TECH VIRTUAL MUSEUM

INFORME SOBRE EL DESARROLLO DE LA APLICACIÓN

Nahima Ortega Rodríguez y Tinizara Rodríguez Delgado

2022

ÍNDICE

1. INTRODUCCIÓN

Tech Virtual Museum es una aplicación nativa, desarrollada tanto para Android como para iOS, dirigida a los usuarios de un museo virtual de carácter tecnológico que tiene un código QR asociado a cada uno de los elementos que se exponen. Desde la aplicación, pueden escanear este QR para acceder a la información sobre el objeto expuesto, de manera que todos los usuarios puedan leer la información desde su dispositivo móvil, de forma más rápida, clara y cómoda.

No solo les proporciona esta facilidad a los usuarios, sino que, además, se centraliza toda la información del museo en una misma aplicación, desde compra de entradas hasta eventos que van a tener lugar próximamente en éste.

2. DESCRIPCIÓN

Tech Virtual Museum ha sido desarrollada empleando la metodología Scrum. Ha pasado por diversas fases, que veremos a lo largo de este informe: diseño, análisis, desarrollo, implementación y soporte.

En un prototipo inicial, hemos diseñado la aplicación para que tuviese componentes de realidad aumentada, para hacer la estancia de los usuarios más amena y atractiva. El elemento distintivo del museo sería ofrecer una experiencia inmersiva a través de su aplicación.

La funcionalidad básica de la aplicación es el escaneo de un código QR asociado a cada elemento, a partir del cual podremos ver todos los detalles del elemento, así como una sección de comentarios y un vídeo.

Existen otras funcionalidades desde la aplicación como la posibilidad de ver los eventos, charlas, seminarios, exposiciones, etc., que van a tener lugar próximamente en el museo, de tal manera que podemos reservar una entrada para tener acceso a esta actividad o evento. Además, podemos realizar una compra de entradas normal dentro de la propia aplicación, con la finalidad de acceder directamente al museo sin tener que realizar una cola para comprarlas de manera presencial en el establecimiento.

Dentro de la aplicación, podemos registrarnos o iniciar sesión para agilizar el procedimiento de compra y para realizar comentarios en la sección de comentarios de los elementos escaneados. De igual manera, podremos visualizar los datos de nuestro perfil, así como modificarlos, ya sea nuestro nombre, email o contraseña. También podremos cerrar sesión y, en el caso de que no recordemos nuestra contraseña, podemos recuperarla con un correo que llegará a nuestro email.

3. ELECCIÓN DE COLOR, DISEÑO DEL LOGO Y OTRAS CARACTERÍSTICAS VISUALES

Imagen 1: Color escogido

Como se podrá ver a lo largo de este informe, la aplicación tiene unos colores específicos, que harán que el usuario los asocie con el museo, así como para aumentar la coherencia con la empresa. El color principal del museo es el color del año 2018 de Pantone (Ultra Violet).

Según la psicología del color, cada color posee un significado oculto tras él, y es capaz de hacernos percibir y comportarnos de manera diferente ante cada uno de estos. Por lo tanto, podemos aprovecharnos de la psicología del color para generar las emociones adecuadas en nuestro público. Además, hay estudios recientes que avalan que el color aumenta el conocimiento de la marca en un 80%, por lo que, mediante el uso de un mismo color, hacemos que los usuarios recuerden la marca, aun cuando no incluimos ningún texto o logotipo.

El color elegido, similar al violeta, es el color de la innovación, creatividad, enseñanza y tecnología. No es extraño que este color, junto a tonalidades de azul, se nos venga a la mente cuando pensamos en este concepto. Tech Virtual Museum es un museo innovador, ya que ofrece una experiencia que cualquier otro museo no suele dar a sus usuarios, introduciendo, además, realidad aumentada.

A partir de este color principal, vamos a generar la siguiente paleta de colores. Emplearemos colores muy similares para mantener la coherencia y consistencia en nuestra aplicación.

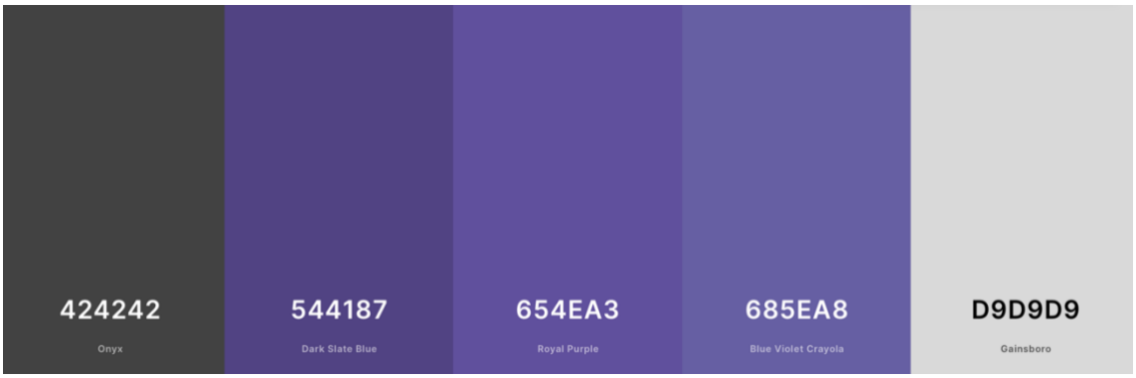


Imagen 2: Paleta de colores empleada

En la imagen anterior, los colores en los extremos van a ser empleados, principalmente, para el color de la fuente, ya que el uso del negro puro en interfaces de usuario está totalmente desaconsejado. Además, estos han sido seleccionados de manera que exista suficiente contraste entre el texto y el fondo, de tal manera que pueda ser leído correctamente por todos los usuarios, incluso aquellos con problemas visuales de algún tipo, consiguiendo asi uno de los principios de las ocho reglas del oro para el diseño de las interfaces como es la usabilidad universal.

Haciendo uso de estos colores, respetamos la regla 60:30:10 del diseño de interfaces de usuario. Esta regla dice que el 60% de los colores empleados en nuestra aplicación deben corresponder al color principal o primerio de nuestra aplicación. El 30% debe corresponder al color secundario, que debe ser usado para algunos componentes como

carrousel, botones, etc. Por último, el 10% restante debe corresponder al color terciario, que no va a ser tan predominante como los anteriores y que servirá para resaltar algunos detalles.

Para el diseño del logo, vamos a seguir una línea minimalista y vamos a emplear los colores seleccionados con anterioridad. Como vemos, realizamos un degradado del color principal de la aplicación para crear el logo. Generamos varias imágenes del logotipo, así como el isotipo, ya que nos serán necesarios para nuestra aplicación. En las imágenes siguientes vemos cómo cambiamos el fondo y el nombre del museo para generar las diferentes formas del logotipo.

El logo ha sido generado empleando el software Canva.

Imagen 3: Logo con nombre

Imagen 4: Logo con nombre

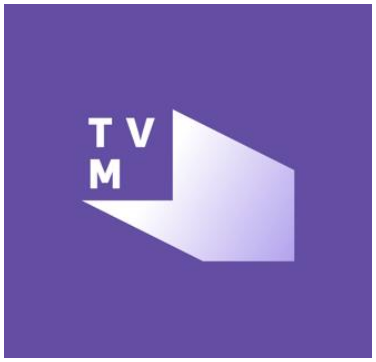


Imagen 5: Logo con iniciales

Imagen 8: Logo con nombre

Imagen 7: Isotipo

Imagen 6: Logo con iniciales

Por otro lado, hemos escogido las fuentes Roboto y Lato, principalmente por su legibilidad y minimalismo, que encajan con el diseño de la aplicación. Estas fuentes han sido seleccionadas, también, en base a las pautas del diseño de interfaces de usuario. En estas guías, se desaconseja el uso de fuentes predeterminadas, como Arial o Times New Roman, entre otras, ya que se encuentran usadas en exceso y los usuarios se encuentran cansados de verlas. Además, se recomienda el uso de, exactamente, dos tipos de letra, ya que, de esta manera, obtendremos un resultado más llamativo y pulido. En nuestro caso, una de las fuentes (Roboto) será empleada para títulos, botones, etc., mientras que la otra (Lato) se empleará para el texto del cuerpo de apartados, secciones, etc.

Como se aprecia, ambas fuentes tienen un cierto contraste entre ellas, de manera que se pueden diferenciar tranquilamente, haciendo que aquellos textos en los que se usa Roboto (títulos principalmente) destaquen.

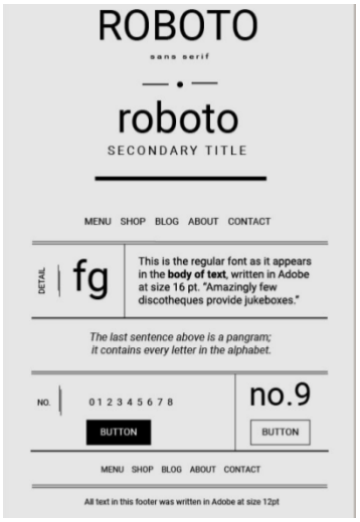


Imagen 10: Fuente Roboto

Imagen 9: Fuente Lato

4. MOCK-UPS EN FIGMA

4.1 Wireframes

En esta primera parte del desarrollo del diseño, se ha optado por empezar con unos pequeños bocetos realizados a mano, donde se plasmarán todas las ideas sin seguir un orden para posteriormente haciendo uso de la herramienta Figma y tras un filtrado de las ideas anteriormente comentadas, empezar a diseñar una idea casi final de cómo debería quedar la aplicación al momento de la entrega, siendo este diseño la base en la que se fundamentaría el aspecto visual de la misma.

Asimismo, generamos el prototipado no funcional, es decir, generamos el flujo que debería seguir la aplicación cuando el usuario navegase por las diferentes pantallas y opciones de cada una de ellas. Gracias a este punto, entre otras ventajas, pudimos minimizar los errores en la experiencia de usuario, además de estimar el tiempo de desarrollo de cada una de las actividades individuales, entendiéndose como actividad, el conjunto de las partes que formarían cada apartado, como por ejemplo la actividad de compra de las entradas para acudir a algún evento. Esta actividad estaría formada por cuatro pantallas de carga: en una primera, encontramos la lista de éstos, que al hacer *click* en alguno se nos abriría la pantalla de sus detalles y donde accederíamos al botón de compra. Si hiciéramos *click* en él, se nos llevaría al proceso de compra donde el usuario debería seleccionar el número de entradas e introducir sus datos de pago; para, por último, llegar a la pantalla informativa que nos avise de que se ha realizado la compra de forma correcta.

A continuación, se muestran las imágenes de todas las actividades sin entrar en detalles de que tipografía, colores, logo y recursos se usarán.

a. Splash Screen

Esta sería la primera pantalla que le aparecerá al usuario al hacer *click* en el icono de la aplicación. Las *splash screen* se usan como una pantalla de carga para que al usuario no

se le haga molesto el tiempo de espera hasta que se inicie por completo la aplicación, es decir, se puede entender como un punto intermedio entre nuestro escritorio en el teléfono móvil y la aplicación en cuestión.

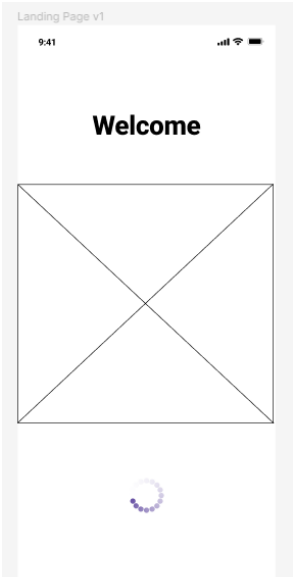


Imagen 11. Splash screen

b. Inicio de sesión. Registro de usuario. ¿Has olvidado tu contraseña?

A continuación, encontramos un grupo de tres vistas, que muestra los diferentes campos que el usuario deberá rellenar para iniciar sesión o registrarse, así como su ubicación en las diferentes pantallas. Estos campos se validarán haciendo uso de una herramienta propia de Firebase Auth que se explicará posteriormente. Además, desde la propia aplicación, antes de enviar los datos a Firebase Auth, se validarán.

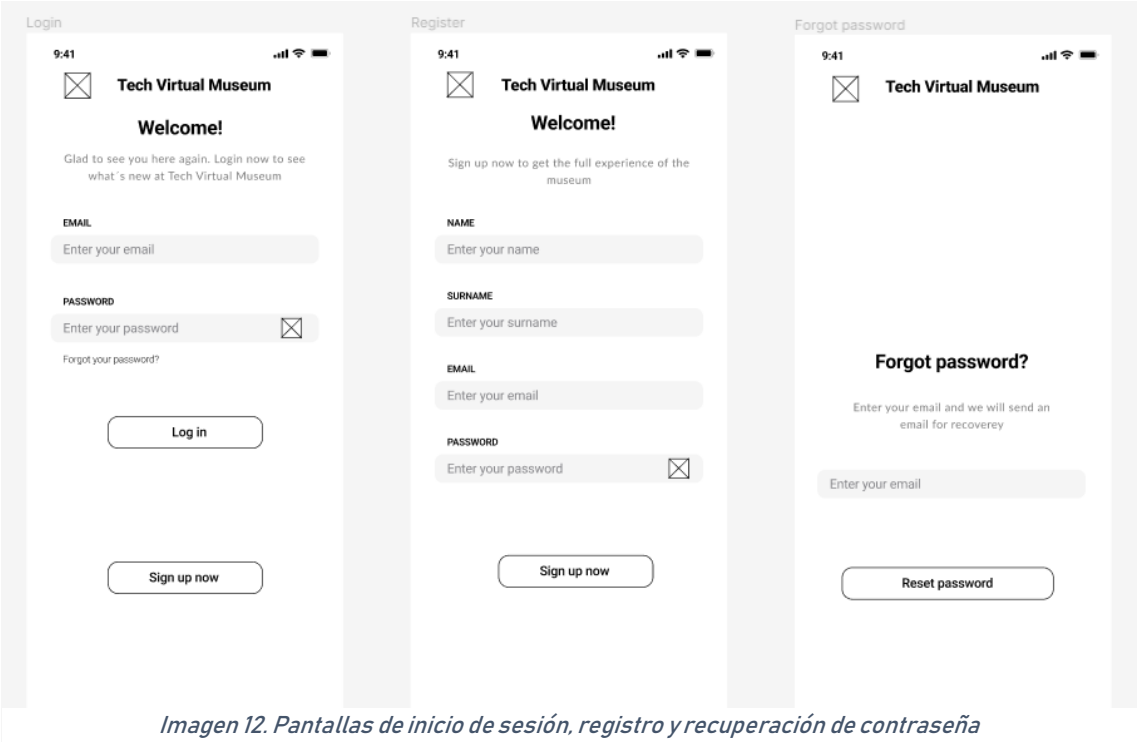


Imagen 12. Pantallas de inicio de sesión, registro y recuperación de contraseña

c. Home page

A la pantalla de inicio, el usuario llegará tras haber iniciado sesión o haberse registrado. En ella, primera se encontrará con un *slider* formado por los ítems del museo que sean más populares para, a continuación encontrar un botón que, al hacer *click,* nos llevaría a la vista mediante la cual el usuario podría comprar las entradas para acudir al museo.

En la parte final de la pantalla, nos encontraremos con un mapa que ayudará al usuario a saber dónde se encontraría el museo. Podemos interactuar con él como un mapa normal y, si lo pulsamos, nos llevará a la aplicación de Google Maps o Apple Maps, con la ubicación del museo, para poder visualizarlo a pantalla completa o para poder iniciar una ruta que nos lleve hasta él.

Asimismo, en la parte superior derecha nos encontraremos con un botón que nos lleva a la pantalla de perfil del usuario.

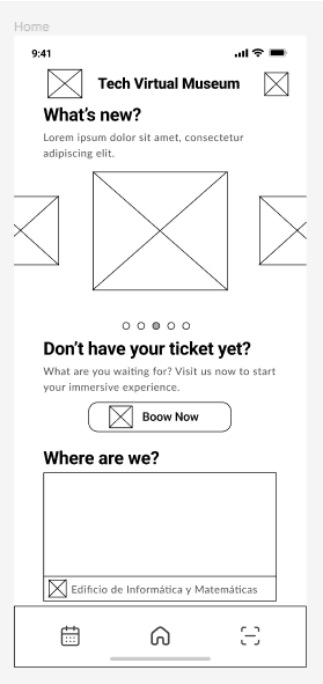


Imagen 13. Home page

d. Pantalla de información personal. Edición de datos.

La pantalla de perfil del usuario es una pantalla en la que se pueda visualizar la información del usuario como su nombre, apellidos o email.

Asimismo, si hacemos *click* en el botón de editar el perfil, el usuario podrá modificarlos en cualquier momento. Estos datos se actualizarán haciendo uso de Firebase Auth y Firestore. Además, se validarán los campos que se introduzcan, al igual que se hacía para el registro y el inicio de sesión.

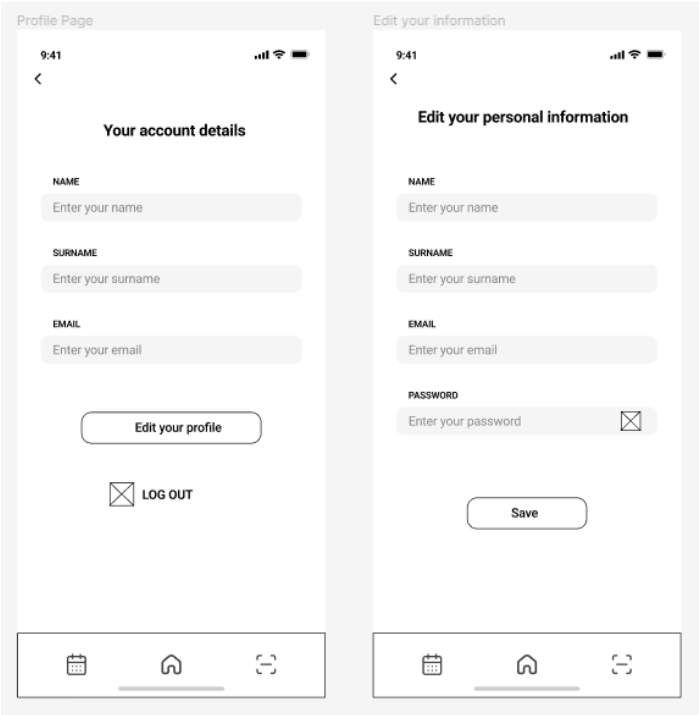


Imagen 14. Pantalla de muestra de la información de perfil y pantalla de edición de los datos

e. Compra de entradas

Estas pantallas hacen referencia al proceso de elección de fechas y compra de las entradas para acudir al museo.

Está formado por, en una primera pantalla, un calendario y unos botones de selección de horas para elegir en qué fecha y hora el usuario quiere ir al museo. A continuación, elegirá cuantas entradas quiere y que tipo de usuario será(n); para finalmente mostrarle un resumen del número de entradas y el total que deberá pagar, junto con su información personal y unos botones para seleccionar que método de pago.

En la parte superior de cada pantalla, tendremos un indicador de en qué momento del proceso nos encontramos, para hacer que el usuario pueda seguir el progreso de este proceso, así como saber en qué paso de éste se encuentra y cuánto falta para completarlo. De esta manera, estamos aplicando el principio de progreso visual de las interfaces de usuario, de forma que mejoramos la experiencia del usuario al proporcionar una sensación de avance y una mayor comprensión de lo que está sucediendo.

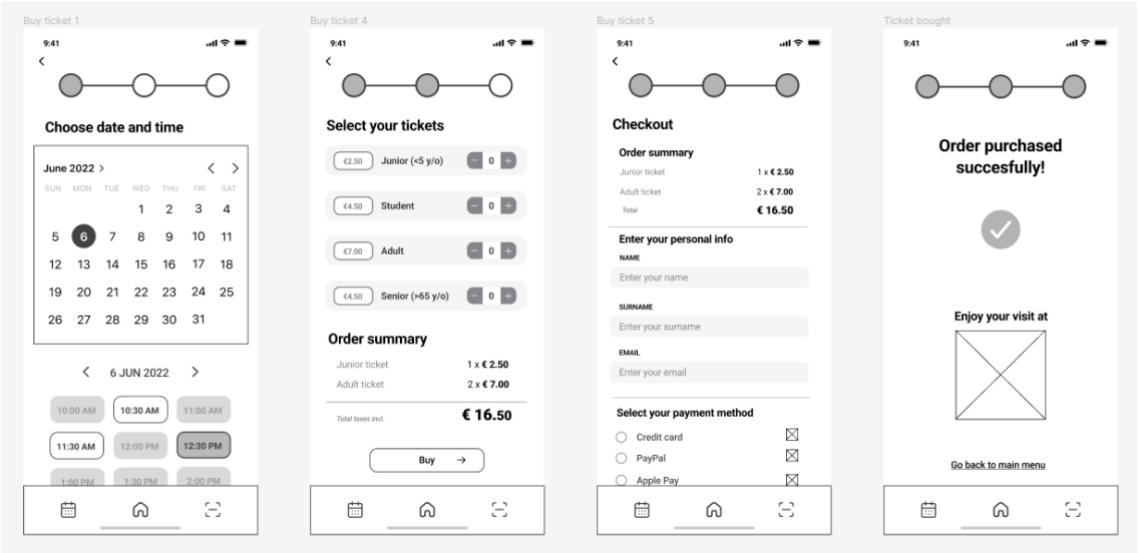


Imagen 15. Conjunto de pantallas a través de las cuales el usuario podrá comprar una entrada para acceder al museo

f. Lista de eventos y pantalla de detalle

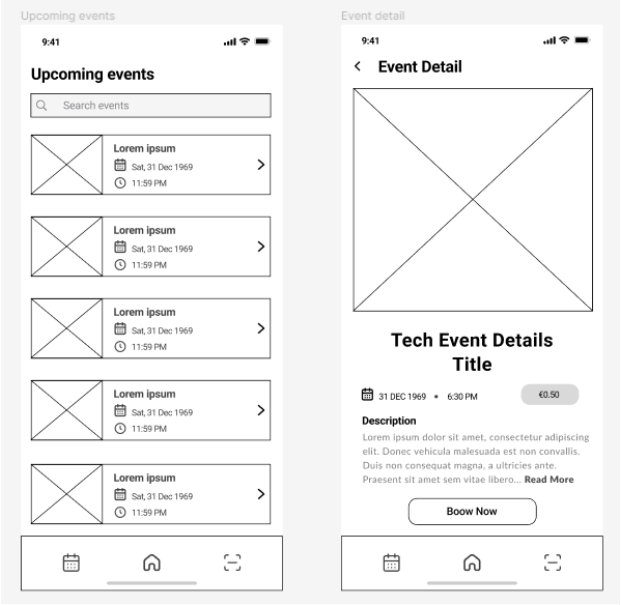


Imagen 16. Lista de los eventos y pantalla de detalles

La primera pantalla de la lista de eventos, que aparece al pulsar el botón de la barra inferior de navegación con el icono del calendario, estará formada por una lista de eventos que se obtiene desde Firestore y en la que se muestra el nombre, fecha y hora de cada evento.

Si el usuario quiere visualizar más detalles de un evento concreto, solo tiene que hacer *click* en él y se accederá a la pantalla de la derecha, donde, además de los datos anteriores, se podrá ver una pequeña descripción, el precio de su entrada y el botón para comprar la entrada para ese evento.

g. Compra de entradas para un evento seleccionado

Cuando el usuario hace *click* en el botón inferior de la anterior vista, accede a esta nueva vista, donde solo debe seleccionar el número de entradas y elegir el método de pago, ya que el apartado de su información personal se rellena automáticamente con los datos que él mismo ha introducido al registrarse por primera vez en la aplicación.

Finalmente, si el pago se procesa correctamente, al usuario se le muestra una pantalla informativa de que así ha ocurrido para no generar incertidumbre en él y proporcionar *feedback*.

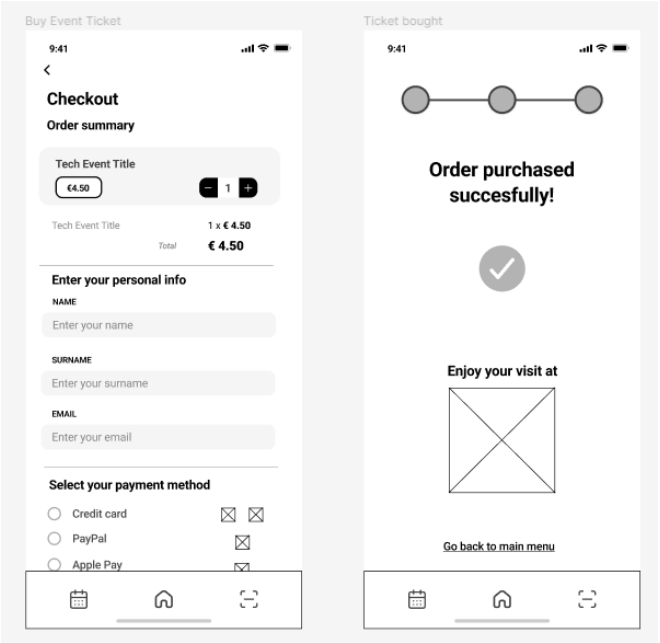


Imagen 17. Compra de tícket específica para un evento

h. Escáner QR y detalles del producto

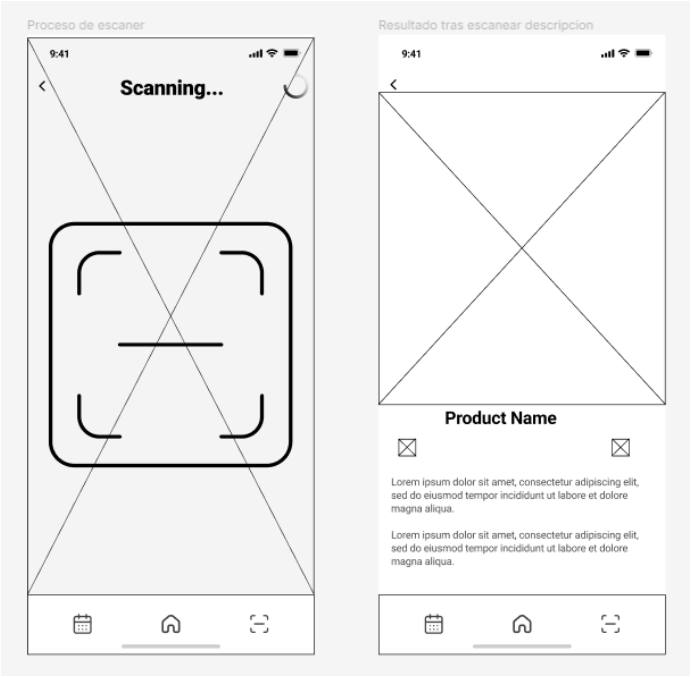


Imagen 18. Proceso de escanear un código QR y pantalla de los detalles del producto escaneado

Aunque el usuario puede acceder a esta pantalla en cualquier momento, le dará realmente uso cuando se encuentre en el museo, ya que será ahí donde pueda escanear los diferentes códigos QR de cada objeto para poder visualizar una serie de datos sobre él, además de reproducir un pequeño vídeo explicativo o acceder a una pantalla de muestra y creación de comentarios sobre el ítem correspondiente. A esta pantalla se accede presionando el botón de escáner QR que se encuentra en la barra de navegación inferior.

i. Creación y muestra de los comentarios. Reproductor de vídeo

A la vista del usuario, pueden ser las pantallas más atractivas debido a que puede dar su *feedback* a través de un comentario sobre las diferentes exposiciones a las que acuda en el museo, además de poder leer también las opiniones y reseñas del resto de usuarios que también hayan visitado esos objetos.

Asimismo, el usuario también podrá acceder a un reproductor en el que se mostrará un pequeño vídeo explicativo sobre el ítem escaneado o el anuncio publicitario de la época en el que ítem fue sacado a la venta.

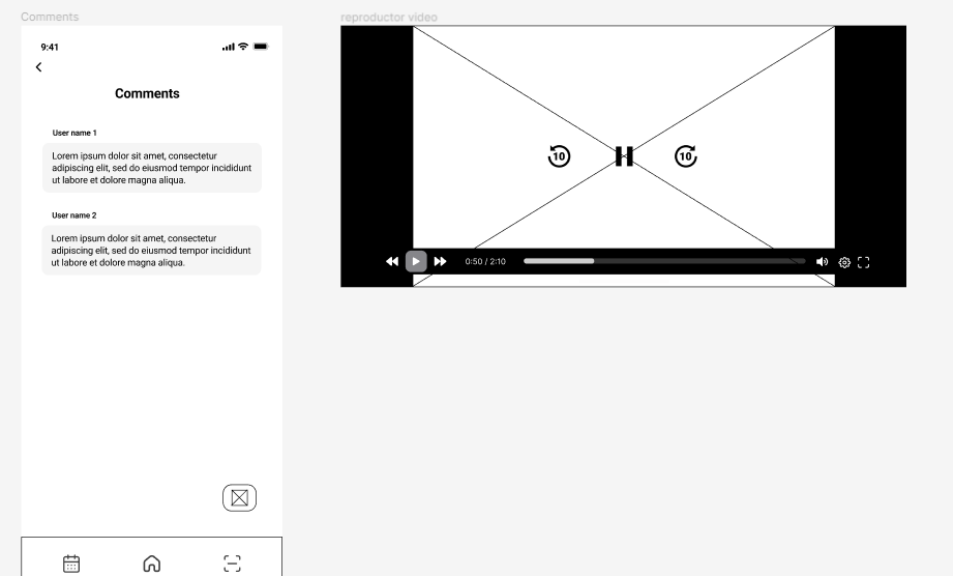


Imagen 19. Muestra de los comentarios y del reproductor del vídeo

4.2 Mockups

Tras el análisis de los *wireframes* anteriores, y poniendo en conjunto los colores corporativos y tipografía anteriormente comentados, se obtuvo como resultado el siguiente diseño de la interfaz de usuario. Este prototipo, a diferencia del anterior, es de alta fidelidad, ya que es una versión más detallada y realista de la interfaz de usuario. Incluyen elementos como transiciones, animaciones y funcionalidades básicas que se van a introducir en la interfaz de usuario definitiva.

En este punto, se ha intentado crear una coherencia y armonía entre las diferentes pantallas, haciendo uso de los mismos estilos en todos los botones y menús, facilitando de igual manera la memoria a corto plazo del usuario. Es decir, hemos aplicado el principio de la coherencia definido en los principios para de diseño de interfaces de usuario.

De igual manera, también podremos ver cómo se han aplicado los diferentes colores de la paleta creada como colores corporativos junto con las tipografías usadas dependiendo de si eran títulos, como los nombres de los eventos o de los productos, o cuerpos de texto, como pueden ser las descripciones de los propios eventos u objetos del museo, brindando, así, al museo una imagen corporativa completa en conjunto con la creación de los logos, también explicados anteriormente.

a. Splash screen

Las *splash screen,* aunque empezaron siendo un elemento estético, esconden una funcionalidad bastante importante como es darle al usuario una pantalla de presentación cuando en realidad, "*en la parte de atrás"* de la aplicación, se están terminando de cargar algunos componentes de ella, para que, al iniciarse por completo, se encuentre a pleno rendimiento.

Esta será la pantalla de carga definitiva, donde nos encontraremos junto con un mensaje de bienvenida, el logo del museo en combinación con el color de fondo que se corresponde con el color corporativo. El uso de uno u otra depende

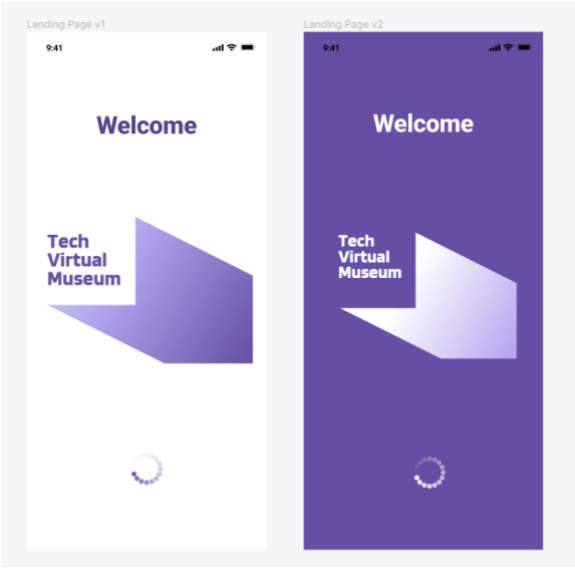


Imagen 20. Pantalla de carga al iniciar a la aplicación

b. Inicio de sesión. Registro de usuario. ¿Has olvidado tu contraseña?

En un primer momento, al iniciar la aplicación cuando no tenemos una sesión iniciada, nos encontraremos con la pantalla de inicio de sesión donde se le dará la bienvenida al usuario, además de encontrarnos con dos campos de texto donde se deberán introducir las credenciales (correo electrónico y contraseña). Los datos introducidos se corresponden con los creados anteriormente en la pantalla de registros en el caso de que no tuviésemos una cuenta. Para la contraseña en el caso de iOS, se encuentra habilitado el empleo del llavero de contraseñas de iCloud.

Si el usuario se encontrase en este último caso comentado, deberá hacer *click* en el botón de registrarse (botón de fondo blanco), para que se le redirija a la pantalla de registro, donde se encuentra con dos campos extras (nombre y apellidos). Esta información se le solicita al usuario no solo para identificarlos, sino para funcionalidades que se comentaran en los próximos apartados, entre las que destacarían la compra de entradas de forma online, o la identificación de qué usuario ha realizado un comentario sin exponer datos sensibles como su correo electrónico.

Todos los datos introducidos por el usuario en estos campos se guardarán en la base de datos, para, por ejemplo, si se diera el caso de que el usuario se olvida de su contraseña, pueda recuperarla. Para ello, puede acceder a la última pantalla que se muestra en la imagen inferior, en donde deberá introducir el correo que utilizó para registrarse. En ese momento, le llegará un correo desde el cual podrá crear una nueva contraseña con la que podrá iniciar sesión.

Como medida de seguridad se solicita el correo electrónico, ya que éste debe ser único, es decir, en la base de datos referente a la gestión de los usuarios no existirán dos usuarios diferentes con el mismo correo, pero, en cambio, sí pueden existir dos usuarios con el mismo nombre y apellido (algo totalmente común en la sociedad).

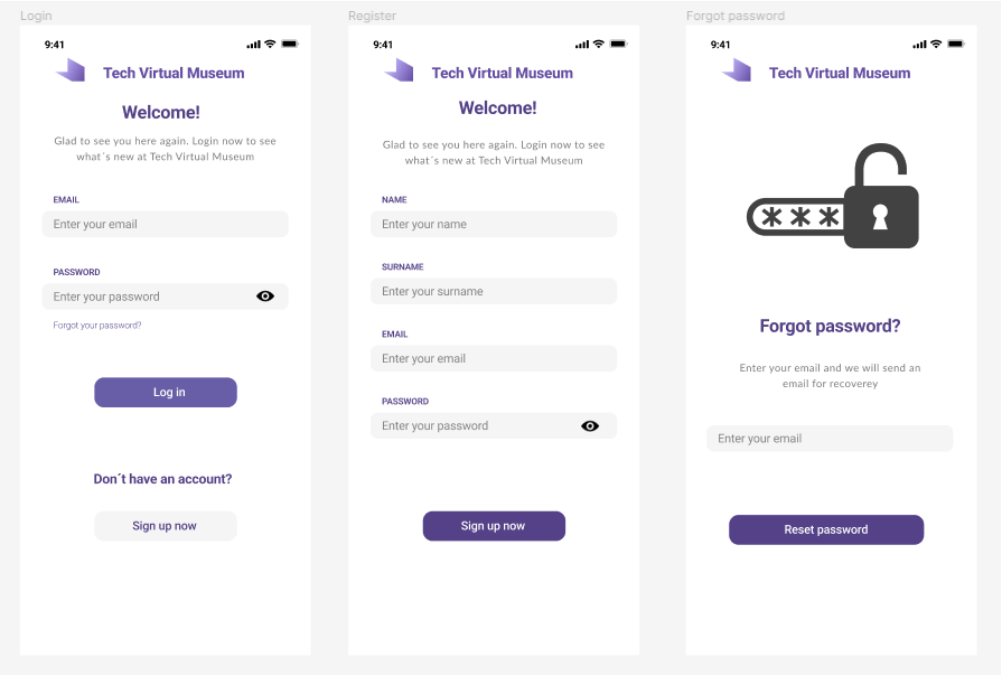


Imagen 21. Inicio y registro del usuario, y apartado por si has olvidado la contraseña

c. Home page

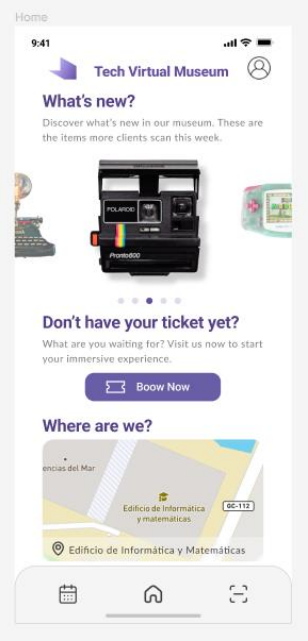


Imagen 22. Pantalla de inicio y principal

Es a partir de esta pantalla de la que nacen el resto de las vistas de la aplicación, ya que es la pantalla principal a través de la cual el usuario puede acceder y editar su información personal como nombre, apellidos y correo electrónico. Por otro lado, puede visualizar cuáles son los ítems más destacados del museo, comprar las entradas que desee y, por último, puede encontrar la ubicación de este de forma fácil, ya que se le muestra un mapa interactivo desde el que iniciar la ruta si desconoce cómo llegar.

Mediante la barra inferior de navegación, el usuario podrá llegar a las tres otras pantallas principales, como son la lista de eventos, la pantalla actual de inicio y el escáner de códigos QR de los productos cuando se encuentre dentro del museo.

d. Pantalla de información personal. Edición de datos.

En esta pantalla, se obtendrá la información del usuario que ha iniciado sesión desde la base de datos con el fin no solo de mostrarla, sino, además, si en algún momento, éste decide modificar algún apartado pueda realizarlo haciendo *click* en el botón inferior (fondo violeta), que le abrirá una nueva pantalla donde se encontrará con unos campos de textos similares a los del formulario de registro.

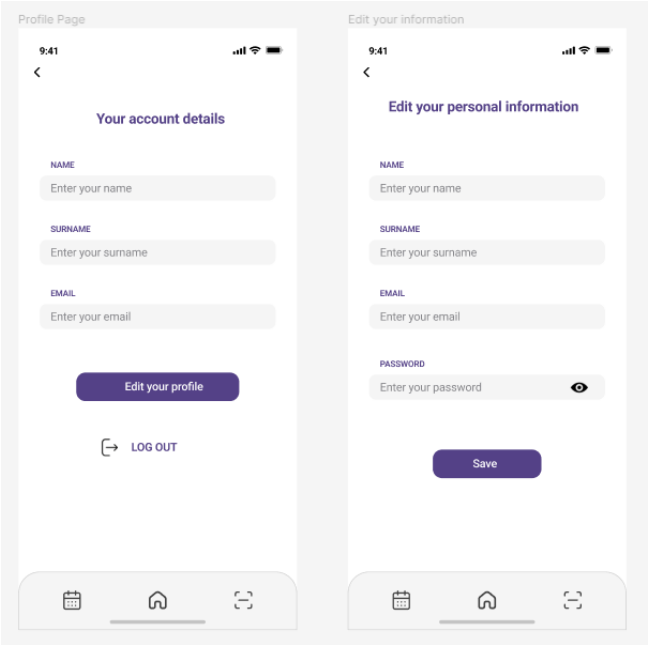


Imagen 23. Pantalla de información y edición del perfil

e. Compra de las entradas

Estéticamente, observamos que, en color verde, se encuentran los precios de los diferentes tipos de entrada, de forma que destaca sobre el juego de las diferentes tonalidades del color base. Asimismo, se ha realizado una diferenciación de los precios dependiendo de en qué intervalo de edad o situación laboral se encuentre el usuario o las diferentes personas que acudirían al museo.

De igual manera, encontramos que la pantalla final es una pantalla informativa en la que también destaca, en color verde, un *tick,* que le indica al usuario si ha realizado de forma correcta el pago de la(s) entrada(s). Se ha optado por usar el color verde en ambos casos (tanto en el precio como en el *tick* al finalizar la compra), para que el elemento destacase a la vista del usuario, ya que, si hubiese optado por usar el mismo color que en el resto de la pantalla, podría haber desapercibido, generando, así, una confusión en el usuario. De igual manera, el color verde es un color entendido internacionalmente de forma que, combinado con el precio y el icono del *tick*, es interpretado correctamente independientemente de la procedencia o edad del usuario. El color verde empleado ha sido seleccionado teniendo en cuenta que tenga una buena ratio de contraste con el blanco, de manera que sea visible por todos los usuarios, independientemente de si tienen alguna discapacidad visual.

Otro punto importante en estas pantallas es el que se había comentado en el apartado del bloque de "*inicio de sesión y registro del usuario*", donde decíamos que añadíamos el campo de nombre y apellidos en el registro, aunque, en principio, no era necesario ni para iniciar sesión, ni si el usuario había olvidado la contraseña. En cambio, para la compra de las entradas sí es necesario, ya que éstas, en principio, estarán a nombre del usuario registrado, ya que estos campos se rellenarán de forma automática cuando el usuario accede a la tercera pantalla del proceso de compra. A pesar de que se autocompleten con los datos del usuario registrado, estos campos se pueden modificar en caso de que se quieran cambiar. Simplemente se autocompletan para ahorrar tiempo al usuario y que no tenga que introducir estos datos constantemente. La idea detrás de introducir el correo y el nombre es que se enviará un recibo de la compra de las entradas al correo indicado por el usuario, de manera que, presentando éste, pueda acceder al museo.

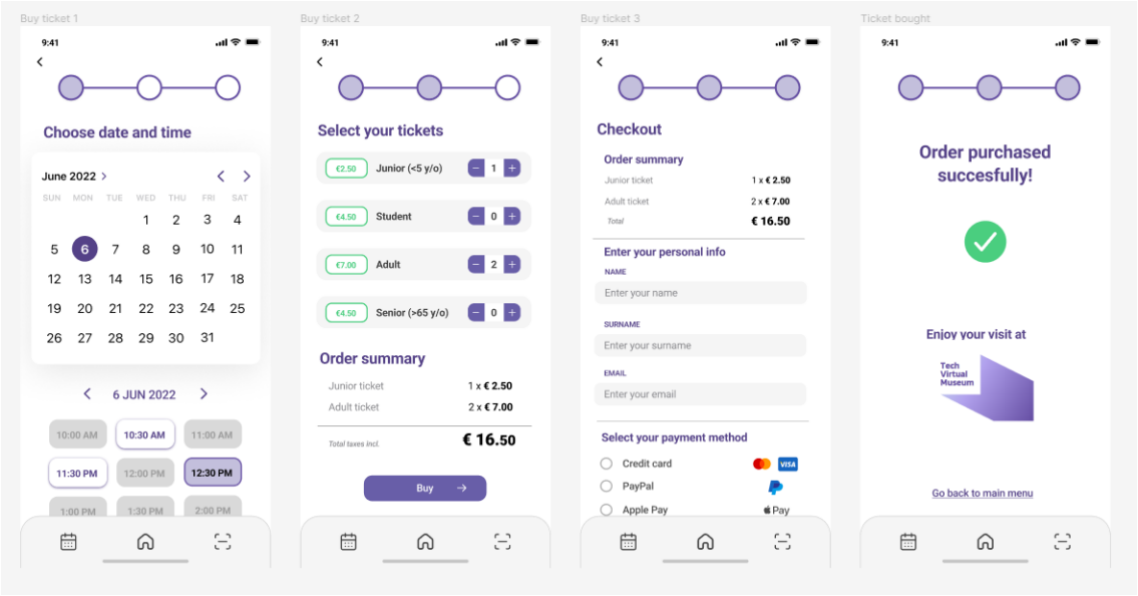


Imagen 24. Conjunto de actividades que conforman la compra de una o varias entradas para el museo

f. Lista de eventos y pantalla de detalles

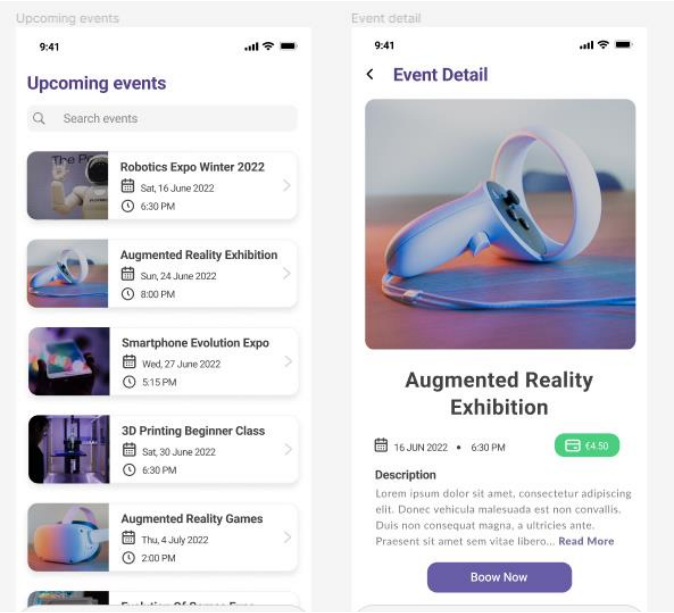


Imagen 25. Lista de los eventos y pantalla de detalle

En esta pantalla, la lista de eventos se obtiene de Firebase haciendo uso de un *adapter*, que nos hará de intermediario entre la propia base de datos donde se encuentra almacenada toda la información y la propia aplicación.

A continuación, tras hacer *click* en el evento que el usuario quiera conocer más detalles, accede a la segunda pantalla donde además de los datos anteriores encontrará el precio y descripción además de un botón que le redirija a la pasarela de pago desde donde podrá comprar las entradas.

g. Compra de entradas para un evento seleccionado

A esta pantalla el usuario accede tras hacer *click* en el botón de comprar entradas situado al final de cada pantalla de detalles de cada evento.

En ella, el usuario solo debe seleccionar cuantas entradas quiere y el método de pago que usará, ya que al igual que en la actividad de la compra de las entradas para el museo, la información personal se obtiene automáticamente de la base de dato del usuario.

De igual manera, aunque en la imagen se muestre un precio total estático, mientras el usuario aumente o disminuya el número de entradas, ese número ira variando en consonancia con lo seleccionado.

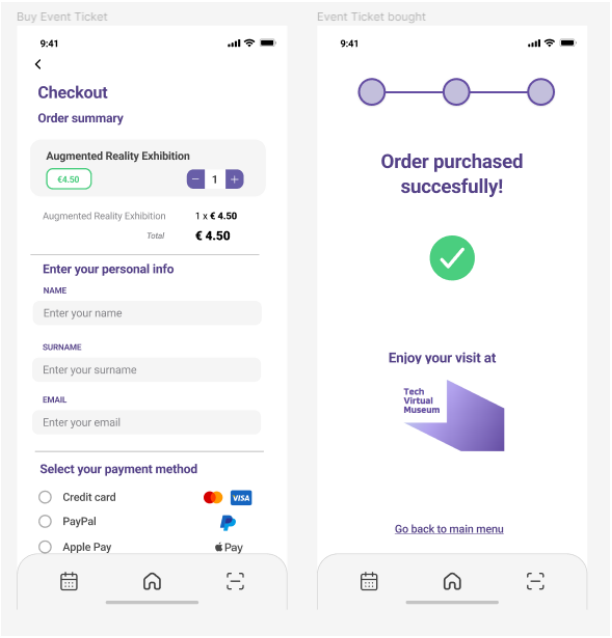


Imagen 26. Pantalla de compra de entrada para un evento

Finalmente, el usuario tras realizar el pago se encontrará con la misma pantalla informativa que en la actividad de la compra de la entrada normal.

h. Escáner QR y detalles del producto

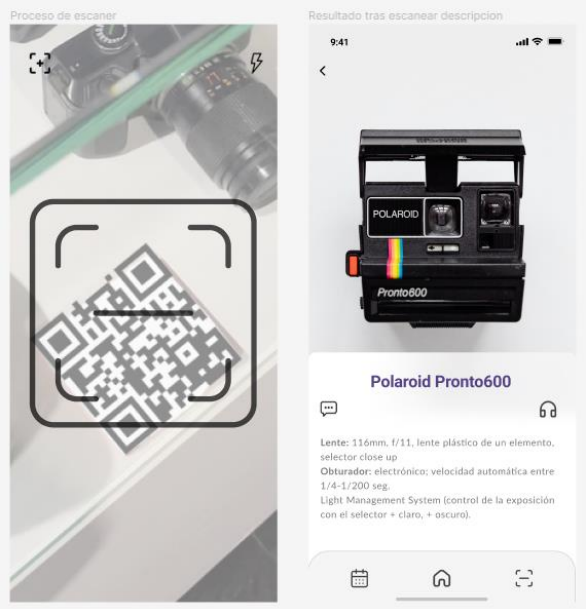


Imagen 27. Escáner de código QR y muestra de los detalles del ítem

Este es el único apartado que, aunque el usuario puede acceder a él en cualquier momento, al igual que al resto de la aplicación, no puede sacar el máximo rendimiento a la funcionalidad ya que los códigos QR que el usuario debe escanear para poder ver la información en referencia a cada objeto, se encuentran dentro del museo.

El escáner tiene por defecto la opción de *autofocus* activada lo que le facilita al usuario usarla, ya que solo debe situar la cámara encima del QR y automáticamente se mostrará la información sobre el objeto escaneado.

Asimismo, dentro de la pantalla de detalles del producto, a parte de una descripción sobre el mismo, tendremos a cada lado un botón con dos funciones extras: el de la derecha con el icono de los auriculares, nos permitirá reproducir un pequeño video explicativo o ilustrativo del ítem; y en el botón que tiene como icono el símbolo de comentarios, le permitirá al usuario visualizar los comentarios que han añadido otro usuario, o por otro lado añadir uno.

i. Creación y muestra de los comentarios. Reproductor de video

Cada ítem que se encuentra en el museo y sea escaneado por el usuario tiene su propio tablón de comentarios, es decir, cada uno tendrá su propia colección en la base de datos en la que se alojen los respectivos comentarios.

Cada visitante que acceda a ese apartado de detalles del ítem, podrá visualizar qué han opinado otros visitantes, además de añadir un comentario si así lo desea, visualizándose instantáneamente debajo del último comentario añadido anteriormente.

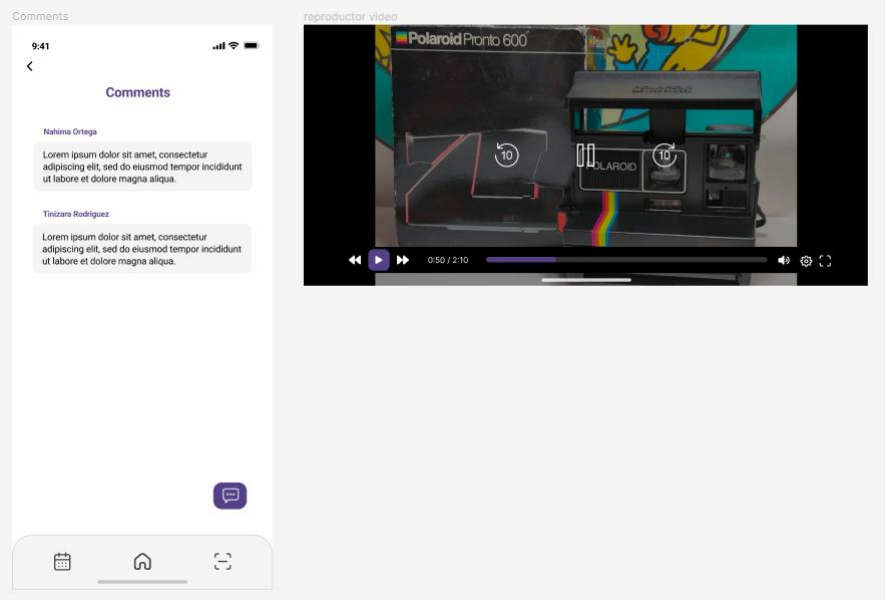


Imagen 28. Creación y visualización de los comentarios y del reproductor de video

4.3 Flow de navegación entre las diferentes actividades

En la imagen inferior, nos encontramos con el flujo final de navegación entre las diferentes pantallas representado empleando Figma. Se representan mediante flechas tanto el origen como el destino que puede seguir el usuario pulsando cada botón que se encuentre en la pantalla o realizando una determinada acción.

Como se puede observar, desde la barra de navegación inferior, se puede navegar a tres diferentes pantallas, comenzando de izquierda a derecha encontramos: el icono de un calendario que nos llevará hasta la vista de la lista de eventos; el icono de una casa, que hace referencia al *home page* y, por último, un icono de escáner, que nos dirige a la pantalla de escanear el QR.

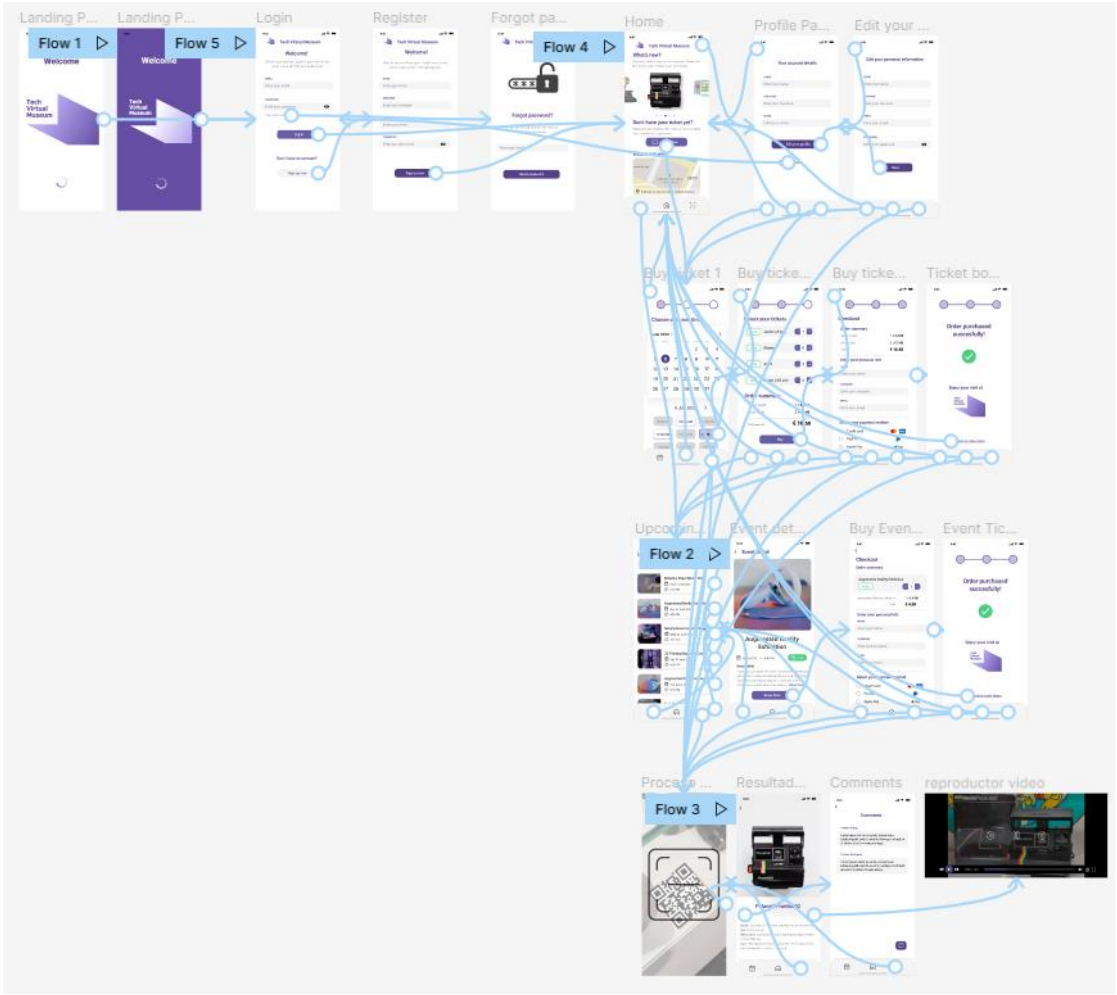


Imagen 29. Navegación entre las diferentes pantallas

5. METODOLOGÍA DE DESARROLLO EMPLEADA

En este apartado, vamos a describir las diferentes fases por las que hemos pasado para crear una primera versión funcional de la aplicación, desde el prototipado hasta el lanzamiento. Describiremos en detalle, también, la metodología empleada, así como las herramientas para la coordinación.

En un principio, pasamos por la fase de análisis y planificación, donde se realiza un análisis de la viabilidad, del público objetivo, etc., y se establecen los objetivos y requisitos de la aplicación. Comenzamos planteando las posibles funcionalidades de la aplicación mediante un *brainstorming*. De aquí, surgieron muchas ideas que tomamos para implementar en la aplicación y otras que fueron descartadas. Por ejemplo, una idea que nos pareció interesante implementar es una visita en realidad aumentada, en la que, mediante la cámara del dispositivo, mostramos al usuario dónde se encuentran los códigos QR que pueden escanear y a qué objeto pertenecen, de tal manera que la experiencia es mucho más inmersiva para los usuarios.

Otras funcionalidades que no considerábamos que pudieran encajar con la aplicación fueron descartadas. Un ejemplo de ellas podría ser contar con un apartado de favoritos donde guardar los elementos que resultaban interesantes al usuario, junto a toda la información de estos. Los elementos que fueran marcados como favoritos mediante un botón, quedarían guardados en esta sección. Esta funcionalidad fue descartada inicialmente, ya que, al ser el museo de pago, es más conveniente no guardar información que se brinda solo si compramos una entrada para acceder a él. De esta manera, evitamos la difusión de la información y de los vídeos o audios que elabora el museo únicamente para sus usuarios.

El tiempo empleado para esta fase de análisis y planificación fue, de aproximadamente, unas cuatro horas, que correspondería a dos horas para cada persona del equipo.

Con las ideas seleccionadas del *brainstorming*, realizamos un análisis de éstas para terminar de definirlas. De aquí, elaboramos la pila del producto (*product backlog*). Cada una de las funcionalidades quedará dividida como una o más historias de usuario, con sus respectivas estimaciones, puntos de historia, prioridad y dependencias. Además, añadimos los criterios de aceptación para, una vez desarrollada la historia, saber si funciona acorde a lo esperado. Con todos estos datos, podremos ordenar la pila del producto. Como sabemos, los elementos en la parte superior del *product backlog*, son aquellos que se encuentran más definidos, con menor riesgo y más urgentes. Explicaremos en apartados posteriores de este informe más detalles acerca de la pila del producto y otros artefactos de Scrum que hemos empleado.

Una vez creamos la pila del producto, pasamos a la fase de diseño y prototipado, donde realizamos los wireframes y mockups, así como, de igual manera, definimos la navegación de la aplicación. No solo diseñamos cómo sería el front-end de la aplicación, sino también el back-end: patrones o librerías que íbamos a emplear, cómo realizar ciertas funcionalidades, etc. El tiempo empleado para esta fase ha sido de, aproximadamente, 15 a 20 horas.

El desarrollo de la aplicación se ha fundamentado en el uso de metodologías ágiles, mediante las cuales, el proceso de desarrollo ha sido incremental y adaptado a las circunstancias y el contexto en el que nos encontrábamos en cada sprint; y en específico la metodología Scrum, cuyos principios son: por un lado, desarrollar el software mediante

iteraciones denominadas *sprints* con una duración no mayor de 30 días, siendo de 15 días en nuestro caso; y por otro lado, son las reuniones entre los miembros del equipo donde se discuten aspectos sobre los requisitos y características del producto, así como para la sincronización del equipo. Los sprints realizados han sido, en total, cuatro. En estos sprints, hemos desarrollado toda la funcionalidad con la que cuenta la aplicación para la versión actual. En términos de horas empleadas, si contamos un total de 40 horas aproximadas por semana entre las dos personas del equipo, esto sumaría un total de 160 horas empleadas para el desarrollo de la aplicación (80 horas por persona).

Con relación al despliegue e instalación, hemos probado la aplicación tanto en simuladores como en dispositivos móviles reales. De esta manera, comprobamos que no haya problemas de compatibilidad o problemas con las versiones o con ciertos módulos que hagan que nuestra aplicación no funcione. Todos los problemas que surgieron fueron solventados eficientemente y se añadieron nuevas características a la aplicación como que, en el caso de iOS, el iPhone realice una breve vibración cuando reconozca un código QR.

Con respecto a la fase de documentación, hemos elaborado un PowerPoint describiendo las funcionalidades básicas de la aplicación, así como esta memoria. Además, se ha descrito el funcionamiento de la app, así como las herramientas empleadas en el ReadMe que se encuentra junto al código en GitHub. La temporización para esta fase ha sido un total de 10 horas por persona, es decir, un total de 20 horas.

En cuanto a las reuniones que hemos tenido los miembros del equipo, hemos llevado a cabo reuniones similares a un *Daily Scrum* (Scrum diario), en las que comentamos, de manera breve y general, cómo llevamos el proyecto, las dificultades que hemos encontrado o las que tengamos actualmente, así como qué funcionalidades hemos desarrollado y cuáles vamos a comenzar a desarrollar. Aunque, inicialmente, el *Daily Scrum* está planteado para llevarse a cabo diariamente con un time-box de quince minutos, nosotras lo hemos separado más en el tiempo, realizándolo varias veces a la semana, pero no diariamente, aunque con un time-box similar.

Otra reunión que solíamos hacer es la de planificación del sprint, para saber qué historias de usuario íbamos a incluir en el nuevo sprint, así como qué tareas íbamos a realizar dentro de cada historia o cómo pensábamos implementarla. De esta manera, lográbamos una buena coordinación en las dos versiones de la aplicación (iOS y Android).

Por otro lado, otras reuniones eran el *Sprint Review*, en el que mostrábamos cómo se encontraba cada aplicación, las funcionalidades que tiene y posibles aspectos de mejora para incorporar al *backlog* ; y el *Sprint Retrospective*, en el que sugeríamos aspectos de mejora como equipo para trabajar más ágilmente, como por ejemplo, nuevas maneras de comunicarnos o la frecuencia con la que lo hacíamos.

6. USO DE JIRA

La coordinación y gestión del trabajo y pilas del producto y del sprint, se realizó mediante Jira Software, ya que es una herramienta bastante potente para la visión general y planificación del proyecto. En ella, nos coordinábamos estableciéndonos, a cada una de nosotras, nuestras respectivas tareas en el *sprint backlog* (se explica en el siguiente apartado), además de documentar todos los pasos y errores abriendo tickets, que posteriormente serían asignados y procesados.

A diferencia de otras herramientas como Trello, aquí hemos podido crear y administrar el *product backlog*, de manera que podemos añadir funcionalidades que nos gustaría implementar, no solo en este *sprint*, sino también en los siguientes.

Por un lado, hemos empleado dentro de Jira un tablero (similar al de Trello) con las tareas a las que se les establece una etiqueta en función de su estado: por hacer, haciendo o terminada (To-Do, Doing y Done). Estas tareas han sido generadas tras un filtrado de todas las ideas que se obtuvieron del primer *brainstorming*, y tras filtrarlas según la capacidad y la programación de los diferentes *sprints* en los que se dividió el desarrollo de la aplicación.

6.1 Product backlog y Sprint backlog

En la metodología Scrum, se entiende como *product backlog* al listado de todas las historias de usuario, mejoras y/o tecnologías que se pretenderán realizar y utilizar durante el proceso de desarrollo de la aplicación. Además, se realiza la estimación de tiempo que se empleará en cada una de ellas y se determina la importancia de cada tarea. Dentro del *product backlog*, nos encontramos con los *product backlog items*, que hacen referencia a cada función, problema o trabajo pendiente que encontramos en la lista de trabajo.

Cuando definimos las funciones, también podemos hablar de las historias de usuario, que se corresponden con la definición, además de las funciones, de las estimaciones en forma de tiempo y prioridad que puede llevar desarrollarlas, diferenciándose a su vez en historias simples, si la carga de trabajo es aceptable; y épicas, en el caso de que se corresponda con una función compleja.

Las historias de usuario, a su vez, están formadas por cuatro grandes bloques que las constituirán, como son: la reparación de errores (cuando se genera un error en alguna parte del desarrollo, se debe generar un *ticket* con toda la información de contexto, y asignarle una prioridad determinada con el fin de solventarlo lo más rápido posible, en el caso de que el error así lo requiera), definición de las funciones (comentadas anteriormente), deuda técnica (hace referencia al momento en el que se comienza a

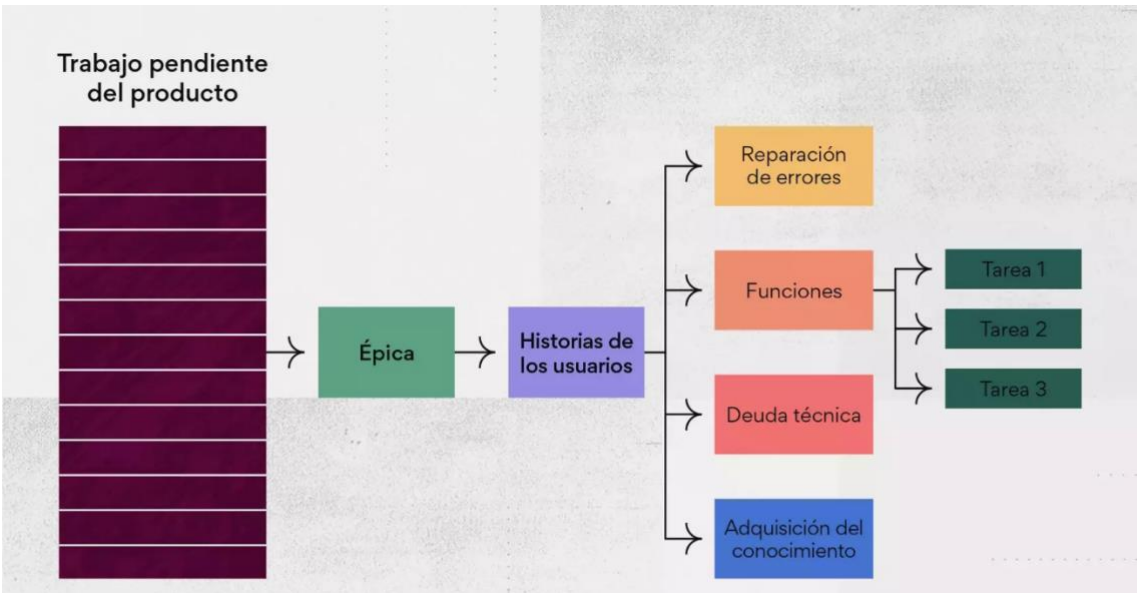


Imagen 30. Esquema ilustrativo sobre la clasificación del product backlog

acumular trabajo técnico debido a errores o a mala planificación temporal y, como consecuencia, se crea un cuello de botella), y la *adquisición del conocimiento* (consiste en acumular toda la información posible con el objetivo de poder cumplir con tareas que se sucederán más adelante en el *product backlog*).

En nuestro caso, hemos seguido los conceptos teóricos que hemos descrito, aunque adaptándolo al proyecto en el que nos encontramos. Scrum no se trata de unas reglas que se deben seguir estrictamente, sino que éstas se pueden adaptar a las necesidades del equipo. Siguiendo esto, hemos adaptado esta definición convencional de historia de usuario. En nuestro caso, por ejemplo, al tratarse de un proyecto nuevo, no hemos empezado a considerar la deuda técnica.

Para la realización del *product backlog*, se siguen los siguientes pasos:

- Crear la hoja de ruta del trabajo. La entendemos como la hoja en la que se define el proceso de cambio que irá siguiendo la aplicación a medida que se avance en los *sprints*, es decir, es una visión al futuro donde se define el estado en el que queremos que la aplicación se encuentre en cada momento.

- Crear un listado con los elementos del *product backlog*

- Asignar prioridades a los elementos anteriormente creados, priorizando las historias de usuario con mayor valor o más urgentes; es decir, tareas más necesarias para el usuario si la aplicación se publicara en ese momento, por ejemplo. En un principio, los elementos con más prioridad serán aquellos que nos permitan tener un mínimo viable que pueda ser usado por los clientes.

- Actualizarlo de forma regular, ya sea añadiendo tareas o eliminándolas si se han finalizado, además de modificar las prioridades según se vaya desarrollando.

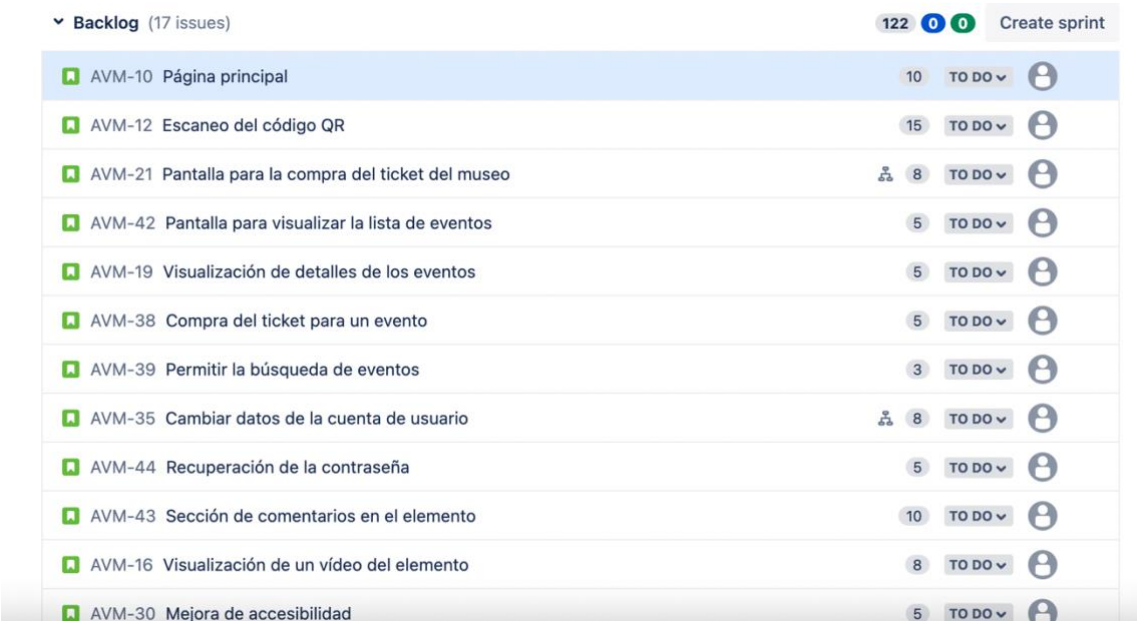


Imagen 31: Product Backlog del proyecto

En cuanto al *sprint backlog*, podemos definirlo como el subconjunto de elementos tomados del *product backlog* para realizarlas durante este sprint, así como el objetivo del sprint y el trabajo para convertirlos en un incremento con valor. En este punto, se asigna quién se encarga de cada tarea y el tiempo de trabajo (denominado *Story Point Estimate*).

En el caso de que alguna de las tareas no se pueda terminar en el sprint, se añadirán al siguiente sprint.

Para crear el *sprint backlog*, hemos tenido en cuenta que la duración de cada sprint es de dos semanas, en nuestro caso, y, como consecuencia, el volumen de trabajo debe adecuarse a esta estimación. Es por ello, por lo que, en las historias de usuario, se añade una estimación de cada una de las tareas que se crean.

En total, las horas dedicadas por semana por cada una de las integrantes del equipo ha sido de, aproximadamente, 20 horas semanales, sumando unas 40 horas semanales en total empleadas en el proyecto. Es importante tener esto (la capacidad del equipo) en cuenta también para determinar las historias de usuario que vamos a incluir en el sprint.

Por lo tanto, para crear el sprint backlog, hemos tomado en consideración el tamaño de las historias de usuario, así como el tamaño del sprint, la capacidad del equipo, la prioridad de las historias de usuario, así como la dependencia entre ellas.

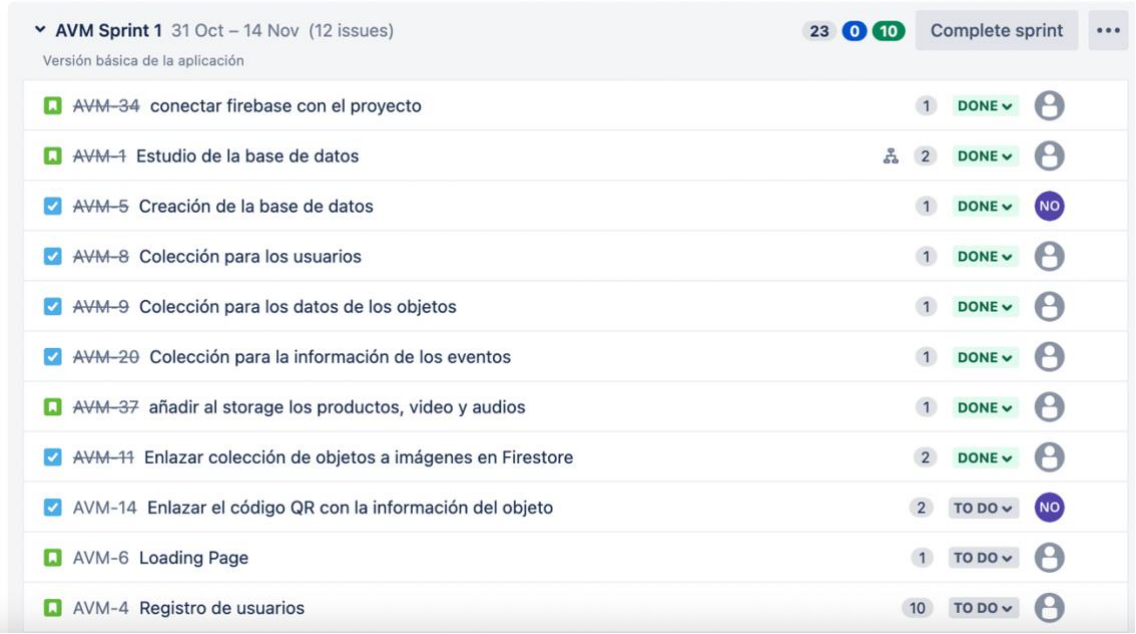


Imagen 32: Sprint Backlog

7. ARQUITECTURA Y PATRONES DE DISEÑO

Los patrones de diseño son soluciones generales, reutilizables y aplicables a problemas comunes que nos encontramos cuando desarrollamos software. Con el uso de estos patrones, se pueden corregir algunos problemas comunes de una manera segura y estable, independientemente del lenguaje de programación que estemos utilizando en ese momento.

Existen tres categorías de patrones:

- Patrones creacionales, que se corresponden con aquellos patrones que proporcionan formas flexibles de crear objetos, minimizando las dependencias entre las clases, pero maximizando la reutilización de código. Existen 5 patrones que son: *Abstract Factory*, *Builder*, *Factory Method*, *Prototype* y *Singleton*; siendo

este último junto al *Builder* y el *Factory Method* los más usados en Android. En cambio, para el desarrollo de aplicaciones iOS, el más utilizado es el *Builder.*
El patrón de diseño Builder consiste en la creación de objetos con cierta complejidad empleando objetos más simples paso a paso. Un ejemplo del empleo de este patrón de diseño en el código realizado puede ser al crear la lista de eventos, ya que se crea la lista de eventos compuesta de celdas que tienen su propio constructor.

- Patrones estructurales, son aquellos que ayudan a organizar las clases de manera eficiente, haciendo que las relaciones entre ellas sean mínimas y directas. En este caso, son 7 los que componen los patrones estructurales: *Adapter*, *Bridge*, *Composite*, Decorator, *Facade*, *Flyweight* y *Proxy*. Los más destacados y usados en el desarrollo de las aplicaciones tanto Android como para iOS son: *Adapter*, *Decorator* y *Facade*
Adapter es un patrón de diseño que permite que los objetos con interfaces incompatibles trabajen juntos, por ejemplo, cuando queremos utilizar una clase de terceros, pero su interfaz no coincide con el resto del código de la aplicación.
Decorator permite proporcionar nuevas funcionalidades a un objeto envolviéndolo en *wrappers* útiles.

- Patrones de comportamiento, responsables de la distribución de manera eficiente y segura de los comportamientos entre los objetos del software. Los patrones que lo componen son 10*: Chain of Responsability*, *Command*, *Iterator*, *Mediator*, *Memento*, *Observer*, *State*, *Strategy*, *Template Method*. En este caso para el desarrollo de aplicaciones para Android, los más utilizados son *Observer* e *Iterator*; y para las aplicaciones en iOS son: *Template Method* y *Decorator.*
Template Method se emplea cuando delegamos la responsabilidad de algunos pasos a subclases, de manera que podemos emplearlo cuando tenemos varias clases responsables de acciones similares, de manera que podemos reutilizar código y centralizar esa acción similar en una función o clase separada.

En iOS, hemos empleado la Arquitectura Modelo-Vista-Controlador (MVC). De esta manera, tenemos tres responsabilidades separadas y conectadas entre sí, cumpliendo, así, el Principio de responsabilidad única (*Single Responsibility Principle*).

Por un lado, el modelo contiene las clases que vamos a emplear o a representar en la vista de nuestra aplicación. La clase contiene la estructura correspondiente a un componente. Por ejemplo, tenemos la clase "Evento" (que especifica los atributos que debe tener un evento, como fecha, hora, título, etc.), o la clase Celda (empleada para representar las celdas de una lista). Para el modelo, se emplea el lenguaje Swift.

Por otro lado, la vista se corresponde con la representación visual de la aplicación. La vista sirve al usuario para comunicarse con la aplicación. La vista se ha realizado empleando *UIView*, siendo de ayuda el uso de Storyboards para agilizar el proceso de creación de las vistas.

Por último, el controlador (*UIViewController*) va a encargarse de realizar la coordinación entre el modelo y la vista, de manera que, si hay alguna actualización en los valores del modelo, esto se vea reflejado en la vista. Además, va a encargarse de ejecutar las tareas correspondientes a determinados *triggers* o detonantes desde la vista, como cuando se presiona un botón, se selecciona una celda, se realiza una búsqueda en la barra de búsqueda, etc.

En cuanto al acceso a Firebase, se ha encapsulado la lógica de este servicio en un subconjunto de clases, que se encuentra bajo la carpeta *Services*, de manera que proporciona una interfaz para el resto de las clases para interactuar con este módulo. De esta manera, facilitamos la reutilización y escalabilidad de la lógica del servicio. Estaríamos, además, aplicando el patrón *Template Method*, ya que describe el esqueleto de los métodos necesarios para interactuar con Firebase (como escritura o lectura de un documento o colección).

En Android se ha hecho uso de tres clases adapter (*AdapterComments, AdapterEventos, AdapterProduct*) y tres clases Modales (*CommentModal, EventosModal, DataModal*), las cuales en su código, nos encontramos con una clase objeto que se utilizará para poder almacenar y trabajar con la información relacionada con los comentarios, eventos y los productos de la aplicación y donde cada uno tiene una serie de atributos que hacen referencia al nombre, precio, descripcion, etc, dependiendo de con cual clase estemos trabajando. Asimismo, en ella, debemos integrar dos constructores donde uno debe estar vacío para Firebase debido a que existen algunos métodos que asi lo requieren, y por otro lado, otro constructor al que si se le pasen los parámetros que se inicializaran con los valores que se obtienen de Firestore mediante la clase Adapter.

Las clases Adapter son utilizadas para *adaptar* la lista de objetos que se obtienen de las clases Modales (objetos cuyos valores se encuentran en la base de datos) a una lista de elementos visuales en la aplicación, permitiendo asi que se muestren de una manera más ordenada y estética, haciendo uso asi del patrón estructural de Adapter

8. HERRAMIENTAS USADAS

La primera herramienta que se ha utilizado ha sido Figma, que nos ha ayudado, como se ha podido comprobar a lo largo del informe, a crear los prototipos que seguimos para crear la interfaz de usuario.

Un prototipo sirve tanto para que el cliente se pueda imaginar cómo será el producto final, como para facilitar la labor de los desarrolladores. Son la guía que usarán los programadores cuando estén desarrollando, con el objetivo de crear una interfaz de usuario lo más parecida posible. En los prototipos, no solo se puede observar la apariencia de las vistas, sino que también se puede ver el flujo de navegación, de manera que, al programar, también se pueda tener más claro el funcionamiento esperado.

De igual manera, los prototipos pueden ser de alta fidelidad o de baja fidelidad. El prototipo de baja fidelidad es aquel donde se representan la estructura básica que seguirá la aplicación, así como la localización de botones, imágenes, iconos, etc. En nuestra aplicación, se corresponden con los bocetos (realizados a mano alzada) y *wireframes* mostrados anteriormente, que han sido creados en Figma. En los prototipos de alta fidelidad, se añade todos aquellos elementos que se correspondan con la identidad visual de la empresa (en este caso el museo), como pueden ser las diferentes fuentes, las imágenes y texto. En este último, se genera un prototipo interactivo donde se define cual será la ruta de navegación entre las diferentes pantallas, como comentábamos, con el objetivo de acercarnos lo máximo posible a lo que conformará la aplicación final.

Para la coordinación del equipo y para aplicar la metodología Scrum, hemos empleado Jira Software, además de Confluence.

Jira nos ha servido como herramienta para la gestión de proyectos y organización de los miembros del equipo. Como comentábamos en el apartado de metodologías ágiles, esto lo hacemos mediante la administración de la pila del producto, la pila del sprint y la asignación de tareas. Por otro lado, cuando añadimos Confluence, estamos agregando un espacio donde podemos organizar todas las ideas y tareas, así como la posible documentación que ambas necesitásemos. De esta manera, evitamos el uso de unidades compartidas en otras herramientas, ya que todo se encuentra en el mismo lugar.

Para poder visualizar vídeos dentro de la propia aplicación, se ha utilizado la API de Youtube. Una API es un conjunto de protocolos, herramientas y definiciones que se utilizan a la hora de desarrollar aplicaciones, y que permiten que diferentes sistemas se puedan comunicar entre ellos, para realizar alguna función determinada en una aplicación, como puede ser el caso de reproducir vídeos de Youtube o usar alguna aplicación de mapas.

Con la API propia de Youtube, se nos permite incorporar todas las funciones que podríamos ejecutar dentro de la propia web o aplicación de Youtube, pero en su lugar, se ejecutarían dentro de la aplicación, sin necesidad de redirigir al usuario a la aplicación. Para conseguirlo se debe obtener y activar la API KEY de Youtube desde la consola de desarrolladores de Google e introducirla en los archivos de configuración de la aplicación. Además, en el caso de iOS, accedemos a los vídeos empleando el ID del vídeo, no la URL.

Por otro lado, hemos trabajado con las aplicaciones de mapas Google Maps, en el caso de Android, y Apple Maps, en el caso de iOS. Hemos tenido que interactuar con estas aplicaciones para, por una parte, mostrar el mapa directamente en la aplicación y, por otra, hacer que, al redirigir al usuario a la aplicación, se le muestren los detalles de la ubicación del museo correctamente.

La última herramienta utilizada se corresponde con la plataforma de Firebase. En ella, el desarrollador se puede encontrar con diversos módulos o con diferentes herramientas, que corresponderían a diferentes partes de desarrollo de una aplicación. Firebase ofrece herramientas que pueden ayudar al desarrollo, herramientas de crecimiento y monetización, o herramientas de análisis de datos de la aplicación.

Entre las herramientas que nos ofrece Firebase, podemos encontrarnos con Firebase Authentication, Firebase Firestore y Firebase Storage, entre otras, aunque nos centraremos en las mencionadas, ya que se corresponden con las utilizadas en el desarrollo de esta aplicación.

En las herramientas que nos proporciona, correspondientes al crecimiento, se trabajan la gestión y captación de usuarios, destacando Firebase Authentication y Firebase Cloud Messaging. Es con esta última con la que se permite enviar notificaciones al usuario para informarles de ciertas actividades como, por ejemplo, en aplicaciones de mensajería instantáneas, o la llegada de un nuevo mensaje por parte de algún contacto.

Con las herramientas de monetización, lo que se busca es generar ingresos, siendo la principal fuente la publicidad, mostrándoles a los usuarios, anuncios que se basen en sus intereses en función de sus búsquedas o navegación. Es por ello por lo que Firebase cuenta con AdMob, donde permite a la empresa generar ingresos mostrando anuncios

junto con el contenido de la aplicación como sucede cuando navegamos por páginas web, donde encontramos banners publicitarios a ambos lados o en el medio del contenido de la página. Ofrece la posibilidad de poder personalizarlos tanto estéticamente para que continue la línea de diseño que se trabaja como a nivel de relevancia para el usuario.

El último modulo que forma Firebase es el de análisis de los resultados con el fin de tomar decisiones sobre el marketing a seguir, y sobre las funcionalidades que se podrían añadir o eliminar dependiendo de las interacciones de los usuarios. Destaca Firebase Analytics, donde obtendremos informes en forma de estadísticas y gráficos, donde se podrá visualizar la forma en la que los usuarios interactúan con la aplicación, además de poder ver cómo funciona a nivel de marketing.

En el próximo apartado se explicará en más detalles las tres utilidades de Firebase que se han utilizado en la aplicación (Authentication, Firestore y Storage).

8.1 Android

Para el desarrollo de la parte de Android, se ha utilizado Android Studio como entorno de desarrollo (se entiende como entorno de desarrollo, el software que le brinda al programador todas las herramientas que son necesarias para el desarrollo de una aplicación).

La gran ventaja de usar Android Studio es que permite la interoperabilidad de los principales lenguajes de programación en las aplicaciones para móviles Android como son Kotlin y Java.

Esta ventaja se ha usado para el desarrollo de la aplicación del museo, ya que, aunque en la gran mayoría se ha hecho uso de Kotlin, en las clases *Adapter* se ha recurrido a usar Java por la facilidad, aunque este código se podría transcribir con el fin de generar una aplicación desarrollada en su totalidad en Kotlin.

Otro punto que destacar de Android Studio, es el apartado de generar el diseño de la vista de la aplicación. Esto se debe a que nos permite incluir diferentes componentes como botones, campos de texto, imágenes o recursos multimedia, arrastrando desde la librería de componentes y soltando en la mesa de trabajo.

Además, otra ventaja que nos ofrece es que se puede visualizar en tiempo real cualquier cambio realizado tanto a nivel de interfaz de usuario como a nivel de código, gracias a que podemos instalar un emulador que se ejecute en una ventana emergente, evitando, así, tomar mucho espacio de la pantalla y ocultar parte del código.

8.2 iOS

Para el desarrollo de la aplicación para sistemas iOS, el entorno utilizado ha sido Xcode, que cuenta con la capacidad de integrarse con el lenguaje Swift, que ha sido el lenguaje utilizado para el desarrollo. Hemos optado por emplear Swift en lugar de Objective-C, debido a la potencia que tiene Swift al estar más optimizado. Además de su uso incremental por parte de los desarrolladores, Swift, a simple vista, nos permite desarrollar un código más simple y legible.

En conjunto con Swift, se hace uso de SwiftUI, mediante el cual podemos programar las vistas. Aunque la gran parte de las vistas han sido desarrolladas empleando Storyboards, ha sido necesario emplear SwiftUI en momentos puntuales. La ventaja de usar Storyboards es que podemos crear la vista arrastrando componentes de la librería de componentes a las vistas, haciendo más sencilla la labor de desarrollo. Además, se puede apreciar no solo el diseño de las vistas sino la interacción entre éstas, por lo que es especialmente útil. Algunas de las vistas que han tenido que ser desarrolladas empleando SwiftUI es, por ejemplo, la vista de escanear el código QR.

Para el escaneo del código QR, se ha empleado AVFoundation, que es un framework (se define como una planilla que ofrece al desarrollador una estructura base a partir de la que, después de modificar algunos aspectos, poder desarrollar su propio software) que nos permite realizar diferentes acciones como acceder a la cámara para poder grabar un vídeo, reproducir y visualizar imágenes o vídeos, crear animaciones, etc. En nuestro caso, se utiliza, como comentábamos, para poder implementar el escáner QR con el que poder acceder a los detalles de los productos. Para ello, ha sido necesario añadir un aviso de permiso para acceder a la cámara del dispositivo al iniciar la aplicación.

9. FIREBASE COMO BASE DE DATOS

Firebase es una plataforma de Google que facilita ciertas tareas a la hora de desarrollar aplicaciones de escritorio o móviles, así como páginas web, ya que se puede integrar en diferentes tecnologías.

Se diferencia en tres grandes módulos, que se encargan de diferentes apartados de una aplicación, como son: por un lado el módulo referente a las tareas de compilación, donde podremos desarrollar una base de datos no relacional sin necesidad de tener que administrar servidores; el segundo modulo hace referencia a la parte de lanzamiento y supervisión de la aplicación; y el ultimo a la interacción del usuario final con ella, analizando los diferentes datos de inicios de sesión (se puede contar como la participación), o como se desenvuelve el usuario por la navegación del usuario.

En nuestro caso, se ha hecho uso de Firebase Firestore como base de datos para almacenar y gestionar toda la información en referencia a los objetos, comentarios, eventos y recursos visuales, además de administrar los registros e inicios de sesión.

De igual manera, también se ha hecho uso de Firebase Authentication, para el proceso de validación de las credenciales al registrarse o iniciar sesión un usuario.

9.1 Firestore

Firestore es un base de datos no relacional, es decir, entre otras características, la información se guardará en documentos donde ya se declaren sus atributos en vez de en tablas, como se realiza en las bases de datos relacionales. Que se empleen documentos ayuda a la posible escalabilidad que pueda tener la aplicación, debido a que está diseñada para soportar grades volúmenes de datos, facilitando la tarea de modificar, añadir o eliminar campos, además de ser capaz de trabajar sin conexión. Si en algún momento el usuario pierde la conexión, puede seguir accediendo a cierta información,

dado que la información se guarda en la caché. Esa información se sincronizaría con la que se encuentra en la base de datos una vez se recupera la conexión.

En la siguiente imagen podemos ver un ejemplo de las colecciones que se han creado para que la aplicación opere correctamente. Las colecciones cuyos documentos son creados por las acciones del usuario son: *eventoComprado*, ya que se crea después de que el usuario haya comprado una entrada para un evento y se guarda el id del usuario, así como el nombre, fecha y precio del evento que ha comprado; y *users,* puesto que se crea cuando el usuario se registra por primera vez en la aplicación.

Por otro lado, las colecciones cuyos documentos ya se encuentran creados son las que necesita la aplicación para poder mostrar información de los eventos o de los productos como son: *eventos,* almacena la información de cada evento, como nombre, fecha, hora, precio y descripción de cada evento; *products,* almacena toda la información de los productos; y *slider,* que guarda en link a las imágenes (almacenadas en Cloud Storage) que se muestran en la pantalla de inicio.

Con respecto a lo anterior, en cuanto a la implementación del escáner del código QR, cada código QR se ha generado a partir de una ristra de caracteres que identifica al objeto correspondiente. Esta *string* es la misma que el ID del documento que contiene toda la información del objeto. De esta manera, cuando escaneamos el código QR, obtenemos el ID del documento que debemos leer de Firestore para extraer la información del objeto.

Asimismo, existe una colección que ya está creada y se actualiza después de las acciones de los usuarios, y esta es la que almacena los comentarios: *comments.* Esta se trata de la colección más compleja, debido a que se encarga de almacenar todos los comentarios que se muestran en la aplicación. Dentro de la colección de comentarios, nos encontramos documentos cuyo ID coincide con los identificadores de cada objeto, para poder distinguir qué comentarios pertenecen a qué objeto. Cada uno de estos documentos, contendrá, a su vez, una subcolección con todos los comentarios realizados por los usuarios. De esta manera, separamos las secciones de comentarios de los diferentes objetos en documentos independientes.

De igual manera, la colección *users,* se actualiza cuando el usuario en cuestión modifica alguno de sus datos, en el apartado de edición de la información de perfil, comentado en los apartados anteriores.

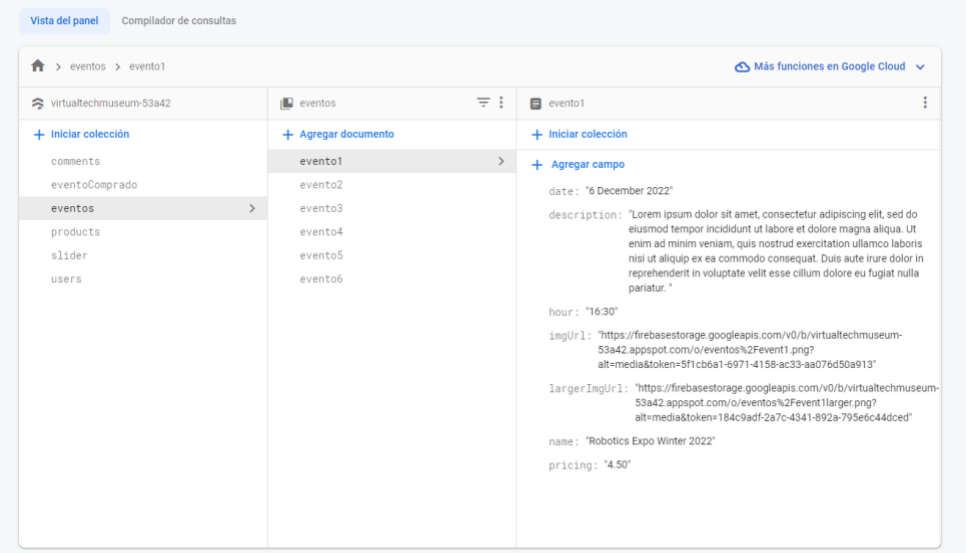


Imagen 33: Panel de Firestore donde se pueden ver las distintas colecciones que conforman la base de datos

9.2 Auth

Haciendo uso de Firebase Authentication, Firebase permite diferentes formas de autenticación de los usuarios. En nuestro caso, solo hemos hecho uso de la autenticación mediante correo electrónico y contraseña, tal y como se muestra en la imagen inferior.

Esta información además de guardarse en el propio Authentication, junto con la fecha de creación y de acceso, así como el UID único para cada usuario, se guardará en un base de datos en Firestore que se llama *users,* que va a guardar información relacionada con el usuario como nombre y apellidos. Cuando el usuario inicia sesión, se produce una llamada a Firebase, donde se busca el UID del usuario y se compara con el del usuario que está intentando iniciar sesión, con el fin de comprobar que está registrado en la base de datos. En caso de que no se encuentre, se devolverá una respuesta, en forma de mensaje de error. En cambio, si existe y los datos introducidos corresponden con la información que se encuentra en la base de datos, el usuario puede iniciar sesión de forma correcta, accediendo al contenido de la aplicación.

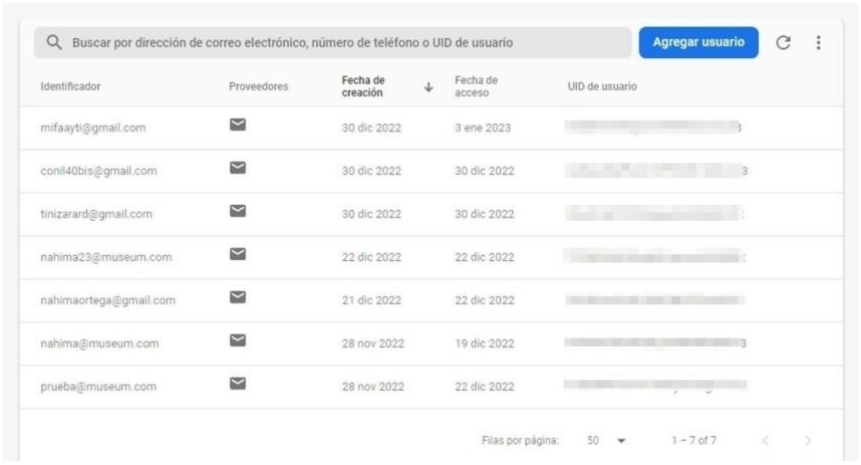


Imagen 34. Firebase Authentication con los diferentes usuarios que se han registrado

9.3 Cloud Storage

La funcionalidad principal de Cloud Storage es almacenar los recursos visuales que se usaran en la aplicación, como las imágenes. En nuestro caso, tenemos tres directorios (eventos, productos y *slider*), donde se almacenan las imágenes que se van a visualizar de los productos, eventos u objetos más escaneados (en el carrousel de imágenes de la pantalla inicial). Desde Firestore, se accede a las imágenes almacenadas aquí mediante la URL.

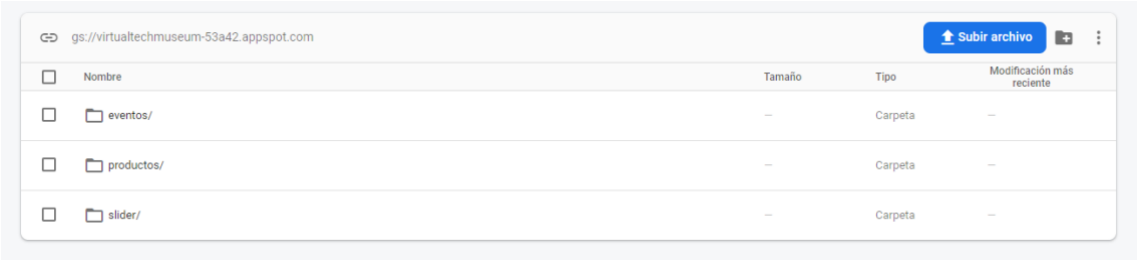


Imagen 35. Storage donde se guardan los diferentes recursos visuales

10. FUNCIONALIDADES IMPLEMENTADAS

En la pantalla de inicio, a la que el usuario llega tras haberse registrado o haber iniciado sesión, nos encontramos, en primer lugar, con un *slider,* que mostrará una serie de imágenes que se corresponden con los productos más escaneados. El *slider* se desplaza de una imagen a otra de forma automática, tras pasar un pequeño intervalo de tiempo, aunque el usuario también puede navegar entre las imágenes deslizando el dedo hacia izquierda o derecha.

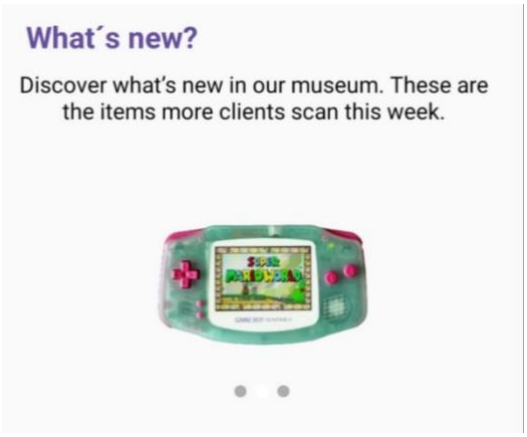


Imagen 36. Slider ubicado en la pantalla de inicio

De igual manera, y bajando un poco más en la pantalla, podemos visualizar un mapa. Si pulsamos en él, nos llevará a la respectiva aplicación de mapas del sistema, es decir, en los móviles Android se corresponde con Google Maps, y en iOS, con Apple Maps. En iOS, el mapa será interactivo, es decir, podemos hacer el movimiento de expandir o reducir dependiendo de si queremos agrandar o disminuir el zoom del mapa, así como arrastrar para desplazarnos en él.

Si continuamos en la pantalla de inicio, también tenemos la opción de comprar las entradas para acudir al museo.

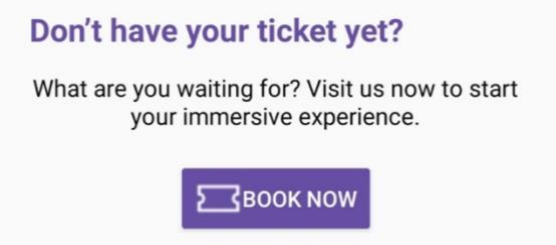


Imagen 37. Botón de compra de entradas

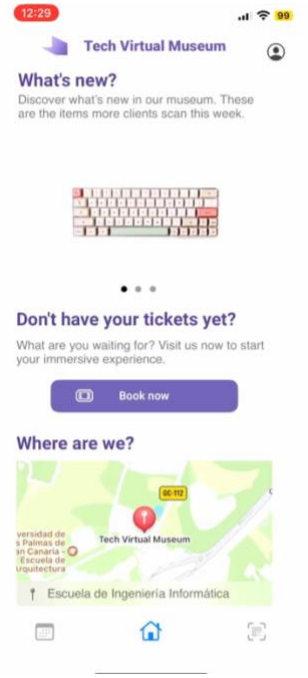


Imagen 39: Pantalla inicial en iOS

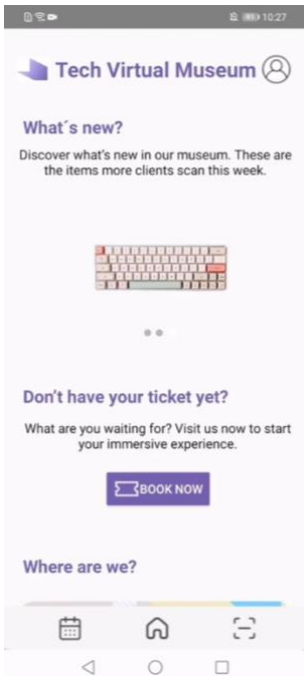


Imagen 38: Pantalla inicial en Android

La pantalla de compra de las entradas está compuesta por tres pantallas, las cuales se dividen en:

- En la primera pantalla, nos encontramos con un calendario dinámico. Partimos del día actual, con un calendario en la parte superior y una tabla con botones de compra de entrada sin haber seleccionado correctamente la fecha y hora de la visita. con las horas a las que se puede realizar la visita al museo. Mediante el calendario, podemos ir avanzando a través de los diferentes días. En cuanto a robustez en este apartado (tolerancia a posibles fallos), no se permite seleccionar un día menor al actual, ni se puede pasar a la siguiente fase

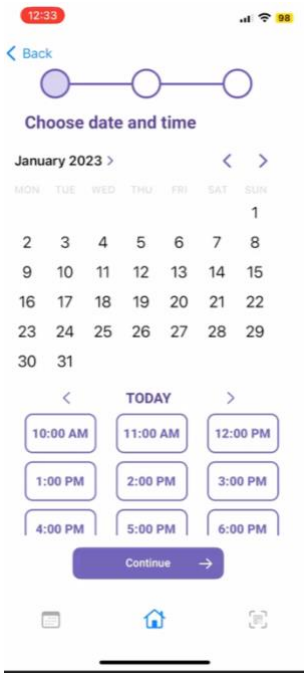


Imagen 41: Primera pantalla de compra en iOS

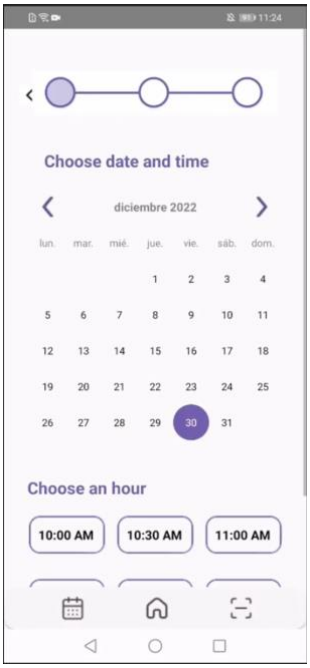


Imagen 40: Primera pantalla de compra en Android

- En la siguiente pantalla, nos encontramos los botones de selección de entradas, que podemos seleccionar en base a las edades de las personas que van a realizar la visita. Cada tarjeta de precio está formada por dos botones que aumentarán o disminuirán el número de entradas (*stepper*). A la misma vez que se incrementa o decrementa el número de entrada, se actualiza de forma simultánea el precio total de la compra, así como el número de entradas de cada tipo. En este paso, no se puede acceder a la siguiente pantalla si no hay entradas para comprar (la suma total de las entradas sea cero).

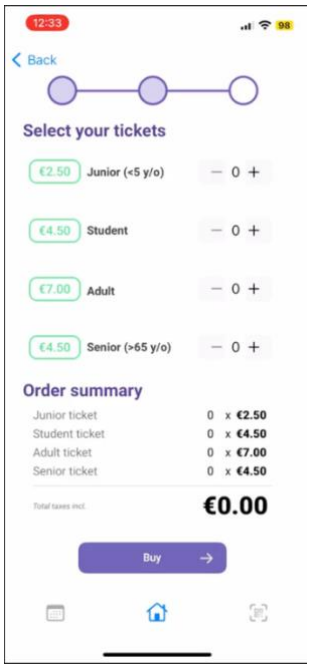


Imagen 43: Segunda pantalla de compra en iOS

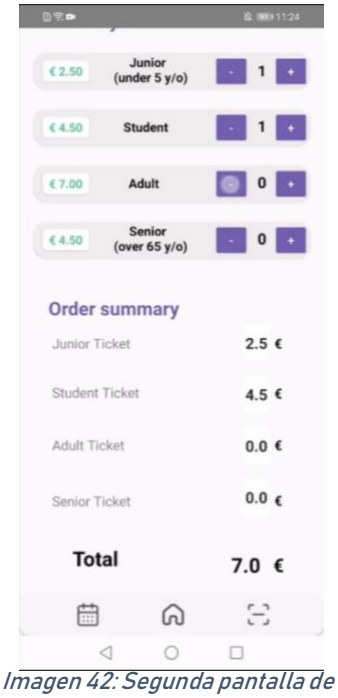


Imagen 42: Segunda pantalla de compra en Android

- En la tercera pantalla, se muestra un pequeño sumario de las entradas que se han seleccionado, así como el número de entradas, y el precio total que se paga por cada una de ellas. También nos encontraremos con un bloque donde el usuario puede visualizar su información personal (nombre, apellidos y email), que se obtiene de la información con la que se registró, aunque se puede modificar en el caso de que así el usuario lo desee. Estos datos se corresponden con los de la persona que efectúa el pago de las entradas, con la idea de que la información del pago sea, posteriormente, enviada al correo que indica. Además, se tiene un conjunto de botones para la selección del método de pago. No se podrá acceder a la siguiente pantalla si no se selecciona correctamente el método de pago.

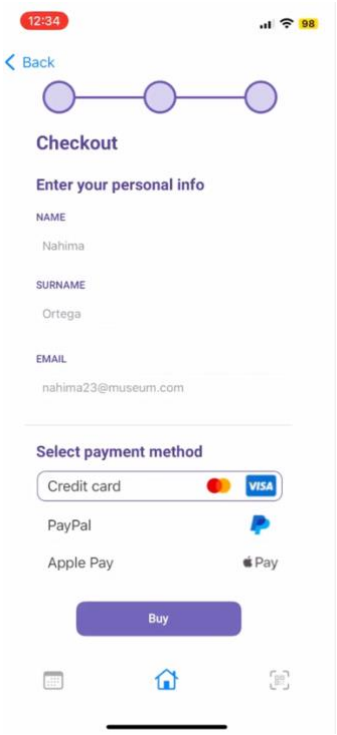


Imagen 44: Tercera pantalla de compra en iOS

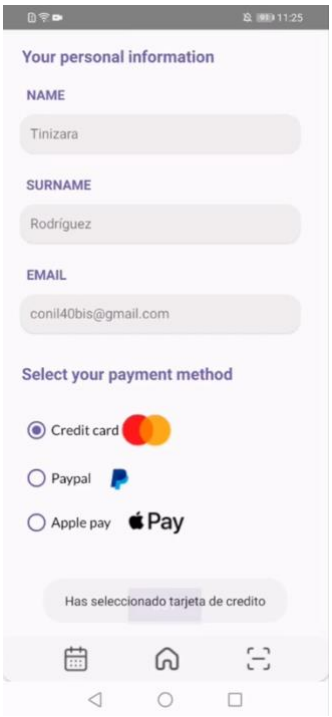


Imagen 45: Tercera pantalla de compra en Android

- Por último, y tras haber realizado de forma correcta el pago, se muestra una pantalla donde se le informa al usuario que la compra ha sido exitosa y que puede volver a la pantalla de inicio.

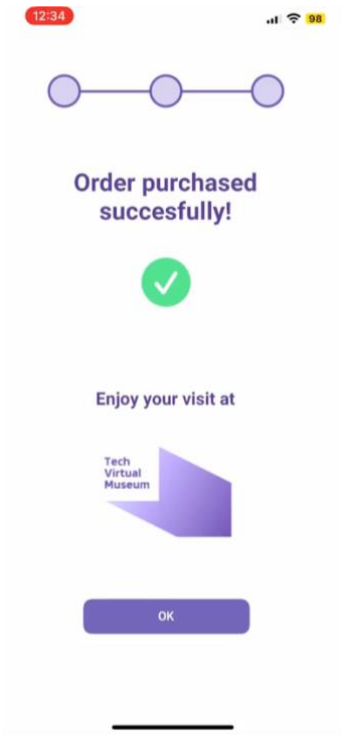


Imagen 46: Pantalla final de compra en iOS

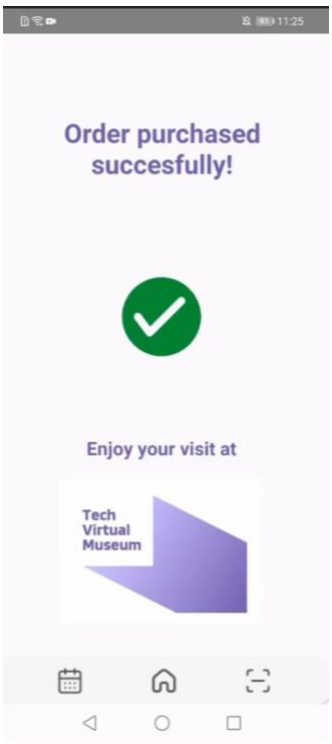
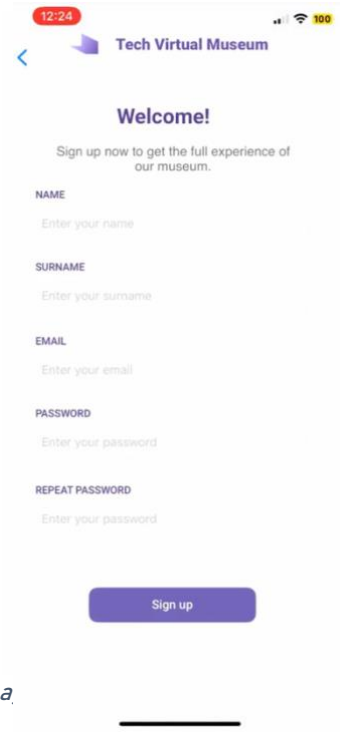


Imagen 47: Pantalla final de compra en Android

Otra de las funcionalidades que se ha desarrollado en esta aplicación, son utilidades principales para el usuario como son los formularios de inicio y cierre de sesión y de registro del usuario. Estos campos de texto son validados y gestionados por la función de



Ima

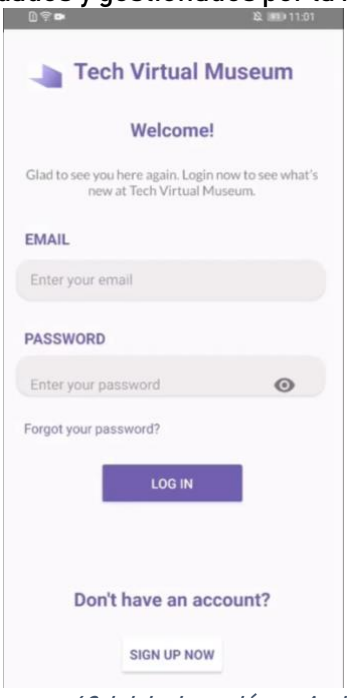


Imagen 48: Inicio de sesión en Android

Firebase Authentication, que se ha explicado anteriormente. Además, se hace uso de expresiones regulares como patrón para comprobar que los correos introducidos y las contraseñas creadas cumplen con las medidas para que sean contraseñas seguras y correos reales. En el caso de que sean detectados campos no válidos, será notificado al usuario mediante alertas para su corrección.

Otra posibilidad que tiene el usuario con sus datos es la opción de recuperar contraseña en el caso de que se le haya olvidado. Para ello, se debe introducir el correo electrónico con el que se haya registrado y, automáticamente, Firebase, enviará un correo al usuario con un enlace mediante el cual puede resetear su contraseña. Esta función también es gestionada Firebase Authentication. En nuestro caso, hemos cambiado ciertos parámetros en Firebase Auth para que el correo que se le envíe al usuario sea más descriptivo, indicando el nombre del museo, por ejemplo. En el caso de que el correo no se encuentre registrado, se le avisa al usuario mediante una alerta.

Imagen 52: Olvido de contraseña en iOS

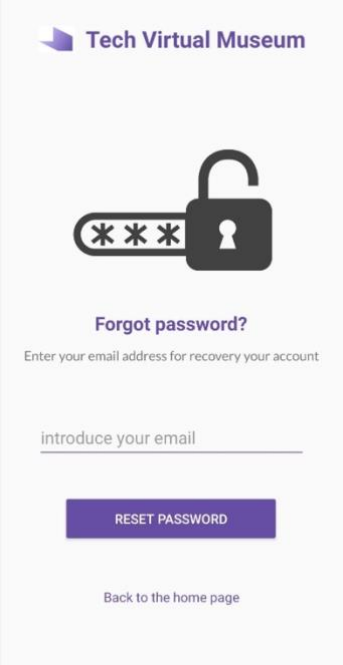


Imagen 53: Olvido de contraseña en Android

Otra opción que puede realizar el usuario es editar sus datos, tanto nombre y apellidos como correo y contraseña con los que se registró, mediante un formulario. Estos datos se actualizarán, tanto en su respectiva colección de usuarios de Firebase Firestore, como los datos que están guardados en Firebase Authentication. Al principio, cuando se accede a la página de editar, se muestran los datos actuales de la cuenta, sobre los cuales el usuario puede comenzar a realizar modificaciones.

Además, puede visualizar su nombre y apellidos, así como su correo, al pulsar el icono que se encuentra arriba a la derecha en la página principal. Desde aquí, puede cerrar sesión también.

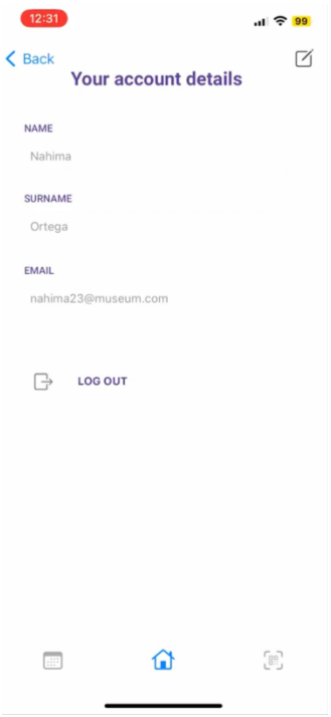


Imagen 57: Ver datos en iOS

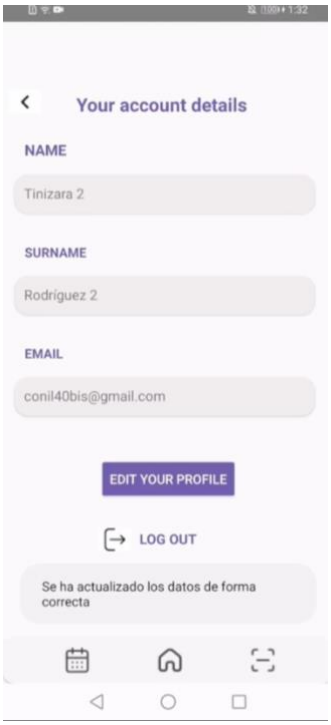


Imagen 56: Ver datos en Android

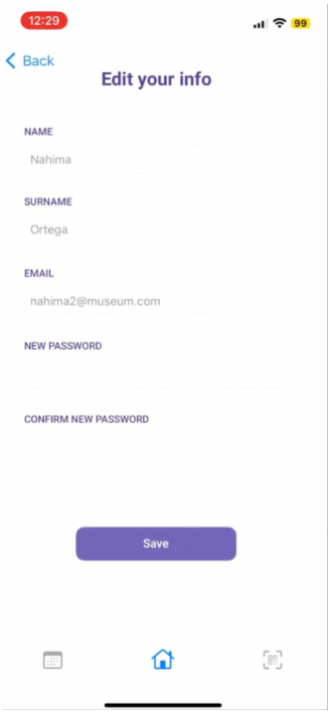


Imagen 54: Editar datos en iOS

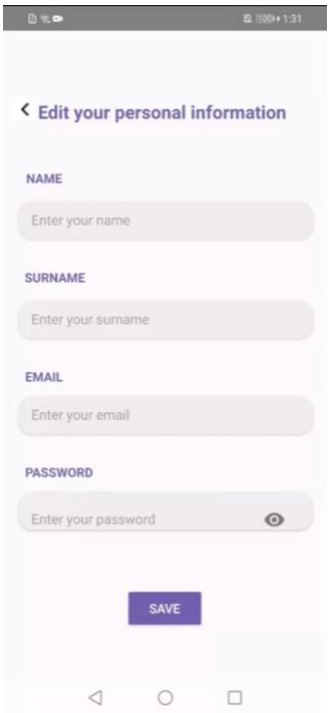


Imagen 55: Editar datos en Android

A continuación, en la parte inferior de las pantallas, el usuario siempre se encontrará con una barra de navegación formada por 3 iconos que representan las funcionalidades principales como son, en primer lugar, la pantalla donde podremos visualizar y comprar entradas para acudir a una charla o evento; el segundo, que nos dirige a la pantalla de inicio; y el último, que es el escáner QR que tiene implementado la propia aplicación.

Ilustración 1. Barra de navegación

Si nos dirigimos a la barra de navegación de la aplicación, encontramos el primer botón que nos dirigirá a la pantalla de los eventos, y es en ella donde encontramos las siguientes funcionalidades:

- Barra de búsqueda, donde el usuario podrá introducir un término y se realizará una búsqueda en los elementos mostrados para mostrar los resultados que contengan ese término introducido.

- Lista con todos los eventos que se encuentran en la base de datos, ordenados por fecha, de más cercana a más lejana. Al hacer *click* en alguno de ellos, accederemos a su pantalla de detalles donde se podrá visualizar información sobre él, así como la fecha y hora, además de su correspondiente precio. Asimismo, tenemos un botón que nos redirigirá a su portal de compra, en donde seleccionaremos el número de entradas que deseamos a la vez que nos muestra el precio total que el usuario deberá pagar, además de poder modificar ahí también la información de usuario y seleccionar el método de pago. Este proceso de compra es bastante similar al que mostrábamos anteriormente, pero se trata de una compra para un único evento, que no nos da acceso al museo. En cuanto a la robustez, no se permite realizar la compra si hay algún campo inválido (no se ha seleccionado método de pago o no se han añadido entradas).

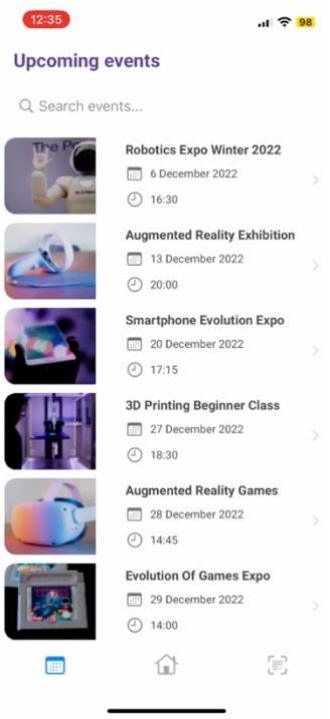


Imagen 61: Lista de eventos en iOS

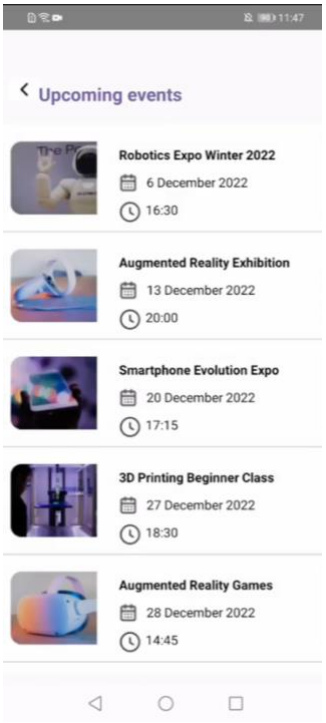


Imagen 60: Lista de eventos en Android

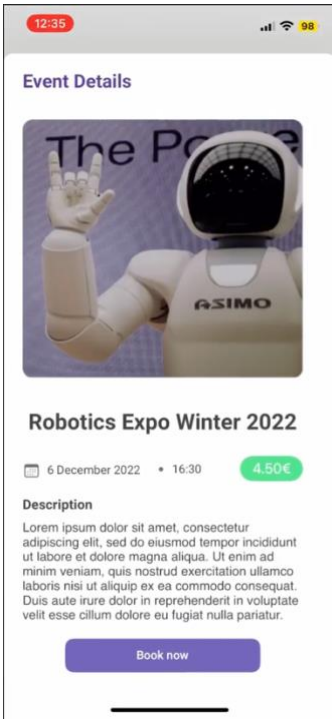


Imagen 59: Detalles del evento en iOS

Imagen 58: Detalles del evento en Android

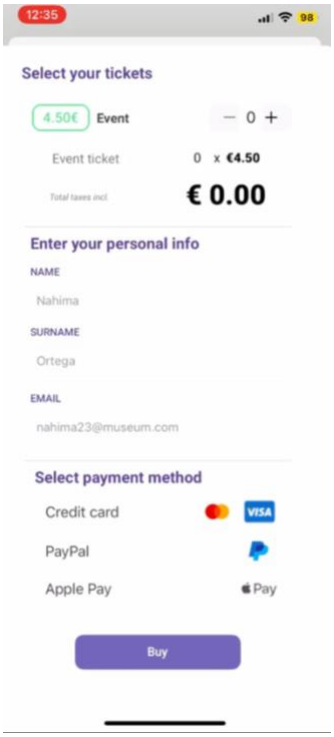


Imagen 63: Compra de eventos en iOS

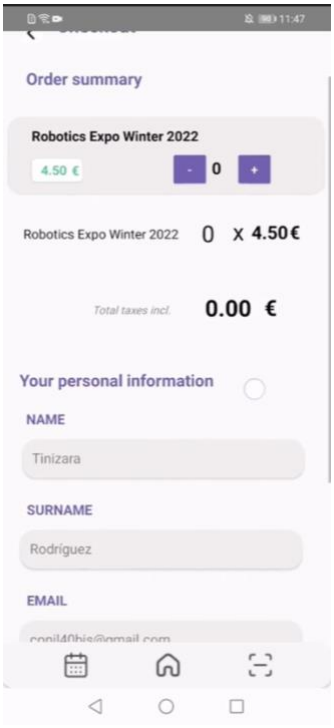


Imagen 62: Compra de eventos en Android

La última opción que encontramos en la barra de navegación es la de escanear códigos QR. Para que funcione de manera correcta, cada código tiene asignado un identificador que se corresponde con el identificador que tiene el producto en la base de datos, por lo que al escanear cada imagen se mostrará el producto junto con su nombre, foto y descripción. Cuando el usuario se encuentre en esa pantalla, podrá realizar dos acciones destacables que son:

- Por un lado, el usuario podrá visualizar un vídeo relacionado con el objeto escaneado. Este vídeo se puede reproducir gracias a la integración de la API de Youtube, la cual permite hacer uso de todas las funciones que ofrece en la web, pero en la propia aplicación.

- Por otro lado, también existe la posibilidad de visualizar comentarios, o incluso añadirlos en tiempo real. Cuando el usuario accede a esta función de la aplicación, lo primero que aparece en la pantalla son las parejas nombre-comentario, que otros usuarios han creado previamente o, en el caso de que no haya ninguno, aparecerá una pantalla en blanco además del botón para crear un comentario. Cuando el usuario añade la opinión o valoración, al hacer *click* en el botón de enviar, ésta se publicará en el tablón de comentarios en conjunto a su nombre. De esta manera, puede ver al instante su comentario reflejado en la sección de comentarios.

Imagen 67: Escáner QR en iOS

Imagen 66: Escáner QR en Android

Imagen 65: Información del objeto en iOS

Imagen 64: Información del objeto en Android

Imagen 71: Ver vídeo en iOS

Imagen 70: Ver vídeo en Android

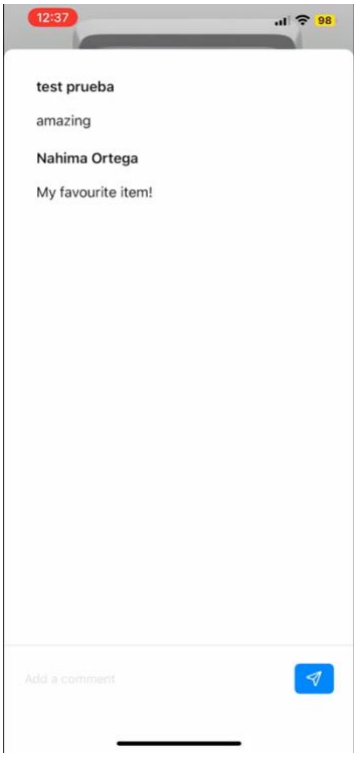


Imagen 69: Comentarios en iOS

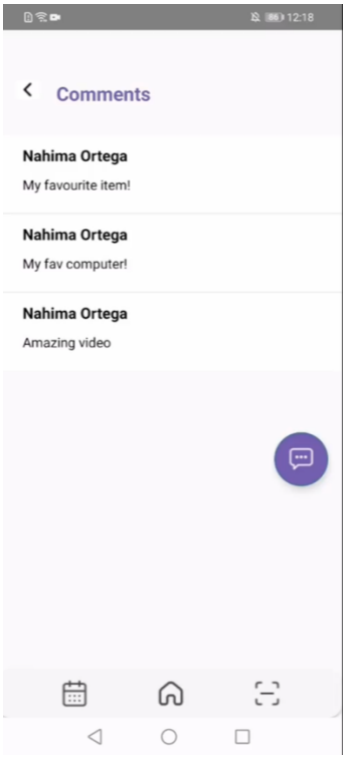


Imagen 68: Comentarios en Android

11. CONCLUSIONES

El principal atractivo de la aplicación desarrollada es el escáner QR, que se incluye sin necesidad de tener que descargar aplicaciones de terceros que pudiesen frustrar la experiencia de usuario tanto en el museo como en el manejo de ésta.

Cada producto que se encuentre físicamente dentro del museo consta de un código QR, el cual podrá ser escaneado haciendo uso de esta utilidad. Tras esto, en la pantalla, se mostrará como resultado una página de información sobre el ítem, y nos ofrecerá dos funcionalidades más que son: por un lado, la integración de un tablón de comentarios independiente de los otros productos, que también tendrán su correspondiente tablón de comentarios. En él se podrían visualizar los comentarios que han hecho el resto de los usuarios, o bien, se podrá añadir uno nuevo, que se publicará de forma instantánea en el tablón, pudiendo visualizarse en conjunto con los anteriores.

La otra opción que se nos brinda es la posibilidad de reproducir un vídeo, que, en algunos casos, se corresponderá con el anuncio que se emitía en el momento en el que el objeto fue puesto a la venta, y, en otros casos, se corresponderá con un vídeo explicativo de alguna función en específico del ítem. Para ello, se ha hecho uso de la API de Youtube.

En la aplicación de iOS, un aspecto a destacar es la posibilidad de sincronizar los datos que se usan en el registro del usuario con el llavero de iCloud, que es una funcionalidad propia de iOS que permite al sistema guardar las contraseñas e inicios de sesión. De esta manera, la próxima vez que el usuario vaya a iniciar sesión, se realizará de forma automática.

Asimismo, la aplicación permite mantener la sesión iniciada una vez que el usuario se ha registrado en el sistema, si es la primera vez que hace uso de ella, o en el caso de que inicie sesión. Es decir, si el usuario sale de la aplicación y la mantiene en ejecución en segundo plano, no tendrá que iniciar sesión de nuevo. Con ello se favorece la memoria a corto plazo del usuario, ya que no tiene que estar constantemente iniciando sesión, además de ofrecer una navegabilidad menos frustrante.

Por último, cabe destacar la conectividad e interoperabilidad que existe entre ambos sistemas, ya que la base de datos es compartida por la aplicación en Android y la de iOS. Por ello, si un usuario se registra en Android e inicia sesión en un teléfono Apple, ese *login* se hará de manera correcta; o, por ejemplo, un usuario Android podrá visualizar también los comentarios que hayan añadidos los usuarios de Apple y añadir él uno que se podrá visualizar por ambos usuarios de ambos sistemas.

12. FUTURAS IMPLEMENTACIONES

Al finalizar el proyecto, se han finalizado por completo las funcionalidades que surgieron en el *brainstorming,* y que fueron diseñadas con los *mockups* en Figma, por lo que las implementaciones futuras son añadidos que puedan aportar más valor a la aplicación.

Por un lado, una implementación futura sería integrar las respectivas pasarelas de pago. Esto hace referencia a servicios que conectarían nuestra cuenta bancaria con el responsable del pago; es decir, conecta el banco emisor, que sería el banco del usuario, con el banco adquiriente, que en este caso sería el banco del museo, en el caso de que el

método de pago usado fuera la tarjeta de crédito. Los diferentes métodos de pago que aceptaría la aplicación serían PayPal, ApplePay, y tarjeta de crédito (Visa y MasterCard).

Otra de las funcionalidades sería integrar realidad aumentada en el escáner QR. La idea consiste en que, cuando el usuario escanee un código que se encuentra a la entrada de la habitación, por ejemplo, se abra una ventana que le vaya mostrando en tiempo real diferentes pop-ups del objeto al que esté enfocando con la cámara, sin necesidad de escanear el QR respectivo de cada ítem. Asimismo, en ese pop-up, se encontraría la información del objeto, como el nombre y una imagen, además de un botón que, si se hiciera *click* en él, redirigía a la pantalla de detalles del producto.

Esto conseguiría transmitir un mensaje más completo sobre la exposición además de llamar la atención del usuario, generándolo un interés que puede que previamente no tuviera. Además, se logra una experiencia guiada y más inmersiva.

En las imágenes inferiores, podemos ver cómo sería el prototipo de baja fidelidad y a la derecha, el prototipo de alta fidelidad, de la vista correspondiente a esta funcionalidad.

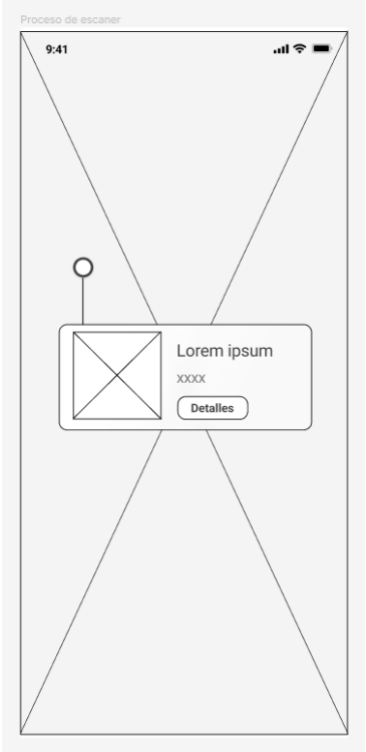


Ilustración 5. Prototipo de baja fidelidad

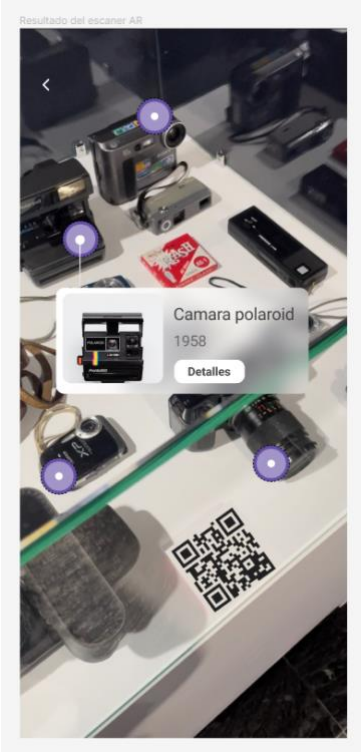


Ilustración 6. Prototipo de alta fidelidad

13. ANEXO I: CÓDIGOS QR

A continuación, se añaden los códigos QR de los productos de prueba que hemos incluido en la aplicación. Se utilizan para la funcionalidad de escáner QR implementada en la aplicación. Estos códigos no se pueden escanear haciendo uso de un escáner externo, ya que cada uno de ellos tiene implementado un identificador único que hace referencia a su respectiva colección almacenada en Firebase Firestore.

- Objeto 1: Ordenador Macintosh

- Objeto 2: Grabadora de Voz Sony ICD-30

- Objeto 3: Nintendo GameBoy

- Objeto 4: Polaroid Pronto 600

- Objeto 5: Family Computer Nintendo

14. ANEXO II: VÍDEO DE LA APLICACIÓN

En el siguiente link, se puede acceder a un vídeo que muestra el funcionamiento de la aplicación para esta versión, tanto para Android como para iOS.

15. ANEXO III: PRESENTACIÓN DE LA APLICACIÓN

En el siguiente link, se puede acceder a una presentación en Microsoft PowerPoint con los puntos básicos de la aplicación, es decir, un resumen breve de esta memoria.

16. ANEXO IV: SITUACIONES ADVERSAS

En cuanto a los archivos ejecutables que corresponderían a la instalación de la aplicación en los dispositivos móviles, la referente al sistema iOS, se nos muestra una dificultad, ya que para poder generar el archivo .ipa en Xcode, que se corresponde con el .apk de Android; solo puede ser generado si eres miembro del

programa de desarrolladores de Apple, por lo que solo se podría probar teniendo un dispositivo apple conectado al ordenador o usando el emulador propio del ide.

Es por ello por lo que, en este caso, para poder testearla, es necesario descargarse el código que se encuentra en el github del equipo e importarlo en el entorno de desarrollo Xcode propio.