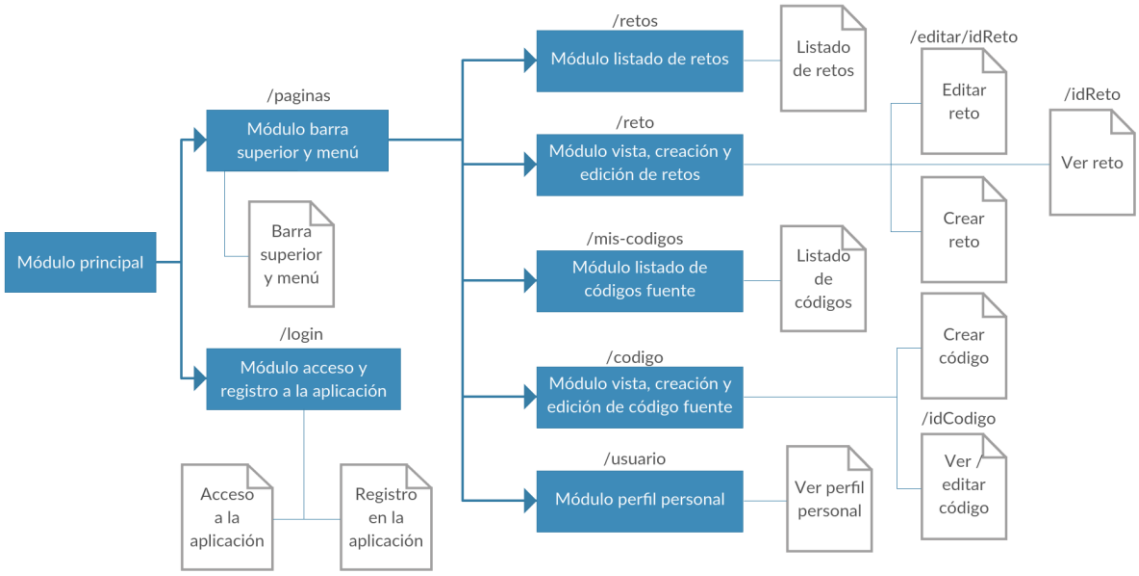

Figura 16: Mapa de módulos de la aplicación

En este mapa se puede observar la jerarquía de módulos que tiene la aplicación. En un primer nivel se encuentra el módulo de login, que carga las páginas de acceso a la aplicación y de registro, ya que esta sección de la aplicación no debe estar protegida por la seguridad de la aplicación. El módulo que contiene la barra superior y el menú, carga además de estos dos elementos un marco donde se introducirán todas las páginas de los módulos hijos. Por este motivo, el resto de los módulos que contiene la aplicación son hijos de este.

Como se puede apreciar, algunos módulos contienen diferentes páginas y, para poder diferenciar cuál cargar, se pueden definir diferentes rutas para cada una. De esta manera, siguiendo el mapa de la Figura 16, si se quiere acceder a la vista de un reto con identificador “1a2b3c” y la aplicación está desplegada en la máquina local en el puerto 8080 se debe introducir la siguiente URL:

`http://localhost:8080/paginas/reto/1a2b3c`

6.3.3 Internacionalización

Esta aplicación está orientada a utilizarse en la *Universitat Politècnica de València* que tiene dos idiomas oficiales (castellano y valenciano), además de que tiene alumnos internacionales que pueden sentirse más cómodos con el uso de inglés. Teniendo esto en mente se ha decidido ofrecer la aplicación en estos tres idiomas, proceso que se conoce como internacionalización.

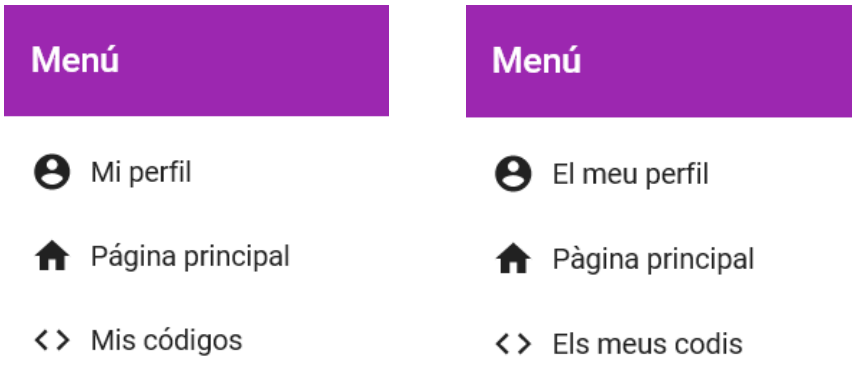

Figura 17: Menú de la aplicación en castellano y valenciano

En el ejemplo de la Figura 17 se pueden apreciar las diferencias entre el menú cuando la aplicación se muestra en castellano y cuando se muestra en valenciano.

Como se ha explicado en la sección 3.5.3 para este proceso se va a hacer uso de la librería *ngx-translate*, una librería que no es oficial de Angular pero que tiene un uso muy extendido entre los usuarios de este framework.

Esta librería se compone de dos elementos, por un lado, el *http-loader* que se encarga de obtener y leer los archivos con traducciones y por otro, el *core* que provee utilidades para aplicar las traducciones en las pantallas.

La carga de los archivos de traducciones se puede hacer de manera local, es decir que estos archivos se incluyen en la aplicación, o de forma remota, donde los archivos de traducciones se encuentran en el servidor y se obtienen mediante un servicio web. Debido a que inicialmente esta aplicación no va a contener un gran volumen de textos se ha optado por tratarlos de manera local, haciendo que el cambio entre idiomas sea mucho más rápido. Sin embargo, se considera una buena práctica obtenerlos de manera remota ya que permite realizar modificaciones en las traducciones sin tener que volver a desplegar la aplicación en el servidor. Por este motivo se ha dejado preparada la configuración de esta librería para que en una futura ampliación se puedan obtener los archivos de configuración de manera remota sin tener que realizar grandes modificaciones.

El formato con el que trabaja esta librería para los archivos de traducciones es JSON. Este tipo de archivos tienen un formato clave-valor y en este caso la clave es el código de la traducción y el valor el texto traducido. Además, para facilitar la organización, se pueden crear niveles permitiendo así agrupar conjuntos de claves-valores.

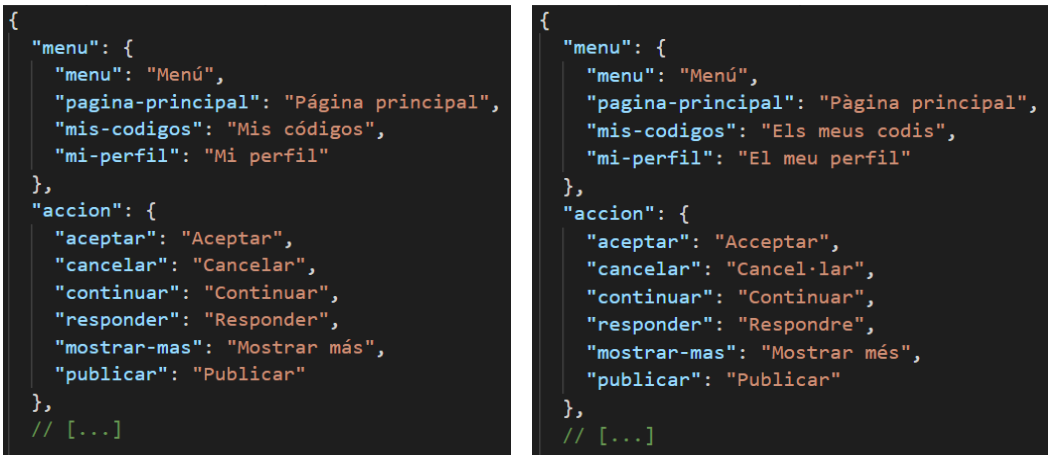

Figura 18: Comparativa ficheros de internacionalización de castellano y valenciano

Como se puede apreciar en la Figura 18 donde se muestra una comparativa entre los ficheros de internacionalización para los idiomas castellano y valenciano, las claves del documento (los textos azules) son idénticas pero los valores contienen el texto traducido para cada idioma. Por ejemplo, si al implementar la página se utiliza la clave "accion.cancelar" y se le indica al servicio de traducción que se debe traducir, un usuario que tenga configurado el idioma en castellano verá el texto "Cancelar" mientras que otro que lo tenga en valenciano verá el texto "Cancel·lar".

6.3.4 Página de acceso y registro a la aplicación

La primera página que se ha desarrollado es la de acceso a la aplicación ya que es una página bastante simple que permite empezar a poner en práctica todos los elementos explicados anteriormente.

La pantalla de acceso se ha diseñado con una barra superior, dos campos para el usuario y la contraseña, el botón acceder y un enlace al registro. Este tipo de pantallas donde se deben rellenar ciertos datos y realizar son muy usuales en el desarrollo de aplicaciones web, tanto es así que en HTLM existe la etiqueta "form" que permite crear este tipo de formularios de una manera más cómoda. Además, Angular ha querido potenciar todavía más este elemento con los denominados *ngForm*, una capa adicional de los formularios que permite gestionarlos de una manera más simple y enlazarlos con el resto de elementos de Angular.

Como la única diferencia entre la pantalla de acceso y la de registro es el formulario que contienen, se han creado estas dos secciones como componentes Angular. En el bloque central de la página se carga el componente que se requiera según si se está iniciando sesión o registrando a la aplicación. El cambio de modo se realiza mediante el enlace situado bajo el botón de acceso/registro.

Como se puede apreciar en la Figura 19, se ha seguido el diseño propuesto, pero quedaba mucha pantalla sin usar por lo que se ha decidido incluir el logotipo de la universidad. Aprovechando que es el 50 aniversario de la universidad, se ha escogido el logotipo conmemorativo que se ha diseñado para la ocasión.

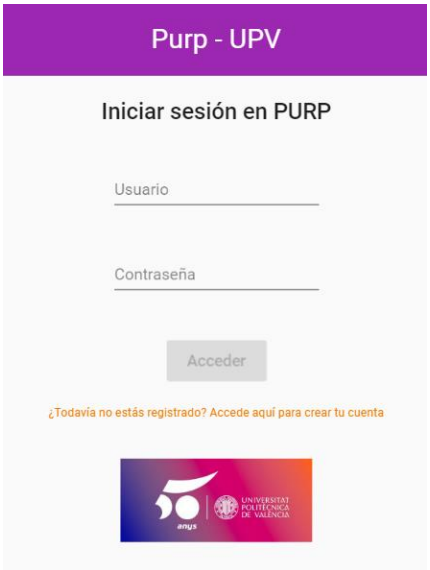

Figura 19: Acceso a la aplicación para pantallas pequeñas

Se debe aclarar que la utilización del nombre y logotipo de la universidad no significa que la aplicación esté integrada con la universidad. Se utilizan para mostrar de una manera más fiel lo que podría ser una versión final puesta en marcha e institucionalizada por la *Universitat Politècnica de València*.

Una característica de la aplicación es que va a estar disponible tanto para móviles como para ordenadores, por lo que con este diseño en pantallas más grandes seguían teniendo demasiado espacio sobrante. Por este motivo, se han aprovechado las características que Angular ofrece para el diseño adaptativo y se ha creado un segundo diseño que se activa únicamente para pantallas de mayor tamaño, así quedan estéticamente mejor los espacios sin utilizar.

Este diseño para pantallas más grandes se puede apreciar en la Figura 20, en él se ha creado un bloque central elevado lo que reduce el espacio útil de la pantalla sin que dé sensación de estar desaprovechado.

Otro recurso que se ha utilizado es dividir el bloque central verticalmente y rellenar la sección izquierda del color principal, incluyendo el título de la aplicación y el logotipo de la universidad. De esta manera el componente con el formulario, que es lo realmente útil de la página, queda enmarcado en un contenedor más ajustado.

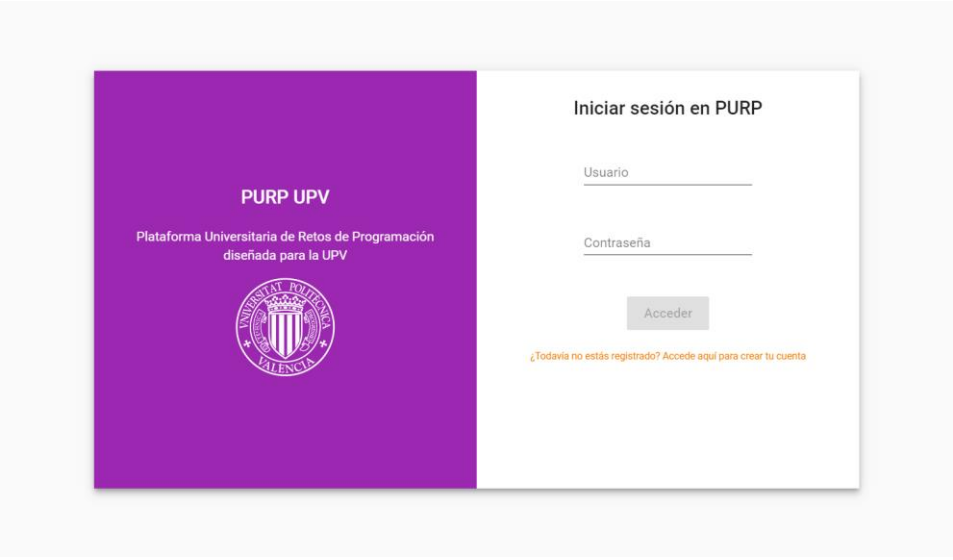

Figura 20: Acceso a la aplicación para pantallas grandes

6.3.5 Marco de la aplicación

Una vez se accede a la aplicación, esta cuenta con una barra superior y el menú visibles en todo momento, a esto se le ha denominado marco de la aplicación. Para estos elementos se ha vuelto a hacer uso de las herramientas que ofrece Angular para el diseño adaptativo. De esta manera el menú de la aplicación, cuando se accede desde el móvil, se despliega haciendo uso de un "*Hamburger Button*" situado a la izquierda en

la barra superior y se superpone a la pantalla en la que se encuentre el usuario. Por el contrario, al acceder desde un ordenador no existe este botón y el menú está siempre visible en una sección a la izquierda de la pantalla. En la Figura 21 se muestra una comparativa mostrando la diferencia:

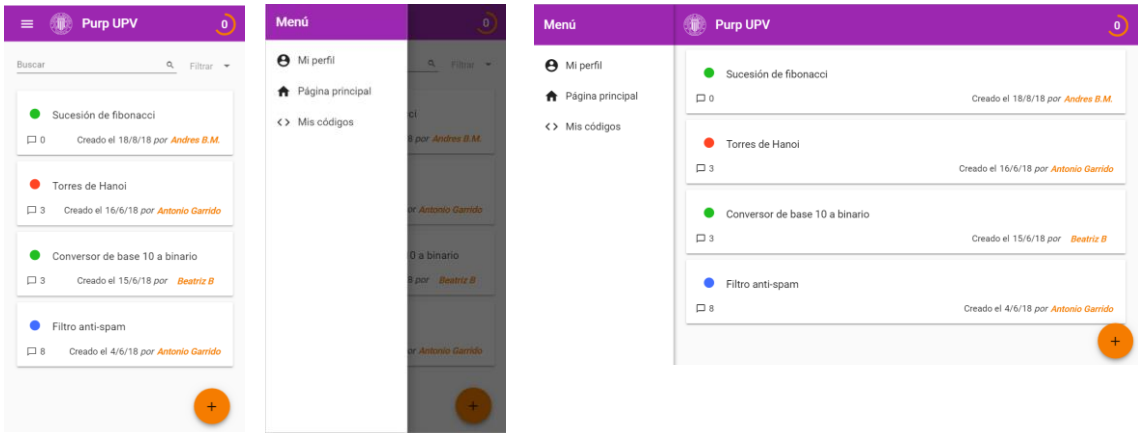

Figura 21: Menú en un dispositivo móvil y en un ordenador

A la derecha de la barra superior se encuentra un icono que representa el nivel y la experiencia del usuario. Este sistema de experiencia se ha creado para potenciar la gamificación de la aplicación, creando una competencia entre los usuarios que los motive a continuar haciendo uso de la aplicación. La experiencia es un parámetro del usuario que se actualiza al realizar determinadas acciones como publicar un reto o recibir un voto positivo en un comentario. La actualización de este parámetro se realiza automáticamente por el backend de la aplicación por lo que para este proyecto únicamente se debe recuperar y mostrar.

El espacio restante de la pantalla se reserva para un contenedor donde incluir el contenido de las pantallas que conforman la aplicación. En la comparativa de la Figura 21 se muestra como la pantalla cargada en el contenedor del marco es la página principal.

6.3.6 Página principal de la aplicación

La página principal es la primera a la que se accede al iniciar sesión y contiene un listado de retos. En esta pantalla los usuarios pueden encontrar los retos en los que participar y se incluye una sección para filtrarlos y que el usuario pueda encontrar aquellos retos en los que está interesado en participar. Inicialmente se pretendía crear dos pantallas diferenciadas para retos y competiciones, pero se ha encontrado redundante ya que se ha incluido un selector para filtrar los retos según su tipo (de alumnos, de profesores y competiciones) dando la posibilidad de obtener las competiciones desde la propia página principal.

Para el desarrollo de la página principal se ha creado un componente que muestra la información de un reto en una tarjeta. Este componente se ha creado en un módulo común para que pueda ser utilizado en otras pantallas de la aplicación, de esta manera si se quiere mostrar información resumida de un reto no es necesario implementar de nuevo un elemento como este. El componente de reto se muestra en la Figura 22:

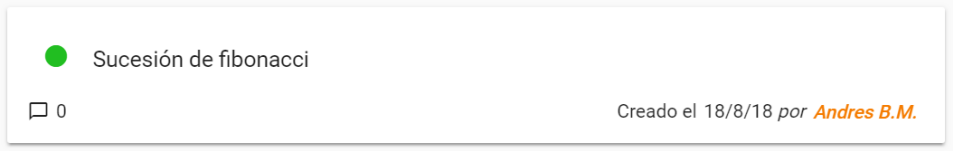

Figura 22: Tarjeta de reto

Esta tarjeta de reto contiene la información más relevante del reto de forma visual para que los usuarios puedan recibirla en un primer vistazo.

En la parte izquierda de la tarjeta se incluye un marcador que indica el tipo de reto mediante colores, utilizando el rojo para competiciones, azul para retos de profesores y verde para retos de alumno. Esta distinción se realiza para dar más importancia a las competiciones y retos de profesores. Por un lado, las competiciones son retos especiales enfocados a gamificar la aplicación ya que disponen de un tiempo límite y al acabar se mostrará un ranking con los usuarios que hayan participado. Por otro lado, los retos de profesores teóricamente no son distintos a los retos de alumnos, no obstante, al ser creados por profesores la explicación y el objetivo del reto estarán generalmente más cuidados que los retos que puedan proponer los alumnos y por eso se ha querido diferenciar entre ambos.

El texto central de la tarjeta se corresponde al título del reto, el cual debe indicar de manera breve qué se va a pedir en la descripción del reto para que el usuario pueda saber sin necesidad de acceder al detalle si le interesa o no participar en él. En la parte inferior se muestra otra información que puede resultar relevante al usuario como puede ser la cantidad de comentarios que se han realizado, la fecha en la que se ha publicado o el usuario que lo ha publicado.

El principal problema que se ha tenido que abordar durante la creación de esta pantalla ha sido la carga de retos. En esta pantalla se pueden encontrar todos los retos de la plataforma desde sus inicios, algo que en este momento de desarrollo no supone un problema pero que sí lo sería una vez la aplicación estuviese activa en un entorno real debido al volumen de datos que se deberán manejar. Para abordar este problema se ha implementado un sistema llamado “scroll infinito” que consiste en cargar los elementos de la lista únicamente cuando el usuario se acerca al final de la lista. Para realizar esta implementación se ha hecho uso del sistema de paginación que proporciona el backend y que permite obtener una sección (o página) del listado total. También se ha implementado la lógica necesaria que captura el evento de scroll sobre la página y realiza la petición al servidor para cargar más elementos conforme se acerca a la parte inferior. Sobre estas peticiones se hablará más adelante.

Otro elemento que se ha introducido es un botón flotante para la creación de nuevos retos. Este elemento no se encontraba en los bocetos del diseño de la aplicación, pero siguiendo la guía de diseño *Material Design* se ha decidido incluir ya que representa una función única y principal de la pantalla[19].

[19] Material Design – Floating Action Button (Acceso Junio 2018): https://material.io/design/components/buttons-floating-action-button.html

6.3.7 Creación y edición de reto

Los retos planteados se deben crear desde la aplicación, para ello se ha desarrollado una pantalla que incluye un editor de texto que permite aplicar estilos al texto de la descripción.

Figura 23: Editor de retos

Como se puede observar en la Figura 23 el editor de textos contiene múltiples herramientas que ofrecen al creador del reto diversas posibilidades para formatear el texto. La característica principal de este editor ofrecido por TinyMCE es que se puede visualizar el estilo final mientras se está editando el texto, lo que facilita saber cuál será el resultado final. No obstante, para que se pueda tener una visión más fiel al resultado final de la aplicación, se ha implementado un botón de previsualización que muestra al usuario cómo quedará el reto una vez publicado.

Adicionalmente esta pantalla cuenta con la posibilidad de añadir etiquetas al reto. Dichas etiquetas contienen un texto libre y sirven para que se puedan incluir términos que serán analizados cuando un usuario realice una búsqueda de retos.

6.3.8 Vista de reto

Esta pantalla contiene la información completa de un reto. Para desarrollar esta página se ha tenido que afrontar el problema de visualizar el reto con los estilos con los que se ha creado. La descripción del reto se guarda en la base de datos en formato HTML para que pueda contener el formato con el que se ha diseñado. Para mostrar este texto de manera correcta se embebe el código en un contenedor con la propiedad *innerHTML* que permite que el texto introducido sea interpretado como HTML en vez de texto plano. Gracias a esto se puede obtener un resultado como el que se observa en la Figura 24:

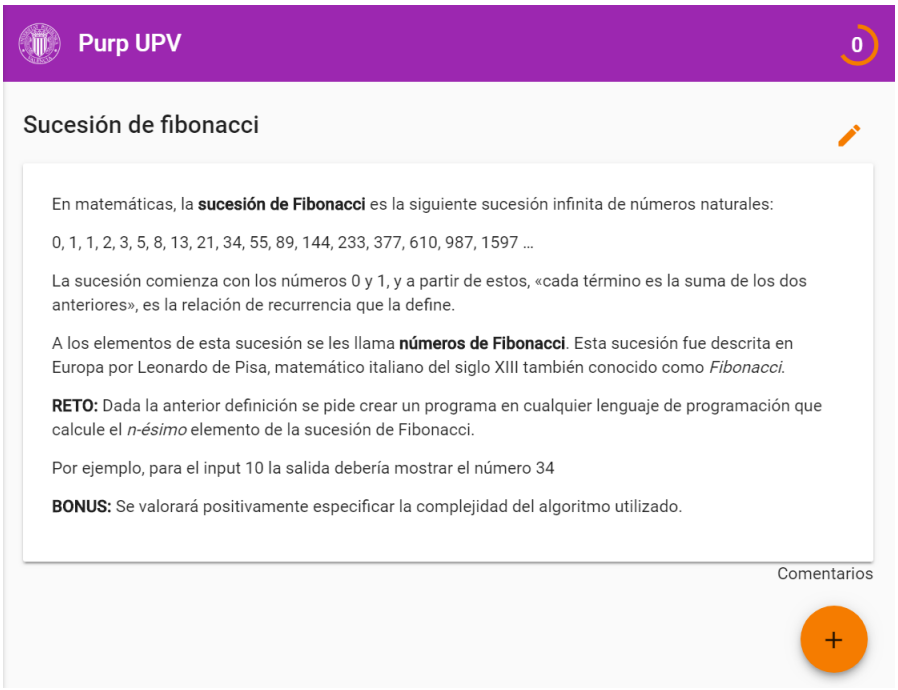

Figura 24: Vista de reto

Una funcionalidad indispensable de esta página es la publicación de comentarios, ya que es la manera que tienen los usuarios de participar en los retos. Para ello se ha dispuesto un botón flotante en la parte inferior derecha como en la página principal que permite al usuario publicar un reto. Al pulsar este botón, se despliega un panel en la parte inferior con un campo de texto donde introducir el texto del comentario que se desea realizar.

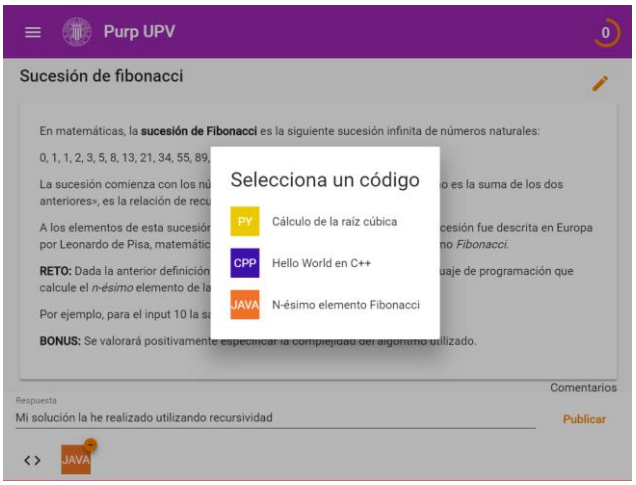

Figura 25: Publicación de un comentario

Como muestra la Figura 25, el panel para añadir un comentario también cuenta con un botón para adjuntar códigos al comentario que, al pulsarlo, muestra un cuadro con los códigos que tiene el usuario en su espacio personal. Una vez seleccionados, estos códigos se representan bajo el texto del comentario mediante un cuadrado con las siglas o abreviaciones de los lenguajes y el color de fondo predominante en los logotipos de cada uno de ellos, de esta manera los usuarios pueden distinguirlos de una manera rápida. Además, para aquellos que utilicen la aplicación desde un ordenador al pasar el ratón por encima de este recuadro muestra el título del código fuente. Para poder

eliminar un código adjuntado, se ha incluido un botón con el símbolo "-" que realiza esta acción.

Tras la descripción se ha añadido un botón que muestra la cantidad de comentarios que han sido publicados en ese reto y, al pulsarlo, despliega un listado de comentarios. Para este listado se ha seguido la misma técnica de "Scroll infinito" descrita en el desarrollo de la página de inicio.

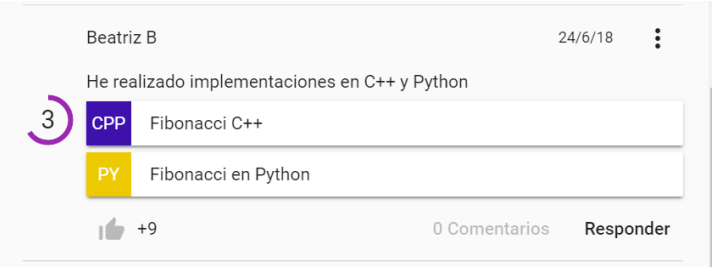

Figura 26: Tarjeta de comentario

Para mostrar un comentario se ha creado un componente como el que muestra la Figura 26. Este se compone del indicador de nivel del usuario a la izquierda y el nombre de usuario en la parte superior, esto se hace para motivar a los alumnos a superarse entre ellos ya que en todo momento pueden ver el nivel que tienen sus compañeros dentro de la plataforma. Dado que los profesores también pueden comentar pero no están incluidos en este sistema de experiencia, se les asigna un icono que los identifica como profesores. Esta distinción hace que resulte más sencillo identificar los comentarios más relevantes (los comentarios de profesores) de otras respuestas que pueden ser más imprecisas. Esto potencia el aspecto didáctico de la plataforma creando una comunidad donde no solamente se resuelven retos, sino que también se puede obtener una retroalimentación fiable sobre los errores o mejoras de la solución que se ha propuesto.

El contenido principal del componente de comentario se compone de dos partes: el comentario en formato texto y los códigos adjuntos. La respuesta textual del usuario es donde debe justificar la solución al reto propuesto o realizar una crítica a otro comentario. Los códigos adjuntos se listan a continuación del texto del comentario debido a que las respuestas a los retos pueden requerir de una implementación. Además, los códigos fuente adjuntados son enlaces directos a la vista del código de modo que al pulsar sobre ellos se pueden visualizar y ejecutar.

En la sección inferior del componente se encuentra en la parte izquierda un botón para dar *me gusta* al comentario con la forma de pulgar hacia arriba, y en la parte derecha dos botones para mostrar las respuestas que haya recibido ese comentario y para añadir una respuesta. Las respuestas del comentario hacen uso del mismo componente pero aparecen con un pequeño margen izquierdo y el indicador de nivel aparece en diferente color, facilitando así la distinción entre soluciones al reto y respuestas a estas soluciones.

6.3.9 Vista y edición de código fuente

Para crear los códigos fuente de las soluciones a los retos se facilita a los usuarios una herramienta para poder implementar el código desde la propia aplicación. Esta herramienta es la pantalla de edición de código fuente.

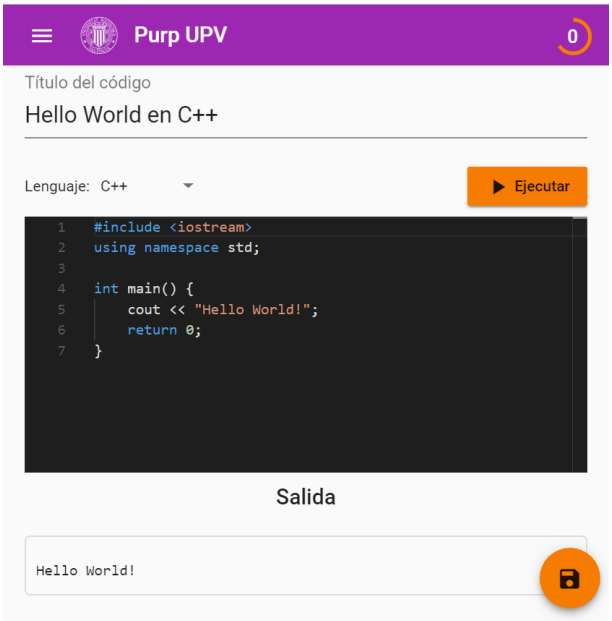

Figura 27: Edición de código fuente

Esta pantalla incluye un editor de código embebido que permite al usuario de la plataforma realizar sus implementaciones. Para incluir este editor se ha hecho uso de la librería *Monaco Editor* ya que proporciona resaltado de sintaxis para los principales lenguajes de programación, incluidos los que permite esta plataforma.

Como se puede apreciar en la Figura 27, en la parte superior de la pantalla se incluye un selector de lenguaje. Este selector no solamente indica qué lenguaje se ha utilizado, sino que se ha implementado una funcionalidad por la cual, al cambiar de lenguaje de programación, se carga en el editor una plantilla básica para aquellos lenguajes que requieran alguna estructura inicial como puede ser un método *main* para su ejecución.

Una vez creado el código puede ser ejecutado para obtener el resultado ya sea la salida por consola que se haya implementado o los errores de compilación que pueda tener el código. Esta ejecución del código no la realiza la propia aplicación, sino que se consigue realizando una llamada a una API ofrecida por la plataforma SoloLearn que devuelve el resultado de la ejecución. Esta herramienta es fundamental en la aplicación ya que permite a los usuarios probar sus soluciones sin necesidad de salir de la aplicación.

Esta pantalla se utiliza también para la visualización de códigos fuente de otros usuarios, pero en ese caso se elimina el botón guardar. De esta manera al ver una solución propuesta por otro usuario puedes ejecutarla tal cual la publicó el creador o realizar modificaciones y ver cómo varía el resultado.

6.3.10 Perfil personal

La última pantalla desarrollada es la de perfil personal, donde el usuario puede revisar su información y realizar ajustes en la aplicación. Esta pantalla se divide en secciones que se reorganizan para ocupar el espacio disponible en la pantalla.

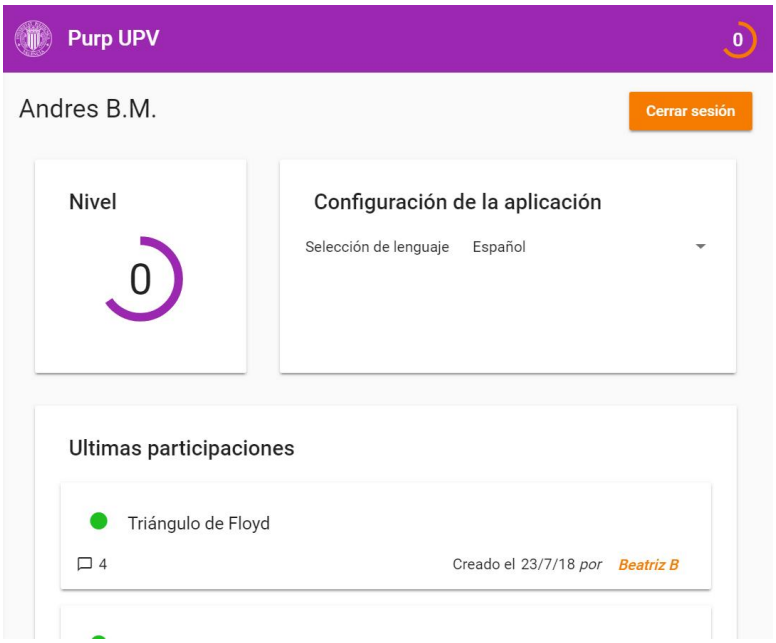

Figura 28: Perfil personal

El primer bloque de la pantalla contiene un componente que muestra el nivel del usuario. Este componente ya se ha visto tanto en la barra superior de la aplicación mostrando el nivel del usuario como en los comentarios de los retos donde muestra el nivel de quien lo ha publicado. En él se calcula el nivel del usuario en base a la experiencia siguiendo la siguiente fórmula:

$$Nivel\ de\ usuario = \sqrt[5]{\left(\frac{Experiencia\ del\ usuario}{100}\right)^4}$$

Ecuación 1: Fórmula para el cálculo del nivel del usuario

En la Ecuación 1 se calcula el nivel del usuario utilizando una fórmula que permite incrementar la cantidad necesaria para subir de nivel de forma progresiva. Esta fórmula se ha creado con el objetivo de crear una curva exponencial que permita configurar dos aspectos del cálculo del nivel:

- La experiencia necesaria para alcanzar el nivel 1, ya que esto fija la base en la que se calcula este sistema de experiencia y niveles. Este parámetro se corresponde con el divisor de la ecuación que en este caso es 100.
- La progresión de experiencia necesaria al ir avanzando de nivel, ya que de este modo los niveles más bajos se obtendrán más rápidamente y los más altos requerirán de un esfuerzo mayor. Esta progresión viene dada por la ratio entre el exponente y el radical en este caso 4/5.

De esta manera para subir al primer nivel se requieren 100 puntos de experiencia mientras que para avanzar al segundo nivel se requiere obtener 138 puntos de experiencia. Para no mostrar decimales en el nivel se muestra la parte entera como número y una circunferencia exterior que se va rellenando cuanto más te acercas al siguiente nivel.

El segundo bloque de la aplicación se utiliza para incluir la configuración que los usuarios pueden realizar sobre la aplicación. En esta primera versión de la aplicación únicamente permite configurar el idioma de esta mediante un selector que incluye los

idiomas castellano, valenciano e inglés como se ha mencionado en la sección de internacionalización.

Finalmente se incluye un bloque que muestra la actividad reciente del usuario, más concretamente los últimos retos en los que ha participado, para poder llevar un seguimiento de las actualizaciones.

6.4 SERVICIOS DE OBTENCIÓN DE DATOS

Todos los datos mostrados en la aplicación se deben de obtener del servidor y este proceso se realiza mediante peticiones a servicios web. Para gestionar estas peticiones se ha desarrollado una capa de servicios que se encarga de hacer las llamadas a las direcciones y con el formato que define la API publicada por el TFG complementario a este. Sin embargo, al comienzo del desarrollo estos servicios no están publicados todavía por lo que se debe simular las respuestas obtenidas.

6.4.1 Simulación de peticiones

La disponibilidad de los servicios web es un problema recurrente en el desarrollo de aplicaciones web, por este motivo Angular ha desarrollado un servicio que permite simular la respuesta que debería dar el servidor al realizar peticiones web.

Para simular las peticiones con este servicio se debe definir una base de datos haciendo uso de la estructura de objetos propia de JavaScript, en el ejemplo de la Figura 29 se puede apreciar esta estructura.

```
createDb() {
  const usuarios: Usuario[] = [···
  const retos: Reto[] = [
    {
      id: 'db4e1073f2ef5d665af8',
      usuario: usuarios[1],
      titulo: 'Longest Common Subsequence',
      tipo: 'A',
      fechaCreacion: new Date(2018, 4, 1),
      estado: 0,
      comentarios: 15,
      descripcion: '<p>La búsqueda de la secuencia...'
    }
  ];

  return { retos, usuarios };
}
```

Figura 29: Base de datos para la simulación de servicios

La base de datos se define como un diccionario donde la clave es el nombre de la tabla y el valor es el contenido. En el ejemplo se ve que se han definido dos tablas: usuarios y retos.

El contenido de cada tabla se representa mediante un array donde un registro de la tabla es un elemento del array, siguiendo el ejemplo la tabla "retos" contiene únicamente una fila.

Finalmente, cada registro se representa como un diccionario donde la clave es el nombre de la columna y el valor es el contenido del registro en esa columna.

Una vez creada esta base de datos simulada, Angular captura las peticiones REST que se realicen en la aplicación que tengan el formato "/api/[tabla]" donde tabla es el nombre de una tabla definida en esta base de datos, y devuelve o actualiza su contenido siguiendo la metodología *RESTful*[20]. A grandes rasgos esta metodología asocia métodos HTTP a acciones CRUD sobre la base de datos siguiendo el siguiente esquema:

- **GET:** Recupera los registros de la tabla.
- **POST:** Crea nuevos registros.
- **PUT:** Actualiza registros existentes.
- **DELETE:** Elimina un registro de la tabla.

[20] What is REST API (Acceso Julio 2018): https://restfulapi.net/

De esta manera se pueden implementar las peticiones web que realizará la aplicación al servidor de backend, pero serán redirigidas para actuar sobre esta base de datos simulada evitando la problemática de que no estén disponibles los servicios web necesarios.

6.4.2 Implementación de los servicios.

Para mantener estructurado el código del proyecto se ha creado una capa de servicios encargada de realizar las comunicaciones con el servidor de la aplicación.

Para realizar estos servicios se ha hecho uso de varias técnicas y librerías que permiten gestionar tanto las peticiones como las respuestas obtenidas, facilitando el acceso a datos desde el resto de la aplicación.

La principal librería que se utiliza es *HttpClient*, que viene incluida en el paquete *core* de Angular. Esta librería es la que permite realizar las peticiones web de una manera sencilla además de ofrecer un control total sobre la estructura de la petición.

Las peticiones realizadas haciendo uso de la librería anterior devuelven un objeto de tipo *Observable* definido en la librería *rxjs,* orientada a facilitar el uso de programación reactiva. Este tipo de programación, a diferencia de la secuencial, se basa en reaccionar ante eventos como por ejemplo la recepción de una respuesta a una petición web.

Finalmente, aprovechando que TypeScript es un lenguaje tipado, se han definido las clases de los objetos que se pueden recibir como respuesta del servidor. La definición de estas clases es muy útil ya que permite realizar transformaciones a los objetos recibidos como el casteo de fechas, que se reciben como un campo numérico representando los milisegundos, a objetos de tipo *Date* que los navegadores pueden interpretar como fechas.

```
getReto(id: string): Observable<Reto> {
  // Se realiza una llamada de tipo GET al servidor
  return this.http.get(this.utils.getURL(PATH, id)).pipe(
    // Se transforma la respuesta recibida a un objeto de tipo Reto
    map(reto => new Reto(reto))
  );
}
```

Figura 30: Ejemplo de llamada HTTP

Como se puede observar en el ejemplo de la Figura 30, se utiliza la librería *HttpClient* (a través de la instancia definida como “this.http”) para realizar una petición al servidor de tipo GET y, una vez se obtenga la respuesta, esta será mapeada a un objeto de tipo Reto. El método devuelve un *Observable* que emitirá un evento con el valor de la respuesta ya mapeada a Reto, de manera que se pueda reaccionar a esta respuesta y utilizar los datos recibidos.

En la Figura 30 destaca la manera en la que se obtiene la URL ya que se hace uso de un método “getURL” del servicio “utils”. Este servicio se ha creado para evitar repetir código en múltiples puntos de la aplicación. El método utilizado en este caso en concreto permite obtener la URL adecuada dependiendo del entorno donde se esté ejecutando la aplicación. Esto resuelve una problemática común en el desarrollo de aplicaciones web, que los servicios que se consumen durante el desarrollo y las pruebas de la aplicación no sean los mismos que utilizará la aplicación cuando esté ejecutándose en un entorno real. Por este motivo se definen las direcciones correctas para cada entorno de ejecución y este método utiliza aquella que corresponde al entorno donde la aplicación se esté ejecutando en ese momento.

En ocasiones estas peticiones obtienen un listado demasiado extenso como para manejarlo de forma eficiente por la aplicación, por lo que se ha decidido implementar un sistema de paginación que obtiene estos resultados en segmentos más pequeños. Para ello se inserta en la petición al servidor una petición de página y el servidor responde a esta con el subconjunto de datos requerido. La implementación de este sistema en la aplicación consiste en tres elementos:

- **Clase PageRequest:** Se ha creado una clase *PageRequest* que contiene la información que se necesita proporcionar al servidor al realizar una petición con paginación.
- **Método buildPageParams:** Se ha implementado un método en el servicio *utils* que construye los parámetros de la petición a partir del objeto *PageRequest* recibido.
- **Clase Page:** Se ha creado una clase *Page* que contiene el modelo de página que se recibe del servidor. En ella se incluye tanto el fragmento de lista que compone la página como información referente al número de elementos por página, elementos totales, página actual, etc.

```
getRetoComents(idReto: string, pageRequest: PageRequest): Observable<Page<Comentario>> {
  // Se construyen los parámetros de la aplicación a partir de la petición de página
  const params = this.utils.buildPageParams(pageRequest);
  return this.http.get<Page<Comentario>>(this.utils.getURL(PATH, idReto, 'comentarios'), {params}).pipe(
    map(page => {
      // Se mapea el contenido de la página al tipo Comentario
      page.content = page.content.map(comentario => new Comentario(comentario));
      return page;
    })
  );
}
```

Figura 31: Obtención de comentarios utilizando paginación

En la Figura 31 se muestra cómo se consume el servicio “Recuperar comentarios del reto” que a través del id de reto y los parámetros de la petición permite obtener todos los comentarios del reto de manera paginada. Se puede encontrar la documentación de la API en el anexo del TFG complementario creado por Beatriz Benedicto Granell. En él se incluyen tanto este como el resto de los servicios que se consumen en la aplicación.

6.4.3 Integración de los servicios en las páginas

Una vez se dispone de los servicios, se puede hacer uso de estos desde cualquier página o componente de la aplicación. Estos servicios son un pilar fundamental de la aplicación ya que toda la funcionalidad que implementa se basa en la manipulación de los datos recibidos por el servidor.

Gracias a Angular, y más concretamente a su inyector de dependencias, la utilización de estos servicios no requiere de mucho esfuerzo. El inyector de dependencias es una herramienta que se encarga de mantener una instancia de cada servicio que se defina y, cuando una página o componente requiere hacer uso de uno de ellos, provee la instancia del servicio solicitado.

Al utilizar un método de los servicios de obtención de datos se devuelve un *Observable*. A este objeto se le puede especificar una función de retorno que se ejecutará cuando finalmente se obtenga la respuesta del servidor. Esta función contiene las acciones necesarias para hacer uso de los datos recibidos como en el ejemplo siguiente extraído de la lógica de la pantalla de vista de reto mostrado en la Figura 32. En este ejemplo se recibe la información del reto solicitado y, en caso de tratarse de una competición, se calcula el tiempo límite del que dispone el usuario para contestar.

```
// Obtiene el reto a partir de su ID
this.service.getReto(id).subscribe(reto => {
  this.reto = reto;
  // Si el reto es una competición, calcula el tiempo restante hasta su finalización
  if (this.reto.tipo === 'C') {
    const timeDiff = Math.round((this.reto.fechaLimite.getTime() - new Date().getTime()) / 1000);
    this.secondsLeft = Math.max(timeDiff, 0);
  }
});
```

Figura 32: Ejemplo de utilización de servicios

6.5 SEGURIDAD DE LA APLICACIÓN

Una aplicación es segura cuando posee un mecanismo de autentificación, de manera que solo pueden acceder a la aplicación los usuarios permitidos.

El sistema más común cuando se desea hacer segura una aplicación móvil es la utilización de tokens, concretamente los *JSON Web Tokens* o *JWT*. Este tipo de tokens contienen información del usuario y están criptográficamente firmados, lo que permite asegurarse de que esa información no ha sido alterada en ningún momento.

El uso de este sistema de autentificación viene definido por el servidor backend de la aplicación. A continuación se describe a grandes rasgos los pasos que sigue este sistema para tener una idea de su funcionamiento:

1. Se envían las credenciales del usuario al servidor.
2. El servidor valida las credenciales y devuelve un token *JWT* con la información del usuario.
3. Se recibe el token y se almacena para identificarse en el futuro.
4. Cada vez que se realiza una petición, se adjunta el token para identificarse.
5. El servidor recibe la petición, extrae el token de la petición y lo valida.
6. Finalmente, si el token es válido, la petición es respondida.

Siguiendo los pasos anteriores se puede observar que hay dos operaciones principales que se deben implementar en la aplicación. Estas son el proceso de adquisición de un token a partir de las credenciales y la autenticación mediante el token en posteriores peticiones.

6.5.1 Adquisición del token

La adquisición del token se realiza a través de la pantalla de acceso a la aplicación donde el usuario introduce su usuario y contraseña. Una vez pulsa el botón acceder, se hace uso del servicio de usuario que se ha implementado para gestionar las peticiones de acceso y registro de la plataforma:

```
login(username: string, password: string): Observable<boolean> {
  return this.http.post<any>(this.utils.getURL('login'), { username: username, password: password }).pipe(
    map(res => {
      // Si la respuesta contiene un token es que se ha validado el usuario
      if (res && res.token) {
        // Se almacena el token del usuario en la memoria local del navegador
        localStorage.setItem(environment.userStorageKey, JSON.stringify({ username, token: res.token }));
        return true;
      }
      return false;
    }),
    catchError((error, caught) => of(false))
  );
}
```

Figura 33: Obtención y almacenamiento del token de identificación.

Como podemos ver en la Figura 33, se realiza una petición al servidor enviándole el usuario y contraseña introducidos en la página de acceso. Esta petición llega a un

servicio publicado por el TFG complementario a este el cual se encarga de encriptar la contraseña, compararla con la contraseña encriptada almacenada en base de datos y, siempre que las credenciales sean válidas, generar el token de autentificación. Finalmente devuelve el token generado o un error de autentificación.

Una vez se recibe la respuesta del servidor se comprueba si incluye el token de usuario y, en ese caso, lo almacena en un espacio de la memoria del navegador que pueden utilizar las aplicaciones web llamado *LocalStorage*. Al almacenarlo en este espacio el token no se elimina al hacer un refresco de la página o cerrar el navegador, esto implica que la sesión del usuario se mantendrá activa a pesar de que se abandone la página evitando tener que estar constantemente introduciendo las credenciales.

Un aspecto importante que tener en cuenta al utilizar los tokens JWT es que están encriptados de tal forma que en caso de ser interceptados no se pueda obtener información del usuario. Es por esto por lo que, junto con el token, se almacena el nombre de usuario de manera que se pueda saber a quién pertenece el token. Esta información es necesaria al realizar ciertas acciones en la aplicación como responder a un comentario.

6.5.2 Inclusión del token en las peticiones

Una vez se dispone de un token, todas las peticiones que se hagan al servidor deben incluirlo para que este pueda confirmar que proceden de un usuario autentificado. Para este proceso se ha hecho uso de la interfaz *HttpInterceptor* provista por Angular. Esta interfaz captura todas las peticiones a direcciones externas a la aplicación y permite realizar modificaciones, como en este caso la inclusión del token.

El interceptor que se ha desarrollado implementa el método “intercept” que es llamado en cada petición que se realiza al servidor. En este método se comprueba que exista un token almacenado en la memoria del navegador y en ese caso lo inserta en la cabecera de la petición. Es importante destacar que si no se encuentra el token no se lanza ninguna excepción ya que las peticiones de acceso y registro a la aplicación se pueden realizar sin haber iniciado sesión previamente.

El interceptor no es útil únicamente para incluir el token, también se puede utilizar para gestionar los errores devueltos por el servidor. Para ello se ha implementado una función que captura las respuestas de error que vengan del servidor y las muestra al usuario en una banda que aparece en la parte inferior de la pantalla. Además, si el error se produce porque el token no es válido (Error 403) redirige a la página de acceso a la aplicación para que vuelva a introducir los datos de inicio de sesión y generar un nuevo token.

7 PRUEBAS

Un paso clave en el ciclo de vida de todo proyecto es la realización de pruebas. En esta sección se explican las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación.

La realización de pruebas es un proceso que, si no se organiza con antelación, puede resultar caótico y consumir gran parte del tiempo y esfuerzo del desarrollo de un proyecto. Es por esto que se ha dividido el proceso de pruebas de este proyecto en tres fases:

1. Realización del plan de pruebas
2. Ejecución de las pruebas
3. Corrección de errores

7.1 PLAN DE PRUEBAS

Un plan de pruebas es un documento que recoge el conjunto de pruebas que se deben realizar sobre la aplicación. Estos documentos deben ser capaces de cubrir los diferentes casos de uso de modo que se maximice la probabilidad de identificar fallos en la aplicación, siempre teniendo en cuenta que se dispone de un tiempo limitado por la fecha límite de entrega. Se puede encontrar el plan de pruebas realizado en el anexo de este documento.

Para la confección del plan de pruebas se ha partido de la toma de requisitos realizada en la sección **¡Error! No se encuentra el origen de la referencia.** ya que en ella se especifican las funcionalidades que debe tener la aplicación. Cada una de las pruebas que se ha realizado consta de 4 secciones:

- **Objetivo:** Define cuál es el objetivo de la prueba a realizar.
- **Requisitos previos:** Especifica los requerimientos que se deben cumplir para poder realizar la prueba.
- **Pasos de reproducción:** Se enumeran los pasos que se deben seguir para completar la prueba.
- **Criterio de éxito:** Define el comportamiento esperado de la aplicación para considerar la prueba como exitosa.

A continuación, se muestra un ejemplo de prueba en el que se comprueba la funcionalidad de editar reto:

Objetivo:	Editar un reto publicado.
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • El usuario debe tener un reto publicado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Acceder a la página principal. 3. Pulsar sobre un reto publicado por el usuario de la sesión. 4. En el detalle del reto, pulsar en el botón editar de la parte superior derecha de la página. 5. Realizar alguna modificación en la descripción del reto. 6. Pulsar el botón guardar situado en la esquina inferior

	derecha de la pantalla.
Criterio de éxito:	Tras realizar los pasos, se carga la pantalla de detalle del reto donde se observan las modificaciones realizadas.

7.2 EJECUCIÓN

Una vez completado el plan de pruebas se procede a la ejecución de estas. Este proceso consiste en seguir los pasos indicados en la descripción de cada prueba y comprobar que el resultado obtenido es el esperado, en caso contrario se debe identificar qué es lo que falla para poder resolverlo posteriormente.

Si el plan de pruebas está correctamente formulado, la ejecución de las pruebas puede ser realizada por cualquier persona independientemente de sus conocimientos informáticos. Es incluso recomendable que las pruebas las realicen personas ajenas al desarrollo ya que estas pruebas deben ser pruebas de caja negra, es decir, quien las ejecuta no conoce el funcionamiento del sistema y únicamente provee datos a la aplicación y comprueba las respuestas obtenidas. Esto es así porque el conocimiento de la implementación de la aplicación puede influir en las acciones realizadas para completar las pruebas. Por este motivo la ejecución del plan de pruebas la hemos realizado entre mi compañera Beatriz Benedicto, quien ha implementado la API REST a la que se conecta la aplicación, y yo.

Para la ejecución de las pruebas se ha tenido en cuenta que la aplicación debe ser funcional tanto desde ordenadores como dispositivos móviles. Para ello se ha realizado cada una de las pruebas al menos una vez en cada tipo de dispositivo, de esta manera se pretende asegurar el correcto funcionamiento de la aplicación independientemente de dónde se esté haciendo uso de ella.

7.3 CORRECCIÓN DE ERRORES

Tras la ejecución del plan de pruebas se ha recopilado una lista con los errores que han ido surgiendo al utilizar la aplicación. La creación de esta lista permite priorizar la corrección de aquellos errores más graves que afectan en gran manera a la experiencia del usuario.

Gran parte de los errores surgidos en este proceso estaban relacionados con los estilos de la aplicación, concretamente por los siguientes motivos:

- **Navegador utilizado:** Realizando las pruebas se ha detectado que muchos de los efectos y funcionalidades de los componentes no funcionaban correctamente en el navegador Internet Explorer. Esto es debido a que el motor de renderizado de este navegador no incluye muchas de las funcionalidades que utiliza Angular. Para solucionar estos errores se han incluido unos scripts llamados *polyfills* que permiten agregar aquellas características de las que no dispone el navegador.

- **Aplicación adaptable:** El que la aplicación pueda adaptarse a diferentes tamaños de pantalla ha provocado en múltiples ocasiones que textos o botones queden fuera de la pantalla o que el contenido sea *scrollable* horizontalmente lo que resulta poco intuitivo cuando se está haciendo uso de la aplicación en un dispositivo móvil. Para solucionar estos problemas se han creado reglas en los estilos de la aplicación que cambian la distribución de los elementos de la pantalla según el tamaño del dispositivo que se esté utilizando.
- **Internacionalización:** Este proceso permite mostrar el texto en diferentes idiomas lo que provoca que los textos que muestra la aplicación varíen en longitud. Debido a esto, en ocasiones textos más largos no cabían en el espacio asignado o se desalineaban con el resto de contenido provocando un efecto visual de desorden. Estos problemas han sido más sencillos de solucionar ya que consistían en aumentar los espacios o corregir la alineación de los elementos.

Además de los errores visuales también se han corregido errores en la lógica de la aplicación, sin embargo, estos son más específicos de cada componente o servicio. Algunos de los errores recurrentes en la lógica de la aplicación consistían en los siguientes aspectos:

- **Condiciones de visualización:** Ciertos elementos de las páginas están vinculados a una lógica que decide si se deben mostrar o no dependiendo de condiciones como el perfil del usuario, la cantidad de comentarios u otros atributos. En determinados casos no se tenían en cuenta todos los factores lo que provocaba inconsistencias en la visualización.
- **Atributos opcionales:** En ocasiones, hay atributos de los objetos que se reciben al realizar peticiones al servidor que son opcionales y pueden venir vacíos. Si no se tiene en cuenta esto se producen errores de acceso a valores nulos que provocan un mal funcionamiento de la aplicación. Para solucionar dichos errores se evalúa que el atributo que se pretende utilizar no sea nulo.

Tras corregir los errores que han surgido al realizar una prueba, se vuelve a ejecutar la prueba en cuestión para comprobar que no se reproduce de nuevo el error.

8 CONCLUSIONES

En este capítulo se plasman las conclusiones obtenidas tras el desarrollo de la aplicación. Estas conclusiones se van a comentar en base a los objetivos marcados en el capítulo 1.2, analizando el cumplimiento de cada uno y los problemas que se han debido de afrontar.

En primer lugar, cabe destacar que todos los objetivos marcados al inicio del documento se han completado satisfactoriamente, dando lugar a una aplicación capaz de comunicarse con la API proveedora de servicios de una manera segura.

El primer objetivo corresponde a la **creación del proyecto base**. Para esta tarea, un elemento clave ha sido la elección del framework Angular y más concretamente su herramienta Angular CLI. La elección venía condicionada por un conocimiento previo básico del framework, no obstante gracias a este proyecto he podido adentrarme en la utilización de la herramienta Angular CLI, la cual me ha sorprendido por su gran versatilidad y la agilidad que proporciona si se hace un uso adecuado de ella. Además, durante el desarrollo se han intentado seguir estándares y buenas prácticas de codificación, lo que me ha permitido afianzar mis conocimientos sobre el framework de una manera más correcta desde el punto de vista técnico.

El segundo objetivo marcado era la **implementación de servicios** que realicen las peticiones REST a la API publicada por el TFG complementario a este. Realizar esta integración ha sido un proceso extenso ya que la cantidad servicios de los que hace uso la aplicación ha sido mucho mayor a lo previsto inicialmente. En el desarrollo de estos se ha ocasionado el problema más destacable del desarrollo, la implementación de la paginación. Este sistema para aligerar las peticiones que debe procesar la aplicación ha supuesto un gran esfuerzo ya que, por un lado se han tenido que modelar los diferentes elementos que requieren las peticiones de este tipo, y por otro se ha tenido que profundizar en el uso del servicio de realización de peticiones que ofrece Angular para parametrizar correctamente todas las peticiones que requieren paginación.

Como tercer objetivo se encontraba la **implementación de la seguridad**. Este objetivo suponía a priori el mayor reto a afrontar debido a que no tenía conocimientos en el ámbito de la segurida. Sin embargo, aprender sobre este tema me ha resultado de gran interés y la implementación que se ha realizado en este proyecto me ha permitido poner en práctica todos los conocimientos adquiridos.

El cuarto objetivo consistía en **internacionalizar la aplicación**. En una primera aproximación se ha intentado realizar esta tarea haciendo uso de la herramienta por defecto que trae Angular, sin embargo, tras enfrentarse a algunas dificultades de implementación se optó por analizar otras alternativas. Para ello se barajaron varias opciones y se hicieron pruebas con cada una de ellas, un proceso que me sirvió para familiarizarme con el gestor de dependencias *npm* y la importación de módulos externos en un proyecto Angular.

Uno de los objetivos con los que más he disfrutado ha sido el de **embeber un editor de código** en la aplicación pese a que su implementación ha sido la que más necesidades requería. Los pasos que se han seguido para su desarrollo han sido:

1. Primero se han tenido que analizar opciones para el editor.
2. El siguiente paso ha sido embeberlo en una página de la aplicación manteniendo el formato adaptable para que funcionase en cualquier tamaño de pantalla.

3. Una vez se disponía del editor de código, se han analizado varias opciones para la compilación del código.
4. Finalmente, se ha tenido que implementar un servicio para realizar las peticiones web a una API de otra plataforma que permite compilar y ejecutar el código que se le envía.

Como último objetivo se proponía **realizar la aplicación de manera adaptativa** de forma que se pueda utilizar tanto en dispositivos móviles como en ordenadores. Este ha sido quizá uno de los más tediosos de implementar ya que me ha obligado incluso a reestructurar páginas enteras como es el caso del acceso a la aplicación. Para esta tarea se ha decidido usar el sistema *flexbox* introducido en *CSS3* el cual ha sido de gran ayuda, sin embargo, este sistema ofrece un gran abanico de posibilidades que no se ha podido dominar durante el desarrollo de este proyecto y que posiblemente hubiesen facilitado todavía más el cumplimiento de este objetivo.

Un apunte que se debe realizar es que, pese a haber completado satisfactoriamente todos los objetivos, no se han podido implementar todos los requisitos comentados en la sección **¡Error! No se encuentra el origen de la referencia.**. Debido a la falta de tiempo para implementar la solución se ha decidido priorizar todo lo referente al uso de la aplicación por parte de un alumno, dejando para posteriores ampliaciones las acciones que pueden realizar en exclusiva los profesores como lo son publicar una competición o verificar un comentario.

8.1 Relación del trabajo desarrollado con los estudios cursados

Este trabajo culmina mis estudios del Grado en Ingeniería Informática, gracias al cual he adquirido múltiples conocimientos que me han permitido afrontar este proyecto.

En primer lugar, cabe destacar que este es un proyecto de desarrollo software, lo que implica que se ha debido de programar por completo la aplicación. Para esto, los conocimientos conseguidos durante estos años han sido fundamentales permitiéndome aplicar las técnicas de programación aprendidas a pesar de que el lenguaje de programación utilizado (Typescript) no se imparte en ninguna asignatura de la titulación.

Concretando en asignaturas, gracias a “Interfaces Persona Computador” he podido aprender a crear interfaces que resulten agradables e intuitivas consiguiendo un resultado profesional. Además, para la comunicación con el servidor han sido de gran ayuda los conocimientos adquiridos en “Tecnología de Sistemas de Información en la Red”. También debo destacar la asignatura de “Gestión de Proyectos” que me ha brindado la base de conocimientos necesaria para la realización de las tareas de análisis, diseño y planificación del trabajo.

Finalmente, las influencias más destacables para la realización de este proyecto provienen de “Ingeniería de Software” y las prácticas en empresa. En la universidad he podido aprender los métodos, técnicas y herramientas actuales para el desarrollo de software de calidad. Durante las prácticas en empresa he podido ver de primera mano cómo se aplican todos estos conocimientos para resolver problemas del mundo real.

8.2 Trabajos futuros

A continuación se presentan las posibles correcciones y mejoras que se podrían aplicar a este proyecto en versiones posteriores.

La primera línea de desarrollo que se debería tomar es finalizar la implementación de aquellos requisitos que no se han podido realizar, especialmente la creación de competiciones. Este tipo de retos se espera que sean el principal atractivo de la aplicación debido a la rivalidad que va a generar en los alumnos para superarse entre sí. Para este desarrollo se puede seguir el análisis y diseño realizado en este documento lo que supone un punto de partida que reduce los costes de implementación.

Una vez finalizada una versión con todas las funcionalidades presentadas en este trabajo implementadas, existen varias mejoras que se podrían aplicar.

- Una primera mejora que se plantea es la creación de un pequeño tutorial para el primer acceso a la aplicación. Aunque la aplicación se ha diseñado para ser lo más intuitiva posible, este tutorial podría ayudar a los usuarios recién incorporados a la plataforma a entender su funcionamiento.
- Otra mejora interesante es la integración del sistema de seguridad de la universidad en la aplicación. Esto evitaría a los alumnos la necesidad de registrarse en esta plataforma ya que accederían a ella con su cuenta de la universidad.
- También se quiere añadir un sistema de moderación que permita a los profesores gestionar los comentarios que se realizan en los retos ya que, como los usuarios pueden comentar libremente en los retos, podrían aparecer malas conductas por parte de algunos alumnos. Dado que el objetivo de esta plataforma es crear un espacio donde los alumnos compartan sus soluciones y se apoyen entre ellos, se debe intentar procurar crear un ambiente que motive a los alumnos a ayudarse evitando cualquier tipo de abuso o menosprecio.
- Finalmente, esta aplicación está enfocada completamente al ámbito de la programación, pero haciendo uso de ella se ha visto que el potencial del sistema de retos puede ser aplicable en otros grados. Los retos no tienen por qué ser siempre de programación, pueden consistir en preguntas abiertas, resolución de problemas matemáticos o de cualquier otro tipo dependiendo de la escuela donde se utilice. Es por esto por lo que se podrían adaptar versiones de la aplicación a las diferentes facultades de la universidad.

9 REFERENCIAS

Angular Blogs. (2018). *Schematics - An Introduction*. Recuperado el Julio de 2018, de https://blog.angular.io/schematics-an-introduction-dc1dfbc2a2b2

BARZANALLANA, R. (2012). Historia del desarrollo de aplicaciones Web.

CABERO, J. (2006). Bases pedagógicas del e-learning. *3*(1).

Class Central. (2016). *Class Central's Top 50 MOOCs of All Time*. Recuperado el Julio de 2018, de https://www.class-central.com/report/top-moocs/

CORTIZO PÉREZ, J. C. (2011). Gamificación y Docencia: Lo que la Universidad tiene que aprender de los Videojuegos.

GUSTAFSON, A. (2017). *A list apart*. Recuperado el Julio de 2018, de Yes, That Web Project Should Be a PWA: https://alistapart.com/article/yes-that-web-project-should-be-a-pwa

IEEE. (1998). *IEEE Std 830-1998*. Recuperado el Agosto de 2018, de Recommended Practice for Software Requirements Specifications: https://ieeexplore.ieee.org/document/720574/

MELNIK, I. (2018). *Single Page Application (SPA) vs Multi Page Application*. Recuperado el Julio de 2018, de http://merehead.com/blog/single-page-application-vs-multi-page-application/

StackOverflow. (2018). *Developer Survey Results 2018*. Recuperado el Julio de 2018, de Most popular technologies: Frameworks, Libraries and Tools: https://insights.stackoverflow.com/survey/2018/#technology-frameworks-libraries-and-tools

TURNER, J. (2015). *Microsoft Blogs*. Recuperado el Julio de 2018, de Angular 2: Built on TypeScript: https://blogs.msdn.microsoft.com/typescript/2015/03/05/angular-2-built-on-typescript/

10 ANEXO

10.1 PLAN DE PRUEBAS DE LA APLICACIÓN

Objetivo:	Registrar un usuario
Requisitos previos:	-
Pasos de reproducción:	1. Acceder a la pantalla de acceso de la aplicación. 2. Pulsar sobre el enlace que indica "¿Todavía no estás registrado? Accede aquí para crear tu cuenta". 3. Rellenar los datos del formulario. 4. Pulsar el botón "Crear".
Criterio de éxito:	El usuario accede a la aplicación con el usuario creado.

Objetivo:	Iniciar sesión
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Acceder a la pantalla de acceso de la aplicación. 2. Rellenar los campos de usuario y contraseña. 3. Pulsar el botón "Acceder".
Criterio de éxito:	El usuario accede a la aplicación con el usuario creado.

Objetivo:	Filtrar retos por tipo
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Acceder a la página principal y comprobar que existen retos de diferentes tipos (se indica el tipo en el color del marcador de cada reto). 3. En el selector superior derecho con el texto "Filtrar" seleccionar "Por alumnos". 4. Comprobar que únicamente aparecen retos de alumnos marcados en verde. 5. Cambiar el selector a la opción "Por profesores". 6. Comprobar que únicamente aparecen retos de profesores marcados en azul. 7. Cambiar el selector a la opción "Competiciones". 8. Comprobar que únicamente aparecen competiciones marcadas en rojo.
Criterio de éxito:	Se aplican los diferentes filtros correctamente.

Objetivo:	Filtrar retos por texto
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Acceder a la página principal. 3. En el campo de texto superior que indica “Buscar”, escribir un término de búsqueda. 4. Pulsar el botón con forma de lupa junto al campo de texto.
Criterio de éxito:	Se muestra un listado de retos filtrados por el texto indicado en el campo de texto.

Objetivo:	Visualizar el detalle de un reto
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Pulsar sobre un reto de la página principal.
Criterio de éxito:	Se visualiza la pantalla con el detalle del reto.

Objetivo:	Crear un reto
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	3. Iniciar sesión en la aplicación. 4. Pulsar sobre el botón “+” de la esquina inferior derecha. 5. Rellenar el título y la descripción del reto. 6. Pulsar sobre el botón “Vista previa”. 7. Pulsar sobre e botón de guardar en la esquina inferior derecha.
Criterio de éxito:	La vista previa del reto se muestra correctamente y tras guardar el reto se puede visualizar en la pantalla de detalle del reto.

Objetivo:	Editar un reto publicado.
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • El usuario debe tener un reto publicado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Acceder a la página principal. 3. Pulsar sobre un reto publicado por el usuario de la sesión. 4. En el detalle del reto, pulsar en el botón editar de la parte superior derecha de la página. 5. Realizar alguna modificación en la descripción del reto. 6. Pulsar el botón guardar situado en la esquina inferior derecha de la pantalla.
Criterio de éxito:	Tras realizar los pasos, se carga la pantalla de detalle del reto donde se observan las modificaciones realizadas.

Objetivo:	Mostrar comentarios de un reto
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • Disponer de un reto con comentarios.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Pulsar sobre un reto que contenga comentarios (se puede observar el número de comentarios del reto junto al icono que muestra un diálogo). 3. Pulsar sobre el botón “Comentarios” debajo del panel con la descripción del reto.
Criterio de éxito:	Se muestra un listado con los comentarios del reto.

Objetivo:	Responder a un reto.
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Pulsar sobre un reto. 3. Si el reto contiene comentarios pulsar sobre el botón “Comentarios”. 4. Pulsar sobre el botón “+” en la esquina inferior derecha. 5. Escribir el texto del cometario. 6. Pulsar sobre el botón “Publicar”.
Criterio de éxito:	Se muestra el comentario creado en el listado de comentarios del reto.

Objetivo:	Adjuntar código fuente a comentario
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • Disponer de códigos en el espacio personal del usuario.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Pulsar sobre un reto. 3. Si el reto contiene comentarios pulsar sobre el botón “Comentarios”. 4. Pulsar sobre el botón “+” en la esquina inferior derecha. 5. Escribir el texto del cometario. 6. Pulsar sobre el botón “<>”. 7. Seleccionar un código del listado mostrado. 8. Pulsar sobre el botón “Publicar”.
Criterio de éxito:	Se muestra el comentario creado con el código adjunto en el listado de comentarios del reto.

Objetivo:	Crear código fuente
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Si se ha accedido desde un dispositivo móvil pulsar sobre el botón de menú en la esquina superior izquierda. 3. Pulsar sobre la opción de menú “Mis códigos”. 4. Pulsar sobre el botón “+” en la esquina inferior derecha. 5. Introducir el título del código fuente. 6. Seleccionar el lenguaje de programación en el selector “Lenguaje”. 7. Introducir el código fuente. 8. Pulsar el botón guardar en la esquina inferior derecha.
Criterio de éxito:	Se muestra el código fuente creado en el espacio personal del usuario.

Objetivo:	Crear código fuente
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • Disponer de un código fuente en el espacio personal.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Si se ha accedido desde un dispositivo móvil pulsar sobre el botón de menú en la esquina superior izquierda. 3. Pulsar sobre la opción de menú “Mis códigos”. 4. Seleccionar un código fuente de la lista. 5. Modificar el título, el lenguaje o el código fuente. 6. Pulsar el botón guardar en la esquina inferior derecha.
Criterio de éxito:	Se muestra el código fuente modificado en el espacio personal del usuario y al acceder a su detalle contiene las modificaciones realizadas.

Objetivo:	Ejecutar código fuente
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • Disponer de un código fuente en el espacio personal.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Si se ha accedido desde un dispositivo móvil pulsar sobre el botón de menú en la esquina superior izquierda. 3. Pulsar sobre la opción de menú “Mis códigos”. 4. Seleccionar un código fuente de la lista. 5. Pulsar el botón “Run” en la parte superior derecha del reto.
Criterio de éxito:	Se muestra el resultado de la ejecución en el panel inferior que indica “Salida”.

Objetivo:	Ejecutar código fuente
Requisitos previos:	• Disponer de un usuario registrado en la aplicación. • Disponer de un código fuente en el espacio personal.
Pasos de reproducción:	6. Iniciar sesión en la aplicación. 7. Si se ha accedido desde un dispositivo móvil pulsar sobre el botón de menú en la esquina superior izquierda. 8. Pulsar sobre la opción de menú “Mis códigos”. 9. Seleccionar un código fuente de la lista. 10. Pulsar el botón “Run” en la parte superior derecha del reto.
Criterio de éxito:	Se muestra el resultado de la ejecución en el panel inferior que indica “Salida”.

Objetivo:	Cambiar el idioma de la aplicación
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	1. Iniciar sesión en la aplicación. 2. Si se ha accedido desde un dispositivo móvil pulsar sobre el botón de menú en la esquina superior izquierda. 3. Pulsar sobre la opción de menú “Mi perfil”. 4. Seleccionar un idioma en el selector “Selección de lenguaje”.
Criterio de éxito:	Se muestran los textos de la aplicación en el idioma seleccionado.

Objetivo:	Cerrar la sesión de la aplicación
Requisitos previos:	• Disponer de un usuario registrado en la aplicación.
Pasos de reproducción:	5. Iniciar sesión en la aplicación. 6. Si se ha accedido desde un dispositivo móvil pulsar sobre el botón de menú en la esquina superior izquierda. 7. Pulsar sobre la opción de menú “Mi perfil”. 8. Pulsar sobre el botón “Cerrar sesión” en la parte superior derecha.
Criterio de éxito:	Se cierra la sesión y se redirige a la pantalla de acceso a la aplicación.