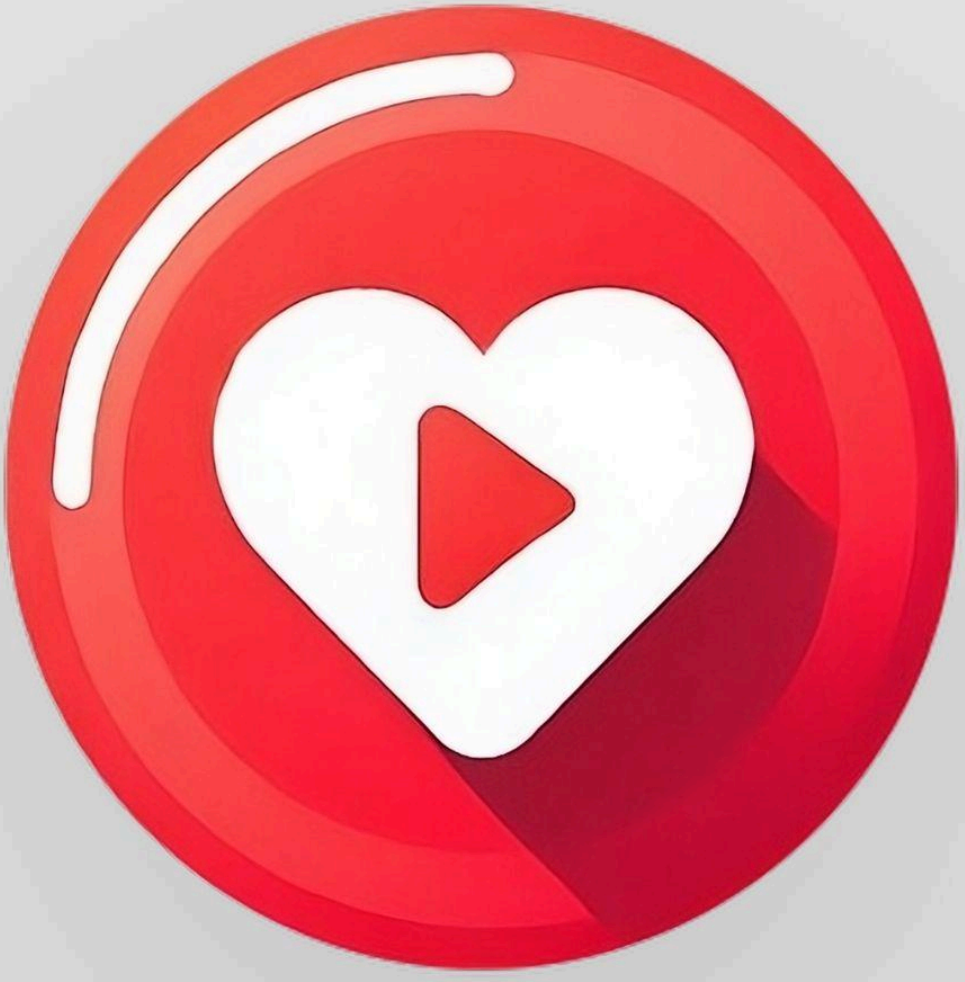
PLAN DE GESTIÓN, ANÁLISIS, DISEÑO Y MEMORIAL DEL PROYECTO LOVEROOM

GRUPO 06. SUSAN KARE



LoveRoom

ANDREI DUMBRAVA 844417
MIGUEL ARASANZ 839385
ALEJANDRO BENEDI 843826
SERGIO GARCÉS 843214
GUILLERMO BAJO 842748
CARLOS DE VERA 848481
JAVIER JULVE 840710
JAVIER FERRERAS 816410

GITHUB

FECHA: 11/03/2024

ÍNDICE

1. Introducción

Este documento define cómo va a ser el proceso de desarrollo y gestión de LoveRoom, una revolucionaria aplicación de citas web y móvil basada en el emparejamiento de personas y la creación de salas privadas, donde las parejas pueden reproducir vídeos de YouTube de manera sincronizada mientras chatean en directo para conocerse.

El desarrollo de este proyecto se llevará a cabo en un plazo estimado de unos tres meses, con seguimientos mensuales y entregables entre los que destacan la Memoria de la 1ª iteración del proyecto (15/03/24), Resultados de la P4 evaluación del proyecto y final de la Fase 1 (19/04/24), para cuando se tendrá la funcionalidad principal desarrollada. Destaca en la segunda fase la demostración final de Software en dispositivos reales (13/5/24) y la versión final y actualizada del plan de gestión (20/5/24).

El principal funcionamiento de la app se resume a continuación; Tras registrarse, los usuarios completarán su perfil, indicando ciertos atributos como su género, edad, localización, géneros y edades que les interesan, y podrán añadir fotos, descripción, etc. Después, en la página principal del usuario se mostrarán todos los vídeos que están siendo vistos por las personas de interés del usuario (misma ciudad, género y edades de interés de ambos lados coinciden).

Si en la página principal el usuario ve algún vídeo que le llame la atención, podrá clicar en él y automáticamente se le emparejará (match) con una de las personas de interés que estaban viendo ese vídeo. En ese momento se creará una sala privada (room), que se guardará en el perfil de ambas personas, y a la que ambos podrán conectarse en todo momento para chatear y conocerse de forma entretenida y emocionante mientras ven vídeos de manera sincronizada (pudiendo cambiar de vídeo, pausar, adelantar, etc). Dentro de la sala ambos podrán consultar el perfil del otro usuario para ver sus fotos y más detalles. Los usuarios podrán desconectarse y reconectarse a la sala libremente, y en cualquier momento podrán abandonar definitivamente la sala (si deciden que no quieren continuar conociéndose), y esta se borrará de sus perfiles y se evitará que ambos usuarios se emparejen en el futuro.

Si en la página principal no hay ningún vídeo que le interese al usuario, siempre podrá optar por hacer una búsqueda de cualquier vídeo de YouTube y seleccionar uno para ver. Al entrar al vídeo, si no había ninguna persona de interés para hacer match, el usuario se quedará viendo el vídeo a la espera de que alguien clique también en ese vídeo para hacer match con él, ya que el vídeo ahora aparecerá en las páginas principales de otras personas. Cuando ocurre un match se repite el proceso descrito en el párrafo anterior.

2. Organización del proyecto

En aras de una mejor organización, se han repartido diversos roles dentro del grupo, asignados a quien se ha considerado más adecuado atendiendo a diversos factores, tales como un nivel más avanzado de conocimientos en el ámbito. Dadas sus dotes de liderazgo e ilusión por el proyecto, Javier Ferreras lidera el proyecto como director. Por otra parte, Miguel Arasanz desempeña la labor de coordinador, facilitando la distribución de tareas y resolución de conflictos cuando sea necesario. También comunicará las horas y dedicación realizadas por cada miembro del equipo de forma semanal a través de un formulario. Además, Sergio Garcés destaca como responsable de la versión web del sistema, dados sus elevados conocimientos en desarrollo web, mientras que Andrei Dumbrava toma el rol de responsable de la versión móvil, debido a sus extensas habilidades en los respectivos ámbitos, las cuales pueden servir de gran ayuda para el resto de integrantes de la organización. Por último, Carlos de Vera tomará responsabilidad con lo respectivo a la comunicación con el backend y su implementación. Alejandro Benedi, Javier Julve y Guillermo Bajo participarán activamente en el desarrollo de la aplicación.

Cada sección principal está compuesta por tres integrantes menos web que estará formado por dos. Estos llevarán a cabo por norma general el desarrollo en su respectivo ámbito: Frontend React por Andrei Dumbrava, Guillermo Bajo y Miguel Arasanz; Frontend Web por Sergio Garcés, Javier Julve y Backend por Carlos de Vera, Javier Ferreras y Alejandro Benedi.

Aunque parezca menos importante, también se repartieron las responsabilidades de la realización de formularios, despliegues o del control del github. Por norma general se decidió que la realización de formularios la realizara el coordinador, que es la persona encargada de distribuir tareas y resolver conflictos. En cuanto a los despliegues, el equipo del backend sería el responsable de realizar estas tareas y guiar al resto del equipo para que puedan usarlas. El control del github lo realizan todos los integrantes por igual, siguiendo una distribución equitativa.

Aún teniendo esta distribución de responsabilidades, la gestión del proyecto se tratará de realizar de una forma horizontal en su mayor parte mediante comunicación activa entre individuos, democratizando el proceso de toma de decisiones y mejorando por tanto el flujo de ideas, la satisfacción general de sus integrantes y el progreso del proyecto como tal.

3. Plan de gestión del proyecto

Inicio del proyecto

Para asegurar la excelencia en el desarrollo del proyecto, se ha diseñado un riguroso plan de pruebas que abarca diversos aspectos clave. Por una parte, nos enfocamos en la validación y optimización de la experiencia de usuario en dispositivos móviles Android. Para ello, se ha utilizado la versátil aplicación Expo Go, la cual permite ejecutar pruebas exhaustivas en una amplia gama de dispositivos, asegurando así la funcionalidad y compatibilidad de la aplicación en distintas plataformas.

En paralelo, se han dedicado esfuerzos a perfeccionar la aplicación web mediante pruebas continuas en un entorno de desarrollo local proporcionado por Angular. Este servidor de desarrollo ofrece la flexibilidad necesaria para realizar pruebas de diseño de manera eficiente, garantizando que la aplicación web cumpla con los estándares de calidad y rendimiento esperados.

En cuanto a la gestión de datos, se ha optado por una solución en la nube para alojar la base de datos y los servicios web. Clever Cloud ha sido seleccionado como proveedor de servicios en la nube, brindando la infraestructura necesaria para almacenar y gestionar eficientemente los datos del proyecto.

Además, para enriquecer la funcionalidad de la aplicación con contenido multimedia, se propone integrar videos de YouTube. Para lograr esto, se ha procedido con el registro en la API de YouTube, lo que nos permite acceder a su vasta biblioteca de videos y ofrecer una experiencia más completa a los usuarios.

En el contexto de la planificación de formación para el desarrollo del proyecto, se ha optado por utilizar PostgreSQL como sistema de gestión de base de datos. Para el desarrollo del servidor web, se ha seleccionado Node.js junto con TypeScript, y se integrará Prisma y Angular para potenciar la funcionalidad y la interfaz de usuario. Respecto a la aplicación móvil, se ha optado por emplear React Native en combinación con Expo para garantizar un desarrollo ágil y efectivo.

Dado que la mayoría de los miembros del equipo no poseen experiencia previa en estas tecnologías, se han asignado roles dentro del proyecto basándose en el nivel de experiencia existente. Se ha asignado cada integrante al equipo de trabajo en el cual mostraba unas dotes más avanzadas, ya sea en desarrollo de servidores web, de aplicaciones móviles o de backend.

Para abordar esta necesidad, se ha implementado un plan de formación que incluye una variedad de recursos, como videotutoriales, documentación oficial y cursos en plataformas educativas en línea, como GitHub Education o Codeacademy. Estos recursos han permitido adquirir el conocimiento necesario para llevar a cabo las responsabilidades de manera efectiva y contribuir al éxito del proyecto. Asimismo, se ha compartido entre los integrantes del equipo un directorio de documentación en el que se han ido anotando aspectos importantes sobre las diversas tecnologías empleadas. Esto ha resultado ser una práctica muy útil para garantizar que todos los miembros del equipo estén al tanto de los detalles esenciales de las herramientas y plataformas utilizadas en el proyecto. Este enfoque promueve la transparencia y el intercambio de conocimientos dentro del equipo, lo que a su vez mejora la eficiencia y la colaboración en el proyecto.

Control de configuraciones, construcción del software y aseguramiento de la calidad del producto

Como se ha mencionado previamente en la organización del equipo, Javier Ferreras asumirá el papel de director del proyecto, siendo responsable de garantizar que el producto se desarrolle correctamente y siguiendo el plan acordado por todos. No obstante, todos los miembros del equipo trabajarán en conjunto para cumplir con los objetivos y completar todas las tareas dentro de los plazos establecidos.

En cuanto a las convenciones de nombrado de documentos y ficheros, se ha establecido un enfoque coherente y claro para garantizar la organización y la facilidad de navegación dentro del proyecto. Se ha adoptado la convención de nombrar los documentos de manera descriptiva y significativa, reflejando su contenido y función dentro del proyecto. Esto incluye el uso de nombres que sean comprensibles para todos los miembros del equipo y que sigan una estructura coherente en todo el proyecto.

Además, se han seguido las convenciones específicas para el nombrado de archivos dentro de cada tecnología utilizada en el proyecto. Por ejemplo, en el caso de la carpeta "src" en el desarrollo web y móvil, se seguirá una estructura de nomenclatura que refleje la función y el tipo de archivo, facilitando así su identificación y mantenimiento a lo largo del tiempo. En React Native, las vistas de pantallas se describirán con sufijo Screen y se encontrarán en la carpeta homónima, mientras que los componentes se encontrarán en la pantalla componentes y estarán formados por un nombre descriptivo.

Respecto a los estándares de código y guías de estilo, se han seleccionado y adoptado aquellos que mejor se adapten a las tecnologías utilizadas en el proyecto. Se ha dado prioridad a las guías de estilo recomendadas por la comunidad y las prácticas de codificación establecidas por las tecnologías específicas, como la style guide oficial de Angular o la de React de AirBnb. Estos estándares y guías proporcionarán consistencia en la estructura y el formato del código en todo el proyecto, lo que facilita la colaboración entre los miembros del equipo y mejora la legibilidad y mantenibilidad del código a lo largo del ciclo de vida del proyecto. Además, se ha fomentado la revisión periódica de las guías de estilo y la retroalimentación entre los miembros del equipo para garantizar su eficacia y relevancia continuas. También nos ayudaremos de herramientas formateadores de código como Prettier para confirmar el correcto seguimiento de las guías de estilo. El nombrado de las variables, rutas y ficheros será en inglés y la tabulación de 4 espacios. Se tratará de evitar comentar excesivamente ya que el propio nombrado debe ser lo suficientemente descriptivo, pero se podrá hacer comentarios en caso de que aporte información necesaria.

Como se ha adelantado anteriormente, para la construcción del software, se han utilizado herramientas como Visual Studio Code y Git. Visual Studio Code ha sido fundamental para la escritura y edición de código, proporcionando un entorno de desarrollo intuitivo y funcionalidades útiles para la productividad del equipo. La instalación de diversas extensiones en VSC facilita significativamente el trabajo. La extensión LiveShare para trabajar diversos integrantes en tiempo real sobre el mismo código es una de las extensiones más útiles.

Por otra parte, Git ha sido esencial para el control de versiones del proyecto, permitiendo el seguimiento de los cambios en el código, la colaboración entre los miembros del equipo y la gestión eficiente de las ramas de desarrollo. El proyecto se encuentra en una organización, teniendo un repositorio distinto para el backend, frontend app y frontend web. Para realizar cambios en los repositorios no se ha establecido ninguna restricción de gran peso, cualquier integrante del equipo puede realizar modificaciones y hacer commits en cualquier momento y sin necesidad de añadir un mecanismo de aprobación de commits. De todas formas, todo lo que se guarde en la rama principal debe estar testeado de manera local y no contener errores que impidan el correcto funcionamiento, siendo esta rama la versión de última *release* en todo momento y pudiendo ser utilizada por miembros que desarrollan otra sección sin problema.

Para el desarrollo de componentes y nuevas funcionalidades se crearán nuevas ramas, con nombre descriptivo sobre la funcionalidad que se está desarrollando. Una vez esté terminado y probado se realizará un merge a la rama principal. En caso de que dos integrantes se enfrenten a un merge, estos deben consensuar sus cambios para evitar conflictos. Esta coordinación garantiza una integración suave y sin problemas de los cambios, asegurando un desarrollo fluido y eficiente del proyecto.

En cuanto a la documentación del proyecto, se ha utilizado un sistema con historial de versiones integrado a través de Google Drive y sus gestores de documentos Google Docs y Google Sheets. Esto nos permite un desarrollo concurrente y con control sobre qué miembro del equipo ha escrito algo y la capacidad de retroceder a versiones anteriores en caso de que fuera necesario. Los documentos realizados deberán seguir las normas y estructura de documentos técnicos, con portada, índice y secciones, referenciación cruzada, bibliografía, correcta redacción, concisa y sin ambigüedad, numeración de tablas y con una tipografía de entre 10 y 12 puntos.

Para asegurar la calidad del código, se han llevado a cabo una serie de medidas de control. Entre ellas destacan las revisiones de código por pares, donde cada pieza de código es revisada por al menos otro miembro del equipo. Esta práctica no solo ayuda a identificar posibles errores o problemas de implementación, sino que también fomenta la colaboración y el aprendizaje entre los distintos miembros del equipo. Además, se han realizado revisiones tanto de requisitos como de diagramas UML para validar y refinar los requisitos antes de comenzar la implementación. Para asegurar la calidad del código en el servidor de datos se utilizó Jest, una robusta herramienta para la realización de pruebas. Su utilización simplificó la creación y ejecución de pruebas, las cuales se encuentran alojadas en un subdirectorio dedicado denominado “tests”.

Por otra parte, se ha creado una guía de estilo para asegurar un diseño homogéneo entre los distintos dispositivos, abordando aspectos como el color, el diseño de botones y otros elementos visuales. Estéticamente, a lo largo de todas las implementaciones de LoveRoom, trataremos de mantener una única identidad visual, dando continuidad a la marca en distintos dispositivos. Para ello usaremos una paleta de colores que muestre que es una aplicación de citas, tomando derivados del rojo pasión pero evitando el uso directo de colores excesivamente saturados para conseguir una mejor legibilidad y accesibilidad a personas con discapacidades visuales. Estará principalmente formada por rosa palo (#F89F9F) y blanco (#FFFFFFF), aunque se utilizarán otros como negro (#000000) y gris claro (#343a40) para detalles. El logotipo se usará sobre fondos claros para asemejarse en mayor medida a Youtube. La tipografía será Arial y el tamaño de fuente para encabezados 24px, texto principal 16px y texto secundario 14px. Los botones tendrán los bordes redondeados. Aún así, el diseño debe ser responsive para dispositivos de diverso tamaño y forma, por lo que estas especificaciones pueden cambiar según sea necesario.

El diseño tratará de ser minimalista y dar importancia al contenido, consiguiendo un uso efectivo e intuitivo para todo tipo de usuarios, siendo fácil su aprendizaje y evitando los posibles errores de los usuarios. El tono de voz debe ser amigable, pero respetuoso, además de animado y positivo. Buscamos una experiencia positiva y auténtica para los usuarios finales, por lo que se fomentará el respeto mutuo y la esencia despreocupada de la aplicación. La creación y seguimiento de esta guía garantiza una experiencia de usuario coherente y atractiva en todas las plataformas donde se despliega el software diseñado.

Mapas de navegación:

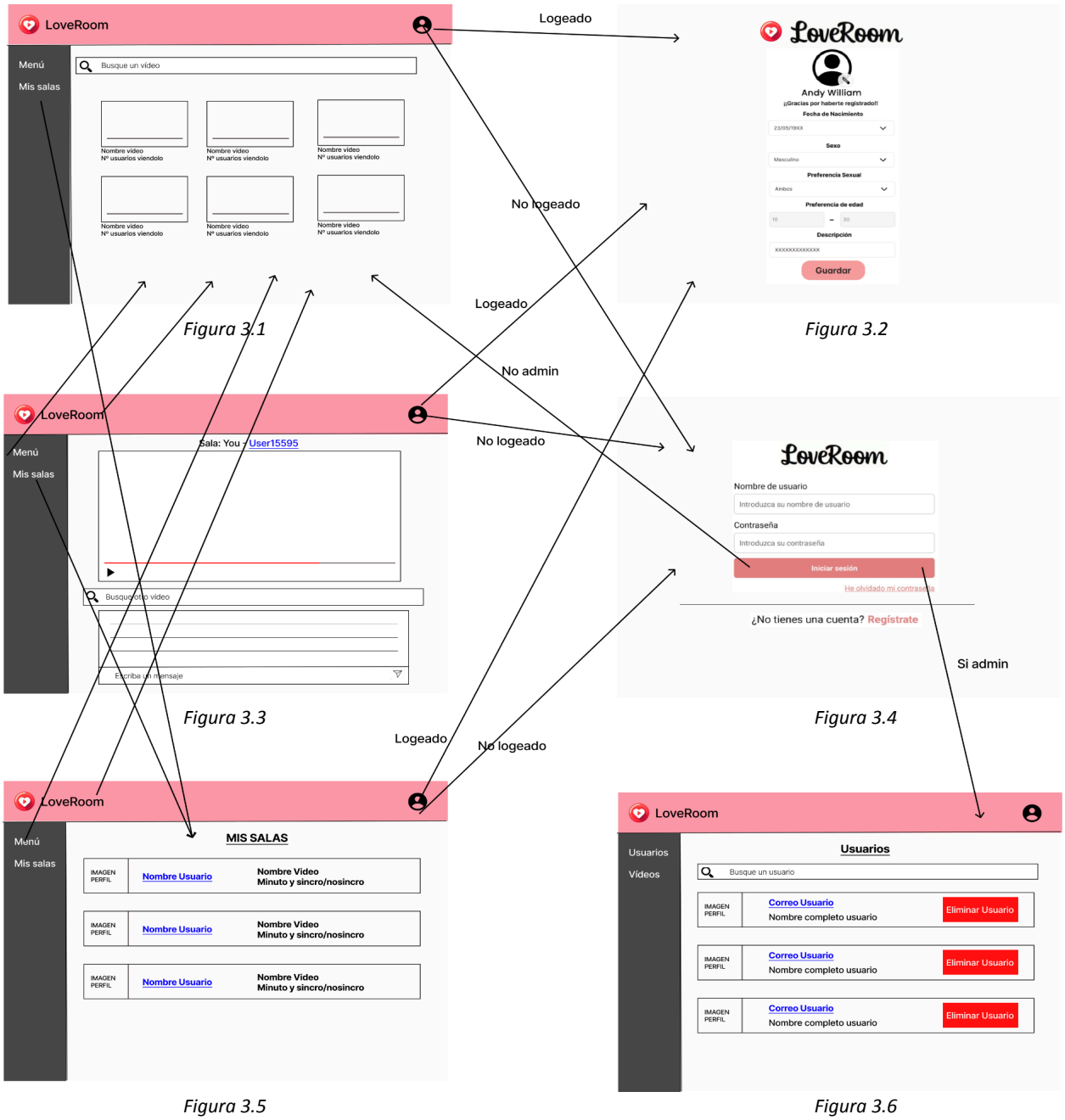


Figura 3.1

Figura 3.2

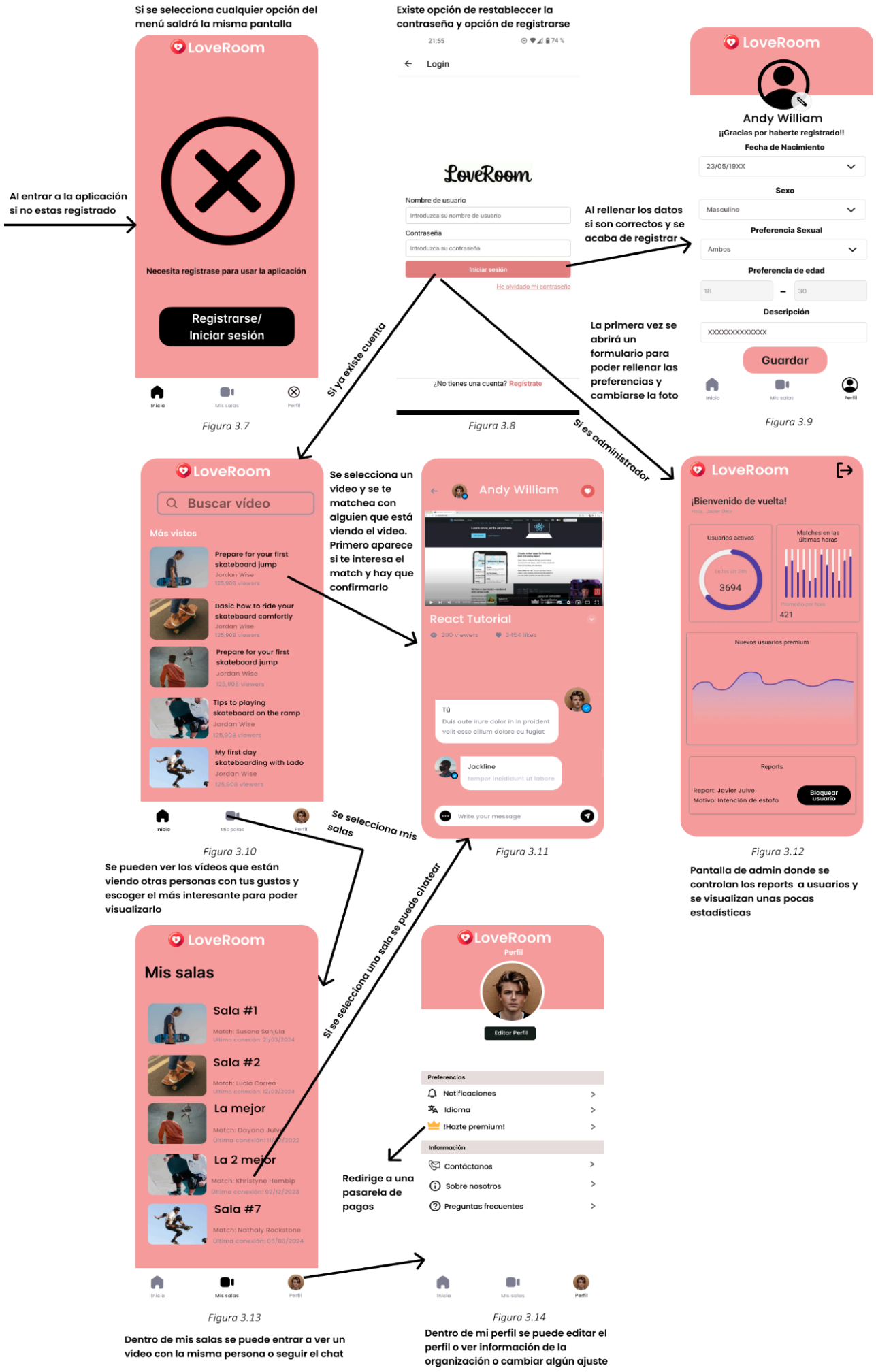
Figura 3.3

Figura 3.4

Figura 3.5

Figura 3.6

Mapa de navegación básico de la aplicación web



Mapa de navegación básico de la aplicación móvil

La muestra de resultados a clientes se realizará en estos dispositivos reales, sin usar emulación, y con las versiones mínimas tanto de navegadores como de sistemas operativos especificadas en el diseño del sistema. Para el android se entregará el APK listo para instalarse y la web y base de datos con una imagen Docker configurada para un despliegue fácil. También se entregará el código fuente, el cual será público en GitHub.

Calendario del proyecto, división del trabajo, coordinación, comunicaciones, monitorización y seguimiento

Los responsables del proyecto ya se han explicado en el punto 2, destacando a Javier Ferreras como director del proyecto.

Para la primera iteración (Fase 1), la cual suponemos como fecha final la evaluación y entrega hasta P4 del 19 de abril, el objetivo es tener implementados todos los requisitos imprescindibles para la funcionalidad del proyecto. Esto incluye todos los requisitos de creación y utilización de salas, creación y gestión de usuarios, interacción con la API de Youtube incluyendo búsquedas, sincronización de los videos. (RF 1, 2, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16, 18, 19). También habremos terminado el presupuesto y actualizado el plan de gestión, aunque tampoco será la versión definitiva.

Con ello conseguimos el principal objetivo del proyecto, dejando para la Fase 2, con fin en la demostración de software y la entrega final del examen toda la gestión de pagos y cuentas premium (RF3), el chat (RF9), fotos y videos locales para cada sala (RF10, 11), la capacidad de buscar otro video mientras ves uno (RF17) y la GUI de administrador y sus funciones (RF 19, 20, 21, 22, 23). También contaremos con la versión definitiva de la memoria.

Se ha realizado un diagrama de Gantt en el cual se muestran las distintas actividades a realizar a lo largo del proyecto, las fechas límite estrictas, y fechas de revisión del calendario y análisis de rendimiento las cuales hemos situado el viernes 29 de marzo y el miércoles 1 de mayo. Se ha utilizado un sistema EDT por códigos jerárquicos, divididos en Gestión, Backend y Frontend conjunto de app y web ya que las tareas a realizar y los tiempos que conllevan son similares. El frontend se ha dividido en interfaz y lógica, siendo en su mayoría el desarrollo de ambas concurrente y en tareas de componentes similares, dado que están vinculados y de esta manera se permite un proceso más eficiente. Se ha subdividido en las tareas necesarias para diferenciar correctamente todos los pasos pero sin microgestionar los procedimientos.

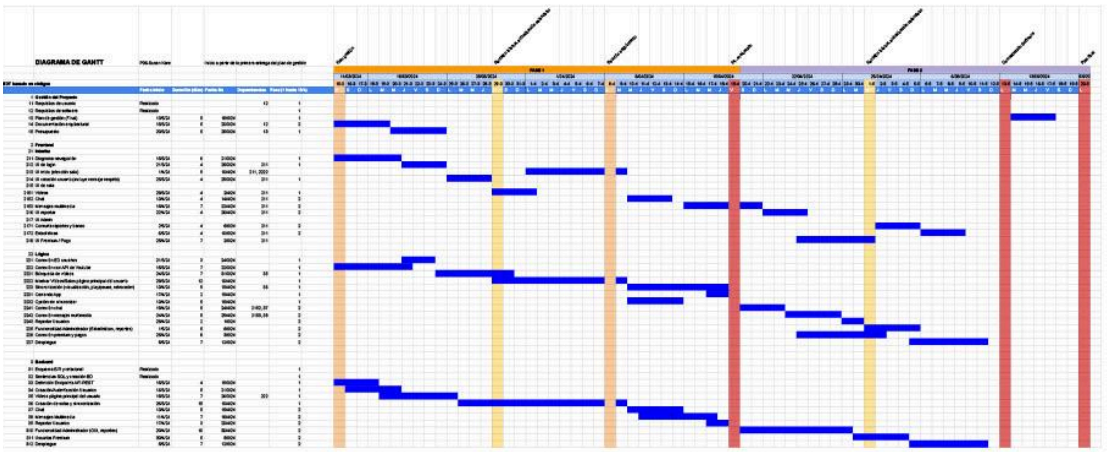


Figura 3.15 Diagrama de Gantt

Como se ha detallado anteriormente, cada sección principal está compuesta por tres integrantes que llevarán a cabo por norma general el desarrollo en su respectivo ámbito, internamente dividiendo las tareas modularmente en la medida de lo posible, con desarrollo en las ramas necesarias. Aún así ante bloqueos y para mejorar la productividad a través de la cooperación, cada equipo se ayudará mutuamente, facilitando el trabajo y transferencia de conocimientos utilizando comunicación activa. Se tratará que dentro de cada grupo la carga de trabajo sea equitativa, resolviendo en caso de conflicto el responsable de la sección. Tanto el diseño gráfico como las pruebas o los manuales serán repartidos por cada grupo de la manera más conveniente en el momento dado, teniendo un manejo flexible por tanto de responsabilidades facilitando el eventual seguimiento del plan. Si un grupo tiene una mayor carga significativa, será ayudado por el resto; siempre y cuando el motivo de ello no haya sido un menor rendimiento de este.

Dentro de cada grupo habrá una persona más enfocada a la gestión, siendo estos el director Ferreras, coordinador Arasanz y Garcés. Aunque participarán de igual manera en la implementación del sistema, tomarán un papel más activo en temas como la redacción y corrección del plan de gestión, presupuesto y recursos humanos, además de corroborar las medidas de progreso y las actas de reuniones. También para la resolución de disputas tendrán un mayor papel, escuchando a ambas partes y llegando a un acuerdo común donde todos los involucrados queden satisfechos. En el raro caso en que esto se dificulte excesivamente, la última palabra la tendrán el director y el coordinador del proyecto.

Para las comunicaciones internas se utilizará un grupo de Whatsapp conjunto, dado que resulta lo más cómodo y rápido, además de trabajo y comunicación presencial cuando es posible y Discord para trabajo conjunto remoto. Se realizarán dos reuniones de seguimiento para actualizar el calendario de actividades y comprobar el correcto progreso del trabajo. Estas se realizarán el 29 de marzo y el 1 de mayo y se realizará un acta de sesión donde se detallará el estado del proyecto y los cambios realizados.

En cuanto a las medidas de progreso, no utilizaremos métricas absolutas ya que pueden resultar arbitrarias (en especial siendo nuestro primer proyecto de estas características) y propensas a cumplir la ley de Goodhart. Por ello utilizaremos un modelo mixto, fijándonos en especial en el rendimiento por grupos, comprobando el cumplimiento de requisitos a lo largo del tiempo, el correcto seguimiento del diagrama de Gantt inicial y el seguimiento de esfuerzos individuales. También métricas basadas en el producto como las tendencias de la tasa de cambio de código y los commits de git. No utilizaremos LOC ya que al usar tres tecnologías distintas se convierte en una medida excesivamente inexacta de productividad y esfuerzo, pero sí nos servirá para realizar una estimación del esfuerzo y coste necesario. Como métrica de despliegue inicial será el tiempo hasta el despliegue de la funcionalidad útil más sencilla, teniendo como objetivo antes de la primera iteración. Para medir la productividad tendremos en cuenta el grado de satisfacción entre miembros, el número de issues resueltos y el tiempo dedicado al proyecto. Se buscará una productividad similar para todos los miembros, siendo delegado en el día a día a cada grupo.

En las reuniones de seguimiento tendremos en cuenta todos estos factores y los analizaremos en conjunto para encontrar el mejor funcionamiento posible del proyecto.

4. Plan de gestión del proyecto

Análisis de requisitos

Código	Descripción
RF-1	La aplicación debe permitir la creación de una cuenta para usar la aplicación, guardando para cada usuario información sobre su nombre, sexo, edad, orientación sexual, descripción y fotos.
RF-2	La orientación sexual de cada usuario se definirá mediante un rango de edades y sexo objetivo, y se utilizará para determinar qué usuarios pertenecen al conjunto de personas de interés de cada usuario.
RF-3	La aplicación debe permitir a los usuarios elegir entre una cuenta estándar, limitada a tres salas máximo o una cuenta premium con un coste económico, que les permita obtener salas ilimitadas.
RF-4	La aplicación debe permitir a los usuarios buscar vídeos de Youtube.
RF-5	La aplicación debe permitir a los usuarios seleccionar y reproducir un vídeo buscado.
RF-6	La aplicación debe mostrar a cada usuario en su menú principal los distintos vídeos que están siendo vistos por personas pertenecientes a su conjunto de interés.
RF-7	La aplicación debe permitir a los usuarios entrar en un vídeo que ya estén viendo otras personas.
RF-8	La aplicación debe emparejar a dos usuarios y crear una sala privada para ellos, cuando ambos estén viendo el mismo vídeo y ambos estén en el conjunto de personas de interés del otro.
RF-9	La aplicación debe permitir a los usuarios chatear dentro de su sala, enviando mensajes de texto.
RF-10	La aplicación debe permitir a los usuarios subir y enviar fotos y vídeos locales dentro de una sala para que puedan ser vistos por la otra persona.
RF-11	Las fotos y vídeos locales junto con los mensajes de cada sala se almacenarán de manera persistente y se recuperarán cuando un usuario entre en la sala.
RF-12	La aplicación debe permitir al usuario abandonar definitivamente la sala si no quiere seguir conociendo a la persona con la que ha hecho match, y ambos no volverán a emparejarse nunca. La sala será borrada.
RF-13	La aplicación debe guardar las salas activas de cada usuario para que pueda volver a usarlas cuando desee.
RF-14	La aplicación debe permitir a los usuarios de una sala ver un mismo vídeo de forma sincronizada y no sincronizada a elección de los usuarios.

RF-15	La aplicación debe permitir a un usuario que esté en una sala consultar el perfil del otro usuario, pudiendo ver su nombre, sexo, edad, orientación sexual, descripción y fotos.
RF-16	La aplicación debe permitir a cualquiera de los dos usuarios que esté en una sala pausar el vídeo, avanzar o retroceder en él, manteniéndose la sincronización (si estaba activada) entre ambos.
RF-17	La aplicación debe permitir a ambos usuarios que estén en una sala buscar otro vídeo para reproducirlo en la sala, manteniéndose la sincronización (si estaba activada).
RF-18	La aplicación debe permitir a cualquiera de los dos usuarios que esté en una sala cerrar la aplicación y entrar de nuevo en la sala sin que se pierda el punto del vídeo, ni se desincronicen (si estaba activada la sincronización).
RF-19	La aplicación mostrará un mensaje a los nuevos usuarios con normas de comportamiento que deberán cumplir para garantizar el respeto y tolerancia entre usuarios.
RF-20	La aplicación debe permitir a los usuarios de una sala reportar mensajes por conducta inapropiada o contenido gráfico, violento, sexualmente explícito, vulgar u ofensivo.
RF-21	La aplicación tendrá un tipo de usuario especial llamado Administrador.
RF-22	Los Administradores podrán consultar los reportes realizados por los usuarios y banear a aquellos usuarios que infrinjan las normas de comportamiento establecidas.
RF-23	La aplicación tendrá una GUI para Administradores con estadísticas sobre los usuarios de la aplicación.
RNF-1	La aplicación requerirá de conexión a internet en todo momento.
RNF-2	La aplicación asegurará el cifrado de los datos y las conexiones.
RNF-3	La aplicación estará disponible en formato web y como app móvil para Android.

Diseño de sistema

En el mundo del desarrollo de software, entender cómo está estructurado un sistema es vital. ¿Cómo se organizan las piezas y cómo se comunican entre sí. Dos herramientas clave para visualizar esta estructura son los diagramas de módulos y de despliegue. Estos diagramas nos dan una vista panorámica de los elementos que conforman el sistema y cómo interactúan entre sí.

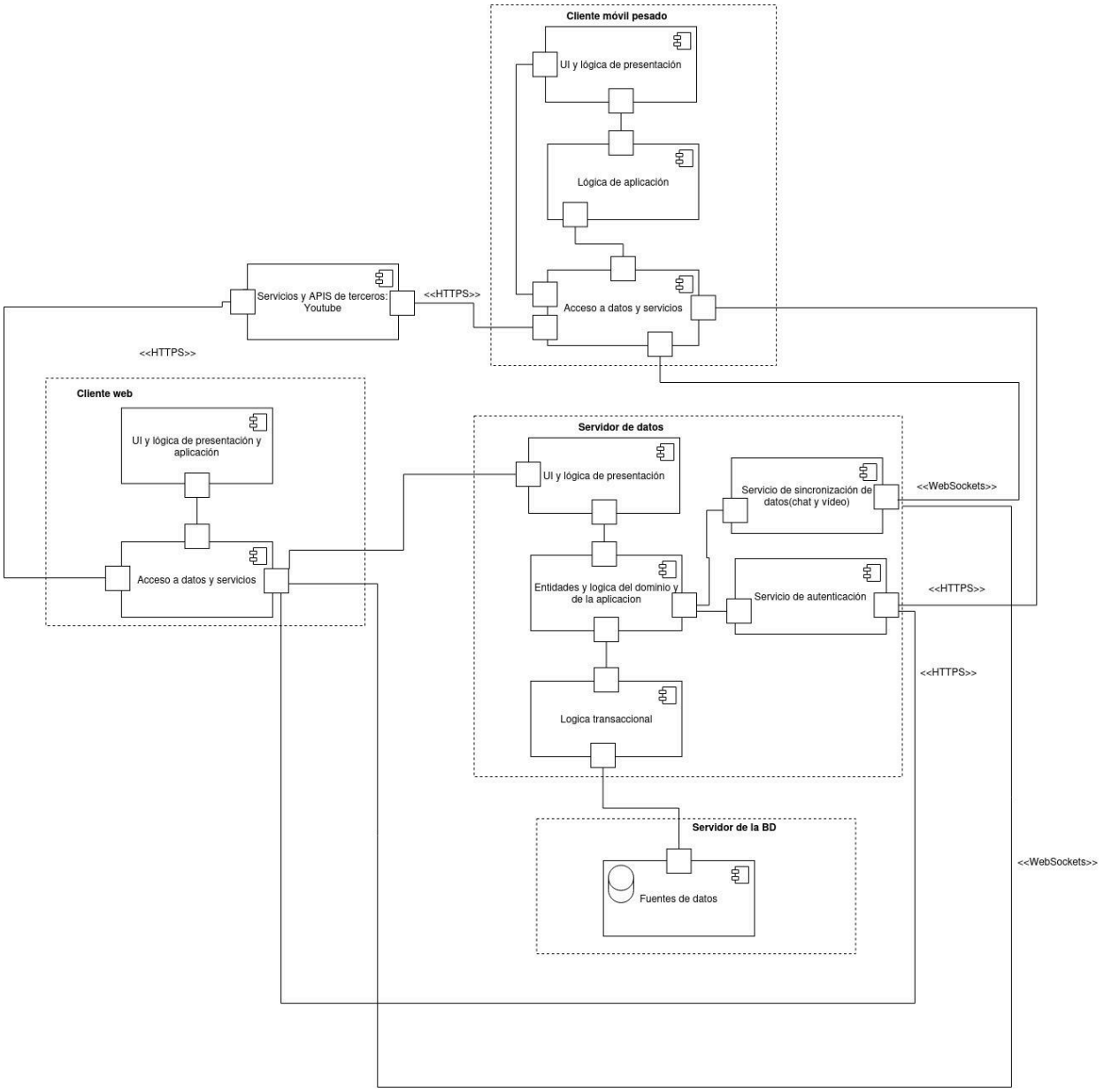


Figura 4.1 Diagrama de componentes del SI Loveroom

El sistema está diseñado para ofrecer una experiencia unificada tanto en navegadores web como en dispositivos móviles. Se ha optado por una arquitectura monolítica, en la cual un servidor web principal asume la responsabilidad de gestionar la lógica de la aplicación, la interfaz de usuario destinada a los clientes web, así como la gestión de datos. Toda la comunicación referente a la sincronización entre clientes es gestionada mediante Websockets por lo que los clientes app y web son ligeros, ya que se tiene que garantizar en todo momento que haya conexión a Internet, debido a que dicha sincronización se gestiona en el servidor de datos.

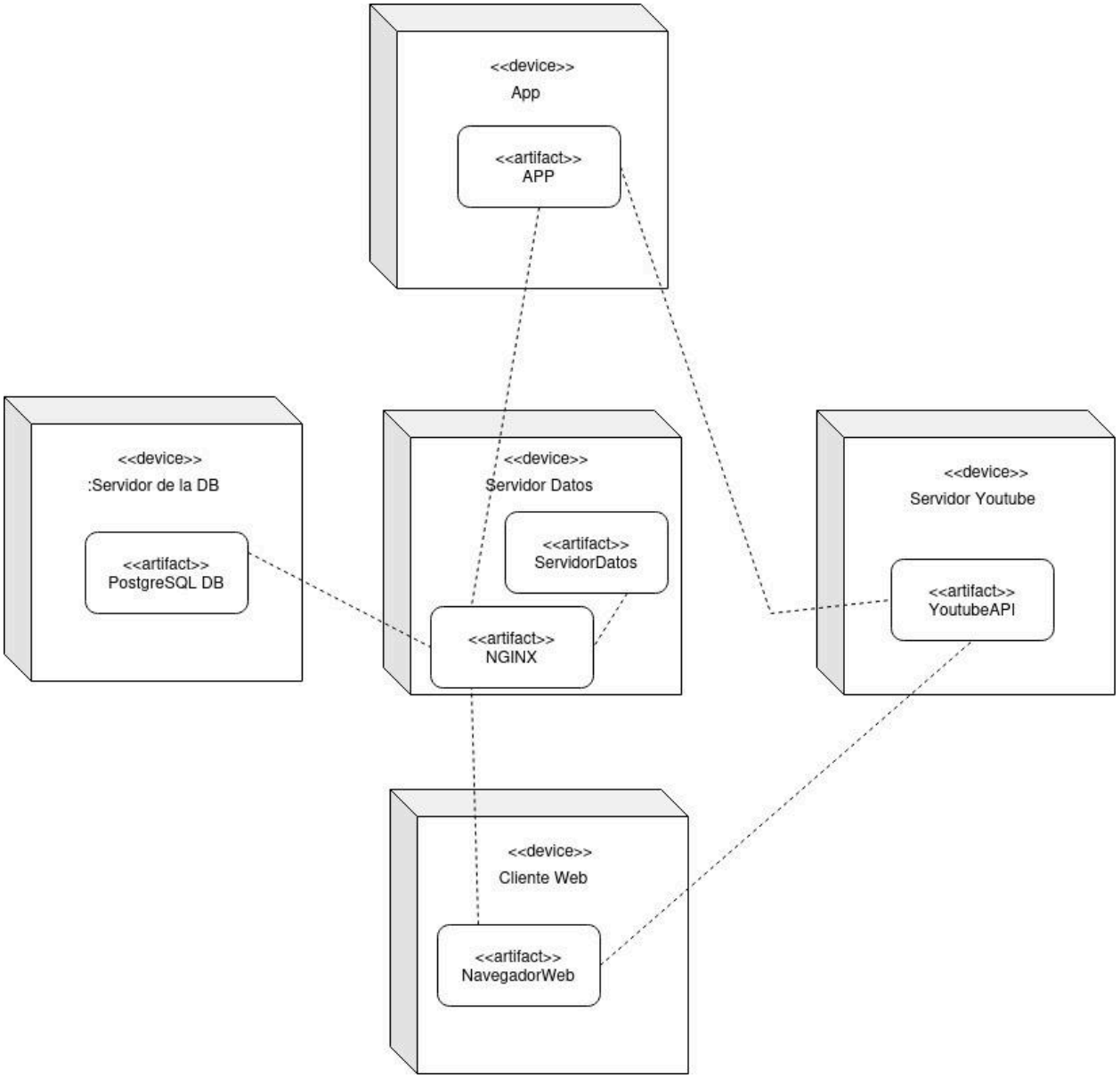


Figura 4.2 Diagrama de despliegue del SI Loveroom

Para el desarrollo se va a utilizar el patrón MVC (Modelo - Vista - Controlador), un enfoque arquitectónico que proporciona una organización estructurada y modular para nuestras aplicaciones y mantenibilidad del código, además se ha utilizado el patrón Adapter para la pasarela de pagos, para que la entidad usuarioController pueda interactuar con la pasarela de pagos a través de operaciones de alto nivel sin preocuparse de cómo se realizan internamente esas operaciones. Asi como se va a realizar un despliegue en 3 niveles.

Tecnologías utilizadas

A continuación se detallan las tecnologías utilizadas para el desarrollo del sistema.

Base de datos:

- PostgreSQL(v15): PostgreSQL será la elección como sistema de gestión de bases de datos relacional. Su soporte de SQL avanzado y robustez serán clave para el manejo de grandes conjuntos de datos, en la etapa de desarrollo se va a utilizar Clever cloud para alojar la base de

datos, una vez se pase a producción se integrará con Azure ya que ofrece mejores servicios que AWS y Clever cloud.

Servidor Web:

- Node.js(v18.19.0): Se utilizara node como entorno de ejecución para el servidor web debido a su eficiencia y escalabilidad en entornos basados en eventos, habiéndo elegido esta opción frente a ruby on rails debido a lo expuesto anteriormente.

- TypeScript(v5.3.3): TypeScript será el lenguaje de programación principal para el servidor frente a Javascript debido a que el uso de tipificación estática mejora la robustez del código, especialmente durante pruebas y mantenimiento.

- Prisma(v5.10.2): Prisma se emplea como un ORM específico de Node.js para interactuar con la base de datos PostgreSQL. Facilitará la abstracción de la capa de persistencia y proporcionará una interfaz intuitiva frente al uso de raw queries.

- Angular(v17.2.2): Angular en su versión más reciente a día de hoy, la 17, se utilizara como framework para el diseño del frontend de la página web. Este framework permite una fácil división de los elementos de la web en componentes, lo que facilita su reutilización en las diferentes partes de la aplicación. Angular nos permite en la fase de diseño compilar e iniciar un servidor de desarrollo que proporciona recarga en caliente del navegador cuando se detectan cambios en los archivos del código fuente. Además tiene una buena compatibilidad con TypeScript e incorpora técnicas de carga para obtener un rendimiento mejorado. Se barajó la opción de trabajar con Ruby on Rails, dado que algunos miembros del equipo ya los habían utilizado previamente, pero tras la consulta a un experto en la materia, y descubriendo la posibilidad de Angular, vimos que Ruby on Rails es más recomendable para aplicaciones web de tipo formulario, y Angular se adapta mejor a nuestras necesidades.

Aplicación:

- React Native(v0.73.4): la aplicación móvil se ha desarrollado con React Native, permitiendo la creación de aplicaciones nativas para Android utilizando Javascript y React. Este framework creado por Facebook permite un desarrollo multiplataforma con un desarrollo bastante rápido y sencillo, pues mezcla los conceptos del desarrollo móvil con los de React, permitiendo una experiencia de usuario nativa y adaptándola a cada plataforma. Además tiene una gran comunidad y soporte; y un rendimiento sólido, con una capacidad de renderizado bastante eficiente.
- Expo(v50.0.8): Expo se ha utilizado para simplificar el desarrollo y la implementación en React Native, proporcionando herramientas y servicios que faciliten el proceso. Expo proporciona un inicio rápido, encargándose de la configuración de compilación y el empaquetamiento de la aplicación. Su despliegue es muy sencillo, permitiéndonos probar nuestra aplicación desde la aplicación escaneando un simple código QR.

Otras consideraciones a tener en cuenta son, el desarrollo de una API RESTful para poder asegurar una comunicación fiable entre la frontend y el backend, este elemento es crítico ya que se necesita asegurar que el cambio de un usuario entre dispositivos resulte transparente; además se implementarán medidas de seguridad como la autenticación de usuarios mediante JWT(JSON Web Token) y cifrado de datos en tránsito (HTTPS). También, aunque ya se ha mencionado, nuestra base de datos es SQL, se barajaron distintas opciones, pero como la mayoría de integrantes tienen más conocimientos sobre este tipo de bases que sobre las NoSQL, se acabó optando por esta opción.

En cuanto a la compatibilidad con distintas versiones de dispositivos, para su uso en web, el framework que se va a utilizar es Angular, las versiones mínimas que puede tener el navegador utilizando Angular(v17.2.2), siendo en los más populares: Chrome a partir de las dos versiones más recientes, por

lo que tomamos v122.0.6261.128 para evitar actualizar las versiones continuamente, Safari (v17.4) y Microsoft Edge (v122.0.2365.52) a partir de las dos versiones mayores más recientes y en Mozilla Firefox (v123.0.1) con las versiones latest y extended support release (ESR). Estas versiones son las que utiliza la mayoría de los usuarios por lo que no supondrá una reducción significativa dentro del rango de posibles usuarios. Las fijamos en estos valores para evitar tener que actualizar el código para cumplir la compatibilidad de las versiones en el largo plazo.

Para el desarrollo móvil, React Native(v0.73.4) tiene unos requisitos mínimos de compatibilidad de versiones de Android para Android 6.0 (API 23). Aun así, dado que no tenemos un dispositivo con este sistema operativo, tomaremos como versión mínima de Android 13 (API 33), aunque debería ser compatible con versiones más antiguas. Por último, aunque no menos importante, es necesario poseer de una conexión a internet para poder usar la aplicación, pues se realizan conexiones entre usuarios continuamente.

También hay que destacar que nuestro código va a ser en todo momento open-source. Este se regirá por la licencia GNU General Public License, es decir, cualquier persona puede acceder a nuestro código fuente y modificarlo para su propio interés o sugerir distintos cambios, pero se mantiene el derecho de Copyleft, asegurando que todas las versiones modificadas o derivadas se mantengan bajo la misma licencia.

5. Memoria del proyecto

Inicio del proyecto

Como en la mayoría de proyectos, el inicio suele ser bastante complicado. Todos los integrantes del equipo tratan de cumplir con lo establecido y aprender lo máximo posible para desempeñar una correcta función en el trabajo que deben realizar. Al principio la mayoría de integrantes dedicó su tiempo a mirar tutoriales para aumentar el grado de conocimiento de las tecnologías que debían abordar. Tras esto, se inició con la tarea de escribir código, y poco a poco ir integrándolo para que todos puedan usarlo y darle una funcionalidad a la aplicación. Esta fase suele ser bastante costosa, pues se debe entender lo que se está haciendo y cumplir con lo establecido, cosa que a veces no es tan fácil.

En un principio el desarrollo del servidor de datos iba a realizarse en Ruby On Rails, un framework para el desarrollo de aplicaciones web, sobre todo especializada en el uso de formularios, debido a que nuestro sistema se basa en la comunicación front-back se cambió a Node.js ya que ofrece mayor escalabilidad y se ajusta a las características de nuestro sistema, ya que ya están definidas librerías para la gestión de Web Sockets. También se cambió el desarrollo móvil, pues se iba a desarrollar en Kotlin, pero decidimos utilizar React Native porque ofrece un desarrollo más integrado con el ecosistema Android, facilidad para usar componentes nativos y se puede integrar a iOS si en algún punto nos interesa si la aplicación está teniendo éxito.

Ejecución y control del proyecto

El **reparto de trabajo** entre los miembros del equipo se realizó de acuerdo con el plan inicial, dividiendo las tareas modularmente dentro de cada grupo. En el front React, se dividió en secciones como multimedia, perfil, administrador, sincronización, etc., ocupándose cada uno de los tres en especial de determinadas partes. Aún así la comunicación y ayuda entre compañeros fue continua y hubo muchas ocasiones donde se trabajaba conjuntamente y en los mismos objetivos, en especial al principio del proyecto. El desarrollo en el backend se dividió en dos tareas principales que cubrían la mayor parte de requisitos funcionales; usuarios, y sincronización y salas. Debido a que el equipo constaba de 3 personas, uno de los miembros se dedicó a la tarea de usuarios, y los otros dos a la tarea de sincronización y salas. Tras completar las tareas principales, se fueron desarrollando el resto de requisitos funcionales restantes (administración, etc). Además en cada tarea se incluye el desarrollo y prueba de los test correspondientes. Adicionalmente, una vez que el backend fue desplegado correctamente, los integrantes de dicho grupo se repartieron entre los demás grupos de desarrollo, moviéndose dos de los integrantes al front web y el restante al grupo de desarrollo de la App, tratando de ayudar sobre todo en temas integración frontend-backend. Por último en el front Angular el reparto de trabajo durante todo el proyecto fue muy sencillo, ya que durante gran parte del proyecto fue tarea de dos personas únicamente. La comunicación entre estos dos miembros ha sido excelente para una buena organización entre ambos y poder ir completando todos los requisitos de forma secuencial, sin hacer una división del proyecto en distintas partes de forma muy clara. Si es cierto que al final cada uno de los dos miembros acabó controlando más de distintas características de angular, que a la hora de ir haciendo tareas más complejas, permitía a cada miembro poder realizar la parte de cada tarea en la que estaba más especializado, y así realizar la tarea de forma más eficaz. En las últimas semanas del proyecto, algún otro miembro del grupo de back se unía a la parte de front web para ayudar a completar alguna tarea concreta y aliviar de carga a los otros dos compañeros.

La **comunicación interna** se ha realizado en especial a través de un grupo de WhatsApp y Discord, aumentando durante la unión front-back. El primero se utilizaban más para dudas o circunstancias concretas, y el segundo cuando había problemas de mayor dificultad o bloqueos de los desarrolladores.

También se ha utilizado comunicación presencial cuando era posible, y se realizaron las dos reuniones internas para reconsiderar el calendario y el progreso de cada miembro. También se realizaron revisiones de código por pares, asegurando la calidad de este y el seguimiento de las pautas establecidas por la comunidad.

El **progreso del proyecto** se midió mediante un modelo mixto, evaluando tanto el rendimiento grupal en general ha sido una comunicación efectiva y ha permitido sinergias entre los miembros del grupo.como el seguimiento del diagrama de Gantt inicial y los esfuerzos individuales. Para ello, se realizaron seguimientos del cumplimiento de los requisitos a lo largo del tiempo, las horas individuales, la tasa de cambio de código, la satisfacción de los miembros, y aunque con más precaución y sólo como referencia, LOC y commits de GitHub, ya que pueden variar según la parte desarrollada y la frecuencia sobre la que se hace commit Estas medidas se han tenido en cuenta en especial durante las reuniones internas del 28 de marzo y 1 de mayo y autoevaluación. Se estableció como métrica de despliegue inicial el tiempo hasta el despliegue de la funcionalidad útil más sencilla, objetivo que se logró algo más tarde de la primera iteración ya que hubo un mayor desarrollo por sección individual antes de ello.

La herramienta de GitHub Issues no se ha llegado a utilizar en todo su potencial. Esto se debe a que dado el tamaño relativamente pequeño del proyecto y su comunicación activa, todos los integrantes del grupo tenían bastante claro qué trabajos debían realizar y se consideraba más como un mero trámite. A partir de la reunión de seguimiento del 8 de abril, se comenzó a utilizar más este sistema tal y como se recomendó. Aunque reconocemos el valor de esta herramienta para proyectos de mayores dimensiones, facilitando el control del progreso tanto grupal como individual, y permitiendo elegir con mayor criterio dónde dedicar esfuerzos, en nuestro caso no ha sido de excesiva utilidad. Otros factores que ayudan a esto es la integración tardía con el sistema, no tener hábito de utilizarlo y dificultad de crear issues de correctas características para el sistema.

Figura 5.1 Diagrama con las horas dedicadas al proyecto a lo largo del desarollo del SI

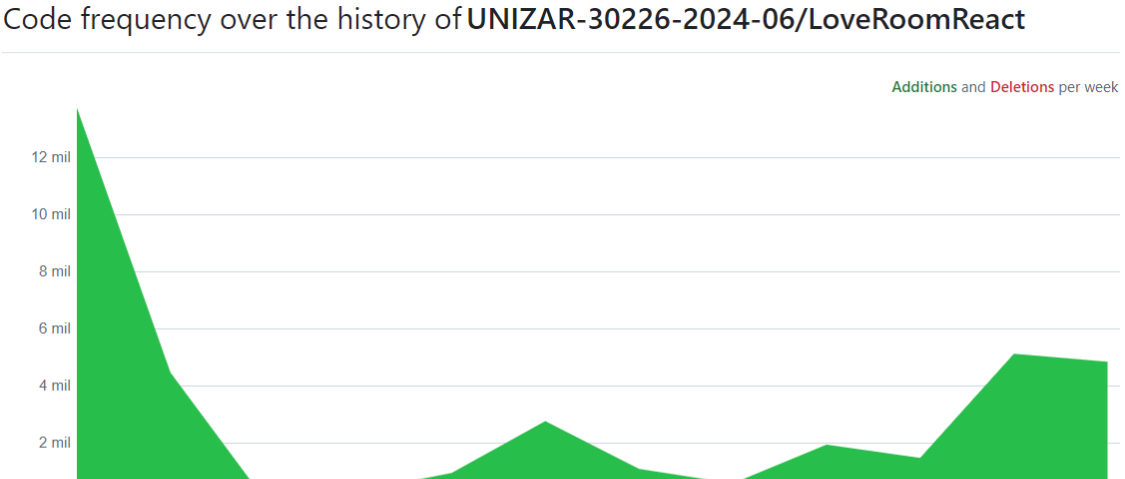


Figura 5.2 Diagrama de code churn del repositorio correspondiente al equipo de desarrollo de la aplicación

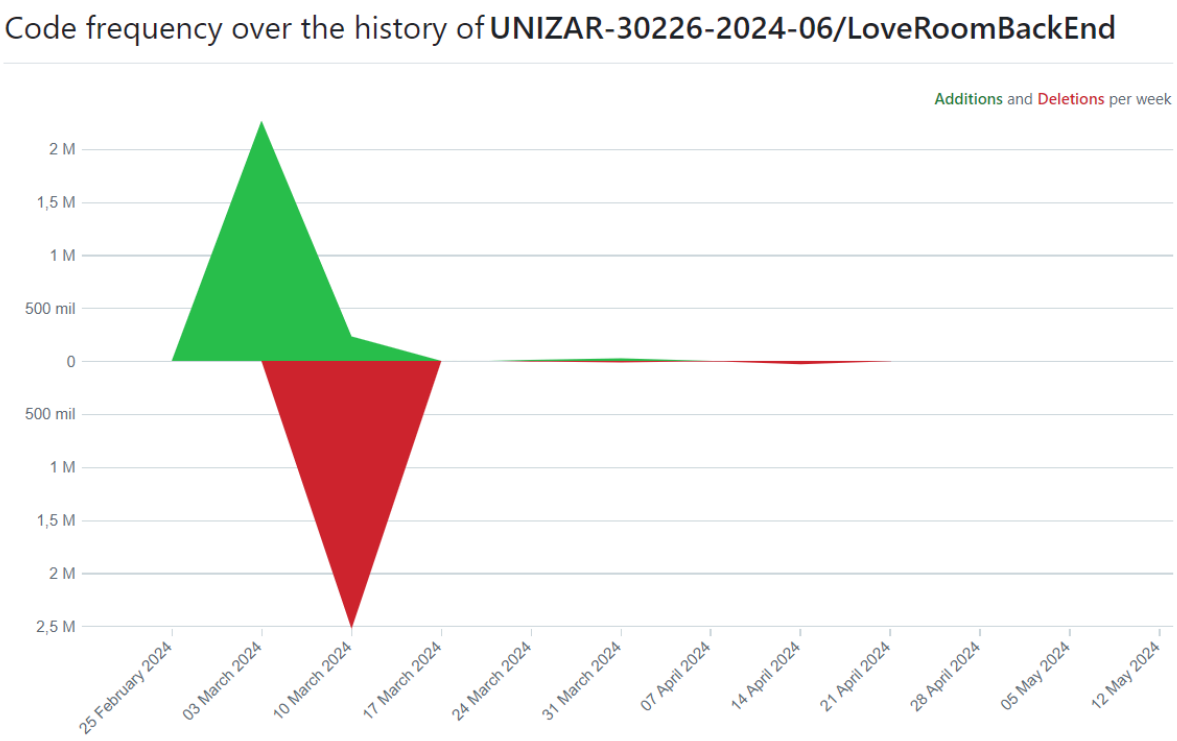


Figura 5.3 Diagrama de code churn del repositorio correspondiente al equipo de desarrollo del backend

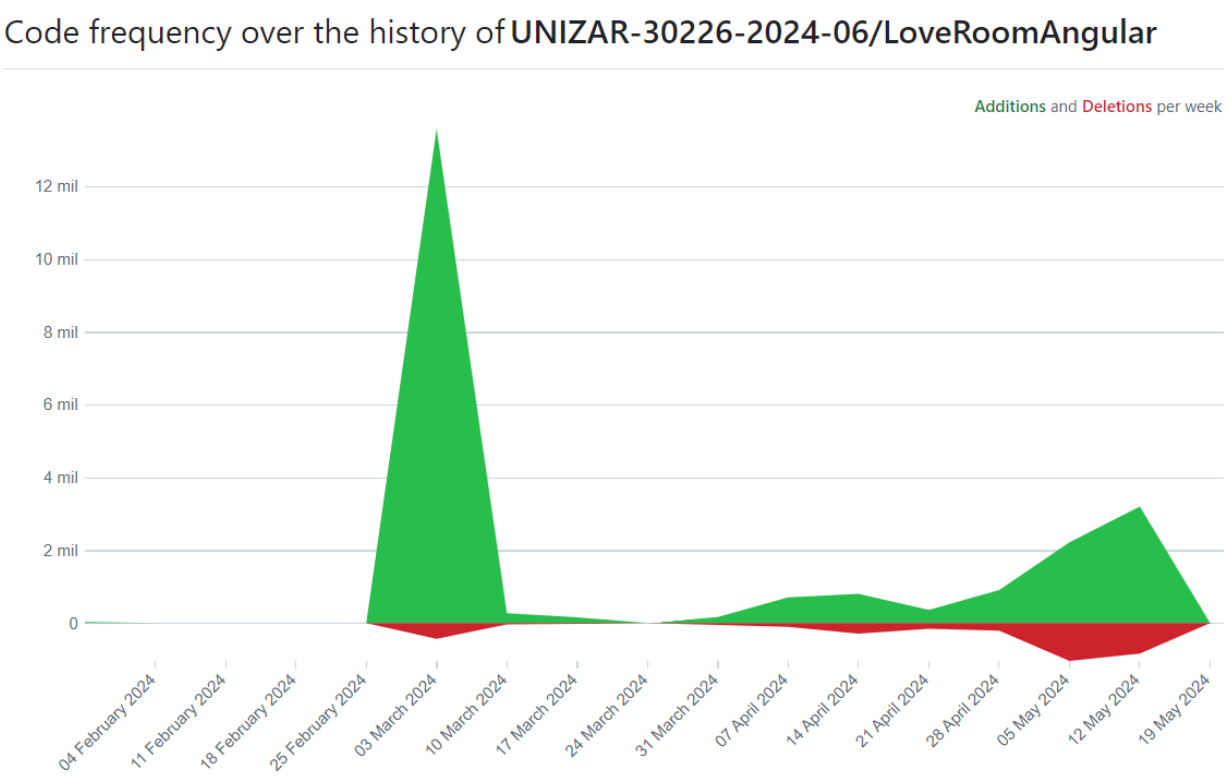


Figura 5.4 Diagrama de code churn del repositorio correspondiente al equipo de desarrollo del front web

Durante la **ejecución del proyecto**, se detectaron algunas divergencias frente al calendario inicial. Estas divergencias se debieron principalmente a imprevistos técnicos y a la variabilidad en la complejidad de ciertas tareas, en especial de la implementación para Frontend y la conexión Front-back. Para abordar estos problemas, durante las revisiones internas tanto del 28 de marzo como del 1 de mayo, además de la autoevaluación, se realizaron ajustes tanto en el trabajo como en el calendario, ajustando tiempos y tareas y priorizando las tareas críticas para el funcionamiento correcto, destacando la ayuda de los integrantes del Backend al desarrollo en Angular y en la sincronización de React.

En general se han realizado las tareas de manera más concurrente, tanto por el tipo de distribución de trabajo que hemos hecho, como debido a que la carga de trabajo del resto del curso no nos permitió dedicarle tanto esfuerzo reiterativo como se había planteado sino que fue más hacia al final donde se realizó un gran aumento de las horas, como se puede apreciar en la figura 5.1. También la tasa de cambio de código aumentó al final aunque tuvo valores generalmente constantes el resto del proyecto, más allá de los picos generados por la creación del mismo, o la inclusión inintencionada de la carpeta *node_modules* en el GitHub del backend. Esto nos indica que pese a haber intentado seguir el calendario, se tardó demasiado en implementar las funcionalidades restantes debido a diversos factores, influenciando en la facilidad del despliegue y pruebas finales.

Las **herramientas y tecnologías** empleadas fueron adecuadas en su mayoría, aunque se realizaron algunos cambios para facilitar la implementación. Aún así, han dado mayores problemas de los esperados, a los cuales se deben gran parte de los retrasos y de las horas no previstas que se han tenido que realizar. Entre ellos problemas de versiones de módulos e implementaciones de componentes en React que dificultaron la implementación de video y multimedia, el bloqueo de peticiones por cabeceras CORS o configuración de nginx, la generación con Expo de la apk, peticiones *http*, la escasa documentación sobre Angular 17. Estos cambios fueron documentados y discutidos en las reuniones de seguimiento para asegurar que todos los miembros estuvieran al tanto de las nuevas decisiones y pudieran adaptarse rápidamente.

El **control de versiones** se manejó eficazmente utilizando Github, dividido en tres repositorios de la misma organización y siguiendo en su mayoría los procedimientos de rama por componente a desarrollar y versión con correcto funcionamiento en la rama main. Se implementó un despliegue automático con Github Actions para el backend, para asegurar la **integración continua**, se creó un *workflow*, para cada commit en cualquier rama que no fuese la rama **main**, ejecutando los test unitarios correspondientes a todo el sistema. Por otro lado para asegurar el **despliegue continuo** se añadio otro workflow donde también se ejecutaban todos los test unitarios, una vez pasados estos test, se procede al despliegue en la plataforma Azure, creando la imagen del contenedor y enviándola a la máquina correspondiente al servidor de datos. Fue de gran valor dado que permitió comprobar el correcto funcionamiento antes de que pueda ser utilizado incorrectamente, y facilitaba el despliegue incluso para participantes del front ajenos al procedimiento en caso de tener que hacer pequeños cambios.

Sin embargo, se presentaron algunos desafíos durante los despliegues en React y en Angular, fallando todos los procedimientos probados. En la parte de React, existieron muchos problemas con la generación de la apk de la aplicación. Android no permite el envío de texto en plano (*http*) y se le debía especificar en el archivo *AndroidManifest.xml*. Dado que Expo no poseía este archivo, se le debía especificar desde el archivo de configuración *app.json*, pero no funcionaba, por lo que se debió exportar el proyecto y compilarlo con Android Studio para permitir esto. Como es obvio, no es el único grupo de desarrollo que ha tenido problemas con el despliegue. El no poder usar *https* ha sido un gran inconveniente para todos, pues el grupo de *Backend* quiso realizar esta tarea con *nginx,* que es un servidor web y proxy inverso de código abierto, conocido por su alto rendimiento, estabilidad y bajo consumo de recursos. La implementación con este servidor funciona correctamente, pero los certificados autofirmados no funcionaron correctamente y no nos permitieron realizar el protocolo deseado. Por último, pero no menos importante, el servidor web para realizar la web tampoco llegó a funcionar correctamente. Aunque realmente se consiguió hacer que la aplicación compilase en producción, a la de realizar el despliegue en Vercel, Github-Pages y en un contenedor en una máquina de Azure con un servidor de Nginx, ninguno de estos despliegues fue exitoso. Además debido a la falta de tiempo no se ha podido descubrir el porqué de estos fallos.

Respecto a las **pruebas del software**, no solo se ejecutaban varios tests en el despliegue automático del backend, sino que también el equipo ha trabajado siguiendo una filosofía basada en comprobar manualmente de forma exhaustiva todos los casos y ejecuciones posibles tras implementar una nueva funcionalidad, y antes de subir commits a la rama main tanto en los fronts como en el backend. Pese a realizar extensas pruebas, la funcionalidad del chat y multimedia implementada en el front de React falló en la reunión de demostración de software con un bug inesperado que no hemos conseguido replicar a posteriori, muestra de que debería haber sido aún más robusto y se debería haber hecho una prueba completa de la ejecución en el test de producto.

Cierre del proyecto

Como se ha visto tanto en el presupuesto como en el seguimiento de esfuerzos, las **estimaciones iniciales** han sido excesivamente optimistas, habiendo una diferencia de unas 180 horas, de las 846 previstas a las 1027 reales. Un factor importante es la poca experiencia de todos los integrantes del grupo, que era la primera vez que se enfrentaba a un proyecto de este calibre, con tecnologías que la mayoría de integrantes no habían utilizado, aumentando notablemente los imprevistos y errores a lo largo del proyecto. También la distribución de las horas con mayor número de estas al final debido a la carga de trabajo del resto del curso, pueden haber dificultado la solución de errores. Estos factores se habían tenido en cuenta a la hora de realizar la planificación, pero se había infravalorado su importancia.

Otra gráfica que se ha estudiado es la de horas por actividad, donde podemos observar que donde se ha necesitado más tiempo ha sido en el front, tanto para Angular como React. Esto nos ha podido enseñar que el desarrollo web y los frameworks que no se controlan pueden ser un gran inconveniente, teniendo además en cuenta que en Angular se había programado dos únicos integrantes, necesitando al final ayuda de otras secciones, ya que es donde más tiempo se ha acabado dedicando. Entre otras **lecciones aprendidas** se podría mencionar el control del tiempo, donde una reserva de tiempo real mayor para el despliegue podría haber sido más importante para llegar a cumplir todas las necesidades. Otra lección aprendida ha sido reflejada en la importancia que tiene leer la documentación de cada herramienta que se emplea para el desarrollo del proyecto, donde ésta puede llegar a ser clave para el correcto desarrollo del proyecto. En resumen, cuando se debe escoger una herramienta, los aspectos más importantes son el estudio de la documentación de ésta y la experiencia previa en herramientas parecidas.

También se ha de tener en cuenta un estudio exhaustivo de todas las herramientas disponibles. Para explicarlo con un ejemplo, en el desarrollo de React los estados globales y parciales se controlan de una manera poco intuitiva, por lo que haber usado bibliotecas como *Redux* podría haber sido una buena opción a tener en cuenta.

Otra de las lecciones aprendidas, en particular respecto al desarrollo de backend, es la importancia de anticipar y prevenir el uso malintencionado o hacking de la aplicación por parte de los usuarios finales. Se ha aprendido que es crucial que los métodos de la API comprueben que el usuario que los invoca tenga privilegios para realizar las acciones correspondientes. A lo largo del desarrollo del proyecto esto es algo que se ha implementado utilizando tokens de usuario, y se ha trabajado de manera profesional y cautelosa, asegurando que incluso usuarios malintencionados no puedan acceder a información restringida. Además, otra lección crucial aprendida en el desarrollo de backend es la importancia de una documentación detallada y efectiva, que facilite la integración frontend-backend. Durante el proyecto, se han documentado debidamente los endpoints de la API y los eventos del socket. Cabe destacar que la comunicación fluida entre los miembros del equipo de desarrollo es esencial para el éxito y la eficiencia en el desarrollo de software.

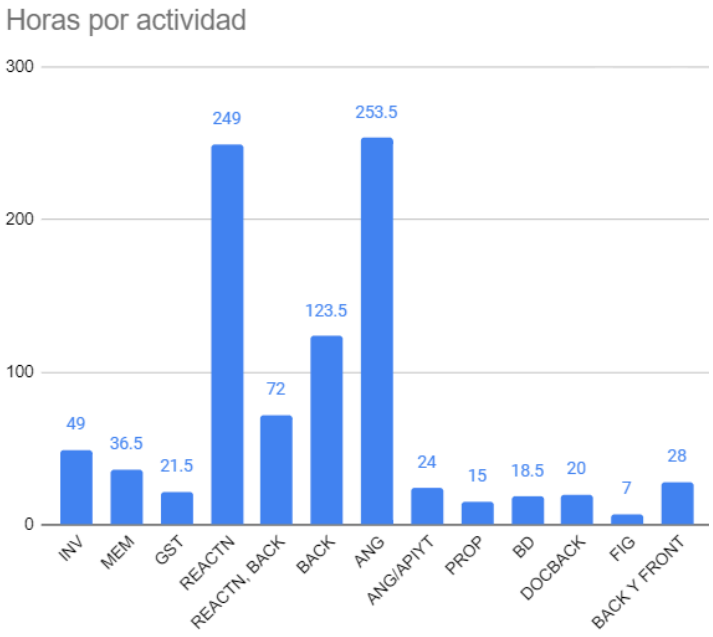
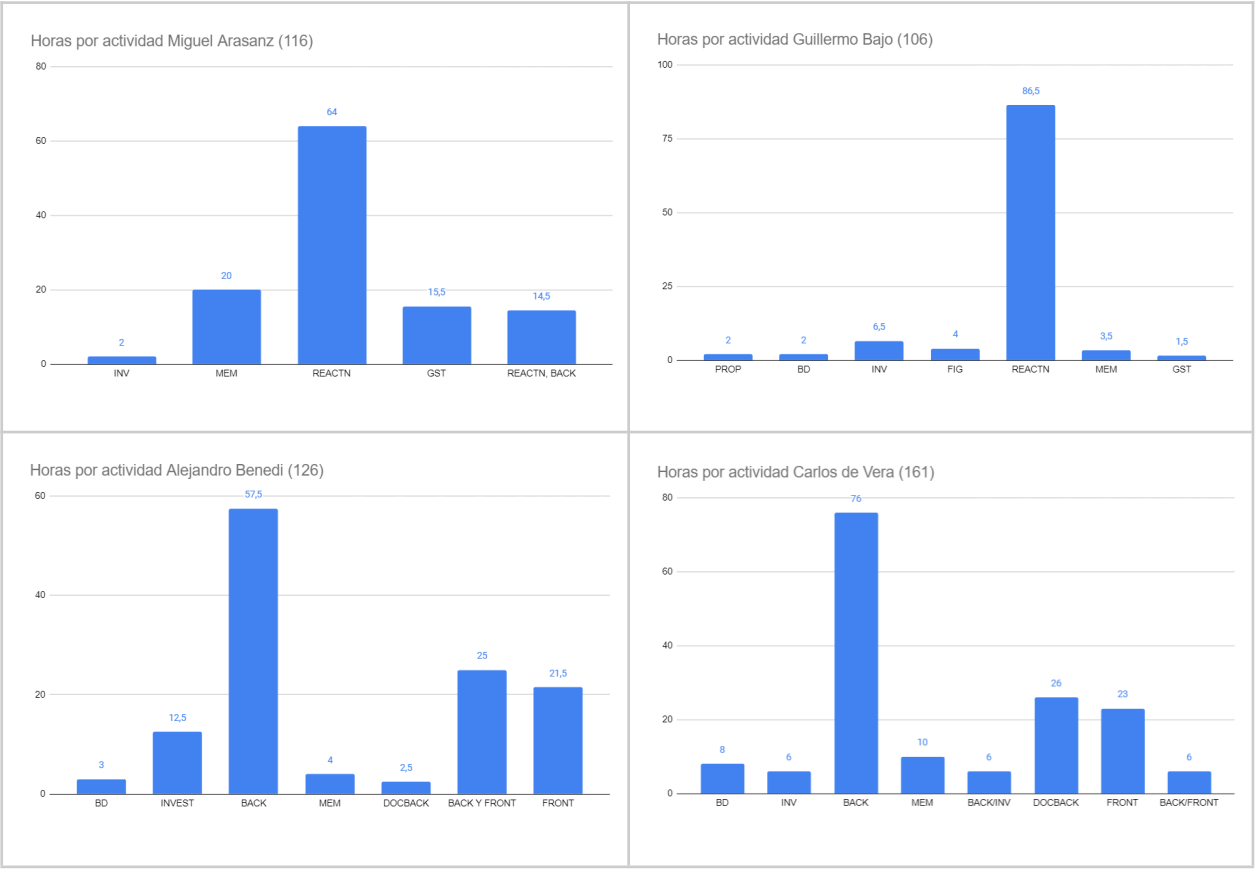


Figura 5.5 Diagrama de barras donde se muestran las horas totales por actividad



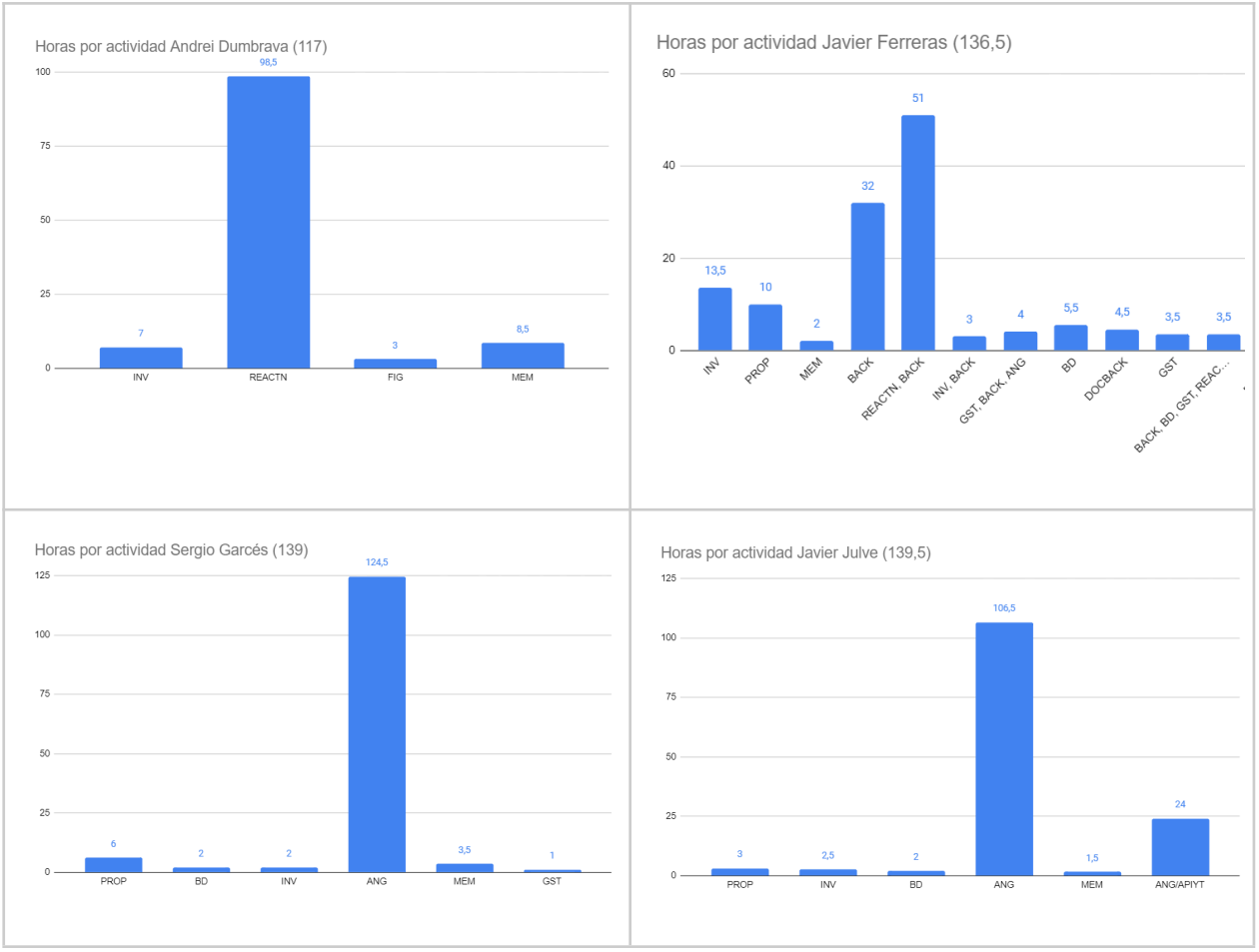


Figura 5.6 Diagramas de barras donde se muestran las horas totales por actividad y por integrante del grupo de desarollo

En cuanto a los esfuerzos realizados por cada integrante, podemos apreciar que son bastante equitativos en cuanto a número de horas, aunque ha habido algo de variabilidad en cuanto a qué se consideraba una hora trabajada por lo que no se puede tomar como valores absolutos sino como una métrica más a tener en cuenta.

En el front React, aparte de las horas conjuntas de ayuda mutua, en especial en el perfil pero en todos las secciones nombradas, cada uno ha tomado mayor carga de trabajo en una sección. Andrei Dumbrava se ha encargado de la búsqueda de videos de Youtube, su sincronización con el back, salas de video, chat y despliegue. Guillermo Bajo ha realizado la UI de administrador y reportes, estadísticas y premium. Miguel Arasanz ha dedicado sus esfuerzos en especial al multimedia y su conexión con el back, aunque dándole mayor importancia a la gestión en la memoria, presupuesto, Gantt y seguimiento semanal.

En el backend, aparte de algunas horas de reuniones conjuntas, cada integrante se ha ceñido principalmente a tareas concretas. A Carlos de Vera se le asignó la tarea de salas, sincronización, multimedia y perfil administrador. Alejandro Benedi se centró principalmente en todo lo relacionado a los usuarios (creación, modificación, autenticación, verificación, cuentas…). Por último Javier Ferreras se encargó también de las tareas de salas y sincronización, además de matches, reportes, y de controlar de una forma global el desarrollo del backend.

Una vez el backend ha sido desplegado correctamente Carlos de Vera fue asignado al grupo de desarrollo del front web como consultor debido a que también se le asignó la tarea de solucionar los posibles errores que se encontrasen en producción en el servidor de datos. Además tras terminar sus respectivas tareas, Alejandro Benedi y Javier Ferreras también fueron asignados a otros grupos de

desarrollo. Alejandro fue asignado al grupo de desarrollo del front web y Javier fue asignado al grupo de desarrollo del front React.

En el front web, la realización del proyecto ha sido de forma conjunta continuamente. Al comienzo del proyecto Sergio Garcés se especializó más en el diseño html y css de las distintas pantallas, mientras Javier Julve empezaba a realizar las conexiones con el backend de las primeras funcionalidades, y la configuración de las rutas de la web. Una vez realizada esta primera parte base, ambos miembros empezamos a ir uniendo todas las funcionalidades con el backend y comprobando su correcto funcionamiento. En esa segunda fase se fue avanzando de forma continua sin demasiados problemas. La tercera y última fase fue la conexión mediante sockets para el correcto funcionamiento de todos los requisitos de la sala. En esta segunda y última fase se acabó uniendo Alejandro Benedi para acabar algún requisito concreto fuera de los sockets y asi llegar a tiempo con la funcionalidad completada.

También se han estudiado los resultados obtenidos con la herramienta git fame. Esta nos permitía realizar un estudio de las líneas de código vivas por cada usuario. Aún así, esta métrica no se ha de tener tanto en cuenta, pues no refleja el trabajo de cada integrante y además incluye líneas de código como el *package.json, package-lock.json, …* Es decir, hay números que pueden no reflejar con exactitud la realidad.

Estos son los resultados realizados por el grupo de aplicación móvil:

```
Total commits: 179
Total ctimes: 4239
Total files: 127
Total loc: 34137
| Author              |  loc   | coms|  fils  | distribution      |
|:---------------------|--------:|-------:|-------:|:--------------------|
| Andrei Dumbrava| 14284 |    54 |    48 | 41.8/30.2/37.8  |
| Miguel Arasanz   | 14114 |    28 |    24 | 41.3/15.6/18.9  |
| Guillermo Bajo    |  5166  |    63 |    48 | 15.1/35.2/37.8  |
| Javier Ferreras    |  572   |    29 |     6 | 1.7/16.2/ 4.7     |
```

Figura 5.8 Resultados de git frame sobre el repositorio LoveRooomReact

Por otro lado en el Backend se obtienen los siguientes resultados:

```
Total commits: 377
Total ctimes: 2263
Total files: 86
Total loc: 12733
| Author              |  loc |  coms |  fils |  distribution   |
|:---------------------|------:|-------:|-------:|:----------------|
| Carlos de Vera      5821 |   152 |    32 | 45.7/40.3/37.2  |
| Alejandro Benedi |  4247 |   116 |    19 | 33.4/30.8/22.1  |
| Javier Ferreras     |  1582 |    76 |    21 | 12.4/20.2/24.4  |
| Miguel Arasanz     |   621 |    18 |     7 | 4.9/ 4.8/ 8.1   |
| Andrei Dumbrava |    3 |     6 |     1 | 0.0/ 1.6/ 1.2   |
```

Figura 5.8 Resultados de git frame sobre el repositorio LoveRooomBackEnd

Por ultimo en el front:

```
Total commits: 136
Total ctimes: 1350
Total files: 191
```

Total loc: 63318

Author	loc	coms	fils	distribution
Sergio Garces	39218	42	81	61.9/30.9/42.4
Javier Julve	22381	27	59	35.3/19.9/30.9
Alex Benedi	1183	32	30	1.9/23.5/15.7
Javier Ferreras	157	3	1	0.2/ 2.2/ 0.5
Carlos de Vera	90	15	3	0.1/11.0/ 1.6

Figura 5.9 Resultados de git frame sobre el repositorio front

6. Conclusiones

Realizar un proyecto muy extenso con un gran grupo de personas dedicándose a realizar lo mismo e intentar coordinarse a la perfección es una tarea muy complicada y que requiere mucho tiempo. Existe la falsa concepción de que los informáticos, se dedican únicamente a escribir código, pero la realidad es muy distinta. Una gran parte del tiempo se dedica a planificar las tareas que se han de realizar, coordinarse con el equipo, comparar resultados parciales… Este proyecto nos ha podido hacer ver que no es menos importante la gestión de un proyecto que la realización de éste mismo, pues una buena gestión nos facilita la otra.

Un aspecto crucial que hemos identificado es la importancia de utilizar versiones completamente estables de los frameworks y librerías en lugar de las versiones más recientes o experimentales, como es el caso de Angular 17, la escasa documentación al final ha derivado en retraso en el desarrollo. Esto habría reducido significativamente los problemas de compatibilidad y errores inesperados en el desarrollo.

La documentación del backend es una parte fundamental del proyecto. En retrospectiva, habría sido más eficiente utilizar herramientas como Swagger para generar la documentación automáticamente, en vez de realizar dicha documentación a mano. Esto no solo habría ahorrado tiempo, sino que también habría garantizado una documentación más precisa y fácil de mantener.

Tener una buena comunicación entre todos los integrantes es clave, pues como se puede ver han habido pequeñas diferencias entre la aplicación para móvil y para web. Obviamente este aspecto está altamente correlacionado con el tiempo en el que se han realizado el despliegue y el final del proyecto, donde se ha ido un poco a contrarreloj para asegurar el cumplimiento de todos los requisitos.

La gestión del proyecto sobre el papel no era incorrecta, se desempeñó una parte del tiempo en realizar esta tarea antes de empezar a desarrollar y escribir código sin sentido, por lo que probablemente esto se llevaría a otros proyectos parecidos, pero se cambiaría la realización de éste, utilizando por ejemplo diagramas de Gantt no tan secuenciales y adaptados mejor al reparto de tareas final. Aún así aunque es cierto que un factor muy importante es la dedicación que los integrantes han tenido que emplear para el resto de asignaturas, además de los distintos proyectos personales de cada uno.

Aunque la organización del equipo fue adecuada, nos desviamos en varias ocasiones de las tareas planificadas. En futuros proyectos, sería beneficioso implementar un seguimiento de tareas más riguroso para asegurar que todos los miembros del equipo se mantengan enfocados en los objetivos prioritarios. Además, se ha aprendido que la ejecución real del proyecto siempre es distinta de la planificación, y es extremadamente importante planificar dejando cierto margen de tiempo para tener en cuenta retrasos y solucionar problemas imprevistos, además que una distribución de las horas donde más parte del esfuerzo se hubiera dedicado al principio hubiera permitido un mejor desarrollo.

Para concluir, este proyecto nos ha demostrado la importancia de una gestión eficaz y una comunicación clara en el desarrollo de software en equipo. La experiencia también subrayó la necesidad de un seguimiento riguroso de tareas. A pesar de que no se consiguió llegar a todos los objetivos establecidos, hemos identificado áreas clave de mejora y estamos preparados para abordar futuros proyectos con una metodología más ágil y colaborativa.

Anexo I. Presupuesto

Para calcular los **costes directos** se ha hecho una estimación con las tareas a realizar y sus duraciones estimadas en días. Estas son las mismas que las definidas en el diagrama de Gantt pero sin la parte de gestión y aseguramiento de la calidad, los cuales se han tomado como valores macro. De las duraciones estimadas en días se ha tomado una media de 2 horas por día en cada tarea, tomando en frontend 4 horas porque hay dos secciones encargadas en ello. El margen de error utilizado es de un 20% tanto para el mínimo como el máximo. También se ha tenido en cuenta como macro la formación específica para el proyecto, aparte de la formación continua, y donde suponemos un bajo porcentaje de dedicación sobre desarrollo ya que aunque son tecnologías novedosas para el personal, el no estar familiarizado con el entorno se ha tenido en cuenta en la estimación de esfuerzos.

Se han mantenido las horas de convenio pero el sueldo bruto se ha tomado un tanto menor en 20.000,00 EUR anuales ya que el personal del proyecto tiene poca experiencia en el sector, conllevando un sueldo menor. Otros costes directos tenidos en cuenta son los viajes para contactar con contratistas. Existe la posibilidad de hacerlo online o de necesitar hasta 5 viajes, los cuales suponemos nacionales (Sobre 100 euros). Las licencias usadas en el proyecto de React, Angular, Azure o VSCode son todas de tipo MIT por lo que no sería necesario pagar. El coste de los servidores cloud se muetra sólo para el proceso de desarrollo, siendo tanto Azure como CleverCloud posible de forma gratuita pero se toma un coste de 3 EUR para Azure y 8 para CleverCloud para poder mantener un mayor número de conexiones y el servidor disponible en todo momento. La amortización de los equipos se ha mantenido a 250 EUR anuales a 4 años.

En cuanto a los **costes indirectos** se han mantenido las estimaciones dadas para una oficina para un equipo de nuestro tamaño, los proyectos no exitosos, formación continua y asesores. Se han añadido estimaciones para el coste de las Licencias Google Workspace, siendo la Licencia Business Standard 11,50 EUR por usuario al mes y Github Enterprise de 19,50 EUR por usuario al mes, siendo el resto de licencias de uso libre. También se ha tomado un seguro de interrupción de negocio para el caso de un incendio u otro evento que interrumpa el funcionamiento normal, siendo entre 500 y 3,000 euros por año y tomando el límite menor dada la posibilidad de trabajo online; un seguro de responsabilidad civil profesional siendo la prima media de los seguros de RC Profesional de profesiones técnicas de 1.414 EUR para hacer frente a los daños personales y materiales que por error, omisión o negligencia se cometan y perjudiquen a terceras personas, como clientes o proveedores; y un seguro de propiedad intelectual para que se respete la licencia tomada Attribution-NonCommercial-NoDerivatives 4.0 International y proteger ante una posible infracción a otra empresa.

Además, se ha tenido en cuenta los costes financieros suponiendo necesario para iniciar las operaciones un préstamo de 50.000 EUR a un 2% TAE. Por último, aunque no serían necesarios gastos en marketing ya que no es una empresa dedicada al comercio como tal, para encontrar clientes y hacer la función de relaciones públicas se ha tenido en cuenta a un miembro del equipo que dedique sobre unos 20 días anuales para contactar con empresas, acudir a ferias, etc. y ejercer de relaciones públicas.

Por tanto, nuestro proyecto tendría probablemente unos costes totales sobre 18.000 EUR, variando desde 14.000 a 22.000. Tomando un 8% de beneficio bruto esperado para la empresa con el proyecto, aumenta en unos 1.500 el coste de contratación. aplicando el IVA para el cliente, se quedaría en unos 23.500 EUR de coste total más probable. Por tanto decidimos redondear a **24.000 EUR** para la factura del cliente, con iva incluido.

En la **finalización del proyecto** se ha evaluado el presupuesto teniendo en cuenta el número de horas reales, siendo estas 1027. Este número se encuentra ligeramente fuera del rango previsto, 15 horas por encima del máximo y más de 180 horas por encima de lo más probable. Teniendo en cuenta el resto de

gastos indirectos y costes directos de licencias, equipos y viajes; esto supone unos gastos de **21.783,98 EUR** frente a los 18.000 EUR previstos. Este aumento de más de 3.500 euros ha igualado y superado al margen de beneficio bruto de 1.438, haciendo que el resultado para la empresa sea negativo, con 2.344,98 EUR de pérdidas. El aumento de horas frente a las previstas para el frontend, tanto en React como en Angular, ha supuesto la mayor parte del error, aunque el coste en desarrollo de backend haya compensado en parte estos errores de predicción. Como se ha explicado anteriormente, se debería tener más en cuenta el factor de tecnologías no utilizadas y errores imprevistos para futuros presupuestos, para permitir el desarrollo conforme a lo planificado y dentro del presupuesto.

Anexo II. Resultados de la revisión intermedia

Anexo III. Recursos

- **Backend**

https://socket.io -> Chat y comunicacion sincrona
https://github.com/expressjs/multer/blob/master/doc/README-es.md -> Gestion y subida de archivos
https://expressjs.com/es/guide -> Documentacion oficial de express
https://nodejs.org/en/about -> Documentacion oficial de node
https://www.youtube.com/watch?v=XfX2Ap30pwU -> Streaming video with Node.js and React pero no usa socket.io, reproducción de videos desde el server a los clientes sin tener en cuenta la sincronización
https://github.com/ChrisArasin/socket-sync-video -> Repositorio de hace 5 años sobre sincronizacion de video en diferentes dispositivos
https://github.com/ShakedAp/synchronized-video-streaming -> Este parece mejor
https://www.nodebeginner.org/index-es.html -> Cursos de node
https://www.nodemailer.com/usage/ -> Enviar mails
https://stackoverflow.com/questions/26717013/how-to-edit-nginx-conf-to-increase-file-size-upload -> Nginx change Max-body-size

- **Frontend React**

https://www.npmjs.com/package/react-youtube -> Paquete para facilitar IFrame
https://reactnative.dev/docs/getting-started -> Documentación oficial de React
https://docs.expo.dev/versions/latest -> Referencia de Expo (ImagePicker, FileSystem, etc.)
https://stackoverflow.com/questions/62945907/how-to-solve-typeerror-network-request-failed-in-react-native -> Network Request Failed soluciones
https://es.react.dev/learn/referencing-values-with-refs#how-does-use-ref-work-inside -> Mejor uso de referencias
https://www.npmjs.com/package/react-native-video -> Reproductor de Video
https://stackoverflow.com/questions/73545401/how-to-fetch-image-via-get-request-in-react-native -> Imágenes a través de GET
https://stackoverflow.com/questions/60018247/react-native-video-cannot-read-property-constants-of-null -> Solventar errores de versiones en videos

- **Frontend Angular**

https://socket.io/docs/v4/client-socket-instance/ -> Documentación oficial socket io

https://socket.io/docs/v3/handling-cors/ -> Documentación oficial socket io pero en concreto el manejo de CORS
https://medium.com/@shubhsalunkhe4199/guide-to-implement-web-sockets-in-angular-d8ce2b01abd4 -> Guía para la creacion y conexión de sockets
https://github.com/Humam-Albitar/Books-Data-Live-Chart -> Proyecto en angular 17 que hace uso de sockets
https://github.com/angular/components/blob/main/src/youtube-player/youtube-player.ts -> Documentación de componente de angular youtube-player
https://angular.io/docs -> Documentación oficial de angular 17
https://www.youtube.com/watch?v=f7unUpshmpA -> Tutorial para iniciar con Angular 17
https://www.youtube.com/watch?v=n7OKfVwClE4 -> Tutorial uso de sockets en Angular
https://www.youtube.com/watch?v=MKaCXBnVxeg -> Tutorial uso de sockets en Angular