GRUPO 8
COMPATIFY

Participantes:

- Emilio Gómez Esteban, emige19@uma.es
- Fernando Javier López Cerezo, lopezalhaurin@uma.es
- Enrique Pérez Haro, enriqueperezharo@uma.es
- Alberto Trigueros Postigo, albertotrigueros22@uma.es
- Jesús Fuentes Moya, jesusfuentesm4@uma.es
- María Peinado Toledo, mariiapt02@uma.es
- Antonio Trujillo Reino, antoniotrujillo@uma.es
- Juan José Rodríguez Hernández, jjrodriguezhernandez@uma.es

Enlaces:

Enlace al repositorio GitHub: https://github.com/fjlc-73/g08-por-determinar
Enlace al espacio de trabajo de Trello: https://trello.com/w/espaciodetrabajodeuser46307107

Índice

1. Introducción

Hoy en día la música se ha convertido en algo indispensable para la vida diaria y Spotify es la principal aplicación para ello. ¿Te gustaría conocer gente afín con tus gustos musicales? ¿Nunca has pensado comparar tus canciones favoritas con las de tus amigos? Con este propósito nace Compatify, software online de uso gratuito.

Nuestra aplicación se trata de un software que estima la compatibilidad musical entre dos individuos. Cada usuario se registra en la aplicación, añadiendo su cuenta de Spotify, y mediante su historial de reproducción se podrá obtener el porcentaje de similitud con otro usuario de Compatify. Esta comparación se basa en aspectos como artistas, canciones o géneros musicales. Finalmente, se mostrará el resultado de forma sencilla y visual.

2. Asignación de roles

En cuanto a los posibles roles que pueden adoptar los participantes en este proyecto tendremos en cuenta los siguientes:

- Analista: especifica la documentación independiente del modelo de implementación.
- Diseñador: especifica la documentación dependiente del modelo de implementación.
- Programador: implementa el programa en los lenguajes y marcos de trabajo escogidos.
- Jefe de proyecto: coordina las operaciones y se responsabiliza de las decisiones tomadas.
- Ingeniero de pruebas: diseña e implementa los casos de prueba unitarios, de integración, etc.

(En caso de que sea necesario en el proyecto, aparecerán nuevos roles como el de grafista o jefe de SCRUM, pero por el momento esos son los esenciales)

Que un integrante tenga un rol significa que será responsable de esa tarea, sin embargo todos los participantes estarán informados de todas las tareas realizadas en el proyecto. De esa manera, la distribución de roles en primera instancia puede quedar de la siguiente manera:

Analista	**Diseñador**	**Programador**	**Jefe de proyecto**	**Ingeniero de pruebas**
Jesús Fuentes	Emilio Gómez	Alberto Trigueros	Fernando Javier López	Alberto Trigueros
Juan José Rodríguez	Jesús Fuentes	Enrique Pérez	Enrique Pérez	Juan José Rodríguez
Fernando Javier López	María Peinado	María Peinado	Antonio Trujillo	Antonio Trujillo
		Emilio Gómez		

3. Gestión de riesgo

Se identificarán los riesgos, es decir, los posibles contratiempos que afecten al desarrollo del proyecto y preparar un plan que minimice sus efectos.

Se abordarán los siguientes tipos de riesgo:

- Los riesgos del proyecto que afectan a la planificación y a los recursos.
- Los riesgos del producto que afectan a la calidad y al rendimiento del software bajo desarrollo.
- Los riesgos del negocio que afectan a la organización que desarrolla el software.

3.1 Límite de peticiones

- Descripción: La API de Spotify establece un límite al número de peticiones que se pueden hacer en una franja de treinta segundos. Si se excede el límite, se reciben respuestas de error 429. Spotify no especifica cuál es este límite.
- Tipo: Producto y negocio
- Probabilidad: Moderada
- Efectos: Serio
- Estrategia de gestión: Intentar optimizar el número de peticiones que se realizan. Spotify tiene para ello algunas recomendaciones en su página web.

3.2 Falta de experiencia

- Descripción: Somos un equipo con poca experiencia en el desarrollo de software, lo cual puede provocar una mayor dificultad y retraso para elaborar el código.
- Tipo: Proyecto
- Probabilidad: Alta
- Efectos: Tolerable
- Estrategia de gestión: Recurrir a recursos webs y tutoriales, así como preguntar a los profesores.

3.3 Enfermedad o baja del personal

- Descripción: Algún miembro del equipo puede caer enfermo o abandonar el proyecto, lo que provocaría un retraso en la ejecución del proyecto.
- Tipo: Proyecto
- Probabilidad: Baja
- Efectos: Tolerable
- Estrategia de gestión: Todo el equipo debe saber qué tareas realizan los demás para que pueda haber mayor flexibilidad y se pueda retomar el trabajo hecho por otro integrante. También es buena idea llevar a cabo buenas prácticas de programación como puede ser comentar adecuadamente el código.

3.4 Restricción de usuarios

- Descripción: Spotify tiene dos modos para las apps que usan su API, el modo en desarrollo y el modo de cuota extendida. Las aplicaciones inicialmente están en el primer modo, el cual requiere que los usuarios estén en una whitelist y limita el número de usuarios autenticados a 25.
- Tipo: Negocio
- Probabilidad: Baja (Poca probabilidad de que la app sea usada por más de 25 usuarios distintos)
- Efectos: Serio
- Estrategia de gestión: En caso de que queramos lanzar la aplicación al mercado y pueda ser usada por un mayor número de usuarios, habría que rellenar un formulario para que el equipo de Spotify apruebe que la aplicación pase al modo de cuota extendida, que no tiene límite de usuarios.

3.5. Cierre de la API Spotify Developers

- Descripción: El cierre temporal o definitivo de la API Spotify por la empresa. Esto nos lleva a perder el acceso a la información de cada usuario y quedaría la actividad paralizada.
- Tipo: Producto
- Probabilidad: Muy baja (No ha sido anunciado tal suceso)
- Efectos: Catastrófico

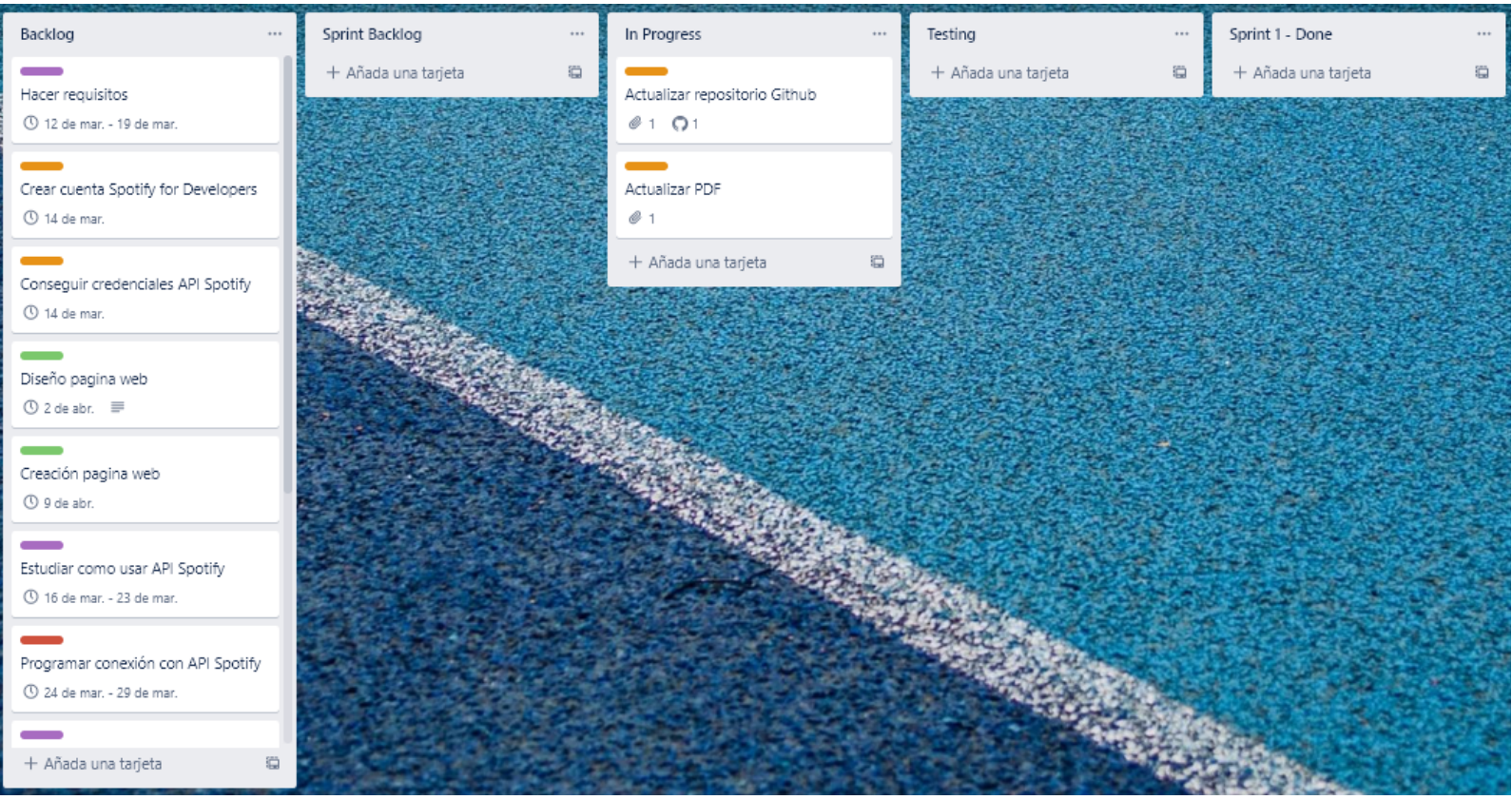
- Estrategia de gestión: Contacto con el equipo de marketing de Spotify con el fin de exponer la idea de proyecto y que se nos permita el acceso a dichos datos, implementado como extensión de la aplicación o como proyecto propio.

4. Planificación

Nos enfrentamos a un proyecto del que no conocemos mucho ya que somos novatos en este tipo de desarrollo, con lo que inevitablemente en ciertos momentos es muy probable que cunda la incertidumbre. Por ello, necesitamos un proceso ágil que nos permita en esos momentos ser capaces de rectificar, adaptarnos y priorizar de nuevo los procesos. Es por todo esto por lo que hemos elegido la metodología "Scrum", ya que tras el análisis de los diferentes modelos de proceso software existentes, "Scrum" es el más flexible y el que más se adapta al cambio. Esto se debe a que "Scrum":

- Fomenta la comunicación entre los miembros.
- Optimiza el tiempo.
- Adaptable a cambios y cambios de prioridades..
- Ciclos cortos lo cual disminuye los riesgos.
- Todo el equipo participa en todas las fases.

Es por todo esto por lo que hemos decidido que este método es el óptimo para nuestro proyecto.



En la anterior imagen se puede observar una captura de "Trello", un software de planificación que nos ayudará durante todo el proceso de desarrollo. Se compone de diferentes tareas y columnas que, al ser Scrum la metodología empleada, se componen de:

- Backlog: Documento a alto nivel del proyecto que reúne todos las tareas o requisitos.
- Sprint backlog: Subconjunto de requisitos que serán desarrollados en el siguiente sprint. Hemos escogido una duración de sprint de dos semanas y está vacío pues no hemos comenzado.
- In progress: Requisitos desarrollados en este momento del actual sprint.
- Testing: Pruebas y revisión de las tareas antes de darlas por terminadas.
- Sprint 1-Done: Tareas que se han completado del actual sprint.

4.1 Power-ups

Una de las funcionalidades de Trello, integraciones que se añaden al tablero central para mejorar la experiencia del usuario, como pueden ser gráficos o conexiones con distintas aplicaciones. Los usados en el proyecto son los siguientes:

- **Git**

- **Google Drive**

- **Burndown Chart**

5. Requisitos

A continuación, describiremos los requisitos de nuestra aplicación. Estos se tratan de los servicios que proporcionamos y las restricciones que hay en la operación de la App.

5.1 Requisistos Funcionales

Consideramos como requisitos funcionales aquellos que describen los servicios que se espera que el sistema suministre a los usuarios.

RF1: El sistema proporcionará visores adecuados
Como desarrollador
quiero que una vez hecha la comparativa entre dos usuarios de spotify, los resultados sean mostrados con gráficos que muestren con claridad la compatibilidad musical entre ellos.
Para que así Compatify sea más atractiva a los clientes

Pruebas de aceptación:
-Hacemos pruebas con distintos compañeros del equipo con alta, media y baja compatibilidad musical esperando unas grandes diferencias gráficas en los resultados.

RF2: Crear cuenta
Como usuario
quiero poder crear cuenta en Compatify para acceder a la página web.

Pruebas de aceptación:
-Introducir información personal(Nombre, Apellido, correo electrónico,edad, usuario, contraseña
-Guardar la información del usuario en la base de datos.

RF3: Enlazar cuenta
Como usuario
puedo iniciar sesión en spotify desde compatify para que se descarguen los datos en la aplicación

Pruebas de aceptación:
-iniciar sesión con su cuenta de spotify
-comprobar que les ha conectado con su cuenta

RF4: Botón actualizar
Como usuario
quiero poder actualizar la información del spotify para realizar las comparaciones con la información más reciente.

Pruebas de aceptación:
-Pulsar botón para actualizar información.
    -Si disponemos del token pedimos la información directamente a spotify.
    -Si no disponemos del token pedimos al cliente usuario y contraseña del spotify para actualizar la información desde spotify

RF5: Búsqueda de Usuario
Como usuario
quiero poder buscar otro usuario dentro de la base de datos que tenga su visibilidad pública

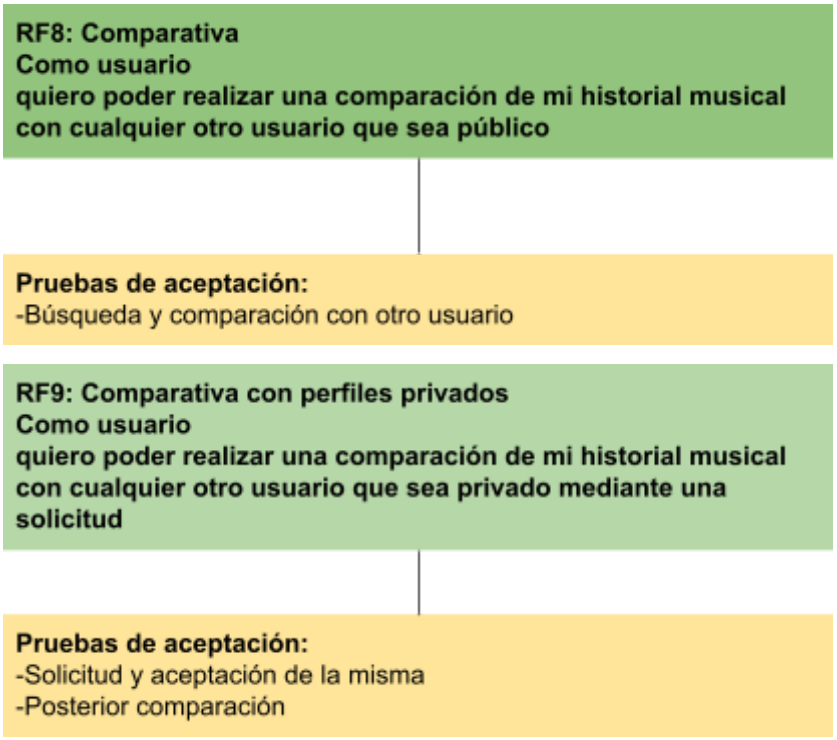
Pruebas de aceptación:
-Añadir un usuario público a la base de datos.
-Probar que puedo buscarlo desde mi usuario

RF6: Elección tipo de cuenta
Como usuario
quiero elegir si mi cuenta en Compatify es pública o privada cuando se crea la cuenta

Pruebas de aceptación:
-Al crear la cuenta seleccionar si la cuenta es pública o privada.
-Modifique el tipo de acceso de la cuenta.

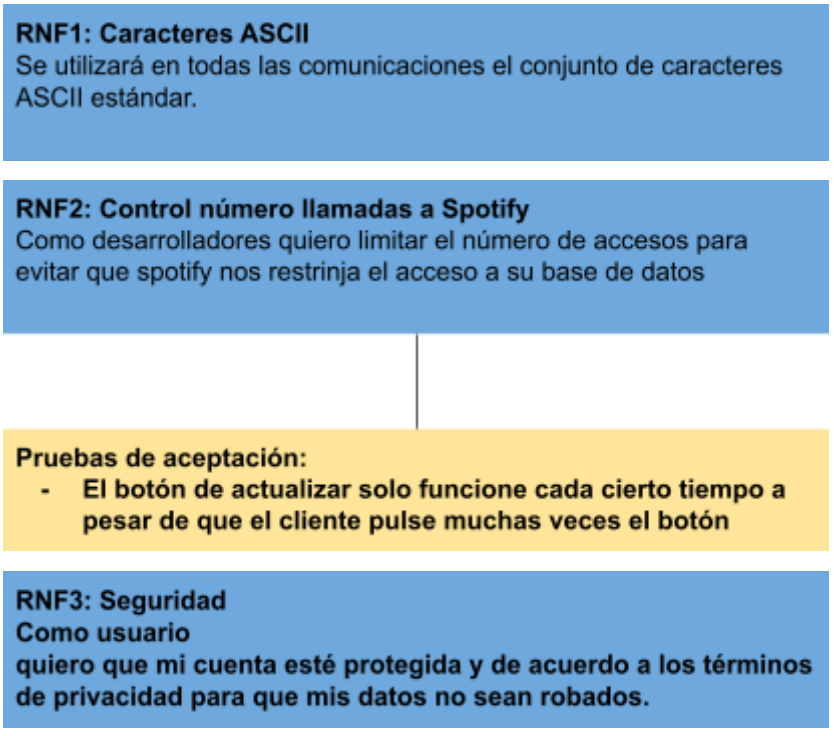
RF7: Iniciar sesión
Como usuario
quiero poder iniciar sesión en compatify en cualquier momento, y que me conecte correctamente con mi cuenta

Pruebas de aceptación:
-iniciar sesión



5.2 Requisitos No Funcionales

Consideramos como requisitos no funcionales aquellos que describen las restricciones sobre los servicios ofrecidos por el sistema.



RNF4: Facilidad de uso
Como usuario
quiero que la aplicación sea fácil e intuitiva de usar
para no tardar mucho tiempo en aprender a usarla.

RNF5: Visibilidad
Como usuario
Si mi cuenta es privada, quiero no poder ser buscado por otro usuario.

Pruebas de aceptación:
-Crear una cuenta privada
-Buscar el usuario desde otra cuenta y comprobar que no existe

RNF6: Rapidez
Como usuario
quiero que la página web funcione de forma eficiente.

RNF7: Robustez
Como desarrollador
quiero que el programa sea robusto para que el tiempo de reinicio en caso de fallo sea bajo y halla baja probabilidad de corrupción de datos.

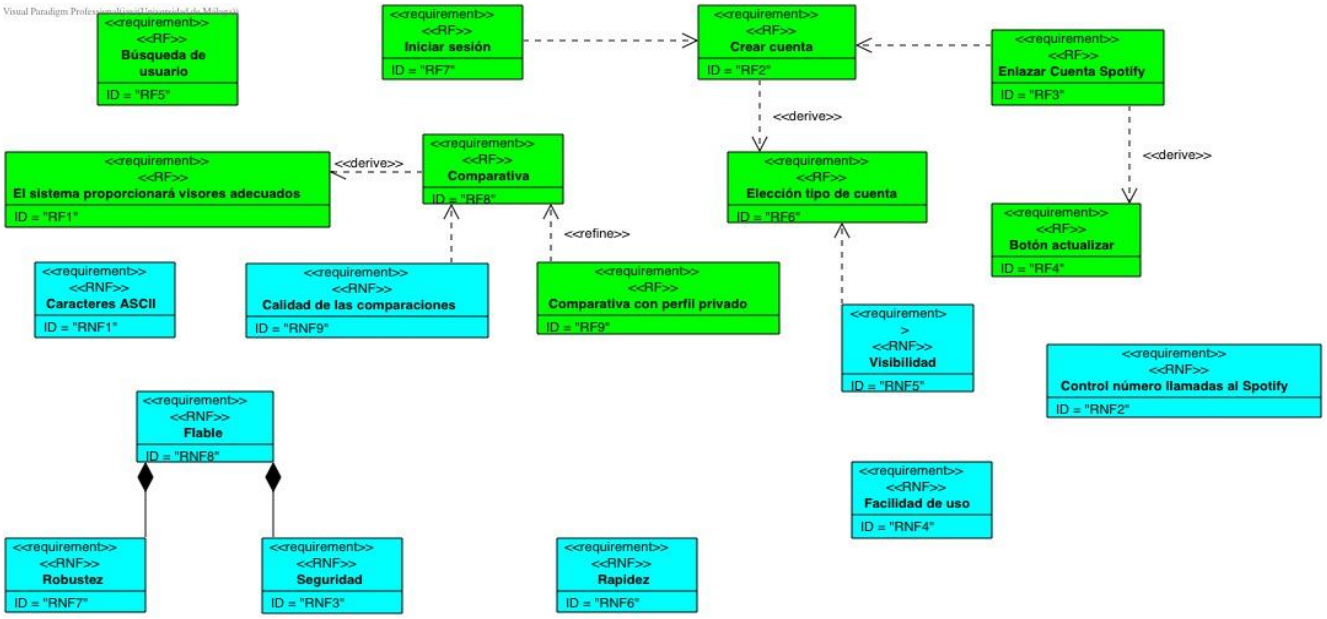
Pruebas de aceptación:
-Provocar un fallo y ver el tiempo que tarda en reiniciarse
-Intentar corromper los datos desde la página web

RNF8: Fiable
Como usuario
Quiero que ocurran la menor cantidad de fallos en las funcionalidades.

RNF9 Calidad de las comparaciones
Como usuario
quiero que al realizar comparaciones con otros usuarios se refleje si realmente tenemos gustos en común
para que la comparativa sea interesante.

Pruebas de aceptación:
-Hacer una comparación y que no refleje como gusto en común un artista que apenas se ha escuchado.

Con ayuda del software Visual Paradigm el cual proporciona diagramas para requisitos con sus respectivas relaciones, podemos verlos visualmente como sigue:



6. Herramientas software

a. **Google docs/Google drive:** Usaremos este software para la elaboración de forma colaborativa de los documentos relacionados con el proyecto.

b. **Paquete Office:** Lo usaremos para redacción y apuntes del proyecto.

c. **Whatsapp:** Software empleado para la comunicación no instantánea para la organización y planteamientos de diferentes ideas sobre el proyecto.

d. **Discord:** software usado para la comunicación instantánea a distancia. Es decir, para mantener una conversación mientras hacemos el proyecto.

e. **GitHub:** Repositorio remoto para trabajar de forma colaborativa sobre el código de la aplicación.

f. **Trello:** Software para la planificación y organización de las diferentes tareas y subtareas del proyecto.

g. **Visual Paradigm:** Software para la creación de los requisitos y establecer relaciones entre ellos.