Estudio de Sistemas de redes de comunicación satelital de baja latencia

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio de 2022

Autor:

Facundo A. Sierra Rodríguez

Tutor:

Angel L. Sánchez Lázaro

VNiVERSiDAD D SALAMANCA
CAMPUS DE EXCELENCIA INTERNACIONAL

DEFENSA DEL TRABAJO FIN DE:
Grado en Ingeniería Informática

Título

Estudio de Sistemas de redes de comunicación satelital de baja latencia
(Referencia: BRKRF)

Title*

Study of low latency satellite communication network systems

FACULTAD DE CIENCIAS

Datos del estudiante

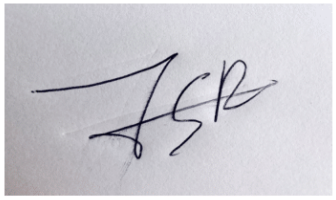
Nombre y apellidos: Facundo Sierra Rodríguez
DNI: 70970389W
Correo electrónico: facundosierra@usal.es
Teléfonos: 607934522, 622204869
Dirección: Calle Frontón
37129 Parada De Arriba (Salamanca)

Presenta el trabajo para su defensa en la convocatoria de Septiembre del curso académico 2022/2023

El tutor considera que el trabajo es Apto para ser presentado (nombre y firma)

Sánchez Lázaro, Ángel Luis	Angel Luis Firmado digitalmente por Angel Luis Nombre de reconocimiento (DN): cn=Angel Luis, o, ou, email=lacertificada@certificado, c=ES Fecha: 2022.09.06 12:59:46 +02'00'

Firma del estudiante



Fecha 05/09/2022

(*) Obligatoriamente se debe cumplimentar el título en inglés para su inclusión en el SET (Suplemento Europeo al Título)

Sr./a. PRESIDENTE/A DE LA COMISIÓN DE TRABAJOS FINALES GRADO EN INGENIERÍA INFORMÁTICA

Resumen

Este Trabajo de Fin de Grado estudia las diferentes propuestas de constelaciones de satélites de órbita terrestre baja que plantean ofrecer conexión a internet de baja latencia y describe el desarrollo de una aplicación nativa para Android que calcula la posición de cualquier satélite que desee el usuario, entre otras funcionalidades.

Hoy en día, se ofrece conexión de gran velocidad en cualquier ciudad de un tamaño mínimo, gracias a las tecnologías de fibra óptica. Sin embargo, en poblaciones más pequeñas y apartadas la conexión que se puede llegar a lograr es mucho más limitada o nula.

Para dar solución a este problema, se plantean sistemas que utilizan satélites con órbitas bajas para ofrecer una conexión rápida y eficaz. Al tener órbitas bajas estos satélites permiten una mayor velocidad de conexión, pero su cobertura es mucho menor. Debido a esto, se plantea el uso de constelaciones de satélites para poder ofrecer una cobertura global. Una de las propuestas más conocidas es la de SpaceX Starlink, que ya tiene varios miles de satélites en órbita y se encuentra operativa. Sin embargo, existen otros proyectos como Kuiper, OneWeb o Telesat Lightspeed que merece la pena estudiar.

En la presente memoria se estudian las características de estas propuestas y se documenta el desarrollo de una aplicación móvil nativa para teléfonos Android en la que el usuario puede acceder a detalles sobre cualquier satélite deseado para poder estudiar las carácteristicas de las órbitas de estos satélites y compararlas con las de otros tipos de satélites.

Además, la aplicación calcula la posición del satélite respecto del usuario y ofrece una simulación en la que el usuario podrá saber dónde tiene que mirar para estar apuntando a cualquier satélite deseado. Esto puede ayudar en la orientación de una antena direccional para comunicarse con el satélite.

Palabras clave: Satélite, Internet, Órbita, Baja Latencia

Abstract

This Final Degree Project studies the different proposals of low-Earth orbit satellite constellations that offer low latency internet connection and describes the development of a native application for Android that calculates the position of any satellite the user wants, among other functionalities.

Today, high-speed connection is offered in any city of a minimum size, thanks to fiber optic technologies. However, in smaller and more remote towns, the connection that can be achieved is much more limited or non-existent.

To solve this problem, systems using satellites with low orbits are proposed to provide a fast and efficient connection. These low orbits allow a higher connection speed, but their coverage is much lower. Because of this, the use of constellations of satellites is proposed to offer global coverage. One of the best known proposals is SpaceX Starlink, which already has several thousand satellites in orbit and is operational. However, there are other projects such as Kuiper, OneWeb or Telesat Lightspeed that are worth studying.

This report studies the characteristics of these proposals and documents the development of a native mobile application for Android phones in which the user can access details about any desired satellite in order to study the characteristics of the orbits of these satellites and compare them with those of other types of satellites.

In addition, the application calculates the position of the satellite with respect to the user and offers a simulation in which the user will be able to know where he/she has to look to be pointing to any desired satellite. This can help in orienting a directional antenna to communicate with the satellite.

Keywords: Satellite, Internet, Orbit, Low-latency

Índice

1. Introducción

En la presente memoria se describen de forma general los principales aspectos del desarrollo del proyecto. A continuación se concretan los apartados:

- Estado del arte: En este apartado se explica el estado actual de las tecnologías relacionadas con el tema que se estudia, además de describir conceptos relacionados como el de constelación de satélites o satélites de órbita terrestre baja.
- Objetivos: Se explican los objetivos del sistema.
- Conceptos teóricos: Se detallan los conceptos teóricos necesarios para comprender el desarrollo de la aplicación, como los sistemas de coordenadas, los elementos keplerianos de una órbita y el proceso de propagación de un órbita.
- Herramientas y metodología: Se detallan las herramientas utilizadas para abordar el desarrollo de la aplicación, tales como lenguajes de programación, entornos de programación, librerías y metodologías.
- Aspectos relevantes del desarrollo: Se detallan los aspectos más relevantes del desarrollo como pueden ser la planificación, los problemas que han surgido durante el desarrollo, el proceso de documentación técnica, etcétera.
- Conclusiones: Se exponen las conclusiones a las que se ha llegado tras finalizar el desarrollo del sistema.
- Líneas de trabajo futuras y posibles mejoras: Se manifiestan las posibles mejoras que se podrían aplicar al proyecto.
- Bibliografía: Se detallan los recursos que se han utilizado para la realización de la presente memoria.

Mientras que en esta memoria están los aspectos más fundamentales, si se desea ampliar la información se pueden consultar diferentes anexos que se han desarrollado para tal fin:

- Anexo I, Planificación del proyecto: Se explica en detalle la metodología utilizada, Scrum, y se describe el proceso de cada iteración.
- Anexo II, Especificación de requisitos: Se enumeran y detallan los requisitos del sistema.
- Anexo III, Manual de usuario: Se explica como manejar la aplicación desarrollada desde el punto de vista del usuario.

2. Estado del arte

a. Introducción

En esta sección se explicará el estado actual de las tecnologías de Internet por satélite de baja latencia. Para facilitar la comprensión de estas tecnologías, a continuación se describen algunos conceptos relevantes.

i. Satélites de órbita terrestre baja

Se denomina satélite de órbita terrestre baja a aquel que orbita la tierra con una altura de 160 a 1000 kilómetros sobre la superficie terrestre. Para mantener una órbita a esta altura es necesario que la velocidad de órbita sea muy grande, alrededor de 7 kilómetros por segundo.

Debido a esto, este tipo de satélite posee un periodo orbital muy pequeño, menor a 128 minutos. Es decir, un satélite con este tipo de órbita completará no menos de 11 periodos por día. Debido a su altura, su cobertura se limita a una pequeña parte de la superficie terrestre. Esto provoca que para ofrecer cobertura completa a un usuario se deben tener varios satélites para que este pueda comunicarse con uno u otro dependiendo de cual le ofrece cobertura en un determinado momento.

Figura 1: Altura de una órbita terrestre baja

ii. Satélites de órbita heliosíncrona

Se dice que un satélite orbita con una órbita heliosíncrona cuando cubre una determinada latitud a un mismo tiempo sidéreo local. Para lograr que una órbita haga esto, se juega con la inclinación y la altitud (Rutgets.edu, 2019).

iii. Constelación de satélites

Recibe el nombre de constelación de satélites aquel conjunto de satélites que cumple una función, actuando entre todos como un único sistema. El principal uso que se da a las constelaciones de satélites es el de dar cobertura en un territorio determinado en todo momento. Un ejemplo muy conocido es el de la constelación GPS, formada por 24 satélites distribuidos en varios planos orbitales.

b. Starlink

Starlink es un proyecto de la empresa SpaceX que tiene como objetivo proveer de una conexión a internet de banda ancha en toda la superficie terrestre. Para conseguir este objetivo, se está poniendo en marcha una gran constelación de satélites de órbita terrestre baja, comenzando con los primeros satélites en el año 2019.

Actualmente la constelación está formada por varios miles de unidades, y se plantea llegar a más de 30 000, cifra que preocupa a los astrónomos.

Figura 2: Logo de Starlink

i. Las cinco capas o "Shells" de Starlink

Los satélites Starlink están divididos en grupos con diferentes características denominados "shells". Los grupos se diferencian entre sí en función a la inclinación de la órbita y a la altitud. Cada una de estas capas posee una cantidad de satélites objetivo con el fin de proveer cobertura en toda la superficie. A continuación se describen brevemente.

- Shell 1: Las unidades de esta capa tienen una inclinación de 53º, una altura de 550 kilómetros y se planea que haya un total de 1584 distribuidos en 72 planos orbitales con 22 satélites por plano.
- Shell 2: Con una inclinación de 70º, orbitan a una altura de 570 km y el objetivo es que sean unos 720 satélites en 36 planos orbitales con 20 satélites por plano

- Shell 3: Esta capa está formada por satélites que poseen órbitas polares. En concreto de 97.6º y con una altura de 560 kilómetros. Se busca que haya un total de 348 satélites en esta capa. Se distribuye en 6 planos orbitales con 58 satélites por plano.
- Shell 4: Esta capa es idéntica a la anterior salvando la altura de la órbita que sería de 540 kilómetros. El número de satélites y la distribución de estos en planos orbitales es la misma.
- Shell 5: Se trata de otra capa con órbitas polares de 97.6º, pero en este caso el número de satélites se reduce a 172 satélites distribuidos en 4 planos de 43 satélites cada uno.

Algunas de estas capas casi llegan al objetivo planteado, como por ejemplo la Shell 1. Sin embargo, las capas 3 y 5 están aún en fase temprana. En concreto, la shell 3 no llega a 50 satélites y la 5 aún no posee ningún satélite en órbita.

ii. Problemática con Starlink y los astrónomos

La comunidad de astrónomos ha expresado su descontento con la idea de lanzar tal cantidad de satélites en órbita alrededor del planeta. En concreto, manifiestan que dificultará el proceso de observación del cosmos, ya que interfieren en las imágenes captadas por los telescopios de larga exposición.

En un principio los satélites de Starlink eran visibles a simple vista hasta en ciudades con contaminación lumínica, ya que reflejaban la luz solar en sus paneles solares.

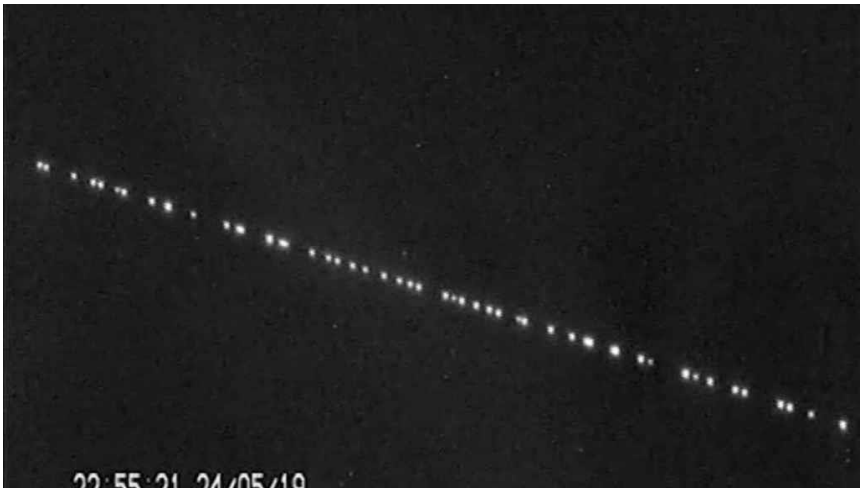


Figura 3: Hilera de satélites Starlink tras su lanzamiento

Sin embargo, tras la queja de la comunidad científica, la compañía de Musk se comprometió a revisar sus diseños para ofrecer una solución con satélites que no reflejaran con tanta intensidad la luz solar. El resultado logró disminuir el brillo en un 30%, pero se plantea que no es suficiente.

iii. Starlink V1.5, Starlink V2 y la conectividad móvil

En la versión 1.5 de Starlink, lanzada en enero de 2021, los satélites son capaces de comunicarse entre sí, en la versión 1.0 esto no era posible y la señal tenía que pasar por estaciones en tierra.

Recientemente, el CEO de SpaceX ha anunciado que se está trabajando en lo que ha denominado Starlink V2, que en principio permitirá la posibilidad de transmitir conectividad a la red directamente a dispositivos móviles con el objetivo de que internet sea accesible en cualquier lugar del planeta. Para llevar a cabo esta tarea, SpaceX trabajará con la empresa de telefonía móvil T-Mobile.

Según el comunicado, estará disponible en 2023 y la velocidad será de 2 a 4 MBits, por lo que será adecuado para realizar llamadas o enviar mensajes de texto pero no para operaciones más demandantes que requieren más ancho de banda. Para esta actividad requerirán de otro modelo de satélite Starlink más grande y pesado.

c. Kuiper

Project Kuiper, o también denominado Kuiper Systems LLC es un proyecto de la empresa Amazon que tiene como objetivo desplegar una constelación de satélites para ofrecer servicios de internet a todo el mundo. Aún no se ha lanzado ningún satélite, pero se planea que en los próximos 5 años se vayan a lanzar un total de 3,236 satélites en 83 operaciones.

Figura 4: Logo de Project Kuiper

d. Telesat Lightspeed

Telesat es una compañía de comunicaciones por satélite canadiense fundada en el año 1969. En el año 2016 anunció su intención de lanzar una constelación de 120 satélites distribuidos en 6 planos orbitales, pero más adelante se amplió la cifra a 1600 satélites.

Figura 5: Logo de Telesat Lightspeed

e. OneWeb

Se trata de una compañía de comunicaciones que planea construir una constelación de satélites y que tiene como mercado objetivo a negocios, gobiernos y operadores móviles. Actualmente han lanzado más de 200 satélites de los 648 totales que espera lanzar. En julio de 2022 anunciaron su unión con la empresa Eutelsat.

3. Objetivos

Se describen los objetivos que se pretenden cumplir para el desarrollo del TFG.

a. Objetivos generales del proyecto

i. Estudio de propuestas actuales

Se estudiarán las propuestas de conexión a internet a través de constelaciones de satélites de baja frecuencia, en particular las redes OneWeb, Kuiper y Starlink.

ii. Desarrollo de una aplicación nativa en android

Se desarrollará una aplicación android que permita visualizar en pantalla los satélites visibles desde la localización del usuario, así como información sobre dichos satélites. La ubicación del satélite se realizará propagando la órbita a partir de unos datos conocidos de posición y velocidad de un satélite en un determinado instante, además de los que caracterizan la órbita. Estos datos son accesibles de manera pública en diferentes fuentes y formatos. En el caso de este sistema, se utilizan ficheros en formato TLE, obtenidos realizando llamadas a una API externa. El proceso de propagación se describe en detalle más adelante.

Se pretende además que el software desarrollado cumpla con unos estándares de calidad, a saber: que sea funcional, estable, eficiente, escalable, reusable, y que el rendimiento sea el adecuado, además de estar documentado correctamente.

b. Objetivos funcionales

En esta sección se exponen los objetivos funcionales del sistema divididos en paquetes. Para más información consultar el Anexo II, Especificación de Requisitos.

OBJ-01	Gestión de propagación de órbita
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Descripción	*El sistema debe ser capaz de propagar la órbita de un satélite a partir de un TLE, obteniendo la anomalía verdadera en un instante de tiempo.*
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Figura 6: Tabla de OBJ-01

OBJ-02	Gestión de coordenadas
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Descripción	*El sistema debe proporcionar al usuario la posibilidad de establecer las coordenadas del observador y las coordenadas de un satélite en función al observador.*
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta

Comentarios	Ninguno

Figura 7: Tabla de OBJ-02

OBJ-03	Gestión de búsqueda y filtrado de satélites
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Descripción	*El sistema debe proporcionar al usuario la posibilidad de buscar un conjunto de satélites.*
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Figura 8: Tabla de OBJ-03

OBJ-04	Gestión de vista de detalle
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Descripción	*El sistema debe proporcionar al usuario la posibilidad de ver todos los detalles de un satélite.*
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Figura 9: Tabla de OBJ-04

OBJ-05	Gestión de simulación

Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Descripción	*El sistema debe proporcionar al usuario la posibilidad de acceder a una simulación en la que pueda conocer la orientación a la que tiene que mirar para apuntar a un satélite.*
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Figura 10: Tabla de OBJ-05

OBJ-06	Gestión de ajustes
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Descripción	*El sistema debe proporcionar al usuario la posibilidad de modificar sus preferencias*
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Media
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Figura 11: Tabla de OBJ-06

c. Objetivos personales

Se busca poner en práctica las competencias adquiridas en el plan de estudios del grado, en especial sobre los bloques de Programación, Ingeniería de Software, Sistemas Operativos e Interacción Persona-Ordenador.

Por otra parte, conocer el desarrollo de aplicaciones móviles nativas, y en concreto familiarizarse con el entorno de desarrollo integrado Android Studio utilizando como lenguaje de programación Kotlin.

Dentro de la mecánica orbital, conocer el proceso de propagación de órbita para satélites artificiales que orbitan la Tierra a partir de los ficheros TLE proporcionados por la NORAD.

También se ha planteado el objetivo de aplicar en un proyecto real el marco de trabajo SCRUM para desarrollo ágil de software.

4. Conceptos teóricos

a. Elementos de dos líneas (TLE). Elementos orbitales.

Un TLE (Two-line element set) es un formato de datos que contiene una lista de parámetros orbitales que describen sin ambigüedades la órbita de un satélite terrestre. De entre todos los datos que provee un TLE, seis de ellos son de interés para hallar la posición de un satélite en un instante determinado. Estos son denominados elementos orbitales.

En el campo de la mecánica orbital, los elementos orbitales son una serie de parámetros necesarios para especificar las características de la órbita de un astro para el problema de los dos cuerpos.

Se trata de una simplificación que describe una órbita ideal ignorando las perturbaciones provocadas por la relatividad general y la fuerza gravitatoria de objetos externos. Esta representación idealizada de una órbita real se denomina órbita de Kepler. Debido a esto, los elementos orbitales son también llamados elementos keplerianos. Se describen brevemente a continuación.

- Excentricidad, e: Es la relación entre la semidistancia focal y el semieje mayor de la elipse que representa la órbita. En el caso de los satélites que estudiamos, tiene un valor en el intervalo $[0, 1)$. Una elipse con excentricidad cero se trata de una circunferencia.
- Semieje mayor, a: Suma de las distancias de perigeo y apogeo dividida entre dos.
- Longitud del nodo ascendente, (Ω): Representa el ángulo que se forma entre el punto de aries y el punto en el que la órbita cruza el plano orbital.
- Inclinación, i: Inclinación vertical respecto del eje de referencia.

- Argumento del perigeo, (ω): Ángulo entre el nodo ascendente y el punto más cercano a la tierra de la órbita, llamado perigeo.
- Anomalía media en la época, (M_0): Ángulo ficticio que representa la fracción del periodo orbital que el satélite ha recorrido en el instante de la medición, expresado en grados.

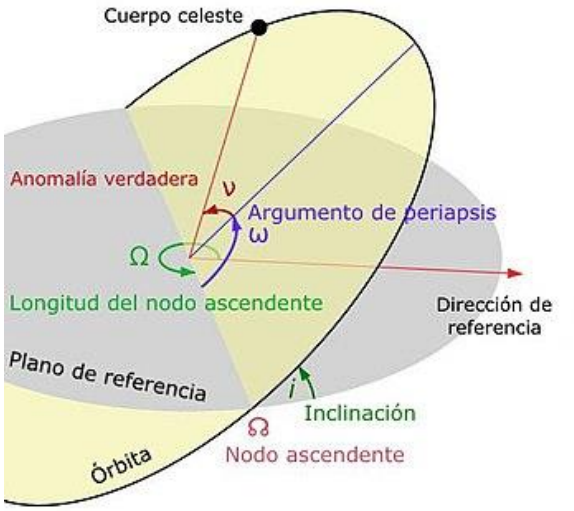


Figura 12: Elementos Keplerianos

En los TLE, en lugar del semieje mayor aparece la velocidad angular, n, ya que ambos influyen en el mismo grado de libertad.

b. Sistemas de coordenadas

En este apartado se describen los sistemas de coordenadas utilizados para el desarrollo de la aplicación.

i. Introducción

Dentro del campo de la geometría, un sistema de coordenadas se define como un sistema de referencia que determina inequívocamente la posición de un objeto en función a uno o más números denominados coordenadas. Existen varias formas de representar unas coordenadas. Las más significativas para el proyecto se describen a continuación.

ii. Coordenadas cartesianas

También llamadas coordenadas rectangulares, las coordenadas cartesianas reciben su nombre en honor al filósofo, matemático y físico René Descartes, que fue el primero en formalizarlas y extender su uso.

Esta representación se basa en expresar cada coordenada en función a las distancias respecto de n hiperplanos (siendo n la dimensión del espacio) que se cruzan de forma perpendicular en un punto denominado origen. Es decir, en un espacio de dimensión 3, cada coordenada se expresa en función a la distancia respecto de los 3 hiperplanos (de dimensión n-1 = 2) X=0, Y=0 y Z=0. En la siguiente figura se observa un ejemplo de ejes de coordenadas cartesianos para un espacio de dos dimensiones.

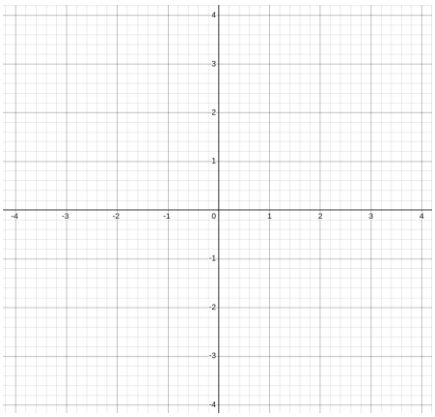


Figura 13: Ejes de coordenadas cartesianas

iii. Coordenadas polares

Esta representación se basa en expresar una posición del espacio en función a la distancia respecto de un origen y al ángulo en sentido antihorario que se forma entre un plano horizontal que contiene al origen y una recta que pasa por el origen y el punto que deseamos representar.

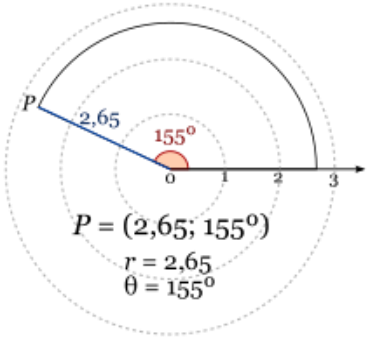


Figura 14: Coordenadas polares

iv. Coordenadas esféricas

Si queremos trasladar la solución de las coordenadas polares a un espacio de tres dimensiones, basta con incluir un tercer parámetro que representa el desplazamiento del punto en el plano horizontal que pasa por el origen.

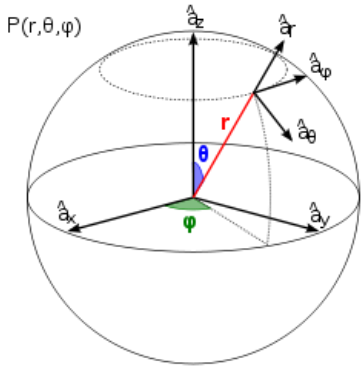


Figura 15: Coordenadas esféricas

Tenemos entonces tres parámetros:

- Radio, (r): Distancia entre el origen O y el punto P que se desea representar.
- Azimutal, (φ): representa el ángulo que se forma entre un eje de referencia y la proyección sobre el plano horizontal de la recta r. Toma valores del intervalo $[0, 2\pi)$.

- Colatitud, (θ): representa el ángulo que se forma entre la recta paralela al eje perpendicular del plano horizontal que pasa por el origen y la recta r. Puede tomar valores del intervalo $[0, \pi]$.

v. Coordenadas geográficas

Para representar una posición horizontal en la superficie terrestre, se utilizan dos parámetros que se describen a continuación.

- Latitud, (ϕ): Ángulo entre el plano ecuatorial y la recta que pasa por el punto a representar y el centro de la tierra. Es el mismo concepto que la colatitud vista en el apartado anterior.
- Longitud, (λ): Es el ángulo que se forma entre el meridiano de Greenwich y el meridiano que contiene al punto a representar.

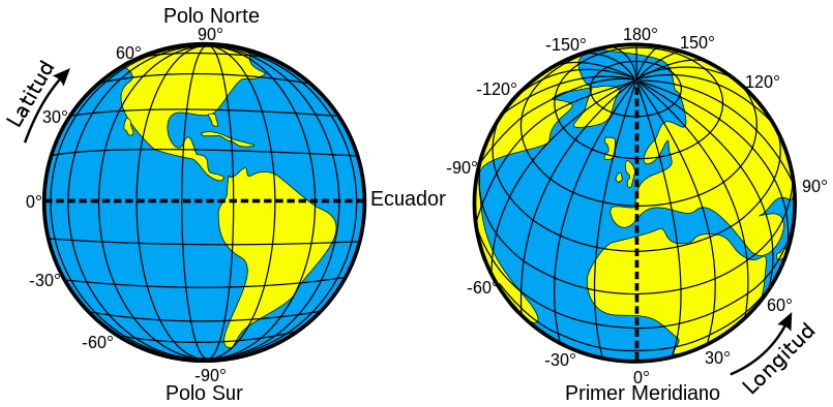


Figura 16: Coordenadas geográficas

vi. Coordenadas locales

Se llaman coordenadas locales a aquellas coordenadas que dependen de la posición del observador. Es decir, para dos observadores situados en dos posiciones diferentes, un mismo se representa con coordenadas diferentes.

c. Propagación de órbita

Se explican los detalles relevantes sobre el proceso de propagación de órbita. Se describirá el modelo utilizado. Este proceso resulta en la obtención de las coordenadas locales esféricas de un satélite a partir de su TLE y la posición de un observador.

i. Introducción

Para abordar la propagación de una órbita existen varios métodos. En este apartado se describen algunos de ellos.

1. Modelo Kepleriano de los dos cuerpos

En este modelo se tienen en cuenta solamente los efectos gravitatorios producidos por los dos cuerpos que se estudian. En nuestro caso concreto, serán la Tierra y el satélite que orbita alrededor como consecuencia de la atracción gravitatoria. Además, se asume que la tierra es esférica y se le trata como una masa puntual. Es decir, se considera que toda la masa y en consecuencia todo el efecto gravitacional de la Tierra se encuentran en su centro.

Este modelo permite obtener resultados rápidamente y gracias a esto se pueden propagar varios millares de órbitas en cuestión de fracciones de segundo. Sin embargo, se trata de una aproximación y no se obtendrán soluciones tan precisas como las que ofrecen otros modelos.

```
com.example.satellitetracker D/TIEMPO: Se han propagado 3320 órbitas en 72 milisegundos
```

Figura 17: Tiempo de propagación de órbitas con el modelo Kepleriano

2. Modelos de perturbaciones simplificados

Dentro de esta categoría se encuentran cinco modelos: SGP, SGP4, SDP4, SGP8 y SDP8, siendo el más usado y representativo SGP4.

Estos modelos tienen en cuenta las perturbaciones que puede sufrir la órbita debido a diferentes factores, entre los que destacan: efectos gravitatorios de terceros, resistencia atmosférica, fuerza electromagnética, efectos relacionados con

la relatividad general, no esfericidad del astro sobre el que se orbita, etcétera. Tienen un error de un kilómetro, que aumenta de uno a tres kilómetros por día.

d. Aplicación del modelo de Kepler

i. Introducción

En el siguiente apartado se mencionan los conceptos de anomalía media, excéntrica y verdadera. Estas son diferentes ángulos que sirven para representar la posición de un satélite en su órbita. A continuación se describen la anomalía media y la verdadera brevemente. Para ampliar información sobre este tema, se recomienda consultar la bilbiografía.

- La anomalía media representa la fracción de periodo que ha recorrido el satélite. Se consideran de 0º en el punto de periapsis.
- La anomalía verdadera representa el ángulo formado entre el semieje mayor de la elipse y el foco en el que está el cuerpo sobre el que se orbita.

ii. Cálculo de anomalía media actual

Para calcular la anomalía actual necesitamos el tiempo transcurrido desde el instante de tiempo del TLE hasta el instante actual:

$$\Delta t = t - t_0$$

Figura 18: Cálculo del tiempo actual

Calculamos ω, la velocidad angular en rad/s a partir de n, las revoluciones diarias obtenidas del TLE.

$$\omega = \frac{2\pi n}{24 \cdot 3600}$$

Figura 19: Cálculo de la velocidad angular en radianes por segundo

Calculamos la anomalía media actual:

$$m = m_o + w \cdot \Delta t$$

Figura 20: Cálculo de anomalía media actual

Siendo m_0 la anomalía del satélite en el instante del TLE en radianes.

iii. Cálculo de anomalía excéntrica

La ecuación de Kepler describe la relación entre la anomalía media y la excéntrica:

$$m = E - e \cdot senE$$

Figura 21: Cálculo de anomalía excéntrica

Dado que esta ecuación no se puede resolver de forma analítica es necesario aplicar métodos numéricos. En concreto se ha aplicado el método de aproximaciones sucesivas, en el que se aplica el siguiente algoritmo:

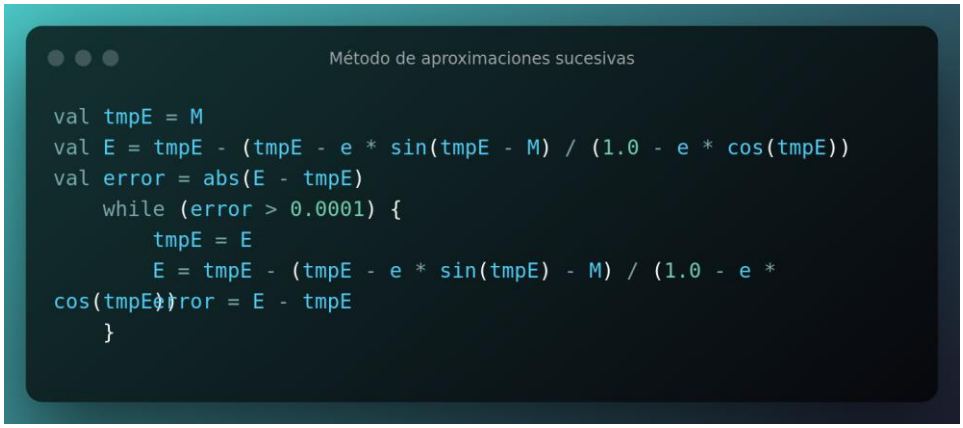


Figura 22: Modelo de aproximaciones sucesivas

Siendo M la anomalía media, E la excéntrica y e la excentricidad de la órbita.

iv. Cálculo de anomalía verdadera

A partir de la anomalía excéntrica podemos calcular la anomalía verdadera con la siguiente fórmula:

$$v = 2 \arctan(\sqrt{\frac{1+e}{1-e}} \tan\frac{E}{2})$$

Figura 23: Cálculo de anomalía verdadera

v. Cálculo de coordenadas cartesianas del satélite

Una vez conocida la anomalía verdadera, debemos calcular las coordenadas cartesianas en el plano orbital como se indica a continuación:

$$a = \sqrt[3]{\frac{\mu}{w^2}}$$

$$r = a \cdot \frac{(1 - e^2)}{1 + (e \cdot \cos(v))}$$

Figura 24: Cálculo del radio

Una vez obtenidos el semieje mayor (a) y el radio (r), se calculan las coordenadas cartesianas de la siguiente manera:

$$\mathrm{x} = \mathrm{r} \cdot cos(v)$$

$$\mathrm{y} = \mathrm{r} \cdot \sin(v)$$

$$\mathrm{z} = 0$$

Figura 25: Coordenadas cartesianas en el plano orbital

Una vez obtenidas, se aplican las matrices de transformación del argumento del pericentro, inclinación y ascensión recta del nodo ascendente, en ese orden:

$$\begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \end{pmatrix} = \begin{pmatrix} \cos\Omega & -\sin\Omega & 0 \\ \sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathrm{cosI} & -\mathrm{sinI} \\ 0 & \mathrm{sinI} & \mathrm{cosI} \end{pmatrix} \begin{pmatrix} \mathrm{cosP} & -\mathrm{sinP} & 0 \\ \mathrm{sinP} & \mathrm{cosP} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{pmatrix}$$

Figura 26: Matrices de transformación

Con esto, obtenemos las coordenadas cartesianas del satélite con origen en el centro de la Tierra. Para obtener las coordenadas incerciales centradas en el observador realizamos la siguiente operación.

$$x' = X - x_u$$

$$y' = Y - y_u$$

$$z' = Z - z_u$$

Figura 27: Coordenadas ecuatoriales inerciales centradas en el observador

Siendo (x_u, y_u, z_u) las coordenadas cartesianas del observador en un sistema inercial con origen en el centro de la tierra.

vi. Cálculo de alt-azimut del satélite respecto del observador

Lo siguiente será aplicar la matriz de conversión de sistema inercial de coordenadas al sistema local. Para ello multiplicamos las coordenadas obtenidas en el apartado anterior por la matriz M de conversión obtenida para tal efecto. Para más información sobre esta matriz, se recomienda consultar el código fuente.

$$(x'',\ y'',\ z'') = M_{3\times 3}\cdot\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

Figura 28: Coordenadas cartesianas en sistema local

El siguiente paso es convertir las coordenadas obtenidas a esféricas. Finalmente, para obtener el alt-azimut realizamos las siguientes operaciones.

$$\mathrm{az} = 180 - \varphi$$
$$\mathrm{alt} = -(\theta - 90)$$

Figura 29: Alt-azimut

Con esto, ya tenemos el alt-azimut del satélite según el modelo que hemos utilizado.

5. Herramientas y metodologías

A lo largo de este apartado se describen las herramientas utilizadas en el proyecto, tanto en el apartado de implementación y pruebas como en el apartado de la planificación.

a. Android Studio

Android Studio es el IDE (Integrated Development Environment) oficial para el desarrollo de aplicaciones nativas en dispositivos Android desarrollado por Google. Fue publicado en 2014, sustituyendo a Eclipse como IDE oficial para Android. Los lenguajes utilizables en el entorno son Java, C++ y Kotlin.

El diseño de la interfaz es realizado con ficheros XML, sin embargo estos se pueden generar con un editor de tipo *drag-and-drop.* Aparte, se pueden instanciar elementos de diseño en tiempo de ejecución.

i. Kotlin

Se trata de un lenguaje de programación de tipado estático basado en Java, ya que es dependiente de su biblioteca de clases y se ejecuta sobre su máquina virtual. Se ha optado por usar Kotlin debido a que desde el año 2019 es el lenguaje que recomienda Google para Android Studio.

ii. XML

XML es un metalenguaje que puede definir lenguajes de marcas. Fue desarrollado por el W3C y se utiliza para el almacenamiento legible de datos. Es el método que ofrece Android Studio para el diseño de la interfaz.

iii. Bibliotecas de Android utilizadas

A continuación se describen algunas bibliotecas que se han utilizado para facilitar el desarrollo de la aplicación

- Volley: Utilizada para realizar las peticiones HTTP de forma sencilla y ágil.

- Gson: Biblioteca para Java cuya utilidad es la de convertir objetos Java en formato JSON, y viceversa. En este caso se ha utilizado para convertir la respuesta del sistema externo, en formato JSON, a una clase de Kotlin.
- AppIntro: Utilizada para crear la introducción a la aplicación de forma sencilla. También se ha utilizado para las instrucciones de la simulación.

b. IntelliJ

IDE para la máquina virtual de Java desarrollado por JetBrains. Se ha utilizado para implementar los primeros prototipos en Kotlin.

c. Postman

Cliente HTTP utilizado para realizar pruebas con APIs mediante peticiones HTTP de forma sencilla gracias a su interfaz de usuario. Se ha utilizado para probar diferentes APIs para la obtención de los TLEs.

d. Git y Github

Se ha utilizado Git, un software de control de versiones para tratar con proyectos software de forma fácil y eficiente. Se ha alojado el repositorio en Github.

e. app.diagrams.net

Para realizar los diagramas se ha utilizado la herramienta alojada en app.diagrams.net, que sigue los estándares del proceso unificado.

f. Celestrak

Es una plataforma centrada en satélites y astronomía en general, creada por T. S. Kelso en el año 1985. Se utilizará su API para la obtención de los TLEs en formato JSON, así como para validar el algoritmo de propagación desarrollado.

g. n2yo - Real Time Satellite Tracking

Plataforma que localiza satélites en tiempo real y proporciona sus coordenadas en función a la posición de un observador. Se ha utilizado para validar los datos obtenidos con el algoritmo de propagación desarrollado.

h. Procreate

Aplicación de ilustración digital desarrollada por Savage Interactive para iPad OS. Se ha utilizado para diseñar el logo de la aplicación.

Figura 30: Logo de la aplicación

i. Scrum

El marco de trabajo utilizado ha sido Scrum. Se trata de un marco de trabajo incluido dentro de la categoría de metodologías ágiles que facilita la adaptación del producto a los requisitos, ya que estos pueden ser cambiantes (Schwaber , K; Sutherland, J, 2020). A lo largo del Anexo I, Planificación del proyecto, se explica en detalle el proceso seguido.

j. ray.so

A lo largo de esta memoria se adjuntan capturas de ciertos bloques de código. Para mejorar la presentación, se ha utilizado la herramienta de capturas de código ray.so.

k. Método de Durán y Bernárdez

En el proceso de especificación de requisitos se han utilizado las tablas de Durán y Bernárdez para describir los casos de uso. (Durán, A.; Bernárdez, B., 2002).

6. Aspectos relevantes del desarrollo

En esta sección se describen los aspectos más relevantes que hacen referencia al desarrollo del sistema así como la planificación, las iteraciones, los requisitos, etcétera.

a. Planificación

El marco de trabajo seleccionado, tal y como se ha dicho en apartados anteriores, se trata de Scrum. Este marco de trabajo nos permite desarrollar el sistema de forma iterativa e incremental, dividiendo el tiempo de trabajo en iteraciones denominadas Sprints. Al final de cada uno de estos Sprints se debe producir un producto entregable denominado incremento, que irá creciendo en cada Sprint recibiendo en cada iteración más funcionalidades.

El conjunto de requisitos del sistema recibe el nombre de pila de producto, y cada uno de los requisitos de esta pila es llamado historia de usuario. El subconjunto de historias de usuario que se selecciona para completar durante un Sprint se llama pila del sprint. A cada historia de usuario se le asignan unos puntos de esfuerzo denominados puntos de historia.

Todos los Sprints durante el desarrollo deben tener la misma duración. Si se observa que los puntos de historia que se realizan durante un Sprint no coinciden con los que se esperaban, para los siguientes Sprints se debe variar la cantidad de puntos de historia por Sprint pero nunca la duración de estos.

Al comenzar el proyecto se decidió que la duración de cada Sprint sería el estándar, es decir, 14 días.

Para ampliar la información sobre el marco de trabajo utilizado, las historias de usuario, el desarrollo de los Sprints, y más datos de interés, se puede consultar el Anexo I, Planificación del Proyecto.

b. Requisitos

En este apartado se describe de forma breve la metodología utilizada para la especificación de requisitos iniciales del sistema. Se han utilizado las tablas del método de Durán y Bernárdez, al igual que en el apartado de objetivos vista anteriormente.

A continuación se muestran algunos ejemplos de las tablas generadas. Para ampliar, se recomienda consultar el Anexo II.

i. Participantes

Participante	Facundo A. Sierra Rodríguez
Organización	Universidad de Salamanca
Rol	Alumno, desarrollador, Scrum Master, Product Owner
Es desarrollador	Sí
Es cliente	No
Es usuario	Sí
Comentarios	Ninguno

Figura 31: Tabla de participante

ii. Requisitos de información del sistema

IRQ-01	Satélite
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Dependencias	• TLE • Coordenadas del observador
Descripción	*El sistema debe poder almacenar información relevante sobre el estado de un satélite en un instante de tiempo.*

Datos	• Nombre • Identificador Norad • Elevación • Azimut • Anomalía verdadera • Vueltas totales • Ascensión Recta • Declinación • Tiempo desde la última actualización del TLE • Periodo • Velocidad Angular • Velocidad Lineal • Altura • TLE	
Tiempo de vida	Medio	Máximo
	-	-
Ocurrencias simultáneas	Medio	Máximo
	-	-
Importancia	Alta	
Urgencia	Media	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Figura 32: Tabla de requisito de información

iii. Actores del sistema

ACT-01	Usuario
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro

Descripción	*Actor que representa al usuario de la aplicación.*
Comentarios	Ninguno

Figura 33: Tabla de actor

iv. Casos de uso del sistema

UC-16	Añadir a favoritos	
Versión	1.0	
Autores	Facundo Agustín Sierra Rodríguez	
Fuentes	Angel Luis Sánchez Lázaro	
Dependencias	-	
Descripción	*El sistema añadirá el satélite seleccionado a la lista de satélites favoritos*	
Precondición	Se debe haber iniciado el caso de uso UC-13 Mostrar datos satélite para un satélite que no esté en la lista de favoritos	
Secuencia normal	Paso	Acción
	1	El usuario pulsa el botón de añadir a favoritos
	2	El sistema añade el satélite seleccionado a la lista de favoritos
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Rendimiento	Paso	Tiempo máximo
	-	-
Frecuencia	Media	
Importancia	Media	

Urgencia	Baja
Estado	Validado
Estabilidad	Media
Comentarios	Ninguno

Figura 34: Tabla de caso de uso

v. Requisitos no funcionales

NFR-01	Estabilidad
Versión	1.0
Autores	Facundo Agustín Sierra Rodríguez
Fuentes	Angel Luis Sánchez Lázaro
Dependencias	-
Descripción	*El sistema debe ofrecer un funcionamiento estable*
Importancia	Media
Urgencia	Baja
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Figura 35: Tabla de requisito no funcional

vi. Diagramas de casos de uso

A continuación se muestran unos ejemplos de los diagramas de casos de uso realizados para la especificación inicial de requisitos.

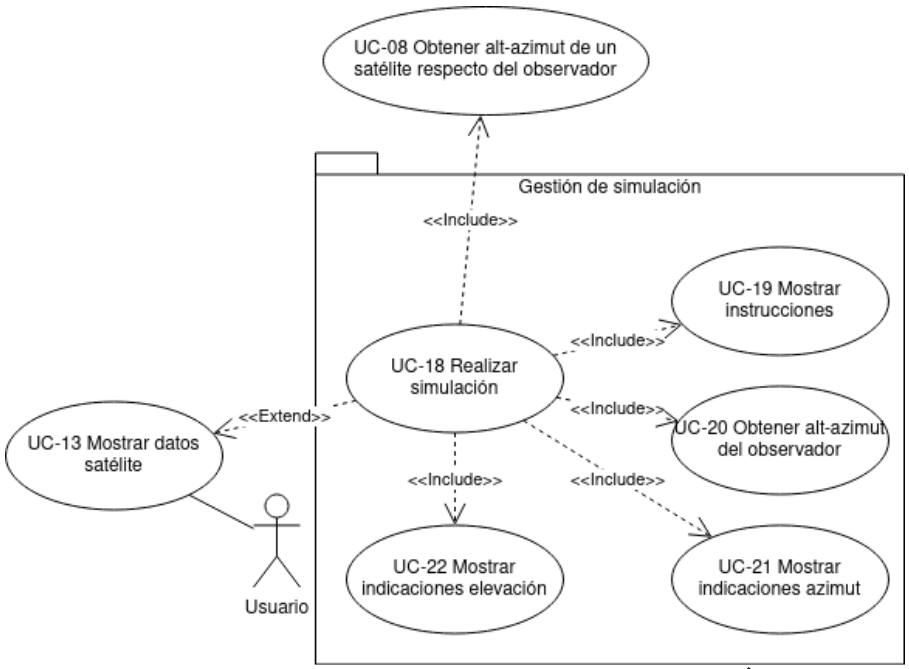


Figura 36: Diagrama de caso de uso del paquete Gestión de simulación

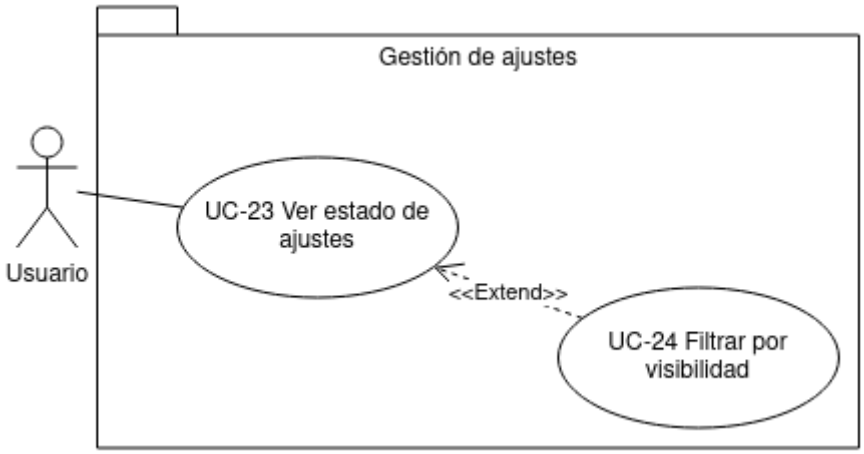


Figura 37: Diagrama del caso de uso del paquete Gestión de ajustes

c. Análisis

A continuación se indica el diagrama de clases realizado al comienzo del desarrollo durante el Sprint 2, en concreto la historia de usuario HU-07.

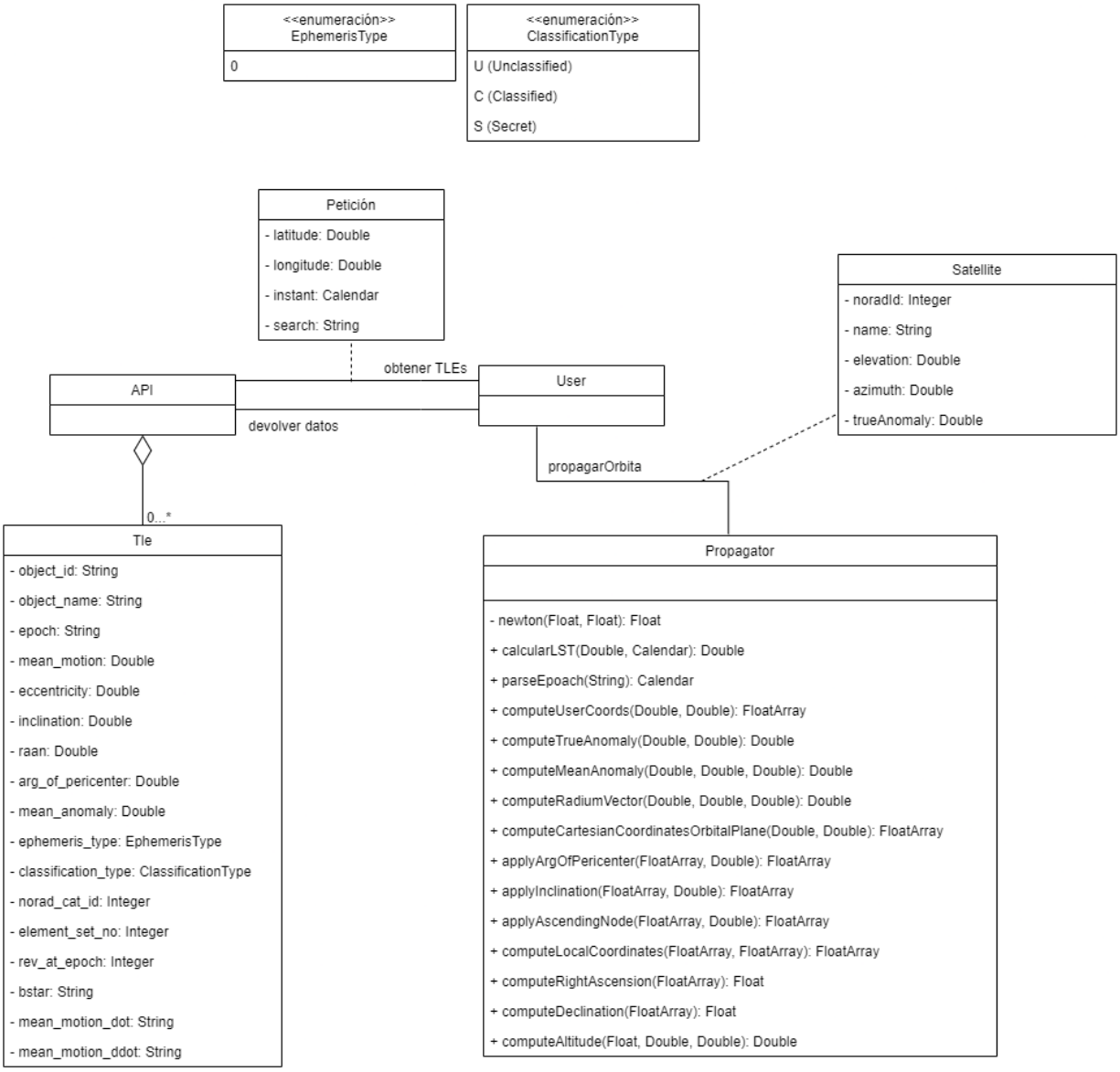


Figura 38: Diagrama de clases

d. Implementación

En esta sección se muestran algunos detalles relevantes en relación a la implementación del sistema.

i. Implementación de la funcionalidad

En esta sección se describirán algunos aspectos de la implementación de la funcionalidad de la aplicación.

1. Documentación técnica

Durante la implementación se ha realizado una documentación técnica que, con la herramienta dokka, genera una documentación visible desde un navegador web. Se adjuntan a la memoria los ficheros generados por la herramienta, y a continuación se muestran algunas capturas de ejemplo.

Figura 39: Fragmento de la documentación generada para la clase Propagator

app / com.example.satellitetracker / Propagator / applyArgOfPericenter

applyArgOfPericenter

```
fun applyArgOfPericenter(coordsOrbitalPlane: FloatArray, argumentOfPericenter: Double): FloatArray
```

Aplica argumento del pericentro/periapsis a las coordenadas cartesianas de un satélite

Return

Coordenadas cartesianas del satélite tras aplicar la matriz

Parameters

coordsArgumentOfPericenter	Array de tamaño 3 que contiene las coordenadas cartesianas de un satélite
argumentOfPericenter	Argumento del pericentro, obtenido de un TLE

Figura 40: Documentación generada para la función applyArgOfPericenter

2. Gestión de propagación de órbita y gestión de coordenadas

Para la propagación de órbita y la gestión de coordenadas se han utilizado principalmente las clases Propagator y Satellite. La clase Propagator contiene todos los métodos necesarios para propagar una órbita a partir de un TLE. La clase Satellite obtiene todos los atributos relevantes resultantes de esa propagación.

```
class Satellite (
    val name: String,
    val noradId: Int,
    val altitude: Double,
    val azimuth: Double,
    val trueAnomaly: Double,
    val totalRevs: Int,
    val rightAscension: Double,
    val declination: Double,
    val timeFromLastTle: Double,
    val period: Double,
    val angularSpeed: Double,
    val linearSpeed: Double,
    val height: Double,
    val tle: Tle,
) : Serializable
```

Figura 41: Clase Satellite

La siguiente figura muestra la función computeMeanAnomaly de la clase Propagator, que se encarga de calcular la anomalía media actual de un satélite a partir de su velocidad angular, el tiempo transcurrido desde la medición de los datos y la anomalía en ese mismo instante.

```kotlin
/**
 * Compute mean anomaly
 * Calcula la anomalía media actual de un satélite
 * @param omega velocidad angular, obtenida de un TLE
 * @param m0 anomalía en t0
 * @param delta tiempo transcurrido desde t0 hasta t
 * @return anomalía media en instante t en radianes
 */
fun computeMeanAnomaly(
  omega: Double,
  m0: Double,
  delta: Double
): Double {
  val n = (omega * (2 * PI)) / (24.0 * 3600.0) // [rev/dia solar] a [rad/s]
  var mt = (m0 * AngleUtils().DEGtoRAD) + n * delta
  val revs = mt / (PI * 2)
  mt -= floor(revs) * PI * 2
  return mt
}
```

Figura 42: Función computeMeanAnomaly

La siguiente función, computeCartesianCoordinatesOrbitalPlane, pertenece a la clase Propagator y calcula las coordenadas cartesianas de un satélite en el plano orbital a partir del radio-vector y la anomalía verdadera.

```kotlin
/**
 * Calcula las coordenadas cartesianas de un satélite en el plano orbital.
 * @param r Radio-vector del satélite
 * @param a Anomalía verdadera del satélite, en radianes
 * @return Array que contiene las coordenadas cartesianas x, y, z del
 * satélite en el plano orbital
 */
fun computeCartesianCoordinatesOrbitalPlane(
  r: Double,
  a: Double
): FloatArray {
  val cartesianCoordsOrbitalPlane = FloatArray(3)
  cartesianCoordsOrbitalPlane[0] = (r * cos(a)).toFloat()
  cartesianCoordsOrbitalPlane[1] = (r * sin(a)).toFloat()
  cartesianCoordsOrbitalPlane[2] = 0.0f
  return cartesianCoordsOrbitalPlane
}
```

Figura 43: Función computeCartesianCoordinatesOrbitalPlane

Para realizar otras operaciones relevantes en el proceso de propagación, se han implementado las clases AngleUtils y Coordinates. A continuación se muestra un par de ejemplos.

```kotlin
/**
 * Contiene recursos para tratar con ángulos y tratar con ellos en diferentes unidades.
 */
class AngleUtils {
    val DEGtoRAD : Double = PI / 180
    val RADtoDEG: Double = 180 / PI
    /**
     * Convierte un ángulo en grados a horas cadenas y segundos en formato String
     */
    fun decimalGradesToHours(angle: Float): String {
        var hour = (angle / 360.0 * 24.0)
        var min = ((hour - hour.toInt()) * 60.0)
        var sec = ((min - min.toInt()) * 60.0)
        return String.format("%.2f%.2f%.2f", hour, min, sec)
    }
}
```

Figura 44: Clase AngleUtils

La siguiente figura muestra la función sphericalToCartesian de la clase Coordinates, que toma unas coordenadas esféricas en grados y las convierte en coordenadas cartesianas.

```
/**
* Convierte las coordenadas esféricas a coordenadas cartesianas y las devuelve en un
* FloatArray de dimension 3.
* @return Coordenadas cartesianas
*/
private fun sphericalToCartesian(): FloatArray {
  val cartesian = FloatArray(3)
  cartesian[0] = (r*sin(alt*AngleUtils().DEGtoRAD) * cos(az*AngleUtils().DEGtoRAD)).toFloat()
  cartesian[1] = (r*sin(alt*AngleUtils().DEGtoRAD) * sin(az*AngleUtils().DEGtoRAD)).toFloat()
  cartesian[2] = (r*cos(alt*AngleUtils().DEGtoRAD)).toFloat()
  return cartesian
}
```

Figura 45: Función sphericalToCartesian

3. Gestión de búsqueda y filtrado de satélites

Lo más destacable de esta sección en cuanto a funcionalidad es la RecyclerView utilizada para mostrar la lista de satélites, usada tanto en el fragmento de búsqueda como en el de favoritos. El código de su adaptador se puede observar en la siguiente figura.

```kotlin
package com.example.satellitetracker
import [...]

class RvSatelliteListAdapter (private val parent: RvSatelliteListAdapterCallback) :
RecyclerView.Adapter<RvSatelliteListAdapter.SatelliteCard> () {

    lateinit var context: Context

    interface RvSatelliteListAdapterCallback{
        fun onClicked(satellite: Satellite)
    }

    var satelliteList : MutableList<Satellite> = ArrayList()
    set(satelliteList){
        field = satelliteList
        notifyDataSetChanged()
    }

    class SatelliteCard(view: View): RecyclerView.ViewHolder(view){
        val tvNombre: TextView
        val tvNoradId: TextView
        val tvVisible: TextView
        val container: ConstraintLayout
        init{
            tvNombre = view.findViewById(R.id.tv_satellite_name)
            tvNoradId = view.findViewById(R.id.tv_satellite_norad_id)
            tvVisible = view.findViewById(R.id.tv_visible)
            container = view.findViewById(R.id.satellite_card_container)
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): SatelliteCard {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.satellite_card, parent,
false)
        context = parent.context
        return SatelliteCard(view)
    }

    override fun onBindViewHolder(holder: SatelliteCard, position: Int) {
        holder.container.animation = AnimationUtils.loadAnimation(context,
R.anim.fade_transition_animation)
        holder.tvNombre.text = satelliteList[position].name
        holder.tvNoradId.text = String.format("NORAD ID: %d", satelliteList[position].noradId)

        if (satelliteList[position].altitude > 20.0){
            holder.tvVisible.text = "Visible: Sí"
        }
        else {
            holder.tvVisible.text = "Visible: No"
        }
        holder.itemView.setOnClickListener{
            parent.onClicked(satelliteList[position])
        }
    }

    override fun getItemCount(): Int {
        return satelliteList.size
    }
}
```

Figura 46: Fichero RvSatelliteListAdapter.kt

4. Gestión de ajustes y satélites favoritos

Las sharedPreferences son a efectos prácticos un fichero xml que se guarda en el dispositivo y permite almacenar diferentes variables con valores que persisten entre diferentes instancias de ejecución. Este método también ha sido utilizado para almacenar la lista de IDs de los satélites favoritos del usuario. A continuación se muestra un ejemplo de acceso a estas variables.

```
private fun loadSettings(){
  val sp = PreferenceManager.getDefaultSharedPreferences(context)
  showOnlyVisibles = sp.getBoolean("show_only_visible_satellites", false)
  manualCoordinatesSelected = sp.getBoolean("manual_coordinates", false)
  favSatellitesList = sp.getStringSet("favSatellites", null)
}
```

Figura 47: Acceso a variables de sharedPreferences

5. Gestión de simulación

La simulación consiste en variar el layout en pantalla dependiendo de la situación en la que se encuentre el positivo respecto de las coordenadas del satélite en función al observador.

Por ejemplo, a continuación tenemos las funciones turnClockWise y raiseDevice, a las que llamaremos cuando sea necesario para dar indicaciones al usuario.

```
/**
 * Indica al usuario que debe girar el dispositivo en sentido horario
 */
private fun turnClockwise() {
  ivClockwise.visibility = View.VISIBLE
  ivClockwise.rotation += 10.0F
  if (ivClockwise.scaleX != -1.0F) {
    ivClockwise.scaleX = -1.0F
  }
}
```

Figura 48: Función turnClockWise

```
/**
 * Indica al usuario que debe alzar el dispositivo
 */
private fun raiseDevice() {
  ivArrowDown.visibility = View.GONE
  ivArrowUp.visibility = View.VISIBLE
}
```

Figura 49: Función raiseDevice

A continuación se muestra el código de la primera fase de la simulación, en la que el usuario debe ajustar el azimut del dispositivo al del satélite. Vemos que se llama a la función turnClockwise o turnCounterClockwise según sea necesario. Cuando el azimut del usuario está dentro de un rango deseado, se espera durante un tiempo para confirmar que está en la posición correcta de forma estable. Si no, se vuelve a empezar con las indicaciones y el contador se pone a cero de nuevo.

```
if (!firstPhaseDone) {
    tvInstructions.visibility = View.VISIBLE
    tvInstructions.text = "Gira tu dispositivo como se indica en pantalla"
    tvSatElevation.visibility = View.GONE
    tvTitle.text = "Ajustar azimut"
    when {
        azimuth < satAzimuth-7 -> {
            val dif  = satAzimuth - azimuth
            val dif2 = azimuth + 360 - satAzimuth
            counterAz = 0
           //Hayamos el camino más corto para ajustar el azimut
            if (dif < dif2) turnClockwise()
            else turnCounterClockwise()
        }
        azimuth > satAzimuth+7 -> {
            val dif  = azimuth - satAzimuth
            val dif2 = satAzimuth + 360 - azimuth
            counterAz = 0
            if (dif < dif2) turnCounterClockwise()
            else turnClockwise()
        }
        else -> {
            counterAz++
            tvInstructions.text = "¡No te muevas!"
            ivClockwise.visibility = View.GONE
            if (counterAz >= counterlimit) {
                firstPhaseDone = true
                tvTitle.text = "Ajustar elevación"
                tvUserAzimuth.visibility = View.GONE
                tvSatAzimuth.visibility = View.GONE
                tvUserElevation.visibility = View.VISIBLE
                tvSatElevation.visibility = View.VISIBLE
            }
        }
    }
}
```

Figura 50: Primera fase de la simulación

6. Gestión de vista de detalle

Con respecto a la vista de detalle, lo más destacable es la implementación de la actualización a tiempo real de los datos del satélite. Para esto se crea un servicio dentro del hilo de la interfaz de usuario que cada 100 milisegundos actualiza las etiquetas correspondientes con los nuevos datos del satélite.

```
val timer = Timer()
timer.scheduleAtFixedRate(object : TimerTask() {
    override fun run() {
        runOnUiThread {
            satellite = updateData(satellite.tle, lat, localSiderealTime)
           observerTime = LocalDateTime.now()
           updateTextViews()
        }
    }
}, 0, 100)
```

Figura 51: Actualización a tiempo real de datos del satélite

ii. Creación de vistas

A lo largo del proceso se han ido desarrollando una serie de vistas para acceder a las funcionalidades de la aplicación de forma lo más cómoda e intuitiva posible.

En primer lugar, se creó una vista para la búsqueda de satélites, que consiste en una barra de búsqueda y una sección de pantalla en la que se muestra la lista de satélites encontrados según la búsqueda introducida.

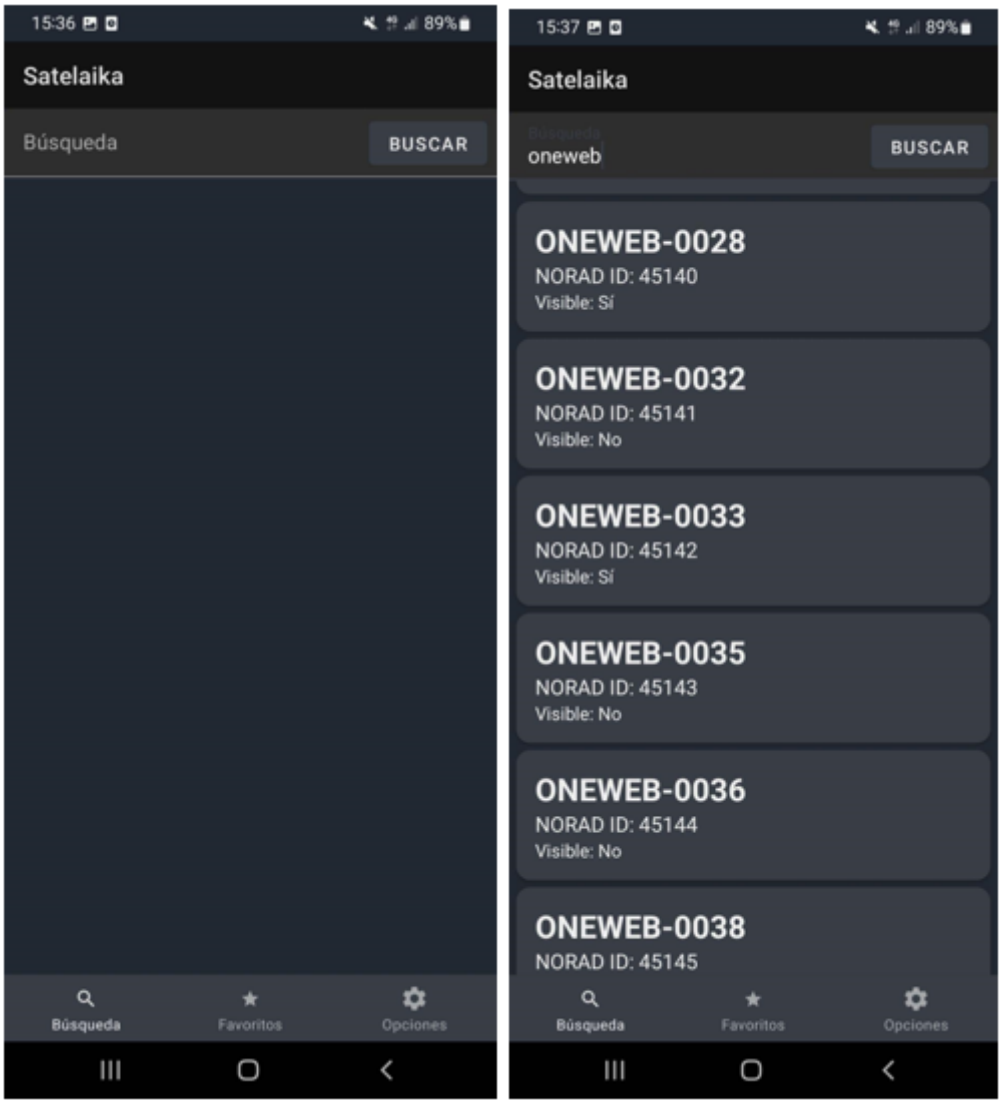


Figura 52: Búsqueda de satélites

Además, se implementó una barra de navegación en la zona inferior para acceder a los diferentes fragmentos de la actividad principal. A continuación se muestra junto al código que la genera. En él describimos cada ítem, que nos redirigirá a un diferente fragmento, y le asignamos un icono que guardamos en la carpeta @drawable, junto al resto de recursos gráficos.

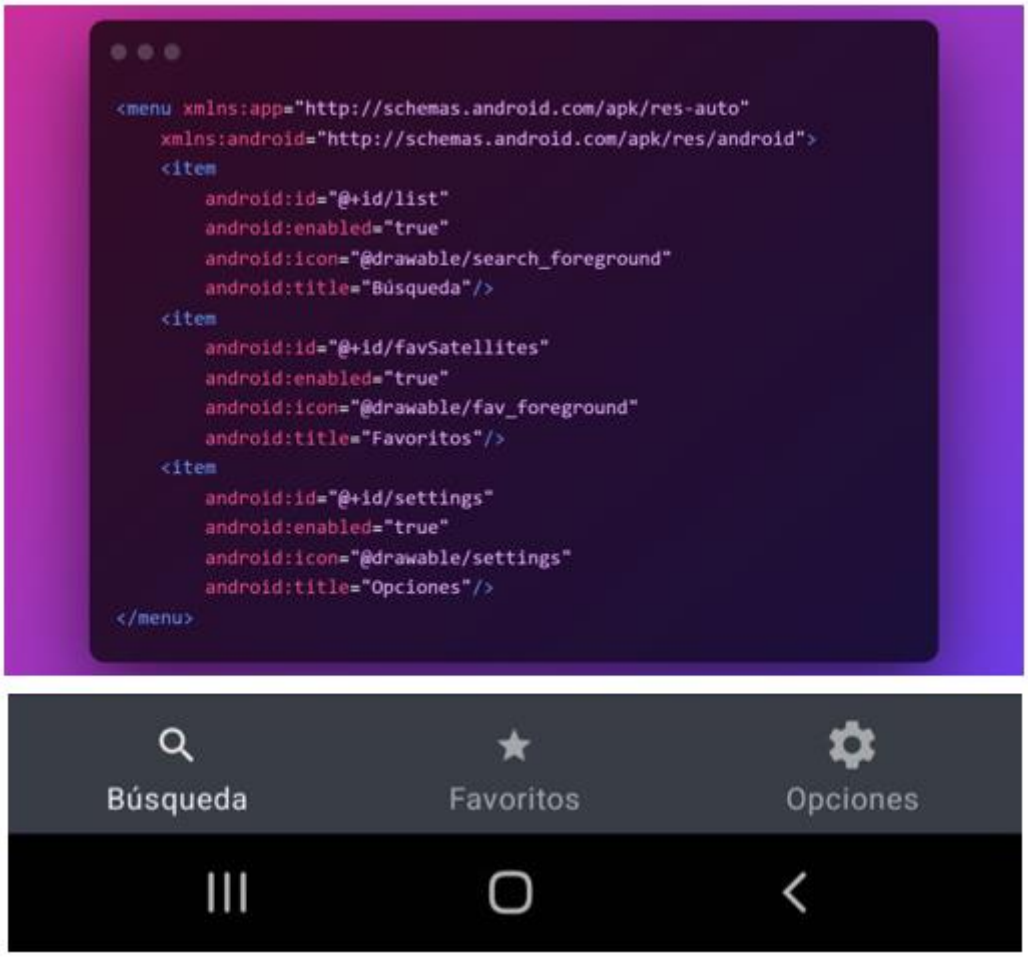


Figura 53: Barra de navegación inferior

Lo siguiente fue crear una vista de detalle para acceder a información sobre un satélite que seleccione el usuario.

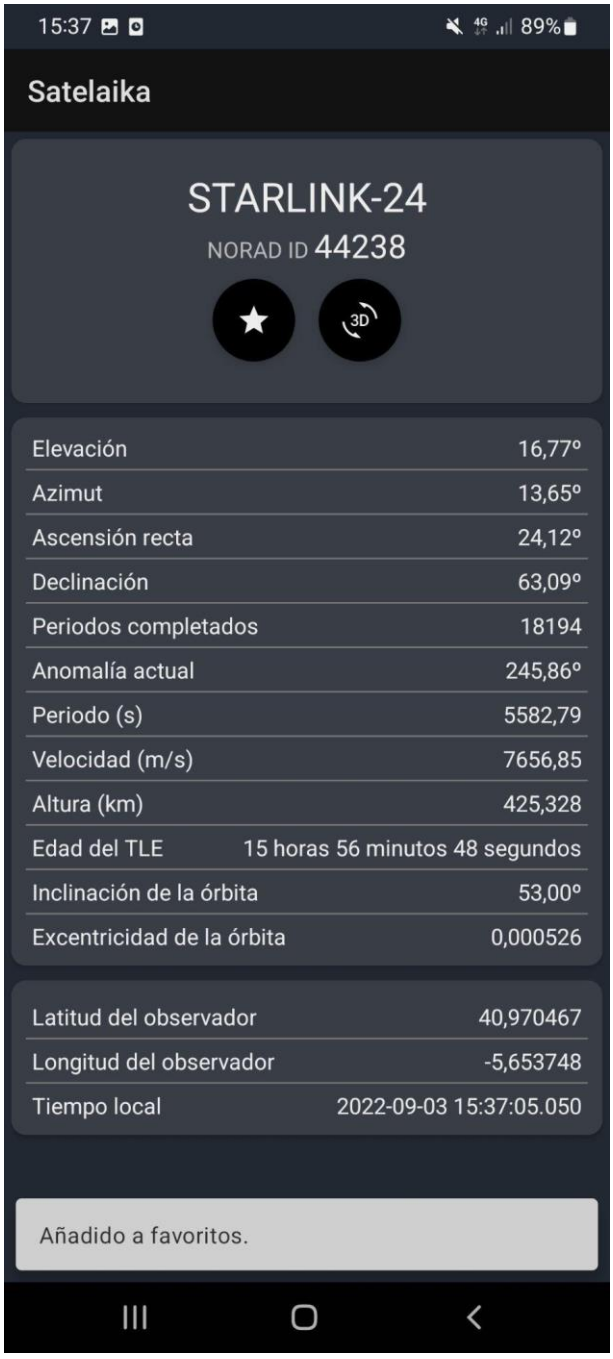


Figura 54: Vista de detalle

Para parte del diseño de esta vista se ha tomado como inspiración la vista de detalles de contacto en teléfonos Samsung.

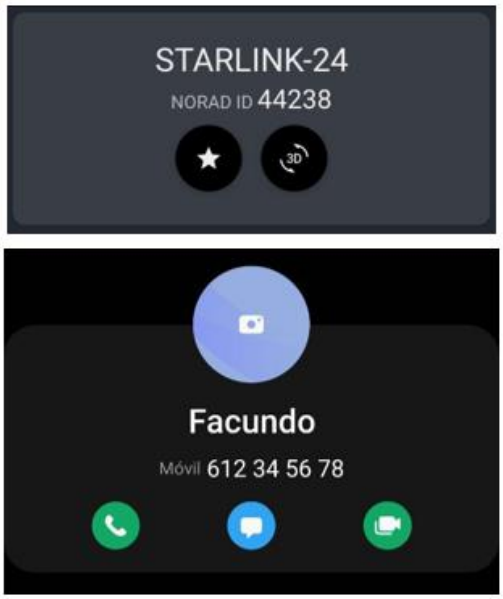


Figura 55: Vista de contactos en teléfonos Samsung

Lo siguiente fue implementar la vista de satélites favoritos. En este caso será igual que la vista de búsqueda pero sin la barra de búsqueda.

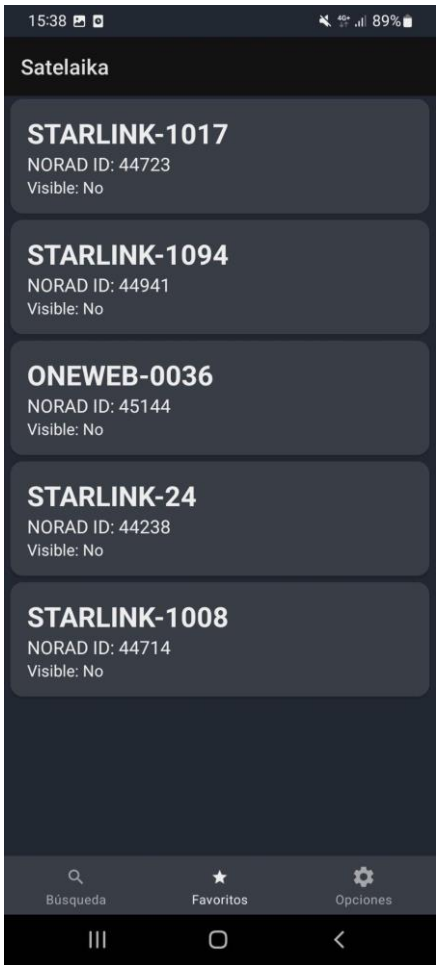


Figura 56: Satélites favoritos

El tercer fragmento de la actividad principal es el de opciones. La vista de opciones es generada con el fichero root_preferences.xml en el que se especifican categorías de opciones y las variables en las que guardar el estado de las opciones.

```xml
<PreferenceScreen xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="Visibilidad"
        android:icon="@drawable/ic_baseline_visibility_24">
        <SwitchPreferenceCompat
            app:key="show_only_visible_satellites"
            app:title="@string/show_fav_satellites"
            app:singleLineTitle="false"
            app:summaryOff="@string/show_fav_satellites_summary_off"
            app:summaryOn="@string/show_fav_satellites_summary_on">
        </SwitchPreferenceCompat>
    </PreferenceCategory>
</PreferenceScreen>
```

Figura 57: Fichero root_preferences.xml

Figura 58: Vista de ajustes

Lo siguiente fue crear unas vistas para la simulación. Se actualizarán en función al estado de la simulación.

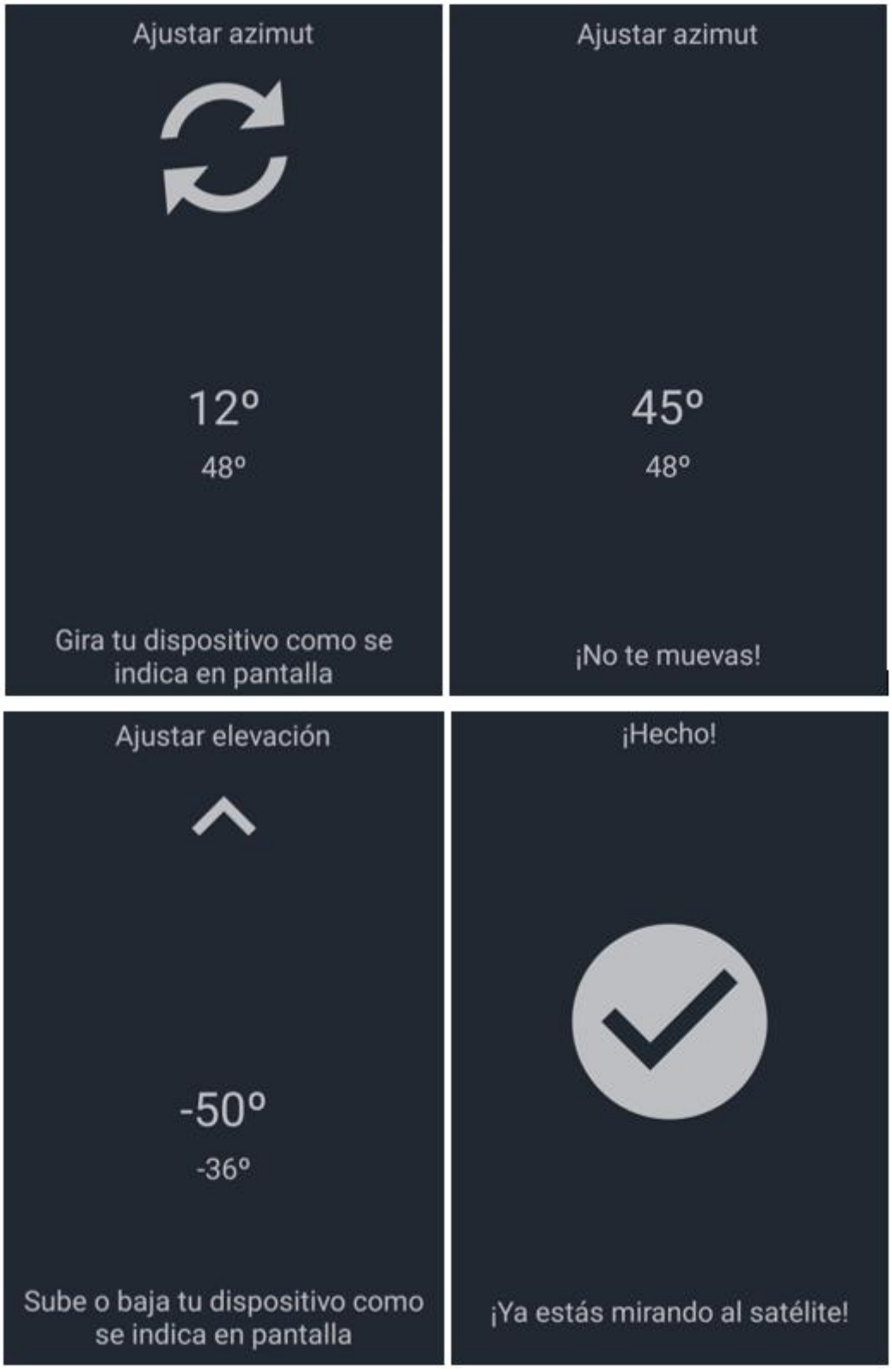


Figura 59: Vistas simulación

Una vez hecho todo esto, con la funcionalidad acabada, se creó el logo de la aplicación, una introducción e instrucciones para la simulación. Se han realizado con una biblioteca open source llamada AppIntro.

Figura 60: Introducción e instrucciones

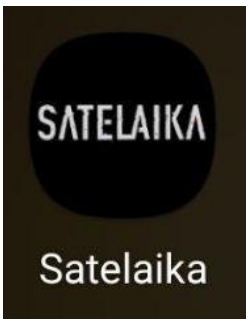


Figura 61: Aplicación desde el menú de aplicaciones

e. Pruebas

Como se ha indicado en el apartado de herramientas, se han realizado pruebas en n2yo para comprobar la validez del modelo. A continuación se muestran los resultados.

Como podemos observar en este primer ejemplo, la elevación y azimut obtenidos muestran valores cercanos a los de n2yo, pero con cierto error. En concreto la elevación muestra un error de 7º, pero el error del azimut aumenta hasta 20º.

Figura 62: Prueba 1

En el segundo ejemplo, observamos que el error en el azimut es de unos 12º mientras que el de la elevación es de unos 20º.

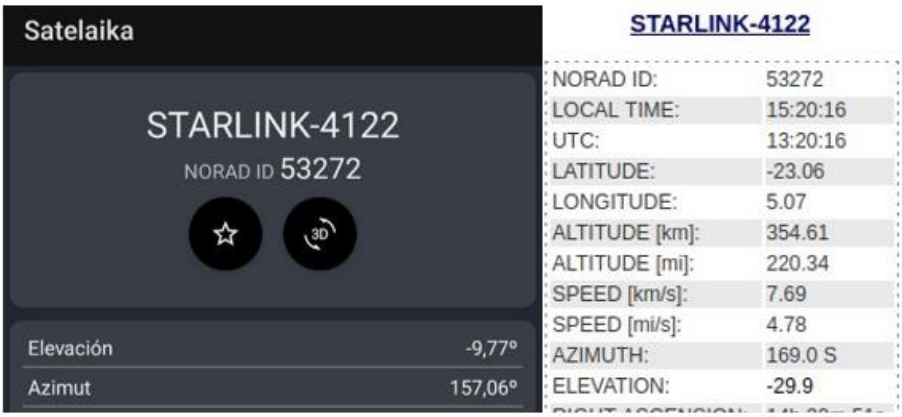


Figura 63: Prueba 2

Se proponen mejoras para un menor error en los apartados de conclusiones y líneas de trabajo futuras.

7.Conclusiones

Tras finalizar el desarrollo del sistema y analizar los objetivos cumplidos y los planteados en un principio, podemos llegar a varias conclusiones.

Durante el inicio del desarrollo planteamos por ejemplo la idea de utilizar alguna API externa para realizar las propagaciones de órbita, pero finalmente se realizó la idea inicial que era implementar la propagación desde cero. Esto acarreó problemas ya que se tuvo que dedicar más tiempo del esperado a la implementación del modelo físico y redujo el tiempo que se pudo dedicar a la construcción de funcionalidades propias de la aplicación.

En caso de llevar a cabo este proyecto fuera del ámbito académico, quizá se podría optar por formar un equipo de desarrollo multidisciplinar en el que algún miembro con conocimientos de mecánica orbital se encargaría de desarrollar todo el modelo físico, mientras que otros miembros tendrían la función de implementar ese modelo físico en lenguaje Kotlin e integrarlo en Android Studio.

Con respecto al tema del estudio de las redes satelitales de baja latencia, la aplicación puede resultar útil para un usuario interesado en estos temas ya que con ella podrá buscar información sobre los satélites que desee, ver qué rasgos tienen en común los satélites de diferentes proyectos de este campo y qué rasgos los diferencian de otros satélites como pueden ser los satélites de órbita terrestre media o alta.

El proyecto me ha servido como acercamiento a diferentes campos como al desarrollo de aplicaciones móviles nativas en Android, al diseño de interfaces de usuario en XML, y al lenguaje Kotlin, que parece ser el futuro de la programación en Android.

Por otro lado, he podido investigar sobre este nuevo tipo de redes universales que dan cobertura mundial y los distintos proyectos que se han planteado para llevarlo a cabo, además de tener una primera aproximación a la mecánica orbital y conocer los diferentes modelos de propagación de órbita y en concreto del modelo Kepleriano de dos cuerpos.

Como conclusión final, considero que el proyecto se ha cumplido de forma satisfactoria y me ha servido para poner en práctica los conocimientos adquiridos durante el grado, poniendo a prueba mis capacidades para aprender nuevas tecnologías que se usan en la industria y preparándome para el mundo laboral.

8. Líneas de trabajo futuras

Tras finalizar el proyecto, en esta sección se plantean posibles mejoras y líneas de trabajo futuras para el sistema desarrollado.

Teniendo en cuenta que el equipo de desarrollo ha sido formado por una persona, podemos observar que la aplicación posee ciertas características mejorables y carece de otras que se podrían añadir. A continuación se describen algunos ejemplos.

- Se podría mejorar la simulación añadiendo la opción de ejecutarla con un conjunto de satélites o con todos los satélites visibles desde la ubicación del usuario. Además, se podría mejorar gráficamente introduciendo modelos 3D para representar los satélites junto a animaciones para representar su posición relativa respecto a la orientación del dispositivo del usuario.
- A la hora de establecer la ubicación del usuario, esta se obtiene directamente de los servicios de ubicación del dispositivo. Se plantea añadir la opción de introducir coordenadas de forma manual.
- Para calcular la posición de los satélites, la aplicación utiliza el modelo Kepleriano de dos cuerpos. Al tratarse de una aproximación que no tiene muchos factores en cuenta, posee errores. Se plantea utilizar un modelo de perturbaciones simplificado como puede ser SGP4. Para esto se podría utilizar algún recurso o librería externo que facilitara el proceso.

9. Bibliografía

Danby, J.M.A. (1962). *Fundamentals of Celestial Mechanics*. Willmann-Bell.

Hoots, R.; Roehrich, R (1988). *Models for Propagation of NORAD Element Sets*

Colerus, E. (1972). *Breve historia de las matemáticas 1 y 2*. Doncel

Durán, A., Bernárdez, B. (2002). *Metodología para la Elicitación de Requisitos de Sistemas Software*

Meyer, B. *Construcción de Software Orientado a Objetos*. Prentice-Hall

Pressman, R.S. (2010) *Ingeniería del Software, un enfoque práctico*. Mc Graw Hill.

Wikipedia. *Órbita terrestre baja* wikipedia.org

Xataka. *La Nasa está preocupada por los 30.000 satélites [...] de Starlink.* xataka.com

Wikipedia. *Starlink.* wikipedia.org

Wikipedia. *Constelación de satélites.* wikipedia.org

El Español. *Qué es Starlink y cómo funciona [...]* elespanol.com

Android Developer. *Volley* developer.android.com

Github. *Gson* github.com

Github. *AppIntro* github.com

Business Insider, *SpaceX has darkened its Starlink internet satellites with visors to avoid disrupting the night sky*. businessinsider.com

NasaSpaceFlight. *Fourth shell of Starlink reaches 1000 satellites.* nasaspaceflight.com

ray.so https://ray.so/

Grush, L. The Verge(2022) *Amazon's Project Kuiper books up to 83 rockets to launch its internet-beaming satellites.* theverge.com

Wikipedia. *Telesat* wikipedia.org

Telesat. *Telesat to Become Public Company [...]* telesat.com

Sheetz, Michael. (2021) *OneWeb CEO: Here's why our product is different than Elon Musk's SpaceX Starlink*. cnbc.com

Castor2.ca. *Orbit anomalies.* castor2.ca

Rutgers.edu (2019) *Types of orbits.* rutgers.edu

Ken Schwaber & Jeff Sutherland (2020) *La Guía Scrum.* scrumguides.org

Wikipedia. *OneWeb*. wikipedia.org