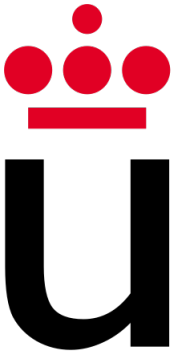


Universidad Rey Juan Carlos

Grado en Ingeniería en Tecnologías de la Telecomunicación

Universidad Rey Juan Carlos Fuenlabrada

Escuela de Ingeniería de Fuenlabrada

TRABAJO FIN DE GRADO

Herramienta de cálculo de capacidad para un sistema de transmisión de clave cuántica

Autor: Iván Fernández García

Tutor: Ángel Álvaro Sánchez

Fuenlabrada, FEBRERO | OCTUBRE de 2024

En primer lugar, quiero dar las gracias a Ángel y al equipo de Thales Alenia Space por darme la oportunidad de desarrollar mi TFG junto a ellos.

En segundo lugar, quiero agradecer a mis compañeros y amigos por su constante apoyo a lo largo de este tiempo, ayudándome a alcanzar la finalización de mis estudios de grado

Por último y más importante, quiero dar gracias infinitas a mis padres Beatriz García Fernández y Eusebio Fernández Rodríguez, así como a mi hermano, Sergio Fernández García, por todo su apoyo, cariño y comprensión hasta el día de hoy. También quiero acordarme de mis familiares más cercanos y de los que ya no están presentes físicamente, pero que nunca se irán, gracias.

Declaración de originalidad:

El autor de este trabajo, Iván Fernández García, con D.N.I. 49101262B, declara que el contenido de este trabajo de fin de grado es original, y en el caso de haber utilizado las ideas o contenidos de otros trabajos de otros autores están convenientemente citados, de acuerdo con las normas de citación establecidas.

El número de páginas de este trabajo es: 76

Índice

Índice de figuras

Resumen

La Distribución de Claves Cuánticas (Quantum Key Distribution, QKD) es un método de comunicación que permite enviar y recibir claves entre usuarios con mayor seguridad debido a la inviolabilidad del canal por su naturaleza. Se trata de la creación de una clave secreta, únicamente conocida por emisor y receptor, que sirve para cifrar y descifrar los mensajes posteriores. Esta comunicación cuántica se utiliza en el ámbito de las comunicaciones satelitales, debido a la gran importancia de la seguridad en la información compartida. Este TFG se define en un escenario donde se realizan comunicaciones entre estaciones terrenas a través de un satélite geoestacionario mediante este tipo de criptografía cuántica. Para realizar una comunicación óptima se destinan una serie de slots a cada comunicación en función de las necesidades de claves de cada estación. El objetivo de este trabajo es realizar una herramienta que optimice la distribución de claves para cada caso, consiguiendo satisfacer las necesidades de claves, pero sin desperdiciar una gran cantidad de estas. Para ello se ha utilizado la herramienta de Matlab, la cual nos permite calcular la mejor manera de distribuir el volumen total de clave y configurar diferentes parámetros en función de las necesidades de cada comunicación. Por último, tras simular varias posibles configuraciones, se presentan los resultados obtenidos para su posterior análisis. Esta herramienta está creada en colaboración con la empresa Thales Alenia Space SA dentro del proyecto GARBO.

Abstract

Quantum Key Distribution (QKD) is a communication method that allows sending and receiving keys between users with greater security due to the inviolability of the channel by its nature. It involves the creation of a secret key, only known to the sender and receiver, which is used to encrypt and decrypt subsequent messages. This quantum communication is used in the field of satellite communications, due to the great importance of security in shared information. This TFG is defined in a scenario where communications are carried out between earth stations through a geostationary satellite, using this type of quantum cryptography. To achieve optimal communication, a series of slots are allocated to each communication depending on the key needs of each station. The objective of this work is to create a tool that optimizes the distribution of keys for each case, managing to satisfy the needs of keys, but without wasting a large amount of them. For this, the Matlab tool has been used,

which allows us to calculate the best way to distribute the total key volume and configure different parameters depending on the needs of each communication. Finally, after simulating several possible configurations, the results obtained are presented for subsequent analysis. This tool is created in collaboration with the company Thales Alenia Space SA within the GARBO project.

Palabras clave

Palabras clave: Ordenador Cuántico, GARBO, Caramuel, QKD, Criptografía cuántica, SKR, fotón, protocolo BB84, optimización, Matlab, interfaz de usuario y slots horarios.

Keywords

Keywords: Quantum Computer, GARBO, Caramuel, QKD, Quantum Cryptography, SKR, photon, BB84 protocol, optimization, Matlab, user interface and time slots.

Capítulo 1: Introducción

En este apartado se muestra una introducción al trabajo realizado y las razones que motivaron su creación. En primer lugar, tenemos el apartado de motivación en el cual se explican los motivos que han llevado a la realización de este trabajo de fin de grado. Posteriormente, se explica la estructura de la memoria, los objetivos a cumplir con este proyecto, un apartado de diversidad y sostenibilidad y, por último, un cronograma donde se aprecia la organización temporal.

1.1 Motivación.

La seguridad en las comunicaciones es algo prioritario, más aún cuando se trata de información con un contenido sensible y que necesita ser tratada de una manera especialmente segura. **La Distribución de Claves Cuánticas (QKD)** es una tecnología que garantiza una transmisión y recepción segura de información a partir de la generación de claves mediante criptografía cuántica para evitar posibles hackeos o ataques cibernéticos. Los hackers en un futuro no muy lejano contarán con ordenadores cuánticos, cuya potencia será capaz de descifrar las contraseñas actuales más inexpugnables. Son capaces de ejecutar en segundos el trabajo que los ordenadores clásicos tardan años en realizar. Hoy en día, los espías son capaces de interceptar mensajes críticos y guardarlos para decodificarlos en un futuro cuando dispongan de la tecnología necesaria, con el objetivo de adueñarse de información crítica de gobiernos, empresas, etc.

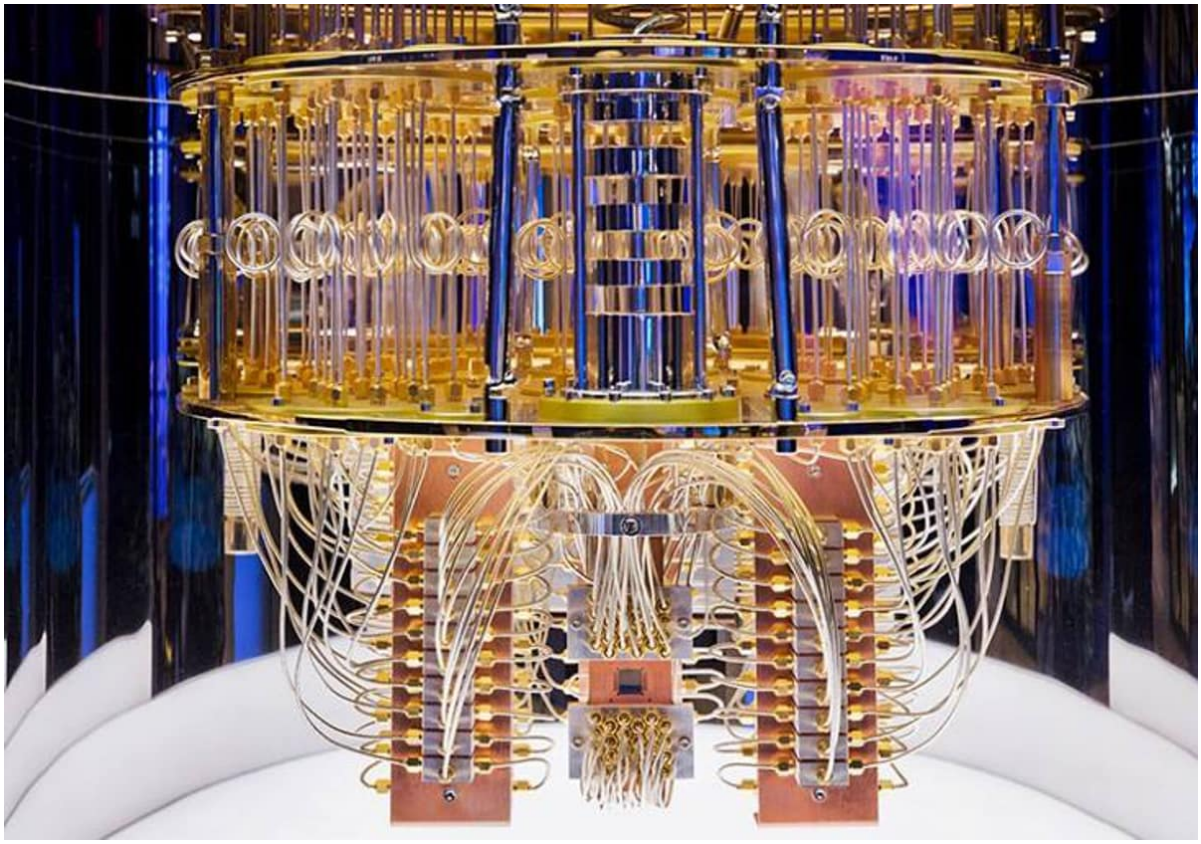


Figura 1: Ordenador Cuántico

Todo esto hace necesario el rápido despliegue de una tecnología robusta frente a los ataques que permitirán realizar los ordenadores cuánticos.

La **comunicación cuántica** se basa en la transmisión, detección y procesado de fotones individuales y por ello a día de hoy presenta dificultades en enlaces terrestres de distancia considerable, debido a las pérdidas que se generan en la fibra óptica a partir de centenares de kilómetros y a la inexistencia de equipos que actúen como repetidores cuánticos de la señal. Por este motivo, la opción de la **comunicación satelital** es la más adecuada para establecer comunicaciones cuánticas de gran distancia.

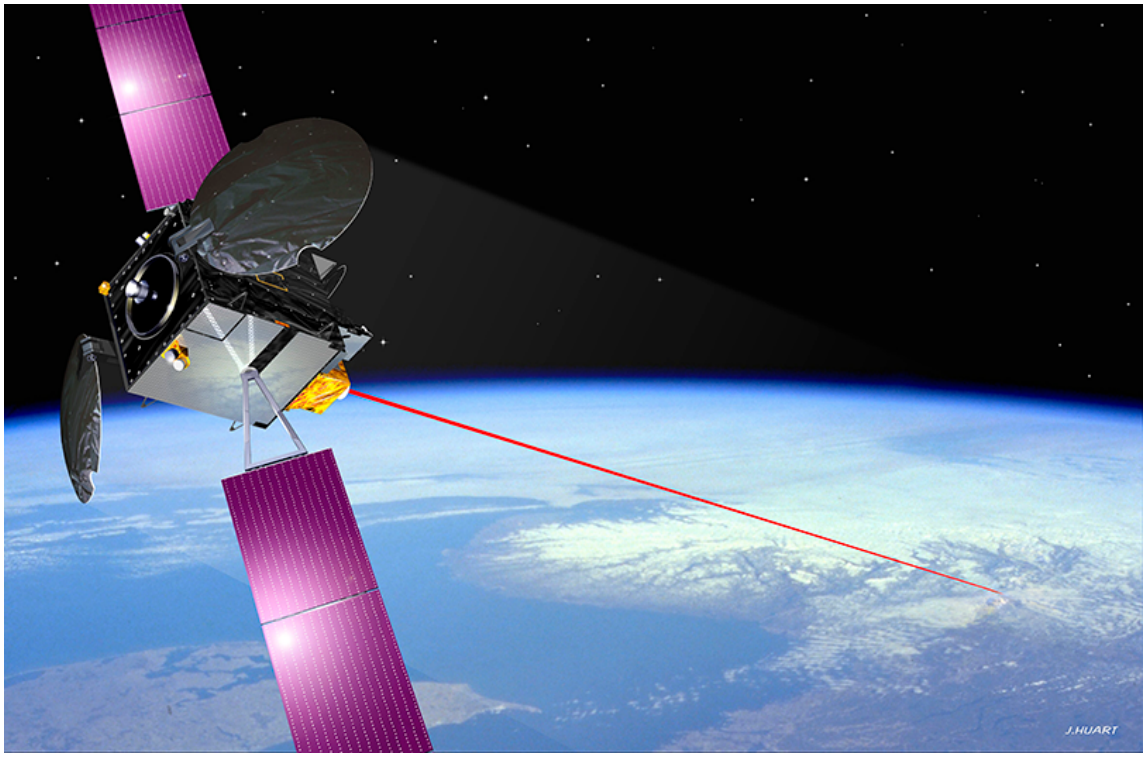


Figura 2: Satélite con tecnología QKD

El Ministerio de Ciencia, Innovación y Universidades, a través del Centro para el Desarrollo Tecnológico y la Innovación (CDTI), el PERTE (Proyectos estratégicos para la recuperación y transformación económica) y la Secretaría de Estado de Telecomunicaciones e Infraestructuras Digitales (SETELECO) del Ministerio de Asuntos Económicos y Transformación Digital colaboran a través de la compra pública precomercial del CDTI (Centro para el Desarrollo Tecnológico y la Innovación) de un sistema real de transmisión de clave cuántica desde órbita geoestacionaria completado y calificado para vuelo mediante prueba y demostración (en tierra o espacio).

Este proyecto ha sido asignado a **Thales Alenia Space** con el nombre **GARBO**, en honor al espía español Joan Pujol, el cual tuvo un papel clave en la operación Fortitude en la Segunda Guerra Mundial.

En este trabajo colaboraremos con Thales Alenia Space España creando una herramienta para obtener una distribución óptima de estos bits de claves cuánticas entre las distintas estaciones del proyecto GARBO.

Figura 3: Sede de Thales Alenia Space en Tres Cantos, Madrid

Por estos motivos, el objetivo del trabajo es el desarrollo de una aplicación que nos permita optimizar el volumen de clave generado por el sistema en función de las necesidades que tenga la comunicación y comprobar los resultados a través de tablas numéricas y gráficos para su posterior estudio y conclusión.

1.2 Estructura de la memoria.

La estructura principal se divide en una serie de índices de figuras, seguidas de un resumen y unas palabras claves, el índice principal del trabajo, los próximos pasos que se podrían dar, los conocimientos adquiridos, las conclusiones del proyecto y, para finalizar, una bibliografía donde se encuentran las citas utilizadas para analizar y describir los diferentes puntos del estudio.

En este trabajo comenzamos con una introducción en el capítulo 1 para poner en contexto al lector, donde se explica la motivación que ha llevado a la realización de este proyecto, este apartado de estructura, donde describimos las distintas partes que componen el trabajo, los objetivos a conseguir, un apartado de diversidad y sostenibilidad y un cronograma en el cuál vemos como el proyecto se ha ido desarrollado a lo largo del tiempo.

En el capítulo 2 describimos varios conceptos, fundamentos y aplicaciones de la criptografía cuántica, haciendo hincapié en su uso en las comunicaciones satelitales.

A lo largo del capítulo 3 se habla sobre el proyecto GARBO, sus requisitos principales, los elementos de la misión, la colaboración con la plantilla de Thales, los objetivos técnicos y el plan desarrollo.

Posteriormente se procede a explicar los motivos que llevaron a elegir el software para realizar la herramienta, como se diseñó en un principio y los cambios que ha tenido, los requisitos, entradas, salidas, la seguridad de los datos manejados, el rendimiento que se alcanza y la arquitectura del código fuente, todo ello recopilado en el capítulo 4.

A continuación, en el capítulo 5 realizamos la implementación de la aplicación, detallando las herramientas que utilizamos, un manual de uso para el usuario, su integración en un **Suite de herramientas**, los planes de pruebas junto a su validación y la experiencia de entrega al usuario final.

Durante el capítulo 6 hablaremos de los conocimientos que hemos adquirido de sistemas con QKD, de herramientas de optimización, de **Matlab App Designer** y de **Latex**, estas últimas herramientas utilizadas para crear este trabajo.

En el capítulo 7 se desarrollan los próximos pasos que se podrían llevar a cabo partiendo de esta herramienta.

Por último, en el capítulo 8 presentamos nuestras conclusiones a partir de los resultados obtenidos tras modificar la configuración de los escenarios.

1.3 Objetivos.

Existen una serie de objetivos detallados en este apartado:

- Comprender el escenario en el que nos encontramos y conocer las necesidades de cada comunicación entre las estaciones terrenas y el satélite.
- Diseñar y desarrollar una aplicación en Matlab App Designer que optimice el volumen de bits de clave generado por el sistema.
- Elegir la herramienta y entorno de desarrollo adecuados.

- Desarrollar una interfaz gráfica que facilite la entrada de datos y el análisis de resultados.
- Describir el funcionamiento de la aplicación para que cualquier usuario pueda usarla independientemente de sus conocimientos.
- Examinar y comparar los resultados que hemos obtenido en cada configuración para su posterior estudio y desarrollo de conclusiones.

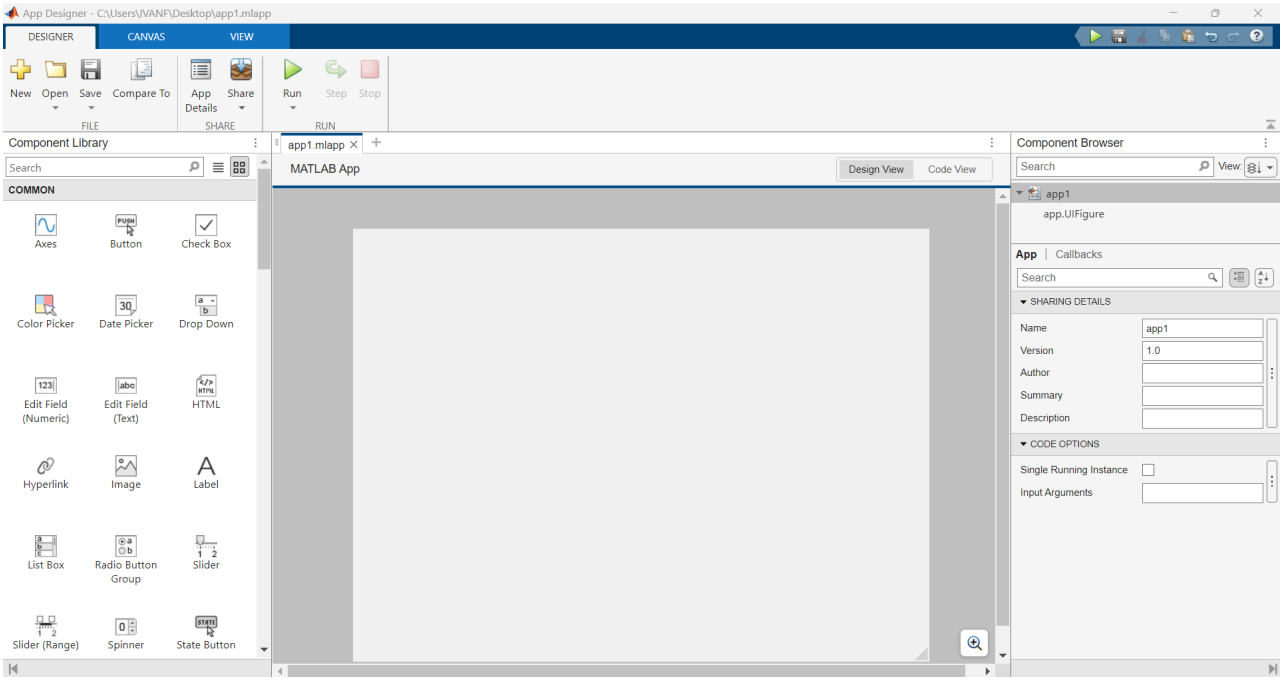


Figura 4: Matlab App Designer

1.4 Diversidad, ética social y sostenibilidad

En materia de diversidad, género y derechos humanos, este trabajo se ha realizado teniendo en cuenta todas las aportaciones relacionadas con referencias, independientemente del género del autor/autora y se ha desarrollado con el objetivo de hacerlo accesible para todo el mundo, independientemente del género, raza, etc.

Por lo que respecta al comportamiento ético y la responsabilidad social, creo que este trabajo tiene un impacto positivo porque contribuye a potenciar el uso de tecnologías aplicadas para mejorar la seguridad de las comunicaciones por satélite. GARBO es un proyecto clave que garantiza la privacidad de las comunicaciones. Este derecho

es clave para el desarrollo de una sociedad más justa y democrática, donde la protección de la información personal y sensibilidades comerciales es fundamental para preservar la libertad de expresión, la confidencialidad y el bienestar de los usuarios. Al promover tecnologías como la distribución cuántica de claves (QKD), este trabajo contribuye no solo a reforzar la seguridad de las comunicaciones, sino también a sentar las bases para un futuro más seguro y ético en las telecomunicaciones, asegurando que la información sensible esté protegida de amenazas crecientes, como los **ciberataques** y la vigilancia no autorizada.

Por último, en cuanto a la sostenibilidad, el trabajo se ha realizado siguiendo los estándares **ISO 14001** e **ISO 50001** con respecto a la gestión ambiental y medio ambiente, al igual que se sigue en Thales Alenia Space, sin generar gases de efecto invernadero ni sustancias prohibidas.

El proyecto GARBO cumple con la directiva europea de **DNSH (Do No Significant Harm)**. El principio DNSH es una condición que obliga a realizar una autoevaluación que asegure que la inversión o reforma no afecta negativamente a uno, o varios, de los 6 objetivos medioambientales definidos en el Reglamento 852/2020: Mitigación del Cambio Climático.

1.5 Cronograma

Este proyecto comenzó en febrero de 2024 como un apoyo para el proyecto GARBO que lidera Thales Alenia Space, con el objetivo de que sea empleado en un futuro por la empresa.

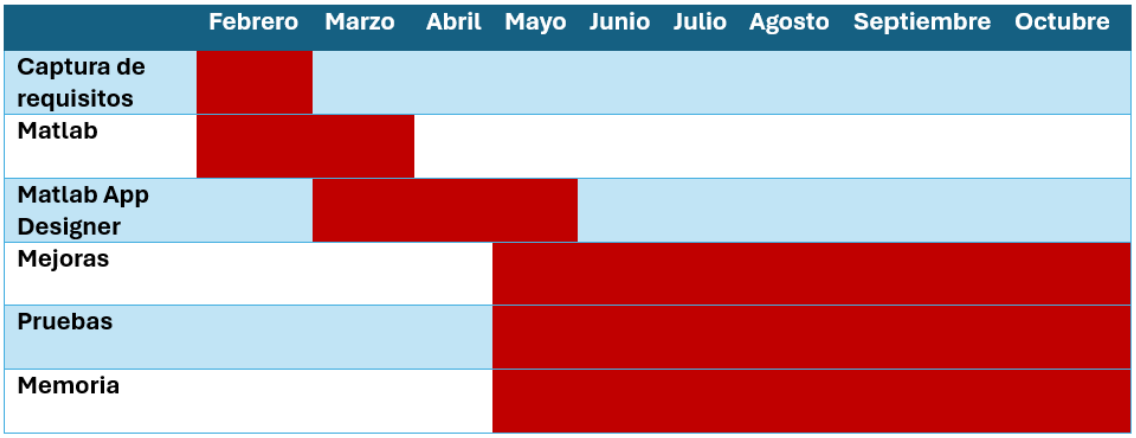


Figura 5: Cronograma del proyecto

Los pasos que se han seguido hasta lograr la versión final de la herramienta han sido los siguientes:

Plan de desarrollo: en primer lugar, se realizó una reunión con un grupo de responsables de Thales Alenia Space y el tutor, donde se establecieron las bases de la aplicación y su funcionamiento.

Cálculos: en esta primera versión se realizaba la lectura de datos de un archivo Excel proporcionado por Thales a través de un archivo de Matlab de tipo .mlx. Este programa realizaba las operaciones a través de la función *intlinprog* de programación lineal y mostraba los resultados obtenidos a través del terminal de Matlab.

Interfaz: reutilizando el código anterior, en esta versión lo que se hizo fue pasar de tener un archivo .mlx a una aplicación de Matlab (.mlapp) utilizando su App Designer, logrando tener una aplicación interactiva con botones, tablas, gráficas, etc.

Mejoras: se realizaron mejoras en cuanto a la funcionalidad del código y la apariencia visual, mediante por ejemplo la adición de un mapa mundial para visualizar la estación terrena, el satélite y las ciudades que participan en esa ejecución o la presencia de cuadros de texto para señalizar si las acciones del usuario se han hecho de manera correcta o no. También se hizo una fase de *debugging* donde se analizaron y corrigieron los errores del código fuente. En cuanto a la información proporcionada en el Excel, también ha sufrido modificaciones a medida que la herramienta iba creciendo en funcionalidades. Finalmente, se llevó a cabo la entrega al usuario final, durante la cual se atendieron sus dudas y se presentó la versión definitiva de la herramienta.

Pruebas: En este apartado vamos a mostrar el funcionamiento de la herramienta realizando una serie de casos de uso que muestran diferentes resultados. Cada prueba se compone de tres partes:

- **1.** Variación de valores dentro del archivo Excel para estudiar diferentes escenarios.
- **2**. La salida que esperamos ver para comprobar la correcta funcionalidad del software.

- **3.** Las conclusiones de cada escenario para comprender la capacidad de la herramienta.

Memoria: esta memoria comienza a escribirse a partir de la primera versión de la herramienta y ha sufrido modificaciones y mejoras desde ese momento hasta tener la versión final disponible presentar como TFG.

Capítulo 2: Fundamentos teóricos.

2.1 Introducción a la criptografía cuántica.

La **criptografía cuántica** es un tipo de criptografía que utiliza los principios de la física cuántica para crear un mensaje indescifrable para todos menos para el receptor previsto. Utiliza partículas cuánticas (**fotones**, electrones, etc) asignando una probabilidad a cada posible estado de la partícula para emitir la información a través de fibras ópticas o comunicaciones satélites.

No es posible hacer una replicación exacta de la información transmitida cuánticamente, por lo que un atacante no puede copiar la información transmitida, como dice el **teorema de la 'no clonación'**, formulado por Wootters y Zurek y Dieks en 1982. Tampoco es posible observar/medir la información cuántica sin introducir algunos rastros detectables, por lo que un atacante no puede medir los datos clave secretos que se intercambian sin dejar rastro. Estas propiedades garantizan la seguridad de QKD contra futuros desarrollos en los campos de la computación cuántica o matemáticas porque no se requieren suposiciones sobre la intratabilidad de los problemas matemáticos subyacentes. Por último, QKD no necesita ser desplegado/utilizado aislado de otros esquemas de seguridad, y, por ejemplo, se puede combinar con **PQC (Post-Quantum Cryptography)** u otras soluciones clásicas para mejorar la seguridad de todo el sistema.

El protocolo de seguridad QKD BB84 ha sido el elegido para llevar a cabo las comunicaciones del proyecto ya que es el más conocido, fiable y compatible con el balance de enlace:

Protocolo QKD BB84: Creado en 1984 por Charles H. Bennett y Gilles Brassard, utiliza la polaridad de los fotones de luz a través de un canal cuántico para comunicar al transmisor y al emisor. Los fotones que se utilizan representan un bit de información y se consigue mediante la codificación de estados no-ortogonales. A continuación, vamos a describir su funcionamiento.

Este protocolo *Prepare and Measure* permite un intercambio de claves entre dos nodos de una manera segura. Los usuarios (**Alice** y **Bob**) se comunican a través de un canal cuántico y un canal clásico. Además, el protocolo permite detectar un

espía (**Eve**) en el canal cuántico (o clásico). Para llevar a cabo este protocolo, la información (bits 0 y 1) se codifica en qubits. Cada bit clásico (0 o 1) puede ser codificado usando varias bases ortogonales de qubits:

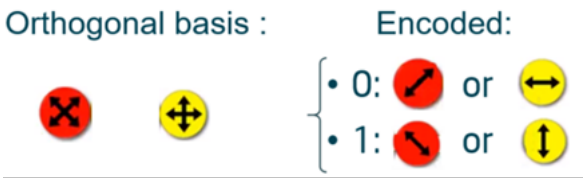


Figura 6: Bits clásicos codificados en dos bases ortogonales

Una vez que los qubits hayan sido codificados, si se miden utilizando la misma base en la que fueron codificados, se obtendrá una medición correcta. Sin embargo, si se miden con la base equivocada, solo hay una **probabilidad del 50 %** de obtener el dato correcto. Además, una vez que se mide el qubit, se colapsará al estado clásico medido. Debido a esto, si un qubit se mide en la base equivocada, la mitad del tiempo cambiará la codificación original del bit clásico. Esta propiedad física de los qubits permite a los usuarios legítimos detectar la presencia de un atacante/eavesdropper (Eve) en el canal de comunicación, ya que medirán incorrectamente la mitad del tiempo y esto modificará el estado del bit original en aproximadamente 1/4 de los bits codificados. La Figura 7 detalla los pasos del protocolo BB84:

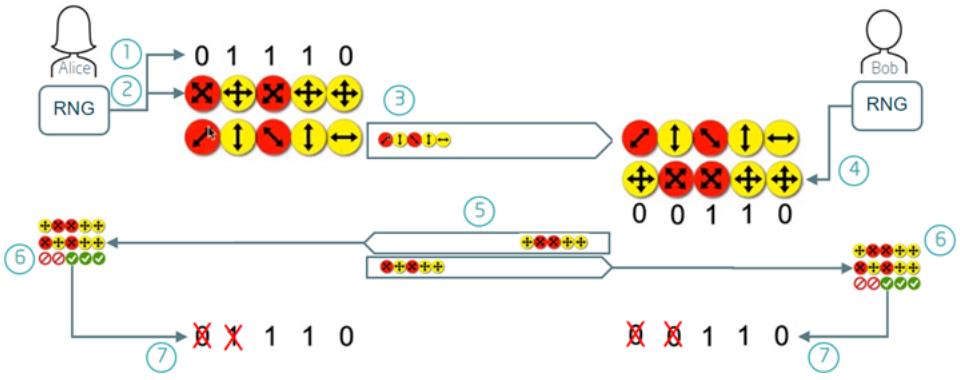


Figura 7: Pasos del protocolo BB84

- Alice elige una secuencia aleatoria para ser enviada a Bob.
- Alice elige aleatoriamente la base para codificar los valores de bits en qubits.
- Alice envía los qubits codificados a Bob por el canal público cuántico.

- Bob elige aleatoriamente la base para medir los valores de qubits recibidos y los mide.
- Bob y Alice intercambian a través del canal clásico público la base elegida para medir y codificar respectivamente.
- Después de compartir la base entre sí, ambos, Alice y Bob, saben qué qubits han sido codificados y medidos con la misma base.
- Alice y Bob descartan los qubits que han sido medidos con bases diferentes, y los restantes (que ahora son bits clásicos) construyen la clave acordada.

Como último paso, Alice y Bob tienen que seleccionar y compartir una muestra de la clave que luego eliminarán de la clave final. La seguridad del protocolo contra un atacante se basa en esta muestra. Si nadie ha medido los qubits, la muestra será idéntica (excepto para algunos errores menores). De lo contrario, si ha habido un atacante que mide los qubits, los valores de los bits en la muestra no coincidirán en aproximadamente un cuarto. Si esto ocurre, el protocolo es abortado y las claves compartidas se consideran inválidas.

Nótese que para garantizar la seguridad del protocolo QKD, el canal clásico público (utilizado, por ejemplo, para intercambiar la base elegida respectivamente) no está obligado a ser confidencial, sino que debe ser autenticado para evitar ataques Man-in-the-Middle.

En nuestro sistema GARBO, el papel de Alice será siempre desempeñado por la carga útil de QKD alojada en el satélite GEO, que, de esta manera, se convierte en un nodo de confianza de nuestra red de QKD, y el papel de Bob será desempeñado a su vez por las diferentes estaciones terrestres que son iluminadas y atendidas por la carga útil de QKD.

En el proyecto GARBO se usará un satélite en una órbita geoestacionaria que será capaz de dar cobertura a las dos orillas del atlántico, el cual será operado por parte de **Hispasat**.

2.2 Aplicaciones de la criptografía cuántica en comunicaciones satelitales.

La función principal es proporcionar una distribución de clave cuántica entre dos puntos remotos (estaciones terrestres) utilizando un satélite. Basado en el protocolo BB84 ya descrito, la Figura 8 a continuación introduce el proceso para dos nodos terrestres (Estación de Tierra A y Estación de Tierra B) para compartir la misma clave usando un satélite:

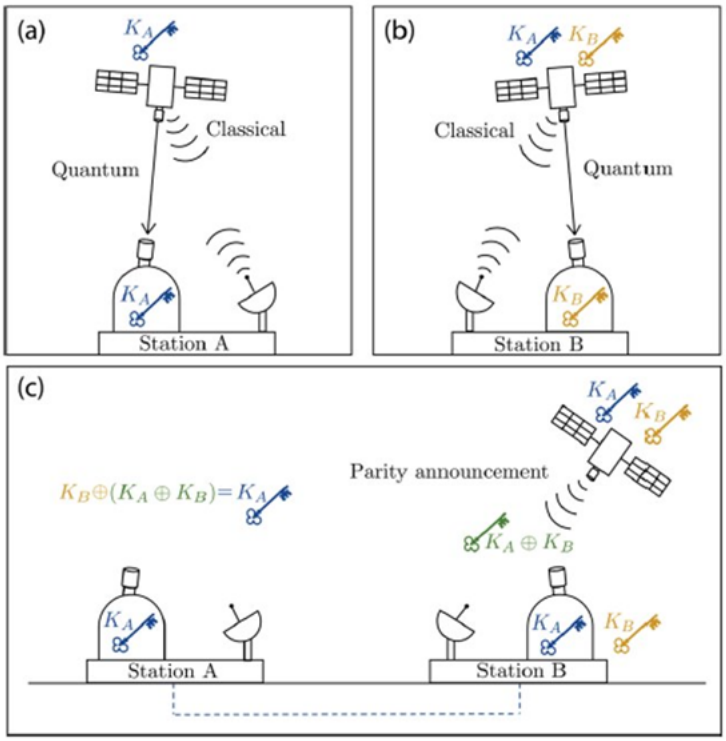


Figura 8: QKD por satélite entre dos estaciones terrestres

- En primer lugar, el satélite realiza un protocolo cuántico BB84 con la estación de tierra A, después de lo cual, ambos comparten la clave K_A
- Luego, de la misma manera, el satélite realiza otro protocolo cuántico BB84 para compartir la clave K_B con la estación de tierra B
- Por último, el satélite utiliza K_B como clave One Time Pad (OTP) para distribuir a través del canal clásico a las estaciones de tierra
- El satélite realiza un XOR = (K_A + K_B) y lo envía a las estaciones de Tierra A y B

- Estación de tierra B obtiene el valor de realizar un XOR = (K_A + K_B) + K_B=K_A
- Igualmente, Estación de tierra A obtiene el valor de realizar un XOR = (K_A + K_B) + K_A=K_B

2.3 Comunicación entre estaciones base y satélite.

La comunicación entre dos estaciones terrestres a través de un satélite en una órbita determinada se realiza utilizando el satélite como un repetidor de señal entre la estación transmisora y receptora.

En función de la órbita utilizada el satélite será capaz de cubrir una determinada superficie terrestre. Estas órbitas se clasifican en función de la altura que tengan sobre la superficie de la Tierra:

Órbita terrestre baja (Low Earth Orbit, LEO): satélites que están **entre 200 y 2.000 km de altura** sobre la Tierra, se suelen usar para observación de la Tierra y comunicaciones móviles.

Ventajas: presentan pocas pérdidas por propagación, necesitando antenas de menor tamaño y tienen un bajo retardo.

Desventajas: se necesitan muchos para cubrir toda la Tierra de forma permanente, sufren el efecto Doppler, tienen visibilidad breve y producen gran cantidad de basura espacial.

Órbita terrestre media (Medium Earth Orbit, MEO): cuando el satélite está **entre 10.000 y 14.000 km de altura** aproximadamente, se usan para sistemas de navegación por satélite (GNSS), como GPS, Galileo o GLONASS.

Ventajas: ofrece cobertura global con 8-12 satélites, los terminales son pequeños y tiene un retardo medio.

Desventajas: sufre **efecto Doppler**, la visibilidad es breve y se necesita una compleja arquitectura de red.

Órbita terrestre geoestacionaria (Geostacionary Earth Orbit, GEO): cuando la altura del satélite está cerca de los **36.000 km de altura**, usados para meteorología y comunicaciones que necesitan que el satélite esté fijo en todo momento.

Ventajas: se consigue una cobertura global con 3 satélites, el efecto Doppler es mínimo y la señal es estable.

Desventajas: no cubre las zonas polares, tiene un retardo considerable, los terminales son más grandes, tienen muchas pérdidas de enlace y un alto coste en el lanzamiento.

Órbita terrestre alta (High Earth Orbit, HEO): Altura entre 1.0000 y 70.000 km, existen 2 tipos:

Molniya: usada por la antigua URSS, sirve para dar cobertura a zonas polares.

Tundra: evita el bloqueo de edificios debido a su inclinación, se considera una órbita geosíncrona.

Ventajas: dan cobertura a zonas polares, el coste de lanzamiento es menor que en GEO y con 3 satélites se da cobertura permanente.

Desventajas: no dan cobertura global, tienen grandes pérdidas por enlace, retardo considerable, sufren efecto Doppler y las órbitas cruzan los **cinturones de Van Allen**.

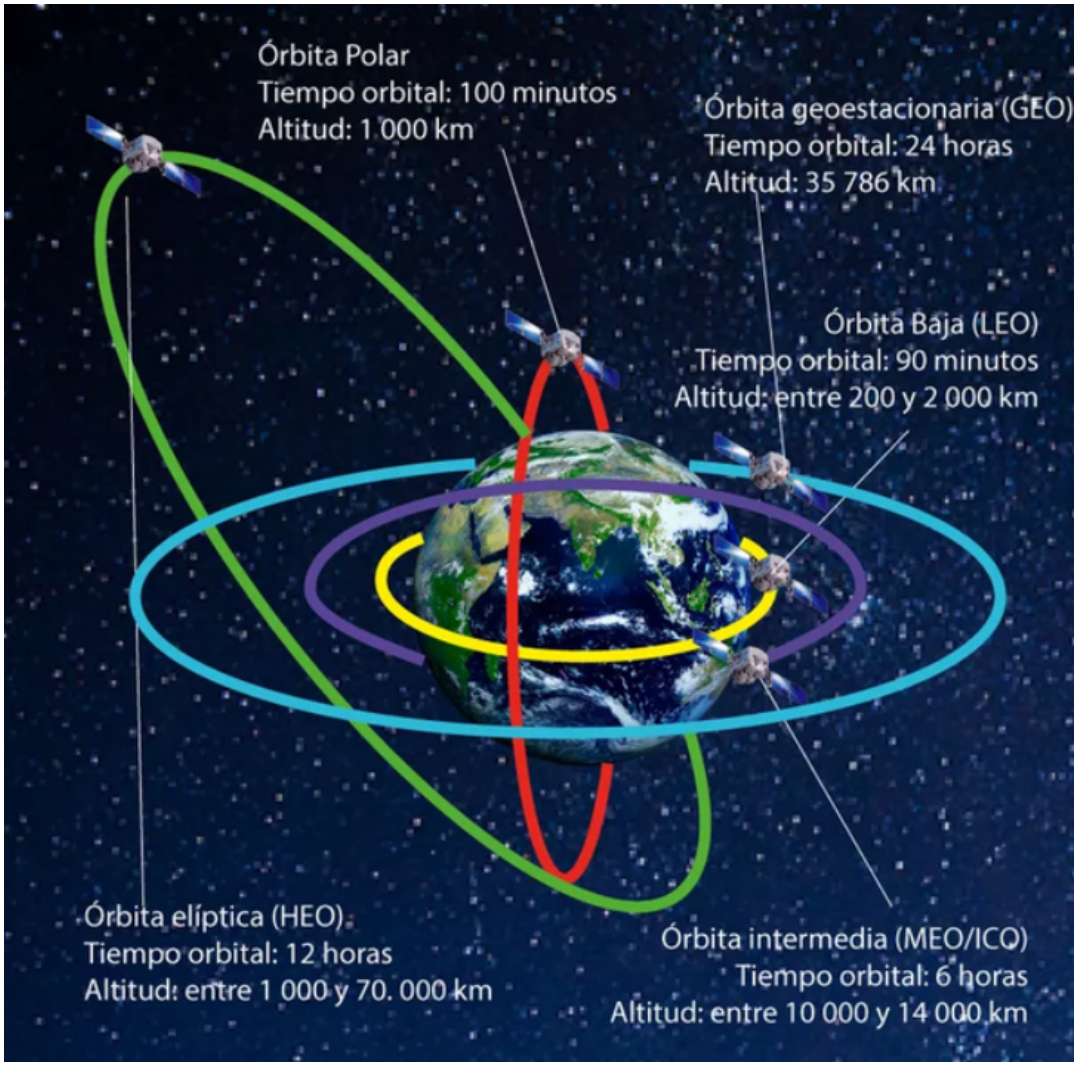


Figura 9: Órbitas satelitales

CARACTERÍSTICAS	NO GEOESTACIONARIOS		GEOESTACIONARIO
	LEO	MEO	GEO
Tipo de Orbita	Circular	Circular	Circular, Geosincrona y Ecuatorial
Coste de lanzamiento	Medio	Medio	Muy Alto
Coste del Segmento Espacial	Alto	Bajo	Medio
Complejidad de explotación	Bajo	Medio	Muy Alto
Vida del Satélite (años)	3 a 7	10 a 15	10 a 15
Coste de las Pasarelas	Muy costoso	Costoso	Economico
Terminales de Bolsillo	Muy Dificil	Posible	Posible
Retardo de Propagación	Medio(20-270)	Bajo(5-20 ms)	Alto(270 ms)
Perdidas de Propagacion	Bajas	Media	Alta
Ángulos de Elevación	45 (Bajo y malo)	55 (Medio)	0
Complejidad de las	Compleja	Media	Sencilla
Handover	Frecuente	Infrecuente	Inexistente
Tiempo de Desarrollo	Largo	Corto	Largo
Efecto Doppler	Alto	Medio	No
Daño por Radiacion	No pasan por cinturones de radiacion	Solo durante el Lanzamiento	Solo durante el lanzamiento
Satélites	40 a 80	8 a 20	3
Visibilidad del satelite	Corta	Media	Siempre
Satelites para una cobertura global	> 20	8 a 16	3
Distancia a la Tierra	200-3000 km	3000-35786 km	35786 km
Aplicaciones	Internet,datos,voz, navegacion	GPS	Satellite TV

Figura 10: Tabla comparativa de órbitas

En el proyecto GARBO se quiere tener cobertura permanente las 24 horas del día, por lo que se usa un satélite en órbita geoestacionaria, aunque imponga algunas limitaciones técnicas al proyecto como puede ser la atenuación, el apuntamiento y el retardo.

En las comunicaciones ópticas existen problemas de disponibilidad debido a la presencia de nubes que pueden bloquear o atenuar la señal, por lo que en regiones con alta nubosidad los enlaces son afectados. Como solución podemos instalar **estaciones terrestres distribuidas**, utilizar longitudes de onda alternativas o usar comunicaciones híbridas (ópticas y RF), que ayudan a mitigar los efectos adversos

de las condiciones meteorológicas.

Debido a estos problemas de disponibilidad nuestros terminales pueden estar en modo activo (1) o inactivo (0).

Capítulo 3: GARBO

El proyecto ha sido bautizado como GARBO, nombre en clave del espía español Joan Pujol, que tuvo un papel clave en la operación Fortitude, haciendo creer al ejército alemán que el desembarco aliado tendría lugar en Calais y no en Normandía. Este proyecto supondrá una aportación clave de la industria española en el futuro de la seguridad europea.

El objetivo de la misión GARBO es el desarrollo de las capacidades espaciales necesarias para diseñar, desarrollar y construir la primera misión cuántica QKD sobre un satélite geoestacionario. La misión se llevará a cabo en España, que participará en la fabricación espacial. Se desarrollará de manera experimental, con la intención de impulsar este tipo de nuevos servicios y mercados de comunicaciones seguras, con la intención de una posterior comercialización para comunicaciones gubernamentales o comerciales. GARBO estará totalmente alineado con las iniciativas europeas actualmente en desarrollo y España será la principal aportación en las mismas. Con estas aportaciones el objetivo del **CDTI (Centro para el Desarrollo Tecnológico y la Innovación)** es posicionar a España como pionera en el desarrollo de este tipo de comunicaciones, posicionando a este tipo de industria como líder de Europa.

La misión GARBO se plantea proporcionar un sistema basado en una carga útil QKD óptica alojada en un satélite geoestacionario que permita el establecimiento de una conexión cuántica entre dos puntos (Estaciones Terrestres) a una distancia mayor que el rango permitido por las redes ópticas, aunque tendrá que lidiar con la precisión de apuntamiento, la atenuación de la señal o la sincronización. Existe otro caso de satélite QKD desarrollado por China que fue lanzado en 2016 (**Micius**) y opera en órbita LEO a 400km de altura, por lo que su servicio es muy irregular.

Se realizará también una validación funcional del segmento terreno y una carga útil pre-integrada, que se comunicarán en un enlace atmosférico de 140 km entre dos islas de Canarias. Esto permite simular las condiciones de contorno que hacen el ensayo totalmente representativo, y garantizará el funcionamiento en órbita del sistema. El siguiente paso, objeto de actividades futuras, sería su acomodación como carga útil a bordo de un satélite de comunicaciones.

3.1 Requisitos principales.

El CDTI tiene como objetivo posicionar a España como líder de los futuros sistemas de distribución de clave cuántica (QKD). Los avances en computación van a hacer posible romper los sistemas criptográficos tradicionales por lo que estos sistemas cuánticos son sin duda, necesarios. Para ello, se propone diseñar un sistema de distribución de claves cuánticas desde un satélite en órbita geoestacionaria. Es un sistema completo innovador que desarrolla no sólo el método de destilación de claves, sino toda la cadena de transmisión, el segmento terreno y la interconexión de todo ello.

Es necesario identificar y analizar los parámetros clave del enlace QKD debido a la distancia entre emisor y receptor, unos 36.000 km. También se deben identificar y definir las funciones de seguridad, que garanticen que la operación de este sistema cumpla con las necesidades de la Administración Española.

La fase A finalizada en 2022 en el marco de la Agencia Espacial Europea se denominó **Proyecto Caramuel**, el cual demostró la viabilidad teórica e identificó las tecnologías clave para conseguir los resultados. Este nuevo proyecto desarrollará los diversos elementos y su integración para demostrar el funcionamiento de la tecnología, y la madurez de la solución.

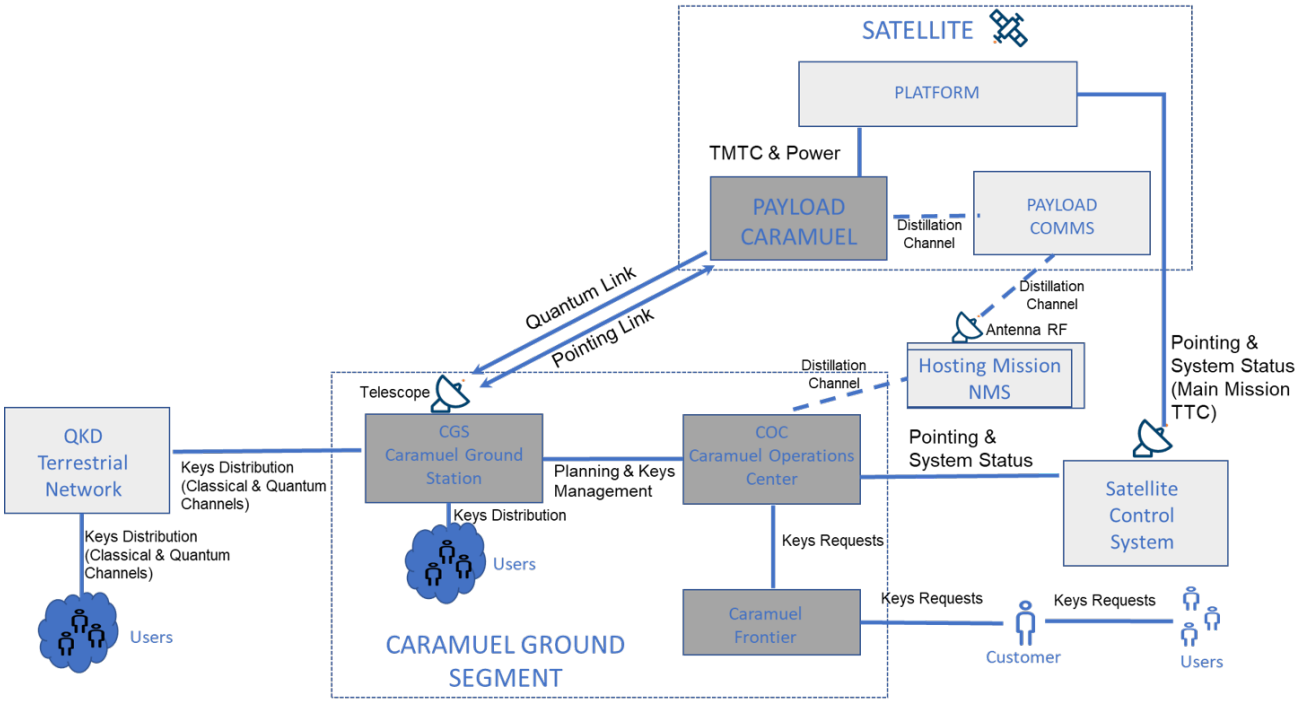


Figura 11: Esquema del proyecto Caramuel

3.2 Elementos de la misión.

El sistema se compone de un segmento espacial y un segmento terreno:

El **segmento espacial** es una carga útil secundaria alojada en un satélite geoestacionario de comunicaciones. Algunos elementos clave son: generador de números aleatorios (QRNG), fuente de fotones polarizados (Signal Source), terminal óptico (Telescopio) y su sistema electrónico que envía la información a tierra, así como los mecanismos encargados del apuntamiento, con su electrónica integrada y un sistema de procesado y sincronización (Protocol Processor) que compila la información generada y sincroniza la parte terrena y espacial. El sistema es independiente de la plataforma satelital de la que sólo utiliza, para reconciliar las claves, un canal tradicional de comunicación, típicamente un enlace de radiofrecuencia.

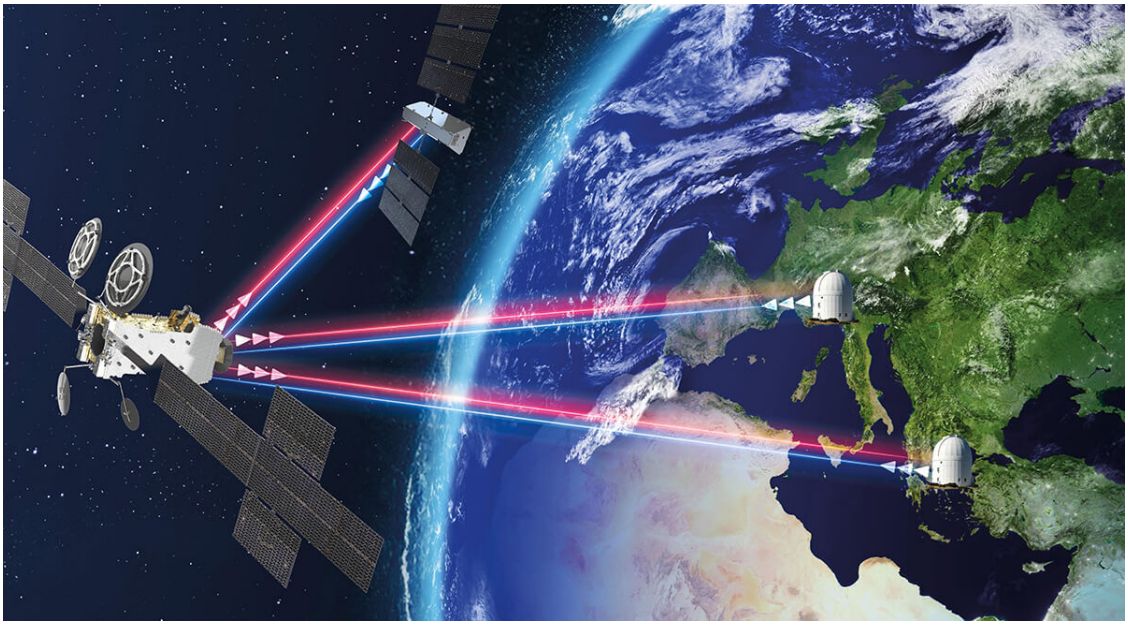


Figura 12: Segmento espacial en satélite geoestacionario

El **segmento terreno** consta de la estación terrena (GOR) y el centro de operaciones (COC). La estación terrena tiene un sistema óptico que recibe los fotones enviados desde el espacio, detectores de señal y funciones de procesado y sincronización para la destilación de las claves. El centro de operaciones organiza todas las actividades del sistema. Controla a las estaciones terrenas y recibe las necesidades de los usuarios para planificar sesiones de comunicación entre el segmento espacial y las diferentes estaciones terrenas. También conecta con los centros de control de la misión principal, enviando y recibiendo los datos que llegan a la carga útil mediante el canal tradicional. Además de todas esas funciones, se encarga de la planificación de la herramienta final de optimización, donde mi proyecto sería una primera aproximación.

Figura 13: Estación terrena y centro de operaciones en Maspalomas, Gran Canaria

Al ser una carga útil secundaria, el sistema tendrá interfaces con otros elementos externos: la plataforma satelital y la misión de telecomunicaciones en el propio satélite y el sistema de gestión de la red (NMS) y el sistema de control del satélite (SCS) dentro del segmento terreno del satélite. La plataforma satelital alojará en un futuro este sistema como carga útil secundaria y la misión de telecomunicaciones deberá proporcionar un canal de radiofrecuencia para el mecanismo de reconciliación de claves mediante el protocolo BB84 para poder configurar y operar la carga útil. En tierra, el sistema de gestión de red deberá proporcionar un canal virtual para comunicar la carga útil con el centro de operaciones COC. El sistema de control del satélite interactuará con el COC para proporcionar los servicios básicos de control y telemetría de la carga útil.

3.3 Integración, Verificación, Validación y Calificación progresiva del sistema.

La estrategia de integración, verificación y validación permitirá demostrar que las funciones más importantes se han implementado correctamente, validando las cadenas funcionales de transmisión de claves cuánticas, apuntamiento y seguimiento, gestión de carga útil, y gestión de claves.

La validación será progresiva, partiendo de modelos software, pasando por pruebas con elementos comerciales, hasta acabar con prototipos de alta madurez tecnológica. Está previsto construir un gemelo digital del sistema. Partirá de las cadenas funcionales definidas y modelará el funcionamiento del sistema. Se construirá a partir de los bloques funcionales y se irá refinando de modo iterativo, desde una definición en

UML, pasando por modelos de pregunta-respuesta, modelos unitarios, simulaciones parciales, hasta conseguir una simulación completa, incluyendo casos extremos y modos de fallo.

El objetivo es que el software objeto de este TFG se integre dentro de este **Digital Twin**, por lo que las interfaces se han definido de forma que aseguren la futura compatibilidad con las nuevas integraciones en este entorno. Esta herramienta se corresponde con la primera versión del **futuro planificador de recursos** del sistema.

Se realizarán pruebas funcionales con los diferentes elementos, usando un simulador de misión anfitriona (Hosting Mission Simulator) que proporcione los estímulos necesarios a la carga útil, incluyendo las características de los enlaces geoestacionarios. Se usará fibra óptica para las pruebas unitarias (mantenimiento, apuntamiento, transmisión, destilación y gestión de claves).

3.4 Colaboración con plantilla de Thales Alenia Space España.

El primer paso para el desarrollo de la herramienta fue la realización de una reunión junto con el tutor y una serie de responsables del equipo de proyecto GARBO. En ella se estableció la necesidad que tenían de crear una herramienta que fuera capaz de realizar un cálculo de optimización independientemente del escenario en el que nos encontremos. Cada escenario tiene una necesidad diferente de volumen de clave, que depende del **SKR (Symbol Key Rate)**, del número de estaciones, del tamaño de telescopio, la elevación, la localización y el entorno. En cuanto a este último puede haber diferentes variables que lo definan, como por ejemplo las turbulencias, la contaminación lumínica o las nubes.

Durante el desarrollo de la herramienta se realizaron varias reuniones con el equipo del proyecto, con el fin de obtener información detallada sobre las funcionalidades requeridas, proponer mejoras potenciales y abordar cualquier duda que surgiera durante el proceso. Estas reuniones facilitaron una comunicación fluida entre todos los involucrados, permitiendo una validación continua del progreso y asegurando que la herramienta cumpliera con las expectativas del usuario final y con los objetivos del proyecto.

Dos de los responsables del equipo de proyecto son **Edoardo Vanin**, ingeniero

de sistemas, encargado del Digital Twin y usuario final de la herramienta, y **Siro Muela**, ingeniero aeroespacial, encargado de la herramienta de balance de enlace y suministrador de los datos de entrada para la herramienta.

3.5 Objetivos técnicos.

Esta aplicación tiene como objetivo optimizar la manera en la que se distribuyen slots de día y de noche para transmitir bits de claves cuánticas entre estaciones terrestres y un satélite.

La elección de slots puede ser de día y noche, o solo de noche, debido a que de día tenemos mucha luz solar difusa, la cual actúa como ruido. Por lo tanto, los slots de día y de noche tienen diferente SKR.

Se ha decidido utilizar una función de programación lineal de enteros (MILP) llamada *intlinprog*, la cual define un conjunto de límites inferiores y superiores de modo que la solución siempre se encuentra en el rango $lb \leq fval \leq up$. Esta función devuelve los slots necesarios tras la optimización y el valor final de volumen de clave generado.

Este es el funcionamiento interno que utiliza para encontrar el mínimo de un problema especificado.

$$\min_x f^T x \text{ subject to } \begin{cases} x(\text{intcon}) \text{ are integers} \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub. \end{cases}$$

Figura 14: Funcionamiento interno intlinprog

f, x, intcon, b, beq, lb y ub son vectores, y A y Aeq son matrices.

f es el vector de función objetivo, una matriz con los valores de volumen de clave de cada tipo de slot.

x será el valor de volumen de clave tras el cálculo de optimización. También podemos obtener los valores de cada slot si sustituimos x por la expresión [slots, fval], donde slots sería una tabla con los slots y fval el volumen de clave generado.

intcon es el número de variables que intervienen en el cálculo, cuatro.

A es la matriz de restricción de desigualdades lineales.

b es el volumen de clave requerido, una restricción de desigualdad lineal.

beq y Aeq serían restricciones de igualdad lineales, es nuestro caso no se necesitan.

lb y ub son las restricciones de límite, para nuestra optimización sería que el volumen no puede ser menor que cero.

Debido a los requerimientos de Thales Alenia Space, se optó por esta opción de optimización en lugar de usar otros métodos como el **Monte Carlo** o el **enjambre de partículas (PSO)**, debido al tipo de datos que había que manejar.

Cuando se generan los bits de claves diarios y se cumplen los requerimientos de cada terminal, los bits restantes dejan de ser útiles debido a que las claves que los usan caducan, por lo que la operación para el siguiente día comienza de nuevo. Además, cuando un terminal no está activo un día su requerimiento de clave se acumula al día siguiente, por lo que la necesidad aumenta. Para representar el estado del terminal tenemos una variable que puede ser "1", en el caso de que esté activo, o "0", no disponible.

En cuanto a la implementación del código Matlab, en la sección de *properties* tenemos todos los componentes de la librería que hemos usado junto al tipo de componente que son. También podemos crear nuestras propiedades privadas como el caso de la ruta donde se encuentra el archivo Excel utilizado. Tenemos el apartado de *callbacks* donde a través de funciones diseñamos el funcionamiento de los componentes de la aplicación. Por último, tenemos una parte de descripción de los componentes donde podemos ver sus características como su posición en el lienzo, su tipo de componente o los nombres de las columnas de las tablas, por ejemplo. Toda esta parte de código está acompañada de comentarios para que cualquier persona que acceda a él conozca el funcionamiento del software.

Esta parte de código se compone de 726 líneas, que incluyen tanto la creación de las figuras como las definiciones de sus cometidos. La primera parte está compuesta por las propiedades de los componentes, definidas dentro de la clase que lleva el nombre de la aplicación. Tras esta parte se encuentran las variables globales, junto con la

función de optimización.

```
properties (Access = private)
    ruta; % Ruta donde se encuentra el archivo Excel
    archivo; % Archivo Excel utilizado
    gx; % Mapa mundial
end

methods (Access = private)

    % Función para optimizar los slots
    function [slots, fval] = optimizacion(~, f1, f2, f3, f4, a1, a2, a3, a4, b)
        f  =  [f1 f2 f3 f4];
        A  =  [a1 a2 a3 a4];
        b = b(:);

        lb=zeros(size(f));
        ub=[];
        intcon=1:4;

        [slots, fval] = intlinprog(f, intcon, A, b, [], [], lb, ub);
    end
end
```

Figura 15: Variables globales y función de optimización

En el apartado de *callbacks* tenemos las funciones que diseñan el funcionamiento de los distintos elementos de la interfaz.

Para finalizar, están definidos los componentes con sus dimensiones y propiedades, junto con unas propiedades que se crean automáticamente al crear la aplicación.

3.6 Plan de desarrollo.

En primer lugar, se produjo una reunión con el usuario final para estudiar la propuesta de herramienta que se quería diseñar, con sus entradas, salidas y arquitectura. Tras estudiar estos requisitos se procedió a diseñar la herramienta.

- El primer paso fue la elección del software para su creación, valorando su versatilidad y sus posibilidades. Finalmente, las dos opciones favoritas fueron Python y Matlab, decantándome por esta última debido a la funcionalidad que requería la herramienta y a los conocimientos que había adquirido anteriormente en el grado, debido a su uso en múltiples asignaturas.
- El siguiente paso fue estudiar el archivo Excel que contiene la información

proporcionada por Thales para determinar cuál era la necesaria para la herramienta y cuál servía para identificar los distintos escenarios.

- Una vez seguidos estos pasos comenzó la creación del código creando un archivo de tipo .mlx. En un principio los requisitos eran que la herramienta tuviera un procesado *offline*, con una tabla de entrada de datos, un procesado y una salida de resultados a través del terminal de Matlab y de la exportación de una tabla en un archivo Excel. Tras conseguir que el código tuviera la funcionalidad deseada se procedió a presentarlo al usuario final para determinar su valoración de la herramienta y sus posibles mejoras.

- La herramienta cumplía con las expectativas funcionales, por lo que el siguiente paso era hacerla mucho más versátil para su posterior uso. El usuario final decidió que el siguiente paso era la creación de una interfaz gráfica interactiva que tuviera un procesado *online*. Para ello disponíamos de una opción dentro de Matlab llamada **App Designer**, lo que nos permitió reutilizar el código anterior sin problema para continuar con su desarrollo.

- A continuación, creamos el archivo de tipo .mlapp y nos dispusimos a explorar la gran cantidad de opciones de diseño y funcionalidades nuevas. Se interactuó con la interfaz gráfica hasta comprender su funcionamiento introduciendo distintos elementos como botones o tablas. Cuando se entendió completamente el funcionamiento de la aplicación se procedió a reutilizar el código para que realizara las mismas acciones que hacía anteriormente, pero teniendo como salida una aplicación en lugar de un terminal.

- Una vez que se consiguió este funcionamiento comenzó la introducción de mejoras, como por ejemplo la posibilidad de variación de datos en las tablas útiles para la herramienta con su inmediata modificación en el archivo Excel, permitiendo modificar datos en función de las necesidades de cada escenario sin la necesidad de salir de la herramienta, lo que haría que fuera poco ergonómica.

- Otras mejoras que se introdujeron fueron la posibilidad de ver los resultados de los slots a través de un gráfico, la introducción de un mapa mundial donde figuren las localizaciones del satélite que se usará, la estación terrena, situada en Maspalomas y los distintos terminales que intervienen en el cálculo de optimización, y la posibilidad de exportar los resultados en un archivo de tipo .csv.

- Desde la primera implementación hasta la última mejora se desarrollaron distintos planes de pruebas para verificar el correcto funcionamiento de la herramienta, comenzando con un terminal que actúa de día y de noche, de noche solo, o poniendo un requisito muy alto de clave para comprobar su convergencia, y acabando con probar con una gran cantidad de terminales con distintas necesidades y disponibilidades.
- Cuando se consiguieron todas estas mejoras se consultó a través de correo electrónico con el usuario para comprobar que estas modificaciones estaban correctamente realizadas y para saber si eran las últimas o había alguna más. Tras analizar la aplicación el usuario nos propuso que además de la funcionalidad que tenía la aplicación, esta debía ser capaz de expresar la **convergencia** de los escenarios a través de otra tabla de resultados complementaria, que contendría los días que necesitaba cada terminal junto con los días objetivo que se habían estipulado, la cantidad de horas de día y de noche de cada slot y la duración de la optimización para comprobar su capacidad de ejecución.
- Finalmente, cuando la aplicación tuvo el diseño deseado se procedió a presentársela presencialmente al usuario final en la sede de Thales en Tres Cantos a través de una demostración práctica, realizando la entrega final de la herramienta y puesta en servicio.

Capítulo 4: Diseño del sistema.

4.1 Elección del entorno de desarrollo.

Para diseñar esta herramienta hemos optado por utilizar Matlab, en concreto Matlab App Designer, que nos permite diseñar una aplicación interactiva con botones, tablas, gráficos, etc. Las ventajas que nos presenta Matlab en esta herramienta frente a otras como Python son la presencia de la función *intlinprog* de programación lineal, la cual se encarga de realizar el cálculo de optimización, la presencia de una interfaz gráfica, que nos permite diseñar la aplicación de una forma más cómoda y eficiente, y la compatibilidad con otras herramientas de Thales Alenia Space, lo que permitirá su futura integración de una manera efectiva.

En lo que se refiere a los conocimientos adquiridos del concepto de optimización, tras estudiar diferentes modelos, como métodos heurísticos, algoritmos estocásticos y algoritmos deterministas, aprendí a valorar las ventajas y limitaciones de cada uno y como podían encajar en el funcionamiento de la herramienta. Esto sumado a los conocimientos que ya había adquirido en la asignatura de **Investigación Operativa** me dieron las herramientas necesarias para realizar el cálculo de optimización.

Método de optimización	Ventajas	Desventajas
Algoritmos deterministas	- Muy eficiente para problemas de programación lineal. - Garantiza encontrar soluciones óptimas. - Predictibilidad del tiempo de ejecución	- Alto costo computacional en problemas grandes - Dificultad para adaptarse a problemas complejos - Falta de escapatoria de mínimos locales
Métodos heurísticos	- Soluciones rápidas para problemas complejos - Escapan de mínimos locales - Pueden manejar problemas con no linealidades, discontinuidades o múltiples óptimos locales	- No garantizan una solución óptima - Resultados variables - Depende fuertemente de la naturaleza específica del problema
Algoritmos estocásticos	- Capacidad para escapar de mínimos locales - Flexibilidad para problemas complejos - Adaptabilidad a diferentes problemas	- Convergencia lenta. - Resultados variables - No son adecuados para problemas pequeños o simples

Figura 16: Comparativa métodos de optimización

En cuanto al aprendizaje de la herramienta Matlab App Designer, esta comenzó con una familiarización con la interfaz gráfica y las funciones básicas, como pueden ser las opciones del menú o las distintas figuras presentes en la librería de componentes

de la vista de diseño, y la elección de un diseño con una interfaz dividida en tres partes debido a que al distribuir los elementos de esta forma la aplicación es mucho más intuitiva para el usuario.

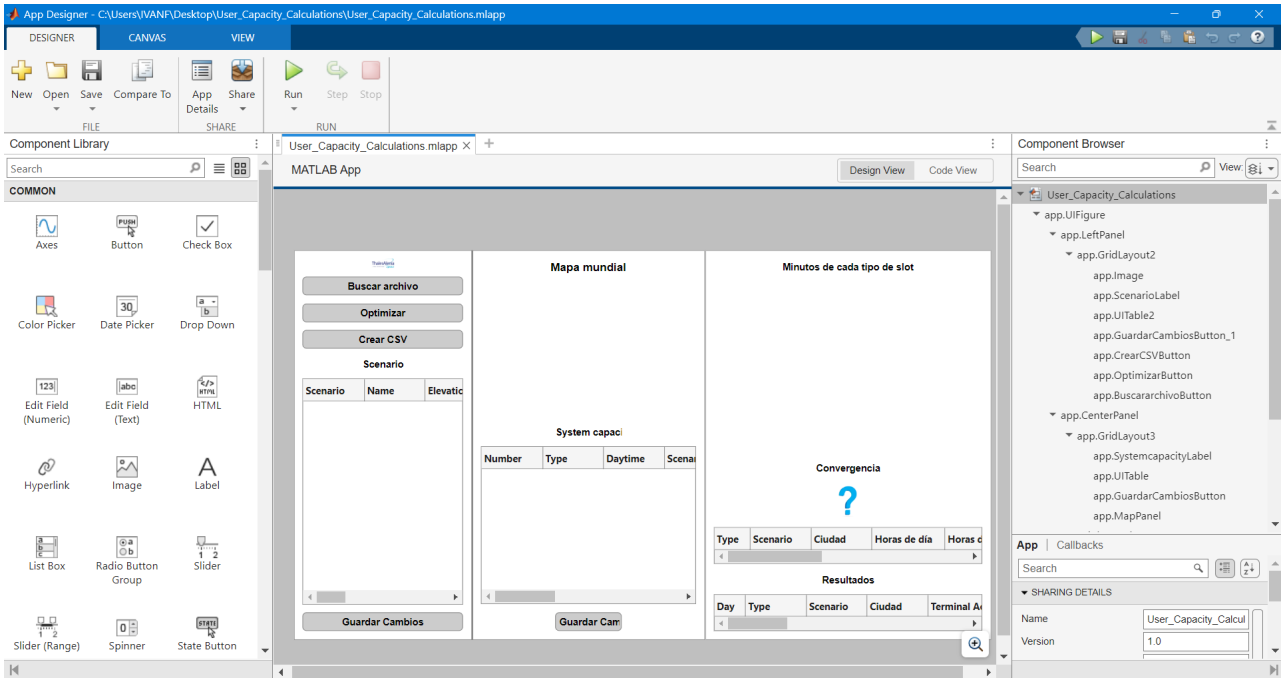


Figura 17: Interfaz de Matlab App Designer

Tras esta fase de familiarización, se procedió a comprender el funcionamiento de los *callbacks*, necesarios para poder programar la funcionalidad de los botones y la presencia del mapa mundial, y de las *properties*, variables globales que permiten su uso en cualquier función intermedia.

Como última fase tenemos el dominio y la personalización, en las cuáles se procedió a retocar la parte de código para hacerlo más eficiente y a modificar la interfaz gráfica para hacerla más vistosa, dando color a los botones o introduciendo imágenes, por ejemplo.

4.2 Interfaz gráfica.

Cuando se inició la herramienta se utilizó Matlab, por lo que no existía una interfaz gráfica, solamente se disponía de un apartado para programar el código y una salida a través de un terminal.

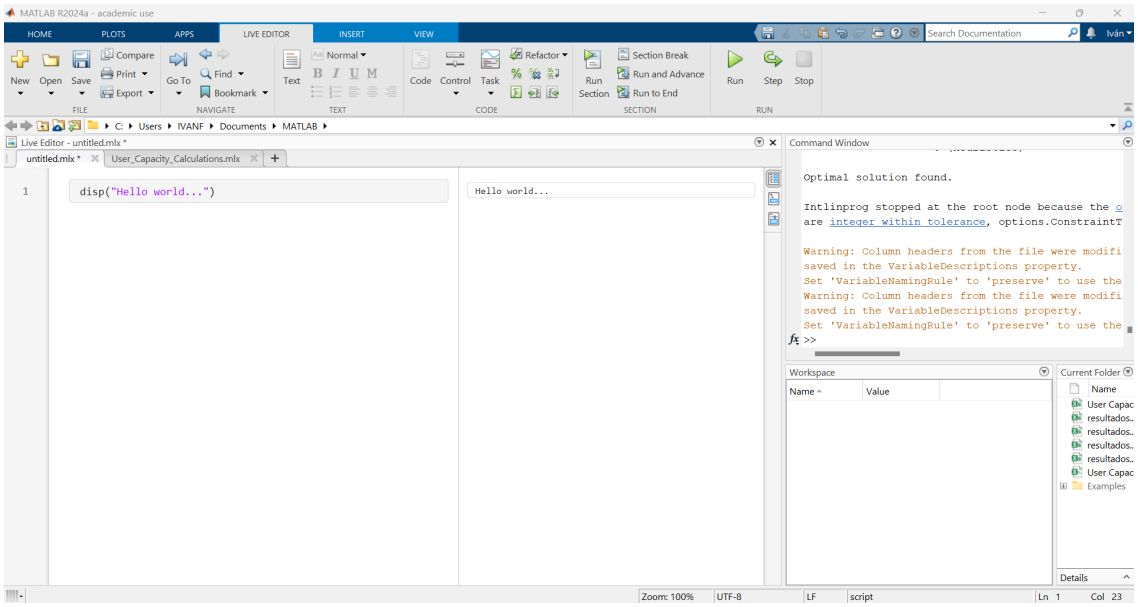


Figura 18: Interfaz de Matlab

En cambio, con App Designer somos capaces de diseñar una interfaz que sea tanto intuitiva como atractiva para el usuario. Para ello el lienzo de la interfaz se divide en **3 partes**:

- La parte izquierda consta de 3 botones funcionales para poder cargar el archivo correspondiente, iniciar la función de optimización o crear un archivo .csv, junto con la tabla *scenario* y un botón para poder guardar los cambios que se consideren oportunos.
- La parte central se compone de un mapa mundial donde se ubica la estación terrena situada en Maspalomas, Gran Canaria, y la tabla *System Capacity*, junto con el botón para guardar cambios al igual que en la tabla anterior.
- Por último, en la parte derecha del lienzo se encuentra la parte de resultados en la que, tras realizar la optimización, veremos un gráfico con porcentajes, una tabla para mostrar la convergencia de los terminales y una última tabla de resultados donde se apreciará el número de cada tipo de slots, entre otra información.

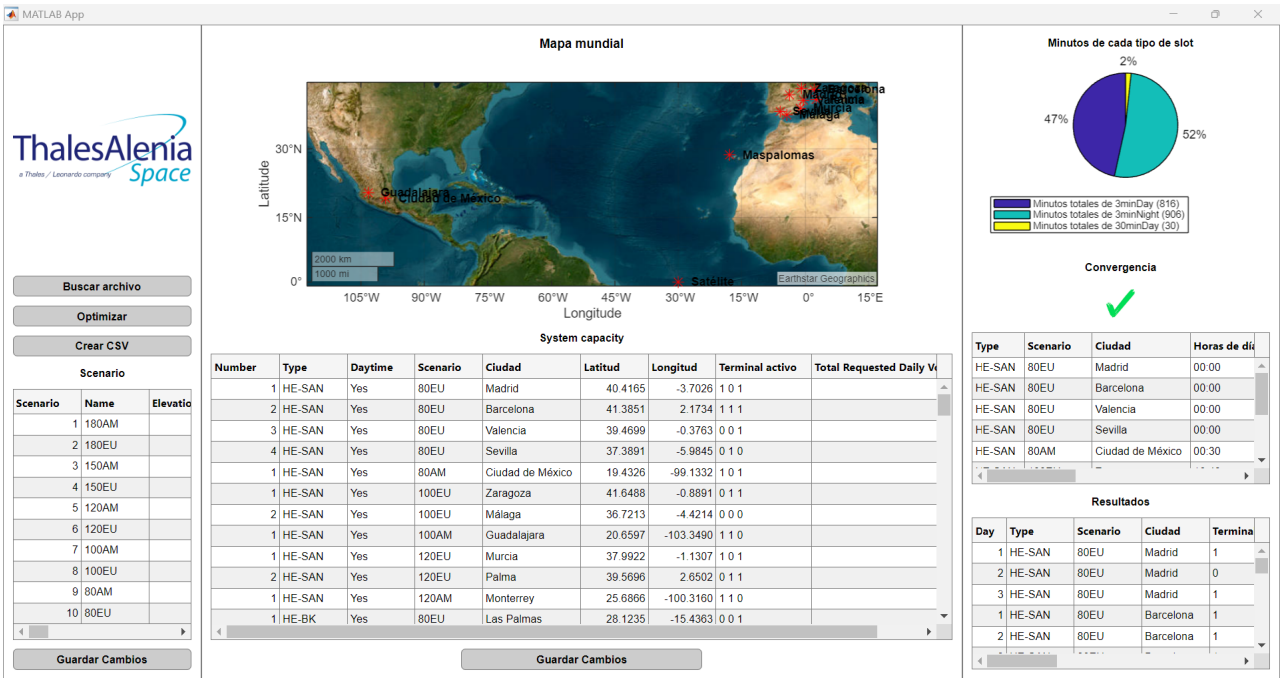


Figura 19: Ejemplo de aspecto final de la aplicación

4.3 Requisitos, entradas y salidas

En un principio los requisitos exigidos por el usuario para la herramienta eran que fuera capaz de realizar un cálculo de volumen de clave requerido por cada terminal a través de la combinación de una serie de slots temporales, donde se indica para cada tipo la capacidad que tiene el sistema de generar esos bits de clave. La herramienta trabaja con slots de **3 y 30 minutos** de día y de noche que son capaces de generar distintas cantidades de volumen de clave en función del escenario en el que se encuentren. Utilizando un algoritmo de programación lineal se consiguió que la herramienta fuera capaz de generar de la manera más óptima ese volumen de clave, satisfaciendo la necesidad de cada terminal a través de una combinación de slots. Como la herramienta adquirió mejoras debido a su paso a grado de aplicación interactiva, esos requisitos aumentaron.

En este apartado se listan los **requisitos funcionales** que la aplicación debe ser capaz de realizar. Estos son:

- Debe permitir al usuario cargar un archivo de tipo Excel donde se encuentra la información a tratar, con un formato igual que el que nos proporciona Thales Alenia Space, si no es así la aplicación fallará para evitar errores futuros.

- Extraer la información necesaria de ese Excel y escribirlo en dos tablas que permiten modificar sus valores.
- Crear un archivo CSV con los resultados y la convergencia.
- Tener la posibilidad de ver en un mapa mundial las distintas ciudades que se están utilizando en el proceso, además de la localización de la estación terrena y del satélite.
- Cuando un terminal no esté activo ser capaz de acumular su necesidad de clave al día activo siguiente.
- Optimizar utilizando los datos visibles y modificables de las tablas, solicitar al usuario el número de terminales que se desean analizar y el tipo de slot, mostrar todas las ubicaciones que intervienen en la comunicación y cuando se muestren los resultados crear dos archivos Excel con las tablas de convergencia y de resultados.
- Mostrar los resultados a través de un gráfico que expresa los minutos de cada tipo de slot, el cumplimiento de los días objetivo y una tabla de resultados más completa.

La aplicación está diseñada para que reaccione mediante cuadros de texto informativos ante comportamiento erróneos por parte del usuario, como pulsar el botón de optimizar o guardar cambios en las tablas sin antes haber importado los datos del archivo Excel correspondiente o crear un archivo CSV sin haber optimizado previamente.

Figura 20: Cuadro informativo de error al optimizar

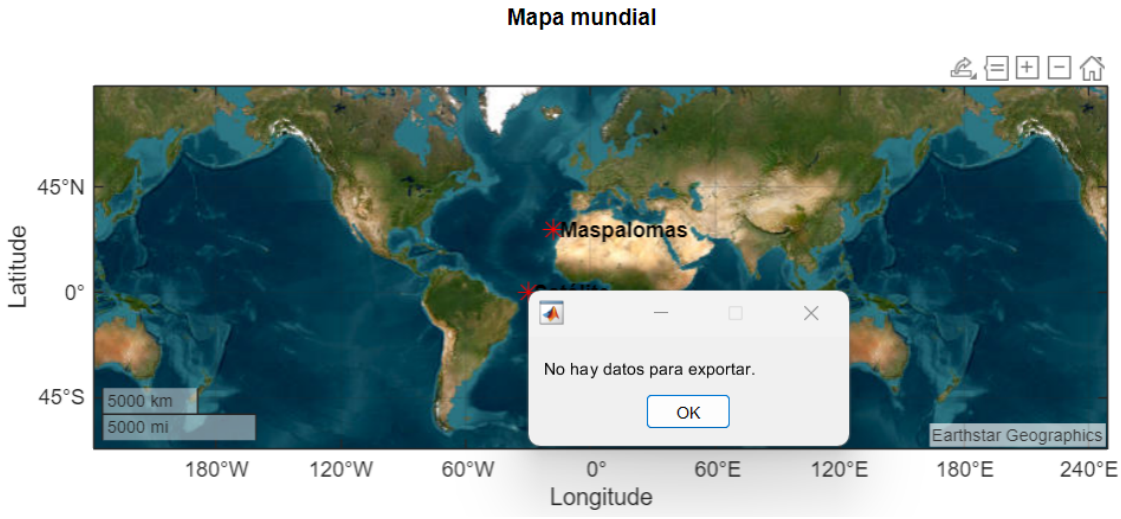


Figura 21: Cuadro informativo de error al crear archivo CSV

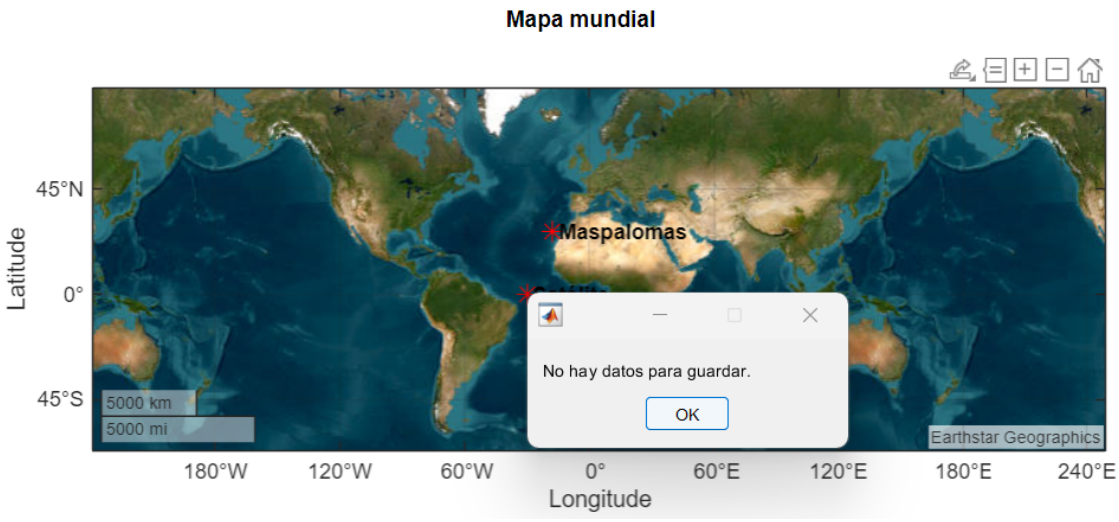


Figura 22: Cuadro informativo de error al guardar cambios

También tenemos unos **requisitos no funcionales**:

- Permitir modificar parámetros de los escenarios desde las tablas de la aplicación sin necesidad de hacerlo en el Excel.
- Documentar los cambios y explicar el funcionamiento de la aplicación después de esos cambios.

Entradas: Se trata del archivo Excel que elijamos en nuestro dispositivo que contenga la información necesaria para realizar el cálculo correctamente.

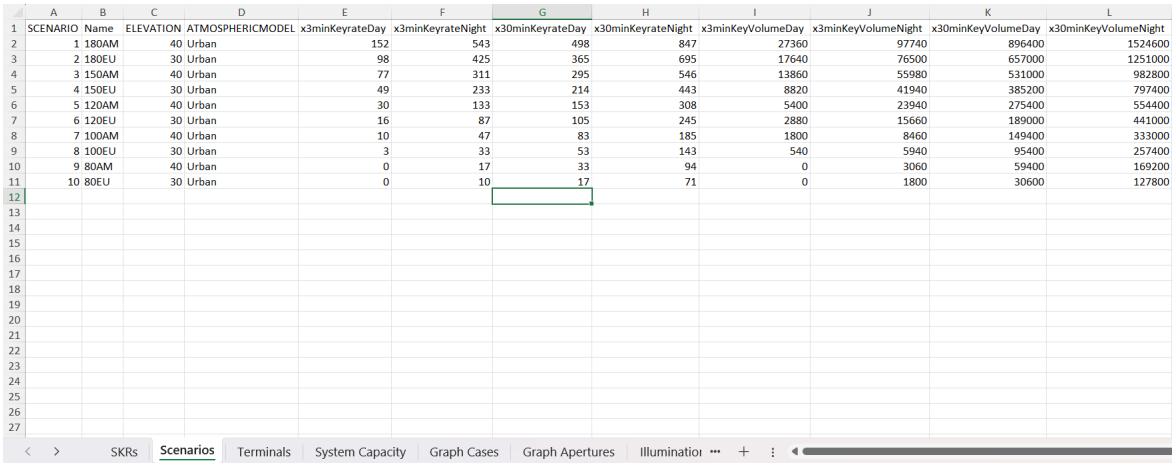
- Partimos de un archivo Excel llamado *User Capacity Calculations* en el cuál tenemos la información separada por hojas.
- En la primera hoja *SKR's* están las sensibilidades de cada tipo de antena según su apertura, elevación, situación del escenario, tiempo y horas del día. Se tratan de valores orientativos que no participan activamente en el cálculo de optimización.

	A	B	C	D	E	F
1	Aperture	Elevation	Scenario	Time	Day SKR	Night SKR
2	1,8	50	Best	30 mins	966	1354
3	1,8	50	Best	3 mins	480	943
4	1,8	50	Regular	30 mins	884	1266
5	1,8	50	Regular	3 mins	420	874
6	1,8	50	Worst	30 mins	602	961
7	1,8	50	Worst	3 mins	220	633
8	1,8	30	Best	30 mins	805	1182
9	1,8	30	Best	3 mins	362	807
10	1,8	30	Regular	30 mins	697	1065
11	1,8	30	Regular	3 mins	285	715
12	1,8	30	Worst	30 mins	366	695
13	1,8	30	Worst	3 mins	99	426
14	1,8	10	Best	30 mins	575	932
15	1,8	10	Best	3 mins	202	610
16	1,8	10	Regular	30 mins	358	687
17	1,8	10	Regular	3 mins	97	419
18	1,8	10	Worst	30 mins	23	183
19	1,8	10	Worst	3 mins	0	46
20	1,5	50	Best	30 mins	605	886
21	1,5	50	Best	3 mins	239	574
22	1,5	50	Regular	30 mins	550	828
23	1,5	50	Regular	3 mins	201	528
24	1,5	50	Worst	30 mins	365	625

SKRs | Scenarios | Terminals | System Capacity | Graph Cases

Figura 23: Hoja *SKR's*

- En la siguiente hoja *Scenarios* tenemos los posibles escenarios dependiendo de la elevación, el modelo atmosférico y los diferentes valores de tasa y volumen de bits de clave en función de la hora del día y la duración. En esta hoja los datos que se utilizan para el cálculo son los valores de volumen de clave, el resto de las columnas sirven para identificar cada escenario y poder vincular esta serie de datos con la tabla de la hoja *System Capacity*.



SCENARIO	Name	ELEVATION	ATMOSPHERICMODEL	x3minKeyrateDay	x3minKeyrateNight	x30minKeyrateDay	x30minKeyrateNight	x3minKeyVolumeDay	x3minKeyVolumeNight	x30minKeyVolumeDay	x30minKeyVolumeNight
1	180AM	40	Urban	152	543	498	847	27360	97740	896400	1524600
2	180EU	30	Urban	98	425	365	695	17640	76500	657000	1251000
3	150AM	40	Urban	77	311	295	546	13860	55980	531000	982800
4	150EU	30	Urban	49	233	214	443	8820	41940	385200	797400
5	120AM	40	Urban	30	133	153	308	5400	23940	275400	554400
6	120EU	30	Urban	16	87	105	245	2880	15660	189000	441000
7	100AM	40	Urban	10	47	83	185	1800	8460	149400	333000
8	100EU	30	Urban	3	33	53	143	540	5940	95400	257400
9	80AM	40	Urban	0	17	33	94	0	3060	59400	169200
10	80EU	30	Urban	0	10	17	71	0	1800	30600	127800

Figura 24: Hoja *Scenarios*

- En la hoja Terminals podemos ver los 7 terminales distintos de los que disponemos que varían según su tipo, número de usuarios, disponibilidad durante el día, número de bits de claves diarias por usuario y volumen total de bits de clave requerido. Esta tabla es un resumen de la cantidad de bits de clave que necesita cada tipo de escenario, a partir de la cual se crea la tabla *System Capacity.* Podemos apreciar que los tipos de terminal CO- son usados por 237 usuarios, por lo que el requerimiento de bits de clave es mucho mayor que en el resto de terminales, los cuales son usados por un único usuario.

Terminal	NAME	TYPE	Nº Users	Allows Daytime	Daily Keys/user	Requested Daily Volume
1	HE-SAN	High End	1	Yes	216	55296
2	HE-BK	High End	1	Yes	72	18432
3	HE-GOV	High End	1	Yes	30	7680
4	HE-ENE	High End	1	Yes	12	3072
5	CO-150EU	Commercial	237	Yes	6	364032
6	CO-180EU	Commercial	237	Yes	6	364032
7	CO-180AM	Commercial	237	Yes	6	364032

SKRs | Scenarios | Terminals | System Capacity | Graph Cases | Graph Apertures | Illuminatio

Figura 25: Hoja *Terminals*

- En System Capacity disponemos de 10 columnas donde están presentes el número y el tipo de cada terminal, la disponibilidad durante el día, el tipo de escenario, la ciudad con sus coordenadas, si el terminal está activo o no en cada día y el volumen total de bits de clave requerido. Esta tabla está vinculada a la de *scenarios*, ya que el volumen de clave de cada slot varía en función del escenario en el que nos encontremos.

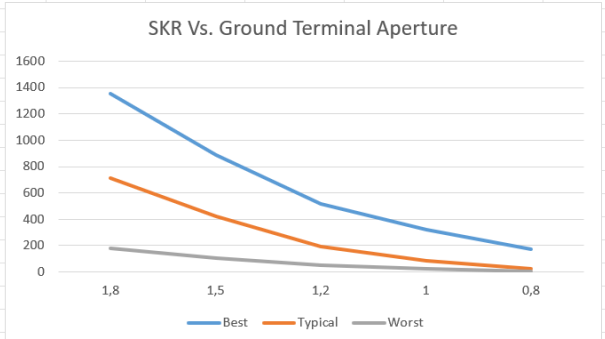
	A	B	C	D	E	F	G	H	I
1	Number	Type	Daytime	Scenario	Ciudad	Latitud	Longitud	TerminalActivo	RequestedDailyVolume
2	1	HE-SAN	Yes	80EU	Madrid	40,4165	-3,7026	1 0 1	55296
3	2	HE-SAN	Yes	80EU	Barcelona	41,3851	2,1734	1 1 1	55296
4	3	HE-SAN	Yes	80EU	Valencia	39,4699	-0,3763	0 0 1	55296
5	4	HE-SAN	Yes	80EU	Sevilla	37,3891	-5,9845	0 1 0	55296
6	1	HE-SAN	Yes	80AM	Ciudad de México	19,4326	-99,1332	1 0 1	55296
7	1	HE-SAN	Yes	100EU	Zaragoza	41,6488	-0,8891	0 1 1	55296
8	2	HE-SAN	Yes	100EU	Málaga	36,7213	-4,4214	0 0 0	55296
9	1	HE-SAN	Yes	100AM	Guadalajara	20,6597	-103,349	1 1 0	55296
10	1	HE-SAN	Yes	120EU	Murcia	37,9922	-1,1307	1 0 1	55296
11	2	HE-SAN	Yes	120EU	Palma	39,5696	2,6502	0 1 1	55296
12	1	HE-SAN	Yes	120AM	Monterrey	25,6866	-100,316	1 1 0	55296
13	1	HE-BK	Yes	80EU	Las Palmas	28,1235	-15,4363	0 0 1	18432
14	2	HE-BK	Yes	80EU	Bilbao	43,263	-2,935	1 1 1	18432
15	3	HE-BK	Yes	80EU	Alicante	38,3452	-0,4815	0 0 1	18432
16	4	HE-BK	Yes	80EU	Córdoba	37,8882	-4,7794	1 0 1	18432
17	5	HE-BK	Yes	80EU	Valladolid	41,6523	-4,7245	0 0 1	18432
18	6	HE-BK	Yes	80EU	Vigo	42,2406	-8,7207	1 0 0	18432
19	7	HE-BK	Yes	80EU	Gijón	43,5322	-5,6611	1 1 1	18432
20	8	HE-BK	Yes	80EU	Granada	37,1773	-3,5986	0 0 1	18432
21	1	HE-BK	Yes	80AM	Puebla	19,0414	-98,2063	1 0 1	18432
22	2	HE-BK	Yes	80AM	Buenos Aires	-34,604	-58,3816	0 1 0	18432
23	3	HE-BK	Yes	80AM	Córdoba	-31,42	-64,1888	1 1 1	18432
24	1	HE-BK	Yes	100EU	Oviedo	43,3603	-5,8448	0 0 1	18432

SKRs | Scenarios | Terminals | System Capacity | Graph Cases | Graph Apertures | Illuminatio

Figura 26: Hoja *System Capacity*

- En *Graph Cases* tenemos una gráfica que expresa la relación entre la apertura de los terminales y el SKR de cada uno.

		1,8	1,5	1,2	1	0,8
4	Best	1354	886	516	322	171
5	Typical	715	420	194	82	26
6	Worst	183	106	52	25	6



SKRs | Scenarios | Terminals | System Capacity | Graph Cases | Graph Apertures | Illuminatio

Figura 27: Hoja *Graph Cases*

- *Graph Apertures* enseña diferentes gráficas que relacionan las aperturas de los terminales con la posible elevación.

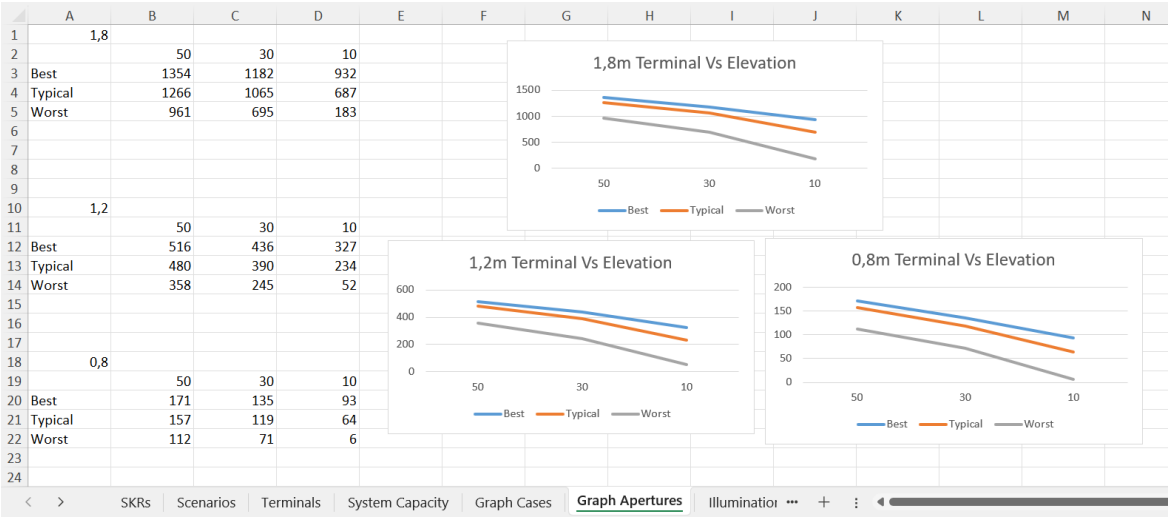
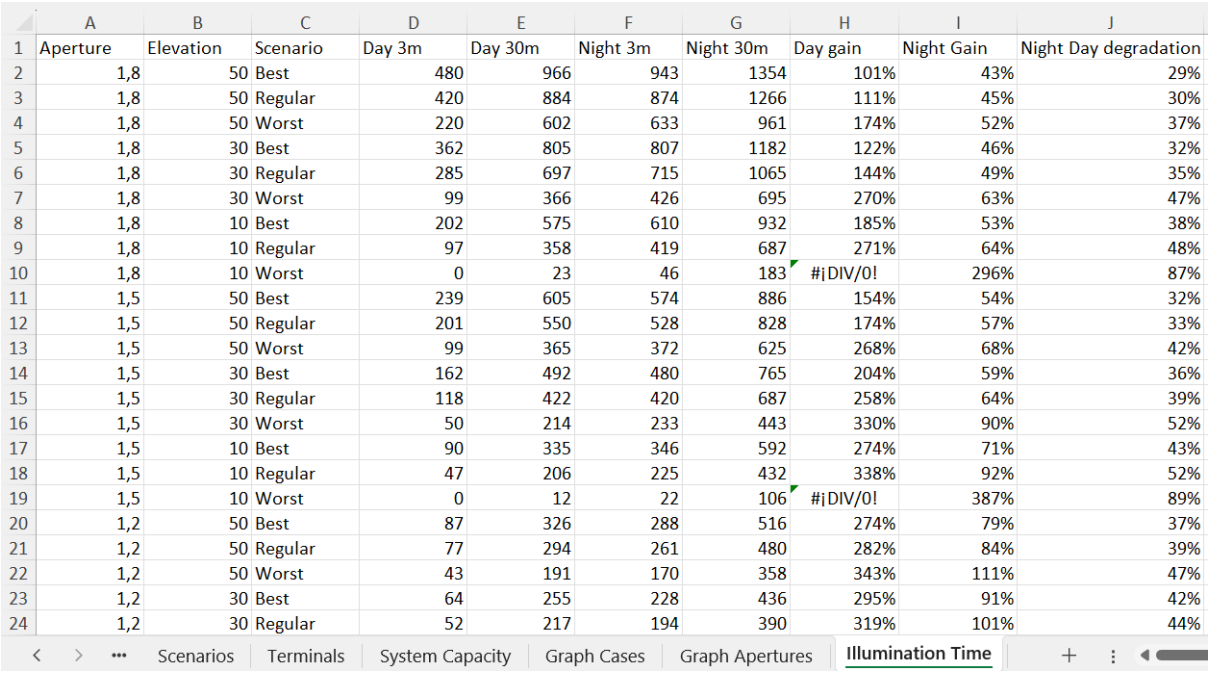


Figura 28: Hoja *Graph Apertures*

- Por último, tenemos la hoja de *Illumination Time*, que demuestra las distintas ganancias de volumen de clave en función de si la actividad se desarrolla de día o de noche para cada tipo de terminal en función de su apertura, elevación y tipo de escenario.



	A	B	C	D	E	F	G	H	I	J
1	Aperture	Elevation	Scenario	Day 3m	Day 30m	Night 3m	Night 30m	Day gain	Night Gain	Night Day degradation
2	1,8	50	Best	480	966	943	1354	101%	43%	29%
3	1,8	50	Regular	420	884	874	1266	111%	45%	30%
4	1,8	50	Worst	220	602	633	961	174%	52%	37%
5	1,8	30	Best	362	805	807	1182	122%	46%	32%
6	1,8	30	Regular	285	697	715	1065	144%	49%	35%
7	1,8	30	Worst	99	366	426	695	270%	63%	47%
8	1,8	10	Best	202	575	610	932	185%	53%	38%
9	1,8	10	Regular	97	358	419	687	271%	64%	48%
10	1,8	10	Worst	0	23	46	183	#¡DIV/0!	296%	87%
11	1,5	50	Best	239	605	574	886	154%	54%	32%
12	1,5	50	Regular	201	550	528	828	174%	57%	33%
13	1,5	50	Worst	99	365	372	625	268%	68%	42%
14	1,5	30	Best	162	492	480	765	204%	59%	36%
15	1,5	30	Regular	118	422	420	687	258%	64%	39%
16	1,5	30	Worst	50	214	233	443	330%	90%	52%
17	1,5	10	Best	90	335	346	592	274%	71%	43%
18	1,5	10	Regular	47	206	225	432	338%	92%	52%
19	1,5	10	Worst	0	12	22	106	#¡DIV/0!	387%	89%
20	1,2	50	Best	87	326	288	516	274%	79%	37%
21	1,2	50	Regular	77	294	261	480	282%	84%	39%
22	1,2	50	Worst	43	191	170	358	343%	111%	47%
23	1,2	30	Best	64	255	228	436	295%	91%	42%
24	1,2	30	Regular	52	217	194	390	319%	101%	44%

Figura 29: Hoja *Illumination Time*

Salidas: Son las diferentes acciones que debe realizar la aplicación con éxito para que el usuario pueda utilizarla

- Tras pulsar el botón 'Crear CSV' debe crearse un archivo llamado *User Capacity Calculations.csv* en la misma ruta donde está el archivo Excel.
- El mapa mundial debe estar actualizado en función de los terminales escogidos, la estación terrena y el satélite.
- Las tablas *User Capacity Calculations* y *Scenarios* tienen que cargar correctamente los datos obtenidos del archivo Excel.

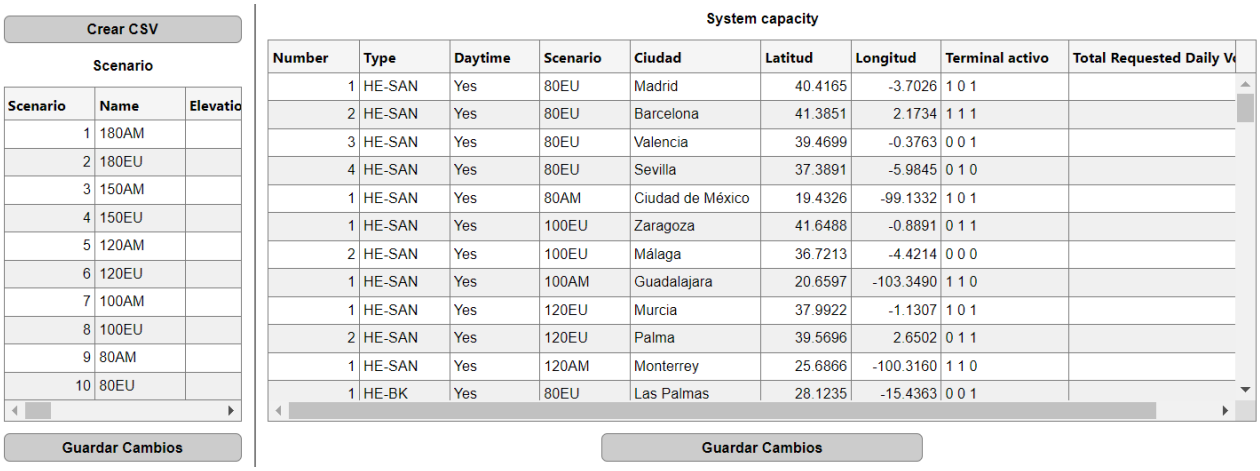


Figura 30: Información de Excel cargada en las tablas

- Tras pulsar el botón 'Optimización', indicar la cantidad de terminales y elegir el tipo de los slots, la aplicación debe mostrar un gráfico que muestre los minutos de cada tipo de slots con los porcentajes en función de la cantidad de cada slot.

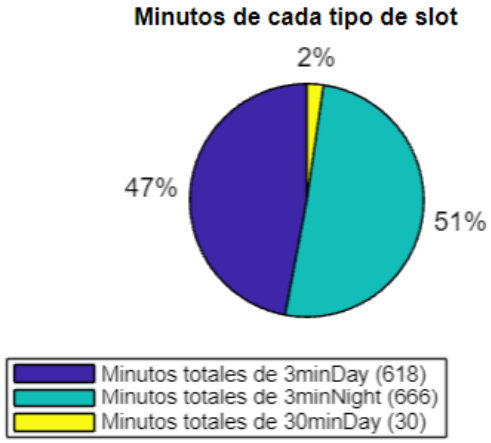


Figura 31: Resultados Slots

- También debe aparecer la tabla que muestre si se cumple el objetivo de días fijado en el Excel y, por último, una tabla de resultado más extensa.

Convergencia

Type	Scenario	Ciudad	Horas de día
HE-SAN	80EU	Madrid	00:00
HE-SAN	80EU	Barcelona	00:00
HE-SAN	80EU	Valencia	00:00
HE-SAN	80EU	Sevilla	00:00
HE-SAN	80AM	Ciudad de México	00:30

Resultados

Day	Type	Scenario	Ciudad	Termina
1	HE-SAN	80EU	Madrid	1
2	HE-SAN	80EU	Madrid	0
3	HE-SAN	80EU	Madrid	1
1	HE-SAN	80EU	Barcelona	1
2	HE-SAN	80EU	Barcelona	1

Figura 32: Resultados y convergencia

Esta tabla se corresponde con los resultados de convergencia que se exportan a un archivo Excel. En ella podemos apreciar el número de horas de día y noche de cada slot en cada escenario junto con el tiempo de ejecución del programa, los días objetivo y necesarios y la convergencia o no del escenario.

	A	B	C	D	E	F	G	H	I
1	Type	Scenario	Ciudad	Tiempo de día	Tiempo de noche	Tiempo de ejecución (seg)	Días Necesarios	Días Objetivo	Converge
2	HE-SAN	80EU	Madrid	00:00	04:39	4	2	3	Sí ✓
3	HE-SAN	80EU	Barcelona	00:00	07:45	4	2	3	Sí ✓
4	HE-SAN	80EU	Valencia	00:00	09:18	4	1	3	Sí ✓
5	HE-SAN	80AM	Ciudad de México	00:30	11:06	4	2	3	Sí ✓
6	HE-SAN	100EU	Zaragoza	10:48	11:06	4	2	3	Sí ✓

Figura 33: Tabla de resultados en Excel

En esta otra tabla apreciamos los resultados de los slots para cada escenario junto

con los volúmenes de clave requeridos y generados.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Day	Type	Scenario	Ciudad	ActiveTerminal	TotalRequestedDailyVolume	VolumenDay3min	VolumenNight3min	VolumenDay30min	VolumenNight30min	TotalKeyVolume	3Day	3Night	30Day	30Night
2	1	HE-SAN	80EU	Madrid	1	55296	0	1800	30600	127800	55800	0	31	0	0
3	2	HE-SAN	80EU	Madrid	0	0	0	1800	30600	127800	0	0	0	0	0
4	3	HE-SAN	80EU	Madrid	1	110592	0	1800	30600	127800	111600	0	62	0	0
5	1	HE-SAN	80EU	Barcelona	0	0	0	1800	30600	127800	0	0	0	0	0
6	2	HE-SAN	80EU	Barcelona	1	55296	0	1800	30600	127800	55800	0	31	0	0
7	3	HE-SAN	80EU	Barcelona	1	55296	0	1800	30600	127800	55800	0	31	0	0
8	1	HE-SAN	80EU	Valencia	0	0	0	1800	30600	127800	0	0	0	0	0
9	2	HE-SAN	80EU	Valencia	0	0	0	1800	30600	127800	0	0	0	0	0
10	3	HE-SAN	80EU	Valencia	1	55296	0	1800	30600	127800	55800	0	31	0	0

Figura 34: Tabla de resultados de slots en Excel

Esta serie de resultados en tablas por días forman parte de una primera planificación de la operación del sistema final.

4.4 Seguridad y rendimiento.

Esta aplicación maneja información sensible proporcionada por Thales Alenia Space contenida en el archivo Excel *User Capacity Calculations*, la cuál puede ser ligeramente modificada en esta memoria para garantizar su privacidad.

En cuanto al rendimiento la herramienta, permite al usuario realizar la optimización de cuantos terminales quiera, poniendo el límite en la capacidad de ejecución que tenga Matlab. Contando a partir de la pulsación del botón de optimizar, la herramienta es capaz de procesar 5 terminales en apenas 3 segundos, 10 terminales en unos 5 segundos o 100 terminales en 48 segundos aproximadamente. Esto permite al usuario analizar una gran cantidad de terminales en un muy poco tiempo. Respecto al resto de botones su funcionalidad es inmediata, mostrando un mensaje informativo para comprobar si la acción se ha realizado correctamente o no ha sido posible.

4.5 Arquitectura

El **flujo de ejecución** se trata de la secuencia de pasos que sigue un programa durante su ejecución, en nuestro caso la siguiente figura representa las diferentes opciones que nos da la herramienta desde la entrada de datos hasta la obtención de los resultados.

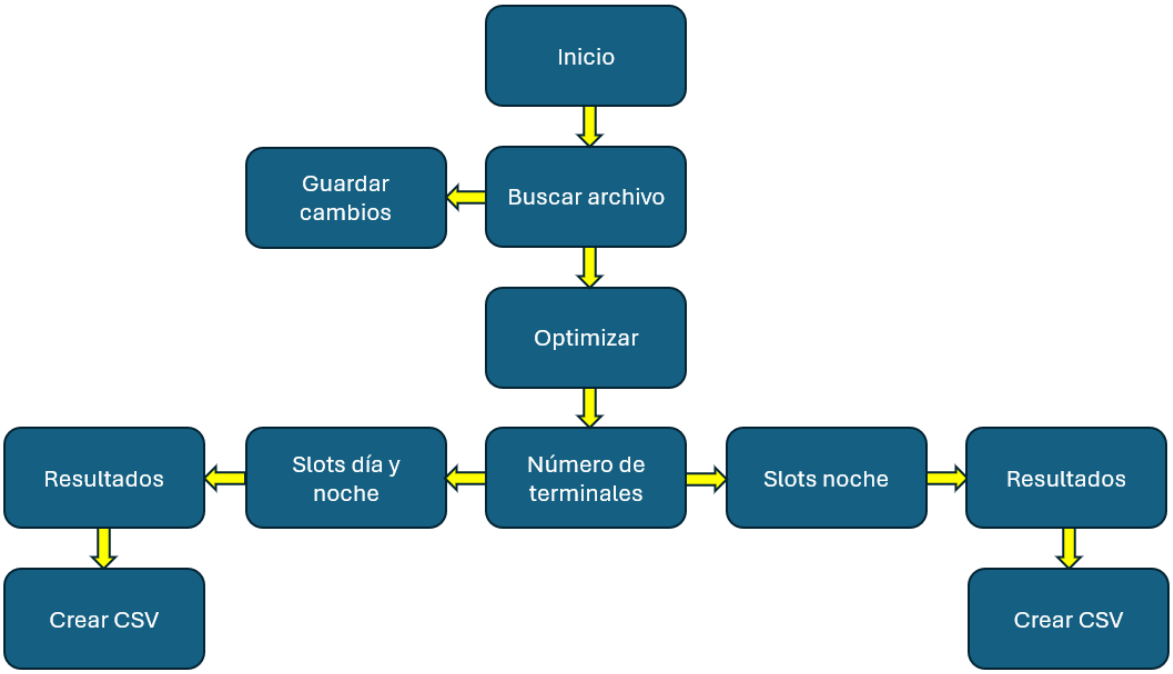


Figura 35: Flujo de ejecución

El software de esta herramienta está diseñado para calcular la cantidad de claves cuánticas necesarias diariamente en una comunicación satelital, y para ello integra módulos de cálculo, optimización y visualización que interactúan entre sí de manera secuencial.

A continuación, describiremos los diferentes módulos:

1. **Recogida de datos:** Se encarga de recolectar y validar la información ingresada por el usuario, a través de un archivo Excel. Para realizar esta acción nos apoyamos en la función *uigetfile*, la cual permite al usuario escoger un archivo cualquiera de su dispositivo, retornando un error si no se trata de tipo .xls o .xlsx. En cuanto a la lectura de esos datos se usa *readtable*, que almacena la tabla entera en una variable data. Estas acciones están recogidas dentro de la función *BuscararchivoButtonPushed*. También podemos incluir en este módulo a la función *StartupFcn*, la cual consigue que el mapa mundial exponga la ubicación del satélite y la estación terrena de la isla de La Palma cuando se inicia la aplicación. Esta función utiliza la función de Matlab *geoaxes* para representar el mapa mundial, la función *geoplot* para los asteriscos de las localizaciones, *geobasemap* para elegir el tipo de mapa y añadiendo la función *text*

a la figura del mapa conseguimos poner las etiquetas con los nombres a cada localización.

2. **Módulo de cálculo:** dentro de este módulo deberían estar los cálculos para saber el volumen de clave que necesita cada terminal dependiendo del escenario, pero esta serie de cálculos ya se realizaron dentro del archivo Excel proporcionado por parte de Thales, como se puede apreciar dentro de la hoja *System Capacity* en la columna *RequestedDailyVolume (bits)*.

3. **Módulo de optimización:** Antes de realizar el cálculo de optimización se permite al usuario modificar cualquier dato de las dos tablas representadas, pulsando después el botón de 'Guardar Cambios'. A través de la función *GuardarCambiosButtonPushed*, donde se habilita que los datos sean modificables y se guarda la tabla de nuevo en el archivo Excel original, podemos guardar los cambios en ambos sitios sin salir de la aplicación. Ahora sí, la parte de optimización está contenida dentro de la función *OptimizarButtonPushed*, que tiene como entrada los valores de volumen de clave de cada slot de cada escenario junto con su volumen requerido diariamente y calcula la combinación de slots más óptima para que el valor de clave supere el mínimo volumen de clave requerido, pero sin desperdiciar una gran cantidad de bits de claves.

   También nos ofrece la opción de elegir el número de terminales y si queremos utilizar los slots de día y de noche o solo de noche. Dentro de esta función podemos encontrar una gran variedad de funciones de Matlab, como por ejemplo *msgbox* para mostrar mensajes de error, *listdlg* para dar opciones de elección al usuario, y por supuesto la función *intlinprog*, para calcular la cantidad necesaria de cada slot y el valor de clave generado. Tras realizar la optimización tenemos el cálculo de la convergencia de cada terminal, calculando los días necesarios mediante la función de redondeo *ceil*, y el cálculo del tiempo de ejecución de la herramienta de optimización, usando las funciones de *floor* y *round*.

4. **Módulo de visualización:** en este módulo nos dedicamos a extraer los resultados y presentarlos al usuario a través de un gráfico y un par de tablas. Para la creación de las tablas utilizamos la función *writetable* para crear el archivo Excel con los resultados y posteriormente leemos esos archivos con la función *readtable* para mostrar esos resultados en las tablas presentes en la interfaz. Para el gráfico se utiliza la función *pie*, que crea un gráfico de forma

circular expresando los resultados de los slots en forma de porcentajes. Por último, tenemos la opción de exportar los resultados en dos archivos de tipo .csv a través de la función *CrearCSVButtonPushed*, que usa *writecell* en lugar de *writetable* debido al formato.

Este software se ha diseñado para que funcione **ininterrumpidamente**, es decir, para que el usuario pueda ejecutar las optimizaciones que requiera sin límite de uso. Es algo muy útil ya que permite ir cambiando valores en las tablas y comprobar los resultados al instante sin tener que volver a iniciar la aplicación de cero.

Esta arquitectura ha sido diseñada a partir de las necesidades de Thales Alenia Space y posteriormente aprobada por su equipo.

Capítulo 5: Implementación.

5.1 Herramientas utilizadas.

Este proyecto se ha realizado utilizando dos herramientas distintas, una para la aplicación (Matlab y Matlab App Designer) y otra para la redacción de la memoria (LaTeX). A continuación, paso a describir estas herramientas en profundidad con sus detalles técnicos.

Matlab y Matlab App Designer: en primer lugar, la aplicación comenzó siendo una archivo User_Capacity_Calculations.mlx de Matlab el cuál realizaba unas funciones parecidos a las que realiza la app final, pero con la diferencia de que en ese archivo los resultados estaban expresados a través del terminal de salida de Matlab, por lo que no eran muy visible cuando se trataba de una gran cantidad de resultados. App Designer nos permite crear apps a través de componentes visuales para crear el diseño de la interfaz gráfica de usuario **(GUI)** y de un editor para programar su comportamiento. Después de tener ese archivo listo el siguiente paso fue reutilizar ese código y añadirle las características de aplicación para crear la versión final operativa.

Para crear estos dos programas he utilizado la documentación de Matlab, disponible dentro de la aplicación, y la de **Mathworks**, su página web.

En el desarrollo del código he utilizado una serie de funciones predefinidas en Matlab, ubicadas dentro de las nuevas funciones diseñadas.

5.2 Funciones.

Estas son las funciones más importantes del software:

Uigetfile: nos permite abrir un cuadro de diálogo de selección de archivos, devolviendo el nombre del archivo y su ruta, en nuestro caso especificamos que tiene que ser de tipo .xlsx o .xls. Si el usuario hace clic en 'Cancelar' o en el botón de cerrar ventana ('X'), *uigetfile* devuelve 0 para ambos argumentos de salida.

Readtable: crea una tabla a partir de un archivo, en la variable data almacenamos la información de la hoja del archivo Excel seleccionado.

```
function BuscarArchivoButtonPushed(app, event)
    [app.archivo, app.ruta] = uigetfile({'*.xlsx';'*.xls'}, 'Seleccionar archivo Excel')

    % Comprobar si se ha seleccionado un archivo
    if isequal(app.archivo, 0)
        return;
    end

    % Ruta completa del archivo seleccionado
    archivoCompleto = fullfile(app.ruta, app.archivo);

    % Manejo de errores al leer las hojas del archivo Excel
    try
        % Cargar los datos de la hoja "System Capacity"
        data = readtable(archivoCompleto, 'Sheet', 'System Capacity');
        app.UITable.Data = data;
        app.UITable.ColumnEditable = true; % Permitir edición

        % Cargar los datos de la hoja "Scenarios"
        data1 = readtable(archivoCompleto, 'Sheet', 'Scenarios');
        app.UITable2.Data = data1;
        app.UITable2.ColumnEditable = true; % Permitir edición
    catch ME
        % Si ocurre un error con el archivo Excel
        msgbox(['Error al cargar los datos: ', ME.message], 'Error', 'error');
        return;
    end
end
```

Figura 36: Función *BuscararchivoButtonPushed* con *uigetfile*

Intlinprog: programación lineal de enteros mixtos (MILP), define un conjunto de límites inferiores y superiores de modo que la solución siempre se encuentra en el rango $lb \leq fval \leq up$. Esta función devuelve los slots necesarios tras la optimización y el valor final de volumen de clave generado. Como estamos buscando el primer valor que satisfaga el volumen de clave, las variables de volumen de clave de cada slot y el volumen requerido tienen que ser negativas, es decir, la matriz A y b.

```
        % Función para optimizar los slots
        function [slots, fval] = optimizacion(~, f1, f2, f3, f4, a1, a2, a3, a4, b)
            f  =  [f1 f2 f3 f4];
            A  =  [a1 a2 a3 a4];
            b = b(:);

            lb=zeros(size(f));
            ub=[];
            intcon=1:4;

            [slots, fval] = intlinprog(f, intcon, A, b, [], [], lb, ub);
        end
    end
```

Figura 37: Función *optimización* con *intlinprog*

Writetable: escribe una tabla en un archivo, con esta función somos capaces de crear o sobrescribir los datos de una tabla en el archivo Excel seleccionado, tras guardar los cambios hechos en las tablas de la interfaz.

```
        % Button pushed function: GuardarCambiosButton,
        % ...and 1 other component
        function GuardarCambiosButtonPushed(app, event)
            % Verificar si hay datos en la tabla antes de guardar
            if isempty(app.UITable.Data) || isempty(app.UITable2.Data)
                msgbox('No hay datos para guardar.');
                return;
            end

            % Obtener los datos modificados
            modifiedData = app.UITable.Data;
            modifiedData1 = app.UITable2.Data;

            % Guardar los datos en un nuevo archivo o sobrescribir el archivo original
            writetable(modifiedData, fullfile(app.ruta, app.archivo), 'Sheet', ...
                'System Capacity', 'WriteMode', 'overwrite');
            writetable(modifiedData1, fullfile(app.ruta, app.archivo), 'Sheet', ...
                'Scenarios', 'WriteMode', 'overwrite');

            msgbox('Cambios guardados', 'modal');
        end
```

Figura 38: Función *guardarCambios* con *writetable*

Geoplot: muestra los datos de latitud y longitud del mapa, tras cargar el paquete *'satellite'* podemos proyectar y mostrar los vectores de latitud y longitud en el mapa mundial a través de puntos rojos.

Text: añade descripciones de texto a puntos de datos, con la función *text* podemos dar nombre a las coordenadas anteriormente representadas por puntos rojos.

```
% Code that executes after component creation
function startupFcn(app)
    app.UIFigure.WindowState = 'maximized'; % Pantalla completa
    %Crear el mapa mundial ajustando su tamaño
    app.gx = geoaxes(app.MapPanel);
    limites = [-60 70];
    geolimits(app.gx, limites, limites);

    % Coordenadas de Maspalomas y Satélite
    latitudes = [28.7636, 0]; % Latitudes de Maspalomas y Satélite
    longitudes = [-17.8947, -30]; % Longitudes de Maspalomas y Satélite
    localizaciones = {'Maspalomas', 'Satélite'}; % Etiquetas de las localizaciones

    % Crear el mapa usando geoplot y establecer los puntos
    hold(app.gx, 'on');
    geoplot(app.gx, latitudes, longitudes, '*', 'Color', 'red', 'MarkerSize', 10);

    % Establecer el tipo de mapa
    geobasemap(app.gx, 'satellite');

    % Añadir etiquetas a los puntos
    for i = 1:length(latitudes)
        offset = 3; %Offset para que las etiquetas no se solapen
        text(app.gx, latitudes(i), longitudes(i) + offset, localizaciones{i}, 'FontSize', ...
            10, 'Color', 'black', 'FontWeight', 'bold');
    end

    hold(app.gx, 'off');

    app.UITable.ColumnEditable = true; % Permitir edición tabla System Capacity
    app.UITable2.ColumnEditable = true;% Permitir edición tabla Scenario
end
```

Figura 39: Función *startupFcn* con *geoplot* y *text*

Listdlg: crea un cuadro de diálogo para seleccionar un elemento de una lista, esta función despliega un menú en el que debemos elegir una de las dos opciones disponibles.

```
    % Selección de tipo de slots
    list = {'Slots durante día y noche', 'Slots durante noche'};
    [opcion, tf] = listdlg('PromptString', {'Elija una opción'}, 'SelectionMode','single' , ...
        'ListString', list, 'ListSize', [140 30]);

    %Si existe un error en la elección
    if tf == 0
        valid = false;
    end
end
```

Figura 40: Función *OptimizarButtonPushed* con *listdlg*

Pie: crea un gráfico circular de colores que representa los minutos de cada tipo de slot, eliminando los porcentajes de menos de un 0.1 %.

```
% Cálculo de los valores para gráfico
y = [3*total_Valor_3D, 3*total_Valor_3N, 30*total_Valor_30D, 30*total_Valor_30N];

% Nombres de cada parte del pie junto con el número de slots de cada tipo
x = {sprintf('Minutos totales de 3minDay (%s)', num2str(3*total_Valor_3D)), ...
    sprintf('Minutos totales de 3minNight (%s)', num2str(3*total_Valor_3N)), ...
    sprintf('Minutos totales de 30minDay (%s)', num2str(30*total_Valor_30D)), ...
    sprintf('Minutos totales de 30minNight (%s)', num2str(30*total_Valor_30N))};

% Calcular el porcentaje de cada sector
total = sum(y);
porcentajes = (y / total) * 100;

% Filtrado de sectores con porcentaje menor al 0.1%
indices = porcentajes >= 0.1;
y_filtrado = y(indices);
x_filtrado = x(indices);

%Reseteo del gráfico
delete(app.GraphicPanel.Children);

% Crear el diagrama de quesitos
ax = axes('Parent', app.GraphicPanel);
pie(ax, y_filtrado);

% Añadir leyenda
legend(ax, x_filtrado, 'Location', 'southoutside');
```

Figura 41: Función *OptimizarButtonPushed* con *pie*

Para la realización de esta memoria he utilizado la herramienta Latex. Esta herra-

mienta de composición de textos nos permite crear documentos, artículos académicos, tesis y libros técnicos a través de comandos del lenguaje TeX con una alta calidad tipográfica. En este proyecto se ha creado una estructura de carpetas que contienen archivos .tex, donde se encuentran los comandos que comienzan con \, y el texto y las imágenes referenciadas desde sus carpetas de imágenes en cada capítulo.

5.3 Manual de uso.

El uso de esta aplicación es muy intuitivo, por lo que en este manual de uso vamos a describir cómo se comporta durante una ejecución usual.

Inicio de la aplicación: para comenzar a utilizar la aplicación debemos *clickar* en el ejecutable User_Capacity_Calculations.exe. Se aconseja tener en una carpeta este ejecutable junto al resto de archivos debido a que de esta manera los futuros archivos de resultados se guardarán dentro de esa carpeta y se podrán localizar más fácilmente. Tras esto se abrirá la aplicación en pantalla completa y podremos ver su interfaz.

El primer paso es buscar el archivo Excel necesario, acción que se hará tras pulsar el botón 'Buscar archivo'. Tras esto se abrirá una ventana que nos permitirá escoger un archivo en cualquier localización. Después de elegir el archivo, veremos que las tablas situadas debajo del mapa mundial se rellenan de datos provenientes del archivo Excel escogido. Estas tablas nos permiten modificar cualquier parámetro sin necesidad de salir de la aplicación y poder guardar esos cambios en el Excel original. En este punto tenemos la opción de crear un archivo CSV llamado 'User Capacity Calculations_csv.csv' con la información contenida en la tabla System Capacity para su posterior uso en otras aplicaciones futuras.

El siguiente paso sería pulsar el botón de 'Optimizar' para que comience este proceso. Se mostrará una pequeña ventana que nos preguntará la cantidad de terminales y si queremos que la optimización de slots se realice teniendo en cuenta que están disponibles los slots durante el día y por la noche o únicamente por la noche. Realizamos nuestras elecciones y seguidamente veremos como en la parte derecha del lienzo, la gráfica y las tablas que estaban vacías ahora contienen los resultados de la optimización. En un segundo plano tras la optimización se crearán dos archivos Excel llamados resultados_slots.xlsx, resultados.xlsx, resultados_slots_noche.xlsx

o resultados_noche.xlsx en función de la elección de los slots.

Por último, la aplicación se mantiene en un estado de pausa, esperando a que el usuario decida si quiere realizar otras acciones como las anteriormente descritas, o si por el contrario ya no quiere seguir usando la aplicación y la cierra.

5.4 Integración en Suite de herramientas.

Esta herramienta se incorporará a un Suite llamado **COQPit** en el que integrara junto a otra serie de herramientas de Matlab. Este Suit está basado en tablas de tipo .csv, motivo por el cual hemos utilizado las tablas como entrada de datos y salida de resultados.

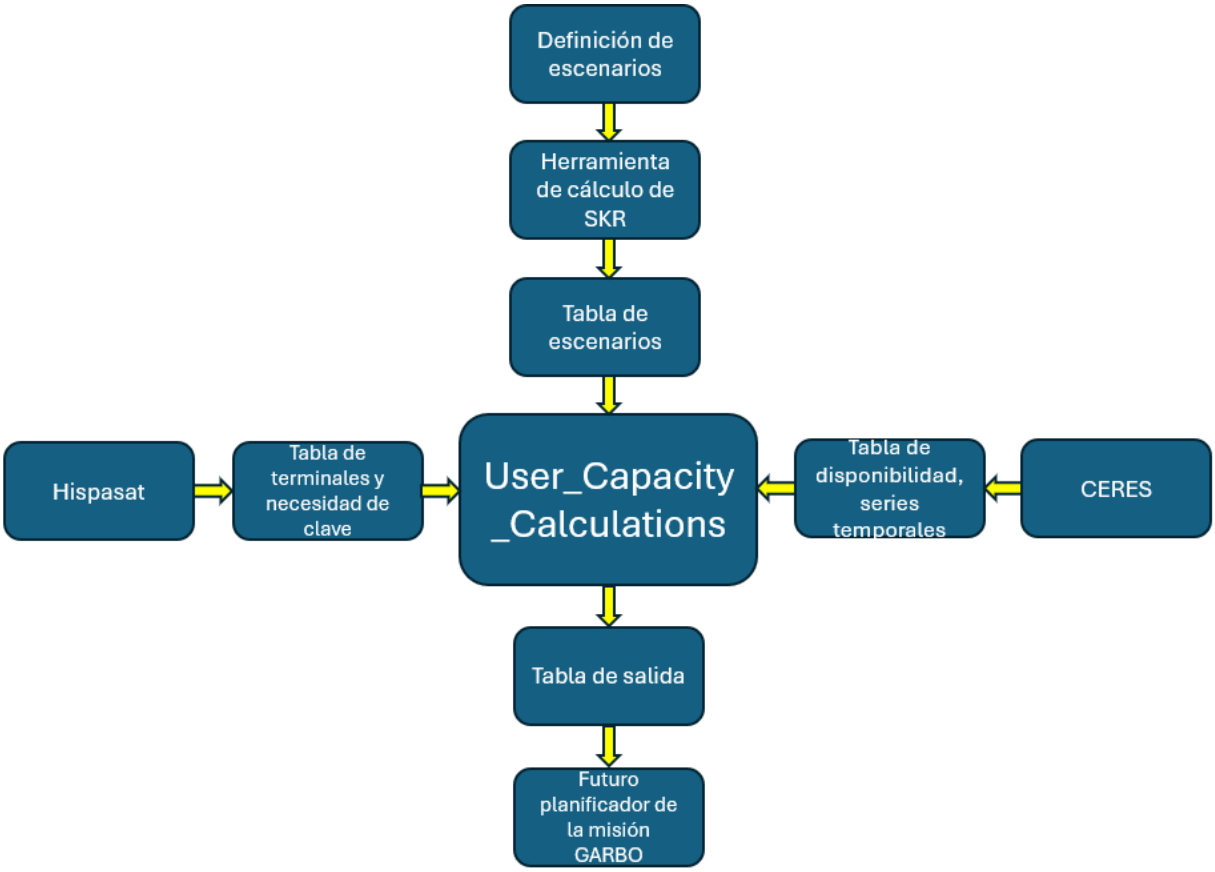


Figura 42: Diagrama de integración de User_Capacity_Calculations en COQPit

Este es el diagrama de bloques de COQPit donde las flechas representan las tablas a las que accederá cada una de las herramientas que lo integran.

5.5 Plan de pruebas.

El objetivo de este plan de pruebas es garantizar que todas las funcionalidades de la aplicación cumplen con los requisitos establecidos a través de una serie de pruebas. De este modo aseguramos que la aplicación es estable y confiable bajo diferentes condiciones de uso.

A continuación, se enumerarán una serie de pruebas que se realizaron:

- Prueba 1: pulsar el botón de 'Optimizar' o 'Guardar Cambios' en las tablas sin antes haber importado los datos del archivo Excel correspondiente.
- Prueba 2: importar un archivo Excel equivocado.
- Prueba 3: crear un archivo CSV sin haber optimizado.
- Prueba 4: comprobar que se introduce un número en la elección del número de terminales y que está entre 1 y la cantidad de escenarios que haya en la tabla.
- Prueba 5: el programa muestra de nuevo los mensajes que necesitan de la interacción del usuario si se cancelan.
- Prueba 6: estado de terminal es distinto de 0 o 1.
- Prueba 7: distinto número de días en un escenario que en el resto.
- Prueba 8: cálculo de optimización de un terminal con slots de día y noche.
- Prueba 9: cálculo de optimización de un terminal con slots de noche.
- Prueba 10: comprobación de no convergencia con un terminal.
- Prueba 11: comprobar convergencia con 20 escenarios.
- Prueba 12: comprobación de no convergencia con 20 escenarios.
- Prueba a plena carga.

5.6 Validación y pruebas.

Para la realización de esta serie de pruebas se han tomado múltiples valores que han sido modificados para comprobar la respuesta de la herramienta. Los resultados que se muestran se corresponden a unos datos de ejemplo.

Ahora vamos a describir en profundidad la serie de pruebas que comentamos en el anterior apartado:

- Prueba 1: iniciamos la herramienta y seguidamente pulsamos cualquiera de los dos botones de guardar cambios en las tablas sin haber importado la información. Se muestra un mensaje informativo de error pidiendo al usuario que primero busque y cargue un archivo. Prueba correcta.
- Prueba 2: cuando iniciamos la aplicación pulsamos el botón de buscar archivo y si ese archivo no se corresponde con el formato del Excel de referencia la aplicación muestra un mensaje de error ya que no encuentra las hojas necesarias. Prueba correcta.

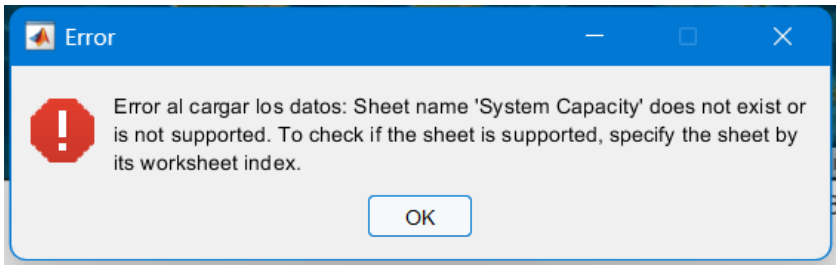


Figura 43: Error al cargar el archivo Excel

- Prueba 3: si pulsamos el botón de crear archivo csv antes de haber importado la información necesaria y sin haber realizado el cálculo de optimización la herramienta revela un mensaje indicando el error. Prueba correcta.
- Prueba 4: cuando procedemos a calcular la optimización se exige introducir una cantidad de terminales que tiene estar comprendida entre 1 y la cantidad de escenarios que haya en la tabla. Prueba correcta.

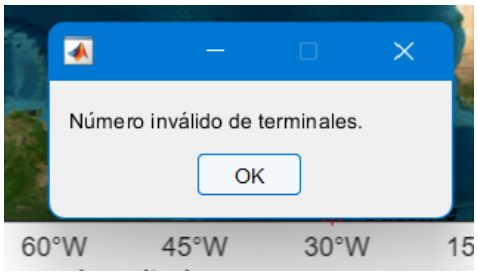


Figura 44: Error al introducir el número de terminales

- Prueba 5: si se cancela uno de los dos mensajes que necesitan la participación del usuario el programa procede a mostrar de nuevos esos mensajes hasta que la información introducida sea correcta. Prueba correcta.
- Prueba 6: el estado del terminal debe ser 0 o 1, en caso contrario la interfaz debe mostrar un mensaje expresando el error e indicando el terminal en el que ocurre. Prueba correcta.

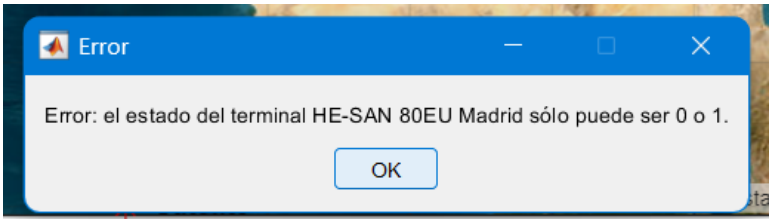


Figura 45: Error en el estado del terminal

- Prueba 7: distinto número de días en un escenario que en el resto. Si tenemos una serie de escenarios que se analizan durante 3 días por ejemplo y hay uno que en el campo terminal activo tiene más días, el programa debe para su ejecución cuando lo encuentre y mostrar que escenario tiene ese error. Prueba correcta.

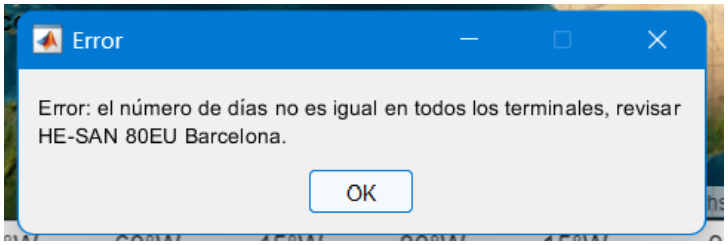


Figura 46: Error en el número de días de un terminal

- Prueba 8: cálculo de optimización de un terminal con slots de día y noche. Para esta primera prueba vamos a comprobar si al cálculo se realiza correctamente para un único terminal.

  En primer lugar, cargamos el Excel para comprobar que las tablas contienen la información necesaria, después pulsamos el botón de optimizar junto con la introducción de un '1' en la pregunta del número de terminales y con la elección de los slots de día y noche. Para este primer terminal vamos a utilizar

el escenario de HE-SAN 80AM en Ciudad de México que requiere un volumen de clave de 55296 bits de clave por día y tiene una configuración de terminales de 1 0 1, lo que quiere decir que el requerimiento del segundo día se tiene que sumar al del tercero.

Tras el cálculo podemos ver en los resultados que el escenario converge ya que según vemos en la tabla de convergencia los días objetivo son 3, pero los días que necesitamos son 2. En cuanto a la elección de slots podemos ver que mediante la combinación de 19 y 17 slots de 3 minutos de noche y 30 minutos de día la cantidad de volumen generado por día es de 58140, 0 y 111420 bits de clave. Si comparamos estos volúmenes con los requeridos podemos ver que el primer día se requieren 55296 y el programa genera 58140, por lo que cumple con el requerimiento de ese día. Si ahora nos vamos al segundo vemos que no existe volumen requerido porque el terminal no está activo, por lo que el volumen generado es 0. Por último, en el tercer día la cantidad necesaria es de 110592 bits debido a la suma del día anterior y lo que se genera son 111420 bits, por lo que el sistema converge ese día también.

Como podemos observar la cantidad de claves que se desperdician es muy pequeña, por lo que la herramienta funciona correctamente. En el gráfico de minutos de slots debe mostrar 108 minutos de slots de 3 minutos y 30 minutos de slots de 30 minutos. A parte de comprobar estos resultados también nos fijamos en el mapa mundial, que tiene que mostrar las coordenadas de Ciudad de México además de las del satélite y de la estación terrena. Prueba correcta.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Day	Type	Scenario	Ciudad	ActiveTerminal	TotalRequestedDailyVolume	VolumenDay3min	VolumenNight3min	VolumenDay30min	VolumenNight30min	TotalKeyVolume	3Day	3Night	30Day	30Night
2	1	HE-SAN	80AM	Ciudad de México	1	55296	0	3060	59400	169200	58140	0	19	0	0
3	2	HE-SAN	80AM	Ciudad de México	0	0	0	3060	59400	169200	0	0	0	0	0
4	3	HE-SAN	80AM	Ciudad de México	1	110592	0	3060	59400	169200	111420	0	17	1	0

Figura 47: Resultados de slots

	A	B	C	D	E	F	G	H	I
1	Type	Scenario	Ciudad	Tiempo de día	Tiempo de noche	Tiempo de ejecución (seg)	Días Necesarios	Días Objetivo	Converge
2	HE-SA	80AM	Ciudad	00:30	01:48	2	2	3	Sí ✓

Figura 48: Resultados de convergencia

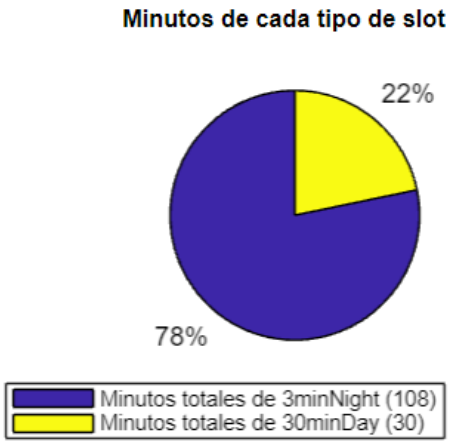


Figura 49: Gráfico de minutos

- Prueba 9: cálculo de optimización de un terminal con slots de noche. Ahora vamos a realizar la misma prueba que la anterior, pero eligiendo únicamente los slots de noche. Si nos vamos directamente a los resultados vemos que únicamente hay slots de noche y que para el primer día se requieren 19 slots de 3 minutos que generan 58140 bits y para el tercero 37 slots de 3 minutos de noche que generan 113220 bits de clave, por lo que también se cumplen los requerimientos diarios. Los minutos totales son 168 repartidos en los slots de 3 minutos. La cantidad de días necesarios sigue siendo 2 mientras que los días objetivos son 3, por lo que el sistema converge. Prueba correcta.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Day	Type	Scenario	Ciudad	ActiveTerminal	TotalRequestedDailyVolume	VolumenDay3min	VolumenNight3min	VolumenDay30min	VolumenNight30min	TotalKeyVolume	3Day	3Night	30Day	30Night
2	1	HE-SAN	80AM	Ciudad de México	1	55296	0	3060	0	169200	58140	0	19	0	0
3	2	HE-SAN	80AM	Ciudad de México	0	0	0	3060	0	169200	0	0	0	0	0
4	3	HE-SAN	80AM	Ciudad de México	1	110592	0	3060	0	169200	113220	0	37	0	0

Figura 50: Resultados de slots

	A	B	C	D	E	F	G	H	I
1	Type	Scenario	Ciudad	Tiempo de día	Tiempo de noche	Tiempo de ejecución (seg)	Días Necesarios	Días Objetivo	Converge
2	HE-SA	80AM	Ciudad	00:00	02:48	3	2	3	Sí ✓

Figura 51: Resultados de convergencia

Figura 52: Gráfico de minutos

- Prueba 10: comprobación de no convergencia con un escenario. Para esta prueba vamos a usar un escenario que tiene un terminal que requiere una gran cantidad de clave cada día. En un terminal con la configuración de disponibilidad de terminal 1 0 1 requiere un volumen de clave de 1155296. En los resultados observamos que se necesitarían 642 slots de 3 minutos de noche para crear 1155600 bits en el primer día y 1284 slots de 3 minutos de noche generando 2311200 bits, lo que conlleva que se necesitarían 9 días mientras que el objetivo es que sean 3. Prueba correcta.

Convergencia

(seg)	Días necesarios	Días objetivo	Converge
1	9	3	No ✗

Resultados

	TotalKeyVolume (bits)	3Day	3Night	30D
00	1155600	0	642	
00	0	0	0	
00	2311200	0	1284	

Figura 53: Resultados de no convergencia

- Prueba 11: comprobar convergencia con 20 escenarios. Para esta prueba vamos a utilizar datos reales proporcionados por Hispasat de 20 escenarios para ver que el programa es capaz de analizarlos todos correctamente. Tras analizar esta prueba cambiando distintos valores observamos que la salida es coherente con lo que se esperaba. Tras validar la herramienta con un terminal para comprobar su correcto funcionamiento vamos a realizar dos pruebas de carga

con datos reales para 20 terminales. Prueba correcta.

- Prueba 12: comprobación de no convergencia con 20 escenarios. Al igual que en la prueba anterior, los resultados concuerdan con lo previsto. Prueba correcta.
- Prueba a plena carga con 117 escenario: en esta última prueba vamos a utilizar datos "simulados"para 117 terminales, que es el tamaño de la red previsto para GARBO. Los resultados son los esperados tras una ejecución que ha durado un minuto aproximadamente en un dispositivo portátil que tiene un procesador AMD Ryzen 7 y 16 GB de memoria RAM. Prueba correcta.

Una vez completado el plan de pruebas se considera el software completo y listo para ser entregado al usuario final.

5.6.1 Entrega y validación por el usuario final

Una vez la herramienta tuvo la versión final deseada y se realizaron todas las pruebas necesarias para comprobar su funcionamiento en múltiples situaciones, se concertó una reunión con el equipo de Thales en su sede donde se expuso la herramienta para su revisión final. Allí se hizo una demostración en vivo para mostrar el funcionamiento y, tras una revisión, una serie de sugerencias y una validación de los resultados, la herramienta tuvo el visto bueno. Tras la demostración se procedió a una entrega formal tanto del código fuente como de un archivo ejecutable.

Capítulo 6: Lecciones aprendidas.

Durante la realización de este TFG se han adquirido una gran cantidad de conocimientos, tanto en la parte teórica del funcionamiento de un satélite con tecnología QKD, como en la parte práctica a la hora de programar con Matlab esta herramienta.

- Se comprendió cómo funciona la tecnología de distribución de claves cuánticas (QKD) en satélites y el potencial que tiene garantizando la seguridad en las comunicaciones de larga distancia mediante el uso de criptografía cuántica.
- En cuanto a los conocimientos de optimización, se estudiaron distintos tipos de algoritmos de optimización viendo sus beneficios y sus desventajas, hasta que se optó por la programación lineal.
- Se adquirieron conocimientos sobre Matlab para desarrollar algoritmos complejos y sobre Matlab App Designer para la creación de una aplicación con interfaz gráfica con la intención de facilitar la interacción con el usuario final.
- La interacción con el equipo de Thales Alenia Space España a través de reuniones fue clave para comprender sus necesidades, expectativas y sugerencias, lo que me permitió desarrollar la aplicación a su medida.
- Con el objetivo de asegurar una experiencia óptima al interactuar con la herramienta, se aprendió como se debía diseñar una interfaz de usuario intuitiva y eficiente, enfocada en la utilidad y accesibilidad.
- Mediante la demostración en vivo que realicé al equipo de Thales aprendí como se evalúa un proyecto, se verifican los requisitos y se hacen ajustes en función del *feedback*, mejorando así la calidad del producto final.
- Se adquirió experiencia en el uso de Latex a través de la redacción de esta memoria, una herramienta esencial para la creación de documentos técnicos y científicos, facilitando la redacción y el formato de textos complejos.

Capítulo 7: Próximos pasos.

Esta herramienta tiene muchas futuras mejoras al tratarse de la primera versión, la cual debe crecer para tener un uso comercial.

La primera mejora que podría implementarse se trata de la consideración de un **tiempo de apuntamiento** del satélite para el establecimiento de la conexión, haciendo totalmente real el caso. Esto obligará a añadir un tiempo inactivo entre cada uno de los terminales, lo que favorecería resultados que minimicen el número de sesiones de iluminación.

Otro futuro paso sería dar la posibilidad al usuario final de que escoja exactamente los terminales que quiera analizar en lugar de elegirlos en el orden en el que se encuentren en la tabla del archivo Excel.

También podríamos hacer estimaciones de clave por meses en lugar de días multiplicando el volumen requerido por 30, sin tener la necesidad de poner 30 números para el estado del terminal. Para este tipo de optimizaciones largas se podrían importar **series temporales** de una fuente externa, por ejemplo, las bases de datos de **CERES NASA** que registran los datos climáticos globales.

Como última mejora, los valores de terminal activo pueden pasar de ser binarios a ser unos **porcentajes de disponibilidad** que varíen en función de la climatología de cada localización, lo que haría variar la cantidad de volumen de clave que se podría proporcionar en cada día y así poder hacer una estimación más adecuada en cada caso.

Todas estas modificaciones necesitarían información que debe ser proporcionada por uno de los operadores de satélite que participan en el proyecto GARBO, como Hispasat, por ejemplo.

Capítulo 8: Conclusiones.

Este TFG hemos estudiado como se puede diseñar una aplicación con Matlab que posteriormente se unirá a otra serie de herramientas, con el fin de crear un planificador de recursos para un proyecto de intercambio de claves cuánticas en un satélite geoestacionario.

En cuanto a la tecnología QKD, se ha demostrado el por qué este tipo de tecnología es capaz de garantizar la privacidad de las comunicaciones cuánticas de larga distancia, dando solución a un gran problema de seguridad y anticipándose a la presencia de ordenadores cuánticos en manos de espías.

El estudio del proyecto se ha creado mediante el desarrollo de un algoritmo implementado con Matlab App Designer, demostrando la importancia de adquirir conocimientos de este tipo de herramientas durante los estudios de grado. Para el cálculo de la aplicación se ha usado una gran cantidad de datos reales proporcionados por Hispasat, empresa que colabora con Thales Alenia Space España en el proyecto GARBO. Tras la realización de las pruebas necesarias, podemos concluir que la herramienta es capaz de calcular una cantidad de volumen de clave con la cual se desperdician muy pocos bits clave diarios en proporción a dicho volumen. A la hora de calcular esos volúmenes se ha priorizado minimizar el desperdicio de bits de clave sobre la búsqueda de una opción que generara un mayor volumen de clave en menos tiempo, aun cuando dicha opción implicara un menor tiempo de uso del terminal.

La experiencia de participar en un proyecto real me ha permitido conocer de primera mano cómo se gestionan los desafíos técnicos y organizativos, desde la captura de requisitos hasta la implementación y validación de soluciones, proporcionando una visión integral del proceso de desarrollo y una comprensión más profunda de las necesidades del usuario final.

Referencias

[1] CDTI. (2022). El CDTI adjudica el contrato de I+D para el primer proyecto español de misión geoestacionaria de distribución cuántica de claves. Disponible en: https://www.cdti.es/noticias/el-cdti-innovacion-adjudica-el-contrato-de-i d-para-el-primer-proyecto-espanol-de-mision

[2] Plan de Recuperación. (2022). Conoce el primer proyecto español de misión geoestacionaria de distribución cuántica de claves. Disponible en: https://plan derecuperacion.gob.es/noticias/Conoce-primer-proyecto-espanol-mision-geoes tacionaria-distribucion-cuantica-claves-perte-aeroespacial-prtr

[3] CDTI. (2022). Iniciativa Quantum Key Distribution. Disponible en: https:// www.cdti.es/qkd-iniciativa

[4] HISPASAT. (2022). Un grupo de empresas españolas lideradas por HISPASAT trabaja en la fase de viabilidad de CARAMUEL, la primera misión geoestacionaria de distribución cuántica de claves. Disponible en: https://www.hispasat .com/es/sala-de-prensa/notas-de-prensa/archivo-2022/449/un-grupo-de-emp resas-espanolas-lideradas-por-hispasat-trabaja-en-la-fase-de-viabilidad-de-car amuel-la-primera-mision-geoestacionaria-de-distribucion-cuantica-de-claves

[5] Red Seguridad. (2022). CARAMUEL: Primera misión satelital geoestacionaria de distribución de claves mediante comunicaciones cuánticas. Disponible en: https://www.redseguridad.com/especialidades-tic/tecnologias-disruptivas/car amuel-primera-mision-satelital-geoestacionaria-de-distribucion-de-claves-med iante-comunicaciones-cuanticas_20221017.html

[6] El Mundo. (2024). Crónica sobre la misión geoestacionaria CARAMUEL y la tecnología cuántica. Disponible en: https://amp.elmundo.es/cronica/2024/06 /13/6662f968e4d4d8223c8b457c.html

[7] MathWorks. (s.f.). Sitio oficial de MATLAB. Disponible en: https://es.mathw orks.com/?s_tid=gn_logo

[8] Wikipedia. (s.f.). Distribución de claves cuánticas. Disponible en: https://es.wikipedia.org/wiki/DistribuciÃşn_de_claves_cuÃąnticas

[9] Wikipedia. (s.f.). Protocolo BB84. Disponible en: https://es.wikipedia.org/wiki/Criptograf%C3%ADa_cu%C3%A1ntica#Protocolo_BB84

[10] NASA. (s.f.). CERES Tool. Disponible en: https://ceres-tool.larc.nasa.gov/ord-tool/jsp/SYN1degEd41Selection.jsp

[11] Bennett, C. H., & Brassard, G. (1984). BB84: Quantum cryptography. Disponible en: https://arxiv.org/abs/2003.06557

[12] Yin, J., et al. (2022). Micius quantum experiments in space. Disponible en: https://arxiv.org/abs/2208.10236

[13] ESA. (s.f.). Caramuel Mission Overview. Disponible en: https://www.esa.int/Safety_Security/Caramuel

[14] González, A., et al. (2022). CARAMUEL: The future of Space Quantum Key Distribution in GEO. IEEE Conference Publication. Disponible en: https://ieeexplore.ieee.org/document/xxxxxxx