App Multiplataforma para la Digitalización y Catálogo de Piezas de Museos

Rafael Espejo Angulo

Resumen—En este proyecto se presenta el diseño y desarrollo de una aplicación móvil con React Native para Android cuya finalidad será poder digitalizar las diferentes piezas del catálogo de un museo. La aplicación no solo servirá para facilitar el trabajo de los trabajadores de un museo, sino que también permitirá al museo administrar los diferentes recursos registrados desde la misma aplicación.

Palabras clave— React Native, Android, NodeJS, ExpressJS, MySQL, Desarrollo de Aplicaciones Móviles, Aplicación Multiplataforma, RETOLD Project, Digitalización de Catálogo, Manejo de Piezas de Museo, Gestión de Recursos de Museos.

Abstract—In this project, the design and development of a mobile application using React Native for Android is presented. The purpose of this application is to digitize the various pieces in a museum's catalog. The application will not only facilitate the work of the museum staff but also allow the museum to manage the different registered resources from within the application.

Index Terms— React Native, Android, NodeJS, ExpressJS, MySQL, Mobile Application Development, Cross-Platform Application, RETOLD Project, Catalog Digitalization, Museum Piece Management, Museum Resource Management.

1 Introducción

Este documento tiene el objetivo de mostrar el diseño e implementación de una aplicación, usando el framework React Native, para móviles Android el cual se encargará de proporcionar una herramienta de fácil uso para digitalizar piezas del catálogo de un museo. Para dar apoyo a la aplicación, se ha diseñado una API y una base de datos, los cuales servirán para gestionar y almacenar dichos materiales digitalmente.

Este documento, también cuenta porque surge la idea de diseñar dicha aplicación y que beneficios aportará a los museos, una vez finalizado. También incluye, una explicación de las diferentes tecnologías que lo componen, qué metodología se ha aplicado para su buen desarrollo, cómo se ha planificado las diferentes etapas para su implementación, las herramientas que se han usado y los problemas que han surgido durante la configuración y su implementación. Para finalizar se presentarán las conclusiones a las que se ha llegado al realizar el proyecto seguido por la bibliografía utilizada para llevarlo a cabo.

2 Contextualización

La idea de desarrollar este proyecto surge de RETOLD [1], un proyecto financiado por el Programa Europa Creativa [2] el cual está destinado a impulsar y fortalecer los sectores culturales y creativos de los países miembros de la Unión Europea. RETOLD hizo un estudio sobre como centralizar los materiales de todos los museos afiliados al programa, para así poder compartir dichos recursos y poder llevar la cultura al público. Para ello y usando el gran apogeo de los Smartphones, se planteó un proyecto para que el público que visite un museo pueda usar su propio Smartphone, en vez de usar las audioguías que se ofrecen y así poder visualizar las distintas piezas que expone el museo.

En este contexto entra el proyecto que se describe en este documento. Como se ha explicado, el programa RETOLD tiene el objetivo, crear una aplicación para poder visualizar las piezas que expone, pero para ello se necesita una aplicación que los trabajadores del museo puedan usar para digitalizar dichos materiales.

Dicha aplicación debe tener todo lo necesario para poder realizar esa tarea. Además, no debe ser complicada de usar sino práctica, por ello su interfaz debe tener una buena usabilidad.

3 Objetivos

El objetivo principal del proyecto es crear una aplicación para móviles Android que sirva para digitalizar las diferentes piezas de un museo. La aplicación dispondrá de una interfaz que será capaz de obtener fotos y videos a través de la cámara del móvil para posteriormente subirlos al servidor. No solo podrá introducir estos datos, sino que también dispondrá de una plantilla tipo formulario, donde el usuario podrá introducir diferentes datos referentes al objeto que está documentando. A parte, no solo

- *E-mail de contacto: rafael.e.a.developer@gmail.com*
- *Mención realizada: Tecnologías de la Información*
- *Trabajo tutorizado por: Hing Fai Kevin Chow (departamento de Ingeniería de la Información y de las Comunicaciones)*
- *Curso 2023/24*

se podrán crear materiales digitales, sino que dichos materiales pueden ser usados para crear distintos eventos, los cuales podrían ser usados por otra aplicación para poder visualizar las distintas piezas del catálogo.

El segundo objetivo es que la aplicación pueda guardar localmente en el dispositivo móvil, los formularios y los archivos multimedia de los materiales o eventos que se creen. Esta funcionalidad se ha creado para solventar el problema de tener una mala conectividad a la red. Al guardarlos en el dispositivo, el usuario puede subirlos al servidor en otro momento sin perder la información introducida.

El tercer objetivo es que la aplicación disponga de una buena usabilidad, en otras palabras, que sea fácil de usar para el usuario y no tenga una interfaz compleja y difícil de manejar.

Como objetivo adicional, la aplicación podrá asumir dos roles de usuario dependiendo de las credenciales introducidas. Esos roles serán el de usuario y el de administrador. La funcionalidad del rol usuario será que podrá usar la aplicación para crear, ver, borrar y enviar materiales o eventos, también podrá crear grupos de trabajo, editarlos, añadir usuarios al grupo y eliminar grupos. Por otro lado, la funcionalidad del rol administrador solo podrá ver los usuarios del museo al que pertenece y añadir nuevos usuarios al museo.

Como objetivo extra, se ha creado una API para realizar todas las funcionalidades que tendrá la aplicación y una base de datos para poder guardar los datos que usará la aplicación.

4 Metodología y planificación

Para llevar a cabo el proyecto se ha utilizado una metodología en cascada. Para ello se han dividido las tareas en cuatro fases análisis, diseño, implementación y test.

Además, el proyecto se ha dividido en tres bloques, el servidor backend, la base de datos y la aplicación. Cada bloque utilizará independientemente las fases de la metodología y se aplicará el mismo concepto de que si no se acaba una fase no se puede empezar la siguiente.

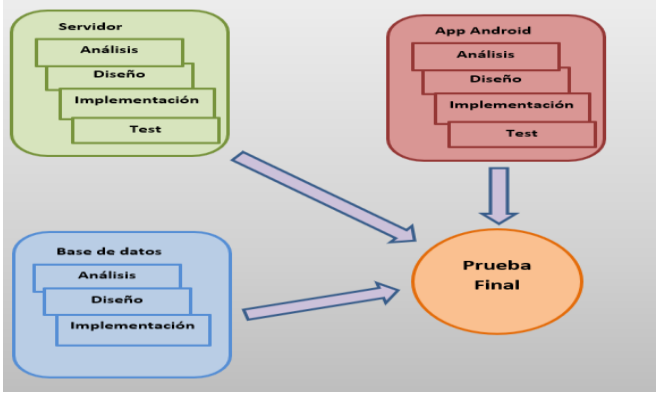

Fig. 1. Metodología para el desarrollo del proyecto.

La planificación se ha diseñado para ejecutarse en un tiempo definido. El proyecto empezará en la fase de análisis donde se buscará los requisitos funcionales y no funcionales de cada bloque y que tecnologías se pueden usar. Acabada esta fase, empezará la fase de diseño en la cual se harán distintos diagramas de los bloques implicados. En el caso de la aplicación se diseñarán las distintas pantallas de usuario.

Una vez acabada las distintas fases de diseño se comenzará con la fase de implementación de cada uno de los bloques del proyecto, los cuales serán programados en los distintos lenguajes definidos en la fase de diseño.

Al finalizar la fase de implementación del servidor y la aplicación se ejecutarán sus fases de test. La fase de test del servidor servirá para comprobar si las rutas de la API funcionan bien. La fase de test de la aplicación será una simulación para comprobar si la aplicación funciona en un dispositivo físico. En el apéndice A1 se puede visualizar la planificación de las distintas fases.

5 Análisis y tecnologías para el desarrollo del proyecto

5.1. Análisis de las tecnologías

5.1.1 Aplicación

Hoy en día existen muchas tecnologías para el desarrollo de aplicaciones móviles Android. Anteriormente solo había la opción de Android usando Java como principal lenguaje para codificar la aplicación, pero con el auge de los frameworks basados en JavaScript y TypeScript ya no es necesario aprender Java para crear una aplicación móvil. Ahora al desarrollar una aplicación se tiene que ver qué tecnología abarata tus costes de desarrollo y mantenimiento, o si quieres que tu aplicación sea multiplataforma o no. Haciendo un estudio del mercado actual, se presentan 3 tecnologías que hoy en día son las más usadas y que nos pueden ayudar a desarrollar una aplicación móvil, las cuales son Flutter con Dart, React Native y Kotlin [3] [4].

Flutter es un framework de código abierto creado por Google, para desarrollar aplicaciones móviles. Esta codificado en Dart por ello Flutter y Dart siempre se usan juntos para el desarrollo de aplicaciones. Flutter suele usarse más para el desarrollo de las interfaces de usuario. La ventaja de usarlo es que permite crear, probar y desplegar aplicaciones nativas de una forma fácil. A parte las aplicaciones desarrolladas pueden ser usadas en múltiples plataformas (móvil, web, escritorio), por esa razón en los últimos años es uno de los frameworks más usados para crear aplicaciones multiplataforma.

React Native es un framework creado por Facebook y que permite crear aplicaciones multiplataforma a través de la API de la plataforma nativa usando el lenguaje JavaScript. Su curva de aprendizaje es rápida si eres un desarrollador web familiarizado con JavaScript. Su venta-

ja principal es que permite crear interfaces de usuario combinando JavaScript con el lenguaje nativo de la plataforma, consiguiendo un rendimiento muy bueno en la creación de aplicaciones multiplataforma. Su desventaja es que no está completamente integrado con los sistemas nativos, por ello en determinadas ocasiones, se tiene de tocar código nativo.

Kotlin es el lenguaje de programación actual para desarrollar aplicaciones Android. Su lenguaje de programación es del tipo "tipado" estático, como muchos lenguajes modernos. Se ejecuta sobre la máquina virtual de Java por ello es muy compatible con dicho lenguaje. Solo es compatible con dispositivos Android, por ello no es ideal para el desarrollo de aplicaciones multiplataformas como iOS. La ventaja es que permite crear aplicaciones nativas de alta calidad y con un gran rendimiento de funcionalidad.

5.1.2 SERVIDOR BACKEND

Actualmente existen múltiples tecnologías para desarrollar un servidor backend. Haciendo un análisis de mercado, JavaScript es líder en la creación de servidores backend por su rápido desarrollo y puesta en marcha. No obstante, no es la única tecnología existente.

Entre las más usadas están Python, PHP y JAVA, cada una tienen sus ventajas y desventajas a la hora de implementar un servidor backend. [5]

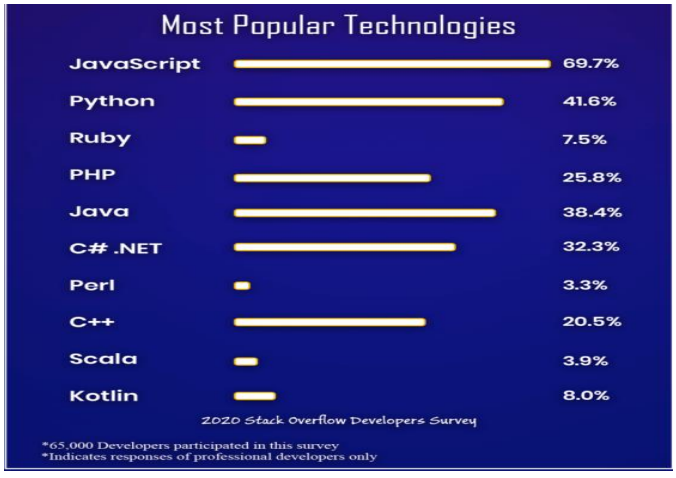

Fig. 2. Ranking de popularidad de tecnologías backend año 2020.

JavaScript, con la ayuda de Node.js, puede ejecutar operaciones para desarrollar un servidor backend. Su modulo Express.js permite tener listo un servidor backend en poco tiempo. Sus principales ventajas son su rápido desarrollo y la reducción de gastos, ya que utiliza secuencias de comandos de una manera muy eficiente y eso reduce los costes de cómputo. Tiene suficiente potencia para manejar miles de solicitudes de Entrada/Salida de los usuarios. Por otra parte, tiene sus limitaciones, la más problemática es entender la asincronicidad de las devoluciones de llamada. El problema es que Node.js se ejecuta en un solo hilo de ejecución. Al desarrollar funciones que pueden tardar tiempo en devolver una respuesta (llamadas a una API remota, lectura o escritura de disco), Node.js mete en una pila estas funciones y espera que se resuelvan. Esto conlleva que, si no tienes en cuenta esto, puedes devolver datos indefinidos ya que el hilo principal ejecuta todo el código sin pararse a esperar la respuesta de estas funciones. Por ello para programadores que empiezan a trabajar con este lenguaje puede resultar un dolor de cabeza saber por qué una función no devuelve nada.

Python es una tecnología backend muy demandada por desarrolladores ya que se apoya en la escritura de líneas de código racionales y explicitas. Sus principales ventajas son que es un lenguaje muy fácil de entenderlo y programarlo por su alta legibilidad, tiene una extensa biblioteca de librerías que dan apoyo a la comunidad a la hora de programar. Es de código abierto y gratuito, por ello los costes de desarrollo son reducidos.

Entre sus limitaciones es que las interrupciones en la programación pueden provocar que la ejecución sea lenta, por ello son necesarias pruebas adicionales para detectar errores o inconsistencias en el código. Es un lenguaje muy dependiente de bibliotecas de terceros para desarrollar algunos proyectos.

Java es una de las tecnologías más poderosas para el desarrollo de servidores backend. Sus principales ventajas es que su código es simple y altamente escalable, esto permite ejecutar numerosas instancias para las solicitudes del servidor. Es una tecnología multihilo que permite manejar las solicitudes en hilos independientes, por ello no hay grandes problemas en el control del retorno de la devolución de datos. Ofrece excelentes funciones para manejar los riesgos de seguridad. Su limitación más importante es el elevado coste de cómputo de hardware que genera y que su programación del lado del servidor consume más tiempo y memoria. Aparte no ofrece control sobre la recolección de basura y le falta soporte de programación de bajo nivel.

5.1.3 BASE DE DATOS

En la actualidad existen múltiples arquitecturas para crear una base de datos. Anteriormente las bases de datos relacionales eran las únicas que existían y todos los proyectos usaban este tipo de base de datos, pero con el paso del tiempo apareció un nuevo modelo de bases de datos, las no relacionales. Estas bases de datos guardan los datos sin una relación obligatoria, sino que como programador tú le das la relación. No hay un mecanismo que diga este dato está relacionado con aquel como en el caso de las relacionales. Hay muchas tecnologías dedicadas a las bases de datos y de libre distribución. Entre las relacionales las que más destacan son Oracle SQL, MySQL y PostgreSQL, entre las no relacionales están MongoDB, Elasticsearch, y Redis. No solo existen estas, sino que hay más, pero estas son las más usadas a fecha de hoy. [6]

Posición	Cambio	Base de Datos	Puntaje	Cambio (Puntaje)
1		Oracle	1256.83	-10.05
2		MySQL	1214.68	+8.63
3		Microsoft SQL Server	949.05	-4.24
4		PostgreSQL	609.38	+2.83
5		MongoDB	488.64	+0.07
6	^	Redis	175.80	-2.18
7	v	IBM Db2	162.88	-1.32
8		Elasticsearch	162.29	+1.54
9	^	Microsoft Access	131.26	+2.31
10	v	SQLite	128.37	+0.94

Fig. 3. Ranking de popularidad de bases de datos año 2022. [7]

5.2. Elección de las tecnologías usadas en los distintos bloques del proyecto

Sobre la base de datos, se compararon los distintos modelos actuales, relacionales y no relacionales, y se llegó a la conclusión que el proyecto utilizará datos que necesitan recurrir a complejas consultas y tener una estructura fija para extraer los datos. Por ello, las bases de datos relacionales son la mejor elección para este proyecto.

El lenguaje para desarrollarlo será MySQL, ya que es un lenguaje maduro y que permite extraer los datos relacionados de distintas tablas en una sola consulta.

En el caso del servidor backend, se llegó a la conclusión que para este proyecto Node.js con la librería Express.js [9] es una tecnología suficiente para desarrollar la API que usará la aplicación. Los factores que han llevado a esta resolución son que el módulo de Express.js tiene mecanismos suficientes para programar los distintos modelos y controladores para solicitar la información a la base de datos. Otro factor es que la curva de aprendizaje del lenguaje es relativamente sencilla ya que utiliza JavaScript, lo único que se tiene de preocupar el programador es de la asincronicidad de dicho lenguaje que puede ser difícil de aprender.

La aplicación tenía como requerimiento ser implementada en React Native, no se ha podido optar por otras tecnologías ya que es el objetivo principal del proyecto.

6 Herramientas del entorno de desarrollo

Para desarrollar el proyecto se han usado una serie de herramientas para poder trabajar. Para la codificación se ha usado un IDE de código gratuito muy usado en los últimos años el cual es Visual Code. Con el IDE solo se ha usado la extensión de MySQL para poder ver la base de datos desde una forma gráfica. Para la codificación del backend se ha usado la versión 20 de Node.js siendo la última de este lenguaje. Para la compilación de la aplicación se ha usado el SDK de Android en su versión 34 junto con la versión 0.73.4 de React Native usando también la versión 20 de Node.js. El montaje del servidor y la base de datos se ha hecho en una máquina virtual con Ubuntu. Para los diagramas se ha usado Dia, una herramienta que permite diseñar diagramas de bases de datos entre otros. Y como último, para diseñar el mockup de las pantallas de la aplicación se ha usado Marvel App [8].

7 Análisis

El análisis de la base de datos se basó en buscar que datos necesita un museo guardar. Para ello se hizo una búsqueda de qué tipo de campos son necesarios para documentar un material o un evento. También se analizó qué tipo de usuarios iban a usar la aplicación y qué debe guardarse sobre ellos.

La base de datos no está diseñada para un solo museo, por tanto, también se buscó qué información relacionada con el museo, como organización, debería guardarse.

Otras de las cosas que surgió mientras se analizaba la base de datos es si los usuarios trabajarán en grupo o en solitario, se llegó a la conclusión que era mejor que los usuarios trabajasen en grupo. Por ello se buscó como gestionar estos grupos y que datos deberían guardarse.

El servidor backend fue analizado para trabajar como un modelo-datos-controlador (haciendo referencia al modelo MVC [9] usado en el diseño de aplicaciones web).

También se analizó como enviar y guardar los archivos multimedia en disco. Para ello todos los archivos se envían al servidor en base64 y se guardan en disco en ese formato.

Cuando la aplicación pide un archivo multimedia, estos se envían de dos formas. Para las imágenes se envían junto con datos del formulario en base64 en una propiedad de la trama resultante al pedir los datos de un material. Los videos por su parte son transmitidos en formato streaming ya que el componente encargado de reproducir los videos en la aplicación solo acepta URL o enlaces locales para visualizar los videos, por ello solo los videos tienen una ruta especial para poder ser reproducidos.

Otra cosa que se buscó es una forma de que las rutas del servidor fuesen accedidas solamente si tienes autorización por token. Estos tokens se agregan en el campo "Authorization" del header de la trama transmitida por la aplicación.

Por último, se analizó qué debería tener la aplicación para poder documentar un material. Lo primero que tiene de tener la aplicación es una forma de acceder a la cámara del dispositivo para capturar el material. Además, tiene de tener un formulario para poder documentar dicho material.

Otra de las cosas que se definió es que los usuarios no son los dueños de los materiales, sino que es el grupo de trabajo quien registra el material documentado. Por ello la pantalla principal de la aplicación debe mostrar los grupos donde el usuario esta asignado y al clicar en un grupo podrá ver y/o crear un material o un evento.

Los usuarios de los grupos tienen permisos, para ello se definieron tres permisos r (lectura), w (escritura) y a (administrador), solo los usuarios con el permiso "a" pueden

eliminar los materiales del grupo, editar el nombre y la descripción del grupo, y añadir nuevos usuarios al grupo.

8 Diseño

Al analizar los requisitos que necesitaría la base de datos, se obtuvieron 10 tablas las cuales definen los datos con los que trabajará la aplicación. En el apéndice A2 se puede visualizar el diagrama relacional.

Sobre el servidor backend se ha diseñado un diagrama el cual se ve cómo funciona el servidor. Dicho servidor tendrá unas rutas las cuales atenderán las peticiones de la aplicación. Cada ruta de la API llamará a un controlador específico el cual usará la función designada del modelo para hacer la operación requerida.

También se ha diseñado dos servicios el de disco y el de autorización. El servicio de disco servirá para poder enviar o guardar los archivos multimedia en el disco físico. El servicio de autentificación servirá para dar autorización a la petición si el token es válido. En el apéndice A3 se puede visualizar el diagrama del servidor.

Por otra parte, el servidor se ha diseñado para que se puedan controlar los distintos errores de código para que el servidor no se apague inexpertamente dejando sin servicio a los usuarios.

Por último y no menos importante, el servidor está diseñado para poder trabajar con conexiones seguras utilizando el protocolo https.

La aplicación se dividió en dos partes de diseño, la lógica y la interfaz de usuario. Las diferentes pantallas de la interfaz gráfica se diseñaron usando el aplicativo Marvel App. En el apéndice A4 se pueden ver el mockup de las pantallas de la aplicación.

Cada pantalla es un componente el cual tiene su lógica asociada dentro de un solo archivo. Un componente de la aplicación tiene una parte donde se definen las funciones necesarias para su funcionamiento y la plantilla de renderizado para que dicha pantalla se pueda visualizar e interactuar.

La estructura de carpetas de la aplicación está dividida entre los componentes, recursos y API. La carpeta de componentes se encontrarán todos los archivos de las interfaces graficas divididos entre las interfaces del rol usuario y las del rol administrador. La carpeta recursos contendrá los archivos con funciones que serán compartidas por los componentes como pueden ser las funciones de control de la sesión, las funciones de guardado de materiales y eventos de forma local y las interfaces de tipado. La carpeta API contiene la función configurada para realizar la llamada a la API, la definición de todas las rutas de la API y su método de llamada GET, POST, PUT o DELETE, junto con la URL base de la API.

9 Implementación

9.1 Servidor Backend (API)

El servidor ha sido codificado usando la librería Express.js [10]. Se ha usado un modelo modelo-datos-controlador (parecido al modelo-vista-controlador de las aplicaciones web) para codificar el funcionamiento de este. Para este modelo se han creado diferentes controladores los cuales llaman a una función del modelo para realizar consultas a la base de datos.

Cada controlador está construido como una clase que extiende de una clase abstracta llamada Controller que contiene el método "runImplements". Este método tendrá que ser implementado por cada controlador de la API y ahí es donde se codificarán las funciones que desempeñará el controlador.

Otro método que tiene la clase "Controller" es "run" que es el encargado de llamar al método "runImplements", y a parte se encarga de llamar al servicio de autentificación de token. Toda la clase "Controller", esta codificada en un try-catch para controlar los errores que se puedan producir. Otra cosa que tiene "Controller" es que, a su vez, extiende de una clase "EResponse" en la cual se han implementado diferentes funciones para la respuesta que dará el controlador al cliente que hizo la petición.

También se han creado los servicios "SAuthorization" y "SDisk". El servicio "SAuthorization" es el encargado de verificar si el token pasado en el campo "Authorization" del header es válido o no. Esta validación se hace únicamente buscando si el token está asignado a un usuario, si existe dentro de la base de datos, se le considera válido. El servicio "SDisk" sirve para guardar los archivos multimedia en el disco físico y leerlos cuando se pidan (los videos no usan este servicio para ser enviados, sino que usan un controlador llamado "RCVideo" que será el encargado de enviar el video en formato streaming).

El servidor se ha codificado para poder trabajar con peticiones https. Para ello se tiene de disponer de un certificado válido para que no se rechacen las peticiones.

Por último, se implementó un sistema de logger el cual registra todas peticiones que se le hacen a la API. En él se registran posibles errores o información de las rutas que se solicitan en tiempo de ejecución a la API. Todo se guarda en un archivo log con la fecha y hora del día. Esto puede servir a los administradores o programadores para encontrar problemas que se puedan producir en el servidor durante su funcionamiento. Para implementar el logger se utilizó la librería Winston [11].

9.2 Aplicación

La aplicación ha sido desarrollada con el framework de React Native [12]. La gran mayoría de los elementos visuales que conforman las distintas pantallas son componentes nativos del framework. No obstante, el framework en si no provee de todo lo que necesitas, así que se ha

utilizado la librería react-native-paper [13] para poder formar algunos de los elementos de las pantallas.

9.2.1 Navegación

Para la navegación entre las distintas pantallas se ha utilizado las librerías @react-navigation/native y @react-navigation/stack [14][15]. Primero es crear un componente NavigationContainer en el archivo App de la raíz del proyecto. Este componente es donde se montará y visualizará las diferentes pantallas. Después se crea un Stack.Navigator usando la función createStackNavigator de la librería. Al crear un componente Stack.Navigator, debes crear un componente Stack.Screen que servirá para llamar a la pantalla que necesites para navegar por la aplicación.

Este Stack.Screen tiene unas propiedades, llamadas name y component, su inicialización es obligatoria. La propiedad name es el nombre de la pantalla y servirá para que el Navigator sepa que componente cargar. La propiedad component sirve para definir el componente que quieres que se cargue. Si queremos definir otra pantalla crearemos otro Stack.Screen.

Todos estos Stack.Screen están definidos en el archivo Main de la aplicación.

Para poder navegar entre las distintas pantallas definidas, usaremos la función navigate(nombre_pantalla, datos). Su primer parámetro sirve para navegar a la pantalla deseada. Debe quedar claro que ese nombre corresponde al name especificado en algún Stack.Screen definido en el archivo Main. El segundo parámetro sirve para pasar datos entre componentes. Para que esto funcione, debes crear tus componentes con dos parámetros en la función principal de cada archivo, si no lo haces no tendrás acceso a las funciones del navigate ni a los datos que te pueda enviar otro componente.

9.2.3 Sesión

Para la creación de la sesión de usuario, se ha usado la librería react-native-storage [16]. Las funciones usadas de esta librería son load, save y removeItems. Su finalidad ha servido para guardar los datos de sesión del usuario como el token, el id_usuario y el id_organizacion para así no tener que loguearse siempre que se inicie la aplicación. Al iniciar la aplicación, siempre comprobará si hay una sesión, si la hay saltará a la pantalla de grupos, no obstante, sino hay la hay, el usuario se tendrá que loguear.

9.2.4 Captura de video e imágenes

Para la captura de archivos multimedia se ha usado la librería react-native-image-picker [17]. Sirve para poder usar la cámara del dispositivo, y así poder obtener fotos o videos. Cuando se captura una foto o video, esta librería devuelve la ruta local donde está el archivo multimedia que se ha capturado. Después usando la librería react-native-fs [18], podemos cargar el archivo multimedia desde el dispositivo, y pasarlo a base64 para poder enviarlo al servidor.

10 Tests

10.1 Servidor backend (API)

Para los tests de la API se ha usado la librería Jest [19] para poder comprobar que el código funcionaba correctamente. Las pruebas realizadas han sido verificar si los diferentes controladores comprobaban si las variables requeridas para hacer las consultas existían y no iban vacíos. También se ha verificado que los controladores enviaban los campos requeridos como respuesta para la consulta realizada.

10.2 Aplicación

Las pruebas para ver que todo lo implementado en la aplicación funcionaba, ha sido probando la aplicación durante la fase de implementación utilizando un Samsung A52s con Android 13. Una vez implementado todo el código se probó la aplicación en modo producción sin la función de debugger que proporciona el framework de React Native. Estas pruebas se llevaron a cabo, probando todos los elementos que componen cada pantalla y si al pedir o enviar datos a la API, respondía como se esperaba. También se probó el funcionamiento sin Internet para comprobar que la aplicación no se quedaba bloqueada o con el spinner en estado de carga infinito. Todas las pruebas realizadas han servido para hallar posibles errores de comportamientos extraños cuando se interactuaba con las distintas pantallas.

11 Resultados finales

En este apartado se muestran las diferentes pantallas que componen la aplicación. Se ha elegido que la visualización de la aplicación sea con colores propios del modo oscuro. Considero que este tipo de tonalidad no perjudica a la vista como las tonalidades claras, en especial pantallas con fondo en blanco. Por eso mismo se ha elegido esta tonalidad para el diseño final de la aplicación.

11.1 Pantalla de Inicio de sesión

Esta pantalla es la primera que se carga al iniciar la aplicación por primera vez o si cierras sesión. En ella podemos encontrar dos campos para introducir el email del usuario y su contraseña. Después clicando en el botón “Log In” inicias sesión en la aplicación. El otro botón sirve para cargar la pantalla de creación de un museo (organización). En ella se requerirá que introduzcas el nombre que tendrá el usuario administrador, un email, una contraseña y el nombre que le quieres dar al museo.

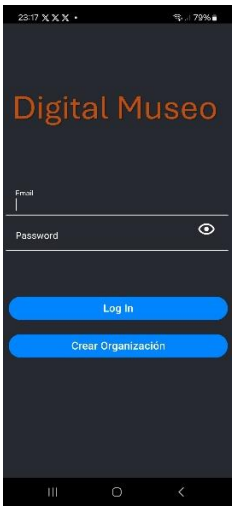

Fig. 4. Pantalla de inicio de sesión.

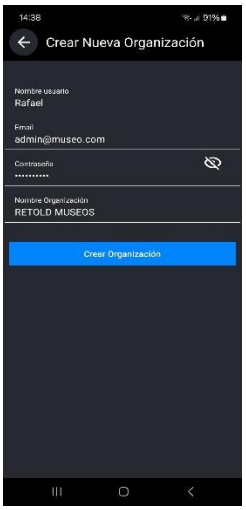

Fig. 5. Pantalla de creación de museo.

11.2 Pantalla de grupos de trabajo

Esta es la pantalla principal, una vez iniciada la sesión o si cierras la aplicación con la sesión guardada al iniciarla otra vez se volverá a cargar esta pantalla primero. En ella aparecerá una lista de los grupos de trabajo del usuario con su nombre, su descripción y su fecha de creación. Cada grupo tendrá dos botones que sirven para editar el grupo o eliminarlo. A parte, fijo en la parte de abajo estará el botón que sirve para crear nuevos grupos.

Fig. 6. Pantalla de grupos de trabajo.

Fig. 7. Pantalla de creación de grupos de trabajo.

11.3 Pantalla de materiales y eventos

Al clicar en un grupo de trabajo se cargará esta pantalla en la cual se podrán ver los materiales y eventos creados para ese grupo. La primera pantalla que se cargará por defecto es la de materiales, en ella podemos ver una lista de los diferentes materiales. Cada elemento mostrará el nombre, la descripción y la fecha en la que se creó el material. También dispone de un botón que sirve para eliminar el material y en la parte de abajo está el botón para poder añadir nuevos materiales. La otra parte de esta pantalla sirve para ver los eventos creados para ese grupo de trabajo. Para poder verlos, se tiene de hacer un movimiento con el dedo de derecha a izquierda en modo horizontal. Una vez hecho este movimiento, se puede ver la lista de eventos, los cuales tendrán su nombre, descripción, fecha de inicio y fecha de finalización del evento. Como en la pantalla de los materiales, cada elemento tiene su botón que sirve para eliminar el elemento. En la parte de abajo, estará el botón para poder añadir nuevos elementos.

Tanto en la pantalla de materiales como en la de eventos, todos lo elementos de la lista se pueden clicar. Si se clica en ellos se carga la pantalla de visualización de cada elemento en la cual se puede ver más información del elemento. Esta pantalla de visualización tiene en la parte de arriba a la derecha un botón que permite editar el elemento.

Fig. 8. Pantalla de materiales.

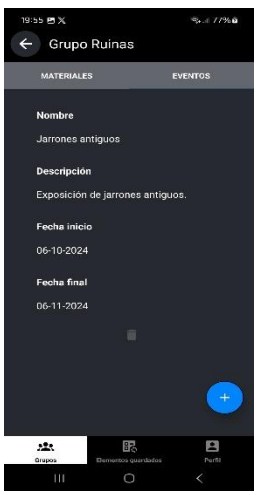

Fig. 9. Pantalla de eventos.

Fig. 10. Pantalla de creación materiales.

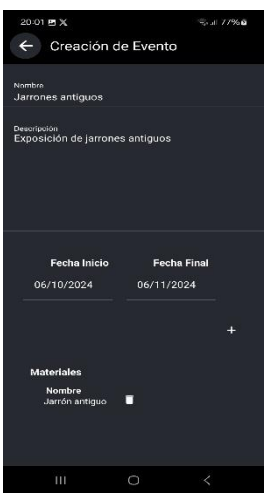

Fig. 11. Pantalla de creación de eventos.

Fig. 12. Pantalla de visualización de un material.

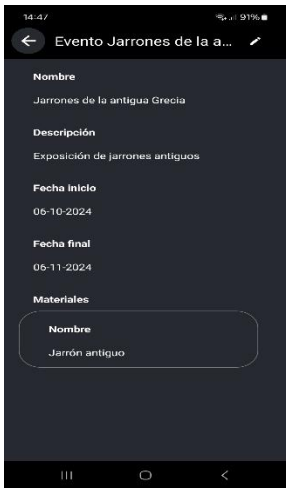

Fig. 13. Pantalla de visualización de un evento.

Fig. 15. Pantalla de elementos guardados localmente.

Fig. 14. Pantalla de edición de un material

11.4 Pantalla de elementos guardados

En esta pantalla se mostrarán, en forma de lista, los materiales o eventos guardados localmente en el dispositivo. Cada elemento mostrado, tendrá dos botones que sirven para enviar el elemento al servidor (si se envía, se elimina de la lista) y para eliminarlo localmente. Otra funcionalidad que tiene cada elemento es que al clicar en él puedes ver la información que se guardó del elemento. Para llegar a esta pantalla se tiene de clicar en el botón inferior de la pantalla que pone "Elementos guardados".

11.5 Pantalla de perfil

Esta pantalla muestra la información del perfil del usuario. En ella se puede ver el nombre del usuario, sus apellidos, su email, su cargo y su teléfono. En la parte inferior hay dos botones, uno sirve para poder editar los datos del perfil y el otro sirve para cerrar sesión del usuario (al cerrar sesión se borrará la sesión guardada en el dispositivo). Para llegar a esta pantalla, el usuario tiene de clicar en el botón inferior de la pantalla "Perfil".

Fig. 16. Pantalla de perfil del usuario.

11.6 Pantalla de administrador

A esta pantalla se accede con unas credenciales del rol de administrador. Una vez iniciada la sesión, la pantalla principal mostrará una lista de los usuarios que pertenecen al museo. A parte en la parte superior de la lista, hay un botón que sirve para ir a la pantalla donde se podrán añadir nuevos usuarios. La pantalla contiene los campos nombre, email y contraseña del usuario que servirán para añadirlo al sistema. Solo el usuario admi-

nistrador puede añadir usuarios, no existe ninguna otra forma de hacerlo.

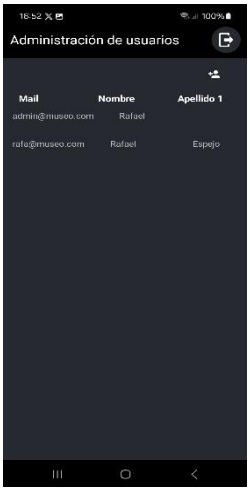

Fig. 17. Pantalla de administración.

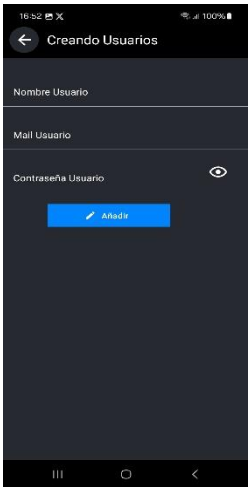

Fig. 18. Pantalla para añadir usuarios.

12 Conclusiones

El objetivo de este proyecto era poder digitalizar piezas de museos, usando un dispositivo móvil. Este objetivo se ha cumplido exitosamente y se ha podido crear una aplicación usando la tecnología React Native para ello. Lamentablemente no se ha podido conseguir del todo que la aplicación tenga un buen feedback con el usuario y no se ha conseguido desarrollar más funcionalidades para el rol de administrador. Por parte de la API se ha conseguido desarrollar un servidor de peticiones exitosamente. Concluyendo este proyecto me ha ayudado a ver las capacidades que tengo para desarrollar proyectos de este calibre y que necesito mejorar para mi desarrollo personal en el mundo laboral.

13 Trabajo futuro

Las futuras actualizaciones de este proyecto serán:

- Mejorar el feedback con el usuario.
- Adaptar la aplicación para que pueda usar diferentes idiomas para interactuar con el usuario.
- Integración del sistema de permisos de usuario sobre los grupos de trabajo.
- Desarrollar más funcionalidades para el rol de administrador.

Agradecimientos

Mis agradecimientos van dedicados a toda mi familia y mis amigos por todo el apoyo aportado para que pudiese desarrollar este proyecto.

Bibliografía

[1] C. Hansen, R. Kelm, "RETOLD: On the Way for a Digital Future of Documentation in Open-air Museums – User Requirements for Data Entry and a Management Product for the RETOLD-Project," EXARC Journal Issue 2022/3, September 2022. [Online]. Disponible: https://exarc.net/issue-2022-3/aoam/retold-user-requirements

[2] "Europa Creativa." Proyecto Cultural Europa Creativa, financiado por la Unión Europea. [Online]. Disponible: https://europacreativa.es/europa-creativa/

[3] F. Medina, "Mejores tecnologías para desarrollo de aplicaciones móviles 2022," ArmadilloAmarillo, Mayo 2022. [Online]. Disponible: https://www.armadilloamarillo.com/blog/mejores-tecnologias-para-desarrollo-de-aplicaciones-moviles-2022/

[4] L. Saez, "11 tecnologías para aprender los desarrolladores móviles en 2021," Linkedin, Abril 2021. [Online]. Disponible: https://es.linkedin.com/pulse/11-tecnolog%C3%ADas-para-aprender-los-desarrolladores-en-2021-saez-irurre

[5] "Las 10 mejores tecnologías de backend. "[Online]. Disponible: https://blog.back4app.com/es/las-10-mejores-tecnologias-de-backend/

[6] S. Martín, "Las mejores bases de datos del año 2017," PandoraFMS, actualizado Marzo 2022. [Online]. Disponible: https://pandorafms.com/blog/es/mejores-bases-de-datos/

[7] Fig. 3. "Las 10 Base de Datos más Populares," nubecolectiva, Marzo 2022. [Online]. Disponible: https://blog.nubecolectiva.com/las-10-bases-de-datos-mas-populares-febrero-2022/

[8] "MarvellApp", aplicación web para crear mockups para aplicaciones móviles. [Online]. Disponible: https://marvelapp.com/

[9] "Patrón de programación modelo-vista-controlador", Wikipedia. [Online]. Disponible: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

[10] "Express.JS", ExpressJS. [Online]. Disponible: https://expressjs.com

[11] "Winston", npm. [Online]. Disponible: https://www.npmjs.com/package/winston

[12] "React Native", React Native Getting Started. [Online]. Disponible: https://reactnative.dev/docs/0.73/getting-started

[13] "React Native Paper", Github. [Online]. Disponible: https://callstack.github.io/react-native-paper/docs/guides/getting-started

[14] "React Navigator", React Native Navigation. [Online]. Disponible: https://reactnative.dev/docs/navigation

[15] "Stack Navigation", React Navigation Stack. [Online]. Disponible: https://reactnavigation.org/docs/stack-navigator/

[16] "React Native Storage", npm. [Online]. Disponible: https://www.npmjs.com/package/react-native-storage

[17] "React Native Image Picker", Github. [Online]. Disponible: https://github.com/react-native-image-picker/react-native-image-picker

[18] "React Native Fs", npm. [Online]. Disponible: https://www.npmjs.com/package/react-native-fs

[19] "Jest", JestJs. [Online]. Disponible: https://jestjs.io/es-ES/docs/api

APÉNDICE

A1. PLANIFICACIÓN DEL PROYECTO

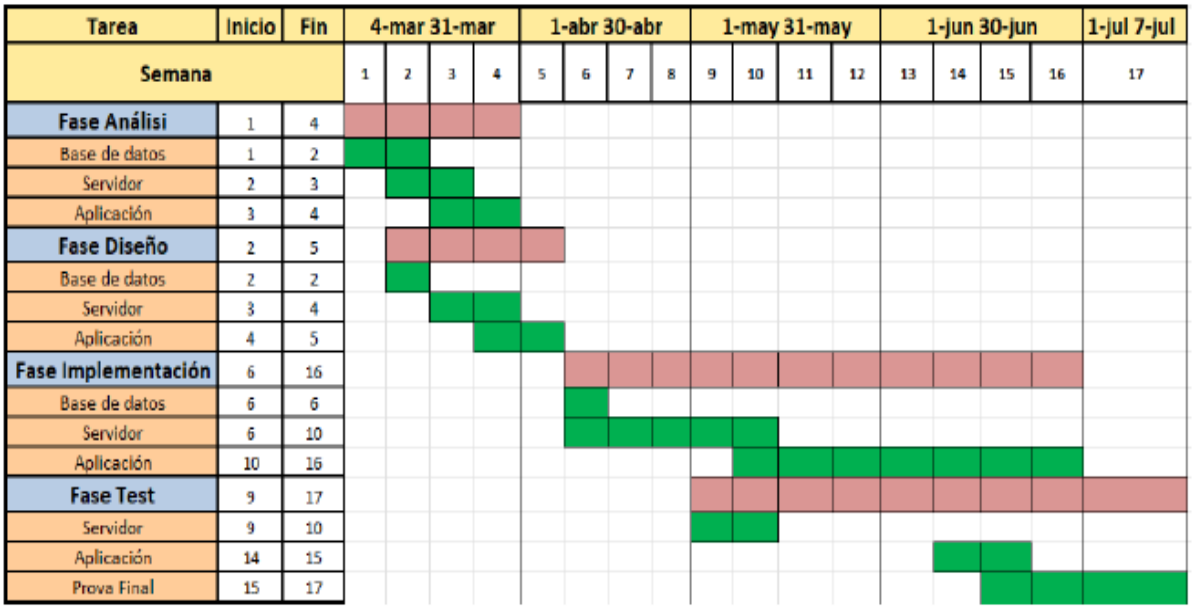

Fig. 19. Planificación del desarrollo del proyecto.

A2. DIAGRAMA DE LA BASE DE DATOS Y SUS PRINCIPALES TABLAS

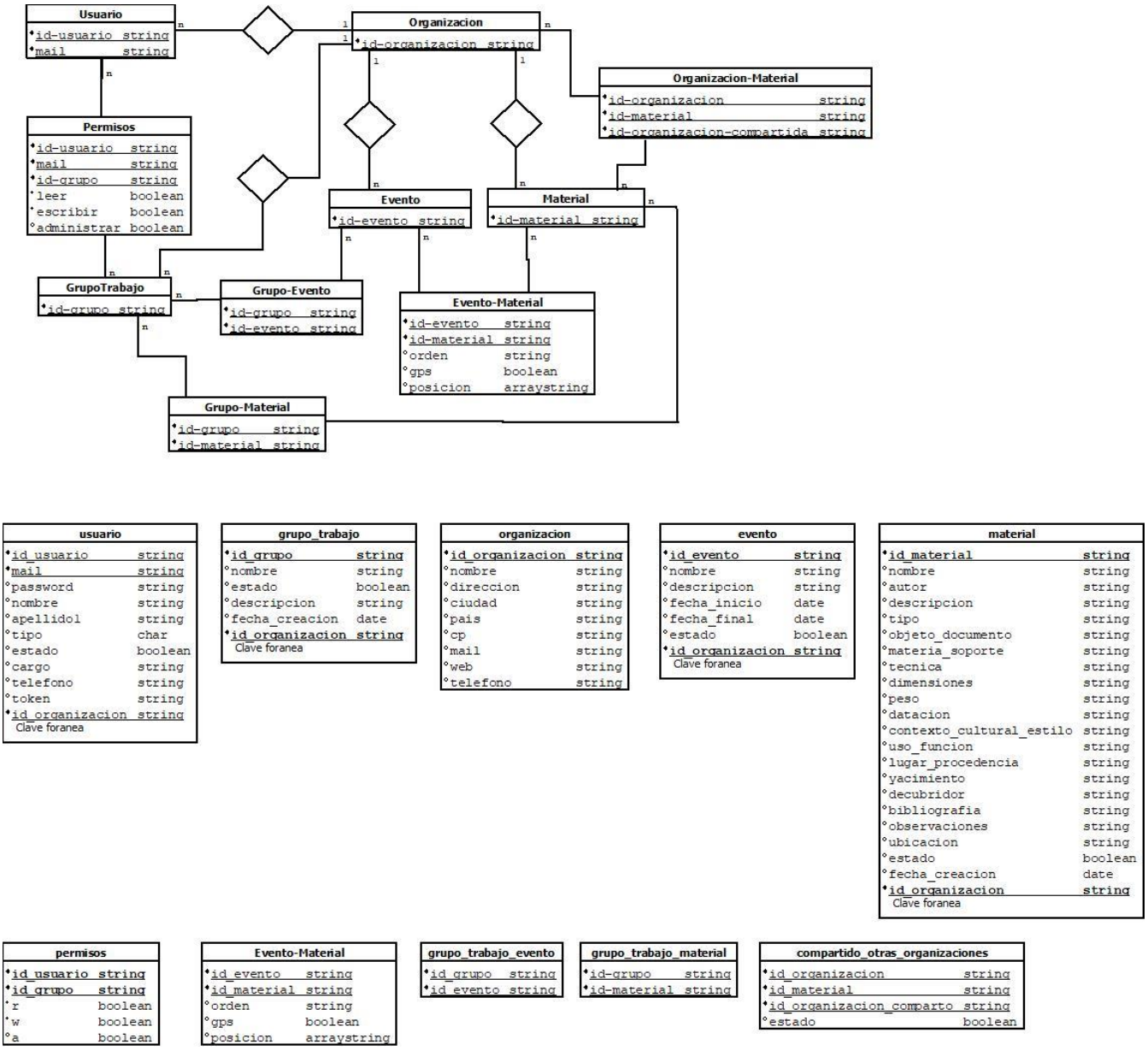

Fig. 20. Diagrama relacional de la base de datos.

A3. DIAGRAMA DEL SERVIDOR BACKEND

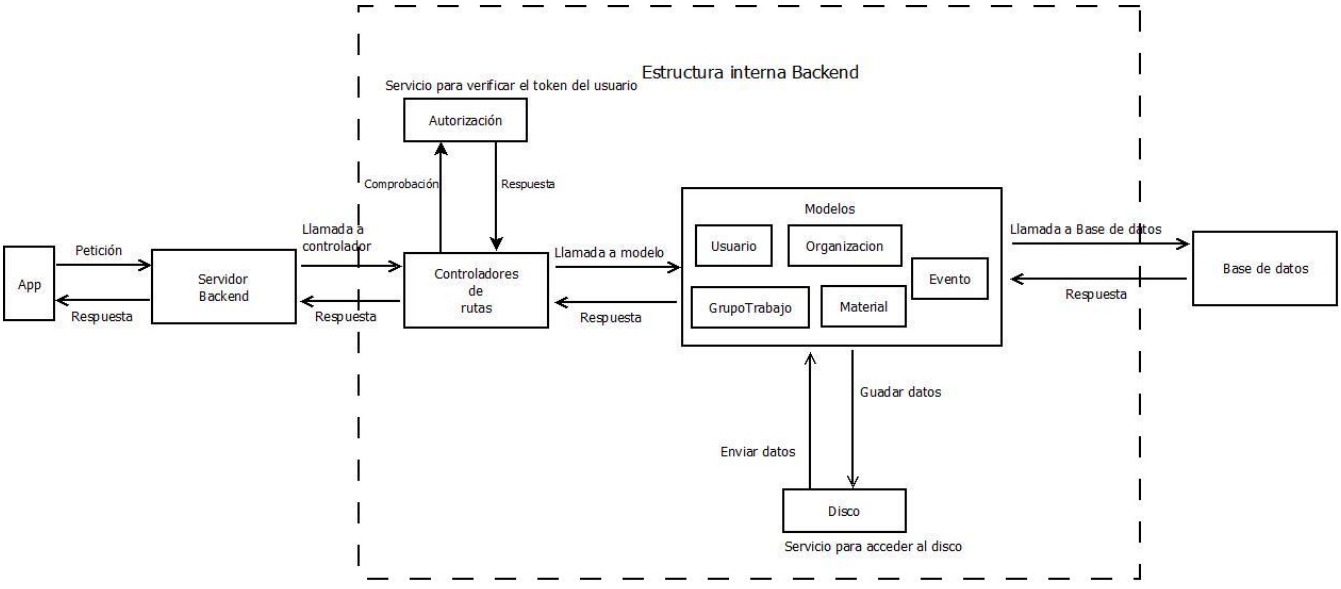

Fig. 21. Diagrama estructural del servidor backend.

A4. Pantallas de la aplicación

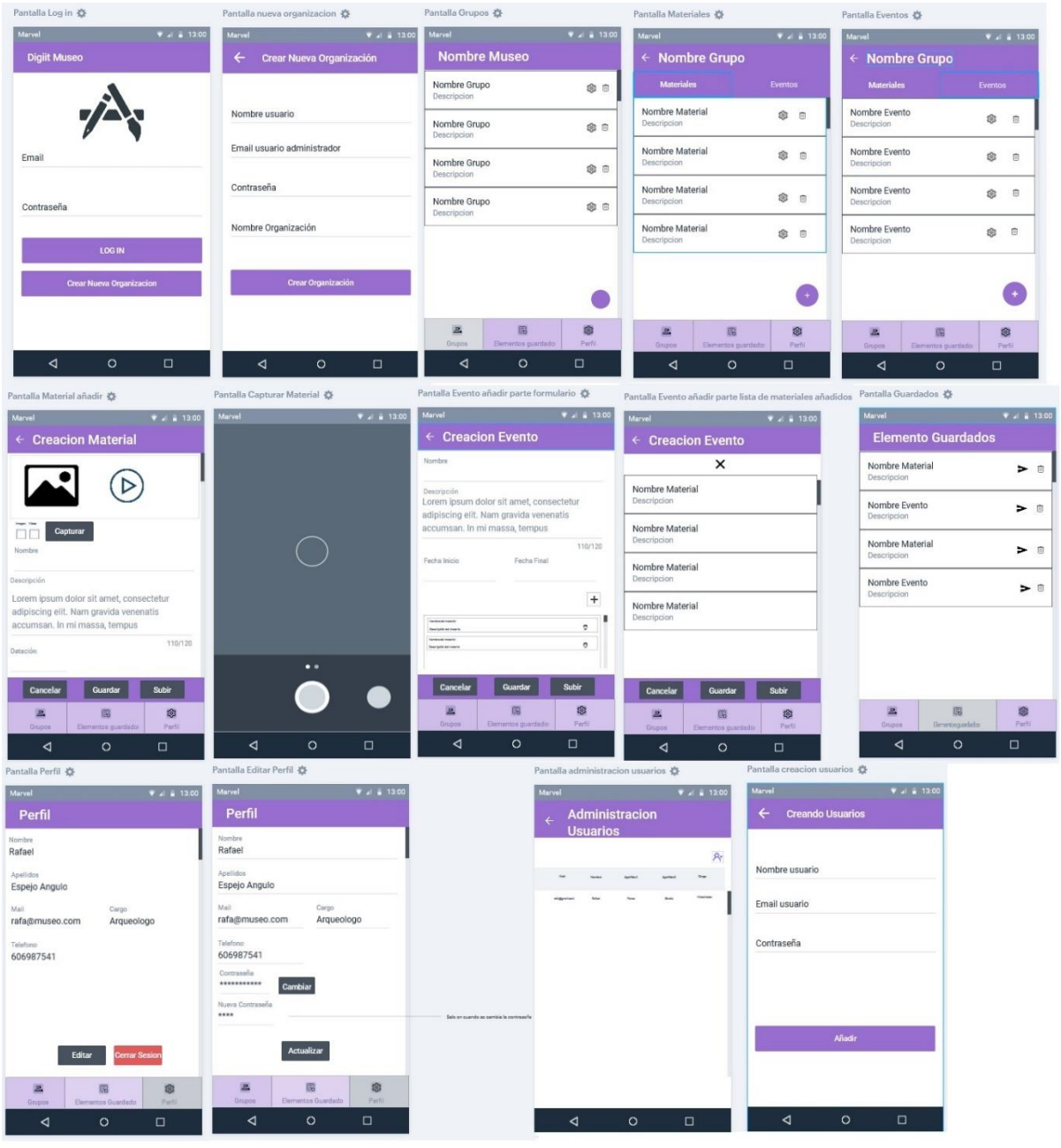

Fig. 22. Mockup de las pantallas de la aplicación.