Plan de gestión, análisis, diseño y memoria del proyecto

SpreadYourMusic$_{Inc}$

ÍNDICE

1. Introducción

➢ Resumen

La aplicación a desarrollar consistirá en un reproductor de música en streaming inspirado en Soundcloud y Spotify.

Es una aplicación orientada a todo tipo de usuario, tanto a los músicos que están empezando su carrera en el mundo de la música como a cualquier persona aficionada a ella. Nuestra aplicación permitirá a los usuarios subir canciones, crear listas de reproducción o escuchar canciones utilizando un reproductor propio entre otras funcionalidades. La versión Android permitirá a los usuarios descargar las canciones para poder escucharlas sin necesidad de estar conectado a Internet. También incluirá características sociales, permitiendo a los usuarios seguir a sus artistas favoritos para ver las novedades que publican o subscribirse a listas de reproducción creadas por otros usuarios para enterarse de cambios en esta. El sistema también poseerá integración con redes sociales, así como la posibilidad de autentificación mediante cuenta de Google.

El usuario tendrá recomendaciones personalizadas para el usuario tanto basadas en su historial de reproducción como en geolocalización y facilitará al usuario encontrar canciones, pudiendo buscar canciones por categorías, autor, nombre, así como también mostrando canciones populares dentro de la aplicación.

Nuestra aplicación tendrá soporte web (en navegadores Chrome y Firefox) y como aplicación Android, de manera que el usuario pueda usarla en web, en el móvil o en ambas ya que incorpora un sistema de sincronización de manera que el usuario puede seguir escuchando en cualquier dispositivo la misma canción justa en el momento en el que la dejó.

2. Organización del proyecto

➢ Equipo

El equipo está formado por 7 estudiantes de Ingeniería Informática. Estos poseen conocimientos sobre programación del frontend y backend de una aplicación Web. También se tiene experiencia en el diseño de base de datos así como con la programación sobre la plataforma Android.

Se han desarrollado varias aplicaciones Android entre las que destacan una aplicación para la gestión de notas y otra enfocada al sector veterinario. También se han desarrollado diversos sistemas de información web como son un portal de venta de libros. Por otro lado también posee experiencia en la programación 3d con la creación de varias modificaciones sobre videojuegos. Por último el equipo ha participado en concursos como Google HashCode y uCode.

Por otro lado los integrantes del grupo ya poseen experiencia trabajando juntos. Para desarrollar esta aplicación se han creado 3 grupos dentro del equipo según la experiencia de los integrantes en las áreas de backend, interfaz Web(frontend) y desarrollo de la aplicación Android. Para gestionar el proyecto se ha designado un coordinador general así como coordinadores específicos dentro de cada área.

3. Procesos

➢ Procesos de inicio de proyecto

La aplicación móvil funcionará en dispositivos con Android 5.0 como mínimo, la aplicación web funcionará en los navegadores Firefox y Chrome en su última versión disponible para Windows y Mac. Para realizar las pruebas de la aplicación móvil se va a usar un dispositivo físico con Android5.0. Por otro lado, para las pruebas de la versión web se va a usar un ordenador con el navegador instalado y actualizado (Chrome y Firefox). El sistema inicialmente funcionará sobre un clúster, que contará con un almacenamiento bruto de 20GB. Se ha estimado este tamaño para el clúster ya que presenta un espacio aceptable para almacenar canciones de prueba en esta arquitectura. Este espacio se usará como base en las pruebas iniciales ya que cuando el software se entregue al cliente este podrá poner el tamaño que desee en su clúster puesto que el sistema es fácilmente escalable.

➢ Procesos de ejecución y control del proyecto

Las comunicaciones del grupo se van a realizar mediante un grupo en la aplicación de mensajería instantánea WhatsApp para tratar temas y comunicaciones poco importantes y eventuales. Sin embargo, para temas que deban ser permanentes y/o deba quedar constancia de esta comunicación, se usarán las Issues o incidencias de la plataforma de alojamiento de los proyectos GitHub. En esta plataforma de alojamiento se almacenará el código fuente del sistema desarrollado y todos los documentos generados durante el desarrollo. Entre estos documentos se encuentra, por ejemplo, las actas de las reuniones con los clientes, que serán redactadas por al menos un miembro del equipo durante dicha reunión. De forma similar se registrarán las reuniones del equipo, los contenidos y las decisiones que puedan tomarse en esas reuniones.

En todas las actas, tanto de reuniones del equipo como reuniones con los clientes se incluirá al menos la fecha y hora de la reunión, la duración de la reunión, los miembros presentes en la reunión y los temas y decisiones que se tomen en la reunión.

Todas las semanas tienen un conjunto de tareas asociadas. Al final de cada semana, el responsable de cada proyecto revisa las tareas que se han realizado esa semana y se realiza una reunión conjunta de todos los miembros del equipo a través de la plataforma Skype, si hay tareas que no se han cumplido, se asigna automáticamente para la siguiente semana, siendo estas tareas las primeras que se deberán hacer. Además, cada semana el responsable del sub-proyecto revisará cuantas tareas se han realizado para la siguiente iteración como métrica de monitorización de la desviación según el plan original. Tras esta revisión el responsable del sub-proyecto asignará las tareas de la semana a todos los miembros del equipo que puedan trabajar esa semana. Durante el desarrollo del proyecto puede haber problemas y disputas entre los miembros del equipo. Para tratar

de resolverlos el responsable del sub-proyecto será el primero en mediar entre los miembros en disputa y, si hay alguna razón que haga imposible esta mediación será el resto del equipo quien deberá mediar.

➢ Procesos técnicos

Las herramientas utilizadas tanto para desarrollo del software (construcción, pruebas y despliegue) serán IntelliJ Idea y Android Studio debido a que ambas dan soporte para los lenguajes de programación Java y Kotlin además de su buena integración con Git, herramienta que se utilizará para el control de versiones. En el caso especial del Frontend se usará como herramienta WebStorm por razones similares a los anteriores.

Para asegurar la calidad del software, al mismo tiempo que se vaya desarrollando se irán creando pruebas unitarias sobre el mismo con JUnit.

Siguiendo una metodología de Diseño incremental, cada dos semanas se tiene planificada la entrega interna de una versión nueva del software. Estas versiones se basan en la versión anterior añadiéndole ciertas funcionalidades y no serán lanzadas al público.

➢ Planes

▪ Plan de gestión de configuraciones

El código desarrollado deberá seguir una serie de estándares, siendo los siguientes los aceptados. Para el nombrado de métodos, variables, clases... se utilizará la filosofía CamelCase en los proyectos de Android y Backend y la filosofía SnakeCase en el caso del Frontend. Además, para la escritura de código se seguirá el estándar recomendado por Google en el caso de Android y Backend y el recomendado por la W3Schools para Frontend. Como medida adicional, para permitir que un mayor número de dispositivos puedan visualizar correctamente la página web, se va a desarrollar la web de forma Responsive.

Para asegurar que el desarrollo sigue los estándares de calidad y de nombrado, el coordinador de cada proyecto se encargará de la revisión de los commits de su proyecto y la aceptación o no aceptación de código nuevo. Del despliegue y la puesta en marcha del sistema se encargará Jorge Pinilla y de la correcta administración del sistema de control de versiones (VCS) Git Á ngel Cañal.

Este Sistema de Control de Versiones está compuesto por 4 repositorios, uno en el que se alojan todos los documentos del sistema (Ejmeplo: ”Propuesta Económica” o ”Plan de Gestión”) y 3 repositorios más, uno por cada uno de los proyectos (Android, Backend y Frontend). Para intentar reducir conflictos o problemas, sólo los miembros más experimentados tendrán acceso total a los repositorios (estos son Ángel Cañal y Abel Chils). El resto del equipo sólo tendrá acceso al repositorio de Documentos y al del proyecto en el que esté trabajando.

Para reducir problemas causados por modificación concurrente del mismo fichero de código fuente, se va a usar un flujo de trabajo en Git denominado Git Flow por el que hay dos ramas principales, una para la última versión estable (master) y otra para la versión en desarrollo (develop). Además, para cualquier nueva característica a añadir habrá que crear una nueva rama en Git y, cuando se haya terminado, se volcará esa rama a la rama de desarrollo. No se permite subir código directamente a las ramas develop ni master. Como este flujo de trabajo puede resultar complejo, se simplifica haciendo uso del comando ”git flow”.

Como el sistema Git soporta la gestión de incidencias, se va a aprovechar este sistema para gestionar todas las situaciones inesperadas que pudieran suceder. Además, con la gestión de incidencias se va a utilizar una incidencia especial, llamada ”tarjeta” por la que se va a poder gestionar las tareas pendientes por hacer. Para ello, el responsable de cada equipo creará una incidencia por tarea a realizar (lo más pequeña posible). Estas incidencias podrán estar en 3 estados, ”Abierto”, ”En progreso” y ”Terminado” además de estar asignado a un punto en el tiempo en el que deberán estar terminadas por completo. Para pasar de ”En progreso” a ”Terminado” el coordinador del equipo debe dar el visto bueno e incorporar el nuevo código al resto (rama develop).

▪ Plan de construcción y despliegue del software

Para la compilación del backend utilizamos Maven que facilita las dependencias entre los paquetes y garantiza que todos compilen con las mismas dependencias.

Para la compilación de Android se utiliza Grandle de una manera similar a Maven.

Para la compilación de la Web se realiza una compilación manual de Jekyll.

Para realizar los test, se realizan de manera manual, primero en los ordenadores personales de cada integrante y luego integrandolo con el servidor de producción.

Los modulos son independientes entre ellos y se comunican principalmente mediante peticiones HTTP, esto permite simular las peticiones antes de probarlo con el entorno real de producción.

No existe una construcción automática del software, sino que se realiza una construcción cuando se ha terminado alguna feature. Se van realizando compilaciones locales y unicamente cuando hay una release se realiza una compilación en el servidor.

▪ Despliegue de software

Debido a presupuesto hemos simulado todo el entorno de pruebas en un solo servidor mediante máquinas virtuales y virtualización anidada.

El servidor host únicamente es el encargado de redirigir las peticiones mediante un HAProxy a modo de router.

Se utilizan 4 máquinas virtuales:

3 máquinas virtuales constituyen el servicio de almacenamiento, encargado de servir los datos, almacenar la información y garantizar la consistencia y seguridad de los datos.

Se utiliza ceph (https://ceph.com/) como sistema de almacenamiento distribuido definido por software sobre debian como sistema operativo, para desplegarlo se utiliza la herramienta semi-automática de ceph-deploy.

Se utiliza RadosGateway (http://docs.ceph.com/docs/master/radosgw/) como interfaz para exportar los datos mediante HTTP, de una manera similar a Amazon S3, las canciones se almacenan en Buckets y disponen de un URL único.

El cuarto servidor contiene Proxmox (https://www.proxmox.com/en/) y kvm. Se encarga de virtualizar tanto la base de datos como el backend y su integración con ceph permite una gestión unificada del servicio de almacenamiento.

La idea de utilizar máquinas virtuales es la gran escalabilidad que permite y la posibilidad de configurar resistencia a fallos automática.

La separación entre backend y el servicio de almacenamiento permite redirigir las peticiones sin sobrecargar ninguno de los dos servicios, un solo servidor de backend atenderá a pequeñas peticiones de consultas mientras que los 3 servidores de almacenamiento atienden a las peticiones de datos de canciones.

Puertos:

ceph01 192.168.200.10 2000 RGW(HTTP) 7480

ceph02 192.168.200.11 2001 RGW(HTTP) 7480

ceph03 192.168.200.12 2002 RGW(HTTP) 7480

proxmox 192.168.200.20 2005 WEB 8006

backend 192.168.200.16 2006 HTTP 7800 HTTPS 7880

postgres 192.168.200.17 2007 POSTGRES 7850

➢ Arquitectura

▪ Arquitectura de la aplicación Android

La aplicación Android está estructurada en 3 capas (siguiendo una arquitectura modelo-vista-controlador).

En la primera de estas se encuentran las APIs con servicios externos (backend, google, facebook y twitter) y una clase que se encargará de mantener la persistencia de la sesión (accediendo a ella podríamos obtener información del usuario que está actualmente en sesión). Esta capa permite no depender del desarrollo del backend, creando una interfaz entre ambos que permiten probar funcionalidades no implementadas en el (simulando la respuesta que ofrece el backend).

Existe una segunda capa en la cual se encuentran funciones para manipular la capa citada anteriormente (controlador). En esta tenemos las auth-functions las cuales se encargan de manipular todo lo que tiene que ver con la identificación del usuario (estas funciones son llamadas cuando el usuario inicial sesión). Por otro lado tenemos las player-functions que idealmente sería el punto de acceso al controlador musical, en cambio como se explicará posteriormente actualmente no está operativa. También se poseen las song-request-functions que son las funciones encargadas de solicitar al backend canciones, listas de reproducción (incluyendo listas de recomendados y novedades) e información de usuarios. Por último, tenemos las innner-social-functions que se encargan de las funciones sociales internas de la aplicación (como seguir a un usuario o dar me gusta a una canción) y las extern-social-functions que se encargan del manejo de redes sociales.

Por otro lado nos encontrados la capa de vista, que se encarga del manejo y dibujado de la interfaz gráfica. Esta capa se divide a su vez en dos. Una de ellas se encarga del manejo y dibujado de la interfaz en las pantallas y la otra se encarga de la persistencia de los datos que se están mostrando. Esta segunda capa es requerida para mejorar el rendimiento de la aplicación evitando rehacer determinadas operaciones.

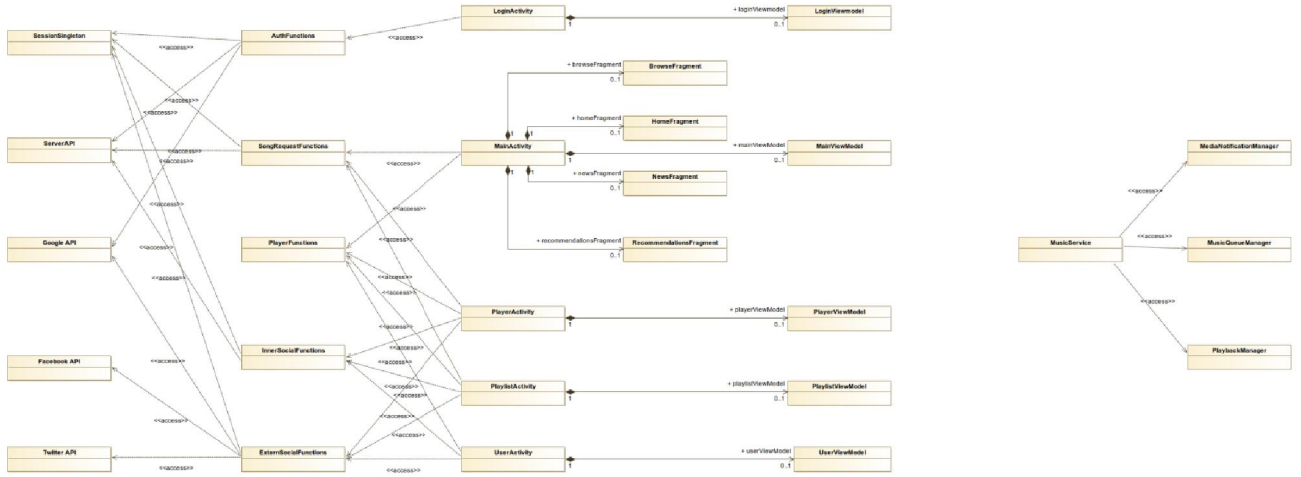
Por último, nos encontramos el controlador musical. Esta capa se encarga de interactuar con el sistema Android para reproducir la música. Se englobaría idealmente en la misma capa en la que se encuentran los servicios externos (comunicación exterior) ya que su funcionamiento es similar y se controlaría mediante el controlador (concretamente mediante las player-functions), pero debido a peculiaridades del diseño en Android y la implementación actual de esta capa, que está muy ligada a la interfaz (en el esquema no se han representado las líneas de comunicación para mejorar su comprensión) hacer esto complicaría la estructura del código a bajo nivel. Aun así esta está sufriendo una reestructuración para hacer posible su encapsulación tras las player-functions.

- ## Versionado de la aplicación Android

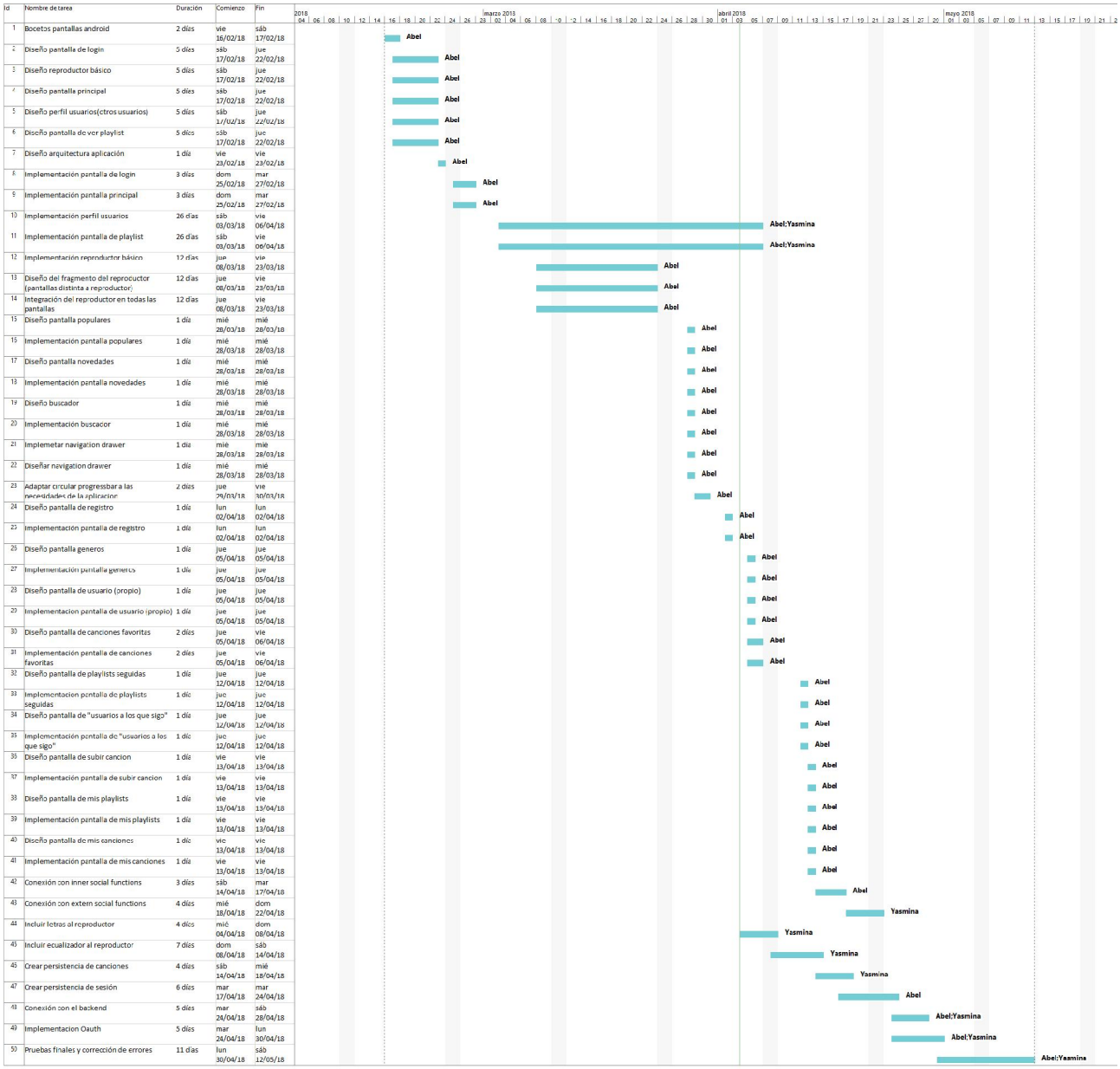
La versión 1, primera iteración, es la versión correspondiente a la fecha 31-03 del diagrama de Gantt.

La versión 2, segunda iteración, es la versión correspondiente a la fecha 12-05 del diagrama de Gantt.

- **Diagrama de clases de la aplicación Android**



- **Diagrama de Gantt de la aplicación Android**



Id	Nombre de tarea	Duración	Comienzo	Fin
1	Bocetos pantallas android	2 días	vie 16/02/18	sáb 17/02/18
2	Diseño pantalla de login	5 días	sáb 17/02/18	jue 22/02/18
3	Diseño reproductor básico	5 días	sáb 17/02/18	jue 22/02/18
4	Diseño pantalla principal	5 días	sáb 17/02/18	jue 22/02/18
5	Diseño perfil usuarios(ctros usuarios)	5 días	sáb 17/02/18	jue 22/02/18
6	Diseño pantalla de ver playlist	5 días	sáb 17/02/18	jue 22/02/18
7	Diseño arquitectura aplicación	1 día	vie 23/02/18	vie 23/02/18
8	Implementación pantalla de login	3 días	dom 25/02/18	mar 27/02/18
9	Implementación pantalla principal	3 días	dom 25/02/18	mar 27/02/18
10	Implementación perfil usuarios	26 días	sáb 03/03/18	vie 06/04/18
11	Implementación pantalla de playlist	26 días	sáb 03/03/18	vie 06/04/18
12	Implementación reproductor básico	12 días	jue 08/03/18	vie 23/03/18
13	Diseño del fragmento del reprocuctor (pantallas distinta a reproductor)	12 días	jue 08/03/18	vie 23/03/18
14	Integración del reproductor en todas las pantallas	12 días	jue 08/03/18	vie 23/03/18
15	Diseño pantalla populares	1 día	mié 28/03/18	mié 28/03/18
16	Implementación pantalla populares	1 día	mié 28/03/18	mié 28/03/18
17	Diseño pantalla novedades	1 día	mié 28/03/18	mié 28/03/18
18	Implementación pantalla novedades	1 día	mié 28/03/18	mié 28/03/18
19	Diseño buscador	1 día	mié 28/03/18	mié 28/03/18
20	Implementación buscador	1 día	mié 28/03/18	mié 28/03/18
21	Implemetar navigation drawer	1 día	mié 28/03/18	mié 28/03/18
22	Diseñar nevigation drawer	1 día	mié 28/03/18	mié 28/03/18
23	Adaptar circular progressbar a las necesidades de la aplicacion	2 días	jue 29/03/18	vie 30/03/18
24	Diseño pantalla de registro	1 día	lun 02/04/18	lun 02/04/18
25	Implementación pantalla de registro	1 día	lun 02/04/18	lun 02/04/18
26	Diseño pantalla generos	1 día	jue 05/04/18	jue 05/04/18
27	Implementación pantalla generos	1 día	jue 05/04/18	jue 05/04/18
28	Diseño pantalla de usuario (propio)	1 día	jue 05/04/18	jue 05/04/18
29	Implementacion pantalla de usuario (propio)	1 día	jue 05/04/18	jue 05/04/18
30	Diseño pantalla de canciones favoritas	2 días	jue 05/04/18	vie 06/04/18
31	Implementación pantalla de canciones favoritas	2 días	jue 05/04/18	vie 06/04/18
32	Diseño pantalla de playlists seguidas	1 día	jue 12/04/18	jue 12/04/18
33	Implementacion pantalla de playlists seguidas	1 día	jue 12/04/18	jue 12/04/18
34	Diseño pantalla de "usuarios a los que sigo"	1 día	jue 12/04/18	jue 12/04/18
35	Implementación pantalla de "usuarios a los que sigo"	1 día	jue 12/04/18	jue 12/04/18
36	Diseño pantalla de subir cancion	1 día	vie 13/04/18	vie 13/04/18
37	Implementación pantalla de subir cancion	1 día	vie 13/04/18	vie 13/04/18
38	Diseño pantalla de mis playlists	1 día	vie 13/04/18	vie 13/04/18
39	Implementación pantalla de mis playlists	1 día	vie 13/04/18	vie 13/04/18
40	Diseño pantalla de mis canciones	1 día	vie 13/04/18	vie 13/04/18
41	Implementación pantalla de mis canciones	1 día	vie 13/04/18	vie 13/04/18
42	Conexión con inner social functions	3 días	sáb 14/04/18	mar 17/04/18
43	Conexión con extern social functions	4 días	mié 18/04/18	dom 22/04/18
44	Incluir letras al reproductor	4 días	mié 04/04/18	dom 08/04/18
45	Incluir ecualizador al reproductor	7 días	dom 08/04/18	sáb 14/04/18
46	Crear persistencia de canciones	4 días	sáb 14/04/18	mié 18/04/18
47	Crear persistencia de sesión	6 días	mar 17/04/18	mar 24/04/18
48	Conexión con el backend	5 días	mar 24/04/18	sáb 28/04/18
49	Implementacion Oauth	5 días	mar 24/04/18	lun 30/04/18
50	Pruebas finales y corrección de errores	11 días	lun 30/04/18	sáb 12/05/18

▪ Arquitectura del Backend

La arquitectura del Backend del sistema está estructurada en 4 capas:

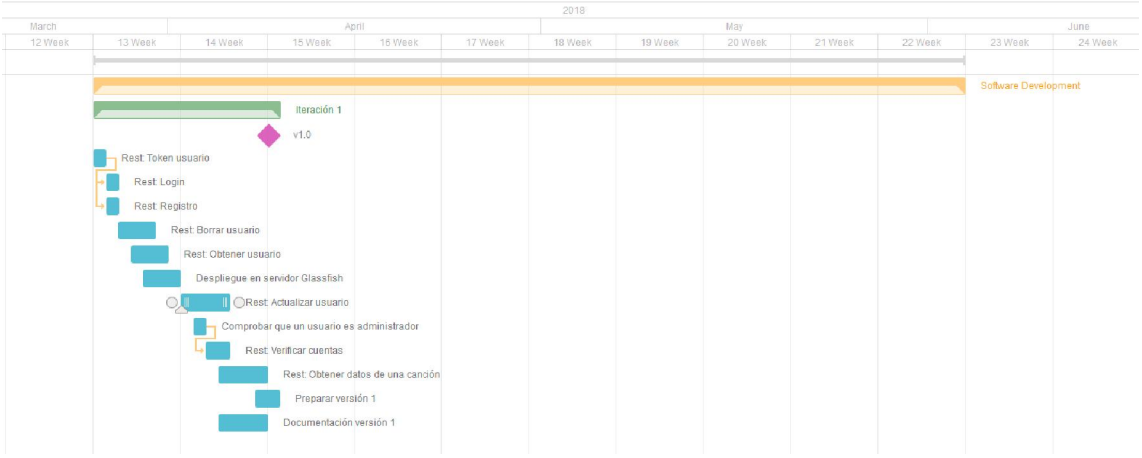
En primer lugar se halla la conexión con la base de datos, la cual será mapeada según el método objeto-relación, haciendo uso de Hibernate.

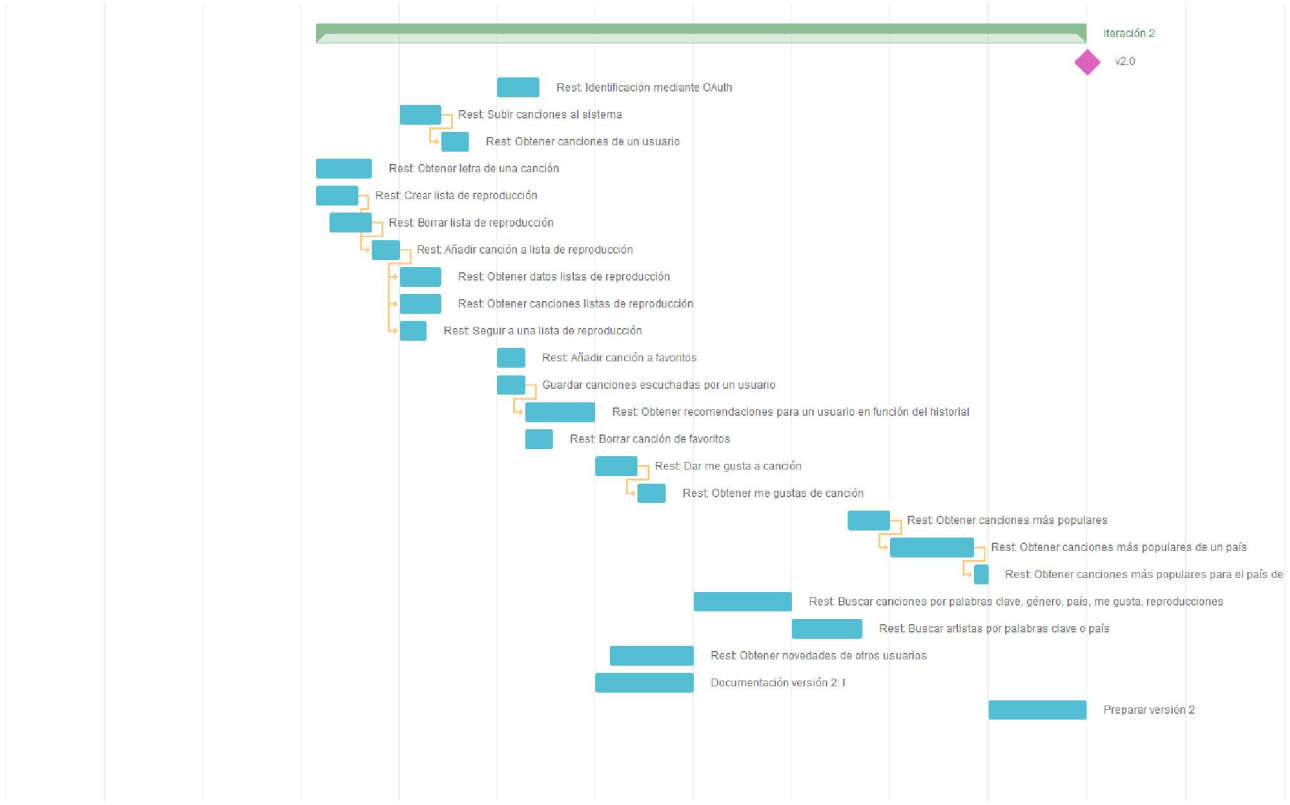
En segundo lugar, existe un servicio que atiende peticiones REST y permite devolver los datos solicitados de usuarios, canciones y listas en los formatos solicitados (JSON,XML).

En tercer lugar, una capa compuesta por una serie de caches cuya finalidad es optimizar las conexiones con la base de datos.

Por último, una cuarta capa que dispone de las utilidades necesarias para realizar funciones como la generación de cadenas aleatorias, el parseo de datos y comprobaciones sobre los mismos.

- **Diagrama de Gantt del Backend**



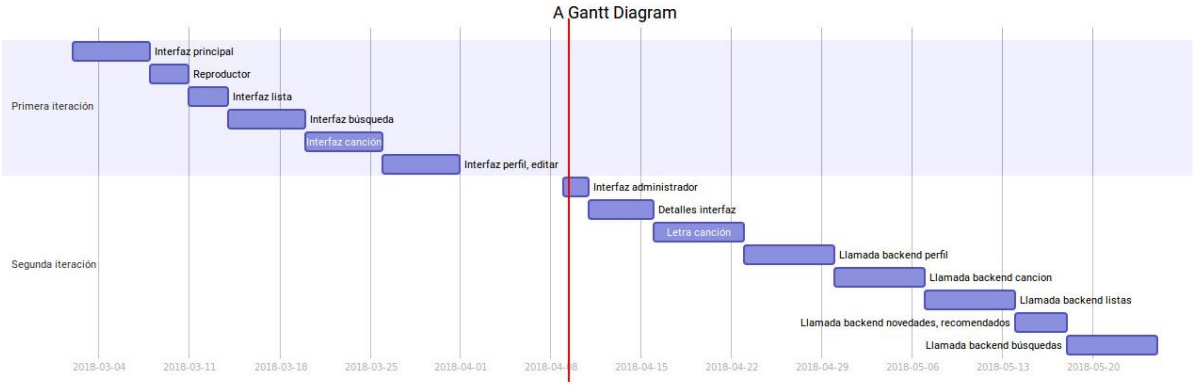


▪ Arquitectura de la aplicación Web

La página Web está estructurada de la siguiente manera:

Existe una página 'Home' desde la cual podemos ubicarnos para acceder al resto de las pantallas web, donde encontraremos las diferentes búsquedas (canciones, listas, artistas), las diferentes listas de reproducción del usuario, el perfil del artista, las canciones para reproducir, la opción de editar el perfil del artista, de editar sus propias listas, subirlas, y en el caso de ser el administrador del sistema, una página especial desde la cual se puede acceder a las diferentes opciones de administrador.

- **Diagrama de Gantt de la aplicación Web**



4. Análisis y Diseño del sistema

➢ Análisis de requisitos

▪ Sistema: Requisitos no funcionales

El sistema tendrá una versión para Android y otra para Web.

El sistema soporta los ficheros MP3, WAV, OGG.

El sistema dispone de un servidor para el almacenamiento de canciones.

La aplicación móvil soportará Android 5.0 y la Web la última versión de Firefox y Chrome.

▪ Backend: Requisitos funcionales

1. El sistema se compone de Usuarios, Canciones y Listas de reproducción.
2. El sistema permite 3 tipos de usuarios:
    - Usuario registrado (usuario que esté registrado en el sistema)
    - Usuario no registrado (que es el cual el sistema no puede identificar con ninguna cuenta)
    - Usuario administrador (aquel que es como un usuario registrado pero unos privilegios especiales, como es verificar cuentas)
3. Una canción se compone de un título, un audio, la transcripción de dicho audio, el país de la canción, cuántos me gusta tiene y el número de reproducciones.
4. Una lista de reproducción o categoría es una lista de canciones generadas por el sistema agrupadas por género, éxito, país, situación para la que es propicia o generada por el usuario.
5. Un usuario registrado se compone de un nombre, un nick único (es decir, que dos usuarios no pueden tener el mismo nick), un correo electrónico, una contraseña para acceder a la aplicación, la fecha de nacimiento, una biografía, una foto de perfil, qué canciones ha reproducido, cuántos me gusta han recibido las canciones que ha subido, cuántos me gusta ha dado, sus canciones, el país y las redes sociales que quiera añadir de forma opcional (Twitter, Facebook e Instagram).
6. El sistema permite obtener todos los datos de una canción.
7. El sistema permite que los usuarios se registren en el sistema.
8. El sistema permite que los usuarios de identifiquen tanto con usuario y contraseña así como con una cuenta de Google.
9. El sistema permite que los usuarios registrado registren nuevas canciones en el sistema.
10. El sistema permite que los usuarios registrados creen y borren listas de reproducción formadas por canciones que pueden ser o no del propio usuario, las cuales serán públicas.
11. El sistema permite que los usuarios registrados añadan una canción a una lista de reproducción que haya creado el mismo.

12. El sistema permite obtener todas las canciones de una lista de reproducción.
13. El sistema permite que los usuarios registrados tengan una lista de canciones favoritas, la cual será privada.
14. El sistema permite que los usuarios registrados añadan o eliminen canciones de favoritos.
15. El sistema permite que los usuarios registrados obtengan la lista de sus canciones favoritas.
16. El sistema permite obtener las letras de una canción.
17. El sistema permite obtener recomendaciones (canciones, listas de reproducción y usuarios), canciones más populares, novedades sobre los usuarios o listas de reproducción que un usuario sigue.
18. El sistema permite que los usuarios registrados modifiquen su nombre, su correo electrónico, su contraseña, su fecha de nacimiento, su biografía, su foto de perfil, su país y sus redes sociales.
19. El sistema permite que los usuarios registrados sigan a una lista de reproducción.
20. El sistema permite buscar canciones mediante palabras clave (título), género, país, me gusta, reproducciones o una combinación de varios.
21. El sistema permite ver las canciones más populares de un país.
22. El sistema permite ver las canciones más populares del país de un usuario.
23. El sistema permite buscar artistas mediante palabras clave (nick, nombre y biografía), país o una combinación de varios.
24. Las búsquedas por palabras clave deberán contener al menos una palabra y, si contiene sólo una palabra, que sea de longitud mayor o igual a 3 caracteres.

- ## Backend: Requisitos no funcionales

1. El sistema permite registrar usuarios cuyo nick esté comprendido entre 4 y 32 caracteres.

▪ Android: Requisitos funcionales

1. El sistema permite un solo tipo de usuario, y es el usuario registrado (usuario que esté registrado en el sistema).
2. El sistema permite que los usuarios de identifiquen tanto con usuario y contraseña así como con una cuenta de Google.
3. El sistema permite que los usuarios suban canciones a la aplicación.
4. El sistema permite que los usuarios se registren en la aplicación.
5. El sistema permite a los usuarios tener una lista de canciones favoritas, la cual será privada.
6. El sistema permite añadir a favorito una canción que se está escuchando.
7. El sistema permite desde cualquier pantalla escuchar canciones y ver los controles básicos de canciones (play, pause y pasar canción)
8. El sistema permite desde una pantalla específica acceder a más controles sobre las canciones.
9. Desde la pantalla específica el sistema permite avanzar o retroceder en la canción a un momento exacto de esta.
10. Desde la pantalla específica el sistema permite ver la letra de la canción dinámica (va al ritmo de la canción) en el caso que la posea.
11. Desde la pantalla específica el sistema permite ver la onda de sonido de la canción.
12. Desde la pantalla específica el sistema permite cambiar el orden de muestra de las canciones aleatorio o lineal).
13. El sistema posee una pantalla principal en la que se mostrarán recomendaciones (canciones, listas de reproducción y usuarios), novedades sobre los usuarios y listas de reproducción seguidos y canciones más populares.
14. El sistema posee una pantalla de usuario (otros usuarios) en la que se puede ver su información, canciones, listas de reproducción creadas, su número de seguidores, la opción de seguirlo así como un enlace a sus redes sociales.
15. El sistema posee una pantalla de usuario (usuario propio) en la que se puede ver las canciones del propio usuario y sus listas de reproducción creadas, así como añadir más canciones, listas de reproducción o eliminar alguna de las dos.
16. El sistema permite crear listas de reproducción formadas por canciones de las que puedes ser autor o no, las cuales serán públicas.

17. El sistema permite modificar los datos de usuario una vez creado.
18. El sistema posee una pantalla específica en la que se puede ver las canciones que posee una lista de reproducción así como seguirla.
19. El sistema permite buscar las canciones más populares de un género.
20. El sistema permite ver las canciones más populares en el país desde el que se conecta el dispositivo.
21. El sistema permite buscar canciones, artistas y listas de reproducción por su nombre mediante un buscador.
22. El sistema posee una pantalla principal en la que se mostrarán recomendaciones (canciones, listas de reproducción y usuarios), novedades sobre los usuarios y listas de reproducción seguidos y canciones más populares.
23. El sistema posee una pantalla de usuario (otros usuarios) en la que se puede ver su información, canciones, listas de reproducción creadas, su número de seguidores, la opción de seguirlo así como un enlace a sus redes sociales.
24. El sistema posee una pantalla de usuario (usuario propio) en la que se puede ver las canciones del propio usuario y sus listas de reproducción creadas, así como añadir más canciones, listas de reproducción o eliminar alguna de las dos.

▪ Web : Requisitos funcionales

1. El sistema permite registrarse mediante usuario o a través de Google.
2. La reproducción actual de un usuario se sincroniza en todos los dispositivos.
3. Se dispone de 3 perfiles de usuarios: registrados, no registrados y administrador.
4. Los usuarios no registrados pueden acceder a la página principal, páginas de los diferentes usuarios, páginas de canciones además de poder realizar búsquedas y pueden reproducir canciones de usuarios o de listas de reproducción públicas.
5. Los usuarios registrados pueden realizar todas las funcionalidades de un usuario no registrado.
6. El sistema permite a los usuarios registrados identificarse.
7. El sistema permite a los usuarios registrados modificar su información personal.
8. El sistema permite a los usuarios registrados subir canciones.
9. El sistema permite administrar una lista de reproducción privada de favoritos.
10. El sistema permite añadir cualquier canción a favoritos.
11. El sistema permite crear listas de reproducción públicas
12. El sistema permite añadir cualquier canción a una lista de reproducción previamente creada por el usuario.

13. El sistema permite eliminar una canción de sus listas de reproducción o de la lista favoritos.
14. El sistema permite seguir/ dejar de seguir a un usuario.
15. El sistema permite seguir/ dejar de seguir a una lista de reproducción.
16. El sistema permite modificar sus enlaces a redes sociales.
17. El sistema permite al usuario administrador puede verificar la identidad de un usuario.
18. El sistema permite al usuario administrador modificar la información de un usuario, canción o playlist.
19. El usuario administrador se identifica en la aplicación con un usuarios y contraseña predefinidos.

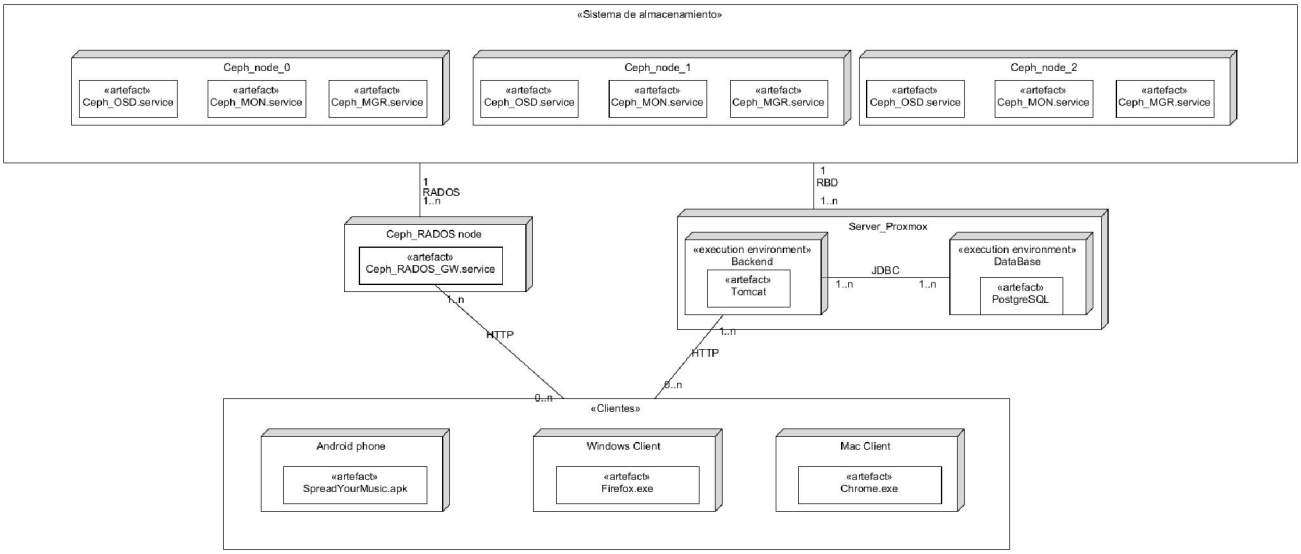
▪ Web: Requisitos no funcionales

1. El sistema tiene una pantalla principal donde se mostrarán novedades de artistas y canciones, y canciones más populares.
2. La pantalla principal de un usuario registrado contiene recomendaciones de artistas, listas o usuarios en base a sus gusto y novedades de sus artistas/listas seguidos.
3. La pantalla de administrador incluye un buscador para la búsqueda de un usuario, lista o canción y una lista con el resultado de la búsqueda.
4. Cada canción, lista o usuario tiene una pantalla de administrador desde donde el sistema permite al administrador la modificación.
5. El sistema tiene una pantalla de usuario pública para todo el mundo donde aparece su información, canciones, listas de reproducción públicas, número de seguidores y enlaces a sus redes sociales
6. El sistema cuenta en todas las pantallas con un reproductor de audio que incluye información sobre la canción que se está reproduciendo y las opciones de avanzar, retroceder a un momento de la canción; avanzar o retroceder a la siguiente o anterior canción si se estará produciendo una lista parar la reproducción y cambiar el orden de reproducción a aleatorio.
7. El sistema tiene en todas las pantallas un buscador para realizar una búsqueda de canciones, listas y usuarios.
8. El sistema tiene una pantalla para cada canción donde se puede ver la información de la canción, la onda de sonido y si tuviese las letras de forma dinámica en reproducción.
9. El sistema tiene una pantalla para identificarse/ registrarse.

10. El sistema dispone de un sistema de verificación de cuentas. Esto supone un indicador gráfico en el perfil de usuario que anteriormente ha verificado su identidad. La finalidad es garantizar la verdadera identidad del usuario.

➢ Diagrama de módulos:

▪ Diagrama de distribución o despliegue:



▪ Tecnologías elegidas:

Durante la programación de la aplicación Android se decidió usar Kotlin y Java, en código heredado de otras aplicaciones (ejemplos de Google sobre uso de las bibliotecas de sonido y biblioteca para crear un seekbar circular), Kotlin interopera con código Java para el funcionamiento de la misma y XML para las vistas de la aplicación.

Durante la programación de la aplicación Web se decidió usar HTML, CSS y Javascript junto con las bibliotecas Jekyll, Jquery, Bootstrap y Wavesurfer para facilitar el desarrollo de la aplicación.

▪ Otros aspectos técnicos de interés:

La base de datos será tipo SQL.

Existe hay una API Web y será RESTful.

La aplicación de los dispositivos Android es nativa y la aplicación de los dispositivos de sobremesa es Web.

5. Memoria del proyecto

➢ Inicio del proyecto

Se planteó el proyecto diferenciando las partes principales de este, que son la aplicación Android, la versión Web para navegadores y el motor que sustenta a ambos, por ello se creó tres equipos para que cada uno trabajase en una parte del proyecto, estos grupos son Grupo Android, Grupo Front-End y Grupo Back-End. Se planteó como lenguaje principal Java y se optó por usar Kotlin (un lenguaje que corre en máquina virtual de Java) para Android por ser más sencillo, ser interoperable con Java y ser un lenguaje oficial de Android. En Web se usaría HTML, CSS y JavaScript. Se procedió entonces a realizar el análisis del sistema, durante esta etapa, uno de los componentes del equipo fue dado de baja médica por dos semanas, tras su vuelta y con el análisis ya realizado se decidió reorganizar los equipos para ajustar horas estimadas de trabajo con la carga de trabajo esperada en cada grupo.

También se procedió a buscar y configurar un clúster con el que trabajar.

➢ Ejecución y control del proyecto

El reparto de tareas se ha realizado, como ya se ha explicado, separando el grupo de trabajo en tres subgrupos de desarrollo enfocado (App Android, Front-end Web y Backend).

Dentro de estos subgrupos el coordinador de ese grupo creaba tareas de trabajo para que cada desarrollador fuera eligiendo para hacer, se detectó que este modelo de trabajo era bastante ineficiente ya que no había fechas ni obligaciones. Con el objetivo de llevar el proyecto al día según los diagramas de Gantt se corrigió el modelo de trabajo de forma que cada tarea a realizar tiene como límite una semana desde su inicio. Las causas de este cambio incluyen: dejadez de trabajo, aferrarse a una tarea, olvido de trabajo, sobrecarga de trabajo externo.

La comunicación interna ha seguido siendo a través de WhatsApp aunque se ha demostrado un poco ineficiente debido a que las conversaciones se mezclan y pierden un poco, por ello y para realizar un control periódico se ha procedido a realizar conversaciones grupales de voz semanales a través de las aplicaciones Skype y Discord en las que se discute y comenta que ha hecho o está haciendo cada grupo y persona, sin embargo en estas reuniones no se redacta ningún acta por ser más relajadas, no estar los miembros en persona y sobre todo no tomar decisiones nuevas o cambios representativos en el proyecto.

El progreso del proyecto se va midiendo con tarjetas de tareas en GitHub tal y como se planeó. Las reuniones con el cliente han sido recogidas en actas en las cuales se ha escrito temas hablados, errores cometidos para su arreglo y planteamiento de mejoras; dentro de estas se incluyen las fechas, duraciones y personal presente.

El proyecto continúa con las tecnologías y lenguajes pensados en un inicio. No ha habido incidencias en integración de código debido al uso de Git-Flow y a la decisión de no realizar pruebas unitarias automatizadas a este. A fecha de realización de este documento todavía no se han

realizado despliegues. Sin embargo, el clúster que se usará como centro del Back-End ya está operativo para conectarse a él.

➢ Cierre del proyecto

La siguiente tabla mide el esfuerzo realizado por cada miembro del proyecto en su respectiva tarea, así como una serie de observaciones/aclaraciones.

Miembro del equipo	Horas dedicadas	Tarea	Observaciones
Yasmina Albero	8h	Desarrollo de la aplicación Android	
Jorge Aznar	24,5h	Desarrollo del software para el Backend	Desarrollo memoria técnica v2
Ángel Cañal	38,25h	Desarrollo del software para el Backend	Líder sub-grupo Backend y desarrollo memoria técnica v1
Abel Chils	74h	Desarrollo de la aplicación Android	Líder sub-grupo Android y director del proyecto
Oscar Fraca	12h	Desarrollo del software para el Backend	Desarrollo memoria técnica v2
Alex Oarga	27h	Desarrollo de la aplicación Web	
Jorge Pinilla	28h	Desarrollo de la aplicación Web	Líder sub-grupo Frontend y gestor servidores Backend

6. ANEXO I

➢ Bibliografía:

Estándar para Android y Web (https://google.github.io/styleguide/javaguide.html)

Estándar recomendado por la W3Schools para Frontend (https://www.w3schools.com/html/html5 syntax.asp)

Git Flow (https://danielkummer.github.io/git-flow-cheatsheet/)