





CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

Gagarin

Fabiola, Fran, Ionut, Pablo

CURSO 2020-21

TÍTULO : Gagarin

AUTORES: Fabiola Decena Figueroa

Francisco Javier Ruiz Joya

Ionut Alin Bozintan

Pablo Benito Lopez

TUTOR DEL PROYECTO: Ernesto Ramiro Córdoba

FECHA DE LECTURA: 7 Junio de 2021

En Madrid

Ernesto Ramiro Córdoba
Tutor del PFC

Resumen:

Gagarin es una aplicación móvil de noticias espaciales con el objetivo de entusiasmar a la gente y generar interés en la carrera espacial. Con los recientes lanzamientos y recuperaciones exitosas de cohetes reutilizables ha comenzado una nueva carrera espacial. Esta vez, el objetivo son los viajes comerciales e interplanetarios, por lo que es importante transmitir a la población que los cohetes pueden ser seguros y confiables. Gagarin tiene como objetivo hacer precisamente eso al proporcionar las últimas noticias espaciales, lanzamientos y mapas interactivos en 3D.

Antes de los cohetes reutilizables, el interés y las esperanzas de los viajes espaciales eran mínimos desde la carrera espacial hacia la Luna. Esto se debió al alto costo de los viajes espaciales pero, lo que es más importante, al riesgo de que los cohetes no funcionen correctamente debido a su baja confiabilidad. Ahora empresas como SpaceX y Rocket Labs tienen cohetes reutilizables y están demostrando que los cohetes pueden ser seguros y fiables. El problema ahora es demostrarle a la gente que los cohetes son lo suficientemente seguros para que los humanos los aborden y los aviones comerciales lo son. Con Gagarin, queríamos que la gente volviera a interesarse por el espacio y comprendiera que la tecnología de cohetes actual es segura. Esta aplicación está dirigida principalmente a una generación más joven (1990+) porque son los que tienen más probabilidades de tener la oportunidad de viajar al espacio.

Abstract:

Gagarin is a mobile space news app with the aim of getting people excited and building interest in the latest space race. With the recent successful launches and recoveries of reusable rockets a new space race has begun. This time the goal is in commercial and interplanetary travel, thus, it's important to convey to the population that rockets can be safe and reliable. Gagarin aims to do just that by providing the latest space news, launches, and interactive 3D maps.

Before reusable rockets the interest and hopes for space travel were at the lowest since the space race to the moon. This was due to the high cost of space travel but most importantly the risk of rockets malfunctioning due to their low reliability. Now companies such as SpaceX and Rocket Labs have reusable rockets and are showing that rockets can be safe and reliable. The problem now is proving to people that rockets are safe enough to board for humans and commercial planes are. With Gagarin, we wanted to have people interested in space again and understand that today's rocket technology is safe. This app is mostly directed towards a younger generation(1990+) because they are the ones that are most likely going to have the opportunity to travel to space.

AGRADECIMIENTOS



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia original completa(jurídicamente válida) que puede encontrarse en:

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>

Gagarin	2
Resumen:	3
Abstract:	4
INTRODUCCIÓN	8
Objetivos	8
Motivación	8
Antecedentes	8
DESARROLLO DEL PROYECTO	10
Herramientas tecnológicas	13
Planificación	16
Descripción del trabajo realizado	19

1. INTRODUCCIÓN

1.1. Objetivos

El objetivo de Gagarin es aumentar el interés en temas del espacio ya que hemos entrado en una nueva era de exploración espacial. Queremos demostrar que aprender sobre el espacio y cohetes es algo que cualquier persona podría hacer. Por último uno de nuestros objetivos es aumentar el número de personas que desean viajar al espacio.

1.2. Motivación

La mayoría de los jóvenes hoy en día no están interesados en temas del espacio, con Gagarin queremos demostrar que saber sobre cohetes y el espacio es algo que cualquiera puede aprender. Otra razón por la que hemos creado esta aplicación es para enseñar que los cohetes de hoy en día son más fiables que nunca y son seguros. Cuando los primeros aviones comerciales se empezaron a usar mucha gente estaba indecisa pero con los años se demostraron que eran seguros y ahora es uno de los modos de transporte más importantes. Nosotros pensamos que lo mismo se va aplicar en el área de vuelos comerciales al espacio y uno de nuestros objetivos es aumentar el número de personas que estén interesados en viajar al espacio.

1.3. Antecedentes

Para crear el prototipo primero listamos en grupo las principales funcionalidades que nos gustaría que tuviese la app que son enseñar noticias actualizadas del espacio, próximos lanzamientos, y un mapa 3D de Marte.

Con estos requisitos iniciales empezamos hacer un estudio del mercado de aplicaciones con funcionalidades y objetivos similares. Encontramos Quibi y Pinterest, dos aplicaciones de mobile de noticias que usamos de inspiración para los layouts y guía de estilos. En la aplicación Pinterest, nos gustaban los layouts por su simplicidad y el uso de cartas. Quibi era un buen ejemplo de como crear una aplicación de noticias con colores de estilo oscuros. Con estos

ejemplos, creamos nuestra guía de estilos y creamos la primera pantalla del prototipo que fue 'Lanzamientos'. Ahí se verán los próximos lanzamientos de cohetes.

Esta pantalla nos sirvió de gran ayuda para experimentar con los colores de las cartas que íbamos a usar. Decidimos hacer las cartas con un efecto que les hace parecer de cristal para darle una apariencia de espacio abierto. Las siguientes pantallas que diseñamos fueron 'Últimas noticias' y 'Mapa de Marte'.

El layout para la pantalla de últimas noticias fue inspirado por la aplicación mencionada antes llamada Pinterest. Usan cartas deslizantes y decidimos adoptar la idea ya que funcionalmente es intuitivo y fácil de navegar. Parte del objetivo de esta app es aumentar el interés y emoción sobre viajes al espacio. Para conseguir esto, usamos un fondo del planeta tierra para dar la sensación de estar viéndolo desde el espacio. Las cartas transparentes ayudan a dar la sensación de que estas detrás de un cristal mirando al planeta. Ambas aplicaciones que usamos de inspiración para nuestro prototipo nos sirvieron de antecedentes ya que tenían layouts intuitivos y un buen uso de paleta de colores oscuros.

2. DESARROLLO DEL PROYECTO

VENTANA SPLASH SCREEN

Para la realización de la ventana Splash Screen , que es la encargada de iniciar la aplicación, hemos optado por incorporar un dibujo de Marte, que es el tema base y principal de la aplicación, el título "Gagarin" que está presente en casi todas las ventanas de la app y un gif animado.

Nuestra idea para la animación de esta ventana, era hacer que el fondo girase mientras el planeta Marte se va acercando y se muestra el título de la aplicación, de esta forma da la sensación de que estamos viajando en el espacio y nos vamos acercando al planeta.

Esta idea surgió a raíz del deseo del ser humano de llegar a Marte, que será el próximo objetivo de la exploración espacial. De esta forma, y con esa idea en mente, hemos intentado darle ese efecto a la ventana.

VENTANA LOGIN SCREEN

Esta ventana es la encargada de loguear a los usuarios a la aplicación.

Los usuarios deberán estar registrados en la aplicación o disponer de una cuenta de Google o Facebook para poder acceder, si no cumplen alguno de estos requisitos, se les restringirá el acceso a la aplicación.

Esta ventana dispone de **3 métodos** de inicio de sesión:

- Inicio de sesión normal→ Para aquellos usuarios que se han dado de alta mediante la ventana de registro.
- Inicio de sesión con Google→ Para aquellos usuarios que disponen de una cuenta de google y quieren acceder de una forma más fácil y rápida sin tener que completar un formulario de registro.
- Inicio de sesión con Facebook→ Igual que Google , está destinada para aquellos usuarios que quieran acceder con Facebook.

Además cuenta con una funcionalidad adicional, y es poder restablecer la contraseña mediante un correo electrónico, sólo para los usuarios normales.

Por último, cuenta con una serie de verificaciones, como por ejemplo , verifica si el campo de usuario o contraseña están vacíos o contienen algún espacio en blanco, también verifica si el usuario ha validado su correo electrónico para poder loguearse y por último, en el caso de que hayamos pulsado sobre la opción "Forgot password" comprobará si hemos introducido algún correo electrónico en el campo de "Email" para enviarnos un correo automático para restablecer nuestra contraseña. En el caso de que no cumplamos alguna de las verificaciones anteriormente descritas se nos mostrará un toast.

Su función es registrar usuarios en la aplicación, para que puedan acceder a ella. Para realizar el diseño, hemos preferido usar el mismo estilo que la ventana de Login, mismas fuentes, tamaños, colores, fondo y desenfocado. Como podemos observar, esta ventana dispone de 4 campos editables, 1 botón y un checkbox. Para realizar el registro correctamente, se tienen que cumplir una serie de verificaciones:

- Los 4 campos no han de ser nulos
- El username no debe estar en uso, tiene que ser único
- Los dos campos de contraseña tienen que coincidir.

Una vez que el usuario haya completado debidamente los campos, se le enviará un mail a su correo electrónico (al correo que se ha introducido en el campo "Email") para poder loguearse en la app.

VENTANA NOTICIA SCREEN

En esta ventana se muestran las noticias más relevantes del mundo de la ciencia, astrología y exploración espacial. Su función es mostrar al usuario la noticia completa con los datos más relevantes y la fuente de la información. Además, los usuarios podrán comentar las noticias a través del chat y debatir con otros usuarios en tiempo real.

También podrán dar like o dislike a los comentarios de otros usuarios. Para el estilo de la ventana, hemos decidido en aplicar un diseño espacial, con un fondo estrellado y una imagen de la noticia.

Al final de la ventana, hemos incorporado el chat, para que, una vez que el usuario haya terminado su lectura, pueda comentar la noticia.

El diseño de la ventana consta de un scrollView adaptable al contenido de la noticia y el chat dispone de una vista recycler que se va actualizando con los nuevos comentarios.

VENTANA SETTINGS SCREEN

Su función es cambiar ajustes de la cuenta de los usuarios como usuario y contraseña (de momento disponible sólo para los usuarios normales, que se hayan registrado mediante la ventana Register).

Los usuarios podrán activar o desactivar las notificaciones de la aplicación en el apartado de "Notifications" o ver información sobre los creadores de la app en el about us.

Por último, dispone de un textView para cerrar sesión de la aplicación.

VENTANA HOME SCREEN

Es la pantalla principal de la aplicación, en ella se pueden ver noticias relacionadas con el espacio, próximos lanzamientos de cohetes al espacio y secciones para aprender más sobre el espacio. Los usuarios podrán interactuar con todas las cards para ver más acerca de toda la información disponible desde la app.

VENTANA MAPA MARTE SCREEN

En esta pantalla, los usuarios podrán interactuar con Marte, pudiendo hacer zoom para agrandar o disminuir el tamaño del mismo, además de poder ver sobre la superficie del planeta los lugares de aterrizaje de los rovers y puntos de interés acerca de Marte.

VENTANA ROVERS SCREEN

Desde el mapa de Marte se podrá acceder a una lista con los rovers más importantes del planeta rojo, Opportunity, Curiosity, Spirit y Perseverance.

Además se podrán ver modelos 3D de ellos además de información sobre sus misiones y fotos que han sacado de marte.

2.1. Herramientas tecnológicas

VENTANA SPLASH SCREEN

Para incorporar el gif a la ventana, ha sido necesario utilizar una librería especial llamada "pl.droidsonroids.gif:android-gif-drawable:1.2.19" que permite crear un elemento xml llamado "Gif Image View" a la cual hemos incorporado el gif. Más información en:

[pl.droidsonroids.gif:android-gif-drawable 1.2.23 on Maven - Libraries.io](https://libraries.io/maven/pl.droidsonroids.gif/android-gif-drawable/1.2.23)

Para crear y cargar las animaciones hemos utilizado la clase Animation de Android Studio que permite cargar una animación sobre un elemento del layout. Más información en:

[Android - Animations - Tutorialspoint](https://www.tutorialspoint.com/android/android_animations.htm)

VENTANA LOGIN SCREEN

Para el diseño de esta ventana se ha utilizado una serie de herramientas y tecnologías como BlurView ('com.eightbitlab:blurview:1.5.0') que es una librería que nos ofrece la posibilidad de desenfocar una imagen, textView, cardView... Más información en:

[Dimezis/BlurView: Dynamic iOS-like blur of underlying Views for Android \(github.com\)](https://github.com/Dimezis/BlurView)

Además, se ha utilizado la librería Material que ofrece posibilidades de diseño más amplias ('com.google.android.material:material:1.3.0').

[Develop - Android - Material Design](https://developer.android.com/material-design)

Se ha utilizado Firebase Authentication para loguear a los usuarios en la aplicación y comprobar que han verificado su email antes. Y además poder

configurar el login mediante Google y Facebook, ya que se necesitan las credenciales de firebase. Se han utilizado las librerías:

```
'com.google.android.gms:play-services-auth:19.0.0'  
'com.google.firebase:firebase-bom:27.1.0'  
'com.facebook.android:facebook-android-sdk:9.1.1'  
'com.facebook.android:facebook-login:[8.1)'
```

VENTANA REGISTER SCREEN

Para el diseño de la ventana se ha utilizado la librería BlurView y Material descritas anteriormente en la Ventana Login Screen.

Para dar de alta los usuarios se ha utilizado las tecnologías Firebase Authentication y Firebase Realtime Database.

[Firebase Realtime Database](#) | [Firebase Realtime Database \(google.com\)](#)

VENTANA NOTICIA SCREEN

Hemos utilizado la librería BlurView para el desenfoque, Firebase Firestore para realizar el chat, ya que es una base de datos en tiempo real que nos ofrece mayor velocidad ('com.google.firebase:firebase-firestore').

[Cloud Firestore](#) | [Firebase \(google.com\)](#)

Para obtener las noticias hemos utilizado la librería Jsoup.
(group: 'org.jsoup', name: 'jsoup', version: '1.13.1').

[Getting Started with JSOUP in Android](#) | [by Damilola Omoyiwola](#) | [Medium](#)

Permite extraer datos de páginas web y usarlos en nuestra aplicación, la página de donde se ha sacado las noticias es:

[Spaceflight Now – The leading source for online space news](#)

Para la imagen circular de los usuarios en el chat, hemos utilizado la librería Glide, que permite cargar imágenes ligeras y darles esa forma circular ('com.github.bumptech.glide:glide:4.11.0')

[Glide v4 : Fast and efficient image loading for Android \(bumptech.github.io\)](#)

VENTANA NOTICIA SCREEN

Hemos utilizado Material para el diseño de las ventanas de ajustes, Firebase Auth para la funcionalidad Logout, Firebase Realtime Database para cambiar el usuario y contraseña y Firebase Cloud Messaging para añadir las notificaciones a la app (implementation 'com.google.firebase:firebase-messaging') (implementation 'com.google.firebase:firebase-analytics').

[Configura una app cliente de Firebase Cloud Messaging en Android \(google.com\)](https://firebase.google.com/docs/cloud-messaging/android/client)

VENTANA HOME SCREEN

Hemos utilizado la librería [Jsoup](https://jsoup.org/), que sirve para recoger datos de páginas web, para recoger información de distintas páginas web. Para recoger los lanzamientos programados <https://www.spacelaunchschedule.com> y para recoger las noticias destacadas <https://spaceflightnow.com/category/news-archive/>

VENTANA MAPA MARTE Y DETALLE ROVERS SCREEN

Para ambas ventanas requería de representar un modelo 3D en la aplicación, para ello se ha optado por utilizar la librería [Rajawali 3D](https://github.com/Rajawali/Rajawali-3D): una librería para java que permite crear, modificar modelos 3D y cargar modelos 3D para visualizarlos en la aplicación.

Además se ha utilizado [Retrofit 2](https://square.github.io/retrofit/), una librería que permite hacer peticiones HTTP a páginas web para recibir datos y luego convertirlos, para hacer peticiones a un API facilitada por la NASA para sacar fotos sobre marte hechas por los diferentes rovers. Y para poder visualizarlas se ha usado [Gson Converter](https://github.com/google/gson), una librería que junto a retrofit te permite recoger un fichero JSON

de internet y parsear la información para poder gestionarla de la manera que quieras.

VENTANA LANZAMIENTOS SCREEN

Para realizar las ventanas que reciben información de los lanzamientos a través de la página spacelaunchschedule.com se ha utilizado la librería Jsoup, (group: 'org.jsoup', name: 'jsoup', version: '1.13.1').

[Getting Started with JSOUP in Android | by Damilola Omoyiwola | Medium](#)

2.2. Planificación

VENTANA SPLASH SCREEN

Como canales de comunicación hemos utilizado principalmente Whatsapp, Github y gmail.

Fabiola y Pablo se han encargado del diseño en figma de la ventana y de obtener los recursos necesarios como imágenes, iconos, tipografía e

Ionut se ha encargado de llevar el diseño a Android Studio, crear las animaciones y añadir el gif mediante la librería anteriormente mencionada. Para la realización de esta ventana hemos necesitado unas semanas ya que ha estado en constantes cambios porque hemos querido darle un toque único y acorde con el tema de nuestra aplicación Gagarin.

VENTANA LOGIN SCREEN

Como canales de comunicación hemos utilizado principalmente Whatsapp, Github y gmail.

Pablo Benito se ha encargado de realizar el diseño de la ventana en figma para posteriormente llevar la ventana a Android Studio.

Para la realización del diseño de esta ventana hemos necesitado 2 semanas ya que hemos tenido que investigar alguna librería para realizar el desenfoque y entender cómo funciona.

La programación completa de la ventana la ha realizado Ionut, comenzando por el Login normal y después la opción de Google y Facebook.

Para la programación completa de la ventana hemos necesitado más de 3 semanas ya que se ha requerido de mucha investigación y pruebas para realizar el acceso con Facebook y Google.

VENTANA REGISTER SCREEN

Se ha utilizado Github y Whatsapp como canales de comunicación.

Primero, se ha realizado el diseño en figma por Pablo Benito y posteriormente, Ionut Alin ha programado la ventana en Android Studio.

Se ha realizado el diseño y la funcionalidad en 1 semana ya que se han utilizado las mismas tecnologías que en el Login Screen.

Posteriormente, se ha añadido el campo "Username" ya que era imprescindible para otros apartados de la aplicación.

VENTANA NOTICIA SCREEN

Como canales de comunicación hemos utilizado principalmente Whatsapp, Github y gmail.

Para comenzar, Pablo Benito ha realizado el diseño en figma de la ventana e Ionut Alin se ha encargado de llevar la ventana a Android Studio y realizar su diseño y programación.

Principalmente se ha realizado el diseño de la ventana, que hemos tardado 4 días en finalizar su diseño.

Después se ha realizado el web scraping de cada noticia a la web [Spaceflight Now – The leading source for online space news](#), que hemos tardado en tenerlo 2 días al completo.

Finalmente, se ha realizado el diseño del chat, comenzando por el diseño de la cardView del mensaje, que contiene la foto del usuario, nombre de usuario, el tiempo transcurrido desde que se ha publicado el comentario y el cuerpo del mensaje, después se ha creado el adaptador del chat, que contiene los

mensajes y por último se ha realizado la programación del chat con Firebase Firestore, Firebase Realtime Database y Firebase Authentication.

VENTANA SETTINGS SCREEN

Se ha utilizado Github, Whatsapp Y Gmail como canales de comunicación. Para comenzar, Pablo Benito ha realizado el diseño en figma y Android Studio de la ventana e Ionut Alin se ha encargado de realizar la programación de las ventanas.

En primer lugar, se ha realizado la ventana de ajustes de la cuenta, seguido de la ventana de notificaciones, about us y por último la funcionalidad de cerrar sesión.

En total hemos tardado dos días en finalizar su diseño y dos días en implementar la programación.

2.3. Descripción del trabajo realizado

VENTANA SPLASH SCREEN

Esta ventana ha pasado por varias fases y diseños:

Versión 1:



Figura 1: Vista Splash Screen

Primera versión del splash screen, sin animación.

Podemos observar el característico planeta rojo , con el título de la aplicación y el espacio de fondo.

Para ello hemos utilizado 2 ImageView, uno para el fondo de la ventana y otro para la imagen de marte y un TextView para el título de la aplicación.

Hemos creado el intent con 6 s de delay para que transcurrido ese tiempo de paso

a la ventana de Login.

Versión 2:



Figura 2: Vista Splash Screen 2

Segunda versión del Splash Screen, totalmente renovado y con animación. En esta versión habíamos decidido cambiar el diseño totalmente, para dar paso a un diseño minimalista que contiene un fondo oscuro, con un gif y el característico título de la app, con una animación "Fade-in" que consiste en ir mostrando el nombre gagarin según se va cargando la pantalla, desde Alpha 0 (transparente) hasta 1.0 (Visible 100%):

```
<alpha  
    android:fromAlpha="0"  
    android:toAlpha="1.0"  
    android:startOffset='2000'  
    android:duration="2000"/>
```

Listado 1: Animación Fade-In

Versión 3:



Figura 3: Vista Splash Screen Definitiva

Versión final del Splash Screen con animación de Scale, Fade-in y gif animado.

Finalmente hemos decidido volver a renovar el diseño del splash screen, añadiendo el planeta marte ya que al tener la app un mapa de marte, era más adecuado.

Además hemos añadido un fondo del espacio animado, la animación Fade-in al texto y la animación Scale a Marte, que todo ello conjuntamente, da la sensación de viajar por el espacio e ir acercándose al planeta rojo.

Para la animación Scale hemos utilizado la etiqueta "<scale>" junto con sus atributos "FromXScale", "ToXScale", "FromYScale", "ToYScale" y "Duration"

Para hacer que la imagen vaya aumentando desde un tamaño inicial hasta alcanzar el tamaño final en un tiempo determinado:

```
<android:fromXScale="0.1"
    android:toXScale="1.0"
    android:fromYScale="0.1"
    android:toYScale="1.0"
    android:duration="3800"
```

Listado 2: Animación Scale

VENTANA LOGIN SCREEN

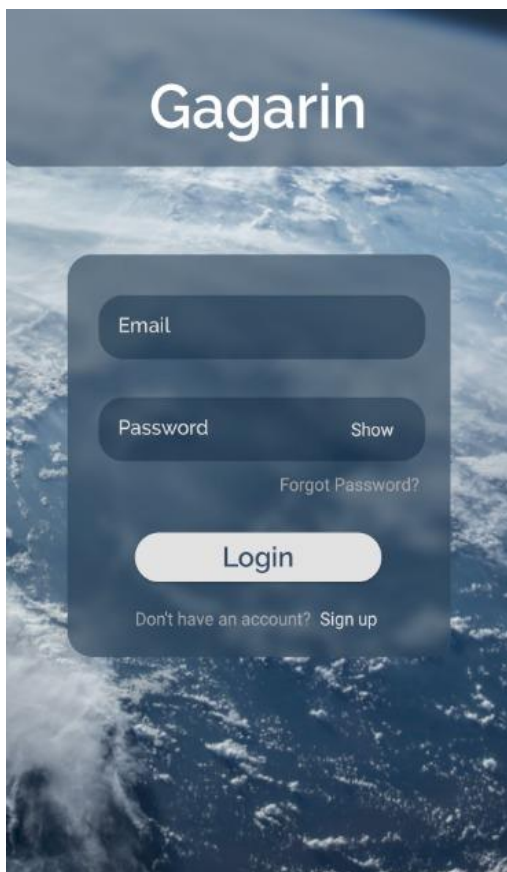


Figura 4: Vista Login 1

Versión 1:

Diseño ventana y Login usuarios normales.

En esta versión se ha realizado el diseño de la ventana, utilizando la librería Material ('com.google.android.material:material:1.3.0') para diseñar

componentes más estéticos. Los botones se encuentran dentro de un `cardView` con un fondo blurred (desenfocado) con la librería de `BlurView`, para ello:

```
<eightbitlab.com.blurview.BlurView
    android:id="@+id/blur_card"
    android:layout_width="311dp"
    android:layout_height="440dp"
    <androidx.cardview.widget.CardView
        android:id="@+id/card_content_login"
        style="@style/CardViewStyle
        ...
```

Listado 3: BlurView XML

Debemos crear una etiqueta "BlurView" en el código xml del layout e introducir dentro de esta etiqueta el `cardview` o la vista que deseemos desenfocar y mediante el código:

```
card.setupWith(root)
    .setFrameClearDrawable(windowBackground)
    .setBlurAlgorithm(new SupportRenderScriptBlur(this))
    .setBlurRadius(radius)
    .setHasFixedTransformationMatrix(true);
```

Listado 4: BlurView Java

Le asignamos un valor al `radius`, que a mayor valor, mayor desenfocado y al `cardview` le asignamos la variable `root` que es el `groupview` (el `constraint layout` que engloba toda la ventana).

Para realizar el Login normal se ha utilizado `Firebase Authentication` ya que tiene algunas funciones interesantes como el cifrado de contraseñas y enviar

correos de confirmación de email para verificar que email existe y restablecimiento de contraseña.

Se ha tenido que realizar una serie de verificaciones al pulsar sobre el botón "Login":

- Comprobar que los campos son correctos y no hay valores nulos o espacio
- Después, comprobar que se ha verificado el correo, mediante un método que ofrece Firebase Auth bastante sencillo
"firebaseAuth.getCurrentUser().isEmailVerified()"

Si todo es correcto, damos de alta el usuario en Firebase Realtime Database, con su email y username, que obtenemos del registro mediante el intent con el

getStringExtra() y que posteriormente utilizaremos el username para otras funciones.

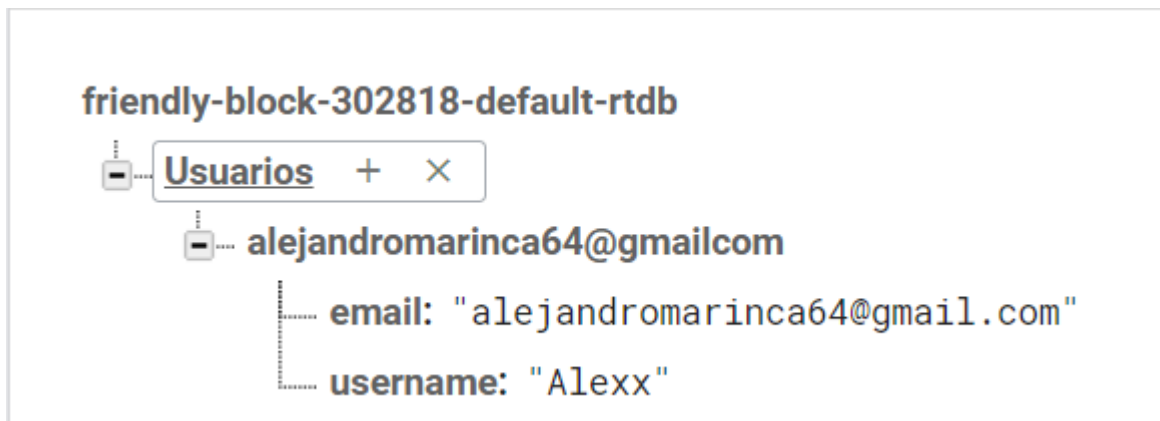


Figura 5:Firebase Realtime Database

Versión 2:

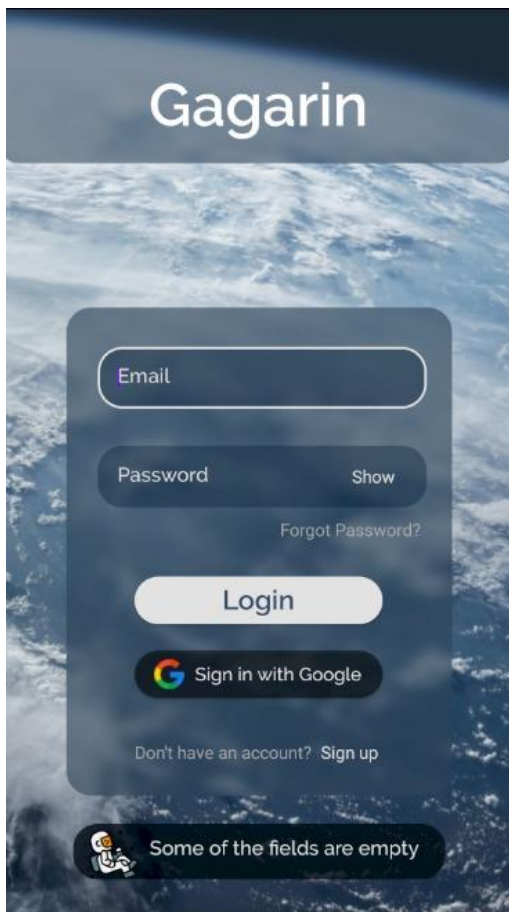


Figura 6:Vista Login 2

Login con Google y Toast personalizado.

En esta versión se ha realizado la funcionalidad de login mediante Google, para ello, como primer paso después de configurar Firebase correctamente y

agregar el json google services a nuestro proyecto, debemos habilitar en Firebase Authentication lo siguiente:




Proveedor	Estado
 Correo electrónico/contraseña	Habilitada
 Teléfono	Inhabilitado
 Google	Habilitada

Figura 7:Firebase Authentication 1

Después agregamos las librerías en el build gradle (module):

'com.google.android.gms:play-services-auth:19.0.0'

Y en el build gradle (project):

classpath 'com.google.gms:google-services:4.3.8'

Una vez en el código, debemos obtener el api de autenticación de Google para poder loguearnos con nuestra cuenta:

```
GoogleSignInOptions gso = new GoogleSignInOptions
    .Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)

    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
// Build a GoogleSignInClient with the options
specified by gso.
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

Listado 5: Google Sign Java

En el método sign in, que es llamado al pulsar el botón de google, creamos un intent y le pasamos un código de petición que es una constante de clase que le podemos dar cualquier valor. Este método invocará a la función

onActivityResult que retornará el requestCode del intent anteriormente mencionado.

```
private void signIn() {  
    Intent signInIntent =  
    mGoogleSignInClient.getSignInIntent();  
    startActivityForResult(signInIntent, RC_SIGN_IN);  
}
```

Listado 6: Google Sign Java 2

Si el Request Code concuerda con nuestra constante de clase, se llamará al método firebaseAuthWithGoogle y se le pasa como parámetro la cuenta de google creada en el anterior método para autenticarse con google.

Primero se obtiene el credencial de la cuenta que le hemos pasado por parámetros y después le pasamos la variable credential a firebaseAuth para loguearnos mediante Firebase Authentication, pero con nuestra cuenta de

google. Si todo es correcto podremos obtener los datos de la cuenta del usuario creando un `FirebaseUser` y pidiendo el usuario que está logueado:

```
private void firebaseAuthWithGoogle(GoogleSignInAccount account) {
    AuthCredential credential =
    GoogleAuthProvider.getCredential(account.getIdToken(), null);
    firebaseAuth.signInWithCredential(credential)
        .addOnCompleteListener(this, new
    OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                FirebaseUser user = firebaseAuth.getCurrentUser();
                String uid = user.getId();
                String email = user.getEmail();
                Log.d("tag", "email:"+email);
                Log.d("tag", "uid:"+uid);

                if(task.getResult().getAdditionalUserInfo().isNewUser()){
                    showToast("Account Created...");
                }
            } else {
                // If sign in fails, display a message to the
                user.
                Log.w("tag", "signInWithCredential:failure",
                task.getException());
            }
            loadMain();
            finish();
        }
    });
}
```

Listado 7: Google Sign Java 3

Por último llamamos al método `loadMain()` que se encarga de abrir mediante un `Intent` la ventana `HomeScreen`, es decir, la ventana principal y al método

finish(), que como su nombre indica, finaliza la actividad de la ventana de Login.

Versión 3:

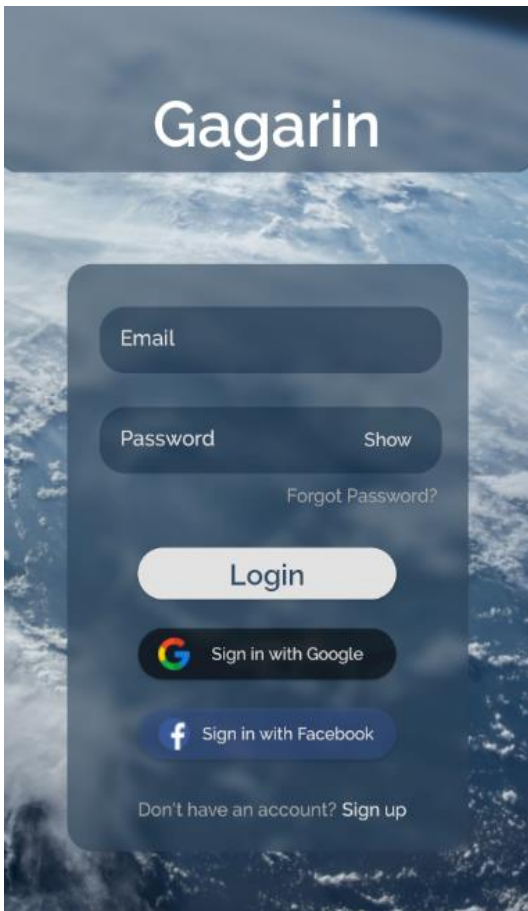


Figura 8: Vista Login 3

Versión final de la ventana, que incluye login con facebook y reseteo de contraseña.

Para realizar el login con facebook se ha utilizado Firebase Authentication y se ha habilitado la opción "Facebook" en "Sign-in methods":



Habilitada


Figura 9: Firebase Authentication 2

Se han utilizado las librerías de Facebook anteriormente descritas y se ha configurado el proyecto en Facebook Developers:

Identificador de la aplicación	Clave secreta de la aplicación
<input type="text" value="242835757644573"/>	<input type="password" value="••••••••"/> Mostrar
Nombre para mostrar	Espacio de nombres
<input type="text" value="Gagarin"/>	<input type="text"/>
Dominios de la aplicación	Correo electrónico de contacto ⓘ
<input type="text"/>	<input type="text" value="ionutalin64@gmail.com"/>
URL de la política de privacidad	URL de las Condiciones del servicio
<input type="text" value="https://friendly-block-302818.firebaseio.com/__/auth/handler"/>	<input type="text" value="Condiciones del servicio del cuadro de diálogo de inicio de sesión y l"/>

Figura 10:Firebase Authentication 3

y enlazado con Firebase Auth.


 Facebook

☒ Habilitar

ID de la app

Secreto de app

Para completar la configuración, agrega este URI de redireccionamiento de OAuth a la configuración de tu app de Facebook. [Más información](#) ⓘ



Cancelar Guardar

Figura 11:Firebase Authentication 4

Además, se ha agregado en el manifest el intent-filter del botón de la librería de Facebook:


```

<activity android:name="com.facebook.CustomTabActivity"
android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category
android:name="android.intent.category.DEFAULT" />
        <category
android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="@string/fb_login_protocol_scheme"
/>
    </intent-filter>
</activity>

```

Listado 8: Facebook Sign Manifest

En el OnClickListener del botón se llama al método registerCallback que devuelve el resultado de la autenticación , si ha sido exitoso, recuperamos la foto de perfil del usuario, que se usará más adelante y llamamos al método graphLoginRequest que utilizaremos para obtener información del usuario cómo por ejemplo el nombre de usuario:

```

login_facebook.registerCallback(callbackManager,
new FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) {
        try {
profileImg=new
URL("https://graph.facebook.com/"+loginResult.getAccessToken().getU
serId()+"/picture?type=square"+"&access_token="+loginResult.getAcce
ssToken().getToken());
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        graphLoginRequest(loginResult.getAccessToken());
        condicion_facebook = true;
        Intent intent = new Intent(LoginScreen.this,
HomeScreen.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }
}

```

Listado 9: Facebook Sign Java

Para la funcionalidad "Forgot password" se comprueba que el campo del email es válido y mediante el método `firebaseAuth.getInstance().sendPasswordResetEmail(String email)` se envía un email al usuario para restablecer su contraseña:

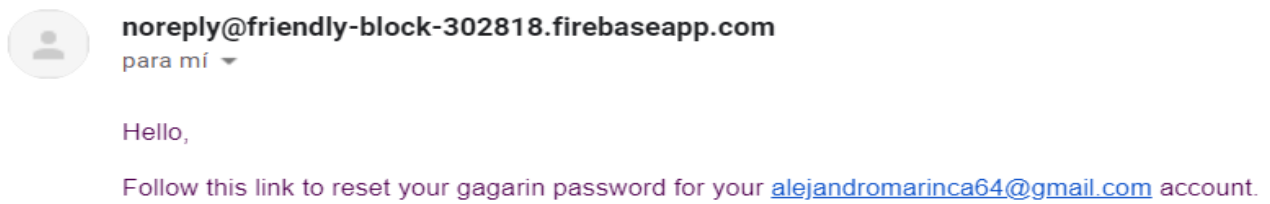


Figura 12: Login Forgot Password

VENTANA REGISTER SCREEN

Versión 1:

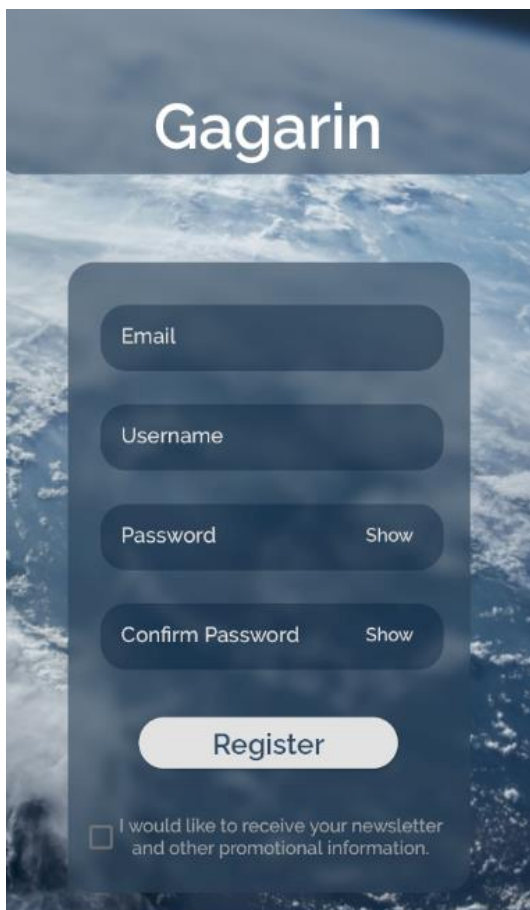


Figura 13: Vista Register

En esta versión se realiza el diseño de la vista y su funcionalidad completa.

Para realizar el registro de los usuarios se ha utilizado Firebase Realtime Database y Firebase Authentication.

Se comprueba el valor de los campos antes de dar de alta al usuario y se comprueba que el usuario no exista en Firebase:

```
public static void devolverUsuarios(RegisterScreen llamante,
NoticiaScreen llamante2, LoginScreen llamante3) {
    DatabaseReference ref = conexionBBDD();
    ref.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            Iterable<DataSnapshot> datos = snapshot.getChildren();
            for (DataSnapshot d : datos) {
                Usuario userBBDD = d.getValue(Usuario.class);

usuariosBBDD.put(userBBDD.getEmail(),userBBDD.getUsername());
            }
        }
    });
}
```

Listado 10: Get Username in RegisterScreen

Este método recupera los usuarios que ya están dados de alta en firebase y rellena un HashMap con clave, el email del usuario y valor, su username. Al pulsar en el botón comprueba que el username introducido en el campo "Username" no se encuentre en el HashMap. Si existe, se le informa al usuario de que debe elegir un username diferente.

Si todos los datos son correctos, se guarda el username y el email del usuario en Firebase y se crea el usuario en Firebase Authentication

El método crea el usuario con el email y la contraseña en Firebase Auth. Una vez que la tarea termina correctamente, se envía al usuario un email de verificación mediante el método `sendEmailVerification()`.

VENTANA HOME SCREEN

Figura 14.1: Home Screen



Figura 14.2: Home Screen II



El Home(Figura 14.1) es la pantalla principal de la aplicación, es la que se va a mostrar nada más abras la aplicación, en ella se mostrarán diferentes secciones, dichas secciones serán:

Noticias, acerca del espacio y de todos los temas relacionados con él, se mostrarán las noticias más recientes.

Lanzamientos, en esta sección se verán los 6 lanzamientos más próximos en el tiempo mostrados en diferentes cards para cada uno de ellos. Se mostrará el nombre del cohete, la fecha prevista y el modelo de dicho cohete (Figura 14.2).

Aprender más, en esta sección habrán dos cartas donde se podrá ver un mapa 3D de Marte e información sobre los rovers y una donde se podrán ver los satélites orbitando alrededor de la Tierra (Figura 14.2).

Además se tiene un slider horizontal para acceder de forma más rápida a todos los contenidos de la app (Figuras 15)

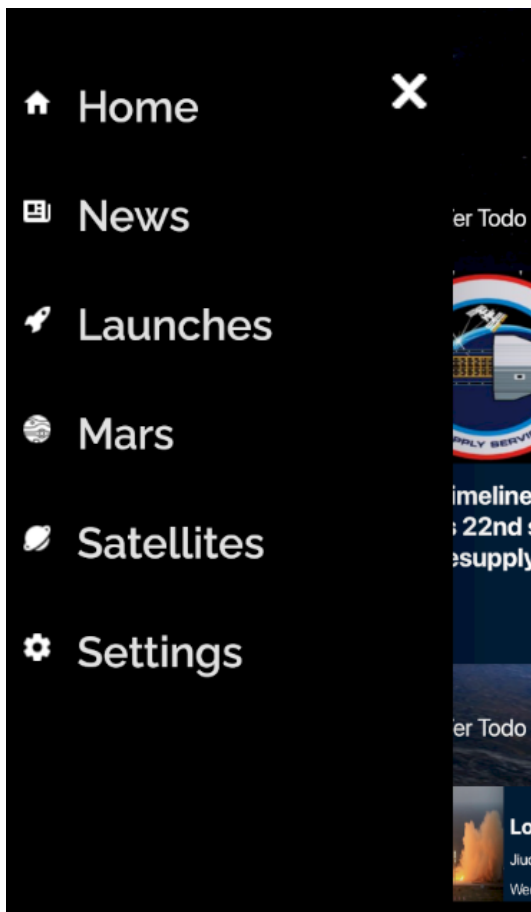


Figura 15.1: Slider menu



Figura 15.2: Slider menu

Diseño: Para el diseño del Menú hemos utilizado un cardView con ConstraintLayout que contiene los diferentes TextView e ImageView. En el xml hemos ocultado el cardview asignándole la propiedad "layout_constraintEnd_toStartOf="parent""

Animación: Para crear la animación se ha utilizado la clase ObjectAnimator para que, al pulsar el imageView del menú el cardview se deslice desde el eje $x=0$ hasta el $x=880$ y agregándole una duración de 1 segundo para crear el efecto de "slide".

```
ObjectAnimator objectAnimator= ObjectAnimator.ofFloat(menu_view_home,
"translationX", 0, 880);
objectAnimator.setDuration(1000);
objectAnimator.start();
```

La animación para ocultar el menú es prácticamente igual , salvo que ahora le decimos al ObjectAnimator que deslice el menú desde $x=880$ hasta $x=0$ al pulsar el aspa del menú.

VENTANA HOME SCREEN

Si desde la sección de cohetes en el home hacemos tap en el texto de ver más se abrirá una ventana que mostrará de una forma más visual y detallada de todos los próximos lanzamientos (figura 17).

Esta vez si hacemos tap en una de las cards nos llevará a la ventana de detalles del cohete (figura 18), a esta ventana se puede acceder también dando tap a una de las cards del home, en ella nos mostrará una cuenta atrás hasta el lanzamiento del mismo, información sobre el evento del lanzamiento como el tiempo y empresa desarrolladora del cohete e información acerca del lugar desde el que será desplegado además de un mapa que marca dónde se va a lanzar y una foto(figura 18), toda la información relativa a los lanzamientos y la información es obtenida mediante web scraping utilizando la librería Jsoup la página <https://www.spacelaunchschedule.com>.

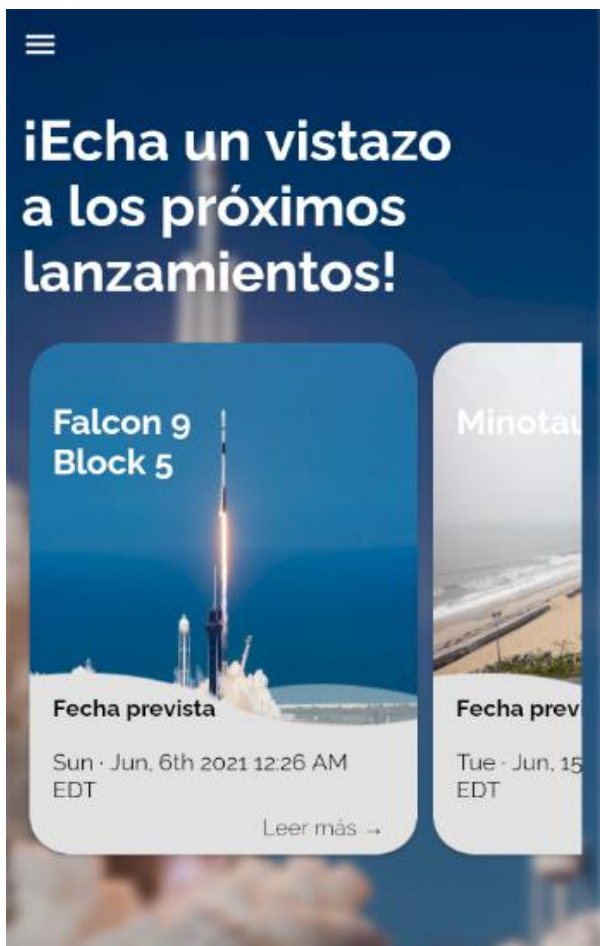


Figura 17: Lanzamientos Screen



Figura 18: Detalle Lanzamiento

Jsoup requiere mínimo de una clase Java que implemente Runnable para poder

hacer las peticiones en otro hilo y no ralentice la carga de la aplicación. Una vez tengas la clase que se va a ejecutar en un hilo aparte, es importante conectarse a la página web de la que queremos sacar la información una vez conectado se podrá recoger la información (Listado 12).

```
Document doc =  
Jsoup.connect("https://www.spacelaunchschedule.com/").get();
```

Listado 12: Conectarse a una Web.

VENTANA MAPA DE MARTE



Figura 18: Mapa 3D Marte

De vuelta al home si hacemos scroll hacia abajo veremos dos cards debajo de la sección de lanzamientos, Marte en 3D y Satélites de la Tierra (Figura 14.2), al hacer tap en el primero de ellos nos llevará a una ventana donde se verá un modelo 3D de Marte que se podrá rotar (Figura 19). Para hacer esto, se ha usado la librería Rajawali 3D. Esta librería requiere de una clase para definir y renderizar el objeto en cuestión además de un componente dentro del layout de la activity que es donde se cargará el objeto. Dentro de esta clase, definiremos, entre otras cosas, la textura del objeto que

estará en la carpeta drawable y tendrá extensión .png.

```
try{  
    material.addTexture(earthTexture);  
} catch (ATexture.TextureException error){  
    Log.d( ".initScene", error.toString());  
}
```

Listado 13: Añadir la textura al material

Una vez creada la textura debemos crear un objeto Material y añadirlo dentro de un try catch para evitar errores. Una vez hayamos añadido la textura al material se crea un objeto "Sphere" y se le añade el material, eso creará una esfera que tendrá la textura seleccionada, en nuestro caso esa textura es una imagen de la superficie de marte, entonces al crear la esfera aparecerá como si fuera un modelo de marte. Para hacer que podamos rotar ese objeto con el dedo tenemos que crear y asignar al objeto en cuestión un objeto "arcballCamera".

```
ArcballCamera arcball = new ArcballCamera(mContext,  
((Activity)mContext).findViewById(R.id.rajawali_surface)  
);  
    arcball.setTarget(mMarsSphere); //your 3D Object  
    arcball.setPosition(0,0,4); //optional  
  
getCurrentScene().replaceAndSwitchCamera(getCurrentCamera(), arcball);
```

Listado 14: Objeto que permite la rotación

VENTANA ROVER MARTE



Figura 19: Rovers de Marte

Desde el mapa de Marte se ve un texto para cambiar a la activity de los rovers que hay desplegados en Marte. Por el momento son 4, Opportunity, Curiosity, Spirit y Perseverance, en la ventana puedes hacer tap en el rover que quieras(Figura 19).

VENTANA ROVER MARTE

Una vez elegido el rover que se quiere ver, se abrirá una ventana que mostrará un modelo 3D del rover, información relativa a él y una serie de fotos sacadas por el rover recibidas gracias al API Mars Rover Photo API que proporciona la NASA (<https://api.nasa.gov>, <https://github.com/chrisccerami/mars-photo-api>).



Figura 19: Detalles Rovers de Marte

Para cargar un modelo 3D ya hecho a una aplicación Android usando Rajawali 3D hay que tener en cuenta múltiples factores.

Primero, el objeto 3D tiene que estar en un formato compatible con la librería, en nuestro caso usamos una pareja de archivos .obj (contiene el esqueleto del modelo) y un archivo .mtl para cargar las texturas del objeto (normalmente las texturas vienen acompañadas de archivos .png que habrá que guardarlas en la carpeta drawable-nodpi), una vez se tienen estos dos archivos se van a guardar con un nombre específico, nombreObjeto_obj.obj y nombreObjeto_mtl.mtl, dentro de la carpeta raw dentro de los recursos de la app (Figura 20).

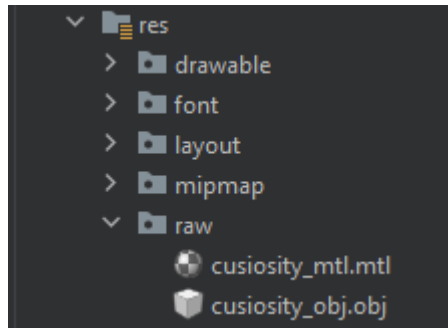


Figura 20: Directorio

Segundo, dentro del archivo .mtl hay que cambiar las rutas de las imágenes para que cuadren con la localización donde están y en el archivo .obj, hay que cambiar una de las primeras líneas a "mtllib nombreObjeto _mtl", sin la extensión del archivo.

Ahora hay que parsear el objeto personalizado para que aparezca en la aplicación. Hay que crear una nueva clase renderer y llamar crear allí un objeto LoaderOBJ (Listado 15), si se utiliza otro tipo de archivo se creará el Loader correspondiente.

```
objParser = new LoaderOBJ(mContext.getResources(),  
mTextureManager, R.raw.cusiocity_obj);
```

Listado 15: Objeto para Parsear .obj

```
try {  
    objParser.parse();  
    rover = objParser.getParsedObject();  
    ...  
}
```

Listado 16: Recibe el resultado del parseo

VENTANA NOTICIA SCREEN

Versión 1:



Figura 21: Vista Noticia Screen

Realización diseño ventana y web scraping noticias.

Hemos utilizado la librería BlurView para desenfocar la parte superior de la ventana, Material CardView para el diseño personalizado del card que contiene el cuerpo y el título de la noticia.

Además, le hemos incorporado un scrollview para que se ajuste al contenido de la noticia al deslizar la pantalla.

Para realizar la petición de noticias, se ha creado una clase Runnable que se encarga de hacer la petición de las noticias a la clase PeticionBodyNoticias que es la encargada de extraer los datos del body de cada noticia. Dependiendo de la url que le pasemos a esa clase va a extraer una noticia u otra (La url de petición de cada noticia la obtenemos del web scraping que ha realizado mi compañero Fran Ruiz en la ventana HomeScreen):

```

public class HiloPeticionBodyNoticias implements Runnable {
    private NoticiaScreen context;
    private String url_noticia;

    public HiloPeticionBodyNoticias(NoticiaScreen context, String
url_noticia) {
        this.context = context;
        this.url_noticia = url_noticia;
    }

    @Override
    public void run() {
        PeticionBodyNoticias.conectar(url_noticia);
        String cuerpoNoticia = PeticionBodyNoticias.pedirCuerpo();
        context.devolverBodyNoticias(cuerpoNoticia);
    }
    public interface InterfazBodyNoticias{
        public void devolverBodyNoticias(String cuerpoNoticia);
    }
}

```

Listado 17: Request News

Como podemos observar, tenemos la clase HiloPeticionNoticias que es una clase que implementa Runnable, que se encarga de realizar la petición de las noticias cuando invocamos al método run() del hilo.

Una vez que ha realizado la petición, devuelve los datos de la noticia a la Ventana Noticia Screen mediante la InterfazBodyNoticias{}, que deberemos implementar esta interfaz en la ventana Noticia Screen.

```

public class PeticionBodyNoticias {
    private static Document doc;
    public static void conectar(String url_noticia) {
        try {
            doc = Jsoup.connect(url_noticia).get();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static String pedirCuerpo() {
        String cuerpoNoticia = "";

        Elements newsHeadlines = doc.getElementsByClass("entry-
content");
        for (Element headline : newsHeadlines) {
            Elements phrases = headline.getElementsByTag("p");
            for(Element phrase : phrases){

```

```

        if(!phrase.text().contains("<span")){
            cuerpoNoticia += phrase.text()+"\n\n";
        }
    }}
    return cuerpoNoticia;
}
}

```

Listado 18: Request News

Esta clase se encarga de conectarse a la url de la noticia que le hemos pasado por parámetro mediante la librería jsoup.

En el método pedirCuerpo(), jsoup se encarga de buscar en la página web todos los elementos html que contienen la clase "entry-content".

Por cada elemento de la clase "entry-content" seleccionamos las etiquetas <p> que contienen el texto de la noticia y en el condicional descartamos las etiquetas que contienen texto html para quedarnos únicamente con el texto plano.

```

@Override
public void devolverBodyNoticias(String cuerpoNoticia) {
    //Set Body Story
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            tv_body_story.setText(cuerpoNoticia);
            pg_progress_story.setVisibility(View.GONE);
        }});
}

```

Listado 19: Request News

En la ventana Noticia Screen, implementamos la interfaz devolverBodyNoticias y con el método runOnUiThread nos traemos la ejecución del hilo al hilo principal y ya disponemos de los datos de las noticias.

Versión 2:

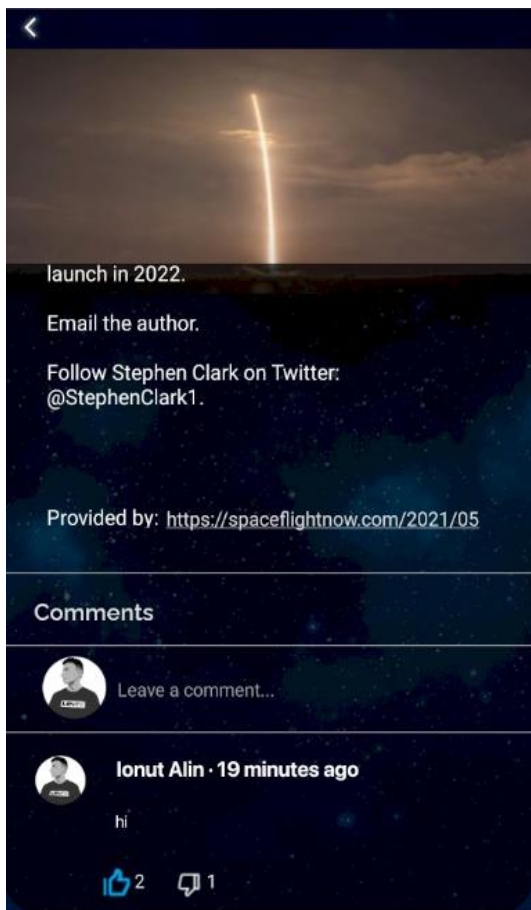


Figura 22: Vista Noticia Screen Chat

Realización del diseño del chat y funcionalidad chat.

Para la realización del diseño del chat se ha utilizado un cardView junto con un recyclerView.

Cuando escribimos en el EditText y pulsamos enter:

```
et_comment_story.setOnKeyListener(new View.OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        // If the event is a key-down event on the "enter" button
        if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
            (keyCode == KeyEvent.KEYCODE_ENTER)) {
            // Perform action on key press
            addMessageToChat();
            return true;
        }
        return false;
    }
});
```

Listado 20: Key Listener

Llama al método addMessageToChat() que se encarga de añadir el mensaje a Firestore:

```

private void addMessageToChat() {
    String message = et_comment_story.getEditableText().toString();
    String time = String.valueOf(System.currentTimeMillis());
    String message_id = tv_title_story.getText().toString();
    if(!message.trim().isEmpty()){
        Mensaje messageObj = new
Mensaje(message,username,time,message_id);
        if(firebaseAuth.getCurrentUser().getPhotoUrl()!=null &&
LoginScreen.condicion_facebook == false){

messageObj.setPhoto(firebaseAuth.getCurrentUser().getPhotoUrl().toString
());
        }else if(LoginScreen.condicion_facebook){
            messageObj.setPhoto(LoginScreen.profileImg.toString());
        }else{
            messageObj.setPhoto("por defecto");
        }

        FirebaseFirestore.getInstance().collection(message_id.substring(0,15).re
place(" ", "").replace("'", "").replace("-",
", "")) .document(messageObj.getTime()).set(messageObj);
        et_comment_story.setText("");
    }
}

```

Listado 21: Adding Message

Recuperamos el mensaje que quiere enviar el usuario, el tiempo actual en milisegundos, el username y la foto que dependiendo del tipo de login vamos a obtener estos dos datos de diferentes formas:

- Si es un usuario de Google obtenemos el username mediante `firebaseAuth.getCurrentUser().getDisplayName()` y la foto mediante `firebaseAuth.getCurrentUser().getPhotoUrl()`.
- Si es un usuario de Facebook recuperamos la foto y el username del Login mediante el `accessToken` del usuario.
- Si es un usuario que se ha logueado normal, mediante registro, buscamos el usuario en Firebase Realtime Database y le asignamos una foto por defecto.

Como clave del mensaje vamos a utilizar el tiempo, en milisegundos y como nombre de la colección vamos a utilizar los primeros 15 caracteres del titular de la noticia, de esta forma, cada noticia tendrá su propio chat.


```

FirebaseFirestore.getInstance().collection(getIntent().getStringExtra("Home").substring(0,15).replace("","").replace("-", "").orderBy("time",
Query.Direction.DESCENDING)
    .addSnapshotListener(new EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(@Nullable QuerySnapshot
queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
            for (DocumentChange mDocumentChange :
queryDocumentSnapshots.getDocumentChanges()) {
                if (mDocumentChange.getType() ==
DocumentChange.Type.ADDED) {

lista_mensajes.add(mDocumentChange.getDocument().toObject(Mensaje
.class));

                                adaptador.notifyDataSetChanged();

rv_mensajes.smoothScrollToPosition(lista_mensajes.size());}
                }}
    });

```

Listado 22: Refresh Chat

Listener que va actualizando el recyclerview cuando se inserta un nuevo mensaje en Firestone.

Funcionalidad Likes & Dislikes:

```

private void addLike(String like, int childAdapterPosition) {
    Mensaje msj = lista_mensajes.get(childAdapterPosition);
    if(true){
        List lista=msj.getUsers_who_like_list();
        lista.add(username);
        msj.setUsers_who_like((ArrayList<String>) lista);
    }
    //msj.setUsers_who_like(userss);
    msj.setLike(Integer.parseInt(like));

    FirebaseFirestore.getInstance().collection(tv_title_story.getText()
().toString().substring(0,15).replace("","").replace("'", "").repl
ace("-", "").document(msj.getTime()).set(msj);
}

```

Listado 23: Save Likes

Si el usuario pulsa sobre el tv_like o tv_dislike se llama al método addLike() o addDislike() según corresponda, que se le pasa la cantidad de likes o dislikes que tiene el comentario y la posición del recycler sobre la cual ha pulsado, se actualiza los likes o dislikes del comentario en firestone y se añade el nombre de usuarios a la lista de personas que ya han pulsado sobre el botón, para así evitar que pulse varias veces y que se vayan sumando, de esta manera, cada usuario solo puede dar like o dislike una vez a cada comentario.

Funcionalidad tiempo transcurrido del mensaje:

Para realizar esta función, hemos creado un timer en el onBindViewHolder del adaptador del recyclerview, que recupera de cada posición del adaptador la variable time del mensaje y hace un cálculo para sacar el tiempo transcurrido, restando el tiempo actual en milisegundos - el tiempo que se ha guardado en la bbdd cuando se ha añadido el mensaje, después se pasa a minutos y horas (de momento) y va actualizando el TextView cada minuto.

```

Timer timer = new Timer();
TimerTask timerTask;
timerTask = new TimerTask() {
    @Override
    public void run() {
        noticiaScreen.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Long time =System.currentTimeMillis() -
Long.valueOf(m.getTime());
                Log.d("time",time.toString());
                if (time > 0) {
                    int minutes = (int) ((time / (1000 * 60)) %
60);
                    int hours = (int) ((time / (1000 * 60 * 60)) %
24);
                    String texto =
holder.tv_id_comment.getText().toString().split(" · ")[0];
                    if(minutes>=1 && minutes <= 60){
                        holder.tv_id_comment.setText(texto+" · "+
minutes +" minutes ago");
                    }if(hours>=1 && hours<=60){
                        holder.tv_id_comment.setText(texto+" · "+
hours +" hours ago");
                    }
                }
            }
        });
    }
};
timer.schedule(timerTask, 0, 60000);

```

Listado 24: Timer

VENTANA SETTINGS SCREEN

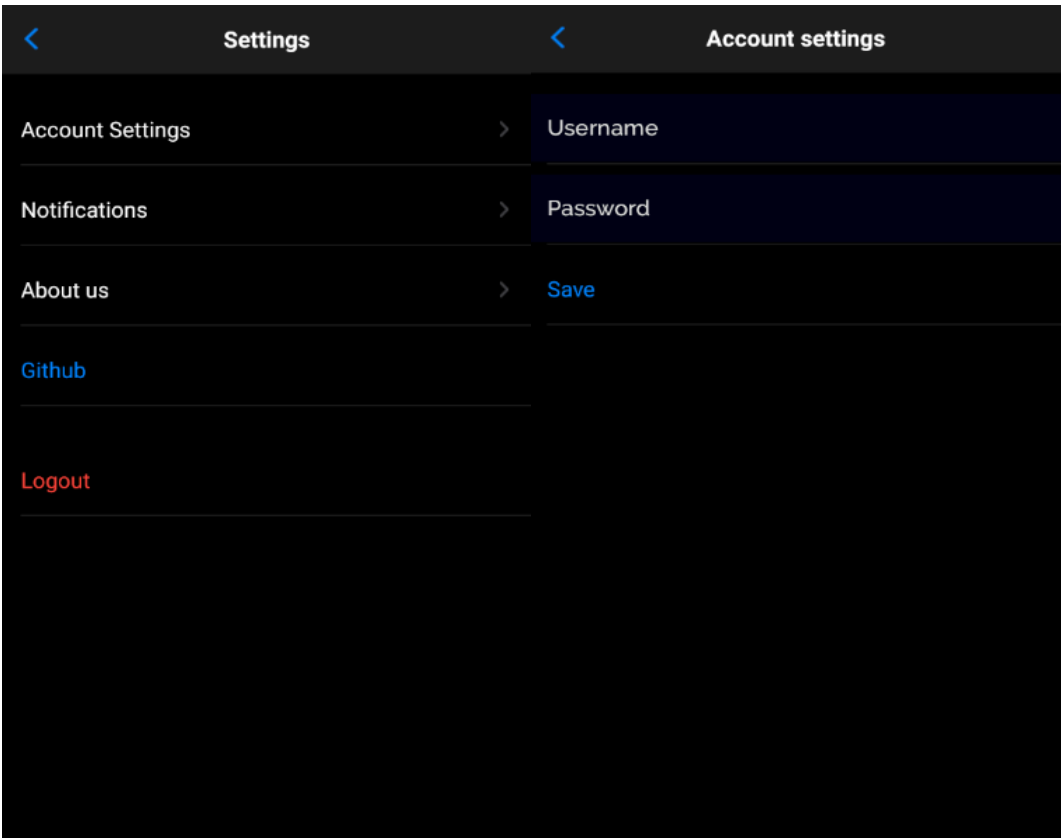


Figura 23.1: Settings

Figura 23.2: Account Settings

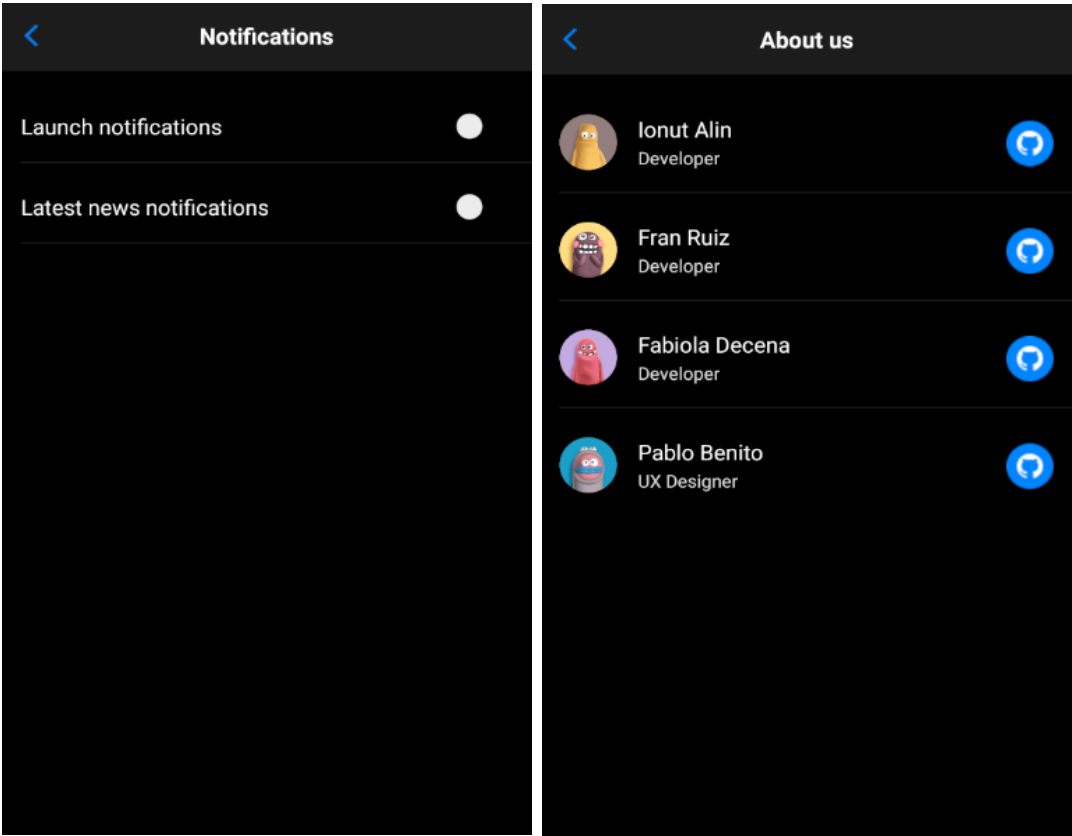


Figura 23.3: Notifications

Figura 23.4: About us

Funcionalidad Logout:

Para conseguir la funcionalidad de cerrar sesión de los usuarios normales se ha utilizado el método `FirebaseAuth.getInstance().signOut()`

Funcionalidad Notifications:

Para ello hemos utilizado Firebase Cloud Messaging , hemos configurado Cloud Messaging y hemos creado las notificaciones y desde el código accedemos a la notificación mediante la clase `RemoteService`.

Funcionalidad Account Settings:

Al pulsar sobre el imageView "Save" se recuperarán los datos de los editText y se sobrescribirán en Firebase Realtime Database.

Funcionalidad About us:

Cada vez que el usuario pulsa sobre un ImageView con la foto de un desarrollador de la aplicación se le abrirá un webView con la cuenta de Github.

CONCLUSIONES

En finalización de plazos, hemos logrado plasmar nuestras ideas y planteamientos principales en lo que se ha convertido "Gagarin", materializando desde sus inicios su identidad como aplicación y los objetivos que le destinamos, obteniendo un entorno que brinde a través de una gentil e interactiva interfaz, información e interés hacia el espacio y las novedades de la interacción que se ha logrado a lo largo de los años con este, ya que muy pocos recursos de interés se le muestran al público a atraer. Gagarin ha cumplido nuestras expectativas de creación, iniciando otras de crecimiento y expansión.

4. BIBLIOGRAFÍA Y WEBGRAFÍA

- DEV.TO.(2021) <https://dev.to/>. Fecha de consulta: 09:18, mayo 28,2021 de [Cloud Firestore Basics in Flutter: How to Get, Add, Edit, and Delete Data in Cloud Firestore, Demonstrated in a Real Flutter App - DEV Community](#)
- TUTORIALSPPOINT(2021) <https://www.tutorialspoint.com/> . Fecha de consulta: 13:08, mayo 28,2021 de [How to use Fade In and Fade Out Android Animation in Java? \(tutorialspoint.com\)](#)
- STACKOVERFLOW(2021) <https://stackoverflow.com/> . Fecha de consulta: 16:16, mayo 24,2021 de [com.google.android.gms.common.api.ApiException: 12500 - Stack Overflow](#)
- YOUTUBE(2021) <https://www.youtube.com/> . Fecha de consulta: 09:01, mayo 21,2021 de [\(98\) formaprofestic - YouTube](#)
- STACKOVERFLOW(2021) <https://stackoverflow.com/> . Fecha de consulta: 16:16, abril 30,2021 de [Edit text Password Toggle Android - Stack Overflow](#)
- STACKOVERFLOW(2021) <https://stackoverflow.com/> . Fecha de consulta: 12:16, abril 30,2021 de [Android material design: remove hint animation - Stack Overflow](#)
- STACKOVERFLOW(2021) <https://stackoverflow.com/> . Fecha de consulta: 14:19, abril 29,2021 de [java - Hide or remove Action Bar on specific Activity | Android - Stack Overflow](#)
- YOUTUBE(2021) <https://www.youtube.com/> . Fecha de consulta: 12:10, abril 29,2021 de [\(98\) 2019 Awesome Blurry Login UI & Sign up design Android Studio XML tutorial With Simple Codes - YouTube](#)
- GITHUB(2021) <https://www.github.com/> . Fecha de consulta: 17:07, abril 29,2021 de [Awesome-looking-Blurry-Login-Page-UI-design-2019/app/src at master · MonsterTechnoGit/Awesome-looking-Blurry-Login-Page-UI-design-2019 \(github.com\)be](#)