Memoria

Herramienta visual para la gestión y análisis de la carga docente

Trabajo de Fin de Grado

Grado en Ingeniería Informática

VNiVERSiDAD D SALAMANCA

Septiembre de 2022

Autor

Álvaro Martín Martín

Tutor

Rodrigo Santamaría Vicente

Certificado del tutor

D. Rodrigo Santamaría Vicente, profesor titular del Departamento de Informática y Automática de la Universidad de Salamanca.

Hace constar:

Que el trabajo titulado “Herramienta visual para la gestión y análisis de la carga docente” ha sido realizado por D. Álvaro Martín Martín, con DNI 70830456R y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad.

En Salamanca, a 6 septiembre de 2022

Firmado por SANTAMARIA VICENTE RODRIGO - 70879303L el día 06/09/2022 con un certificado emitido por AC FNMT Usuarios

Resumen

Este documento describe el desarrollo del proyecto titulado "**Herramienta visual para la gestión y análisis de la carga docente**" del Trabajo de Fin de Grado en Ingeniería Informática.

La aplicación tiene una arquitectura cliente-servidor, donde el cliente proporciona una interfaz que permite al usuario solicitar servicios al servidor, programa que maneja las peticiones y devuelve una respuesta para que el cliente muestre los resultados.
El lado del cliente ha sido desarrollado con VueJS, mientras que el lado del servidor con Spring Boot y toda la aplicación ha sido desplegada en producción en AWS.

El objetivo principal es que el usuario pueda realizar búsquedas sobre profesores, titulaciones y asignaturas mediante filtros, ordenaciones y paginaciones; estos datos se podrán visualizar en unas tablas acompañadas de gráficos.

Esta herramienta podría facilitar las labores de gestión del equipo de dirección en la planificación docente de los cursos en la parte relativa a la carga docente del profesorado, además de permitir al profesor examinar fácilmente su carga docente y asignaturas asociadas de una manera visual en un navegador web.

Se ha utilizado una metodología de desarrollo ágil para trabajar de manera más eficiente y rápida, consiguiendo una aplicación con mayor flexibilidad e inmediatez.

Palabras clave

Aplicación web, API, Asignaturas, Profesores, Titulaciones, VueJS, Spring Boot, AWS

Abstract

This document describes the development of the project entitled "**Visual tool for management and analysis of course load**" of the Final Degree Project in Computer Engineering.

The application has a client-server architecture, where the client provides an interface that allows the user to request services from the server, program that handles the requests and returns a response for the client to display the results.
The client side has been developed with VueJS, while the server side with Spring Boot and the whole application has been deployed in production on AWS.

The main objective is to allow the user to search for professors, degrees and subjects by means of filters, sorting and pagination; this data can be displayed in tables accompanied by graphs.

This tool could facilitate the management team's management tasks in the teaching planning of the courses in the part related to the teaching load of the teaching staff, in addition to allowing the teacher to easily examine his teaching load and associated subjects in a visual way in a web browser.

An agile development methodology has been used to work more efficiently and quickly, achieving an application with greater flexibility and immediacy.

Palabras clave

Web application, API, Subjects, Professors, Degrees, VueJS, Spring Boot, AWS

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Lista de abreviaturas

- **API:** Application Programming Interface
- **AWS:** Amazon Web Services
- **BBDD:** Base de Datos
- **CORS:** Cross-Origin Resource Sharing
- **CPU:** Central Processing Unit
- **CSS:** Cascading Style Sheets
- **EC2:** Elastic Compute Cloud
- **HTML:** HyperText Markup Language
- **HTTP:** HyperText Transfer Protocol
- **HTTPS:** HyperText Transfer Protocol Secure
- **IoT:** Internet Of Things
- **IP:** Internet Protocol
- **JS:** JavaScript
- **JSON:** JavaScript Object Notation
- **MVC:** Modelo-Vista-Controlador
- **MVP:** Minimum Viable Product
- **REST:** REpresentational State Transfer
- **S3:** Simple Storage Service
- **SOAP:** Simple Object Access Protocol
- **SPA:** Single-Page Application
- **SSL:** Secure Sockets Layer
- **TCP:** Transmission Control Protocol
- **TFG:** Trabajo Fin de Grado
- **TLS:** Transport Layer Security
- **UI:** User Interface
- **URI:** Uniform Resource Identifier
- **URL:** Uniform Resource Locator
- **XML:** eXtensible Markup Language.

1. Introducción

Esta memoria forma parte de la documentación del proyecto titulado "**Herramienta visual para la gestión y análisis de la carga docente**" del Trabajo de Fin de Grado en Ingeniería Informática.

En este apartado se explicará la estructura y el contenido de la memoria, así como unas cuestiones relevantes relacionadas con la aplicación.

El proyecto consiste en el desarrollo de una aplicación web con una arquitectura cliente-servidor que permite realizar búsquedas de profesores, titulaciones y asignaturas mediante filtros, ordenaciones y paginaciones.

Además, permite visualizar gráficos para detectar relaciones en los datos y facilitar su visualización y comprensión.

La herramienta facilita las labores de planificación docente de los cursos y permite al profesor examinar su carga docente y asignaturas asociadas.

Se puede acceder a la aplicación a través de la siguiente dirección:

http://tfg-client.s3-website-us-east-1.amazonaws.com

Se ha seguido una metodología ágil durante el desarrollo de la aplicación.

1.1. Estructura y contenido de la memoria

Siguiendo esta estructura se expondrán los aspectos más relevantes del desarrollo de la memoria:

1. **Introducción:**
   Se especifica el tema y el enfoque del proyecto al lector.
2. **Objetivos:**
   Se establecen los objetivos funcionales, no funcionales y personales que se quieren lograr durante el desarrollo de la aplicación.
3. **Conceptos teóricos:**
   Se destacan los aspectos teóricos más relevantes necesarios para entender la aplicación.

4. **Técnicas y herramientas:**
   Se describen las técnicas y herramientas utilizadas durante el desarrollo de la aplicación.
5. **Aspectos relevantes del desarrollo:**
   Se explican los aspectos más relevantes del desarrollo de la aplicación.
6. **Conclusiones:**
   Se valoran los objetivos cumplidos tras la realización de la aplicación
7. **Líneas futuras de trabajo:**
   Se resumen las adaptaciones y ampliaciones que se pueden desarrollar en el futuro.
8. **Bibliografía:**
   Se nombran las referencias utilizadas para crear esta memoria.

Además de esta memoria, se entrega el resto de la documentación, formada por los siguientes anexos:

- **Anexo I – Plan de Proyecto Software:**
  Se encuentra la planificación temporal del proyecto.
- **Anexo II – Especificación de Requisitos:**
  Se especifican los actores, objetivos y diagramas de Casos de Uso.
- **Anexo IV – Documentación Técnica:**
  Se encuentra información para los programadores.
- **Anexo VI – Manual de Usuario:**
  Se explica la funcionalidad de la aplicación al usuario final.

Por último, también se encuentra el código fuente de la aplicación.

1.2. Cuestiones relacionadas con la aplicación

1. Función de la aplicación:

La aplicación ofrece una interfaz que facilita la visualización de la carga docente, la búsqueda de titulaciones, asignaturas y profesores, además de la relación entre asignaturas, profesores y cursos.

También dispone de gráficos sencillos de carga: diagramas de barras horizontales y circulares.

La aplicación se encuentra desplegada en producción, en los servidores de AWS, para que cualquier usuario pueda visitarla en cualquier momento:

http://tfg-client.s3-website-us-east-1.amazonaws.com

2. Creación de la aplicación:

Para el desarrollo del proyecto se define una arquitectura cliente-servidor y una aplicación web completamente funcional e intuitiva.

Se utilizan las tecnologías VueJS para el cliente, Java Spring Boot para el servidor y AWS para el despliegue en producción.

3. Enfoque de la aplicación:

La aplicación web puede ser usada por cualquier usuario interesado en los datos mencionados anteriormente, pero principalmente está enfocada en el equipo de dirección de la planificación docente de los cursos, para facilitar las labores de gestión.

2. Objetivos

El objetivo principal del proyecto es desarrollar una herramienta que facilite las labores de gestión del equipo de dirección en la planificación docente de los cursos, en la parte relativa a la carga docente del profesorado.
Además, la herramienta deberá permitir al docente examinar fácilmente su carga docente y asignaturas asociada de una manera visual en un navegador web.

En este apartado se describirán los objetivos funcionales, los no funcionales y los personales.

2.1. Objetivos funcionales

Los objetivos mínimos funcionales son los designados en la presentación del TFG y son los siguientes:

- Diseño de una interfaz que facilite la visualización de la carga docente.
- Conexión o realización de una BBDD sobre la carga docente del departamento de informática y automática.
- Búsqueda de profesores o asignaturas.
- Relación entre asignaturas, profesores y cursos.
- Realización de gráficos sencillos de carga: scatter plots e histogramas fundamentalmente
- Puesta a punto en producción para el curso siguiente a aquél en que se desarrolle.

2.2. Objetivos no funcionales

Los objetivos no funcionales del sistema que se plantean para esta aplicación son los siguientes:

- Accesibilidad: El sistema deberá ser accesible por todos los usuarios.
- Disponibilidad: El sistema deberá encontrarse en un estado operativo específico.
- Escalabilidad: El sistema deberá estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos

- Portabilidad: El sistema deberá poder ser ejecutado en diferentes plataformas y/o sistemas operativos.
- Rendimiento: El sistema deberá consumir el menor número de recursos.
- Usabilidad: El sistema deberá tener una interfaz sencilla e intuitiva para los usuarios.

2.3. Objetivos personales

En este apartado, como autor, hablaré en primera persona al tratarse de una descripción de los objetivos personales que se desean conseguir mediante la realización de este proyecto:

- Completar los objetivos funcionales y no funcionales anteriormente propuestos para la aplicación que voy a programar.
- Demostrar que soy capaz de desarrollar y conectar ambas partes de una aplicación cliente-servidor y obtener un producto completo, útil para otras personas.
- Demostrar todo el conocimiento adquirido durante la carrera.
- Aprender y manejar diferentes herramientas, técnicas y lenguajes de programación.

3. Conceptos teóricos

3.1. Aplicación web

Una aplicación web es un programa almacenado en un servidor remoto al que se accede a través de Internet, por medio de una interfaz de navegador. Las aplicaciones web pueden diseñarse para gran variedad de usos y cualquier persona puede utilizarlas.
Las aplicaciones web de uso común pueden ir desde correo web hasta tiendas de comercio electrónico o calculadoras en línea.

¿Cómo funcionan?
A una aplicación web se accede mediante una red, por lo que no es necesario descargarla. De esta forma, los usuarios tienen acceso a una aplicación web a través de un navegador web como puede ser Google Chrome, Safari o Mozilla Firefox.
Para el correcto funcionamiento de una aplicación web es necesario un servidor web, un servidor de aplicaciones y una base de datos. Por su parte, los servidores web tienen la tarea de administrar las solicitudes realizadas por un cliente, mientras que el servidor de aplicaciones se encarga de completar la tarea solicitada. Además, puede utilizarse una base de datos para el almacenamiento de cualquier información que sea necesaria.
Las aplicaciones web pueden ser creadas por equipos de desarrollo pequeños, y de hecho, normalmente sus ciclos de desarrollo son cortos. Mayoritariamente, las aplicaciones web se encuentran escritas en JavaScript, HTML5 o CSS (hojas de estilo en cascada). Por lo general, la programación del lado del cliente hace uso de estos lenguajes, pues ayudan a constituir el frontend. En cambio, debido a que la programación del lado del servidor tiene como fin la creación de los scripts que utilizará una aplicación web, se suelen utilizar lenguajes como Python, Java o Ruby.

Ventajas
Entre las ventajas de la aplicación web encontramos las siguientes:

- La posibilidad de acceso de diferentes usuarios a la misma versión de una aplicación.
- No se requiere la instalación de aplicaciones web.

- Es posible acceder a las aplicaciones web mediante diversas plataformas, como una computadora de escritorio, una computadora portátil o un dispositivo móvil.
- Es posible el acceso a través de múltiples navegadores. [1]

3.2. Arquitectura cliente-servidor

Uno de los estilos arquitectónicos distribuidos más conocidos es la arquitectura cliente-servidor, formado por dos componentes denominados: cliente (demandante) y servidor (proveedor de recursos o servicios).

El servidor es el sistema centralizado a través del cual el servicio es solicitado y recibido por los clientes. La interfaz proporcionada por las computadoras cliente permite la solicitud de servicios del servidor por parte del usuario y la visualización de los resultados devueltos por el servidor. Los servidores responden a las solicitudes de los clientes tras haberlas recibido.

Los servidores y los clientes se encuentran ubicados en partes diferentes de la red. Por lo general, mientras que los clientes se encuentran en computadores personales o en estaciones de trabajo, los servidores se ubican en máquinas más potentes.

Los computadores cliente y el servidor se consideran dispositivos independientes. Mientras que numerosos clientes pueden acceder a la información del servidor de forma simultánea, es posible que una computadora cliente realice otras tareas. [2]

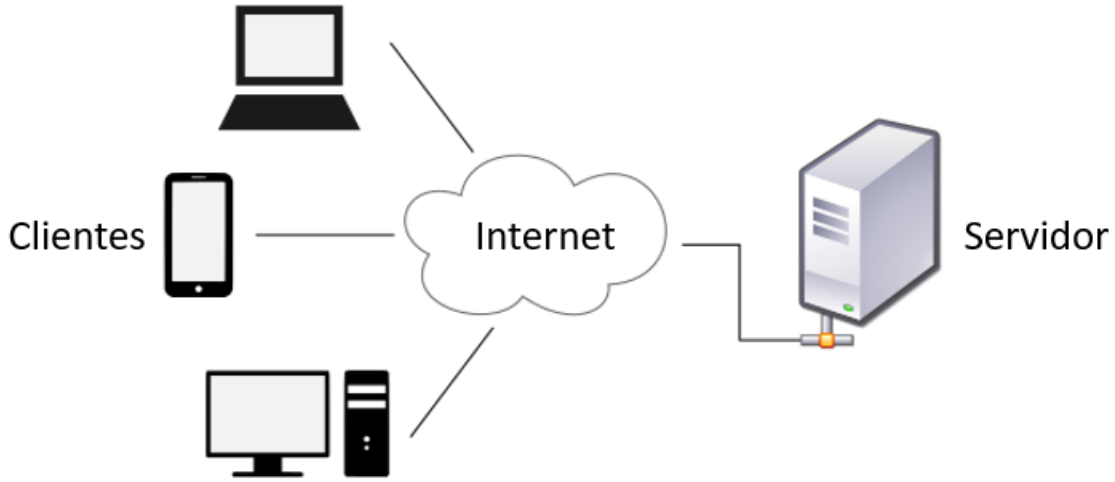

Ilustración 1: Arquitectura cliente-servidor

Ventajas

- El sistema se encuentra en un solo lugar con los datos centralizados.
- Es destacable la eficiencia del modelo en la entrega de recursos al cliente y su mantenimiento de bajo costo.
- La administración es sencilla y la entrega de los datos al cliente se realiza fácilmente.
- Este sistema es más seguro y brinda una mayor seguridad a los datos gracias a la centralización de estos.
- Es posible la integración de más clientes y servidores en el servidor dentro de este tipo de modelo, esto provoca un aumento en la flexibilidad general del modelo y que el rendimiento sea sobresaliente.

Inconvenientes

- Es posible encontrar un virus o cualquier script malicioso en los sistemas de los clientes en caso de que se esté ejecutando alguno en el servidor.
- Para evitar la falsificación de los datos entre la transmisión, es necesario agregar seguridad adicional.
- Es necesario tener en cuenta que el problema principal es la caída del servidor. Esto trae como consecuencia la pérdida de la conexión por parte del cliente, no pudiendo acceder a los datos. [3]

3.3. HTTP

HTTP es el acrónimo de HyperText Transfer Protocol (Protocolo de Transferencia de HiperTexto). El hipertexto es un tipo de texto especialmente codificado con la ayuda de un lenguaje de codificación estándar denominado HTML.

Es el protocolo utilizado para transferir hipertexto entre dos computadoras. HTTP proporciona un estándar entre un navegador web y un servidor web con el fin de establecer la comunicación. Se trata de un conjunto de reglas destinadas a trasladar datos de una computadora a otra. Siempre que un usuario web abre su navegador web, el usuario hace uso de HTTP de manera indirecta. [4]

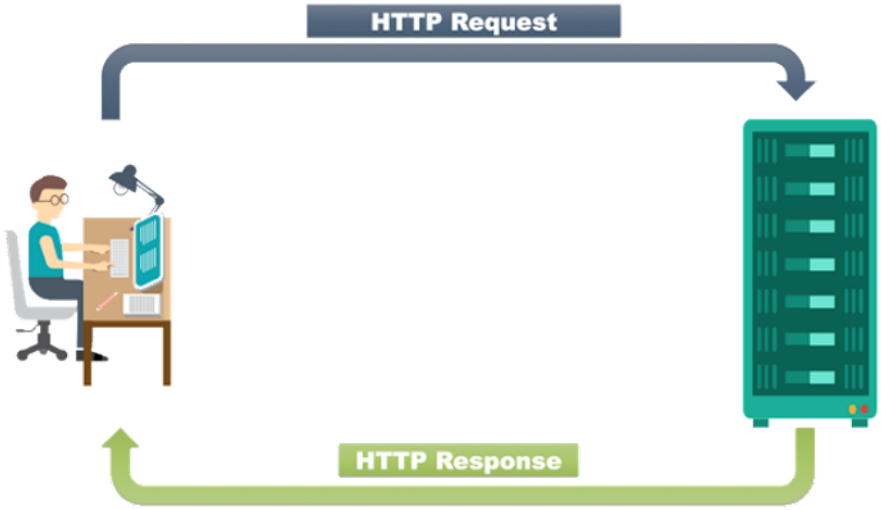

Ilustración 2: HTTP

Ventajas

- Al existir un menor número de conexiones simultáneas, el uso de la memoria y el de la CPU son bajos.
- La congestión de la red es menor gracias a que hay pocas conexiones TCP.
- Al realizarse el protocolo de enlace en la etapa de conexión inicial, se ve reducida la latencia porque no existe necesidad del protocolo de enlace en solicitudes posteriores.
- HTTP permite combinar múltiples consultas en una sola conexión TCP sin esperar las respuestas para cada consulta

Inconvenientes

- Al no utilizar ningún método de encriptación como HTTPS, que usa TLS para encriptar las solicitudes y respuestas HTTP normales, es menos seguro.
- No está optimizado para teléfonos móviles.
- Al ser menos seguro, no ofrece un intercambio genuino de datos.
- Es necesario tener en cuenta que el cliente no cierra la conexión hasta haber recibido datos completos del servidor, lo que conlleva que el servidor espere a que se completen los datos, así que no puede estar disponible durante ese periodo de tiempo para otros clientes. [4]

3.4. API REST

REST, acrónimo de Representational State Transfer (Transferencia de Estado Representacional), es un estilo arquitectónico para sistemas distribuidos. Estos sistemas permiten elaborar y presentar al usuario documentos complejos con enlaces a otros documentos cuya residencia física se encuentre en otras máquinas de Internet. [6][7]
Es utilizado sobre todo para facilitar el uso de servicios web con un enfoque universal y estandarizado.
Estos servicios web ofrecen sus recursos a través de una representación textual, permitiendo su lectura y procesamiento con un protocolo sin estado. Las operaciones que puede realizar un cliente se basan sobre todo en los protocolos HTTP.
El propósito es la imposición de determinadas reglas a la API con el fin de que adquiera mayor rendimiento, escalabilidad, simplicidad, modificabilidad, visibilidad, portabilidad y confiabilidad. [8]
Los seis principios rectores o restricciones de la arquitectura RESTful son:

1. Interfaz uniforme

La restricción de interfaz uniforme define la interfaz entre clientes y servidores. Simplifica y desacopla la arquitectura, lo que permite que cada parte evolucione de forma independiente.
Esto significa que se tiene:

- Verbos HTTP (GET, POST, PUT, DELETE).
- URI (nombre del recurso).
- Respuesta HTTP (estado y cuerpo).

Los cuatro principios rectores de la interfaz uniforme son:

1. Identificación del recurso: Es necesario que cuente con un URI específico y cohesivo.
2. Representación del recurso: La representación puede ser en HTML, XML, JSON, TXT, entre otras.
3. Mensajes autodescriptivos: Todos los mensajes cuentan con la información suficiente para describir la forma de procesamiento del mensaje.

4. Hipermedia como motor del estado de la aplicación: Los clientes entregan el estado por medio del contenido del cuerpo, los parámetros de cadena de consulta, los encabezados de solicitud y el URI solicitado (el nombre del recurso).

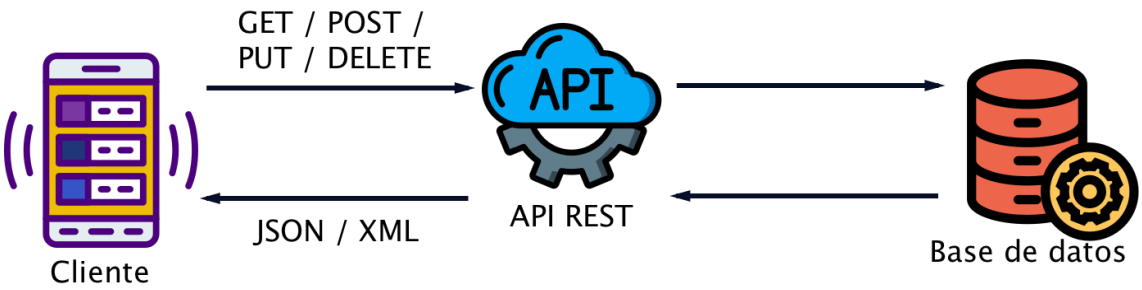

Ilustración 3: API REST

2. Apátrida

Es posible que un mismo cliente envíe varias solicitudes al servidor siempre y cuando estas sean independientes. Por tanto, es necesario que cada solicitud contenga toda la información requerida para que el servidor la entienda y procese.

3. Caché

Muchos clientes tienen acceso a un mismo servidor y pueden solicitar los mismos recursos, por lo que se necesita el almacenamiento en caché de estas repuestas, aumentando así el rendimiento de forma significativa.

4. Cliente-Servidor

Los clientes y los servidores se separan por medio de una interfaz uniforme. Siempre y cuando la interfaz no se altere, es posible reemplazar y desarrollar los servidores y clientes de forma independiente. Los servidores pueden ser más simples y escalables porque no se preocupan por el estado del usuario o la interfaz.

5. Sistema en capas

Los servidores intermediarios permiten el equilibrio de carga y proporcionan cachés compartidas, esto puede mejorar la escalabilidad del sistema. Las capas también tienen la posibilidad de aplicar políticas de seguridad. [9]

3.5. Arquitectura de microservicios

La arquitectura de microservicios, abreviada comúnmente como “microservicios”, es un estilo utilizado para el desarrollo de aplicaciones. Gracias a ellos es posible separar una aplicación grande, en partes más pequeñas e independientes entre sí, contando cada una con su dominio propio de responsabilidad. Una aplicación basada en microservicios, para atender una sola solicitud de usuario, puede llamar a varios microservicios internos con los que construir su respuesta. [11]

Esta arquitectura estructura una aplicación como una colección de servicios que son:

- Altamente mantenibles.
- Débilmente acoplados.
- Desplegables de forma independiente.
- Organizados en torno a las capacidades empresariales.
- Propiedad de un pequeño equipo.

Gracias a la arquitectura de microservicios es posible la entrega rápida, frecuente y confiable de aplicaciones grandes y complejas. Además, posibilita que una organización evolucione su pila de tecnología. [12]

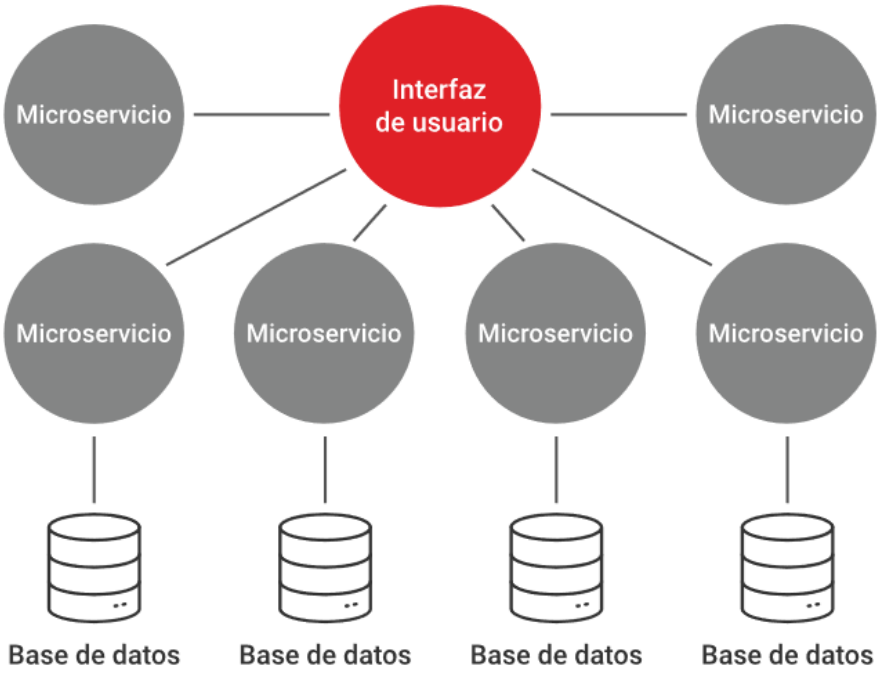

Ilustración 4: Arquitectura de microservicios

3.6. Biblioteca

Al conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que presenta una interfaz bien definida para la funcionalidad invocada, se le llama en informática biblioteca o, por vicio del lenguaje librería (traducción errónea del inglés library).

A diferencia de un programa ejecutable, el comportamiento que implementa una biblioteca no espera ser utilizada de forma autónoma (un programa sí tiene un punto de entrada principal), sino que su fin es ser utilizada por otros programas, independientes y de forma simultánea. Por otra parte, el comportamiento de una biblioteca no tiene por qué diferenciarse demasiado del que pudiera especificarse en un programa. Es más, unas bibliotecas pueden requerir de otras para funcionar.

La mayoría de los sistemas operativos modernos proporcionan bibliotecas que implementan los servicios del sistema. De esta manera, estos servicios se han convertido en una «materia prima» que cualquier aplicación moderna espera que el sistema operativo ofrezca. Como tal, la mayor parte del código utilizado por las aplicaciones modernas se ofrece en estas bibliotecas. [13]

3.7. La nube

Aunque la definición de la nube pueda parecer compleja, lo cierto es que realmente es un término utilizado para la descripción de una red global de servidores, cada uno con su propia función. La nube, lejos de ser una entidad física, se trata de una extensa red de servidores remotos alrededor de todo el mundo conectados todos ellos entre sí y destinados a operar como un solo ecosistema. Estos servidores se han diseñado para almacenar y administrar datos, ejecutar aplicaciones o entregar contenido o un servicio como transmitir vídeos, correo web, software de productividad de oficina o redes sociales. No accede a los datos y archivos desde una computadora local o personal, sino que lo hace en línea desde cualquier dispositivo que tenga acceso a Internet; de esta forma, la información se encontrará disponible en cualquier lugar y en cualquier momento que sea necesaria.

Ilustración 5: La nube

Para llevar a cabo la implementación de recursos en la nube, las empresas hacen uso de cuatro métodos distintos:

- Una nube pública, que además de compartir recursos es capaz de ofrecer servicios al público a través de Internet.
- Una nube privada, que ofrece servicios por medio de una red interna privada, hospedada normalmente en las instalaciones, pero que no se comparte.
- Una nube híbrida, destinada a compartir servicios entre públicos y privados en función de su propósito.
- Una nube comunitaria, que intercambia recursos únicamente entre organizaciones, tales como instituciones gubernamentales. [14]

3.8. Metodología Scrum

Scrum consiste en una metodología de desarrollo ágil que se basa en un proceso iterativo e incremental y de la que se hace uso durante el desarrollo de software. Se encuentra diseñado para dar valor al cliente durante todo el desarrollo del proyecto, siendo por tanto un marco ágil adaptable, rápido, flexible y eficaz.

El desarrollo parte de una idea general de lo que hay que construir, elaborando una lista de características ordenadas por prioridad que el propietario del producto quiere obtener.

La metodología Scrum se asienta sobre un conjunto de prácticas y roles muy definidos que han de intervenir durante el proceso de desarrollo de software.

Los Sprints son los bloques temporales, cortos y periódicos en los que se ejecuta Scrum, y estos suelen oscilar entre 2 y 4 semanas, que es precisamente el plazo de retroalimentación y reflexión. Cada Sprint, al ser una entidad en sí misma, ofrece un resultado completo, una variación del producto final que pueda ser entregado con el menor esfuerzo posible al cliente en el momento en que así lo solicite.
El punto de partida del proceso es una lista de objetivos o requisitos que forman el plan del proyecto, priorizados por el cliente del proyecto tras considerar un equilibrio entre el valor y el coste de estos. De esta forma, es posible determinar las iteraciones y las siguientes entregas.

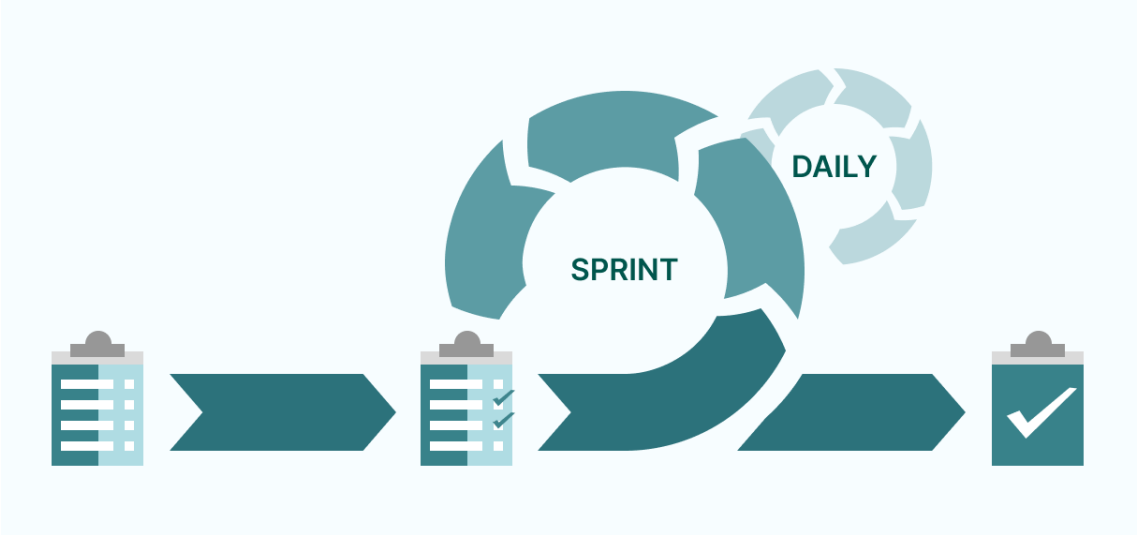

Ilustración 6: Metodología Scrum

Ventajas

Siendo Scrum el marco de referencia más empleado y fiable en la industria del software, es evidente que son numerosas las ventajas que presenta frente al resto de metodologías de desarrollo ágil. Algunos de los beneficios conocidos son:

- Fácilmente escalable: Gracias a la iteración de los procesos de Scrum y a su manejo dentro de períodos específicos de trabajo, resulta más sencillo que el equipo se enfoque en funcionalidades determinadas para cada período.

- Cumplimiento de expectativas: El cliente, para determinar sus expectativas, debe indicar cuál es el valor que aporta cada requisito o historia del proyecto. El equipo procede a estimarlas para que el Propietario de Producto establezca su prioridad a partir de dicha información.
- Flexibilidad ante los cambios: Reacción rápida a los cambios en los requisitos generados por las necesidades del cliente o la evolución del mercado.
- Reducción del tiempo de comercialización: Incluso sin estar el producto completamente listo, cabe la posibilidad de que el cliente comience a utilizar las funcionalidades construidas de la aplicación.
- Mayor calidad del software: El sistema de trabajo y la necesidad de alcanzar una versión funcional después de cada iteración, contribuyen a la obtención de un software de mayor calidad.
- Reducción de riesgos: Como consecuencia de que las funcionalidades que tienen mayor valor se realizan en primer lugar y del conocimiento de la velocidad con la que el equipo va avanzando en el proyecto, se eliminan los riesgos de forma anticipada y eficaz.

4. Técnicas y herramientas

4.1. Desarrollo del cliente

4.1.1. Visual Studio Code

El editor de código fuente desarrollado por Microsoft se denomina Visual Studio Code (VS Code). Se trata de un software libre y multiplataforma que cuenta con disponibilidad para Windows, GNU/Linux y macOS. También cuenta con una buena integración con Git, tiene numerosas extensiones que brindan la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación, además cuenta con soporte para depurar código. [16]

4.1.2. VueJS

La creación de UI (interfaces de usuario) y SPA (aplicaciones de una sola página) puede llevarse a cabo a través de VueJS, un marco JavaScript progresivo, famoso porque cuenta con una rápida curva de aprendizaje. Al ser una biblioteca accesible y fácil de aprender, es posible empezar a crear aplicaciones web en VueJS necesitando únicamente conocimientos en HTML, CSS y JavaScript. Se trata de un framework completo para la creación de aplicaciones web enormes.

Ventajas

- Curva de aprendizaje y documentación bien escrita.
- Código modular y reutilizable.
- Fácil desarrollo. [17]

4.1.3. HTML

El lenguaje informático que posibilita la creación de sitios web es HyperText Markup Language (o HTML). Al igual que en el resto de los lenguajes, se encuentran palabras de código y sintaxis. Es destacable su facilidad relativa de comprensión, siendo cada vez

más poderoso en lo que le permite a alguien crear. HTML se encuentra en constante evolución con el fin de satisfacer las demandas y requisitos de Internet bajo la figura del World Wide Web Consortium, la organización encargada del diseño y mantenimiento del lenguaje, siendo ejemplo de ello la transición a la Web 2.0.
Los usuarios de Internet utilizan un método denominado hipertexto para navegar por la web. A su vez, tienen acceso a nuevas páginas al pulsar en los hipervínculos, que son un texto especial. [18]

4.1.4. CSS

CSS, cuyo significado es "hojas de estilo en cascada", es el lenguaje utilizado para describir la presentación de las páginas web, incluyendo las fuentes, los colores y el diseño. De esta forma, se consigue que sean presentables para los usuarios.
Es necesario tener en cuenta que CSS es independiente de HTML y que está diseñado para la realización de hojas de estilo para la web. Además, es posible su uso con cualquier lenguaje de marcado basado en XML. A continuación, se interpreta el acrónimo:

- Cascada: Caída de estilos.
- Estilo: Agregar diseños/Estilizar las etiquetas HTML.
- Hojas: Escribir el estilo en diferentes documentos. [19]

4.1.5. JavaScript

JavaScript consiste en un lenguaje de programación basado en texto, es utilizado tanto en el lado del cliente como en el del servidor, permitiendo que las páginas web sean interactivas.
Al contrario que HTML y CSS, que son lenguajes que proporcionan a las páginas web estilo y estructura, JavaScript facilita elementos interactivos a las páginas web con el fin de captar la atención del usuario.
La experiencia de usuario de la página web se ve mejorada gracias a la incorporación de JavaScript, pasando de ser una página estática a ser una página interactiva.
Teniendo en cuenta todo lo comentando anteriormente, se puede decir que JavaScript agrega comportamiento a las páginas web. [20]

4.1.7. Librerías

Axios

Axios es una librería de JavaScript que ofrece la posibilidad de hacer sencillas las operaciones como cliente HTTP y puede ser ejecutada en el navegador. Esto permite configurar y realizar solicitudes a un servidor, recibiendo respuestas fácilmente sencillas de procesar.

Al hacer uso de un framework JavaScript (en este caso VueJS), se cuenta con un cliente HTTP encargado de realizar tareas de request contra un servidor, que recibirá a su vez el response. Si se compara con otras librerías, se puede observar que esta es la mejor opción. Por ejemplo, utilizando Ajax podrían existir complicaciones en las peticiones, y no es posible usar Fetch en todos los navegadores. Tampoco es factible recurrir a jQuery, puesto que es una librería excesivamente pesada.

Por lo que Axios es una alternativa con bastantes ventajas:

- Cuenta con una API unificada para las solicitudes Ajax.
- Optimizado para facilitar el consumo de servicios web, API REST y que devuelvan datos JSON.
- Su uso resulta sencillo y es un complemento perfecto para las páginas convencionales.
- Su peso es muy pequeño, de apenas 13KB, pudiendo ser incluso menor si se envía comprimido al servidor.
- Tiene compatibilidad con cualquier navegador en su versión actual. [21]

Vue3-easy-data-table

Esta librería ofrece un componente de tabla para la fácil visualización de datos y con todas las funciones para aplicaciones VueJS 3.

Además de admitir tanto fuentes de datos fijas como dinámicas, cuenta con numerosas funciones avanzadas, tales como la búsqueda en vivo, el filtrado de columnas o la paginación. [22]

Su uso es posible tanto en modo cliente como en modo del lado servidor. Por su parte, el modo cliente se concibe para aquellos casos en los que todos los datos ya se hayan cargado; es decir, su llamada inicial tiene como fin la solicitud de todas las páginas de un

servidor. En cambio, en el modo del lado del servidor, cada vez que se navega a una nueva página, la solicitud realizada debe consistir en datos limitados de un servidor. [23]

Vue-apexcharts

Vue-ApexCharts es una librería completa para JavaScript, presenta un componente contenedor para ApexCharts listo para ser integrado en la aplicación VueJS.

Por otra parte, ApexCharts se trata de una biblioteca moderna de gráficos cuyo fin es ayudar a los desarrolladores en la creación de gráficos atractivos e interactivos para páginas web. Consiste en un proyecto de código abierto, gratuito en aplicaciones comerciales, que cuenta con licencia del MIT.

Ventajas de ApexCharts:

- Responsive: Teniendo en cuenta que puede contar con distintos diseños para diferentes tamaños de pantalla, escala tanto en computadoras de escritorio como tabletas y dispositivos móviles.
- Interactivo: Cuando el usuario pasa el cursor sobre los puntos de datos, es capaz de mostrar información adicional.
- Dinámico: Es posible la carga de datos en selecciones y la creación de otros gráficos basados en dichas selecciones.
- Alto rendimiento: Se puede renderizar una gran cantidad de puntos de datos.
- Anotaciones: Es posible colocar etiquetas con el fin de facilitar a los usuarios la interpretación de los gráficos.
- Animaciones suaves: Al cambiar conjuntos de datos, cargar datos dinámicos e interactuar con los gráficos, la experiencia interactiva que proporciona es fluida.
- Paletas y estilos: Se puede escoger entre distintas paletas de colores, existiendo también degradados, imágenes, patrones y sombras paralelas a los elementos del gráfico. [24]

EmailJS

Esta librería permite enviar correos electrónicos sin desarrollo de backend, directamente desde las tecnologías Javascript del lado del cliente. Los desarrolladores crean plantillas y eligen el correo electrónico al que desean que vayan sus mensajes automatizados.

EmailJS es una herramienta de la categoría API de correo electrónico. [25]

4.2. Desarrollo del servidor

4.2.1. Maven

Maven, desarrollada por Apache Group, se trata de una herramienta de compilación de código abierto destinada a la compilación, publicación e implementación de diversos proyectos a la vez para mejorar su gestión.
Maven tiene como propósito facilitar a los desarrolladores:

- Un modelo para proyectos integral, mantenible, simple y reutilizable.
- Un conjunto de herramientas y complementos con capacidad para interactuar con el modelo declarativo.

Algunas de las características más notables son:

- Un gran repositorio en continuo crecimiento de bibliotecas de usuarios.
- Capacidad de configurar proyectos de forma sencilla haciendo uso de las mejores prácticas.
- Gestión de dependencias, con actualización automática
- Compatibilidad con anteriores versiones.
- Informes sólidos de errores e integridad.
- Control de versiones principal automático.
- El uso uniforme está garantizado en todos los proyectos.
- Es extensible y puede escribir fácilmente complementos utilizando lenguajes de secuencias de comandos o Java. [26]

4.2.2. Eclipse STS

STS es el acrónimo de Spring Tool Suite, un entorno de desarrollo que está basado en Eclipse y se encuentra personalizado para desarrollar aplicaciones Spring.
Proporciona un entorno listo para depurar, ejecutar e implementar aplicaciones. STS se ha construido como adición a las últimas versiones de Eclipse. [27]

4.2.3. Spring Boot

Spring Boot se trata de un marco de código abierto basado en Java cuyo uso está destinado a la creación de microservicios.

Gracias a la arquitectura de microservicios, los desarrolladores pueden, de manera independiente, desarrollar e implementar servicios. Además, cada servicio en ejecución cuenta con un proceso propio.

Ventajas de los microservicios:

- Fácil implementación.
- Escalabilidad sencilla.
- Compatible con contenedores.
- Configuración mínima.
- Menor tiempo de producción.

Los desarrolladores de Java pueden desarrollar una aplicación Spring independiente y de grado de producción posible de ejecutar gracias a la plataforma proporcionada por Spring Boot. No es necesaria una configuración completa de Spring, pudiendo así comenzar con configuraciones mínimas.

Ventajas de Spring Boot:

- No solo resulta sencillo de entender, sino que además es fácil desarrollar aplicaciones REST.
- Incrementa la productividad.
- El tiempo de desarrollo se ve reducido. [28]

4.2.4. Java

Java consiste en un lenguaje de programación y una plataforma informática destinada al desarrollo de aplicaciones. Es uno de los lenguajes de programación más utilizados, lanzado en primer lugar por Sun Microsystem en 1995, aunque posteriormente Oracle Corporation lo adquirió.

La plataforma Java cuenta con un conjunto de programas que contribuyen al desarrollo y la ejecución de programas escritos en este el lenguaje de programación. A su vez, esta plataforma incluye un motor de ejecución, un compilador y un conjunto de bibliotecas.

Por último, es necesario destacar que Java no es un lenguaje específico de ningún procesador o sistema operativo, sino que se trata de un lenguaje independiente de la plataforma. [29]

4.2.5. Swagger

Swagger es un conjunto de reglas, especificaciones y herramientas de código abierto utilizado para el desarrollo y la descripción de API REST. A su vez, el marco Swagger brinda la posibilidad a los desarrolladores de crear documentación de API interactiva y legible tanto por humanos como por máquinas.

Por lo general, dentro de las especificaciones de API nos encontramos con información como operaciones admitidas, parámetros y resultados, requisitos de autorización, puntos finales disponibles o licencias necesarias. Swagger tiene la capacidad de generar dicha información de forma automática a partir del código fuente, solicitando a la API que le devuelva un archivo de documentación a partir de sus anotaciones.

Además, Swagger ayuda al usuario en la creación, documentación, prueba y consumo de servicios web REST.

Swagger cuenta con numerosas ventajas:

- Tiene una interfaz de usuario amigable que traza el plan para las API.
- Es posible que comprendan la documentación tanto los desarrolladores como los no desarrolladores, como lo gerentes de proyectos o clientes.
- Las especificaciones son legibles por humanos y máquinas.
- La documentación que genera es interactiva, además de que se puede comprobar fácilmente
- Admite la creación de bibliotecas API en más de 40 idiomas.
- Para facilitar las ediciones, se acepta el formato en JSON y YAML.
- Ayuda a automatizar los procesos relacionados con la API. [30]

4.2.6. SoapUI

SoapUI es una herramienta multiplataforma de prueba de servicios web de herramientas de código abierto tanto para arquitecturas orientadas a servicio, como para transferencias de estado de representación.

SOAP, cuya definición fue dada por el World Wide Web Consortium (W3C), es la palabra que se utiliza para el Protocolo Simple de Acceso a Objetos. Este protocolo se basa en XML y es usado para intercambiar información en un entorno descentralizado y distribuido.

Ventajas de SoapUI:

- Cuenta con una interfaz gráfica de usuario sencilla de usar.
- Proporciona transporte de datos para servicios web.
- Es posible su uso como transmisión de mensajes.
- Útil para realizar pruebas funcionales.
- Es un protocolo diseñado para comunicarse con la ayuda de Internet.
- Desempeña el papel tanto de cliente como de servicio.
- Además del sencillo uso, es fácil transformar pruebas funcionales en pruebas no funcionales. [31]

4.2.7. Postman

A través de Postman, una plataforma de API (Interfaz de Programación de Aplicaciones) de prueba de software independiente, es posible crear, probar, diseñar, modificar y documentar una API. Cuenta con una interfaz gráfica de usuario simple para enviar y ver solicitudes y respuestas HTTP.
Con esta herramienta se construyen suites de prueba denominadas colecciones, que interactúan con la API.

Casi toda funcionalidad que cualquier desarrollador pueda necesitar se integra en esta herramienta, que es capaz de llevar a cabo diferentes tipos de solicitudes HTTP (como GET, POST, PUT, PATCH) y transformar la API en código para lenguajes como Python y JavaScript. [32]

4.2.7. Python

Python es un lenguaje de programación de alto nivel interpretado, orientado a objetos y con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con la escritura dinámica y el enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de secuencias de comandos o pegamento para conectar componentes existentes entre sí. [33]

Pandas

Pandas es una biblioteca de software escrita para el lenguaje de programación Python para la manipulación y el análisis de datos. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. [34]

4.3. Despliegue en producción

4.3.1. AWS

AWS es el acrónimo de Amazon Web Services, la plataforma en la nube ofrecida por Amazon. Son numerosos y completamente distintos los productos y servicios de computación que ofrece esta plataforma. Gracias a la división altamente rentable de Amazon proporciona servidores, redes, almacenamiento, computación remota, correo electrónico, desarrollo móvil y seguridad.

Ventajas de AWS:

- Ahorro de costes
- Escalabilidad
- Adaptabilidad
- Seguridad
- Confiabilidad [35]

Amazon S3

Amazon Simple Storage Service, también denominado Amazon S3, es un servicio líder en la industria de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Este servicio puede ser utilizado por clientes e industria de cualquier tamaño para el almacenamiento y la protección de datos, sea en la cantidad que sea, para una variedad de casos de uso, como: lagos de datos, sitios web, aplicaciones móviles, copia de seguridad y restauración, archivos, aplicaciones empresariales, dispositivos IoT y analítica Big Data.

Además, Amazon S3 cuenta con diferentes características de administración para que sea posible la optimización, organización y configuración del acceso a los datos y así satisfacer los requisitos comerciales, organizacionales y de cumplimiento específicos. [36]

Amazon EC2

Una instancia de Amazon EC2 es un servidor virtual en EC2 (Elastic Compute Cloud) de Amazon para llevar a cabo la ejecución de aplicaciones en la infraestructura de AWS, pudiendo servir como un conjunto prácticamente ilimitado de VM (máquinas virtuales). Mientras que EC2 es un servicio que hace posible la ejecución de en el entorno informático por parte de los suscriptores comerciales, AWS es una plataforma informática en la nube integral y en evolución.

Con el fin de adaptarse a las necesidades de cada usuario, Amazon ofrece varios tipos de instancias con configuraciones distintas de CPU, memoria, almacenamiento y recursos de red. Por supuesto, cada tipo se encuentra disponible en diferentes tamaños para así abordar los requisitos específicos de carga de trabajo. [37]

4.5. Herramientas para el control de versiones

4.5.1. Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia. [38]

4.5.2. GitHub

GitHub es un recurso de programación cada vez más popular que se utiliza para compartir código. Es un sitio de redes sociales para programadores que muchas empresas y organizaciones utilizan para facilitar la gestión y colaboración de proyectos. [39]

4.6. Herramientas para el control de versiones

4.6.1. Microsoft Word

Microsoft Word es un procesador de textos comercial ampliamente utilizado diseñado por Microsoft. Microsoft Word es un componente del paquete de software de productividad de Microsoft Office. [40]

4.6.2. Microsoft Project

Microsoft Project es un software de gestión de proyectos que se utiliza para crear cronogramas, planes de proyectos, administrar recursos y realizar un seguimiento del tiempo. Tiene funciones como diagramas de Gantt, tableros kanban y calendarios de proyectos para profesionales de gestión de proyectos. [41]

4.6.3. Visual Paradigm

Visual Paradigm es una herramienta de gestión y diseño potente, multiplataforma y fácil de usar. Proporciona a los desarrolladores de software la plataforma de desarrollo de vanguardia para crear aplicaciones de calidad más rápido, mejor y más barato. Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los IDE líderes. [42]

4.6.4. REM

REM (Gestión de Requerimientos) es una herramienta experimental y gratuita de Gestión de Requerimientos diseñada para dar soporte a la fase de Ingeniería de Requerimientos de un proyecto de desarrollo de software. [43]

5. Aspectos relevantes del desarrollo

Siguiendo esta estructura se exponen los aspectos más relevantes del desarrollo de la aplicación:

1. **Metodología:**
   Se comentan los motivos de trabajar con scrum.
2. **Arquitectura:**
   Se establece la arquitectura de la aplicación construida.
3. **Estudios previos:**
   Se explica el trabajo previo al desarrollo de la aplicación.
4. **Iteraciones:**
   Se describe el proceso realizado en cada iteración, los problemas encontrados, cómo se han solucionado y por qué se han tomado determinadas decisiones.
   - **Primera iteración**
   - **Segunda iteración**
   - **Tercera iteración**
   - **Cuarta iteración**

5.1. Metodología

En el apartado de los conceptos teóricos se desarrolla en qué consiste la metodología scrum. A continuación, se explican los motivos por los que se ha decidido utilizar:

- El equipo de desarrollo es pequeño:
  - Está formado por un único miembro.
- Permite dimensionar mejor el proyecto:
  - Como la metodología scrum ejecuta el proyecto en fases cortas de dos a cuatro semanas, llamadas iteraciones o sprints, se segmenta el desarrollo en pequeños bloques más gestionables que si se trata de abarcar el proyecto entero de principio a fin.
    Esto permite mucha flexibilidad a la hora de emprender cambios a mitad del proyecto, ya que tras cada fase se pueden identificar fácilmente los

objetivos e incluso los posibles contratiempos que se pueden encontrar en el camino.

- Rápido aprendizaje:
    - Ya que las iteraciones se completan en un breve espacio de tiempo y que estas son independientes al resto, permite obtener un aprendizaje que puede ser utilizado en los próximos sprints del proyecto.
      Es decir, no hace falta acabar el proyecto para darse cuenta de los errores, sino que se va aprendiendo y corrigiendo según se desarrolla el propio proyecto.
- Obtención de un producto mínimo viable (MVP):
    - Lograr un MVP tiene como ventajas que no es necesario tener el producto o servicio completamente acabado para ponerlo en marcha. Al final de cada sprint se obtiene un producto mejorado.
- Software funcional por encima de la documentación:
    - No es necesaria mucha documentación, las expectativas son las entregas rápidas y tener mucho control sobre el proyecto.
- Autonomía y responsabilidad.

Una de las reglas del scrum que no se ha cumplido son los feedbacks rápidos y precisos, esta metodología propone reuniones rápidas diarias donde el equipo se junta y expone de manera individual 3 factores:

- Qué se ha hecho desde la última reunión.
- Qué se va a hacer hoy.
- Los impedimentos que hay para realizarlo.

No se ha podido llevar a cabo debido a que el equipo está formado por una sola persona y no se han realizado reuniones diarias, las decisiones tomadas se han realizado de manera autónoma pensando en el producto.

Se ha decidido no trabajar con el marco Proceso Unificado de Desarrollo Software y emplear una metodología ágil para así lograr un software funcional en el menor tiempo posible, sacrificando la documentación exhaustiva.

5.2. Arquitectura

Se ha desarrollado un proyecto Spring Boot y VueJS con una aplicación CRUD.
El servidor back-end usa Spring Boot con Spring Web MVC para el controlador REST y servicios para interactuar con las bases de datos:

- Servicios web de diaweb
- Excel

La parte front-end está programada con VueJS.

La arquitectura de la aplicación es la siguiente:

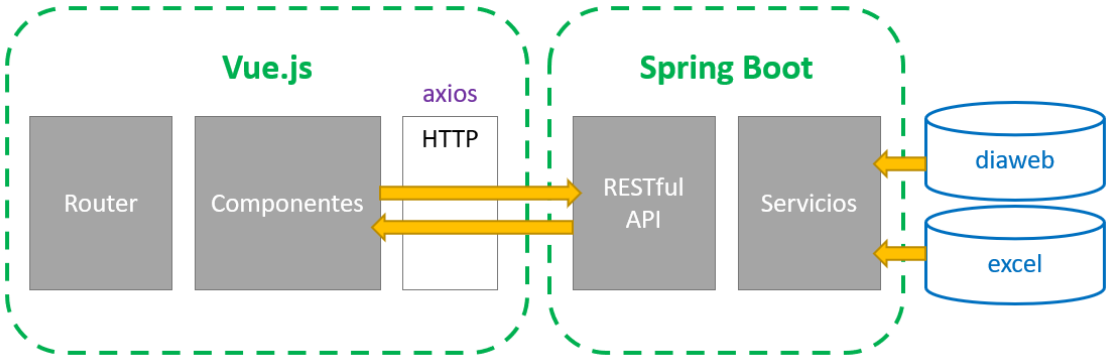

Ilustración 7: Arquitectura de la aplicación

– Spring Boot exporta REST API usando Spring Web MVC e interactúa con las bases de datos.
– El cliente Vue envía solicitudes HTTP y recupera respuestas usando axios, muestra los datos recibidos sobre los componentes. También usa Vue Router para navegar por las distintas páginas.

5.2.1. Spring Boot back-end

Visión general

Las API que la aplicación Spring Boot exporta son las siguientes:

MÉTODO	URL	ARGUMENTOS	COMPORTAMIENTO
POST	/api/asignaturasTitulacion	Código de una titulación	Obtiene las asignaturas que tiene una titulación
POST	/api/asignaturasProfesor	Número de columna de un profesor	Obtiene las asignaturas que tiene un profesor
POST	/api/asignaturaNombre	Código de una asignatura	Obtiene el nombre de la asignatura
GET	/api/asignaturas		Obtiene todas las asignaturas
POST	/api/gruposTeoria	Código de una asignatura	Obtiene los grupos de teoría que tiene una asignatura
POST	/api/gruposPractica	Código de una asignatura	Obtiene los grupos de práctica que tiene una asignatura
GET	/api/titulaciones		Obtiene todas las titulaciones
POST	/api/profesoresAsignatura	Código de una asignatura	Obtiene los profesores que imparten una asignatura
POST	/api/profesorNombre	Número de columna de un profesor	Obtiene el nombre del profesor
GET	/api/profesores		Obtiene todos los profesores

Tabla 1: API

La dirección IP estática donde se encuentra el servidor es 54.174.126.218 y se pueden comprobar los recursos de la API accediendo a la siguiente dirección:

http://54.174.126.218:8080/swagger

En el Anexo III se puede consultar la documentación técnica donde se puede encontrar los argumentos de entrada y salida de cada recurso. También, en el Anexo IV se encuentra el manual de usuario donde se muestra como navegar por la aplicación y se visualiza en la interfaz los datos que devuelve cada recurso.

Estructura del proyecto

Se ha seguido una buena práctica que es dividir el proyecto en capas.

Se puede dividir el proyecto en un paquete para los controladores, otro para los servicios y otro para el modelo de los datos.

Spring ofrece los estereotipos y propone una división en Controller, Service, Repository a fin de ordenar mejor el código. Aprovechando esto se pueden crear los paquetes para cada estereotipo de este modo:

- Paquete controller para todos los endpoints que tenga la aplicación.
- Paquete service para agregar las clases que respondan a la funcionalidad y lógica.
- Paquete model para las representaciones del modelo de datos (entidades).

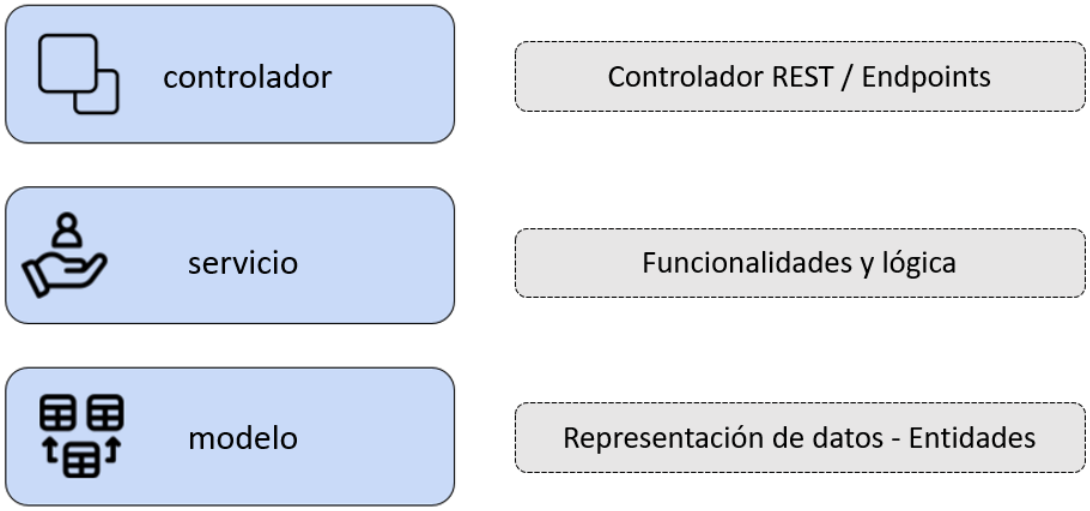

Ilustración 8: Estructura Spring Boot

Siguiendo esta buena práctica se obtiene la siguiente estructura:

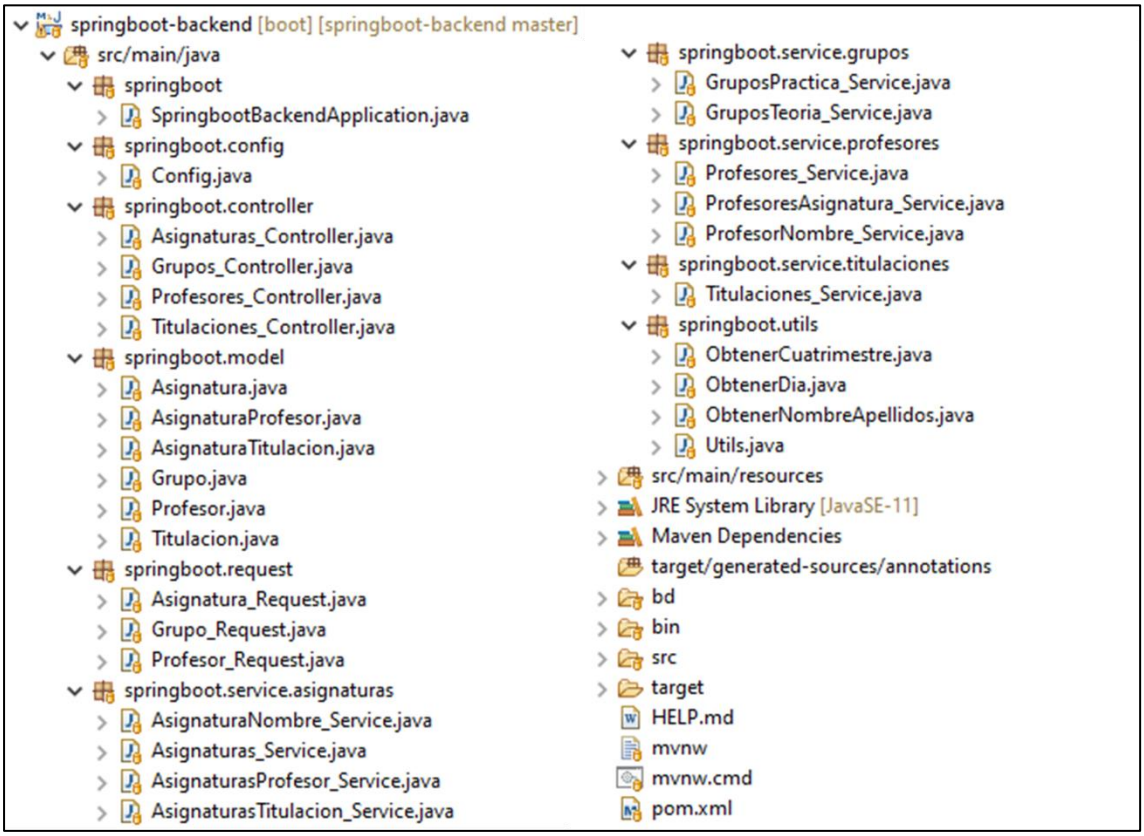

Ilustración 9: Estructura backend de la aplicación

5.3. Estudios previos

Esta fase de estudios previos tiene una duración de 2 semanas, comenzando el 14/02/2022 y finalizando el 01/03/2022; no se ha considerado una iteración debido a que no se redactó ninguna historia de usuario cuyo objetivo fuese obtener funcionalidad software.

Durante esta fase se instalaron las herramientas necesarias para poder comenzar el desarrollo del proyecto, se establecieron los objetivos y se realizaron las primeras pruebas.

Las tecnologías seleccionadas fueron Spring Boot para el back-end debido a que durante el verano de 2021 se realizaron prácticas curriculares en las que se presentó esta herramienta y fueron tres meses de desarrollo en los que se trabajó con ella de manera muy cómoda. Para la parte del front-end se recomendaba el uso de JavaScript y D3.js ya que se trataba de un entorno web visual, se decidió utilizar VueJS ya que durante el segundo cuatrimestre de 2022 se realizó un curso con esta tecnología, que pareció ser más fácil de aprender que Angular, el framework con el que inicialmente se iba a programar el front-end.

La primera reunión con el tutor Rodrigo Santamaría Vicente se realizó el día 16/02/2022, donde se presentaron las tecnologías seleccionadas, se concretaron los objetivos, se resolvieron dudas y se comentó el uso de metodologías ágiles. Además, se hizo entrega del Excel con los datos que formaban la base de datos de donde se obtendría la carga docente, aunque se comentó la idea de obtener también esos datos de los servicios web de diaweb.

El día 17/02/2022 se instala NodeJS y VueJS, se crea el proyecto inicial utilizando el editor de código Visual Studio Code y se realizan las primeras pruebas.

El día 19/02/2022 se instala Maven y Eclipse STS, se crea el proyecto inicial y se establece una división en capas, creando los paquetes controller, service y model.

El día 23/02/2022 el tutor crea un usuario en el servidor cpg3.der.usal.es, en ese servidor se desplegaría la aplicación en producción.

El día 28/02/2022 la profesora Ángeles María Moreno Montero envía los manuales sobre los servicios web en diaweb. Ese mismo día se realiza la instalación de la herramienta SoapUI y se ejecutan las primeras pruebas, peticiones SOAP en formato XML al servidor.

Una vez que la primera reunión se ha llevado a cabo, se han descargado las herramientas necesarias, se ha realizado un estudio sobre ellas y los servicios web devuelven la información solicitada finaliza la fase de estudios previos y pruebas. Ya se pueden definir las primeras historias de usuario para el primer sprint.

5.4. Iteraciones

Una vez finaliza la fase de estudios previos, comienza el desarrollo del software, se ha conseguido obtener una aplicación web totalmente funcional a través de cuatro iteraciones con una duración total de doce semanas.

En la primera iteración se crean los repositorios, se incorporan las librerías necesarias para realizar las primeras peticiones y termina con un pequeño prototipo funcional.

Durante la segunda iteración se realizan los primeros gráficos, se piensa en un futuro diseño y termina con una segunda tutoría presencial donde se presenta la aplicación y se proponen cambios.

La tercera iteración es la más importante y donde hubo un gran rendimiento, se incorporan los cambios y se termina básicamente de programar toda la aplicación.

En la cuarta iteración se realizan las últimas pruebas y se aplican los cambios necesarios para desplegar la aplicación en producción.

Tarea	Comienzo	Fin
Sprint 1	**mié 02/03/22**	**jue 24/03/22**
Primer prototipo funcional	jue 24/03/22	jue 24/03/22
Sprint 2	**jue 21/04/22**	**jue 12/05/22**
Segundo prototipo funcional	jue 12/05/22	jue 12/05/22
Sprint 3	**lun 27/06/22**	**lun 25/07/22**
Aplicación final	lun 25/07/22	lun 25/07/22
Sprint 4	**mar 26/07/22**	**lun 01/08/22**
Aplicación desplegada en la nube	lun 01/08/22	lun 01/08/22

Tabla 2: Planificación iteraciones

En el Anexo I se puede consultar toda la planificación temporal del proyecto detallada y las tareas asociadas a cada iteración.

5.4.1. Primera iteración

Esta primera iteración tiene una duración de tres semanas, comenzando el día 02/03/2022 y finalizando el 24/03/2022 con la obtención de un primer prototipo funcional.

Backlog:

- Como usuario quiero una aplicación para poder visualizar datos sobre titulaciones, asignaturas y grupos.
- Como usuario quiero poder navegar fácilmente por la aplicación.
- Como desarrollador quiero una aplicación con una arquitectura cliente-servidor.
- Como desarrollador quiero documentar mi API REST.
- Como desarrollador quiero obtener los datos de los servicios web de diaweb.

Desarrollo:

Durante la fase de estudios previos se crearon los proyectos, creando una división entre el cliente y el servidor, como si de dos aplicaciones independientes se tratasen, dotando a al proyecto de una arquitectura cliente-servidor, en esta iteración se inician los

repositorios en GitHub y se conectan a los proyectos, se comienzan a hacer los primeros commits.

Para documentar la API REST se implementó Swagger, a partir de la importación de la librería springdoc-openapi-ui en el archivo pom.xml del back. De esta manera, todos los endpoints que se agregarían en un futuro estarían documentados.

Para poder navegar fácilmente por la aplicación se instaló vue-router en el front, se importó en el proyecto y se creó el archivo index.js con todas las rutas. A lo largo de las siguientes iteraciones este archivo será modificado, añadiendo las nuevas rutas según las nuevas vistas creadas.

Para obtener los datos de los servicios web se crearon los services que realizaban las peticiones SOAP y modelaban los datos, este punto fue complicando debido a que nunca se había realizado este tipo de peticiones y era necesario construir mensajes en formato XML. Las instrucciones necesarias que el código realiza para obtener los datos son:

1. Crear la dirección del servicio, la URL donde se encuentra el endpoint con la información
2. Crear contexto SSL y confiar en todos los certificados, este paso fue añadido durante la tercera iteración debido a que el protocolo de diaweb cambió de HTTP a HTTPS, en la descripción de esta iteración se explicará el problema con mayor profundidad.
3. Abrir conexión a la dirección del servicio.
4. Establecer los parámetros:
    - Configuración del modo de envío: GET.
    - Establecer el formato de datos: XML.
    - Establecer entrada y salida debido a que la conexión predeterminada creada no tiene permisos de lectura y escritura.
5. Crear la solicitud SOAP.
6. Enviar la solicitud SOAP en una secuencia.
7. Recibir la respuesta del servidor y construir el modelo de datos.
8. Cerrar la conexión a la dirección del servicio.

Con ayuda de la herramienta Postman se iban realizando pruebas unitarias a los endpoints creados en la API, se creó una colección con peticiones a todos los recursos.

Para poder visualizar los datos en el front era necesario realizar peticiones a la API, para ello se utilizó axios y se mostraron los datos en una tabla mapeando a elementos el array recibido con v-for en la parte de código HTML.
El motivo por el que las peticiones al servidor se realizan con protocolo HTTP es debido a que se utiliza axios, una biblioteca que facilita las peticiones al servidor y nos ahorra tener que crear certificados SSL para peticiones HTTPS, además de que estas peticiones se tendrían que realizar con Fetch, que no es posible usar en todos los navegadores.
Se crearon los services, las llamadas con axios y las tablas necesarias para poder mostrar tanto las titulaciones, como las asignaturas de una titulación y los grupos de teoría y práctica de una asignatura. Para navegar entre estos componentes se utilizó v-on:click para manejar los eventos desde los templates de Vue al pulsar en una fila de la tabla.

Hito de finalización:
Se obtuvo un primer prototipo funcional en el que un usuario podía desplazarse fácilmente entre las distintas vistas, visualizando las titulaciones, las asignaturas y los grupos.

5.4.2. Segunda iteración

Esta segunda iteración tiene una duración de tres semanas, comenzando el día 21/04/2022 y finalizando el 12/05/2022 con la obtención de un segundo prototipo mejorado.

Backlog:

- Como usuario quiero una aplicación para poder visualizar gráficos sobre las titulaciones, asignaturas y grupos.
- Como usuario quiero poder acceder a la aplicación desde un dispositivo móvil.
- Como desarrollador quiero una aplicación responsive con una interfaz de usuario mejorada.
- Como desarrollador quiero agregar filtros, paginación y ordenación a las tablas.

Desarrollo:
Al comienzo de esta fase, ya se tiene un prototipo funcional sobre el que trabajar. En primer lugar, se mejoró la interfaz de la aplicación; para ello, se añadió una barra de navegación con iconos de Google Material Icons y un footer adaptativos, se importó el

tipo de letra Fira Sans y se crearon animaciones en CSS. Además, se creó un componente que mostraba un gif de carga mientras el front recibía del back los datos solicitados.
Para poder acceder a la aplicación desde un dispositivo móvil se tuvo que cambiar el host del front y habilitar las CORS en el back para que cualquier cliente pudiese acceder a los recursos del servidor.
La parte que más tiempo se tardó en desarrollar fue la visualización de gráficos, se probaron distintas librerías que ayudasen a cumplir este objetivo, pero o no funcionaban como se esperaba o eran muy complejas de manejar. Finalmente se consiguió mostrar gráficos haciendo uso de ChartJS. Programando diferentes funciones en JavaScript se permitía al usuario elegir el color de los gráficos y ocultar las barras de un profesor.
Por último, se agregó una retícula de tres columnas:

- En la primera, se encontraba un filtro para poder buscar datos concretos en la tabla.
- En la segunda, se encontraba un formulario para poder introducir el número de filas de cada página de la tabla.
- En la tercera, se encontraba un desplegable para ordenar la tabla por el campo seleccionado de manera ascendente o descendente.

El problema de estas tres funcionalidades es que había que escribir una gran cantidad de líneas de código y que éstas no eran reutilizables. En cada vista que tuviera una tabla se tenía que modificar el código cambiando el nombre de los campos.

Hito de finalización:
Se obtuvo un segundo prototipo con toda la funcionalidad del primero, además de poder visualizar gráficos, ordenar, filtrar y paginar los datos de las tablas, junto con un diseño mejorado, accesible también desde cualquier dispositivo.

El día 12/05/2022 se realiza la segunda reunión con el tutor; en ella se presenta la aplicación, se resuelven dudas para poder seguir con el desarrollo y se dictan los cambios a realizar:

- Era indispensable mostrar la carga docente de los profesores del Excel que se entregó durante la fase de estudios previos; hasta el momento todos los datos eran obtenidos de los servicios web de diaweb.
- Los datos que se mostraban en los gráficos no aportaban demasiada información.

- Algunas tablas contenían columnas con información innecesaria.
- Se encontraron algunos errores ortográficos.

5.4.3. Tercera iteración

Esta tercera iteración tiene una duración de cuatro semanas, comenzando el día 27/06/2022 y finalizando el 25/07/2022. Es el sprint con mayor duración pero en el que se trabaja con un mayor rendimiento y se obtiene la aplicación final, completando así la mayoría de los objetivos.

Backlog:

- Como desarrollador quiero obtener y mostrar los datos del Excel.
- Como desarrollador quiero reducir código y que éste sea reutilizable.
- Como desarrollador quiero que mi aplicación tenga páginas de error.
- Como usuario quiero gráficos que me aporten más información.
- Como usuario quiero una interfaz de usuario más simple.

Desarrollo:

La primera semana de esta iteración se destinó a conseguir leer y operar con los datos del Excel. Aquí se encuentra el primer gran problema: el documento contiene una gran cantidad de información dividida en distintas hojas y con muchas celdas combinadas.

Únicamente era necesaria la información de la primera hoja, el problema era que tenía datos referenciados de otras hojas, por lo que no se podía eliminar el resto de las hojas. Y el problema de las celdas combinadas hacía muy complejo el desarrollo de la lógica que manejase esos datos.

La solución fue crear un script en Python utilizando la biblioteca Pandas. Este pequeño programa leía la primera hoja del Excel original y volcaba los datos en otro nuevo Excel. De este modo, se consigue tener los datos de la primera hoja y los referenciados de las otras en una única hoja y sin celdas combinadas. Ya solo faltaba eliminar las columnas que tuviesen información que no era necesaria.

La utilización de la librería Pandas fue propuesta por el tutor Rodrigo Santamaría Vicente; sin su ayuda no se hubiese conseguido llegar a la solución propuesta.

Una vez que los datos estaban en un Excel de una sola hoja sin celdas combinadas, se agregaron las librerías poi y poi-ooxml en el archivo pom.xml del back. Con esas librerías se podía desarrollar la lógica necesaria para moverse por las celdas del archivo obteniendo los datos necesarios en cada servicio.

Una vez que esa información era modelada, se creaban los controladores REST que la devolvían al front para ser mostrados en tablas acompañadas de gráficos.

Utilizando la librería vue3-easy-data-table, se dejaron de mostrar los datos con v-for. Además, ya solo era necesario configurar las tablas para tener los filtros, paginación y ordenación sin programar todas las funciones; estas funcionalidades eran proporcionadas por la librería. Por consiguiente, el número de líneas de código se redujo notablemente y estas nuevas tablas eran reutilizables únicamente cambiando el nombre de los campos.

La librería ChartJS se cambió por vue-apexcharts, una librería menos compleja de manejar y sobre la que se volcaron datos que aportaban más información.

En esta iteración se agregó una vista que mostraba el error 404 cuando el usuario intentaba acceder a una página inexistente de la aplicación y otra vista que mostraba un mensaje, en caso de que un servicio no devolviese ninguna información.

La barra de navegación y el footer fueron eliminados y sustituidos por una barra lateral. De esa manera, se simplificaba la interfaz de usuario y se mostraban los accesos a las distintas páginas de una manera más sencilla e intuitiva.

Durante esta iteración se descubre otro gran problema, toda la información manejada en sprints anteriores ya no estaba disponible debido a se obtenía de los servicios web de diaweb, que habían cambiado el protocolo de HTTP a HTTPS, una versión más segura. Para solucionar este problema el código Java tuvo que ser modificado añadiendo los siguientes cambios:

- Sustituir la apertura de conexión HTTP por conexión HTTPS.
- Agregar una clase con una implementación ficticia de la interfaz HostnameVerifier que confía en todos los hosts.
- Agregar una clase ficticia que implementara X509TrustManager para confiar en todos los certificados.
- Crear un contexto SSL confiando en todos los certificados.
- Realizar la petición SOAP a través de HTTPS sin tener certificados válidos.

Hito de finalización:

Se obtuvo una aplicación web completa con una interfaz de usuario más sencilla y que cumplía todos los objetivos funcionales a excepción del despliegue en producción.

5.4.4. Cuarta iteración

Esta cuarta y última iteración tiene una duración de una semana, comenzando el día 26/07/2022 y finalizando el 01/08/2022 con la aplicación desplegada en producción.

Backlog:

- Como usuario quiero una aplicación web completa.
- Como desarrollador quiero una aplicación web desplegada en producción.

Desarrollo:

Durante esta iteración se corrigen pequeños errores y se pule el diseño final de aplicación.
Se añade un video en la página principal explicando la aplicación y un formulario en la página de contacto, haciendo uso de la librería EmailJS para que los usuarios puedan proponer futuros cambios o dar su opinión.
Asimismo, se realizan pruebas funcionales para verificar que todo funciona correctamente y pruebas de carga y rendimiento con Postman al sistema, para determinar su comportamiento en condiciones de carga normal o carga máxima.
Una vez que el código del back ya no va a ser modificado, es necesario exportar y construir el ejecutable Jar.
Para desplegar el back en la plataforma AWS es necesario:

- Crear una instancia EC2, una máquina virtual en la nube que funcionará como servidor y generar un par de claves para acceder a él.
- Copiar el archivo Jar al EC2.
- Acceder por SHH a la máquina virtual e instalar Java 1.8.
- Ejecuta el archivo Jar.
- Habilitar el puerto 8080 en el grupo de seguridad de la instancia.

La dirección IP estática asignada es 54.174.126.218 y se puede comprobar que el servidor funciona correctamente accediendo a la documentación de la API:

http://54.174.126.218:8080/swagger

Una vez que se tiene una IP estática asignada, es necesario cambiar en el código del front la dirección donde se encuentra la API, luego se compila el proyecto para poder ser desplegado.

Para desplegar el front es necesario:

- Crear un servicio S3, que proporciona almacenamiento de objetos.
- Desbloquear el acceso al público y generar una política de acceso al servicio.
- Cargan todos los archivos del proyecto compilado.

La dirección ofrecida por AWS para acceder a la aplicación es la siguiente:

http://tfg-client.s3-website-us-east-1.amazonaws.com

Hito de finalización:

Se obtuvo la aplicación final desplegada en producción, cumpliendo así el último objetivo funcional.

6. Conclusiones

Una vez ha finalizado el desarrollo del proyecto se exponen las conclusiones explicando individualmente el cumplimiento de los objetivos funcionales, no funcionales y personales descritos en el apartado 2.

6.1. Cumplimiento de objetivos funcionales

- Diseño de una interfaz que facilite la visualización de la carga docente:

Para justificar el cumplimiento de este primer objetivo se comentan las características que tiene la interfaz de usuario de la aplicación:

- Clara: La información es transmitida de manera precisa gracias a tablas y gráficos.
- Coherente: El usuario puede desarrollar patrones de uso y aprender la función de los diversos botones, iconos y elementos de la interfaz.
- Interactiva: La interfaz muestra los datos de manera rápida.
- Familiar: El usuario se siente familiarizado con todos los elementos y puede acceder a la página deseada en cada momento gracias al menú lateral.
- Eficiente: El usuario es capaz de visualizar lo que busca en el momento sin opciones adicionales sobre lo que desea hacer, que demoran el proceso y dañan la experiencia del usuario.
- Atractiva: Se utilizan iconos de Google Material Icons, el tipo de letra Fira Sans y una paleta de colores simple y flexible que permite distinguir todos los elementos de la interfaz.

- Conexión o realización de una BBDD sobre la carga docente del departamento de informática y automática:

La aplicación obtiene los datos de dos fuentes diferentes:

- Conexión con una BBDD Excel con la carga docente del Departamento de Informática y Automática facilitada por el tutor.
- Llamadas a los servicios web de diaweb que devuelven distinta información de la BBDD del Departamento de Informática y Automática.

- Búsqueda de profesores o asignaturas.

Se ha desarrollado una aplicación que permite buscar tanto profesores como asignaturas haciendo uso de filtros y visualizando la información en tablas acompañadas de gráficos.

- Relación entre asignaturas, profesores y cursos.

Las tablas y gráficos que se acaban de nombrar muestran datos sobre las asignaturas, profesores y cursos y su relación entre ellos.

- Realización de gráficos sencillos de carga:

Los gráficos sencillos de carga de este objetivo eran scatter plots e histogramas, por lo que no se ha cumplido correctamente.

Utilizando la librería vue-apexcharts se consiguen mostrar otros tipos de gráficos que ayudan a visualizar relaciones en los datos:

- Gráfico de barras horizontales para el número de titulaciones por centro.
- Gráfico de barras horizontales para el número de asignaturas por titulación.
- Gráfico de barras horizontales para el número de horas por profesor.
- Gráfico de anillos para el número de alumnos por grupo de teoría y práctica.
- Gráfico de anillos para el número de horas de una asignatura por profesor.
- Gráfico de anillos para el número de horas de un profesor por asignatura.

- Puesta a punto en producción para el curso siguiente a aquél en que se desarrolle:

Durante el cuarto sprint el proyecto se desplegó la aplicación en producción en la nube de AWS con la creación de una instancia EC2 para el back y un servicio S3 para el front, conectados entre ellos, de esta manera el usuario puede acceder a la aplicación en cualquier momento y desde cualquier dispositivo, a través de la siguiente dirección:

http://tfg-client.s3-website-us-east-1.amazonaws.com

6.2. Cumplimiento de objetivos no funcionales

- Accesibilidad:

Toda la información que maneja la aplicación es pública, por lo que no se desarrolló la funcionalidad de iniciar sesión, haciendo que la aplicación sea accesible para todos los usuarios.

- Disponibilidad:

Debido a que la aplicación ha sido desplegada en la nube, el sistema se encuentra disponible en estado operativo en todo momento.

- Escalabilidad:

Actualmente se tiene contratada una instancia EC2 t2.micro con 1 vCPU y 1 GiB, en cualquier momento se puede contratar una máquina con un mayor número de vCPU y memoria para hacer más grande el sistema y no perder calidad en los servicios ofrecidos en caso de que la aplicación contase con muchos usuarios.

- Portabilidad:

El sistema puede ser ejecutado en cualquier sistema que tenga instalado Java y Node.
Al tratarse de una aplicación web es accesible desde cualquier plataforma y dispositivo que tenga un navegador web.

- Rendimiento:

El sistema desarrollado apenas consume recursos.

- Usabilidad:

La interfaz de usuario de aplicación es sencilla e intuitiva, el cumplimiento de este objetivo también se argumenta en el punto anterior debido a que también es un requisito funcional.

6.3. Cumplimiento de objetivos personales

- Completar los objetivos funcionales y no funcionales anteriormente propuestos:

Se acaba de argumentar como tanto los objetivos funcionales como los no funcionales han sido cumplidos con éxito.

- Desarrollar y conectar ambas partes de una aplicación cliente-servidor y obtener un producto completo útil para otras personas:

La aplicación web desarrollado puede ser útil para que los profesores que quieran consultar su carga docente, además para cualquier otro usuario interesado en la información que se maneja.
Se ha logrado desarrollar una aplicación con una arquitectura cliente-servidor programada en Vue y Spring Boot en la que ambas partes se comunican entre ellas.

- Demostrar todo el conocimiento adquirido durante la carrera:

Se ha conseguido poner en práctica y asentar los conocimientos adquiridos a lo largo del Grado en Ingeniería Informática de asignaturas como Sistemas Distribuidos, Ingeniería del Software, Programación, entre otras.

- Aprender y manejar diferentes herramientas, técnicas y lenguajes de programación.

Se han adquirido una gran cantidad de conocimientos sobre las herramientas utilizadas durante el desarrollo del proyecto, principalmente en los lenguajes de programación JavaScript y Java, el framework Vue, la tecnología Spring Boot y la metodología ágil scrum.
La experiencia obtenida durante el proceso ha sido gratificante y se desarrollado un gran interés en seguir con el aprendizaje de Spring Boot y la especialización en AWS.

7. Líneas futuras de trabajo

Se ha conseguido desarrollar una aplicación completamente funcional cumpliendo en gran medido los objetivos marcados, pero siempre quedan aspectos que mejorar o nuevas funcionalidades que añadir, en este apartado se describirán las principales ideas de mejora.

Por falta de tiempo no se pudo programar una funcionalidad muy deseada desde el principio del desarrollo, la posibilidad de que en la propia aplicación se pudiera gestionar la carga docente de los profesores, quitando horas a unos profesores y añadiéndoselas a otros, de esta manera se produciría escritura en la base de datos, y no solo lectura. Respecto a esta posible modificación, sería interesante añadir otra funcionalidad como la descarga de la base de datos modificada en formato Excel o similares, con la aplicación presentada esto carece de sentido debido a que el contenido es el original, proporcionado al comienzo del proyecto.

Otra de las mejoras sería tener la aplicación desplegada en un dominio personalizado HTTPS y un certificado SSL propio, esta mejora es fácil de implementar gracias a las herramientas que proporciona AWS, como puede ser Amazon Route 53.

8. Bibliografía

[1] TechTarget, Contributor (2019, agosto). *Web application (Web app)*. TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

[2] Britannica, T. Editors of Encyclopaedia (2021, 13 de mayo). *client-server architecture*. Encyclopedia Britannica. https://www.britannica.com/technology/client-server-architecture

[3] Sarangam, A. (2020, 30 de diciembre). *What Is Client Server Architecture? An Overview*. Jigsaw Academy. https://www.jigsawacademy.com/blogs/cyber-security/what-is-client-server-architecture

[4] Yambadwar, S. (2022, 23 de agosto). *HTTP Full Form*. GeeksforGeeks. https://www.geeksforgeeks.org/http-full-form

[5] Rugamba, M. (2018, 1 de noviembre). *HyperText Transfer Protocol — HTTP*. Medium. https://medium.com/@rrugamba/hypertext-transfer-protocol-http-f7c0665f7695

[6] Gupta, L. (2022, 7 de abril). *What is REST*. REST API Tutorial. https://restfulapi.net

[7] Segura, J. A. (1994). *WORLD WIDE WEB: UN SISTEMA HIPERMEDIA DISTRIBUIDO PARA LA DOCENCIA UNIVERSITARIA*. Nuevas Tecnologías de la Información Aplicadas a la Educación. https://nti.uji.es/docs/nti/badajoz.html

[8] Marica, S. (2021, 23 de julio). *Understanding REST APIs and their Constraints*. Web Scraping API. https://www.webscrapingapi.com/rest-api-architecture-constraints

[9] Rajput, D. (2016, 15 de octubre). *What is REST? REST Architecture and REST Constraints* Dinesh on Java. https://www.dineshonjava.com/what-is-rest-and-rest-architecture-and-rest-constraints

[10] *Top 3 benefits of REST APIs*. MuleSoft. https://www.mulesoft.com/resources/api/top-3-benefits-of-rest-apis

[11] Google Cloud. *¿Qué es la arquitectura de microservicios?*. Google Cloud. https://cloud.google.com/learn/what-is-microservices-architecture

[12] Microservice Architecture. *What are microservices?*. Microservice Architecture. https://microservices.io

[13] Deshpande, P. (2013). *Metamorphic Detection Using Function Call Graph Analysis*. San José State University. https://scholarworks.sjsu.edu/etd_projects/336

[14] Azure. *What is the cloud?*. Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-the-cloud

[15] Digité. *¿Qué es Scrum?*. Digité. https://www.digite.com/es/agile/que-es-scrum

[16] Flores, F. (2022, 22 de julio). *Qué es Visual Studio Code y qué ventajas ofrece*. OpenWebinars. https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece

[17] Azam, S. (2020). *What is Vue.js, and Why is it Cool?*. linuxhint. https://linuxhint.com/about_vue_js

[18] Hayes, A. (2020, 13 de noviembre). *HyperText Markup Language – HTML*. Investopedia. https://www.investopedia.com/terms/h/html.asp

[19] Parvez, F. (2021, 1 de agosto). *Introduction to CSS | CSS Tutorial for Beginners*. Great Learning. https://www.mygreatlearning.com/blog/css-tutorial

[20] Hack Reactor (2021, 26 de agosto). *What is JavaScript used for?*. Hack Reactor. https://www.hackreactor.com/blog/what-is-javascript-used-for

[21] García de Zúñiga, F. (2019, 5 de abril). *Axios Javascript: analizamos las características de este ligero cliente HTTP*. Arsys. https://www.arsys.es/blog/axios

[22] HC200ok (2022, 13 de agosto). *Easy Data Table Component For Vue 3*. Vue Script. https://www.vuescript.com/easy-data-table-component

[23] HC200ok (2022, 9 de junio). *Modes*. vue3-easy-data-table. https://hc200ok.github.io/vue3-easy-data-table-doc/modes.html

[24] ApexCharts. *What is ApexCharts?*. ApexCharts. https://apexcharts.com

[25] StackShare. *What is EmailJS?*. StackShare. https://stackshare.io/emailjs

[26] Gaba, I. (2022, 11 de julio). *What is Maven: Here's What You Need to Know*. Simplilearn. https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven

[27] baeldung. (2022, 16 de abril). *A Guide to Spring in Eclipse STS*. Baeldung. https://www.baeldung.com/eclipse-sts-spring

[28] Tutorialspoint . *Spring Boot – Introduction*. Tutorialspoint. https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm

[29] Guru99 (2022, 3 de julio*). ¿Qué es Java Platform?*. Guru99. https://guru99.es/java-platform

[30] TechTarget, Contributor (2019, agosto). *Swagger*. TechTarget. https://www.techtarget.com/searchapparchitecture/definition/Swagger

[31] Pedamkar, P. *What is SoapUI?*. EDUCBA. https://www.educba.com/what-is-soapui

[32] Javatpoint. *Postman Tutorial*. Javatpoint. https://www.javatpoint.com/postman

[33] Python. *What is Python? Executive Summary*. Python. https://www.python.org/doc/essays/blurb/

[34] Ravulakollu, N. *Introduction to Python Pandas for Beginners*. AlmaBetter. https://www.almabetter.com/blogs/introduction-to-python-pandas-for-beginners

[35] Page, V. (2021, 12 de agosto). *What Is Amazon Web Services and Why Is It So Successful?*. Investopedia. https://www.investopedia.com/articles/investing/011316/what-amazon-web-services-and-why-it-so-successful.asp

[36] AWS. *What is Amazon S3?*. AWS. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[37] Wigmore, I. (2021, julio). *Amazon EC2 instance*. TechTarget. https://www.techtarget.com/searchaws/definition/Amazon-EC2-instances

[38] Git. https://git-scm.com

[39] Gaba, I. (2022, 29 de julio). *What is GitHub And How To Use It?*. Simplilearn. https://www.simplilearn.com/tutorials/git-tutorial/what-is-github

[40] Techopedia. (2022, 10 de agosto). *Microsoft Word*. Techopedia. https://www.techopedia.com/definition/3840/microsoft-word

[41] Keup, M.(2022, 22 de marzo). What Is Microsoft Project?. ProjectManager. https://www.projectmanager.com/blog/what-is-microsoft-project

[42] Visual Paradigm. https://www.visual-paradigm.com

[43] ISA. *REM*. ISA. https://www.isa.us.es/3.0/tool/rem