Universidad Nacional Autnoma de Mxico

Reflexin final del proyecto de Ingeniera de Software 2022-2

Demian Alejandro Monterrubio Acosta
Ncolas Kano Chavira
Erik Federico del Ro Pulido
Emiliano Dominguez Cruz
Rodrigo Garca Padilla

23 de junio de 2022

1. Introduccin

1.1. Propsito del documento.

En este documento se encuentran las especificaciones de la nueva pgina web del servicio musical FCmusic. Esta plataforma le dar al cliente acceso a toda su msica desde la comodidad de su telfono y/o computadora de manera rpida y sencilla. Este documento planea guiar el diseo e implementacin de la plataforma.

1.2. Resumen del Proyecto

Nombre del proyecto: FCmusic.

Desarrolladores:

- del Ro Pulido Erik Federico
- Garca Padilla Rodrigo
- Monterrubio Acosta Demian Alejandro
- Kano Chavira Ncolas
- Domnguez Cruz Emiliano

Usuarios finales: Miembros de la comunidad de la Facultad de Ciencias que quieran escuchar la msica que tienen.

1.3. Antecedentes

Actualmente existen diversas aplicaciones especializadas en escuchar msica, algunas de las ms famosas son: Spotify, Apple music, Amazon music, Deezer, Google music, YouTube music, etc. La mayora de estas aplicaciones son de pago, o requieren de una suscripcin para usar todas las funcionalidades. A travs de este proyecto vamos a replicar las caractersticas ms importantes de estas aplicaciones e implementar un sistema gratuito que permite escuchar la msica que se tiene compartida mediante archivos en google drive.

1.4. Propsito del sistema

1.1.1 Usuarios

El sistema, (idealmente) estar disponible para cualquier usuario en la web (de la facultad de ciencias).

- Usuarios: miembros de la comunidad de la Facultad de Ciencias, que quieran escuchar y compartir msica previamente guardada en Google Drive.
- Administradores: administradores del sistema, encargados de dar mantenimiento.

1.1.2 Ubicacin

El sistema, (idealmente) estar disponible para cualquier usuario en la web (de la facultad de ciencias).

1.1.3 Necesidad

Este sistema est hecho para poder acceder a las funcionalidades de cualquier sistema de msica sin tener que pagar.

2. Requerimientos del Cliente

El cliente requiere un sistema para escuchar msica; este sistema debe ser accesible desde cualquier navegador, pero tambin debe tener una aplicacin mvil. El sistema deber poder acceder a toda la msica guardada en una carpeta de google drive, obtener su informacin, administrarla, reproducirla, y tener las funciones ms importantes de las aplicaciones de reproduccin de msica (tipo Spotify, Amazon Music, etc.). El sistema estar disponible para toda la comunidad de la facultad de ciencias.

3. Objetivos

3.1. Objetivos Funcionales

3.1.1. Alta Prioridad

El sistema podr conectarse con Google Drive para extraer las canciones, guardar su informacin en la base de datos y utilizarlas.

El sistema deber poder guardar (en una base de datos):

- Canciones
  - Artista
  - Nombre de cancin
  - Ao de lanzamiento
  - Gnero
  - Duracin
  - Ao de lanzamiento
  - lbum al que pertenece
  - Estrellas
  - Vistas
- Usuarios:
  - Nombre
  - Correo electrnico
  - Canciones que escuchan
  - Canciones que les gustan

        - Playlist
        - Escuchado recientemente
    - Listas de reproduccin
        - Estado (pblico / privado)
        - ndice de cancin
    - lbumes
        - Nombre del lbum
        - Canciones del lbum
    - Artistas
        - Nombre del artista.
        - Canciones del artista.
    - Estadsticas

El sistema deber tener listas de reproduccin y podr ponerlas en aleatorio (shuffle). Las listas sern:

- Personales
- Disponibles para todos

El sistema contar con un sistema de bsqueda difusa para encontrar canciones. (Que autocomplete canciones o ayude en las maysculas y minsculas para encontrar la cancin).

El sistema podr reproducir canciones, pausarlas, adelantarlas, retrocederlas, pasar a la siguiente cancin o volver a la cancin anterior.

El sistema tendr un sistema de usuarios.

3.1.2. Media Prioridad

El sistema contar con un sistema de calificacin con estrellas.

El Sistema tendr un modelo de estadstica a base de:

- Calificacin dada
- Canciones ms escuchadas
    - todo el tiempo
    - en el momento (popularidad)

El sistema tendr una biblioteca local con canciones descargadas por el usuario.

3.1.3. Baja Prioridad

El sistema tendr un sistema de recomendaciones.

- Basado en ao, gnero, artista.

El sistema tendr un modelo de interaccin entre usuarios, (i.e. Formar listas de reproduccin compatibles de acuerdo a los gneros que ambos escuchan).

El sistema permitir seguir a un artista o gnero

- con un sistema de notificaciones. (i.e. Te avisa cuando hay una nueva cancin de un artista o algn usuario crea una lista de reproduccin)

3.2. Objetivos No Funcionales

3.2.1. Fiabilidad

- El sistema intentar no colapsar, en caso de ocurrir un error se debe notificar al usuario sin cerrar la aplicacin.

3.2.2. Usabilidad

- Un usuario que conoce cmo usar otro sistema de msica ser capaz de entender cmo usar este sistema en poco tiempo.
- Los administradores podrn usar eficientemente el sistema despus de haber recibido un entrenamiento.

3.2.3. Rendimiento

- El sistema ser lo suficientemente bueno en rendimiento como para ser usado cmodamente por un usuario.

3.2.4. Seguridad

- Las cuentas de los usuarios tendrn sus contraseas protegidas para evitar el robo de su informacin.
- Tendremos la seguridad que ofrece Google.

3.2.5. Compatibilidad

- El sistema funcionar en la mayora de los navegadores ms usados.
- El sistema podr funcionar en la aplicacin de dispositivos mviles.

3.2.6. Documentacin de usuario en lnea y ayuda

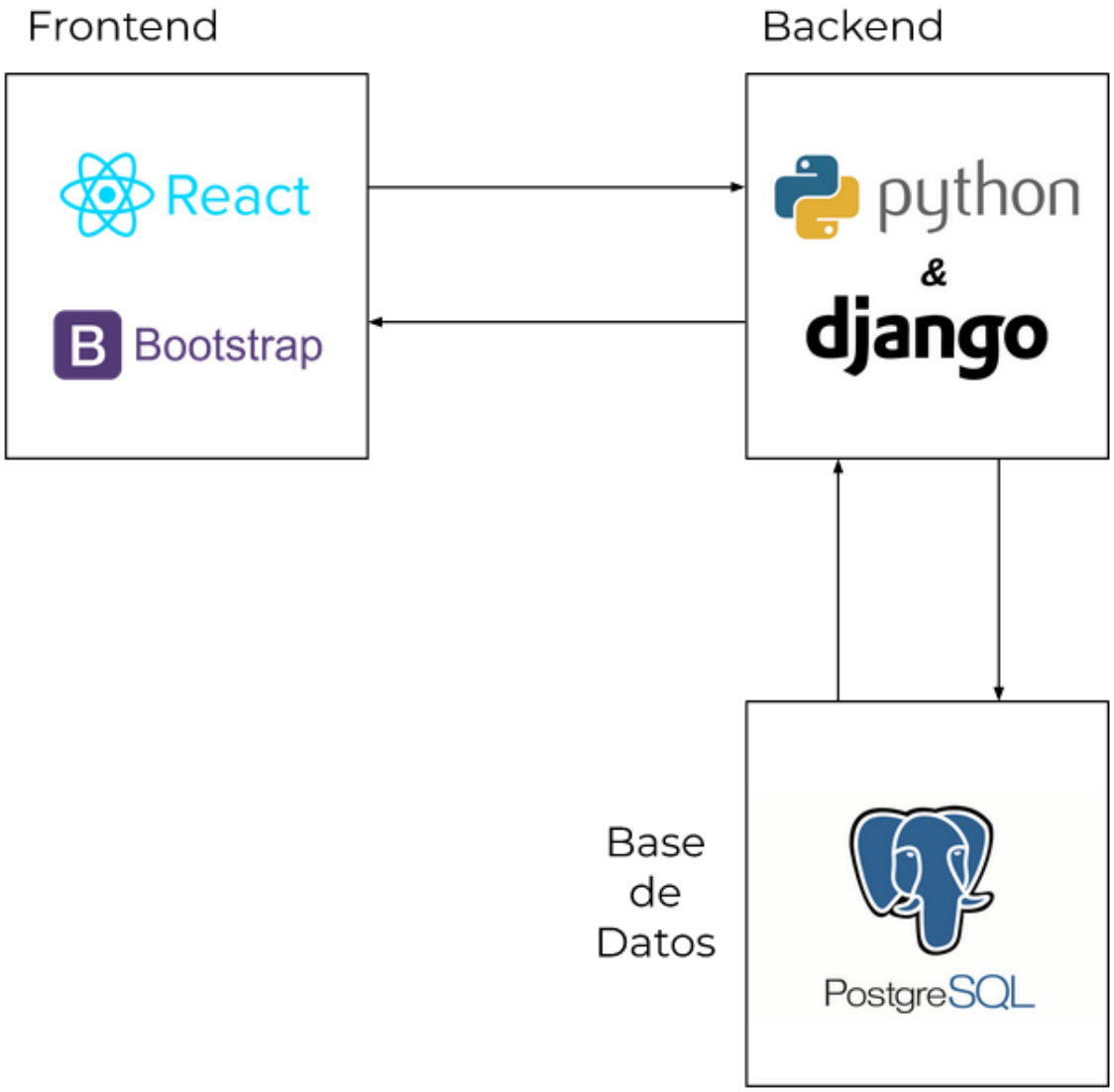
- El sistema tendr un manual claro y documentado para el usuario, al cual se podr acceder fcilmente desde la pantalla principal del sistema.

3.2.7. Robusto

- El sistema ser robusto respecto a errores.

4. Arquitectura del Sistema

Las tres partes ms importantes del sistema son: Frontend: la parte del sistema con la que el usuario interacta. Backend: la parte del sistema que se encarga de implementar todas las funcionalidades y la comunicacin entre las otras partes. Base de datos: la parte que guarda toda la informacin y la consulta cuando sea necesario.

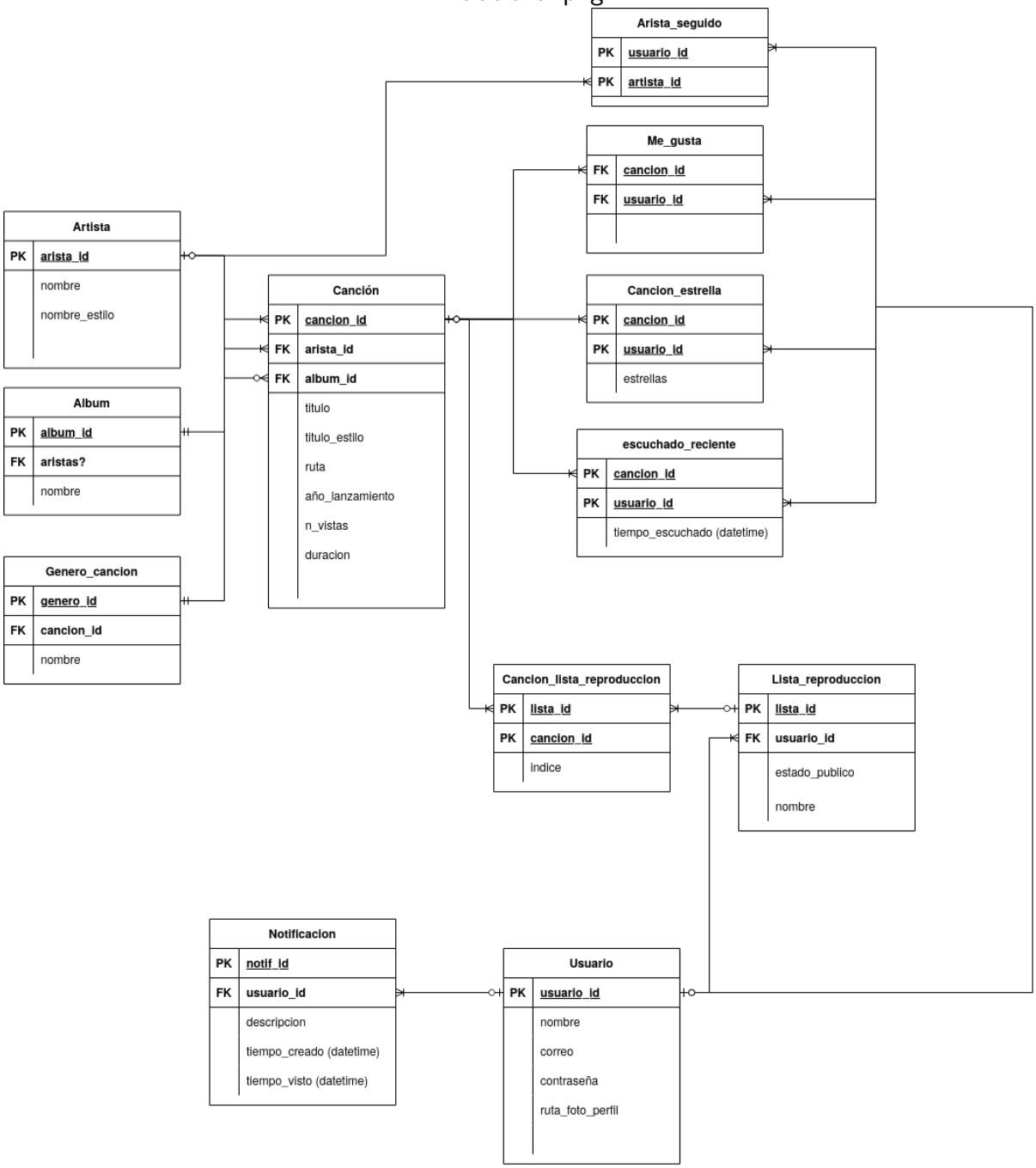


En este diagrama podemos ver la comunicacin entre los mdulos ms importantes, y dentro de cada mdulo, la tecnologa que usaremos para su implementacin. Para la base de datos usaremos PostgreSQL, para el backend usaremos Python con Django, y para el frontend usaremos React y Bootstrap. Estamos usando el modelo vista controlador.

5. Modelos del Sistema

5.1. Modelo relacional de la base de datos:

Relacional.png



Este modelo representa la informacin que guardaremos en la base de datos; cada cuadro es una tabla en la base de datos, que almacena la informacin listada. Las lneas que conectan las tablas indican la relacin entre ellas. (Este modelo es tentativo).

5.2. Modelo de clases del backend (simplificado):

de Clase Backend.png

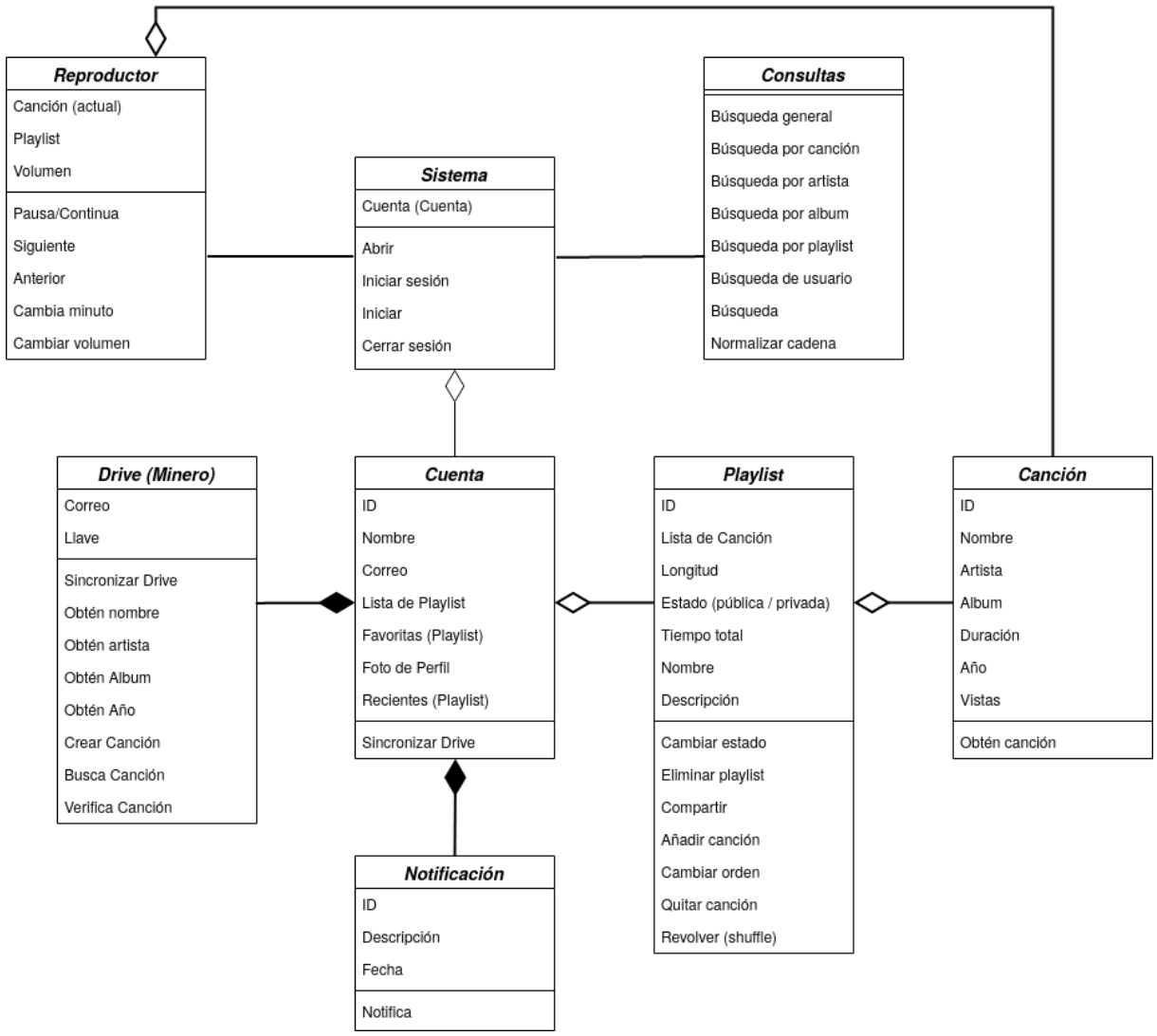


Diagrama tipo UML (simplificado) de clases correspondiente al funcionamiento del backend. Las clases que harn que funcione el sistema estn representadas por cuadros, sus atributos estn en el primer rectngulo, y sus mtodos en el segundo rectngulo. Los nombres de los atributos y mtodos no son los nombres finales del sistema, representan nombres descriptivos de lo que son / hacen. El diagrama de clases corresponde a las funcionalidades de alta prioridad.

6. Conclusiones

El sistema que planeamos en este documento tiene las funcionalidades ms importantes y esenciales para su funcionamiento; hay muchas otras caractersticas que podran ser aadidas, aunque en principio no se

planea que el sistema evolucione. La principal deficiencia del sistema es que no cuenta con una base de datos con las canciones, depende de que estas canciones sean aadidas en Google Drive por los usuarios, esta decisin la tom nuestro administrador (Canek), para que el sistema no tuviera problemas legales con el uso de msica; si el sistema llegara a evolucionar, es bastante posible que esta caracterstica se mantenga. Como obtenemos la informacin de los archivos mp3, podramos tener ciertas inconsistencias con la informacin real de las canciones; intentaremos que la informacin que mostremos de las canciones sea correcta; pero priorizaremos que el sistema sea funcional y consistente con sigo mismo. Esperamos que el sistema cuente con todas las caractersticas planteadas y pueda estar disponible para la facultad de ciencias.

7. Apndice

7.1. Descripcin de casos de uso (para casos selectos).

Caso de Uso:	Acceder a cuenta existente
Resumen:	Se inicia sesin en la cuenta del usuario existente.
Flujo bsico:	1.Se redirige a la pgina de inicio de sesin de google. 2.Se ingresa su nombre de usuario o correo. 3.Se ingresa la contrasea correspondiente a la cuenta. 4.Se da click en el botn de acceder 5.El usuario entra a su cuenta
Flujos alternativos:	Si el usuario no recuerda su contrasea, google se encargar de restablecerla.
Postcondiciones:	El usuario ingresa a su cuenta..
Reglas de negocio:	El usuario no debe crearse dos veces, es decir cada persona tiene derecho a una sola cuenta por correo. Se debe cumplir con el dominio del correo.

Caso de Uso:	Escuchar una lista de reproduccin en shuffle.
Resumen:	Se escucha una lista de reproduccin en orden aleatorio.
Flujo bsico:	1.El usuario se va a la seccin playlists 2.El usuario entra en una lista de reproduccin 3.El usuario selecciona el botn shuffle 4.La lista de reproduccin se ordena de manera semi aleatoria. 5.El sistema empieza a reproducir la primera cancin de la lista de reproduccin con el orden cambiado. 6.El sistema contina reproduciendo canciones siguiendo el orden alterado.
Flujos alternativos:	4.5. El usuario puede volver a seleccionar el botn shuffle las veces que quiera. 7.Si ya se escucharon todas las canciones de la lista, el sistema deja de reproducir canciones.
Precondiciones:	El usuario est dentro del sistema. El usuario est en la pgina principal del sistema. El usuario tiene al menos una lista de reproduccin.
Postcondiciones:	El usuario escucha su lista de reproduccin en un orden semialeatorio.
Reglas de negocio:	Seleccionar el botn shuffle no afectar a ninguna lista de reproduccin que tenga guardado el usuario, no crear ni eliminar nada; y si el usuario vuelve a la pgina principal, la lista volver a estar ordenada como antes del shuffle.

Caso de Uso:	Agregar una cancin a una playlist.
Resumen:	Se modifica una playlist existente para agregar una cancin.
Flujo bsico:	1. El usuario elige una cancin, ya sea que la est escuchando o la haya buscado. 2. El usuario hace click en el botn agregar a playlist. 3. El sistema muestra una lista de playlist candidatas para agregar. 4. El usuario selecciona la playlist a cul agregar. 5. El sistema agrega la cancin al final de la playlist.
Flujos alternativos:	5. Si la cancin ya se encuentra en el playlist no se agrega una copia.
Precondiciones:	El usuario ha ingresado a su cuenta. Debe de existir una playlist en la cuenta del usuario
Postcondiciones:	La cancin ahora se encuentra en el playlist seleccionado
Reglas de negocio:	

Caso de Uso:	Un usuario sube una nueva cancin al sistema
Resumen:	Este caso de uso le permite a un usuario aadir una nueva cancin al sistema para que se pueda escuchar.
Flujo bsico:	1. El usuario quiere subir una cancin al sistema. 2. El usuario sube la cancin a su carpeta de google drive. 3. El usuario inicia sesin en el sistema. 4. El usuario se va a la seccin de su cuenta. 5. El usuario presiona el botn de sincronizar. 6. El sistema automticamente encuentra la cancin nueva en su cuenta de drive y saca la informacin de la cancin usando los meta-datos del mp3. 7. El sistema presenta una pantalla de sincronizacin exitosa. 8. La cancin ya est dentro del sistema y se puede buscar en el buscador. 9. El sistema manda una notificacin al usuario con las nuevas canciones aadidas.
Flujos alternativos:	6. El usuario corrige cualquier dato incorrecto. 7. El usuario da aceptar para guardar los cambios. 10. La cancin se sube exitosamente.
Precondiciones:	El usuario puede iniciar sesin en el sistema.
Postcondiciones:	Como la cancin ya est en el sistema, ya se puede usar para cualquier otro caso en el que se necesite.
Reglas de negocio:	Las canciones ya existentes, ya no se guardan. Es la misma cancin si es del mismo artista y tiene el mismo nombre.

8. Resultados y Reflexiones:

La versin final del proyecto no cumpli con los requerimientos descritos en este documento; a continuacin presentaremos una reflexin grupal e individual sobre el proyecto y por qu no pudimos cumplir con las expectativas.

8.1. Generales

A partir de una discusin del equipo llegamos a que, de manera general, estas son las razones por las que no acabamos el proyecto:

- Integracin tarda de componentes: Se dej la unin de los componentes al final, cosa que impidi que el producto final se lograra, subestimando la complejidad del despliegue web del mismo, as como una base de datos en la nube.
- Fragmentacin de los integrantes: Los primeros elementos del equipo terminaron dejando la materia, cosa que evit la continuidad del proyecto, y a su vez trajo conflictos internos de decisin de temtica del mismo.
- Falta de comunicacin entre el equipo: Poca comunicacin entre integrantes, lo cual trajo incertidumbre entre los avances significativos de cada uno y conciencia de otros integrantes de los mismos.
- Mala tcnica de organizacin: Carencia de tcnicas vistas durante el curso tales como SCRUM, debido a que aunque se implementaron en un inicio, terminaron por abandonarse debido a que no resultaron orgnicas, las cuales hubieran permitido llevar un ritmo constante de trabajo.
- Falta de experiencia: Nuestro equipo careca de personas que hubieran llevado la realizacin de un proyecto de principio a fin (en proyectos web), por lo cual no sabiamos como abordar el proyecto, ni sus componentes y hubo detalles no previstos hasta una etapa tarda del desarrollo.
- Implementacin tarda del proyecto: Se dej la parte de las tecnologas del proyecto para el final, lo cual trajo problemas con la estimacin de tiempos y por tanto finalizacin.
- Falta de buenas de prcticas de ingeniera de software: Carencia de uso de interfaces para divisin del trabajo, y de herramientas coolaborativas.

Individualmente, estas son las reflexiones que tenemos:

8.2. Demian

Como mencionamos antes hubo muchos factores que afectaron el resultado final del proyecto, desde mi punto de vista, los factores ms determinantes fueron una mala planeacin y organizacin, y un comienzo tardo del proyecto.

Al principio s intentamos seguir buenas prcticas, como escribir minutas, reuniones semanales, un pizarrn gil, un desarrollo estructurado y una buena planeacin del proyecto; pero, creo que en la primera etapa del proyecto nos cost trabajo comunicarnos como equipo, hablbamos muy poco en las reuniones (yo creo que por ser en lnea), y nadie tena iniciativa; esto junto con que los primeros dos meses hubo pocos avances, hizo que se volviera cansado y hasta innecesario seguir con las buenas prcticas.

Algo que tambin me parece importante, es que no supimos como abordar el proyecto, nadie en el equipo tena suficiente experiencia como para guiar el desarrollo, y adems no conocamos la mayora de las herramientas que bamos a usar; cremos que dividir las tareas sera una buena forma de avanzar, pero como no habamos planeado bien el proyecto, no estructuramos interfaces, (lo intentamos, pero la planeacin que tenamos en un principio se vio muy deficiente), separar las tareas hizo que no estuviramos conscientes de los dems componentes del proyecto.

En particular yo trabaj nicamente en el minero durante los primeros tres meses del semestre, como era la parte que me tocaba pens que no sera necesario aprender las tecnologas de los dems componentes; pero, cuando lleg la hora de juntar las partes, yo no entenda el funcionamiento del sistema, ni tampoco entenda el cdigo escrito en java script, por lo que fue muy complicado integrar mi parte al sistema; adems, despus de terminar el minero, tena que trabajar con otras partes para las cuales necesitaba java scrit, como sent

el tiempo encima, en lugar de aprender el funcionamiento del sistema, intent solo solucionar el problema especfico que tena; al final esto no funcion y no pude integrar los otros componentes en los que estaba trabajando. Considero que fue una muy mala manera de abordar el proyecto por parte ma.

En lo personal, siento que me tard mucho en empezar a trabajar, como en las primeras dos presentaciones no era necesario llevar cdigo (o no tanto), solo estuve trabajando en el documento de requerimientos; deb haber empezado antes a investigar para el proyecto y a escribir cdigo desde antes, as me hubiera topado ms rpido con los problemas, y hubiera tenido el tiempo para buscar la solucin correcta.

Las ltimas semanas sent que era dficil avanzar en el proyecto porque haba una buena parte del sistema que no entenda a profundidad, y cuando escriba ms cdigo aparecan nuevos errores. Facores que no consideramos empezaron a aprarecer hasta el final (porque nos tardamos en integrar); por ejemplo; cuando queramos lanzar el docker, haba problemas porque las urls en el docker eran diferentes, y google no las permitan pues no eran pblicas; o que desde un inicio no us ambientes para el desarrollo (no los conoca), al final tuve que ponerme a separar lo que s era necesario para el funcionamiento del sistema de lo que no era.

Fueron muchos factores los que fueron sumando al resultado del proyecto, pero siento que s obtuve cierta experiencia que me facilitara realizar un proyecto similar si tuviera que hacerlo; y aunque no segu muchas buebas prcticas, s siento que aprend su importancia y los resultados que se obtienen al no seguirlas.

8.3. Ncolas

Me centre mucho en el trabajo de la aplicacion movil lo cual causo que me perdiera de como estaba hecha la aplicacion web y se me dificultara el acoplamiento de la parte de la app que contactaba a la base de datos de la aplicacion web. debi estar mas al pendiente de lo que estaba haciendo mi equipo en vez de aislarme en mi parte

Otro problema que tuve fue que sobreestime mi habilidad/infraestime la dificultad de algunas caracteristicas que dijimos que tendria la aplicacion y no pude llevarlas acabo en el producto final

8.4. Erik

Al dividirnos el trabajo entre varios de mis compaeros, me sent perdido cuando trabajaba en las partes de mis compaeros. Se me hizo muy difcil entender lo que hacan otras partes del proyecto.

Creo que pude haber motivado a mis compaeros para trabajar ms y de manera mejor organizada durante las diferentes etapas del desarrollo de nuestro trabajo. Y as avanzar de manera ma rpida y eficiente.

Sent que mi productividad baj hacia el final del semestre porque ya estaba fatigado. Y creo que pude haber avanzado mucho ms si no hubiera bajado el ritmo de trabajo.

8.5. Emiliano

Subestime la complejidad de aprender un framework, mientras que en el equipo donde originalmente me encontraba todos fuimos aprendiendo al mismo ritmo, al hacer el cambio de equipo tuve que adaptarme al aprendizaje ya realizado por el resto de mi nuevo equipo, que a pesar de sus mejores intentos de explicar el funcionamiento me tomo bastante tiempo de experimentacin tener una idea del funcionamiento del proyecto. De manera similar, poder ejecutar el docker causo que me retrasara en las tareas que se me asignaron. Pude haber invertido ms tiempo en resolver por mi cuenta los problemas con los que me

encontr, pero permit que las dems materias ocuparan mi tiempo, tambin pude haber pedido ayuda para aclarar las dudas que tena sobre el funcionamiento del proyecto, pero (y lo admito como falla personal y de nadie ms) no ped ayuda de mis compaeros en el tiempo adecuado para poder solucionar eficientemente los problemas.

8.6. Rodrigo

Por mi parte falt velocidad en el aprendizaje de las tecnologas usadas tales como el framework django (a diferencia de React en mi anterior equipo), as como la estructuracin del progreso del proyecto luego de mi incorporacin al nuevo equipo. Adems de falta de comunicacin para evitar trabajar en lo mismo.