ASTRORETOS

Proyecto final del Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma

Juan Manuel Perez Lopez
3DAMN IES EL RINCON

Contenido

OBJETIVOS

Objetivo de este documento

El objetivo de este documento es el registro detallado del proceso de desarrollo y de los resultados obtenidos del Proyecto Final del Ciclo Formativo de Grado Superior de **Desarrollo de Aplicaciones Multiplataforma**. Su objetivo es documentar las decisiones tomadas y las acciones realizadas durante el ciclo de vida del proyecto, desde la planificación hasta la finalización. Este documento también se utiliza para evaluar el rendimiento del proyecto y para identificar las lecciones aprendidas y las mejores prácticas que se pueden aplicar a proyectos futuros. Además, este documento sirve como fuente de información para los *stakeholders* y para futuros miembros del equipo que puedan necesitar revisar o actualizar el proyecto en el futuro.

Objetivo del Proyecto

El objetivo de este proyecto es aplicar los conocimientos adquiridos durante el Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma para crear un proyecto funcional. El proyecto se centrará en la creación de una aplicación informática que cubra una necesidad específica del usuario y que utilice tecnologías actuales y herramientas de desarrollo modernas. Para lograr este objetivo, se seguirá un proceso de desarrollo iterativo y se aplicarán las prácticas de programación y de gestión de proyectos aprendidas durante el curso. El proyecto también incluirá un análisis detallado de los requisitos del usuario, del modelo de datos y casos de uso. Al finalizar el proyecto, se espera obtener una aplicación informática completamente funcional y documentada que pueda ser utilizada por el usuario final y que represente todo lo aprendido durante el curso.

Descripción del Proyecto

AstroRetos consiste en una aplicación multiplataforma destinada a ofrecer una ayuda a astrónomos aficionados en una noche de observación.

Durante una noche de observación hay tantas cosas que observar que un observador novel puede llegar a sentirse abrumado por no saber qué puede observar.

AstroRetos ofrece un listado de objetos y efemérides en forma de retos para que el observador pueda completarlos y así tener un objetivo durante la observación.

Estos retos se ofrecen como tarjetas con una imagen y una descripción detallada sobre cómo es posible conseguir el reto, además de información de interés sobre el objeto o efeméride a observar.

Dentro de las tarjetas también se podrá ver la dificultad del reto y el tipo de observación para el que está concebido (ocular, con prismáticos o con telescopio).

Desde la tarjeta también se podrá añadir el reto a nuestra lista de Favoritos para poder hacer un seguimiento a los retos que más nos interesan.

Además, estos retos otorgan una serie de puntos en función de su dificultad, ofreciendo la posibilidad de ver el ranking de los observadores más activos.

Los retos los podrá crear tanto el usuario administrador cómo los usuarios registrados, llamados *Retadores*.

Como añadido, también se ha añadido una sección de noticias (en inglés) con toda la actualidad del Espacio y la Astronomía.

También se ha querido potenciar la Documentación de un proyecto usando componentes específicos para este cometido, consultables desde la propia aplicación.

Tecnologías usadas

Para el desarrollo completo de la aplicación se han utilizado una serie de Frameworks, IDEs, APIs y componentes varios.

Framework

Angular es un *framework* de ingeniería de software de código abierto mantenido por Google, usado para desarrollar aplicaciones web de estilo Single Page Application (SPA) y Progressive Web App (PWA). Sirve tanto para versiones móviles como de escritorio.
Una de las principales ventajas es su documentación detallada. La documentación de Angular está cuidadosamente escrita y dotada de una gran variedad de ejemplos de código para tener mayor claridad en el proceso y permitir que los desarrolladores encuentren soluciones rápidas a cualquier problema conforme crean una aplicación.

Utiliza un patrón de diseño MVVM (Modelo-Vista-VistaModelo) que separa la lógica de presentación de la lógica de negocio, lo que lo hace más fácil de mantener y depurar.

SDK

Ionic es un SDK de *frontend* de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Es decir, un *framework* que nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código.
Es multilenguaje, aunque en este caso se usara con Angular.

Permite desarrollar y desplegar aplicaciones híbridas, que funcionan en múltiples plataformas, como iOS nativo, Android, escritorio y la web como PWA, todo ello con una única base de código.
Ofrece un diseño limpio, sencillo y funcional y es compatible con multitud de *plugins.*

Backend

Se ha optado por Google Firebase. Se trata de una plataforma de desarrollo de aplicaciones móviles y web de Google que proporciona una variedad de herramientas y servicios para ayudar a los desarrolladores a crear aplicaciones de alta calidad. Firebase ofrece servicios de backend que se integran fácilmente con aplicaciones móviles y web. Dentro de todos esos servicios, para este proyecto se han utilizado:

- **Firebase Authentication**: un servicio de autenticación de usuario que agregar fácilmente la autenticación basada en correo electrónico y redes sociales a sus aplicaciones.

- **Cloud Firestore**: un servicio de base de datos en tiempo real y sin servidor que permite almacenar y sincronizar datos en tiempo real.

- **Cloud Storage**: un servicio de almacenamiento de objetos en la nube que permite almacenar y recuperar archivos de manera fácil y segura.

Control de Versiones

Se ha utilizado GIT, es un sistema de control de versiones distribuido de código abierto que se utiliza para realizar un seguimiento de los cambios en el código fuente de un proyecto de software y para coordinar el trabajo en equipo en un proyecto de software.

Para alojar el proyecto se ha recurrido a GitHub, una plataforma en línea de alojamiento y gestión de código fuente para proyectos de software.

La creación del proyecto en GitHub se ha realizado con GitKraken, una herramienta de interfaz gráfica de usuario (GUI) para Git que ayuda a los desarrolladores a visualizar y gestionar fácilmente los repositorios de Git. Permite a los desarrolladores realizar tareas de Git como crear ramas, fusionar

cambios, hacer revertir cambios y colaborar en proyectos de Git de manera visual e intuitiva. Se integra perfectamente con servicios de alojamiento como GitHub.

Editor de Código

Se ha tomado la decisión de usar Visual Studio Code. es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Se utiliza para escribir y depurar código en una variedad de lenguajes de programación, incluidos JavaScript, TypeScript, Python, C++, Java y muchos más.

VS Code ofrece características avanzadas para el desarrollo de software, como la resaltado de sintaxis, el autocompletado, la navegación de código y la depuración. También es altamente personalizable, con soporte para una amplia gama de extensiones que pueden agregar características y funcionalidades adicionales al editor.

Algunas de estas extensiones usadas son:

- **Angular Snippets**. Facilita la introducción de código predefinido permitiendo una codificación más ágil.
- **Ionic Snippets**: de la misma forma que Angular Snippets, facilita la introducción de código.
- **JSON to TS**: Convierte objetos JSON a Typescripts.

IDE

Para poder preparar la aplicación para su uso y probarla en emuladores se ha usado Android Studio, diseñado específicamente para desarrollar aplicaciones para dispositivos móviles con el sistema operativo Android. Está desarrollado por Google y es la herramienta de desarrollo recomendada para los desarrolladores de Android.

Android Studio nos proporciona una amplia variedad de características y herramientas para el desarrollo de aplicaciones de Android, incluyendo la edición de código, la depuración, la creación de interfaces de usuario, la gestión de recursos, el empaquetado de aplicaciones, y mucho más.

Otras Utilidades

diagrams.net

Herramienta de código abierto y gratuita que se ha utilizado para crear el esquema de la Base de Datos. Anteriormente conocida como draw.io, la herramienta se ejecuta completamente en línea y está disponible como una extensión para Google Drive, como una aplicación de escritorio y como una aplicación web.

Postman

Herramienta de colaboración y prueba de API (interfaz de programación de aplicaciones) que se utiliza para diseñar, probar y documentar API. Está disponible como una aplicación de escritorio, una extensión de navegador y una aplicación web.

Backend

Dentro del Backend de Google Firebase se han utilizado varios de los componentes que ofrece. A continuación se explicará el uso de cada uno de ellos.

Firebase Authentication

Se ha utilizado este Servicio de autenticación ya que forma parte de Google Firebase y se integra perfectamente en aplicaciones PWA y móviles. Con Firebase Authentication, se pueden agregar funciones de autenticación de usuarios a sus aplicaciones con solo unas pocas líneas de código, lo que permite a los usuarios registrarse y autenticarse en una aplicación con su correo electrónico y contraseña, o mediante una cuenta de proveedor de autenticación externo como Google, Facebook, Twitter…

En este caso, se ha usado sólo la autenticación por email. El usuario podrá crear desde la propia aplicación su cuenta en **AstroRetos,** y usarla posteriormente para participar en la aplicación.

Cloud Firestore

Cloud Firestore es una base de datos NoSQL de documentos en tiempo real y en la nube. Es una base de datos escalable y flexible que permite almacenar, sincronizar y consultar datos para aplicaciones web y móviles.

Firestore utiliza un modelo de datos basado en documentos, donde cada documento representa una única entidad, y se organiza en colecciones de documentos. Los datos dentro de los documentos se almacenan en formato JSON, lo que hace que Firestore sea fácil de integrar con aplicaciones basadas en JavaScript.

En AstroRetos se han creado 4 colecciones para almacenar todos los documentos:

usuarios

Contiene los documentos relativos a los usuarios registrados en la aplicación. Cuando un usuario se registra a través de Firebase Authentication, se crea automáticamente el documento con la información de ese usuario:

- **ID_USUARIO**: Almacena el mismo ID que autogenera Firestore para facilitar el manejo del documento.
- **NOMBRE**: Almacena el nombre escrito por el usuario al registrarse.
- **EMAIL**: Dirección de email del usuario y que coincide con la del registro.

- **ROL**: Por defecto tendrá el rol de retador, aunque se puede convertir a Administrador desde dentro de la aplicación.
- **AVATAR**: Almacena el token de acceso a la imagen de su avatar en Storage. Por defecto se le asigna una imagen sencilla, aunque desde la aplicación el usuario podrá cambiarla.
- **PUNTOS**: Total de puntos conseguidos por el usuario al ir completando retos.

retos

En esta colección se guardan todos los documentos referidos a los Retos. Se crean desde la propia aplicación.

- **ID_RETO**: Almacena el mismo ID que autogenera Firestore para facilitar el manejo del documento.
- **TITULO**: Título del Reto.
- **DESCRIPCION**: Texto detallado del Reto.
- **TIPO**: tipo de reto a observar. Puede ser Ocular, Prismáticos o Telescopio, elegible desde la aplicación.
- **NIVEL**: Nivel de dificultad del Reto. Puede ser Fácil, Intermedio o Difícil. Elegible desde la aplicación.
- **ACTIVO**: Marca si el reto esta activo o no. En la aplicación solo se muestran los activos, aunque desde el panel de gestión de Retos se puede cambiar el estado.
- **RETADOR**: email del usuario que crea el Reto.
- **IMAGEN**: Imagen descriptiva del Reto. Seleccionable desde la galería del dispositivo.

favoritos

Almacena todos los favoritos que marcan los usuarios en la aplicación.

- **ID_FAV**: Id del documento, igual al autogenerado.
- **ID_RETO**: Id del Reto.
- **USER**: Email del usuario que marca el Reto como Favorito.

retosconseguidos

Colección que guarda todos los retos conseguidos por los usuarios.

- **ID_RETO_CONSEGUIDO**: Id del documento, igual al autogenerado.
- **ID_RETO**: Id del Reto.
- **USER**: Email del usuario que marca el Reto como Favorito.
- **PUNTOS**: Puntos conseguidos con ese Reto.

En el Anexo se podrá ver el esquema de la Base de Datos y la lógica de relaciones entre colecciones, recordando que se trata de una Base de Datos **No Relacional**.

Cloud Storage

Es el servicio de almacenamiento en la nube proporcionado por Firebase que permite a los desarrolladores almacenar y recuperar datos, como imágenes, videos, archivos y otros tipos de contenido multimedia en la nube.

Una vez alojada la imagen en Storage, se crea un token de acceso que es el que se almacena dentro del Documento de la Colección correspondiente.

Para AstroRetos se han creado dos carpetas.

- Avatar: Para guardar los avatares de los usuarios.
- Retos: dónde se guardan las imágenes de los Retos.

La subida de las imágenes se hace desde la aplicación.

Roles

AstroRetos cuenta con 2 roles de uso además del rol de invitado.

Invitado: Podrá ver los Retos y compartirlos. También podrá acceder a la sección de Noticias y compartirlas

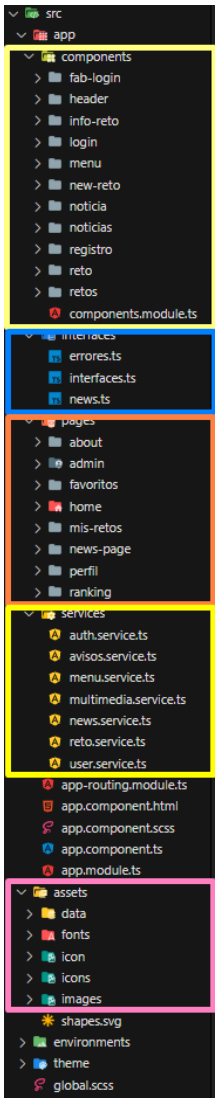
Retador: Tendrá acceso a todos los apartados de la aplicación excepto al de Administración. Además de poder hacer lo mismo que un Invitado, podrá marcar Retos como conseguidos, como Favoritos, gestionar esos favoritos y ver los Retos creados por él y gestionarlos.

Administrador: Además de todas las funciones del Retador, también puede gestionar al resto de usuarios y los Retos de todos.

Frontend

Desarrollo de la aplicación

Estructura



Components: Son elementos de la interfaz reutilizables. Facilitan la construcción de una interfaz de usuario consistente y escalable en toda la aplicación. Se controlan desde el components.module.ts, dejando el app.module.ts por defecto para los componentes genéricos de Ionic.

Interfaces: Aquí está el modelo de datos. Objetos con los mismos campos que las colecciones en Firestore. También está la interface para las noticias y los errores.

Pages: Las páginas principales del proyecto con las distintas secciones: retos (home), noticias, perfil del usuario, listado de favoritos…

Services: Servicios accesibles desde distintas partes del proyecto, como los avisos, los CRUD que conectan con Firestore, la carga de noticias o el manejo de la cámara.

Assets: Aquí guardo el JSON con las opciones de menú, y los distintos recursos gráficos que se usarán durante el proyecto.

Components

Components.module.ts

Módulo donde se declaran todos los componentes personalizados que se han ido creando. Para facilitar el mantenimiento del código, en el archivo *app.module.ts* solo se declaran componentes externos y el propio

components.module.ts, dejando para éste último las declaraciones del resto de componentes creados.

Fab-Login

Este componente se encarga de mostrar el *fabIcon* que da acceso al componente de Login o al de Registro. Sólo se muestra si no hay ningún usuario logado, y tras el *login*, se oculta.

Header

Se encarga de mostrar la cabecera de las páginas. Al hacerlo a través de un componente, es más fácil de mantener en caso de tener que modificarlo y ahorra el código repetitivo.

InfoReto

Componente modal que se muestra desde la tarjeta de un Reto o desde el listado de Retos desde las páginas de gestión.

Login

Componente modal que muestra el formulario de acceso con usuario registrado. Este formulario tiene validadores que comprueban si el formato del email es válido y si la contraseña tiene al menos 6 caracteres.

En el mismo modal, aparece el enlace para abrir el formulario de registro en caso de que no tengamos cuenta.

Menu

He optado por no usar el menú estándar de aplicaciones Ionic y usar en su lugar un favIcon que al pulsar muestra una serie de botones dinámicos que nos llevan a las distintas secciones de la aplicación. Dependiendo de si estamos logamos o no (y del rol del usuario logado) mostrará unas opciones u otras.

NewReto

Muestra el formulario de creación de un nuevo Reto como un modal. En él, los usuarios con rol de Retador o Administrador podrán crear Retos rellenando el formulario. Los campos obligatorios están señalados y son validados por el propio formulario. Este formulario tiene un *ion-select* para acotar el tipo de Reto a tres, y otro *ion-select* para acotar la dificultad.

Noticia y noticias

Para facilitar el manejo de cada noticia, se han creado dos componentes. Uno de noticias que recibe toda la lista de noticias, y otro componente llamado *noticia* que muestra la noticia en in *ion-card* pasándole el *id* desde *noticias*.

Registro

Muestra el formulario de registro como un modal. Nos pide tres campos. Nombre, email y contraseña. Tanto el email como la contraseña se validan para que tenga el formato y longitud correctas.

Reto y Retos

Al igual que las noticias, se crean dos componentes para facilitar su manejo. *Retos* carga el array de retos y llama al componente Reto pasándole el *id*. Este componente *Reto* es que el que después tiene toda la lógica.

Interfaces

Errores

Interface con los tipos de error de email y password para facilitar su manejo.

Interfaces

Aquí es dónde se encuentra el modelo de datos de todos los documentos de todas las colecciones de la Base de Datos con los mismos campos que están en Firestore.

También tiene la interface de las opciones de menú, con los campos de icono, nombre, *url* a la que navegar y el rol que tiene acceso a él.

News

Interface que se tiene que usar para manejar los datos recibidos por la API de *NewsApi*.

Esta interface se crea obteniendo el el objeto *JSON* a través de *PostMan* y pasándolo por el plugin de *Visual Studio Code JSON to TS*, que lo convierte a lenguaje *TypeScript*.

Pages

About

Página con la información de AstroRetos, con enlaces a los perfiles de su autor y a la documentación.

Admin

Página a la que sólo tienen acceso los administradores. En ella se gestionan Retos, pudiéndolos marcar activos/inactivos y eliminarlos.

También se gestionan los Usuarios. Se les puede dar rol de Administrador o Retador, y eliminarlos.

Favoritos

En esta página aparecerán los retos favoritos del usuario logado. Solo los Retadores tienen acceso a es página. También se puede quitar el favorito deslizándolo o mostrar sus detalles pulsando en él.

Home

Página principal de **AstroRetos**. En ella se muestran todos los retos activos a través de *ion-cards*. Cada *ion-card* tiene toda la información del Reto: tipo, nivel, acceso a los detalles, iconos de acción y una imagen del Reto.

MisRetos

Página que muestra los Retos creados por el usuario logado. Desde ella se puede acceder a los detalles del Reto pulsando sobre él y eliminarlo o ponerlo activo/inactivo deslizando hacia la izquierda.

NewsPage

Página de Noticias. Muestra noticias obtenidas a partir de la API gratuita de *NewsApi*. Puedes compartir cada noticia por Redes Sociales y al pulsar sobre ella la abre en un navegador integrado en la propia aplicación a través del plugin *InAppBrowser.* La página incorpora un *infiniteScroll* para no cargar todas las noticias a la vez y la navegación sea más ágil.

Perfil

En esta página se muestra el perfil del usuario logado. En ella puedes cambiar la imagen del Avatar, seleccionando una imagen desde la galería o usando la cámara del dispositivo. También se puede cambiar la contraseña, cerrar sesión y eliminar la cuenta.

Ranking

Una página que muestra los usuarios con mayor puntuación ordenados de mayor a menor.

Services

Auth

Servicio que gestiona todo lo relacionado con Firebase Authentication. Se encarga se la creación, eliminación y modificación de cuentas de usuario.

Avisos

Usado para mostrar mensajes de aviso mediante un *toast.* Se ha creado para ahorrar código repetitivo. Recibe por parámetros el mensaje a mostrar y el color.

Menu

Este servicio gestiona el menú. Comprueba si hay usuario logado. Si no lo hay carga el array de opciones con los objetos de menú a los que tiene acceso. Si está logado comprueba el rol y carga las opciones en consecuencia. El array de objetos es un JSON.

Multimedia

Servicio creado exclusivamente para la subida de la imagen a *Google Firestorage*. Sube el archivo y recibe el *token* de acceso a la imagen para luego

adjuntarla al documento del usuario o del Reto, que son dónde se usan las imágenes.

News

Desde este servicio se gestionan las consultas a la API de NewsApi para recoger las noticias. Se elabora la consulta a lanzar y recoge los datos para luego mostrarlas en la página.

Reto

Este servicio es el encargado de gestionar toda la lógica de los Retos. Tiene los métodos de creación, modificación, eliminación, consulta de nivel, dificultad, gestión de favoritos y gestión de retos conseguidos y puntuación.

User

Servicio encargado de la gestión de la colección de usuarios para actualizar sus valores. También recoge los valores que se pasan al servicio de Manu para la mostrar las opciones.

Assets

En estas carpetas se encuentran distintos recursos gráficos usados por la aplicación. También se encuentra el JSON con las opciones de menú.

Environments

Aquí están registradas las variables de entorno que usa la aplicación.

Firebase: variable de entorno con todo lo necesario para la comunicación con *Firebase*.

News: la *apiKey* y la URL de cabecera para la consulta a la API de *NewsApi*.

Theme y global.scss

Aquí están las hojas de estilo de la aplicación para personalizar su aspecto. En *theme/variables.scss* se incidan los colores por defecto de la aplicación tanto para tema claro o tema oscuro.

En el *global.scss* están distintas clases para personalizar distintos elementos de la aplicación, como los modales, *fablcons*, títulos, fuentes…

Documentación

Se ha querido dotar a la documentación del proyecto una gran importancia otorgándole un espacio propio. La documentación sirve como recurso de gran valor para futuros desarrolladores o personas que necesiten mantener o mejorar el proyecto. Proporciona información sobre la estructura del proyecto, los componentes utilizados, las decisiones de diseño y los detalles técnicos importantes. Esto facilita el mantenimiento y la evolución del proyecto a medida que se realizan actualizaciones o se agregan nuevas funcionalidades.

Para ello se ha utilizado un componente externo llamado **Compodoc**.

Compodoc es una herramienta de generación de documentación para proyectos de Angular. Proporciona una manera fácil y automática de generar documentación detallada y bien estructurada para el código fuente de una aplicación Angular.

También analiza el código fuente de un proyecto Angular y extrae información clave, como componentes, directivas, servicios, módulos y sus respectivas propiedades, métodos y eventos. Luego, genera una documentación interactiva basada en esta información, que incluye una descripción general del proyecto, una lista de componentes y sus características, ejemplos de uso, diagramas de dependencia y mucho más.

La documentación generada por Compodoc se presenta en un formato amigable para el navegador, lo que facilita la navegación y la búsqueda de información. Además, es altamente personalizable, lo que permite a los desarrolladores adaptar la apariencia y el contenido de la documentación según sus necesidades.

Compodoc también proporciona métricas e informes detallados sobre la calidad del código, como la cobertura de pruebas, el acoplamiento y la cohesión, lo que ayuda a los equipos de desarrollo a evaluar y mejorar la calidad de su código.

Tan solo hay que poner un script en el archivo *package.json* para posteriormente ejecutarlo desde consola.

---

npm run compodoc

npx compodoc -p tsconfig.doc.json -d documentacion -n \"Documentación de AstroRetos\" -o -s -w --language es-ES --theme material –hideGenerator

---

Costes

Cuantificar los costes de una aplicación es una tarea complicada ya que se trata de una aplicación lúdica planteada como un hobby.

Aún así, las horas de programación si han tenido un coste cuantificable.

Se han invertido sobre unas 80 horas en el proyecto entre el diseño y la codificación.

Un desarrollador de angular en España está de media sobre los 19€/hora por lo que sólo en costes de desarrollo, AstroRetos costaría unos 1520€

Con ese importe ya se podría lanzar la aplicación en una fase inicial creando el archivo apk compatible con Android o alojando la PWA en un espacio web gratuito. Recordemos que una de las ventajas de Ionic es que la aplicación se puede desarrollar para web, Android o Apple.

Si se quiere mayor visibilidad lo ideal es publicarla tanto en Google Play como en la Apple App Store.

Publicar una aplicación en Google Play cuesta actualmente sobre los 100$ (tarifa única de creación de cuenta de desarrollador).

Si decidiéramos publicarla además en Apple App Store, el registro como desarrollador sería de $100 anuales.

En cuanto a la API de NewsApi, se está usando el plan gratuito que permite 100 peticiones al día.

También hay que tener en cuenta un posible éxito de la aplicación.

Actualmente se está utilizando el Plan Spark de Google Firebase, gratuita, con unos límites de 20000 operaciones de lectura, 50000 de escritura y 20000 documentos borrados al día.

Un posible éxito nos llevaría a cambiar de plan y pagar $0.08/$0.16/$0.02 por cada 100000 operaciones de lectura/escritura/borrado.

Esto también repercutiría en el precio de la API de noticias, que se nos iría a $449/mes o $1749/mes según el éxito.

Teniendo en cuenta que AstroRetos tiene mucho margen de mejora y es posible añadir otras funcionalidades más avanzadas, se podría crear un sistema de Roles Premium que sufraguen este costo.

Conclusiones

A lo largo de todo el desarrollo AstroRetos se han podido poner en práctica gran parte de lo aprendido en el **CFGS de Desarrollo de Aplicaciones Multiplataforma**.

- Definición de requisitos recopilando y analizando la aplicación que se quiere conseguir para buscar las herramientas adecuadas.
- Diseño y planificación. Diseñar la Base de Datos que guardará la información que manejará la aplicación. Crear un prototipo estático de la aplicación con *mockups*. Planificar las funciones que va a tener la aplicación y como organizarlas.
- Desarrollo de la interfaz que tendrá la aplicación. En esta fase se crearon todas las vistas de la aplicación, sin funcionalidad, para ver como quedaría el producto final.
- Desarrollo de funcionalidades. Se desarrollan las características y componentes del software de acuerdo con los requisitos establecidos. Es importante seguir las buenas prácticas de desarrollo y realizar pruebas regulares para asegurar la calidad del código.
- Testeo y pulido de código, dónde se revisan las funcionalidades y se depuran posibles problemas de funcionamiento.
- Despliegue de la aplicación. Publicación de la aplicación como PWA y preparación en Android Studio para poder publicarla en Play Store.
- Documentación de todo el proceso, incluida la documentación *inline*.

En cuanto a la tecnología elegida, Ionic, me ha dejado buen sabor de boca debido a la curva de aprendizaje constante que tiene y al jugo que se le puede sacar. La documentación oficial es muy consistente, aunque tengo que decir que cuando buscar algo concreto que no está en esta documentación se vuelve un poco errático ya que la mayoría de ejemplos que hay son de versiones Ionic anteriores.

Un punto negativo es ese, el *framework* está en constante cambio, y muchos métodos que te funcionaban en una versión, podían dejar de funcionar en la siguiente.

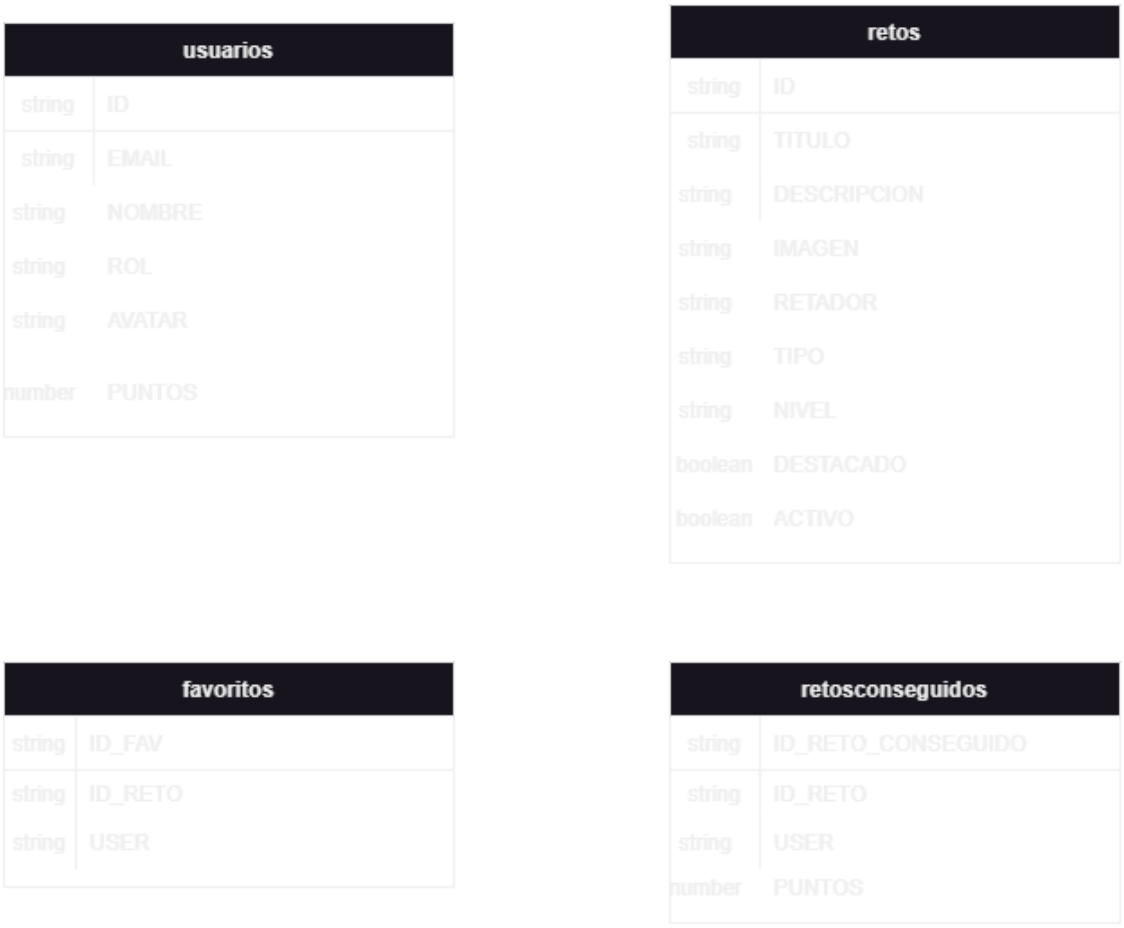
En cuando a Firebase, a pesar de no usar todas las funciones que ofrece, lo veo una opción muy válida para proyectos que requieren inmediatez y alta disponibilidad con Bases de Datos no demasiado complejas. Quizá para una aplicación empresarial flaquee un poco a no tener una estructura tan sólida como puede ser MySQL, SQL Server o PostgreSQL.

Pero en general, muy contento con el resultado final teniendo en cuenta el tiempo disponible.

Bibliografía

- https://ionicframework.com/
- https://angular.io/
- https://firebase.google.com/?hl=es-419
- https://compodoc.app/
- https://www.udemy.com/share/1013fO3@0J0Mwl2RZl23O9AqSAfIzCJgpdcoLcDnWBPW8VIeRMman4x1FFaXSrZNk51FgOsL/
- https://www.udemy.com/share/101tFG3@XVTPQIlanlHm3p1SZpp-sE-Sy7sFPNA1HlNY8FRfnsdIPBFDu-1jhbtyIDO7Z11j/
- https://youtu.be/MRXO2WQMxNM
- https://stackoverflow.com/questions/74745954/error-angular-fire-build-incorrectly-extends-interface
- https://capacitorjs.com/
- https://github.com/apache/cordova-plugin-inappbrowser
- https://danielsogl.gitbook.io/awesome-cordova-plugins/
- https://app.diagrams.net/

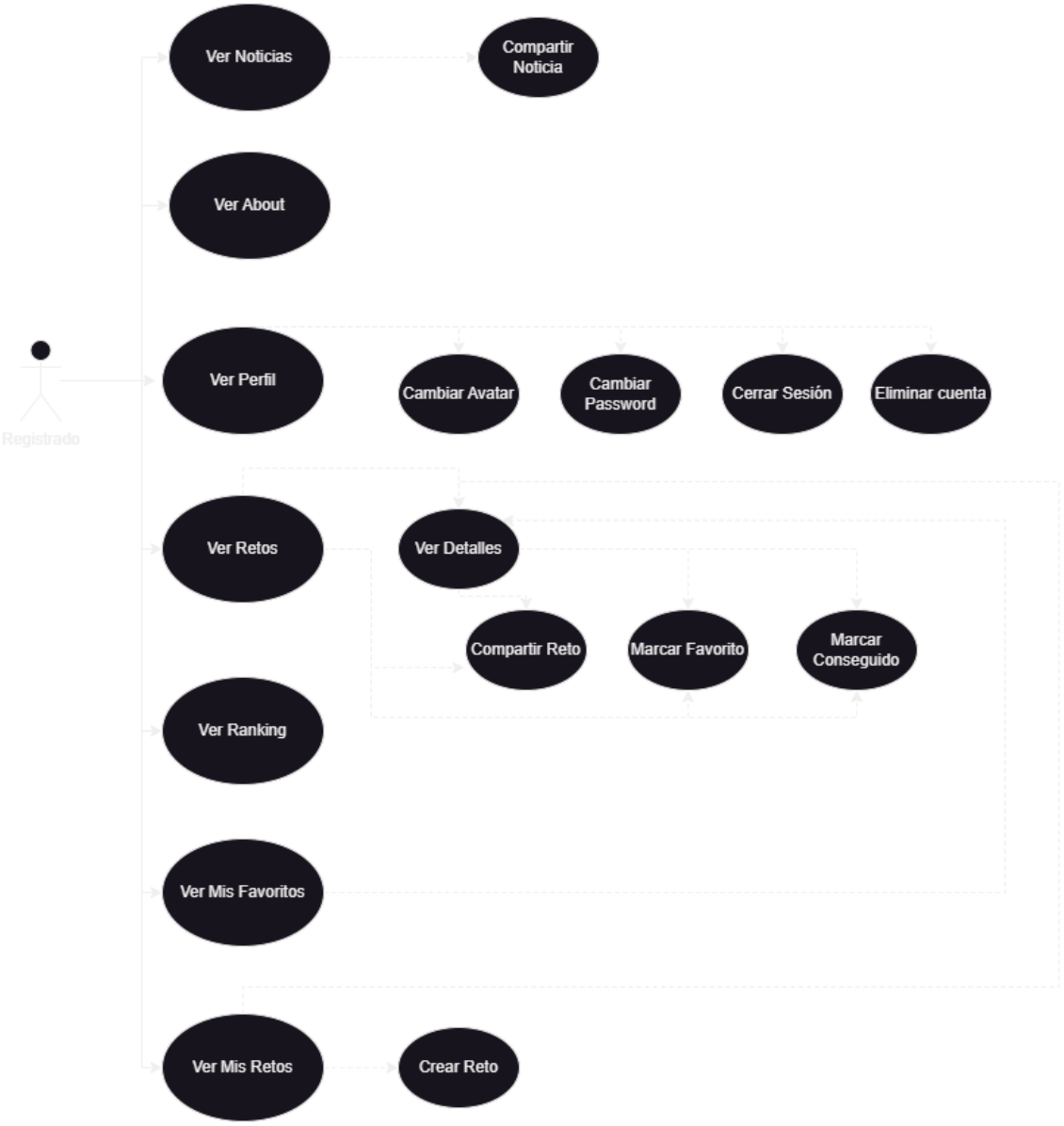
Anexo 1: Modelo de Base de Datos



Anexo 2: Casos de uso

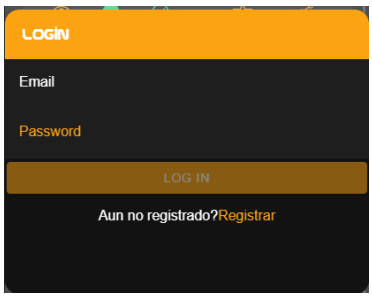
Usuario Anónimo

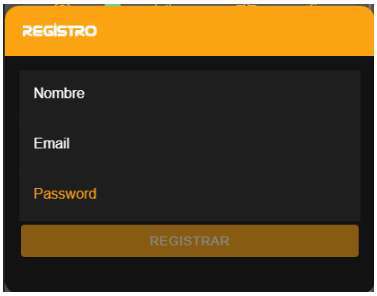
Usuario Registrado



Anexo 3: Vistas de la aplicación

Sistema de Login y registro





Menús

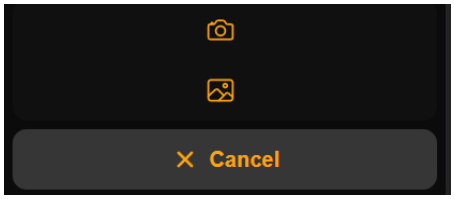
Anónimo

Retador

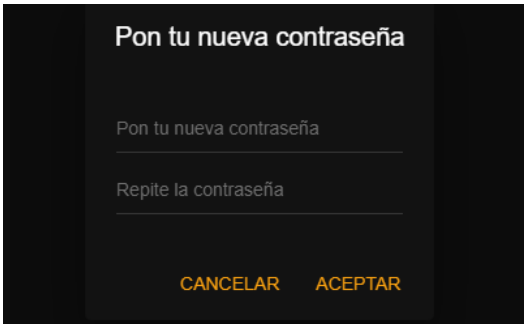
Admin.

Página de Perfil

Cambio de Avatar



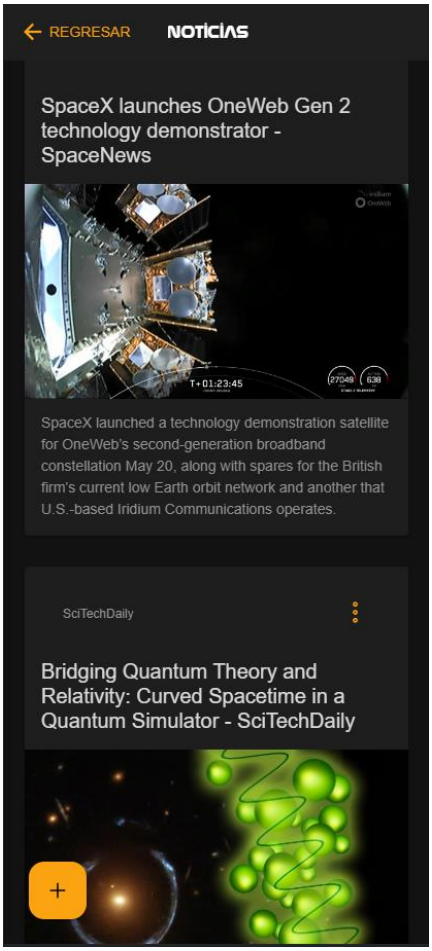
Cambio de contraseña



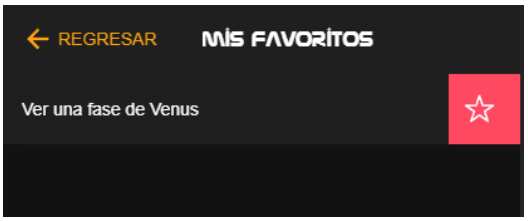
Retos

Detalle del Reto

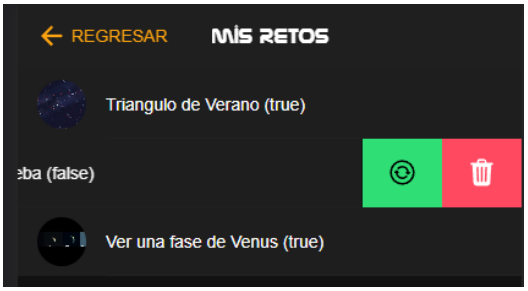
Página de Noticias



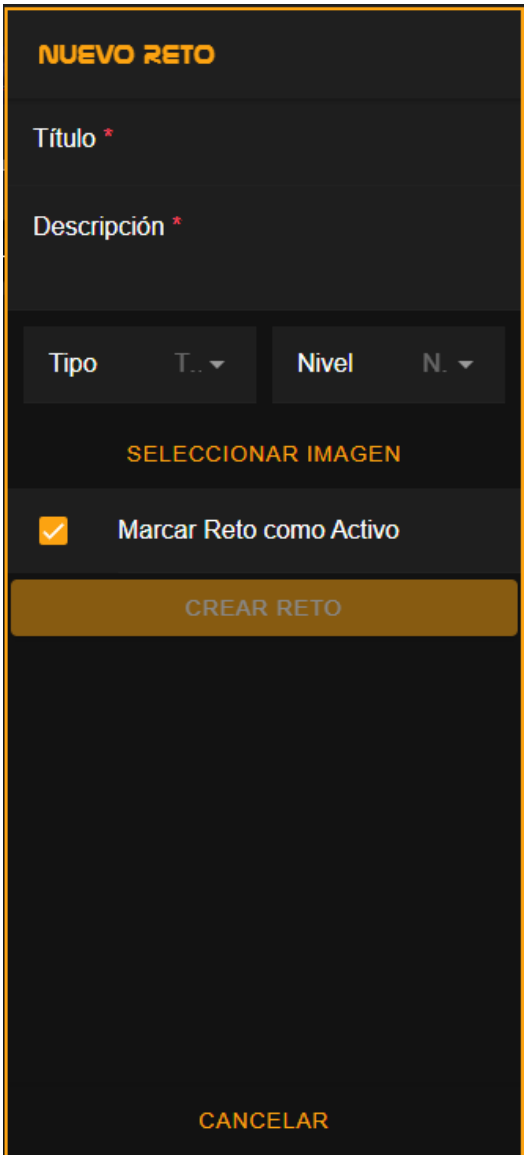
Página Mis Favoritos



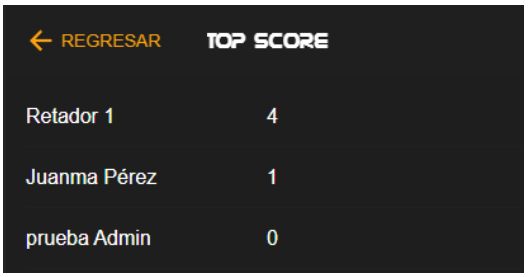
Mis Retos



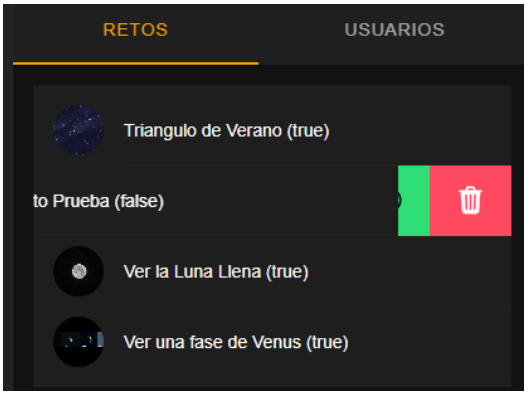
Nuevo Reto



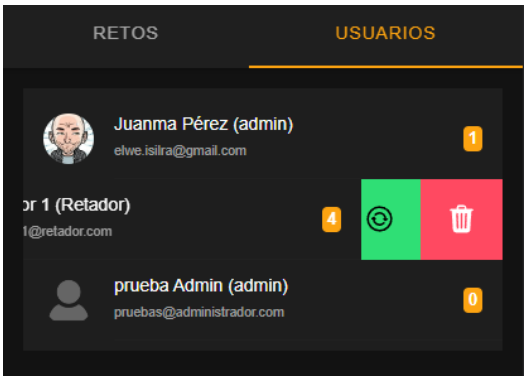
Página Ranking



Página Admin. Retos



Página Admin. Usuarios



Página About