A logo of a coat of arms

Description automatically generated UNIVERSIDAD DE BURGOS

Escuela Politécnica SuperiorA logo of a machine

Description automatically generated

Gº en Ingeniería Informática

Imagen que contiene Logotipo

Descripción generada automáticamente

A black and white logo

Description automatically generated

**TFG Ingeniería Informática:**

**Digitalización de Partituras Antiguas**

Presentado por Martín González Saiz

en Burgos el X julio de 2024

Tutor D. José Manuel Galán Ordax

**Índice General**

**Índice de Tablas**

**Anexo A Planificación del proyecto**

1. **Introducción**

En este apartado de los anexos se va a explicar detalladamente el desarrollo del proyecto durante los sprints así como su viabilidad económica.

1. **Planificación temporal**

En cada uno de los sprints ha habido una reunión previa en la que se ha acordado las tareas y objetivos a alcanzar. Para llevar a cabo esta gestión se ha utilizado el gestor de proyectos Trello, para administrar tanto la categoría, como el estado de cada tarea.

* + Sprint 1: <https://trello.com/b/s2npoqgv/tfg-digitalizacion-de-partituras-sprint-1-15-02-2024-01-03-2024>
  + Sprint 2: <https://trello.com/b/SEnQJM6h/tfg-digitalizacion-de-partituras-sprint-2-01-03-2024-17-03-2024>
  + Sprint 3: <https://trello.com/b/5qrTSteY/tfg-digitalizacion-de-partituras-sprint-3-18-03-2024-15-04-2024>
  + Sprint 4: <https://trello.com/b/W4oJQ7Ah/tfg-digitalizacion-de-partituras-sprint-4-15-04-2024>
  + Sprint 5: <https://trello.com/b/XWatG1p6/tfg-digitalizacion-de-partituras-sprint-5-23-05-2024-20-06-2024>
  + Sprint 6: <https://trello.com/b/OuDMWHQl/tfg-digitalizacion-de-partituras-sprint-6-21-06-2024>

Al inicio del proyecto se plantearon unos objetivos a alcanzar y como podría llegar a ser el desarrollo, así como las posibles herramientas a utilizar. Con ello se empezó la primera reunión en la que se centró su enfoque en los preparativos del proyecto.

* + 1. **Sprint 1: 15/02/2024 - 01/03/2024**

En la primera reunión para el primer sprint planeamos y acordamos preparar todos los repositorios en cada una de las plataformas, además de todos los programas, que son utilizados a lo largo del desarrollo del proyecto. Por tanto en estas semanas creé repositorios para GitHub y Trello.

GitHub para poder reflejar la evolución del trabajo a nivel de código y programación así como para tener un buen control de las versiones y cambios, mientras que Trello reflejaría el trabajo que hay detrás de todo el tiempo de investigación, desarrollo y estudio de diferentes aspectos del proyecto.

Además instalé y configuré Mendeley para poder registrar todas las referencias a las que haya accedido para posteriormente poder implementarlo en la memoria. También me serviría para poder acceder a paginas útiles siempre que quiera durante el desarrollo de la herramienta.

Otro aspecto importante fue en estudio del entorno en el que se utilizaría la aplicación. Hice una investigación para elegir entre una aplicación web o una de escritorio, valorando sus ventajas e inconvenientes.

Por último me decanté por desarrollar la memoria en Word por su simplicidad en comparación con Latex.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha | Tarea | Etiqueta | Estado | Tiempo estimado | Tiempo real |
| 16/02/2024 | Creación y preparación de la tabla de Trello | Configuration | Completado | 0.5 | 0.25 |
| 19/02/2024 | Creación del repositorio de GitHub | Configuration | Completado | 0.5 | 0.25 |
| 20/02/2024 | Instalación y configuración de Mendeley | Configuration | Completado | 1 | 1 |
| 21/02/2024 | Estudio de GitHub | Research | En curso | 1.5 | 2 |
| 22/02/2024 | Estudio de GitHub | Research | En curso | 1.5 | 1.5 |
| 23/02/2024 | Estudio de GitHub | Research | Completado | 1 | 1 |
| 26/02/2024 | Investigar sobre aplicaciones web | Research | Completado | 2 | 1.5 |
| 27/02/2024 | Investigar sobre aplicaciones escritorio | Research | Completado | 2 | 1.5 |
| 28/02/2024 | Investigación general de herramientas OMR | Research | Completado | 2 | 1.5 |
| 29/02/2024 | Decidir entorno en el que desarrollar la memoria | Research | Completado | 1 | 0.75 |

* + 1. **Sprint 2: 01/03/2024 –17/03/2024**

Este sprint se dedicó el trabajo a la investigación y el estudio de las herramientas a utilizar en el proyecto para levantar un servidor web y desarrollar la aplicación.

Entre ellas se valoró Flask o Django. Flask ofrece mucha simplicidad y flexibilidad dejando libertad para la implementación de otras herramientas y librerías, además, su aprendizaje es muy rápido permitiendo tener un mayor control en el desarrollo de la aplicación. Sin embargo, puede ser un problema si el proyecto crece lo suficiente con demasiadas funcionalidades, ya que habría que implementar muchas características. Por otro lado, Django es un framework que ofrece una amplia variedad de funcionalidades integradas de serie, también no permite un desarrollo tan rápido ya que está enfocado en aplicaciones complejas haciendo uso de las características de este framework, así como escalabilidad para cualquier proyecto. Valorando todos estos aspectos y analizando la presente y futura situación he considerado que la mejor opción es usar Flask para mi proyecto.

También realicé una investigación de los paquetes OMR más sólidos y de código abierto en la actualidad. La elección más llamativa es Audiveris debido a su larga trayectoria y la comunidad que hay detrás ya que es muy activa y grande, además de por su amplia documentación, explicado cada punto con detalle. Otras tecnologías que valoré son SharpEye Music Reader, MuseScore.

En este sprint también cambié el plan estructural de las tablas de Trello y la base de datos donde guardo las referencias de Mendeley a Zotero.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha | Tarea | Etiqueta | Estado | Tiempo estimado | Tiempo real |
| 06/03/2024 | Cambiar el plan estructural de Trello | Documentation | Completado | 0.5 | 0.3 |
| 07/03/2024 | Cambiar la base de datos de referencias de Mendeley a Zotero | Documentation | Completado | 0.25 | 0.25 |
| 10/03/2024 | Documentar Sprint 1 | Documentation | En curso | 1 | 2 |
| 11/03/2024 | Documentar Sprint 1 | Documentation | Completado | 1 | 1 |
| 12/03/2024 | Documentar Sprint 2 y avances de los paquetes OMR en la memoria | Documentation | En curso | 1 | 1.5 |
| 12/03/2024 | Investigar servidor web: Django vs Flask | Research | En curso | 1 | 1 |
| 13/03/2024 | Investigar servidor web: Django vs Flask | Research | En curso | 2 | 2 |
| 14/03/2024 | Investigar servidor web: Django vs Flask | Research | Completado | 1 | 2 |
| 16/03/2024 | Investigar paquetes OMR para mi proyecto y elegir el más ideal | Research | Completado | 2 | 2.5 |

* + 1. **Sprint 3: 18/03/2024 – 15/04/2024**

En esta etapa comencé con el desarrollo de la aplicación web, enfocándome en las funcionalidades más básicas de esta.

En primer lugar planteé una primera versión de los tipos de usuarios que habrá en mi aplicación, empezando por un usuario corriente y un usuario registrado. Además le asigné unas primeras funcionalidades básicas a estos usuarios, que son entrar a la ventana home, logearse y registrarse en el caso del usuario corriente y poder subir ficheros y hacer log out en el caso de un usuario registrado como funciones extra para este.

Tras ello creé un entorno virtual en el que trabajaré para el desarrollo de mi aplicación. Tomé esta decisión con el fin de evitar conflictos de dependencias de paquetes y versiones con otros proyectos.

Después creé una primera versión de mi proyecto siguiendo una estructura de ficheros común en este tipo de proyectos, luego instalé Flask. Gestionar el fichero de configuración en este punto fue clave para poder hacer las migraciones del modelo de las tablas a la base de datos.

Hice una primera implementación de un login y register haciendo uso de la librería de FlaskWTF para facilitarme hacer los forms. Como tuve problemas con el uso adecuado de esta herramienta cambié el código, retirando esta librería.

Para poder almacenar las entradas en base de datos los usuarios registrados instalé xampp y configuré la base de datos en phpadmin. Elegí este entorno debido a su facilidad de uso y su libre acceso a cualquier navegador web.

Para reflejar los modelos creados en código en base de datos, uso migraciones mediante comandos en la terminal para que las tablas se creen automáticamente. Estos pasos me dieron varios problemas debido a que la carpeta de migraciones almacenó más de un cambio y se solaparon entre ellos. Además los ficheros de la caché almacenaban datos antiguos apareciendo atributos fantasma. Para arreglarlo borré la carpeta y recreé la base de datos.

Por último terminé de vincular las ventanas y de estructurar la aplicación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha | Tarea | Etiqueta | Estado | Tiempo estimado | Tiempo real |
| 19/03/2024 | Documentar Sprint 2 | Documentation | Completado | 1 | 1.5 |
| 20/03/2024 | Cambiar las gráficas de los sprints | Documentation | Completado | 1 | 1 |
| 21/03/2024 | Plantear los tipos de usuarios, sus roles y funcionalidades | Research | Completado | 4 | 0.5 |
| 21/03/2024 | Preparar la aplicación web | Development | Completado | 2 | 1.5 |
| 24/03/2024 | Desarrollo de ventana login y register | Development | En curso | 2 | 2 |
| 01/04/2024 | Desarrollo de ventana login y register | Development | En curso | 2 | 1.6 |
| 02/04/2024 | Desarrollo de ventana login y register | Development | En curso | 3 | 3 |
| 03/04/2024 | Instalación e implementación de xampp | Configuration | Completado | 1 | 1.5 |
| 03/04/2024 | Instalación e implementación de phpMyAdmin | Configuration | Completado | 2 | 2 |
| 04/04/2024 | Desarrollo de ventana login y register | Development | En curso | 2.5 | 2.5 |
| 07/04/2024 | Desarrollo de ventana login y register | Development | En curso | 2 | 2 |
| 08/04/2024 | Desarrollo de ventana login y register | Development | En curso | 3 | 3 |
| 09/04/2024 | Desarrollo de ventana login y register | Development | En curso | 3 | 3 |
| 10/04/2024 | Desarrollo de ventana login y register | Development | Completado | 1 | 1 |
| 10/04/2024 | Vinculación entre ventanas | Development | Completado | 1 | 1 |
| 10/04/2024 | Desarrollo de registro de partituras | Development | Completado | 1 | 1 |
| 11/04/2024 | Gestión de usuarios y sesiones | Development | Completado | 2 | 2 |
| 15/04/2024 | Documentación del sprint 3 | Documentation | Completado | 3 | 3 |

* + 1. **Sprint 4: 15/04/2024 – 22/05/2024**

Este sprint está enfocado en la importación de la tecnología OMR Audiveris en mi aplicación.

Para ello, primero, tuve que investigar y revisar bien la documentación que aportan los autores de la herramienta o los desarrolladores interesados en esta. Cloné la herramienta en mi sistema y la ejecuté.

Tuve problemas para ejecutarlo porque el nombre de los ficheros y archivos no se correspondían del todo con los ejemplos de la documentación.

Analicé e investigué los diferentes aspectos del programa para saber cómo se trabaja con esta, a la par que me documentaba con el pdf de ayuda para los desarrolladores.

Luego tuve que cambiar el formato en el que se aceptaba los archivos subidos a mi aplicación (solo se podía pdfs) para darle flexibilidad a la aplicación.

Creé un proyecto en Google Cloud para tener un servicio de almacenamiento para las partituras en la nube, me dio alguna dificultad adaptarlo a mi proyecto por lo que deseché la idea por no malgastar tiempo.

Intenté encapsular mi proyecto junto con audiveris para moverlo a un servicio de nube que desplegara mi proyecto públicamente para abrir su acceso a los usuarios, también aseguraría que el comportamiento es el mismo en un entorno de producción y localmente.

Tuve que registrar todas las dependencias del proyecto y registrar las instalaciones del terminal del docker en un fichero que ejecutaría más tarde. Tuve problemas por la compatibilidad de los paquetes al construir el docker y, tras construirlo, el acceso a las rutas de las carpetas tuvo muchas ambigüedades, las cuales tuve que resolver.

Otro inconveniente fue que en un inicio por cada cambio en el código del proyecto tuve que construir un nuevo contenedor e imagen, por lo que el nombre de estos se pisaban, ralentizaban el ordenador y colapsaron el almacenamiento. Dejé aparcada la idea debido a la ralentización del proceso y la incertidumbre de si llegaría a algo funcional.

Volví con la programación de una función que ejecutara Audiveris como un miniservicio de mi aplicación en segundo plano. Me documenté de las vías de ejecución que tiene Audiveris en su documentación para construir un comando que ejecute el servicio sin interfaz de usuario, haga el proceso de digitalización y exporte el fichero resultante.

La construcción de este comando nuevamente dio problemas que llevaron un tiempo resolver. Esto se debe a que la ubicación de los ficheros de ejecución de java (.jar) no eran accesibles para la carpeta del proyecto. Cloné el repositorio de Audiveris a el directorio de la aplicación web para facilitar el acceso. También tuve que cambiar la versión de mi carpeta jdk de Java a la 17 y configurarlo en mis variables de entorno para asegurar un buen comportamiento, como indican las especificaciones de la documentación de Audiveris.

Mejoré las vistas de mi aplicación web con bootstrap para darle un mejor diseño a la web y un toque más profesional.

Exploré las posibilidades para el acceso a metadatos en base de datos en Firebase. La configuración y adaptación de este servicio a mi proyecto llevaron tiempo, debido a las dificultades que se me presentaron para la importación de paquetes así como adaptarme a la nueva estructura, sintaxis y funciones de este.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha | Tarea | Etiqueta | Estado | Tiempo estimado | Tiempo real |
| 16/04/2024 | Documentación del sprint 3 y cambios en las tablas y gráficos | Documentation | Completado | 0.5 | 0.5 |
| 23/04/2024 | Investigación para la implementación de la tecnología OMR | Research | En curso | 2 | 2 |
| 24/04/2024 | Investigación para la implementación de la tecnología OMR | Research | En curso | 2 | 2 |
| 25/04/2024 | Investigación para la implementación de la tecnología OMR | Research | En curso | 2 | 2 |
| 27/04/2024 | Subir archivos en todos los formatos | Development | Completado | 2 | 2 |
| 27/04/2024 | Investigación para la implementación de la tecnología OMR | Research | En curso | 3 | 3 |
| 30/04/2024 | Investigación para la implementación de la tecnología OMR | Research | En curso | 1.5 | 1 |
| 01/05/2024 | Investigación para la implementación de la tecnología OMR | Research | En curso | 4 | 4 |
| 03/05/2024 | Implementación de servicio de almacenamiento | Development | En curso | 3 | 2 |
| 04/05/2024 | Implementación de servicio de almacenamiento | Development | En curso | 5 | 5 |
| 05/05/2024 | Implementación de un docker para Audiveris | Development | En curso | 3 | 3 |
| 06/05/2024 | Implementación de un docker para Audiveris | Development | En curso | 5.5 | 5 |
| 07/05/2024 | Implementación de un docker para Audiveris | Development | En curso | 3 | 3 |
| 08/05/2024 | Implementación de un docker para Audiveris | Development | En curso | 2 | 2 |
| 09/05/2024 | Implementación de un docker para Audiveris | Development | En curso | 2.5 | 2.5 |
| 11/05/2024 | Implementación de un docker para Audiveris | Development | Completado | 1 | 1 |
| 11/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | En curso | 2 | 2.5 |
| 12/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | En curso | 7.5 | 8 |
| 13/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | En curso | 5 | 5 |
| 14/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | En curso | 3.5 | 3.5 |
| 15/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | En curso | 4 | 4 |
| 17/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | En curso | 3 | 3 |
| 18/05/2024 | Implementación de una funcionalidad que ejecute audiveris | Development | Completado | 3 | 4.5 |
| 18/05/2024 | Mejora de las vistas | Development | Completado | 2 | 2 |
| 18/05/2024 | Implementación de Firebase | Research | En curso | 2 | 2 |
| 19/05/2024 | Implementación de Firebase | Development | En curso | 2 | 2 |
| 19/05/2024 | Implementación de Firebase | Development | En curso | 3 | 3 |
| 20/05/2024 | Implementación de Firebase | Development | En curso | 1.5 | 2 |
| 20/05/2024 | Implementación de Firebase | Development | En curso | 3 | 5.5 |
| 21/05/2024 | Implementación de Firebase | Development | En curso | 5 | 5 |
| 22/05/2024 | Implementación de Firebase | Development | En curso | 5 | 7 |

* + 1. **Sprint 5: 23/05/2024 – 20/06/2024**

En este sprint valoré otras alternativas para la digitalización de partituras. Consideré hacer uso de Firebase ML que permite integrar aprendizaje automático en aplicaciones web. De esta forma podría implementarlo en mi web para que, en vez de usar Audiveris para transcribir las partituras, usar esta IA. Valoré la posibilidad de usar un modelo preentrenado con esta funcionalidad para facilitar el proceso, pero no obtuve resultados viables. Investigué como podría entrenar una por mi cuenta en base a mis preferencias, pero no lo llevé adelante debido a la falta de experiencia en este sector y el tiempo que me llevaría desarrollarlo y en base al tiempo que me quedaba.

También empecé la documentación de la memoria empezando por puntos esenciales y sencillos de implementar, como los objetivos de desarrollo sostenible o la introducción.

Retomé la funcionalidad de ejecutar Audiveris construyendo un comando que se ejecutaría como batch en segundo plano, además también replanteé la estructura de las funciones de cada archivo del directorio, ordenando las definiciones, evitando que Firebase se inicializara más veces de las necesarias.

El siguiente paso fue implementar una funcionalidad que permitiera visualizar las partituras que se consiguieran digitalizar. Para ello usé Verovio, instalándolo junto con sus dependencias en mi máquina. Tuve bastantes problemas en este punto ya que hubo muchos conflictos con la instalación de las dependencias y con las correcta configuración y establecimiento en mis variables de entorno. Incluí la funcionalidad de visualizar las partituras y Verovio en las vistas con bibliotecas, mostrándose las partituras mediante bloques y scripts.

Luego implementé una funcionalidad para que el usuario pudiera eliminar las partituras que desee en el listado de partituras registradas en la aplicación y Firebase. Aparqué la implementación debido a que me dio problemas con el acceso al bucket y al servicio de almacenamiento de Firebase.

Añadí alertas a las funcionalidades de la aplicación para que el usuario se mantenga informado acerca de los procesos fallidos al intentar ciertas interacciones.

Aprovechando esta última implementación, añadí una funcionalidad para que, en caso de que la digitalización de una partitura fallé, el usuario pueda alterar la imagen llegando a un proceso de preprocesamiento, facilitando a Audiveris poder procesar la imagen y llegar a una digitalización completa. Esta funcionalidad consiste en que el usuario inserta valores a funciones de preprocesamiento.

Por último, empecé el desarrollo para desplegar la aplicación públicamente. Para ello tomé dos caminos, el primero fue desplegar la aplicación haciendo uso de un archivo Procfile y buildpacks en Heroku. Esta opción me dio problemas debido a que Heroku no permite la ejecución de scripts en tiempo de compilación sin configurar sus buildpacks, y ello conllevaba realizar las instalaciones necesarias de Audiveris en estos. Además los archivos que se generaban con los scripts no persistían por alguna razón. La opción que dio resultados fue utilizar el Docker creado para tener un mejor control del entorno de ejecución y usar Heroku para configurar la aplicación y desplegarla. También dio problemas por no estar bien adaptada al contexto la inicialización de la aplicación en código y los errores en las interpretaciones del puerto en el Dockerfile al desplegar la aplicación con Heroku. El despliegue está terminado pero la funcionalidad de digitalizar las partituras no se puede completar en la versión desplegada de Heroku ya que sobrepasa la memoria permitida en mi plan gratuito de mi cuenta.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fecha | Tarea | Etiqueta | Estado | Tiempo estimado | Tiempo real |
| 23/05/2024 | Documentar sprint 4 | Documentation | Completado | 2 | 3 |
| 24/05/2024 | Investigar herramientas de procesamiento de imágenes para Firebase | Research | En curso | 0.4 | 0.4 |
| 24/05/2024 | Investigar herramientas de procesamiento de imágenes para Firebase | Research | Completado | 1 | 1 |
| 25/05/2024 | Documentación de la memoria | Documentation | En curso | 1 | 1.5 |
| 26/05/2024 | Reimplementación de un docker | Development | Completado | 1 | 2 |
| 26/05/2024 | Documentación de la memoria | Documentation | En curso | 1 | 1 |
| 27/05/2024 | Reimplementación de audiveris | Development | En curso | 2 | 2 |
| 28/05/2024 | Reimplementación de audiveris | Development | Completado | 2 | 3 |
| 28/05/2024 | Reestructuración para Firebase | Development | Completado | 2 | 3.5 |
| 31/05/2024 | Documentar la memoria | Documentation | En curso | 1 | 2.5 |
| 01/06/2024 | Implementación de Verovio | Development | En curso | 3 | 3 |
| 02/06/2024 | Implementación de Verovio | Development | En curso | 2 | 2 |
| 03/06/2024 | Implementación de Verovio | Development | En curso | 2 | 2 |
| 06/06/2024 | Implementación de Verovio | Development | En curso | 6 | 6 |
| 07/06/2024 | Implementación de Verovio | Development | En curso | 7 | 7 |
| 08/06/2024 | Implementación de Verovio | Development | En curso | 8 | 10 |
| 09/06/2024 | Implementación de Verovio | Development | Completado | 9 | 5 |
| 09/06/2024 | Funcionalidad para eliminar partituras | Development | Bloqueado | 2 | 2 |
| 10/06/2024 | Arreglos en la aplicación | Development | Completado | 3 | 6.5 |
| 11/06/2024 | Alertas en las vistas | Development | Completado | 4 | 6 |
| 12/06/2024 | Función de preprocesamiento | Development | Completado | 4 | 5.5 |
| 13/06/2024 | Despliegue de la aplicación | Development | En curso | 5 | 8 |
| 14/06/2024 | Despliegue de la aplicación | Development | En curso | 4 | 8 |
| 15/06/2024 | Despliegue de la aplicación | Development | En curso | 5 | 10 |
| 16/06/2024 | Despliegue de la aplicación | Development | Bloqueado | 5 | 7.5 |
| 17/06/2024 | Documentación del sprint 5 | Documentation | Completado | 3 | 5 |
| 18/06/2024 | Documentación de la memoria | Documentation | En curso | 5 | 7 |
| 19/06/2024 | Documentación de la memoria | Documentation | En curso | 7.5 | 7.5 |
| 20/06/2024 | Funcionalidad para eliminar partituras | Development | Completado | 3 | 3 |
| 20/06/2024 | Documentación de la memoria | Documentation | En curso | 4 | 4 |

* + 1. **Sprint 6: 21/06/2024 –**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Tarea | Etiqueta | Estado | Tiempo estimado | Tiempo real |
| 21/06/2024 | Documentación de la memoria | Documentation | En curso | 8 | 8 |
| 22/06/2024 | Documentación de la memoria | Documentation | En curso | 8 | 9 |
| 23/06/2024 | Documentación de la memoria | Documentation | En curso | 10 | 10 |
| 24/06/2024 | Documentación de la memoria | Documentation | En curso | 5 | 6.5 |
| 26/06/2024 | Documentación de la memoria | Documentation | En curso | 6.5 | 6 |
| 27/06/2024 | Documentación de la memoria | Documentation | En curso | 7 | 7.5 |
| 28/06/2024 | Documentación de la memoria | Documentation | En curso | 7 | 9 |
| 29/06/2024 | Documentación de la memoria | Documentation | Completado | 1 | 3 |
| 29/06/2024 | Documentación de los anexos | Documentation | En curso | 5 | 5 |
| 30/06/2024 | Documentación de los anexos | Documentation | En curso | 7 | 7 |
| 01/07/2024 | Documentación de los anexos | Documentation | En curso | 7 | 7 |
| 02/07/2024 | Documentación de los anexos | Documentation | En curso | 7 | 8.5 |
| 03/07/2024 | Estelización de las vistas | Development | Completado | 6 | 9 |
| 04/07/2024 | Documentación de los anexos | Documentation | En curso | 4 | 4 |
| 04/07/2024 | Arreglos en código | Development | Completado | 3 | 3 |
| 05/07/2024 | Comentar el código | Development | Completado | 1 | 1 |
| 05/07/2024 | Documentación de los anexos | Development | En curso | 8 | 8 |

* + 1. **Histórico de los sprints**

|  |  |  |  |
| --- | --- | --- | --- |
| Tarea | Etiqueta | Tiempo estimado | Tiempo real |
| Creación y preparación de la tabla de Trello | Configuration | 0.5 | 0.25 |
| Creación del repositorio de GitHub | Configuration | 0.5 | 0.25 |
| Instalación y configuración de Mendeley | Configuration | 1 | 1 |
| Estudio de GitHub | Research | 4 | 4 |
| Investigar sobre aplicaciones web | Research | 2 | 1.5 |
| Investigar sobre aplicaciones escritorio | Research | 2 | 1.5 |
| Investigación general de herramientas OMR | Research | 2 | 1.5 |
| Decidir entorno en el que desarrollar la memoria | Research | 1 | 0.75 |
| Cambiar el plan estructural de Trello | Documentation | 0.5 | 0.3 |
| Cambiar la base de datos de referencias de Mendeley a Zotero | Documentation | 0.25 | 0.25 |
| Documentar Sprint 1 | Documentation | 2 | 3 |
| Documentar Sprint 2 | Documentation | 1.5 | 2.25 |
| Avances de los paquetes OMR | Documentation | 0.5 | 0.75 |
| Investigar servidor web: Django vs Flask | Research | 4 | 5 |
| Investigar paquetes OMR | Research | 2 | 2.5 |
| Cambiar las gráficas de los sprints | Documentation | 1 | 1 |
| Plantear los tipos de usuarios, sus roles y funcionalidades | Research | 4 | 0.5 |
| Preparar la aplicación web | Development | 2 | 1.5 |
| Desarrollo de ventana login y register | Development | 17.5 | 18.1 |
| Instalación e implementación de xampp | Configuration | 1 | 1.5 |
| Instalación e implementación de phpMyAdmin | Configuration | 2 | 2 |
| Vinculación entre ventanas | Development | 1 | 1 |
| Desarrollo de registro de partituras | Development | 1 | 1 |
| Gestión de usuarios y sesiones | Development | 2 | 2 |
| Documentación del sprint 3 | Documentation | 3 | 3 |
| Documentación del sprint 3 y cambios en las tablas y gráficos | Documentation | 0.5 | 0.5 |
| Investigación para la implementación de la tecnología OMR | Research | 14.5 | 14 |
| Implementación de servicio de almacenamiento | Development | 8 | 7 |
| Implementación de un docker para Audiveris | Development | 17 | 16.5 |
| Implementación de una funcionalidad que ejecute audiveris | Development | 28 | 30.5 |
| Mejora de las vistas | Development | 2 | 2 |
| Implementación de Firebase | Development | 21.5 | 26.5 |
| Documentar sprint 4 | Documentation | 2 | 3 |
| Investigar herramientas de procesamiento de imágenes para Firebase | Research | 1.4 | 1.4 |
| Reimplementación de un docker | Development | 1 | 2 |
| Reimplementación de audiveris | Development | 4 | 5 |
| Reestructuración para Firebase | Development | 2 | 3.5 |
| Implementación de Verovio | Development | 37 | 35 |
| Funcionalidad para eliminar partituras | Development | 5 | 5 |
| Arreglos en la aplicación | Development | 3 | 6.5 |
| Alertas en las vistas | Development | 4 | 6 |
| Función de preprocesamiento | Development | 4 | 5.5 |
| Despliegue de la aplicación | Development | 19 | 13.5 |
| Documentación del sprint 5 | Documentation | 3 | 5 |
| Documentación de la memoria | Documentation | 72 |  |
| Documentación de los anexos | Documentation | 39.5 | 40.5 |
| Estelización de las vistas | Development | 6 | 9 |

1. **Viabilidad económica**

En este apartado se va a evaluar la rentabilidad que supondría el desarrollo de este proyecto en el supuesto de que fuera producido empresarialmente. Para ello se van a analizar los costes en base a datos reales de proyectos cotidianos.

* + 1. **Coste personal**

En este apartado se van a evaluar los costes de personal que tendría el proyecto en base a los empleados. En el desarrollo de este proyecto han entrado en juego un único desarrollador *junior* trabajando mas o menos 4 meses, con un total de 327,3.

Se toma el salario promedio de un programador junior, que es 21000 € brutos al año, siendo el precio por hora de 10,77 €. Calculamos el precio total en función de las horas invertidas en el proyecto y concluimos con un total de 3525,021 €, que vendrían siendo 881,25 € mensuales brutos. (1)

Ahora hay que añadir al análisis los impuestos con los que tiene que contar la empresa. En España, las empresas tienen que pagar impuestos a la Seguridad Social por cada empleado. Teniendo en cuenta esto y sabiendo que el porcentaje de cotización a la Seguridad Social para la empresa es del 30% sobre el salario del empleado. (2)

Impuestos=3525,021€×0.30=1057,506€

Por lo tanto, el costo total del proyecto, incluyendo los impuestos, sería:

Costo total del proyecto=3525,021€+1057,506€=4582,527€

* + 1. **Hardware**

En la sección de costes de hardware, es necesario calcularlos en función del material usado en su desarrollo.

En cuanto al ordenador se ha hecho uso de un portátil Lenovo Ideapad Gaming de 829 € y de pantalla se ha usado uno monitor Philips de 90€. Calculando que la amortización de todo el material va a llegar a los 4 años, el cálculo ser

919 / 4 = 229,75

Se hace la operación correspondiente para sacar su costo por mes y se multiplica por el numero de meses de desarrollo:

(229,75€ al año / 12 meses) \* 4 meses = 76 €

* + 1. **Software**

En este apartado se va a evaluar el coste que ha tenido el uso del software.

Al haber usado librerías y bibliotecas totalmente gratuitas, en lo que respecta a esto el coste es 0 €, si bien algunas de las herramientas usadas tienen planes de pago, no han sido usados en el proyecto por lo que no se va a tener en cuento.

El software utilizado en el proyecto es totalmente gratuito y público, teniendo un coste total de 0 €.

* + 1. **Beneficios**

Este proyecto ha sido desarrollado sin intención de sacar beneficio económico de él, cuyo fin es aumentar las posibilidades al estudio de material musical.

|  |  |
| --- | --- |
| **Concepto** | **Coste (€)** |
| Personal | 4582,527 |
| Hardware | 76 |
| Software | 0 |
| **Total** | 4.658,527 |

* + 1. **Viabilidad legal**

En esta sección se va a analizar profundamente las leyes y regulaciones que tiene que cumplir el proyecto en el desarrollo y en el uso de la aplicación.

Para ello hay que analizar cada una de las librerías y herramientas usadas en el proyecto y comprobar y estudiar su licencia.

|  |  |  |
| --- | --- | --- |
| **Herramienta** | **Versión** | **Licencia** |
| GitHub | 2.43.0. | MIT License |
| Scrum | - | CC BY-SA 4.0 |
| GitHub | - | MIT License |
| Trello | - | - |
| Microsoft Word | - | - |
| GitHub Desktop | - | - |
| Zotero | - | AGPL |
| Python | 3.11.4 | PSF License |
| Flask | 3.0.2 | BSD-3-Clause |
| Bootstrap | 5.3.2 | MIT |
| Cloud Firestore | 2.16.0 | - |
| Google Cloud Storage | 2.16.0 | - |
| Bcrypt | 4.1.3 | ISC |
| pdf2image | 1.17.0 | GPL |
| OpenCV | 4.10.0.82 | Apache 2.0 |
| NumPy | 1.26.4 | BSD-3-Clause |
| Audiveris | 5.1.1 | AGPL |
| Verovio | 3.9.0 | LGPL |
| Docker | 24.0.5 | Apache 2.0 |

Prácticamente la totalidad de las librerías y herramientas del proyecto son de libre uso y públicas, aunque algunas de ellas se someten a ciertas condiciones que la aplicación debe cumplir para su uso. Las licencias mas restrictivas de las anteriores son:

**General Public License (GPL)**

Esta licencia exige que el software que incorpore GPL sea liberado con la misma licencia cuando se ditribuya, con sus modificaciones y extensiones.

**Affero General Public License (AGPL)**

Es muy similar a GPL pero en este caso al hacer una modificación y uso del software bajo APGL debe tener disponible el código fuente modificado a los usuarios a través de la red.

**Lesser General Public License (LGPL)**

Es más flexible que GPL, siendo necesario simplemente que cualquier modificación en el software en cuestión tenga que ser liberada bajo la misma licencia.

**Anexo B Especificación de Requisitos**

* 1. **Introducción**

A continuación, se va a desarrollar el apartado referente a los objetivos y requisitos necesarios para el proyecto. De esta manera se podrán encontrar las funcionalidades necesarias para crear los casos de uso de la aplicación.

* 1. **Objetivos generales**

Para crear los casos de uso, primero es necesario tener claros los objetivos generales del proyecto. Los objetivos generales son los siguientes:

* Incorporar una autenticación segura y efectiva.
* Dar un servicio de almacenamiento de partituras online para cualquier usuario con una interacción cómoda y rápida.
* Implementar un proceso de transcripción de partituras eficiente a través del uso del software de reconocimiento óptico de música (OMR).
* Proporcionar un sistema de almacenamiento independiente para las partituras que han sido digitalizadas.
* Implementar una funcionalidad de eliminación de partituras tanto subidas sin digitalizar, como partituras digitalizadas.
* Implementar una opción de preprocesamiento manual mediante el uso de tecnologías de procesamiento de imágenes.
* Implementación de la posibilidad de visualización de las partituras digitalizadas.
  1. **Catálogo de requisitos**

El siguiente paso va a ser definir los requisitos funcionales y no funcionales que tiene el proyecto.

* + 1. **Requisitos funcionales**
* R.F-01. Inicio de sesión: El usuario debe poder iniciar sesión.
* R.F-02. Registro de usuario: El usuario debe poder registrarse.
* R.F-03. Cerrar sesión: La aplicación debe permitir al usuario cerrar sesión.
* R.F-04. Subir partituras: El usuario debe poder subir una partitura en cualquier formato.
* R.F-05. Listar las partituras registradas: La aplicación debe permitir al usuario ver su lista de partituras registradas.
* R.F-06. Listar las partituras digitalizadas: La aplicación debe permitir al usuario ver su lista de partituras digitalizadas.
* R.F-07. Digitalizar una partitura: El usuario debe poder digitalizar una partitura de su lista de partituras registradas.
* R.F-08. Preprocesar una partitura: El usuario debe poder preprocesar una partitura de su lista de partituras registradas.
* R.F-09. Visualizar una partitura: El usuario debe poder visualizar una partitura digitalizada.
  + 1. **Requisitos no funcionales**
* R.NF-01. Seguridad: La aplicación debe asegurar la protección de los datos del usuario haciendo uso de métodos de encriptación para el almacenamiento de las credenciales.
* R.NF-02. Disponibilidad: La aplicación debe estar disponible durante el periodo de actividad.
* R.NF-03. Escalabilidad: La aplicación debe ser capaz de soportar y tener una capacidad de integración sencilla a modificaciones y aumento de funcionalidades.
* R.NF-04. Usabilidad: La interfaz de usuario debe ser intuitiva y fácil de usar, permitiendo a los usuarios realizar todas las operaciones con no más de tres clics.
* R.NF-05. Compatibilidad: La aplicación debe ser compatible con los principales sistemas operativos de escritorio, incluyendo Windows, macOS, Android e iOS.
* R.NF-06. Mantenibilidad: El código de la aplicación debe seguir una buena metodología y estructura de programación para asegurar que sea fácil de mantener y actualizar.
  1. **Especificación de requisitos**

En esta sección se van a especificar los casos de uso en base a los requisitos descritos anteriormente.

* + 1. **Casos de uso**

**Casos de Uso de Autenticación de Usuario**

CU-1 Inicio de sesión

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios iniciar sesión de forma segura.

Actores Usuario.

Precondiciones El usuario debe estar registrado.

Acciones 1. El usuario introduce sus credenciales.

2. El sistema verifica las credenciales.

3. El usuario es autenticado y redirigido al menú de la aplicación.

Postcondiciones El usuario ha iniciado sesión.

Requisitos asociados R.F-01, R.NF-01

CU-2 Registro de usuario

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios registrarse en el sistema.

Actores Usuario.

Precondiciones Ninguna.

Acciones 1. El usuario entra en el formulario de registro

2. El usuario completa todos los campos del formulario.

3. El sistema guarda los datos y registra al usuario.

Postcondiciones El usuario ha sido registrado.

Requisitos asociados R.F-02, R.NF-01

CU-3 Cerrar sesión

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios cerrar sesión en el sistema.

Actores Usuario.

Precondiciones El usuario debe haber iniciado sesión.

Acciones 1. El usuario selecciona la opción de cerrar sesión.

2. El sistema cierra la sesión del usuario.

Postcondiciones El usuario ha cerrado sesión.

Requisitos asociados R.F-03, R.NF-01

**Casos de Uso de Gestión de Partituras**

CU-4 Subir una partitura

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios subir una partitura en cualquier formato.

Actores Usuario.

Precondiciones El usuario debe estar autenticado.

Acciones 1. El usuario selecciona la opción de subir una partitura.

2. El usuario introduce el nombre para la partitura.

3. El usuario selecciona el archivo de la partitura en su sistema.

4. El usuario selecciona la opción de subir la partitura.

5. El sistema guarda el archivo con los datos y lo asocia al usuario.

Postcondiciones La partitura es subida y registrada.

Requisitos asociados R.F-04

CU-5 Listar las partituras registradas

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios ver su lista de partituras registradas.

Actores Usuario.

Precondiciones El usuario debe estar autenticado.

Acciones 1. El usuario selecciona la opción de ver sus partituras registradas.

2. El sistema muestra la lista de partituras registradas.

Postcondiciones La lista de partituras es visualizada.

Requisitos asociados R.F-05

CU-6 Listar las partituras digitalizadas

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios ver su lista de partituras digitalizadas.

Actores Usuario.

Precondiciones El usuario debe estar autenticado.

Acciones 1. El usuario selecciona la opción de ver sus partituras digitalizadas.

2. El sistema muestra la lista de partituras digitalizadas.

Postcondiciones La lista de partituras digitalizadas es visualizada.

Requisitos asociados R.F-06

CU-7 Digitalizar una partitura

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios digitalizar una partitura de su lista de partituras registradas.

Actores Usuario.

Precondiciones El usuario debe estar autenticado, tener partituras registradas y estar en la lista de partituras registradas.

Acciones 1. El usuario selecciona el botón de digitalizar de la partitura en el listado.

2. El sistema digitaliza la partitura.

Postcondiciones La partitura es digitalizada y guardada.

Requisitos asociados R.F-07

CU-8 Preprocesar una partitura

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios preprocesar una partitura de su lista de partituras registradas.

Actores Usuario.

Precondiciones El usuario debe estar autenticado, tener partituras registradas y estar en la vista de la lista de partituras registradas.

Acciones 1. El usuario selecciona el botón de preprocesar de la partitura en el listado.

2. El sistema muestra el formulario de los valores de las funciones de procesamiento.

3. El usuario introduce los valores de las funciones de procesamiento.

4. El sistema aplica las técnicas de procesamiento en la imagen de la partitura en cuestión.

5. El sistema comienza la descarga del nuevo archivo.

Postcondiciones La partitura es preprocesada y descargada.

Requisitos asociados R.F-08

CU-9 Visualizar una partitura digitalizada

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios visualizar una partitura de su lista de partituras registradas.

Actores Usuario.

Precondiciones El usuario debe estar autenticado, tener partituras digitalizadas y estar en la vista de la lista de partituras digitalizadas.

Acciones 1. El usuario selecciona el botón de visualizar de la partitura en el listado.

2. El sistema muestra una vista de la partitura.

Postcondiciones La partitura es visualizada.

Requisitos asociados R.F-09

**Casos de Uso del Menú**

CU-10 Ir al menú

Versión 1.0

Autor Martín González Saiz.

Descripción Permitir a los usuarios visualizar el menú de acciones

Actores Usuario.

Precondiciones El usuario debe estar autenticado.

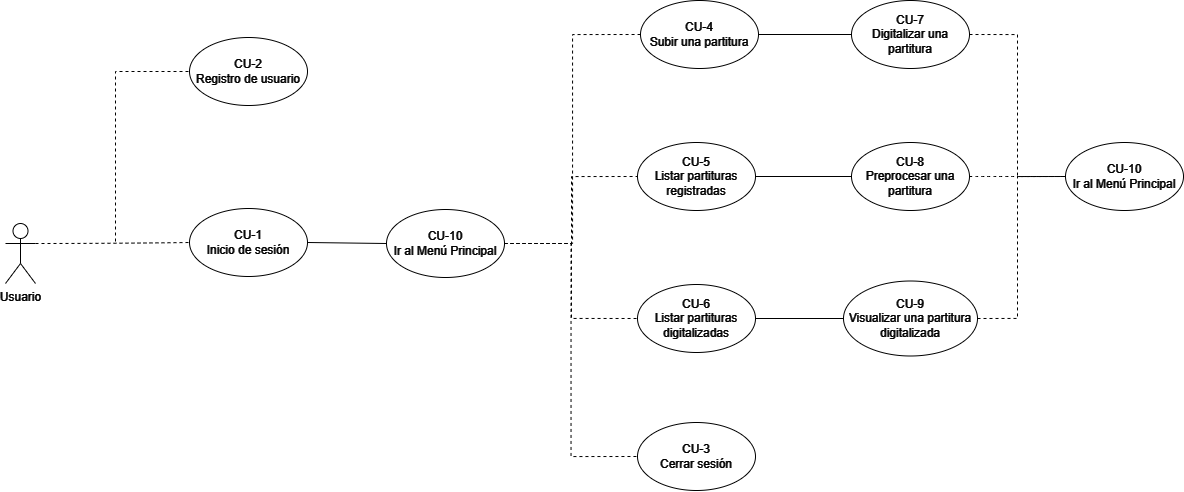
Acciones 1. El usuario completa el proceso de inicio de sesión.

2. El sistema redirige al usuario al menú principal.

Postcondiciones El usuario ha sido redirigido al menú principal

Requisitos asociados R.F-01

* 1. **Diagrama de Casos de Uso**



**Anexo C Especificación de diseño**

* 1. **Introducción**

En esta parte de la documentación del proyecto se va a explicar la estructura y modelo de datos que se ha seguido, teniendo en cuenta que para el desarrollo de esta aplicación se ha hecho uso de servicios externos.

* 1. **Diseño de datos**

En el desarrollo de la aplicación se utilizan distintos conceptos de estructura de datos, debido a que se hace uso de servicios externos como Firebase, por ello se va a seccionar el diseño de datos en distintos apartados.

* + 1. **Modelos de datos**

**User**

Este modelo representa a un usuario que se ha registrado en la aplicación tras rellenar el correspondiente formulario, siendo apto en la autenticación y autorizado para interactuar con la aplicación en ciertas funcionalidades.

Sus atributos son los siguientes:

* id (str): Nombre del usuario único que actúa como clave primaria.
* email (str): Direccion de correo electrónico del usuario.
* passwordHash (str): Contraseña usuario encriptada para aumentar la seguridad.

Y sus métodos:

* \_\_init\_\_(username, email, passwordHash): Es el constructor del modelo de usuarios

**SheetMusic**

Este modelo representa los registros de datos referentes a una partitura subida por un usuario.

Sus atributos son los siguientes:

* title (str): Título de la partitura.
* file\_path (str): Ruta del archive de la partitura almacenada en Firebase Storage.
* upload\_date (datetime, opcional): Feha y hora de la carga del archivo.

Y sus métodos son:

\_\_init\_\_(title, file\_path, upload\_date=None): Es el constructor que inicializa una nueva partitura.

* + 1. **Almacenamiento y gestión de datos**

**Firebase Firestore**

Esta herramienta es usada para almacenar los datos tanto de los usuarios como de las partituras en una base de datos NoSQL. En Firebase la base de datos de Firestore se organiza por colecciones y documentos, de manera similar al concepto de tablas, filas y columnas, siendo las colecciones lo equivalente a tablas y los documentos a las filas.

* Colección de users: Contiene los documentos los cuales cada uno representa los datos de un usuario en concreto. Cada documento tiene un id que es el nombre del usuario.
* Colección de sheet\_music: Contiene los documentos los cuales cada uno representa una partitura, tanto para las registradas como para las digitalizadas.

**Firebase Storage**

Es la herramienta utilizada para el almacenamiento y gestión de los archivos que representan las partituras. Se ha seguido una estructura sistema de directorios que distinga los archivos por formato y usuario, manejando su acceso mediante rutas.

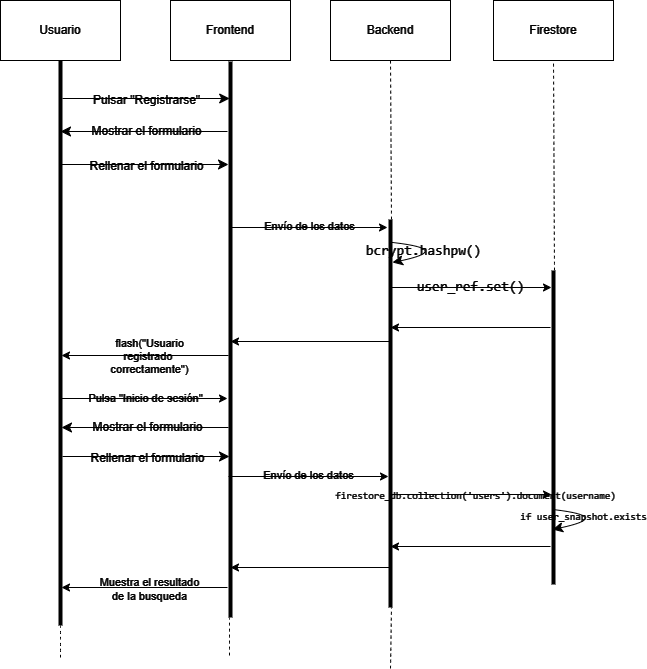
La estructura utilizada es: partituras/<user\_id>/<file\_type>/<filename> donde <user\_id> es el ID del usuario, <file\_type> es el formato del archivo de la partitura y <filename> es el nombre del archivo.

* 1. **Diseño procedimental**

En esta sección se va a mostrar con un diagrama de secuencias el proceso de un flujo de trabajo completo de la aplicación. Como el diagrama quedaría muy grande se va a seccionar.

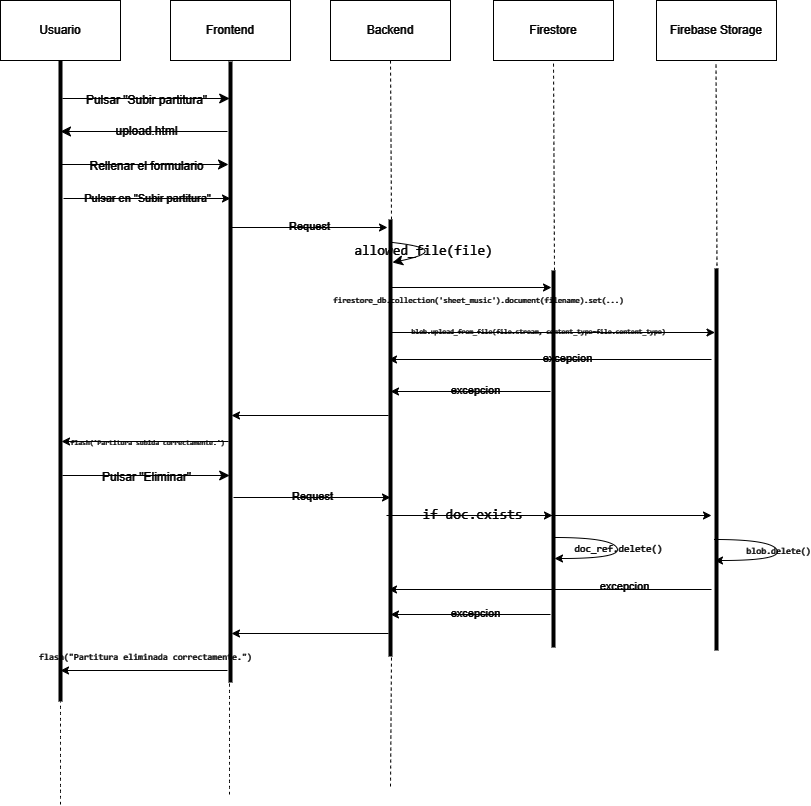
* + 1. **Registro e inicio de sesión**

Este diagrama muestra el proceso que sigue el programa para registrar una nueva cuenta de usuario. Para ello tras recibir la petición del formulario del usuario se encripta su contraseña en la capa del backend para luego enviar todos los datos al servicio Firestore. Seguido de este proceso, se realiza un inicio de sesión donde para autenticar un usuario, se verifican las credenciales. En ambos procesos se sigue un flujo de intercambio de datos entre los distintos servicios de Firebase.



* + 1. **Subida y eliminación de una partitura**

El siguiente diagrama es referente al flujo de subir una partitura a la aplicación y posteriormente eliminarla. Para ello el proceso se basa en esperar la petición del usuario de cada una de las acciones y hacer las comprobaciones correspondientes, siendo en el caso de subida verificar el formato de archivo y en el caso de la eliminación comprobar su existencia. Tras ello se intercambia información con los servicios de Firebase para actualizar los registros. En caso de excepción o éxito se le comunica al usuario.



* + 1. **Digitalización y preprocesamiento de una partitura**

Ahora se va a mostrar un flujo de digitalización de una partitura con el lanzamiento de una excepción seguido de preprocesamiento, acabando con un intento de digitalización con éxito.

El usuario hace la petición de ver el listado de partituras, desde el backend se accede al listado iterando sobre los elementos del bucket de Firebase Storage y para cada elemento se busca en la colección de Firestore el nombre correspondiente añadiéndolo a un array de elementos. Esto se manda a la vista para mostrarlo al usuario.

A continuación, el usuario hace una petición para digitalizar una partitura en concreto, se descarga de Firebase Storage para luego intentar digitalizarla en otra función. El resultado del proceso de digitalización es analizado buscando su log y cadenas de caracteres concretas para mandar alertas al respecto al usuario. El primer caso es el fallido por lo que se le envía al usuario un mensaje que indica que la partitura no es apta.

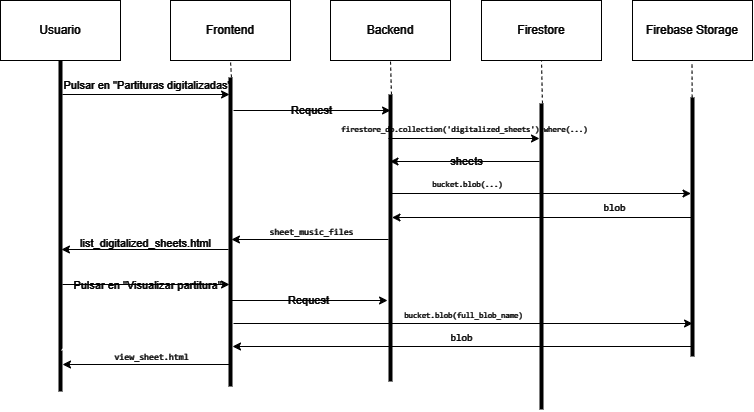
El siguiente intento es un éxito por lo que el resultado devuelto por audiveris es registrado en ambos servicios de Firebase y se envía un mensaje de éxito al usuario.

A black screen with white squares

Description automatically generated

* + 1. **Visualización de una partitura**

Por ultimo se va a representar el flujo que se sigue cuando un usuario solicita visualizar una partitura. El usuario solicita ver la lista de partituras digitalizadas, la request llega al backend y este accede al listado a través de la colección de Firestore, y de los registros de Firebase Storage. Tras esto se envían las partituras a la vista para que el usuario las visualice y elija cual debe mostrarse. Realiza la solicitud para visualizar la partitura y el backend accede a esta a través de Firebase Storage para visualizarla haciendo uso de Verovio.



* 1. **Diseño arquitectónico**

En esta sección se va a detallar el patrón de diseño que se ha seguido en el desarrollo del proyecto para mantener una escalabilidad.

* + 1. **Estructura Modular**

En primer lugar, se va a hablar de la estructura modular seguida en el proyecto. Se ha hecho uso de blueprints para modularizar las rutas y vistas, de manera que se gestionen las áreas de la aplicación de manera distintiva y separada. A su vez, también se han separado los archivos y módulos en función de su funcionalidad, pudiendo ser modelos, rutas, configuraciones, extensiones, servicios de Firebase, etc.

* + 1. **Configuración Externa**

Gran parte de los parámetros de configuración de la aplicación se han detallado y gestionado en un archivo externo, de manera que cualquier modificación y manejo en el comportamiento de la aplicación se facilite en gran medida. Además, se integran variables de entorno para las credenciales de Firebase y Google Cloud para mantener una buena seguridad en todos los entornos, sobre todo públicamente.

* + 1. **Gestión de Usuarios**

Toda la gestión de sesiones de usuario se ha hecho mediante una combinación de librerías de Flask con los servicios que proporciona Firebase para el almacenamiento y validación de los datos del usuario.

* + 1. **Servicios Externos**

Principalmente para el manejo de datos se ha utilizado servicios externos.

* Firebase Admin: Se ha utilizado los servicios de Firebase para la gestión y almacenamiento de datos, siendo Firestore la base de datos y Firebase Storage el servicio de almacenamiento de las partituras de la aplicación, de esta manera se centraliza de forma adecuada los datos para cada usuario.
* Manejo de archivos: la carga, subida y manipulación de los datos se gestiona de forma distribuida y validada según el formato de estos y los usuarios, además se intenta hacer un manejo seguro.
  + 1. **Tratamiento de errores**

Se ha hecho uso de excepciones como retroalimentación del flujo de la aplicación, así como de interacción informativa para el usuario con el fin de mantenerlo informado con cada acción en todo momento.

* 1. **Diseño de interfaces**

Para el diseño de interfaces de la aplicación se ha hecho uso de plantillas HTML que implementan la librería Bootstrap como framework de frontend. Todas las páginas utilizan Bootstrap para hacer que visualmente se mantenga una consistencia, ajustándose a varios dispositivos y tamaños de pantalla, esto ha ayudado a mantener en todas las vistas una estética uniforme.

Además, se ha ajustado la estructura de las vistas con el uso de secciones de estilo css para mantener una homogeneidad entre las ventanas.

En las vistas se hace uso de varios componentes propios de Bootstrap, ya sean botones, alertas y formularios, para mejorar la usabilidad de la aplicación para el usuario.

**Anexo D Documentación técnica de programación**

* 1. **Introducción**

En este apartado se va a explicar y mostrar la organización y estructuración que tiene el proyecto, así como un manual del programador para dar pautas e instrucciones a los usuarios de cómo preparar su entorno e instalar la aplicación correctamente.

* 1. **Estructura de los directorios**
* db\_versions: Es una carpeta en la que se ha guardado alguna versión de la base de datos para evitar pérdidas debido a migraciones.
* DigitalizaciónPartiturasApp: Es el directorio en el que se ha trabajado en la aplicación. Aquí se encuentran las carpetas y directorios que contienen el código de la aplicación, así como recursos necesarios para el funcionamiento de ciertas funcionalidades.
  + .vscode: Es la carpeta que contiene configuraciones y ajustes del proyecto utilizado por VS Code.
  + \_\_pycache\_\_: Es una carpeta generada automáticamente por Python con versiones optimizadas del código.
  + app: Es la carpeta principal que contiene los archivos principales de la aplicación.
    - templates: Contiene todos los archivos html necesarios para que se muestren las vistas
    - \_\_init\_\_: Utilizado para inicializar y configurar la aplicación web, ya sea de extensiones o servicios.
    - extensions.py: Este archivo se encarga de cargar las credenciales de Firebase y de inicializar esta herramienta en la aplicación.
    - firebase\_utils.py: Se encarga de toda la interacción que tiene la aplicación con Firebase y con la base de datos NoSQL.
    - models.py: Contiene la definición de los modelos de la aplicación.
    - routes.py: Contiene todas las funcionalidades y lógica de la aplicación.
  + audiveris: Directorio clonado del software de Audiveris.
  + audiveris\_output: Directorio de salida para las partituras digitalizadas.
  + logs: Carpeta que contiene los archivos del logging utilizados a lo largo del desarrollo del proyecto.
  + migrations: Directorio que contiene las migraciones realizadas durante el desarrollo del proyecto.
  + verovio: Directorio clonado de Verovio utilizado para visualizar las partituras.
  + config: Archivo en el que se realiza toda la configuración necesaria para el correcto funcionamiento de la lógica de la aplicación. Se definen variables para guardar la configuración de los servicios, como Firebase, claves y archivos .json utilizados posteriormente en otros módulos, así como la configuración de los logs para controlar el flujo de la aplicación.
  + DockerFile: Archivo creado para la creación del Docker donde se especifica las base, herramientas y dependencias necesarias para el correcto funcionamiento de la aplicación, así como la compilación de Audiveris y la ejecución final de la aplicación.
  + requirements: Contiene todos los requerimientos necesarios para el correcto funcionamiento de la aplicación. Este archivo es creado automáticamente mediante el comando: pip freeze > requirements.txt.
  + run.py: Este archivo se encarga de la ejecución del servidor de Flask con el host 0.0.0.0 para que sea accesible para cualquier IP.
  + run\_audiveris.sh: Este archivo construye el comando necesario para ejecutar Audiveris sin interfaz de usuario, exportando los resultados a una carpeta definida.
  + sheet-transcribe-firebase-adminsdk: Son las credenciales json del servicio Firebase.
* memoria: Este directorio alberga los documentos referentes a la documentación.
* generate\_base64: Este script se encarga de codificar las credenciales de Firebase.
  1. **Manual del programador**

En este apartado se van a detallar los programas y configuraciones utilizados para el desarrollo del proyecto. Por lo tanto, los requisitos y pasos de instalación son:

* Python 3.11ç
* Git
* Docker
* Visual Code

A continuación, se va a explicar cómo instalar cada programa paso a paso en el sistema operativo Windows, que es en el que se ha llevado a cabo el desarrollo del proyecto.

**Python 3.11**

Python es el lenguaje de programación en el que ha sido desarrollada la aplicación.

Es necesario usar la versión de Python 3.11. Para su descarga es necesario acceder a <https://www.python.org/downloads/>. Tras esto, durante la instalación hay que marcar la opción “Add Python to PATH”.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Git**

Es una herramienta de control de versiones usado para la gestión del desarrollo de la aplicación.

Es necesario descargar e instalar Git, por ello hay que acceder a <https://git-scm.com/download/win>.

A screenshot of a computer

Description automatically generated

**Docker Desktop**

Para la importación y ejecución del Docker es necesario tener instalado Docker Desktop que es el programa de gestión de contenedores e imágenes Docker. Para ello hay que acceder a <https://www.docker.com/>

A screenshot of a computer

Description automatically generated

**Visual Studio Code**

El editor de código utilizado para construir y trabajar en el desarrollo ha sido Visual Studio Code. Para ello es necesario acceder a <https://code.visualstudio.com/> para descargarlo e instalarlo. Al igual que antes en esta instalación es necesario activar la casilla de “Agregar al PATH”. Algunas extensiones de Visual Studio que han resultado útiles son Python, Docker, GitLens y Prettier.

A screenshot of a computer

Description automatically generated

* 1. **Compilación, instalación y ejecución del proyecto**

En primer lugar, sería necesario acceder al repositorio del proyecto para clonarlo <https://github.com/martingonzsaiz/Digitalizacion-Partituras>. Para ello hay que realizar el comando siguiente en la terminal CMD o en el PowerShell:

**git clone** [**https://github.com/martingonzsaiz/Digitalizacion-Partituras.git**](https://github.com/martingonzsaiz/Digitalizacion-Partituras.git)

Luego sería necesario moverse al directorio en cuestión para seguir con las siguientes instrucciones.

**cd Digitalizacion-Partituras**

Después se creó un entorno virtual en el que se trabajó y se llevó el desarrollo. La creación del entorno virtual se alcanza con los siguientes comandos:

**python -m venv entorno\_partituras**

**entorno\_partitiuras\Scripts\activate**

De esta manera se facilita el manejo del entorno y se asegura el aislamiento de las dependencias y su mayor gestión. Así las dependencias de este proyecto no entran en conflicto con dependencias externas. Si el entorno está activado aparece su nombre al principio de la línea de comandos.

Tras esto, es necesario instalar las dependencias del proyecto, para llevarlo a cabo se utiliza el comnado:

**pip install -r requirements.txt**

A continuación, es necesario tener claros ciertos aspectos de la aplicación. En esta aplicación se utilizan valores de variables para preparar el entorno de manera segura y efectiva. Estos valores son FIREBASE\_CREDENTIALS\_JSON\_BASE64 y FIREBASE\_BUCKET\_NAME.

Las credenciales de Firebase sirven para que la aplicación se autentique y pueda acceder a los servicios de Firebase. En el caso de esta aplicación se hace uso de Firestore y Firebase Storage, por ello es necesario generar estas claves y definirlas como variables. Para ello habría que seguir unos pasos extra:

En Firebase Storage:

1. Ir a Firebase Console: <https://console.firebase.google.com/>
2. Crear o seleccionar el proyecto.
3. Ir a Configuración del proyecto > Cuentas de servicio.
4. Seleccionar “Generar nueva clave privada” y guardar el archivo JSON generado.
5. Se podría codificar las credenciales con el siguiente comando: **certutil -encode input.json output.txt**

A screenshot of a computer

Description automatically generated

En Firebase Bucket:

1. Acceder a Firebase Console y entrar en el proyecto.
2. Entrar en Storage.
3. Copiar el nombre del bucket.

Ahora es necesario configurar las variables de entorno para poder cargar y preparar las credenciales desde el entorno del sistema haciendo uso de la biblioteca os, tal y como se hace en el archivo config.py.

1. Ir a Sistema>Configuración avanzada del sistema>Variables de Entorno
2. En Variables del sistema, es necesario crear las nuevas variables de entorno: FIREBASE\_CREDENTIALS\_JSON\_BASE64, FIREBASE\_BUCKET\_NAME.
3. Aplicar los cambios

Con esta configuración se podría reutilizar el archivo config.py del repositorio.

Ahora, tras asegurarse de que el entorno virtual esta ejecutado, se podría ejecutar la aplicación con el comando:

**python run.py**

Y acceder a ella a través del navegador con <http://127.0.0.1:5000>, pudiendo interactuar con todas las funcionalidades de la aplicación localmente.

* 1. **Pruebas del sistema**

Durante todo el desarrollo del proyecto se han realizado pruebas de integración para comprobar el correcto funcionamiento de los servicios utilizados de Firebase. Estas pruebas han consistido en comprobaciones e impresiones en el flujo de la aplicación para comprobar que las interacciones con Firestore funcionan correctamente y que el intercambio de datos con Firebase Storage se realiza como se desea.

Muchas de estas pruebas han sido un proceso de hacer varias pruebas de registro e inicio de sesión para comprobar que todo funcionaba correctamente con Firebase en diferentes contextos. Otro tipo de pruebas que se han realizado ha sido con el manejo de cargas y descargas de partituras con Firebase Storage.

A su vez también se ha hecho varias pruebas para comprobar el comportamiento de las funciones ajenas a los servicios utilizados en la aplicación. Dentro de este grupo, las funcionalidades en las que mas se ha probado su funcionamiento es con las funciones de preprocesamiento de imágenes. El proceso ha consistido en manejar distintos valores para los campos del formulario de preprocesamiento de imágenes para comprobar visualmente los resultados obtenidos en la imagen. De esta manera se ha podido valorar cuales son las mejores combinaciones o valores. Otro aspecto en el que se ha hecho pruebas es la comprobación de los formatos en los que se admiten partituras, teniendo que ser .pdf, .png, .jpg, .jpeg.

Otro tipo pruebas que se ha realizado es referente al proceso de digitalización de las partituras. Debido al ratio de fallo que tiene Audiveris con lo que respecta a digitalizar partituras en ciertas circunstancias o con ciertas características, ha sido necesario afianzar y agrupar cierto grupo de ejemplares de partituras que valen como testeo y llegan al fin. Para ello ha habido que realizar un proceso de prueba y error hasta dar con una colección.

En general se ha hecho pruebas para verificar que todas las rutas de la aplicación devuelven las respuestas y esperadas en diferentes contextos y condiciones. Además, cada vez que se ha implementado una funcionalidad nueva se ha comprobado que no alteraba el correcto funcionamiento del resto, haciendo un proceso de flujo completo de trabajo a lo largo de la aplicación.

**Anexo E Documentación de usuario**

* 1. **Introducción**

En este apartado se va a explicar como debe ser la descarga e instalación de la aplicación para cualquier usuario casual interesado en el uso de la aplicación. Además, se explicarán los requisitos necesarios para que el usuario pueda hacer uso de ella. Tras ello se explicará cómo es la interacción de la aplicación con el usuario visualmente.

* 1. **Requisitos de usuarios**

<https://melodymatrix-abf474f50a15.herokuapp.com/menu>

Sin embargo, como la aplicación desplegada de Heroku no es capaz de digitalizar las partituras, debido a que la aplicación tiene mucho peso por Audiveris, se plantea otra alternativa para que los usuarios puedan probar la aplicación.

**Apéndice F Anexo de sostenibilización curricular**

En este apartado se van a listar los objetivos de desarrollo sostenible que alcanza este proyecto.

**Objetivo 4: Educación de calidad**

El proyecto contribuye al Objetivo de Desarrollo Sostenible 4, ‘Educación de calidad’, al aportar en gran medida un recurso funcional para la educación musical. Siendo su acceso libre, permite la edición y manipulación de las partituras en formato digital, abriendo múltiples posibilidades a profesores, alumnos y personas autodidactas para gestionar diferente material del sector.

**Objetivo 9: Industria, innovación e infraestructura**

El proyecto contribuye al Objetivo de Desarrollo Sostenible 9, ‘Industria, innovación e infraestructura’, al implementar una tecnología para la digitalización y edición de partituras, mejorando la infraestructura tecnológica y fomentando la innovación en el sector.

**Objetivo 12: Producción y consumo responsables**

El proyecto contribuye al Objetivo de Desarrollo Sostenible 12, ‘Producción y consumo responsables’, mediante el uso de la herramienta los usuarios reducirán el consumo de papel, al optar por un formato digital. Esta práctica fomenta prácticas más sostenibles que alienta por un recurso más eco-friendly y eficiente.

**Bibliografía**

1. Talent.com. «Salario para Programador Junior en España - Salario Medio». Accedido 29 de junio de 2024. <https://es.talent.com/salary>.
2. Agency, Monster Digital. «Sueldos de un desarrollador junior: Salario de un developer junior». Epitech Spain (blog), 10 de diciembre de 2021. https://www.epitech-it.es/sueldo-desarrollador-junior/.