Aplicación para la administración de archivos de bandas de música

Memoria de Proyecto Final de Grado

Grado de Ingeniería Informática

Autor: Martin José Moreno Jiménez
Consultor: Gregorio Robles Martínez
Profesor: Santi Caballe Llobet

Enero del 2022

Agradecimientos,
A Dorma Diseño por formarme y darme la primera oportunidad. A la familia Medina Cuadros por confiar en mí. A la otra familia de los Medina, mis compañeros, ya que ellos son el espejo al que quiero parecerme. A mi familia y, en especial, mi madre por enseñarme tres palabras: perseverancia, decisión y valentía. Y a la Estrella que tengo en mi vida, por ser la luz de mi camino. Si estoy aquí es por vosotros.

FICHA DEL TRABAJO FINAL

Título del trabajo:	Mibanda.online: una aplicación web para la gestión del archivo de bandas de música.
Nombre del autor:	Martín José Moreno Jiménez
Nombre del consultor:	Gregorio Robles Martínez
Fecha de entrega:	01/2023
Área del Trabajo Final:	Desarrollo de aplicaciones web
Titulación:	*Grado de ingeniería informática*

Resumen del Trabajo (máximo 250 palabras):

El TFG tiene como finalidad la creación de una aplicación que permita gestionar el archivo de una banda de música. La aplicación busca dos objetivos: por un lado, que los administradores de la banda puedan administrar el archivo de partituras de esta y los componentes que la forman. Y, por otro, que los miembros de la banda puedan acceder a sus partituras de forma fácil.

Para lograr este objetivo se ha desarrollado e implementado una aplicación web utilizando C# para la gestión de backend, Angular 14 para la gestión del frontend y SQL para la gestión de base de datos.

Además, el proyecto está dockerizado para poder desplegarlo en cualquier entorno de forma fácil y, sobre todo, escalable.

Abstract (in English, 250 words or less):

The purpose of the TFG is the creation of an application that allows managing the archive of a music band. The application seeks two objectives: on the one hand, that the band's administrators can manage its sheet music file and the components that make it up. And, on the other, that the members of the band can access their scores easily.

To achieve this goal, a web application has been developed and implemented using C# for backend management, Angular 14 for frontend management and SQL for database management.

In addition, the project is dockerized to be able to deploy it in any environment easily and,

above all, scalable.

Palabras clave (entre 4 y 8):

C#, Angular, HTML5, SQL, Desarrollo Web, Archivo de bandas de música

Contenido

1. Tabla de ilustraciones

2. Objetivo del proyecto

El objetivo de este proyecto es el desarrollo de una *webapp* pensada tanto para directores/archiveros como para músicos que gestione el archivo de obras que posee una banda de música o asociación musical. La aplicación propone una alternativa para resolver el problema que nos encontramos muchas veces los músicos cuando vamos a ensayar o tocar, ya que muchas veces faltan partituras bien porque no las posee o bien porque las ha perdido.

De igual forma, se está extendiendo el uso generalizado de tabletas electrónicas a la hora de realizar actuaciones ya que la compra de estos dispositivos a la larga es más económica que la impresión física en papel.

La *app* constará de un panel de autentificación, se debería determinar si será propia, mediante terceros (Google y Facebook) o mixta. Hay que tener en cuenta que un mismo perfil puede estar en varias agrupaciones y puede tener distintos roles en ellas.

Cuando un usuario entre a la *app*, aparecerán listadas las agrupaciones a las que pertenece y en cada una de ellas según el perfil que disponga tendría la posibilidad de:

- Si es director: Gestionar el archivo: dar de alta obras, añadir partes a las mismas, creación, lectura, actualización y borrado de componentes, asignar instrumentos a compones, etc.
- Si es músico: le aparecería las bandas a las que pertenece y podría buscar las obras que tiene asociadas. Una vez dentro de la obra podría abrir o descargarse la partitura vinculada a cada instrumento vinculado.

De igual forma existiría un proceso de "archivado" automático, ya que las obras la mayoría de las veces vienen en un solo archivo, este archivado se basaría en Hash de cada archivo para reducir espacio de disco en el servidor (por ejemplo, se una asociación quiere subir una obra ya existente no duplicaríamos los ficheros).

En definitiva, el proyecto básicamente busca resolver un problema muy común que tenemos músicos de banda de música: disponer de forma inmediata de las partituras que nos faltan partituras cuando se va a tocar de forma inmediata.

3. Plan de trabajo

3.1. Análisis

En esta fase se han realizado un análisis en profundidad de cómo será el proyecto determinando: su alcance, arquitectura, base de datos, etc.

- **Fase 1**
  Determinar alcance de la aplicación, estudiar casos de uso y escribir las distintas Historias de usuario que estarán dentro del proyecto.

- **Fase 2.**
  Determinar la arquitectura de la aplicación, *frameworks*[1] a utilizar y estructura de la base de datos.

  En un principio, el proyecto se desarrollará en C# en backend, con Angular en *frontend*, sobre SQL server (enlazado con *Entity Framework* de ORM[2]) y, probablemente, estaría dockerizado.

  El proyecto utilizará *Code First*[3] basado en migraciones, es decir, el modelado de BBDD se realiza por iteraciones desde el mismo código. El control de código fuente es gestionado por Git y almacenado en un repositorio de Github.

3.2. Implementación:

Durante esta fase se ha desarrollado, testado y puesto en producción la solución propuesta.

- **Fase 3.**
  Prototipado de la APP.

  En esta fase determinaríamos el Core o núcleo de la APP:

  - Gestión de excepciones y manejadores de logs.
  - Gestión de Usuarios, roles, autentificación y permisos (incluyendo login).
  - Integración continúa de la aplicación.
  - Gestión de los datos maestros de la *app* como, por ejemplo, instrumentaciones que dispone una banda de música. Plantillas de tipos de banda (banda de música, charanga, etc).

- **Fase 4**.
  Servicios vinculados a los gestores de archivo:

  - CRUD[4] de usuarios dentro de una organización (asignación de un usuario a una banda).
  - CRUD de obras y asignación de partes.

- **Fase 5**.
  Servicios vinculados a un usuario:
  - Ver obras asignadas

3.3. Finalización del proyecto.

Esta última fase se ha documentado la solución propuesta con el objetivo de ser presentada ante el tribunal.

- **Fase 6.**
  Documentación del proyecto, testing y calidad del software (gran parte se realizará de forma paralela al desarrollo).

- **Fase 7.**
  Memoria y presentación. Ciertas partes de la memoria, entiendo que también se realizarán de forma paralela al desarrollo.

- **Fase 8.**
  Defensa del proyecto

Se han intentado alinear las fases con las distintas PECs que tenemos previstas, por tanto, un calendario aproximado de desarrollo sería el siguiente:

- Fase 1: 04-11-2022
- Fase 2: 11-11-2022
- Fase 3: 25-11-2022
- Fase 4: 09-12-2022
- Fase 5: 23-12-2022
- Fase 6: 30-12-2022
- Fase 7: Primera memoria 07-01-2023. Segunda memoria 11, tercera y última el 19.
- Fase 8: Entre el 27-01 y 01-02-22

Así mismo debemos tener en cuenta que puede existir algún tipo de retraso. Por eso, aunque sean fechas festivas entiendo que se podrían asumir estos retrasos en tiempos de la fase 5 y 6. Además, se intentaría cerrar antes la fase 1 y 2.

4. Tecnologías usadas

Tal como se ha descrito en el listado de fases se proponen las siguientes tecnologías:

4.1. Backend

El proyecto se desarrollará en C# bajo .Net *framework*. Se utilizará librerías de apoyo de este *framework* como pueden ser: *Entity Framework*, Inyección de dependencias, etc.

.Net Framework[5] es un entorno de ejecución desarrollado por Microsoft que dota al usuario de distintas herramientas y lenguajes de programación con el fin de poder desarrollar aplicaciones de informáticas, servicios, librerías, etc. El proyecto utilizará su versión LTS más reciente .Net Framework 6. Así mismo el proyecto de backend está codificado en C#.

Entity Framework[6], es una solución ORM (Object Relational Mapping o Mapeo Objeto-Relacional en castellano) de código abierto de Microsoft que nos permite gestionar desde el código fuente la estructura de la base de datos y realizar operaciones de creación, lectura, actuación y borrado de los datos que esta contiene de forma fácil mediante el lenguaje de consultas integrado LINQ.

4.2. Frontend

Se desarrollará en la versión más actual de Angular[7] bajo *Typescript*[8]. Además de html5 y *JavaScript*. De igual forma se utilizarán distintas librerías de apoyo para el desarrollo y maquetación basadas en *Bootstrap*.

Angular es un framework de código abierto desarrollado principalmente por Google. Está orientado a la creación de aplicaciones web de una sola página en Typescript. El proyecto de Frontend implementa una aplicación basada en Angular 14, que corresponde a la ver más reciente LTS. Igualmente se utilizan distintas librerías y componentes satélites open source de apoyo.

Bootstrap[9] es un framework, principalmente, de maquetación web que permite desarrollar aplicaciones web de modo sencillo y rápido. Está desarrollado principalmente por Twitter. Para el proyecto se ha utilizado la versión 5.

4.3. Base de datos

Utilizaremos SQL Server[10] bajo una imagen Docker en su última versión. SQL server es un sistema de gestión de bases de datos relacionadas desarrollado por Microsoft. Aunque dispone de versiones de pago, para el proyecto se ha utilizado la versión Express. Esta versión dota al proyecto de todas las funcionalidades de SQL Server con bases de datos de hasta 5gb de forma gratuita[11].

4.4. Otros

Se ha dockerizado el proyecto y se usa Github para el control de código fuente.

Docker[12] es un proyecto multiplataforma de código abierto que automatiza el despliegue de aplicaciones basadas en contenedores. Estos contenedores son máquinas virtuales construidas de forma modular independientes del sistema operativo que las ejecuta.

Github[13] es un servicio de almacenamiento y control de versiones de archivos propiedad de Microsoft. Está enfocado principalmente a alojar código fuente de aplicaciones informáticas. Este control del versionado de archivos permite a los desarrolladores trabajar en un mismo

proyecto de forma simultánea y establecer mecanismos de supervisión, revisión y testeo de cambios de forma manual como automática.

Además, está integrado con distintos entornos de desarrollo integrado (IDE)[14] facilitando la importación y exportación de los cambios mediante el uso de ramas de trabajo. Para el proyecto se ha utilizado un repositorio privado gratuito.

4.5. IDE y software adicional

Se utilizará Visual Studio Community[15] en su versión 2022 para el desarrollo de la parte de *backend* y Studio Code[16] para el *frontend.* Para la gestión de la base de datos y consultas se utilizar Sql Management Studio. Tanto Visual Studio Community como Visual Studio Code son entornos de desarrollo propiedad de Microsoft con licencia gratuita de uso[17], [18].

Las pruebas de funcionamiento serán bajo Chrome de escritorio principalmente, aunque se ha comprobado su correcto funcionamiento en otros navegadores y formatos (Tablet y móvil), realizando un desarrollo responsivo de la aplicación

5. Historias de Usuario

5.1. Actores del proyecto

La aplicación contará con tres actores principales:

- **Administrador**: Persona autentificada en el aplicativo que se encargará del mantenimiento de la aplicación, es decir, el mantenimiento de los parámetros maestros y el alta de una nueva agrupación en el sistema, gestionar usuarios e integrantes y cambiar permisos de las agrupaciones. Corresponde al grupo de usuarios marcados con el rol Admin.
- **Usuario Autentificado**: Persona autentificada en el aplicativo con acceso a, al menos, una agrupación musical. Por defecto podrá ver sus partituras y calendario de las agrupaciones que sea miembro. Además, si dispone de permisos de gestión de archivo en dicha agrupación podrá gestionar a los miembros de la banda, el archivo y el calendario de esta. Un usuario puede ser administrador de una banda y miembro de otra de forma concurrente.
- **Usuario**: Persona no autenticada (no registrada) o autentificada dentro del aplicativo que no pertenece a ninguna agrupación ni es administrador.

5.2. Tips generales en todas las historias de usuario: *Definition of Done* y *Definition of Ready*[19] del aplicativo

Se han definido los siguientes requisitos que deben cumplir **todas las historias de usuario de forma común**:

5.2.1. Backend

- Se debe prestar especial atención a la hora de generar cualquier clase. Esta debe ir ubicada en la capa que por alcance le pertenece.
- Todos los elementos que lo necesiten dispondrán de control de excepciones y registros en el log de la aplicación
- Uso de expresiones *LINQ*
- El código tendrá un patrón *async/await*[20]
- El *endpoint* principal de backend será una API accesible mediante cliente (*Postman*) o *Swagger*.
- Se debe comprobar que los *endpoints* son accesibles consumibles por *Swagger*[21] para su consumo sin el *front* y estos están documentados.
- Los datos recibidos por cualquier *endpoint* deben ser validados en función del requerimiento de la tarea.
- De igual forma, el *endpoint* enviará una respuesta que cumpla con los códigos de repuesta estándares de C#[22] y sea gestionable por el *front*
- Se establece *JSON* como formato para el formato de comunicación con la *API*.

5.2.2. Frontend

- Se prestará especial atención a la ubicación de los componentes y módulos. Estos deben estar en la ruta que les pertenezca
- Se crearán rutas específicas y módulos según el alcance de la parte. Se prestará especial atención en la importación y exportación de elementos.

- Se seguirá un patronaje *async/await* basado en *callbacks* y promesas.[23]
- Cualquier formulario debe validarse previamente al envío. Se implementarán mecanismos que garanticen que no se remiten formularios erróneos.
- Se implementarán mecanismos que revisen si el usuario tiene permisos para acceder a las rutas y servicios.
- En caso de que la respuesta non sea correcta, se debe mostrar el error notificado por el back
- Propiedades del *layout*:
    - Debe ser responsivo y adaptable a Ordenadores de escritorio y tabletas. Se tendrá en cuenta que la resolución nativa de escritorio es FullHD y la tableta un Ipad Pro de 11 pulgadas.
- Propiedades de los listados:
    - Salvo que se indique lo contrario, tendrán paginación y deben permitir buscar por sus campos claves.
    - El usuario podrá determinar la cantidad de ítems.

5.2.3. Comunes para ambos entornos

- Se debe seguir la arquitectura y patronaje marcada, prestando atención a la creación de un código limpio y documentado
- Se crearán test unitarios en la medida que sea posible

5.2.4. Esquema de comunicación base entre entornos

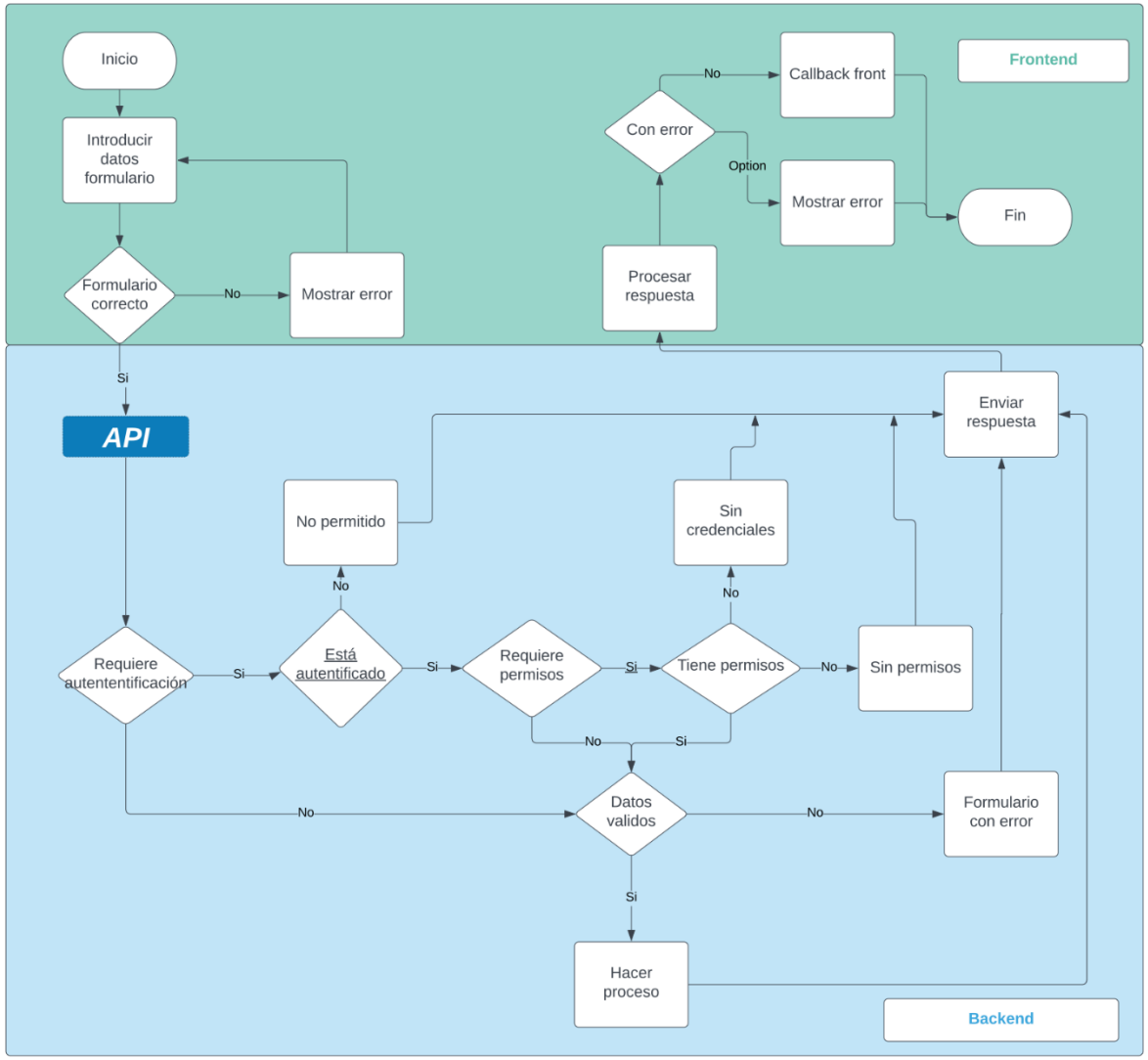


Ilustración 1 - Diagrama de comunicación de la aplicación

La ilustración 1 muestra el esquema de comunicación entre las diferentes partes de la aplicación. Todas las historias de usuario seguirán este patrón, el proceso de backend y sus casuísticas están descritas en cada historia de usuario.

La aplicación desarrollada contará con las siguientes historias de usuario:

5.3. Relacionadas con la autentificación

5.3.1. Un usuario podrá iniciar sesión en la aplicación

Esta historia de usuario busca desarrollar la lógica que de autentificación en la aplicación.

Backend

El objetivo, a nivel de *backend*, es la creación de un servicio de autentificación contra la BBDD, este servicio comprobará que:

- La encriptación de la contraseña para su posterior comparación con el existente en la base de datos.
- Si hay algún fallo en el formulario, contraseña o proceso se notificará al usuario
- En caso de que la autentificación sea correcta se generará un token *JWT*[24], que será almacenado para comprobado por los servicios que requieran autentificación.
- Si se cumple la premisa anterior se enviará una respuesta correcta al *front*, esta correcta debe incluir los campos que este necesite además del token *JWT*.
- Se debe desarrollar un servicio que devuelva a la *APP* el menú vinculado al usuario en base a sus permisos. Este menú tendrá en cuanta lo siguiente:
    - Si posee un rol de administrador debe disponer de acceso a la ruta de administración del sistema
    - Se genera un item por cada agrupación a la que pertenezca teniendo en cuenta que:
        - Todos los usuarios pueden ver su calendario y partituras
        - Solo los usuarios gestores de dicha agrupación pueden gestionar los integrantes y archivo de esta.

Frontend

A nivel de frontend esta historia busca desarrollar una página de *login* que cumpla las siguientes características:

- Se debe validar el formulario antes de su envío
- En caso de que la respuesta sea correcta, se deben generar y almacenar las clases y servicios que sean necesarias para el uso de los mecanismos de control de permisos y peticiones funcionen correctamente.
- El *front* navegará al área autentificada y:
    - Cargará el menú vinculado al usuario logado.
    - Mostrará los datos del usuario logado en las distintas partes destinadas del *front*.

5.3.2. Un usuario podrá registrarse en la aplicación

El objetivo de esta historia de usuario es desarrollar la lógica permita a cualquier usuario registrarse en la base de datos.

Backend

Se creará un servicio que registre al usuario en la base de datos, este usuario podrá autenticarse en el aplicativo una vez registrado, pero no tendrá acceso a ningún módulo hasta que una agrupación o administrador lo marque como miembro de esta. Esta tarea será abordada en otra historia de usuario.

Se deberá tener en cuenta lo siguiente en el desarrollo:

- La contraseña debe encriptarse con el mismo sistema que utiliza el *login*.
- Solo se permite un usuario con el mismo correo electrónico.

Frontend

De igual manera, a nivel de *frontend*, se debe desarrollar un controlador que sea accesible desde una ruta específica que permita a un usuario registrarse en el aplicativo.

5.4. Relacionadas con la administración de la aplicación

5.4.1. Un usuario con permisos administrativos podrá administrar agrupaciones.

El objetivo de esta historia de usuario es gestionar que cualquier usuario con el rol Amin pueda:

- Dar de alta una nueva agrupación
- Actualizar los datos de una agrupación
- Marcar una agrupación como dada de baja en el sistema.

Backend

Se generará un servicio que realice las funcione anteriores.

Frontend

Se genera una opción accesible desde el menú que:

- Liste las agrupaciones existentes de forma paginada.
- Permita realiza búsquedas sobre el listado

5.4.2. Un usuario con permisos administrativos podrá gestionar los instrumentos de la aplicación

El objetivo de esta historia de usuario es la gestión global de los instrumentos que serán accesibles por los archiveros, estos servicios son:

- Dar de alta un instrumento
- Modificar un instrumento
- Eliminar un instrumento

Backend

Generar un servicio que realice las operaciones anteriores, teniendo en cuenta que:

- No se podrá eliminar un instrumento si pertenece a alguna obra

Frontend

Se genera una opción accesible desde el menú que:

- Liste los instrumentos disponibles de forma paginada.
- Permita realiza búsquedas sobre el listado
- Permita generar nuevos instrumentos
- Permita realizar modificaciones o eliminar los existentes

5.4.3. Un usuario con permisos administrativos podrá gestionar los usuarios

El objetivo de esta historia es permitir que un usuario con los permisos necesarios pueda gestionar los usuarios existentes en la base de datos. Tras su realización el usuario podrá:

- Marcar un usuario como inactivo de forma global, es decir, hacer un borrado lógico en la BBDD.
- Asignar o desasignar un usuario a una agrupación
- Modificar datos del usuario

Backend

Se realizará un servicio que implemente lo anterior.

Frontend

Se genera una opción accesible desde el menú que:

- Liste los usuarios paginada.
- Permita realiza búsquedas sobre el listado.
- Permita realizar las operaciones administrativas nombradas en la historia con un usuario.

5.5. Relacionadas con los usuarios autentificados

5.5.1. Un usuario con permisos de gestión podrá gestionar los usuarios de una agrupación

Un gestor podrá vincular a otros usuarios como miembros de su agrupación. Además, podrá marcar a otros usuarios como administradores. Se tendrá en cuenta que:

- Sólo podrá dar de alta al usuario si conoce el correo electrónico del mismo.
- Podrá gestionar los miembros existentes pudiendo:
    - Asignar o desasignar instrumentos a los integrantes.
    - Eliminar al usuario de la agrupación.
    - Marcar o desmarcar a un usuario como administrador, esta función no podrá hacerla contra sí mismo.

Backend

Realizar un servicio que implemente lo anterior.

Frontend

Se genera una opción accesible desde el menú que:

- Liste los usuarios existentes en la agrupación de forma paginada.
- Permita realiza búsquedas sobre el listado.
- Permita realizar las operaciones vinculadas a la historia de usuario.
- Permita dar de alta a un usuario mediante su email.

5.5.2. Un usuario con permisos de gestión podrá gestionar el archivo de una agrupación

Tras realizar esta HU un usuario con privilegios de administración de una agrupación podrá:

- Dar de alta una nueva obra.
- Administrar las partituras vinculadas a la obra.
- Consultar partituras de una obra especifica.

Backend

Se realizará un servicio que permita realizar las operaciones anteriores teniendo en cuenta que:

- En la medida que sea posible se ofrecerá al usuario obras ya existentes mediante el autocompletado del nombre de la obra.
- Si una obra existe en el sistema se cargará de forma automática los instrumentos que pertenecen a la misma.
- Se permitirá asignar nuevos a la obra y vincular su partitura. No se guardará el instrumento si no tiene partitura asociada.
- Se permitirá modificar el archivo vinculado al instrumento siempre que el Hash de este sea distinto al que está almacenado.

Frontend

Se genera una opción accesible desde el menú que:

- Liste las obras existentes en la agrupación de forma paginada.
- Permita realiza búsquedas sobre el listado
- Permita ver las partituras e instrumentos vinculadas a la obra
- Permita realizar las operaciones vinculadas a la historia de usuario

5.5.3. Un usuario con permisos de gestión podrá gestionar el calendario de la agrupación

Tras la realización de esta HU un usuario con permisos de gestión de la agrupación podrá:

- Generar o modificar eventos vinculados a la agrupación.

  Cada evento dispondrá de una lista de obras a interpretar

Backend

Realizar un servicio que implemente lo anterior

Frontend

Realizar una vista que permita ver los eventos futuros de la banda y administrarlos.

5.5.4. Un usuario autentificado podrá ver o descargar sus partituras

Tras la realización de esta HU cualquier miembro de la banda podrá ver el listado de obras de la agrupación y visualizar o descargar las partituras vinculadas a estas. Se tendrá en cuenta que:

- Solo se permitirá ver las partituras de instrumentos que estén asignados al usuario en la agrupación.
- No se listarán obras que el usuario no tenga partituras asignadas.

Backend

Realizar un servicio que implemente lo anterior.

Frontend

Se genera una opción accesible desde el menú que:

- Liste las obras que el usuario puede acceder de la agrupación de forma paginada.
- Permita realiza búsquedas sobre el listado.
- Permita ver o descargar la partitura vinculada a la obra.

5.5.5 Un usuario autentificado ver los próximos eventos de una agrupación

Un usuario podrá ver los eventos próximos vinculados a una agrupación. De igual forma podrá ver o descargar las partituras vinculadas a dicho evento en los instrumentos que tenga asignados.

Backend

Modificar el servicio de calendario para implementar la consulta de este en usuario no gestores.

Frontend

Modificar la vista de calendario para que sea compatible con usuarios no gestores. Añadir la funcionalidad de vista de las partituras vinculadas a un usuario.

6. Modelo de pantallas

6.1. Login

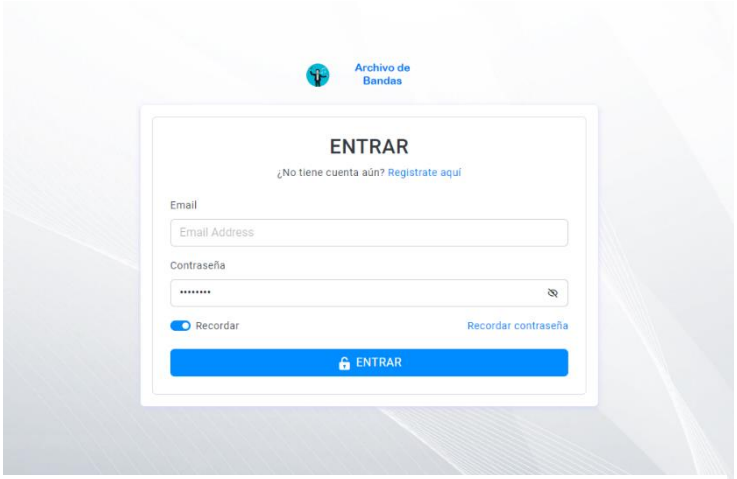
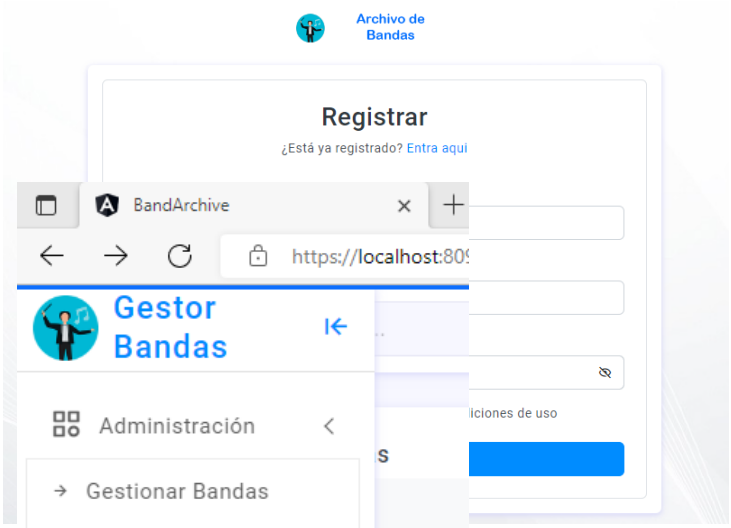
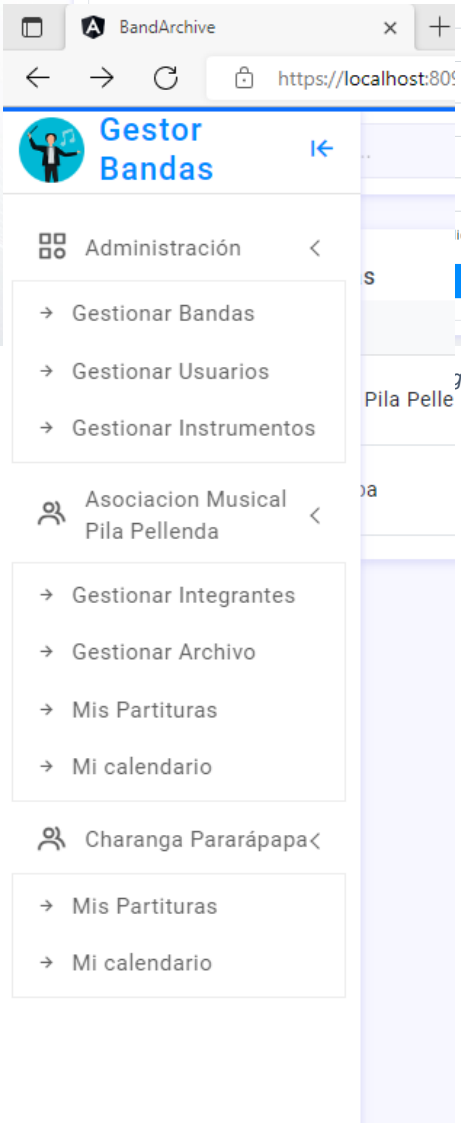


Ilustración 2 - Ventana de Login

La pantalla de *Login,* representada por la ilustración 2, será la encargada de autenticar al usuario dentro del aplicativo.

6.2. Registro





gistro

La ilustración 3 representa la pantalla de registro será la encarga de registrar a nuevos usuarios de la aplicación.

6.3. Pantalla Inicial

Menú

El menú lateral de la aplicación, ilustración 4, se dividirá en dos partes:

- **Administración:** Sólo será visible para los usuarios con rol administrador. Permite realizar las operaciones de gestión de las bandas, usuarios e instrumentos

- **Gestor de agrupaciones:** Se podrá visualizar de dos formas. Si el usuario es administrador del fichero tendrá acceso a todas las opciones del menú. Por otro lado, si no posee el permiso anterior, sólo podrá ver las partituras vinculadas a los instrumentos asignados en la agrupación.

  El menú por defecto está colapsado y podrá acceder a el mediante un botón en la barra superior.

Barra superior

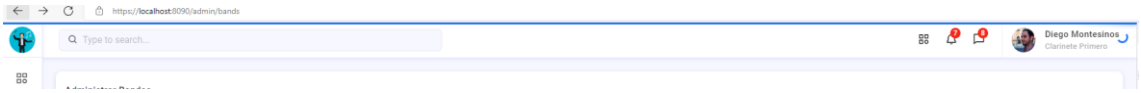


Ilustración 5 - Barra superior

La ilustración 5 nos muestra el encabezado de la página. El encabezado permitirá acceder al menú en su parte izquierda. En la parte derecha mostrará los datos básicos de la persona que esta logada en el sistema, así como permitirá cerrar sesión.

6.4. Administración de la APP

6.4.1. Gestionar Bandas

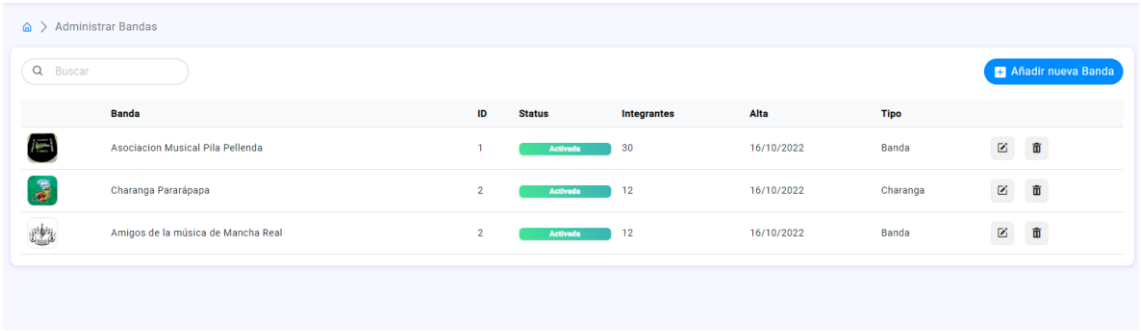


Ilustración 6 - Gestión de bandas de música

Mostrará el listado de bandas registradas en el sistema de una forma similar a la ilustración 6. Mediante un menú de opciones permite activar o desactivar las bandas de música mediante un borrado lógico en base de datos.

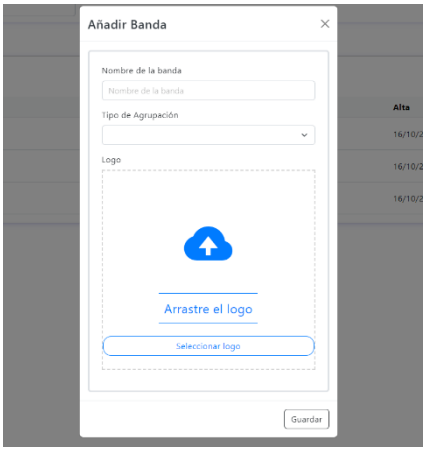


Ilustración 7 - Añadir banda de música

Si se pulsa sobre añadir nueva banda se abrirá un formulario análogo a la ilustración 7 de forma modal, donde el usuario insertará los datos necesarios para añadir la misma

El formulario de edición de una banda existente será similar al de añadir, pero con los datos ya cargados.

6.4.2. Gestionar Usuarios

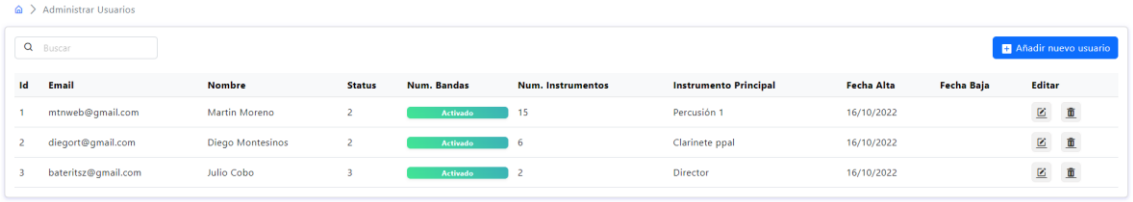


Ilustración 8 - Gestionar usuarios

Mostrará el listado de los usuarios de la app para su gestión global (ilustración 8). El botón añadir nuevo usuario mostrará una ventana modal para la gestión del usuario. Esta ventana será similar si se está editando el usuario (ilustración 9):

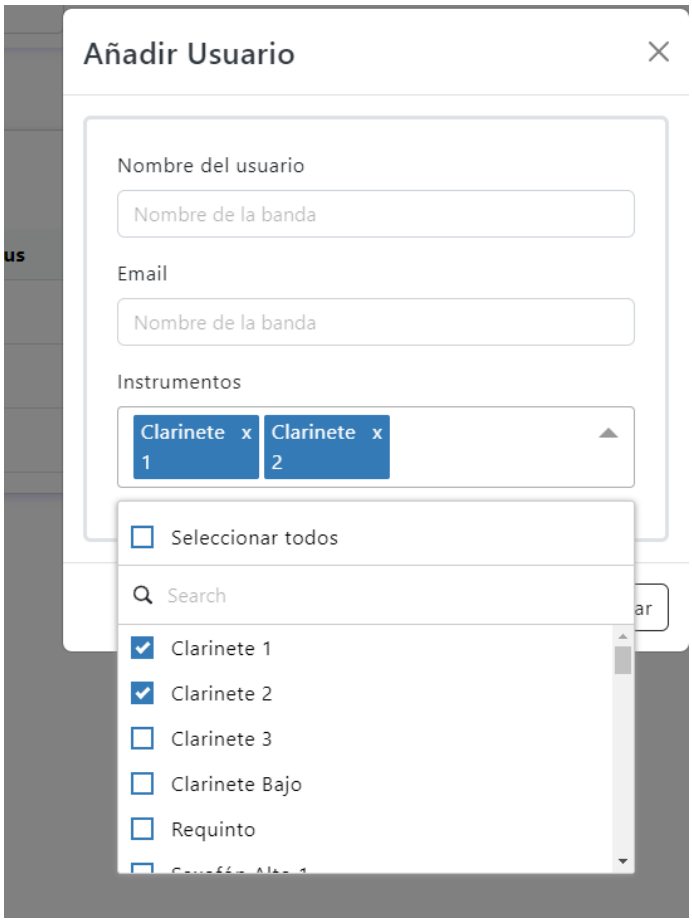


Ilustración 9 - Añadir usuario

6.4.3. Gestionar Instrumentos

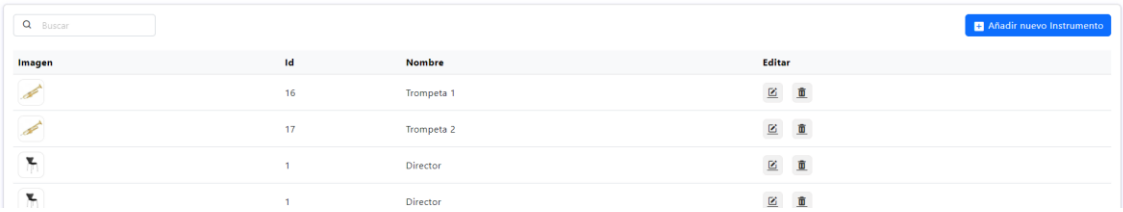


Ilustración 10 - Gestionar instrumentos

La ilustración 10 muestra el listado de los instrumentos que existen actualmente en la aplicación. Si se añade un nuevo instrumento aparecerá la siguiente ventana siguiente modal (ilustración 11). Esta ventana también servirá para la edición del instrumento:

Ilustración 11 - Añadir instrumento

6.5. Gestionar Agrupación musical

6.5.1. Gestión de integrantes

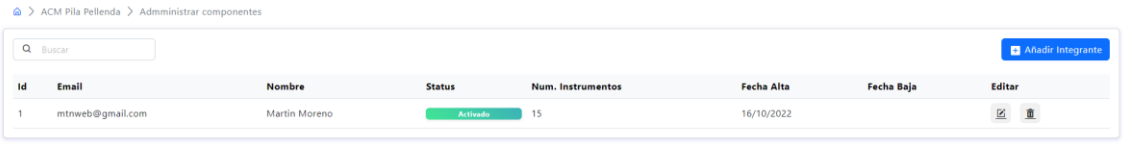


Ilustración 12 - Gestiona agrupación

La ilustración 12 muestra la ventana donde se gestionarán los miembros que pertenecen a una agrupación específica. En la edición del usuario sólo se le permitirá asignar o desasignar instrumentos dentro de la agrupación (ilustración 13).

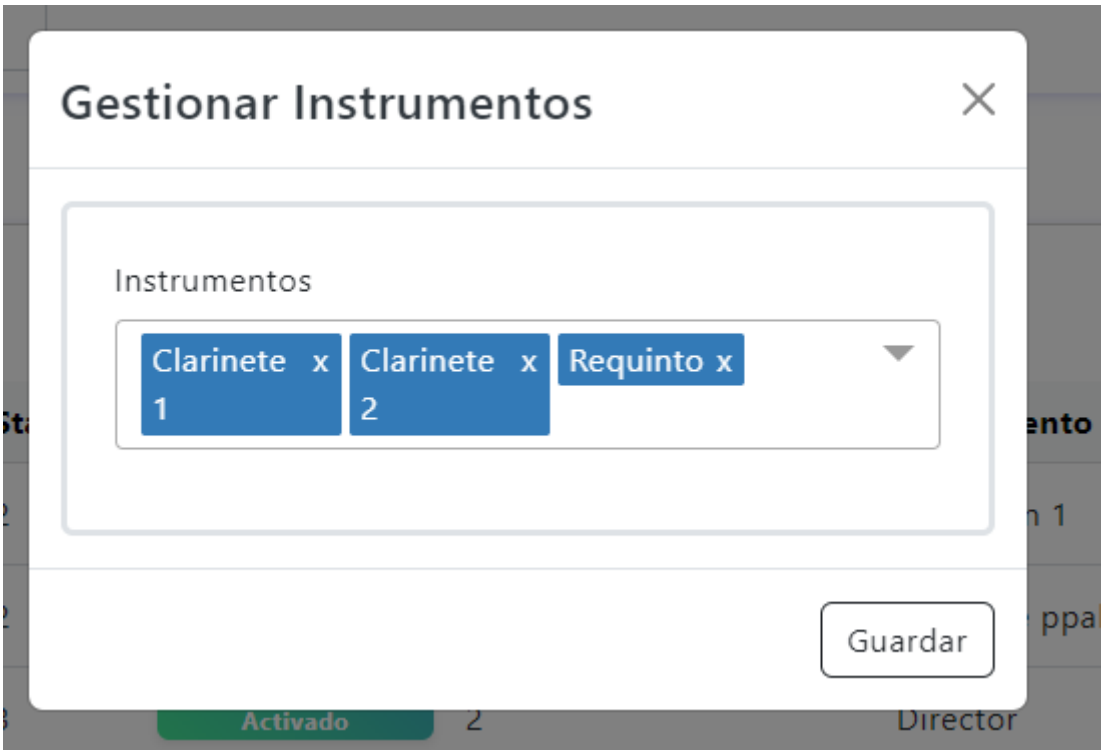


Ilustración 13 - Gestionar instrumentos

6.5.2. Gestión del archivo

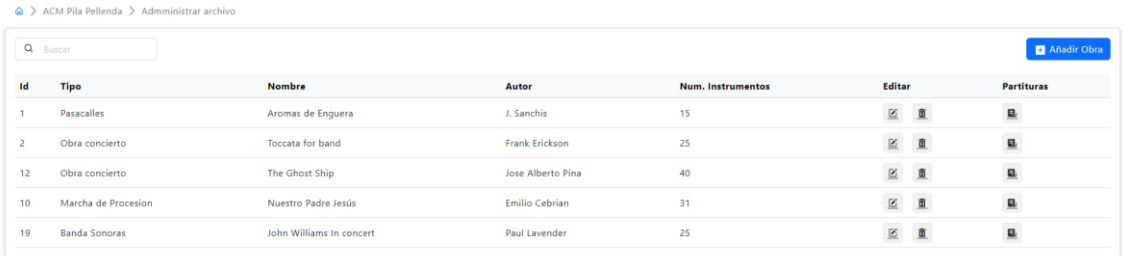


Ilustración 14 - Gestionar archivo

En la ventana representada en la ilustración 14. Se listarán todas las obras vinculadas a una banda. Se podrán editar los campos básicos de la obra y añadir nuevas obras. Si se pulsa sobre el botón de partituras nos llevará a la ventana de gestión de partituras en modo edición.

La ventana de añadir obra será similar a las modales mostradas anteriormente.

6.5.3. Mis partituras

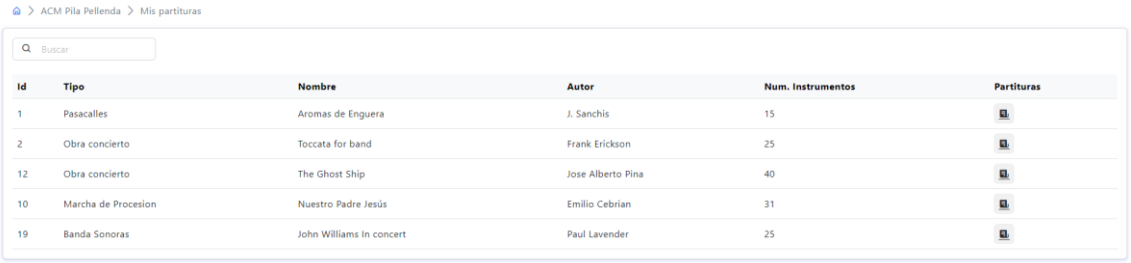


Ilustración 15 - Mis partituras

La ilustración 15 nos muestra una ventana similar a la gestión archivo para representar las partituras pertenecientes a una banda que tiene acceso un usuario, la principal diferencia es que esta ventana será en modo sólo lectura. De igual manera en la vista de partituras no se permitirá ninguna edición.

6.5.4. Visualización de una partitura

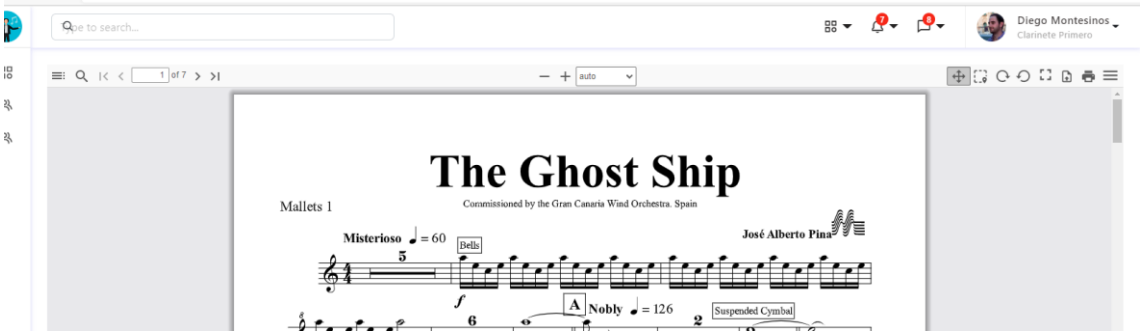


Ilustración 16 - Visor pdf

La *aplicación* dispondrá de un visor *pdf* incorporado en el que poder visualizar e interactuar con la partitura similar a la representación 16.

6.5.5. Gestión de calendario – Mi calendario

Ilustración 17 - Mi calendario

Al igual que las anteriores ventanas esta pantalla tendrá dos modos:

- **Modo administrador:** Se podrá gestionar los eventos futuros: cambiar fecha, lugar, obras, etc. De igual forma se permitirá añadir nuevos eventos.
- **Modo usuario:** Solamente se permitirá visualizar los eventos futuros, la representación 17 muestra esta casuística.

En ambos casos se permitirá acceder a la visualización de la obra desde el nombre de esta.

7. Modelo de BBDD

La ilustración 18 muestra el modelo de base de datos que tendrá la aplicación:

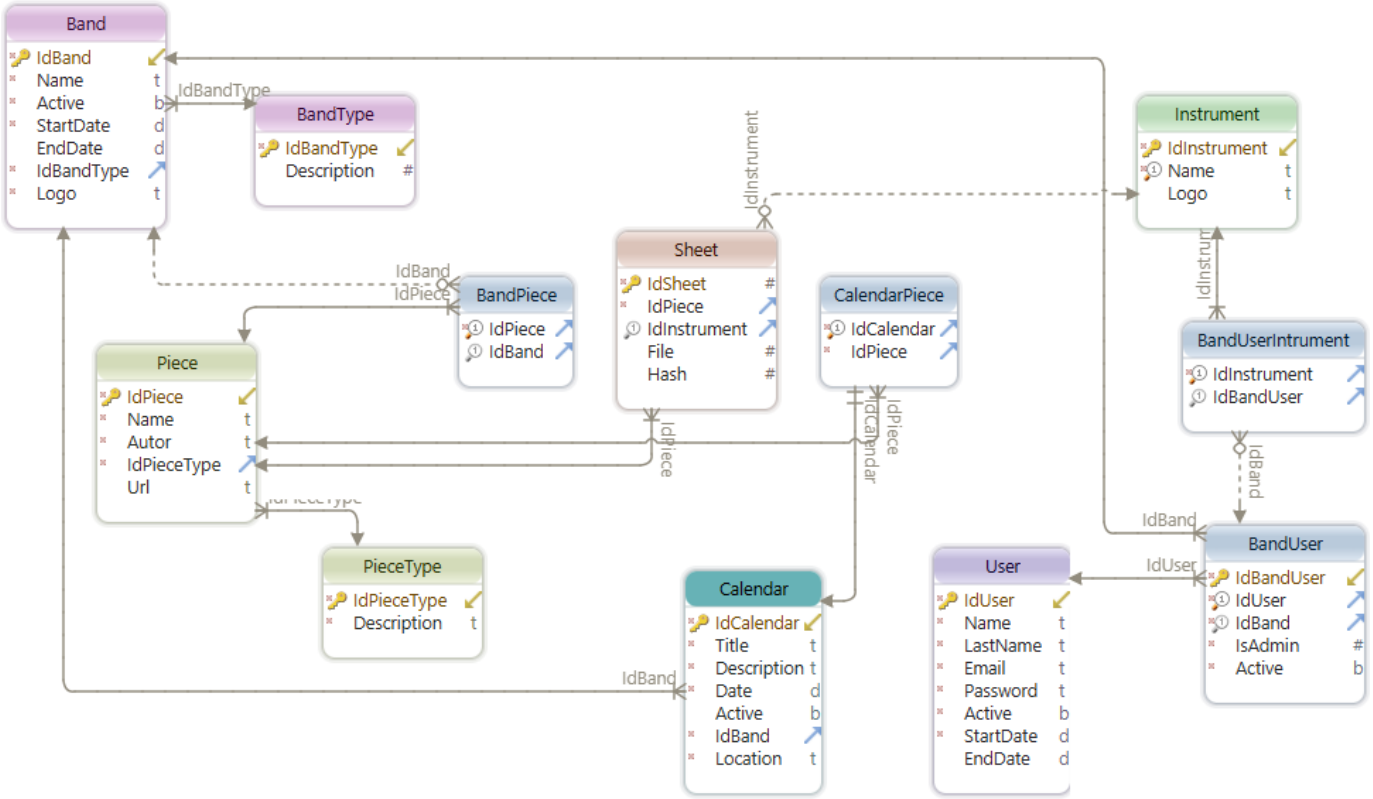


Ilustración 18 - Modelo de BBD

8. Arquitectura de la Aplicación

Para el desarrollo de la aplicación tenemos que distinguir que se han seguido dos patronajes distintos en función de si estamos en *backend o Frontend*. En ambos casos se ha tomado la determinación de trabajar bajo principios *SOLID* y patrones de código limpio (*Clean Code*).

8.1. Backend

Para el backend, tal como muestra la ilustración 19, se ha optado por una arquitectura basada en tres capas principales:

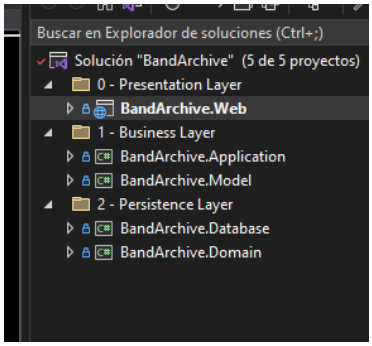


Ilustración 19 - Capas de la aplicación

- **Capa de presentación:** Será la encargada de ofrecer los diferentes Interfaces de comunicación de la *APP*, esta comunicación se realizará mediante una *API*
- **Capa de negocio:** Como un nombre indica será la encargada de gestionar la lógica de negocio
- **Capa de datos o persistencia:** Es la encargada del acceso a datos y persistencia de estos, bajo un patronaje de código basado en entidades de dominio.

Se ha elegido este modelo ya que garantiza una arquitectura donde las capas y, por ende, sus componentes cumplen con el principio de responsabilidad única.

Si entramos un poco más en detalle de cada capa podremos observar que:

8.1.1 Capa de presentación

Está compuesta por un proyecto el cual contiene los controladores, ilustración 20, que se presentan en la API, esta está basada en **AspNetCore 6**[25]**.**

Los paquetes principales usados en esta capa son:

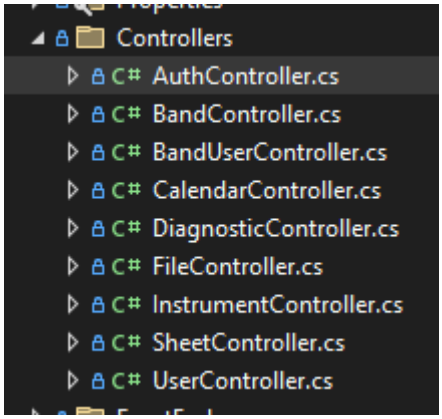


Ilustración 20 - Controladores

- ***Swagger***[26]: Permite consumir de una forma rápida la API, además incorpora patrones de documentación y diseño de esta que facilitan la lectura y estandarización del código.
- ***DotNetCore.IOC***[27] ***y DotNetCore.Logging***[28]: Paquetes que facilitan el log de eventos, trazabilidad y gestión de la inyección de dependencias.

Los principales tipos de clases de esta capa son los controladores, equivalentes a los *endpoints* del api, estando desarrollados a día de hoy visibles en la imagen lateral de esta sección.

8.1.2. Capa de negocio

Contiene dos proyectos: *Application y Models* como se observa en la ilustración 21. El primero es el encargado de gestionar toda la lógica de negocio, operaciones y comunicaciones entre la capa de presentación y datos. Está basado en un patrón factoría y de servicios[29]. El proyecto de modelos contiene las clases modelos que son usadas por la API y los servicios.

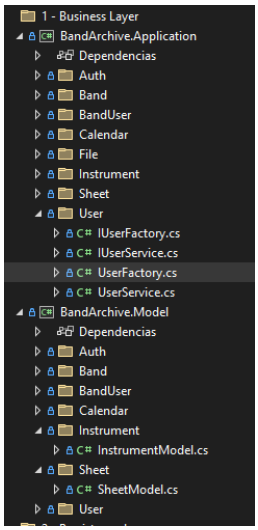


Ilustración 21 - Capa de negocio

Los paquetes principales utilizados en esta capa son:

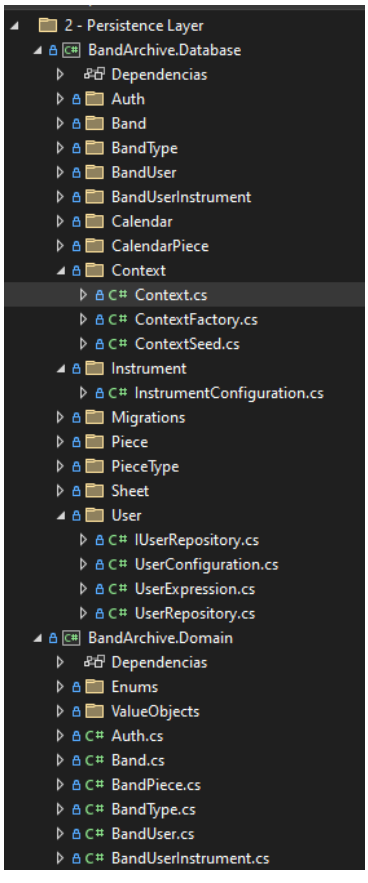
- **DotNetCore.Security**[30]**:** Encargado de gestión de la seguridad, acceso a datos y servicios.

Los principales objetos de clases de esta capa son:

- **Servicios:** encargados de realizar la lógica negocio de la aplicación, todos parten de una interfaz y una clase que implementa su lógica.
- ***Factory's:*** encargados de facilitar la implementación de lógicas complejas de forma centralizada. Al igual que los servicios parte de una interfaz que es implementada en una clase.
- **Modelos:** Utilizados por la capa y la capa superior.

8.1.3. Capa de persistencia o acceso a datos

Como su nombre indica es la encargada de gestionar las comunicaciones entre la BBDD y la capa de negocio. Está compuesta por dos proyectos *Database y Domain* (ilustración 22). El primero se encarga de gestionar las distintas peticiones a la base de datos, las semillas de la APP y el control de cambios de estructura en la BBDD. EL segundo contiene los modelos de dominio utilizados por la *APP* que corresponde a las tablas de la BBDD.



Los principales paquetes utilizados en esta capa son:

- ***EntityFramework:*** Es el ORM más utilizado por los desarrolladores en .NET gracias a su versatilidad y facilidad de uso.
- ***DotNetCoreDomain***[31]***:*** Que facilita la implementación de un patronaje de código basado en Dominio.

Los principales objetos de clases utilizados en esta capa son:

- ***Configurations:*** Son los encargados de indicar al ORM la configuración de campos de cada tabla, sus relaciones, restricciones, etc.
- **Repositorios**: Ofrecen el catálogo de acciones que se pueden realizar en la base de datos. Están basados en una interfaz que es implementado en una clase.

- **Contexto:** Encargado de gestionar la conexión con la BBDD, realizar el mantenimiento de la estructura de esta y la inserción de las semillas.
- ***Expressions:*** Son extensiones de clase que ayudan a los repositorios.
- **Clases de dominio:** Equivalentes a los campos de tabla en la BBDD.

8.2. FrontEnd

En el Frontend hay que tener en cuenta que el *framework* utilizado, Angular, se basa en un patronaje *MVC*[32], por lo que toda la parte relacionada con Frontend mantendrá dicha estructura. Además, se ha optado por una organización del proyecto siguiendo los estándares de facto en el desarrollo de aplicaciones Angular.

La estructura principal por la que se ha optado es la siguiente:

Dentro de la carpeta src (*source* del proyecto) nos encontramos los componentes principales de la app de angular, archivos auxiliares y entornos del aplicativo.

Nos centraremos el directorio app que contiene distintos subdirectorios especializados según su comportamiento dentro del *front*:

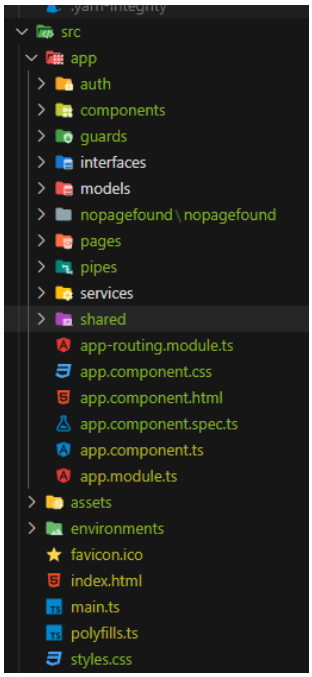


Ilustración 23 - Organización del frontend

- ***Auth***: Contendrá los componentes vinculados a la autentificación del usuario (principalmente el formulario de login). Estos componentes no requieren estar autentificados en el sistema para su acceso.
- ***Components***: Es donde ubicaremos los componentes reutilizables por nuestro *front*.
- ***Guard***: Extensión vinculada a las rutas para identificar si el usuario puede acceder o no a la misma
- ***Model***: Modelos de la aplicación
- ***Pages***: Componentes de la aplicación que requieren estar autentificado
- ***Pipes***: Contiene ayudas en la composición de peticiones al back
- ***Services***: Servicios del *front*, se encargará de gestionar las peticiones al back. Estas peticiones serán de forma totalmente asíncrona.
- ***Shared***: Componentes que son utilizados por la parte autentificada como la que no está autenticada.

Además, para asegurarnos que el código esté lo más limpio posible se ha tomado la decisión que cada parte controle mediante un módulo sus propios componentes y rutas.

Por ejemplo, dentro de la parte autentificada (*pages*). Disponemos de la siguiente estructura que se repite en todas las subpartes anteriores que disponen de componentes:

Por último, se ha optado por una carga perezosa (*Lazy loading*[33]) en algunos componentes. Esto es importante ya que nos ayuda a agilizar los procesos de carga de datos.

Todo lo anterior puede apreciarse en la ilustración 23.

8.2.1. Maquetación de la aplicación

Para facilitar el maquetado y un diseño responsivo de la aplicación de forma fácil se ha optado por utilizar Bootstrap. Se ha optado por este *framework* por las siguientes razones:

- Es un *framework* muy reconocido y usado en la comunidad de desarrollos, que dispone de un alto grado de madurez y mantenimiento.
- Es fácilmente integrable con Angular.
- Dispone de *layouts* responsivos de forma nativa lo que implica que, si se aplican correctamente sus decoradores, nuestra aplicación sea compatible con múltiples resoluciones y dispositivos.

9. Implementación

9.1. Servicio dockerizado en producción

Para la implementación se ha optado por realizar una aplicación "auto hospedable" basada en un contenedor Docker esto permite al proyecto hacerlo más modular ya que permite alojar en distintos servidores back, front y BBDD

La implementación del proyecto es accesible a través de la url: www.mibanda.online con las siguientes credenciales de prueba

- Administrador:
  - Usuario: mtnweb@gmail.com
  - Password: admin

- Usuario de pruebas:
  - Usuario: bateriszt@gmail.com
  - Password: Julio2022

Para el hospedaje se ha utilizado un servidor Linux que es accesible desde internet que, mediante un proxy inverso redirecciona la petición en función del dominio que llega:

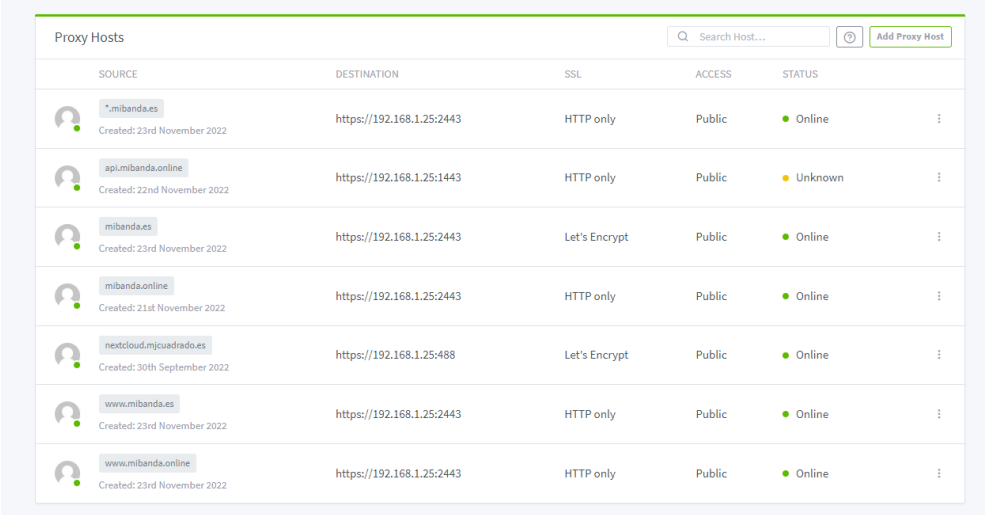


Ilustración 24 - Detalle del proxy inverso

La ilustración 24 muestra el proxy inverso configurado basado en una configuración nginxproxymanager[34], [35] junto a letsEncrypt[36], [37] para la generación de certificados SSL que garanticen la seguridad punto a punto en la aplicación.

9.1.1. Proceso de puesta en producción

El proyecto cuenta con varios archivos del tipo Docker-compose[38] que facilitan la puesta en producción del aplicativo según se necesite.

La puesta en producción, una vez tenemos descargado el código, se realiza mediante el siguiente script:

docker compose -f docker-compose.yaml up --build -d

La ilustración número 25 muestra el contenido del fichero “Docker-compose.yaml”

```yaml
version: "3.7"
services:
    api:
        image: bandarchiveapi
        container_name: bandarhiveapi
        restart: always
        build:
            context: ./source
            dockerfile: dockerfile
        environment:
            - ConnectionStrings__Context=Server=192.168.1.25;Database=tfgDB;User Id=tfg;Password=mjcuadrado;
            - Serilog__WriteTo__1__Args__path=/app/logs/
            - ASPNETCORE_URLS=https://+:443;http://+:80
            - ASPNETCORE_Kestrel__Certificates__Default__Password=P4ssW0rd!
            - ASPNETCORE_Kestrel__Certificates__Default__Path=/https/mibanda.online.pfx
        networks:
            - proxynet
        ports:
            - 8090:80
            - 1443:443
        volumes:
            - ./conf.d/mibanda.online/:/https/
            - /mnt/disk3/tfg/Files:/app/Files
    front:
        image: bandarchivefront
        container_name: bandarhivefront
        restart: always
        build:
            context: ./FrontEnd
            dockerfile: dockerfile
        ports:
            - 8888:80
            - 2443:443
        networks:
            - proxynet
        volumes:
            - ./conf.d/mibanda.online/:/certs/
networks:
    proxynet:
        external: true
```

Ilustración 25 - Fichero Docker compose producción

Este fichero genera dos contenedores en una misma red proxy.

El primero de ellos contiene una imagen Linux de .NET Core y la compilación del backend (ilustración 26).

```
# .NET SDK
FROM mcr.microsoft.com/dotnet/sdk:6.0-alpine AS dotnet
WORKDIR /api



# Copy Projects
COPY . .

RUN ls -la

# .NET Restore
RUN dotnet restore ./Web/BandArchive.Web.csproj
RUN ls -la


# .NET Publish
RUN dotnet publish ./Web/BandArchive.Web.csproj -c Release -o dist --no-restore
RUN ls -la

# .NET Runtime
FROM mcr.microsoft.com/dotnet/aspnet:6.0-alpine as app
WORKDIR /app
RUN apk add --no-cache icu-libs
ENV DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=false
COPY --from=dotnet /api/dist .
ENTRYPOINT ["dotnet", "BandArchive.Web.dll"]
```

Ilustración 26 - Proceso de compilación del backend

El segundo contenedor tiene la imagen del front bajo servidor de autohospedaje nginx (ilustración 27). Al ser código html5 puro no necesita ninguna otra librería.

```
# Angular
FROM node:18-alpine AS angular
WORKDIR /front
COPY . ./
RUN npm run restore
RUN npm run build

# BUILD WEB
FROM nginx:1.17.1-alpine
COPY ./nginx.conf /etc/nginx/conf.d/nginx.conf
COPY --from=angular /front/dist /usr/share/nginx/html
```

Ilustración 27 - Configuración del frontend

La BBDD de producción está basada en una imagen independiente de SQL Server tal como se puede observar en la configuración de la ilustración 25.

De igual forma, de cara a garantizar la persistencia de los datos, el proyecto de back tiene vinculada una carpeta del servidor que lo hospeda que contiene las imágenes y partituras de la app.

9.2. Test por parte de la UOC

Así mismo se ha elaborado un fichero Docker-compose[39], [40] para que la app pueda ser testada por la UOC.

Para dicho testeo es necesario disponer del siguiente software:

9.2.1. Docker

La instalación de Docker se puede realizar de forma gratuita en cualquier sistema operativo, basta con acudir a su página web y seguir los pasos para:

- Windows: https://docs.docker.com/desktop/install/windows-install/
- Linux (Ubuntu): https://docs.docker.com/engine/install/ubuntu/
- Linux (Debian): https://docs.docker.com/engine/install/debian/
- Mac: https://docs.docker.com/desktop/install/mac-install/
- Otros sistemas pueden ser consultados en la página web de Docker.

9.2.2. .NET Framework

El framework es necesario para la generación de certificados SSL válidos para las pruebas en contenedores locales. De la siguiente URL https://dotnet.microsoft.com/en-us/download/dotnet/6.0 se debe descargar e instalar:

- SDK 6 en su última versión (actualmente 6.0.404)
- ASP.NET Core Runtami en su última versión (actualmente 6.0.12)

9.2.3. Ejecución del proyecto basado en Docker Windows

Una vez instalado Docker en nuestro ordenador, deberemos acceder a la carpeta del proyecto descargado y ejecutar un símbolo de sistema en la raíz para ejecutar los siguientes pasos:

Solo la primera vez que se ejecute el proyecto:

1. mkdir conf.d
2. dotnet dev-certs https –clean
3. dotnet dev-certs https -ep ./conf.d/https/dev_cert.pfx -p Madison
4. dotnet dev-certs https –trust

Resto de ejecuciones posteriores

5. docker compose -f docker-compose-uoc.yaml up --build -d

Este paso autodescargará los servicios de autohospedaje del proyecto y levantará los servicios de BBDD, back y front.

Si todo ha ido correctamente deberemos tener los contenedores que se muestran en la ilustración 28 en la consola de Docker y se podrá acceder al servicio desde http://localhost:8080

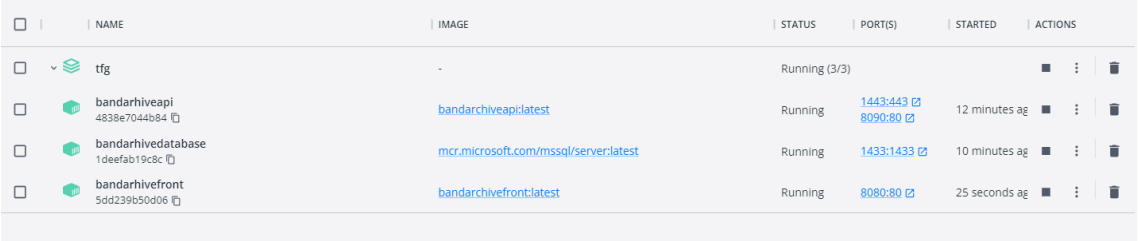


Ilustración 28 - Docker desktop, contenedores generados

Las credenciales de acceso para la prueba del aplicativo son las mismas que las de producción.

La principal diferencia entre este entorno de pruebas y el de producción es que además de los dos contenedores de back y de front, se usa un contenedor de BBDD para ahorra tener que instalar SQL en el equipo de la persona que lo pruebe.

Los archivos de configuración son idénticos cambiando las rutas de acceso a back y varios cambios menores de configuración.

9.2.4. Otros métodos para testar la aplicación

Por último, se puede testar la aplicación mediante debug en Visual Studio (Back) y Visual Studio Code (front). Si se opta por este método el usuario tendrá que cambiar la cadena de conexión del App.config y copiar la carpeta “Files” en el directorio de compilación de backend.

Además, deberá de configurar una BBDD en SQL accesible desde la aplicación.

10. Código fuente.

El código fuente puede consultarse en el repositorio de github:

https://github.com/mjcuadrado/BandArchive

Al ser un repositorio privado, se debe solicitar previamente acceso por correo electrónico.

11. Evaluación de la propuesta por parte de los usuarios

Para garantizar el correcto funcionamiento de la aplicación se han realizado diferentes pruebas con usuarios reales con el fin de ver el grado de adaptación y posibles mejoras que se pueden realizar a el desarrollo entregado.

Se han buscado distintos perfiles con el fin de obtener una imagen lo más transparente posible de cara a la evaluación. Para la realización de la propuesta se ha habilitado el acceso a estos usuarios durante 48 horas explicando de una forma breve y sin entrar en profundidad qué pueden hacer con los permisos que disponían. Pasadas estas horas se han realizado una serie de entrevistas para evaluar el grado de satisfacción obtenido.

11.1. Entrevista realizada

A todos los usuarios que han testado la aplicación se les ha hecho una entrevista base que consta de las siguientes preguntas:

- Datos de la persona: Nombre, edad
- Función dentro de la banda de música
- Nivel musical que posee
- Nivel informático que posee
- ¿Pertenece a más de una banda de música? ¿Ha podido acceder a los instrumentos asignados a cada una de ellas?
- ¿Me puede decir desde qué dispositivos ha accedido a la aplicación?
- ¿Ha podido gestionar alguna obra o la banda asignada? (solo a roles de administración)
- ¿Le ha resultado fácil realizar estas operaciones?
- ¿Qué opinión tiene de la aplicación? ¿La usuaría? ¿Qué mejoraría?

11.2. Entrevista a Julio Cobo

- **Datos de la persona: Nombre, edad, función dentro de la banda de música**
  Nombre: Julio Cobo
  Edad: 33 años
  Sexo: Masculino
- **Función dentro de la banda**
  Soy profesor de la asignatura de Orquesta del conservatorio Profesional de música José Salinas de Baza (Granada). Además, soy miembro de distintas agrupaciones musicales como Amigos de la Música de Mancha Real de la cual fui director hasta el año pasado y actualmente toco de forma esporádica. Además, colaboro con otras bandas de música.
- **Nivel musical que posee**
  Tengo el Grado Superior en Piano y una especialización en dirección de orquesta. Además, soy percusionista.
- **Nivel informático que poseen**
  La informática me manejo con ella
- **¿Pertenece a más de una banda de música? ¿Ha podido acceder a los instrumentos asignados a cada una de ellas?**

Lo primero, gracias por ponerme como director de la Banda de Torres. Ya sabes que me encantaría volver algún día a ella como director. Como te he comentado anteriormente pertenezco a Amigos de la música de Mancha Real y desarrollo el rol de percusionista con percusión sinfónica.

He podido acceder a todos los instrumentos y he visto que no falta ninguno.

- **¿Me puede decir desde qué dispositivos ha accedido a la aplicación?**
  Al principio entré desde mi móvil un iPhone y puede ver todo y realizar algunos ajustes. Pero donde realmente he visto el potencial de la aplicación ha sido en el ordenador.
- **¿Ha podido gestionar alguna obra o la banda asignada? (solo a roles de administración)**
  Sí, he realizado alguna gestión que otra: como administrar componentes, subir y sustituir una partitura.
- **¿Le ha resultado fácil realizar estas operaciones?**
  Me ha parecido muy fácil realizar los cambios
- **¿Qué opinión tiene de la aplicación? ¿La usuaria? ¿Qué mejoraría?**
  En general me ha parecido una aplicación con gran potencial. Pero a diferencia de como tú, la vez creo que inviable utilizarla en la calle. Para actuaciones que conlleven no moverse como conciertos o ensayos creo que es la aplicación que muchas bandas necesitan.

11.3. Entrevista a Diego Montesinos

- **Datos de la persona: Nombre, edad, función dentro de la banda de música**
  Nombre: Diego Montesinos
  Edad: 34 años
  Sexo: Masculino
- **Función dentro de la banda**
  Realizo funciones de Archivo dentro de la Asociación Musical Pila Pellenda de Torres, de la cual soy músico.
- **Nivel musical que posee**
  Tengo el Grado Medio de clarinete y llevo tocando 20 años en distintas agrupaciones.
- **Nivel informático que poseen**
  Soy Ingeniero Superior Informático
- **¿Pertenece a más de una banda de música? ¿Ha podido acceder a los instrumentos asignados a cada una de ellas?**
  Principalmente toco en la banda de Torres, también toco en otras agrupaciones de forma esporádica
- **¿Me puede decir desde qué dispositivos ha accedido a la aplicación?**
  He probado la aplicación en distintos dispositivos como móviles, ipad y ordenadores.
- **¿Ha podido gestionar alguna obra o la banda asignada? (solo a roles de administración)**
  He gestionado componentes y archivo de la banda
- **¿Le ha resultado fácil realizar estas operaciones?**
  Aunque la funcionalidad básica está hecha me parece que la subida de partituras es mejorable ya que, como sabes, normalmente están todas en un fichero y no de forma individual.

- **¿Qué opinión tiene de la aplicación? ¿La usuaria? ¿Qué mejoraría?**
  Era algo que teníamos en mente durante muchos años y creo que el enfoque que tiene es el correcto. Ahora solo queda mejorarla y meter más datos y partituras, por tanto, sí que la voy a utilizar.

11.4. Entrevista a Antonio Garrido

- **Datos de la persona: Nombre, edad, función dentro de la banda de música**
  Nombre: Antonio Garrido
  Edad: 40
  Sexo: Masculino
- **Función dentro de la banda**
  Soy director de la Banda de Música Municipal de Bailén y de la Banda de Música de la Carolina ambas en Jaén.
- **Nivel musical que posee**
  Grado Superior acabado de Flauta travesera. Distintos cursos de dirección
- **Nivel informático que posee**
  No me manejo muy bien con la informática
- **¿Pertenece a más de una banda de música? ¿Ha podido acceder a los instrumentos asignados a cada una de ellas?**
  Toco con quien me deja, incluso contigo, aunque cada vez menos por las obligaciones con las dos bandas que dirijo. Me has asignado todos los instrumentos que toco, incluido el flautín. Una alegría ver que todavía alguien recuerda que toco el trombón y el bombardino.
- **¿Me puede decir desde qué dispositivos ha accedido a la aplicación?**
  Solo lo he probado desde el ordenador.
- **¿Ha podido gestionar alguna obra o la banda asignada? (solo a roles de administración)**
  Solamente he subido una partitura y creado una obra para ver si funciona y he visto que sí. No he podido ni gestionar mi banda porque no hay usuarios registrados
- **¿Le ha resultado fácil realizar estas operaciones?**
  En general me ha parecido fácil realizar los cambios.
- **¿Qué opinión tiene de la aplicación? ¿La usuaria? ¿Qué mejoraría?**
  Creo que no la utilizaría ya que soy de la antigua escuela, es decir, el papel. Pero para actuaciones esporádicas me parece una idea muy buena

11.5. Entrevista a Estrella Serrano

- **Datos de la persona: Nombre, edad, función dentro de la banda de música**
  Nombre: Estrella Serrano
  Edad: 34
  Sexo: Femenino
- **Función dentro de la banda**
  Soy música
- **Nivel musical que poseen**
  Tengo sin terminar el grado medio de piano, en banda toco el requinto. También toco la flauta.

- **Nivel informático que poseen**
  Nivel medio
- **¿Pertenece a más de una banda de música? ¿Ha podido acceder a los instrumentos asignados a cada una de ellas?**
  Pertenezco a una banda de música solamente, he visto que me has asignado en otra banda que no es la mía, pero los instrumentos que toco están todos
- **¿Me puede decir desde qué dispositivos ha accedido a la aplicación?**
  He probado la aplicación desde la tablet
- **¿Le ha resultado fácil realizar estas operaciones?**
  Sí, aunque algunas partituras salen al revés, después me di cuenta de que se pueden girar.
- **¿Qué opinión tiene de la aplicación? ¿La usuaria? ¿Qué mejoraría?**
  Si me vienen todas mis partituras ahí sí la usuaria ya que es más cómodo que el papel, aunque en pasacalles creo que no podría usarla.

11.6. Entrevista a Francisco Martínez

- **Datos de la persona: Nombre, edad, función dentro de la banda de música**
  Nombre: Francisco Martínez
  Edad: 24
  Sexo: Masculino
- **Función dentro de la banda**
  Trompeta
- **Nivel musical que posee**
  Soy músico desde pequeño, pero no tengo titulación
- **Nivel informático que posee**
  Tengo los conocimientos que cualquier joven tiene a nivel de informática
- **¿Pertenece a más de una banda de música? ¿Ha podido acceder a los instrumentos asignados a cada una de ellas?**
  Pertenezco a la banda de Torres y a la charanga en la que tú tocas
- **¿Me puede decir desde qué dispositivos ha accedido a la aplicación?**
  He probado la aplicación desde el móvil y el ordenador
- **¿Le ha resultado fácil realizar estas operaciones?**
  En el móvil las partituras se ven muy pequeñas, por lo que he mirado en el ordenador que salen perfectas.
- **¿Qué opinión tiene de la aplicación? ¿La usuaria? ¿Qué mejoraría?**
  Me parece una muy buena idea para dejar de utilizar los papeles.

11.7. Evaluación de las entrevistas

En general para los usuarios es una aplicación fácil de utilizar y han podido realizar todas las operaciones que se les había pedido sin ayuda. Como mejoras futuras se deberían de realizar los siguientes cambios:

- Cambiar el sistema en el que un usuario sube una partitura para poder hacerlo de forma global.
- Corregir el bug del menú en dispositivos móviles.
- Buscar cómo poder utilizarla en dispositivos que puedan ser utilizados en la calle.

12. Objetivos conseguidos y cambios realizados frente al planteamiento inicial

En general, se puede decir que se han conseguido casi todos los objetivos marcados al inicio del desarrollo de este trabajo de fin de grado. Si bien se han tenido que modificar el alcance de algunos puntos ya que en el transcurso del desarrollo cuando se ha hablado con las agrupaciones que van a iniciar la prueba piloto de la aplicación se ha visto que había puntos en los que se podía mejorar la aplicación haciéndola más simple.

El primer punto donde ha existido un cambio significativo frente al planteamiento inicial ha sido la forma en la que se gestionan las partituras y obras del archivo. En un principio se consideraba que cada banda tendría su archivo único con sus obras.

Después de varias reuniones se consideró en pensar una aplicación mucho más colaborativa en el que el archivo es único para todas. Es decir, cualquier agrupación que en un futuro ingrese se encontrará que tendrá acceso a todo el catálogo sin restricciones. Esto se ha hecho ya que la mayoría de las obras que se tocan en agrupaciones actualmente son compartidas entre agrupaciones y estas, a su vez, las comparten con otras.

El siguiente punto que ha tenido cambios en su alcance es el calendario. Al igual que el archivo en las reuniones mantenidas con las bandas implicadas en el proyecto hemos observado que puede realizarse un desarrollo mucho más completo en el que además de ver las obras que se tocarán en un determinado evento la persona tenga acceso a sus partituras.

En relación de los objetivos conseguidos creo que se han cumplido todos los demás. La aplicación es capaz de gestionar distintas bandas, sus integrantes, los instrumentos que cada uno de ellos tiene y, el objetivo principal, las obras y partituras de su archivo.

Además, los usuarios de cualquier banda pueden entrar y ver solo las partituras de los instrumentos que tienen asignados. Esto es muy importante ya que no siempre se tienen asignados los mismos instrumentos en las bandas y cuando se tienen que compartir partituras con personas ajenas, los llamados refuerzos de actuación, se pierde un tiempo considerable buscándolas, optando la mayoría de las veces por compartirle la obra entera y no la parte del instrumento que toca.

A nivel de administración también se han quedado desarrollados todos los puntos. Permitiendo gestionar los usuarios, bandas e instrumentos.

Si nos centramos en el desarrollo propiamente dicho, se han cumplido de igual forma todos los objetivos.

A nivel de *backend* se ha diseñado una arquitectura que permite implementar nuevas funcionalidades de forma rápida bajo patrones factoría. Además, se han generado distintos *endpoints* en la API para gestionar distintas funcionalidades que aún no están implementadas. Todo lo anterior es supervisado por un servicio de autorización que, en función del usuario, controla si la petición realizada sea de la índole que sea puede realizarse. Así mismo la aplicación cuenta con un log interno que garantiza la trazabilidad y un gestor propio de excepciones globales que hace que la aplicación sea robusta y tolerante a los fallos.

El desarrollo del front también ha cumplido todos sus objetivos ya que se ha realizado un desarrollo muy modular. En este apartado se debe destacar la implementación de un

componente de listados que facilita de forma exponencial la gestión general de la aplicación. De igual manera que el back contamos con mecanismos para la gestión centralizada de peticiones, sus respuestas y, en caso de que existan, de los errores.

A nivel de integración continua se han realizado incluso más objetivos de los marcados. En un principio, se indicó que la posibilidad de *dockerizar* los distintos entornos para generar contenedores que pudieran ser implementados de forma fácil en cualquier entorno. No solo se ha implementado este sistema, sino que se han generado distintos entornos que facilitan el *testing* y el despliegue de la aplicación.

Mención aparte merece la persistencia de los archivos ya que, aunque la solución se encuentre en un contenedor, los archivos se enrutan a una unidad del hosting que está en un servidor NAS garantizando que existan copias de seguridad en caso de algún problema.

Por último, a nivel de base de datos podemos indicar que también se ha cumplido todos los objetivos ya que esta se gestiona íntegramente desde el control de código fuente mediante migraciones. Esto nos permite versionar la base de datos y saber cuándo y cómo se realizaron los cambios. Además, el framework utilizado, genera de forma fácil todas las relaciones, dependencias y estructuras de tabla, pudiendo consultarlas de forma inmediata ante una duda.

En conclusión, si tenemos en cuenta todo los anterior podemos destacar que se ha realizado un desarrollo totalmente satisfactorio en el que se han cumplido en tiempo y forma todos los objetivos marcados. Además, se han dejado acotadas nuevas funcionalidades y desarrollos que pueden realizarse en un futuro como mejora de este.

13. Mejoras futuras

En los anteriores puntos se han indicado distintas mejoras que serán regidas en este epígrafe del trabajo en distintas áreas de la aplicación.

A nivel de funcionalidad la mejora más importante que se debe realizar es en el calendario, ya que se deben acceder a las partituras asignadas al evento de todos los instrumentos vinculados en la banda del usuario.

De igual forma, en la lista de obras, tanto en edición por parte del administrador de la banda como en lectura por parte del usuario, se ha dejado preparado un botón para visualizar la en video. Este desarrollo debería ser relativamente rápido puesto la obra ya tiene un campo a nivel de base de datos donde registrar el video que es devuelto por el *backend.* También, como complemento a lo anterior, se está barajando la opción de disponer de un botón que permita descargarlas todas para que el usuario pueda imprimirlas con facilidad.

El segundo desarrollo importante que se debe realizar sobre el existente es la generación de un *wizard* de importación. En este *wizard* se cargaría el archivo pdf con la obra completa y se marcarían las páginas en las que están cada instrumento. Una vez reviera la petición el *backend* realizaría un proceso de trocear el pdf y guardarlo por partes, generando estas en el archivo.

También debe corregirse distintos problemas que existen con el menú. Este está implementado en JQuery y no en Angular, lo que hace que en dispositivos móviles cuando pulsas sobre un *item* aparece siempre desplegado y no colapsado.

En nuevas funcionalidades que habría que hacer desde cero la que más consenso tiene entre las distintas bandas es la generación de un catálogo de métodos que, al igual que con las obras, permita asignar a alumnos de las escuelas musicales métodos para su instrumento o solfeo. La funcionalidad anterior abriría la puerta a otros desarrollos como, por ejemplo, que el profesor de la escuela pueda mandar una página específica al alumno para su estudio.

Debido a que no entraba dentro del alcance, es difícil de documentar y no funciona aún al 100% el sistema de integración continúa basado en Jenkins[41] debe mejorarse. Actualmente vigila las *pull request* de GitHub[42], se baja los cambios y despliega, pero el proceso es mejorable.

Por último, se quiere implementar un sistema de pago por uso en el que las bandas pagaran una cantidad de dinero mensual por usuario o grupos de usuarios. Esta cantidad serviría para pagar los servidores donde actualmente está alojada la aplicación, los desarrollos nuevos que se realicen y otros gastos como pueden ser los dominios contratados.

14. Conclusiones

En este apartado me gustaría hablar de forma personal. Esta aplicación es un sueño que teníamos un gran amigo mío (Diego Montesinos) y yo durante muchos años. Tanto es así que en su momento, hace casi 10 años, empezamos a realizarla pero se quedó a medias.

Este proyecto ha servido para replantearlo y, aprovechando la tecnología actual, llevarlo a cabo. No les voy a mentir, apostaba a caballo ganador, el proyecto estaba bien estructurado en mi cabeza y solamente ha sido necesario unir los puntos. Además, a diferencia de ese momento, he elegido un lenguaje que domino como es C# y un marco de trabajo como es .NET.

En contraprestación a lo anterior, creo que es conveniente indicar que, aunque poseía ciertos conocimientos en Angular, no era un entorno que dominara y quizás es en este apartado donde más he aprendido. En este sentido, antes de escribir cualquier línea de código he intentado documentarme sobre qué buenas prácticas debería de realizar y cómo estructurar el código para que fuera lo más profesional posible.

También he aprovechado la tesitura para aprender algo nuevo en el entorno que trabajo a diario. La utilización de Docker al principio fue de las partes más complicadas del proyecto, pero, al final, ha permitido que con un solo comando se desplieguen los entornos en un servidor.

En mis casi 20 años de experiencia desarrollando, tengo algún que otro proyecto del que me siento orgulloso y, desde hace relativamente poco, puedo decir que este es uno de ellos. Por lo que mi conclusión global, y más viendo cómo está funcionando con distintas agrupaciones en producción, es que el proyecto ha merecido la pena y que es mucho mejor que como estaba en mi cabeza.

15. Cambios en la planificación prevista

Se han realizado varias modificaciones que afectan al proyecto final entregado:

- El archivo, tras pactarlo con las bandas que explotaran el sistema, será único para todas. Es decir, cualquier banda que tenga acceso al aplicativo tendrá acceso a todas las obras existentes en él. Se ha optado por este enforque ya que no se duplicarán las obras y será una *app* "colaborativa".
- El borrado de todas las entidades de la BBDD es lógico para prevenir errores del usuario y que puede deshacer estos de una forma rápida.
- El calendario está pendiente de desarrollo.
- La edición de instrumentos está pendiente de desarrollo

Por lo demás, la aplicación cumple con todas las historias de usuario propuestas y se espera que el entorno de producción sea testado por, al menos, 6 bandas de música en los próximos meses.

Para las pruebas por parte de la UOC se ha volcado un pequeño catálogo con varias obras.

Referencias

[1] wikipedia, «Qué es un framework», *https://es.wikipedia.org/wiki/Framework*. https://es.wikipedia.org/wiki/Framework (accedido ene. 14, 2023).

[2] L. del Valle Hernandez, «¿Qué es un ORM?», *https://programarfacil.com/blog/que-es-un-orm/*. https://programarfacil.com/blog/que-es-un-orm/ (accedido ene. 14, 2023).

[3] Microsoft, «Code First en una nueva base de datos», *https://learn.microsoft.com/es-es/ef/ef6/modeling/code-first/workflows/new-database*, sep. 28, 2022.

[4] H. Baumann, «Conoce qué es CRUD y por qué es fundamental para desarrollar sitios y aplicaciones», *https://www.crehana.com/blog/transformacion-digital/que-es-crud/*, feb. 08, 2022. https://www.crehana.com/blog/transformacion-digital/que-es-crud/ (accedido ene. 14, 2023).

[5] Microsoft, «Introducción a .NET Framework», *https://learn.microsoft.com/es-es/dotnet/framework/get-started/*, sep. 22, 2022.

[6] M. Romanos, «Entity Framework y Linq», *https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/entity-framework-y-linq*. https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/entity-framework-y-linq (accedido ene. 14, 2023).

[7] F. Cristancho, «¿Qué es Angular?», *https://talently.tech/blog/que-es-angular/*, jul. 25, 2022. https://talently.tech/blog/que-es-angular/ (accedido ene. 14, 2023).

[8] U. Hernández, «Qué es TypeScript», *https://codigofacilito.com/articulos/typescript*, jun. 03, 2018. https://codigofacilito.com/articulos/typescript (accedido ene. 14, 2023).

[9] Wikipedia, «Bootstrap», *https://es.wikipedia.org/wiki/Bootstrap_(framework)*. https://es.wikipedia.org/wiki/Bootstrap_(framework) (accedido ene. 14, 2023).

[10] ¿QUÉ ES MICROSOFT SQL SERVER Y PARA QUÉ SIRVE?, «¿QUÉ ES MICROSOFT SQL SERVER Y PARA QUÉ SIRVE?», *https://intelequia.com/blog/post/2948/qu%C3%A9-es-microsoft-sql-server-y-para-qu%C3%A9-sirve*, oct. 18, 2021. https://intelequia.com/blog/post/2948/qu%C3%A9-es-microsoft-sql-server-y-para-qu%C3%A9-sirve (accedido ene. 14, 2023).

[11] A. Nocentino, «Container Limits and SQL Server», *https://www.nocentino.com/posts/2021-07-25-container-limits-and-sql-server/*, jul. 25, 2021. https://www.nocentino.com/posts/2021-07-25-container-limits-and-sql-server/ (accedido ene. 14, 2023).

[12] Amazon, «¿Qué es Docker?», *https://aws.amazon.com/es/docker/#:~:text=Docker%20es%20un%20sistema%20operativo%20(o%20runtime)%20para%20contenedores.,crear%2C%20iniciar%20o%20detener%20contenedores.* https://aws.amazon.com/es/docker/#:~:text=Docker%20es%20un%20sistema%20operativo%20(o%20runtime)%20para%20contenedores.,crear%2C%20iniciar%20o%20detener%20contenedores. (accedido ene. 14, 2023).

[13] Wikipedia, «GitHub», *https://es.wikipedia.org/wiki/GitHub*. https://es.wikipedia.org/wiki/GitHub (accedido ene. 14, 2023).

[14] Amazon, «¿Qué es un IDE?», *https://aws.amazon.com/es/what-is/ide/*. https://aws.amazon.com/es/what-is/ide/ (accedido ene. 14, 2023).

[15] Microsoft, «Visual Studio Community». https://visualstudio.microsoft.com/es/vs/community/ (accedido ene. 14, 2023).

[16] Microsoft, «Visual Studio Code», *https://code.visualstudio.com/*. https://code.visualstudio.com/ (accedido ene. 14, 2023).

[17] «Licencia de VS Community», *https://visualstudio.microsoft.com/es/vs/community/license-terms/vs2022/-ga-community/*. https://visualstudio.microsoft.com/es/vs/community/license-terms/vs2022/-ga-community/ (accedido ene. 14, 2023).

[18] «Licencia VS Code», *https://code.visualstudio.com/license?lang=es*. https://code.visualstudio.com/license?lang=es (accedido ene. 14, 2023).

[19] Giovanny Cifuentes, «Definition of Done y Definition of Ready», *https://giovannycifuentes.com/definiciones-ante-empezar-primer-sprint-dor-dod/*, feb. 09, 2018. https://giovannycifuentes.com/definiciones-ante-empezar-primer-sprint-dor-dod/ (accedido ene. 14, 2023).

[20] Microsoft, «Asynchronous programming in C#. », *https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/*, ene. 06, 2023. https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/ (accedido ene. 14, 2023).

[21] Microsoft, «Documentación de la API web de ASP.NET Core con Swagger/OpenAPI», *https://learn.microsoft.com/es-es/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-6.0*, nov. 28, 2022. https://learn.microsoft.com/es-es/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-6.0 (accedido ene. 14, 2023).

[22] Microsoft, «Códigos de error de REST del Centro de partners», *https://learn.microsoft.com/es-es/partner-center/developer/error-codes*, dic. 31, 2022. https://learn.microsoft.com/es-es/partner-center/developer/error-codes (accedido ene. 14, 2023).

[23] Balram Chavan, «Using async-await feature in Angular», *https://balramchavan.medium.com/using-async-await-feature-in-angular-587dd56fdc77*, abr. 18, 2018. https://balramchavan.medium.com/using-async-await-feature-in-angular-587dd56fdc77 (accedido ene. 14, 2023).

[24] Luis Miguel López Magaña, «Qué es Json Web Token y cómo funciona», *https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/*, ene. 17, 2020. https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/ (accedido ene. 14, 2023).

[25] Microsoft, «Asp .NET 6», *https://learn.microsoft.com/es-es/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-7.0*, nov. 28, 2022. https://learn.microsoft.com/es-es/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-7.0 (accedido ene. 14, 2023).

[26] https://www.chakray.com/, «SWAGGER Y SWAGGER UI», *https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/*. https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/ (accedido ene. 14, 2023).

[27] Microsoft, «Inyección de depedencias en .NET Core», *https://learn.microsoft.com/es-es/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0*, nov. 28, 2022. https://learn.microsoft.com/es-es/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0 (accedido ene. 14, 2023).

[28] Microsoft, «Logging in .NET Core and ASP.NET Core», *https://learn.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-6.0*, sep. 21, 2022. https://learn.microsoft.com/en-us/aspnet/core/fundamentals/logging/?view=aspnetcore-6.0 (accedido ene. 14, 2023).

[29] Albert Capdevila, «EL PATRÓN FACTORÍA SIMPLE EN C# DESDE UN PUNTO DE VISTA DE MANTENIBILIDAD», *https://albertcapdevila.net/patron-diseno-factoria-simple-csharp/*, 2018. https://albertcapdevila.net/patron-diseno-factoria-simple-csharp/ (accedido ene. 14, 2023).

[30] «Paquete DotNetCore.Security», *https://github.com/rafaelfgx/DotNetCore/tree/main/source/Security*. https://github.com/rafaelfgx/DotNetCore/tree/main/source/Security (accedido ene. 14, 2023).

[31] «Paquete DotNetCore.Domain», *https://github.com/rafaelfgx/DotNetCore/tree/main/source/Domain*. https://github.com/rafaelfgx/DotNetCore/tree/main/source/Domain (accedido ene. 14, 2023).

[32] https://desarrolloweb.com/, «Qué es MVC», *https://desarrolloweb.com/articulos/que-es-mvc.html*, jul. 28, 2020. https://desarrolloweb.com/articulos/que-es-mvc.html (accedido ene. 14, 2023).

[33] Angular, «Lazy loading in Angular 14», *https://angular.io/guide/lazy-loading-ngmodules*. https://angular.io/guide/lazy-loading-ngmodules (accedido ene. 14, 2023).

[34] «nginxproxymanager.com», *https://nginxproxymanager.com/*. https://nginxproxymanager.com/ (accedido ene. 14, 2023).

[35] lordpedal, «NPM en Docker», *https://lordpedal.github.io/gnu/linux/docker/npm-docker/*, feb. 12, 2022. https://lordpedal.github.io/gnu/linux/docker/npm-docker/ (accedido ene. 14, 2023).

[36] «Cómo funciona Let's Encrypt», *https://letsencrypt.org/es/how-it-works/*, oct. 18, 2019. https://letsencrypt.org/es/how-it-works/ (accedido ene. 14, 2023).

[37] Euda, «Certificación de dominio con Let's Encrypt, NGINX y Docker Compose», *https://dev.to/euda_ar/certificacion-de-dominio-con-let-s-encrypt-nginx-y-docker-compose-16p4*, sep. 03, 2020. https://dev.to/euda_ar/certificacion-de-dominio-con-let-s-encrypt-nginx-y-docker-compose-16p4 (accedido ene. 14, 2023).

[38] https://docs.docker.com/, «Try Docker Compose», *https://docs.docker.com/compose/gettingstarted/*, 2020. https://docs.docker.com/compose/gettingstarted/ (accedido ene. 14, 2023).

[39] R. Carrasco, «¿CÓMO ES LA ESTRUCTURA DEL FICHERO COMPOSE DE DOCKER? DOCKER-COMPOSE Y DOCKER STACKS», *https://www.ramoncarrasco.es/es/content/es/kb/151/como-es-la-estructura-del-fichero-compose-de-docker-docker-compose-y-docker*. https://www.ramoncarrasco.es/es/content/es/kb/151/como-es-la-estructura-del-fichero-compose-de-docker-docker-compose-y-docker (accedido ene. 14, 2023).

[40] «Uso de Docker Compose para implementar varios contenedores», *https://learn.microsoft.com/es-es/azure/cognitive-services/containers/docker-compose-recipe*, nov. 29, 2022. https://learn.microsoft.com/es-es/azure/cognitive-services/containers/docker-compose-recipe (accedido ene. 14, 2023).

[41] Sentrio, «Introducción a Jenkins: ¿qué es, para qué sirve y cómo funciona?», *https://sentrio.io/blog/que-es-jenkins/*, sep. 16, 2021.

[42] M. Reigen, «Integrate Jenkins builds into GitHub Pull Requests», *https://mreigen.medium.com/integrate-jenkins-builds-into-github-pull-requests-33bc053d6210*, dic. 30, 2017.