Plan de gestión, análisis, diseño y memoria del proyecto

Spread Your Music

March 2, 2018

Contents

1 Introducción

1.1 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 la música. 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. Tambin incluirá características sociales, permitiendo a los usuarios seguir a sus artistas favoritos para ver las novedades que publican o suscribirse a listas de reproducción creadas por otros usuarios para enterarse de cambios en esta. El sistema tambin 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 tambin 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

2.1 Equipo

El equipo está formado por 7 estudiantes de Ingeniería Informática: Jorge Aznar López, ángel Cañal Muniesa, Abel Chils Trabanco, Alicia Yasmina Albero Escudero, óscar Fraca Ferrández, Alexandru Ioan Oarga Hategan y Jorge Pinilla López.

De estos, cinco poseen conocimientos sobre programación Web (*Frontend*) así como práctica en el desarrollo del *Backend* de una aplicación Web. Por otro lado, todos tienen experiencia en el diseño de base de datos, así como experiencia con la programación sobre la plataforma Android. Además, los integrantes del grupo ya poseen experiencia trabajando juntos, lo cual facilitará la comunicación entre ellos.

Para desarrollar esta aplicación se han creado 3 grupos y cada uno se centrará en una parte del desarrollo. Un equipo se encargará del *Backend*, otro de la interfaz Web (o *Frontend*) y por último otro de la aplicación Android. La asignación de los integrantes del grupo a cada una de las partes se ha realizado en base a la experiencia que posea el integrante en cuestión en dicha área.

El equipo de *Backend* está formado por óscar, ángel y Jorge Aznar. El equipo de la plataforma Android está formado por Abel y Yasmina. Por último, el equipo de *Frontend* está formado por Alexandru y Jorge Pinilla.

Para gestionar el proyecto se ha designado un coordinador general y coordinadores especficos dentro de cada rea. El coordinador general es Abel, el coordinador de *Backend* es ngel, el de Android es Abel y el de *Frontend* es Jorge Pinilla

3 Plan de gestión del proyecto

3.1 Procesos

3.1.1 Procesos de inicio de proyecto

La aplicación móvil funcionará en dispositivos con Android 5.0, la aplicación web funcionará en los navegadores Firefox y Chrome. Para realizar las pruebas de la aplicación móvil se requerirá de un dispositivo con Android 5.0. Por otro lado, para las pruebas de la versión web será necesario un ordenador con el navegador instalado (Chrome o Firefox). El sistema se funcionará sobre un clúster, que contará con un almacenamiento bruto de 20GB. Se ha estimado este tamaño para el clúster teniendo en cuenta la arquitectura del sistema.

3.1.2 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* 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 revisará las tareas que se han realizado esa semana y, si hay tareas que no se han cumplido, se asignarán automáticamente para la siguiente semana, siendo estas tareas las primeras que se deberán hacer. Además, cada semana el responsable del subproyecto 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 subproyecto 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 subproyecto 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.

3.1.3 Procesos técnicos

- Describir los métodos, herramientas y técnicas necesarios tanto para construir el software (p.ej. herramientas de desarrollo), desplegarlo, probarlo (todos los necesarios para dar soporte a los planes descritos en la sección 3.2).

3.2 Planes

3.2.1 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 **snake_case** 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* (https://google.github.io/styleguide/javaguide.html) y el recomendado por la W3Schools para *Frontend* (https://www.w3schools.com/html/html5_syntax.asp). 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 Ángel Cañal.

Este Sistema de Control de Versiones está compuesto por 4 repositorios, uno en el que se alojan todos los documentos del sistema (e.g. ”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 (https://danielkummer.github.io/git-flow-cheatsheet/) 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 envíar 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*).

3.2.2 Plan de construcción y despliegue del software

- Cómo se construye e integra el software: si hay scripts de construcción automatizada o no (en ese caso qué se usa, y cómo se garantiza que todos los participantes compilan igual y con las mismas dependencias), qué se incluye en la construcción (descarga y actualización de dependencias, compilación, ejecución de tests automáticos...) y cada cuánto se construye (compila, integra, prueba) el sistema completo, cómo se configuran los computadores de los desarrolladores.

- Cómo se despliega el software más allá de las máquinas de desarrollo: contenedores, máquinas virtuales, servidor en cloud etc. y cómo se configuran esos entornos (rutas, usuarios y contraseñas, puertos y otros elementos).

3.2.3 Plan de aseguramiento de la calidad

- Estándares de código y otros (se pueden definir guías para la documentación de diseño y otros documentos del proyecto).

- Actividades de control de calidad del código que se realizarán: revisiones de código por pares, revisiones de requisitos o diagramas UML por pares, tipos de tests automáticos o manuales que se llevarán a cabo.

3.2.4 Calendario del proyecto y división del trabajo

- Diagrama de Gantt que recoja las tareas a realizar. Tened en cuenta que trabajáis con dos iteraciones y por tanto que hay una entrega intermedia y una final, y reflejarlo en este diagrama. Tened en cuenta que es normal que lo tengáis que actualizar conforme avance el proyecto (cuándo y cómo establezcáis en la sección 3.1.2).

  - Debe quedar claro qué requisitos van a estar completados en la primera iteración y cuáles en la segunda. Es posible que para la primera iteración no se planifique completar ningún requisito, pero en ese caso tiene que planificarse qué se hará y que faltará por hacer para cada requisito.

- División del trabajo en partes (los módulos del software a desarrollar, pero también la documentación, el diseño gráfico, instalaciones o despliegues, pruebas manuales etc.) y reparto de los mismos entre el equipo de desarrollo, al menos a alto nivel (el reparto de labores concretas en el día a día no se detalla aquí, pero hay que explicar bajo qué criterios y quién/cómo se hace en la sección 3.1.2). Debe haber una correspondencia con las tareas que aparecen en el diagrama de Gantt (que no necesariamente tiene que ser una relación 1 a 1).

  - Verificar que esta división del trabajo cubre todos los requisitos.

4 Análisis y diseño del sistema

4.1 Análisis de requisitos

Completar y detallar los requisitos preliminares incluidos en la propuesta técnica y económica. Recordad que los requisitos deben ser completos, concretos, medibles cuando tenga sentido y lo menos ambiguos posible. También es importante que estén identificados para facilitar su trazabilidad.

4.2 Diseño del sistema

- Diagramas arquitecturales (de módulos, de componentes y conectores, de distribución), patrones de diseño y estilos arquitecturales que se aplicarán. Las interfaces (de módulos y de componentes) son especialmente importantes. También lo son los protocolos de comunicación entre componentes.
- Tecnologías elegidas (lenguajes de programación, componentes que se integrarán, API web externas con las que se conectará etc.).
- Otros aspectos técnicos de interés (p.ej. si hay base de datos si va a ser SQL o NoSQL, si hay una API Web va a ser RESTful o no, si algunas de las operaciones van a ser asíncronas o no, si va a ser una aplicación móvil o de escritorio será nativa o se van a usar tecnologías web, cómo se van a considerar los requisitos de seguridad o de prestaciones, cómo y dónde se harán las instalaciones y despliegues etc.)

Hay que justificar todas las decisiones de diseño. Esto exige contestar a dos preguntas sobre cada decisión: qué alternativas se barajaron? y por qué se eligió una y no las otras?