Mixcrowd, 600 91 11 64, joaquin.gonzalez97@gmail.com+

PROYECTO 6

PLAN DE GESTIÓN, ANÁLISIS, DISEÑO Y MEMORIA DEL PROYECTO

ÍNDICE

1. **Introducción** 3
2. **Organización del proyecto** 3
3. **Plan de gestión del proyecto** 4
   - **Procesos** 4
     - Procesos de inicio del proyecto 4
     - Procesos de ejecución y control del proyecto 5
     - Procesos técnicos 5
   - **Planes** 6
     - Plan de gestión de configuraciones 6
     - Plan de construcción y despliegue del software 8
     - Plan de aseguramiento de la calidad 11
4. **Análisis y diseño del sistema** 16
   - **Análisis de requisitos** 16
   - **Diseño del sistema** 17
     - Arquitectura de la aplicación 17
     - Interfaz de la aplicación 18
     - Mapa de navegación 22
     - Tecnologías elegidas 23
     - Otros aspectos técnicos 24
     - Base de datos 25
     - Diagrama entidad-relación 25
     - Instalación gestor 26
     - Creación de la base de datos 27
     - Poblado de la base de datos 27
5. **Memoria del proyecto** 28
   - **Inicio del proyecto** 28
   - **Ejecución y control del proyecto** 28
   - **Cierre del proyecto** 29
6. **Conclusiones** 31

I. **Anexo I. Glosario** 34

II. **Anexo II. Actas de las reuniones** 35
   - **Acta de la primera reunión** 35
   - **Acta de la segunda reunión** 37
   - **Acta de la tercera reunión** 39
   - **Acta de la cuarta reunión** 41
   - **Acta de la quinta reunión** 42
   - **Acta de la sexta reunión** 43

III. **Anexo III. Diagrama de clases** 44

IV. **Anexo IV. Readme** 46

V. **Anexo V. Horas trabajadas** 48

VI. Anexo VI. Bibliografía 49

1. Introducción

En la sociedad de información abierta en la que vivimos, hay una forma de construir límites al Poder, una sola herramienta: Internet. Las personas pueden colaborar creando proyectos de toda clase proyectos que, tradicionalmente, realizaban empleados o contratistas, dejándolos a cargo de un gran número de personas que conforman una comunidad, a través de una convocatoria abierta. En concreto, en MixCrowd se quiere permitir la creación colaborativa de música en línea.

Tras analizar las diferentes propuestas que se pueden encontrar en la red, se ha podido ver cómo había una parte del mercado sin cubrir entre "los DAWs onlines" (softwares de edición online) y las redes sociales planas para relacionar a los diferentes individuos que participan en el proceso de producción de música, para que luego estos se pusieran de acuerdo por medios de comunicación tradicionales. Por ello, MixCrowd dará todas las herramientas posibles para producir y de red social entre músicos. Sin embargo, se centrará en crear un espacio donde los proyectos sean el pilar fundamental de la comunidad, se pueda colaborar en ellos, sean promocionados y animen por medio de la misma difusión o por concursos a dicha comunidad.

El proyecto, como se explicará más adelante se hará en pocos niveles de despliegue para una comunidad de usuarios limitados. Sin embargo, por medio de buenas prácticas y una metodología ágil, este podrá escalar de una manera adecuada dependiendo del número de visitas.

2. Organización del proyecto

Como en muchas organizaciones informáticas, tras analizar la viabilidad económica de dar nuestro servicio a terceros por medio de una API, el back-end estará oculto del usuario final y solo podrá ser usado por los administradores que se encargará de gestionar el sistema de información.

En concreto, los responsables del back-end serán Osmar, Pedro y Darío. Que abarcará, tanto la puesta en marcha de la base de datos, las máquinas virtuales y los sistemas de ficheros de estas sobre las que funcionará el sistema como del diseño e implementación de los servicios del API web y la lógica del dominio y de la aplicación mediante el micro-framework Flask así como de la interacción con este último con la base de datos con la ayuda de SQLAlchemy y con los sistemas de ficheros con el intérprete Bash y el software libre FFmpeg. Además, este equipo creará contenido para pruebas y para el día de la de la demostración a final de curso.

De la parte de la aplicación que interactúa con los usuarios, es decir, el front-end se encargará Alejandro, Gabriel, Diego, Joaquín, desarrollando cierta lógica de la propia aplicación. La UI y la lógica de la presentación estarán hechas con HTML5, CSS3 y Bootstrap, mientras la lógica de dominio se hará en Javascript y jQuery.js. Aunque se pueda realizar una división por tecnologías como están todas muy realizadas se ha decidido que los miembros del grupo se dividan por pantallas para subdividir el trabajo.

3. Plan de gestión del proyecto

Procesos

En este apartado se va a presentar todo aquello referente a los diferentes procesos involucrados en el desarrollo del sistema, desde su parte inicial hasta las últimas fases de éste.

Procesos de inicio del proyecto

Para la realización de las pruebas del sistema se utilizará, por un lado, una máquina virtual con sistema operativo Debian 4.6.3-14, cuya función será albergar la base de datos. La máquina virtual correrá sobre un equipo Ubuntu 18.04 que actuará como servidor y como almacenamiento del sistema de ficheros. Es decir, habrá un único equipo que contendrá las pistas musicales y se conectará con el cliente, y que además dispondrá de una máquina virtual con la función de actuar como base de datos. Ambos sistemas están en este caso en una misma red local, pero pueden perfectamente estar localizados en distintas redes.

Para redireccionar el tráfico de las peticiones de los clientes al servidor, se va a utilizar una máquina que actúe como forwarding. Debido a que van a existir 2 servidores, el forwarding permitirá que las peticiones se dirijan a aquel de los servidores que tenga menor congestión. Esta solución además permite la escalabilidad, puesto que en el caso de que se quiera incluir un mayor número de servidores para dar acceso a más clientes será posible redirigir el tráfico a cada uno de ellos mediante el forwarding. La máquina que va a actuar como forwarding es una RaspberryPi3.

Por último, se simularán los clientes sobre las propias máquinas de los integrantes del grupo, realizando peticiones al servidor desde diferentes sistemas operativos y navegadores distintos: Google Chrome en Windows 10, Safari en MacOS y Mozilla Firefox en Ubuntu.

En cuanto a las herramientas utilizadas, algunas de ellas son de uso libre y otras requieren un registro para su utilización.

Se ha estimado conveniente dividir el desarrollo del proyecto en varios apartados:

- Base de datos: lo concerniente a la base de datos va a correr a cargo de Osmar Alí, que estudia la rama de Sistemas de información y ha trabajado en varios proyectos de desarrollo de bases de datos;

- Front-End: en cuanto al desarrollo del front-end, éste va a comprender varias herramientas, HTML5, CSS3, Javascript, Bootstrap y jQuery. Esta tarea será realizada por Alejandro, Diego, Gabriel y Joaquín. Ninguno de ellos cuenta con experiencia en desarrollo de front-end web, salvo Diego, así que será necesaria una cierta autoformación y documentación por su parte, para poder llevar a cabo el trabajo.

- Back-End: por último el trabajo de back-end será realizado por Pedro y Darío. Será necesario para ambos, al igual que en el caso anterior, un trabajo de documentación y autoformación, que permita el dominio de las tecnologías a utilizar, las cuales son Flask, FFmpeg (una herramienta que permitirá realizar las diferentes modificaciones en los archivos de audio) y por último la gestión del sistema de ficheros, que almacena la información de las pistas musicales.

Procesos de ejecución y control del proyecto

La comunicación entre los miembros del grupo se lleva a cabo principalmente mediante la aplicación de mensajería Whatsapp, a través de un grupo creado expresamente con este propósito.

Por otra parte, en cuanto a las reuniones, además de aquellas pactadas con los profesores, se llevan a cabo otras que atañen únicamente a los miembros del grupo, en las que se pone en común el trabajo realizado y se discuten las diferentes propuestas y estrategias a llevar a cabo. En todas las reuniones se designa a un miembro del grupo para que tome nota de todo aquello tratado, plasmándolo en un acta. Esta función le corresponde en cada reunión a un miembro diferente del grupo.

Para la resolución de conflictos, se ha estimado oportuno que si surgen tales situaciones estas deben ser solucionadas de forma rápida en presencia del director del proyecto, Pedro, para que no se prolonguen en el tiempo y puedan retrasar el proyecto.

Para registrar el avance del proyecto a lo largo del tiempo se ha planeado realizar semanalmente una puesta en común de todo lo avanzado hasta el momento, que además implicará la realización de pruebas específicas de cada una de las partes del software desarrollado. Una vez se haya alcanzado el momento en el que alguno de los subsistemas se pueda integrar, se procederá a la realización de pruebas conjuntas, que se ajusten en mayor medida al funcionamiento final del sistema, hasta que finalmente, todos los subsistemas estén correctamente desarrollados y puedan integrarse por completo dando lugar al sistema definitivo. En caso de que ocurran errores en la puesta en común, estos deberán ser subsanados con la mayor brevedad posible, preferiblemente antes del final de la semana, para que el desarrollo pueda continuar con el progreso previsto. Por otra parte de esta manera, si se detectan retrasos en el proceso de diseño se añadirá una nueva puesta en común por semana, haciendo un total de 2, hasta que vuelva a recuperarse el ritmo normal de desarrollo establecido.

Para la entrega de resultados, se hará al cliente una demostración del sistema desplegado y funcionando, que permita visualizar el correcto funcionamiento de todas sus características. Esta demostración consistirá en testeo completo de las funcionalidades ofrecidas por la aplicación. Además, al cliente se le entregarán todos los archivos necesarios para que pueda disfrutar del comportamiento del sistema que se le ha mostrado; no se le entregará ningún archivo de código fuente, binarios o cualquier archivo relativo a la implementación del sistema.

Procesos técnicos

Para la realización del back-end se utilizará el IDE Psycharm, que integra el framework web Flask; por otra parte, para el front-end, se utilizará el IDE Webstorm, que integra los lenguajes de diseño web CSS, HTML y Javascript.

Para la ejecución de los scripts necesarios para la puesta en marcha del sistema se utilizará el intérprete de comandos Bash, y para la edición en general de código el editor de texto Sublime. Por último, además se utilizarán los navegadores nombrados anteriormente para las pruebas desde el punto de vista del cliente.

Planes

Plan de gestión de configuraciones

Convenciones y estándares

En el proyecto se seguirán las siguientes convenciones sobre nombres y estándares para la gestión de las configuraciones, asegurando el entendimiento del código y la cooperación entre los distintos miembros del grupo independientemente de sus responsabilidades y tareas asignadas:

- El código deberá de ser comprensible y entendible. Para ello se harán uso de nombres de funciones y variables que den información sobre su significado y uso dentro del código. De esta manera, se asegura que, con una lectura rápida se podrá comprender lo que se quiere conseguir en la función o sentencia del código.

- Los nombres de funciones y variables serán lo más cortos posibles (no superior a 20 caracteres de ser posible). Así se asegura mayor legibilidad y mayor rapidez a la hora de poder desarrollar código (rapidez en la escritura). Del mismo modo, no se acortarán si la abreviatura no es entendible ni legible ni identificable.

- El código deberá atender a las normas del formato de codificación UTF-8, sin embargo se escribirá el código mediante nombres de variables y funciones usando expresiones y frases, así como la documentación, en español, puesto que es el idioma predominante dentro del grupo de desarrolladores. Sin embargo, puede darse la posibilidad de que exista algún elemento en inglés en caso de código ya existente probado y válido y de cierta complejidad con el objetivo de hallar más fácil la fuente original y evitar errores posteriores de traducción.

- La documentación ha de ser concisa, clara y dar pistas o indicar lo que la función o sentencia del código en cuestión intenta conseguir. Deberá de realizarse de forma informal, no mediante notación científica. Así simplificando el entendimiento del código.

Todo lo realizado en este aspecto, será exclusivamente para facilitar el trabajo de los desarrolladores puesto que ni el cliente ni nadie más podrá acceder o tener el código fuente.

Responsables

Coordinación de los grupos de trabajo: Joaquín González Oller

Control de versiones y copias: Diego Santolaya Martinez

Actas y reuniones con clientes: Osmar Ali De la Fuente Maicas

Apoyo y supervisión de trabajos: Darío Ferrer Chueca y Pedro Ramoneda Franco

Evaluación y pruebas: Alejandro Francés Rubio y Gabriel Garcia Remirez de Ganuza

Recursos

Para el desarrollo del proyecto se hará uso del controlador de versiones Git. A través de la plataforma Github se podrá supervisar, ver y guardar las distintas versiones que se realizan durante el proceso de construcción, despliegue y mejora del proyecto.

Además dicha herramienta, Git, es robusta y asegura que todo cambio quede guardado y las versiones puedan recuperarse en cualquier momento. En un inicio habrá un único repositorio para el grupo de desarrolladores. Sin embargo cada uno trabajara sobre su parte asignada.

Todos los cambios se guardarán en dicho repositorio. De esta forma se asegura que el proyecto o sistema en su conjunto se encuentra en un único lugar guardado de forma segura. Y en caso de necesitar, cada uno podrá recuperar las versiones sobre las partes en las que trabaja.

Todos los miembros tendrán permisos de todo tipo sobre el repositorio, siendo colaboradores de este mismo. Los cambios y subidas se realizarán cuando se quiera una opinión o testeo por parte de los demás desarrolladores, así como para guardar la versión definitiva y correcta.

Una vez el trabajo de desarrollo termine el repositorio será el lugar donde se almacene el sistema entero con el código fuente de los distintos componentes del sistema software para su construcción y despliegue.

Procedimiento para realizar cambios

El control de versiones será realizado de forma automática mediante Git. Este mismo será el encargado de guardar las modificaciones que se realicen.

Será el desarrollador el encargado de tomar una práctica responsable y realizar commits cuando se considere que se han realizado modificaciones lo suficientemente importantes como para que dicha versión se guarde.

Los commits se realizarán por lo tanto basados en la opinión de cada uno de los miembros del equipo. Siendo cada uno responsable de sus propias partes y responsables de guardar las versiones. El propio controlador de versiones Git será el encargado de poner a disposición del desarrollador las distintas versiones realizadas.

Sin embargo en todo momento el usuario de Git deberá de controlar lo que está realizando y los comandos que necesita para llevar a cabo sus tareas de desarrollador: recuperación y guardado de versiones. Puesto que un mal uso puede dificultar o provocar la inaccesibilidad a las versiones guardadas. No obstante una persona será la encargada de supervisar las distintas acciones realizadas por cada desarrollador.

Sin embargo no será necesaria una aceptación previa sino que está sometido a criterio individual. Las incidencias y actividades realizadas, el propio sistema gestor Git será el que las anote y el supervisor podrá comprobarlas.

Al repositorio compartido solo se subirán aquellas versiones que pasen los tests iniciales y se consideren que ya se pueden incorporar al conjunto del sistema final y que asegura el correcto funcionamiento del proyecto.

Plan de construcción y despliegue del software

Construcción e integración

El proyecto o sistema que se desarrolle se construirá e integrará a través del uso de scripts, de construcción automatizada, los cuales permitan a los desarrolladores el desplegar el sistema sin esfuerzo alguno. Además así se garantiza que todos los participantes compilen y ejecuten con las mismas dependencias.

Estos scripts pueden ser usados para cada una de las partes de forma individual o para el conjunto del sistema. No obstante cada usuario también es libre de compilar y ejecutar mediante línea de comandos, pero en este caso deberá de comunicar el procedimiento a seguir a los distintos miembros para que ellos mismos también puedan hacerlo. Siendo necesario así que las instrucciones estén por escrito. Este método en caso de que automatizar suponga un esfuerzo o una desventaja mayor que la forma manual.

La construcción por lo tanto necesitará de una compilación en una primera instancia. Posteriormente solo con la ejecución de los archivos ya generados en la compilación es suficiente. Para ello se harán scripts los cuales aúnan las distintas partes y sea un único fichero o varios los que ejecuten las distintas partes. Ante cualquier mejora u optimización será necesaria una nueva compilación.

El sistema completo no se construirá hasta que todas las partes se hayan probado por individual, cada parte deberá asegurar un mínimo de funcionalidad. De este modo se consigue seguir un modelo incremental que aísla las partes y luego las va juntando una por una para asegurar que la construcción del sistema se realiza de forma correcta y en el menor número de dificultades.

De esta forma primero se construirán las distintas partes y se asegurará su funcionamiento, luego se irán acoplando entre ellas asegurando que cada parte que se acopla se comunica con las demás de forma correcta. Así hasta juntar el proyecto al completo y tras lo cual se realizarán las pruebas necesarias para determinar que el sistema es robusto y no genera problemas de funcionalidad.

Los computadores personales de los desarrolladores se configuran de la manera que cada individuo considere oportuna no obstante atendiendo a que pueden surgir problemas de compatibilidad por lo que se tratará de optar por aquella configuración más genérica. Además, deberá de comunicarse y estar atento a las acciones de los distintos miembros.

No obstante las herramientas a desarrollar se han mirado y analizado para que se puedan trabajar de forma individual, de tal manera que el problema de compatibilidad no pueda surgir entre las distintas partes ni entre desarrolladores que trabajan en la misma tarea.

Despliegue Software

El software, puesto que se puede desplegar en distintos componentes hardware dependiendo de la demanda a la que el servicio pretenda atender, se explicará su despliegue basando en una sencilla demo.

Esta demo pretende mostrar un sistema sencillo y funcional, así como un esquema escalable de despliegue y construcción. Con esto, se muestra al cliente que cualquier alteración al hardware que se realice no altera la funcionalidad del sistema y que cualquier componente puede ser usada. Es decir, software y arquitectura hardware son independientes, una no depende de la otra.

En la demo como ya se ha explicado se seguirá el siguiente esquema. Los clientes se

conectarán a un único servidor de forwarding el cual dirigirá el tráfico a los distintos servidores, los cuales serán los que a su vez se comuniquen con la base de datos y el sistema de ficheros.

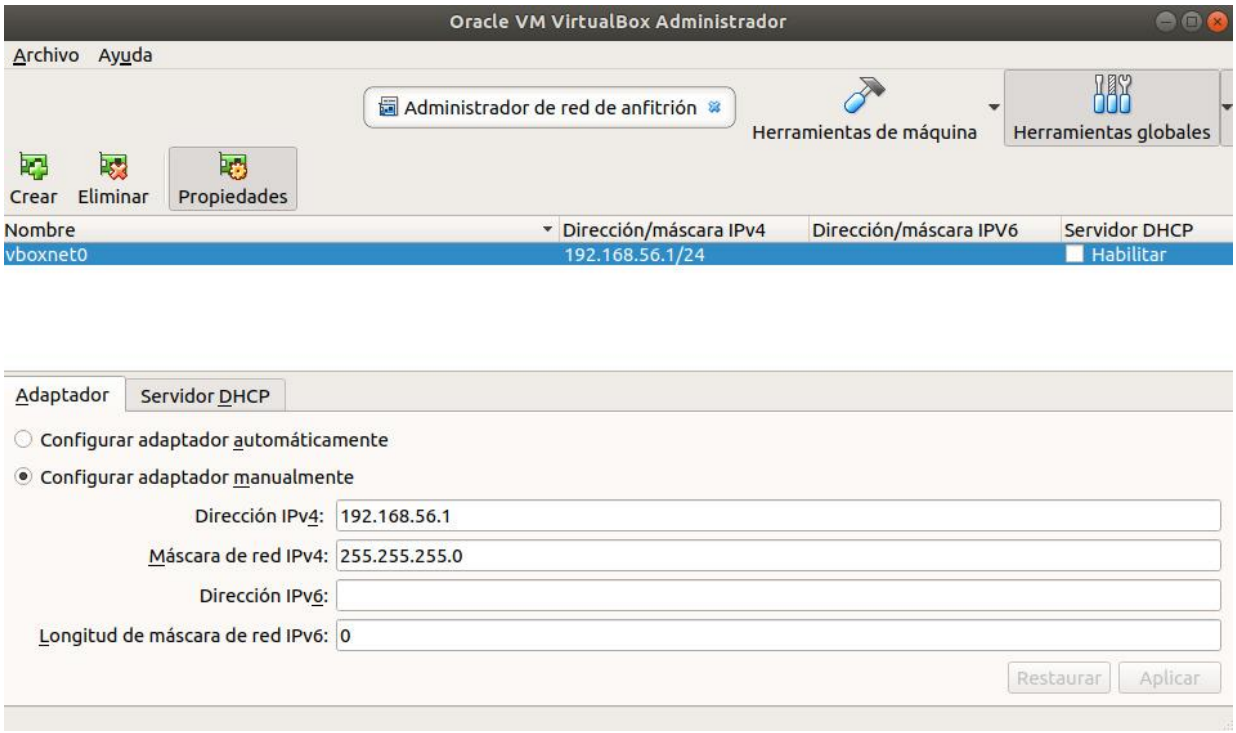
Por lo que en este caso en teoría se dispondría en la arquitectura hardware de un servidor de forwarding, varios servidores y una única base de datos y sistema de ficheros.

Por lo que en la demo se emplea:

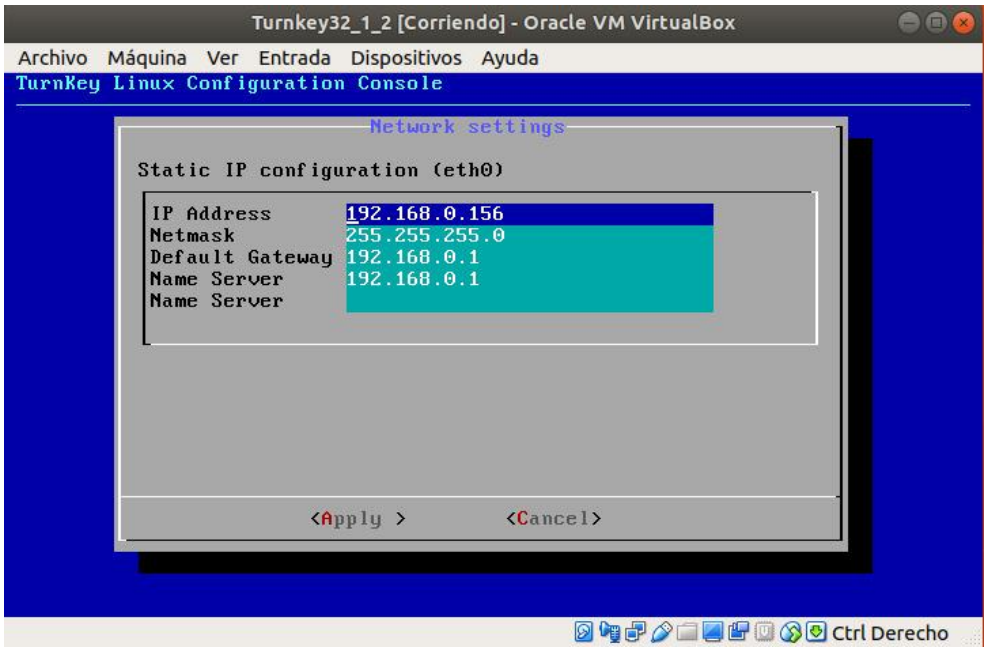
- La raspberry donde se desplegará el software referido a la funcionalidad del servidor de forwarding.

- Varias máquinas virtuales, en concreto 2 que sirvan como servidores. Con lo cual en ellas se desplegará todo el software relativo a la mezcla de pistas y la comunicación entre la base de datos, ellos y el servidor de forwarding.

- Una máquina física donde se guardará el sistema de ficheros y la base de datos. En ella el software desplegado será de mantenimientos y gestión de los datos almacenados así como la comunicación con los servidores.



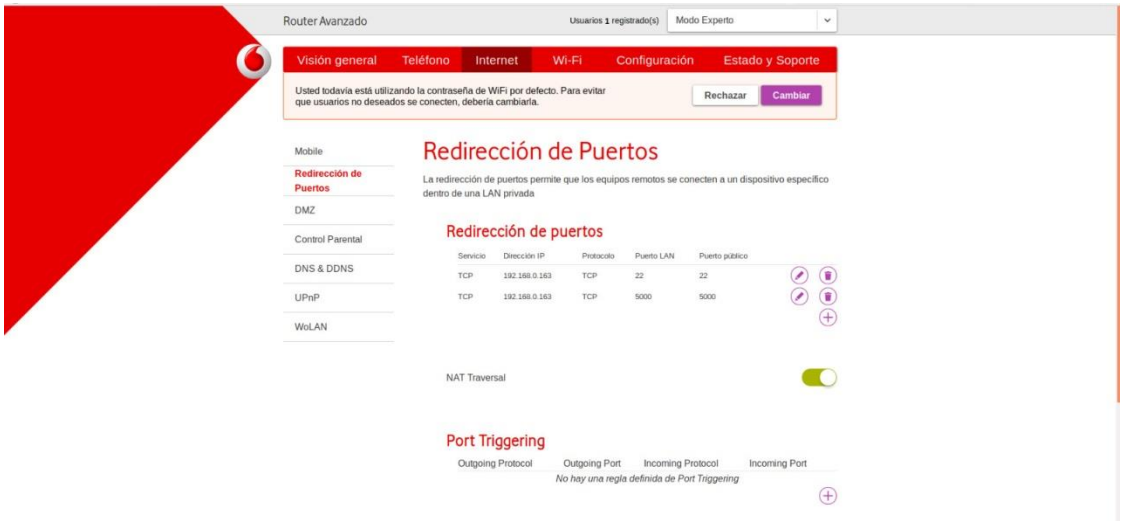
1. Ejemplo de una posible configuración de la máquina virtual



2. Ejemplo de la configuración de la máquina virtual

Sin embargo, por falta de recursos hardware para el proyecto se ha optado por montar un único servidor sin forwarding para realizar la demo. Toda esta arquitectura está descrita en el apartado “Arquitectura de la aplicación” en “Diseño del sistema” del punto 4 “Análisis y diseño del sistema”.

Todo esto es por simplificar el sistema a desarrollar. El cliente puede utilizar otro hardware sin problemas: por ejemplo usar máquinas individuales y físicas para cada servidor puesto que considera que la demanda va a ser superior de la que se podría manejar con la virtualización en una única máquina física.



3. Redirección de puertos del router necesaria para una máquina virtual

La configuración para la comunicación entre las distintas maquinas será mediante la selección de puertos. Las maquinas operarán en puertos designados con prioridad por los cuales llegarán y enviarán las distintas peticiones. Así como el sistema de ficheros se

gestionará mediante rutas. Puesto que los ficheros de audio se encontrarán en distintos directorios o subdirectorios dentro del sistema.

En cuanto a la app web, la parte de interacción con el cliente esta estará gestionada por permisos los cuales se analizarán antes de realizar una acción. Puesto que un usuario puede no tener los permisos necesarios para realizar tal acción como por ejemplo: un usuario no administrador intentar borrar una pista de un proyecto.

Además el acceso a distintos proyectos, los de carácter privado se realizará de forma automática mediante el identificativo de los usuarios, el cual es de carácter único.

Todo este proceso está más descrito y mejor detallado en el README de Github, cuya referencia está en el anexo Bibliografía a la carpeta que lo contiene, con todos los documentos necesarios.

Plan de aseguramiento de la calidad

Las siguientes pautas serán seguidas por los distintos desarrolladores del grupo para asegurar la calidad y eficiencia del proyecto o sistema a desarrollar:

- El uso de bibliotecas y funciones de librerías ya desarrolladas dentro de los lenguajes que se empleen. Es decir, los desarrolladores deberán consultar y hacer uso de los recursos ya implementados por los propios desarrolladores de los lenguajes en los que se trabaje. De esta manera aseguramos el uso de herramientas de gran eficiencia, así como la simpleza y lectura del código que se desarrolla.

- Las anotaciones o documentación del proyecto deberán aparecer al menos como cabecera de línea en aquellas que se consideren complejas de entender en la codificación empleada, así como para todas las funciones empleadas. Así se asegura el entendimiento de las tareas que se desarrollen por el resto de desarrolladores que puedan trabajar en la misma tarea o en otras.

- Todo código a usar deberá de atender a los siguientes principios: simpleza y eficiencia. El código que se desarrolle deberá de ser lo más eficiente posible. Por lo que se intentará siempre pensar antes en qué estructuras, métodos, o algoritmos son los mejores. Atendiendo al coste en memoria y en tiempo que estas opciones puedan suponer, así como la complejidad. Entre métodos de misma eficiencia siempre se elegirá el de más fácil comprensión e implementación.

- Automatizar procesos. Todo proceso que se pueda realizar de forma automática así se hará. De esta forma se asegura que el sistema en la mayor medida pueda estar exento de fallos puntuales por parte del componente humano, así como asegurar el rápido y sencillo despliegue del sistema al cliente.

- Garantizar recuperación ante errores. Valorar todos los aspectos posibles o sucesos que puedan tomar lugar y asegurar que el sistema da una correcta respuesta a estos.

Actividades de control

Todo desarrollador deberá seguir las siguientes normas para garantizar un mínimo control

de calidad del código que se realiza:

- Todo desarrollador deberá, antes de integrar al conjunto del proyecto su parte, comprobar que esta funciona correctamente. Asegurando que funciona primero de forma independiente de las otras piezas del sistema. Las pruebas deberán de realizarse en la medida de lo posible simulando que se encuentra de forma conjunta al resto del sistema.

- Una vez se pruebe su funcionamiento deberá de supervisar su código y ver con otra persona si este puede o no ser mejorado. En caso afirmativo realizará las mejoras, y lo guardará con el resto del proyecto para que el resto de desarrolladores pueda verlo y opinar. En caso negativo directamente lo guardará.

- Antes de realizar ningún código, el desarrollador será consciente de los requisitos a cumplir manteniendo estos en su cabeza cuando escriba el código y revisando . Los requisitos estarán a disposición de todos los miembros en caso de que sea necesario consultarlos.

- Los desarrolladores buscarán cumplir los requisitos de forma eficiente y simple, pero sin pasarse. Hay que tener en cuenta, que solo se han de cumplir las condiciones pactadas, sólo estas y nada más.

- Cada uno de los desarrolladores, anterior a la implementación, deberá desarrollar y esquematizar en pseudocódigo lo que implementa en el posterior código

- Todos los tests serán automáticos a poder ser para que el resto de miembros puedan ver el correcto funcionamiento de lo desarrollado. Si fueran manuales, en este caso cada miembro notificará al desarrollador las pruebas realizadas, así como lo que necesita modificar. El desarrollador también notificará las pruebas manuales realizadas a los demás para que así estos puedan ver, que en efecto, el sistema se comporta como debe.

Todas estas pautas están diseñadas para asegurar la calidad de lo realizado. Y asegurar un proceso que dificulte lo menos posible el desarrollo proyecto.

Calendario del proyecto y división del trabajo

En este apartado se presenta el diagrama de Gantt que recoge toda la información de planificación de las tareas a realizar. Se ha tratado de representar prácticamente desde el inicio de la asignatura, para tener presente en todo momento en que parte de la planificación estamos y si vamos según el planning.

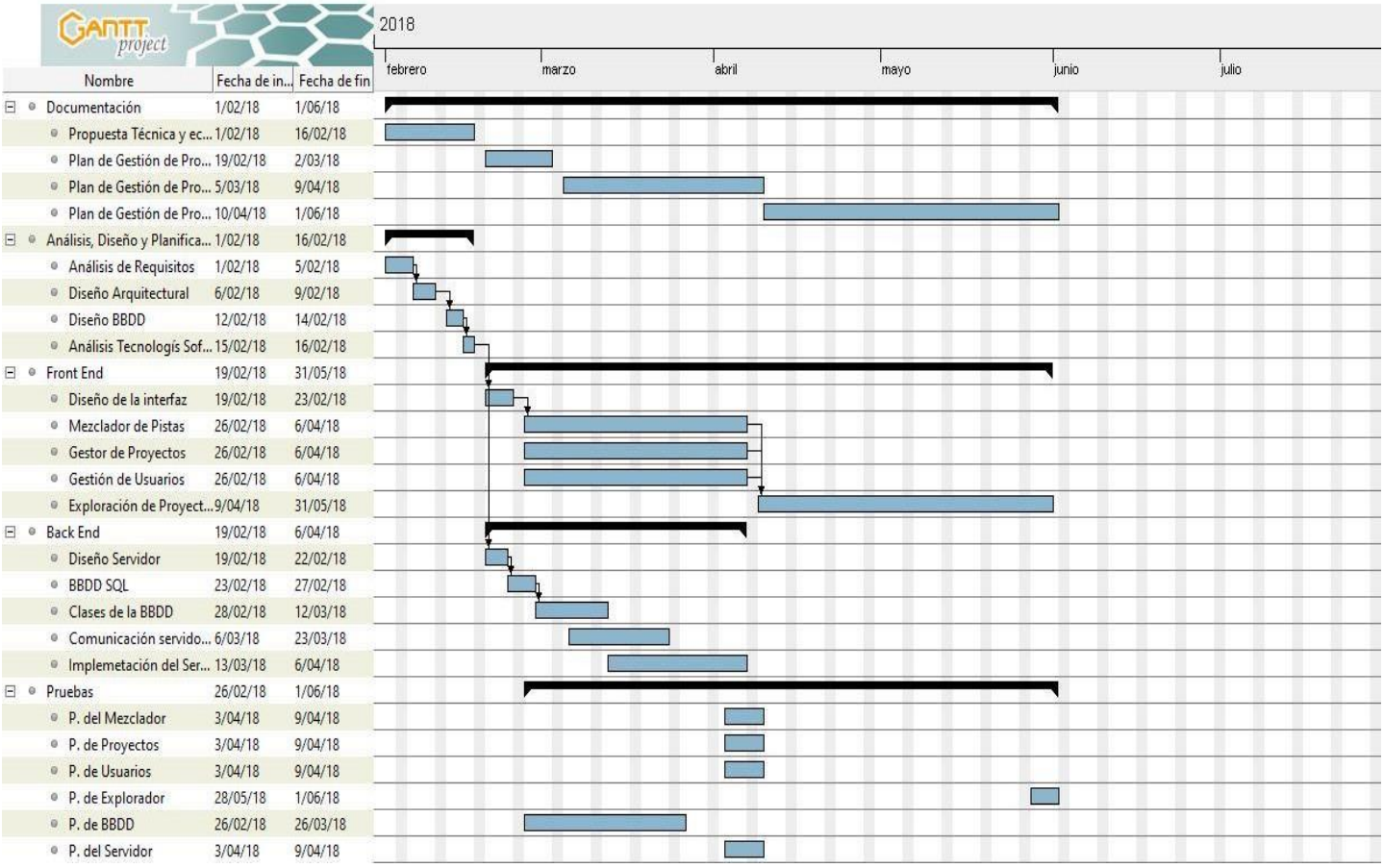


Figura: Diagrama de Gantt general.

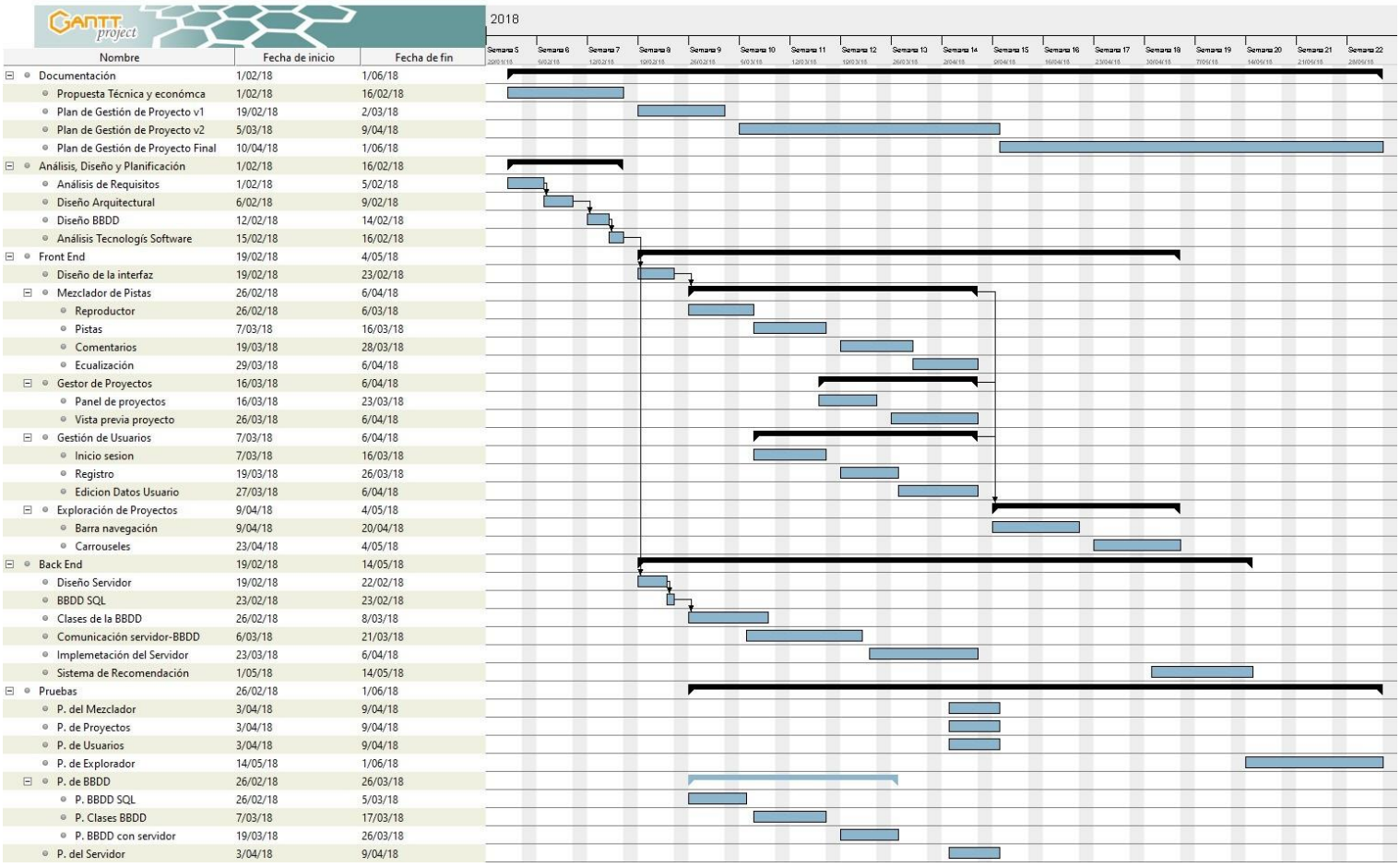


Figura: Diagrama de Gantt desglosado.

Como se aprecia en el diagrama, la idea es que para la primera iteración se debería de tener acabado todo el sistema de datos y el servidor, además del mezclador de pistas, el gestor de proyectos y el gestor de usuarios. De esta forma, dispondremos de tiempo suficiente en la segunda iteración para hacer el sistema de exploración correctamente y corregir posibles errores detectados en la primera iteración.

La forma en la que se va a repartir el trabajo es por áreas de trabajo. Se ha dividido principalmente el proyecto en front-end y back-end, a los cuales se han asignado a 4 y a 3 miembros del grupo respectivamente. Para el back-end, los tres miembros del grupo se encargarán de diseñar el servidor, de implementarlo y de hacerles las pruebas pertinentes, pudiéndose organizar el trabajo internamente como ellos mejor consideren. Para el front-end, al haber cuatro personas y tres aspectos clave que zanjar en la primera iteración, dos se encargaran del mezclador de pistas, funcionalidad principal de nuestro sistema, mientras que los otros dos se encargaran uno de la gestión de proyectos y otro de la de usuarios. Si algún módulo termina antes de lo previsto, deberá de involucrarse con otro miembro para finalizar su tarea más rápido.

Tras una puesta en común para tomar las decisiones más importantes, cada miembro es responsable del diseño e implementación de su parte correspondiente, además de las pruebas que se consideren oportunas para validar su correcto funcionamiento. Es posible que se realicen más puestas en común (bien por vía oral o por vía digital) para poder resolver dudas o conflictos.

También, cada miembro del grupo es responsable de llevar al día la documentación detallada de su tarea para posteriormente, un responsable de documentación pueda juntar todo para organizar y redactar la memoria final.

Siguiendo esta distribución del trabajo y la planificación de tareas anteriormente mencionada, se cumplirán todos los requisitos de la aplicación, mientras que si alguna tarea queda pendiente, es posible que algún requisito no se cumpla. Por esto es de fundamental importancia seguir nuestro planning para la correcta realización del proyecto.

4. Análisis y diseño del sistema

Análisis de requisitos

A continuación, se presentan los requisitos funcionales agrupados por la funcionalidad a la que hace referencia.

- Usuarios:

1. El sistema debe permitir el registro de un nuevo usuario, introduciendo los datos personales.
2. El sistema identifica a cada usuario por su nombre de usuario.
3. Para la creación de proyectos y las recomendaciones personalizadas será necesario el registro en el sistema.
4. En caso de que el usuario no esté registrado únicamente podrá ver diferentes proyectos públicos que hayan sido creados por otros usuarios.
5. El sistema debe permitir el login de un usuario existente. Una vez registrado, será necesario realizar el login para entrar al sistema y utilizar todas sus funciones.
6. El sistema cuenta con un sistema de recomendaciones para cada usuario.
7. La aplicación cuenta con un sistema de "amistad" entre usuarios, que permite a cada uno de estos contar con una lista de personas que pueden colaborar en todos sus proyectos.

- Proyectos:

1. El sistema debe permitir la creación de un nuevo proyecto, estableciendo un título, una descripción y una imagen para el proyecto.
2. El sistema identifica cada proyecto por su título, descripción e imagen.
3. El sistema va reconocer dos tipos de usuarios que acceden a un proyecto, el administrador que es el usuario que ha creado el proyecto y el colaborador que es el usuario que participa en el proyecto añadiendo nuevas pistas de música.
4. El sistema debe permitir al administrador establecer un proyecto como público o privado.
5. El sistema debe permitir al administrador y los colaboradores añadir una pista a un proyecto.
6. El sistema debe permitir al administrador eliminar una pista de un proyecto.
7. El sistema debe permitir al administrador y a los colaboradores mezclar varias pistas de un proyecto.
8. El sistema debe permitir al administrador y a los colaboradores descargar la mezcla resultante de un proyecto.
9. El sistema debe permitir a los colaboradores realizar comentarios en un proyecto.
10. El sistema debe permitir que los usuarios establezcan una valoración de un proyecto.

- Búsqueda:

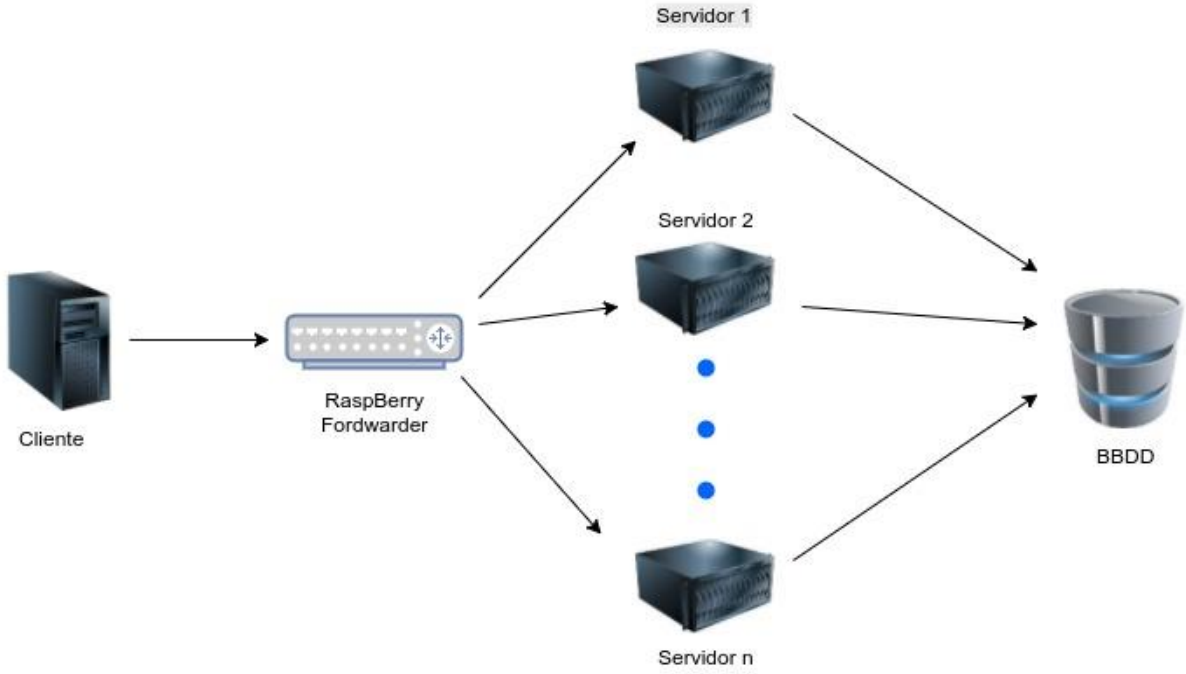
1. El sistema permite la búsqueda de un proyecto concreto ya sea indicando la categoría a la que pertenece, el nombre del proyecto o el usuario que lo ha creado.
2. El sistema permite la búsqueda de un usuario concreto indicando el nombre de usuario.

- Interfaz:

1. La interfaz gráfica ha de tener un diseño responsable, es decir, que al variar el tamaño de la pantalla se equilibren los diferentes componentes de la interfaz.
2. El sistema debe permitir al marcar una pista para mezclar seleccionar la distribución de la señal de sonido en la panorámica del estéreo
3. El usuario debe poder elegir a partir de que determinado momento va a sonar su pista dentro de la mezcla general. Este parámetro puede ser cambiado por Administrador del proyecto.

Diseño del sistema

Arquitectura de la aplicación



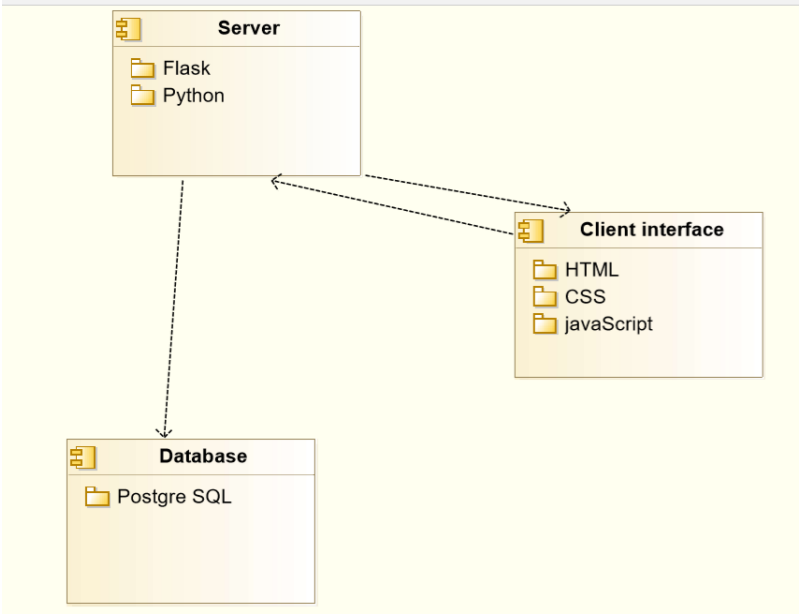
En el sistema pensado para entregarse al cliente con escalabilidad (no implementado por falta de recursos físicos) hay 4 partes bien diferenciadas. En primer lugar, está la interfaz de la aplicación. En segundo lugar, hay una raspberry forwarder cuya función es dirigir el tráfico de cada usuario a un servidor. El servidor asignado será aquel que tenga menor carga de trabajo.

En tercer lugar, están los servidores. Estos se encargan de devolver las peticiones y los datos pedidos por los usuarios. Finalmente, se tiene otra máquina en la cual se monta la base de datos y paralelamente un sistema de ficheros que almacena las pistas de audio del sistema.

Esta máquina se encarga de suministrar la información almacenada a los servidores.

En la demo real y final del sistema que se ha desarrollado se cuenta con 3 partes bien diferenciadas. En primer lugar, está interfaz de la aplicación web/móvil. En segundo lugar está el servidor , que se encarga de devolver las peticiones y los datos pedidos por los usuarios. Finalmente, se tiene otra máquina (de la misma forma que en la arquitectura ideal) en la cual se monta la base de datos y paralelamente un sistema de ficheros que almacena las pistas de audio del sistema.

Esta máquina se encarga de suministrar la información almacenada a los servidores.

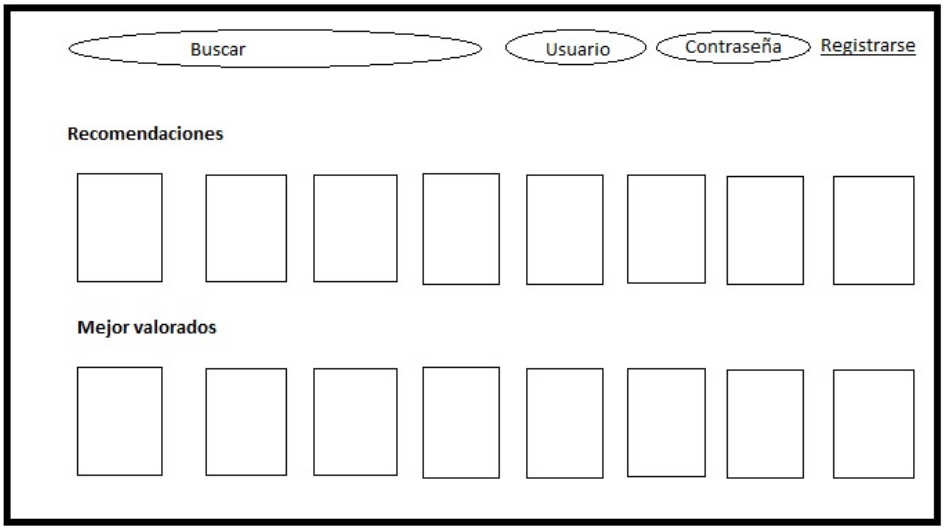


4. Diagrama de componentes

La misma imagen se ilustra en un diagrama de componentes mostrando las distintas partes involucradas en el sistema.

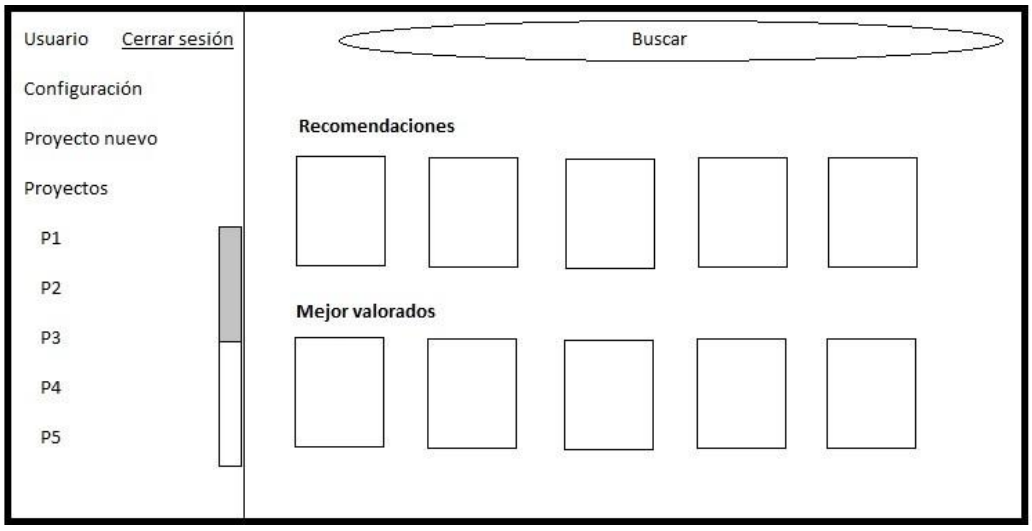
Interfaz de la aplicación

La aplicación web constará de varias páginas: una portada o home inicial con las recomendaciones del programa a cualquier usuario o personalizadas al usuario que está logueado en caso de estarlo; una de registro donde se pedirán todos los datos necesarios; una de crear un proyecto nuevo; y variaciones de una página con un proyecto abierto según la relación que tenga el usuario con dicho proyecto.

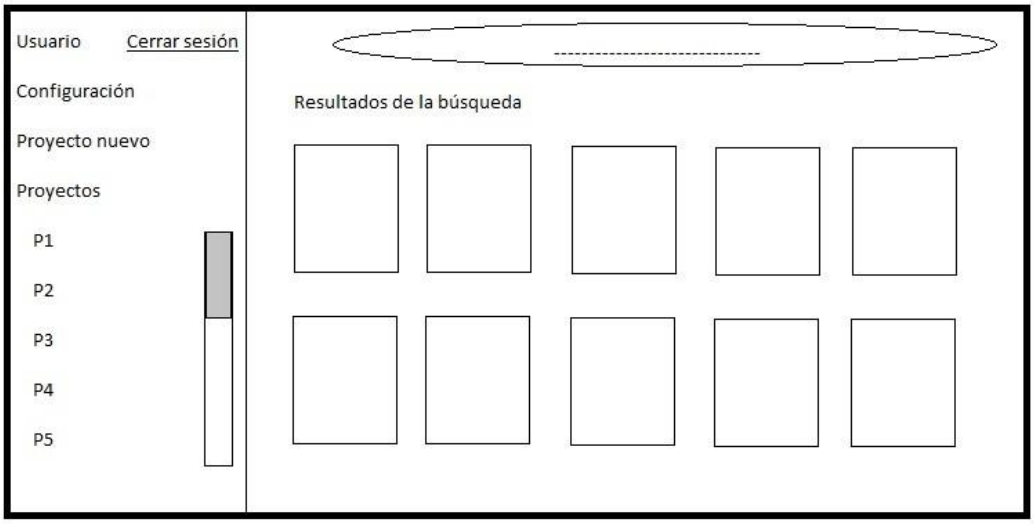


5. Boceto de la página principal

6. Boceto de la página de registro

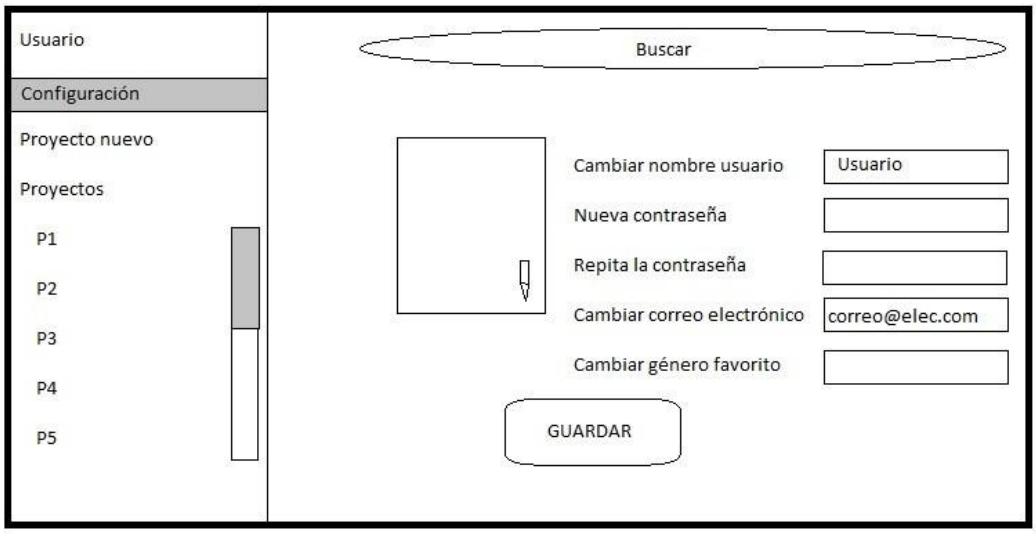


7. Boceto la página principal de un usuario



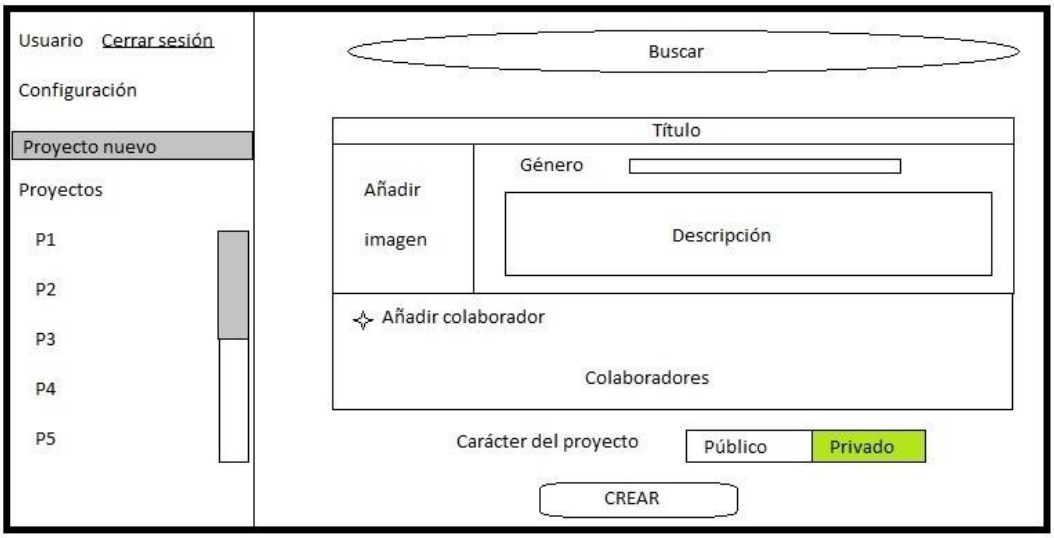
8. Boceto del resultado de una búsqueda

Esta página tiene dos variaciones; la mostrada que es una búsqueda estando logueado como usuario, y una segunda que es no estando logueado como usuario y que únicamente se diferencian entre sí por la columna de la izquierda donde aparece el usuario, la configuración…



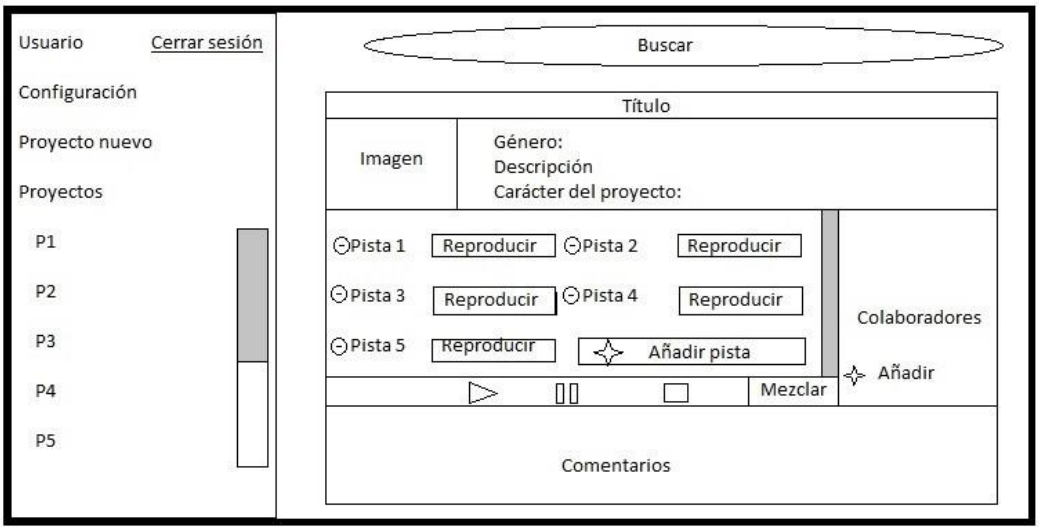
9. Boceto de la página de configuración

Esta página permite cambiar información personal.



10. Boceto de la página de creación de un proyecto nuevo

En esta página se crea un proyecto al que será obligatorio enmarcarlo en un género, ponerle un título y dotarle de carácter público o privado.



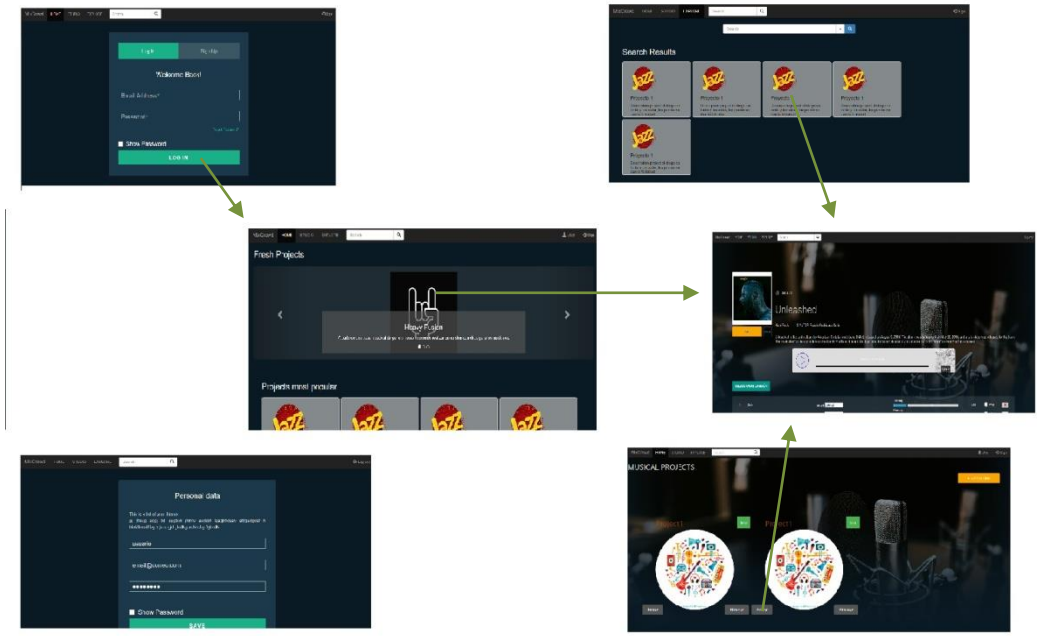
11. Boceto de la página con un proyecto abierto siendo colaborador

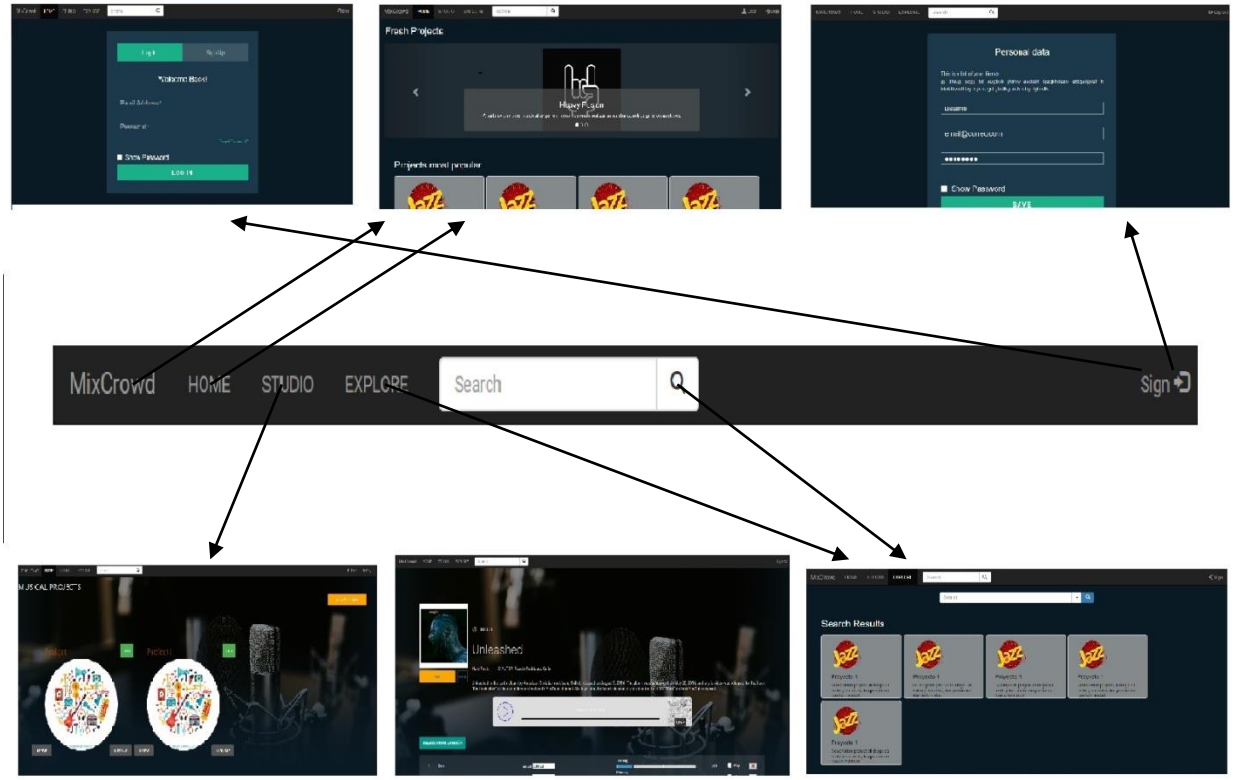
Esta página sufre pequeñas variaciones según la relación que haya entre el usuario y el proyecto. Las opciones de añadir pista, añadir colaborador y mezclar están a disposición de los colaboradores, al igual que la de eliminar pistas. Si se es administrador del proyecto, se puede además cambiar la imagen, el título, el género, la descripción y el carácter del proyecto. Estas variaciones se pueden ver representadas con un símbolo de edición un lugar cercano a este, un lápiz al final del título, por ejemplo, que aparecerá en esa pantalla. Las opciones del administrador de hacer colaboradores administradores y eliminar colaboradores estarán representadas con sendos símbolos, uno que indique que se hace a tal colaborador administrador y otro que se le elimina. En caso de no ser ni administrador ni colaborador, solo estará disponible la opción de reproducir, y si se está logueado también la de añadir pistas. El botón de reproducir puede estar marcado para distintas pistas a la vez. En ese caso y a la hora de darle al botón de play, se reproducirán todas las pistas seleccionadas a la vez.

Todas estas páginas interactúan entre sí de cierta manera:

1. Al darle al botón de registrarse en la primera imagen, se mostrará la segunda imagen por pantalla;
2. Al darle al introducir el usuario y la contraseña correctamente en la primera imagen, la pantalla será sustituida por la tercera imagen;
3. Al introducir y realizar una búsqueda en cualquier pantalla se mostrará la cuarta imagen acorde a si el usuario está logueado o no;
4. Al clicar en la configuración del usuario, donde esté disponible, se cambiará a la quinta imagen;
5. Al darle a “Proyecto nuevo”, se pasará a la penúltima imagen;
6. Al cerrar sesión, se cambiará a la primera imagen
7. Al abrir un proyecto recomendado, resultado de una búsqueda o en el que se participa, se mostrará la última imagen por pantalla de acuerdo a la relación entre el usuario y el proyecto abierto.

Mapa de navegación





La primera imagen, revela como se relacionan las interfaces entre sí clicando en elementos propios no comunes entre ellas. Mientras, en la segunda imagen se muestra la relación entre las interfaces con la barra de navegación.

Como vemos, en la primera imagen hay una interfaz que no se relaciona con el resto. Sin embargo, a esta misma interfaz se llega desde la barra de navegación estando logueado. En caso de no estarlo, se irá a la página de login. Hay que tener en cuenta que todas las interfaces son dinámicas y que por lo tanto tanto la barra de navegación como la interfaz sufrirán cambios si se está logueado o no y entre usuarios con los proyectos que se mostrarán, entre otros elementos.

Tecnologías elegidas

La aplicación está compuesta en varios lenguajes de programación. El front-end tendrá como base Javascript, CSS3, HTML5, jQuery y Bootstrap. Todas ellas son tecnologías muy usadas hoy en día y con mucha documentación. Además, hay múltiples ejemplos de su uso de donde se pueden obtener una parte de código que podemos reutilizar, ya sea por su sencillez a la hora de resolver un problema o una estructura similar a la nuestra. Un motivo extra por el que hemos elegido estas tecnologías es porque un miembro ya sabe emplearlas y puede guiar al resto en el proceso de autoformación de esta nueva tecnología nueva.

El back-end estará compuesto en Python principalmente, aunque se hará uso de comandos para poder conectarnos con la API FFmpeg. FFmpeg es un software libre que usaremos para mezclar las pistas que los usuarios deseen y obtener la resultante. Por otro lado, usaremos SQLAlchemy para poder interactuar con la base de datos SQL. Dicha base de datos se gestionará con PostgreSQL, el cual es un software libre que permite trabajar con bases de datos en SQL. Hemos optado por emplear SQL debido a que ya nos es familiar y además nos proporciona una seguridad en la consistencia de los datos que una base de datos noSQL no da. Por ello, necesitamos un lenguaje que nos permita unificar el trabajo que se realiza en comandos y en SQL, y el elegido es Python. Su facilidad de uso y los varios tutoriales que existen en Internet sobre cómo manejar con él un sistema de ficheros fueron clave para decantarnos por él. Para manejarlo con mayor comodidad haremos uso del framework Flask.

Otros aspectos técnicos

La base de datos va a ser SQL ya que es el lenguaje especifico del dominio que permite gestionar bases de datos y que más extendido está a nivel de software, siendo PostgreSQL el gestor de bases de datos que se ha elegido para gestionar todo lo relacionado con la base de datos. Como alternativas se plantearon como gestores Oracle y mySQL, en ambas las licencias no son gratuitas, habiendo una licencia en mySQL que es gratuita pero no te ofrece todas las funcionalidades disponibles. Por esta razón se ha elegido PostGradeSQL ya que la licencia es gratuita e incluye la gran totalidad de funcionalidades que ofrecen las otras dos alternativas planteadas.

La API Web que se va a desarrollar va a ser RESTful ya que se va a desarrollar una arquitectura que se ejecuta sobre http.

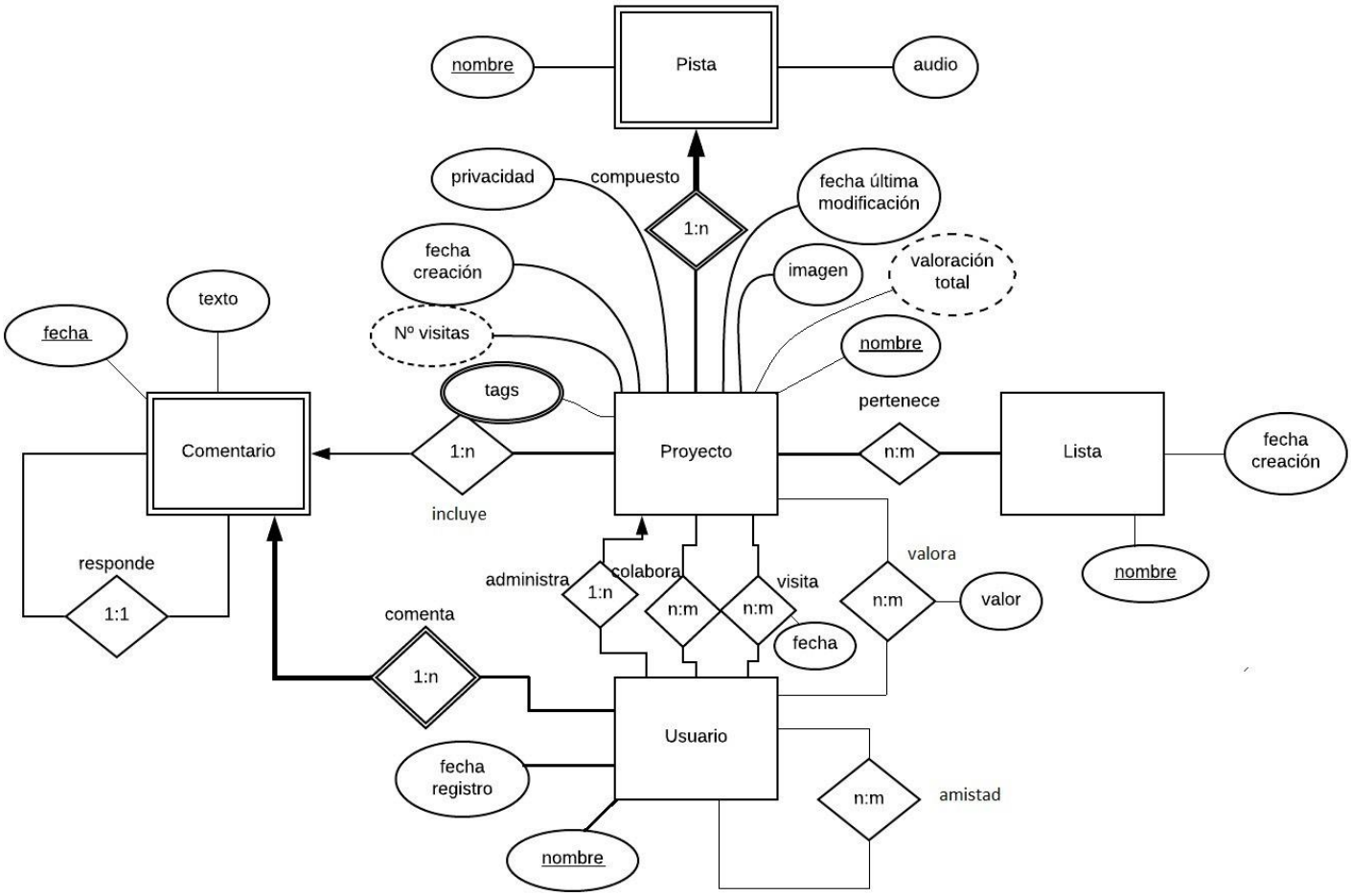
Las operaciones que realizara la aplicación serán síncronas en el sentido de que un usuario cuando realiza una operación no puede realizar otra hasta que no haya acabado la otra. Pero en el sentido del servidor las operaciones serán asíncronas ya que el servidor es capaz de realizar varias operaciones concurrentemente.

La aplicación va a realizarse usando tecnologías web para escritorio, pero va a poder adaptarse como si la estas visualizando por un navegador web en un ordenador como en un smartphone. Otras alternativas habrían sido realizar el sistema como una aplicación móvil o de escritorio pero se ha elegido usar tecnologías web ya que un navegador web está presente en la mayoría de dispositivos por tanto vas a poder acceder a la aplicación desde cualquier dispositivo.

La instalación y los despliegues, en caso de querer un sistema escalable, se realizarán mediante distintos scripts en las máquinas correspondientes. En el forwarding se ejecuta un script que se encarga de la redirección, mientras que en el servidor se ejecuta un script propio del servidor. Ambos scripts pueden ser ejecutados por un único script que se encargue de lanzar los dos. La base de datos se lanza automáticamente gracias a un demonio de la máquina en la que esté corriendo, y por ende no es necesario un script para este servicio.

Base de datos

Diagrama entidad-relación



Como se puede apreciar en el diagrama Entidad-Relación, la base de datos se compone de cinco entidades: Pista, Proyecto, Lista, Usuario y Comentario.

Como cabe esperar en este proyecto, la mayor parte del peso recae sobre la entidad Proyecto, en la cual se pueden resaltar sus atributos "Nº visitas" y "valoración total", los cuales son calculables a partir de las relaciones visita y valora, respectivamente. Por otra parte, también llama la atención el atributo multievaluado "tags", para almacenar las etiquetas representativas de un proyecto.

Por otra parte, merecen especial mención las entidades Comentario y Pista, las cuales necesitan de la clave extranjera de otra entidad (Usuario y Proyecto, respectivamente) para formar su clave, por lo que pasan a ser entidades débiles, como está indicado.

Instalación gestor

La instalación de PostgreSQL se ha realizado mediante el propio sistema gestor de paquetes de la distribución, usando el siguiente comando:

```
$ apt-get install postgresql postgresql-contrib
```

De esta forma se instala el gestor y el paquete "contrib" que incluye algunas funcionalidades aparte. La instalación genera un nuevo usuario "postgres" en el sistema que es el único que puede conectarse al servidor PostgreSQL. Para poder conectarse con él primero se debe cambiar la contraseña con el usuario "root" del sistema y luego hacer login con él:

```
$ passwd postgres
$ login postgres
```

Una vez en el "shell" del nuevo usuario podemos conectarnos al servidor. En PostgreSQL los usuarios funcionan con roles, lo que significa que puedes crear un nuevo rol en el servidor, y podrás conectarte con ese rol si existe en el sistema un mismo usuario con el nombre del rol. Se creará una nueva base de datos, "mixCrowd", un nuevo rol llamado "admin" y a continuación se creará un usuario en el sistema con el mismo nombre:

```
postgres$ createuser admin
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n Shall the new role be allowed to create more new roles? (y/n) n postgres$ createdb mixCrowd
postgres$ psql
```

Ahora ya se ha creado el nuevo rol, pero se le deben otorgar permisos en la nueva base de datos, por ello se accede al servidor mediante el comando "psql", y una vez dentro se ejecutan las siguientes sentencias:

```
ALTER USER admin WITH ENCRYPTED PASSWORD 'password'; GRANT ALL PRIVILEGES ON DATABASE practica1 TO admin;
```

Ahora ya se tiene el nuevo rol en el servidor PostgreSQL, pero es necesario que exista en el sistema un usuario con el mismo nombre que el rol, para ello, desde el "shell" del usuario "root" se puede crear un nuevo usuario mediante el comando "useradd".

```
$ useradd -m admin -p password
```

Se puede comprobar que el servidor se inicia automáticamente al reiniciar la máquina con la salida del siguiente comando:

```
$ service postgresql status
```

Para acceder al servidor como "admin" y trabajar en la base de datos "mixCrowd" tras encender la máquina se deben ejecutar los siguientes comandos:

```
$ login admin
admin$ psql -d mixCrowd
```

Creación de la base de datos

A la hora de implementar la base de datos relacional en PostgreSQL se han creado las tablas de las entidades que aparecen en el diagrama Entidad-Relación, las tablas de las relaciones n:m y la tabla del atributo multievaluado "etiqueta" normalmente.

Se ha añadido la restricción "NOT NULL" a todos los atributos ya que se han considerado imperativa la inclusión de todos los atributos al crear cada tabla; no obstante, hay algunos atributos que no llevan esta cláusula: los atributos calculables "numVisitas" y "valoracion" se especifican como "DEFAULT 0" para que al insertar un proyecto con estos atributos nulos (como es normal) se inserte el cero en ese campo automáticamente, y todos los atributos de tipo date se especifican como "DEFAULT CURRENT_TIMESTAMP" para que al insertar un proyecto con estos atributos nulos (como es normal) se inserte la fecha actual en ese campo automáticamente. Por último, cabe comentar que se han añadido checks para comprobar que no se añaden claves vacías en ninguna tabla y que en la tabla amistad se ha añadido un check que impide que "usuario_1" sea igual que "usuario_2", con lo que no puede haber una relación de amistad de un usuario consigo mismo.

Poblado de la base de datos

Para probar el correcto funcionamiento de la base de datos creada siguiendo el esquema relacional en PostgreSQL, se ha generado un fichero de inserción con datos falsos generados automáticamente para rellenar todas las tablas.

La generación de los "insert" se ha hecho mediante la redirección de la salida de un script en Python (fakeData.py) a un fichero sql (crearPostgre.sql) mediante el comando "python fakeData.py > crearPostgre.sql".

El script en Python imprime por pantalla una serie de inserciones que se consiguen mediante un bucle para la inserción de las filas de cada tabla; los datos falsos se consiguen mediante invocaciones a un objeto "fake", por el cual ha sido necesario incluir una nueva libreria (Python Fake) mediante el comando "pip install fake-factory".

En el script se han creado tres listas (usernames, projectnames, listnames) para guardar valores clave de nombres de usuario, de proyecto y de lista respectivamente, que luego son utilizados para referenciar filas existentes.

5. Memoria del proyecto

Inicio del proyecto

En un principio, se pensó en la necesidad de tener un servidor propio mediante Amazon durante el tiempo que durara este proyecto. Sin embargo, debido a la posibilidad de tener que realizar pagos, se optó por un servidor propio. Dicho elemento se obtuvo mediante la generosidad de un integrante del equipo técnico.

En cuanto a trabajar con APIs, estas se escogieron rápidamente gracias al hecho de ser estudiantes de Unizar; que tiene como privilegios el disponer de diversas APIs durante un tiempo de prueba muy superior al de un usuario no estudiante, o incluso emplear tales herramientas libremente durante el periodo en el que se forma parte de la comunidad Unizar. Para poder disfrutar de ellas, fue necesario un registro verificando la condición de estudiante, caso de Webstorm, o directamente descargándolas, caso de SQLAlchemy.

Durante este proceso, el grupo se dio cuenta de la necesidad de documentarse y formarse y para ello se emplearan recursos. En consecuencia, tras unas semanas de autodidacción, se obtuvieron las cualidades necesarias para poder ejecutar el proyecto propuesto en este documento.

En cuanto al testeo de la aplicación, Google Chrome ha sido el buscador por referencia junto con Mozilla Firefox, y por ende se ha comprobado la ejecución de diversas funcionalidades.

Ejecución y control del proyecto

Durante un proyecto siempre hay problemas varios, los cuales se resuelven lo antes posible para afectar lo mínimo. Uno de ellos, de índole personal, fue la disparidad de opciones, pensamientos y acciones realizadas, pero estos se han solucionado por los involucrados, a veces con ayuda de opiniones del resto en caso de estar relacionados con el código del proyecto. Sin embargo en otras ocasiones, esta solución ha desembocado en un nuevo problema que debe ser solucionado por más miembros, repitiendo así el proceso hasta encontrar una solución sin conflictos.

Otro problema, pero a nivel comunicativo, fue la necesidad de crear de un grupo de Whatsapp para poder comunicarse entre miembros debido a la condición de estudiantes y la imposibilidad de conseguir verse todos regularmente. Entre integrantes se dividió el trabajo en 2 partes diferenciadas, front-end y back-end. 4 miembros se encargaron de lo primero mientras que de los 3 restantes, 1 se encargó de la base de datos y los otros 2 del soporte hardware y software del servidor. En cualquier caso, mediante el grupo todos sabían de la parte de la cual se encargaba cada integrante, teniendo como fechas límite aquellas marcadas por el diagrama de Gantt. Aun así, durante el proceso fue necesaria la reestructuración de dicho elemento. Sin embargo, debido a falta comunicativa y la creencia de la disponibilidad del grupo completo, este no fue seguido de forma concienzuda, dando lugar a prisas posteriores, que se vieron acrecentadas por la subestimación en horas de trabajo de integración, de las últimas fases del proyecto.

Por otro lado, durante el proyecto han surgido distintos problemas que no se pueden reflejar en el diagrama de Gantt. Uno de esos problemas fue la diferencia de estilos entre páginas HTML del proyecto. Para resolverlo, se hizo necesario el trabajo de unas horas de homogeneizar el estilo. De la misma forma, también han surgido problemas durante la integración entre front-end y back-end. Este proceso ha sido largo y tedioso debido a la necesidad de coincidir miembros de diferentes partes para poder completar el trabajo, de forma que se pudieran ayudar entre ellos en la parte que el otro no domina.

También, ha surgido un problema llamativo en el back-end. Este consistía en que una máquina virtual no se lanzaba correctamente impidiendo la conexión correctamente con ella, siendo lo más curioso que sólo fallaba de vez en cuando, obligando a la reinicialización de dicho servicio. Al reinicializarse con el mismo proceso dejaba de existir dicho problema.

Del mismo modo, en el back-end no se llega a probar el sistema con múltiples servidores en activo. Sin embargo, debido a cómo se realiza la conexión con un servidor, el cliente debería conectarse a un puerto del servidor y esperar que éste no esté congestionado. En caso de estarlo, siempre puede cambiarlo, cambiando el puerto al que se conecta.

En la parte de la base de datos y el sistema de ficheros, se intentó almacenar los audios en la propia base sin la necesidad de un sistema de ficheros, como se propuso en una reunión, pero por desgracia, no se aseguraba un retorno del archivo completo solicitado, dando como resultado la vuelta a la idea inicial de disponer de ambos.

A la hora de probar el software, se marcaban unas pruebas manuales evolutivas, que iban ampliándose conforme se integraba la parte correspondiente. Estas pruebas están divididas en dos tipos: de código y de aplicación. En las pruebas manuales de código se aseguraba la funcionalidad del elemento añadido, mientras que en las pruebas manuales de aplicación se comprobaba que dicho elemento no interfiriera con el resto de elementos ya existentes y funcionales.

En cuanto a control de la memoria, todos los miembros del equipo conocían la parte asignada a cada uno. Luego, a la hora de juntar los diferentes apartados, un miembro del equipo, aquel con menor volumen de trabajo en ese momento, se encargaba de de unirlos y aportarle coherencia global en caso de diferencias. Posteriormente, el resto de los miembros lo leían y criticaban el documento hasta llegar al documento final, el cual era entregado con la aceptación y conocimiento de todos en su momento. En las actas, se discernían errores en él; y tras ellas, se hacía que cada miembro modificase la parte asignada previamente en caso de errores o mejoras posibles.

A la hora de ampliar el documento, se evaluaba el tamaño a ampliar y se asignaba otra vez por el mismo método, aquel con menor carga de trabajo. Al final, conforme se iban integrando elementos correctamente, se descubrían nuevos miembros libres, dando lugar a un dilema; hacer que esos miembros libres se encargaran de la memoria o hacer que ayudaran a otros con la integración y el testeo. Esto se resolvió por: nivel de conocimientos globales, nivel de involucración en la parte que se integre en dicho momento y miembros involucrados en la integración, Si el primero es alto, se hace que ayude y en caso de haber 3 o más miembros testear y buscar fallos. Si el primero es bajo pero el segundo es alto, se encarga de de buscar fallos, mejoras y tests; para posteriormente unirse al grupo encargado de la memoria. En resumen, quedan 2 grupos diferenciados: evolución y corrección de este documento e integración. Hay que tener en cuenta que debido a la evolución, los miembros asignados a cada parte iban cambiando.

Cierre del proyecto

El proyecto ha mostrado que a veces conviene aprender el uso de una nueva tecnología que facilite el trabajo, Python, con el objetivo posterior de facilitar la tarea a ejecutar. En caso de haber usado C++, seguiría siendo necesario una ampliación de alta complejidad de nuestro conocimiento sobre ese lenguaje para poder realizar las tareas que con Pyhton se realizan antes y de forma más simple: conexión con la base de datos, conexión con el software FFmpeg y dinamismo de las páginas web

Como consecuencia directa de la elección de lenguajes y tecnologías empleadas, se ha aprendido el trabajo que realizan otras IDEs que disponen de otras funcionalidades más

avanzadas. Por ejemplo, el login está realizado de forma manual debido al desconocimiento en el momento de integrarlo de la herramienta proporcionada por Flask que se encarga de realizar dicho montaje. Posteriormente, en la integración, el grupo se encontró con que la elección de tecnologías pudo haber sido mejor para realizar el proceso y que una mejor documentación sobre ellas podría haber dado menos problemas en ese apartado.

En cuanto al cálculo de estimaciones, la precisión a nivel de horas necesarias para la integración fue bastante bajo, frente a la precisión para el resto de componentes, la cual fue bastante correcta incluso para herramientas con las que se trabaja por primera vez.

Si bien es cierto que se ha empleado Github, nunca se ha sacado partido al completo de su utilidad debido a los problemas ya mencionados con el tiempo y el estilo de trabajo final. Por ende, tampoco ha sido posible obtener la mejor productividad, ni la más constante del grupo.

En global, se han trabajado 807 horas. Estas horas, fragmentadas en el anexo V, se dividen, a grandes rasgos, en distintos trabajos realizados por cada integrante:

Alejandro: diseño de la página de exploración mediante HTML, CSS y Javascript. Además, ha aportado horas en la integración con jQuery, Javascript y Flask.

Darío: funcionamiento del servidor mediante el uso de Flask y python. Por otro lado, ha trabajado también en la integración con jQuery, Javascript y Flask

Diego: diseño de la página de proyectos mediante HTML, CSS y Javascript. También ha realizado trabajo en la integración con jQuery, Javascript y Flask

Gabriel: diseño de las páginas de login y configuración mediante HTML, CSS y Javascript

Joaquín: diseño de la página de estudio mediante HTML, CSS y Javascript. Ha aportado horas de trabajo en la integración con jQuery, Javascript y Flask

Osmar: arquitectura de la aplicación, diseño de la base de datos mediante PostgreSQL y funcionamiento del servidor mediante HTML, CSS y Javascript

Pedro: funcionamiento del servidor mediante el uso de Flask y python, además de un trabajo de integración con jQuery, Javascript y Flask

Todos los integrantes del grupo han aportado un número similar en las horas de testing y de trabajo en la memoria y además, todos los miembros se han encargado de realizar correcciones y mejoras al código existente.

6. Conclusiones

En este apartado se expondrán las distintas conclusiones llegadas una vez realizado el proyecto y presentado por tanto el producto final. Estas conclusiones están orientadas a aspectos de organización grupal, trabajo cooperativo en grupos, administración de tareas y desarrollo software.

No se centrarán tanto, las conclusiones obtenidas, sobre el trabajo en sí sino sobre aspectos más generales sobre el proceso completo de lidiar con un proyecto de este calibre. Los cuales serán tenidos en cuenta en el futuro si un trabajo similar se llegará a realizar.

Pues la experiencia ha demostrado enseñar lo siguiente:

- **La documentación y la información sobre tecnologías serán tus aliados**. En el grupo de trabajo, la mayoría de personas no habían tenido un contacto previo con diseño web, así como el montaje de tu propio servicio o app de este tipo. Este hecho suponía que el grupo en cuestión no tenía una idea clara sobre las tecnologías a usar y de cómo usarlas.

La decisión de que herramientas usar fue optar por aquella solución más sencilla y la más usada con un mayor número de información en internet. No obstante, esto supuso no optar por el uso de tecnologías, en concreto frameworks, que abstraigan y faciliten ciertos ámbitos de la programación y configuración del sistema.

En su lugar se optaron por otras herramientas más básicas y primitivas que resultaron alargar el proceso y dificultar las tareas de programación. Puesto que la programación y corrección del código recae directamente sobre el programador cuando estos aspectos en un framework hubieran sido más sencillos y automáticamente realizados por la herramienta.

A su favor, indicar que este método de trabajo nos mostró los entresijos, es decir, cual es el trabajo de abstracción que los framework realizan. Y así conocer en más detalle lo que se obviaría por uso de herramientas de mayor nivel.

- **Conoce a lo que te enfrentas**. Otro fallo fue asumir que ciertas partes del proyecto iban a ser muy fáciles de realizar. En concreto la integración de front end y back end.

En las primeras reuniones se dividió el grupo en dos grandes subgrupos. Cada subgrupo se encargaba de realizar una parte del proyecto: front end o back end. No obstante en ningún momento se consideró un grupo especializado el cual realizará o se dedicará a la integración de ambas partes. El hecho fue el desconocimiento de dicha integración.

Al no realizar un proyecto igual antes, se vio la parte de integración y unión de ambas partes como un trabajo simple puesto que lo realmente costos era realizar ambas partes. Pues, los esfuerzos del grupo se centralizaron en realizar esas dos partes.

Sin embargo a pesar de asegurar el funcionamiento individual y la corrección de ambas se vió, luego al integrarlas que había ciertas diferencias y problemas de compatibilidad. Así como nueva documentación que buscar e información sobre cómo unirlas.

Esto provocó que muchas horas fueran malgastadas al final del trabajo intentando solventar problemas que si se hubieran tratado desde un inicio no hubieran perjudicado tanto el desarrollo y avance del proyecto.

- **La planificación y la organización harán que llegues a tu destino sin grandes dificultades**. Otro problema fue la organización y la planificación. Estos dos aspectos son el alma y corazón de cualquier trabajo sobre todo en grupos numerosos.

Al ser siete personas, con horarios muy dispares y estudiantes de distintas ramas con distintas asignaturas, el quedar en persona era realmente difícil. En vez de realizar una organización y planificación por escrito para que fuera seguida por todos los miembros de forma estricta, se decidió hacer uso de una planificación más flexible que se compusiera de largos periodos para que todos realizaran su trabajo sin agobios.

Lo cual supuso enormes pérdidas de tiempo, que una planificación y organización más estricta, que hubiera marcado unos objetivos a cumplir diarios o semanales, hubiera supuesto un mayor aprovechamiento de los plazos dados para realizar el proyecto así como una constante conexión con el trabajo que se realiza y así mantener en mente el uso de tecnologías en las que se es novato y no se tienen dominadas.

- **La comunicación es esencial**. Debido a nuestra disparidad en horarios, el quedar en persona para trabajar era toda una odisea de mensajes. Por lo que la mayoría de la comunicación se realizó a través de Whatssapp.

Este medio aunque es una herramienta en teoría apropiada suponía que cada miembro leyera o se informará sobre asuntos de forma tardía y no en el momento. Lo cual generaba que cosas ya pactadas con anterioridad se tuvieran que volver a discutir añadiendo horas perdidas.

Además la no comunicación en persona añadía un problema de unión entre las partes. El trabajo era dividido y al no haber personas encargadas de enlazar o unir las partes, se creaba una situación donde las personas encargadas de las partes debían de estar presentes para entender qué es lo que se intentaba realizar y cómo operaba el código puesto que muchas veces los comentarios eran insuficientes.

Claro el problema está cuando dicha persona encargada no está presente. Esto provocaba parones, problemas y errores a solucionar, puesto que se intentaba modificar cosas realizadas por otro programador sin una comprensión del código en sí, y todo hacía que el cómputo de horas aumentará.

- **Como novatos nuestro trabajo es doble**. Ninguno de los integrantes ya se había topado con un trabajo de este calibre, y sin duda la falta de experiencia fue un factor clave. Sin una guía clara y disponible, solo con conocimientos globales, las horas dedicadas a la búsqueda de información y la comprensión de esta fueran bastante grandes. Además sin el uso de frameworks conocer por completo nuevos idiomas e intentar conseguir dominarlos es un plazo de tiempo tan corto es prácticamente imposible.

Esto se aprecia en una interfaz muy básica y simple sin un gran artificio. Además suponía el tener que hacer un esfuerzo extra. Puesto que trabajar con herramientas nuevas puede ser muy frustrante debido a la obtención de errores desconocidos o poco habituales que suponen horas de búsqueda extra a las horas totales gastadas.

- **Estos proyectos son complejos y extensos**. Al ser novatos, no conocíamos la extensión de estos proyectos. La cantidad de partes involucradas (API, interfaz, bbdd, etc..) Cuya integración en un único sistema suponía una ardua tarea de depuración en herramientas de depuración con las que nunca se había trabajado (de nuevo horas extras gastadas en aprender a usar herramientas).

Sin un conocimiento global la evaluación de la dificultad fue errónea subestimando la carga de trabajo. Al final se vio que dichos proyectos exigen gran trabajo y dedicación.

Por lo que podemos decir como grupo es que, a pesar de todos los problemas y las horas invertidas, ha valido la pena. Este trabajo nos ha abierto los ojos a un nuevo campo de la informática que nunca antes habíamos visto, así como enseñarnos que somos capaces de

adaptarnos a la situación y aprender por nosotros mismos nuevas cosas. Entender la importancia de pasos y procesos explicados anteriormente que en otros proyectos de menor calibre hemos pasado por alto pero que aquí resultan ser muy necesarios.

Por lo que podemos decir que para la siguiente vez que nos topemos con un trabajo parecido podremos lidiar con el y asegurar que los mismos errores no ocurrirán.

I. Anexo I. Glosario

Back-end: parte de una aplicación relacionada con el guardado y mostrado de datos requeridos por un usuario.

Forwarding: equipo electrónico que permite dirigir el tráfico creado por un usuario a un equipo que actúa como servidor concreto.

Framework: conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problema particular.

Front-end: parte de una aplicación relacionada con los elementos visibles por un usuario

Interfaz: pantalla que el usuario ve en su monitor

Login; acceder o registrarse en un sitio mediante un nombre de usuario y una contraseña única para el sitio en cuestión

Máquina virtual: software que permite la simulación de un sistema operativo y sus funcionalidades.

Script: programa que permite la ejecución de distintos programas de forma automática.

Servidor: equipo electrónico que guarda y devuelve datos a un usuario de una aplicación

II. Anexo II. Actas de las reuniones

Acta de la primera reunión

Fecha: 15 de febrero de 2018
Hora: 16:00 a 17:00 pm
Duración: 1 h

En el acta se recogieron los principios sobre el desarrollo de la aplicación web o sistema a realizar a lo largo de la asignatura conjuntamente con el profesor comentado los siguientes aspectos:

- Presentación de una app web que permita la colaboración de distintos usuarios en la realización de proyectos musicales. Una primera presentación del modelo distribuido a realizar, con el hardware necesario para realizar dicho modelo. Todo esto descrito en más profundidad en la propuesta económico y técnico que se presentará.

- Existencia de distintos usuarios. Administradores: creadores de un proyecto los cuales tendrán permisos completos para la manipulación y configuración de su propio proyecto. Colaboradores: resto de usuarios que solo aportan pistas al proyecto con el fin de que al administrador le gusten y las considere para realizar su mezcla personal.

- Se buscará código libre y de terceros para conseguir la mezcla de las distintas pistas que formen parte de un proyecto (FFmpeg).

- Interfaz asíncrona, que no se bloquee la aplicación a la espera del mezclado de las distintas pistas. El usuario podrá seguir realizando otras tareas mientras que la mezcla tiene lugar.

- Posible uso de un servidor Tomcat para la mezcla de las pistas musicales así como para atender a los clientes y hacer consultas a la base y sistema de ficheros (lugar donde se almacenarán los distintos archivos de audio).

- Las pistas deberán ser subidas en formato mp3, para poder partir los temas y enviarlos más rápidamente.

- Uso de una pantalla inicial de registro y login (puede ir en otra pantalla, o realizar un registro del usuario en la misma página en una sección donde se pueda realizar rápidamente).

- Uso de una lista de proyectos para mostrar en una primera pantalla: novedades (nuevos proyectos), los mejores valorados (conjunto de proyectos con una gran valoración), etc.. Además de un navegador habilitado para la búsqueda de distintos proyectos o usuarios. De esta forma se facilita el acceso a proyectos de interés y la colaboración con usuarios en específico.

- Usuarios no registrados sólo podrán navegar por la app web como meros espectadores y escuchar las pistas en proyectos ya existentes y creados. Su pantalla será la misma que la de todo usuario donde aparecen las listas con los distintos proyectos más novedosos, mejor valorados, etc...

- Existirán dos tipos de proyectos públicos y privados. Los privados solo podrán ser vistos por los usuarios permitidos por el administrador mientras que los públicos podrán ser vistos y participar todos los usuarios en la web app.

- Todos los requisitos deberán de ser cumplidos y realizados para poder alcanzar los

puntos pactados: 3 puntos.

- Los distintos miembros del grupo deberán de consultar las tecnologías a usar y decidir por ellos mismo cuales son las que se deberían de usar.

Acta de la segunda reunión

Fecha: 27 de febrero de 2018
Hora: 11:00 a 12:00 am
Duración: 1 h

En el acta se revisó el trabajo realizado en la propuesta económica y técnica, así como posibles mejoras y tareas a realizar para el plan de gestión, análisis, diseño y memoria del proyecto (la primera versión). Así como la forma de planear y controlar la realización del trabajo:

Aspectos para el plan de gestión

- Posibilidad sobre incluir anuncios en la aplicación web. El cliente en cuestión tendría una fuente de ingresos constante. Podría conseguir beneficio de otras empresas interesadas en publicitarse dentro de la web app. Lo cual debería de ser introducido como un nuevo requisito. Deberá notificarse al profesor si se decide realizar dicho requisito.

- El servidor web se realizarán con herramientas como CSS3, HTML5 y Bootstrap. Puesto que JSP ya se encuentra obsoleto y no es una tecnología que se use ya.

- Hacer una estimación o recomendación al cliente sobre el hardware necesario, indicando el número de usuarios que puede soportar. Para ello consultar la documentación del Tomcat, y demás tecnología de la que se haga uso.

- Posibilidad de virtualizar mediante Virtualbox los distintos servidores a usar. De esta manera obtener dos máquinas de una única máquina física.

- Incluir un software de balanceo de tráfico para la no sobrecarga de los distintos servidores.

- Para el despliegue de las máquinas valorar las opciones a elegir. Automatizar este proceso.

- No es necesario argumentar la elección de las tecnologías elegidas o que se vayan a usar.

- Prestar especial atención a la gestión de planificación del proyecto. Punto importante para la siguiente reunión.

- Realizar el esquema de la planificación para indicar en futuras reuniones el estado en el que se encuentra el desarrollo del proyecto.

- -Pensar en otras funcionalidades extras, puesto que la calificación pactada en un no es alcanzable con lo expuesto en la propuesta.

Corrección de la memoria

- El dinero que se vaya a cobrar al cliente debe redondearse. De lo contrario la cifra puede parecer extraña, y se redondea teniendo en cuenta el IVA.

- Los requisitos se encuentran desordenados dentro de la propuesta realizada. Hay que organizarlas por funcionalidad, por ejemplo: reproducción de música, usuarios, sincronización, búsqueda, etc…

- -Falta del requisito de alta de usuarios. Como se va a realizar, mediante correo electrónico o mediante Facebook. Describir mejor el proceso de creación de un proyecto

y el papel de los distintos involucrados y el administrador de dicho proyecto.

- No se describe en la propuesta las características de los proyectos privados y lo que estos suponen.

- -La interacción entre los distintos usuarios mediante comentarios debe de ser añadido como un requisito funcional. Debe de corregirse.

- En la propuesta no se habla sobre la identificación de los distintos proyectos. Se realizará mediante imagen, un título, y una descripción. Añadir esto en la parte de los requisitos.

- Forma de búsqueda de usuarios y proyectos. Los usuarios mediante nombre o identificador. Los proyectos por categoría, nombre del proyecto y usuario, administrador del sistema.

- No incluir pies de foto al menos que éstos expliquen la imagen

- Fallos de navegabilidad en la interfaz. No se implementa un método que posibilite la vuelta atrás del usuario. Posibilidad de añadir un menú que permita al usuario una libre navegación y la consulta de los distintos apartados de la app web.

- Se deben añadir requisitos no funcionales a la propuesta económica y técnica.

Otros aspectos

- En Moodle añadir los esfuerzos por semana de cada componente del equipo indicando que se ha realizado y cuantas horas se ha invertido.

Acta de la tercera reunión

Fecha: 22 de marzo de 2018
Hora: 17:00 a 18:00 pm
Duración: 1 h

Durante la reunión se han tratado diferentes puntos, que pueden resumirse de la siguiente manera:

DESARROLLO DEL TRABAJO:

El trabajo de desarrollo a realizar está muy concentrado, como se puede ver en el diagrama de Gantt. Esto se ha realizado así premeditadamente, ya que se pretende dedicar un gran número de horas al desarrollo durante el periodo de Semana Santa. Aún así, sería deseable reducir un poco la carga de trabajo durante esos días.

También se ha planteado la idea de reducir la magnitud de las tareas a realizar, para que no sean periodos de tiempo tan largos, como mucho 1 o 2 semanas. De esta manera se podrán detectar retrasos con mayor rapidez y tener una idea aproximada del porcentaje de desarrollo que queda por realizar. Todos los cambios producidos deberán reflejarse en nuevas versiones del diagrama de Gantt.

ARQUITECTURA DEL PROYECTO:

Es necesario concretar cuanto antes la arquitectura del proyecto, ya que hasta ahora principalmente se ha trabajado con bocetos o prototipos. Mañana viernes se dedicará el día a delimitar estos aspectos.

CONTROL DE TAREAS Y VERSIONES:

Se ha hablado en la reunión de una herramienta llamada TRELLO, para la planificación del desarrollo de tareas. También se ha hablado de la utilización de Github, mediante diferentes ramas, para que cada miembro del grupo pueda desarrollar su propio software para unirlo posteriormente, cuando se constate que funciona de forma correcta.

ITERACIONES:

La idea que se planteaba era realizar un sistema cerrado en primera iteración para posteriormente añadir la funcionalidad para los usuarios no registrados. En vez de eso es más conveniente realizar en primera iteración el sistema completamente abierto y posteriormente corregir errores y añadir el resto de funcionalidades en una segunda iteración. De esta forma se evita trabajar “dos veces”.

SERVIDORES Y BASE DE DATOS:

Se ha tratado el tema de la actualización de la base de datos, puesto que, el diseño planteado tiene opción de trabajar con varios servidores, lo cual implica que cada uno de estos tiene que tener una copia de la base de datos.

Se va a suponer que no es necesario garantizar las transacciones, puesto que esto debería constar como un requisito funcional importante y sería necesario mucho tiempo de desarrollo, que iría en detrimento del resto de funciones planteadas en los requisitos.

Por otra parte, para el desarrollo del proyecto, ha surgido la idea de tener una máquina virtual que guarde la base de datos en funcionamiento más actualizada, para que los diferentes miembros del grupo puedan trabajar con ella sin miedo a perder los datos o a que se produzcan problemas.

En cuanto al esquema de la base de datos, será necesario añadir nuevas relaciones para representar las recomendaciones que se van a realizar a los diferentes usuarios.

Por último, se ha hablado de la posibilidad de eliminar el sistema de ficheros pensado en un primer momento para almacenar las pistas de música, e incluir los ficheros en la propia base de datos mediante algún tipo de dato en Postgre que permita esto.

Acta de la cuarta reunión

Fecha: 10 de abril de 2018
Hora: 11:00 a 12:00
Duración: 1 h

Durante la reunión se han tratado diversos apartados que se pueden agrupar en:

Desarrollo del trabajo

El trabajo de desarrollo, acorde al diagrama de Gantt, está vacío en las fechas posteriores al día de esta acta. Se ha realizado así premeditadamente, pero durante la reunión se ha visto que diversos apartados estaban incompletos y que se retoman después de esta reunión. Por ello, se crean nuevas tareas, especialmente aquellas relacionadas con el frontend, para expresar esos cambios que se realizan, independientemente de la simplicidad que conllevan o la menor carga de trabajo que acarrean en comparación con su primer periodo de trabajo.

Además, falta integrar el front-end y el back-end, objetivo que tampoco está en el diagrama de Gantt. Cierto es que por separado se han ido probando los distintos apartados, pero nunca en conjunto. Es por esta razón que una tarea de integración pueda ser necesaria para su realización, además de los tests poteriores.

Arquitectura del proyecto

Se ha comentado la arquitectura de la aplicación en la reunión, pero para facilitar el trabajo con ella, se habla de la posibilidad de indicar las dependencias entre las clases de distintas capas entre sí.

Mapa de navegación

Se ha visto que se tienen las interfaces claras y dominadas, pero falta una visualización de ellas en la memoria indicando cómo interactúan entre sí. Es decir, añadir un mapa de navegación de las interfaces es vital si se está valorando el producto entre distintos competidores según su documentación, paso previo a la decisión de compra. Además, puede darse el caso de que al realizarlo se pueda ver si es necesaria alguna interfaz más, que de otro modo no se sabría hasta que se detecte muy posteriormente.

Objetivo para la siguiente reunión

Se ha tratado de la necesidad de que para la próxima reunión, haya integración entre back-end y el front-end, ya que en esta reunión se esperaba una mayor según el diagrama. Este apartado de integración ya se ha comentado antes, y como falta definir su horario de trabajo, será necesario modificar el diagrama de Gantt.

Presentación del proyecto

Finalmente, se han visto varias formas de presentar el proyecto en la entrega final, pero en todas ellas primaba la necesidad de contar con al menos dos portátiles a la hora de mostrar el correcto funcionamiento. Uno de ellos actuará como cliente mientras que el resto de portátil(es) harían de servidor, de esta forma se podría ver la comunicación entre el cliente y el servidor.

Acta de la quinta reunión

Fecha: 4 de mayo de 2018
Hora: 13:00 a 14:00
Duración: 1 h

Durante la reunión se han tratado diversos apartados que se pueden agrupar en:

Arquitectura del proyecto

Como se comentó en la reunión anterior, el diagrama que hay en el apartado Arquitectura del proyecto, es realmente un diagrama de clases que debería ir en un anexo. En sustitución, es necesario ver un diagrama que indique las relaciones entre la base de datos, el servidor, la API, etc. Este diagrama puede ir acompañado de documentación que indique qué se ha empleado para implementar alguno o cada uno de los bloques, por ejemplo SQLAlchemy para la base de datos.

Especificación del contenido y de los tests

Se ha observado que el contenido de la memoria generaliza en varios apartados en los que son recomendables profundizar más. En cuanto a los estándares que se siguen, profundizar un poco más en que características se siguen.

En relación a los tests a los que se somete el código, lo ideal sería comentar cuáles son esos y cómo se realizan. Aunque en nuestro caso, toda pequeña modificación en el código es comprobada manualmente inmediatamente después de ser implementada, disponer de pruebas de caja blanca ayudaría mantener la relación entre implementaciones distintas, pues un cambio mínimo en una parte podría afectar a otra parte que puede no ser comprobada manualmente. Además, estos tests automatizados, pueden ser de más utilidad en diversos proyectos posteriores.

Por otro lado, es recomendable dividir los tests por grupos debido a las partes tan diferenciadas del proyecto (back-end, front-end…).

Finalmente, los tests parciales con los que se prueba el código están reflejados en el diagrama de Gantt, pero no así el test global final de la aplicación.

Control de la memoria

En la memoria se refleja perfectamente cómo evoluciona el código y las versiones de éste, pero no hay nada en relación a la propia documentación. Se ha explicado que se divide por partes y que cada integrante del grupo se encarga de una parte que se encarga de mantener, mientras que el documento final, completo y actual está a disposición de todos en Git. Por lo tanto, en Git sólo está dicho documento, del que se puede recuperar una versión anterior si es necesario.

Con vistas a las horas trabajadas puestas en las encuestas de Moodle, es recomendable añadir un nuevo anexo indicando las horas trabajadas por integrante y completarlas con la información de lo trabajado en esos momentos. Estos datos pueden ser dispuestos mediante la propia aplicación con la que se modifica la memoria, o empleando otras herramientas que puedan ayudar, tales como Excel.

Acta de la sexta reunión

Fecha: 30 de junio de 2018
Hora: 17:00 a 18:00
Duración: 1 h.

Durante la reunión se han tratado diversos apartados que se pueden agrupar en:

Demostración del proyecto.

Durante media hora se ha presentado el trabajo realizado a lo largo del curso. Validando que se habían cumplido los requisitos y funcionalidades propuestas en el anteproyecto al principio de evaluación.

Despliegue del sistema

Como se había desplegado el sistema a través de un dominio redireccionado a nuestra red local en casa de Dario, donde están los servidores utilizados, se ha conectado a ellos por ssh para comprobar su funcionamiento, que verdaderamente estaba sucediendo todo.

Problemas encontrados

La mayoría de los problemas encontrados han estado relacionados con la deuda técnica con la que contábamos los miembros del equipo del proyecto. Pero ya no solo de uso tecnologías sino de desconocimiento sobre el propio proceso de creación de este tipos de proyectos. Todo habría sido mucho más fácil, por ejemplo, si hubiéramos escogido desde un principio un framework de Javascript para desarrollar el front-end como puede ser Angular o Vue.js con una pequeña base de datos firebase en vez de utilizar Javascript a pelo. Aunque es verdad que gracias a esto y utilizar Flask comprendemos muy bien cómo funciona toda nuestra aplicación, como se relaciona el front-end con el back-end y como interactúa esto en la red, y en futuras asignaturas ya utilizaremos herramientas de más alto nivel.

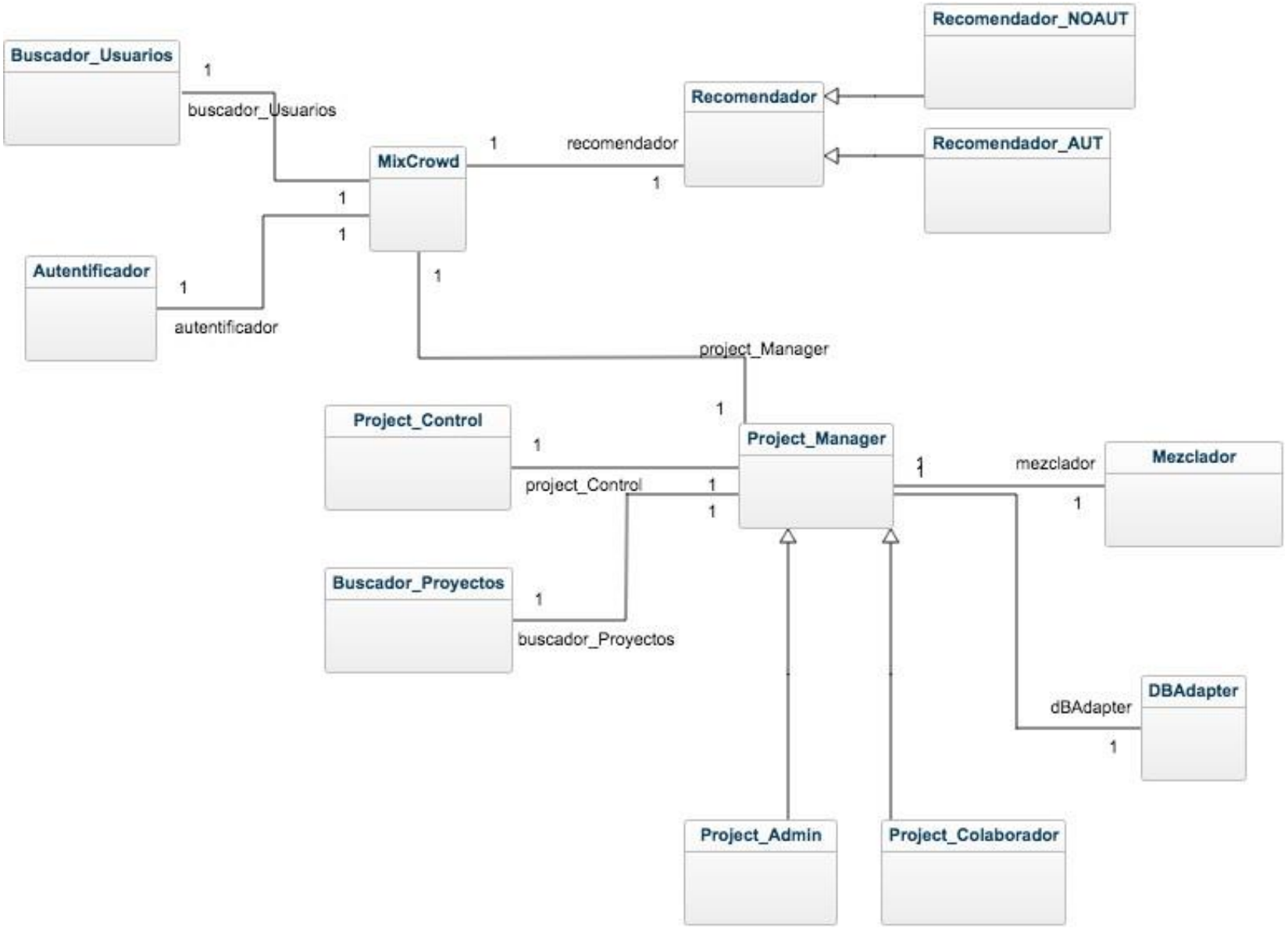
Lo que hemos aprendido en la asignatura

Hemos aprendido que hay que tener muchísimas cosas en cuenta para poder organizar un equipo de personas. Que el software es muy poco flexible y las integraciones son partes muy difíciles del proceso de desarrollo. Pero sobre todo que hacer cualquier sistema de información robusto, escalable y con código bien hecho y reusable para futuros proyectos, iteraciones o refactorizaciones es verdaderamente muy difícil.

Valoración de la asignatura y cosas a mejorar.

Como hemos comentado con Javier sería muy útil para gente que no hemos participado nunca en un proyecto de esta envergadura (y tenemos una deuda técnica total) una guía de tecnologías a usar en el desarrollo web así de cuáles son las fases necesarias para integrar el código según las tecnologías escogidas.

III. Anexo III. Diagrama de clases



Se ha representado la arquitectura de la aplicación mediante un diagrama de clases. En este se muestran las relaciones entre todos los objetos de los que se compone el sistema.

La clase ***MixCrowd*** es la que soporta el funcionamiento del sistema, dentro de ella encontramos diferentes elementos. En primer lugar un ***Autentificador***, que gestiona el registro y el login de los usuarios en la aplicación. Por otra parte, también cuenta con un ***Recomendador***, que puede ser no autentificado, si no se realiza el login, o autentificado si ya se ha realizado el login del usuario. Si el recomendador es autentificado, las recomendaciones son específicas para cada usuario, pero si no es autentificado las recomendaciones serán por proyectos más visitados y mejor valorados.

Posteriormente se encuentra un ***Project_Manager***, que permite la gestión de los proyectos. El ***Project_Manager,*** como se puede apreciar en el diagrama, puede ser de dos tipos***,*** ***Project_Admin*** o ***Project_Colaborador.*** En el primer caso significa que el usuario que accede al proyecto es el administrador del mismo, por lo que tiene un mayor control sobre éste y por lo tanto cuenta con una mayor cantidad de funciones. Sin embargo, si se es colaborador de un proyecto, se instanciará un ***Project_Colaborador,*** que tiene un número de funciones limitado, debido a que las acciones a realizar por un colaborador de un proyecto son menos que las que posee el administrador del mismo.

Dentro del ***Project_Manager***, a su vez, hay diferentes elementos, un ***Project_Control,*** para comprobar los permisos que tiene cada usuario dentro de un proyecto, un ***Mezclador*** para hacer la mezcla de las pistas musicales, un ***DBAdapter,*** que hace de adaptador entre la API y la base de datos, donde se encuentran todas las llamadas a realizar a la base de datos***,*** y finalmente un buscador de proyectos, que permite realizar listado de proyectos o búsquedas de proyectos por nombre.

Finalmente, ***MixCrowd***, también cuenta con un buscador de usuarios, que funciona de forma análoga al de proyectos, permite la búsqueda de usuarios por nombre o la muestra de un listado de usuarios.

IV. Anexo IV. Readme

Este manual está hecho con el fin de detallar el procedimiento a seguir para poder desplegar correctamente la aplicación mixcrowd en un servidor sencillo (una máquinas), sin forwarder ni varios servidores (si se tienen varias máquinas se puede utilizar una como forwarder y las demás como servidores conectados a una misma base de datos).

REQUISITOS:

- Máquina Linux (Ubuntu, Debian,…) con :
  - Virtual Box
  - Flask (python)
  - Máquina virtual Ubuntu (no necesaria interfaz gráfica) con base de datos PSQL

PROCEDIMIENTO A SEGUIR:

En primer lugar, se instala una máquina virtual en virtual box y se configura el adaptador red como “BridgedAdapter”, lo cual permite a la máquina virtual conectarse directamente a la puerta de enlace del router al que se conecta la máquina física que alberga la máquina virtual, permitiendo situar la base de datos en otra red local diferente. La ip privada de la maquina virtual se fija para que la posterior redirección de la ip pública del router sea inmutable (puestoque si dejásemos que el protocolo DHCP asignase la ip a la máquina virtual, a cada conexión de la maquina al Gateway se reiniciaría su ip privada y habría que cambiar la redirección del router).

Posteriormente, la instalación de PostgreSQL serealiza mediante el propio sistema gestor de paquetes de la distribución, usando el siguiente comando:

$ apt-getinstallpostgresqlpostgresql-contrib

De esta forma se instala el gestor y el paquete "contrib" que incluye algunas funcionalidades aparte. La instalación genera un nuevo usuario "postgres" en el sistema que es el único que puede conectarse al servidor PostgreSQL. Para poder conectarse con él primero se debe cambiar la contraseña con el usuario "root" del sistema y luego hacer login con él:

$ passwdpostgres

$ loginpostgres

Una vez en el "shell" del nuevo usuario podemos conectarnos al servidor. En PostgreSQL los usuarios funcionan con roles, lo que significa que puedes crear un nuevo rol en el servidor, y podrás conectarte con ese rol si existe en el sistema un mismo usuario con el nombre del rol. Se creará una nueva base de datos, "mixCrowd", un nuevo rol llamado "admin" y a continuación se creará un usuario en el sistema con el mismo nombre:

postgres$ createuseradmin

Shallthe new role be a superuser? (y/n) n

Shallthe new role be allowedtocreatedatabases? (y/n) n

Shallthe new role be allowedtocreate more new roles? (y/n) n

postgres$ createdbmixCrowd

postgres$ psql

Ahora ya se ha creado el nuevo rol, pero se le deben otorgar permisos en la nueva base de datos, por ello se accede al servidor mediante el comando "psql", y una vez dentro se ejecutan las siguientes sentencias:

```
ALTER USER admin WITH ENCRYPTED PASSWORD 'password';

GRANT ALL PRIVILEGES ON DATABASE practica1 TO admin;
```

Ahora ya se tiene el nuevo rol en el servidor PostgreSQL, pero es necesario que exista en el sistema un usuario con el mismo nombre que el rol, para ello, desde el "shell" del usuario "root" se puede crear un nuevo usuario mediante el comando "useradd".

```
$ useradd -m admin -p password
```

Se puede comprobar que el servidor se inicia automáticamente al reiniciar la máquina con la salida del siguiente comando:

```
$ servicepostgresqlstatus
```

Para acceder al servidor como "admin" y trabajar en la base de datos "mixCrowd" tras encender la máquina se deben ejecutar los siguientes comandos:

```
$ loginadmin

admin$ psql -d mixCrowd
```

Una vez instalado Postgre SQL y configurado correctamente, se puede proceder a redireccionar la ip pública del router desde un puerto de este hacia la ip privada de la red local correspondiente a la máquina donde se corre el servidor (y hacia un puerto local de esta máquina en concreto).

Por último, mediante la web no-ip.com se asocia un dominio a una ip (la ip pública del router). Para solucionar este aspecto además hace falta correr un software que te proporciona la propia página que te vuelve a asociar la ip del router al dominio cuando se reinicia el router y se cambia la ip pública dinámica.

Se comprobará la conexión desde la máquina principal con la virtual y tras esto, se cargará la base de datos disponible en el fichero “postgresql.sql”

Basta con configurar el router para permitir la entrada de tráfico del exterior el cual se conectará a flask.

Se lanzará el flask con la orden:

Python practica1.py

Una vez hecho esto, ya se está listo para acceder desde el exterior usando la dirección http://mixcrowd.sytes.net:5000

V. Anexo V. Horas trabajadas

PROYECTO SOFTWARE																			
MIXCROWD																			
DEDICACIÓN DE HORAS																			
	HORAS (POR SEMANA)																		
COMPONENTES DEL GRUPO	NIA	S1 19-25 Fe	S2 26-4 Mar	S3 5-11 Mar	S4 12-18 Ma	S5 19-25 Mar	S6 26-1 Ab	S7 2-8 Ab	S8 9-15 Ab	S9 16-22	S10 23-29 Al	S11 30-6 Ma	S12 7-13 Ma	S13 14-20 May	S14 21-27 May	S15 28-31 May	ENTREGA	Horas Totales	TOTAL HORAS DEDICADAS
Pedro Ramoneda	717976	4	3	6	2	10	20	20	4	3	3	3	5	0	15	30		128	
Joaquin González	715821	7	8	4	3	10	8	8	8	6	4	3	2	5	15	9		100	
Darío Ferrer	721132	4	4	3	3	5	10	6	5	5	6	6	6	10	26	35		134	
Gabriel García	723553	0	8	8	3	7	10	5	4	3	3	4	6	8	12	20		101	
Alejandro Francés	717361	6	4	3	3	4	12	3	4	3	3	2	0	13	22	19		101	
Osmar Ali De La Fuente	719152	0	2	5	7	0	20	5	40	0	5	1	2	0	10	30		127	
Diego Santolaya	727209	4	5	10	4	4	5	3	5	2	3	5	8	10	18	30		116	
																			807

VI. Anexo VI. Bibliografía

https://github.com/UNIZAR-30226-2018-06/MixCrowd-BackEnd

https://github.com/UNIZAR-30226-2018-06/MixCrowd-FrontEnd

https://es.wikipedia.org/wiki/Panning

www.flask.pocoo.org

www.getbootstrap.com

https://www.jetbrains.com/webstorm/

https://www.postgresql.org

www.stackoverflow.com Empleo de múltiples enlaces del blog

www.3wschools.com Empleo de múltiples enlaces del blog