Desarrollo de un prototipo de sistema de seguridad integrado usando IoT y detección de movimiento para reforzar la seguridad de viviendas ubicadas en áreas de expansión y rurales

Eudrey Didney Reyes Silva, Víctor Manuel Carrillo Jaimes

Trabajo de grado para optar el título de Magister en Gestión y Consultoría en TIC

Director

Elvis Humberto Galvis Serrano

Magister en Redes y Sistemas de Comunicaciones Móviles

Co-Director

Yuli Andrea Álvarez Pizarro

Magister en Ingeniería Eléctrica

Universidad Santo Tomás, Bucaramanga

División de Ingenierías y Arquitectura

Maestría en Gestión y Consultoría en TIC

2025

Dedicatoria

Dedicamos este trabajo de grado a nuestra familia, quienes han sido nuestro pilar y fuente de inspiración a lo largo de este camino.

A nuestros hijos Isabel y Victor, por su amor incondicional, paciencia y comprensión. Cada esfuerzo, cada sacrificio y cada día de estudio tienen como propósito ser un mejor ejemplo para ustedes y demostrarles que con perseverancia todo es posible.

A nosotros mismos, por el compromiso, la resiliencia y el trabajo en equipo que nos permitió superar cada desafío de esta etapa. Este logro es el reflejo de nuestra constancia y pasión por crecer juntos, tanto personal como profesionalmente.

Finalmente, a todas aquellas personas que, de una manera u otra, han sido parte de este proceso y nos han impulsado a seguir adelante.

Eudrey Didney Reyes Silva y Victor Manuel Carrillo Jaimes

Agradecimientos

Este trabajo de grado es el resultado de un proceso académico y personal que no habría sido posible sin el apoyo de muchas personas e instituciones. Expresamos nuestro más profundo agradecimiento a todos aquellos que, de una u otra manera, contribuyeron a la culminación de este proyecto.

En primer lugar, queremos agradecer a la Universidad Santo Tomás (Bucaramanga), y en especial a la División de Ingenierías y Arquitectura, por brindarnos los conocimientos y herramientas necesarias para desarrollarnos en el manejo de nuevas tecnologías y en la gestión de proyectos de tecnología de la información.

A nuestros directores de proyecto, Yuli Andrea Álvarez Pizarro y Elvis Humberto Galvis Serrano, por su guía, paciencia y valiosos consejos durante el desarrollo de este trabajo. Su dedicación y experiencia en metodologías ágiles fueron fundamentales para lograr este resultado.

A nuestra familia, especialmente a nuestros hijos Isabel y Victor, gracias por su infinita paciencia y comprensión durante nuestras largas horas de estudio. Su apoyo incondicional fue nuestra mayor motivación.

Finalmente, agradecemos la fortaleza y perseverancia que nos permitieron alcanzar esta meta, así como a todas las personas que, de una forma u otra, contribuyeron en este proceso.

Eudrey Didney Reyes Silva y Victor Manuel Carrillo Jaimes

Contenido

Lista de tablas

Lista de figuras

Resumen

Las viviendas ubicadas en zonas rurales y de expansión urbana en Colombia presentan altos niveles de vulnerabilidad debido a la limitada cobertura de sistemas de seguridad tradicionales. El objetivo de este trabajo de investigación es desarrollar un prototipo de sistema de seguridad integrado que utilice tecnologías IoT y detección de movimiento para reforzar la seguridad en este tipo de viviendas. Para su implementación ***s***e diseñó e implementó una arquitectura basada en dispositivos IoT, un DVR con capacidad de detección de personas, una Raspberry PI con librerías desarrolladas en Python y una aplicación móvil construida en Android Studio (versión Koala). La metodología de desarrollo empleada fue SCRUM, organizando el trabajo en épicas, tareas e iteraciones. Se utilizaron plataformas como Tuya para facilitar la integración y gestión de dispositivos inteligentes de diferentes marcas. El prototipo desarrollado permitió realizar monitoreo en tiempo real, detección de personas, envío de notificaciones mediante servicios *push* y control remoto desde una app móvil. El uso de funcionalidades ya incorporadas en los DVRs comerciales resultó ser más eficiente que el desarrollo desde cero de algoritmos de reconocimiento de imágenes. Se documentó un plan de pruebas que validó cada funcionalidad según criterios de aceptación definidos previamente. La solución desarrollada fué técnicamente viable, económicamente accesible para hogares de clase media, y escalable a contextos similares. Su implementación demuestra que mediante la integración de tecnologías existentes es posible mejorar la seguridad en viviendas con limitaciones en infraestructura tecnológica.

Palabras clave: IoT, seguridad residencial, zonas rurales, detección de movimiento, prototipo de sistema

Abstract

Homes located in rural and urban expansion areas in Colombia exhibit high levels of vulnerability due to the limited coverage of traditional security systems. The objective of this research project is to develop a prototype of an integrated security system that leverages IoT technologies and motion detection to enhance safety in these types of homes. For its implementation, an architecture was designed and deployed using IoT devices, a DVR with person detection capabilities, a Raspberry Pi running Python-based libraries, and a mobile application built with Android Studio (Koala version). The development methodology employed was SCRUM, structuring the work into epics, tasks, and iterations. Platforms such as Tuya were used to facilitate the integration and management of smart devices from different brands. The developed prototype enabled real-time monitoring, person detection, push notification delivery, and remote control through a mobile app. The use of built-in functionalities in commercial DVRs proved to be more efficient than developing image recognition algorithms from scratch. A testing plan was documented, validating each feature according to previously defined acceptance criteria. The resulting solution was technically feasible, economically accessible for middle-income households, and scalable to similar contexts. Its implementation demonstrates that by integrating existing technologies, it is possible to improve home security in areas with limited technological infrastructure.

Keywords: IoT, home security, rural areas, motion detection, system prototype

Glosario

Alexa: asistente virtual desarrollado por Amazon, integrado en dispositivos inteligentes como los altavoces Echo. Utiliza reconocimiento de voz para interactuar con los usuarios, responder preguntas, reproducir música, controlar dispositivos del hogar inteligente y proporcionar información en tiempo real.

Áreas de expansión urbana: zonas designadas para el crecimiento planificado de las ciudades, con el objetivo de acomodar el aumento de la población y la demanda de infraestructura. En Colombia, estas áreas se determinan a través de los Planes de Ordenamiento Territorial (POT), que buscan garantizar un desarrollo urbano sostenible y ordenado. Las áreas de expansión urbana deben contar con la infraestructura necesaria, como servicios públicos, vías de acceso y espacios verdes, y deben integrarse de manera armoniosa con el entorno natural y las áreas urbanas existentes (Departamento Nacional de Planeación, s.f.).

AWS IoT: conjunto de servicios y soluciones proporcionados por Amazon Web Services (AWS) para conectar, gestionar y analizar dispositivos del Internet de las Cosas (IoT). AWS IoT permite a los usuarios desarrollar aplicaciones inteligentes y escalables mediante la integración de tecnologías de inteligencia artificial (IA) y aprendizaje automático (ML). Los servicios incluyen AWS IoT Ccore, que facilita la conexión segura de dispositivos a la nube, y AWS IoT Greengrass, que permite la ejecución de aplicaciones IoT en el borde (Amazon Web Services, s.f.).

C++: lenguaje de programación de propósito general, conocido por su eficiencia y flexibilidad. Es una extensión del lenguaje C y soporta tanto programación procedimental como programación orientada a objetos. C++ se utiliza ampliamente en el desarrollo de sistemas operativos, software de aplicación, juegos, y sistemas embebidos debido a su capacidad para manejar tareas de bajo nivel y su rendimiento optimizado.

DRV (Digital Video Recorder): dispositivo electrónico que graba video en formato digital en un medio de almacenamiento, como un disco duro. Es comúnmente utilizado en sistemas de cámaras de seguridad para almacenar y gestionar las grabaciones de video.

Ethernet: tecnología de red de área local (LAN) que utiliza cables para conectar dispositivos en una red, permitiendo la transmisión de datos a alta velocidad. Es una de las tecnologías de red más comunes y se utiliza ampliamente en entornos domésticos y empresariales para proporcionar conexiones de red estables y seguras.

Firebase Cloud Messaging (FCM): servicio en la nube de Google que permite enviar notificaciones y mensajes a aplicaciones en Android, iOS y la web.

Google Cloud IoT: conjunto de servicios gestionados por Google Cloud que permiten conectar, gestionar y analizar dispositivos del Internet de las Cosas (IoT) a escala. Incluye Google Cloud IoT Core, que facilita la conexión segura y la gestión centralizada de dispositivos distribuidos globalmente, y se integra con otros servicios de Google Cloud para proporcionar análisis de datos en tiempo real y obtener insights operativos. Google Cloud IoT soporta diversos protocolos de comunicación y ofrece herramientas para la autenticación, autorización y gestión de dispositivos (Google Cloud, 2017).

IoT (Internet of Things): red de dispositivos físicos interconectados que utilizan sensores, software y otras tecnologías para conectarse e intercambiar datos a través de Internet. Estos dispositivos pueden incluir desde electrodomésticos inteligentes hasta sistemas de seguridad y vehículos, permitiendo una mayor automatización y control remoto.

ISAPI (Intelligent Security API): protocolo desarrollado por Hikvision que permite la comunicación y gestión de dispositivos de seguridad, como DVRs y cámaras IP, a través de interfaces estándar basadas en HTTP/HTTPS. ISAPI proporciona funcionalidades para configurar

dispositivos, acceder a transmisiones de video, y recibir notificaciones de eventos en tiempo real, facilitando la integración de sistemas de videovigilancia con aplicaciones externas.

LoRa (Long Range): tecnología de modulación de radio de largo alcance diseñada para la conectividad de dispositivos IoT (Internet de las Cosas) y redes M2M (Machine to Machine). LoRa permite la comunicación a larga distancia con un bajo consumo de energía, lo que la hace ideal para aplicaciones donde la eficiencia energética y el alcance son cruciales. Es utilizada en una variedad de aplicaciones, incluyendo monitoreo ambiental, gestión de recursos, y sistemas de seguridad. LoRa se complementa con LoRaWAN, un protocolo de red que gestiona la comunicación entre dispositivos y la infraestructura de red (Ferrer, s.f.)

Ngrok: es una herramienta y servicio en la nube que permite exponer servidores locales a internet mediante túneles seguros; Facilita exponer servicios locales sin necesidad de configurar redes o abrir puertos en el router.

Prototipo: versión preliminar de un sistema que combina programas, dispositivos y tecnologías, creada para probar y validar conceptos, funcionalidades y diseño antes de la producción final.

Python: lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su sintaxis clara y legible. Es ampliamente utilizado en desarrollo web, análisis de datos, inteligencia artificial, automatización y muchas otras áreas debido a su versatilidad y la gran cantidad de bibliotecas disponibles.

Raspberry PI: computadora de placa única (Printed Circuit Board PCB) de bajo costo y tamaño reducido, desarrollada por la Fundación Raspberry PI. Es utilizada en una amplia variedad de proyectos de electrónica y programación, incluyendo sistemas de automatización del hogar, servidores multimedia y dispositivos de seguridad inteligentes.

Sigfox: red de comunicación inalámbrica global de baja potencia y largo alcance (LPWAN) diseñada específicamente para dispositivos del Internet de las cosas (IoT). Su objetivo es conectar objetos cotidianos a internet de forma económica y eficiente, permitiendo la transmisión de pequeñas cantidades de datos a intervalos regulares. Sigfox es conocida por su bajo consumo de energía y su capacidad para operar en bandas de frecuencia libres de licencia, lo que la hace ideal para aplicaciones que requieren una conectividad fiable y de bajo costo (AlfaIOT, 2023).

TUYA: plataforma global de AIoT para gestionar la conectividad, almacenamiento y procesamiento de datos del sistema. Permite a fabricantes, marcas, OEM y minoristas desarrollar y gestionar dispositivos inteligentes, ofrece servicios en la nube, herramientas de desarrollo de hardware y aplicaciones, facilitando la creación de productos inteligentes y su integración en diversos ecosistemas.

Wi-Fi: tecnología de red inalámbrica que permite la conexión de dispositivos a Internet y entre sí sin necesidad de cables. Utiliza ondas de radio para transmitir datos y es comúnmente utilizada en hogares, oficinas y espacios públicos para proporcionar acceso a Internet. Wi-Fi es esencial en la implementación de sistemas IoT, ya que facilita la comunicación y el control remoto de dispositivos inteligentes.

1. Introducción

El hurto a viviendas en Santander, particularmente en el área metropolitana de Bucaramanga y en el municipio de Floridablanca, evidencian cifras que no se reducen en los últimos 3 años a pesar de los esfuerzos de la policía y las alcaldías municipales, con cifras promedio de más de 900 hurtos a residencias en Santander (según estadísticas oficiales de la Policía Nacional).

En este contexto y con de la creciente expansión urbana de las ciudades, específicamente en zonas alejadas del centro urbano y formadas por veredas con conjuntos que tienen seguridad privada y parcelas particulares; este factor de la inseguridad se ha convertido en una preocupación apremiante. En los últimos años, estas zonas han experimentado un aumento alarmante en los casos de robos y los delincuentes aprovechan la topografía de las zonas, para despojar a los habitantes de sus pertenencias irrumpiendo en las viviendas, perpetrando robos cuando los propietarios están en casa o fuera de ella. Con este antecedente, este proyecto de maestría propone desarrollar un prototipo de sistema de seguridad integrado utilizando tecnología de Internet de las cosas (IoT), que incluye elementos como sensores de detección de movimiento, cámaras de seguridad, iluminación, alarmas y otros esenciales, para mejorar la seguridad, tanto de las viviendas y parcelas de área de expansión urbana. De igual forma, detectará movimientos sospechosos, enviará alertas en tiempo real, permitirá la integración de cámaras y activará luces de seguridad según sea necesario.

En el ámbito tecnológico, los dispositivos inteligentes se han convertido en una parte integral de nuestra vida cotidiana. El Internet de las cosas (IoT) ha transformado cómo interactuamos con el mundo digital, permitiendo la comunicación y el intercambio de datos entre dispositivos de manera fluida y efectiva. Sin embargo, las soluciones de seguridad existentes aún

no satisfacen completamente las necesidades de los hogares santandereanos. El problema que identificamos radica en la falta de soluciones integrales asequibles en el mercado. Actualmente, las opciones disponibles a menudo implican una inversión considerable y no abarcan todas las necesidades de seguridad. Nuestro objetivo es reducir esta falencia, desarrollando un prototipo que sea accesible para las familias de clase media. En última instancia, este estudio no solo aspira a ofrecer un prototipo funcional que demuestre la viabilidad de esta solución, sino también a proporcionar una guía de referencia de seguridad y especificaciones técnicas. Se espera que este proyecto ofrezca una valiosa contribución al campo de la seguridad del hogar, proporcionando a las familias de clase media una opción asequible y completa para proteger sus hogares en una era digital en constante evolución.

1.1 Planteamiento del problema

El crecimiento de áreas de expansión urbana ha llevado a la ocupación de tierras antes deshabitadas, lo que ha incrementado la vulnerabilidad de estas áreas a eventos delictivos. La falta de patrullaje y la limitada presencia de sistemas de vigilancia, tanto públicos como privados, han dejado a los propietarios de parcelas en una posición de desventaja.

A continuación, se listan las diversas causas de la falta de seguridad en estas áreas

1. *Despoblación y escasa presencia policial.*

En áreas de expansión y zonas rurales, la densidad de población tiende a ser baja, lo que a menudo resulta en una presencia policial limitada. La distancia entre las parcelas y las estaciones de policía contribuye a una respuesta lenta a las emergencias, lo que permite a los delincuentes operar con relativa impunidad.

2. *Falta de infraestructura de vigilancia.*

Estas zonas suelen carecer de infraestructura de vigilancia, como cámaras de seguridad, sistemas de alarma y patrullas regulares. Esto facilita el acceso no autorizado y el aumento en los actos delictivos en la zona.

3. *Aislamiento geográfico.*

Muchas parcelas rurales se encuentran en áreas geográficamente aisladas, lo que dificulta la comunicación y la coordinación con las autoridades en caso de emergencias. Esta falta de conectividad puede retrasar la respuesta ante situaciones de peligro.

4. *Escasez de recursos económicos.*

Los propietarios de parcelas en zonas rurales a menudo carecen de los recursos económicos necesarios para invertir en sistemas de seguridad privados efectivos. (En actualidad acceder a este servicio es caro y se necesita de conectividad) Esto los deja vulnerables a la falta de protección contra actividades delictivas.

Las principales consecuencias por la falta de seguridad se listan a continuación.

1. *Robos y vandalismo.*

Una de las consecuencias más graves de la inseguridad son los robos y el vandalismo, especialmente en parcelas y propiedades rurales. Estos actos delictivos ocasionan pérdidas considerables de bienes materiales para los propietarios y representan una amenaza directa para la seguridad personal. Además de las pérdidas económicas, existe el riesgo latente de confrontaciones con delincuentes, lo que pone en peligro la integridad física y emocional de las personas afectadas.

2. *Impacto negativo de la inversión.*

La carencia de seguridad en áreas de expansión y zonas rurales constituye un fuerte disuasivo para la inversión y el desarrollo económico en estas regiones. Los propietarios pueden

mostrarse reticentes a invertir en nuevas parcelas o mejorar las existentes debido a los riesgos asociados con la inseguridad. Esta situación genera un obstáculo significativo para el crecimiento económico y el progreso en estas áreas, lo que podría impactar negativamente en su valor comercial al ser consideradas lugares inseguros para establecerse.

3. *Desconfianza y aislamiento social.*

La ausencia de seguridad puede conducir a un aumento en la desconfianza entre los miembros de la comunidad y a un mayor aislamiento social. Las personas pueden mostrarse renuentes a interactuar y cooperar con sus vecinos, lo que repercute negativamente en el sentido de comunidad en estas áreas.

4. *Dificultades legales y administrativas.*

Los propietarios de parcelas pueden enfrentar dificultades legales y administrativas al intentar abordar incidentes delictivos. La falta de pruebas sólidas y sistemas de seguridad efectivos puede dificultar que las autoridades persigan y enjuicien a los delincuentes.

La implementación de dispositivos inteligentes permite la creación de un sistema de información, que garantiza la seguridad de los habitantes y fomenta un entorno más seguro en estas áreas. Con este proyecto, se busca diseñar un sistema de vigilancia asequible para familias de clase media, que permita la instalación de cámaras en la propiedad, envíe alertas por alarmas activadas, detecte movimientos en áreas específicas, active luces en función de alertas, almacene y organice toda la información recopilada de manera accesible. También se contempla la posibilidad de contar con una alarma de pánico que pueda contactar directamente a las autoridades competentes en caso de emergencia.

1.2 Justificación

A continuación, se expone la relevancia del problema identificado y las motivaciones que dieron origen al desarrollo del presente proyecto. Además, se describen las principales contribuciones previstas, así como los beneficiarios y el impacto que este generará en los actores involucrados.

1.2.1 Justificación del Proyecto

La seguridad en las viviendas ubicadas en áreas de expansión y rurales es un tema de creciente importancia debido a la vulnerabilidad inherente a estos entornos. Estas zonas, a menudo caracterizadas por la despoblación, la escasa presencia policial, la falta de infraestructura de vigilancia y la limitación de recursos económicos, presentan desafíos únicos que requieren soluciones innovadoras y accesibles. El proyecto “Desarrollo de un prototipo de sistema de seguridad integrado usando IoT y detección de movimiento para reforzar la seguridad de viviendas ubicadas en áreas de expansión y rurales” es fundamental no solo desde un punto de vista conceptual y técnico, sino también por su relevancia social, económica y formativa.

1.2.2 Importancia Conceptual y Valor Científico-Técnico

La elección del tema se fundamenta en la necesidad de explorar y aplicar tecnologías emergentes, como el Internet de las Cosas (IoT), en un contexto de seguridad doméstica en áreas rurales. Desde una perspectiva científica y técnica, el proyecto aporta al campo de la seguridad mediante el diseño y la implementación de un sistema innovador que integra sensores de movimiento, cámaras de vigilancia y alertas automatizadas. Este enfoque permite no solo la recolección y análisis de datos en tiempo real, sino también la creación de sistemas de respuesta

inmediata que antes estaban fuera del alcance de las comunidades rurales. La investigación generará conocimiento sobre la integración y optimización de tecnologías IoT en contextos con limitaciones de conectividad e infraestructura, aportando significativamente al desarrollo de soluciones tecnológicas para entornos de baja densidad poblacional.

1.2.3 Importancia para la Formación de Recursos Humanos

Este proyecto también es relevante para la formación de recursos humanos en el campo de las TICs, especialmente en la gestión y consultoría tecnológica. Involucra la aplicación práctica de conceptos avanzados de seguridad, sistemas integrados y gestión de datos, ofreciendo una plataforma para que los estudiantes y profesionales desarrollen habilidades críticas en la implementación de soluciones IoT en escenarios reales. Esto fomenta la capacidad de innovación y adaptación en el uso de tecnologías emergentes, habilidades esenciales para el desarrollo profesional en un mundo cada vez más digitalizado.

1.2.4 Importancia Económica y Social

Económicamente, la implementación de sistemas de seguridad accesibles y eficientes puede reducir las pérdidas materiales por robos y aumentar la percepción de seguridad, lo que a su vez mejora la calidad de vida de los habitantes. Al disminuir los costos asociados con la inseguridad, se liberan recursos que pueden ser destinados a otros aspectos del desarrollo comunitario, como la educación o la salud. Socialmente, el proyecto contribuye a la cohesión de la comunidad al promover una vigilancia cooperativa y al fortalecer el sentido de seguridad colectiva, lo cual es especialmente relevante en zonas rurales con un fuerte sentido de pertenencia y apoyo mutuo.

1.2.5 Campos de Aplicación y Potencial Impacto

El sistema desarrollado puede aplicarse no solo en viviendas individuales, sino también en pequeñas comunidades, fincas y negocios locales que enfrentan los mismos desafíos de seguridad. Su adaptabilidad y bajo costo lo hacen ideal para regiones con escasa infraestructura tecnológica. A nivel científico y tecnológico, el proyecto anticipa un impacto significativo al demostrar cómo la integración de dispositivos IoT puede adaptarse a contextos rurales, impulsando nuevas líneas de investigación y desarrollos en seguridad, conectividad y sostenibilidad.

El impacto potencial de esta tesis en los campos científico y tecnológico reside en su capacidad para generar soluciones replicables y escalables, que pueden servir como modelos para futuros desarrollos en seguridad aplicada en áreas de difícil acceso. En el campo económico-social, la propuesta aborda una problemática real con una solución concreta, accesible y efectiva, mejorando no solo la seguridad física de las viviendas, sino también el bienestar psicológico de sus habitantes.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un prototipo de sistema de seguridad integrado usando IoT y detección de movimiento para reforzar la seguridad de viviendas ubicadas en zonas rurales y/o de expansión urbana. El prototipo se desarrollará en una vivienda ubicada en la zona de expansión urbana de la Vereda de Los Cauchos del municipio de Floridablanca (Santander). La vivienda está comprendida por un lote de 3.000 metros cuadrados, con una casa construida y la cual se encuentra expuesta a

los factores de inseguridad como son no tener una vigilancia urbana tradicional (portería, vigilante, rondas de seguridad, Centros de Atención Inmediata (CAI) de policía cercano, etc.) y en el sector hay aproximadamente 9 viviendas en similares circunstancias, una área extensa de Lotes (con su casa) distantes entre ellas y sin presencia de vigilancia, policía y/o entidades similares de seguridad urbana.

1.3.2 Objetivos específicos

- Diseñar un modelo de seguridad para viviendas ubicadas en áreas rurales y de expansión urbana, integrando tecnologías de detección de movimiento y ajustándose al entorno rural para reducir las alertas no válidas y optimizar los recursos limitados de Comunicación e Internet.
- Desarrollar un prototipo IoT que integre el sistema de vigilancia por cámaras con tecnologías de detección de movimiento, con el propósito de llevar a cabo pruebas de funcionamiento y fiabilidad en un entorno controlado.
- Validar el funcionamiento del prototipo desarrollado como sistema de seguridad que permita dar una respuesta ante eventos identificados y reportados.

2. Marco referencial

A continuación, se detalla el marco conceptual, donde se establecen los términos clave y definiciones fundamentales; el marco teórico, que revisa la literatura relevante y las teorías existentes; el marco legal, que examina las leyes y regulaciones pertinentes; y el estado del arte referencial, donde se analizan los avances más recientes en la investigación y la práctica relacionados con el tema del proyecto.

2.1 Marco teórico

2.1.1 Desarrollo de un prototipo de Sistema de Seguridad Integrado utilizando IoT

Para su desarrollo, en términos generales es necesario contar con los siguientes componentes para la identificación y selección de los requisitos del Hardware.

- Raspberry PI: Es el dispositivo compatible con capacidades de IoT para realizar la función del cerebro del sistema.
- Sensores de Movimiento: Uso de sensores de movimiento como PIR (infrarrojos pasivos), cámaras y/o módulos de ultrasonido (para detectar movimientos).
- Conectividad y Comunicación: Los dispositivos IoT deben tener capacidad de conexión a Internet, ya sea a través de Wi-Fi, Ethernet o un módulo de comunicación IoT como LoRa, Sigfox, etc.

  De igual forma, se debe ejecutar la configuración del Hardware.
- Conexión de Sensores: La conexión de los sensores de movimiento al dispositivo IoT según las especificaciones del hardware verificando su funcionamiento.
- Conexión a Internet: La configuración de la conexión a Internet en el dispositivo controlador para permitir la comunicación en línea.

  Posterior a esto se implementa el siguiente desarrollo.
- Programación del Dispositivo IoT: El desarrollo de un software en lenguajes de programación como Python o C++ para el dispositivo controlador IoT (Raspberry PI o Microcontrolador IoT). Las funcionalidades del software desarrollado pueden incluir la captura de datos del sensor de movimiento y la comunicación con servicios en la nube.

- Almacenamiento de Datos y Análisis: Incluye la definición del almacenamiento de los datos recolectados (como videos o imágenes) y el análisis para detectar movimientos sospechosos.

Se continua con el desarrollo del prototipo, ejecutando las acciones de integración de los servicios en la nube.

- Uso de Plataformas de IoT: Uso de servicios en la nube como la plataforma global de AIoT Tuya para gestionar la conectividad, almacenamiento y procesamiento de datos del sistema. Tuya permite a fabricantes, marcas, OEM y minoristas desarrollar y gestionar dispositivos inteligentes, ofrece servicios en la nube, herramientas de desarrollo de hardware y aplicaciones, facilitando la creación de productos inteligentes y su integración en diversos ecosistemas. Según Tuya (s.f.) la plataforma soporta una amplia gama de dispositivos y proporciona soluciones para el hogar inteligente, la automatización industrial y otros sectores.
- Implementación de Alertas y Acciones: Configuración de las alertas para notificar sobre detecciones de movimiento y definición de las acciones a seguir, como son enviar notificaciones por correo electrónico o mensajes a dispositivos móviles.

  Finalmente se ejecutan pruebas y validación.
- Pruebas del Sistema: Pruebas para asegurar que los sensores detectan correctamente los movimientos y que las alertas y/o acciones se comportan según lo esperado.
- Ajustes y Mejoras: Realización de ajustes según se evidencie en las pruebas y definición de mejoras adicionales para una mayor precisión y eficiencia del sistema.

2.1.2 Marco de trabajo SCRUM para el desarrollo del prototipo

Scrum es un marco de trabajo ágil ampliamente utilizado en proyectos de desarrollo de software para optimizar la gestión y entrega de productos de manera iterativa e incremental. Su utilidad radica en su enfoque colaborativo y adaptable, permitiendo a los equipos responder de manera efectiva a cambios en los requisitos y prioridades del cliente. Scrum se basa en la transparencia, la inspección y la adaptación continua, promoviendo la entrega de productos funcionales en ciclos cortos llamados sprints. Está implementado por roles y personas; entre los cuales se definen: El rol de Product Owner, el cual define las prioridades del producto, rol Scrum Master, el cual facilita el proceso, y el equipo de desarrollo autoorganizado.

Los procesos clave de Scrum incluyen la planificación del sprint, el desarrollo iterativo, la revisión del sprint y la retrospectiva. Estos elementos permiten a los equipos mantener una alta flexibilidad, enfocarse en la entrega de valor y mejorar constantemente sus prácticas de desarrollo [11].

En SCRUM la asignación de tiempos es una práctica muy importante, maneja los siguientes conceptos:

- Sprint: Es un periodo de tiempo (o llamada iteración de tiempo) con una duración variable normalmente de mínimo una semana y máximo seis semanas de duración, en este tiempo se desarrolla trabajo por parte del equipo, para la entrega de un producto o componente funcional y entregable (que puede ser validado) por el cliente.
- Daily: El Daily Meeting es una reunión de corta duración y periodicidad diaria (aproximadamente 15 minutos), en la cual el equipo se reúne para dar informe de la evolución del proyecto. Se deben responder preguntas tales como ¿Que termine ayer?, ¿Qué terminare hoy? y ¿Qué dificultades (si existen) se presentó actualmente?

A continuación, se describen las fases de SCRUM y los procesos en el desarrollo del proyecto.

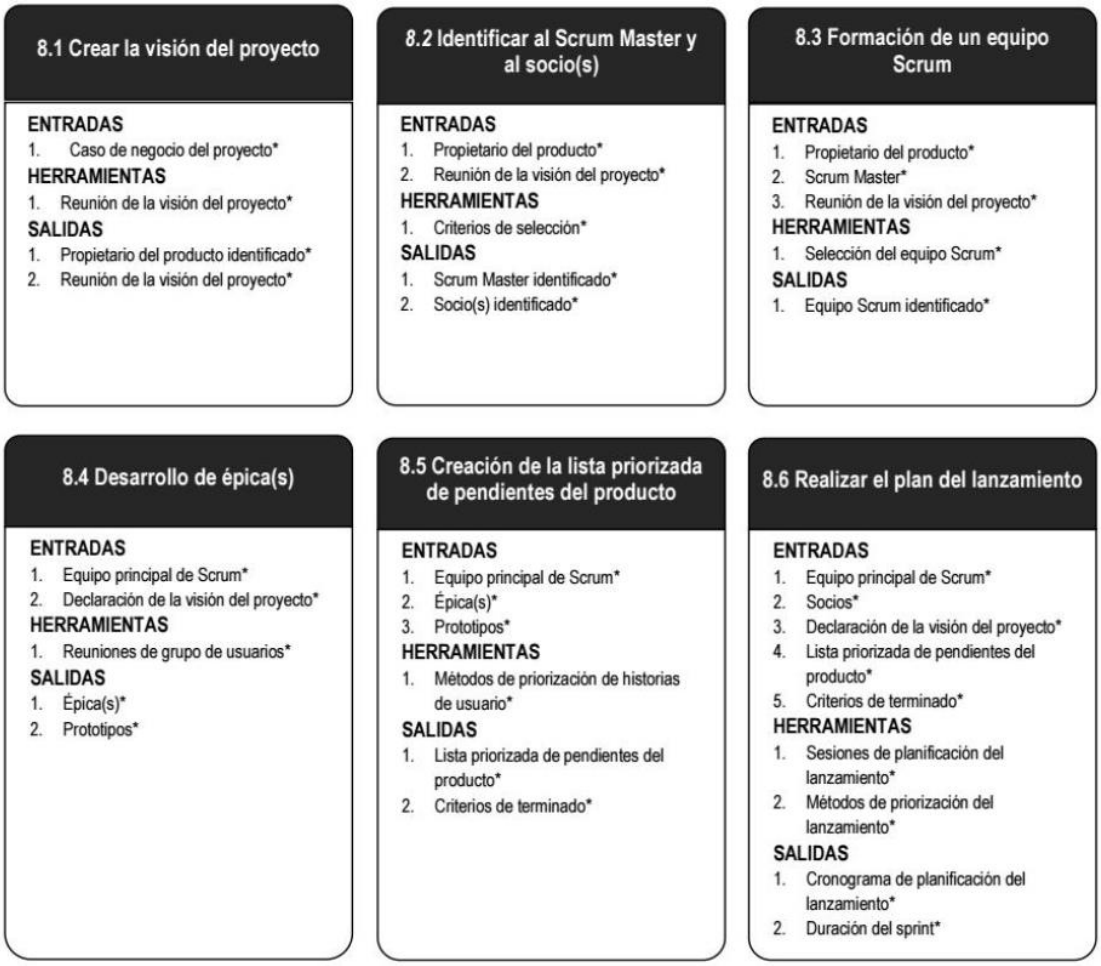
Figura 1. *Fases y Procesos del marco de trabajo SCRUM.*

Capítulo	Fase	Procesos
8	*Iniciar (Initiate)*	1. *Crear la visión del proyecto (Create Project Vision)* 2. *Identificar al Scrum Master y al socio(s) (Identify Scrum Master and Stakeholder(s))* 3. *Formación de un equipo Scrum (Form Equipo Scrum)* 4. *Desarrollo de épica(s) (Develop Epic(s))* 5. *Creación de la lista priorizada de pendientes del producto (Create Prioritized Product Backlog)* 6. *Realizar el plan de lanzamiento (Conduct Release Planning)*
9	*Planear y Estimar (Plan and Estimate)*	7. *Elaborar historias de usuario (Create User Stories)* 8. *Aprobar, estimar y asignar historias de usuarios (Approve, Estimate, and Commit User Stories)* 9. *Elaboración de tareas (Create Tasks)* 10. *Estimar tareas (Estimate Tasks)* 11. *Elaboración de la lista de pendientes del Sprint (Create Sprint Backlog)*
10	*Implementar (Implement)*	12. *Crear entregables (Create Deliverables)* 13. *Llevar a cabo el Standup diario (Conduct Daily Standup)* 14. *Mantenimiento de la lista priorizada de pendientes del producto (Groom Prioritized Product Backlog)*
11	*Revisión y Retrospectiva (Review and Retrospect)*	15. *Convocar Scrum de Scrums (Convene Scrum of Scrums)* 16. *Demostración y validación del Sprint (Demonstrate and Validate Sprint)* 17. *Retrospectiva de Sprint (Retrospect Sprint)*
12	*Lanzamiento (Release)*	18. *Envío de entregables (Ship Deliverables)* 19. *Retrospectiva del proyecto (Retrospect Project)*

Tomado de [6, p. 16]

2.1.2.1 Fase Inicial (Initiate). En esta fase, se define el alcance del proyecto y se forman los equipos. El Product Owner identifica las características y prioridades, creando así el Product Backlog. Los procesos obligatorios por realizar en esta fase son los que describen en la siguiente figura:

Figura 2. *Entradas obligatorias, herramientas y salidas para los procesos en la Fase Inicial.*

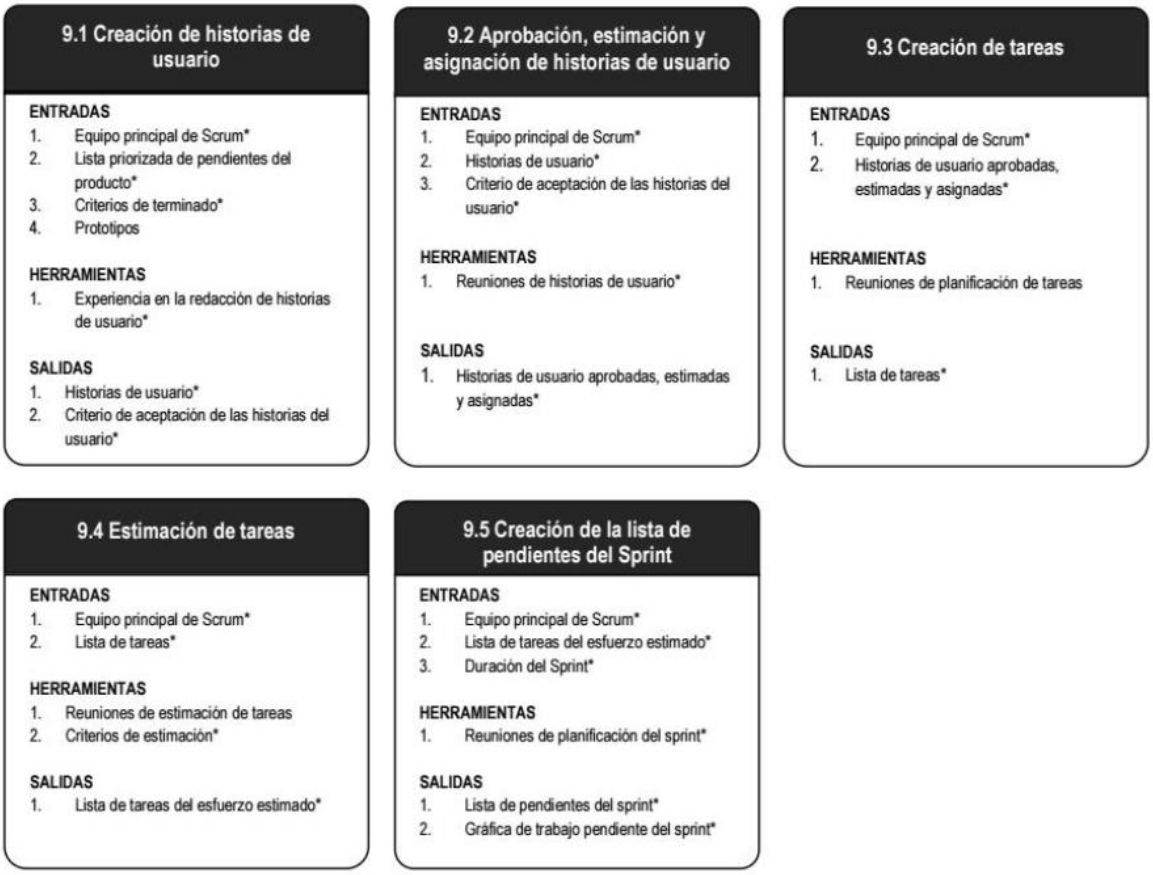


Tomado de [6, p. 145].

2.1.2.2 Fase Planificación y Estimación (Plan and Estimate). La fase de planificación y estimación se realiza al inicio de casa Sprint y su objetivo es definir (planificar) qué actividades o acciones (tareas) se desarrollarán en el Sprint y estimar el tiempo para completar dichas actividades o acciones. En resumen, se define que hacer y cuánto tiempo demora hacerlo, en términos del Sprint que se está desarrollando y tomando como base una porción del total de elementos del Backlog del Producto (lista centralizada de requisitos y deseos del cliente y meta del proyecto en desarrollo). Los elementos del Backlog del Producto seleccionados en esta fase se priorizan según

el valor para el cliente y el negocio. Los procesos obligatorios por realizar en esta fase son los que describen en la siguiente figura:

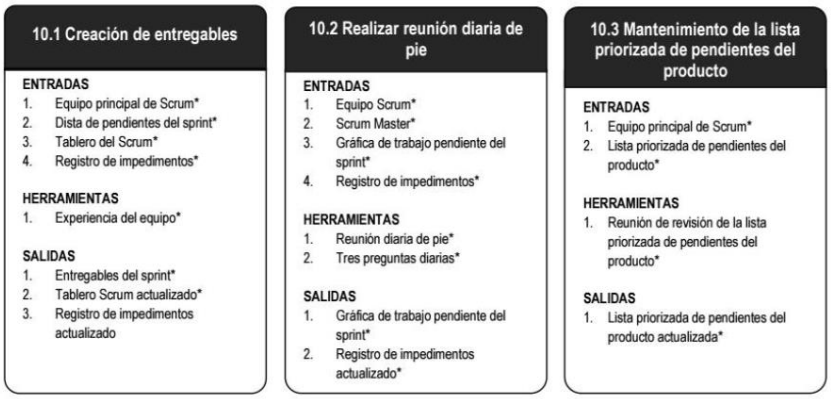
Figura 3. *Entradas obligatorias, las herramientas y las salidas de los procesos en la fase de Planificación y Estimación.*



Tomado de [6, p. 194].

2.1.2.3 Fase de Implementación (Implement). En esta fase se construyen o desarrollan actividades o acciones (tareas) para crear el producto del proyecto. Entre estas actividades se incluyen la realización de Dailys, seguimiento de avance y la creación de entregables. Los procesos obligatorios por realizar en esta fase son los que describen en la siguiente figura:

Figura 4. *Entradas, herramientas y salidas obligatorias para los procesos en la fase de Implementación.*



Tomado de [6, p. 225].

2.1.2.4 Fase de Revisión y Retrospectiva (Review and Retrospect). Es un evento realizado al final de cada Sprint y tiene como objetivo principal permitir mejorar el producto y todo el proceso de desarrollo. La ejecución de esta fase permite al todo el equipo revisar y adaptar (de ser necesario) su proceso de desarrollo y al hacerlo de forma regular, permite una mejora continua a lo largo del tiempo. Los procesos obligatorios por realizar en esta fase son los que describen en la siguiente figura:

Figura 5. *Entradas, herramientas y salidas obligatorias para los procesos en la fase de Revisión y Retrospectiva.*

Tomado de [6, p. 249].

2.1.2.5 Fase de Lanzamiento (Release). En esta fase se lanza una versión finalizada y lista para el mercado o para los usuarios finales. Implica la entrega de un grupo de funcionalidades que representan un avance importante en el total del producto. Su desarrollo requiere planificación, coordinación y comunicación efectivas de todo el equipo de desarrollo. Los procesos obligatorios por realizar en esta fase son los que describen en la siguiente figura:

Figura 6. *Entradas obligatorias, herramientas y salidas para los procesos en fase de Lanzamiento.*

Tomado de [6, p. 271].

2.1.3 Protocolos y estándares tecnológicos aplicados al prototipo

Para el desarrollo e implementación del prototipo de Sistema de Seguridad Integrado, se utilizan diferentes tecnologías y protocolos (estándares) de comunicación para lograr una arquitectura funcional, eficiente y adaptable. Se detallan los fundamentos teóricos de los principales componentes tecnológicos implementados en los prototipos del proyecto.

2.1.3.1 Protocolo ISAPI (Intelligent Security API). El protocolo ISAPI (Interfaz de Programación de Aplicaciones de Seguridad Inteligente) es un protocolo propietario desarrollado por Hikvision, basado en HTTP/HTTPS, con el cual se permite la comunicación entre dispositivos de videovigilancia y sistemas de terceros. Se estructura como una *API RESTful*, permitiendo su integración a través de solicitudes HTTP estándar [1].

En el prototipo final desarrollado, ISAPI es utilizado para establecer la comunicación entre el DVR Hikvision (modelo iDS-7216HQHI-M2/S) y la Raspberry PI. Através de ISAPI el DVR transmite eventos previamente parametrizados, como una respuesta automática del sistema, entre los eventos que pueden definirse están:

- Detección de movimiento
- Cruce de línea
- Detección de intrusos o rostros

Entre las funcionalidades de ISAPI que se resaltan, se encuentran:

- Lectura de eventos en tiempo real mediante peticiones GET.
- Acceso a configuración de dispositivos de videovigilancia.
- Extracción de capturas de video o imagen.
- Control remoto de cámaras PTZ.

Estas funcionalidades permiten una integración eficiente entre sistemas diversos dentro de una arquitectura modular y segura, gracias principalmente a su implementación sobre protocolos estándar como HTTP/HTTPS y al soporte para autenticación mediante tokens o credenciales cifradas. Además, eliminan la necesidad de intervención física en dispositivos externos, ya que el sistema opera directamente aprovechando las capacidades integradas del DVR.

2.1.3.2 Estándar de compresión de video H.264. El códec H.264 (MPEG-4 AVC), conocido como MPEG-4 Part 10 o AVC (Advanced Video Coding), es uno de los estándares más utilizados a nivel mundial de compresión de video, especialmente en aplicaciones de videovigilancia e IoT. Su diseño permite una compresión de alta eficiencia, logrando reducir significativamente el tamaño de los archivos de video sin perder calidad perceptible [2].

En el prototipo inicial (Sprint 1), se hace uso del DVR Dahua DHI-HCVR5216AN-NT, que envía la señal de video por HDMI a la Raspberry PI a través del convertidor HDMI a CSI-2. Esta señal se comprime en formato H.264. La Raspberry PI, con el lenguaje de programación Python utilizan la librería OpenCV para decodificar el video, separarlo en frames y procesarlo mediante algoritmos de detección de movimiento y reconocimiento de personas usando MediaPipe y TensorFlow Lite.

2.1.3.3 Firebase Cloud Messaging (FCM). Servicio de mensajería en la nube de Google que permite enviar notificaciones *push* a aplicaciones móviles [3].

FCM puede ser utilizado para:

- Enviar notificaciones *push* a *apps* móviles (Android/iOS).
- Notificar cambios o eventos desde un *backend* (por ejemplo, una Raspberry PI, servidor Node.js, etc.).
- Enviar mensajes silenciosos para sincronizar datos en segundo plano.
- Dirigir campañas de marketing mediante notificaciones personalizadas.
- Comunicar alertas urgentes (por ejemplo, seguridad, errores, eventos IoT, etc.).

FCM permite los siguientes tipos de mensajes:

- Notificación: Mensajes que se muestra automáticamente en la bandeja de notificaciones del sistema, no requieren la app en primer plano y pueden contener título, cuerpo, icono, entre otros.
- Mensaje de datos: Mensajes que no se muestra automáticamente, la app lo recibe y el usuario decide su visualización, útiles para mensajes sin notificación visual.
- Mensaje combinado: Mensaje de notificación y datos, permiten mostrar una notificación y también procesar datos en la app.

En el prototipo final desarrollado, se utiliza para enviar alertas al usuario cuando se detecta un evento de seguridad. Su integración con el lenguaje de programación Python y el entorno de desarrollo integrado Android Studio, permite una comunicación fluida entre la Raspberry PI y la aplicación móvil.

2.1.3.4 Herramienta NGROK. Es una herramienta que permite exponer servicios locales a internet mediante túneles seguros *HTTPS*; es decir crea un túnel seguro (*HTTPS*/*HTTP*/*TCP*) entre tu dispositivo local y el mundo exterior (internet), sin necesidad de abrir puertos en el *router* (local) o usar una IP pública [4].

Ngrok hace uso de túneles cifrados con *TLS/SSL*, garantizando:

- Confidencialidad: Nadie puede ver lo que se transmite entre *Ngrok* y tu servicio.
- Autenticidad: *Ngrok* genera una URL única para cada túnel.
- Integridad: No se alteran los datos durante el tránsito.
- Versatilidad: Permite habilitar autenticación *HTTP* básica, tokens, inspección de tráfico, y restricciones de acceso si es requerido.

Es importante porque permite que los servicios *REST* desarrollados en Python y alojados en un dispositivo Raspberry PI sean accesibles remotamente desde una aplicación móvil. Esto es esencial para hacer uso de funcionalidades y programas desarrollados y en una red local, desde una aplicación móvil con acceso a internet.

2.1.3.5 Tuya Developer Platform para gestión IoT. Tuya es una plataforma global de AIoT (Artificial Intelligence of Things) que permite la integración, gestión y automatización de dispositivos inteligentes. En este proyecto, la Raspberry PI se comunica con Tuya mediante API para activar dispositivos IoT como tomas, luces o alarmas, de diferentes fabricantes y proveedores, según los eventos detectados y las condiciones definidas en la parametrización del sistema.

2.1.3.6 FastAPI como backend en Python. FastAPI es un framework para construir *APIs RESTful* con Python [5]. Se utiliza en un dispositivo Raspberry PI para desplegar servicios web que gestionan eventos, condiciones, escenarios y dispositivos. FastAPI permite una documentación automática, soporte para autenticación por token y manejo eficiente de peticiones asíncronas.

2.1.3.7 Raspberry PI como unidad de procesamiento embebido. La Raspberry PI es el centro operativo del sistema, permitiendo ejecutar múltiples procesos en paralelo: captura de eventos (ISAPI), envío de alertas (Firebase), gestión IoT (Tuya), exposición de servicios web (Ngrok) y procesamiento de imágenes (OpenCV + IA). Es un dispositivo de tamaño compacto y bajo consumo, y con alta compatibilidad con el lenguaje Python para el desarrollo de aplicaciones de alta complejidad.

2.2 Marco conceptual

2.2.1 Prototipo de un Sistema

Por definición general se considera a un prototipo como una versión inicial, parcial o simplificada del sistema completo que se desea desarrollar. Su propósito es comprobar los conceptos inicialmente identificados, validar las principales funcionalidades definidas, evaluar la viabilidad técnica y los resultados obtenidos de forma anticipada previo al desarrollo del sistema final [11]. Este dispositivo es limitado en las funcionalidades desarrolladas, ofrece solo una parte del total identificado para el sistema total. Se centra en validar la viabilidad técnica del diseño y en la correcta selección e interacción de los componentes seleccionados para el sistema. De igual forma, se construye de forma iterativa, permitiendo identificar e implementar ajustes, mejoras y cambios, producto de la retroalimentación de resultados. Adicionalmente es adaptable a la identificación de nuevos requisitos hallados en su proceso de desarrollo.

Es importante resaltar que un prototipo no es la versión final, por lo tanto presenta una menor complejidad y suele hacer uso de componentes más simples o menos costosos en comparación al sistema final. Finalmente, suele ser utilizado para mostrar el diseño conceptual y/o las potencialidades específicas del sistema (en su versión final) a los potenciales interesados [12].

2.2.2 Sistema de Seguridad Integrado

Es un sistema de seguridad que busca proporcionar una protección integral al unir diferentes componentes que trabajaran en conjunto para detectar, prevenir, responder y mitigar situaciones de peligro o intrusión. Estos sistemas pueden incluir una variedad de dispositivos,

como lo son sensores de movimiento, cámaras de vigilancia, cerraduras inteligentes, alarmas, sistemas de control de acceso (entre otros) [13].

En estos sistemas, los diferentes dispositivos y componentes del sistema están conectados entre sí para compartir información y trabajar de manera coordinada. Por lo anterior puede detectar eventos o anomalías mediante estos componentes (sensores o cámaras) y monitorear constantemente el entorno. Estos sistemas tienen una capacidad de respuesta, puede activar alarmas, notificaciones o acciones automáticas en respuesta a eventos detectados. En muchos casos, estos sistemas permiten el monitoreo y control remoto a través de dispositivos móviles o plataformas en línea. Finalmente, y en algunos casos la integración hace uso de Tecnología IoT (Internet de las cosas), incorporando la capacidad de conectar y controlar dispositivos a través de internet para una mayor automatización y gestión [14].

2.2.3 Zona de Expansión urbana

En el contexto legal de Colombia, una zona de expansión urbana identifica áreas específicas designadas para el crecimiento planificado y ordenado de las ciudades. Estas zonas son áreas periféricas o adyacentes a los límites de las áreas urbanas existentes y se destinan para su desarrollo futuro con infraestructura y servicios básicos limitados [15].

2.2.4 Relación entre los conceptos

A través de la construcción de un prototipo de un Sistema Integrado de Seguridad se ofrecerá una solución en un corto plazo de tiempo al problema de seguridad (situaciones de peligro e intrusión) en zonas periféricas y limítrofes a las zonas urbanas, donde se dispone de servicio de

energía eléctrica e internet; pero no se posee la infraestructura de seguridad (estaciones de policía, CAI, vigilancia privada, etc.) de las zonas urbanas.

2.3 Marco legal

2.3.1 Ley de Protección de Datos Personales (Ley 1581 de 2012)

En Colombia, la Ley 1581 de 2012 establece las disposiciones generales para la protección de datos personales. Las organizaciones que adoptan soluciones en la nube deben garantizar que el tratamiento de dicha información cumpla con esta normativa, especialmente si los datos se almacenan o procesan fuera del país [15].

2.3.2 Normativa de Videovigilancia

Estas normativas establecen requisitos y limitaciones para el uso de sistemas de cámaras de vigilancia en Colombia. La Resolución 1384 de 2000 y la Circular Única de la Superintendencia de Vigilancia y Seguridad Privada son documentos clave que regulan aspectos técnicos y legales del uso de la videovigilancia [16].

2.3.3 Normativas de Seguridad Privada

La Superintendencia de Vigilancia y Seguridad Privada (Supervigilancia) en Colombia regula y supervisa la prestación de servicios de vigilancia y seguridad privada. Cuando el prototipo de seguridad integrado requiere la contratación de dichos servicios, se deben cumplir los requisitos y procedimientos establecidos por la normativa vigente [17].

2.3.4 Legislación Penal y Derechos Humanos

Se incluyen las leyes penales y los derechos humanos relacionados con la privacidad, el respeto a la intimidad y la no discriminación, en caso de que el prototipo involucre la recolección y tratamiento de datos personales. Este marco legal proporciona una visión general de las principales normas y regulaciones vigentes en Colombia, especialmente en lo referente al uso de cámaras de vigilancia, dispositivos de monitoreo y otros elementos requeridos en el desarrollo del prototipo de Sistema de Seguridad Integrado. En particular, se consideran la Ley 1581 de 2012 [15], que establece disposiciones generales para la protección de datos personales, y la Resolución 1384 de 2000 [11], que fija los requisitos para el uso de cámaras de vigilancia. Adicionalmente, se tienen en cuenta los principios constitucionales de protección a la intimidad y no discriminación consagrados en los artículos 15 y 13 de la Constitución Política de Colombia, así como la normativa general vigente en el sector de la vigilancia privada [17].

3. Método

En el desarrollo del prototipo de sistema de seguridad, utilizamos el marco de trabajo Scrum, que se centra en la adaptabilidad y la mejora continua a través de iteraciones rápidas, conocidas como sprints. Esta metodología facilita la planificación, ejecución y revisión constante de cada avance, permitiendo ajustes en función de las necesidades específicas del proyecto.

A continuación, se describen las fases de SCRUM y las actividades realizadas en cada proceso para el desarrollo del proyecto.

3.1 Fase Inicial

En esta fase se realizaron las siguientes actividades:

3.1.1 Formación del Equipo

El equipo tiene 2 integrantes entre los cuales se asignan los diferentes cargos. El equipo Scrum se comprometió a seguir los principios y prácticas de Scrum y a trabajar de manera colaborativa para alcanzar los objetivos del proyecto.

Tabla 1. *Relación de Cargo, Responsable y Principales Actividades de desarrollo SCRUM.*

Cargo	Responsable	Principales Actividades
Product Owner	Víctor Manuel Carrillo Jaimes	• Definir y comunicar la visión del producto. • Gestionar y priorizar el Product Backlog. • Asegurar que el equipo tenga una comprensión clara de los elementos del Product Backlog. • Tomar decisiones sobre el alcance y la calidad del producto.
Scrum Máster	Eudrey Didney Reyes Silva	• Facilitar las ceremonias Scrum (Daily Standups, Sprint Planning, Sprint Review, Sprint Retrospective). • Eliminar impedimentos que afecten al equipo. • Promover y apoyar el uso de Scrum según la guía de Scrum. • Ayudar al equipo a mejorar continuamente sus procesos y prácticas.
Equipo de Desarrollo	Víctor Manuel Carrillo Jaimes Eudrey Didney Reyes Silva	• Desarrollar y entregar incrementos del producto que sean potencialmente entregables. • Autoorganizarse y colaborar para lograr los objetivos del Sprint. • Participar activamente en las ceremonias Scrum. • Comprometerse con los objetivos del Sprint.

3.1.2 Definición de los requisitos de Producto

En esta actividad, se identificaron los usuarios finales, los cuales en este caso seremos nosotros mismos, dado que el sistema de seguridad en desarrollo estará destinado a ser implementado en nuestra vivienda. A partir de esta identificación, se documentaron las historias de usuario, que servirán como base para la definición de los requerimientos del sistema.

3.1.3 Historias de Usuario

Las historias de Usuario SCRUM se muestran a continuación:

Tabla 2. *Historias de Usuario de desarrollo SCRUM.*

ID	¿Cómo? (Quien)	¿Quisiera? (Que)	¿Para?
H-001	Usuario	El sistema detecte cualquier movimiento en áreas específicas de la vivienda	Para ser alertado en caso de intrusión.
H-002	Usuario	El sistema identifique personas en el campo de visión de la cámara	Para recibir una alerta si alguien entra sin autorización
H-003	Usuario	Recibir una notificación en mi dispositivo móvil cuando se detecte movimiento o presencia de personas	Para mantenerme informado en tiempo real de posibles intrusos.
H-004	Usuario	Personalizar los tipos de notificación que recibiré para diferentes escenarios de seguridad (por ejemplo, cuando no estoy en casa o cuando es de noche)	Para recibir solo alertas relevantes en cada situación
H-005	Usuario	El Sistema pueda encender o apagar luces y otros dispositivos IoT de forma autónoma al identificar una posible violación de seguridad	Para disuadir posibles intrusos o simular presencia en casa cuando esté ausente.
H-006	Usuario	El sistema IoT esté integrado para poder controlarlo desde una sola aplicación o interfaz	Para gestionar todo el sistema de seguridad de manera más sencilla.
H-007	Usuario	Definir escenarios de seguridad (por ejemplo, modo "ausente" o "noche") que activen o desactiven dispositivos automáticamente	Para no tener que configurarlos manualmente cada vez.
H-008	Usuario	Que los dispositivos se ajusten automáticamente según el escenario de seguridad elegido	Para asegurarme de que el sistema se comporta de acuerdo con mis necesidades de protección en cada momento
H-009	Usuario	El sistema registre cada evento de detección de movimiento y reconocimiento de personas	Para revisar actividades pasadas y conocer patrones en la seguridad de mi vivienda.
H-010	Usuario	Consultar el historial de eventos de seguridad cuando lo necesite	Para poder hacer un seguimiento de posibles incidentes en la vivienda
H-011	Usuario	Tener la opción de agregar cámaras adicionales al sistema sin necesidad de cambiar la arquitectura y/o la programación del sistema.	Para expandir la cobertura de seguridad de la vivienda cuando lo requiera.
H-012	Usuario	Integrar nuevos dispositivos IoT en el sistema	Para ampliar su funcionalidad según mis necesidades de seguridad.
H-013	Usuario	Una interfaz de administración donde pueda configurar los dispositivos IoT y consultar las notificaciones	Para manejar todo el sistema desde un solo lugar.
H-014	Usuario	Una interfaz intuitiva y fácil de usar	Para configurar y controlar el sistema sin necesidad de conocimientos técnicos avanzados
H-015	Usuario	El sistema permita agregar nuevos dispositivos IoT en el futuro sin afectar su rendimiento	para ampliar la cobertura de seguridad de mi hogar cuando lo necesite.

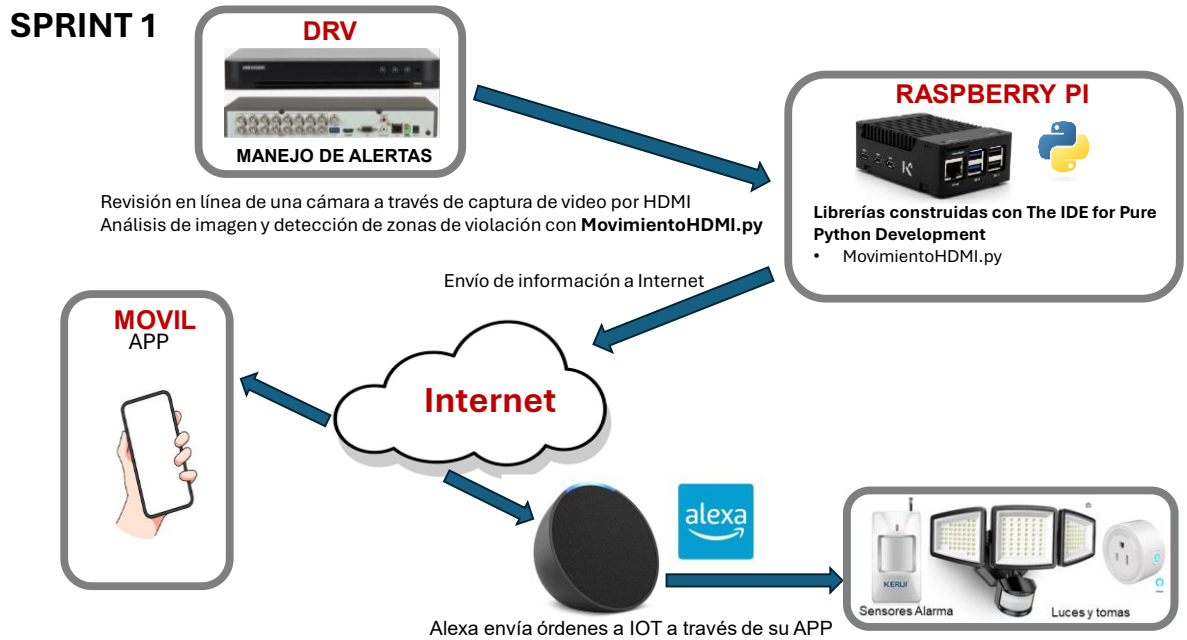
ID	¿Cómo? (Quien)	¿Quisiera? (Que)	¿Para?
H-016	Administrador del sistema	La arquitectura del prototipo debe permitir el uso de dispositivos económicos y de fácil reemplazo (por ejemplo, Raspberry PI o DVR).	Para facilitar el mantenimiento y reemplazo de dispositivos cuando estos sufran daños.
H-017	Administrador del sistema	El sistema detecte movimiento y personas en menos de 20 segundos.	Para recibir alertas en tiempo real ante cualquier intrusión.
H-018	Administrador del sistema	Las acciones de control remoto (encender luces, activar modo de seguridad, etc.) se ejecuten en menos de 20 segundo	Para tener una respuesta ágil en situaciones de emergencia.
H-019	Administrador del sistema	Los datos de mi sistema estén protegidos mediante autenticación básica	Para que solo personas autorizadas puedan acceder a la información de seguridad.
H-020	Usuario	Acceder al sistema desde mi celular	Para controlar el sistema desde el dispositivo que más uso.
H-021	Usuario	El sistema funcione con dispositivos económicos como Raspberry PI	Para que el prototipo sea accesible y fácil de mantener.
H-022	Administrador del sistema	El código y la arquitectura del sistema estén documentados	Para facilitar su actualización y resolución de errores en el futuro.
H-023	Usuario	El sistema consuma poca energía	Para no aumentar excesivamente los gastos de electricidad en mi hogar.
H-024	Administrador del sistema	El sistema use algoritmos de detección de movimiento ligeros	Para que funcione sin problemas en dispositivos de bajo rendimiento como Raspberry PI.
H-025	Usuario	El sistema esté implementado con dispositivos y componentes de bajo costo	Para mantener el proyecto accesible y ajustado a mi presupuesto.
H-026	Administrador del sistema	Los costos de operación, como almacenamiento en la nube o servicios de mantenimiento, sean mínimos	Para que el sistema sea económicamente sostenible.

3.1.4 Identificación de Sprints

Con las historias de usuario documentadas se realizó el **Sprint 0** o **fase de inicio**, en esta etapa se definió la arquitectura general del prototipo y un diseño preliminar de alto nivel. Esta actividad es importante porque se establecieron los lineamientos básicos como la estructura general de la aplicación, la arquitectura, y decisiones sobre tecnologías, con estas definiciones se pueden definir el backlog.

3.1.4.1 Definición y Arquitectura de Sprint 1. Se define el *Sprint 1* con la estructura general del proyecto.

Figura 7. *Estructura del Proyecto de Sprint 1 de desarrollo SCRUM.*



Se crea el backlog que prioriza y organiza las historias de usuario para el desarrollo del prototipo del sistema de seguridad integrado con IoT y detección de movimiento, el backlog se estructura en *épicas*, agrupando tareas relacionadas para facilitar la organización de los Sprints.

3.1.4.1.1 Descripción de la Arquitectura del Prototipo del Sprint 1. RASPBERRY PI: Se utiliza una computadora de placa única (Printed Circuit Board PCB) Raspberry 3 para la implementación de todos los programas requeridos para el proyecto, especialmente el programa de detección de movimiento.

Python versión 3.12: Se define el desarrollo de los programas requeridos para el Sistema a ejecutarse en la Raspberry 3 el lenguaje Python; específicamente la versión 3.12. dada su alta

compatibilidad con la Raspberry PI, las recientes y específicas mejoras que lo hacen desarrollar y ejecutar programas más rápidos y seguros (teniendo en cuenta el hardware limitado como la Raspberry PI 3), adicionalmente se recomienda para proyectos de IoT especialmente por su conexión con APIs y servicios web, también por su amplio ecosistema de librerías permitiendo el desarrollo fácil y rápido de programas con código portable y reutilizable.

Convertidor HDMI a CSI-2 de Raspberry PI: Dispositivo que permite conectar una señal HDMI (de video digital) al puerto CSI (Camera Serial Interface) de la Raspberry PI. Con este dispositivo se habilita la entrada HDMI a la Raspberry PI interpretando la como una cámara y permitiendo su uso a través de Python con el uso de librería OpenCV para capturar el video y separarlo en frames.

DVR Dahua DHI-HCVR5216AN-NT: DVR Trivido de 16 Canales de resolución 720p o 200 FPS 1080p. Se utiliza salida HDMI del DVR para conectarse al convertidor HDMI/CSI-2 en la Raspberry PI.

Alexa Echo Dot (5.ª generación, modelo de 2022): Se realizan pruebas configurando diferentes dispositivos IoT controlados por Alexa. Se realizan pruebas de control de programa Python a Alexa para comunicación y control con dispositivos IoT.

Enchufe inteligente redondo marca Geeni: Dispositivos IoT con Wi-Fi para ser activados al momento de presentarse un evento identificado por el programa de detección de Movimiento y reconocimiento de Personas. Se realizan pruebas para el control de los dispositivos directamente a través de programa Python.

Desarrollo de programa Python de Detección de Movimiento y Reconocimiento de Personas

Algoritmo de detección y análisis de imágenes implementado, permitió analizar las imágenes de una cámara de vigilancia e identificar una figura humana.

La imagen es capturada desde el DRV al Raspberry PI a través de convertidor HDMI a CSI-2.

Para la capturar del video y separarlo en frames se utilizó la en Python la librería OpenCV. Posteriormente los frames se utilizaron para hacer la detección de personas usando la librería MediaPipe Task, la cual permite el uso sencillo de modelos IA (machine learning). Específicamente se usó el módulo vision (análisis de imágenes y video) y el modelo pre - entrenado efficientdet_lite0.tflite. Se desarrolla el programa *MovimientoHDMI.py*.

3.1.4.1.2 Revisión de resultados y modificación de arquitectura del proyecto. A pesar de lograr un porcentaje aceptable de Precisión (confiabilidad de las detecciones), Sensibilidad (completitud del reconocimiento), tiempo de respuesta (rapidez para responder el sistema), estabilidad (tiempo de funcionamiento sin errores) y eficiencia (cantidad de CPU/RAM consumida en la Raspberry PI), estos valores corresponden solo a una cámara y el DVR y el Sistema de vigilancia está comprendida por al menos las 16 cámaras instaladas. Adicionalmente el control de Python sobre los dispositivos IoT de forma directa no lográ los resultados esperados y se identifican se encuentran muy particulares al dispositivo IoT que se instala (implementa).

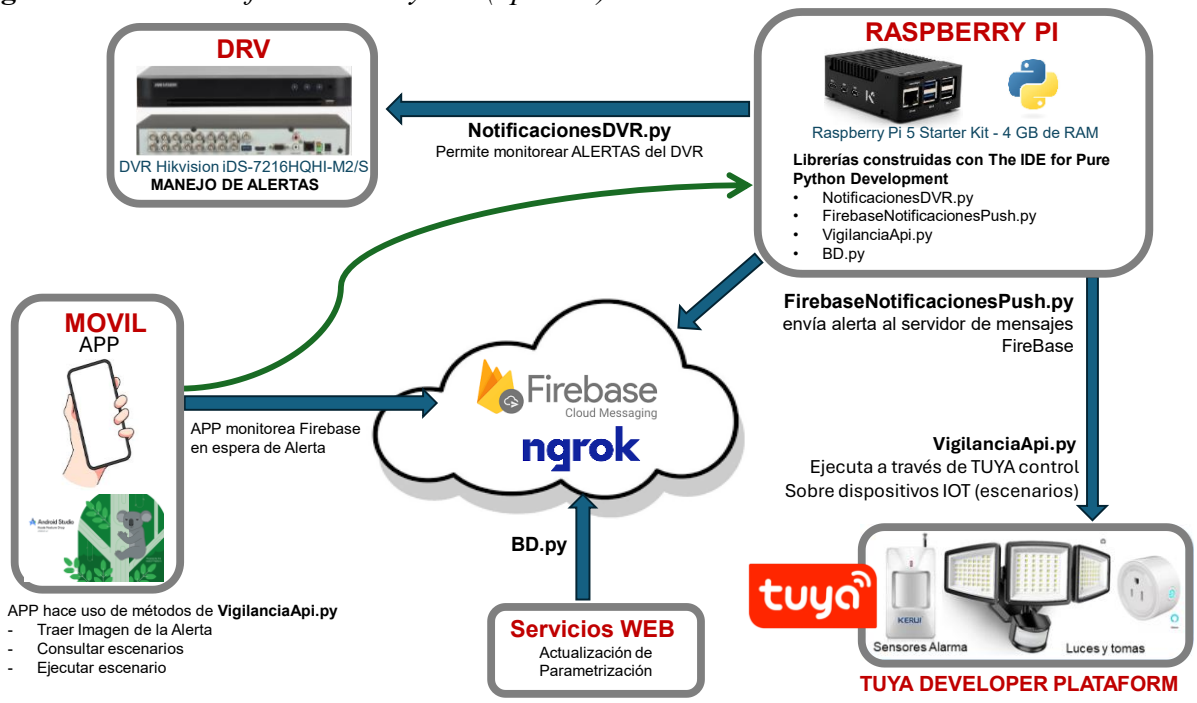
3.1.4.2 Arquitectura del Prototipo de Sistema de Seguridad Integrado usando IoT y detección de movimiento (Sprint 2). Se identifican cambios en la arquitectura inicialmente definida, definiendo una siguiente épica o funcionalidades principales, en un *Sprint 2* (definitivo):

- Modificación en la funcionalidad de Detección de Movimiento y Reconocimiento de Personas, se define hacer uso de las funcionalidades nativas del DVR Hikvision (modelo iDS-7216HQHI-M2/S), en las que se encuentran Detección de Rostros (*Face Detection*), Detección

de Intrusos (*Intrusion Detection*), Cruce de Línea (*Line Crossing Detection*) y Detección de Movimiento (*Motion Detection*).

- Comunicación de los eventos parametrizados en cada una de cámaras del DVR con la Raspberry PI a través del protocolo ISAPI (*Intelligent Security API*), protocolo desarrollado por Hikvision que permite la comunicación y gestión de dispositivos de seguridad, como DVRs a través de interfaces estándar basadas en HTTP/HTTPS.
- Modificación del control e integración de Dispositivos IoT a través de la Plataforma global de AIoT para gestionar su conectividad, almacenamiento y procesamiento denominada TUYA.
- Gestión de Notificaciones y Alertas desde la Raspberry PI hacia la aplicación móvil a través del uso de Firebase Cloud Message.
- Desarrollo de la Aplicativo Móvil (APP) en Android Studio (versión Koala), comunicándose con la Raspberry PI a través de Firebase Cloud Message para las notificaciones y Alertas; y a través de Ngrok para el llamado de servicios y programas (en la Raspberry PI) de consulta y control de los dispositivos IoT.
- Desarrollo en Python de un servicio WEB, implementado en la Raspberry PI para la parametrización del Sistema.

Figura 8. *Estructura final del Proyecto (Sprint 2) de desarrollo SCRUM.*



3.1.4.2.1 Arquitectura General del Sistema. Los componentes principales del Prototipo son:

- DVR Hikvision iDS-7216HQHI-M2/S
- Raspberry PI 5 Starter Kit (4 GB RAM)
- Firebase Cloud Messaging
- Tuya Developer Platform
- Aplicación móvil (Android Studio versión Koala)
- Servicios API de Parametrización y Operación del Sistema expuestos con NGROK y desarrollados en Python (implementados en la Raspberry PI).

Los eventos detectados (previamente configurados) por el DVR se envían vía protocolo ISAPI a la Raspberry PI, donde se procesan y notifican al usuario (en la aplicación móvil).

Simultáneamente, se consulta la parametrización del sistema para ejecutar acciones automáticas sobre dispositivos IoT.

3.1.4.2.2 Componentes Principales. *DVR Hikvision*

Dispositivo al que se conectar las 16 cámaras del sistema de vigilancia, el cual permite configurar detección de rostros (Face Detection), detección de intrusos (Intrusion Detection), cruce de línea (Line Crossing Detection) y detección de movimiento (Motion Detection). Realiza generación de alertas accesibles en tiempo real y permite comunicación con la Raspberry PI vía ISAPI (conectados a la misma LAN).

Raspberry PI

Despliega los servicios desarrollados en Python:

- NotificacionesDVR.py: Captura eventos desde el DVR mediante ISAPI.
- FirebaseNotificacionesPush.py: Envía alertas a Firebase Cloud Messaging.
- VigilanciaApi.py: Gestiona escenarios IoT mediante la plataforma Tuya.
- BD.py: Gestiona base de datos local para parametrización, usuarios y almacenamiento de eventos.

Hace uso de NGROK para exponer a internet los servicios de:

- Entregar imagen de la alerta, consultar y ejecutar escenarios de IoT a través del programa VigilanciaApi.py.
- API de Parametrización del Sistema (BD.py).

Y realiza el procesamiento de los eventos (recibidos del DVR a través de ISAPI), consultando la parametrización definida para asociar eventos con condiciones y ejecutar los escenarios de IoT.

Firebase Cloud Messaging y NGROK

Firebase es una plataforma de mensajería en la nube utilizada para enviar notificaciones push a la aplicación móvil desde el programa FirebaseNotificacionesPush.py (ejecutándose en la Raspberry PI). Las notificaciones incluyen información textual del evento y de la imagen a ser visualizada en la APP construida.

Ngrok es una herramienta y servicio en la nube que permite exponer los servicios de la Raspberry PI a internet mediante túneles seguros.

Aplicación Móvil

Desarrollada en Android Studio (versión Koala, 2024.1.1), proyecto AppNotificaciones (en Java), sus funcionalidades incluyen:

- Login seguro (usuario y contraseña mínimo 8 caracteres).
- Recepción de alertas con imagen.
- Consulta de historial de notificaciones.
- Consulta y ejecución de escenarios IoT.
- Consulta de parametrización (eventos, condiciones, dispositivos).
- Visualización y filtrado de eventos por tipo, fecha, dispositivo.

Tuya Developer Platform

Plataforma para control y monitoreo de dispositivos IoT (luces, alarmas, sensores, etc.). Se integra con VigilanciaApi.py para consultar y/o ejecutar escenarios automáticos según condiciones predefinidas.

Servicios Web de parametrización y Consulta

Servicios desplegados en la Raspberry PI para la entregar (a la APP) imagen de la alerta, consultar y ejecutar escenarios de IoT a través del programa VigilanciaApi.py. Se incluyen otras

funcionalidades como Autenticación de usuarios y verificación de sesiones.
También se incluye API de Parametrización del Sistema (BD.py), incluyendo endpoints para CRUD de eventos, condiciones, escenarios y acciones.

3.1.4.2.3 Flujo de acciones, información y trabajo. El prototipo construido tiene el siguiente flujo de acciones:

1. El DVR detecta movimiento o personas no autorizadas.
2. NotificacionesDVR.py captura la alerta mediante protocolo ISAPI.
3. Se registra el evento e imagen en la base de datos (BD.py).
4. Se consulta la parametrización para determinar si debe ejecutarse un escenario IoT.
5. Se ejecuta el escenario usando VigilanciaApi.py (utilizando la plataforma Tuya).
6. Se envía una notificación push a la app (a través de firebase) con FirebaseNotificacionesPush.py.
7. El usuario recibe la alerta en su dispositivo (consultando texto de mensaje y datos de la imagen de firebase), la imagen del evento es consultada a través de un servicio expuesto por Ngrok desde la Raspberry PI.
8. Desde la APP el usuario puede consultar el historial o ejecutar acciones manuales sobre los dispositivos IoT.
9. Se exponen servicios CRUD para parametrización del servicio a través de Ngrok expuestos desde la Raspberry PI.

3.1.4.2.4 Parámetros y Configuraciones. El sistema permite definir y modificar:

- Eventos del DVR a monitorear.

- Condiciones del sistema (día/noche, presencia, etc.).
- Escenarios y dispositivos IoT a ejecutar.
- Asociación de eventos, condiciones y acciones específicas.

3.1.4.2.5 Aspectos de Seguridad. Se implementan las siguientes funcionalidades:

- Login seguro en la APP con credenciales robustas y contraseña mínima de 8 caracteres.
- Autenticación por token y reglas de acceso en servicios REST.
- Encriptación y ocultamiento de contraseñas en la APP.
- Acceso autenticado a datos y servicios desde la APP y la nube.

3.1.4.2.6 Tecnologías Utilizadas. En resumen, se hace uso de las siguientes tecnologías y/o herramientas:

Hardware: DVR Hikvision, Raspberry PI

Lenguajes: Python, Java (Android)

Servicios: Firebase, NGROK, Tuya

Librerías: PyHik, FastAPI, smtplib, email.mime, sqlite

IDE: Android Studio (versión Koala)

3.1.4.2.7 Consideraciones Técnicas. El prototipo elaborado debe tener en cuenta las siguientes consideraciones:

- Todos los módulos y componentes se comunican por HTTP/HTTPS.
- El DVR debe tener habilitado el protocolo ISAPI y estar accesible a la Raspberry PI desde la red local.

- Se requiere conectividad estable en la Raspberry PI para mantener el servicio NGROK activo.
- La plataforma de mensajería Firebase requiere configuración del canal de mensajería push.
- La base de datos implementada debe estar correctamente configurada y segura para manejar datos sensibles.

3.1.5 Definición de Requerimientos

3.1.5.1 Requerimientos Funcionales. A continuación, se relacionan las Épicas, Requerimientos de usuario e Historias de Usuario identificados:

Tabla 3. *Estimación de Historias de Usuario y Requisitos de desarrollo SCRUM.*

ID	Historia ID	Tarea	Prioridad	Estimación (días)
EPICA 1:		**Detección de Movimiento y Reconocimiento de Personas**		
R-001	H-001 H-024	Implementar funcionalidad en Python que permita analizar las imágenes de una cámara de vigilancia e identifique una figura humana.	Alta	10
R-002	H-017	El proceso de detección de una persona no debe ser superior 10 segundos.	Alta	5
R-003	H-002 H-016 H-021	Configurar DVR para detectar personas en todas las cámaras instaladas. Conectar DVR con la Raspberry PI usando el protocolo ISAPI y leer las alertas emitas por el DVR.	Alta	5
EPICA 2:		**Control e Integración de Dispositivos IoT**		
R-004	H-011 H-012 H-015	Identificar desde un aplicativo desarrollado en Python los dispositivos IOT registrados en la plataforma de TUYA.	Alta	5
R-005	H-005 H-018	Activar o desactivar un dispositivo IOT desde un aplicativo desarrollado en Python	Alta	2
EPICA 3		**Gestión de Notificaciones y Alertas**		
R-006	H-003	Crear una funcionalidad para enviar notificaciones al dispositivo móvil cuando se detecte movimiento o una persona.	Alta	2
R-007	H-009	Registrar los eventos de detección de movimiento y reconocimiento de personas.	Alta	3
R-008	H-004	Notificar al usuario a través del correo electrónico cuando se detecte movimiento o personas no autorizadas	Alta	3

ID	Historia ID	Tarea	Prioridad	Estimación (días)
R-009	H-004	Incluir la imagen de la persona no autorizada en la notificación enviada al correo.	Media	1
EPICA 4		**Aplicativo Móvil**		
R-010	H-006 H-014 H-020	Implementar APP en Android Studio Koala con una interfaz sencilla y fácil de usar.	Alta	5
R-011	H-019	Implementar opción de LOGIN el cual solicita un usuario y una contraseña mínimo de 8 caracteres.	Alta	2
R-012	H-010	Crear funcionalidad que permita consultar el historial de eventos de seguridad.	Alta	3
R-013	H-011	Crear funcionalidad que permita consultar los dispositivos registrados en Tuya y asociados al proyecto de Seguridad.	Alto	6
R-014	H-008 H-022	Consultar la parametrización de Dispositivos, Escenario de Dispositivos IoT y Notificaciones	Alta	6
EPICA 5		**Monitoreo de Seguridad en línea**		
R-015	H-007 H-008 H-009 H-013 H-023 H-024	Monitorear las notificaciones enviadas por el DVR, identificando el evento y realizando de forma automática la notificación a la APP, el envío de correo y la ejecución de acciones de dispositivos IoT según parametrización realizada.	Alta	6
EPICA 6		**Servicios WEB de Actualización de Parametrización**		
R-016	H-025 H-026	Actualización de la parametrización de Eventos, Condiciones (del Sistema), Escenario de Dispositivos IoT, Acciones de Escenarios de Dispositivos IoT y asociación de Eventos, Condiciones y Escenarios de Dispositivos IoT.	Media	6

3.1.5.2 Requisitos No Funcionales. Para el desarrollo del prototipo se identificaron los siguientes requisitos no funcionales:

Tabla 4. *Requisitos No funcionales del desarrollo SCRUM.*

Id	Característica	Descripción
RNF-01	Usabilidad	La interfaz del sistema debe ser intuitiva y fácil de usar para usuarios sin conocimientos técnicos avanzados, permitiendo una navegación simple para configurar dispositivos y escenarios.
RNF-02	Escalabilidad	El sistema debe permitir la integración futura de dispositivos adicionales (por ejemplo, cámaras o sensores IoT) sin afectar su rendimiento.
RNF-03	Rendimiento	El tiempo de respuesta para detectar movimiento, identificar personas y enviar notificaciones debe ser inferior a 20 segundos.
RNF-04	Rendimiento	Las operaciones básicas, como encender luces o activar el modo de seguridad, deben ejecutarse en menos de 20 segundo para garantizar una respuesta ágil.
RNF-05	Documentación del Sistema	La documentación debe estar completa y actualizada para facilitar el mantenimiento y las actualizaciones.
RNF-06	Mantenibilidad	El código y la arquitectura del prototipo deben ser simples y estar documentados para facilitar la corrección de errores o la actualización de funcionalidades.
RNF-07	Costos	Implementar una solución de bajo costo, utilizando componentes económicos y de fácil acceso para que sea accesible para usuarios en áreas rurales. Bajo costo se entiende por un presupuesto alrededor de 2 SLMV.

Id	Característica	Descripción
RNF-08	Portabilidad	La arquitectura del prototipo debe permitir el uso de dispositivos económicos y de fácil reemplazo (por ejemplo, Raspberry PI o DVR).
RNF-09	Seguridad	Asegurar que los datos del usuario, como el historial de eventos y configuraciones, estén protegidos mediante autenticación básica para el acceso a la aplicación y encriptación de datos sensibles.
RNF-10	Disponibilidad	La plataforma debe estar disponible 98% del tiempo, permitiendo la supervisión en tiempo real y el control de dispositivos en cualquier momento.

3.1.5.3 Restricciones. Para el desarrollo del prototipo se identificaron las siguientes restricciones:

Tabla 5. *Restricciones del desarrollo SCRUM.*

ID	Restricción
RT-001	Dado que el sistema se enfoca en áreas rurales y de expansión, debe funcionar con una conexión de internet inestable o de baja velocidad. Esto limita el envío de video en tiempo real y exige priorizar notificaciones y alertas de baja carga de datos.
RT-002	El prototipo debe desarrollarse con un presupuesto ajustado, lo cual restringe la elección de sensores, cámaras y dispositivos de control, obligando a priorizar componentes esenciales de bajo costo.
RT-003	El sistema debe cumplir con todas las normativas y regulaciones locales relacionadas con la seguridad y la privacidad de los datos.
RT-004	Los dispositivos deben ser resistentes a condiciones climáticas adversas.
R-005	El sistema debe ser energéticamente eficiente debido al acceso limitado o inestable a la energía.
RT-006	Se debe utilizar tecnologías y estándares de comunicación compatibles con la infraestructura existente.
RT-007	El proyecto debe completarse en un plazo determinado.
R-008	El sistema debe incluir materiales de capacitación para usuarios con conocimientos técnicos limitados.
RT-009	El software del sistema debe ser compatible con plataformas y dispositivos de bajo costo, por lo que se limita a sistemas operativos y herramientas que puedan ejecutarse en dispositivos como Raspberry PI.
RT-010	La integración de plataformas como Tuya implica que el sistema dependa de APIs y aplicaciones específicas, lo que podría limitar la personalización y flexibilidad del sistema.
RT-011	Los costos operativos, incluyendo almacenamiento en la nube y mantenimiento, deben mantenerse al mínimo para garantizar que el sistema sea económicamente viable para su público objetivo.
RT-012	Dado que se trata de un prototipo, el tiempo de desarrollo es limitado, lo cual restringe el alcance de pruebas y mejoras detalladas en cada componente.
RT-013	La planificación y ejecución de las funcionalidades avanzadas (como comandos de voz e integración de escenarios) deben limitarse a las capacidades básicas, dado que el objetivo es validar el concepto más que crear un producto final detallado.

3.1.5.4 Planificación y Estimación. Considerando el tiempo definido para el desarrollo del prototipo y los requerimientos a implementar se acuerda con el equipo scrum realizar 2 iteraciones o Sprints.

Con la primera iteración (Sprint 1) se buscar obtener la identificación de personas y la integración de dispositivos IOT a través de TUYA. Con la segunda iteración (Sprint 2) se busca obtener el prototipo del sistema de seguridad con la aplicación móvil y los servicios de parametrización del sistema.

En el primer sprint, definimos la identificación de personas en la imagen de una cámara: un programa desarrollado en Python y ejecutándose en una Raspberry PI 3 recibe la imagen de la cámara (a través del puerto HDMI) y analiza imagen por imagen, identificando si hay una figura humana; adicionalmente la Raspberry PI 3 envía un mensaje a través de un servidor (Firebase Clound Message) para ser recibido por una App (mensaje indicando que se visualizó una figura humana). En este sprint se configuró la plataforma Tuya para la gestión de los dispositivos IoT que se desean controlar y se establece comunicación de la Raspberry PI 3 con la plataforma Tuya para activación de los dispositivos IoT.

En el segundo sprint se modifica el diseño inicialmente planteado del prototipo habilitando y configurando en el DVR sus funcionalidades nativas de: Detección de Movimiento, Protección Perimetral y Detección Facial del DVR; y se configuran Notificaciones que son las identificadas por una aplicación desarrollada en la Raspberry PI. Se hace uso de un programa desarrollado en Phyton en la Raspberry PI 3 para la ejecución secuencial de varios dispositivos IoT (mediante la plataforma Tuya), nos enfocamos en la configuración de escenarios de seguridad y la consulta de notificaciones. Se finaliza el Sprint desarrollando el programa de Monitoreo de Seguridad en línea y los Servicios WEB de parametrización de Eventos, Condiciones (del Sistema), Escenario de Dispositivos IoT, Acciones de Escenarios de Dispositivos IoT y asociación de Eventos, Condiciones y Escenarios de Dispositivos IoT (requeridos para el funcionamiento automático del monitoreo de seguridad en línea).

A continuación, se muestran los requerimientos a implementar en cada sprint:

Tabla 6. *Estimación de Requerimientos y Épicas del desarrollo SCRUM.*

Sprint	ID	EPICA	Estimación (días)	Responsable
1	R-001	Detección de Movimiento y Reconocimiento de Personas	10	Víctor Carrillo
	R-002		5	Víctor Carrillo
	R-003		5	Víctor Carrillo
	R-004	Control e Integración de Dispositivos IoT	5	Eudrey Reyes
	R-005		2	Eudrey Reyes
	R-006	Gestión de Notificaciones y Alertas	2	Víctor Carrillo
	R-007		3	Víctor Carrillo
	R-008		3	Víctor Carrillo
	R-009		1	Víctor Carrillo
		SPRINT 1	36	
2	R-010	Aplicativo Móvil	5	Eudrey Reyes
	R-011		2	Eudrey Reyes
	R-012		3	Eudrey Reyes
	R-013		6	Víctor Carrillo
	R-014		6	Víctor Carrillo
	R-015	Monitoreo de Seguridad en línea	6	Víctor Carrillo
	R-016	Servicios WEB de Actualización de Parametrización	6	Víctor Carrillo
		SPRINT 2	34	

3.2 Fase de Implementación

El propósito de esta fase es construir y probar las funcionalidades necesarias para que el sistema de seguridad cumpla con los objetivos propuestos, aprovechando las tecnologías de IoT y Python para asegurar la eficacia y la integridad del sistema.

Actividades y Tareas de Implementación por Historia de Usuario

Se realiza el sprint 1. Se detallan y realizan las actividades de implementación.

Figura 9. *Requerimiento funcional de Implementar en Python análisis de las imágenes de una cámara de vigilancia e identificación de una figura humana.*

ID	Tarea	Estimación (días)
EPICA 1	**Detección de Movimiento y Reconocimiento de Personas**	
R-001	Implementar funcionalidad en Python que permita analizar las imágenes de una cámara de vigilancia e identifique una figura humana.	10
TAREAS	Implementación del algoritmo de detección y análisis de imágenes en Python	
	Pruebas de precisión y ajuste en diversos niveles de iluminación.	
	Registro de detecciones con detalles de la imagen y marcas de tiempo.	
Criterios de Aprobación		**Aprobado**
El sistema detecta y reconoce figuras humanas en tiempo real desde la cámara.		SI
La detección tiene una precisión de al menos un 60% en condiciones de iluminación media.		SI
Evidencia		

Figura 10. *Requerimiento funcional del proceso de detección de una persona.*

ID	Tarea	Estimación (días)
EPICA 1	**Detección de Movimiento y Reconocimiento de Personas**	
R-002	El proceso de detección de una persona no debe ser superior a 10 segundos.	5
TAREAS	Compra e instalación de convertidor HDMI a CSI-2 de Raspberry PI.	
	Identificación de mejor calidad de captura de video, que permita una identificación confiable en la menor tasa de transferencia de información gráfica.	
	Modificación del algoritmo de detección y análisis de imágenes en Python.	
	Prueba de comparación de confiabilidad previo y posterior a la modificación del algoritmo de detección y análisis de imágenes en Python.	
Criterios de Aprobación		**Aprobado**
Las alertas de detección de una persona se generan y envían en menos de 20 segundos		SI
Las pruebas de rendimiento muestran que la detección se mantiene bajo el tiempo especificado con variabilidad mínima.		SI
Implementación		

Programa desarrollado en Python MovimientoHDMI.py

MovimientoHDMI.py

D: > Personales > Maestria2023 > Proyecto > Vigilancia > MovimientoHDMI.py

```python
import mediapipe as mp
from mediapipe.tasks.python import vision
from mediapipe.tasks.python import BaseOptions
import cv2

# Especificar la configuración del detector de objetos
options = vision.ObjectDetectorOptions(
    base_options=BaseOptions(model_asset_path="Modelo/efficientdet_lite0.tflite"),
    max_results=5,
    score_threshold=0.2,
    running_mode=vision.RunningMode.VIDEO)
detector = vision.ObjectDetector.create_from_options(options)

# Leer el video de entrada
cap = cv2.VideoCapture("Capturas/video.mp4")
frame_count = cap.get(cv2.CAP_PROP_FRAME_COUNT)
fps = cap.get(cv2.CAP_PROP_FPS)

for frame_index in range(int(frame_count)):
    ret, frame = cap.read()
    if ret == False:
        break

```

Figura 11. *Requerimiento funcional de Configuración de DVR y Conexión a Raspberry PI con protocolo ISAPI. Parte 1.*

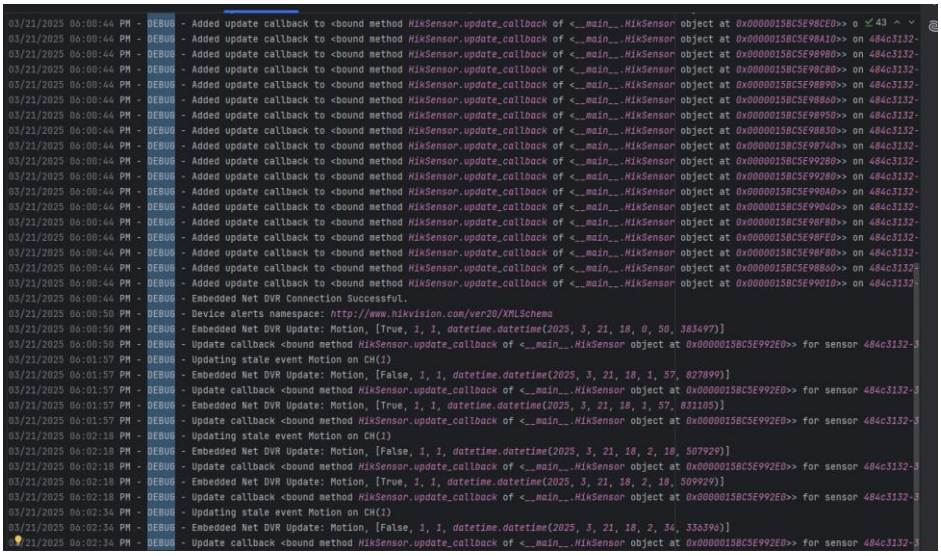
ID	Tarea	Estimación (días)
EPICA 2	**Detección de Movimiento y Reconocimiento de Personas**	
R-003	Configurar DVR para detectar personas en todas las cámaras instaladas. Conectar DVR con la Raspberry PI usando el protocolo ISAPI y captura de las alertas emitas por el DVR.	5
TAREAS	Creación de Eventos en DVR HikVision. En la configuración avanzada del DVR, el modelo del DRV utilizado en el prototipo (DVR iDS-7216HQHI-M2/S) permite la creación de 3 clases de eventos: • Detección de Movimiento. • Protección Perimetral. • Detección Facial del DVR. Todo evento de los 3 tipos anteriormente mencionados que es definido en el DRV son accesibles a través del protocolo ISAPI suministrado por el DVR (el cual se debe habilitar en el DVR para su uso).	
	Configuración del Entorno de desarrollo: • Instalar Python y la biblioteca pyHik en la Raspberry PI. • Verificar en el DRV el protocolo ISAPI y que el DRV y la Raspberry PI estén conectados a la misma red. • Configurar las credenciales del DVR (dirección IP, usuario y contraseña) en el programa de Python y establecer uso de la biblioteca pyHik para establecer una conexión con el DVR a través de ISAPI. El programa en Python, inicializa un cliente de pyHik, que autentica las credenciales y verifica la conexión con el DVR.	
	Monitoreo de Eventos: • Enviar una solicitud al DVR (utilizando ISAPI), recibir en tiempo real los eventos generados por las cámaras conectadas al DVR y procesarlas en la Raspberry PI.	
	Procesamiento y Tratamiento de la información. Se debe hacer lectura del evento, el programa debe registrar: • Cámara / dispositivo Origen. • Tipo de Evento: Clasificados en Detección de Movimiento, Protección Perimetral y Detección Facial, los Tipos de Evento son: *Motion*, *LineDetection*, *FieldDetection*, VMD, *AlarmInput*, *VideoLoss*, *TamperDetection*, *FaceDetection*, *AudioInput*, *RegionEntrance*, *RegionExiting*, *HeatMap*, entre otros.	

Criterios de Aprobación	Aprobado
La detección de personas se activa correctamente en cada cámara y se registra como evento en el sistema del DVR.	SI
Las alertas de detección son emitidas por el DVR para cada cámara en menos de 20 segundos tras el reconocimiento de una figura humana.	SI
La Raspberry PI recibe las alertas del DVR en tiempo real cada vez que se detecta una persona.	SI
Las alertas recibidas en la Raspberry PI se almacenan con detalles de la cámara que originó la detección y la hora del evento.	SI
Se realizan pruebas en cada cámara conectada al DVR para verificar la correcta identificación de personas y el envío de alertas a la Raspberry PI.	SI
El sistema emite un mensaje de confirmación en la interfaz de la Raspberry PI por cada evento de detección recibido correctamente.	SI

Figura 12. *Requerimiento funcional de Configuración de DVR y Conexión a Raspberry PI con protocolo ISAPI. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 2	**Detección de Movimiento y Reconocimiento de Personas**	
R-003	Configurar DVR para detectar personas en todas las cámaras instaladas. Conectar DVR con la Raspberry PI usando el protocolo ISAPI y captura de las alertas emitas por el DVR.	5

Evidencia



Implementación

Programa desarrollado en Python NotificacionesDvr.py

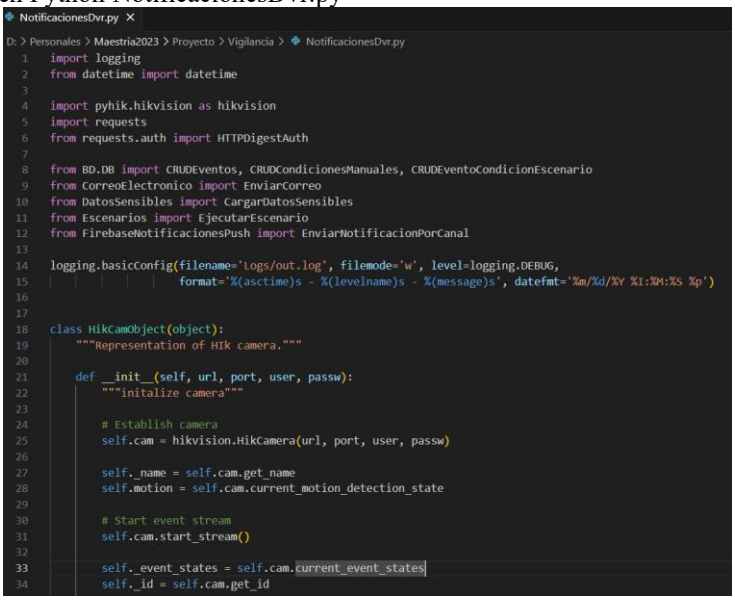
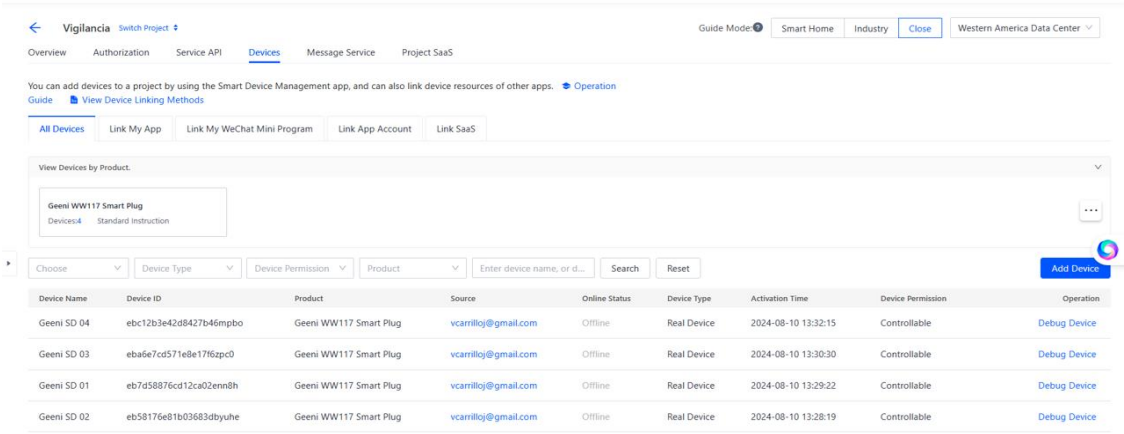


Figura 13. *Requerimiento funcional de Identificar desde un aplicativo desarrollado en Python los dispositivos IOT registrados en la plataforma de TUYA. Parte 1.*

ID	Tarea	Estimación (días)
EPICA 2	**Control e Integración de Dispositivos IoT**	
R-004	Identificar desde un aplicativo desarrollado en Python los dispositivos IOT registrados en la plataforma de TUYA.	5
TAREAS	Configuración e instalación del entorno: • Instalar de librerías en Phyton. • Registrar la cuenta de desarrollador en *Tuya IoT Platform*.	
	Obtención de credenciales de acceso, para interactuar con la API, AccessID, Password y URL de acceso.	
	Conexión de Phyton con la API de Tuya: • Configurar de la autenticación para interactuar con la API de Tuya. • Implementar de la conexión inicial y obtén un token de acceso.	
	Identificación de los dispositivos registrados: • Utilizar los *endpoints* de la API de Tuya para obtener la lista de dispositivos IoT vinculados a la cuenta. • Identificar tipo, estado y ubicación de los dispositivos.	
	Integración con la aplicación Phyton: • Integrar la información obtenida en aplicación (desarrollada en Python). • Configurar los eventos para actualizar automáticamente la lista de dispositivos.	
	Realización de pruebas, depuración y Documentación: • Codificar el programa para verificar que manejar errores como son: o Dispositivos no encontrados. o Credenciales inválidas. o Tiempo de espera en la API. • Documentar el proceso de configuración y uso del código para facilitar futuras actualizaciones.	
Criterios de Aprobación		**Aprobado**
El aplicativo en Python se autentica y conecta exitosamente a la API de TUYA utilizando las credenciales proporcionadas.		SI
La interfaz del aplicativo muestra la lista de dispositivos de manera clara y organizada, permitiendo una fácil visualización y comprensión por parte del usuario.		SI
Las credenciales de autenticación están almacenadas de manera segura y no se exponen en el código fuente ni en la interfaz de usuario.		SI
El aplicativo incluye documentación básica que describe cómo configurar y conectar con la plataforma de TUYA [14].		SI

Figura 14. *Requerimiento funcional de Identificar desde un aplicativo desarrollado en Python los dispositivos IOT registrados en la plataforma de TUYA. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 2	**Control e Integración de Dispositivos IoT**	
R-004	Identificar desde un aplicativo desarrollado en Python los dispositivos IOT registrados en la plataforma de TUYA.	5
Evidencia		



Implementación

Programa desarrollado en Python TuyaDispositivos.py

TuyaDispositivos.py

D: > Personales > Maestria2023 > Proyecto > Vigilancia > TuyaDispositivos.py

```python
import json
import logging
from tuya_connector import TuyaOpenAPI, TUYA_LOGGER

from BD.Tablas import Acciones
from DatosSensibles import CargarDatosSensibles

ACCESS_ID = CargarDatosSensibles("ACCESS_ID")
ACCESS_KEY = CargarDatosSensibles("ACCESS_KEY")
API_ENDPOINT = "https://openapi.tuyaus.com"

openapi = TuyaOpenAPI(API_ENDPOINT, ACCESS_ID, ACCESS_KEY)
openapi.connect()

def EjecutarComandos(acciones: list[Acciones]):
    try:
        for accion in acciones:
            commands = json.loads(accion.comando.replace("'", "\""))
            id_device = accion.id_dispositivo
            openapi.post('/v1.0/devices/{}/commands'.format(id_device), commands)
    except Exception as e:
        print(f"Error: {e}")

def ObtenerDispositivos():
    response = openapi.get("/v2.0/cloud/thing/device?page_size=20")
    if response.get("success"):
        dispositivos = response.get("result", [])
        return dispositivos
    else:
        raise Exception(f"Error al obtener dispositivos: {response}")

```

Figura 15. *Requerimiento funcional de Activar o desactivar un dispositivo IOT desde un aplicativo desarrollado en Python. Parte 1.*

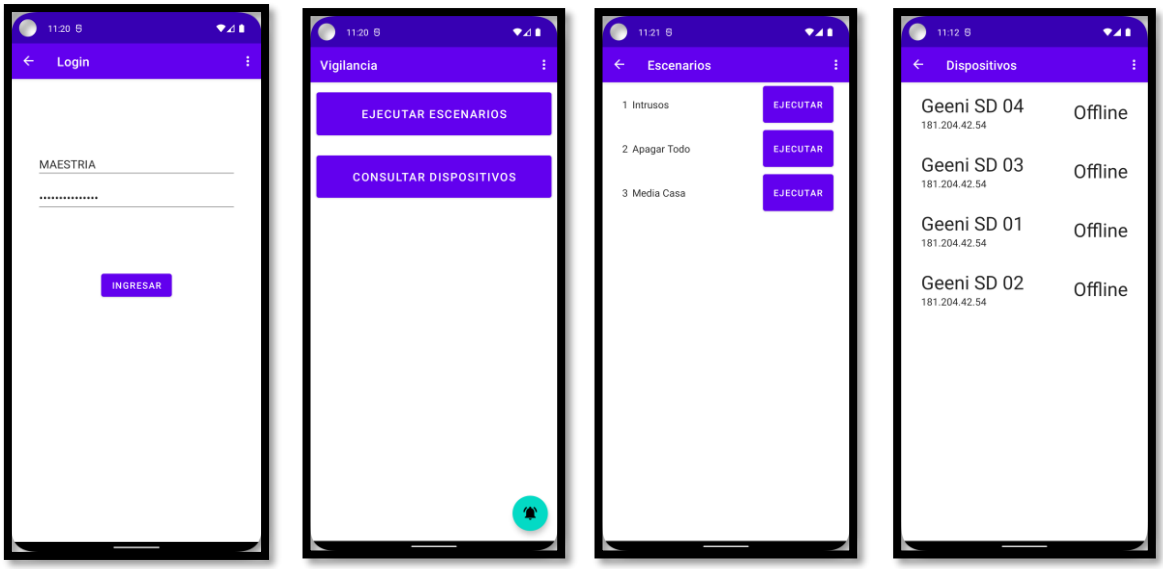
ID	Tarea	Estimación (días)
EPICA 2	**Control e Integración de Dispositivos IoT**	
R-005	Activar o desactivar un dispositivo IOT desde un aplicativo desarrollado en Python.	2
TAREAS	Identificación y/o registro de los dispositivos IoT que se desean controlar, validar que se encuentren registrados en el proyecto y conoce su DeviceID.	
	Verificación de permisos y credenciales: Confirmar las credenciales de Tuya (Access ID y Access Secret) con acceso para controlar dispositivos IoT (no solo consultar).	
	Identificación del estado actual del dispositivo: Uso la API de Tuya para consultar estado actual del dispositivo previo al envío de comando de encendido/apagado.	
	Envío de comando para activar o desactivar el dispositivo y confirmación de la ejecución del comando • Utilizar el *endpoint* de control de dispositivos, con los comandos de encendido/apagado. • Verificar nuevo estado del dispositivo para confirmar que la activación fue exitosa.	
	Realización de pruebas, depuración y Documentación: • Codificar el programa para control de errores/eventos como son: o El dispositivo no está en línea. o El comando enviado no es compatible con el dispositivo. o Credenciales inválidas. • Implementar el manejo de excepciones y/o la verificación de respuesta de la librería en el código para evitar que el programa falle con los errores relacionados. • Integrar con la aplicación, para seleccionar el dispositivo y enviar comandos. • Realizar pruebas al programa con diferentes dispositivos IoT y escenarios: o Dispositivo encendido/apagado correctamente. o Dispositivo fuera de línea o con problemas de conexión. o Respuesta incorrecta de la API. • Documentar el proceso de activación/desactivación y uso del código para facilitar futuras actualizaciones.	

Criterios de Aprobación	Aprobado
El aplicativo permite al usuario activar un dispositivo IoT registrado en la plataforma de TUYA.	SI
El aplicativo permite al usuario desactivar un dispositivo IoT registrado en la plataforma de TUYA.	SI
La acción de activar o desactivar un dispositivo se ejecuta con un retraso máximo de 20 segundos desde que el usuario realiza la solicitud en el aplicativo.	SI
La interfaz muestra el estado actual del dispositivo (activado/desactivado) de manera clara e intuitiva antes y después de cada acción.	SI
El aplicativo solo permite la activación o desactivación de dispositivos a usuarios autenticados.	SI

Figura 16. *Requerimiento funcional de Activar o desactivar un dispositivo IOT desde un aplicativo desarrollado en Python. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 2	**Control e Integración de Dispositivos IoT**	
R-005	Activar o desactivar un dispositivo IOT desde un aplicativo desarrollado en Python.	2

Evidencia



Implementación

Programa desarrollado en Python TuyaDispositivos.py

TuyaDispositivos.py

D: > Personales > Maestria2023 > Proyecto > Vigilancia > TuyaDispositivos.py

```python
import json
import logging
from tuya_connector import TuyaOpenAPI, TUYA_LOGGER

from BD.Tablas import Acciones
from DatosSensibles import CargarDatosSensibles

ACCESS_ID = CargarDatosSensibles("ACCESS_ID")
ACCESS_KEY = CargarDatosSensibles("ACCESS_KEY")
API_ENDPOINT = "https://openapi.tuyaus.com"

openapi = TuyaOpenAPI(API_ENDPOINT, ACCESS_ID, ACCESS_KEY)
openapi.connect()

def EjecutarComandos(acciones: list[Acciones]):
    try:
        for accion in acciones:
            commands = json.loads(accion.comando.replace("'", "\""))
            id_device = accion.id_dispositivo
            openapi.post('/v1.0/devices/{}/commands'.format(id_device), commands)
    except Exception as e:
        print(f"Error: {e}")

def ObtenerDispositivos():
    response = openapi.get("/v2.0/cloud/thing/device?page_size=20")
    if response.get("success"):
        dispositivos = response.get("result", [])
        return dispositivos
    else:
        raise Exception(f"Error al obtener dispositivos: {response}")

```

Figura 17. *Requerimiento funcional de enviar notificaciones al dispositivo móvil cuando se detecte movimiento o una persona. Parte 1.*

ID	Tarea	Estimación (días)
EPICA 3	**Gestión de Notificaciones y Alertas**	
R-006	Crear una funcionalidad para enviar notificaciones al dispositivo móvil cuando se detecte movimiento o una persona.	2
TAREAS	Monitorización de los eventos de movimiento o detección de personas desde el DVR, a través de ISAPI utilizando la biblioteca pyHik, se debe consultar regularmente el DVR para obtener las alertas de los eventos mencionados en tiempo real.	
	Consulta del evento detectado, contra los eventos registrados en la entidad Eventos parametrizada: • Si se identifica asociación del nombre y descripción del evento almacenado al evento detectado del DVR (se identifica por los atributos Cámara/Dispositivo Origen y Tipo de Evento) • Si no se identifica, uso de la información de Cámara/Dispositivo Origen y Tipo de Evento.	
	Creación de lógica para determinar cuándo enviar una notificación. Se enviarán los eventos que se identifican registrados en la entidad Eventos parametrizada.	
	Integración con un servicio de notificaciones *push*: • Seleccionar de una plataforma de notificación *push*, servicio de notificaciones push para enviar las alertas al dispositivo móvil. • Se elige *Firebase Cloud Messaging* (FCM).	

	Envío de la notificación *push*:
	• Obtener el token del registro con FCM para obtener un token único (*token* requerido para enviar notificaciones).
	• Configurar FCM.
	  o Registrar aplicación en *Firebase* y configurar las credenciales para envío de notificaciones.
	  o Utilizar el token del dispositivo móvil para enviar las alertas.
	  o Enviar la notificación *push*.
	• Enviar notificaciones al dispositivo móvil a través de FCM.	

	Programación del control de notificaciones y frecuencia para evitar enviar notificaciones repetidas en poco tiempo, puedes implementar un control de frecuencia.	

	Realización de pruebas, depuración y Documentación:
	• Codificar el programa para control de errores/eventos como son:
	  o El dispositivo no está en línea.
	  o El comando enviado no es compatible con el dispositivo.
	  o Credenciales inválidas.
	• Implementar el manejo de excepciones y/o la verificación de respuesta de la librería en el código para evitar que el programa falle con los errores relacionados.
	• Integrar con la aplicación para seleccionar el dispositivo y enviar comandos.
	• Realizar pruebas al programa con diferentes dispositivos IoT y escenarios:
	  o Dispositivo encendido/apagado correctamente.
	  o Dispositivo fuera de línea o con problemas de conexión.
	  o Respuesta incorrecta de la API.
	• Documentar el proceso de activación/desactivación y uso del código para facilitar futuras actualizaciones.	

Figura 18. *Requerimiento funcional de enviar notificaciones al dispositivo móvil cuando se detecte movimiento o una persona. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 3	**Gestión de Notificaciones y Alertas**	
R-006	Crear una funcionalidad para enviar notificaciones al dispositivo móvil cuando se detecte movimiento o una persona.	2
Criterios de Aprobación		**Aprobado**
El sistema envía una notificación al dispositivo móvil del usuario dentro de 20 segundos tras detectar movimiento o una figura humana.		SI
Evidencia		

Implementación

Programa desarrollado en Python FirebaseNotificacionesPush.py.

D: > Personales > Maestria2023 > Proyecto > Vigilancia > FirebaseNotificacionesPush.py

```python
import firebase_admin
from firebase_admin import credentials, messaging

# Inicializar la aplicación Firebase con el archivo de credenciales
cred = credentials.Certificate('llave/vigilancia.json')
firebase_admin.initialize_app(cred)


def EnviarNotificacion(registration_token, message_title, message_body):
    # Crear el mensaje
    message = messaging.Message(
        notification=messaging.Notification(
            title=message_title,
            body=message_body,
        ),
        token=registration_token,
    )

    # Enviar el mensaje
    response = messaging.send(message)
    return response


def EnviarNotificacionPorCanal(canal, titulo, mensaje, foto):
    data = {
        'foto': foto,
        "titulo": titulo,
        "mensaje": mensaje
    }
```

Figura 19. *Requerimiento funcional de registrar los eventos de detección de movimiento y envío a la APP construida.*

ID	Tarea	Estimación (días)
EPICA 3	**Gestión de Notificaciones y Alertas**	
R-007	Registrar los eventos de detección de movimiento y envío a la APP construida.	3
TAREAS	Creación de rutina en Phyton para el registro de evento en estructura de datos creada para los Eventos	
	Integración con la rutina para registrar eventos, con la funcionalidad de Identificación de Eventos y demás programas desarrollados	
	Verificación los datos registrados	
Criterios de Aprobación		**Aprobado**
Cada notificación enviada al usuario se registra en el sistema con la fecha, hora y el tipo de evento detectado, permitiendo un historial de alertas.		SI
Los eventos de seguridad deben almacenarse en una base de datos accesible desde la funcionalidad construida para consulta.		SI
Evidencia		

Implementación

Programa desarrollado en Python NotificacionesDvr.py.

```python
import logging
from datetime import datetime

import pyhik.hikvision as hikvision
import requests
from requests.auth import HTTPDigestAuth

from BD.DB import CRUDEventos, CRUDCondicionesManuales, CRUDEventoCondicionEscenario
from CorreoElectronico import EnviarCorreo
from DatosSensibles import CargarDatosSensibles
from Escenarios import EjecutarEscenario
from FirebaseNotificacionesPush import EnviarNotificacionPorCanal

logging.basicConfig(filename='Logs/out.log', filemode='w', level=logging.DEBUG,
                    format='%(asctime)s - %(levelname)s - %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')


class HikCamObject(object):
    """Representation of HIk camera."""

    def __init__(self, url, port, user, passw):
        """initalize camera"""

        # Establish camera
        self.cam = hikvision.HikCamera(url, port, user, passw)

        self._name = self.cam.get_name
        self.motion = self.cam.current_motion_detection_state

        # Start event stream
        self.cam.start_stream()

        self._event_states = self.cam.current_event_states
        self._id = self.cam.get_id
```

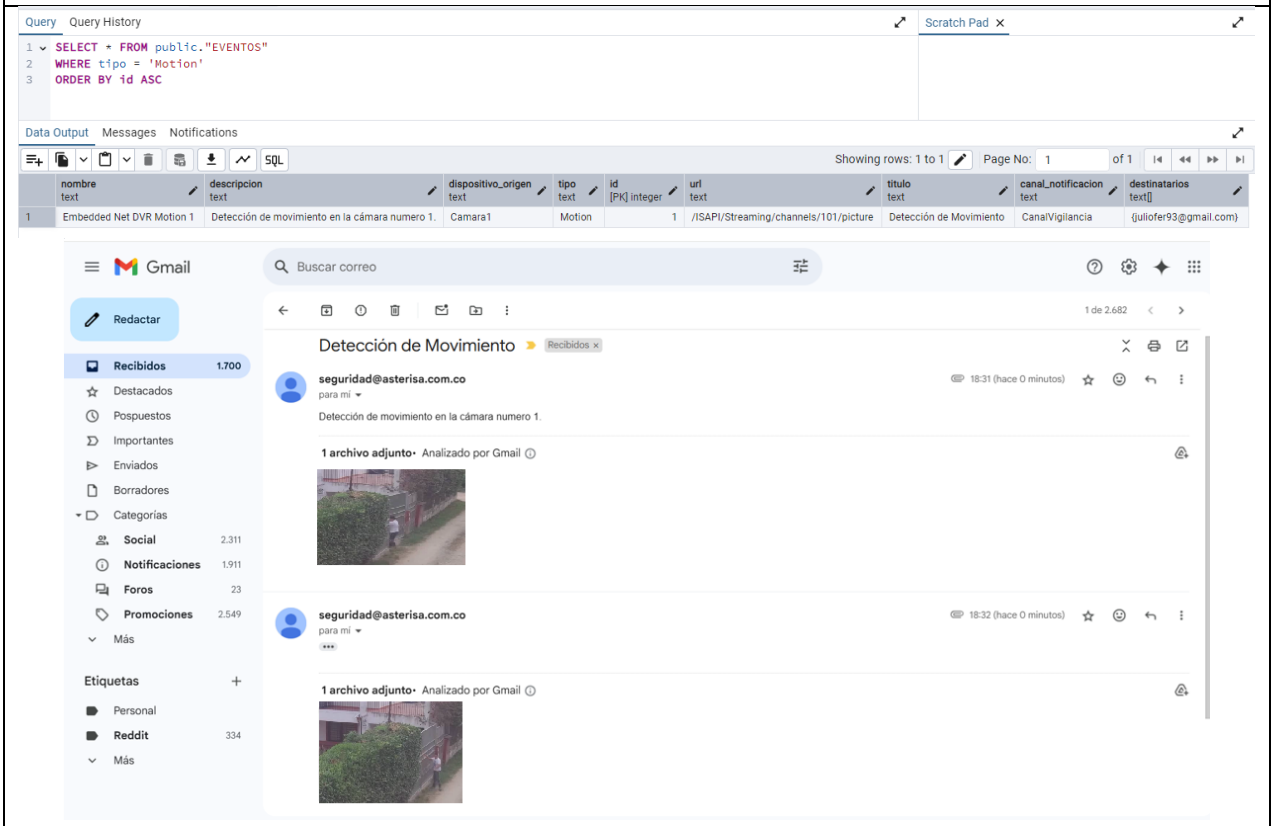
Figura 20. *Requerimiento funcional de notificar al usuario a través del correo electrónico cuando se detecte movimiento o personas no autorizadas. Parte 1.*

ID	Tarea	Estimación (días)
EPICA 3	**Gestión de Notificaciones y Alertas**	
R-008	Notificar al usuario a través del correo electrónico cuando se detecte movimiento o personas no autorizadas	3
TAREAS	Definición y/o creación de cuenta de correo para ser utilizada por la funcionalidad para el envío del correo. Se requiere tener la información de: • Dirección del servidor SMTP. • Puerto de conexión (465 para SSL). • Usuario y contraseña de autenticación.	
	Revisión y/o instalación de las librerías requeridas por Phyton para el manejo correos y archivos adjuntos, se identifican iniciamente las siguientes librerias: • *Smtplib* (*Python Standard Library* PSL): Permite la comunicación con servidores SMTP para el envío de correos electrónicos. • *email.mime* (*Python Standard Library* PSL): Proporciona clases para construir correos con texto, HTML y archivos adjuntos.	
	Construcción del programa Phyton, el cual debe: • Configurar los datos del correo. • Capturar la Imagen y el Texto. • Crear y Enviar el Correo con la Imagen Adjunta.	
	Realización de pruebas, depuración y Documentación: • Verificar conexión con el servidor SMTP. • Verificar credenciales. • Verificar el envío de correos. • Integrar con la aplicación para consulta de correos a enviar según parametrización definida. • Realizar pruebas al programa con diferentes notificaciones recibidas y enviadas por correo. • Documentar el proceso de parametrización del correo emisor y de los correos receptores, y uso del código para facilitar futuras actualizaciones.	
Criterios de Aprobación		**Aprobado**
El sistema envía automáticamente un correo electrónico al usuario dentro de 20 segundos tras detectar movimiento o la presencia de una persona no autorizada.		SI
El correo electrónico se envía a la dirección configurada previamente por el usuario en la aplicación.		SI
El correo electrónico incluye un asunto descriptivo que indique el tipo de evento detectado (por ejemplo, "Alerta: Movimiento detectado" o "Alerta: Persona no autorizada detectada").		SI

Figura 21. *Requerimiento funcional de notificar al usuario a través del correo electrónico cuando se detecte movimiento o personas no autorizadas. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 3	**Gestión de Notificaciones y Alertas**	
R-008	Notificar al usuario a través del correo electrónico cuando se detecte movimiento o personas no autorizadas	3

Evidencia



Implementación

Programa desarrollado en CorreoElectronico.py

CorreoElectronico.py

D: > Personales > Maestria2023 > Proyecto > Vigilancia > CorreoElectronico.py

```python
import os
import smtplib
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

SMTP_SERVER = "10040183.ferozo.com"
SMTP_PORT = 465  # Se usa SSL en este caso
EMAIL_SENDER = "seguridad@asterisa.com.co"
EMAIL_PASSWORD = "ProyectoMaestria2025*"


def EnviarCorreo(destinatarios: list[str], asunto: str, cuerpo: str, adjunto: str):
    msg = MIMEMultipart()
    msg["From"] = EMAIL_SENDER
    msg["To"] = ", ".join(destinatarios)
    msg["Subject"] = asunto
    msg.attach(MIMEText(cuerpo, "plain"))

    image_path = adjunto

    if os.path.exists(image_path):
        with open(image_path, "rb") as img_file:
            img = MIMEImage(img_file.read())
            img.add_header("Content-Disposition", f"attachment; filename={os.path.basename(image_path)}")
            msg.attach(img)
    else:
        raise Exception("La imagen no existe en la ruta especificada.")
```

Figura 22. *Requerimiento funcional de incluir la imagen de la persona no autorizada en la notificación enviada al correo.*

<table>
<tr><th>ID</th><th>Tarea</th><th>Estimación (días)</th></tr>
<tr><td>EPICA 3</td><td colspan="2">Gestión de Notificaciones y Alertas</td></tr>
<tr><td>R-009</td><td>Incluir la imagen de la persona no autorizada en la notificación enviada al correo.</td><td>1</td></tr>
<tr><td rowspan="2">TAREAS</td><td colspan="2">Construcción del programa Phyton, el cual debe:
• Capturar la Imagen y el Texto.
• Crear y Enviar el Correo con la Imagen Adjunta.</td></tr>
<tr><td colspan="2">Realización de pruebas, depuración y Documentación:
• Realizar pruebas al programa con diferentes textos e imágenes enviadas por correo.
• Documentar el proceso y uso del código para facilitar futuras actualizaciones.</td></tr>
<tr><td colspan="2">Criterios de Aprobación</td><td>Aprobado</td></tr>
<tr><td colspan="2">Si el evento reportado permite la captura de una imagen de la cámara que lo genera, el correo enviado debe incluir la imagen capturada disponible.</td><td>SI</td></tr>
<tr><td colspan="3">Implementación</td></tr>
<tr><td colspan="3">Programa desarrollado en CorreoElectronico.py</td></tr>
</table>

CorreoElectronico.py

D: > Personales > Maestria2023 > Proyecto > Vigilancia > CorreoElectronico.py

```python
import os
import smtplib
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

SMTP_SERVER = "l0040183.ferozo.com"
SMTP_PORT = 465  # Se usa SSL en este caso
EMAIL_SENDER = "seguridad@asterisa.com.co"
EMAIL_PASSWORD = "ProyectoMaestria2025*"


def EnviarCorreo(destinatarios: list[str], asunto: str, cuerpo: str, adjunto: str):
    msg = MIMEMultipart()
    msg["From"] = EMAIL_SENDER
    msg["To"] = ", ".join(destinatarios)
    msg["Subject"] = asunto
    msg.attach(MIMEText(cuerpo, "plain"))

    image_path = adjunto

    if os.path.exists(image_path):
        with open(image_path, "rb") as img_file:
            img = MIMEImage(img_file.read())
            img.add_header("Content-Disposition", f"attachment; filename={os.path.basename(image_path)}")
            msg.attach(img)
    else:
        raise Exception("La imagen no existe en la ruta especificada.")
```

Figura 23. *Requerimiento funcional de Implementar APP en Android Studio Koala con una interfaz sencilla y fácil de usar. Parte 1.*

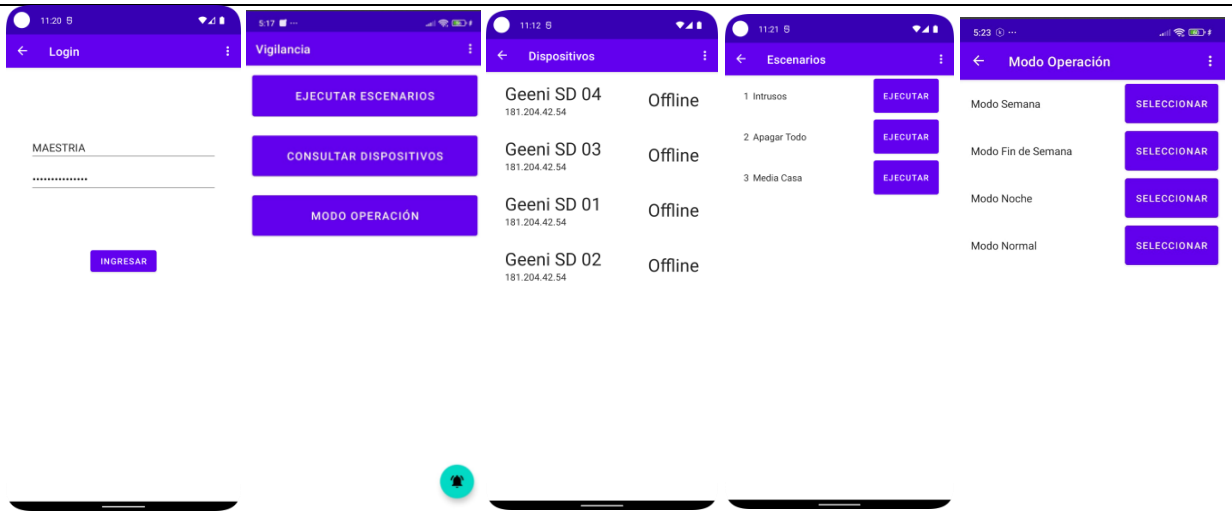
ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-010	Implementar APP en Android Studio Koala con una interfaz sencilla y fácil de usar.	5

TAREAS	Planificación del Proyecto.
	• Definir el propósito de la APP y los requisitos principales.
	• Determinar los componentes clave:
	  o Recepción de notificaciones con datos del evento.
	  o Visualización del mensaje y la imagen.	

	Configuración del Entorno de Desarrollo. • Instalar y configurar Android Studio Koala. • Crear un nuevo proyecto en Android Studio. • Configurar dependencias necesarias en build.gradle.	
	Desarrollo de la Interfaz de Usuario (UI/UX). • Diseñar una interfaz sencilla y clara con XML. • Definir las pantallas principales.	
	Pruebas y Debugging • Pruebas unitarias para validar que la app recibe y muestra las notificaciones correctamente. • Pruebas en dispositivos reales y emuladores. • Optimización del rendimiento para garantizar fluidez.	
	Publicación y Mantenimiento • Generar el APK o AAB para distribución. • Subir la app a la Google Play Store (si aplica). • Mantener y actualizar la aplicación según necesidades.	
Criterios de Aprobación		**Aprobado**
La interfaz principal de la APP debe mostrar de manera clara y accesible la información del Sistema.		SI
La navegación en la APP debe ser intuitiva, permitiendo que el usuario acceda a las funciones con un máximo de 2 clics desde la pantalla principal.		SI
Los elementos de la interfaz (botones, íconos, y texto) deben ser grandes y legibles en dispositivos de diferentes tamaños de pantalla, garantizando una buena usabilidad.		SI
La APP debe ser compatible con dispositivos Android a partir de la versión 7.0 en adelante.		SI
Si ocurre un error en la conexión o en la ejecución de una acción (por ejemplo, al controlar un dispositivo IoT), la APP notifica al usuario la desconexión.		SI
La APP debe ser probada con usuarios para asegurar que pueden comprender y utilizar las funciones principales sin dificultad.		SI

Figura 24. *Requerimiento funcional de Implementar APP en Android Studio Koala con una interfaz sencilla y fácil de usar. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-010	Implementar APP en Android Studio Koala con una interfaz sencilla y fácil de usar.	5
Evidencia		



Implementación

Repositorio VigilanciaPython en GitHub. (github.com/vcarrilloj/VigilanciaPython)

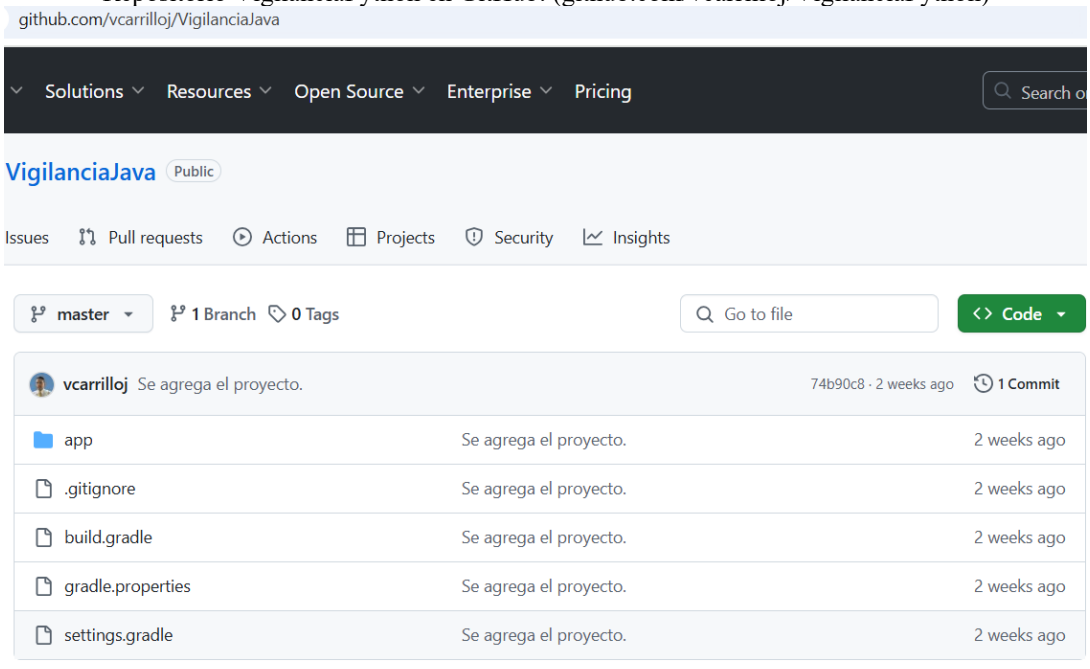


Figura 25. *Requerimiento funcional de Implementar opción de LOGIN el cual solicita un usuario y una contraseña mínimo de 8 caracteres. Parte 1.*

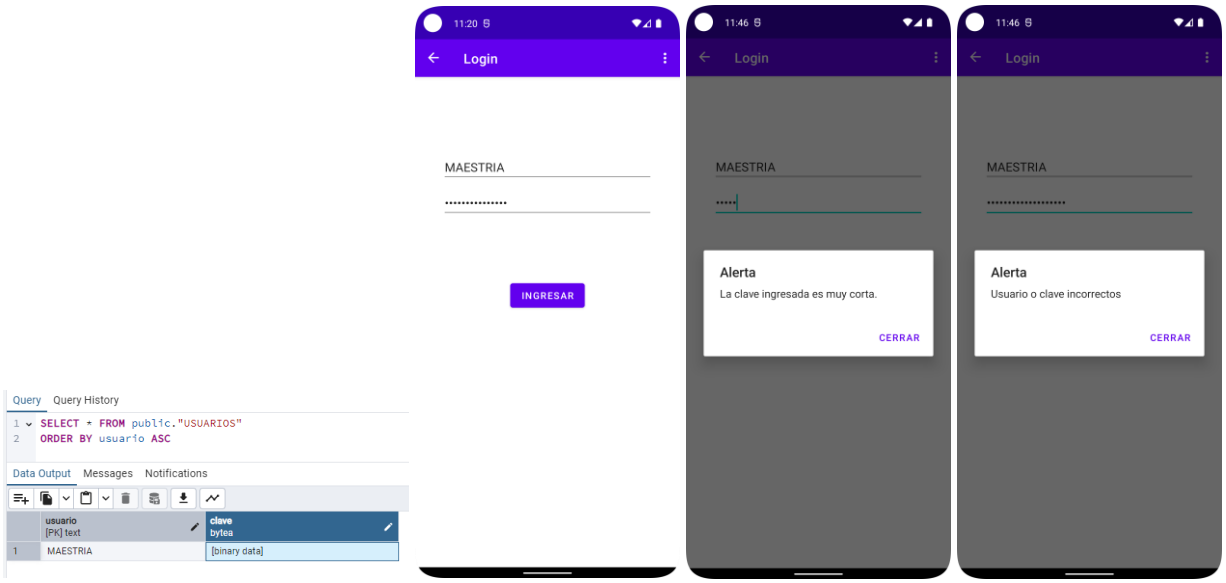
ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-011	Implementar opción de LOGIN el cual solicita un usuario y una contraseña mínimo de 8 caracteres.	2
TAREAS	Definición del Tipo de Autenticación, la opción seleccionada es la autenticación con información registrada en base de datos implantada en la Raspberry PI.	
	Configuración del Proyecto en Firebase • Acceder a la Consola de *Firebase* y seleccionar el proyecto de la APP. • Ir a la sección *Authentication* y habilitar el método de correo y contraseña. • Descargar el archivo google-services.json e incluirlo en el proyecto Android. • Agregar las dependencias necesarias en build.gradle.	
	Desarrollo de la Interfaz de Usuario (UI) para el Login, Pantallas en XML o *Jetpack Compose* de Inicio de Sesión.	
	Implementación del Registro de Usuario. • Capturar el correo y contraseña del usuario. • Verificar la información ingresada con la registrada en la base de datos.	
	Implementación del Inicio de Sesión, verificación de credenciales del usuario y autenticarse con *Firebase*.	
	Manejo de Sesiones y Restricción de Acceso, verificando si hay un usuario autenticado antes de mostrar la pantalla de notificaciones.	
	Implementación del Cierre de Sesión, agregar un botón de Cerrar sesión en la pantalla principal.	
	Realización de pruebas, depuración y Documentación: • Probar el registro y inicio de sesión en diferentes dispositivos. • Validar manejo de errores (correo en uso, contraseña incorrecta, etc.). • Revisar la persistencia de sesión cuando la app se cierra y vuelve a abrir. • Documentar el desarrollo y uso del código para facilitar futuras actualizaciones.	
	Validación de seguridad y optimización. • Habilitar Reglas de Seguridad en *Firebase* para restringir accesos no autorizados. • Activar verificación de email para mayor seguridad.	

Criterios de Aprobación	Aprobado
La pantalla de inicio de sesión debe solicitar al usuario ingresar un nombre de usuario y una contraseña.	SI
La contraseña debe tener al menos 8 caracteres. Si el usuario ingresa una contraseña con menos de 8 caracteres, el sistema debe mostrar un mensaje de error indicando este requisito.	SI
El sistema verifica que las credenciales ingresadas coincidan con un usuario registrado en la base de datos antes de permitir el acceso.	SI
La contraseña debe ocultarse (mostrando solo puntos o asteriscos) mientras el usuario la escribe, para garantizar la privacidad.	SI
Si el usuario ingresa un nombre de usuario o contraseña incorrectos, el sistema debe mostrar un mensaje de error claro, como "Usuario o contraseña incorrectos. Intente nuevamente."	SI
La autenticación debe someterse a pruebas de seguridad para verificar que los datos ingresados se manejen de forma segura (por ejemplo, encriptación de contraseñas en la base de datos).	SI
Solo es posible ingresar al aplicativo con un usuario y contraseña valido.	SI

Figura 26. *Requerimiento funcional de Implementar opción de LOGIN el cual solicita un usuario y una contraseña mínimo de 8 caracteres. Parte 2.*

EPICA 4	**Aplicativo Móvil**	
R-011	Implementar opción de LOGIN el cual solicita un usuario y una contraseña mínimo de 8 caracteres.	2

Evidencia



Implementación

Repositorio VigilanciaPython en GitHub. (github.com/vcarrilloj/VigilanciaPython)

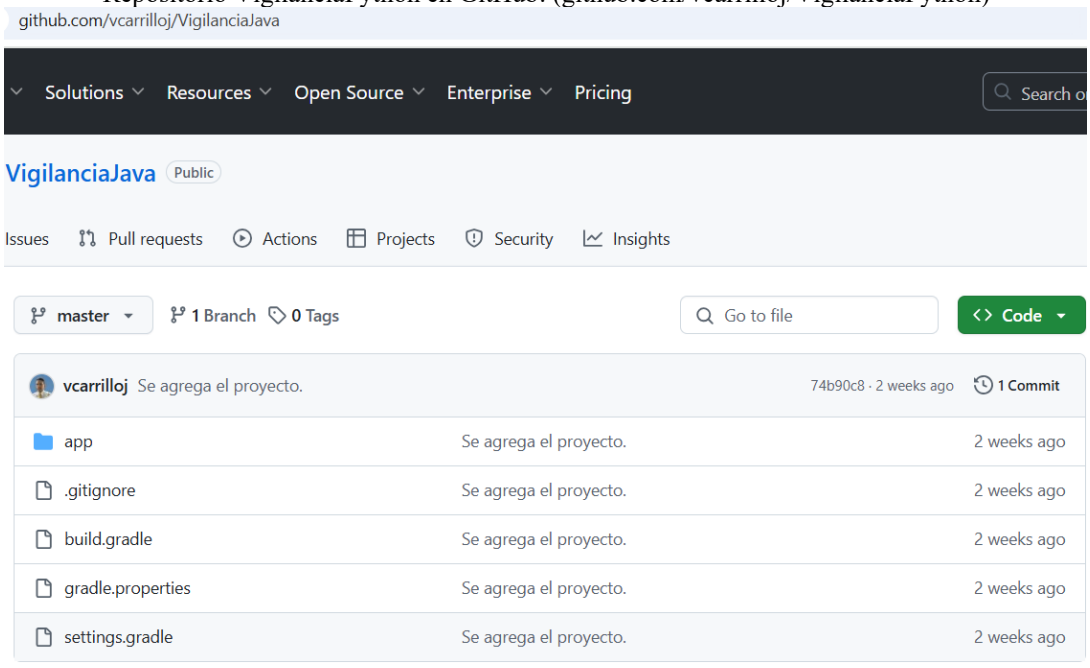


Figura 27. *Requerimiento funcional que permite visualizar las notificaciones y el historial de seguridad. Parte 1.*

ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-012	Crear funcionalidad que permita visualizar las notificaciones de seguridad, con imagen fotográfica. Adicionalmente consultar el historial de las notificaciones de seguridad (historial de notificaciones con imagen fotográfica).	3
TAREAS	Recepción del evento enviado por el programa desarrollado en Phyton (ejecutándose permanentemente en la Raspberry PI), a través de un mensaje (a través de *Fire Cloud Message*), conteniendo la identificación y descripción del Evento y con la URL de la imagen enviada por el DRV (almacenada en la Raspberry PI).	
	Invocación en *Android Studio Koala,* del programa desarrollado en Phyton y a través de la herramienta de exposición de redes *ngrok*: • Obtener la URL pública de ngrok y configurarla para que la app pueda acceder a la información en la Raspberry PI. • Implementar seguridad *API Key* de llamado a *ngrok* (autenticación o tokens si es necesario) desarrollada en Phyton.	
	Implementación del cliente HTTP en la APP para visualizar a la imagen: • Desarrollar una función en la APP que realice solicitudes HTTP a la Raspberry PI mediante la URL de *ngrok*. • Manejar la respuesta y formatear la información para mostrarla en la interfaz de usuario.	
	Configuración del Entorno y Dependencias del proyecto en *Android Studio Koala* para el uso de SQLite (librería build.gradle incluida en Android SDK).	
	Diseño del Modelo de Datos: • Definir la estructura de la base de datos con los campos necesarios. • Definir el almacenamiento de la ruta y recuperación de la imagen (desde el almacenamiento interno o externo del dispositivo). • Crear la Clase para el Manejo de *SQLite*.	
	Implementación de Función para Guardar Datos y de Función para Recuperar Datos.	
	Diseño de la Interfaz de Usuario (UI).	
	Integración con la Interfaz de Registro.	
	Manejo de errores y reconexión • Implementar validaciones para manejar fallos de conexión con *ngrok* o *Firebase*. • Diseñar una estrategia para reintentar la conexión en caso de fallos.	
	Realización de pruebas y optimización: • Probar la APP en diferentes dispositivos y escenarios de acceso a la información. • Validar que los datos se guarden correctamente en la base de datos. • Comprobar la carga de imágenes en la interfaz. • Implementar manejo de errores para evitar fallos en la consulta o guardado. • Mejorar la experiencia del usuario en la visualización de eventos e imágenes.	

Figura 28. *Requerimiento funcional que permite visualizar las notificaciones y el historial de seguridad. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-012	Crear funcionalidad que permita visualizar las notificaciones de seguridad, con imagen fotográfica. Adicionalmente consultar el historial de las notificaciones de seguridad (historial de notificaciones con imagen fotográfica).	3
Criterios de Aprobación		**Aprobado**
La aplicación visualiza en línea la notificación con detalle Tipo, Texto, Fecha/Hora e Imagen del evento.		SI
El usuario puede consultar el historial de notificaciones desde la aplicación, con detalles de cada evento y el estado de entrega de cada alerta (enviada/recibida/error).		SI
La interfaz debe mostrar una lista de eventos de seguridad en orden cronológico (del más reciente al más antiguo), incluyendo la fecha, la hora, y el tipo de evento		SI
El usuario puede aplicar filtros para visualizar eventos específicos, como por tipo de evento, rango de fechas, o por dispositivo específico.		SI
Si ocurre un error al cargar el historial de eventos, el sistema debe mostrar un mensaje de error claro y brindar la opción de intentar nuevamente.		SI
La información del historial de eventos debe ser accesible solo para el usuario autenticado, protegiendo la privacidad y seguridad de los datos de eventos.		SI
El sistema debe cargar el historial de eventos en menos de 20 segundos bajo condiciones normales de red.		SI
Evidencia		

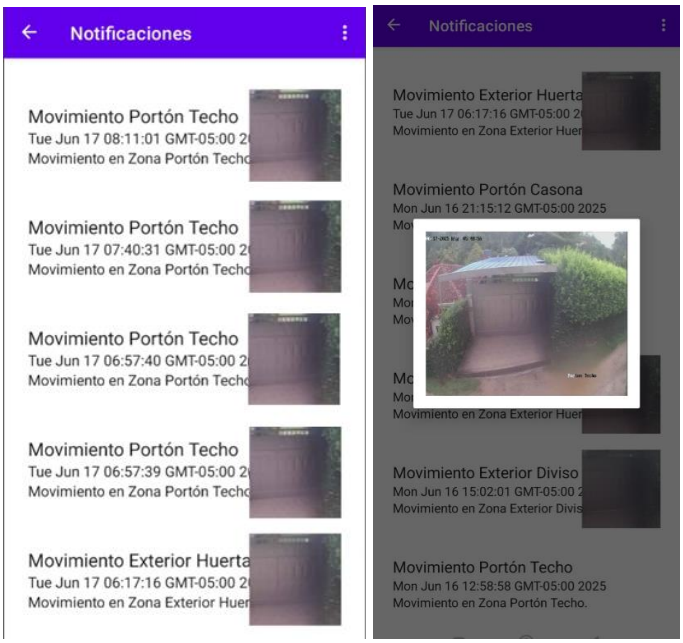


Figura 29. *Requerimiento funcional de que permite consultar los dispositivos registrados en Tuya y asociados al proyecto de Seguridad.*

ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-013	Crear funcionalidad que permita consultar los dispositivos registrados en Tuya y asociados al proyecto de Seguridad	6
TAREAS	Configuración y/o Verificar el Proyecto en la plataforma Tuya • Crear una cuenta en [Tuya IoT Platform](https://iot.tuya.com/). • Configurar un nuevo *Cloud Development Project* en el portal de Tuya. • Obtener las API Keys (Access ID y Access Secret). • Vincular los dispositivos IoT al proyecto.	
	Invocación de la API desarrollada en Phyton en *Android Studio Koala*.	
	Comunicación entre la APP y la Raspberry PI • La Raspberry PI debe ejecutar un programa en Python que monitorea los dispositivos IoT. • exponer un endpoint con el uso de *ngrok*, para que la APP consulte datos directamente desde la Raspberry PI.	
	Desarrollo de la Interfaz en la APP • Crear una pantalla para listar los dispositivos IoT registrados. • Mostrar el estado actual de cada dispositivo (Encendido/Apagado, temperatura, consumo, etc.).	
	Realización de pruebas y optimización: • Verificar que la APP recibe correctamente la información de los servicios expuestos por la Raspberry PI. • Simular eventos y notificaciones de dispositivos IoT para comprobar el monitoreo en tiempo real. • Optimizar la comunicación entre la APP y la Raspberry PI para minimizar latencias.	
Criterios de Aprobación		**Aprobado**
El usuario puede consultar los dispositivos registrados en el proyecto de Tuya administrador por la aplicación desarrollada en Phyton y ejecutándose en la Raspberry PI.		SI
La información los dispositivos registrados en el proyecto de Tuya debe ser accesible solo para el usuario autenticado, protegiendo la privacidad y seguridad de los datos de eventos.		SI
El sistema debe cargar el total de los dispositivos registrados en el proyecto de Tuya en menos de 20 segundos bajo condiciones normales de red.		SI
Evidencia		
		

Figura 30. *Requerimiento funcional para Consultar la parametrización de Dispositivos, Escenario de Dispositivos IoT y Notificaciones.*

ID	Tarea	Estimación (días)
EPICA 4	**Aplicativo Móvil**	
R-014	Consultar la parametrización de Dispositivos, Escenario de Dispositivos IoT y Notificaciones.	6

TAREAS	Configuración en la Raspberry PI:
	• Instalar y configurar el servicio ngrok que permitirá acceso de al APP a la información publicada en la Raspberry PI.
	  ○ Descarga e instala ngrok en la Raspberry PI.
	  ○ Configura ngrok para exponer un puerto específico con un túnel HTTP o HTTPS.
	• Implementar un servicio web en la Raspberry PI
	  ○ Desarrolla un servidor web en la Raspberry usando FastApi (Python) para manejar las solicitudes de datos.
	  ○ Define endpoints para consultar y devolver la información requerida.
	• Ejecutar ngrok para exponer el servicio
	  ○ Ejecutar ngrok.
	  ○ Obtener la URL pública que genera ngrok y úsala para acceder desde la app.	

	Desarrollo en Android Studio:
	• Configurar permisos en Android.
	• Implementar conexión a la API.
	• Desplegar los datos en la interfaz de la APP.
	  ○ Dispositivos y escenario de Dispositivos IoT.
	  ○ Notificaciones de Eventos.
	  ○ Modos de Operación	

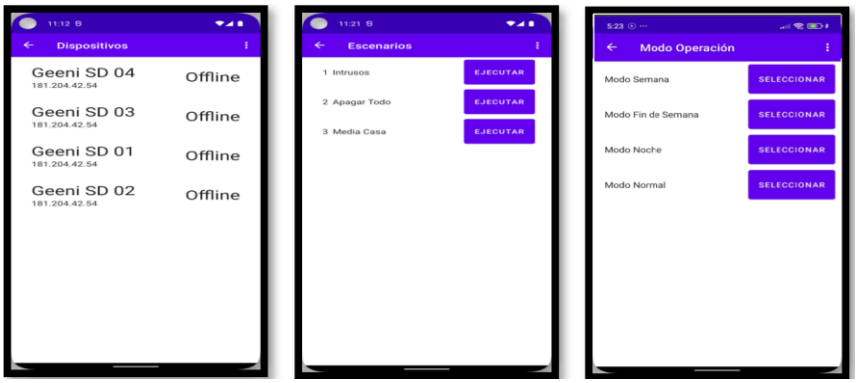
	Seguridad y Mantenimiento: • Actualizar la URL de ngrok dinámicamente. • Implementar manejo de errores, excepciones y validaciones en el código de la APP con la conexión a la API.	
Criterios de Aprobación		**Aprobado**
La interfaz de usuario debe mostrar en una pantalla de forma organizada y visualmente clara, una lista de Dispostivios, Escenario de Dispositivos IoT y Notificaciones de Eventos.		SI
La información de de Dispostivios, Escenario de Dispositivos IoT y Notificaciones de Eventos, debe ser accesible solo para el usuario autenticado, protegiendo la privacidad y seguridad de los datos de eventos.		SI
Se debe permitir seleccionar un Escenario y/o un Modo de Operación y que se ejecute la opción de Ejecución de escenario y/o de Selección de modo de operación.		SI
Evidencia		
		

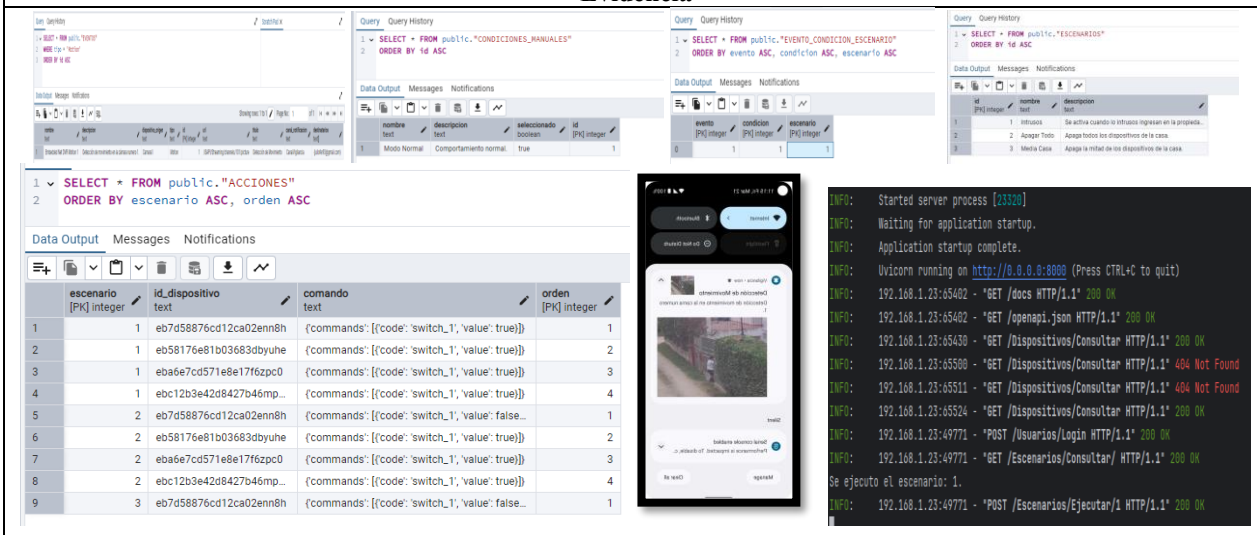
Figura 31. *Requerimiento funcional para monitorear las notificaciones enviadas por el DVR. Parte 1.*

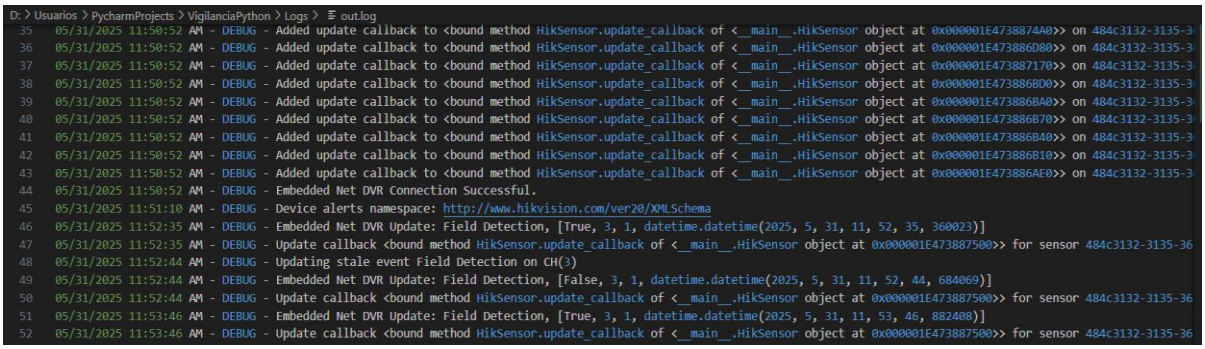
ID	Tarea	Estimación (días)
EPICA 5	**Monitoreo de Seguridad en línea**	
R-015	Monitorear las notificaciones enviadas por el DVR, identificando el evento y realizando de forma automática la notificación a la APP, el envío de correo y la ejecución de acciones de dispositivos IoT según parametrización realizada.	6
TAREAS	Validación del correcto funcionamiento de los requerimientos R-004, R-005, R-006 y R-008: Requerimientos de Control e Integración de Dispositivos IoT. • Identificar desde un aplicativo desarrollado en Python los dispositivos IOT registrados en la plataforma de TUYA. • Activar o desactivar un dispositivo IOT desde un aplicativo desarrollado en Python. Requerimientos de Gestión de Notificaciones y Alertas. • Crear una funcionalidad para enviar notificaciones al dispositivo móvil cuando se detecte movimiento o una persona. • Registrar los eventos de detección de movimiento y envío a la APP construida. • Notificar al usuario a través del correo electrónico cuando se detecte movimiento o personas no autorizadas.	
	Creación de rutina en Phyton para consulta de en estructura de datos creada para los Eventos	
	Integración con la rutina para la consulta de las condiciones parametrizadas e identificación de la condición actual establecida.	
	Integración con la rutina para la consulta del escenario (o escenarios) parametrizados para la Condición actual y el Evento identificado.	
	Integración con la rutina para ejecutar las acciones del escenario (o los escenarios) identificados anteriormente.	
	Realización de pruebas, depuración y Documentación: • Verificar todas las funcionalidades requeridas para el monitoreo. • Realizar pruebas al programa con Eventos ejecutados y Condiciones parametrizadas, verificando correcta ejecución de las acciones de las condiciones parametrizadas. • Documentar el proceso y uso del código para facilitar futuras actualizaciones.	
Criterios de Aprobación		**Aprobado**
El sistema debe monitorear permanentemente la información del DVR para identificar el evento detectado (con base en los Eventos registrados en el sistema).		SI
El sistema debe enviar un mensaje con a la APP construida visualizando el evento en menos de 20 segundos.		SI
El sistema ejecuta las acciones de los dispositivos IoT definidos en los escenarios identificados, en el orden parametrizado y dada la Condición actualmente parametrizada y el evento detectado.		SI

Figura 32. *Requerimiento funcional para monitorear las notificaciones enviadas por el DVR. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 5	**Monitoreo de Seguridad en línea**	
R-015	Monitorear las notificaciones enviadas por el DVR, identificando el evento y realizando de forma automática la notificación a la APP, el envío de correo y la ejecución de acciones de dispositivos IoT según parametrización realizada.	6

Evidencia





Implementación

Programa desarrollado en Python VigilanciaApi.py

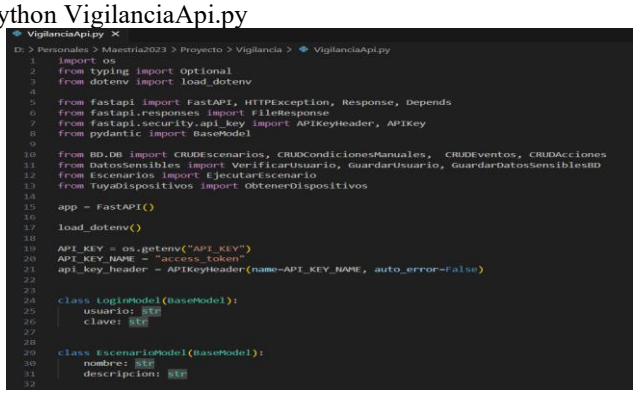
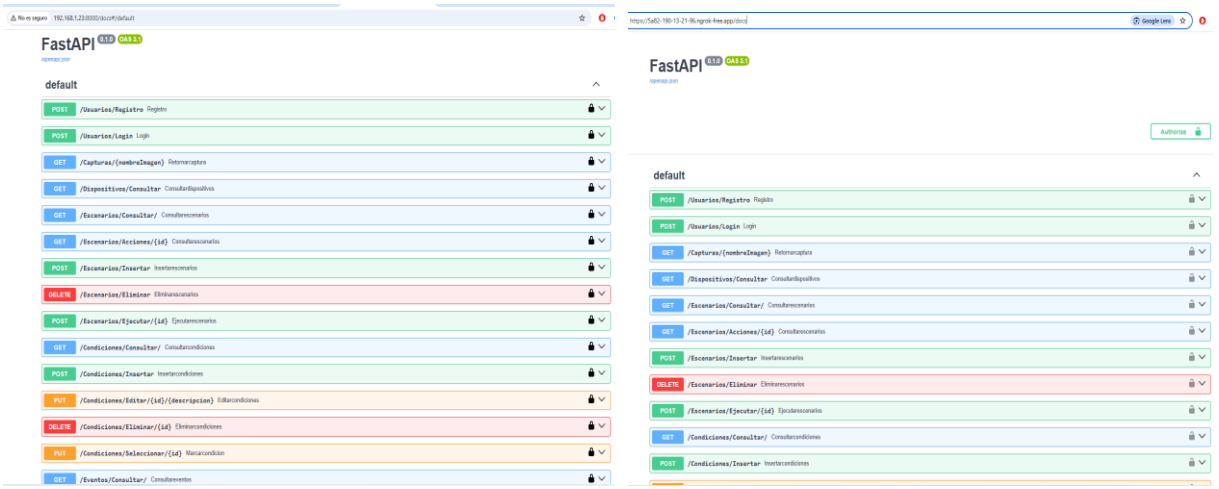


Figura 33. *Requerimiento funcional para la actualización de la parametrización del monitoreo de eventos de seguridad. Parte 1.*

ID	Tarea	Estimación (días)
EPICA 6	**Servicios WEB de Actualización de Parametrización**	
R-016	Actualización de la parametrización de Eventos, Condiciones (del Sistema), Escenario de Dispositivos IoT, Acciones de Escenarios de Dispositivos IoT y asociación de Eventos, Condiciones y Escenarios de Dispositivos IoT.	6
TAREAS	Configuración del Entorno en la Raspberry PI • Instalar Python y dependencias necesarias. • Instalar FastAPI y Uvicorn. • Instalar ngrok y configurar el túnel para exponer el servicio.	
	Implementación del Servicio con FastAPI Diseñar la estructura del API REST. • Crear los modelos de datos para Eventos, Condiciones (del Sistema), Escenario de Dispositivos IoT, Acciones de Escenarios de Dispositivos IoT y asociación de Eventos, Condiciones y Escenarios de Dispositivos IoT. • Definir y diseñar validaciones estructurales de la información de las diferentes entidades. • Implementar *endpoints* para CRUD de cada entidad.Implementar autenticación y seguridad en el servicio (ej. tokens, API keys).	
	Integración con ngrok • Configurar el túnel en ngrok para exponer el servicio. • Implementar reglas de acceso. • Probar la conectividad desde internet al servicio FastAPI.	
	Realización de pruebas, depuración y Documentación: • Realizar pruebas unitarias y funcionales de cada componente. • Implementar el manejo de excepciones en el código para evitar que el programa falle con los errores relacionados • Simular eventos y condiciones para validar la respuesta del sistema. • Monitorear logs y rendimiento del servicio. • Documentar la arquitectura, configuración del sistema y uso del código para facilitar futuras actualizaciones.	
	Despliegue y Mantenimiento • Automatizar el inicio del servicio en la Raspberry PI. • Implementar backups y recuperación en caso de fallos.	

Criterios de Aprobación	Aprobado
Se pueden crear, leer, actualizar y eliminar registros en las entidades de Eventos, Condiciones (del Sistema), Escenario de Dispositivos IoT, Acciones de Escenarios de Dispositivos IoT y asociación de Eventos, Condiciones y Escenarios de Dispositivos IoT.	SI
No se permite la creación de registros duplicados o inconsistentes, con lo cual el servicio valida las asociaciones entre Eventos, Condiciones y Escenarios antes de registrarlas.	SI
El servicio solo permite acceso autenticado mediante tokens o claves de API, con tiempo de respuesta para las operaciones CRUD no mayor a 20 segundos en condiciones normales.	SI

Figura 34. *Requerimiento funcional para la actualización de la parametrización del monitoreo de eventos de seguridad. Parte 2.*

ID	Tarea	Estimación (días)
EPICA 6	**Servicios WEB de Actualización de Parametrización**	
R-016	Actualización de la parametrización de Eventos, Condiciones (del Sistema), Escenario de Dispositivos IoT, Acciones de Escenarios de Dispositivos IoT y asociación de Eventos, Condiciones y Escenarios de Dispositivos IoT.	6
Evidencia		
		
Implementación		
Programa desarrollado en Python DB.py		

```python
import os

from dotenv import load_dotenv
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

from BD.Tablas import CondicionesManuales, Evento, Escenarios, Acciones, EventoCondicionEscenario

load_dotenv()

# Acceder a las variables
database_url = os.getenv("DATABASE_URL")

engine = create_engine(database_url)
Session = sessionmaker(bind=engine)
session = Session()


# Métodos CRUD CONDICIONES_MANUALES
class CRUDCondicionesManuales:
    @staticmethod
    def Insertar(condiciones):
        try:
            for condicon in condiciones:
                nuevo_registro = CondicionesManuales(
                    nombre=condicon.nombre,
                    descripcion=condicon.descripcion,
                    seleccionado=condicon.seleccionado
                )
```

3.3 Fase de Revisión y Retrospectiva

Para llevar a cabo la Sprint Retrospective después de ejecutar los tres (2) sprints, se adoptó la técnica de retrospectiva Start-Stop-Continue. Durante esta sesión, el equipo demostró un compromiso genuino con la escucha activa y la toma de decisiones colaborativas, llegando a diversos acuerdos que reflejan la disposición del equipo para trabajar en conjunto hacia una meta compartida. Los acuerdos establecidos se alinean de manera coherente con los elementos identificados en la técnica, abordando lo que se debe comenzar a hacer (Start), detener (Stop) y continuar (Continue) para mejorar y optimizar el enfoque en los próximos sprints, este proceso reafirma el espíritu de trabajo en equipo y la dedicación del equipo para lograr una mejora continua en la ejecución del proyecto.

Figura 35. Aplicación de r*etrospectiva Empezar – Parar – Continuar de desarrollo SCRUM.*

Empezar	Parar	Continuar
Compartir lecciones aprendidas de los sprint anteriores	No solicitar ayuda cuando NO se posee un conocimiento profundo sobre un tema	Distribución de tareas con base en las capacidades de cada integrante
Realizar reuniones con lapsos de tiempo más cortos	El incumplimiento en el registro del *Scrum Board*	Registro de memorias ejecutivas de las reuniones

Esta dinámica se desarrolla teniendo en cuenta las lecciones aprendidas y las experiencias adquiridas durante la ejecución de cada uno de los sprints en el proyecto, considerando que estas experiencias adquiridas pueden contribuir a maximizar el rendimiento y el compromiso de los integrantes del equipo Scrum en futuros proyectos.

3.4 Fase de Lanzamiento (Release)

En esta fase se lanza una versión prototipo finalizada y lista para validaciones finales. Implica la entrega de un grupo de funcionalidades que representan un avance importante en el total del producto. Su desarrollo requiere planificación, coordinación y comunicación efectivas de todo el equipo de desarrollo. Los procesos obligatorios por realizar en esta fase son los que describen en la siguiente figura:

Figura 36. *Entradas obligatorias, herramientas y salidas para los procesos en fase de Lanzamiento.*

Tomado de [6, p. 16].

4. Resultados

El proyecto logró desarrollar y validar un prototipo funcional de sistema de seguridad integrado basado en tecnologías IoT y detección de movimiento, orientado a fortalecer la seguridad de viviendas ubicadas en zonas rurales y de expansión urbana. A continuación, se describen los principales resultados obtenidos durante el proceso de diseño, implementación y validación del sistema:

- La implementación del sistema de seguridad desarrollado, compuesto por cámaras, sensores, microordenadores y un DVR con funcionalidades inteligentes, tiene un costo aproximado de $3.499.583 COP, lo que representa alrededor de 2.46 salarios mínimos mensuales legales vigentes para el año 2025. Este valor se encuentra dentro del rango de asequibilidad para una familia de clase media en Colombia, la cual, según el DANE, está conformada por personas con ingresos mensuales entre $853.608 y $4.596.352 COP (El Cronista, 2024).

  Considerando un periodo de uso de 5 años, los costos totales proyectados serían:

  Costo inicial: 3.499.583, costos de operación y mantenimiento: 1.253.333 x 5 = 6.266.665 para un costo total a 5 años: 9.766.248. El costo promedio anual: 1.953.249 lo que equivale aproximadamente a 1.37 salarios mínimos anuales.

  Los costos son accesibles comparados con servicios tradicionales de seguridad privada, que pueden superar 6.000.000 COP anuales en zonas rurales.

Figura 37. *Costos detallados del prototipo desarrollado.*

Dispositivo	Cant.	Precio Unit. (COP)	Costo Total (COP)
DVR Hikvision iDS-7216HQHI-M2/S 	1	944.000	944.000
Raspberry PI 5 Starter Kit - 4 GB de RAM 	1	389.085	389.085
Fuente de alimentación de 5.1 V 3A compatible con Raspberry PI 4 	1	38.744	38.744
Interruptor de luz Wi-Fi inteligente, 2 canales, interruptor de relé inalámbrico linptech con aplicación Tuya 	1	86.151	86.151
Amico Luz de seguridad LED inteligente de 40 W con sensor de movimiento, funciona con Alexa/Google Home 	1	135.714	135.714
Enchufe inteligente redondo, con Wi-Fi, color blanco, GN-WW117-199 		86.151	172.302
Cámara de seguridad para exterior 	16	70.000	1.120.000
Instalación cámaras	16	60.000	960.000
Total de Implementación			3.499.583

Tabla 7. *Costo de Operación y Mantenimiento Anual.*

Concepto	Descripción	Costo Estimado Anual COP
Energía eléctrica	Consumo de DVR, cámaras, luces inteligentes y Raspberry Pi	240.000
Servicio de internet básico (para monitoreo remoto)	Plan mínimo requerido de 10 Mbps	600.000
Reemplazo/actualización de dispositivos IoT (cada 3 años, prorrateado)	Estimación de renovación de sensores, interruptores y cámaras	233.333
Mantenimiento preventivo (limpieza, ajustes)	Visita anual técnica	180.000
Total Operación y Mantenimiento Anual		**1.253.333**

- El prototipo final del sistema de seguridad fue desarrollado durante el segundo sprint, integrando diversos componentes tecnológicos en una arquitectura funcional. Se utilizó la plataforma Tuya para facilitar la gestión de dispositivos IoT de diferentes marcas, aprovechando una única librería de programación. Además, se incorporaron funcionalidades avanzadas de los DVRs actuales para la detección eficiente de eventos, evitando la necesidad de desarrollar algoritmos propios de interpretación de imagen. El uso de microordenadores como la Raspberry PI permitió mantener bajos costos sin sacrificar capacidades de procesamiento. La arquitectura refleja la integración efectiva de herramientas, plataformas y tecnologías disponibles para el desarrollo de soluciones IoT aplicadas a la seguridad residencial (como se detalla en la Figura 8. Estructura final del Proyecto (Sprint 2) de desarrollo SCRUM.).

- Como resultado del proyecto, se desarrolló un prototipo funcional de sistema de seguridad basado en tecnologías IoT y detección de movimiento. El sistema incluye una aplicación móvil construida en Android Studio (versión Koala) y una serie de librerías desarrolladas en Python que se ejecutan en una Raspberry PI. Las librerías creadas son:

    1. NotificacionesDVR.py
    2. FirebaseNotificacionesPush.py
    3. VigilanciaApi.py
    4. BD.py

  Durante el desarrollo, cada una de las épicas definidas en la metodología SCRUM detalló las tareas, sus entregables y productos asociados. En los anexos del documento se incluye la guía de instalación del prototipo, con instrucciones paso a paso para su implantación.

- Como parte fundamental del desarrollo del sistema, se elaboró y ejecutó un plan de pruebas para validar el funcionamiento del prototipo. En cada una de las épicas y sus respectivas actividades se definieron criterios de aceptación específicos, los cuales sirvieron como base para verificar el cumplimiento de los requerimientos establecidos.

  El resultado de estas pruebas permitió demostrar que cada funcionalidad desarrollada opera según lo esperado, garantizando la confiabilidad del sistema. Se documentaron los casos de prueba aplicados, los procedimientos seguidos y los resultados obtenidos, los cuales respaldan la efectividad del prototipo como solución de seguridad integrada basada en IoT y detección de movimiento.

5. Conclusiones

El desarrollo del prototipo de sistema de seguridad basado en tecnologías IoT permitió validar la viabilidad técnica y operativa de implementar soluciones de bajo costo para mejorar la seguridad en viviendas ubicadas en zonas rurales y de expansión urbana, como es el caso de la vereda Los Cauchos en Floridablanca, Santander. A continuación, resaltamos las principales conclusiones:

5.1 Eficiencia y conveniencia del uso de la tecnología IoT en la seguridad residencial

La integración (e implementación) de cámaras de seguridad, sensores de movimiento y otros dispositivos IoT, junto con protocolos de comunicación adecuados, demostró ser una solución efectiva para mejorar la seguridad en viviendas con acceso limitado a sistemas tradicionales de vigilancia. La arquitectura del sistema permitió una respuesta rápida y oportuna ante eventos detectados, validando su funcionalidad en un entorno controlado.

5.2 Accesibilidad y relación de costo / beneficio

Se logró diseñar y desarrollar un prototipo de seguridad con bajo costo, ofreciendo una alternativa viable para familias de clase media, que normalmente no pueden acceder a soluciones comerciales de alto costo.

La implementación de dispositivos IoT económicos y la integración con plataformas como Tuya facilitaron la escalabilidad del sistema sin comprometer su eficiencia.

5.3 Eficiencia del modelo de notificación / respuesta

El sistema logró integrar satisfactoriamente la captura de eventos mediante sensores y cámaras, enviando notificaciones en tiempo real a través de una aplicación móvil. Las alertas, acompañadas de imágenes y datos relevantes, permitieron al usuario tomar decisiones informadas. El tiempo de respuesta del sistema fue adecuado para el contexto rural, validando su utilidad en la práctica.

5.4 Optimización mediante integración de tecnologías preexistentes

Durante las pruebas iniciales del sistema, se implementó un script en Python para realizar el reconocimiento de personas a partir de imágenes capturadas por una cámara. Si bien el enfoque demostró ser funcional, presentó limitaciones significativas en términos de escalabilidad, ya que solo permitía el análisis en una cámara de manera simultánea y requería recursos computacionales locales considerables.

Posteriormente, al evaluar las capacidades nativas del DVR utilizado, se identificó que este contaba con funciones integradas de reconocimiento de personas, aplicables a múltiples canales de video en paralelo. Esta funcionalidad permitió detectar eventos de intrusión de forma rápida y efectiva, generando alertas automatizadas y transmitiendo los eventos como mensajes estructurados al sistema de notificación.

Este hallazgo evidenció que, en proyectos orientados a entornos de producción o prototipos funcionales, resulta más eficiente integrar tecnologías ya implementadas y probadas —como las funciones inteligentes de los DVR comerciales— que desarrollar desde cero componentes que ya están disponibles y optimizados en el mercado. La reutilización e integración de soluciones

tecnológicas robustas permite reducir tiempos de desarrollo, minimizar errores y mejorar la fiabilidad del sistema sin comprometer su personalización o capacidad de respuesta.

5.5 Metodología SCRUM y desarrollo ágil

El uso de SCRUM como metodología de desarrollo permitió una evolución iterativa del prototipo, facilitando la validación progresiva de funcionalidades en cada sprint. La organización del proyecto en épicas y requisitos específicos mejoró la gestión del tiempo, recursos y tareas técnicas, asegurando un producto funcional al cierre del ciclo de desarrollo.

5.6 Impacto en la seguridad de áreas de expansión urbana

La solución propuesta no solo aborda el problema de inseguridad en viviendas aisladas, sino que también puede adaptarse a comunidades rurales y conjuntos residenciales sin vigilancia constante.

La escalabilidad del sistema permite su potencial aplicación en otros sectores con condiciones similares, contribuyendo a la reducción de riesgos de intrusión.

5.7 Retos, Limitaciones, Proyecciones y Futuras Mejoras Identificados

La efectividad del sistema depende en gran medida de la estabilidad de la conectividad a internet en áreas rurales, lo que podría requerir futuras mejoras en la redundancia de comunicación.

La responsabilidad de la detección de los eventos se traslada al DVR, el cual en versiones futuras y asegurando los protocolos de comunicación (ISAPI, Intelligent Security API), mejoras en la detección y funcionalidades nuevas serán asimiladas por el proyecto de forma trasparente.

Sin impactar el prototipo desarrollado, la integración con sistemas de respaldo de energía y comunicación satelital podría mejorar la confiabilidad del sistema en zonas con baja infraestructura tecnológica.

En conclusión, el proyecto logró desarrollar un prototipo funcional de seguridad basado en IoT, realizando monitoreo permanente y tomando decisiones automáticas definidas por el usuario, proporcionando una alternativa viable para la seguridad de una vivienda. Se espera que proyecto sirva como base para ideas y desarrollos similares que integren dispositivos IoT, microordenadores, aplicaciones móviles, entre otros elementos; trabajando sistemáticamente para un propósito común.

6. Trabajos Futuros

Con base en los resultados obtenidos durante el desarrollo e implementación del prototipo de sistema de seguridad basado en IoT, se presentan las siguientes recomendaciones y líneas de proyección para futuros desarrollos y aplicaciones

6.1 Mejora de la infraestructura de comunicación

- Se recomienda implementar mecanismos de redundancia en la conectividad, como el uso combinado de redes móviles (3G/4G/5G), enlaces satelitales o enlaces de radiofrecuencia en zonas donde el acceso a Internet es intermitente o inexistente.
- Se sugiere incorporar protocolos de gestión de fallos que aseguren la continuidad operativa del sistema ante interrupciones de red, evitando pérdidas de datos críticos o fallos en las notificaciones.

6.2 Integración con sistemas de respaldo energético

- Para mitigar los efectos de fallos eléctricos comunes en zonas rurales, se recomienda la inclusión de fuentes de energía alterna como paneles solares, UPS o baterías inteligentes, que mantengan el sistema operativo durante cortes prolongados.
- El sistema puede beneficiarse de algoritmos de gestión energética que prioricen funciones críticas, permitiendo un uso más eficiente de los recursos disponibles.

6.3 Integración comunitaria y enfoque colaborativo

- Se recomienda explorar modelos colaborativos entre vecinos o comunidades rurales para implementar sistemas compartidos que permitan monitoreo conjunto, reduciendo costos y aumentando la cobertura.
- La creación de redes de seguridad comunitaria conectadas a través del sistema podría facilitar la detección temprana de situaciones anómalas y la coordinación de respuestas.

6.4 Evaluación de interoperabilidad con sistemas institucionales

Para aumentar el alcance del sistema, se sugiere establecer mecanismos de interoperabilidad con centros de control locales o regionales (policía, bomberos, defensa civil), mediante estándares de comunicación abiertos y seguros.

6.5 Consideraciones éticas y de privacidad

Dado que el sistema involucra la captura y transmisión de imágenes, se recomienda establecer lineamientos claros sobre privacidad, uso de datos, consentimiento y

almacenamiento seguro de la información de manera que el sistema sea percibido como una herramienta de protección y no de vigilancia invasiva.

7. Acerca de los apéndices

En el apéndice se incluye el Manual de Instalación del Sistema de Seguridad Integrado (Apéndice A), el cual describe paso a paso el proceso requerido para implementar correctamente todos los componentes del prototipo. Este manual cubre aspectos como la instalación física de los equipos, la configuración del DVR, la conexión con la Raspberry PI, la ejecución de los scripts Python, la exposición de servicios mediante NGROK, la integración con Firebase y Tuya, y la instalación y uso de la aplicación móvil. Su propósito es servir como guía técnica detallada para el despliegue y puesta en marcha del sistema en ambientes reales o pilotos. También se incluye el Manual de Usuario de la APP construida (Apéndice B), el cual documenta el uso de la aplicación móvil construida.

Adicionalmente se incluye reporte detallado del repositorio programado en Python para la Raspberry Pi denominado VigilanciaPython (Apéndice C) y del repositorio correspondiente al desarrollo de la aplicación móvil para Android denominada VigilanciaJava (Apéndice D), estos apéndices detallan los archivos y carpetas que conforman el desarrollo, incluyendo sus tamaños y permisos de acceso. Adicionalmente se reporta el número de líneas de los archivos relacionados.

Referencias

[1] Hikvision, "Documentación oficial del protocolo ISAPI", Hikvision, s.f. Disponible en: https://www.hikvision.com/en/support/tools/digicap/

[2] ITU-T, "Recommendation H.264: Advanced video coding for generic audiovisual services", ITU, s.f. Disponible en: https://www.itu.int/rec/T-REC-H.264

[3] Firebase, "Firebase Cloud Messaging (FCM)", Firebase Docs, s.f. Disponible en: https://firebase.google.com/docs/cloud-messaging?hl=es-419

[4] Ngrok, "Documentación de Ngrok: túneles seguros para servicios locales", Ngrok, s.f. Disponible en: https://ngrok.com/docs

[5] Tiangolo, "FastAPI: Fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints", s.f. Disponible en: https://fastapi.tiangolo.com

[6] Universitat de Lleida, "¿Qué es un prototipo?", *Universitat de Lleida*, s.f. Disponible en: https://mpiua.invid.udl.cat/fases-mpiua/prototipado/que-es-un-prototipo/

[7] Freed Tools, "Prototipo", *Freed Tools*, s.f. Disponible en: https://freed.tools/blogs/ux-cx/prototipo

[8] A. Cohen, "¿En qué consiste un sistema integrado de seguridad?", *LinkedIn*, 10-feb-2023. Disponible en: https://www.linkedin.com/pulse/en-qu%C3%A9-consiste-un-sistema-integrado-de-seguridad-ariel-cohen/?originalSubdomain=es

[9] Blog Siete24, "La importancia de un sistema integrado de gestión en seguridad privada", *Blog Siete24*, 11-ago-2023. Disponible en: https://blog.siete24.com/la-importancia-de-un-sistema-integrado-de-gesti%C3%B3n-en-seguridad-privada

[10] Departamento Nacional de Planeación, *Cartilla de expansión* [Documento PDF], 2017. Disponible en: https://portalterritorial.dnp.gov.co/KitOT/Content/uploads/Cartilla%20Expansion.pdf

[11] SCRUMstudy, *A Guide to the Scrum Body of Knowledge (SBOK Guide)*, 2016. Disponible en: https://www.scrumstudy.com/SBOK/6/en/

[12] HogarSense, "Seguridad y domótica", *HogarSense*, 2-feb-2023. Disponible en: https://www.hogarsense.es/domotica/seguridad-domotica

[13] El Cronista, "¿Cuánto debe ganar una persona para ser clase media en 2025?", *El Cronista*, 2-abr-2024. Disponible en: https://www.cronista.com/colombia/finanzas-y-economia/cuanto-debe-ganar-una-persona-para-ser-clase-media-en-2025/

[14] Tuya, "Query Devices in Project", *Tuya Developer*, s.f. Disponible en: https://developer.tuya.com/en/docs/cloud/734e8088a6?id=Kcspwthd1f5tb

[15] Congreso de Colombia, *Ley 1581 de 2012: Por la cual se dictan disposiciones generales para la protección de datos personales*, Diario Oficial No. 48.587, 17-oct-2012. Disponible en: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

[16] Superintendencia de Vigilancia y Seguridad Privada, *Resolución 1384 de 2000: Por la cual se establecen los requisitos para el uso de cámaras de vigilancia*. Disponible en: https://www.supervigilancia.gov.co/documents/20143/49486/Resolucion+1384+de+2000.pdf

[17] Superintendencia de Vigilancia y Seguridad Privada, "Normativa general del sector", *Supervigilancia*, s.f. Disponible en: https://www.supervigilancia.gov.co/

Apéndices

Apéndice A. Manual de Instalación del Sistema de Seguridad Integrado

Instalar PosgreSQL Docker

1. Instalar Docker y Docker Compose, en consola:
   ```
   curl -sSL https://get.docker.com | sh
   sudo usermod -aG docker $USER
   ```
2. Reiniciar.
3. En un directorio crear el archivo docker-compose.yml con el siguiente contenido:

   ```
   version: '3.8'

   services:
     postgres:
       image: arm64v8/postgres:latest
       container_name: postgres-rpi
       restart: unless-stopped
       environment:
         POSTGRES_USER: miusuario
         POSTGRES_PASSWORD: miclave
         POSTGRES_DB: midatabase
       ports:
         - "5432:5432"
       volumes:
         - postgres_data:/var/lib/postgresql/data

   volumes:
     postgres_data:
   ```

4. En consola ubicarse en el directorio y ejecutar el comando

   ```
   docker-compose up -d
   ```

5. Conectarse a la base creada y ejecutar el archivo Vigilancia.sql.

Instalar Python

1. En una consola, actualizar el sistema:

   ```
   sudo apt update && sudo apt upgrade -y
   ```

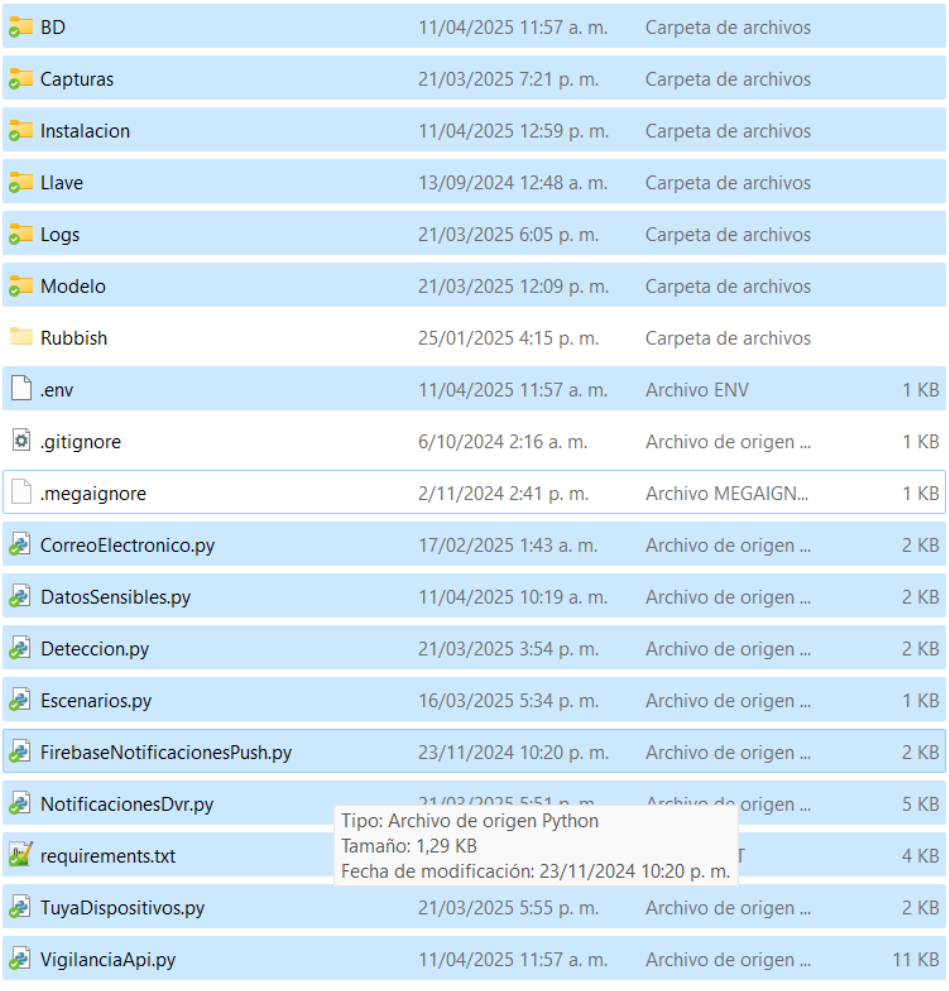
2. Instalar Python

```
sudo apt install python3 -y
```

3. Instalar pip y venv

```
sudo apt install python3-pip python3-venv -y
```

4. Crear directorio para el proyecto y ubicar los archivos del proyecto.



5. Crear un entorno virtual en el directorio.

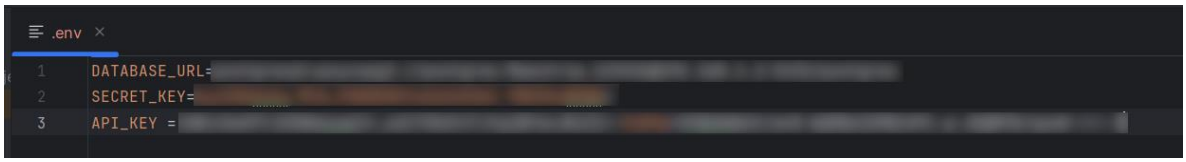
```
cd mi_proyecto
python3 -m venv venv
```

6. Activar el entorno virtual e instalar las librerías del proyecto (archivo requirements.txt).

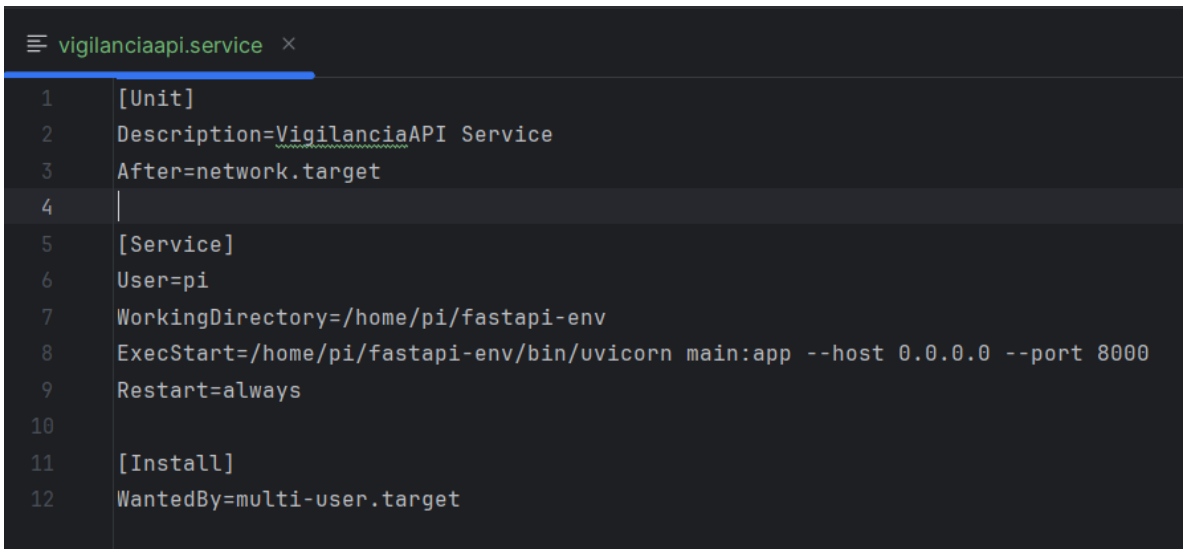
```
source venv/bin/activate
pip install -r requirements.txt
```

Desplegar Api VigilanciaApi

1. Modificar el archivo ingresando los siguientes valores:
    a. DATABASE_URL: Información de conexión con la Base de Datos.
    b. SECRET_KEY: Llave con la que se encriptan los datos Sensibles.
    c. API_KEY: Llave de validación de acceso al API.



2. Copiar el archivo vigilanciaapi.service en el directorio */etc/systemd/*.
3. Modifique los campos *User, WorkingDirectory* y *ExecStart* según la configuración de la raspberry PI y del entorno virtual creado previamente.



4. Recargar el systemd:

    sudo systemctl daemon-reload

    sudo systemctl enable vigilanciaapi

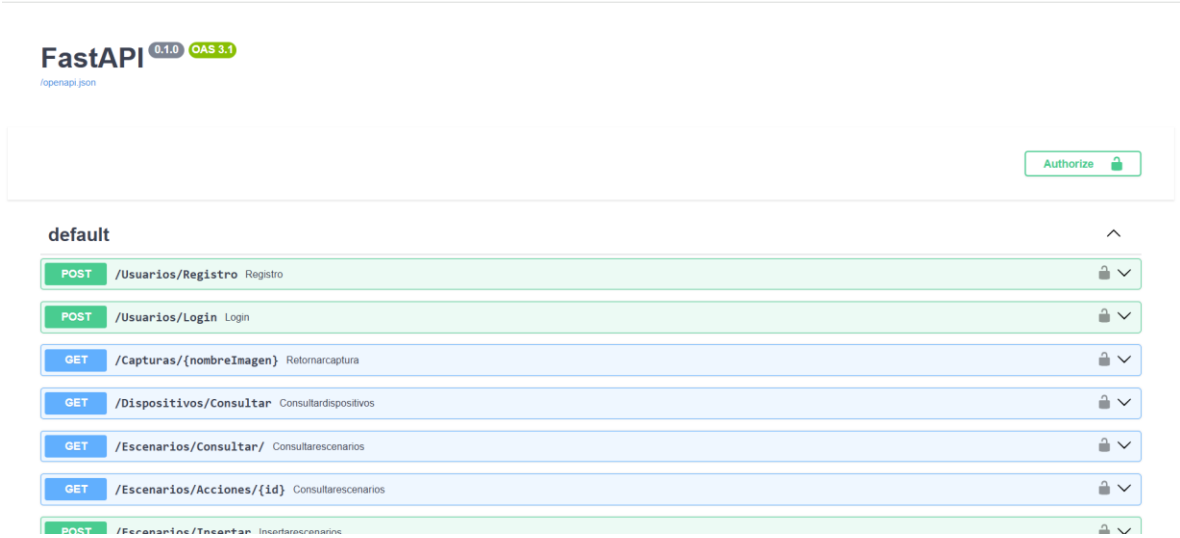
    sudo systemctl start vigilanciaapi

5. Verificar el despliegue del servicio:

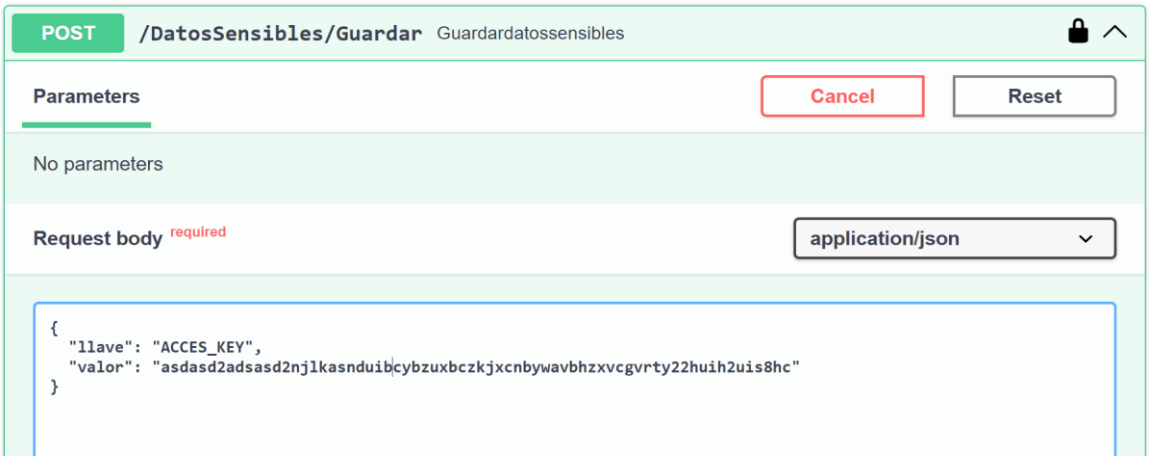
    sudo systemctl status vigilanciaapi

Ingresar Datos Sensibles

1. Ingresar al API VigilanciaApi.

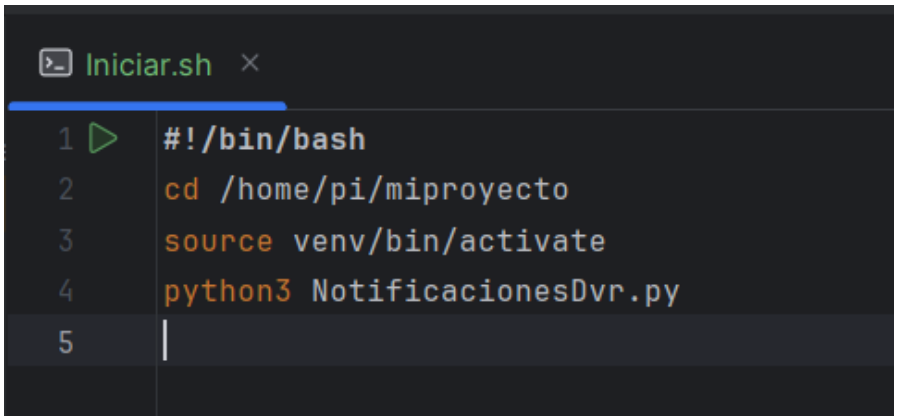


2. Usar el endpoint */DatosSensibles/Guardar* para guardar los siguientes datos sensibles:
    a. IP_DVR: Dirección IP del DVR
    b. USUARIO_DVR: Usuario para conectarse al DVR.
    c. CLAVE_DVR: Clave de acceso al DVR.
    d. ACCESS_ID: ID de acceso a plataforma TUYA.
    e. ACCESS_KEY: Llave de acceso plataforma TUYA.



Agregar Tarea Notificaciones DVR

1. Copiar el archivo Iniciar.sh en una ruta.
2. Editar el archivo según la ruta del entorno virtual previamente creado.



3. Darle permisos de ejecución con el siguiente comando:

   chmod +x /home/pi/miproyecto/Iniciar.sh

4. Abrir una terminar y editar crontab

   crontab -e

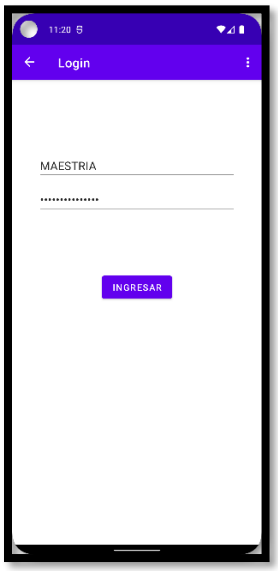
5. Agregar la siguiente línea al final.

   @reboot /usr/bin/python3 /home/pi/miproyecto/Iniciar.sh

Apéndice B. ***Manual de Usuario de la APP***

1. Ingreso a la App:

   Para ingresar a la APP se requiere ingresar el usuario y la contraseña la cual es asignada por el administrador del sistema.



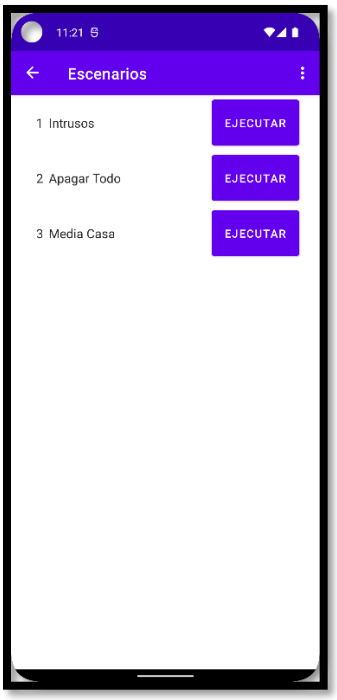
2. Menú Principal:

   El sistema cuenta con las siguientes opciones

   a. Ejecutar Escenarios:

Permite ejecutar los escenarios predefinidos, se han parametrizado para el prototipo tres escenarios:

1. *Intrusos**:** Se encarga de encender todos los dispositivos.*
2. *Apagar Todo**:*** Se encarga de apagar todos los dispositivos.
3. *Media Casa: Apaga la mitad de los dispositivos de la casa.*

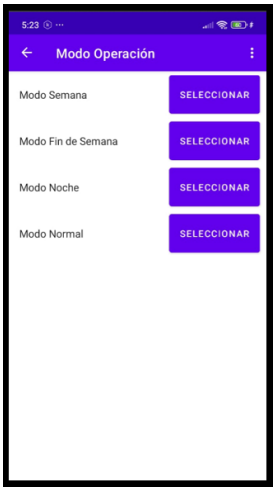


b. Consultar Dispositivos:

Permite consultar los dispositivos integrados al sistema, estos dispositivos se encuentran registrados en la plataforma de Tuya.

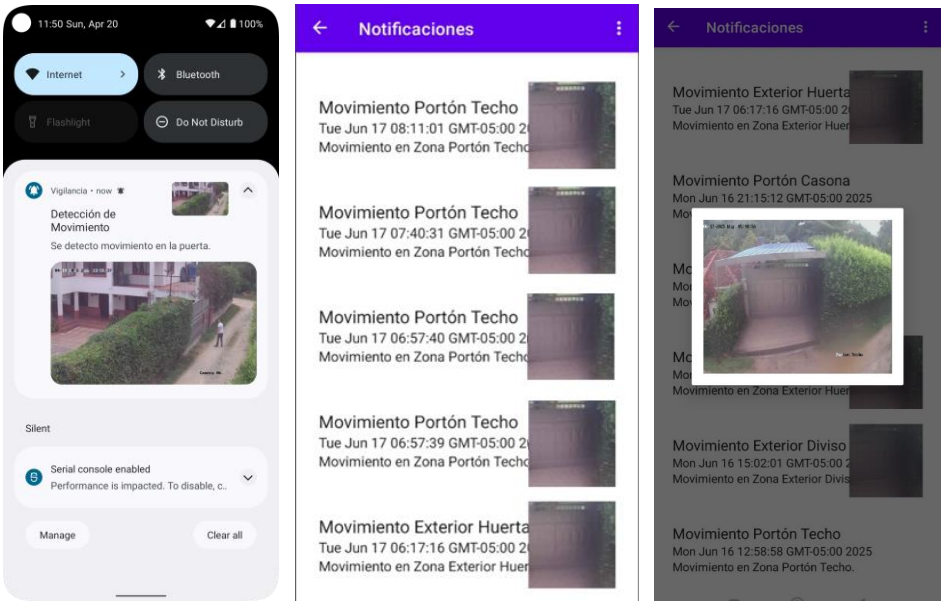
c. Modo Operación

Permite seleccionar el Modo de Operación, que se encuentra el sistema actualmente, este modo es parte de la parametrización que el monitoreo de vigilancia utiliza, junto con el evento que ocurre, para determinar los escenarios de dispositivos que ejecutará de forma automática.



d. Notificaciones:

Permite consultar las notificaciones que fueron recibidas en el dispositivo móvil.



Apéndice C. ***Proyecto desarrollado con Python en Raspberry Pi***

A continuación, se presenta el enlace al repositorio correspondiente al desarrollo programado en Python para la Raspberry Pi, denominado VigilanciaPython:

Repositorio VigilanciaPython en GitHub

(github.com/vcarrilloj/VigilanciaPython)

Se incluye un reporte detallado del repositorio implementado, se detallan los archivos y carpetas que conforman el desarrollo, incluyendo sus tamaños y permisos de acceso. Adicionalmente se reporta el número de líneas de los archivos relacionados:

```
ESTRUCTURA DE ARCHIVOS Y CARPETAS (DETALLADO)
.:
total 58K
-rw-r--r-- 1.4K  ApiDemo.py
drwxr-xr-x    0  BD
-rw-r--r-- 1.4K  CorreoElectronico.py
-rw-r--r-- 1.5K  DatosSensibles.py
-rw-r--r-- 2.4K  Deteccion.py
drwxr-xr-x    0  Doc
-rw-r--r--  409  Escenarios.py
-rw-r--r-- 1.3K  FirebaseNotificacionesPush.py
drwxr-xr-x    0  Instalacion
drwxr-xr-x    0  Modelo
```

```
-rw-r--r-- 4.9K  NotificacionesDvr.py
-rw-r--r-- 1.9K  TuyaDispositivos.py
-rw-r--r--  11K  VigilanciaApi.py
-rwxr-xr-x 1.6K  generar_reporte.sh
-rw-r--r--  178  reporte.txt
-rw-r--r--  753  reporte0.txt
-rw-r--r-- 3.6K  requirements.txt

./BD:
total 28K
-rw-r--r--  11K  DB.py
-rw-r--r-- 2.2K  Tablas.py
-rw-r--r--  11K  Vigilancia.sql

./Doc:
total 372K
-rw-r--r-- 371K  Documento Instalacion.docx

./Instalacion:
total 2.0K
-rwxr-xr-x  91  Iniciar.sh
-rw-r--r-- 261  vigilanciaapi.service
./Modelo:
total 7.0M
-rw-r--r-- 7.0M  efficientdet_lite0.tflite

NÚMERO DE LÍNEAS POR ARCHIVO
2     ./.env
13    ./.git/config
1     ./.git/description
1     ./.git/HEAD
15    ./.git/hooks/applypatch-msg.sample
24    ./.git/hooks/commit-msg.sample
174   ./.git/hooks/fsmonitor-watchman.sample
8     ./.git/hooks/post-update.sample
14    ./.git/hooks/pre-applypatch.sample
49    ./.git/hooks/pre-commit.sample
13    ./.git/hooks/pre-merge-commit.sample
53    ./.git/hooks/pre-push.sample
169   ./.git/hooks/pre-rebase.sample
24    ./.git/hooks/pre-receive.sample
42    ./.git/hooks/prepare-commit-msg.sample
78    ./.git/hooks/push-to-checkout.sample
77    ./.git/hooks/sendemail-validate.sample
128   ./.git/hooks/update.sample
8     ./.git/index
6     ./.git/info/exclude
1     ./.git/logs/HEAD
1     ./.git/logs/refs/heads/master
1     ./.git/logs/refs/remotes/origin/HEAD
3     ./.git/objects/pack/pack-76b9ad8464c558fa6b9435f1fc68a2c0c9545ae3.idx
24379 ./.git/objects/pack/pack-76b9ad8464c558fa6b9435f1fc68a2c0c9545ae3.pack
2     ./.git/objects/pack/pack-76b9ad8464c558fa6b9435f1fc68a2c0c9545ae3.rev
2     ./.git/packed-refs
```

```
1     ./.git/refs/heads/master
1     ./.git/refs/remotes/origin/HEAD
48    ./ApiDemo.py
319   ./BD/DB.py
61    ./BD/Tablas.py
399   ./BD/Vigilancia.sql
38    ./CorreoElectronico.py
52    ./DatosSensibles.py
61    ./Deteccion.py
1049  ./Doc/Documento Instalacion.docx
11    ./Escenarios.py
48    ./FirebaseNotificacionesPush.py
58    ./generar_reporte.sh
3     ./Instalacion/Iniciar.sh
11    ./Instalacion/vigilanciaapi.service
17051 ./Modelo/efficientdet_lite0.tflite
149   ./NotificacionesDvr.py
91    ./reporte.txt
47    ./reporte0.txt
93    ./requirements.txt
59    ./TuyaDispositivos.py
324   ./VigilanciaApi.py
```

Apéndice D. ***Proyecto de aplicación desarrollado en Android Studio (versión Koala)***

A continuación, se presenta el enlace al repositorio correspondiente al desarrollo de la aplicación móvil para Android, denominada VigilanciaJava:

Repositorio VigilanciaJava en GitHub

(github.com/vcarrilloj/VigilanciaJava)

Se incluye un reporte detallado del repositorio implementado, se detallan los archivos y carpetas que conforman el desarrollo, incluyendo sus tamaños y permisos de acceso. Adicionalmente se reporta el número de líneas de los archivos relacionados:

```
ESTRUCTURA DE ARCHIVOS Y CARPETAS (DETALLADO)
.:
total 15K
drwxr-xr-x     0  app
-rw-r--r--   304  build.gradle
-rwxr-xr-x  1.6K  generar_reporte.sh
-rw-r--r--  1.4K  gradle.properties
-rw-r--r--     0  reporte.txecho
-rw-r--r--   178  reporte.txt
-rw-r--r--  3.8K  reporte0.txt
-rw-r--r--   350  settings.gradle
```

```
./app:
total 8.0K
-rw-r--r--  2.2K  build.gradle
-rw-r--r--   770  proguard-rules.pro
drwxr-xr-x     0  src

./app/src:
total 0
drwxr-xr-x  0  androidTest
drwxr-xr-x  0  main
drwxr-xr-x  0  test

./app/src/androidTest:
total 0
drwxr-xr-x  0  java

./app/src/androidTest/java:
total 0
drwxr-xr-x  0  com

./app/src/androidTest/java/com:
total 0
drwxr-xr-x  0  vigilancia

./app/src/androidTest/java/com/vigilancia:
total 0
drwxr-xr-x  0  maestria

./app/src/androidTest/java/com/vigilancia/maestria:
total 4.0K
-rw-r--r--  785  ExampleInstrumentedTest.java

./app/src/main:
total 8.0K
-rw-r--r--  1.7K  AndroidManifest.xml
drwxr-xr-x     0  java
drwxr-xr-x     0  res

./app/src/main/java:
total 0
drwxr-xr-x  0  com

./app/src/main/java/com:
total 0
drwxr-xr-x  0  vigilancia

./app/src/main/java/com/vigilancia:
total 4.0K
drwxr-xr-x  0  maestria

./app/src/main/java/com/vigilancia/maestria:
total 16K
drwxr-xr-x     0  Adaptadores
```

```
drwxr-xr-x     0  Bd
drwxr-xr-x     0  Comun
drwxr-xr-x     0  Fragments
drwxr-xr-x     0  Globales
drwxr-xr-x     0  Json
-rw-r--r--  4.5K  MainActivity.java
-rw-r--r--  4.4K  MyFirebaseMessagingService.java

./app/src/main/java/com/vigilancia/maestria/Adaptadores:
total 12K
-rw-r--r--  2.1K  DispositivosAdapter.java
-rw-r--r--  4.0K  EscenariosAdapter.java
-rw-r--r--  3.2K  NotificacionesAdapter.java

./app/src/main/java/com/vigilancia/maestria/Bd:
total 10K
-rw-r--r--   407  Converters.java
-rw-r--r--  1.4K  Notificacion.java
-rw-r--r--   520  NotificacionDao.java
-rw-r--r--  1.1K  NotificacionesDatabase.java

./app/src/main/java/com/vigilancia/maestria/Comun:
total 8.0K
-rw-r--r--  1.5K  SecureStorageUtil.java
-rw-r--r--  1.2K  Utilidades.java

./app/src/main/java/com/vigilancia/maestria/Fragments:
total 28K
-rw-r--r--  3.4K  DispositivosFragment.java
-rw-r--r--  3.8K  EscenariosFragment.java
-rw-r--r--  5.5K  LoginFragment.java
-rw-r--r--  2.3K  NotificacionesFragment.java
-rw-r--r--  4.1K  Principal.java

./app/src/main/java/com/vigilancia/maestria/Globales:
total 1.0K
-rw-r--r--  280  App.java

./app/src/main/java/com/vigilancia/maestria/Json:
total 14K
-rw-r--r--   419  Accion.java
-rw-r--r--   519  Command.java
-rw-r--r--  4.8K  Dispositivos.java
-rw-r--r--  1.2K  Escenarios.java

./app/src/main/res:
total 4.0K
drwxr-xr-x  0  drawable
drwxr-xr-x  0  drawable-hdpi
drwxr-xr-x  0  drawable-mdpi
drwxr-xr-x  0  drawable-v24
drwxr-xr-x  0  drawable-xhdpi
drwxr-xr-x  0  drawable-xxhdpi
drwxr-xr-x  0  drawable-xxxhdpi
```

```
drwxr-xr-x  0  layout
drwxr-xr-x  0  menu
drwxr-xr-x  0  mipmap-anydpi-v26
drwxr-xr-x  0  mipmap-hdpi
drwxr-xr-x  0  mipmap-mdpi
drwxr-xr-x  0  mipmap-xhdpi
drwxr-xr-x  0  mipmap-xxhdpi
drwxr-xr-x  0  mipmap-xxxhdpi
drwxr-xr-x  0  navigation
drwxr-xr-x  0  values
drwxr-xr-x  0  values-land
drwxr-xr-x  0  values-night
drwxr-xr-x  0  values-w1240dp
drwxr-xr-x  0  values-w600dp
drwxr-xr-x  0  xml

./app/src/main/res/drawable:
total 12K
-rw-r--r--  5.7K  ic_launcher_background.xml
-rw-r--r--  2.5K  icononotificacion.png

./app/src/main/res/drawable-hdpi:
total 4.0K
-rw-r--r--  1.1K  icononotificacion.png

./app/src/main/res/drawable-mdpi:
total 1.0K
-rw-r--r--  616  icononotificacion.png

./app/src/main/res/drawable-v24:
total 4.0K
-rw-r--r--  1.7K  ic_launcher_foreground.xml

./app/src/main/res/drawable-xhdpi:
total 4.0K
-rw-r--r--  1.2K  icononotificacion.png

./app/src/main/res/drawable-xxhdpi:
total 4.0K
-rw-r--r--  2.1K  icononotificacion.png

./app/src/main/res/drawable-xxxhdpi:
total 4.0K
-rw-r--r--  2.5K  icononotificacion.png

./app/src/main/res/layout:
total 33K
-rw-r--r--  1.1K  activity_main.xml
-rw-r--r--   927  content_main.xml
-rw-r--r--  1.6K  fragment_dispositivos.xml
-rw-r--r--   658  fragment_dispositivos_list.xml
-rw-r--r--  1.3K  fragment_escenarios.xml
-rw-r--r--   652  fragment_escenarios_list.xml
-rw-r--r--   400  fragment_image.xml
```

```
-rw-r--r--  3.5K  fragment_login.xml
-rw-r--r--  1.6K  fragment_notificaciones.xml
-rw-r--r--   672  fragment_notificaciones_list.xml
-rw-r--r--  1.6K  fragment_principal.xml
-rw-r--r--   435  popup_image.xml

./app/src/main/res/menu:
total 1.0K
-rw-r--r--  412  menu_main.xml

./app/src/main/res/mipmap-anydpi-v26:
total 2.0K
-rw-r--r--  276  ic_launcher.xml
-rw-r--r--  276  ic_launcher_round.xml

./app/src/main/res/mipmap-hdpi:
total 8.0K
-rw-r--r--  1.4K  ic_launcher.webp
-rw-r--r--  2.9K  ic_launcher_round.webp

./app/src/main/res/mipmap-mdpi:
total 8.0K
-rw-r--r--   982  ic_launcher.webp
-rw-r--r--  1.8K  ic_launcher_round.webp

./app/src/main/res/mipmap-xhdpi:
total 8.0K
-rw-r--r--  1.9K  ic_launcher.webp
-rw-r--r--  3.9K  ic_launcher_round.webp

./app/src/main/res/mipmap-xxhdpi:
total 12K
-rw-r--r--  2.9K  ic_launcher.webp
-rw-r--r--  5.8K  ic_launcher_round.webp

./app/src/main/res/mipmap-xxxhdpi:
total 12K
-rw-r--r--  3.8K  ic_launcher.webp
-rw-r--r--  7.6K  ic_launcher_round.webp

./app/src/main/res/navigation:
total 4.0K
-rw-r--r--  2.6K  nav_graph.xml

./app/src/main/res/values:
total 10K
-rw-r--r--   387  colors.xml
-rw-r--r--   301  dimens.xml
-rw-r--r--  1.5K  strings.xml
-rw-r--r--  1.2K  themes.xml

./app/src/main/res/values-land:
total 1.0K
-rw-r--r--  68  dimens.xml
```

```
./app/src/main/res/values-night:
total 4.0K
-rw-r--r--  834  themes.xml

./app/src/main/res/values-w1240dp:
total 1.0K
-rw-r--r--  69  dimens.xml

./app/src/main/res/values-w600dp:
total 1.0K
-rw-r--r--  68  dimens.xml

./app/src/main/res/xml:
total 2.0K
-rw-r--r--  490  backup_rules.xml
-rw-r--r--  569  data_extraction_rules.xml

./app/src/test:
total 0
drwxr-xr-x  0  java

./app/src/test/java:
total 0
drwxr-xr-x  0  com

./app/src/test/java/com:
total 0
drwxr-xr-x  0  vigilancia

./app/src/test/java/com/vigilancia:
total 0
drwxr-xr-x  0  maestria

./app/src/test/java/com/vigilancia/maestria:
total 1.0K
-rw-r--r--  400  ExampleUnitTest.java

NÚMERO DE LÍNEAS POR ARCHIVO
13   ./.git/config
1    ./.git/description
1    ./.git/HEAD
15   ./.git/hooks/applypatch-msg.sample
24   ./.git/hooks/commit-msg.sample
174  ./.git/hooks/fsmonitor-watchman.sample
8    ./.git/hooks/post-update.sample
14   ./.git/hooks/pre-applypatch.sample
49   ./.git/hooks/pre-commit.sample
13   ./.git/hooks/pre-merge-commit.sample
53   ./.git/hooks/pre-push.sample
169  ./.git/hooks/pre-rebase.sample
24   ./.git/hooks/pre-receive.sample
42   ./.git/hooks/prepare-commit-msg.sample
78   ./.git/hooks/push-to-checkout.sample
```

```
77   ./.git/hooks/sendemail-validate.sample
128  ./.git/hooks/update.sample
59   ./.git/index
6    ./.git/info/exclude
1    ./.git/logs/HEAD
1    ./.git/logs/refs/heads/master
1    ./.git/logs/refs/remotes/origin/HEAD
16   ./.git/objects/pack/pack-cb169bfb677efa6a6be902bf15f08abb5cb13b55.idx
274  ./.git/objects/pack/pack-cb169bfb677efa6a6be902bf15f08abb5cb13b55.pack
1    ./.git/objects/pack/pack-cb169bfb677efa6a6be902bf15f08abb5cb13b55.rev
2    ./.git/packed-refs
1    ./.git/refs/heads/master
1    ./.git/refs/remotes/origin/HEAD
2    ./.gitignore
60   ./app/build.gradle
20   ./app/proguard-rules.pro
25   ./app/src/androidTest/java/com/vigilancia/maestria/ExampleInstrumentedTest.java
45   ./app/src/main/AndroidManifest.xml
63   ./app/src/main/java/com/vigilancia/maestria/Adaptadores/DispositivosAdapter.java
104  ./app/src/main/java/com/vigilancia/maestria/Adaptadores/EscenariosAdapter.java
91   ./app/src/main/java/com/vigilancia/maestria/Adaptadores/NotificacionesAdapter.java
18   ./app/src/main/java/com/vigilancia/maestria/Bd/Converters.java
66   ./app/src/main/java/com/vigilancia/maestria/Bd/Notificacion.java
21   ./app/src/main/java/com/vigilancia/maestria/Bd/NotificacionDao.java
31   ./app/src/main/java/com/vigilancia/maestria/Bd/NotificacionesDatabase.java
40   ./app/src/main/java/com/vigilancia/maestria/Comun/SecureStorageUtil.java
32   ./app/src/main/java/com/vigilancia/maestria/Comun/Utilidades.java
91   ./app/src/main/java/com/vigilancia/maestria/Fragments/DispositivosFragment.java
102  ./app/src/main/java/com/vigilancia/maestria/Fragments/EscenariosFragment.java
144  ./app/src/main/java/com/vigilancia/maestria/Fragments/LoginFragment.java
69   ./app/src/main/java/com/vigilancia/maestria/Fragments/NotificacionesFragment.java
109  ./app/src/main/java/com/vigilancia/maestria/Fragments/Principal.java
11   ./app/src/main/java/com/vigilancia/maestria/Globales/App.java
19   ./app/src/main/java/com/vigilancia/maestria/Json/Accion.java
27   ./app/src/main/java/com/vigilancia/maestria/Json/Command.java
216  ./app/src/main/java/com/vigilancia/maestria/Json/Dispositivos.java
51   ./app/src/main/java/com/vigilancia/maestria/Json/Escenarios.java
122  ./app/src/main/java/com/vigilancia/maestria/MainActivity.java
105  ./app/src/main/java/com/vigilancia/maestria/MyFirebaseMessagingService.java
7    ./app/src/main/res/drawable/icononotificacion.png
170  ./app/src/main/res/drawable/ic_launcher_background.xml
10   ./app/src/main/res/drawable-hdpi/icononotificacion.png
3    ./app/src/main/res/drawable-mdpi/icononotificacion.png
29   ./app/src/main/res/drawable-v24/ic_launcher_foreground.xml
8    ./app/src/main/res/drawable-xhdpi/icononotificacion.png
10   ./app/src/main/res/drawable-xxhdpi/icononotificacion.png
7    ./app/src/main/res/drawable-xxxhdpi/icononotificacion.png
24   ./app/src/main/res/layout/activity_main.xml
19   ./app/src/main/res/layout/content_main.xml
43   ./app/src/main/res/layout/fragment_dispositivos.xml
12   ./app/src/main/res/layout/fragment_dispositivos_list.xml
33   ./app/src/main/res/layout/fragment_escenarios.xml
12   ./app/src/main/res/layout/fragment_escenarios_list.xml
12   ./app/src/main/res/layout/fragment_image.xml
```

```
80   ./app/src/main/res/layout/fragment_login.xml
45   ./app/src/main/res/layout/fragment_notificaciones.xml
12   ./app/src/main/res/layout/fragment_notificaciones_list.xml
40   ./app/src/main/res/layout/fragment_principal.xml
12   ./app/src/main/res/layout/popup_image.xml
9    ./app/src/main/res/menu/menu_main.xml
4    ./app/src/main/res/mipmap-anydpi-v26/ic_launcher.xml
4    ./app/src/main/res/mipmap-anydpi-v26/ic_launcher_round.xml
6    ./app/src/main/res/mipmap-hdpi/ic_launcher.webp
15   ./app/src/main/res/mipmap-hdpi/ic_launcher_round.webp
1    ./app/src/main/res/mipmap-mdpi/ic_launcher.webp
7    ./app/src/main/res/mipmap-mdpi/ic_launcher_round.webp
8    ./app/src/main/res/mipmap-xhdpi/ic_launcher.webp
15   ./app/src/main/res/mipmap-xhdpi/ic_launcher_round.webp
9    ./app/src/main/res/mipmap-xxhdpi/ic_launcher.webp
18   ./app/src/main/res/mipmap-xxhdpi/ic_launcher_round.webp
18   ./app/src/main/res/mipmap-xxxhdpi/ic_launcher.webp
33   ./app/src/main/res/mipmap-xxxhdpi/ic_launcher_round.webp
81   ./app/src/main/res/navigation/nav_graph.xml
9    ./app/src/main/res/values/colors.xml
6    ./app/src/main/res/values/dimens.xml
26   ./app/src/main/res/values/strings.xml
24   ./app/src/main/res/values/themes.xml
2    ./app/src/main/res/values-land/dimens.xml
15   ./app/src/main/res/values-night/themes.xml
2    ./app/src/main/res/values-w1240dp/dimens.xml
2    ./app/src/main/res/values-w600dp/dimens.xml
12   ./app/src/main/res/xml/backup_rules.xml
18   ./app/src/main/res/xml/data_extraction_rules.xml
16   ./app/src/test/java/com/vigilancia/maestria/ExampleUnitTest.java
6    ./build.gradle
58   ./generar_reporte.sh
23   ./gradle.properties
0    ./reporte.txecho
375  ./reporte.txt
228  ./reporte0.txt
16   ./settings.gradle
```