UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

CONTENERIZACIÓN, INTEGRACIÓN DE SISTEMAS Y MIGRACIÓN DE APLICACIÓN WEB CON APLICACIÓN MÓVIL COMPLEMENTARIA.

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Téllez Mérida, José Antonio

Tutor/a: Ballester Merelo, Francisco José

Cotutor/a: Ramos Peinado, Germán

Cotutor/a externo: PONS PUIG, AURELIO-JOSE

CURSO ACADÉMICO: 2022/2023

Resumen

Este trabajo final de grado aborda el desarrollo de una aplicación web completa, su implementación dentro de un contenedor, y su posterior integración dentro del proyecto BEETool del Grupo TecEner de la universidad CEU. El proyecto aborda el desarrollo de una herramienta computacional con la que se puede evaluar y predecir la forma en la que se comportan distintos materiales utilizados en la construcción de las palas de las turbinas eólicas, con el objetivo de mejorar la eficiencia en la producción de energía y minimizar los costes de mantenimientos.

Se trata de un TFG colaborativo en el que cada uno de los dos miembros se ha encargado de algunas de sus partes: aplicación web y contenerización e integración.

Este trabajo se centra en la contenerización de la aplicación web desarrollada en la otra parte del TFG colaborativo, las comunicaciones de la aplicación vía API (Application Programming Interface) que permiten llevar a cabo las simulaciones, y la posterior integración en servidores Linux. Además, se ha desarrollado una aplicación móvil para visualizar algunos de los datos.

Para la contenerización se ha empleado la tecnología Docker. Esto ha permitido llevar a cabo el desarrollo sin necesidad del uso de un servidor específico. Se ha desarrollado una API para recibir y gestionar la información introducida en la aplicación web, y así facilitar su uso. Igualmente, se ha creado otra API para enviar los informes generados por los cálculos de la herramienta de simulación, permitiendo así la correcta visualización de los resultados. Finalmente, se ha desarrollado también una aplicación móvil que permite al cliente visualizar los campos de la herramienta.

Para el desarrollo de este trabajo se han empleado multitud de lenguajes de programación en función de las necesidades como PHP, R, R Markdown y JavaScript. También se ha realizado uso de diferentes frameworks como RedBean y Slim. Para el desarrollo de las APIs se ha empleado la herramienta específica Postman. Por último, todo el trabajo colaborativo realizado se trasladó a un servidor Linux real, alojando el contenedor Docker con los archivos necesarios.

Esta aplicación web proporciona al proyecto BEETool una herramienta global en la que poder predecir los resultados de eficiencia y mantenimiento de palas de turbinas eólicas. Los

resultados finales son totalmente operativos y están siendo ya ofertados a empresas del sector energético.

Palabras clave: BEETool, herramienta computacional, Docker, API, aplicación móvil, servidor Linux.

Resum

Aquest treball final de grau aborda el desenvolupament d'una aplicació web completa, la seua implementació dins d'un contenidor, i la seua posterior integració dins del projecte BEETool del Grup TecEner de la universitat CEU. El projecte aborda el desenvolupament d'una eina computacional amb la qual es pot avaluar i predir la forma en la qual es comporten diferents materials utilitzats en la construcció de les pales de les turbines eòliques, amb l'objectiu de millorar l'eficiència en la producció d'energia i minimitzar els costos de manteniments.

Es tracta d'un TFG col·laboratiu en el qual cadascun dels dos membres s'ha encarregat d'algunes de les seues parts: aplicació web, conteneització i integració.

Aquest treball se centra en la conteneització de l'aplicació web desenvolupada en l'altra part del TFG col·laboratiu, les comunicacions de l'aplicació via API que permeten dur a terme les simulacions, i la posterior integració en servidors Linux. A més, s'ha desenvolupat una aplicació mòbil per a visualitzar alguns de les dades.

Per a la conteneització s'ha emprat la tecnologia Docker. Això ha permés dur a terme el desenvolupament sense necessitat de l'ús d'un servidor específic. S'ha desenvolupat una API per a rebre i gestionar la informació introduïda en l'aplicació web, i així facilitar el seu ús. Igualment, s'ha creat una altra API per a enviar els informes generats pels càlculs de l'eina de simulació, permetent així la correcta visualització dels resultats. Finalment, s'ha desenvolupat també una aplicació mòbil que permet al client visualitzar els camps de l'eina.

Per al desenvolupament d'aquest treball s'han emprat multitud de llenguatges de programació en funció de les necessitats com PHP, R, R Markdown i JavaScript. També s'ha realitzat ús de diferents frameworks com RedBean i Slim. Per al desenvolupament de les APIs s'ha emprat l'eina específica Postman. Finalment, tot el treball col·laboratiu realitzat es va traslladar a un servidor Linux real, allotjant el contenidor Docker amb els arxius necessaris.

Aquesta aplicació web proporciona al projecte BEETool una eina global en la qual poder predir els resultats d'eficiència i manteniment de pales de turbines eòliques. Els resultats finals són totalment operatius i estan sent ja oferits a empreses del sector energètic.

Paraules clau: BEETool, eina computacional, Docker, API, aplicació mòbil, servidor Linux

Abstract

This final degree project deals with the development of a complete web application, its implementation within a container, and its subsequent integration into the BEETool project of the TecEner Group of the CEU University. The project addresses the development of a computational tool with which the behavior of different materials used in the construction of wind turbine blades can be evaluated and predicted, with the aim of improving energy production efficiency and minimizing maintenance costs.

This is a collaborative final degree project in which each of the two members has been responsible for some of its parts: web application, containerization and integration.

This work focuses on the containerization of the web application developed in the other part of the collaborative final degree project, the application communications via API that allows simulations to be carried out, and the subsequent integration into Linux servers. In addition, a mobile application has been developed to visualize some of the data.

Docker technology has been used for containerization. This has made it possible to carry out development without the use of a specific server. An API has been developed to receive and manage the information entered in the web application, thus facilitating its use. Likewise, another API has been created to send the reports generated by the simulation tool calculations, allowing the correct visualization of the results. Finally, a mobile application has also been developed that allows the client to visualize the fields of the tool.

For the development of this work, a multitude of programming languages have been used depending on the needs, such as PHP, R, R Markdown, and JavaScript. It has also made use of different frameworks such as RedBean and Slim have also been used. The Postman specific tool has been used for the development of APIs. Finally, all the collaborative work done was moved to a real Linux server, hosting the Docker container with the necessary files

This web application provides the BEETool project with a global tool for predicting the efficiency and maintenance results of wind turbine blades. The final results are fully operational and are already being offered to companies in the energy sector.

Keywords: BEETool, computational tool, Docker, API, mobile application, Linux server.

A mi familia por su amor incondicional y su apoyo constante

A mi tutor Germán, cuya paciencia, conocimientos y dedicación han sido invaluables. Agradezco su guía constante, su disposición para resolver mis dudas y su motivación para superar los desafíos.

A José, quien ha sido un pilar fundamental en este viaje académico. Su ayuda, colaboración y apoyo mutuo han sido clave para alcanzar nuestros objetivos y superar las dificultades.

A Aurelio, por otorgarme la oportunidad de llevar a cabo este proyecto.

A todos ellos, les estoy sinceramente agradecido. Sus contribuciones y presencia en mi vida han dejado una huella imborrable, y este trabajo no habría sido posible sin su inestimable ayuda.

Índice general

Índice de figuras

Índice de extractos de código

Listado de siglas empleadas

API Application Programming Interfaces. 20

AVI Agencia Valenciana de la Innovación. 8

BEETool Bleed Erosion Evaluation Tool. 8

CEO Chief Executive Officer. 14

CEU Universidad CEU Cardenal Herrera. 8

FEDER Fondo Europeo de Desarrollo Regional. 8

GNU GNU's Not Unix. 18

HTTP HyperText Transfer Protocol. 23

IDIT Instituto de Innovación, Diseño y Tecnología. 13

IoT Internet de las cosas. 62

IP Internet Protocol. 20

JS JavaScript. 52

JSON JavaScript Object Notation. 22

ODS Objetivos de Desarrollo Sostenible. 8

ORM Object-Relational Mapping. 33

PDF Portable Document Format. 44

PNG Portable Network Graphics. 32

REST REpresentational State Transfer. 20

SSH Secure SHEll. 24

TecEner Investigación y Desarrollo de Tecnologías en Aplicaciones Energéticas. 8

TFG Trabajo de Fin de Grado. 3

UCH Fundación San Pablo-CEU. 8

UE Union Europea. 8

URL Uniform Resource Locators. 32

WSL Windows Subsystem for Linux. 18

YAML Yet Another Markup Language. 26

Parte I

Introducción

1. Introducción

1.1. Motivación

A día de hoy, el proceso de análisis de erosión de las palas de las turbinas eólicas es un serio problema. Este es un proceso costoso tanto en el ámbito económico como temporal debido a que estos generadores eólicos deben estar expuestos a condiciones atmosféricas adversas para poner a prueba la resistencia de sus materiales y sus diseños.

Actualmente, existen dos métodos para realizar este análisis. Por una parte, observando su deterioro en una situación real, lo que implica un elevado coste y la espera de un largo periodo de tiempo para que se produzcan todas las situaciones climáticas posibles. Por otra parte, se puede optar por otra alternativa, la cual permite introducir las palas en un banco de pruebas[1], simulando de esta forma las condiciones reales. Se puede observar el banco en la figura 1.1. Este proceso permite observar dicho deterioro sin la necesidad de esperar a que ocurra de manera natural, acelerando todo el proceso y reduciendo costes totales.

Figura 1.1: Banco de pruebas [1].

Sin embargo, este método puede acabar resultando más complejo que la observación del suceso en situaciones reales. Y por lo tanto, debido a esta situación, los operadores y fabricantes de equipos de turbinas eólicas están buscando soluciones innovadoras, puesto que actualmente no existen metodologías específicas que tengan la suficiente validez como

[1]En el banco de pruebas, una pieza del material de las palas que se va a probar se fija y se hace girar mientras la máquina simula la lluvia, exponiendo la pieza a cargas equivalentes a 20 años en el campo [1].

para poder evaluar el rendimiento de sus equipos de manera eficiente y pudiendo predecir de manera más efectiva su desajuste y los posibles fallos [2].

1.2. Marco del proyecto

La solución al problema anteriormente mencionado es la creación de una herramienta computacional junto con una aplicación web con la que se puede evaluar y predecir la forma en la que se comportan distintos materiales utilizados en la construcción de las palas de las turbinas eólicas, con el objetivo de mejorar la eficiencia en la producción de energía y minimizar los costes de mantenimiento [2]. Además, esta alternativa cuenta con una base de datos con la capacidad de almacenar resultados obtenidos del deterioro de las palas en un entorno real o en los bancos de pruebas. Con esto, el cliente tiene la posibilidad de generar informes, contrastando de esta manera los resultados obtenidos en bancos de pruebas o en entornos reales con las evaluadas por la propia herramienta. En consecuencia, se podrá mejorar la resistencia y durabilidad de los materiales empleados en las palas de las turbinas eólicas, siendo un gran avance en el estudio de este proyecto. En último término, sería útil disponer de una aplicación móvil para visualizar los resultados de los cálculos e informes de manera más cómoda.

Para obtener un contexto completo y comprender la terminología utilizada, en el Capítulo 4 se explicará con detalle todo lo correspondiente a ello. Específicamente, se recomienda prestar atención en la subsección 4.1.1 en esta se consigue entender cómo se utiliza Docker [3] en el proyecto y cuál es su papel en el desarrollo y despliegue de las aplicaciones. Además de palabras técnicas y conceptos que se presentan más adelante en el documento.

Este TFG (Trabajo de Fin de Grado) se basa en un proyecto colaborativo dividido en dos partes. Por un lado, el desarrollo de la aplicación web. Por otro lado, el proceso de contenerización de la aplicación web y la herramienta computacional junto con sus respectivas integraciones. Y finalmente, la migración[2] del contenedor de la herramienta computacional a la máquina alojada en la nube junto con el desarrollo de una aplicación móvil con la capacidad de visualizar los parámetros e informes generados.

[2]Proceso de trasladar aplicaciones de software de un entorno informático a otro. En este caso del contenedor en local, al entorno de un proveedor de nube.

1.3. Objetivos

Como se ilustra en la figura 1.2, se pueden observar los objetivos a cumplir para el correcto funcionamiento de la aplicación y sus relaciones. Entre los objetivos se encuentra el empaquetado de la propia web dentro de un contenedor para su correcto funcionamiento en un entorno local, junto con la contenerización de la herramienta computacional. Más adelante la integración de sistemas, la gestión de usuarios y la migración de la herramienta computacional a la máquina alojada en la nube. Finalmente, el desarrollo de la aplicación móvil.

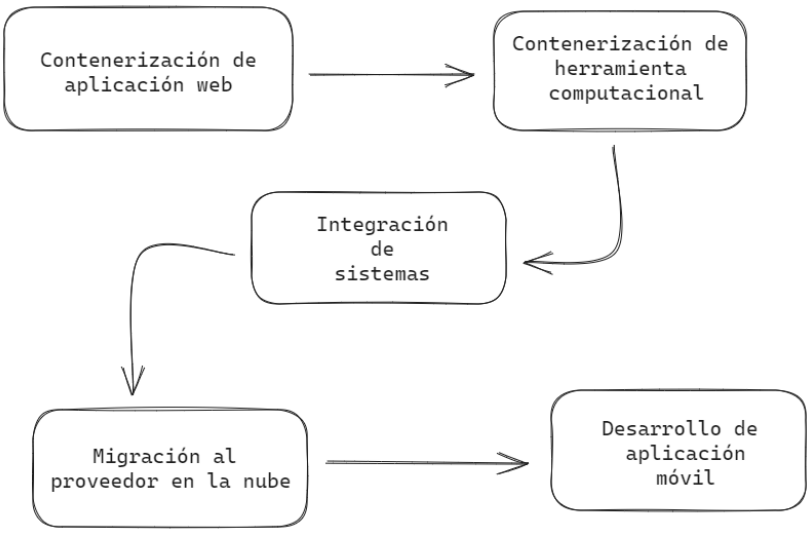

Figura 1.2: Diagrama de objetivos.

Mediante estas implementaciones se tendrá la capacidad de poder alojar la aplicación web y la herramienta computacional en los respectivos contenedores de cada una. Esto será favorable para posteriormente poder permitir la comunicación entre ellos y la realización de peticiones y respuestas. Por otra parte, permitirá la gestión de usuarios teniendo en cuenta unas funcionalidades u otras dependiendo de los permisos que se posean.

Posteriormente, se podría migrar la aplicación web contenerizada, y el propio contenedor Docker de la herramienta computacional, a la máquina alojada en la nube.

Se puede apreciar en la figura 1.3 como el funcionamiento e integraciones de la aplicación web difieren según el entorno. En el entorno local, se realiza la integración entre contenedores, mientras que en la máquina alojada en la nube, la integración se lleva a cabo entre la propia aplicación y la herramienta contenerizada. Cabe destacar que, donde anteriormente de manera local se realizaba la integración entre contenedores, ahora la integración

será entre el contenedor de la herramienta computacional y la aplicación web alojada en la máquina en la nube. Este proceso está detallado más adelante en el capítulo 7.

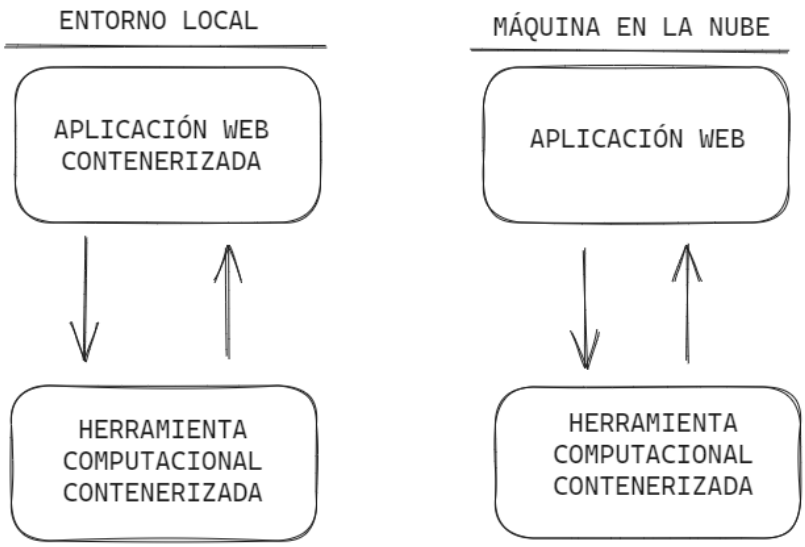

Figura 1.3: Diagrama de integraciones.

El proceso de generar los dos entornos es crucial, debido a que, cuando se realiza un proyecto de esta envergadura, se debe desarrollar e implementar de manera local para posteriormente trasladarlo a la máquina en la nube. Esto permite que cuando la aplicación web esté en producción[3] el funcionamiento sea correcto, además, de ofrecer la capacidad de poder desarrollar, implementar y modificar la web sin necesidad de hacer cambios en la aplicación web alojada en la nube.

En último lugar, y ya citado anteriormente, se propone como parte de los objetivos llevar a cabo parte del desarrollo de la aplicación móvil para la visualización y consulta de datos.

1.4. Estructura del documento

El presente trabajo cuenta con diversos apartados, los cuales tienen por objeto guiar y explicar en profundidad el proyecto. Este primer apartado está compuesto de una introducción desarrollada por la motivación del proyecto, el marco de este y los objetivos establecidos.

[3]Web que está en funcionamiento y disponible para ser utilizada por los usuarios finales. En este estado, la página web ha pasado por el desarrollo y pruebas necesarias, y se considera estable y lista para ser utilizada en un entorno de producción real.

En el segundo capítulo se realiza una descripción y planificación del proyecto, además de la metodología utilizada en el proyecto y la organización del mismo.

En el tercer capítulo se observa el desarrollo y las herramientas que se han utilizado, junto con todos los procesos realizados para el correcto funcionamiento de la aplicación web junto con la herramienta computacional.

En el cuarto capítulo se analizan las pruebas realizadas con el fin de comprobar su calidad y funcionamiento. Asimismo, los resultados obtenidos.

Por último, en el quinto capítulo, se verá las conclusiones que se ha sacado durante la elaboración del proyecto y el trabajo futuro que se podría llevar a cabo.

Parte II

Descripción y planificación del proyecto

2. Definición del proyecto

2.1. Definición del proyecto

Este proyecto BEETool (Bleed Erosion Evaluation Tool) ha sido llevado a cabo por el Grupo TecEner (Investigación y Desarrollo de Tecnologías en Aplicaciones Energéticas) de la CEU (Universidad CEU Cardenal Herrera) UCH (Fundación San Pablo-CEU) y la empresa A3P Interacción Digital contando con la participación de los alumnos José Antonio Téllez Mérida y José Lopez Conejero que han desarrollado este TFG colaborativo. Contando con la colaboración de la AVI (Agencia Valenciana de la Innovación) y cofinanciado por la UE (Unión Europea) a través del Programa Operativo FEDER (Fondo Europeo de Desarrollo Regional).

También es relevante mencionar la conexión existente entre el proyecto BEETool y los ODS (Objetivos de Desarrollo Sostenible) [4]. Estos objetivos fueron adoptados a nivel mundial en 2015 como parte de una nueva agenda para el desarrollo sostenible, con el propósito de erradicar la pobreza y preservar el planeta durante los próximos quince años. Dentro de los ODS se incluyen metas como "Garantizar el Acceso a una Energía Asequible, Segura, Sostenible y Moderna" o "Promover Ciudades y Comunidades Sostenibles". El presente TFG, en conjunto con el proyecto BEETool, impulsa la generación de energía eólica, una fuente de energía limpia que no emite gases de efecto invernadero ni produce residuos. Además, la implementación de aerogeneradores en áreas urbanas contribuye a la sostenibilidad energética de las ciudades.

Respecto al propósito del proyecto, se puede enunciar que se basa en la creación de una herramienta computacional junto con una aplicación web que haga de interfaz y permita predecir el deterioro de las palas de las turbinas eólicas con el objetivo de mejorar la resistencia, durabilidad y calidad de los materiales empleados en las palas de las turbinas eólicas [2].

El proyecto en sí consta de dos partes, en primer lugar, la creación de la interfaz web y en segundo lugar, la creación de una herramienta computacional, la cual su funcionamiento es específicamente llevar a cabo un conjunto de cálculos y distintas funciones que se ofrecen

una serie de gráficos, tablas y representaciones que permiten entender el proceso de deterioro de los distintos materiales.

La empresa A3P se ha encargado de desarrollar la aplicación web, la cual posteriormente se comunica con la herramienta computacional proporcionada por el Grupo TecEner. Gracias a la colaboración de ambas partes se pueden visualizar los resultados obtenidos por la herramienta computacional en la aplicación web, generando un informe permitiendo así observar los resultados de manera más clara y concisa.

El proceso del proyecto BEETool se puede observar en la figura 2.1, es continuo y pasa por tres etapas: la primera, la cual obtiene los materiales, estudiando sus características, posteriormente; la segunda, la cual permite el análisis y modelado; y por último, la estimación. Este proceso circular, produce que se repita constantemente el ciclo desde la primera etapa una vez finalizada la última. Cabe añadir que, la principal finalidad de esto también es poder comparar los modelos físicos[1] con las predicciones que ofrece la herramienta computacional.

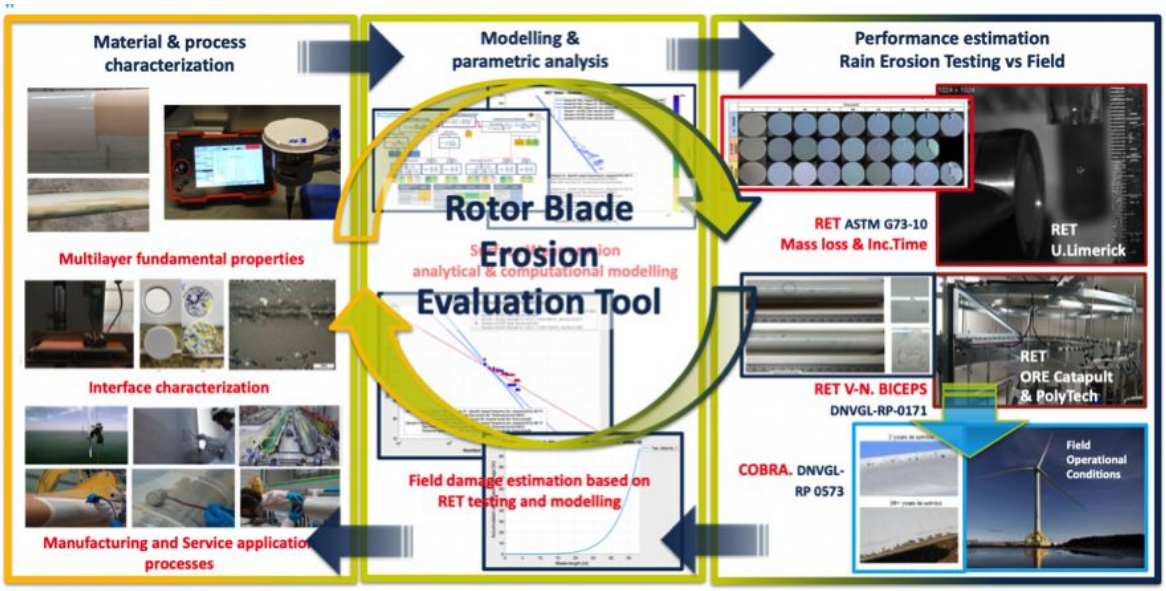

Figura 2.1: Marco tecnológico desarrollado en proyecto BEETool [2].

2.2. Alcance del proyecto

El sistema está diseñado para empresas fabricantes y operadoras de equipos de turbinas eólicas que están buscando activamente tecnologías innovadoras [2], empresas como SIE-

[1]Un modelo físico es una prueba que se ha llevado a cabo en un escenario real o utilizando los bancos de pruebas anteriormente mencionados.

MENS o LG, entre otras. El cliente objetivo[2] que conforma este proyecto son las grandes empresas, estos son los que ofrecen mantenimiento y desarrollo de turbinas eólicas.

Este sistema se encargará de dotar al cliente la capacidad de poder realizar tantas simulaciones como desee sin necesidad de gastar mucho dinero ni esperar mucho tiempo.

Además, cada cliente puede contar con su propia aplicación web alojada en su propio dominio[3] y personalizada, que le prestará la capacidad de almacenar sus propios datos y generar sus propios informes con el formato seleccionado. Paralelamente a esto, se ha desarrollado una aplicación móvil para poder visualizar sus propios informes, esta aplicación cuenta con un inicio de sesión. El desarrollo de esta aplicación móvil se detalla en el capítulo 8.

Otra prestación que se le ofrece al cliente es la gestión de permisos para que dentro de su propia empresa pueda dotar a sus usuarios de unos permisos u otros, teniendo así la capacidad de restringir el acceso a ciertas funcionalidades.

2.3. Subsistemas del proyecto

Debido a la envergadura del proyecto y que involucra diferentes tareas muy distintas, se ha dividido en subsistemas para facilitar su desarrollo. Principalmente, el proyecto se ha dividido en dos partes: la herramienta computacional, y la aplicación web, como se ha comentado previamente. Paralelamente a esto, la aplicación web también se ha dividido debido a que este es un proyecto colaborativo. Por una parte, el desarrollo de la web y por otra las siguientes implementaciones:

- Contenerización de aplicación web.
- Contenerización de herramienta computacional.
- Integración de sistemas entre contenedores.
- Desarrollo de gestión de permisos.
- Migración a servidor.
- Integración de sistemas entre contenedor de herramienta computacional y servidor.

[2] Rrecorte demográfico, socioeconómico y comportamental de un grupo compuesto por futuros consumidores del producto o servicio de la empresa [5].

[3] Nombre único que recibe un sitio web en internet. Este nombre identifica a una página web concreta sin que puedan existir dos o más sitios web que compartan el mismo nombre de dominio [6].

- Desarrollo de aplicación móvil.

Para desarrollar la explicación de manera más detallada de estos subsistemas se usará durante la explicación el diagrama de la figura 2.2, se pueden apreciar distintos colores, el verde muestra la parte desarrollada por el Grupo TecEner, la parte amarilla por el alumno José López Conejero y la azul por el alumno José Antonio Téllez. Cabe destacar que este TFG está enfocado en la parte sombreada de color azul, en consecuencia es la más desarrollada, las otras tan solo están representadas para poder apreciar el hilo del proyecto de manera más completa y contextualizada. Es más, el diagrama muestra el proceso en orden del proyecto.

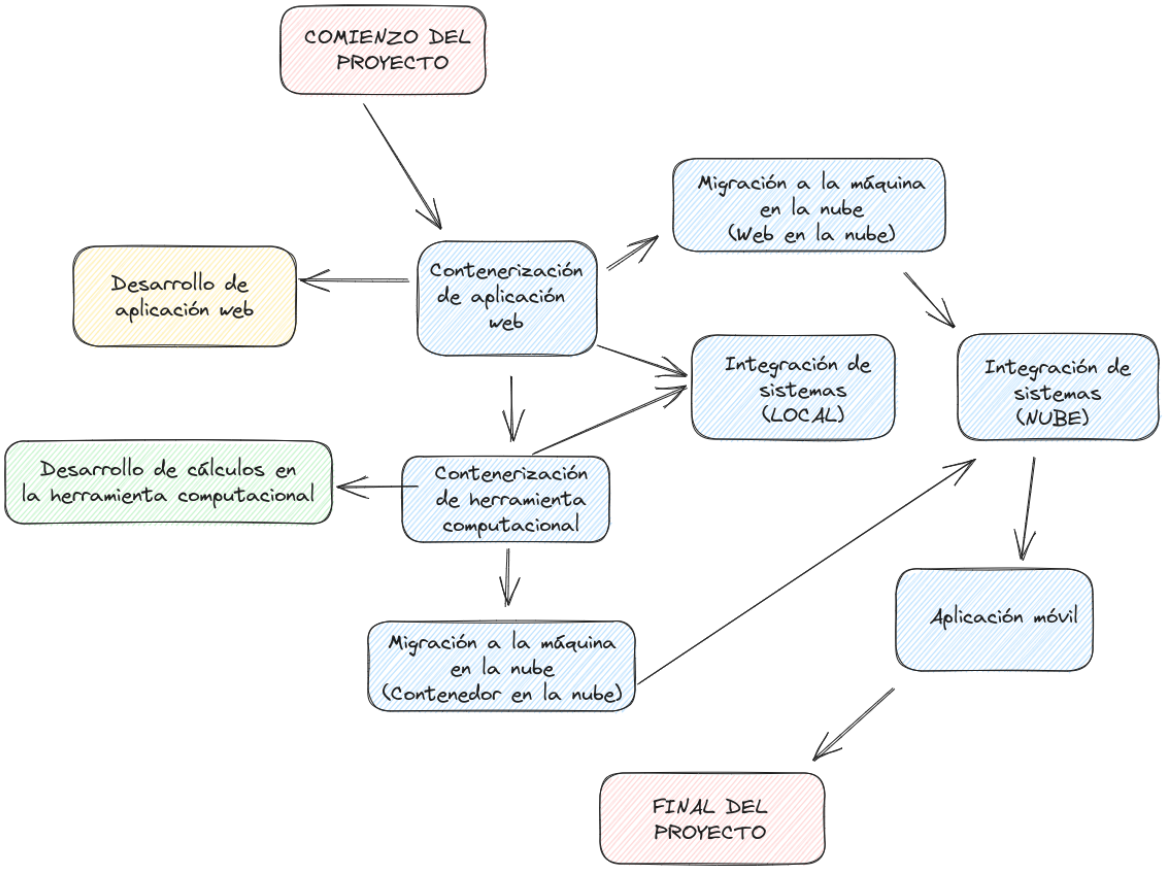

Figura 2.2: Hilo del proyecto.

3. Plan de gestión del proyecto

3.1. Metodología de desarrollo

La Real Academia Española (RAE) define la metodología como "el conjunto de métodos utilizados en una investigación científica o en una exposición doctrinal". Al comprender su significado y realizar una investigación, se puede concluir que la metodología se refiere al conjunto de estrategias, técnicas y herramientas empleadas para llevar a cabo una tarea.

En el contexto del desarrollo de software, la metodología puede entenderse como una guía que se debe seguir a lo largo de todas las etapas del proyecto (análisis, diseño, desarrollo, pruebas y mantenimiento) con el objetivo de cumplir los requisitos del cliente y entregar un producto de calidad en el menor tiempo posible. Si no se utiliza una metodología, es muy probable que el proyecto termine de forma no adecuada debido a la falta de organización entre los diferentes miembros del equipo, dependencias de tareas, errores de arquitectura o código, etc.

Entre las distintas metodologías ágiles, se ha decidido utilizar una metodología basada en objetivos. Mediante este enfoque, el desarrollo del proyecto se establece alrededor de fines específicos que deben ser alcanzados en una fecha marcada. Junto con lo dicho, se trabaja de manera colaborativa entre los miembros para poner en común los puntos establecidos previamente, que han sido conseguidos por estos participantes, y en segundo lugar, establecer nuevos objetivos y metas que promuevan la participación activa de todos los integrantes del equipo.

Paralelamente a esto, se trabaja en *pair programming*[1] cuando surgen situaciones en las que hay que llevar a cabo la toma de decisiones importantes. Respectivamente, cuando el enfoque va más desarrollado a un ámbito u otro, el rol de conductor[2] va variando.

Durante el desarrollo del proyecto, se ha utilizado Trello[3]. Esta herramienta permite

[1] La programación en pareja (pair programming en inglés) es una técnica empleada en el desarrollo ágil de software, consistente en trabajar en el mismo equipo dos programadores de forma conjunta. Uno de ellos (el conductor) escribe el código, mientras que el otro (observador) lo supervisa [7].

[2] Cuando se realiza programación en parejas, el participante encargado de escribir el código.

[3] Herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas [8].

trabajar con un tablero que posee distintas tarjetas, las cuales van pasando por distintos estados dependiendo en el momento en el que estén, como se puede observar en la figura 3.1.

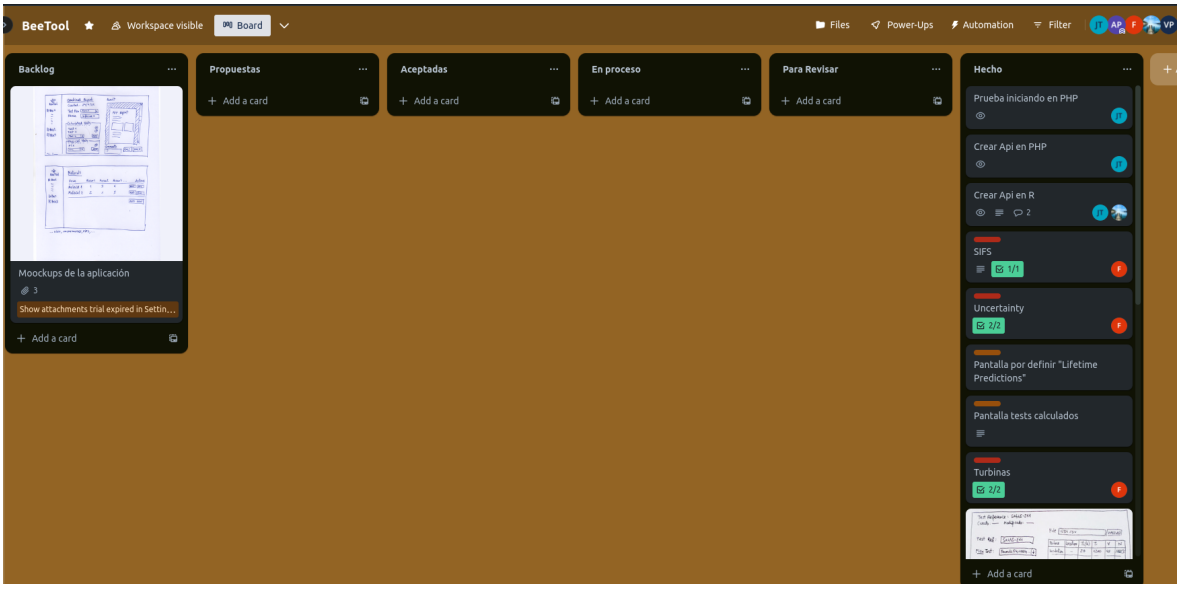

Figura 3.1: Tablero de Trello para proyecto BEETool a día 28/06/2023.

3.2. Planificación y organización del proyecto

El proyecto, al contar con la cofinanciación de la UE, estableció un conjunto de patrones específicos que debían cumplirse. La planificación se basaba en lograr los objetivos establecidos para poder recibir los pagos correspondientes y continuar avanzando con el proyecto. Se procuraba tener demostraciones preparadas para las fechas acordadas, si bien es importante destacar que estas fechas eran orientativas y dependían del progreso del proyecto. A medida que el proyecto avanzaba, se establecían patrones y pautas a seguir para garantizar su correcta evolución.

Durante el desarrollo del proyecto, el grupo TecEner, liderado por Fernando Sánchez, director del IDIT (Instituto de Innovación, Diseño y Tecnología), se enfocó en trabajar de manera colaborativa y diaria para llevar a cabo las operaciones y cálculos necesarios para el correcto funcionamiento de la herramienta computacional para de esta manera realizar predicciones precisas sobre la erosión del borde de ataque de las palas. Por otro lado, A3P llevaba a cabo reuniones diarias para garantizar el adecuado avance del proyecto respecto a la interfaz web e integración de sistemas, trabajando en parejas cuando surgían dificultades. Se utilizó la plataforma Trello para gestionar las tareas acordadas por el gerente de proyecto,

quien se encargaba de planificar, coordinar y supervisar todas las actividades relacionadas con el proyecto. En este caso, Aurelio Pons, CEO (Chief Executive Officer) de A3P Interacción Digital S.L., asumió dicha responsabilidad.

Es relevante destacar que se llevaban a cabo reuniones semanales con el objetivo de coordinar y alinear los esfuerzos de ambas entidades, asegurando así el progreso armónico del proyecto. Esta comunicación regular y la colaboración estrecha entre el grupo TecEner y A3P fueron fundamentales para cumplir con los objetivos establecidos y mantener altos estándares de calidad en el desarrollo del software.

Parte III

Desarrollo

4. Tecnologías utilizadas

4.1. Plataformas de software

Debido a que es un proyecto de desarrollo de software, se han utilizado distintas plataformas. Durante este punto se explica cada uno de los programas informáticos utilizados, su terminología y las funcionalidades que ofrece cada uno de ellos. En consecuencia, ayudará a clarificar y facilitar la comprensión de este TFG.

4.1.1. Docker

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Ésta plataforma empaqueta el programa informático en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el sistema se ponga en funcionamiento, incluidas bibliotecas, herramientas del software, código y tiempo de ejecución. Con Docker, se puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno. Permite conocer con certeza si su código se ejecutará correctamente [3]. El programa del que se habla está basado en características kernel de Linux, las cuales permiten la virtualización ligera y el aislamiento de contenedores.

Docker cuenta con una aplicación nativa llamada Docker Desktop diseñada para Windows y MAC OS, que es capaz de ejecutar, construir y enviar aplicaciones o servicios acoplados en contenedores [9]. Esta aplicación cada vez está más actualizada y ofrece una interfaz más completa, ofreciendo así la capacidad de realizar otro tipo de acciones que anteriormente solo se podían realizar por terminal de comandos.

En la figura 4.1, se puede observar el flujo de construcción de un contenedor en Docker. Este proceso contempla, en primer lugar, la creación de un fichero *Dockerfile*, en el cual se implementan los comandos pertinentes para construir la imagen deseada. A partir de aquí, se construye la imagen con todas las bibliotecas, herramientas y dependencias que se han implementado en el fichero *Dockerfile*. Cabe destacar que, una imagen es una plantilla formada por un conjunto de capas, y es una pieza fundamental para la creación del contenedor. Finalmente, se utiliza la imagen para poner en marcha el contenedor, el cual es una entidad

lógica, una agrupación de procesos que se ejecutan de forma nativa como cualquier otra aplicación en la máquina host [10].

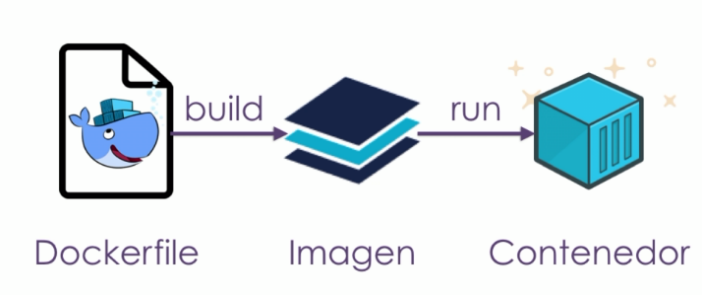

Figura 4.1: Flujo de construcción en Docker [10]

Otra característica del tiempo de ejecución de Docker son los volúmenes, estos permiten que los contenedores conserven datos desde el sistema de archivos del host [11], así como también permite el poder conectar las redes en el programa, permitiendo conectar contenedores entre sí.

Con respecto a este proyecto, Docker permite gestionar la contenerización de la herramienta computacional y de la aplicación web en contenedores aislados. Llegados a este punto, debe señalarse que se entiende como contenerización al proceso de empaquetar una aplicación junto con sus dependencias y configuraciones en un contenedor aislado y autosuficiente.

Siguiendo con todo lo dicho previamente, es importante destacar que se hace uso de los volúmenes de Docker para tener la capacidad de poder conservar los datos o incluso tener la posibilidad de leer ficheros de un contenedor a otro. Algunas de las funcionalidades y ejemplos de su uso son el permiso de poder visualizar los informes generados en la herramienta computacional desde la aplicación web o almacenar la base de datos en un volumen. Para contextualizar, se puede enunciar que en este proyecto se usan cuatro contenedores conectados entre sí y con la capacidad de comunicarse, esto se debe a que están interconectados mediante redes de Docker.

4.1.2. WSL

WSL (Windows Subsystem for Linux) es el Subsistema de Windows[1] para Linux[2] permite a los desarrolladores ejecutar un entorno de GNU (GNU's Not Unix)/Linux, incluyendo la mayoría de herramientas de línea de comandos, utilidades y aplicaciones, directamente en Windows, sin modificar y sin la sobrecarga de una máquina virtual tradicional o una configuración de arranque dual [14].

Para la contenerización de la herramienta y de la aplicación se precisó del uso de Linux, exactamente la distribución[3] Ubuntu 22.04. Windows no ofrece un rendimiento eficiente con Docker, puesto que, ya citado anteriormente, este fue originalmente desarrollado para ejecutarse en sistemas Linux y utiliza características del kernel de Linux. La manera óptima si se posee un dispositivo Windows y no tienes la posibilidad de obtener un dispositivo Linux es trabajar en un entorno WSL2.

WSL 2 es una nueva versión de la arquitectura del Subsistema de Windows para Linux que permite que se ejecuten archivos binarios de ELF64 de Linux en Windows. Sus principales objetivos son aumentar el rendimiento del sistema de archivos y agregar compatibilidad completa con las llamadas [14].

4.1.3. GitHub

Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, esta fue comprada por Microsoft. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuarios no únicamente puedan descargarse la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo [16].

Utilizar GitHub ha permitido tener un repositorio en la nube donde se aloja el código del proyecto, permitiendo así poder llevar a cabo el proyecto colaborativo. Este programa ha facilitado el control de versiones y un desarrollo cronológico del proyecto. Además, ha permitido alojar en la nube distintos commits, dotando al usuario de un aumento en la

[1] Sistema operativo desarrollado por la compañía estadounidense Microsoft [12].

[2] Nombre que reciben una serie de sistemas operativos de tipo Unix bajo la licencia GNU que son su mayoría gratuitos y con todo lo necesario para hacer funcionar un PC, con la peculiaridad de que se puede instalar un sistema muy ligero e ir añadiendo todo lo necesario posteriormente o según se vaya necesitando [13].

[3] Versión personalizada del sistema operativo original, el kernel o núcleo de Linux [15].

sencillez en cuanto al poder viajar a un commit antiguo, en el caso de que se requiera, entre muchas otras funcionalidades. Se entiende como commit a una versión del proyecto que se almacena en el repositorio local que posteriormente se puede subir al repositorio en la nube.

Básicamente, el proyecto se ha basado en el desarrollo de código, crear versiones, subirlas al repositorio en la nube y cuando otro integrante actualizaba la versión en la nube, actualizar el repositorio local, y así sucesivamente.

Paralelamente a esto se han creado un par de claves SSH[4] que permiten la autenticación de usuarios en sistemas remotos de manera segura. Esto ha permitido que el desarrollador tenga en su host el repostorio remoto siempre actualizado, puesto que la actualización se realiza automáticamente sin la necesidad de tener que estar introduciendo usuario y contraseña cada vez que necesitas hacer un fetch, (se entiende como fetch a obtener los últimos cambios de un repositorio remoto pero sin fusionarlos en tu rama local).

4.1.4. Visual Studio Code

Visual Studio Code es un editor de código moderno y gratuito, que puede ser utilizado en múltiples plataformas y soporta una gran variedad de lenguajes de programación.

Cuenta con múltiples extensiones que facilitan el desarrollo de los proyectos, en el caso de este proyecto, estas han sido las más usadas:

- Github Copilot: Una extensión de programador de pares con tecnología de IA que proporciona finalizaciones de código relacionadas con el contexto, sugerencias e incluso fragmentos de código completos. Esta eficaz herramienta ayuda a los desarrolladores a escribir código de forma más eficiente, facilita la reducción del tiempo invertido en tareas repetitivas y minimiza los errores [17].
- GitLens: Su función es brindar información específica sobre los cambios en Git. Se utiliza para saber quién hizo un cambio y en qué momento. La extensión también añade herramientas en la interfaz, como outline, timeline y un gráfico de commits como se puede apreciar en la figura 4.2.
- Docker: Una extensión que facilita la creación, administración e implementación de las funcionalidades de Docker [18]. Este cuenta con un menú lateral, el cual se asemeja a la interfaz de Docker Desktop. En consecuencia, es como tener dicho programa dentro

[4]Explicado en detalle en la sección 4.2.3

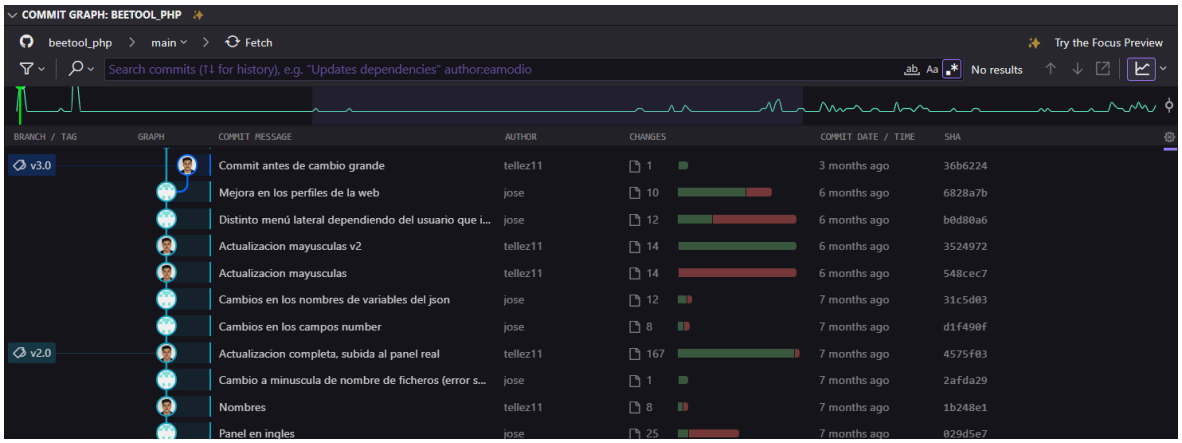

Figura 4.2: Gráfico de commits en GitLens.

del editor de código. De esta forma se mejora el rendimiento del dispositivo, además, es más rápido, fácil e intuitivo.

4.1.5. Postman

Postman en sus inicios nace como una extensión que podía ser utilizada en el navegador Chrome de Google y básicamente permite realizar peticiones de una manera simple para testear APIs[5] de tipo REST[6] (REpresentational State Transfer) propias o de terceros.

Gracias a los avances tecnológicos, Postman ha evolucionado y ha pasado de ser de una extensión a una aplicación que dispone de herramientas nativas para diversos sistemas operativos [21].

Postman se ha utilizado en el proyecto durante el desarrollo de los métodos *POST*[7] de la interfaz web hacia la herramienta computacional y viceversa.

Este programa ofrece la facilidad de crear una colección y añadir peticiones, las cuales se pueden realizar tantas veces como se desee, incluso estableciendo variables, como podría ser la IP (Internet Protocol). De igual modo, posee distintos tipos de parámetros (params, headers, body, pre-request script, authorization, entre otros). Asimismo, como Postman permite la autorización, ofrece la capacidad de poder hacer peticiones incluso necesitando autenticación.

[5]Conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas [19].

[6]Interfaz para conectar varios sistemas basados en el protocolo HTTP y sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON [20].

[7]Explicado en detalle en la sección 4.2.2

En la figura 4.3, se puede apreciar lo anterior citado, es una captura del software de Postman donde a la izquierda se observa la colección creada, la autenticación básica establecida, el método POST y la url requerida para realizar la petición.

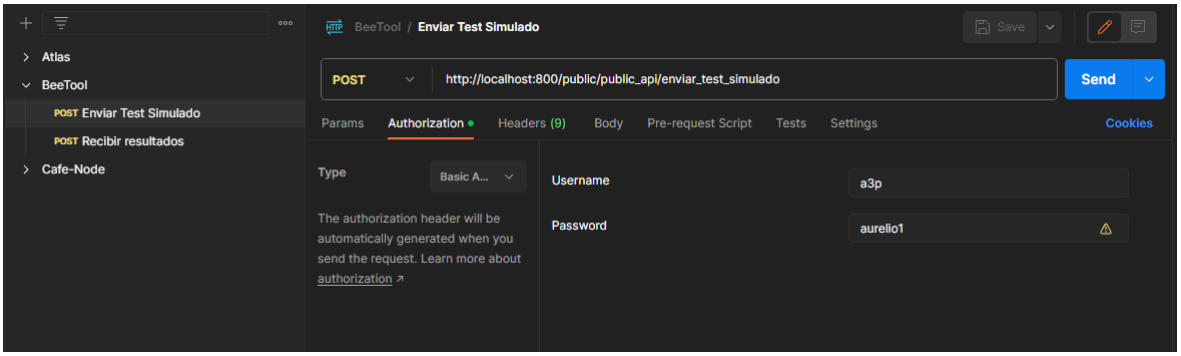

Figura 4.3: Captura del software Postman.

También, como se puede observar en la figura 4.6 de la sección 4.2.2, Postman cuenta con una consola donde se pueden observar los errores, el proceso y en último término la respuesta, de una manera más intuitiva y detallada.

4.1.6. Monaca

Monaca hace que el desarrollo de aplicaciones móviles híbridas HTML5 con Apache Cordova sea simple y fácil. Es una plataforma de desarrollo de aplicaciones híbridas abierta y está lista para conectarse de inmediato a su flujo de trabajo y entorno de desarrollo existentes. Desde Cloud IDE, CLI hasta depurador y compilación en línea remota [22].

Se ha hecho uso de Monaca en el proyecto para el desarrollo de la aplicación móvil que posteriormente se explica en detalle en el Capítulo 8.

4.2. Protocolos y mecanismos de comunicación

Los protocolos y mecanismos de comunicación son elementos fundamentales en el intercambio de información entre sistemas o dispositivos. Estos establecen las reglas y procedimientos que permiten que los datos se transmitan, se interpreten y se procesen de manera consistente y confiable. Gracias a estos protocolos y mecanismos se ha podido llevar a cabo la integración de sistemas correctamente, entre otras funcionalidades.

4.2.1. JSON

JSON (JavaScript Object Notation) es un formato que almacena información estructurada y se utiliza principalmente para transferir datos entre un servidor y un cliente. Los archivos JSON contienen solo texto y usan la extensión `.json` [23].

Hay dos elementos centrales en un objeto JSON:

- Las *keys* deben ser cadenas de caracteres.
- Los *values* son un tipo de datos JSON válido. Puede tener la forma de un arreglo (array), objeto, cadena (string), booleano, número o nulo [23].

El formato JSON está muy presente en el proyecto, principalmente durante la integración de sistemas, los archivos que se intercambian son archivos en formato JSON con la información pertinente.

En la figura 4.4 se puede observar un ejemplo de JSON que contiene los parametros introducidos por el usuario que se le van a enviar a la herramienta computacional para que realice los cálculos.

Figura 4.4: JSON con los parámetros introducidos por el usuario.

Posteriormente, se puede apreciar en la siguiente figura 4.5 el JSON con el que res-

ponderá la herramienta hacia la interfaz web con las *keys* y *values* que en la simulación se desee. En este caso, aparece un parámetro data y el nombre del informe, en vez de un JSON completo con resultados específicos. Esto se debe a que el desarrollo de los cálculos de la herramienta es información confidencial desarrollada por el grupo TecEner, en consecuencia se ha ejemplificado con un campo data y el nombre del informe.

Figura 4.5: JSON con los datos recibidos.

Cabe destacar que el informe varía constantemente demostrando así que la aplicación y la herramienta computacional están en correcto funcionamiento. Los informes actualmente muestran información detallada de los parámetros de entrada.

4.2.2. HTTP

HTTP (HyperText Transfer Protocol) es un protocolo de transferencia sobre el que se basa la red de internet. Funciona como base para los intercambios de datos realizados en la web, además, mantiene una estructura basada en los clientes y servidores y orientada a transacciones. La arquitectura del protocolo HTTP, implica que programas clientes, como Chrome y Firefox, establezcan conexión y realicen peticiones de datos a programas servidores como Apache, Nginx, entre otros [24].

Este protocolo implementa varios métodos de peticiones GET, POST, PUT, DELETE, entre otros. Estos cumplen con distintas funciones, en el caso del desarrollo del proyecto BEETool se ha hecho uso de este protocolo HTTP, realizando peticiones POST en la integración de sistemas. La solicitud POST tiene la función de enviar datos para que sean procesados en un recurso especificado [24].

HTTP cuenta con unos códigos de respuesta con la información de respuesta del recurso que se haya solicitado. De esta manera se comprueba que tipo de respuesta se recibe, las cuales pueden ser desde respuestas informativas y correctas hasta erróneas en las que indique incluso el tipo de error ocasionado.

Cabe destacar que, HTTP cuando realiza una transacción, envía información por sus

cabeceras. Estas cuentan con información que permite la capacidad de poder ampliar sus funciones, como puede ser *Agent* y *Content-Type*, entre otros. En el caso de las peticiones de BEETool la *key*:*Content-Type* tendrá como *value*:*application/json*, puesto que se realiza la comunicación mediante JSON. La cabecera permite llevar a cabo un control de las cookies.

En la figura 4.6, se puede apreciar una captura de la respuesta que ofrece la petición POST. Como se ha citado anteriormente, el método POST tiene la función de enviar datos, con esto se puede observar que devuelve un JSON con un código de respuesta, un mensaje y la data pertinentes. El código de respuesta es el "200", este indica que la petición es correcta y que ha podido responder sin problemas.

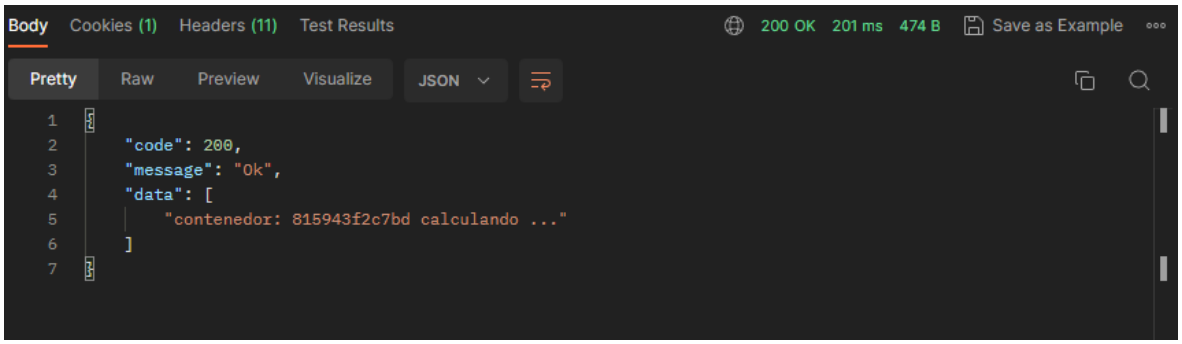

Figura 4.6: Respuesta de método POST en Postman.

4.2.3. SSH

SSH (Secure SHEll) es un protocolo para acceder de forma remota a un servidor privado. Además, da nombre al programa que permite su implementación. Posibilita el acceso y la administración de un servidor a través de una puerta trasera (backdoor). Y a diferencia de otros protocolos como HTTP o FTP, SSH establece conexiones seguras entre los dos sistemas [25].

Como se ha citado anteriormente en la sección 4.1.3, se han generado un par de claves criptográficas SSH, exactamente una clave pública y una clave privada. La clave privada permanece solo en el equipo del cliente, mientras que la clave pública se localiza en el servidor. Estas claves se almacenan en el directorio local y han sido generadas mediante la terminal de comandos, exactamente con *ssh-keygen*[8]. Posteriormente, se traslada la clave pública dentro de los ajustes de GitHub, es decir, en el servidor. De esta forma se accede de

[8]Herramienta de línea de comandos utilizada para generar pares de claves criptográficas para el protocolo SSH.

forma remota al repositorio del proyecto, en consecuencia no se tiene la necesidad de estar introduciendo los datos de inicio sesión cada vez que se quiera realizar una operación.

5. Contenerización Docker

La contenerización de aplicaciones ofrece ventajas como portabilidad, mayor eficiencia y utilización de recursos, escalabilidad rápida y aislamiento de seguridad. Los contenedores son más livianos, se ejecutan rápidamente y permiten adaptarse fácilmente a cambios de carga. Además, proporcionan mayor seguridad al aislar los procesos y recursos de cada contenedor. La contenerización es una opción moderna y preferida para desplegar aplicaciones en entornos dinámicos. Por esta razón, se ha llevado a cabo la contenerización por encima de un enfoque basado en máquinas virtuales[1]

La correcta implementación y funcionamiento del proyecto en Docker se requiere de la herramienta Docker Compose junto con un fichero *Makefile*. Dicha herramienta define y ejecuta aplicaciones de Docker de varios contenedores. En esta, se usa un archivo YAML[2] (Yet Another Markup Language) para configurar los servicios de la aplicación. Después, con un solo comando, se crean y se inician todos los servicios que se requieren para el correcto funcionamiento [28].

Sin embargo, explicar la contenerización no es posible sin antes entender los servicios que se han generado en el fichero *docker-compose.yaml*. Estos servicios son estrictamente necesarios.

Junto con todo esto, en las siguientes secciones se explicará en detalle cada uno de estos servicios que se conectan entre sí y los cuales dan lugar a una red de contenedores que permiten el correcto funcionamiento del proyecto BEETool.

Además, cabe señalar que, un fichero *Makefile* contiene las órdenes que debe ejecutar la utilidad make, así como las dependencias entre los distintos módulos del proyecto [29]. Es importante comentar que la utilidad make determina qué archivos deben ser recompilados y los comandos que se deben utilizar para realizar esta tarea [30].

En primer lugar, en el fichero *Makefile* de este proyecto se han establecido distintos comandos, en el extracto de código 5.1 se observan los principales comandos que se utilizan

[1]Software capaz de cargar en su interior otro sistema operativo, haciéndole creer que es un PC de verdad [26].

[2]Lenguaje de declaración de datos que facilita la legibilidad y la capacidad de escritura del usuario. Se encarga de almacenar archivos de configuración [27].

durante el transcurso de este.

Por otra parte, la creación de las imágenes de los contenedores se ha llevado a cabo mediante Docker Compose, ejecutando el comando *make build*, el cual establecería por línea de comandos la instrucción *docker compose build*, construyendo las imágenes de Docker a partir del fichero *docker-compose.yaml* detallado anteriormente.

Una vez ya está realizado el *make build* y se tienen las imagenes creadas, solo se debería usar *make start* o *make stop* dependiendo si se desea ejecutar la red de contenedores o eliminarla. En caso de querer ejecutar la red de contenedores y mostrar los registros de salida en la terminal con una vista detallada de los procesos en tiempo real de los contenedores, se deberá ejecutar *make start-view*. Esta directiva se suele utilizar para depurar problemas. Cabe destacar que, con el comando anterior se ejecuta un *docker compose up* pero sin la instrucción *-d*, esto permite que se realice la ejecución en segundo plano, modo *detached*.

```
start-view:
    docker compose up
start:
    docker compose up -d
build:
    docker compose build
stop:
    docker compose down
```

Extracto de código 5.1: Comandos relevantes de fichero Makefile.

5.1. Contenerización de Aplicación web

Para explicar la contenerización primeramente se debe enunciar que se necesita para desarollar una aplicación web. Necesitarás una combinación de elementos clave. En el front-end, diseñarás la interfaz de usuario. En el back-end, emplearás un lenguaje de programación y un framework para desarrollar la lógica de la aplicación. Además, requerirás una base de datos para almacenar y recuperar datos, una API para la comunicación entre el front-end y el back-end, un servidor web para alojar la aplicación. Por último, deberás implementar y alojar la aplicación en un servidor.

En relación con lo que engloba la contenerización de la parte de la aplicación web se

implementa mediante tres servicios generados en el fichero *docker-compose*. Los nombres de los tres servicios son “web”, “phpmyadmin” y “db”.

Cabe señalar que, el servicio “phpmyadmin” actúa como una interfaz de usuario web para administrar y manipular bases de datos MySQL, en este caso el servicio “db” contiene MySQL, el cual será detallado en la sección 5.1.1. Por otra parte, el servicio “web” está conectado al “db”, para así estar todos los servicios relacionados entre sí.

5.1.1. Contenedor MySQL

MySQL es un sistema de administración de bases de datos relacionales. Es un software de código abierto desarrollado por Oracle[3]. Además, se considera como la base de datos de código abierto más popularizada para almacenar y administrar datos. Es más, es un sistema comúnmente utilizado debido a sus funciones y características [32].

Se puede apreciar en el extracto de código 5.2 la configuración del servicio de base de datos MySQL, el cual será aclarado y especificado en los siguientes párrafos.

En este servicio de base de datos, se utiliza una imagen de Docker llamada “mysql”, que proporciona una instancia del sistema de gestión de bases de datos MySQL. Se configura para escuchar en el puerto 3306, esto que permite a otras aplicaciones y servicios comunicarse con él a través de ese puerto.

Así mismo, se establece un comando específico para configurar la forma de autenticación predeterminada para los usuarios de la base de datos. En este caso, se utiliza el método de autenticación *mysql_native_password*. Este método verifica las credenciales de los usuarios almacenadas en la base de datos utilizando un algoritmo de hash[4] específico.

El nombre del contenedor se establece como “mysql”, lo que facilita la identificación y el manejo del contenedor dentro de la red creada en Docker.

No obstante, por un lado, se definen variables de entorno para configurar la base de datos. Estas son especificadas con el nombre de la base de datos “dbname”, requiere de una contraseña para los usuarios normales como *test* y la contraseña del usuario *root*[5] también como *test*. Estas variables posteriormente se definen en el contenedor de PHP para poder mantener relación con la base de datos, esto último está explicado con más detalle en la

[3]Empresa multinacional estadounidense dedicada al ámbito de la tecnología informática [31]
[4]Algoritmo matemático que transforma los datos de entrada en un código único [33].
[5]Usuario con privilegios administrativos

sección 5.1.3.

Por otro lado, se establecen volúmenes para almacenar y compartir datos entre el contenedor y el sistema anfitrión. Se utilizan volúmenes denominados “dump”, “conf” y “database” para cargar el *dump*[6] de la base de datos al iniciar el contenedor, guardar la configuración del contenedor y la información de la base de datos, respectivamente. Esto permite que no se pierda la configuración o todo el sistema de almacenamiento de la información cuando se reinicie el contenedor.

```
db:
    image: mysql
    ports:
        - 3306:3306
    command: --default-authentication-plugin=mysql_native_password
    container_name: mysql
    environment:
        MYSQL_DATABASE: dbname
        MYSQL_PASSWORD: test
        MYSQL_ROOT_PASSWORD: test
    volumes:
        - dump:/docker-entrypoint-initdb.d
        - conf:/etc/mysql/conf.d
        - database:/var/lib/mysql
```

Extracto de código 5.2: Servicio db en fichero docker-compose.

En resumen, este servicio de base de datos MySQL configurado en un entorno Docker proporciona una instancia funcional de MySQL con opciones de autenticación, nombres de base de datos y contraseñas personalizables, que junto con los volúmenes permiten la persistencia de los datos y la configuración entre las ejecuciones del contenedor.

5.1.2. Contenedor PhpMyAdmin

PhpMyAdmin es una aplicación web que sirve para administrar bases de datos MySQL de forma sencilla y con una interfaz amistosa. Se trata de un software muy popular basado en PHP. La ventaja de usar una aplicación web es que permite conectar a los usuarios con

[6]Fichero que engloba una copia de seguridad o backup de algo

servidores remotos, a los cuales no siempre se puede acceder usando programas de interfaz gráfica.

Para usar PhpMyAdmin simplemente se necesita subir el conjunto de archivos PHP que componen la aplicación a un servidor web, configurar con los datos de acceso a MySQL y empezar a administrar las bases de datos. Con phpMyAdmin se puede hacer todo tipo de operaciones, desde la creación y borrado de bases de datos a la administración de las tablas (crear, modificar y eliminar) y, por supuesto, de sus propios datos [34].

Para la ejecución del contenedor se ha configurado el servicio “phpmyadmin” como se puede ver en el extracto de codigo 5.3.

En el código, se especifica la imagen de Docker a utilizar, que en este caso es una imagen oficial Docker de PhpMyAdmin. Acto seguido, se establece una conexión con el contenedor de MySQL denominado “mysql” utilizando el enlace “db:db. Con todo esto, se explica el significado de lo que se puede conseguir, es decir, lograr que PhpMyAdmin se comunique con el contenedor de base de datos MySQL.

Al mismo tiempo, se es necesario indicar que, el contenedor de PhpMyAdmin estará accesible a través del puerto 8001 en el host local, redirigiendo el tráfico al puerto 80 del contenedor. Para ello, se debe asignar un nombre al contenedor, en este caso “database”, para facilitar su identificación y gestión dentro de la red de contenedores de Docker.

Por último, definir las variables de entorno es el paso final para configurar la conexión a la base de datos MySQL. No se debe olvidar el establecimiento de las mismas variables que las citadas en la configuración del servicio de MySQL anterior. Además, es importante señalar que, por un lado, estas variables de entorno permiten que PhpMyAdmin establezca la conexión con el servidor MySQL y tenga los privilegios necesarios para administrar la base de datos. Y, por otro lado, dicho código autoriza el desplegar PhpMyAdmin junto con una instancia de MySQL en el entorno del proyecto.

```
1 phpmyadmin:
2         image: phpmyadmin
3         links:
4             - db:db
5         ports:
6             - 8001:80
7         container_name: database
8         environment:
```

```
9               MYSQL_USER: root
10              MYSQL_PASSWORD: test
11              MYSQL_ROOT_PASSWORD: test
```

Extracto de código 5.3: Servicio phpmyadmin en fichero docker-compose.

5.1.3. Contenedor PHP

PHP es un lenguaje de programación creado para el desarrollo de aplicaciones y la creación de sitios web. Es conocida por ser fácil de usar y en constante perfeccionamiento. Además, es una opción segura para aquellos que desean trabajar en proyectos sin complicaciones [35].

Para el desarrollo de la aplicación web, se solicitó tener un contenedor con la versión de PHP 7.2 teniendo en cuenta las dependencias y extensiones de PHP requeridas para el correcto funcionamiento, considerando el servidor Web Apache, el manejador de paquetes Composer y RedbeanPHP.

5.1.3.1 Generación de imagen PHP con fichero Dockerfile

Debido a las instalaciones tan específicas solicitadas por el desarrollador web José López Conejero, se ha tenido que generar un fichero *Dockerfile* el cual generara la imagen con los requerimientos necesarios. Este proceso de creación de imagen a partir de un *Dockerfile*, se detalló anteriormente en la sección 4.1.1 usando como apoyo a la explicación la figura 4.1.

De acuerdo con esto, es necesario explicar en este punto el desarrollo del fichero *Dockerfile* para la creación de la imagen deseada. Para ello, en el extracto de código A.1 se puede apreciar dicho archivo, el cual se aclarará a continuación.

En primer lugar, se establece la imagen base “php:7.2-apache”. Esto significa que el contenedor se construirá utilizando la versión 7.2 de PHP junto con el servidor web Apache.

Seguidamente, se establece el directorio de trabajo dentro del contenedor. En este caso, `/var/www/html`. Esto significa que todas las operaciones posteriores se realizarán en este directorio.

Posteriormente, se ejecutan comandos con la instrucción RUN. Estos comandos se ejecutan dentro del contenedor y se creará una nueva capa en la imagen que captura el

resultado del comando ejecutado.

Para dar inicio al proceso se debería iniciar en primer lugar *a2enmod rewrite*, el cual habilita el módulo rewrite de Apache, que permite la reescritura de URLs (Uniform Resource Locators). Con esto, se le otorga al contenedor la capacidad de implementar enlaces amigables o redireccionamientos en la aplicación web. A partir de aquí, se realizarían los comandos pertinentes para instalar paquetes, introduciendo primeramente *apt-get update* y posteriormente *apt-get install*. Con todo esto, no se puede apartar de la presente explicación la mención de utilización de las librerías siguientes que actualizan la lista de paquetes disponibles y permiten instalar las dependencias necesarias en el contenedor.

- libpng-dev: Permite a los programas compilar y ejecutar funcionalidades relacionadas con imágenes PNG (Portable Network Graphics).
- libzip-dev: Proporciona funciones y herramientas para comprimir y descomprimir archivos ZIP, así como para manipular su contenido.
- curl: Capacita a los programas de la posibilidad de realizar solicitudes y descargas de archivos, en este caso, para realizar la instalación de Redbean.
- tar: Instala las herramientas necesarias para trabajar con archivos tar.

Así pues, se eliminan los archivos de lista de paquetes descargados por *apt-get* durante la instalación, lo cual ayuda a reducir el tamaño de la imagen final.

Para finalizar con la aclaración de las instrucciones RUN, se ejecuta el comando *docker-php-ext-install*. Este se utiliza para instalar y habilitar varias extensiones de PHP necesarias para el proyecto. Además, estos formatos son utilizados comúnmente en el desarrollo de aplicaciones web. Las extensiones que se incluyen son:

- zip: Permite trabajar con archivos comprimidos en formato ZIP.
- pdo_mysql: Facilita a las aplicaciones PHP interactuar con bases de datos MySQL.

Una vez ejecutadas las instrucciones RUN, es momento de pasar a explicar las COPY. La directiva COPY se utiliza para copiar archivos desde el directorio local al contenedor. La primera instrucción COPY copia el archivo de configuración *apache2.conf* al directorio `/etc/apache2/apache2.conf` dentro del contenedor. Esto permite personalizar la configuración de Apache según las necesidades del proyecto. En este caso, se ha añadido un parámetro "ServerName" inicializado a localhost para evitar un warning[7] que emite por consola.

[7]Mensajes de advertencia que suele emitirse en situaciones en las que es útil alertar al usuario de

Siguiendo con la explicación, se copia el archivo ejecutable de Composer desde la imagen oficial en su última versión al directorio `/usr/bin/composer` dentro del contenedor. Esta es una herramienta popular utilizada en el desarrollo de proyectos PHP para gestionar las dependencias del proyecto.

Posteriormente, se copian los archivos *composer.json* y *composer.lock* desde el directorio local al directorio de trabajo dentro del contenedor. Dichos archivos contienen información sobre las dependencias del proyecto. Exactamente, contienen las dependencias solicitadas por el desarrollador web José López Conejero. Estos archivos han sido creados mediante el comando *composer init*, acto seguido, se le han agregado las dependencias requeridas.

La instrucción RUN *composer install* permite descargar e instalar las dependencias especificadas en el archivo *composer.json*. En este caso, se ha instalado sin verificar los requisitos de plataforma y sin ejecutar scripts[8] post-instalación. Composer se encarga de ejecutar las acciones de descarga e instalación dependientes requeridas para el proyecto gracias al fichero *composer.json*.

Finalmente, es necesario que se comprenda que, la instrucción donde se descarga el archivo `tar.gz` de RedBeanPHP[9] es desde el enlace proporcionado. Es más, este guarda el archivo descargado en el directorio actual del contenedor y permite que se descomprima el contenido del archivo en el catálogo especificado del contenedor. Para finalizar el proceso, se elimina el archivo *redbean.tar.gz* del directorio actual del contenedor.

En resumen, este *Dockerfile* establece un entorno Docker basado en PHP 7.2 y Apache. Configura el servidor Apache, instala las dependencias necesarias, habilita módulos importantes, copia archivos de configuración y dependencias del proyecto, e instala RedBeanPHP como una biblioteca adicional en el contenedor.

5.1.3.2. Creación del servicio web

Una vez se ha desarrollado el *Dockerfile* que creara la imagen deseada, se pasa a implementar el servicio encargado de generar el contenedor de la interfaz web con la imagen creada anteriormente.

alguna condición en un programa, esa condición no justifica que se haga una excepción y se termine el programa [36].

[8]Término usado en programación para hablar de los fragmentos de código usados para dar forma a herramientas. Constituye el código de una aplicación en su totalidad o una de sus funciones [37].

[9]Biblioteca ORM (Object-Relational Mapping). ORM es un patrón de diseño que permite mapear objetos de una aplicación a tablas en una base de datos relacional.

En el extracto de código 5.4, se puede apreciar el servicio web, el cual se expone en detalle en los siguientes párrafos.

Para comenzar se define el servicio llamado “web”. Este servicio utiliza una imagen llamada “web” para construir el contenedor. La imagen “web” se construirá utilizando el archivo de construcción *Dockerfile* anteriormente mencionado. Como se puede apreciar en la figura 5.1, se puede observar como está organizado el directorio, de esta forma se aprecia que el fichero *Dockerfile* se encuentra en el directorio *./web*.

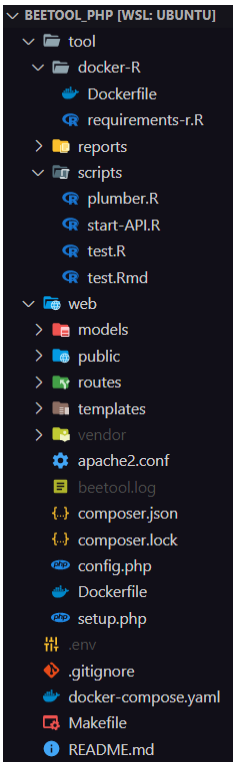

Figura 5.1: Directorio del proyecto en entorno local.

Siguiendo el extracto de código 5.4, se detalla que el servicio estará accesible desde el puerto 800 de la máquina host[10], ya que se realiza un mapeo de puertos entre el puerto 800 de la máquina host y el puerto 80 del contenedor. Esto permitirá acceder a la aplicación web en el contenedor a través del puerto 800 en la máquina host.

El nombre del contenedor se establece como “web”, lo que facilita su identificación y referencia en otras partes del archivo de configuración o en comandos relacionados con el contenedor.

Se definen varios volúmenes para el contenedor, en este caso se establecen los siguientes volúmenes:

[10]Máquina que ha ejecutado la red de contenedores, en otras palabras la máquina local.

1. Se comparte el directorio `./web` de la máquina host con el directorio de trabajo anteriormente inicializado en el *Dockerfile*. Esto significa que los archivos y carpetas ubicados en *./web* de la máquina host estarán disponibles dentro del contenedor.

2. Se comparte el directorio `./tool/reports/pdf` de la máquina host con el directorio `/var/www/html/public/reports/pdf` del contenedor. Esto permite acceder a los informes generados por la herramienta computacional y que se puedan mostrar en la aplicación web. Más adelante, en la sección 5.2.1 se detalla otro volumen en el mismo directorio local pero conectado al directorio del contenedor herramienta computacional.

3. Se establecen los volúmenes "vendor" y "properties" para los directorios `/var/www/html/vendor` y `/var/www/html/public/property-templates` respectivamente. Esto indica que estos directorios dentro del contenedor serán utilizados para compartir y persistir datos, en específico son directorios generados por Composer y tienen archivos necesarios para el funcionamiento de la aplicación web.

Este servicio se enlaza al servicio de MySQL, anteriormente detallado en la sección 5.1.1. Esto indica que hay una dependencia entre estos dos servicios, por otra parte, la aplicación web accede a una base de datos proporcionada por el servicio "db".

Por último, se especifica un archivo de entorno llamado *.env*. Este archivo contiene la variable de entorno "IP" que se utilizará para la tarea de integración de sistemas posteriormente en el capítulo 6.

```
web:
    image: web
    build: ./web
    ports:
        - 800:80
    container_name: web
    volumes:
        - ./web:/var/www/html
        - ./tool/reports/pdf:/var/www/html/public/reports/pdf
        - vendor:/var/www/html/vendor
        - properties:/var/www/html/public/property-templates
    links:
        - db
    env_file:
```

```
15      - .env
```

Extracto de código 5.4: Servicio web en fichero docker-compose

En general, este fragmento de código configura el servicio “web” dentro de un contenedor, establece mapeos de puertos, define volúmenes para compartir archivos y directorios, establece dependencias con otros servicios, y configura variables de entorno para el contenedor. Gracias a esto se ofrece el despliegue y la configuración de la aplicación web en un entorno de contenerizado.

5.2. Contenerización de herramienta de calculo

Con respecto a lo que engloba la contenerización de la parte de la herramienta computacional se implementa mediante un servicio generado en el fichero *docker-compose*. En exactitud, el servicio “tool”, el cual se comunicara con el servicio “web” anteriormente aclarado en la sección 5.1.3.

5.2.1. Contenedor R

R es un entorno y lenguaje de programación abierto, libre y gratis que proporciona una variedad de técnicas estadísticas y gráficas. Es un proyecto GNU, lo que significa que cualquier persona tiene derecho a estudiar, usar, modificar y compartir el software sin que pertenezca a nadie [38].

Para el funcionamiento de la herramienta computacional, se solicitó tener un contenedor que contuviese el lenguaje de programación R, RMarkdown, Plumber, además de distintas librerías y dependencias de R necesarias para el desarrollo de la herramienta y la integración junto con la aplicación web.

De acuerdo con lo dicho previamente, RMarkdown es una herramienta muy útil para la generación de documentos dinámicos y reproducibles en el lenguaje de programación R. Permite combinar código, texto y resultados en un solo archivo, y generar diferentes formatos de salida según las necesidades del usuario. Rmarkdown ha sido utlizado para la creación de los informes de la herramienta. Por otra parte, Plumber es un paquete de R que permite crear APIs usando R.

5.2.1.1. Generación de imagen R con fichero Dockerfile

Debido a los paquetes tan específicos solicitados por el Grupo TecEner, para que la herramienta computacional pueda funcionar correctamente se ha generado un fichero *Dockerfile* el cual va a generar la imagen con todos los paquetes, dependencias y librerías necesarios.

En el extracto de código A.2 podemos ver el fichero *Dockerfile*, el cual va a ser aclarado en los siguientes párrafos.

Primeramente, se puede apreciar que el *Dockerfile* se basa en una imagen base de Ubuntu:22.04. A partir de este punto, se establece una variable de entorno "DEBIAN FRONTEND" en "noninteractive", lo que permite la instalación sin necesidad de interacción del usuario. Para continuar, se establece el directorio de trabajo en `/home/beetool` dentro del contenedor.

Más adelante, se ejecuta una instrucción RUN con distintos comandos, donde se ejecuta el comando para actualizar los repositorios de paquetes. Acto seguido, se instalan los paquetes r-cran-rmarkdown y ºcran-plumber. Por último, se eliminan los archivos temporales generados durante la instalación para reducir el tamaño de la imagen resultante.

A continuación, se añade el script *requirements-r.R* al directorio de trabajo dentro del contenedor. Luego, se ejecuta el script utilizando *Rscript* para instalar las dependencias R necesarias. Posteriormente, se eliminan los archivos temporales generados durante la instalación.

Posteriormente, se ejecuta el comando que permite instalar el paquete tinytex utilizado para documentos LaTeX desde R, este paquete es necesario para la creación del informe.

Seguidamente, se otorgan permisos del directorio de trabajo para permitir el acceso completo a todos los usuarios dentro del contenedor. Y se establece el punto de entrada del contenedor, especificando que se debe ejecutar el código R contenido en el archivo *start-API.R*. Esto permite que el contenedor inicie una API basada en R al iniciarse.

En resumen, este *Dockerfile* configura un entorno Docker utilizando Ubuntu 22.04, instala las dependencias R necesarias, establece permisos adecuados en el directorio de trabajo y configura el punto de entrada para ejecutar un archivo R que inicia una API.

Cabe destacar que se puede apreciar en el extracto de código 5.5. El archivo que inicia la API y luego ejecuta el servidor local en el puerto 8000, lo que permite interactuar con la

API utilizando las rutas y los comportamientos definidos en el archivo *plumber.R*. El fichero *plumber.R* es el encargado de recibir el JSON de la aplicación web, el cual será explicado en el Capítulo 6.

```
1 library(plumber)
2
3 root <- pr("plumber.R")
4
5 root %>% pr_run(port = 8000, host = "0.0.0.0")
```

Extracto de código 5.5: Fichero start-api.R.

5.2.1.2. Creación del servicio tool

Como se ha citado anteriormente en la sección previa, una vez se ha creado la imagen deseada, se debe implementar el servicio encargado de generar el contenedor.

En el extracto de código 5.6 se puede observar el servicio "tool", el cual se explica detalladamente en los siguientes párrafos.

Para comenzar con el servicio se define el nombre del servicio como "tool" y se especifica el nombre de la imagen de Docker que se utilizará para crear el contenedor.

Posteriormente, el siguiente punto a explicar será el *build:* `./tool/docker-R`. Esta línea indica que si la imagen no existe en tu sistema Docker Compose debe construir la imagen utilizando el archivo *Dockerfile* ubicado en la carpeta `./tool/docker-R`. Como se puede apreciar en la figura 5.1 el fichero *Dockerfile* está en el directorio correctamente alojado.

A continuación, se establece el nombre del contenedor Docker creado a partir de esta configuración.

En este caso, se montan dos volúmenes: `./tool` del host se monta en `/home/beetool/tool` del contenedor, y `./tool/reports/pdf` del host se monta en `/home/beetool/tool/reports/pdf` del contenedor. Este último se ha creado con el objetivo de que se repliquen los PDF generados mediante la herramienta computacional en otra carpeta de la aplicación web y así poder visualizar los informes en la web, como se ha citado anteriormente en la sección 5.1.3.

Por lo consiguiente, se indica el directorio de trabajo para cuando se inicie el contenedor,

el cual contiene los scripts necesarios para que la herramienta computacional llevo a cabo todos sus procesos. Estos scripts se pueden apreciar en la figura 5.1, dentro de la propia carpeta scripts. Además, se ha añadido el fichero *.env* que contiene la "IP" como variable de entorno para que de esta forma se pueda llevar a cabo la integración entre contenedores. Este último proceso está más detallado en el Capítulo 6.

Finalmente, se expone el puerto 8000 del contenedor que interactúa con la herramienta computacional.

```
tool:
    image: tool
    build: ./tool/docker-R
    container_name: tool
    volumes:
    - ./tool:/home/beetool/tool
    - ./tool/reports/pdf:/home/beetool/tool/reports/pdf
    working_dir: /home/beetool/tool/scripts
    env_file:
     - .env
    ports:
    - 8000:8000
```

Extracto de código 5.6: Servicio tool en fichero docker-compose.

6. Integración de sistemas

La integración de sistemas es el proceso de conectar diferentes sistemas o software para cooperar, funcionando como un solo sistema compartiendo datos e información [39].

Cabe destacar que, el principal objetivo del proyecto es poder visualizar los resultados de los cálculos e informes generados a partir de los parámetros introducidos, para este desarrollo se ha llevado a cabo la integración de sistemas entre el contenedor de la herramienta computacional y el de la aplicación web.

En este capítulo se detalla cada una de las APIs que hacen posible estas comunicaciones entre contenedores. Exactamente, se explica el proceso de visualización de los informes generados junto con una cadena JSON con los resultados de los cálculos de la herramienta a partir de los parámetros introducidos.

6.1. Flujo de ejecución de un test

En el proyecto BEETool, un "test" se refiere al proceso de introducir parámetros específicos en la aplicación web. Estos parámetros son enviados a la herramienta para realizar un cálculo específico. Una vez que la herramienta ha completado el cálculo, se genera un informe que incluye los resultados obtenidos.

Se pueden realizar dos tipos de pruebas. Uno de ellos es el *simulatedtest*, que consiste en una prueba individual simulada. El otro tipo de prueba es el *testplan*, que es una prueba compuesta por varios *simulatedtest*. En el contexto de BEETool, el *simulatedtest* se utiliza para evaluar aspectos específicos de manera aislada, mientras que el *testplan* permite probar de manera integral y exhaustiva diferentes escenarios al combinar varios *simulatedtest*. Estos dos tipos de pruebas brindan flexibilidad y capacidad para realizar evaluaciones tanto a nivel individual como en un contexto más amplio.

La creación de cada uno de los "test" dentro de la interfaz web se aborda en el TFG del alumno José Lopez Conejero. Sin embargo, para el propósito de este TFG, no es relevante identificar el tipo de test específico que se esté ejecutando, ya que el flujo de ejecución es el mismo en todos los casos, variando únicamente la información que se envía a la herramienta

computacional.

El proceso para visualizar los informes y los resultados ofrecidos por la herramienta computacional se puede apreciar en la figura 6.1, desde que se obtienen los datos de la web y se envían hasta que de nuevo la aplicación web recibe de la herramienta computacional el informe generado junto con los resultados del cálculo. Este proceso se va a explicar de manera detallada en las siguientes secciones.

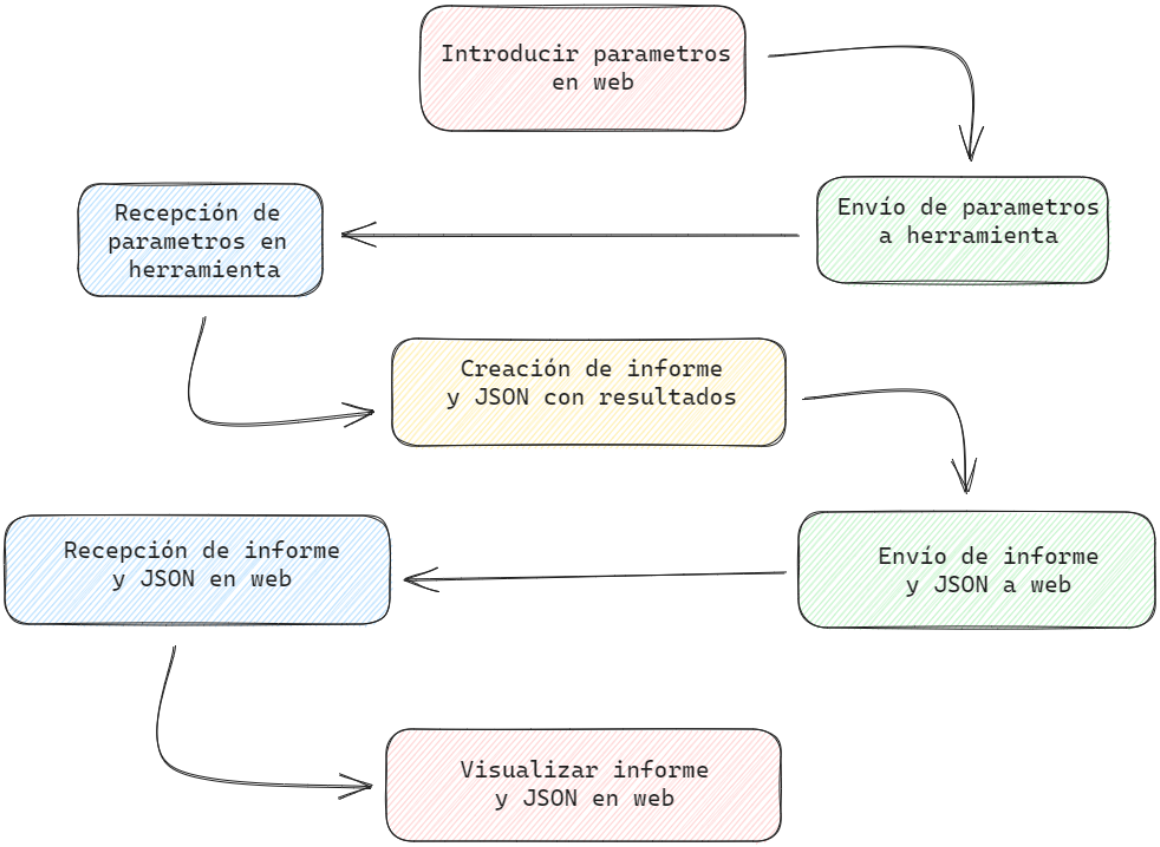

Figura 6.1: Diagrama de flujo de ejecución de test.

6.1.1. Obtención de parámetros en JSON

La obtención de los parámetros se realizará mediante el extracto de código 6.1, el cual se lleva a cabo cuando se pulsa al botón simular. Este extracto utiliza la función *ajax()* de jQuery[1] para realizar una solicitud AJAX[2] al servidor. En concreto, se trata de una solicitud POST a la URL indicada. Los datos enviados al servidor están definidos dentro del objeto

[1]Biblioteca de JavaScript minificada de código.

[2]Conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona, procesando cualquier solicitud al servidor en segundo plano [40].

data, donde se incluyen los valores de “id” y “type”. Estos valores indican el test que se ha simulado para así obtener los parámetros del mismo, estos valores se insertarán en la solicitud antes de su envío. Se especifica que se espera recibir una respuesta en formato JSON.

```
\$.ajax({
    type: "POST",
    url: 'http://localhost:800/public/public_api/
       enviar_test_simulado',
    data: {
        id:"{{ bean.id }}",
        type:"{{ beantype }}"
    },
    dataType: 'json',
});
```

Extracto de código 6.1: Metodo POST que permite obtener los parametros del JSON y enviarlos

La solicitud espera recibir un JSON. En el extracto de código A.4, se observa como se ha generado este JSON de manera simplificada. A su vez se llama a la funcion encargada de enviar el JSON, la cual será aclarada con detalle en la sigiuente sección 6.1.2.

Seguidamente, se puede observar un ejemplo breve de como se ha ido creando el archivo JSON, como se aprecia se usa “R::find” esto forma parte de RedBeanPHP, esto permite obtener los parámetros de la base de datos, una vez que se han obtenido todos los parámetros se crea el JSON “$env_json” el cual se envía posteriormente a la herramienta computacional.

La creación del archivo JSON, escogiendo cada uno de los parámetros, se le atribuye a la parte del TFG colaborativo de José López Conejero, por lo tanto, no están explicados en detalle estos procesos, tanto en la parte de envío como de recepción de la sección 6.1.6.

6.1.2. Envío de JSON a herramienta

Se puede observar en el método POST del extracto de código A.4 que una vez ya se tiene el JSON generado correctamente, se asigna a la función *enviar_post*. Esta recibe dos parámetros, por una parte, la ruta, a la cual se le añadirá delante la ”API_URL”, que en este caso es la ruta del contenedor de la herramienta computacional, quedando de la forma `API_URL/report`. Y, por otra parte, el propio JSON. La función comentada se puede apreciar

en el extracto de código A.3 que será aclarada en los siguientes párrafos.

Ahora bien, en esta se tiene los dos parámetros de entrada anteriormente mencionados. Primeramente, se inicializa una instancia de cURL. Dicha cURL es una herramienta basada en líneas de comandos que se puede usar para transferir datos por medio de protocolos de red [41], además, permite que se configuren varias opciones para establecer el comportamiento de la solicitud. Entre las alternativas más destacadas se incluyen:

- "CURLOPT_URL" se fija utilizando la constante "API_URL" concatenada con la variable ruta para formar la URL de la API, como ya se ha citado anteriormente sería `API_URL/report`
- "CURLOPT_USERPWD" se implanta utilizando las constantes "USER" y "PWD" concatenadas con ":" para proporcionar las credenciales de autenticación básica.
- "CURLOPT_POSTFIELDS" se establece utilizando *json_encode()* para enviar los datos en formato JSON. Los datos se pasan como un array con una clave "items" que contiene el valor de "Datos", en este caso el valor de "Datos" es el JSON proporcionado.

Seguidamente, se ejecuta la solicitud cURL utilizando *curl_exec($ch)* y el resultado se asigna a la variable "$data". Siendo de esta manera "$data" el JSON generado.

Como consecuencia, se verifica si "$data" está vacío. Si está vacío, se asigna el error de cURL utilizando *curl_error($ch)* a "$data" y se devuelve concatenado con "***".

Si "$data" no está vacío, se cierra la conexión cURL..

Finalmente, se devuelve el contenido de "$data" decodificado como JSON utilizando *json_decode($data)*.

6.1.3. Recepción del JSON en herramienta

La recepción del archivo JSON, como se ha citado anteriormente al final de la sección 5.2.1, se apela al fichero *plumber.R*, el cual está a la espera de recibir el JSON, es el que realiza las siguientes interacciones. Se puede apreciar en el extracto de código A.6.

Las líneas iniciales importan las bibliotecas R.utils y jsonlite. Estas bibliotecas son utilizadas para manejar archivos y realizar operaciones de serialización y deserialización de datos en formato JSON.

El código está definido como una función que maneja una solicitud HTTP específica.

Esta función se ejecutará cuando se realice una solicitud HTTP con el método POST a la ruta `/report`. Esta ruta es la que se ha comentado anteriormente en la sección anterior, la cual estaba compuesta por la `API_URL/report`. La anotación #* indica que se trata de una documentación para el enrutamiento del servicio.

Dentro de la función, la variable "req", representa la solicitud HTTP recibida, y "res" es la respuesta que se enviará al cliente.

A continuación, se convierte el cuerpo de la solicitud HTTP en una representación JSON utilizando la función *toJSON* de la biblioteca jsonlite.

El nombre del JSON se construye concatenando la ruta donde se guardaran los archivos, la fecha y hora actual, una cadena aleatoria de 5 letras y la extensión `.json`. Esto crea un nombre de archivo único para cada solicitud recibida. Teniendo así un JSON único recibido por cada informe solicitado. Seguidamente, se guarda el contenido JSON en un archivo utilizando la función *write*.

Con todo esto ya se tiene el JSON recibido y alojado en una carpeta del propio contenedor.

6.1.4. Creación de informe y obtención de resultados de la herramienta

La creación del informe se realiza mediante el fichero *test.R*, el cual se puede apreciar en la primera parte de este fichero en el extracto de código A.7. Este fichero se ejecuta mediante el fichero *plumber.R*, en el extracto de código anterior A.6 se aprecia que se construye la llamada al script *test.R*. El símbolo "&" al final de la cadena indica que se debe ejecutar en segundo plano. Por último, se devuelve una cadena de texto que indica que el contenedor ha recibido el JSON y está calculando los datos.

De acuerdo con la primera parte del fichero *test.R*, cabe destacar que este se encarga de generar el archivo PDF (Portable Document Format) como se aprecia en el extracto código A.7.

Por otro lado, se importan las bibliotecas necesarias para el script. Estas bibliotecas incluyen knitr, rmarkdown, R.utils, httr y jsonlite, además, proporcionan funcionalidades para generar informes, manejar archivos, realizar solicitudes HTTP y manipular datos en formato JSON.

El nombre del PDF se construye de la misma manera que se genera el nombre del

archivo JSON pero en este caso con el archivo PDF.

Finalmente, se realiza la creación del informe mediante la función *render* del paquete rmarkdown, la cual genera el informe en formato PDF utilizando el archivo R Markdown llamado *test.Rmd* este lo ha proporcionado el Grupo TecEner y contiene los cálculos, gráficos y estadísticas que se visualizaran en el informe, además de los resultados a los cálculos establecidos en la propia herramienta.

En último término, el informe se guarda con el nombre de archivo especificado en el directorio `/home/beetool/tool/reports/pdf/`. Es importante recordar que, este directorio se establecía en un volumen el cual acababa siendo una carpeta compartida con el contenedor de la aplicación web.

Cabe destacar que todo lo aclarado ha sido desarrollado por Jose Antonio Téllez Mérida, excepto el fichero *test.Rmd* que ha sido desarrollado por el Grupo TecEner. Este fichero es el encargado de realizar los cálculos pertinentes para llevar a cabo las correctas predicciones. Y mediante este fichero generar el informe.

6.1.5. Envío de JSON con resultados de los cálculos a la interfaz web

Aparte de generar un informe la herramienta computacional, se genera un JSON con distintos parámetros que el cliente podrá observar, como se aprecia en la figura 4.5. En este caso se genera un campo "data" con dos valores por defecto debido a que la información de la herramienta computacional es confidencial y no cuenta con los cálculos reales. En consecuencia, se imprime el nombre del informe generado para demostrar que el JSON de vuelta se recibe y se actualiza.

Este proceso de envío se puede apreciar en la segunda parte del fichero R, exactamente en el extracto de código A.8, el cual se detalla en los siguientes párrafos.

Primeramente, la variable "IP" se obtiene del entorno del sistema, siendo la IP del servidor, en este caso de la máquina local.

Se obtienen los argumentos pasados al script en la línea de comandos. De acuerdo con este, se le cambia el nombre del fichero JSON anteriormente generado. Con esto se guarda el nombre en la variable "filename_json". Acto seguido, se lee el contenido del archivo JSON y se convierte en un objeto de datos en R utilizando la función *fromJSON* de la biblioteca jsonlite.

Posteriormente, se extraen los valores requeridos para hacer la llamada de API externa, en este caso se necesita el "id" y el "type". Uno de ellos es el identificador y el otro el tipo de "test", ya se ha expuesto anteriormente que puede ser un "simulatedtest" o un "testplan".

Al mismo tiempo, se construye una URL utilizando la dirección IP, el tipo y el ID obtenidos anteriormente. Esto parece ser una llamada a una API externa para enviar resultados, el cual tendrá este formato `http:/IP:800/public/public_api/recibir_resultados/type/id`

Seguidamente, se construye una cadena JSON que simula los datos calculados por la herramienta computacional y el nombre del archivo PDF generado. Además, se convierte el JSON en un objeto de datos en R utilizando *fromJSON*.

Finalmente, se realiza una solicitud HTTP POST a la URL especificada anteriormente utilizando la biblioteca httr. Se envía el objeto "mypdf" como cuerpo de la solicitud en formato JSON.

6.1.6. Recepción de JSON en interfaz web

Para recibir el JSON generado por la herramienta, se ha utilizado el extracto de código A.5,el cual define una ruta para recibir resultados en formato JSON a través de una solicitud HTTP POST. La ruta está definida con dos parámetros dinámicos que se capturan para su uso posterior, la "id" y el "type".

Cuando se recibe una solicitud POST en esta ruta, se ejecuta una función anónima con tres parámetros: "$request", "$response" y "$args". Estos representan la solicitud HTTP, la respuesta HTTP y los argumentos capturados de la URL, respectivamente. Este suceso también ocurre en la sección anterior 6.1.1 cuando se ha generado el JSON a partir de los parámetros con el POST *enviar_test_simulado*

Es, por tanto, que dentro de la función, se obtiene el JSON recibido en la solicitud POST mediante *$request-getParams()*. Estos parámetros se guardan en la variable $input, la cual se debe someterla a un *json_encode* para codificarlo en un archivo JSON, que se pueda mostrar por pantalla Además, los valores capturados de los parámetros dinámicos de la URL se asignan a las variables correspondientes, las cuales permitirán saber a qué test se está haciendo referencia.

En resumen, este código define una ruta para recibir resultados en formato JSON a

través de una solicitud POST y los parámetros enviados en la solicitud se capturan y se devuelve una respuesta con el código de estado “200”.

6.1.7. Visualización de informe en interfaz web

La visualización del informe se debe a las carpetas compartidas por la red de contenedores y sus volúmenes, pero no tiene nada que ver con las APIs ya mencionadas. El directorio `./tool/reports/pdf` está compartido en los dos contenedores, en el de la herramienta, el cual es el directorio donde se genera el informe, exactamente `/home/beetool/tool/reports/pdf`. Y en el contenedor web, el directorio `/var/www/html/public/reports/pdf`, esto le permite a la web encontrar el informe generado. Además, como en el JSON anteriormente recibido aparece el nombre del informe, tan solo hace falta buscarlo en el directorio y mostrarlo en la web.

Estas carpetas compartidas y establecidas mediante volúmenes se llevan a cabo gracias a los servicios generados en el fichero *docker-compose*, anteriormente detallado en el capítulo 5.

En la figura 6.2 se muestra el contenido JSON recibido, que contiene el nombre del informe y un campo “data” que simula los resultados ofrecidos por la herramienta computacional. Cabe destacar, aunque ya se ha citado en otras ocasiones, que el resultado de la herramienta se simula con un campo “data” debido a que no se pueden mostrar los cálculos originales, ya que es información confidencial del Grupo TecEner. Debajo, se presenta el informe generado junto con una serie de botones que permiten realizar acciones como imprimir y descargar el informe, entre otras opciones. Esta representación visual proporciona una vista clara y accesible del informe y las funcionalidades asociadas.

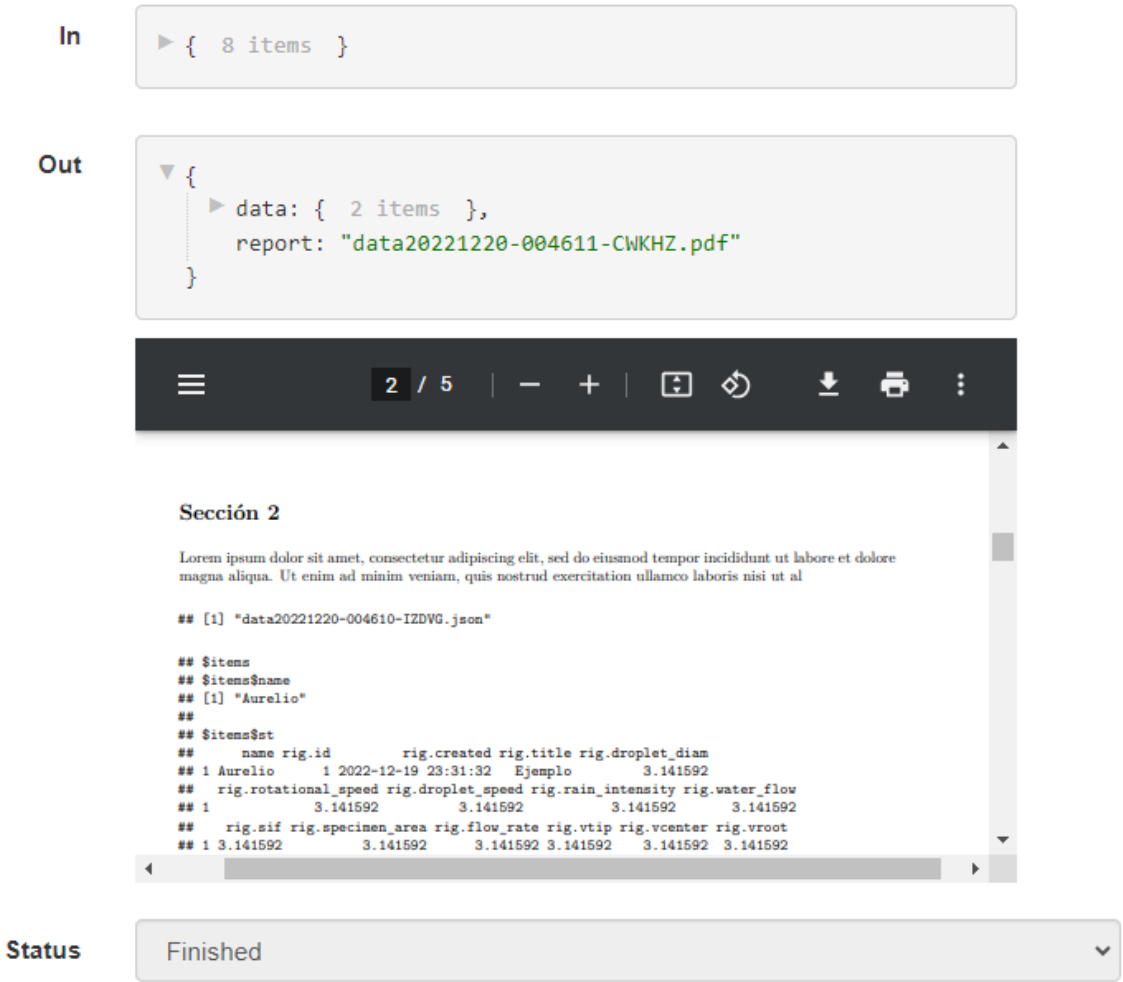

Figura 6.2: Captura de generación de informe en aplicación web

7. Migración a maquina en la nube

La migración se basa en trasladar el directorio “web” con los ficheros de la aplicación web al proveedor de nube, pero sin incluir el fichero *Dockerfile*. Esto se debe a que la aplicación web ya no se encuentra contenerizada.

Además, se copió el directorio “tool” a otro directorio dentro del proveedor de la nube. De esta manera, ya no se utiliza la red de contenedores, por lo que no es necesario generar los servicios que se encargaban de la contenerización de la aplicación web en el archivo *docker-compose.yml* mencionado en el Capítulo 5. Únicamente se establecería el servicio “tool” en el archivo *docker-compose.yml*.

Para aclarar la transferencia de los directorios, se puede observar la figura 5.1, en la cual se muestra la estructura del proyecto en el entorno local. En el proveedor de la nube, la organización del directorio sería similar, pero con los cambios mencionados anteriormente.

7.1. Conexión SSH

Este protocolo SSH está explicado con detalle en la sección 4.2.3. Con respecto al proyecto, cabe destacar que, para conectarse mediante conexión SSH, se ofreció la dirección IP de la máquina junto con un usuario y contraseña asociado. Una vez dentro, tan solo se requirió de utilizar el comando “su” para entrar en modo administrador. Además, para poder conectarse se debe establecer la IP pública[1] del dispositivo que se está conectando en la configuración de la máquina.

7.2. Integración de sistemas

Para llevar a cabo la migración de la aplicación web contenerizada junto con la herramienta computacional, fue necesario adoptar un enfoque diferente en el proyecto. En lugar de realizar integraciones entre contenedores, estas relaciones ahora se realizan entre el

[1]Dirección a la que se puede acceder directamente desde Internet y que su proveedor de servicios de Internet (ISP) asigna a su router de red [42].

contenedor de la herramienta computacional y la aplicación web que se encuentra alojada directamente en el proveedor en la nube. En este caso, dicho distribuidor seleccionado fue A3P, ya que cuenta con sus propias máquinas ubicadas en un centro de datos.

En este caso, como se expone en el capítulo 6, la integración de sistemas para visualizar los resultados de los cálculos junto con sus informes, sigue el mismo flujo que cuando se utiliza la red de contenedores. La diferencia radica en que ahora se han realizado modificaciones en las rutas y direcciones.

Para establecer las rutas, la aplicación web cuenta con un fichero de configuración "config.php". Las variables principales son "API_URL" y "APP_URL, las cuales se definen con la dirección IP de la máquina que aloja la red de contenedores.

En el extracto de código 7.1 se puede observar el contenido del archivo "config.php" en la aplicación de la máquina en la nube. En este caso, se establecen las rutas utilizando la dirección IP de esta y la URL de la página web en producción.

```
1 define("APP_URL","https://panel.beetool.es");
2 define ("API_URL","http://185.31.236.101:8000");
```

Extracto de código 7.1: Extracto de fichero de configuración de web de la maquina en la nube

Finalmente, la API que envía los datos del contenedor de la herramienta computacional a la página web se configura estableciendo la URL con la dirección de la página web en producción. Por lo tanto, la ruta completa sería la que se muestra en el extracto de código A.8, pero reemplazando la parte de la "IP" que anteriormente se obtenía por una variable de entorno con la URL de producción, en este caso "https://panel.beetool.es"

7.3. Base de datos

Previamente, se disponía de dos contenedores, uno para MySQL y otro para PhpMyAdmin. Sin embargo, en la situación actual, no es necesario utilizar ninguno de estos contenedores, ya que nos conectamos directamente a una base de datos alojada en la máquina a través de PhpMyAdmin. En este caso, simplemente se requiere establecer el nombre y la contraseña correspondientes en el archivo de configuración de la aplicación web. Esta configuración se puede ver en el extracto de código 7.2

```
1 \$db_config = array(
```

```
    'servername'    => 'localhost',
    'database'      => 'beetool',
    'username'      => 'beetool',
    'password'      => 'OFe2ViQ'
);
```

Extracto de código 7.2: Extracto de fichero de configuración de base de datos en la maquina en la nube.

7.4. Carpetas compartidas

Otra funcionalidad que cambió mucho fue las rutas de las carpetas compartidas, al desaparecer el contenedor de la interfaz web ahora desaparecían las carpetas compartidas. Por consiguiente, tan solo hacía falta llevar a cabo los volúmenes de la herramienta computacional, para poder replicar las carpetas generadas por el contenedor en el directorio local.

Posteriormente, la interfaz web alojada en la propia máquina tan solo tenía que buscar en sus propios directorios donde se generaban estos informes en formato pdf que posteriormente iban a ser visualizados en la propia aplicación web.

8. Aplicación Móvil

Una vez completado el desarrollo de la herramienta computacional y la interfaz web correspondiente, se plantea la utilización de una aplicación móvil con el propósito de facilitar la visualización de los informes de manera más accesible y clara.

Dicha aplicación móvil ya se encontraba previamente desarrollada por la empresa A3P, únicamente requería la implementación de la funcionalidad de visualización de informes y resultados de los cálculos, así como un sistema de inicio de sesión. Para todo su desarrollo se hizo uso de Monaca software explicado en detalle en la sección 4.1.6.

Es relevante destacar que, en la sección 2.2 se ha mencionado que la aplicación web se ha diseñado con la finalidad de que cada empresa que adquiera el producto del proyecto BEETool disponga de su propia aplicación web, alojada en su dominio exclusivo, lo que le permitirá almacenar sus propios datos, generar informes personalizados y tener acceso a su propia interfaz web y base de datos.

Además, se puede apreciar en la figura B.3, que la aplicación móvil incluye un sistema de inicio de sesión en el cual se debe especificar la empresa correspondiente, así como proporcionar un nombre de usuario y una contraseña. Estos datos de inicio de sesión han sido previamente establecidos en la interfaz web, específicamente en la sección de gestión de usuarios, desarrollada por José López Conejero.

Para la implementación de las nuevas funcionalidades llevadas a cabo se hizo uso de Template7. Esto es un motor de plantillas de JS (JavaScript) para dispositivos móvilles [43]. Asimismo, durante este proceso se utilizan funciones como la del extracto de código del inicio 8.1, en la cual se renderizan las plantillas pasándole unos valores u otros.

```
1 myApp.onPageInit('home', function (page) {
2       renderTemplate("home",{ existe_token: token.length>0 ,footer:
            FOOTER });
3 })
```

Extracto de código 8.1: JS que permite el inicio en la aplicación móvil

8.1. Inicio de sesión

El inicio de sesión se lleva a cabo gracias al registro que se realiza mediante la creación de usuarios dentro de cada interfaz web, como se puede observar en la figura B.1.

En vista de ello, como se puede apreciar en la figura B.2 se muestra la lista de usuarios de la aplicación web, estos usuarios serán los que posean la capacidad de iniciar sesión dentro de la aplicación móvil.

Siguiendo con lo mencionado, cabe destacar que, para el inicio de sesión a nivel de backend en la aplicación móvil se hizo uso del código A.9. Dicho inicio, permite que se maneje un evento con un "id" "#login-button, el cual necesita de un usuario, contraseña y el servidor. Una vez se le han introducido estos tres parámetros, se debe hacer una llamada POST. Llegados a este punto, el presente bloque, realiza una solicitud AJAX utilizando el método POST a una URL determinada y la URL se obtiene llamando a la función *getPublicApiUrl* con los valores de "memo.lagan" y "login". Los datos de usuario y contraseña se envían en el cuerpo de la solicitud. Con la variable "memo.lagan" se consigue obtener el servidor establecido para el inicio de sesión y usar la base de datos conveniente. Una vez se ha realizado el inicio, se guarda el token obtenido y el valor de "memo.lagan" en el almacenamiento local del navegador.

Respecto con el frontend, es sencillo puesto que se usa una variable "existe_token", la cual hace referencia a si existe el token. Si se ha generado como se ha indicado anteriormente en el local storage se pasará a poder ver los informes, si no se pasa al login tal como se puede apreciar en el extracto de código A.11. La variable "existe_token" se obtiene de renderizar la plantilla como se aprecia en el extracto anterior 8.1.

Finalmente, se adjunta la figura B.3, la cual es una captura de la aplicación, en la que se puede apreciar el inicio de sesión. También se muestra la figura B.4, la cual es una captura del inicio una vez se ha iniciado junto con el desplegable de la izquierda abierto. En esta pestaña de inicio informativa se encuentra información sobre el proyecto BEETool y distintas fotos que van cambiando con mensajes.

8.2. Visualización de tests

Para visualizar los tests se tiene en cuenta dos posibles métodos. Por una parte, poder ver el listado de los test, y por otra, y de manera más detallada entrar en cada uno de los test realizados para poder visualizar su informe pertinente y varios datos del test. Las dos implementaciones se han llevado a cabo siguiendo la manera citada anteriormente, como en el de la figura 8.1, se renderiza una plantilla HTML a la que se le pasan ciertos valores que luego se pueden utilizar en el fichero HTML.

8.2.1. Listado de tests

Para implementar el listado de test se realiza una solicitud AJAX utilizando el método GET a una URL determinada, como se aprecia en la figura A.10. Se envían cabeceras en la solicitud, incluyendo un encabezado de autorización con un token de acceso. La URL se obtiene llamando a la función *getPrivateApiUrl* con el argumento "testplan".

Por una parte, se asigna el resultado de la respuesta a la propiedad "testplans" en el objeto "memo" mediante "memo.testplans = result.data.items". Luego, se llama a la función *renderTemplate* para renderizar una plantilla llamada "testplans" con los datos obtenidos (memo.testplans, memo.usuario y *footer*[1]). La plantilla renderizada se utiliza para actualizar el contenido de la página "testplans".

Por otra parte, respecto a la plantilla, se puede apreciar en el extracto de código A.12, y en la figura B.5 que se muestra por pantalla el nombre del test, el id y el estado. Posteriormente, cuando se entre en cada test, se mostrarán más características.

8.2.2. Visualizar un test

Si se quiere visualizar un test en concreto con su informe y sus resultados, solo hará falta seleccionar el que se desee en el listado. Este renderiza una plantilla a la cual se le han introducido más valores, puesto que se visualizan más características del test como se puede apreciar en la figura B.6.

En esta plantilla se puede ver el título del test, su id y el estado. Además, se observa el JSON que se recibe de vuelta por parte de la herramienta computacional junto con el informe

[1]Sección ubicada al final de la página que suele contener información adicional y enlaces relevantes.

y un apartado comentarios.

La plantilla se puede comprobar en el extracto de código A.13, en la cual se puede observar como se entra dentro del item con "#with item" y se va creando con puro HTML los valores que se quieren siempre utilizando {{}}. Por otra parte, como anteriormente, se obtiene el nombre del informe y se le añade el nombre del informe a la ruta donde se tienen almacenados los informes generados por la aplicación web.

8.3. Otras funciones

Otras de las funcionalidades a destacar, las cuales se han llevado a cabo siguiendo la misma lógica, se verán a continuación:

- Formulario de contacto, ver en figura B.7.
- Visualizar datos del usuario, ver en figura B.8.
- Olvido de contraseña, ver en figura B.9.
- Pestaña de About, ver en figura B.10.
- Pestaña de acceso denegado.

Parte IV

Pruebas Realizadas y Resultados

9. Pruebas Realizadas

Respecto con las pruebas realizadas caben destacar distintas dependiendo de cada tarea desarrollada.

9.1. Contenerización

Durante el periodo de pruebas, el proceso de contenerización no recibió una atención exhaustiva. Esto se debe a que las pruebas para generar los archivos de configuración de Docker, como *Dockerfile*, *docker-compose* y *makefile*, forman parte del proceso de contenerización y no requieren pruebas separadas para demostrar su funcionalidad. Si la página web funciona correctamente, indica que todo está correctamente contenerizado.

Es importante resaltar que el proceso de contenerización se llevó a cabo de manera gradual, implementando cada una de las dependencias necesarias para el correcto funcionamiento de la interfaz web. Sin embargo, surgieron algunos desafíos al contenerizar la herramienta computacional, debido a las diversas bibliotecas disponibles para ejecutar archivos RMarkdown y las diferentes imágenes. Finalmente, se decidió utilizar una imagen base de Ubuntu e instalar lo necesario para superar estos obstáculos.

9.2. Integración de sistemas

En el ámbito de la integración de sistemas, se realizaron pruebas utilizando el software Postman, el cual se describe en detalle en la sección 4.1.5. Con Postman, se estableció el método deseado, en este caso POST, se agregó autenticación básica y se establecieron los parámetros deseados, como la “id” y el “type” mencionados anteriormente en la sección 6.1.6.

Además, como se puede observar en el extracto de código, se implementó una función llamada *beetool_log* en PHP que imprime en un archivo lo que se está recibiendo o enviando, dependiendo de dónde se coloque la siguiente línea de código:“beetool_log(print_r($input, true))”. Llegados a este punto, cabe señalar que, se está agregando el JSON recibido desde la

interfaz web de la herramienta computacional al archivo. Para comprender la línea anterior, se puede consultar el extracto de código A.5.

9.3. Migración a la nube

Para el desarrollo de la migración al servidor se utilizó en el método de "prueba y error". Este método es una técnica de resolución de problemas que implica probar diferentes soluciones o enfoques hasta encontrar una que funcione correctamente.

En lugar de intentar planificar y diseñar una solución perfecta desde el principio, se ajusta continuamente el código a medida que se descubren y corrigen errores.

Al utilizar esta técnica, se pueden experimentar diferentes enfoques, ajustar parámetros y verificar los resultados obtenidos. Si una solución no funciona como se espera, se realizan cambios y se vuelve a probar hasta encontrar una solución adecuada.

9.4. Aplicación móvil

Se basó en el método de "prueba y error" anteriormente mencionado en la sección anterior, debido a que a penas surgieron problemas, puesto que ya estaba el código bastante desarrollado, solo había que modificar funcionalidades.

10. Resultados

Dentro de los resultados obtenidos en este trabajo, destaca la implementación exitosa de un emulador local mediante el proceso de contenerización en el proyecto BEETool. Este emulador permite crear un entorno local independiente que facilita el desarrollo de nuevas funcionalidades sin tener que realizar modificaciones en el entorno real de la aplicación (es decir, la máquina en la nube).

Por un lado, a través del proceso de integración, se logró establecer una comunicación fluida entre la interfaz web y la herramienta computacional. Esto incluye la visualización de informes y la presentación de resultados en formato JSON, generados por los cálculos realizados por la herramienta computacional.

Por otro lado, la migración de los desarrollos realizados en el emulador local a la máquina en la nube permitió establecer una interfaz de producción en un entorno real. Esto implicó integrar de manera efectiva las dos partes del proyecto, es decir, la herramienta computacional y la aplicación web.

Finalmente, se logró implementar una aplicación móvil que amplía la funcionalidad del sistema. Esto incluyó el desarrollo necesario para permitir el inicio de sesión basado en los usuarios registrados en la aplicación web, así como la conexión con el panel web correspondiente. Con esta aplicación móvil, se logra una experiencia más sencilla, intuitiva y dinámica para visualizar los resultados de los tests, lo que beneficia tanto a los usuarios individuales como a las empresas que utilizan el sistema.

En resumen, los resultados obtenidos en este trabajo han permitido fortalecer el proyecto BEETool, brindando un emulador local, una sólida integración entre la interfaz web y la herramienta computacional, una migración exitosa a un entorno real en la nube, y una aplicación móvil que mejora la accesibilidad y usabilidad del sistema. Estos avances contribuyen a hacer de BEETool una solución más completa y eficiente para los usuarios.

Parte V

Epílogo

11. Conclusiones

En este último capítulo se detallan las lecciones aprendidas tras el desarrollo del presente proyecto y se identifican las posibles oportunidades de mejora sobre el software desarrollado.

11.1. Objetivos alcanzados

Entre los objetivos alcanzados se encuentra el empaquetado exitoso de la interfaz web en un contenedor para garantizar su correcto funcionamiento en un entorno local, así como la contenerización lograda de la herramienta computacional. Además, se logró la integración efectiva de ambos sistemas y la migración fructífera de la herramienta a una máquina alojada en la nube. Por último, se cumplió con el objetivo de desarrollar una aplicación móvil.

Además, uno de los objetivos planteados fue implementar una gestión de usuarios dentro de la aplicación web. No obstante, este objetivo fue logrado en el proyecto BEETool, su ejecución no se llevó a cabo únicamente por José Antonio Téllez, ya que requería la toma de decisiones por parte de la empresa A3P. En consecuencia, este objetivo se desarrolló en colaboración conjunta.

11.2. Lecciones aprendidas

A continuación, se detallan las buenas prácticas adquiridas, tanto tecnológicas como procedimentales, así como cualquier otro aspecto de interés.

Durante el desarrollo del proyecto se ha aprendido a trabajar en *pair programming*, con una metodología basada en objetivos y metas, promoviendo así la participación activa.

Se ha familiarizado con distintas tecnologías, principalmente Docker, actualmente se tiene la capacidad de contenerizar cualquier entorno, además de realizar integraciones entre varios contenedores utilizando las redes de contenedores y los volúmenes.

Se ha trabajado en WSL, en consecuencia, se han adquirido los conocimientos pertinentes para utilizar este entorno de Linux.

Ahora, se posee la capacidad de llevar un control de versiones con GitHub trabajando con varios integrantes y llevando de esta forma el proyecto colaborativo adecuadamente.

Para el desarrollo de software se ha perfeccionado el uso del editor de código Visual Studio Code, el cual cuenta con múltiples extensiones que han sido de verdadera utilidad durante el transcurso de este proyecto. De este modo, se ha ampliado el saber de los protocolos HTTP, JSON o SSH, los cuales han sido cruciales en el funcionamiento de esta aplicación web, y este último en la migración a la máquina en la nube.

Finalmente, respeto a las tecnologías, cabe destacar el uso de Monaca para el desarrollo de la aplicación móvil, y Postman el cual ya se dominan todas sus funcionalidades para la correcta implementación de APIs.

11.3. Trabajo futuro

El proyecto BEETool es un producto de software que tiene como objetivo principal ayudar a las empresas a mejorar la eficiencia y productividad en la gestión de la cadena de suministro. Dado esto, existen varias posibles direcciones en las que el proyecto podría evolucionar en el futuro. Algunas ideas para posibles trabajos futuros del proyecto podrían incluir:

- Integración con sistemas de inteligencia artificial y aprendizaje automático: Se podrían incorporar capacidades avanzadas de inteligencia artificial y aprendizaje automático para ofrecer análisis predictivos y recomendaciones más precisas.
- Incorporación de tecnologías emergentes: Se podría explorar la integración de tecnologías emergentes como el IoT (Internet de las cosas).
- Mejora de la interfaz de usuario y experiencia de usuario: Se podría centrar en optimizar la usabilidad y la experiencia del usuario, ofreciendo una interfaz más intuitiva y fácil de usar, así como una mayor personalización y adaptabilidad a las necesidades específicas de cada empresa.

Estas son solo algunas ideas y posibilidades para el trabajo futuro del proyecto. La dirección exacta dependerá de las necesidades del mercado y las demandas de los clientes, así como de las capacidades y objetivos del equipo de desarrollo.

Bibliografía

[1] Brande. *El desafío de combatir la erosión de la lluvia*. 2020. URL: https://www.siemensgamesa.com/es-es/descubrir/revista/2020/04/siemens-gamesa-rain-erosion (visitado 28-06-2023).

[2] CEU Universidad Cardenal Herrera. *Innovación para mejorar la eficiencia de la energía eólica*. 2021. URL: https://medios.uchceu.es/actualidad-ceu/innovacion-para-mejorar-la-eficiencia-de-la-energia-eolica/ (visitado 28-06-2023).

[3] Aws Amazon. *¿Qué es Docker?* 2023. URL: https://aws.amazon.com/es/docker/ (visitado 28-06-2023).

[4] Naciones Unidas. *Objetivos de desarrollo sostenible*. 2020. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ (visitado 12-07-2023).

[5] Lia Schüler. *Público objetivo, cliente ideal y buyer persona: ¿cuáles son las diferencias?* 2020. URL: https://www.rdstation.com/blog/es/publico-objetivo-cliente-ideal-buyer-persona/#:~:text=El%20P%C3%BAblico%20objetivo%20o%20target,o%20servicio%20de%20la%20empresa. (visitado 01-07-2023).

[6] Anónimo. *¿Qué es un dominio y cómo funciona?* 2020. URL: https://www.webempresa.com/hosting/que-es-dominio.html (visitado 09-07-2023).

[7] Scrum Manager BoK. *Programación en pareja*. 2021. URL: https://www.scrummanager.com/bok/index.php/Programacil%C3%B3n_en_pareja#:~:text=La%20programaci%C3%B3n%20en%20pareja%20(pair,otro%20(observador)%20lo%20supervisa. (visitado 28-06-2023).

[8] Anónimo. *Trello facilita a los equipos la gestión de proyectos y tareas*. 2023. URL: https://trello.com/es/tour (visitado 28-06-2023).

[9] Asad Ali. *Docker Desktop: la forma más sencilla de contener aplicaciones*. 2022. URL: https://geekflare.com/es/docker-desktop/#:~:text=Docker%20Desktop%20es%20una%20aplicaci%C3%B3n,o%20servicios%20acoplados%20%2F%20en%20contenedores. (visitado 02-07-2023).

[10] Mikel Goig. *Arquitectura de Docker*. 2023. URL: https://docs.mikelgoig.com/docker/arquitectura-de-docker.html#imagenes (visitado 02-07-2023).

[11] Anónimo. *Volúmenes de Docker*. 2023. URL: https://docs.aws.amazon.com/es_es/AmazonECS/latest/bestpracticesguide/storage-dockervolumes.html (visitado 02-07-2023).

[12] Anónimo. *Qué es Windows*. 2020. URL: https://www.arimetrics.com/glosario-digital/windows (visitado 02-07-2023).

[13] Juan Antonio Soto. *¿Qué es Linux y para qué sirve?* 2020. URL: https://www.geeknetic.es/Linux/que-es-y-para-que-sirve (visitado 02-07-2023).

[14] Varios colaboradores. *¿Qué es el Subsistema de Windows para Linux?* 2023. URL: https://learn.microsoft.com/es-es/windows/wsl/about (visitado 02-07-2023).

[15] Juan Antonio Pascual Estapé. *Qué es una distribución Linux, en qué se diferencian y cómo elegir una*. 2017. URL: https://computerhoy.com/noticias/software/que-es-distribucion-linux-que-diferencian-como-elegir-54784 (visitado 02-07-2023).

[16] Yúbal Fernández. *Qué es Github y qué es lo que le ofrece a los desarrolladores*. 2019. URL: https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores (visitado 02-07-2023).

[17] TerryGLee. *¿En qué consiste la extensión de GitHub Copilot para Visual Studio?* 2023. URL: https://learn.microsoft.com/es-es/visualstudio/ide/visual-studio-github-copilot-extension?view=vs-2022 (visitado 02-07-2023).

[18] Anónimo. *Docker in Visual Studio Code*. 2020. URL: https://code.visualstudio.com/docs/containers/overview (visitado 02-07-2023).

[19] Yúbal Fernández. *API: qué es y para qué sirve*. 2019. URL: https://www.xataka.com/basics/api-que-sirve (visitado 01-07-2023).

[20] José Manuel Rosa Moncayo. *Qué es REST: Conoce su potencia*. 2018. URL: https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/ (visitado 04-07-2023).

[21] Yanina Muradas. *Qué es Postman y primeros pasos*. 2019. URL: https://openwebinars.net/blog/que-es-postman/ (visitado 02-07-2023).

[22] Anónimo. *Mobile App Development Fast, Easy and Flexible*. 2019. URL: https://monaca.io/ (visitado 03-07-2023).

[23] Deyimar A. *¿Qué es JSON?* 2023. URL: https://www.hostinger.es/tutoriales/que-es-json (visitado 03-07-2023).

[24] KeepCoding Team. *¿Qué es el protocolo HTTP?* 2023. URL: https://keepcoding.io/blog/que-es-el-protocolo-http/ (visitado 03-07-2023).

[25] Anónimo. *¿Qué es y para qué sirve el protocolo SSH?* 2020. URL: https://linube.com/blog/que-es-protocolo-ssh/#:~:text=El%20protocolo%20SSH%20o%20Secure,seguras%20entre%20los%20dos%20sistemas. (visitado 03-07-2023).

[26] Iván Ramírez. *Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas*. 2020. URL: https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas (visitado 11-07-2023).

[27] KeepCoding Team. *¿Qué es y para qué sirve el fichero YAML?* 2023. URL: https://keepcoding.io/blog/que-es-y-para-que-sirve-el-fichero-yaml/ (visitado 04-07-2023).

[28] diberry. *Uso de Docker Compose para implementar varios contenedores*. 2023. URL: https://learn.microsoft.com/es-es/azure/cognitive-services/containers/docker-compose-recipe (visitado 04-07-2023).

[29] Anónimo. *3. Ficheros makefile*. 2020. URL: http://www.it.uc3m.es/~pedmume/asignaturas/2005/LAO/Lab2/tutorial4/node4.html#:~:text=Un%20fichero%20makefile%20contiene%20las,las%20l%C3%ADneas%20b%C3%A1sicas%20son%20similares. (visitado 04-07-2023).

[30] Marcos Viera. «Instructivo del comando Make». En: (2013), pág. 2. URL: https://eva.fing.edu.uy/pluginfile.php/210228/mod_resource/content/0/Instructivo_Makefile.pdf#:~:text=La%20utilidad%20make%20determina%20autom%C3%A1ticamente,mantener%20el%20ejecutable%20totalmente%20actualizado. (visitado 04-07-2023).

[31] Anónimo. *Oracle, el padre de las empresas de software en el mundo*. 2020. URL: https://tentulogo.com/oracle-padre-las-empresas-software-mundo/ (visitado 05-07-2023).

[32] Pablo Londoño. *Qué es MySQL, para qué sirve y características principales*. 2020. URL: https://blog.hubspot.es/website/que-es-mysql (visitado 05-07-2023).

[33] Redacción KeepCoding. *¿Qué es una función hash?* 2023. URL: https://keepcoding.io/blog/que-es-una-funcion-hash/#:~:text=Una%20funci%C3%B3n%20hash%20es%20un,que%20se%20tenga%20la%20clave). (visitado 07-07-2023).

[34] Fernán García de Zúñiga. *¿Qué es phpMyAdmin y cómo usarlo?* 2021. URL: https://www.arsys.es/blog/phpmyadmin (visitado 05-07-2023).

[35] Ivan de Souza. *Descubre qué es el lenguaje de programación PHP y en qué situaciones se hace útil*. 2020. URL: https://rockcontent.com/es/blog/php/ (visitado 05-07-2023).

[36] Anónimo. *warnings — Control de advertencias*. 2020. URL: https://docs.python.org/es/3/library/warnings.html#:~:text=Los%20mensajes%20de%20advertencia%20suelen,y%20se%20termine%20el%20programa. (visitado 05-07-2023).

[37] Arimetrics. *Qué es Script*. 2020. URL: https://www.arimetrics.com/glosario-digital/script (visitado 06-07-2023).

[38] Datademia. *¿Qué es R?* 2020. URL: https://datademia.es/blog/que-es-r5 (visitado 05-07-2023).

[39] Sky One Solutions. *Systems integration: learn how it works, main types and their importance*. 2022. URL: https://skyone.solutions/en/hub/systems-integration/#:~:text=La%20integraci%C3%B3n%20de%20sistemas%20es,tiempo%20dedicado%20a%20las%20operaciones. (visitado 06-07-2023).

[40] Gustavo B. *¿Qué es AJAX y cómo funciona?* 2023. URL: https://www.hostinger.es/tutoriales/que-es-ajax (visitado 06-07-2023).

[41] Gal EI AI. *cURL: ¿qué es y cómo puede usarlo para raspado de datos??* 2020. URL: https://brightdata.es/blog/procedimientos/como-usar-curl-para-web-scraping#:~:text=cURL%20es%20una%20herramienta%20basada,datos%20desde%20y%20hacia%20servidores. (visitado 06-07-2023).

[42] Anónimo. *¿Qué es una dirección IP pública?* 2020. URL: https://www.avast.com/es-es/c-ip-address-public-vs-private#:~:text=Una%20direcci%C3%B3n%20IP%20p%C3%BAblica%20es,la%20IP%20p%C3%BAblica%20del%20router. (visitado 10-07-2023).

[43] Anónimo. *¿Qué es un dominio y cómo funciona?* 2020. URL: https://v5.framework7.io/docs/template7 (visitado 10-07-2023).

Parte VI

Anexos

A. Código

Si el objetivo es examinar el código completo del proyecto, se recomienda acceder al repositorio de GitHub correspondiente en el siguiente enlace: https://github.com/tellez11/beetool_php. En dicho repositorio, se encuentra disponible el desarrollo completo del proyecto, incluyendo su estructura detallada, lo cual permite una revisión exhaustiva de la implementación"

A continuación, se recogen extractos de código del proyecto BEETool. A lo largo de este TFG se han hecho referencias explícitas a estos extractos de código, facilitando de esta manera comprender el funcionamiento, desarrollo, implementación e integración del proyecto.

A.1. Ficheros Dockerfile

```
FROM php:7.2-apache
WORKDIR /var/www/html

RUN a2enmod rewrite \
    && apt-get update \
    && apt-get install -y libpng-dev libzip-dev curl tar\
    && rm -rf /var/lib/apt/lists/* \
    && docker-php-ext-install zip pdo_mysql

COPY apache2.conf /etc/apache2/apache2.conf

COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
COPY composer.json /var/www/html/composer.json
COPY composer.lock /var/www/html/composer.lock
RUN composer install --ignore-platform-reqs --no-scripts

RUN curl -L https://redbeanphp.com/downloadredbeanversion.php?f=all
   -drivers -o redbean.tar.gz \
```

```
18      && tar xvzf redbean.tar.gz -C /var/www/html/vendor/gabordemooij
           /redbean/ \
19      && rm redbean.tar.gz
```

Extracto de código A.1: Fichero Dockerfile de contenedor PHP.

```
1  FROM ubuntu:22.04
2
3  ENV DEBIAN_FRONTEND noninteractive
4
5  WORKDIR /home/beetool
6
7  RUN apt-get update && \
8      apt-get install -y \
9      r-cran-rmarkdown \
10     r-cran-plumber && \
11     rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
12
13 ADD requirements-r.R /home/beetool
14 RUN Rscript requirements-r.R && \
15     rm -rf /tmp/downloaded_packages/ /tmp/*.rds
16 RUN Rscript -e 'tinytex::install_tinytex()'
17
18 RUN chmod -R 777 /home/beetool
19
20 ENTRYPOINT ["R", "-e", "source('start-API.R')"]
```

Extracto de código A.2: Fichero Dockerfile de contenedor R.

A.2. Aplicación web

```
1  function enviar_post($ruta, $Datos){
2      $ch = curl_init();
3      curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
4      curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
5      curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
```

```
    curl_setopt($ch, CURLOPT_URL,API_URL.$ruta);
    curl_setopt($ch, CURLOPT_USERPWD, USER.":".PWD);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode(array("items"
       =>$Datos)));
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-type:
       application/json'));
    curl_setopt($ch, CURLOPT_VERBOSE, true);

    $data = curl_exec($ch);

    if (empty($data)){
        $data =curl_error($ch);
        return $data.'***';
    }
        curl_close($ch);
        return json_decode(\$data);
}
```

Extracto de código A.3: Función encargada de enviar el JSON a la herramienta.

```
$this->post('/enviar_test_simulado', function ($request, $response,
    $args) {
    $input = $request->getParsedBody();
    $id = $input["id"];
    $type = $input["type"];
    if ($type == "testplan") {
        $testplan = R::findOne('testplan', 'id =?',[$id]);
        $simulatedtest_id = R::find('simulatedtest_testplan', '
           testplan_id =?',[$id]);
    }
    $env_json = array(
        'name'=> $testplan_name,
        'st'=>$simulatedtest_array,
        'et'=>$testedspecimen_array,
        'config'=>$config,
```

```
        'comments'=>$comments,
        'date' => date('l jS \of F Y h:i:s A'),
        'type' => $type,
        'id' => $id
    );
    $respuesta_json=enviar_post("/report", $env_json);
    return setResponse($response, 200, false, respuesta_json);

});
```

Extracto de código A.4: Metodo POST que realiza creación y envío del JSON a la herramienta.

```
$this->post('/recibir_resultados/{type}/{id}', function ($request,
   $response, $args) {
        $input = $request->getParams();
        $type = $args['type'];
        $id = $args['id'];
        return setResponse(response, 200, false, id);
});
```

Extracto de código A.5: Metodo POST que permite recibir resultados en la web.

A.3. Herramienta Computacional

```
library(R.utils)
library(jsonlite)

#* @parser json
#* @post /report
#* @serializer json list(na="string")
function(req, res) {
    jsonData <- toJSON(req$body)
    filename <- paste0(
                "/home/beetool/tool/reports/json/data",
                format(Sys.time(), "%Y%m%d-%H%M%S-"),
                paste0(sample(LETTERS, 5, TRUE), collapse = ""),
```

```
                ".json")
    write(jsonData, file = filename)

    system_call <- paste0("Rscript test.R ", filename, " &")
    system(system_call)

    paste0("El contenedor: ", System$getHostname(), "ha recibido
       el json y está calculando los datos.")
}
```

Extracto de código A.6: Fichero plumber.R que permite la recepción del JSON

```
library(knitr)
library(rmarkdown)
library(R.utils)
library("httr")
library(jsonlite)

filename_pdf <- paste0(
  "data",
  format(Sys.time(), "%Y%m%d-%H%M%S-"),
  paste0(sample(LETTERS, 5, TRUE), collapse = ""),
  ".pdf"
)

rmarkdown::render("test.Rmd", output_file = filename_pdf,
   output_dir = "/home/beetool/tool/reports/pdf/")
```

Extracto de código A.7: Parte del fichero test.R que genera el informe.

```

args <- commandArgs(trailingOnly=TRUE)
filename_json <- args[1]

IP <- Sys.getenv("IP")
json <- fromJSON(filename_json)
id <- json$items$id
id <- as.character(id)
```

```
type <- json$items$type
url <- paste0("http:/",IP,":800/public/public_api/
    recibir_resultados/",type,"/",id)
json <- paste0('{"data":{"a":1,"b":2},"report":"', filename_pdf,
   '"}')
mypdf <- fromJSON(json)

httr::POST(url, body = mypdf, encode = "json", verbose())
```

Extracto de código A.8: Parte del fichero test.R que genera el JSON con los resultados.

A.4. Aplicación Móvil

```
\$('#login-button').click(function(){
  login_user = \$('#login-user').val();
  login_pass = \$('#login-password').val();
  memo.lagan = \$('#login-cliente').val();

  if(login_user && login_pass  && memo.lagan)
        \$.ajax({
          type: "POST",
          url: getPublicApiUrl(memo.lagan,"login"),

          data: {
            user: login_user,
            pass: login_pass
          },
          dataType: 'json'
        })
        .done(function(response){
          if(response.message=="Ok")
           token = response.data.jwt;
           window.localStorage.setItem('beetool_token',token);
           window.localStorage.setItem('beetool_lagan',memo.lagan);
          myApp.closeModal();
          if(token){
```

```
            \$(".open-login-screen").hide();
          }
          mainView.history=[];
          mainView.router.loadPage("mis-datos.html");
          \$("#menu-cuenta").show();
          \$(".menu-cliente").slideUp();
          \$(".menu-cuenta").slideDown();
          cargar_usuario();
        })
});
```

Extracto de código A.9: Función en JS encargada de iniciar sesión en aplicación móvil.

```
myApp.onPageInit('testplans', function (page) {
  if(!memo.testplans.length)
        \$.ajax({
          type: "GET",
          headers: {
            Accept: "application/json",
            Authorization: "Bearer " + token
          },
          url: getPrivateApiUrl("testplan"),

          dataType: 'json'
        })
        .done(function(result){
          memo.testplans=result.data.items;
          renderTemplate("testplans",{
            usuario:memo.usuario,
            items: Object.values(memo.testplans),
            footer:FOOTER
          });
        });
  else renderTemplate("testplans",{
    paciente:memo.paciente,
    items: Object.values(memo.testplans),
    footer:FOOTER
```

```
24    });
25 })
```

Extracto de código A.10: Función de JS encargada de obtener los testplans.

```
{{#if existe_token}}
<div class="standard-row" style="padding-top:40px;">
    <a class="button␣open-testplans">VIEW TEST PLANS</a>
</div>
{{else }}
<div class="standard-row" style="padding-top:40px;">
    <a class="button␣open-login-screen">USER LOGIN</a>
</div>
{{/if}}
```

Extracto de código A.11: HTML de inicio de sesión en frontend.

```
<script type="text/template" id="testplans">
    <div class="standard-row2␣ft-area">
        <div class="list-block">
            {{#items}}
            <ul>
            <a href="testplan.html?id={{@index}}">
            <li class="item-content">
              <div class="item-media"><i class="pe-7s-graph3␣on␣pe
                 -7s-monitor"></i></div>
              <div class="item-inner">
                <div class="item-title"><strong>{{title}}</strong>
                   - {{id}}</div>
                <div class="item-after">{{estado}}</div>
              </div>
            </li>
            </a>
            </ul>
            {{/items}}
        </div>
    </div>
        <div class="footer-row">{{footer}}</div>
```

```
</script>
```

Extracto de código A.12: HTML del listado de testplans.

```
{{#with item}}

<div style="margin-top: 20px">
    <div class="pull-left" style="padding-right: 20px; border-right
       : 1px solid #000">
        <div style="font-weight: bold; text-align: right">Title</
           div>
        <div style="text-align: right; margin-bottom: 10px">{{title
           }}</div>
        <div style="font-weight: bold; text-align: right">ID</div>
        <div style="text-align: right">{{id}}</div>
    </div>
    <div class="pull-left" style="padding-left: 20px">
        <div style="font-weight: bold; text-align: left">Status</
           div>
        <div style="text-align: left; margin-bottom: 10px">{{estado
           }}</div>
    </div>

{{/with}}

{{#if name_report }}
    <div class="pull-right bloque" style="padding-left: 20px">
        <div style="font-weight: bold">Data</div>
        <div style="text-align: left">{{data}}</div>
    </div>
    <div style="clear: both"></div>
</div>

<div class="summary">
    <div> PDF Report </div>
    <div><iframe src="https://panel.beetool.es/uploads/informes/{{
       name_report}}" type="application/pdf" width="100%" height="
```

```
        400px" style="border:␣none"></iframe></div>
</div>
{{/if}}

{{#with item}}
<div class="pull-right␣bloque" style="padding-left:␣20px">
        <div style="font-weight:␣bold;␣text-align:␣left">Comments</
            div>
        <div style="text-align:␣left">{{comments}}</div>
</div>



{{/with}}
```

Extracto de código A.13: HTML de testplan.

B. Maquetas

En este apartado de anexos, se incluyen maquetas que representan visualmente la estructura y diseño de la interfaz de la aplicación, proporcionando una vista previa de cómo se vería. Estas maquetas son un recurso valioso para comprender la propuesta de diseño y la experiencia del usuario que se ha buscado lograr. A través de estas representaciones gráficas, se pretende ofrecer una visión más completa y tangible de la aplicación desarrollada.

B.1. Aplicación web

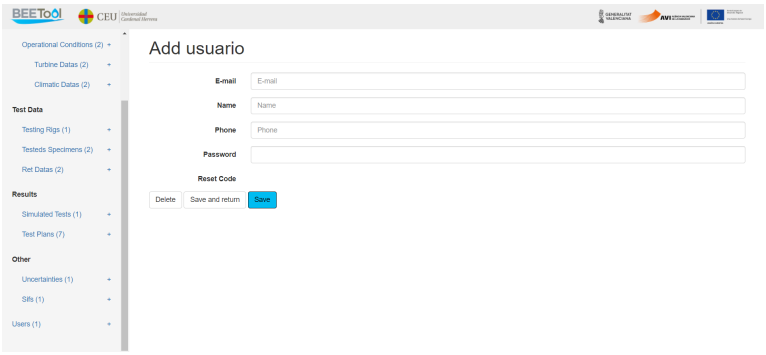

Figura B.1: Captura de creación usuarios.

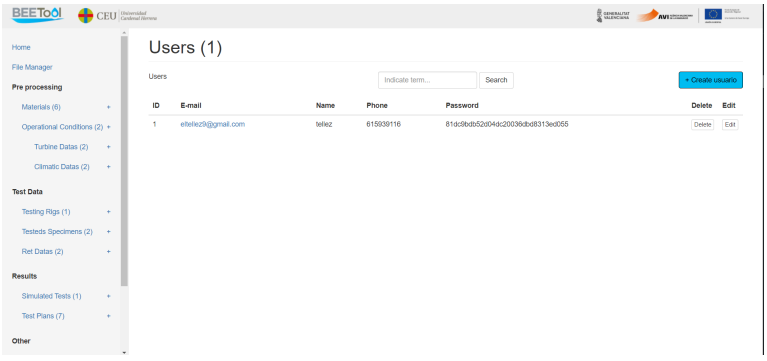

Figura B.2: Captura de lista usuarios.

B.2. Aplicación Móvil

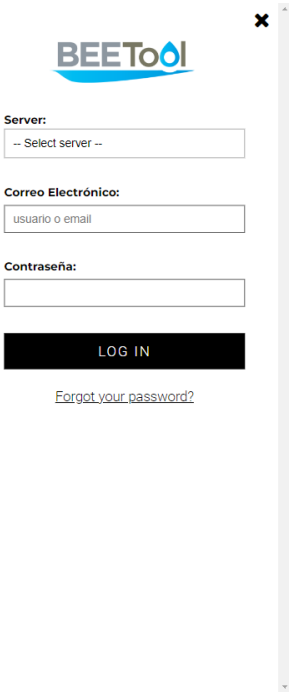

Figura B.3: Captura de inicio de sesión.

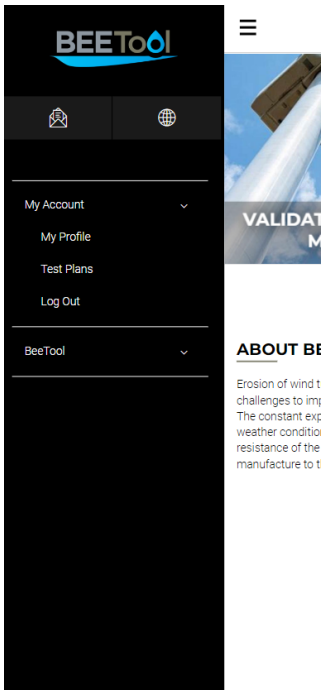

Figura B.4: Captura de sesión iniciada.

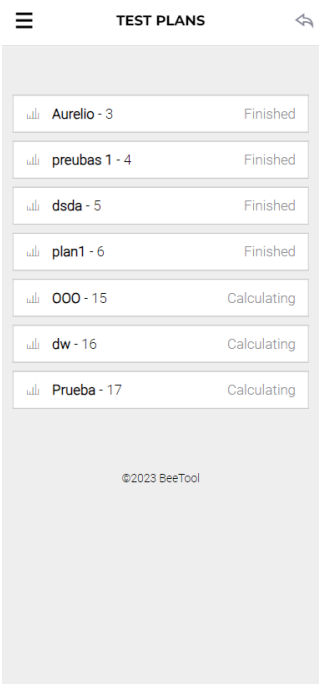

Figura B.5: Captura de listado de testplan.

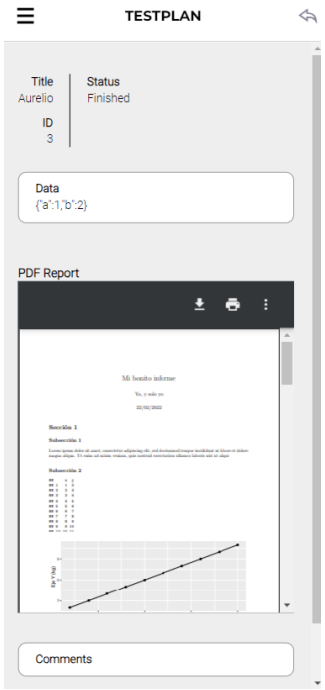

Figura B.6: Captura de testplan.

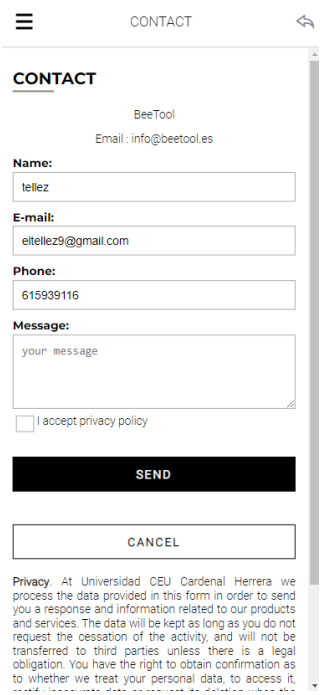

Figura B.7: Captura de pestaña formulario de contacto.

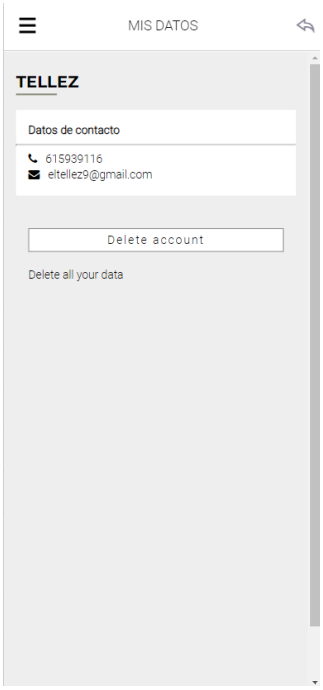

Figura B.8: Captura de datos de perfil de usuario iniciado.

Figura B.9: Captura de pestaña recuperación de contraseña.

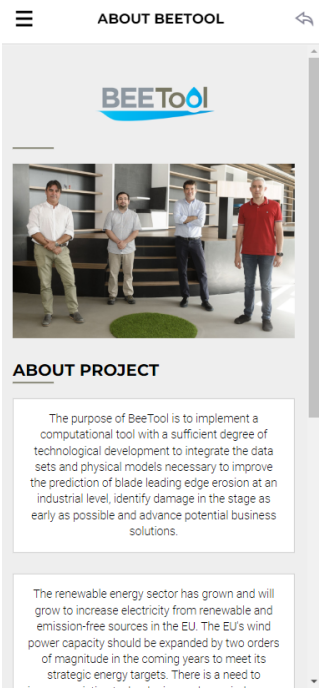

Figura B.10: Captura de pestaña about BEETool.