FIBAjuda

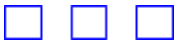
Aplicación para dar clases de repaso a estudiantes de la FIB

TRABAJO DE FIN DE GRADO

Eduard Ortuño Garrote

Directora: María José Casañ Guerrero
Junio 2021

Quiero expresar mi agradecimiento a María José Casañ Guerrero por haber aceptado la propuesta de ser mi ponente en este Trabajo Final de Grado (TFG), por su tiempo dedicado a las reuniones y sus consejos, ayuda, supervisión y rigor durante todas las etapas del proyecto.



Finalmente, agradezco a toda mi familia y a mis compañeros su apoyo y confianza durante toda la carrera, en aquellos momentos buenos y malos.
Gracias por acompañarme.

Resumen

Muchas veces, cuando los estudiantes de la FIB tenemos algún problema o queremos recibir ayuda extra para seguir alguna asignatura concreta especialmente difícil, nos faltan soluciones.

Este proyecto pretende hacer la vida de estos estudiantes más sencilla, a través de una aplicación web llamada “FIBAjuda”, la cual permite que los estudiantes del grado de Ingeniería Informática puedan ofrecerse para dar clases de repaso a otros estudiantes que lo necesiten a cambio de una remuneración pactada.

La solución propuesta en este trabajo puede beneficiar directamente a los estudiantes porque disponen de una ayuda continua para seguir aquella o aquellas asignaturas que más les cuesta. Este proyecto también puede ayudar a los estudiantes de forma indirecta, ya que ayudará a reducir los gastos que estos tendrían si tuvieran que pagar a un profesor particular o matricularse en una academia.

Resum

Moltes vegades, quan els estudiants de la FIB tenim algun problema o volem rebre ajuda extra per a seguir alguna assignatura concreta especialment difícil, ens falten solucions.

Aquest projecte pretén fer la vida d'aquests estudiants més senzilla, a través d'una aplicació web anomenada "FIBAjuda", la qual permet que els estudiants del grau d'Enginyeria Informàtica puguin oferir-se per fer classes de repàs a altres estudiants que ho necessitin a canvi d’una remuneració pactada.

La solució proposada en aquest treball pot beneficiar directament als estudiants perquè disposen d’una ajuda contínua per a seguir aquella o aquelles assignatures que més els hi costa. Aquest projecte també pot ajudar als estudiants de manera indirecta, ja que ajudarà a reduir les despeses que aquests haguessin de pagar a un professor particular o matricular-se a una acadèmia.

Abstract

Many times, when FIB students have a problem or want to receive some extra help to follow a specific subject that we find especially hard, we lack solutions.

The purpose of this project is to make the lives of these students easier, through a web application called "FIBAjuda", which allows students of the Computer Engineering degree to advertise themselves as teachers in order to give support classes to other students who need it in exchange for an agreed remuneration.

The solution proposed in this work can directly benefit students because they have continuous help to follow the one subject or those subjects that are the hardest for them. This project can also help students indirectly since it will help reduce the expenses that they would have if they had to pay a private teacher or to be enrolled in an academy.

Acrónimos y siglas

API Application Programming Interface 43-45, 60, 63-66, 69, 74, 75, 78

BSON Binary JSON 48, 59

DOM Document Object Model 59

ECTS European Credit Transfer System 16

GEI Grau en Enginyeria Informàtica 15

HTML HyperText Markup Language 57

HTTP Hypertext Transfer Protocol 43, 60, 64

HTTPS Hypertext Transfer Protocol Secure 64

IETF Internet Engineering Task Force 62

JSON JavaScript Object Notation 48, 59, 62

JSX JavaScript Syntax Extension 57

JWT JSON Web Token 62, 63

LOPD Ley Orgánica de Protección de Datos de Carácter Personal 18, 42

MVC Modelo-Vista-Controlador 43

REST Representational State Transfer 43, 44, 60, 74

RGPD Reglamento General de Protección de Datos 18

TFG Trabajo de Fin de Grado 36, 37, 66, 71

UML Unified Modeling Language 48

URL Uniform Resource Locator 62, 64, 79-81

XML Extensible Markup Language 57

Glosario

Agile Conjunto de metodologías para el desarrollo de proyectos que precisan de rapidez y flexibilidad para adaptarse a condiciones cambiantes del sector o mercado, aprovechando dichos cambios para proporcionar ventaja competitiva. Es decir, el proyecto se divide en pequeñas partes que tienen que completarse y entregarse en pocas semanas. 20, 22, 34

Backend Parte de una aplicación que se encarga de gestionar la lógica de negocio y de proveer información al frontend, por eso decimos que el backend está en el lado del servidor, corresponde a la capa de acceso a datos. 43-45, 57, 60, 68, 69, 75, 76

Base de datos no relacional Base de dato NoSQL (“Not Only SQL”), que no cumple con el modelo relacional, es decir, los datos no tienen porqué estar relacionados entre sí y por lo tanto no tienen que almacenarse en estructuras fijas como las tablas del modelo de base de datos relacional. 59

Clave Primaria Campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. 68

Clúster Un clúster fragmentado en MongoDB es una colección de conjuntos de datos distribuidos en muchos fragmentos (servidores) para lograr una escalabilidad horizontal y un mejor rendimiento en las operaciones de lectura y escritura. 61

Deployment Mecanismo a través del cual aplicaciones, módulos, actualizaciones y parches son entregadas desde los desarrolladores a los usuarios. 66

Diagrama de Gantt Herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. 27-29, 71, 73

Endpoint Direcciones web que reciben o retornan información de un Web API. 64, 68, 69

Feedback Acto de ofrecer información sobre el resultado de un proceso o de parte de un proceso. Puede involucrar desde consejos, comentarios y evaluaciones. 70, 78

Framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. 27, 59

Frontend Parte de una aplicación que interactúa con los usuarios, por eso decimos que está del lado del cliente, corresponde a la capa de presentación. 43-45, 57-59, 61-63, 68, 69

Hardware Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático. 33

Historia de usuario Pequeña descripción del requerimiento de un cliente. 21, 23, 27, 31, 38-40, 57, 61, 65, 71, 74

Iteración Es un ciclo de tiempo que se va repitiendo. Tras cada iteración, el producto o servicio gana valor. Este se revisa hasta que el cliente queda conforme, por lo que es frecuente facturar por iteración. 20

Librería Conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca. 59, 64

Plan de contingencia Fondos destinados al plan o modelo sistemático de actuación que tiene por objeto anticiparse a situaciones en las que esté próximo un daño o en que exista la posibilidad de que éste suceda o no. 33, 34

Patrón Esqueleto de una solución a un problema común en el desarrollo de software. 43-45

Scrum Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. 20

Software Soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas. 12, 21, 26-28, 31, 32, 37, 43, 44, 51, 66, 67, 75, 76

Stakeholder Persona, organización o empresa que tiene interés en un proyecto. 13

String Secuencia ordenada y limitada de caracteres que se suele utilizar para representar texto. 46-50

Testing Proceso empírico para encontrar bugs o fallos en un programa, ya sea manualmente o automáticamente. 13, 19, 68

Tabla de contenidos

Índice de tablas

Índice de figuras

1. Introducción

1.1. Contextualización

Mi proyecto *“Aplicació per oferir i donar classes de repàs a estudiants de la FIB”* trata de un trabajo de fin de grado para la Facultad de Informática de Barcelona (FIB), concretamente para el grado en Ingeniería Informática en la especialidad de Ingeniería del Software.

En un intento de hacer más sencilla la vida de los estudiantes de la FIB, presento una aplicación web llamada “FIBAjuda” la cual permite que los estudiantes del grado de Ingeniería Informática puedan ofrecerse para dar clases de repaso a otros estudiantes que lo necesiten a cambio de una remuneración pactada.

1.2. Descripción del problema

Actualmente, los estudiantes de la FIB, cuando tienen algún problema o quieren recibir ayuda de alguna asignatura en concreto, tienen varias opciones a decidir.

La primera opción sería preguntar a los compañeros de su propio grupo de amistades dentro de la facultad, o a los compañeros del grado a través de los múltiples grupos de WhatsApp que hay disponibles. Esta opción es relativamente rápida debido a la velocidad a la que se puede responder a estos mensajes, pero puede carecer de veracidad, ya que la persona que te está respondiendo puede no tener los conocimientos necesarios para resolver las dudas expuestas.

Una segunda opción sería acordar una reunión con el profesor que imparte la asignatura en cuestión y que éste resolviera las dudas del estudiante. Si bien esta opción es la que puede ofrecer una mayor fiabilidad de las respuestas dado el conocimiento del profesor, también es verdad que estas reuniones suelen ocurrir con menos frecuencia de lo habitual y exigen una mayor preparación de las preguntas y dudas a hacer.

Y por última opción, el alumnado de la FIB siempre puede acudir a las llamadas “academias” que imparten varios cursos de diferentes asignaturas del grado. En esta opción entra en juego por primera vez la remuneración económica. Estos cursos suelen estar impartidos por exalumnos de la FIB que han sido subcontratados por la empresa para realizar las clases de repaso. Esto hace que si bien los profesores de estos cursos de repaso cobran por ello, hace que el precio de las clases sea un poco elevado.

Además, hay que tener en cuenta que estamos viviendo una pandemia, lo que ha hecho que cada vez haya menos contacto entre estudiantes y menos vida en el campus, lo cual puede derivar en problemas académicos o de rendimiento de algunos alumnos por no tener las herramientas necesarias.

1.3. Stakeholders

Los stakeholders son las denominadas partes interesadas en el proyecto y que por lo tanto son afectadas por la existencia o la actividad de este. Estas partes son de gran importancia para el proyecto, ya que nos ayudarán a definir los requisitos del sistema a través de sus requisitos e intereses.

También es importante mencionar que no solo se ha de identificar bien estas partes, sino intentar mantenerlas involucradas en el proyecto, ya que esto ayudará a que el producto final cumpla con los requisitos necesarios.

Alumnos de la FIB
Ya que el proyecto tiene como objetivo desarrollar una aplicación para que los estudiantes puedan dar o recibir clases de repaso de diferentes asignaturas, estos estudiantes tiene un papel muy importante, ya que serán los principales beneficiados gracias a este proyecto. Dentro de este gran grupo se podría dividir en dos subgrupos con intereses diferentes, los estudiantes que van a dar clase y los que van a recibirla.

Profesores de la FIB
Si bien es cierto que la aplicación está enfocada principalmente a los estudiantes, los profesores también podrían formar parte de este proyecto en un futuro.

Directora del Trabajo de Final de Grado
La directora del proyecto María José Casañ Guerrero, es miembro del “Departament d’Enginyeria de Serveis i Sistemes d’Informació” y está involucrada, ya que tiene que supervisar el proyecto e intentar que vaya en una buena dirección.

Desarrollador
En este caso yo, Eduard Ortuño Garrote, seré el encargado de desarrollar la aplicación, hacer las pruebas de testing y documentar correctamente el proyecto. Yo soy uno de los principales stakeholders del proyecto, ya que este trabajo influirá en mi aprendizaje y puede que también en el mundo laboral posterior.

Alumnos y profesores de otros campus
En caso de que este proyecto fuera exitoso dentro de la FIB, podría exportarse a otros campus de la UPC o incluso a otras universidades.

2. Justificación

2.1. Análisis de mercado

He realizado un análisis de mercado de aplicaciones que permiten a usuarios dar clases particulares a otros usuarios de varias materias.

Tusclasesparticulares

Descripción:
Tusclasesparticulares [1] es una aplicación web que te permite encontrar profesores que se encuentren cerca de ti para darte clases de repaso o anunciarte tú como profesor o inscribir tu negocio.
Hay anuncios de profesores particulares que ofrecen clases particulares en su casa, a domicilio u online. También hay clases y cursos impartidas en academias, centros de formación y escuelas de idiomas. Además en esta página también se puede encontrar formación para empresas y autónomos.

Funcionalidades:

- Permite buscar clases, academias, alumnos, intercambio de idiomas y empleo para profesores dependiendo de la ubicación seleccionada.
- Posee un programa de confianza con centros adscritos a la aplicación.
- Los usuarios profesores tienen un perfil con valoraciones y comentarios, una descripción de su trabajo y fotos y videos.
- Tiene un buscador de clases que permite filtrar por materia que se quiere aprender, lugar, tipo de impartición (desplazándose a algún centro/academia, a domicilio u online), nivel de las clases y otros filtros.
- Las clases online se pueden dar por webcam/videoconferencia a través de Classgap, Skype, Zoom...

Classgap

Descripción:
Classgap [2] es una aplicación web la cual permite recibir clases online en directo, con un profesor particular asignado para cada persona. Pertenece a la misma corporativa que la aplicación anterior Tusclasesparticulares.
Esta aplicación dispone de una aula virtual que tiene material de soporte para hacer las clases: Videoconferencia, pizarra, edición de documentos online…

Funcionalidades:

- Dispone de un aula virtual que cuenta con las siguientes funciones: video, chat, pizarra digital, editor de textos colaborativo, posibilidad de compartir tu pantalla…
- Los usuarios profesores tienen un perfil con valoraciones y comentarios, una descripción de su trabajo, fotos, videos, sus tasas y sus horarios.
- Los usuarios alumnos pueden solicitar reservar una clase con un profesor o iniciar un chat con ellos a través de su perfil.
- Tiene un buscador de clases que permite filtrar por materia que se quiere aprender, por idioma y por precio. También permite buscar por palabras claves.

Infoclases

Descripción:
Infoclases [3] es una aplicación web que permite encontrar profesores particulares filtrando por materia y territorio. También es posible anunciarse como profesor y publicitarse a través de convertirse en un miembro “Premium”, característica la cual permite acceso al Aula Virtual, que es la herramienta de enseñanza en línea que Infoclases ofrece y que hace posible dar clases online a los alumnos, independientemente que se haya contactado con ellos a través de Infoclases o no.

Funcionalidades:

- Tiene un buscador que permite filtrar por criterio (clases particulares, clases online o alumnos), por materia que se quiere aprender y por ubicación.
- Dispone de un blog donde hay artículos para profesores, padres y alumnos que leen, enseñan, comparten y aprenden.
- Los usuarios profesores tienen un perfil con una descripción de su trabajo, una foto de perfil, qué materias imparte ese profesor, cómo se harán las clases (online, a domicilio, por webcam…) y las zonas en las que da clase el profesor.
- Permite contactar con los profesores sin compromiso a través de su perfil.
- Dispone de una aula virtual accesible para los usuarios profesores premium, que tiene funciones como subir material didáctico o grabar las clases para que los alumnos puedan repasar después. Dentro de la clase, se puede utilizar la pantalla compartida, la pizarra virtual, etc.

Aula lliure

Aula lliure [4] es una iniciativa de la FIB (lanzada el Q1 del curso 2017/18) que está diseñada para que los estudiantes de fase inicial del GEI puedan aprovechar la

experiencia académica de los estudiantes más veteranos y así mejorar su rendimiento académico.
Los alumnos veteranos asumen el rol de formadores voluntarios dando clases de repaso a los estudiantes de fase inicial, pero solo de las asignaturas que se imparten en su cuatrimestre natural, en el Q1 - F, IC, FM, PRO1 - y en el Q2 - M1, M2, EC, PRO2-.
Las clases son de 2 horas por asignatura y se programan un mínimo de 10 sesiones. Los formadores voluntarios reciben: 2 ECTS por actividades extrauniversitarias y una bonificación del 20% en el orden de matrícula del siguiente cuatrimestre.

2.2. Resumen funcionalidades

App / Funcionalidad	Recibir clases	Dar clases	Perfil con valoraciones	Compensación económica	Proporciona medios para clase online	Proporciona medios para clase física
Tus clases particulares	X	X	X	X	X	
Classgap	X		X	X	X	
Infoclases	X	X		X	X	
Aula lliure	X	X				X
Mi app (FIBAjuda)	X	X	X	X	X	X

Tabla 1: Resumen de funcionalidades. Elaboración propia

En la Tabla 1 podemos ver un resumen de las funcionalidades de todas las aplicaciones nombradas en el estudio de mercado realizado anteriormente, además de mi aplicación en la última fila.

A partir de este estudio he decidido crear una aplicación en el marco de la FIB que recogiera todas las funcionalidades útiles para los estudiantes. Es decir, una aplicación que permita a los estudiantes tener un perfil propio como en cualquier red social, donde tendrán valoraciones y comentarios de otros usuarios, además de acordar clases con estudiantes veteranos a cambio de un precio negociable. La aplicación también dará soporte para que los estudiantes tengan más fácil hacer clases online (proporcionar algún tipo de aula virtual) y físicas (permitir que los usuarios puedan reservar aulario de la universidad).

3. Alcance

3.1. Objetivos

En esta sección se definen los objetivos del proyecto, que trata de una aplicación web para los estudiantes de las FIB que permita dar clases de repaso de diferentes asignaturas que se imparten en la facultad. Los usuarios de esta aplicación podrán ser estudiantes o profesores de la FIB, que tendrán un perfil con una descripción personal, además de comentarios y valoraciones de otros usuarios. Los objetivos principales serían los siguientes:

- **Permitir que los usuarios puedan iniciar sesión en la aplicación**
  Los usuarios tienen que ser capaces de poder iniciar sesión en la aplicación en cualquier momento y desde cualquier dispositivo.

- **Permitir que los usuarios puedan editar su perfil**
  Los usuarios tienen que ser capaces de entrar y modificar su perfil en cualquier momento, cambiando su descripción, las asignaturas que imparten o el precio de estas.

- **Permitir que los usuarios puedan buscar profesores**
  Los usuarios tienen que ser capaces de buscar profesores dependiendo de algunos criterios, como de qué asignatura quieren el profesor, el rango de precio de estas clases o la disponibilidad.

- **Permitir que los usuarios valoren a otros usuarios**
  Al tratarse de una aplicación web dónde las opiniones de otros usuarios tienen tanta importancia, los usuarios tienen que poder valorar y comentar en el perfil de un profesor que les haya dado alguna clase.

- **Permitir que los usuarios se comuniquen entre ellos**
  Los usuarios tienen que ser capaces de comunicarse entre ellos a través de un chat en vivo, para por ejemplo poder acordar una clase con el profesor.

3.2. Requisitos no funcionales

Una vez descritos los principales objetivos del proyecto, se exponen a continuación una serie de requisitos no funcionales que se han de satisfacer para garantizar un correcto funcionamiento de la aplicación:

- **Apariencia:** La aplicación tiene que ser atractiva y gustar visualmente a los usuarios que la utilicen.

- **Usabilidad:** La aplicación tiene que ser fácil de utilizar para todos los distintos tipos de persona, roles y perfiles, además de proporcionar mensajes de error que sean informativos y orientados al usuario final.

- **Adaptabilidad:** La aplicación tiene que ser capaz de funcionar correctamente en cualquier clase de dispositivo.

- **Disponibilidad:** El sistema tiene que estar operativa en cualquier momento y hay que evitar que ocurran caídas que puedan afectar su funcionamiento.

- **Internacionalización:** La plataforma tiene que ser internacional, es decir, tiene que estar disponible al menos en las tres lenguas que se hablan en el campus (catalán, castellano e inglés).

- **Seguridad y privacidad:** La aplicación tiene que asegurarse de que todas las funcionalidades y datos sean accesibles solo para aquellos usuarios autorizados a utilizarlas.

- **Legislación:** La plataforma debe cumplir con todas las leyes de protección de datos (RGPD Y LOPD).

- **Escalabilidad y extensibilidad:** La aplicación tiene que estar preparada para soportar un incremento de usuarios y de negocios (como por ejemplo incluir otras carreras).

3.3. Riesgos

Hay una serie de posibles riesgos que podrían afectar en el desarrollo del proyecto y por tanto afectar la calidad de este. Para minimizar el impacto, es primordial conocer de antemano los riesgos existentes que pueden hacer que el desarrollo del proyecto se vea afectado. Algunos de estos riesgos son:

- **Inexperiencia en las tecnologías utilizadas:**
  El hecho de utilizar tecnologías en las cuales el desarrollador no es experto (o incluso alguna de ellas no la ha utilizado nunca) hace que el tiempo de aprendizaje sea más alto y por tanto pueda alargarse el tiempo necesario para el desarrollo.

- **Fecha de entrega fija:**
  El hecho de tener una fecha de entrega fija hace que, si se tiene cualquier contratiempo en el desarrollo o en el testing de la aplicación, no se pueda retrasar la entrega del producto y por lo tanto este pierda calidad.

- **Bugs:**
  El hecho de la aparición de errores en el código o “bugs” a lo largo del proyecto es un punto clave. Dependiendo del momento en el que estos bugs sean detectados puede afectar mucho al tiempo que será necesario invertir para arreglarlos.

- **Coronavirus:**
  El hecho de estar trabajando en un proyecto como este mientras estamos en medio de una pandemia global tiene sus consecuencias. Existe el riesgo de quedarse confinado o de ser positivo, lo que podría llegar a imposibilitar por completo el desarrollo del proyecto.

- **Veracidad de la información:**
  Al tratarse de una aplicación basada en las opiniones de los usuarios sobre los profesores, puede ocurrir algún boicot hacia alguna persona en algún momento o que algún usuario introduzca opiniones sin sentido o que no son reales.

4. Metodología

4.1. Introducción

La elección de metodología es una de las decisiones más condicionantes en los proyectos, ya que marcará la forma de trabajar siendo determinante a la hora de alcanzar los objetivos previamente descritos. Esta decisión no es sencilla, ya que hay que tener muy claro los requisitos del proyecto, así como las ventajas e inconvenientes de cada metodología para poder ver cuál es la que se adecua más al proyecto.

A la hora de decidir la de este proyecto, el factor que más ha pesado ha sido experiencia con ciertas metodologías que ya había utilizado anteriormente en otros proyectos.

El desarrollo del proyecto se llevará a cabo utilizando una metodología Agile, concretamente una metodología pseudo-scrum. Scrum es una metodología iterativa e incremental, de forma que se define un nombre de iteraciones o sprints breves, de entre una y tres semanas (en este caso los sprints tendrán una duración de 3 semanas), y en cada iteración se añade valor al producto final. En este caso se utilizará una metodología pseudo-scrum, ya que scrum normalmente se aplica en equipos, y en este caso está adaptada a un solo desarrollador.

Además, al utilizar una metodología Agile habría que hacer reuniones diarias (también conocidas como “dailys”). En este caso, periódicamente, yo, Eduard, tendré reuniones o meetings con mi tutora María José Casañ para mostrarle el producto que tenga en ese momento, y así poder tomar decisiones que influyan al desarrollo o al diseño de la aplicación.

4.2. Herramientas de seguimiento

4.2.1. Taiga

Se utilizará la herramienta online Taiga[5] para tener un control sobre las historias de usuario y poder visualizarlas en un tablero, conociendo en todo momento su estado y viendo si están incluidas en algún sprint. Para ello, se dividirá el tablero en tres columnas (“To do”, “Doing” y “Done”) y las historias se irán moviendo entre estas columnas a medida que se vayan haciendo y vaya cambiando su estado. Cada historia de usuario tendrá una etiqueta para incluirla en un grupo según su ámbito, además de una descripción y unos criterios de aceptación.

4.2.2. Github

Como sistema de gestión de versiones se utilizará Github[6], un sistema de control de versiones muy utilizado en el mundo del software. A través de este sistema se puede llevar a cabo un control exhaustivo de todas las modificaciones que se van haciendo en el código, lo que permite tener un control de los cambios y así detectar los errores de una forma más rápida.

En este caso vamos a trabajar con dos ramas principales, la master y la development. En la rama master estará el código ya testeado y listo para producción, mientras que en la rama development será la rama donde se irán añadiendo las funcionalidades que vayan siendo desarrolladas.

5. Planificación temporal

El proyecto va a estar dividido en seis etapas. La primera etapa es la de gestión del proyecto, que al estar siguiendo una metodología Agile, sería la etapa equivalente a la *inception*, que consiste en definir el alcance del proyecto, estimar costes, hacer una planificación aproximada, definir riesgos, etc (vendría a ser todo lo que se hace en la asignatura de GEP). Después de esta etapa inicial, habrá cuatro etapas de desarrollo, más conocidas como sprints. Estos sprints tendrán una duración aproximada de unas 2-3 semanas cada uno. Por último, habrá una etapa dedicada a acabar la documentación del proyecto y su memoria que tendrá una duración aproximada de una semana.

Cada sprint tendrá unas tareas asignadas, que consistirán en, además de desarrollar la funcionalidad especificada, hacer la documentación necesaria. Al final de cada sprint habrá una retrospectiva, para valorar el trabajo realizado y decidir si hay que tomar algunas decisiones para mejorar la metodología de trabajo o arrastrar alguna tarea al siguiente sprint.

Etapa	Data de inicio	Data de finalización
Gestión del proyecto	15/02/2021	21/03/2021
Sprint I	22/03/2021	18/04/2021
Sprint II	19/04/2021	09/05/2021
Sprint III	10/05/2021	30/05/2021
Sprint IV	31/05/2021	17/06/2021
Memoria	18/06/2021	27/06/2021

Tabla 2: Fechas planificadas de cada etapa. Elaboración propia.

Las tareas las dividiremos y estarán identificadas por su tipo: Gestión del Proyecto (GP), Desarrollo (DS) y Documentación (DC).

5.1. Descripción de las tareas

5.1.1. Gestión del proyecto

- **GP1 - Contextualización y alcance:** Definir el contexto y el alcance del proyecto.
  Duración: 20h
  Dependencias: -

- **GP2 - Planificación temporal:** Definir y planificar las tareas del proyecto.
  Duración: 20h
  Dependencias: GP1

- **GP3 - Gestión económica y sostenibilidad:** Definir un plan de gestión económica y un informe de sostenibilidad.
  Duración: 20h
  Dependencias: GP2

- **GP4 - Documentación fase inicial:** Recoger las tres entregas anteriores y agruparlas en un mismo documento, haciendo las correcciones necesarias.
  Duración: 20h
  Dependencias: GP3

- **GP5 - Definir historias de usuario:** Definir las historias de usuario y ponerlas en el Taiga.
  Duración: 20h
  Dependencias: -

5.1.2. Sprint I

- **DS1 - Inicio:** Preparar el entorno de trabajo, descargando los programas necesarios y crear la estructura del proyecto.
  Duración: 20h
  Dependencias: -

- **DS2 - Autentificación:** Desarrollar el sistema de autentificación, es decir: registro de nuevos usuarios, inicio de sesión, cifrar contraseña.
  Duración: 35h
  Dependencias: DS1

- **DS3 - Modelos:** Crear todos los modelos necesarios para que el sistema funcione.
  Duración: 5h
  Dependencias: DS1

- **DS4 - Editar perfil:** Permitir que todos los usuarios puedan editar los datos de su perfil, como asignaturas que imparten, precios, habilidades, disponibilidad...
  Duración: 20h
  Dependencias: DS2

- **GP6 - Retrospectiva Sprint I:** Valorar el trabajo realizado en el Sprint I, buscar formas de mejorar o solucionar problemas.
  Duración: 5h
  Dependencias: DS1, DS2, DS3, DS4

- **DC1 - Documentación Sprint I:** Redactar la documentación necesaria con relación al Sprint I.
  Duración: 15h
  Dependencias: -

5.1.3. Sprint II

- **DS5 - Buscador de profesores:** Crear la pantalla para buscar profesores, pudiendo filtrar por asignatura, rango de precio o disponibilidad.
  Duración: 45h
  Dependencias: Sprint I

- **DS6 - Acordar clase con profesor:** Permitir que un alumno pueda acordar una clase con un profesor de su elección.
  Duración: 25h
  Dependencias: DS5

- **DS7 - Consultar precio asignatura:** Permitir que los usuarios puedan consultar el precio medio de cualquier asignatura.
  Duración: 15h
  Dependencias: DS5

- **GP7 - Retrospectiva Sprint II:** Valorar el trabajo realizado en el Sprint II, buscar formas de mejorar o solucionar problemas.
  Duración: 5h
  Dependencias: DS5, DS6, DS7

- **DC2 - Documentación Sprint II:** Redactar la documentación necesaria con relación al Sprint II.
  Duración: 15h
  Dependencias: -

5.1.4. Sprint III

- **DS8 - Chat en vivo:** Crear un chat en vivo entre los profesores y sus alumnos asignados.
  Duración: 40h
  Dependencias: Sprint II

- **DS9 - Valoraciones profesores:** Permitir que los alumnos puedan dejar valoraciones y comentarios en el perfil del profesor, y que otros usuarios puedan consultar estas opiniones.
  Duración: 45h
  Dependencias: DS8

- **GP8 - Retrospectiva Sprint III:** Valorar el trabajo realizado en el Sprint III, buscar formas de mejorar o solucionar problemas.
  Duración: 5h
  Dependencias: DS8, DS9

- **DC3 - Documentación Sprint III:** Redactar la documentación necesaria con relación al Sprint III.
  Duración: 15h
  Dependencias: -

5.1.5. Sprint IV

- **DS10 - Aula virtual:** Adaptar o diseñar un aula virtual que permita a los profesores compartir documentos, videos, etc.
  Duración: 45h
  Dependencias: Sprint III

- **DS11 - Soporte clases online:** La aplicación tiene que dar soporte para que los profesores puedan realizar clases online, adaptando algún sistema de streaming.
  Duración: 45h
  Dependencias: DS10

- **GP9 - Retrospectiva Sprint IV:** Valorar el trabajo realizado en el Sprint IV, buscar formas de mejorar o solucionar problemas.

Duración: 5h
Dependencias: DS10, DS11

- **DC4 - Documentación Sprint IV:** Redactar la documentación necesaria con relación al Sprint IV.
  Duración: 15h
  Dependencias: -

5.1.6. Memoria

- **DC5 - Finalizar memoria:** Acabar de redactar el documento final y empezar a preparar la defensa.
  Duración: 40h
  Dependencias: Sprint IV

5.2. Recursos

5.2.1. Recursos humanos

En este proyecto hay cinco roles diferenciados: jefe de proyecto, analista de requisitos, arquitecto de software, programador y *tester*. Sin embargo, al ser este Trabajo Final de Grado realizado por una sola persona, será la encargada de asumir los diferentes roles en función a la tarea a realizar. A continuación se explican las funciones que tiene cada rol:

- Jefe de proyecto: Se encarga de la correcta planificación del proyecto, además de ser la persona que toma las decisiones y documenta todo el trabajo.

- Analista de requisitos: Se encarga de documentar todas las restricciones y todos los requisitos que tiene que cumplir cada funcionalidad.

- Arquitecto de software: Se encarga de diseñar la arquitectura del software a desarrollar.

- Programador: Se encarga de desarrollar el código de cada funcionalidad, siguiendo la arquitectura especificada y cumpliendo con los requisitos necesarios.

- *Tester*: Se encarga de realizar las pruebas de validez del sistema. Debe diseñar las pruebas, ejecutarlas y presentar un informe con los errores encontrados para que puedan ser arreglados.

5.2.2. Recursos materiales

El único recurso material utilizado en este proyecto es un ordenador de sobremesa. A través de este ordenador se realizarán tanto las tareas de gestión del proyecto como las de desarrollo o documentación del mismo. En cuanto a software, serán necesarios los siguientes:

- Visual Studio Code: IDE para el *framework* tanto del *front* como del *back*.
- Github: Herramienta para el control de versiones y repositorios.
- Office 365: Herramientas para la documentación del proyecto.
- Atenea: Fuente de donde sacar los recursos necesarios para la redacción y desarrollo del trabajo.
- Google Chrome: Herramienta para buscar información necesaria.
- Gantt Project: Herramienta para crear el diagrama de Gantt del proyecto.

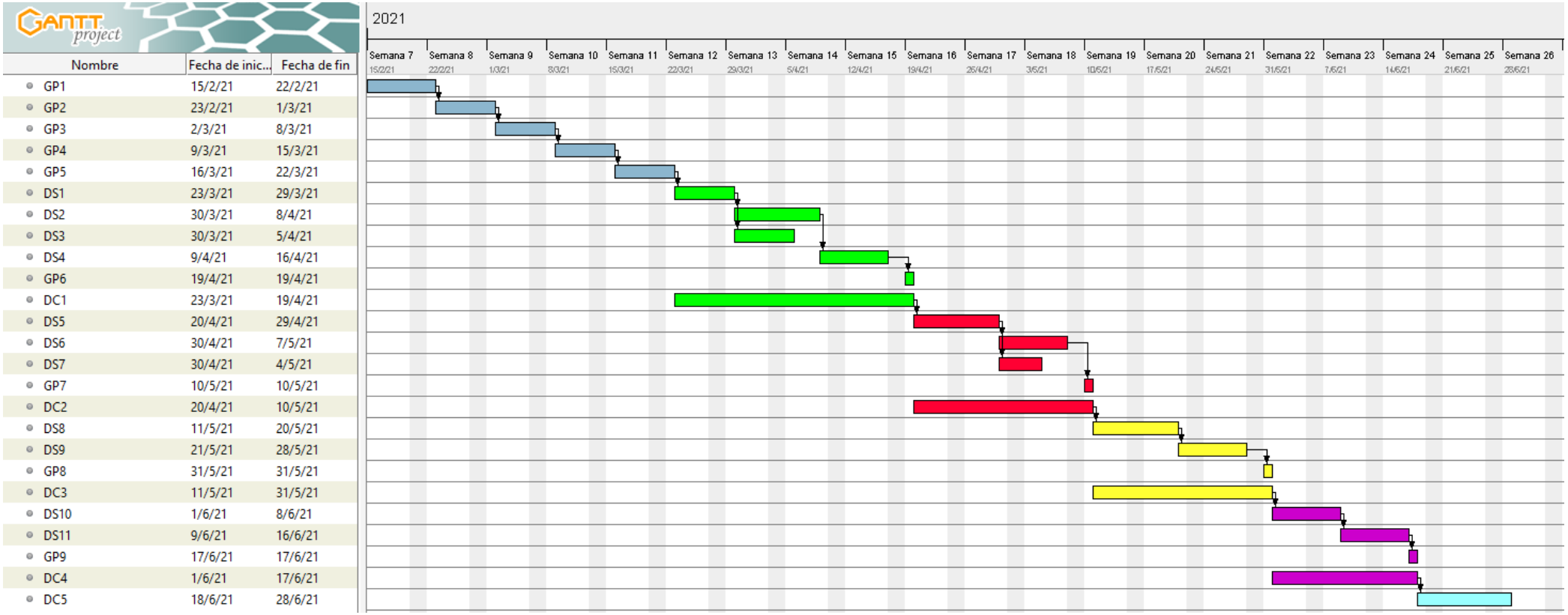
5.3. Estimación de las tareas

Id	Tarea	Tiempo	Dependencias	Recursos	Roles
GP1	Contextualización y alcance	20h		O, A, GC	JP
GP2	Planificación temporal	20h	GP1	O, A, GC, GP	JP
GP3	Gestión económica y sostenibilidad	20h	GP2	O, A, GC	JP
GP4	Documentación fase inicial	20h	GP3	O, A, GC	JP
GP5	Definir historias de usuario	20h	GP4	O, A, GC	JP, AR
DS1	Inicio	20h	GP5	VSC, G, GC	AR, AS, P

DS2	Autentificación	35h	DS1	VSC, G, GC	P, T
DS3	Modelos	5h	DS1	VSC, G, GC	P
DS4	Editar perfil	20h	DS2	VSC, G, GC	P, T
GP6	Retrospectiva Sprint I	5h	DS3, DS4		JP, AR, AS, P, T
DC1	Documentación Sprint I	15h		O, A, GC	JP
DS5	Buscador de profesores	45h	GP6, DC1	VSC, G, GC	P, T
DS6	Acordar clase con profesor	25h	DS5	VSC, G, GC	P, T
DS7	Consultar precio asignatura	15h	DS5	VSC, G, GC	P, T
GP7	Retrospectiva Sprint II	5h	DS6, DS7		JP, AR, AS, P, T
DC2	Documentación Sprint II	15h		O, A, GC	JP
DS8	Chat en vivo	40h	GP7, DC2	VSC, G, GC	P, T
DS9	Valoraciones profesores	45h	DS8	VSC, G, GC	P, T
GP8	Retrospectiva Sprint III	5h	DS9		JP, AR, AS, P, T
DC3	Documentación Sprint III	15h		O, A, GC	JP
DS10	Aula virtual	45h	GP8, DC3	VSC, G, GC	P, T
DS11	Soporte clases online	45h	DS10	VSC, G, GC	P, T
GP9	Retrospectiva Sprint IV	5h	DS11		JP, AR, AS, P, T
DC4	Documentación Sprint IV	15h		O, A, GC	JP
DC5	Finalizar memoria	40h	GP9, DC4	O, A, GC	JP
TOTAL		**560h**			

Tabla 3: Tabla resumen de tareas con su duración, dependencias, recursos y roles. Elaboración propia.
Roles: Jefe de proyecto - JP, Analista de requisitos - AR, Arquitecto de software - AS, Programador - P, Tester - T.
Recursos: Visual Studio Code - VSC, Github - G, Office 365 - O, Atenea - A, Google Chrome - GC, GanttProject - GP

5.4. Diagrama de Gantt



Nombre	Fecha de inic...	Fecha de fin
GP1	15/2/21	22/2/21
GP2	23/2/21	1/3/21
GP3	2/3/21	8/3/21
GP4	9/3/21	15/3/21
GP5	16/3/21	22/3/21
DS1	23/3/21	29/3/21
DS2	30/3/21	8/4/21
DS3	30/3/21	5/4/21
DS4	9/4/21	16/4/21
GP6	19/4/21	19/4/21
DC1	23/3/21	19/4/21
DS5	20/4/21	29/4/21
DS6	30/4/21	7/5/21
DS7	30/4/21	4/5/21
GP7	10/5/21	10/5/21
DC2	20/4/21	10/5/21
DS8	11/5/21	20/5/21
DS9	21/5/21	28/5/21
GP8	31/5/21	31/5/21
DC3	11/5/21	31/5/21
DS10	1/6/21	8/6/21
DS11	9/6/21	16/6/21
GP9	17/6/21	17/6/21
DC4	1/6/21	17/6/21
DC5	18/6/21	28/6/21

Figura 1: Diagrama de Gantt

6. Gestión de los riesgos

A continuación, en la Tabla 4, podemos ver los diferentes riesgos y obstáculos mencionados anteriormente que nos podemos encontrar durante la realización del proyecto, dónde podemos ver el impacto que tendría, el riesgo de que suceda y finalmente el plan alternativo para mitigar estos obstáculos.

Riesgo	Impacto	Probabilidad	Plan alternativo
Inexperiencia en las tecnologías utilizadas	Alto	Alta	Las tareas que implican tecnologías en las cuales se tiene poca experiencia o no se tiene han estado sobreestimadas en cuanto a las horas
Fecha de entrega fija	Medio	Media	Se han planificado unas retrospectivas que sirven entre otras cosas, para hacer replanificaciones de los siguientes sprints en caso de que sea necesario
Bugs	Medio	Alta	Se ha definido un rol de *tester* que será el encargado de solucionar los bugs de todas las funcionalidades en el momento en que se estén desarrollando
Coronavirus	Alto	Baja	Al estar realizando el proyecto desde casa, en caso necesario podría seguir trabajando si fuera posible
Veracidad de la información	Alto	Baja	Se ha definido un rol de administrador dentro de la aplicación, que será el encargado de gestionar las quejas de los usuarios

Tabla 4: Riesgos y planes alternativos. Elaboración propia.

Ninguno de estos riesgos especificados tendrían un impacto directo en el tiempo de desarrollo del proyecto, ya que durante la planificación de este se han tenido en cuenta y se han definido roles, reuniones y horas extra dentro de las tareas donde podrían llegar a ocurrir estos riesgos.

7. Gestión económica

7.1. Identificación y estimación de los costes

7.1.1. Costes de personal

Partiendo de la planificación de las tareas, ahora se calcula el coste del personal. Se tienen en cuenta los cinco roles definidos anteriormente: jefe de proyecto, analista de requisitos, arquitecto de software, programador y *tester*. En la Tabla 5 se ve el coste por hora de cada puesto, los datos han sido obtenidos a través de la herramienta *Payscale* [7].

Rol	**Coste por hora**
Jefe de proyecto	22,82€/h
Analista de requisitos	15,80€/h
Arquitecto de software	23,23€/h
Programador	18,48€/h
Tester	15,99€/h

Tabla 5: Tabla con los costes estimados por rol y hora. Elaboración propia.

En la Tabla 6 se detallan el coste de cada tarea a partir de los costes del personal de la Tabla 5, y se estima el coste de la seguridad social multiplicando el coste por hora estimado por 1,3.

Id	**Tarea**	**Tiempo**	**Roles**	**Coste (€)**
GP1	Contextualización y alcance	20h	JP	593.32
GP2	Planificación temporal	20h	JP	593.32
GP3	Gestión económica y sostenibilidad	20h	JP	593.32
GP4	Documentación fase inicial	20h	JP	593.32
GP5	Definir historias de usuario	20h	JP, AR	1.004,12
DS1	Inicio	20h	AR, AS, P	1.495,26

DS2	Autentificación	35h	P, T	1.568,38
DS3	Modelos	5h	P	120,12
DS4	Editar perfil	20h	P, T	896,22
GP6	Retrospectiva Sprint I	5h	JP, AR, AS, P, T	626,27
DC1	Documentación Sprint I	15h	JP	444,99
DS5	Buscador de profesores	45h	P, T	2.016,49
DS6	Acordar clase con profesor	25h	P, T	1.120,27
DS7	Consultar precio asignatura	15h	P, T	672,16
GP7	Retrospectiva Sprint II	5h	JP, AR, AS, P, T	626,27
DC2	Documentación Sprint II	15h	JP	444,99
DS8	Chat en vivo	40h	P, T	1.792,44
DS9	Valoraciones profesores	45h	P, T	2.016,49
GP8	Retrospectiva Sprint III	5h	JP, AR, AS, P, T	626,27
DC3	Documentación Sprint III	15h	JP	444,99
DS10	Aula virtual	45h	P, T	2.016,49
DS11	Soporte clases online	45h	P, T	2.016,49
GP9	Retrospectiva Sprint IV	5h	JP, AR, AS, P, T	626,27
DC4	Documentación Sprint IV	15h	JP	444,99
DC5	Finalizar memoria	40h	JP	1.186,64
TOTAL		**560h**		**22.206,61**

Tabla 6: Tabla con los costes estimados de cada tarea además del total. Elaboración propia.

7.1.2. Costes genéricos

En este apartado se hablará de los costes en cuanto a recursos materiales y recursos indirectos. En cuanto a los recursos materiales, como bien se ha comentado en apartados anteriores, tan solo se trata de la amortización de un ordenador de sobremesa (además de dos pantallas), ya que los productos software utilizados son gratuitos. Podemos ver su coste en la Tabla 7. Para calcular las amortizaciones usaremos la siguiente fórmula con una dedicación de 5 horas al día:

$$\frac{\text{(Coste (€) * Durada proyecto (h))}}{\text{(Vida útil(años) * 220(días laborables/año) * dedicación/día(h))}}$$

A la hora de calcular la amortización de los elementos de la tabla, usaremos su valor amortizable, que es la diferencia entre su valor inicial y su valor residual. Al haber comprado este material hace dos meses aproximadamente, aún no se ha devaluado apenas su valor, por lo tanto tendremos en cuenta el 90% del importe pagado.

Elemento	Precio compra	Vida útil	Amortización
Ordenador de sobremesa	1500€	4 años	171,81€
BenQ GW2280	100€	4 años	11,45€
Acer Nitro VG240YS	200€	4 años	22,9€

Tabla 7: Costes de los recursos hardware. Elaboración propia.

En cuanto a los costes de carácter general o indirectos, tenemos algunos básicos como la electricidad o el internet. En este caso, el trabajo no lo hará un equipo de 5 trabajadores sino que lo hará todo una sola persona desde casa. Pero aun así, como podemos ver en la Tabla 8, se realizará el cálculo haciendo la suposición de que trabajan un equipo de 5 personas en una oficina de co-working en Barcelona. En concreto, en una oficina de *Aurea co-working* [8], que incluye conectividad a Internet, acceso a impresora/escáner, sala de reuniones y acceso las 24 horas del día los 7 días de la semana.

Elemento	Precio/Mes	Precio/Mes (con IVA)	Meses	Total
Aurea co-working	220€	266,2€	4	1.064,8€

Tabla 8: Costes indirectos. Elaboración propia.

7.1.3. Contingencia

Como en todo proyecto, es importante añadir un sobrecoste en relación con los posibles obstáculos e imprevistos. En este caso, se ha establecido un porcentaje específico del 10%. A continuación, en la Tabla 9, podemos ver los cálculos de la contingencia concretos para cada uno de los costes del proyecto.

Tipo de coste	Coste	Contingencia
Personal	22.206,66€	2.220,67€
Material	206,16€	20,61€
Indirecto	1.064,8€	106,48€

Tabla 9: Tabla contingencia del 10% por tipo de coste. Elaboración propia.

7.1.4. Imprevistos

Los imprevistos vistos en el apartado de gestión de los riesgos afectan principalmente a la planificación temporal, provocando desviaciones en los plazos de las tareas. Pero como bien se explica en ese apartado, las tareas de riesgo ya han sido sobreestimadas. Además, los mecanismos que nos proporciona la metodología Agile nos permite ajustar toda la planificación de estas tareas en caso necesario. De esta forma, se supondrá que los costes de estos posibles desviamientos temporales ya están cubiertos.

Por lo que hace a los imprevistos en cuanto a los recursos materiales, en la Tabla 10 podemos ver el coste de la posible avería de alguno de estos aparatos.

Elemento	Precio reparación	Probabilidad (%)	Coste
Ordenador de sobremesa	150€	10	15€
BenQ GW2280	50€	15	7,5€
Acer Nitro VG240YS	50€	15	7,5€

Tabla 10: Costes de imprevistos. Elaboración propia.

7.1.5. Coste total

Una vez presentados todos los costes del proyecto, en la Tabla 11 podemos ver el presupuesto final del trabajo. El coste total del proyecto es de 25.855,38 euros.

Tipo de coste	Coste
Personal	22.206,66€
Material	206,16€
Indirecto	1.064,8€
Contingencia	2.347,76€
Imprevistos	30€
Coste total	**25.855,38€**

Tabla 11: Presupuesto final del proyecto. Elaboración propia.

7.2. Control de gestión

Una vez presentado el presupuesto final del proyecto, en esta sección se explica el control que se llevará a cabo sobre las posibles desviaciones que pueda haber en el presupuesto. Para conseguir esto, vamos a intentar anotar el tiempo real que nos ha llevado desarrollar cada tarea. De esta manera, podremos saber la desviación que tendremos en nuestro presupuesto. Con el objetivo de obtener este control, definimos las siguientes métricas:

- Desviación de horas por tarea:
  *(horas_estimadas - horas_reales) * coste_real*

- Desviación coste personal por tarea:
  *(coste_estimado - coste_real) * horas_reales*

- Desviación total en la realización de tareas:
  coste_estimado_total - coste_real_total

- Desviación total de recursos (material, indirectos)
  coste_estimado_total - coste_real_total

- Desviación total coste de imprevistos:
  coste_estimado_imprevistos - coste_real_imprevistos

- Desviación total de horas
  horas_estimadas - horas_reales

8. Sostenibilidad

8.1. Autoevaluación

A lo largo de estos años en la universidad nos han explicado en qué consiste la sostenibilidad y lo importante que es pensar en el medio ambiente cada vez que realizamos un proyecto. Sin embargo, hasta que uno no realiza un trabajo como el TFG, no se pone a pensar en su importancia.

Aunque normalmente si pensamos en sostenibilidad pensamos directamente en la parte económica, es verdad que la parte ambiental y social tienen también una fuerte importancia, aunque suelen pasar de forma más desapercibida. Es verdad que siempre tenemos muy presente la necesidad de hacer un análisis económico para asegurar la viabilidad de un proyecto, pues si no fuera por esto no tendría ningún futuro.

La parte ambiental, desde mi punto de vista, es muy importante sobre todo en proyectos que requieren una gran cantidad de recursos, en los cuales sería necesario un análisis más extenso. Aun así, en proyectos como este donde los recursos son mínimos sigue siendo importante minimizar el impacto sobre el medio ambiente.

Pero también es necesario que el proyecto cubra alguna necesidad social, es decir, mejorar la calidad de vida de las personas que utilicen esta aplicación. Además, a nivel personal este TFG me aportará una experiencia y unos conocimientos necesarios para seguir desarrollándome como ingeniero, aparte de representar el último escalón antes de finalizar el grado.

8.2. Dimensión Económica

El coste estimado de este proyecto solo incluye los recursos necesarios para el correcto desarrollo y funcionamiento de la aplicación, por lo tanto no ha habido ningún malgasto de dinero ni de recursos.

Actualmente, se usan otra clase de medios para realizar clases de repaso dentro del campus, entre ellos las academias. Entonces, el proyecto que se va a desarrollar representa un avance económico hacia los estudiantes, puesto que las clases serán más baratas y los beneficios se quedarán dentro de la universidad, iniciando incluso un propio mercado de conocimientos.

8.3. Dimensión Ambiental

Afortunadamente, la realización de este proyecto no tiene un gran impacto ambiental, solo la fabricación del ordenador que se va a utilizar impacta negativamente en el medio ambiente.

Actualmente, muchos de los métodos de repaso que hay actualmente en la universidad utilizan papel para imprimir documentos y/o pruebas. A través de esta nueva aplicación, la cual dispone de un aula virtual donde se pueden colgar cualquier tipo de documentos, se reducirá de forma considerable el consumo de papel y tóner que actualmente se utiliza.

8.4. Dimensión Social

A nivel personal, este proyecto me puede enriquecer en muchos sentidos. Además de aportarme conocimientos nuevos sobre el desarrollo de software, aprender a utilizar nuevas herramientas, etc., también me sirve cómo una experiencia más o menos realista de cómo funciona el desarrollo de un proyecto.

De cara a la sociedad, y en especial a los estudiantes de la FIB, la creación de esta aplicación ayudará a que la transmisión de conocimientos entre alumnos mejore, haciendo incluso que estos propios estudiantes sean más solidarios entre ellos.

Este TFG nace de la necesidad de tener un propio sistema de ayuda académica a los estudiantes de la FIB, que sea más cómodo, rápido y barato que las soluciones existentes actualmente.

9. Especificación de requisitos

Para obtener los requisitos del proyecto hemos tenido que decidir en primer lugar los problemas a los que el proyecto daría respuesta. Entonces, hemos podido marcar unos requisitos y plantear unos requisitos para dar solución a los problemas iniciales. En este proceso ha sido útil realizar el estudio de mercado de sistemas similares ya existentes. Además, algunos requisitos son comunes a muchas aplicaciones (por ejemplo, el login o logout de usuario), por lo que el proceso de obtención se ha simplificado.

Gracias al procedimiento anterior hemos obtenido una serie de requisitos que hemos clasificado según sean funcionales o no funcionales.

Los requisitos funcionales del proyecto hemos decidido especificarlos como historias de usuario y los subdividimos por temáticas o épicas para poder organizarlos de una forma más entendible y eficaz: acceso al sistema, gestión de usuarios, gestión de clases e informativas.

Los requisitos no funcionales hemos decidido especificarlos usando la plantilla de especificación de requisitos de Volere [9], de forma que no sean presentados en un formato informal y sean lo más técnico posible.

9.1. Requisitos funcionales

9.1.1. Acceso al sistema

Historia de usuario: Inicio de sesión
Descripción: Como usuario ya existente, quiero poder acceder a la aplicación usando mis credenciales de la FIB.

Historia de usuario: Cerrar sesión
Descripción: Como usuario ya existente y con sesión abierta dentro de la aplicación, quiero poder cerrar mi sesión en la aplicación de forma que otro usuario pueda iniciar sesión.

9.1.2. Gestión de usuarios

Historia de usuario: Ver perfil de usuario
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder ver el perfil de otro usuario de la aplicación.

Historia de usuario: Modificar perfil de usuario
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder modificar mi perfil, ya sea porque quiero cambiar mi descripción o mi disponibilidad.

Historia de usuario: Buscar profesores
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder buscar usuarios profesores que estén dados de alta filtrando a través de la asignatura, valoración media y precio.

Historia de usuario: Ver perfil de usuario profesor
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder ver el perfil de un usuario profesor de la aplicación y ver tanto su descripción como su disponibilidad, además de todas sus asignaturas, valoraciones y comentarios.

9.1.3. Gestión de clases

Historia de usuario: Enviar petición de clase
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder enviar una petición de clase de una asignatura, con una modalidad (online o presencial), en un día y un horario a un profesor, añadiendo un comentario para más información.

Historia de usuario: Ver peticiones de clase pendientes
Descripción: Como usuario profesor con sesión abierta dentro de la aplicación, quiero poder ver el número de peticiones de clase pendientes que tengo y esas peticiones.

Historia de usuario: Resolver petición de clase
Descripción: Como usuario profesor con sesión abierta dentro de la aplicación, quiero poder resolver una petición de clase que me haya enviado un alumno previamente, y añadir un comentario a mi resolución tanto si la acepto como si la rechazo.

Historia de usuario: Ver valoraciones pendientes
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder ver el número de valoraciones pendientes que tengo.

Historia de usuario: Valorar usuario profesor
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder valorar a otro usuario profesor con el cual haya tenido una clase anteriormente.

Historia de usuario: Ver próximas clases
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder ver mis próximas clases tanto de profesor como de alumno.

9.1.4. Informativas

Historia de usuario: Visualizar información clases presenciales
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder visualizar información sobre las clases presenciales, pudiendo ver las aulas libres dentro del campus y pudiendo reservar.

Historia de usuario: Visualizar información clases online
Descripción: Como usuario con sesión abierta dentro de la aplicación, quiero poder visualizar información sobre las clases online, pudiendo ver las diferentes plataformas disponibles y sus funcionalidades.

9.2. Requisitos no funcionales

Requisito: El sistema deberá ser atractivo para cualquier usuario que lo utilice.
Categoría: 10a) Apariencia - Look and feel.

Requisito: La aplicación podrá ser usada por usuarios sin entrenamiento.
Categoría: 11a) Facilidad de uso - Usabilidad y Humanidad.

Requisito: El sistema será intuitivo para todos los usuarios de manera que no se necesitará una formación previa para poder utilizarla.
Categoría: 11c) Aprendizaje - Usabilidad y Humanidad.

Requisito: Cualquier interacción entre el usuario y la interfaz de la aplicación tendrá un tiempo máximo de respuesta de 3 s.
Categoría: 12a) Velocidad y Latencia - Rendimiento.

Requisito: La aplicación en cualquier navegador web de cualquier dispositivo.
Categoría: 13a) Entorno físico esperado - Operacional y entorno.

Requisito: El sistema interactuará correctamente con sistemas existentes.
Categoría: 13c) Interacción con sistemas existentes - Operacional y entorno.

Requisito: Se actualizará la aplicación de manera periódica.
Categoría: 14a) Mantenibilidad - Mantenibilidad y soporte.

Requisito: Se dará soporte a los usuarios del sistema.
Categoría: 14b) Soporte - Mantenibilidad y soporte.

Requisito: El sistema será compatible con los sistemas operativos que se quiera utilizar.
Categoría: 14c) Adaptabilidad - Mantenibilidad y soporte.

Requisito: La aplicación evitará que el usuario introduzca datos incorrectos.
Categoría: 15b) Integridad - Seguridad.

Requisito: El servidor de la aplicación se ha protegido frente a ataques.
Categoría: 15e) Inmunidad - Seguridad.

Requisito: El sistema no será ofensivo para ninguna etnia ni religión.
Categoría: 16b) Diversidad cultural - Cultura.

Requisito: La información personal se almacenará según la LOPD.
Categoría: 17a) Cumplimiento de aspectos legales - Cumplimiento.

10. Arquitectura del sistema

En este apartado se explica la arquitectura que se ha aplicado al sistema y qué patrones se han visto implicados en el diseño, además del diagrama de clases global correspondiente a la totalidad del sistema y capturas de pantalla de todas las vistas de la aplicación web.

10.1. Visión general

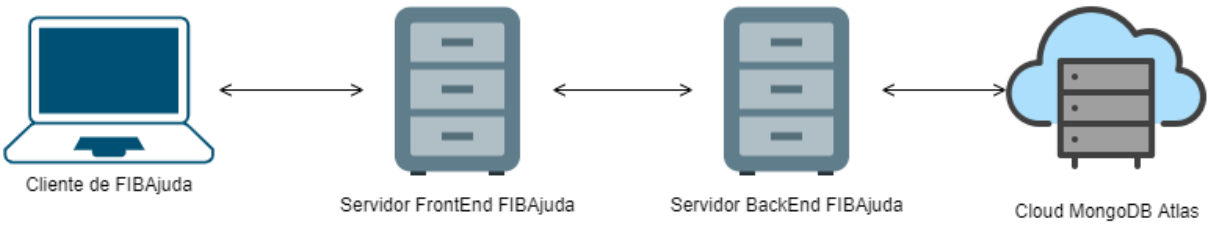


Figura 2: Visión general del sistema

Como podemos ver en la figura 2, el cliente de nuestra aplicación FIBAjuda se conectará al servidor de frontend, que es una aplicación web que se ha desarrollado usando ReactJs[10]. El servidor de frontend se conectará al servidor de backend, que es una API Rest desarrollada con NodeJs[11], a través de llamadas HTTP. Por último, el servidor de backend hará las conexiones necesarias con la cloud de MongoDB Atlas[12], que se encargará de devolver los datos solicitados por la API.

10.2. Patrones implicados

10.2.1. Modelo-Vista-Controlador (MVC)

Modelo-vista-controlador[13] (MVC) es un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

En este proyecto utilizamos este patrón en las dos partes de nuestra aplicación, tanto en el backend como en el frontend. De hecho, es un patrón bastante común.

En el backend lo utilizamos, ya que es un patrón bastante usado cuando se desarrolla una API Rest con NodeJs y MongoDB[14]. En NodeJs se definen todos los modelos utilizados por la aplicación, además de los controladores que son los que interactúan con estos modelos. En MongoDB tendríamos la parte de la vista, ya que podemos visualizar los datos que tenemos de una forma eficiente.

En el frontend lo utilizamos por defecto cuando se trabaja con entidades del backend. Es decir, no tenemos modelos definidos más allá del usuario (para poder llevar el control del usuario activo), pero aun así se definen estructuras de datos, que corresponden a los modelos, las vistas son todas las pantallas o componentes definidos, y como controladores usamos unos servicios que son los encargados de hacer las llamadas a la API.

10.2.2. Cliente-Servidor

La arquitectura cliente-servidor[15] es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y servicios con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se dispone de los requisitos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, etc.

El cliente sería la aplicación web (frontend) y el servidor sería nuestro backend (API Rest y MongoDB).

10.2.3. Fachada

Fachada[16] es un tipo de patrón de diseño estructural. Viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos.

El patrón fachada se aplica cuando se necesita proporcionar una interfaz simple para un subsistema complejo, o cuando se quiere estructurar varios subsistemas en capas, ya que las fachadas son el punto de entrada a cada nivel.

Hemos aplicado este patrón en el frontend, para facilitar su comunicación con el backend, a través de servicios que son los encargados de hacer las llamadas a la API.

10.2.4. Singleton

El patrón Singleton[17] es un patrón de diseño que se usa para garantizar que una clase solo pueda tener una instancia, y que esta sea globalmente accesible en el programa. Se puede pensar que una variable global es a efectos prácticos igual, pero es un razonamiento erróneo, ya que, en primer lugar, el patrón Singleton no expone su constructor al resto de la aplicación y además utiliza el patrón de diseño lazy loading, que evita la inicialización prematura de la instancia de la clase hasta el momento de su uso.

En nuestro proyecto utilizamos este patrón a la hora de guardar y acceder los datos del usuario activo. Disponemos de una clase llamada CurrentUser que utiliza este patrón y permite acceder desde cualquier parte de la aplicación a los tokens de sesión además de datos como su nombre, si es profesor, etc.

10.2.5. Composite

El patrón Composite[18] es un patrón de diseño que sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol.

Utilizamos este patrón en el frontend, ya que utilizamos ReactJs, donde una vista está formada por componentes (como por ejemplo un header, una lista, etc.).

10.3. Esquema de la base de datos

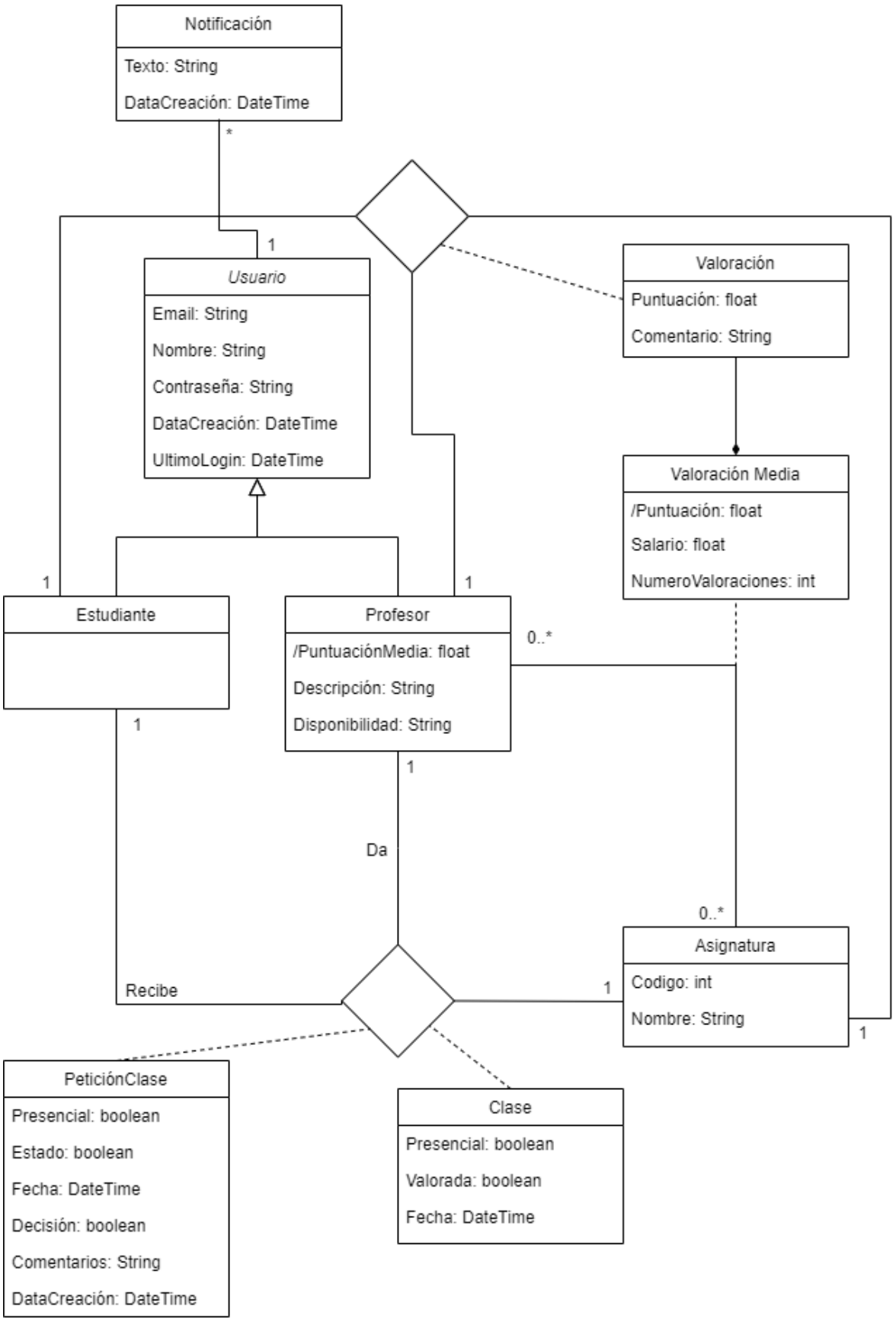


Figura 3: Modelo conceptual de la base de datos

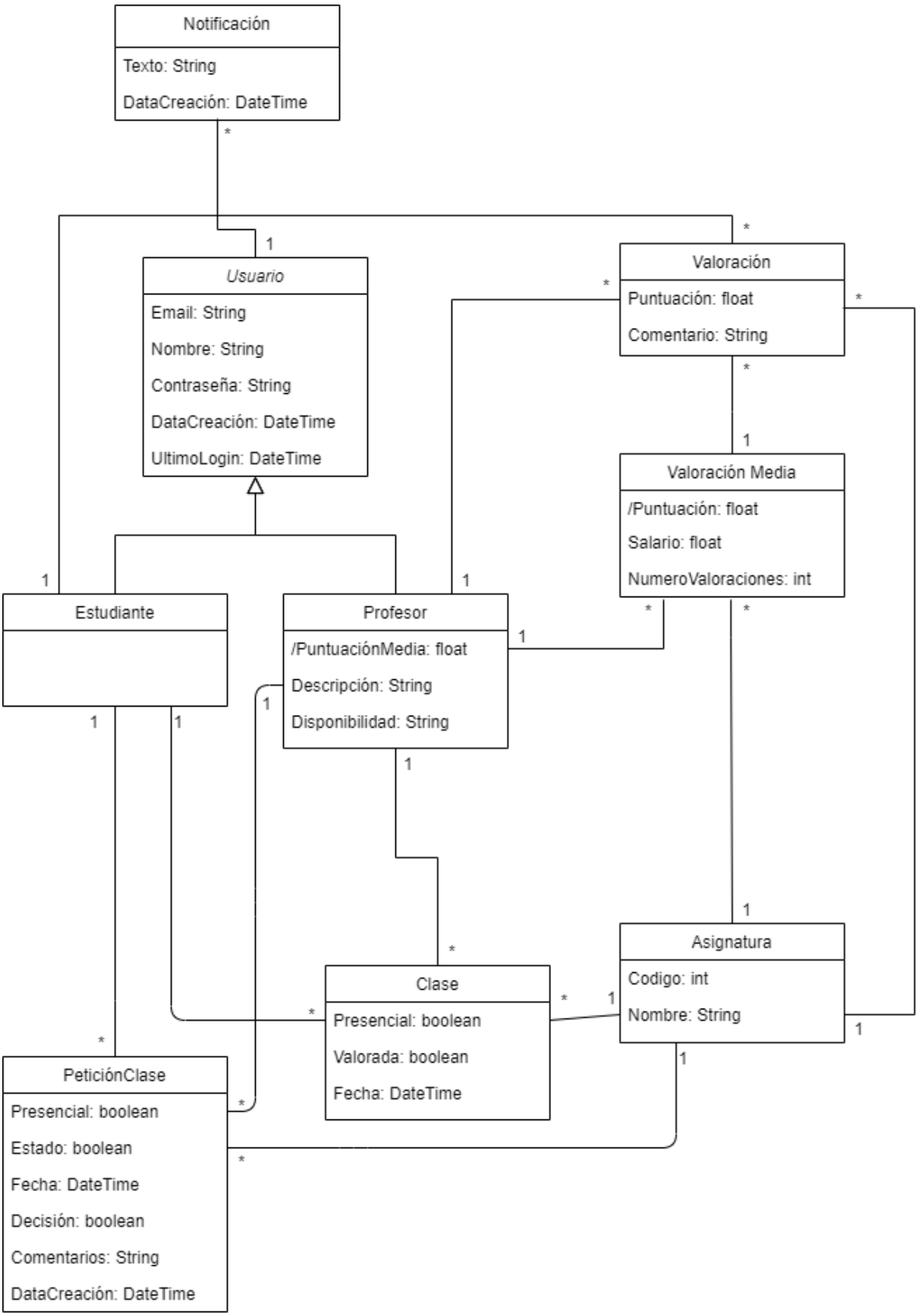


Figura 4: Diagrama de clases de diseño

El diseño en UML de la base de datos, presentado en la figura 3, se ha hecho a partir de unos modelos iniciales simples que hicimos cuando se definió el proyecto, y después de aplicar todas las modificaciones que hemos creído o han sido necesarias (nuevos campos, campos obsoletos, etc.). En la figura 4 podemos ver el esquema de la base de datos o diagrama de clases de diseño, que consiste en eliminar las asociaciones ternarias y las clases asociativas.

Finalmente para el diseño físico de la BD hay que tener en cuenta que se ha escogido MongoDB como sistema de bases de datos. Decidimos usar MongoDB, una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos sea más fácil y rápida. Estas estructuras son conocidas como colecciones, que vendrían a ser las tablas en las bases de datos relacionales, y en vez de tener filas dentro de estas tablas, tenemos documentos (las estructuras BSON).

Figura 5: Ejemplo de documento de MongoDB

En la figura 5 podemos ver un ejemplo de un documento, en específico un documento de la clase *Lesson* (o Clase en el diagrama UML). A continuación veremos un ejemplo de como quedan organizados nuestras clases con MongoDB y explicaremos que representan los atributos:

- Collection avgscorings (Valoración Media):
  - _id: identificador del documento (ObjectId)
  - teacher: identificador del documento donde se encuentra el usuario profesor (ObjectId)
  - subject: identificador del documento donde se encuentra la asignatura (ObjectId)
  - subjectName: nombre de la asignatura (String)
  - salaryHour: precio por hora de la asignatura (Integer)
  - score: puntuación media de la asignatura (Integer)
  - scoreNumber: número de valoraciones de la asignatura (Integer)

- Collection lessonpetitions (PeticiónClase):
    - _id: identificador del documento (ObjectId)
    - decision: si el profesor ha aceptado la petición o no (Boolean)
    - creationDate: fecha y hora en la que se creó la petición (DateTime)
    - sender: identificador del documento donde se encuentra el usuario que creó la petición (ObjectId)
    - senderName: nombre del usuario que creó la petición (String)
    - receiver: identificador del documento donde se encuentra el usuario al que va dirigida la petición (ObjectId)
    - date: fecha en la que se quiere realizar la clase (Date)
    - time: hora en la que se quiere realizar la clase (String)
    - subject: identificador del documento donde se encuentra la asignatura de la petición (ObjectId)
    - subjectName: nombre de la asignatura (String)
    - type: tipo de la petición de clase, presencial u online (Boolean)
    - comments: comentarios extras a la petición de clase (String)
    - state: estado de la petición, si el profesor ha respondido está a true, si sigue pendiente a false (Boolean)

- Collection lessons (Clase):
    - _id: identificador del documento (ObjectId)
    - valued: si el alumno ya ha valorado la clase o no (Boolean)
    - teacher: identificador del documento donde se encuentra el usuario profesor (ObjectId)
    - student: identificador del documento donde se encuentra el usuario estudiante (ObjectId)
    - subject: identificador del documento donde se encuentra la asignatura (ObjectId)
    - date: fecha en la que se va a realizar la clase (Date)
    - time: hora en la que se va a realizar la clase (String)
    - subjectName: nombre de la asignatura (String)
    - type: tipo de la petición de clase, presencial u online (Boolean)

- Collection notifications (Notificación):
    - _id: identificador del documento (ObjectId)
    - creationDate: fecha y hora en la que se creó la notificación (DateTime)
    - receiver: identificador del documento donde se encuentra el usuario al que va dirigida la notificación (ObjectId)
    - notificationText: texto de la notificación (String)

- Collection scorings (Valoración):
    - _id: identificador del documento (ObjectId)
    - teacher: identificador del documento donde se encuentra el usuario profesor (ObjectId)
    - student: identificador del documento donde se encuentra el usuario estudiante (ObjectId)
    - subject: identificador del documento donde se encuentra la asignatura (ObjectId)
    - subjectName: nombre de la asignatura (String)
    - score: puntuación de la asignatura (Integer)
    - comment: comentario del alumno acompañando a la valoración (String)

- Collection subjects (Asignatura):
    - _id: identificador del documento (ObjectId)
    - code: código de la asignatura (Integer)
    - name: nombre de la asignatura (String)

- Collection users (Usuario):
    - _id: identificador del documento (ObjectId)
    - signupDate: fecha y hora en la que se creó el usuario (DateTime)
    - teacher: indica si el usuario es profesor o no (Boolean)
    - averageScoring: valoración media del profesor (Integer)
    - email: correo electrónico del usuario (String)
    - displayName: nombre del usuario con el que aparecerá en la aplicación (String)
    - password: contraseña hasheada del usuario (String)
    - profileDescription: descripción del perfil del usuario (String)
    - availability: disponibilidad del usuario para dar clases (String)

10.4. Diseño de la interfaz gráfica

10.4.1. Consideraciones generales

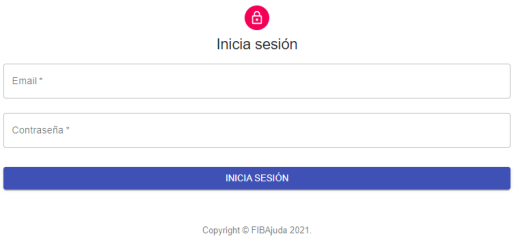
Al tratarse de una propuesta de software que puede ser utilizado por la FIB, durante el diseño de la interfaz de la aplicación hemos tenido en cuenta el diseño del Racó. Es por eso que hemos decidido hacer la pantalla de inicio de sesión lateral con una imagen de la facultad a la izquierda, para mantener el mismo estilo.

Una vez dentro de la aplicación, hemos usado una gama de colores azules, parecidos a los del Racó de la FIB, para mantener esa identidad corporativa. También decir que al ser un proyecto que se puede usar dentro de otras facultades y universidades, el estilo puede variar en función de los colores de la entidad que quiera usar este servicio.

Para la navegación de la aplicación hemos decidido usar un menú con el que los usuarios se sientan familiarizados, ya que no queremos confundirlos. El menú que hemos utilizado consiste en una barra superior o *header*, donde se encuentran todos los puntos de interacción para acceder a todas las páginas de la aplicación.

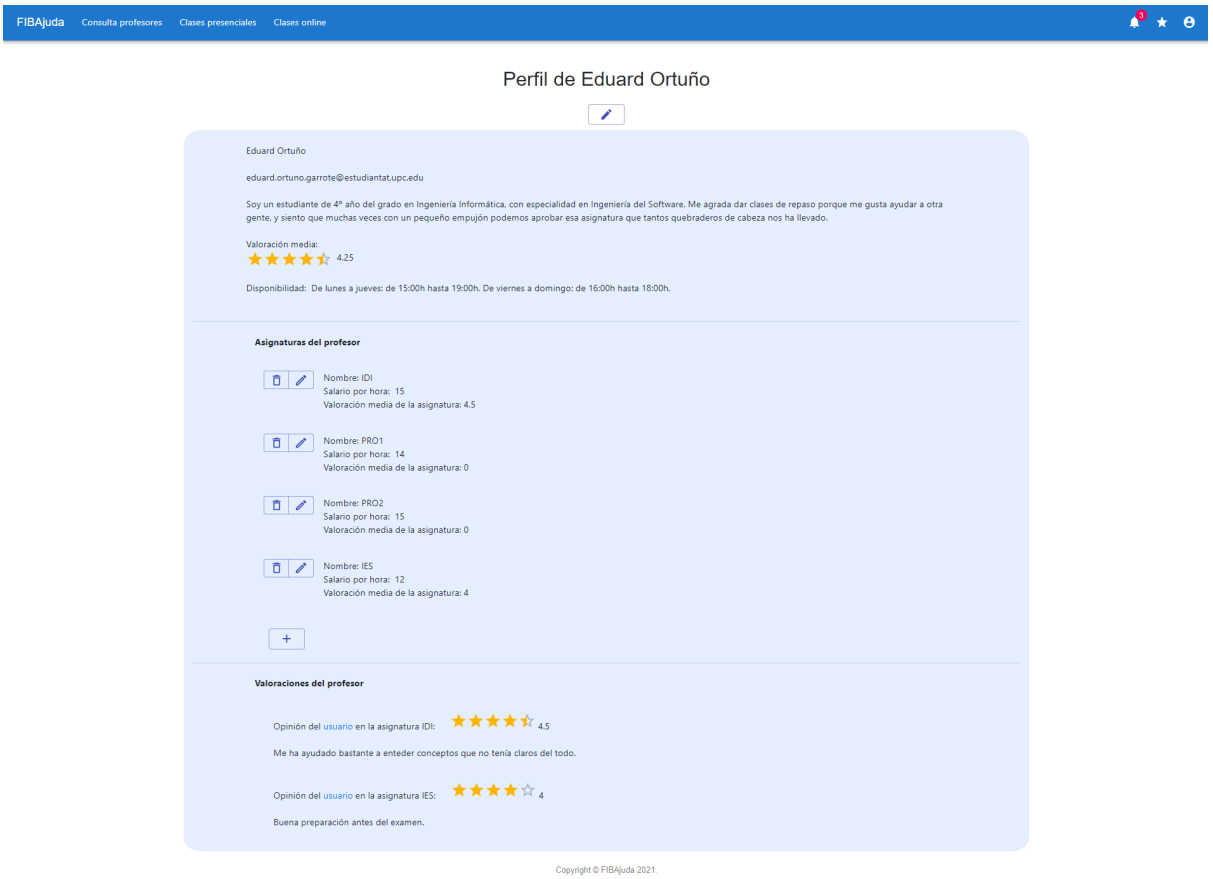
10.4.2. Capturas de la aplicación

(a) Página de inicio

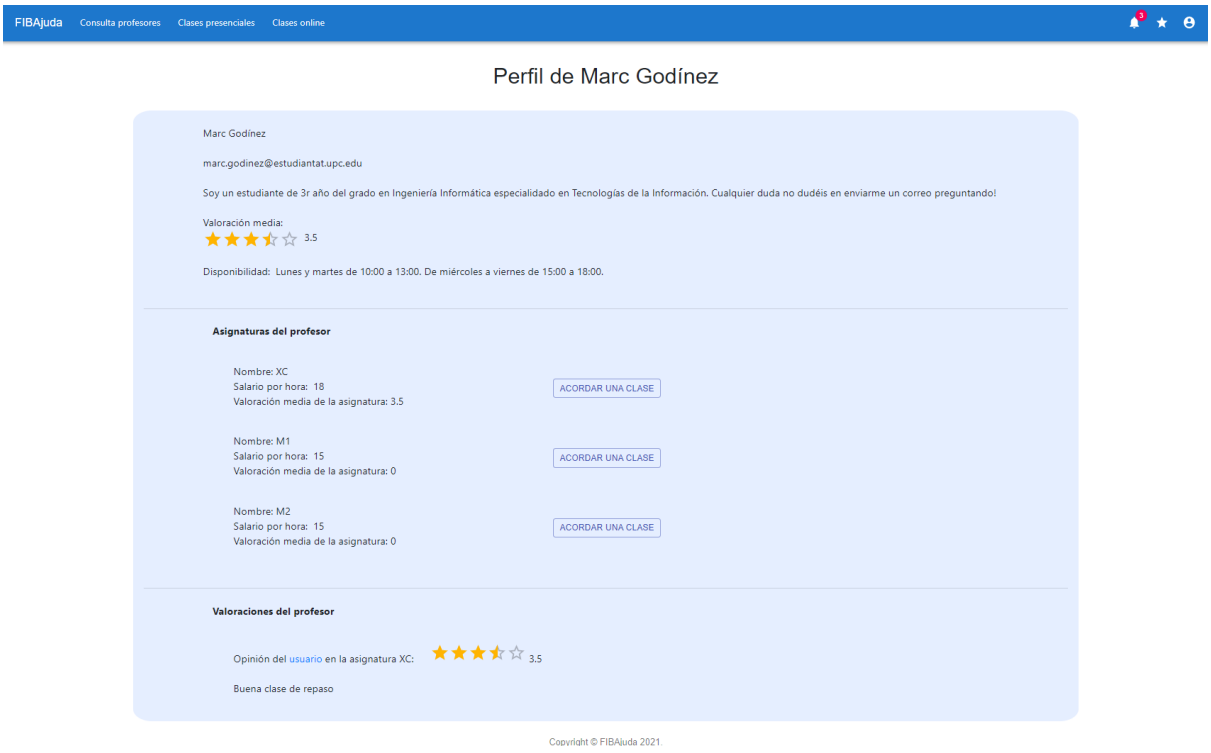


(b) Inicio de sesión

(c) Buscador de profesores

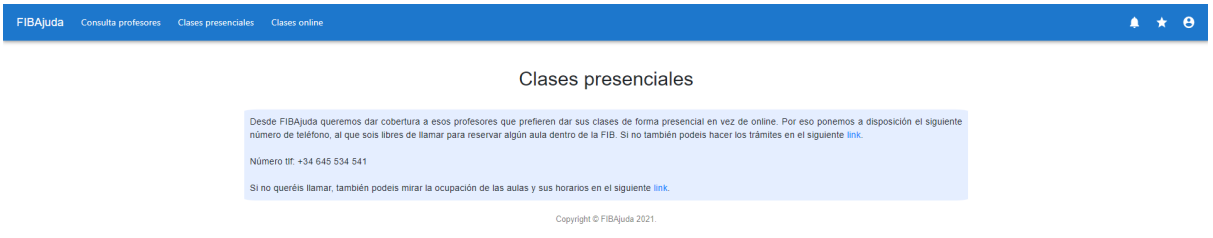


(d) Mi perfil



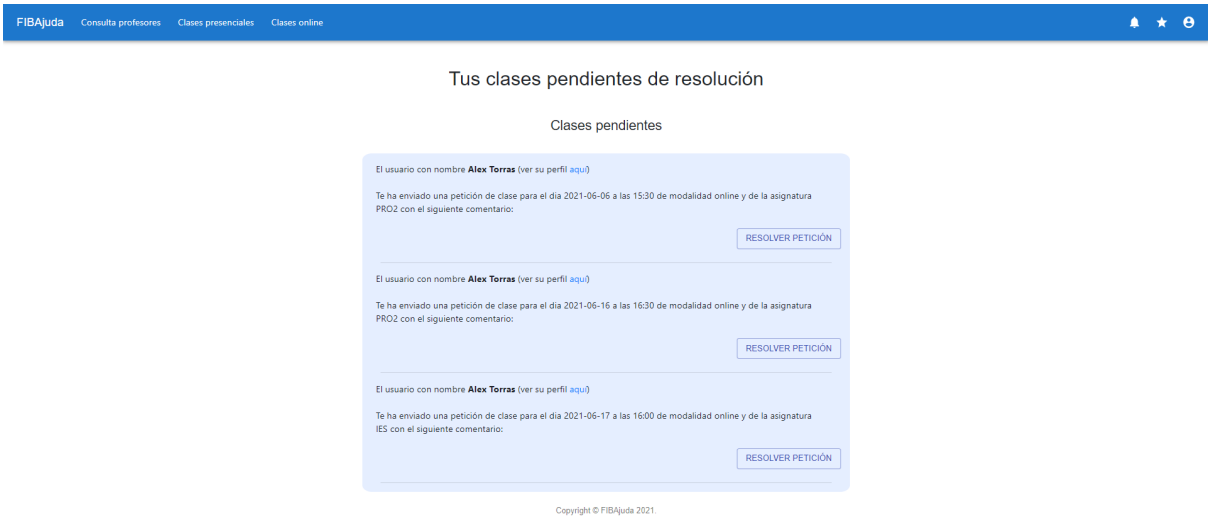
(e) Perfil usuario

(f) Crear petición de clase



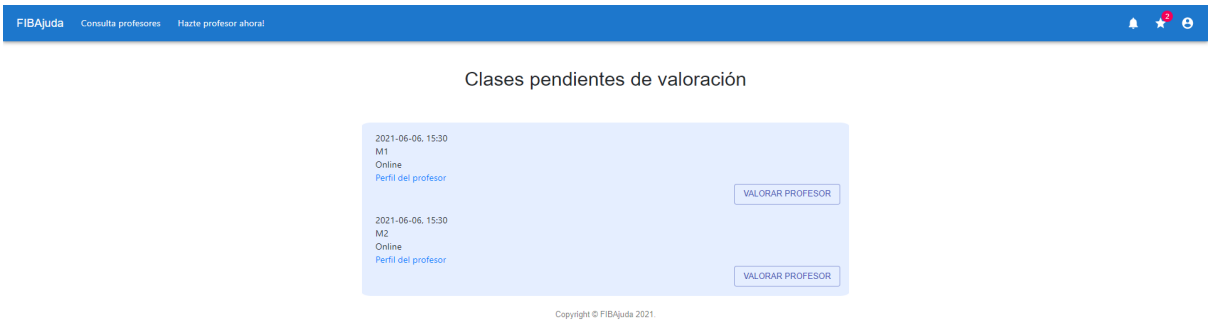
(g) Clases presenciales

(h) Clases online

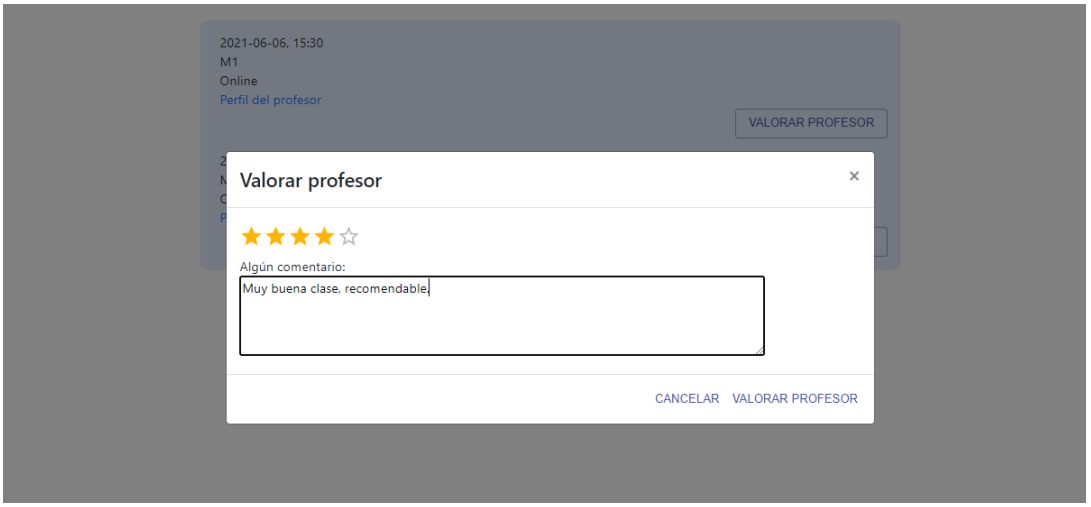


(i) Clases pendientes de resolución

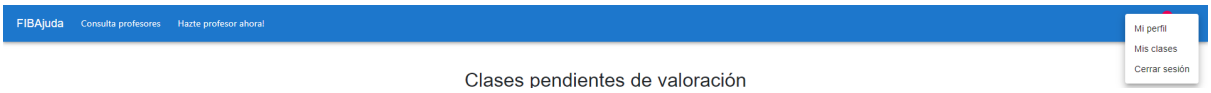
(j) Resolución de una petición de clase



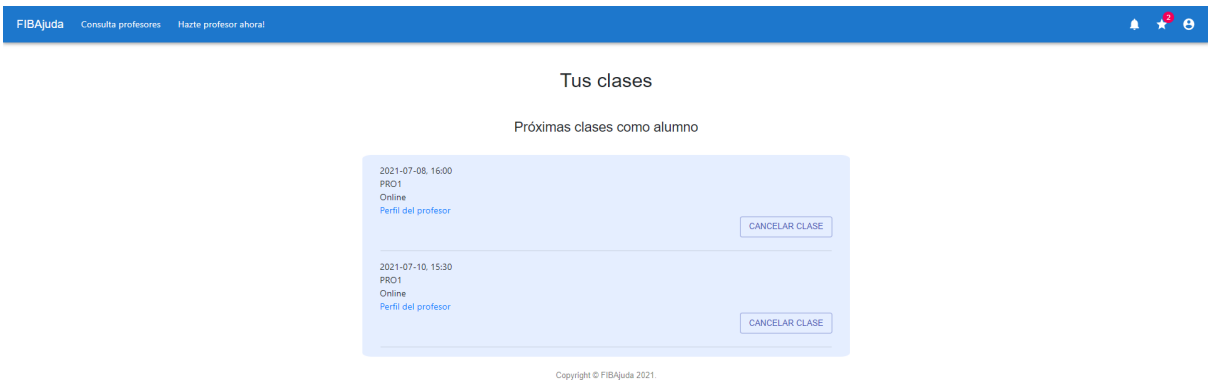
(k) Clases pendientes de valoración



(l) Valoración a un profesor



(m) Menú desplegable



(n) Tus clases

11. Desarrollo

Durante el desarrollo del proyecto hemos utilizado diferentes recursos para poder llevar a cabo la implementación de todas las historias de usuario. En este apartado se explica el funcionamiento de estos recursos o herramientas y cómo las hemos usado para poder realizar y ejecutar nuestra aplicación.

11.1. Recursos utilizados

11.1.1. Lenguaje de programación

11.1.1.1. JavaScript

Javascript[19], publicado en 1995, fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript. Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java.

Figura 7: Logo de JavaScript

Escogimos utilizar este lenguaje debido a que ya lo había usado anteriormente y además es un lenguaje muy utilizado que nos ha permitido desarrollar tanto nuestro backend como nuestro frontend.

11.1.1.2. JSX

JSX[20] (JavaScript Syntax Extension, ocasionalmente llamado JavaScript XML) es una extensión de la sintaxis del lenguaje JavaScript. Parecido en apariencia a HTML, JSX permite estructurar el renderizado de componentes de React usando una sintaxis familiar para muchos desarrolladores, ya que nos permite mezclar JS y HTML.

El motivo por el que usamos este lenguaje es que la parte del frontend de la aplicación la hemos desarrollado usando ReactJs, que usa este lenguaje.

Figura 8: Logo de JSX

11.1.2. Tecnologías usadas

11.1.2.1. NodeJs

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js, creado por Ryan Dahl en 2009, está diseñado para crear aplicaciones network escalables. Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, con E/S de datos en una arquitectura orientada a eventos. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

Decidimos usar NodeJs, ya que era una tecnología que ya habíamos utilizado anteriormente en otro proyecto, además de ser un entorno muy popular entre los desarrolladores con un lenguaje como JavaScript.

Figura 9: Logo de NodeJs

11.1.2.2. ReactJs

ReactJs, creado en 2013 por Jordan Walke y desarrollado por Facebook y una comunidad de desarrolladores y empresas, es una biblioteca Javascript de código abierto diseñada para crear interfaces o componentes de interfaces de usuario. React tiene el objetivo de facilitar el desarrollo de aplicaciones en una sola página o aplicaciones móviles. Sin embargo, React solo se encarga del control del estado y el renderizado del estado al DOM, por eso, cuando se crea una aplicación con React normalmente se necesita usar otras librerías para el routing o funcionalidades para el cliente.

Después de buscar información sobre varios frameworks para desarrollar nuestro frontend, decidimos usar ReactJs, ya que funciona muy bien con NodeJs, además de ser fácil de aprender gracias al desarrollo por componentes.

Figura 10: Logo de React

11.1.2.3. MongoDB Atlas

MongoDB, lanzada en 2009 por MongoDB Inc., es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. Es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos sea más fácil y rápida. Siendo MongoDB Atlas el mismo servicio pero en la nube.

Decidimos usar MongoDB, ya que queríamos utilizar una base de datos no relacional para aprender el funcionamiento de esta. Además, MongoDB ofrece unos

tiempos de lectura más bajos comparado con el resto de bases de datos relacionales.

Figura 11: Logo de MongoDB Atlas

11.1.2.4. Postman

Postman[21], creada en 2014 por y para desarrolladores, es una extensión del navegador Google Chrome y una aplicación de escritorio que permite el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente. Tiene una interfaz fácil de usar que permite construir consultas y leer respuestas. Nosotros lo utilizamos para asegurarnos que la API funciona correctamente.

Figura 12: Logo de Postman

11.1.3. Herramientas

11.1.3.1. Visual Studio Code

Visual Studio Code[22], lanzado en 2015 por Microsoft, es un editor de código fuente desarrollado para Windows, Linux y macOS. Algunas de las herramientas que incluye son: soporte para la depuración, resaltado de la sintaxis, finalización inteligente de código, fragmentos, refactorización de código y control integrado de Git. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado, las preferencias, e instalar extensiones que añaden funcionalidades extra.

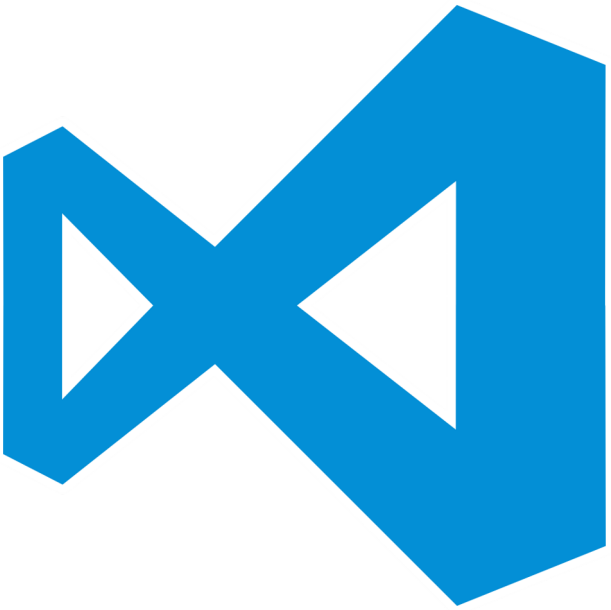


Figura 13: Logo de Visual Studio Code

11.2. Implementación de las historias de usuario

En este apartado explicaremos las historias de usuario más complejas del proyecto y cómo han sido implementadas.

11.2.1. Instalación y configuración de herramientas

En primer lugar, instalamos la herramienta que usaremos durante todo el desarrollo del proyecto: Visual Studio Code. Este editor de código lo utilizamos tanto para el backend como para el frontend. Una vez instalado esto, nos descargamos npm[23] junto a NodeJs (npm se distribuye con NodeJs, lo que significa que cuando nos descargamos NodeJs se nos instala automáticamente npm).

Una vez instalado y configurado el editor junto a NodeJs, configuramos la base de datos. Como hemos dicho en el apartado anterior, hemos utilizado MongoDB Atlas. Para utilizar este sistema hemos tenido que crearnos una cuenta y un superusuario para poder conectarnos más tarde desde el backend. Una vez creado el clúster en la BD, hemos de copiar el link de este con las credenciales del superusuario en el archivo config.js.

Para instalar las herramientas necesarias para el frontend, utilizamos el comando *npm create-react-app* para configurar nuestro ambiente de desarrollo de forma que podamos usar las últimas características de Javascript, brindando una buena experiencia de desarrollo, y optimizando nuestra aplicación para producción.

11.2.2. Inicio de sesión

El inicio de sesión es una de las partes más importantes de la aplicación, ya que sin ella los usuarios no podrían acceder y gestionar los recursos que hay en esta. Sin embargo, muchos desarrolladores no le ponen tanto interés al apartado de seguridad. Por esa misma razón, nosotros hemos decidido que vamos a esforzarnos para que nuestra aplicación cumpla unos requisitos mínimos de seguridad, con el propósito de proteger la información de los usuarios.

Antes que nada, tenemos que explicar que es un JWT[24]. Un JWT (JSON Web Token) es un estándar abierto basado en JSON propuesto por la IETF para la creación de tokens de acceso que permiten la propagación de identidad y privilegios entre dos o más sistemas. Los JWT están diseñados para ser compactos y poder ser enviados por URL de una manera segura. Estos tokens tienen tres partes bien diferenciadas: la cabecera (header), el contenido (payload) y la firma (signature).

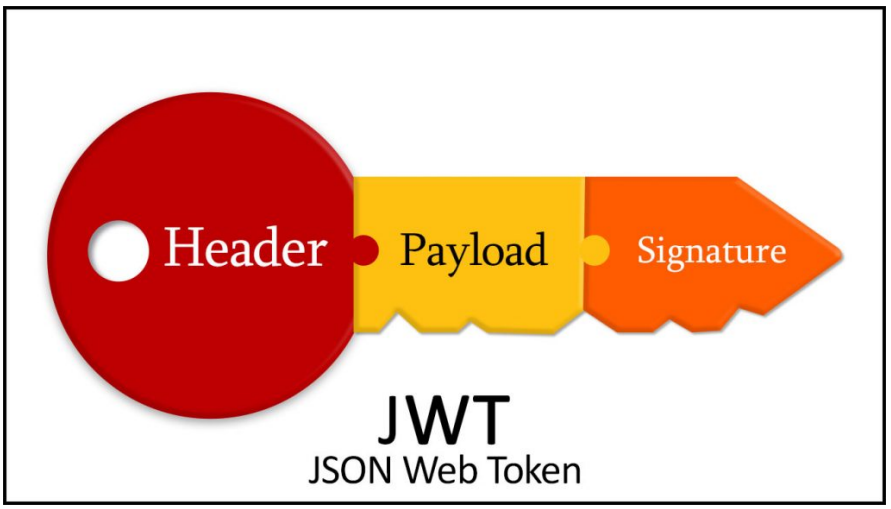


Figura 14: Estructura de un JWT

Pongamos un ejemplo: un servidor podría generar un token indicando que el usuario tiene privilegios de administrador, y proporcionarle este token a un cliente. El cliente entonces podría utilizar el token para probar que está actuando como un administrador o usuario en el cliente o en otro sistema. El token está firmado por la clave del servidor, así que tanto el cliente como el servidor son capaces de verificar que el token es legítimo. En nuestro proyecto, para utilizar esta tecnología hemos utilizado el paquete *jwt-simple*[25].

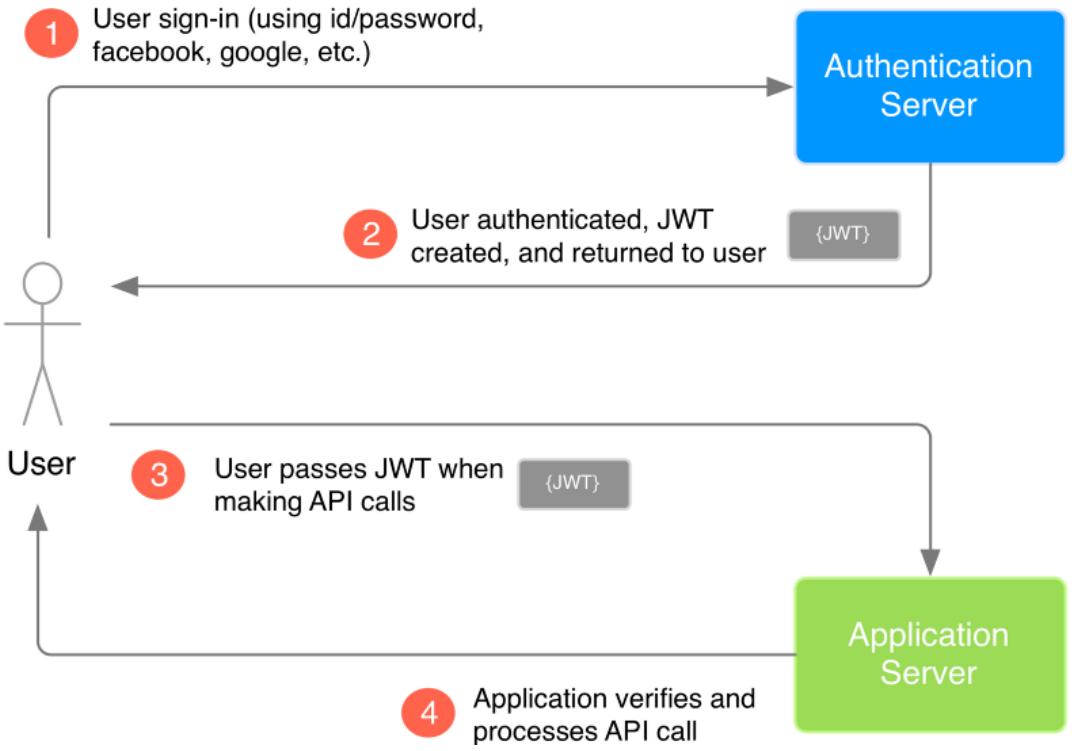


Figura 15: Uso del JWT

Además, a la hora de cifrar la contraseña tanto cuando se crea un nuevo usuario como para validar que las credenciales introducidas son correctas, utilizamos *bcrypt*[26], una función derivadora de claves (también conocida como *hashing*[27]) que incluye una sal (bits aleatorios que se usan como una de las entradas en una función derivadora de claves). En criptografía, una función criptográfica hash (usualmente conocida como *hash*) es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija, es decir, independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud.

Cada vez que un usuario inicia sesión en la aplicación, recibe un JWT que es almacenado por el frontend de la aplicación junto a otro conjunto de datos del usuario dentro de la entidad *CurrentUser*. Entonces, en cada petición que se realiza al servidor que requiere de autenticación, el token se incluye en la llamada a la API (concretamente en la cabecera Authorization de la petición) y de esta forma identificamos al usuario. La durabilidad del token la especificamos nosotros, y en este caso le hemos puesto un valor de 7 días. Cuando este token caduque, el usuario dejará de tener la sesión abierta en la aplicación, haciendo necesario que el usuario tenga que volver a iniciar sesión manualmente. También queremos mencionar que, dado que nos hemos asegurado que todas las comunicaciones de

la aplicación se realizan únicamente por HTTPS, los tokens no podrán ser usurpados por terceros.

11.2.3. Listar profesores

Para poder mostrar la lista de profesores de la aplicación hemos tenido en cuenta varios aspectos técnicos. En primer lugar, decidimos que para solicitar la lista haríamos una petición GET a un endpoint de la API, concretamente /getTeachers.

Para hacer que la llamada fuera eficiente y no trajera datos innecesarios, los filtros que se pueden aplicar (precio, valoraciones, asignaturas), en vez de ser aplicados a la lista de todos los profesores desde el front, decidimos aplicarlos en el back, de esta forma solo se devuelven los datos de los profesores que cumplen los criterios especificados. La función que devuelve el listado de profesores, funciona de manera modular, es decir, tanto si hay filtros especificados en el body de la llamada a la API como si no.

Para hacer las llamadas a la API desde el front, usamos la librería *axios*[28], una librería JavaScript que puede ejecutarse en el navegador y que nos permite hacer sencillas operaciones como cliente HTTP, por lo que podremos configurar y realizar solicitudes a un servidor y recibiremos respuestas fáciles de procesar. El problema que nos encontramos con esta librería fue que tanto las llamadas GET como las DELETE, no pueden tener un body, es decir, no pueden ir acompañadas de parámetros (como por ejemplo los filtros). A partir de esto, teníamos diversas soluciones, como por ejemplo pasar los parámetros por url, pero decidimos cambiar la definición de la llamada en el back a un PUT y de esta forma poder tener un body.

11.2.4. Peticiones de clase

En cuanto a las peticiones de clase, hay diversas historias de usuario que están dentro de esta categoría.

11.2.4.1. Número de peticiones y notificaciones

Tenemos una ruta de la API que devuelve el número de notificaciones y peticiones pendientes de resolución, para así poder mostrar el número en la barra del menú superior de la aplicación. Esta ruta es una petición GET, que cuenta el número de documentos que hay en la base de datos que tienen como receptor al usuario con sesión abierta en la aplicación.

11.2.4.2. Crear nueva petición

Esta funcionalidad consiste en un formulario que se rellena con los campos de fecha, hora, asignatura, modalidad de la clase (online o presencial) y un comentario donde el alumno puede incluir alguna petición extra o alguna duda que pueda tener. Se trata de una petición POST, ya que se crea un nuevo objeto en la base de datos.

Hay algunos controles aplicados a esta funcionalidad, como por ejemplo no poder hacer una petición para una fecha pasada o de una asignatura que el profesor no haya introducido en la aplicación como disponible.

11.2.4.3. Resolver petición

Una vez el usuario alumno ha creado la petición de clase, esta le aparece al profesor en su apartado de peticiones pendientes. Entonces el profesor puede aceptar o rechazar esa petición, además de añadir un comentario respecto a su decisión.

Se trata de una petición PUT, que lo que hace es dejar constancia de la decisión tomada por el profesor, además de enviar una notificación al alumno usando la misma instancia de nuestra clase LessonPetition. Lo que hace es cambiar el receptor de esa petición al usuario alumno que envió la petición en primera instancia, además de cambiar el estado de esta para que cuando el usuario alumno entre en la aplicación le aparezca cuando vea sus peticiones.

En caso de que el profesor decida aceptar la petición de clase, se crea una instancia de clase con los datos de la petición. Esta clase queda registrada tanto para el profesor como para el alumno y aparece en su calendario.

11.2.4.4. Ver peticiones

Esta funcionalidad es bastante sencilla, ya que consiste en una llamada GET a la API que trae todas las peticiones que tenga pendientes el usuario con sesión abierta en la aplicación. Una vez en el front con el listado de peticiones, se filtran dependiendo de su estado, es decir, si eres el primer receptor, es decir, el profesor, y en este caso se tiene la opción de resolver esta petición, o el segundo, el alumno que creó la petición en primera instancia, y en este caso se trata de una notificación con la resolución del profesor y su comentario.

11.2.5. Valorar profesor

Para las valoraciones, también tenemos una ruta de la API que devuelve el número de clases pendientes de valorar, para así poder mostrar el número en la barra del menú superior de la aplicación. Esta ruta es una petición GET, que cuenta el número de documentos que hay en la base de datos que tienen como alumno de la clase al usuario con sesión abierta en la aplicación.

Dependiendo de si el usuario ha valorado anteriormente al profesor en esa asignatura o no, se mostrará la valoración y el comentario anterior (si tiene) para así que el usuario pueda ir modificándolo según su criterio. En cuanto a esta funcionalidad, se trata de una petición PUT, ya que dependiendo de si el alumno ha valorado anteriormente al profesor en esa asignatura, se creará una nueva valoración o se actualizará la que ya existe.

11.3. Deployment

El deployment del software no se ha realizado durante el desarrollo de este proyecto, ya que el software que ha sido realizado en este trabajo no va a ser puesto en producción para intentar monetizarlo. Este software ha sido realizado para poder llevar a cabo el TFG.

No obstante, en caso de que se deseara hacer el deployment del software existen varias herramientas conocidas que podríamos utilizar, como Heroku[29] o AWS[30]

(Amazon Web Services). A continuación hablaremos de las ventajas y desventajas de estos dos software.

Algunas de las principales ventajas de Heroku son:

- Heroku es gratuito para aplicaciones de poco consumo.
- Permite el uso de diferentes lenguajes de programación.
- Es una plataforma fácil de usar.
- Se puede tener acceso desde cualquier lugar y dispositivo compatible con la computación en la nube.

En cuanto a sus principales desventajas:

- Poca personalización y mínima optimización cuando se requiere más infraestructura (versión gratuita).
- Es necesario contratar un servicio de base de datos externo y un hosting.

En cuanto a AWS, algunas de sus ventajas son:

- Bajos costos de implementación.
- Pagos por los recursos empleados.
- Alta escalabilidad.

Mientras que sus inconvenientes son:

- No es recomendable para personal sin experiencia.
- Exige un alto nivel de seguridad.
- No es altamente personalizable.

12. Pruebas

En este apartado hablaremos de las pruebas o tests que hemos aplicado tanto al backend como al frontend para comprobar que el sistema funcionara correctamente y asegurarnos de la integridad de los datos.

12.1. Pruebas del backend

La mayoría de los tests del backend, tanto de los modelos como de los endpoints, han sido testeados con Postman. El resto han sido realizados con revisión manual.

12.1.1. Modelos

Es muy importante asegurarse de que no hay ningún comportamiento no deseado en la lógica de negocio del sistema, por ello decidimos hacer un testing exhaustivo de los modelos comprobando su comportamiento y que sus asociaciones funcionaran correctamente.

En esencia, hemos probado los siguientes aspectos o características de todos los modelos:

- Presencia de atributos: para cada modelo hemos validado qué atributos son requeridos, es decir, que deben estar de forma obligatoria, y cuáles son opcionales.

- Asociaciones de los modelos: para cada modelo hemos validado que las asociaciones de los modelos con los que interactúa se formasen correctamente, por ejemplo, las clases con los profesores y alumnos, las notificaciones con el usuario receptor, etc.

- Reglas de negocio: para cada modelo hemos validado algunas reglas que surgen de la naturaleza del sistema, como por ejemplo que un usuario haga una clase consigo mismo, o que se valore a sí mismo.

- Reglas de integridad: para cada modelo hemos validado que los atributos que forman claves primarias hagan su función correctamente, ya que MongoDB usa por convención identificadores (ids) de los modelos.

12.1.2. Endpoints

Una vez realizadas las pruebas de los modelos, ya nos hemos asegurado de que no haya ninguna inconsistencia en nuestro sistema. Por lo tanto ahora podemos añadir una capa de validación para evitar errores indeseados durante las llamadas a los endpoints de la API.

En esencia, hemos probado los siguientes aspectos o características de todos los controladores:

- Códigos de estado: para cada endpoint hemos comprobado que se devuelve el código de estado apropiado según la naturaleza de la consulta, por ejemplo, devolvemos 404 si el objeto al que se quiere acceder no existe, 200 si se ha creado un nuevo objeto, 500 si ha ocurrido algún error, etc.

- Formato de entrada y salida de atributos: para cada endpoint hemos comprobado que los atributos que venían acompañando a la llamada eran los requeridos por ese endpoint. Además, hemos comprobado el formato de la respuesta de la llamada según el uso que quisiéramos darle a esta.

12.2. Pruebas del frontend

Las pruebas que hemos realizado en el frontend han sido todas manuales. Por cada funcionalidad que se desarrollaba, se iba testeando que funcionara correctamente tanto en la navegabilidad como en la lógica de negocio.

Adicionalmente a los controles que hemos llevado a cabo sobre los atributos desde el backend, en el frontend también asegurábamos la presencia, el formato y los límites de todos los atributos, como por ejemplo en la valoración de un profesor, donde sólamente permitimos que esta vaya del intervalo de 0 hasta 5.

También hemos realizado un conjunto de pruebas con un usuario real mediante un guion, para ver si podíamos encontrar algún problema con la usabilidad de nuestra aplicación. El guion era el siguiente:

1. Iniciar sesión.
2. Convertirse en profesor.
3. Modificar su perfil con una descripción y un horario.
4. Añadir las asignaturas que imparte.
5. Mirar el listado de profesores.
6. Acceder al perfil de uno de los profesores de la aplicación.

7. Realizar una petición de clase a ese profesor.
8. Valorar a un profesor.

Después de que el usuario realizara este guion, nos dio algunos comentarios que nos sirvieron de feedback para mejorar la usabilidad de la aplicación, como por ejemplo los colores de algunos botones, que no concordaban con los de la aplicación, el tamaño de los títulos, que proponía aumentarlos, ya que eran un poco pequeños comparado con el resto del texto, etc.

13. Planificación final y desviaciones

El diagrama de Gantt de la planificación inicial lo podéis encontrar en la página 25, mientras que el Gantt de la planificación final, que ha cambiado en algunos aspectos respecto a la inicial, lo podéis encontrar en la página 69. Estos cambios se deben principalmente a estos factores:

- Los sprints del proyecto se han tenido que acabar antes de lo previsto, alrededor de unas 3 semanas antes, debido a la preparación de la memoria del TFG, ya que las fechas que pusimos en la planificación inicial no eran correctas.

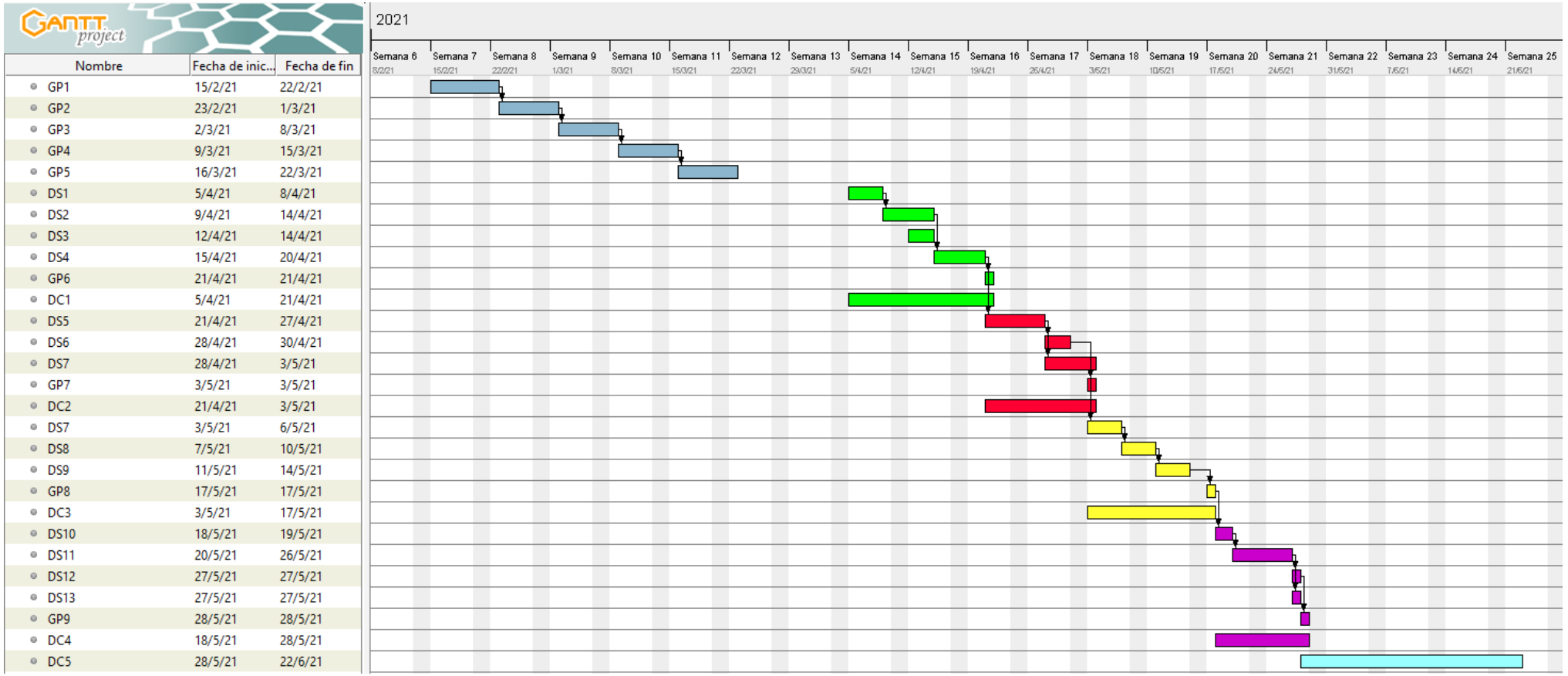
- Algunas historias de usuario como el chat en vivo o el aula virtual han desaparecido completamente (marcadas en rojo), mientras que han aparecido otras nuevas como las relacionadas con las peticiones de clase (marcadas en verde).

- El proyecto se ha empezado más tarde de lo previsto, entre el final de la gestión del proyecto (22 de marzo) y el inicio del desarrollo (5 de abril) transcurrieron 15 días que se tomaron como descanso. Los días del 28 de marzo al 5 de abril pertenecientes a la semana santa se tomaron como vacaciones.

Id	Tarea	Tiempo planificado	Tiempo real	Diferencia
GP1	Contextualización y alcance	20h	23.4h	+3.4h
GP2	Planificación temporal	20h	22.1h	+2.1h
GP3	Gestión económica y sostenibilidad	20h	19.3h	-0.7h
GP4	Documentación fase inicial	20h	18.7h	-1.3h
GP5	Definir historias de usuario	20h	24.5h	+4.5h
DS1	Inicio (instalación y configuración)	20h	22.3h	+2.3h
DS2	Autentificación	35h	30.6h	-4.4h
DS3	Modelos	5h	14.5h	+9.5h
DS4	Editar perfil	20h	33.8h	+13.8h
GP6	Retrospectiva Sprint I	5h	3.4h	-1.6h

DC1	Documentación Sprint I	15h	5.3h	-9.7h
DS5	Buscador de profesores	45h	46.2h	+1.2h
DS6	Acordar clase con profesor	25h	18.7h	-6.3h
DS7	Ver mis clases	-	23.3h	+23.3h
GP7	Retrospectiva Sprint II	5h	2.2h	-2.8h
DC2	Documentación Sprint II	15h	6.8h	-8.2h
DS8	Consultar precio asignatura	15h	18.3h	+3.3h
DS8	Chat en vivo	40h	-	-40h
DS9	Crear petición clase	-	31.4h	+31.4h
GP8	Retrospectiva Sprint III	5h	4.3h	-0.7h
DC3	Documentación Sprint III	15h	3.5h	-11.5h
DS10	Resolver petición clase	-	11.2h	+11.2h
DS11	Valoraciones profesores	45h	50.1h	+5.1h
DS10	Aula virtual	45h	-	-45h
DS12	Soporte clases online	45h	5.5h	-39.5h
DS13	Soporte clases presenciales	-	5.5h	+5.5h
GP9	Retrospectiva Sprint IV	5h	1.3h	-3.7h
DC4	Documentación Sprint IV	15h	4.8h	-10.2h
DC5	Finalizar memoria	40h	64.7h	+24.7h
TOTAL		**560h**	**515.7h**	**-44.3h**

Tabla 12: Diferencia en la planificación del desarrollo. Elaboración propia.



Nombre	Fecha de inic...	Fecha de fin
GP1	15/2/21	22/2/21
GP2	23/2/21	1/3/21
GP3	2/3/21	8/3/21
GP4	9/3/21	15/3/21
GP5	16/3/21	22/3/21
DS1	5/4/21	8/4/21
DS2	9/4/21	14/4/21
DS3	12/4/21	14/4/21
DS4	15/4/21	20/4/21
GP6	21/4/21	21/4/21
DC1	5/4/21	21/4/21
DS5	21/4/21	27/4/21
DS6	28/4/21	30/4/21
DS7	28/4/21	3/5/21
GP7	3/5/21	3/5/21
DC2	21/4/21	3/5/21
DS7	3/5/21	6/5/21
DS8	7/5/21	10/5/21
DS9	11/5/21	14/5/21
GP8	17/5/21	17/5/21
DC3	3/5/21	17/5/21
DS10	18/5/21	19/5/21
DS11	20/5/21	26/5/21
DS12	27/5/21	27/5/21
DS13	27/5/21	27/5/21
GP9	28/5/21	28/5/21
DC4	18/5/21	28/5/21
DC5	28/5/21	22/6/21

Figura 16: Diagrama de Gantt final

14. Conclusiones

14.1. Satisfacción de los objetivos iniciales

A pesar de las dificultades que hemos tenido en la planificación del proyecto, debido a las nuevas historias de usuario que han aparecido durante el desarrollo y al desconocimiento de algunas fechas del proyecto, hemos conseguido satisfacer todos los objetivos iniciales.

Concretamente tal y como mencionamos en los objetivos del proyecto, hemos llevado a cabo el diseño e implementación de una aplicación web (además de su respectiva API Rest) que permite a sus usuarios acceder a su perfil, editarlo, buscar y valorar a profesores y comunicarse con otros usuarios.

Sin embargo, la funcionalidad de que los usuarios se puedan comunicar entre ellos ha cambiado un poco, ya que en un principio se pensó que se llevaría a cabo a través de un chat en vivo en la aplicación, pero finalmente decidimos crear un sistema de peticiones de clase.

Este sistema de peticiones de clase, tiene como objetivo la toma de contacto inicial entre el profesor y el estudiante, ya que como suele ocurrir en la mayoría de aplicaciones, el contacto se acaba migrando a otros servicios de mensajería más comunes, como por ejemplo WhatsApp, correo, y similares.

Con el fin de determinar las historias de usuario más concretamente y viendo el proyecto en retrospectiva, nos hubiese ayudado el hecho de haber considerado todas las posibles pantallas de la aplicación para el ajuste de la planificación de nuestro proyecto.

14.2. Satisfacción de las competencias técnicas

14.2.1. CES1.2

Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles.
Nivel de satisfacción: [Un poco]

Realmente la idea de esta aplicación web es que vaya integrada dentro del Racó (en el caso de la FIB) o dentro de la página de cualquier facultad, convirtiéndose en un servicio interno. El frontend de nuestra aplicación podría fácilmente conectarse a esta API para realizar las llamadas que fueran necesarias. Esto no ha ocurrido, ya que los datos de los estudiantes, que son los que más nos podrían interesar, no están disponibles a través de la API (o al menos no aparece la ruta en su documentación).

El nivel de satisfacción es solo 'Un poco' porque la idea inicial de integrar alguna herramienta de comunicación como Google Meet o Microsoft Teams fue descartada.

14.2.2. CES1.3

Identificar, evaluar y gestionar los potenciales riesgos asociados a la construcción de software que se puedan presentar.
Nivel de satisfacción: [Bastante]

Esta competencia ha estado presente especialmente en el inicio del proyecto, ya que fue el momento donde consideramos todos los riesgos que pudieran aparecer durante la construcción de nuestro sistema. Es por eso que en el servidor para el backend hemos dedicado mucho esfuerzo para hacerlo lo más seguro contra posibles ataques externos, como por ejemplo mediante la utilización de tokens, o validando los inputs del usuario.

Hemos determinado el grado de satisfacción a 'Bastante', ya que hemos gestionado bien todas las posibles amenazas.

14.2.3. CES1.5

Especificar, diseñar, implementar y evaluar bases de datos.
Nivel de satisfacción: [Bastante]

Definimos todos los modelos de la aplicación de una forma que fuera eficiente cuando se tuviera que acceder a objetos relacionados a través de atributos extra. Mediante el servicio que ofrece en el Cloud MongoDB Atlas, hemos gestionado y evaluado el funcionamiento de nuestra base de datos con métricas como el número de operaciones por segundo y su tipo.

Hemos establecido el grado de satisfacción en 'Bastante', ya que hemos trabajado con la base de datos durante todo el desarrollo del proyecto.

14.2.4. CES1.7

Controlar la calidad y diseñar pruebas en la producción de software.
Nivel de satisfacción: [Bastante]

Hemos asegurado que nuestra aplicación funcionara correctamente, tanto desde el punto de vista del uso que los usuarios pudieran darle como de la lógica de negocio y las funcionalidades presentes en el backend. Es decir tenemos validaciones para asegurarnos de que el sistema no está sujeto a errores.

Hemos determinado el grado de satisfacción a 'Bastante', ya que nos hemos asegurado de que no hubiera ningún error o *bug* en el proyecto.

14.2.5. CES2.1

Definir y gestionar los requisitos de un sistema software.
Nivel de satisfacción: [Bastante]

El sistema software de nuestro proyecto define todos sus requisitos, los enumera y los documenta en su respectivo subgrupo. Determinamos que los requisitos funcionales de la aplicación corresponden con las historias de usuario y los no funcionales con aspectos técnicos relacionados con la satisfacción del usuario.

Hemos calificado el grado de satisfacción a ‘Bastante’, ya que hemos trabajado los requisitos con detalle en su respectivo apartado del proyecto.

14.3. Trabajo futuro y posibles mejoras

Si la recepción del público y de la facultad a la aplicación es positiva, hemos planteado las siguientes mejoras o funcionalidades que podrían facilitar al crecimiento de esta:

- Integración con la API del Racó: mediante la gestión ya existente de los usuarios, conseguiremos una mejor lógica y una mayor presencia de los alumnos.

- Mejoras del sistema: podemos añadir nuevas funcionalidades a la aplicación que le otorguen un valor añadido como por ejemplo un chat en vivo, botones para compartir recursos con otros usuarios, un aula virtual, añadir redes sociales de los usuarios, funcionalidades sugeridas por los alumnos recibidas mediante feedback, etc.

- Traducción a otros idiomas: con el objetivo de llegar a un público más extenso como por ejemplo los estudiantes de Erasmus, o incluso se podría considerar la posibilidad de ofrecer este servicio en universidades de otros países.

- Inclusión de publicidad: para poder asumir los gastos relacionados con la escalabilidad de la aplicación, podríamos implementar un sistema en que los profesores puedan publicitarse dentro de la aplicación, generando unos ingresos que ayudarían a pagar el servicio.

- Patrocinio y difusión: para que se haga popular, podríamos pedir a la universidad o facultad que publicite nuestra aplicación a través de los canales oficiales de la universidad como foros o correos.

15. Bibliografía

[1] Tus clases particulares — Página de clases particulares. url: https://www.tusclasesparticulares.com/ (visitado 22-03-2021)

[2] Classgap — Página de clases particulares. url: https://www.classgap.com/es (visitado 22-03-2021)

[3] Infoclases — Página de clases particulares. url: https://infoclases.com/ (visitado 22-03-2021)

[4] FIB UPC Aula Lliure — url: https://www.fib.upc.edu/ca/estudis/graus/pla-daccio-tutorial/aula-lliure (visitado 22-03-2021)

[5] Taiga — Your opensource agile project management software. url: https://www.taiga.io/ (visitado 20-06-2021)

[6] Github — Where the world builds software. url: https://github.com/ (visitado 20-06-2021)

[7] Sueldos de la empresa — payscale.com. url: https://www.payscale.com/rcsearch.aspx?category=Job&CountryName=Spain (visitado 14-03-2021).

[8] Alquiler de oficinas — Aurea co-working. url: https://www.aureacoworking.com/precios-coworking/ (visitado 15-03-2021)

[9] J. Robertson and Suzanne Robertson, Atlantic Systems Guild. Volere. Requirements Specification Template.

[10] ReactJs — Biblioteca de JavaScript para construir UI. url: https://es.reactjs.org/ (visitado 02-06-2021)

[11] NodeJs — Entorno de ejecución para JavaScript. url: https://nodejs.org/es/ (visitado 02-06-2021)

[12] MongoDB Cloud Atlas — Servicio de MongoDB en el cloud. url: https://www.mongodb.com/es/cloud/atlas (visitado 02-06-2021)

[13] Modelo-Vista-Controlador — Que es MVC. url: https://desarrolloweb.com/articulos/que-es-mvc.html (visitado 05-06-2021)

[14] MongoDB — Base de datos NoSQL. url: https://www.mongodb.com/ (visitado 05-06-2021)

[15] Cliente-Servidor — Client-server Architecture. url: https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/ClientServer.pdf (visitado 05-06-2021)

[16] Fachada — Facade pattern: interfaz unificada para proyectos de software. url: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/facade-pattern/ (visitado 06-06-2021)

[17] Singleton — Patron singleton: una clase propia. url: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/patron-singleton/ (visitado 06-06-2021)

[18] Composite — Composite Design Pattern. url: https://sourcemaking.com/design_patterns/composite (visitado 06-06-2021)

[19] JavaScript — MDN Web Docs: JavaScript. url: https://developer.mozilla.org/es/docs/Web/JavaScript (visitado 15-06-2021)

[20] JSX — Presentando JSX. url: https://es.reactjs.org/docs/introducing-jsx.html (visitado 15-06-2021)

[21] Postman — The Collaboration Platform for API Development. url: https://www.postman.com/ (visitado 15-06-2021)

[22] Visual Studio Code — Code editing. Redefined. url: https://code.visualstudio.com/ (visitado 15-06-2021)

[23] npm — Build amazing things. url: https://www.npmjs.com/ (visitado 21-06-2021)

[24] JWT — Qué es Json Web Token y cómo funciona. url: https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/ (visitado 15-06-2021)

[25] jwt-simple — JWT encode and decode module for node.js. url: https://www.npmjs.com/package/jwt-simple (visitado 15-06-2021)

[26] bcrypt — A library to help you hash passwords. url: https://www.npmjs.com/package/bcrypt (visitado 15-06-2021)

[27] hashing — ¿Qué es el Hashing?. url: https://academy.binance.com/es/articles/what-is-hashing (visitado 15-06-2021)

[28] axios — Promise based HTTP client for the browser and node.js. url: https://www.npmjs.com/package/axios (visitado 15-06-2021)

[29] heroku — Cloud application platform. url: https://www.heroku.com/ (visitado 15-06-2021)

[30] AWS — Cloud computing. url: https://aws.amazon.com/es/ (visitado 15-06-2021)

[31] ReactJs Tutorial. Crear una nueva aplicación React. url: https://es.reactjs.org/docs/create-a-new-react-app.html#create-react-app (visitado 15-06-2021)

[32] Agile estimating and planning - Cohn, M, Prentice Hall Professional Technical Reference, 2006.

[33] Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development - Larman, C, Prentice Hall PTR , 2005.

[34] Service design patterns: fundamental design solutions for SOAP/WSDL and RESTful web services - Daigneau, R, Addison-Wesley, 2012.

[35] Principles of distributed database systems - Ozsu, M.T.; Valduriez, P, Springer, 2020.

[36] Agile software development: principles, patterns, and practices - Martin, R.C, Pearson Education Limited , 2014.