

QR Access

FP EDIX

*Desarrollo de Aplicaciones
Multiplataforma*

Antonio Pastor Aranda

Ignacio Viseras Riego

Pablo López Jimenez

Raquel López

Tutora:

Índice

Introducción	4
Justificación.....	4
Objetivos generales.....	4
Agradecimientos	5
Resumen	6
Módulos formativos aplicados en el trabajo	7
Herramientas/Lenguajes utilizados	9
Componentes del equipo y aportación realizada por cada estudiante	11
Fases del proyecto	12
Modelo de datos utilizado	13
Diagrama de clases y casos de uso	14
Diseño del mockup	15
Explicaciones de la funcionalidad del proyecto	17
Servicio de API REST:.....	17
Aplicación Android:	18
Conclusiones y mejoras del proyecto	28
Bibliografía.....	31
Anexos.....	33

Introducción

Tras un intenso segundo trimestre, los exámenes habían terminado. Comenzamos a dar vueltas a una posible aplicación que aunara todos los conocimientos adquiridos y que además fuese apreciable. La tarea fue ardua porque había que encajar en tiempo y forma un proyecto que tuviéramos medianamente funcional en apenas tres meses.

Justificación

En el mercado actual encontramos múltiples formas de acceso. Podemos usar una huella dactilar, un NFC, reconocimiento facial o incluso códigos pictóricos... Esta última forma de acceso nos ocupará el Trabajo de Fin de Grado. En las empresas, en los trenes, en los gimnasios, en los estadios deportivos... En multitud de situaciones tenemos la necesidad de controlar los accesos. Es por ello que, como base para aplicar los conocimientos adquiridos durante todo el itinerario formativo de Desarrollo de Aplicaciones Multiplataforma, hemos decidido realizar un sistema completo para controlar accesos a eventos basándonos en códigos Quick Response o, más conocidos por sus siglas, QR.

Objetivos generales

1. Emplear los recursos aprendidos durante el módulo, demostrando capacidades de programación y diseño de aplicaciones, sistemas y servicios.
2. Crear un servicio de API REST consumible y una aplicación móvil funcional para mostrar el uso de la misma.
3. Coordinar de manera profesional y eficiente el equipo de trabajo, funcionando de manera autónoma pero en equipo.

Agradecimientos

Agradecer a todas las personas que en este año nos han ayudado en nuestro proceso de aprendizaje por hacernos crecer tanto profesional como personal. En especial a las profesoras de EDIX por su apoyo, dedicación y esfuerzo por transmitirnos los conocimientos necesarios para afrontar este reto.

Durante tres meses hemos aumentado nuestras horas de trabajo perdiendo tiempo de vida social y familiar. Pese a todo hoy estamos más cerca y podemos decir que sin la paciencia y el apoyo incondicional de nuestras familias, amigas y amigos no habría sido posible.

Por último, agradecemos mutuamente el buen talante gracias al cuál hemos superado todos los momentos de tensión que han surgido.

Resumen

En nuestra búsqueda de poner en práctica los conocimientos aprendidos, encontramos una posibilidad que atrajo nuestra atención. Esta opción consistía en la gestión de accesos mediante códigos QR.

En nuestro objetivo de mejorar sobre los conocimientos aprendidos hemos intentado implementar una aplicación completamente multiplataforma basada en una API REST consumible por cualquier plataforma mediante HTTP.(1,2)

Esta aplicación está destinada al registro de nuevos eventos y la obtención y/o validación de accesos para dichos eventos.

La aplicación cuenta con dos roles:

- Administrador: puede crear/borrar eventos y validar accesos
- Cliente: puede comprar accesos.

El sistema te permitirá registrarte como administrador o cliente.

Módulos formativos aplicados en el trabajo

Las asignaturas utilizadas para la realización de este proyecto son:

- Acceso a datos:

Para hacer una aplicación más real hemos utilizado dos bases de datos, una relacional en MySQL. Para consumir esta base de datos hemos utilizado JDBC en el servicio java que monta la API. (3,4)

Al dudar en nuestros conocimientos para desplegar el sistema decidimos seguir un desarrollo en paralelo reinventando el producto con una base de datos NoSQL (Firebase).

También hemos usado la base de datos SQLite que genera internamente Android para asegurarnos la persistencia de los datos. Para esto hemos usado la librería ROOM que emplea un modelo de ORM similar a JPA.(5)

- Programación Multimedia y dispositivos web:

Esta asignatura ha sido fundamental para el desarrollo de la aplicación en Android.

- Desarrollo de interfaces:

Para el desarrollo de una buena aplicación ha sido fundamental tener en cuenta el temario de esta asignatura. Un buen desarrollo de la UI/X. Nos queda pendiente el desarrollo de un servicio web que implemente la API. Formaba parte de los planes pero por falta de tiempo no hemos podido implementarlo. (6)

- Programación de servicios y procesos:

El contenido de esta asignatura ha sido crucial para la creación del servicio REST. Nos hemos apoyado en PSP también para la encriptación de las contraseñas de usuario, así como para una pequeña codificación que usamos a la hora de generar el QR en Android. Aunque los hilos se han quedado un poco al margen sí hay una pequeña aplicación de hilos para dormir el proceso durante X segundos en la aplicación de Android.

- DevOps:
Hemos hecho uso de una plataforma en la nube (AWS) para desplegar el proyecto. También hemos planteado implementar integración continua para poder integrar GitHub con AWS pero por falta de tiempo y conocimientos no ha sido viable. (7)
- Sistemas de Gestión Empresarial:
Hemos creado una aplicación basada en API REST con idea de que cualquier Sistema de Gestión empresarial pueda consumirla. Aunque es algo que queda pendiente, es posible crear, por ejemplo, un módulo de Odoo que mediante peticiones HTTP pueda acceder a todas las opciones que da la API. No llegamos a realizarlo, pero comentamos la posibilidad de crear un módulo de Odoo para permitir a un administrador gestionar los eventos y los usuarios clientes.(2,8)

Herramientas/Lenguajes utilizados

Los lenguajes utilizados este proyecto son:

- Java(9)
- SQL
- Xml

Las herramientas utilizadas para la creación del proyecto son:

- Android Studio
Software para el diseño de aplicaciones multiplataforma (10)
- Eclipse IDE
Aplicación para la gestión y creación de proyectos(11)
- Visual Studio Code
Editor de texto y entorno de desarrollo multilenguaje (12)
- Firebase
Plataforma integrada con Google Cloud Platform, ubicada en la nube que puede crear y sincronizar proyectos (13)
- Github
Software que ejecuta un control de versiones. (14)
- Amazon Web Service (AWS)
Es una plataforma de servicios que ofrece soluciones de computación y almacenamiento de datos en la nube. (7)
 - Relational Database Service (RDS)
Es un servicio para operar con bases de datos relacionales en la nube.
 - Elastic Beanstalk
Servicio de administración y despliegue de aplicaciones en la nube.
- XAMP
Es un paquete de software gratuito que proporciona una solución integrada que incluye un servidor web Apache, un gestor de base de datos MySQL y un intérprete de PHP. (15)
 - MySQL Workbench
Es un software para la gestión y administración de bases de datos MySQL (3)

Las herramientas utilizadas para la esquematización y documentación del proyecto son:

- Notions
Software que permite la gestión de documentos online y timeline de tareas. (16)
- PDF
Formato estándar de documentos portátiles. (17)
- Figma
Herramienta generadora de prototipos web. (18)
- Diagrams
Herramienta creadora de diagramas. (19)
- Adobe Color
Generador de colores para nuestra aplicación (20)
- InKscape
Creador de imágenes vectoriales (SVG). (21)
- ChatGPT
Prompt de Inteligencia Artificial (22)
- Codeium
Librería de VSC que usa IA como apoyo al desarrollo (23)

Componentes del equipo y aportación realizada por cada estudiante

Antonio Pastor Aranda: Desarrollo técnico backend y frontend en la aplicación de android, Quality Assurance.

Ignacio Viseras Riego: Desarrollo técnico backend y frontend en la aplicación de android

Pablo López Jimenez: arquitectura, desarrollo del servicio REST, depuración de código

Fases del proyecto

Primera fase:

Realizamos una puesta en conjunto de posibles ideas, exponiendo cómo podíamos mejorar nuestros conocimientos a través de este trabajo.

Segunda fase:

Ejecutamos una búsqueda intensiva para documentarnos acerca del proceso a seguir.

Tercera fase:

Comenzamos a desarrollar el servicio de API REST en java. Llegados a este punto tuvimos bastantes problemas a la hora de implementar la capa de autenticación.

Cuarta fase:

Tras quedarnos atascados en la tercera fase se decidió que Antonio e Ignacio comenzaran el desarrollo de la app en Android mientras Pablo seguía con el servicio de API REST para no quedarnos los tres estancados.

Quinta fase:

Antonio e Ignacio: al ver que se retrasaba la solución para la autenticación se encargan de explorar la opción de prescindir de la API REST y usar en su lugar Firebase.

Pablo consigue solucionar los problemas con la capa de autenticación y consigue terminar el servicio REST completo.

Sexta fase:

Antonio e Ignacio tienen una primera aplicación que registra un usuario en Firebase y genera, guarda y valida datos mediante QR.

Pablo comienza a explorar opciones para desplegar el servicio REST. Al final se decide hacer una cuenta gratuita en AWS para utilizar la capa gratuita de RDS y Elastic Beanstalk.

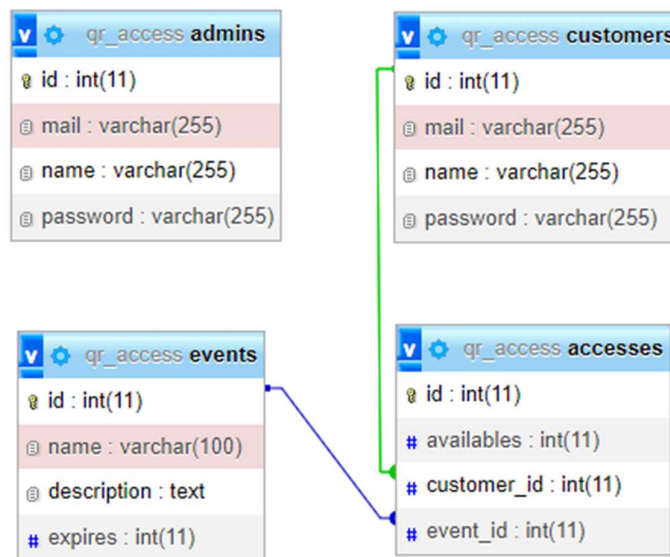
Séptima fase:

Con el servicio REST desplegado optamos por dejar de lado la posibilidad de Firebase y, en base al trabajo ya hecho en Android, mediante la librería Retrofit damos con la clave para autenticar y comenzar a consumir el servicio ya alojado en AWS.

Octava fase: testeo y documentación.

Modelo de datos utilizado

Servicio REST



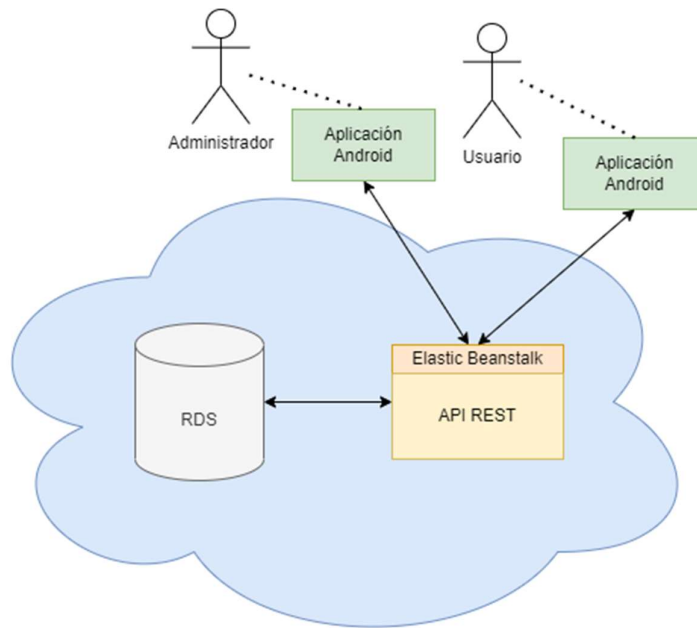
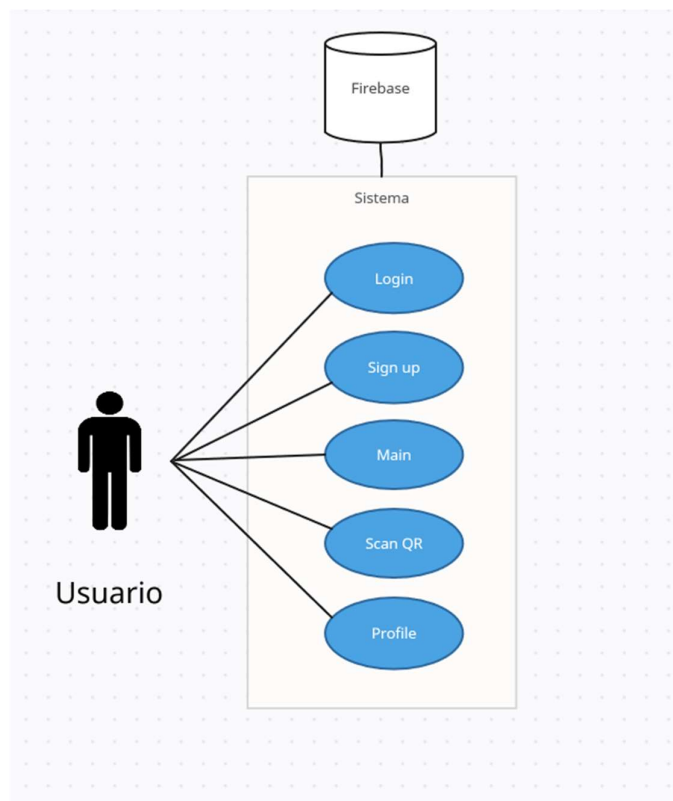


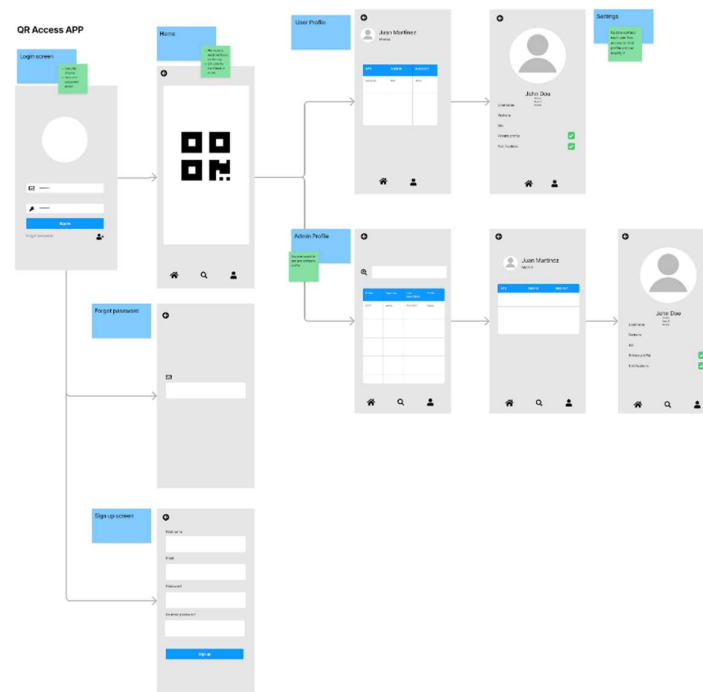
Diagrama de clases y casos de uso

Aplicación Android

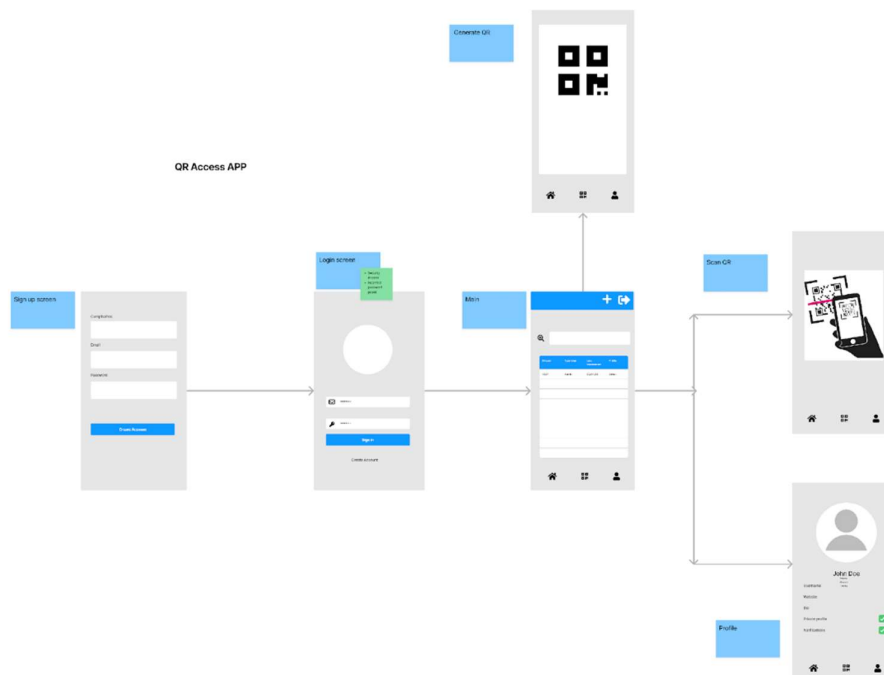


Diseño del mockup

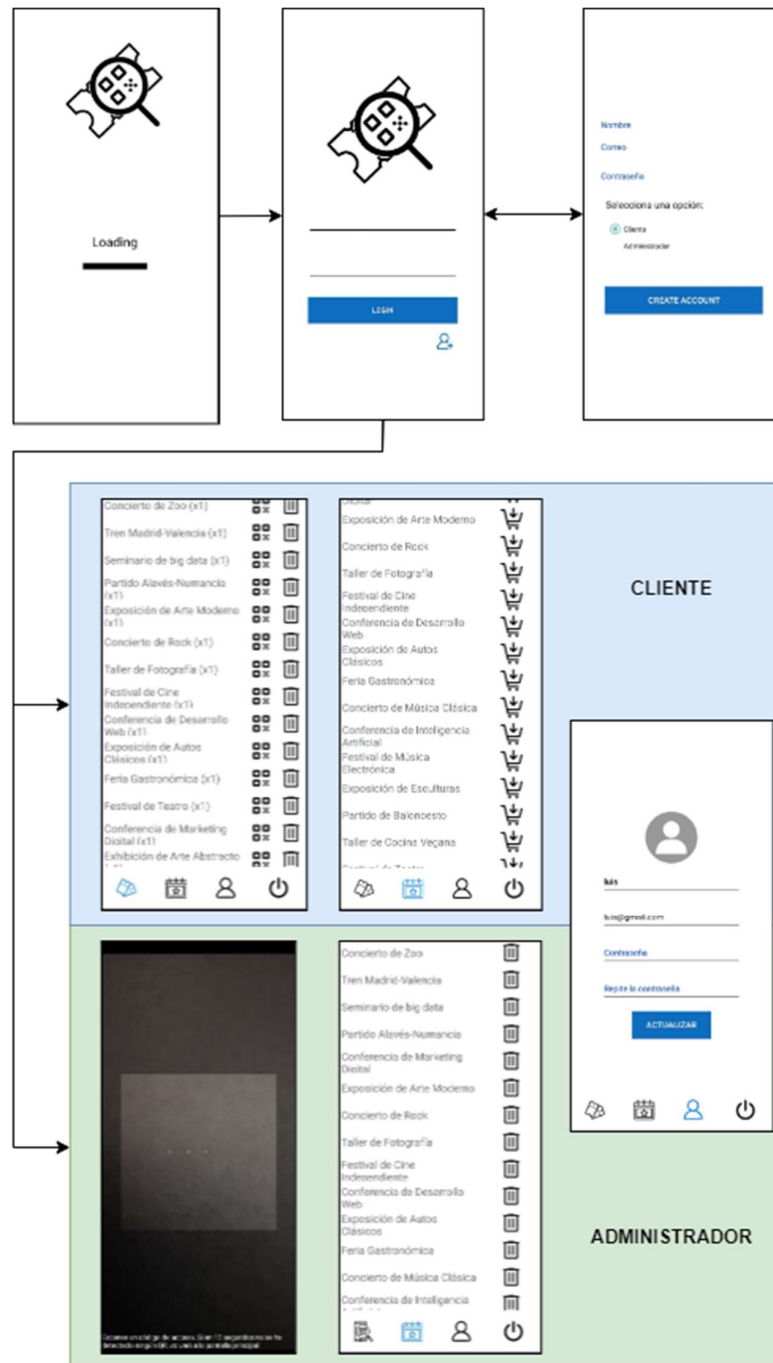
Al presentar la aplicación por primera vez se optó por este diseño:



A medida que el tiempo pasaba se evaluó la posibilidad de modificar algunas pestañas para así mejorar la usabilidad de la app al usuario



Finalmente, tras evaluar los primeros diseños, fuimos rediseñando las distintas vistas al vuelo para hacer la aplicación más agradable a la vista y mejorar la experiencia de usuario. Este es el resultado basado en capturas reales de la aplicación:



Explicaciones de la funcionalidad del proyecto

Servicio de API REST:

Con el objetivo de dar sentido a la parte de multiplataforma desarrollamos un servicio de API REST que servirá para montar la lógica central de negocio y capa de persistencia junto con una base de datos implementada en MySQL que contendrá la arquitectura de datos del proyecto. La capa de base de datos es implementada en el servicio RDS de AWS.

El servicio REST lo implementamos en java 17 con Spring Boot 3.0.5 apoyándonos en Maven. Desplegamos la capa software en AWS montando un entorno de Elastic Beanstalk sobre Corretto 17, una distro de OpenJDK, corriendo en una máquina de 64bit con Linux.(24,25)

Explicación de la capa de bases de datos

La base de datos tiene un esquema sencillo que consta de cuatro tablas:

- Admins: guarda la información relativa a los administradores
- Customers: guarda la información relativa a los clientes
- Events: son los eventos de los cuales un cliente puede comprar un acceso
- Accesses: guarda los accesos a eventos de los clientes. Esta tabla tiene una clave foránea que apunta a la id de customers. Si un cliente es borrado, sus accesos también. En una segunda FK apuntamos a eventos de tal manera que si un evento es borrado, sus accesos también.

Explicación de la capa software

En la capa de modelos se definen las cuatro entidades principales que conforman la base de datos más una quinta cuya función es hacer de apoyo para reducir el código y manejar en paralelo la entidad customer y admin al tener muchas similitudes.

En la capa de Data Access Object, package daos, se implementan las respectivas interfaces de los daos que usa la aplicación y una capa para funcionar con MySQL.

En el package security, se implementa todo lo relacionado con la capa de seguridad: roles, permisos, creación de tokens de accesos, filtros... Todo lo necesario para que la capa de controladores, en el package controllers, pueda validar las peticiones entrantes. Los permisos, que son tres (público, cliente y administrador), marcan la creación de las distintas clases que conforman el controlador. Las peticiones del controlador “public” y las del controlador “event” son públicas y realizables sin necesidad de autenticación. Las peticiones lanzadas contra customer o admin solo las podrá realizar un customer o un admin respectivamente.

Se ha intentado dotar el código de una buena sintaxis y organización para que pueda ser explicado por lo que no entraremos en profundizar más en las explicaciones.

Aplicación Android:

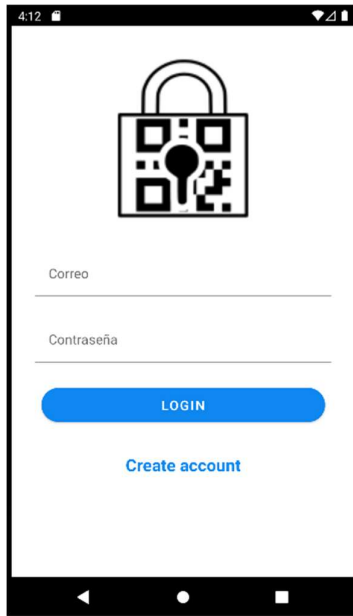
La primera pantalla que podemos encontrar en nuestra aplicación es el Login.

Como podemos observar encontramos dos cajas que recuperamos del xml para poder comprobar los datos y un botón para el envío del login.

Un poco más abajo encontramos un link para la creación de cuenta nueva.

Dentro de las cajas para recuperación de datos, se realiza un control de caracteres mediante el método Regex.

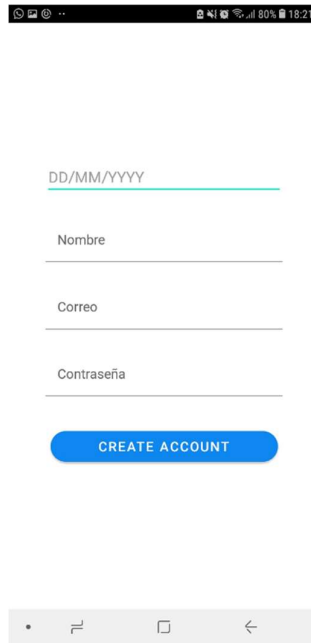
En caso de que alguno de los campos no cumpla saldrá un aviso mostrando el error y de qué tipo es.



Sign Up:

En la pantalla Create account encontramos 4 cajas para rellenar datos en los que se encuentra: edad, nombre, email y contraseña.

Estos datos son recibidos en el método e insertados en la base de datos. La propia base de datos genera un código Hash que asocia al correo y así podemos identificar en todo momento los datos guardados por esta persona.



MainActivity:

En la pantalla main podemos encontrar los botones de menú: Home que nos lleva siempre a la pantalla MainActivity donde encontraremos un listado de los QR y un buscador.

El botón cámara que nos activa el escaneo QR y el botón perfil que nos activa una pantalla para ver los datos de nuestro perfil.

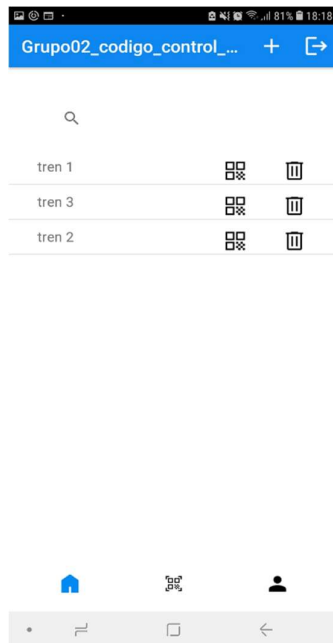
En la parte superior nos encontraremos una barra de búsqueda que filtra por palabra, haciendo desaparecer los demás ítems de la lista. En caso de no utilizar la barra de búsqueda, encontraremos un listado de todos los eventos con opción de abrir el QR o eliminar el evento.

En la barra de menú encontramos un botón para añadir eventos y el botón para salir de la aplicación.

Esta pantalla será el eje central de la aplicación, por ello es desde la única que se podrá salir de la aplicación.

Para estar situado en nuestra aplicación de una manera intuitiva y visual, sabremos en qué apartado nos encontramos observando el cambio de color en la barra de menú inferior.

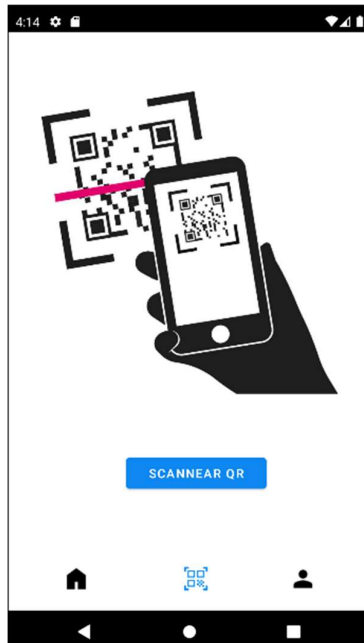
En este caso el botón de la izquierda está azul es por ello que sabemos que estamos en mainActivity o como el usuario lo relacionará “Home”



ScanQr:

Esta pestaña nos permitirá buscar el código QR en nuestra base de datos y comprobar si existe.

En la parte inferior nos encontramos un botón que al presionarlo llama a un método que nos abre la cámara a modo lector de QR . Buscará en nuestro servidor de Firebase si existe en nuestro listado de eventos el código QR leído.



Profile:

El apartado profile consta de los datos del usuario como son: nombre, fecha de nacimiento y correo electrónico.

En la parte inferior de la pantalla se encuentra el botón de cambiar de contraseña donde a modo de hipervínculo nos redirecciona a la gestión de la contraseña por medio de una petición a nuestro servidor enviando un correo al correo de registro añadido por el usuario anteriormente con el cambio de contraseña pedido.

También tenemos un botón para dar de baja de la aplicación que realizará una suspensión de la cuenta.



Una vez alcanzada la séptima fase del proyecto comenzamos un proceso de rediseño de la aplicación. Procedemos a explicar la versión definitiva.

La primera pantalla que se muestra es el Splash. Tomamos ventaja de esta animación inicial para cargar los eventos en la aplicación. Para los eventos no se necesita estar logueado porque son de dominio público. Una vez que se cargan los eventos, y dejando la animación del splash unos segundos más:

Si no te has logueado previamente te lanzará una ventana de login:



En esta pantalla tenemos la opción de entrar en la aplicación o de crear un usuario nuevo. La opción natural para un nuevo usuario es crear un perfil de acceso. Esto nos llevaría a la segunda pantalla:

Nombre

Correo

Contraseña

Selecciona una opción:

☒ Cliente

☐ Administrador

CREATE ACCOUNT

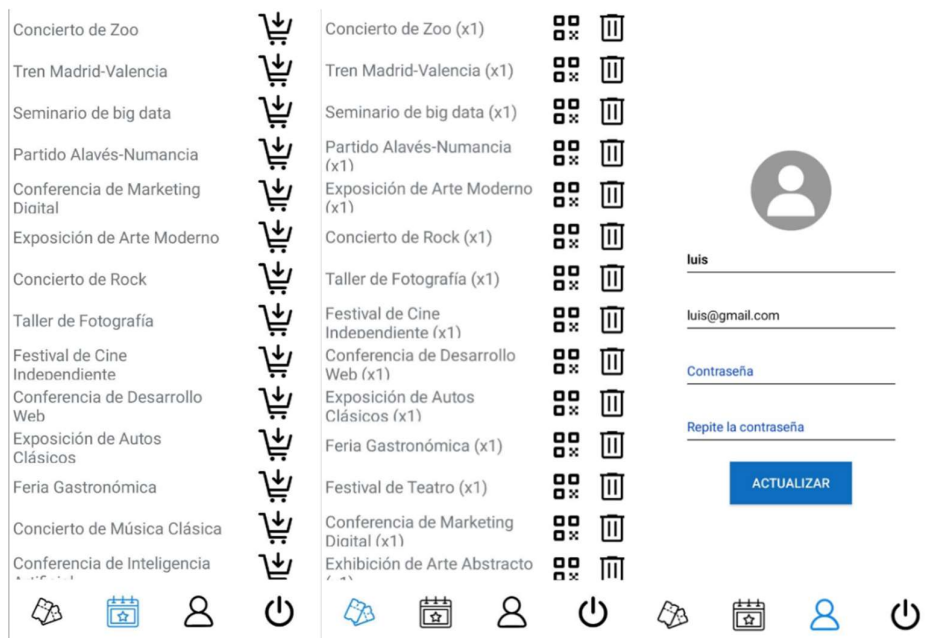
En esta versión inicial del proyecto damos la opción de hacer un perfil cliente o un perfil administrador. No tiene sentido dejar la posibilidad al propio usuario de ser administrador del sistema, pero es algo que tiene una funcionalidad exclusivamente experimental.

Una vez creado el usuario se vuelve a la pantalla de login. Tras loguearse de nuevo, accedemos a la aplicación.

Si accedemos con un usuario cliente tendremos una pantalla con cuatro acciones posibles:

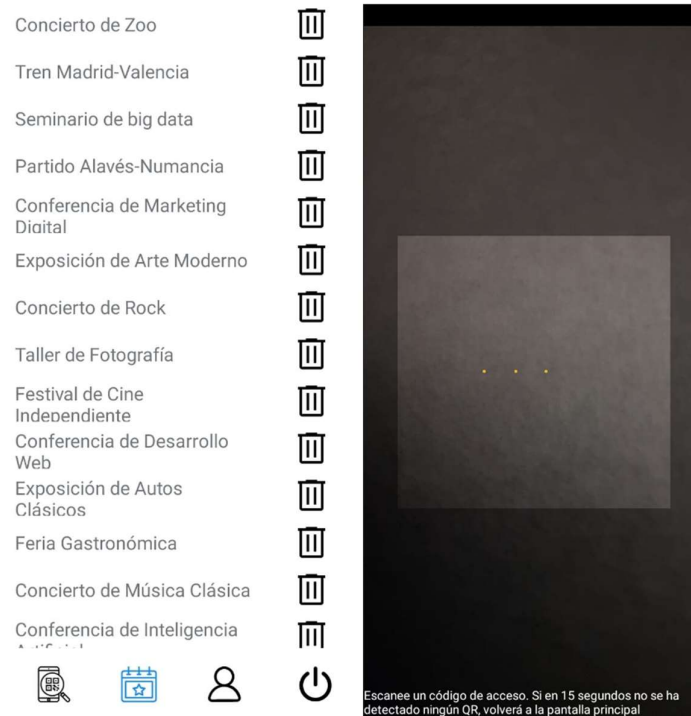
- Ver los accesos comprados para acceder a un evento
- Ver los eventos y comprar un nuevo
- Editar el perfil del cliente
- Salir.

Montamos un menú inferior mediante un fragment para poder navegar por las distintas pantallas de la aplicación:



Si accedemos como usuario administrador, a parte de la opción de salir y editar perfil, podremos:

- Validar un acceso comprado por un cliente con la cámara del móvil
- Ver los eventos y borrarlos.



Como se observa en la primera captura, la opción de consultar accesos se sustituye en este caso por abrir la cámara. La segunda captura muestra la cámara abierta en disposición de validar un QR. Si en 15 segundos no valida nada se cierra automáticamente.

El botón de salir nos desloguea de la aplicación y borra todos los datos persistidos en la base de datos interna. Si el usuario cierra la aplicación sin desloguearse persistirá la información. En caso de que el usuario quiera acceder a un acceso sin conexión de red, si no se ha deslogueado previamente, podrá consultar el acceso y el QR pertinente.

Apuntes técnicos:

- Nos hemos valido de la librería Room para construir el ORM que gestiona la persistencia en Android.
- Para poder atacar la API REST en Android nos hemos valido de la librería Retrofit.
- Hemos implementado `androidx.fragment:fragment:1.4.0` para poder crear un menú común a todas las activities mediante fragments.
- Hemos creado un helper para poder manejar y editar los toasts del sistema.

- Las carpetas en las que se estructura el código intentan explicar su funcionalidad per se. No obstante comentar:
 - La carpeta adapters contiene una implementación extendida de ArrayAdapter para manejar objetos de tipo Access y Event.
 - La carpeta helpers contiene algunos helpers para manejar eventos y accesses. Además añade una clase para manejar toast y otra AppUtils que contiene varias utilidades (como chequear un email o guardar un JWT).

Todo el trabajo del proyecto es posible encontrarlo en este enlace a el repositorio Github:
<https://github.com/ignacioviseras/TFG-DAM.git>

Conclusiones y mejoras del proyecto

Ha sido un proyecto complejo y bastante exigente con el que hemos pretendido abarcar prácticamente todo el curso de 2º de Desarrollo de Aplicaciones Multiplataforma.

Creemos que la transversalidad del proyecto es total respecto a todas las asignaturas técnicas que hemos cursado.

Como punto fuerte destacamos los conocimientos adquiridos en Android y la introducción a la computación en la nube usando Amazon Web Service.

Dificultades a nivel personal

En lo personal la dificultad de compaginar horarios de trabajo y conciliación ha hecho mella en nuestra salud mental. Notamos como tras ocho horas de trabajo, más en algunos casos, era muy difícil sacar tiempo para poder llegar a la presentación de junio. Ha hecho mella en nuestro carácter, en nuestro ánimo e incluso en nuestras relaciones interpersonales llevándonos a momentos de tensión que siempre hemos resuelto de manera satisfactoria.

Dificultades a nivel técnico

- Dudas a la hora de implementar JPA o JDBC en el servicio de API REST. Finalmente nos decantamos por la segunda porque era menos agresiva.
- Problemas para implementar la capa de seguridad de la API REST. Nos encontramos con que la mayoría de las librerías estaban deprecadas. Finalmente, tras mucho tutorial y StackOverflow, dimos con un video actualizado que lo explicaba perfectamente y lo usamos como base para implementar la seguridad.
- Despliegue de la app. Buscando una nube gratuita fuimos con intención de usar Heroku. Tras comprobar que ahora es de pago buscamos una alternativa. Topamos con AWS que permite hacer pequeños desarrollos creando una cuenta gratuita de 12 meses y comenzamos a probar. Entramos en pánico al no poder desplegar la aplicación en Elastic Beanstalk al encontrarnos con un problema de roles en AWS.

Llegados a ese punto valoramos la posibilidad de alquilar algún VPS en Contabo para desplegar la aplicación en Ubuntu Server dockerizando todo el servicio. Por suerte no hizo falta porque conseguimos resolver el problema de roles en AWS y desplegar la aplicación satisfactoriamente.

- A la hora de intentar usar Firebase hemos tenido problemas para poder gestionar los roles. Esto limitaba bastante el proyecto inicial. Por suerte al tener ya el servicio de API REST corriendo pudimos desarrollar 100% la idea inicial de proyecto y dejamos la opción de Firebase.

Como hemos dicho a lo largo de la memoria abarcar todo lo que queríamos hacer en tan poco tiempo es impensable. No obstante, no queríamos dejar pasar la oportunidad de exponer algunas de las mejoras necesarias e innecesarias pero que hacen más usable la aplicación.

Fallos y problema no resueltos:

- Los eventos los estamos metiendo directamente en la base de datos. Es necesario crear una opción para que el usuario administrador pueda meter los datos desde la aplicación móvil.
- Arreglar el fallo de diseño que hay en la base de datos. En la tabla Accesses hay una columna event_id que debe de representar una FK de la tabla events. Es importante que cuando se borre un evento se borren todos los accesos a ese evento. Es un bug importante que hemos descubierto demasiado tarde como para afrontar su resolución, pero no queríamos dejar pasar la oportunidad de exponer que tenemos constancia de ello.
- Un usuario no puede borrarse así mismo. La única forma de borrar tanto usuarios como administradores es por un gestor de base de datos. Al cierre de esta memoria queda pendiente implementarlo.
- A veces los eventos no se cargan a la primera en la aplicación. No sabemos si llegaremos a solucionarlo para presentar el proyecto, pero lo intentaremos.
- No hay implementado un modo oscuro para Android.
- El administrador no puede editar eventos.
- El sistema de testing es muy deficiente e incluso inexistente. Esto es un problema nuestro. Desde el principio no focalizamos en la realización de tests debido a la falta de tiempo.

Mejoras propuestas:

- Crear una app sencilla en JavaScript para dar sentido a la idea central de hacer una API REST multiplataforma. Por falta de tiempo no llegamos.
- Implementar un módulo de Odoo para poder hacer gestiones de administrador sobre todo para poder crear/borrar usuarios y crear/borrar eventos.
- Permitir subir una imagen de perfil
- Añadir opciones para que el usuario pueda personalizar la GUI
- Implementar Integración Continua con GitHub y AWS.
- Crear un mejor sistema de testing implementando test unitarios y automatizando pruebas de api.
- Contar con el testeo de varios usuarios para solicitarles qué harían para mejorar su experiencia de usuario.

Bibliografía

1. ¿Qué es una API REST? | IBM [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.ibm.com/es-es/topics/rest-apis>
2. Generalidades del protocolo HTTP - HTTP | MDN [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://developer.mozilla.org/es/docs/Web/HTTP/Overview>
3. MySQL [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.mysql.com/>
4. ¿Qué es JDBC? - Documentación de IBM [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.ibm.com/docs/es/informix-servers/12.10?topic=started-what-is-jdbc>
5. Cómo guardar contenido en una base de datos local con Room | Android Developers [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://developer.android.com/training/data-storage/room?hl=es-419>
6. UX/UI: ¿Cómo mejorar la experiencia de usuario en desarrollos de software? [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://es.linkedin.com/pulse/uxui-c%C3%B3mo-mejorar-la-experiencia-de-usuario-en-desarrollos-software>
7. AWS | Cloud Computing - Servicios de informática en la nube [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://aws.amazon.com/es/>
8. Open Source ERP and CRM | Odoo [Internet]. [citado 2 de junio de 2023]. Disponible en: https://www.odoo.com/es_ES
9. ¿Qué es Java y por qué lo necesito? [Internet]. [citado 2 de junio de 2023]. Disponible en: https://www.java.com/es/download/help/whatis_java.html
10. Download Android Studio & App Tools - Android Developers [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://developer.android.com/studio/>
11. Eclipse Downloads | The Eclipse Foundation [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.eclipse.org/downloads/>
12. Visual Studio Code - Code Editing. Redefined [Internet]. [citado 11 de mayo de 2022]. Disponible en: <https://code.visualstudio.com/>
13. Firebase [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://firebase.google.com/?hl=es-419>

14. GitHub [Internet]. [citado 11 de mayo de 2022]. Disponible en: <https://github.com/>
15. XAMPP Installers and Downloads for Apache Friends [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.apachefriends.org/es/index.html>
16. Notion – Your wiki, docs & projects. Together. [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.notion.so/>
17. pdfmake/margins.js at master · bpampuch/pdfmake [Internet]. [citado 11 de mayo de 2022]. Disponible en: <https://github.com/bpampuch/pdfmake/blob/master/examples/margins.js>
18. Recently viewed – Figma [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.figma.com/files/recent?fuid=1025466395132524438>
19. draw.io [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://app.diagrams.net/>
20. Rueda de colores, un generador de paletas de colores | Adobe Color [Internet]. [citado 1 de junio de 2023]. Disponible en: <https://color.adobe.com/es/create/color-wheel>
21. Draw Freely | Inkscape [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://inkscape.org/es/>
22. Introducing ChatGPT [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://openai.com/blog/chatgpt>
23. Codeium · Free AI Code Completion & Chat [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://codeium.com/>
24. AWS | Elastic beanstalk para aplicaciones web desarrolladas con Java [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://aws.amazon.com/es/elasticbeanstalk/>
25. OpenJDK [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://openjdk.org/>
26. Implement Barcode QR Scanner in Android Studio Barcode Reader | Cambo Tutorial - YouTube [Internet]. [citado 2 de junio de 2023]. Disponible en: <https://www.youtube.com/watch?v=jtT6oyFPeII>

Anexos

Generación de QR

- Creamos una variable llamada qrData y la utilizamos para almacenar el valor de text. (26)
- Instanciamos QRCodeWriter (es una biblioteca para generar códigos QR)
- Utilizamos el método encode para codificar qrData en un formato QR que será guardado en un objeto BitMatrix, para eso le tenemos que pasar los siguientes parámetros:
 - qrData: texto que se desea encriptar
 - BarcodeFormat.QR_CODE: formato al que se desea encriptar
 - 900x900: tamaño del qr que se desea generar
- Guardamos el tamaño que se indicó anteriormente
- createBitmap: se crea un bitmap vacío con las dimensiones anteriores
- En los for se va a ir pintando para eso lo que se hace es comprobar que en la posición en la que se está tocaría un punto negro de ser así se pone en negro de lo contrario se deja en blanco o del color que se quiera indicar.
- Por último, seteamos el bitmap en la imagen que se instancio al principio del código

```

public void visualizarQr (String text){
    qrImageView = findViewById(R.id.qrImageView);

    String qrData = text;

    QRCodeWriter qrCodeWriter = new QRCodeWriter();
    try {
        BitMatrix bitMatrix = qrCodeWriter.encode(qrData, BarcodeFormat.QR_CODE, width: 900, height: 900);
        int width = bitMatrix.getWidth();
        int height = bitMatrix.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                bitmap.setPixel(x, y, bitMatrix.get(x, y) ? Color.BLACK : Color.WHITE);
            }
        }
        qrImageView.setImageBitmap(bitmap);
    } catch (WriterException e) {
        e.printStackTrace();
    }
}
}

```

Lectura de QR

- Instanciamos la clase ScanOption esta clase nos permite configurar el escaneo del QR.
- setBeepEnabled nos permite activar el sonido de tal forma que sonará cuando scanne un QR.
- setOrientationLoked nos permite establecer la orientación de la pantalla en vertical.
- setCaptureActivity Indicamos que todo esto se realizará en CaptureQR
- launch(options) accionamos el método launch con todas las opciones indicadas.
-

```

public void scanQR() {
    ScanOptions options = new ScanOptions();
    options.setBeepEnabled(true);
    options.setOrientationLocked(true);
    options.setCaptureActivity(CaptureQR.class);
    launcher.launch(options);
}

```

Método launch

- qrResult guarda el texto del qr leído, verificaremos que contiene un texto.
- Luego buscamos en nuestro firebase una coincidencia de los registros y el qr leído
 - En caso de ser correcto llamará a un método que mostrará una ventana emergente avisando que es correcto.
 - En caso de ser incorrecto llamará al mismo método, pero informando que es invalido el qr.

```
ActivityResultLauncher<ScanOptions> launcher = registerForActivityResult(new ScanContract(), result -> {
    if(result.getContents() != null){
        flag = false;
        qrResult = result.getContents(); // texto del qr leído
        if (qrResult != null && !qrResult.isEmpty()) {
            Toast.makeText(context: ScannerQR.this, qrResult, Toast.LENGTH_SHORT).show();
        }
        //validamos que el qr exista en la bd
        CollectionReference registrosRef = db.collection( collectionPath: "Registros");
        registrosRef.whereEqualTo( field: "nombreRegistro", qrResult) Query
        .get() Task<QuerySnapshot>
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                QuerySnapshot snapshot = task.getResult();
                for (DocumentSnapshot document : snapshot.getDocuments()) {
                    validacion = document.getString( field: "nombreRegistro");
                    if (validacion != null && validacion.equals(qrResult)){
                        flag = true;
                        ventana( validacion: "QR correcto");
                    }
                }
            } else {...}
        })
        if(flag == false)
            ventana( validacion: "QR no valido");
    }
});
```

