Aplicación para gestionar alarmas y cámaras domóticas

Iván Peñarando Martínez

Resumen—El desarrollo de los sistemas domóticos ha hecho un avance considerable gracias a los cambios que actualmente el mundo afronta, este desarrollo ha conseguido hacer que nos replanteemos la forma en la que nos ocupamos de nuestro entorno más preciado: nuestro hogar. El siguiente trabajo desarrolla una aplicación móvil que consigue implementar en las manos del usuario el monitoreo de cámaras de seguridad y el control de una alarma. El sistema utiliza una Raspberry Pi como un servidor con cámara de seguridad con capacidades de alarma cargadas a ella. La aplicación permite al internauta monitorear imágenes de video en tiempo real y activar una alarma a voluntad a través de su teléfono móvil; también, incluye la característica de detección de movimiento, que puede utilizarse para alertar sobre cambios sospechosos en el video, dichos movimientos detectados pueden considerarse como una amenaza. Gracias al uso de la aplicación será posible conseguir unos niveles óptimos de seguridad, favoreciendo la tranquilidad y confianza en casa, además de una gran comodidad debido a su fácil accesibilidad a través del teléfono móvil.

Paraules clau—Domótica, Alarma, Cámaras IP, Raspberry Pi, Android, Detección de movimiento, Notificaciones, Vigilancia, Monitoreo.

Abstract—The development of home automation systems has made considerable progress thanks to the changes the world is currently facing. This development has led us to reconsider how we take care of our most cherished environment: our home. The following work develops a mobile application that puts in the hands of the user the monitoring of security cameras and the control of an alarm. The system uses a Raspberry Pi as a server with a security camera and alarm capabilities loaded onto it. The application allows the user to monitor real-time video images and activate an alarm at will through their mobile phone. It also includes a motion detection feature, which can be used to alert about suspicious changes in the video, with the detected movements potentially being considered a threat. Thanks to the use of the application, it will be possible to achieve optimal security levels, fostering peace of mind and confidence at home, as well as great convenience due to its easy accessibility through the mobile phone.

Index Terms—Home Automation, Alarm, IP cameras, Raspberry Pi, Android, Motion detection, Notifications, Surveillance, Monitoring.

1 Introducción - Contexto del trabajo

En la última década, el avance en el diseño de los sistemas de domótica ha revolucionado la forma en que nos ocupamos de nuestro entorno de vida. El control de nuestro entorno se ha vuelto una parte clave del día a día y nos ha hecho prestar más atención a la seguridad del hogar.

Este proyecto es sobre el desarrollo de una aplicación móvil a través de la cual las cámaras de los sistemas domóticos se pueden monitorear y controlar fácilmente, además de tener la posibilidad de, en caso de cualquier peligro, activar una alarma por la que poder proteger nuestro hogar, todo a través de una interfaz intuitiva y efectiva.

La necesidad básica que ha justificado este proyecto es la creciente necesidad de sistemas de seguridad accesibles, eficientes y efectivos para el consumidor promedio. Poder mirar y controlar su hogar en cualquier parte, con un

- *E-mail de contacto: ivan.penarando@autonoma.cat*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Jordi Serra Ruiz (Área de Ciencias de la Computación e Inteligencia Artificial)*
- *Curso 2023/24*

teléfono en su bolsillo, no solo otorgará tranquilidad, sino que también le permitirá tomar decisiones basadas en la situación si surge una emergencia.
La estructura del documento empieza por la sección de objetivos en la que se explicará qué se pretende alcanzar con la aplicación. Continuaremos con el estado del arte donde se revisarán conceptos y tecnologías ya existentes que podrían ser relevantes para la aplicación. Siguiendo con la metodología, donde se describe el enfoque de trabajo que se ha adaptado. Finalizando por los resultados y conclusiones donde se expondrán los resultados y desafíos así como los objetivos alcanzado

2 Objetivos

El propósito general de este proyecto es el desarrollo de una aplicación móvil para controlar fácil y rápidamente cámaras de seguridad en el hogar con el fin de convertir el dispositivo en una herramienta que sea usada y beneficiosa para mejorar la seguridad en casa.

Como característica principal de la aplicación, se permitirá a los usuarios **ver en tiempo real** los videos de la cámara de vigilancia de manera remota, lo cual es una gran manera de permitir a los usuarios monitorear sus hogares desde cualquier parte.

Otro objetivo crucial es establecer un **sistema de detección de movimiento** que tenga la capacidad de alertar al usuario mediante notificaciones móviles si se detecta algún movimiento dudoso, permitiendo así la seguridad bifurcada. Habrá una función implementada que permitirá al usuario **disparar una alarma** utilizando la aplicación móvil en caso de que se detecte una posible amenaza, y la Raspberry Pi tomará la responsabilidad de activar uno de sus pines para llevar a cabo cualquier mecanismo de seguridad que el usuario final desee.

Para asegurar la compatibilidad de la aplicación con diferentes cámaras IP se investigarán los protocolos de vídeo más utilizados por las cámaras para intentar ofrecer la máxima flexibilidad hacia los usuarios.

3 Estado del arte

Desde la última década, el avance en el desarrollo de dispositivos de automatización del hogar ha cambiado el paradigma de cómo interactuamos con nuestros hogares. La monitorización y seguridad en el hogar se ha mejorado extremadamente con la incorporación de cámaras de vigilancia y sistema de alarmas. En este apartado echamos un vistazo a algunos de los proyectos y dispositivos recientes que han dado forma en el desarrollo de la aplicación.

El primero y más relevante es el sistema domótico presentado por Salgado y Peñaloza (2019) [1] que ofrece capacidades de automatización para iluminación, temperatura, seguridad y acceso. El sistema está basado en el uso de una Raspberry Pi, una base de datos Firebase y una aplicación en Android para la interacción del usuario. Admite protección con contraseña y una cámara IP para monitoreo en tiempo real con capacidades de detección de movimiento y notificación al usuario sobre situaciones sospechosas.

Otro de los estudios es el de Utitiaja Mayancha (2019) [2], que trata sobre la implementación de una aplicación de Android para controlar las unidades de automatización del hogar. Esta aplicación se utiliza para controlar una cámara IP y sensores de movimiento con un Arduino en la interfaz de control móvil. El desarrollo de este proyecto se realiza utilizando tecnologías como Android Studio, Java, HTML y MySQL en la gestión de bases de datos. Es único por su interfaz de usuario simple y la mejora que agrega a la calidad de vida de las personas.

Además de esto, se buscaron varios tipos de cámaras de vigilancia [3,4] que podríamos utilizar para el desarrollo de la aplicación. Una de ellas son las cámaras analógicas, aunque han sido menos utilizadas en las aplicaciones modernas, toman video que se transmite a un grabador de video digital. El problema de ellas es que no hay una manera sencilla de acceder al vídeo en directo desde la propia Raspberry Pi, se tratan de cámaras más autónomas.

Otra de las cámaras serían las IP, estas utilizan redes de datos, tienen la capacidad de ser accedidas remotamente en cuanto a vídeo y se pueden integrar fácilmente con dispositivos de automatización del hogar debido a sus modos de instalación y operación simples. Una variante de estas, serían las cámaras PTZ tienen mayores capacidades de movimiento y zoom remotas y, por lo tanto, son más adecuadas para seguir objetos de interés o áreas dinámicas.

De los protocolos de transmisión, también se tendría que hablar de diversos estándares disponibles [5]. UDP es un protocolo rápido pero tiene la tendencia a perder datos en redes congestionadas. El protocolo TCP, en cambio, es mucho más estable y más correcto en lo que se refiere a errores; este último sería perfectamente adecuado para las transmisiones de vídeo de alta resolución. RTSP hace uso de estos protocolos y está especializado en la transmisión de multimedia en tiempo real [6], y ONVIF es un protocolo a través del cual diferentes fabricantes de dispositivos de video pueden tener una interoperabilidad armoniosa mediante la comunicación [7].

Para la creación de la aplicación móvil, muchas herramientas han sido investigadas [8]. Entre las herramientas, Android Studio es una de las recomendadas para las instalaciones avanzadas de pruebas y desarrollo de aplicaciones en Android. Otras buenas opciones son React Native y Flutter, que están dirigidas al desarrollo de aplicaciones multiplataforma, pero para este proyecto particular, Android Studio sería utilizado para la integración con los dispositivos Android, así como la disponibilidad de herramientas avanzadas de desarrollo.

Como tal, para finalizar, el estado de la arte para la monitorización inteligente de hogares y sistemas consiste en la capacidad para desarrollar una solución barata y

confiable para la monitorización de la seguridad residencial al aprovechar el poder de soluciones como la Raspberry Pi, cámaras IP y estándares de transmisión comunes. Se basará en estos conceptos para desarrollar la aplicación y proporcionar una solución fácil de usar y efectiva que aumente la seguridad y el control del entorno para los usuarios.

4 METODOLOGÍA

En cuanto a la metodología del proyecto se empleará un método de desarrollo incremental e iterativo con los valores de las prácticas ágiles. Esto es para tratar todas las posibles modificaciones de los requisitos y trabajar en las cuestiones de tal manera que la entrega del proyecto pueda ser exitosa. Las etapas más críticas de la metodología utilizada son las siguientes:

La primera etapa del proyecto consiste en investigar el software y el hardware necesarios. El objetivo es recopilar toda la documentación pertinente sobre cámaras de vigilancia domésticas para decidir qué software y hardware se deben utilizar. Luego, investigar los diversos entornos de desarrollo de aplicaciones móviles para elegir los que sean más adecuados para los requisitos del proyecto.

En la siguiente etapa se tratan los prototipos de interfaces de usuario y conceptos. Con la premisa principal de crear una interfaz intuitiva y fácil de usar. Se tratará de afinar mejoras basadas en las pruebas de usabilidad y el feedback para asegurar la mejor experiencia de usuario final.

Para la tercera etapa se incorporarán las características de tiempo real mencionadas anteriormente, como las transmisiones en vivo de las cámaras, la detección de movimiento y la activación remota de la alarma.

Como última etapa la aplicación se someterá a pruebas en varios entornos y escenarios. Se corregirán los problemas relacionados con el rendimiento y la seguridad de la aplicación.

En el desarrollo, se adoptará un control de versiones GitHub, para rastrear el código fuente y el progreso del proyecto de manera que se pueda realizar una edición sin el miedo a perder datos. La documentación será una parte vital del proyecto desde la investigación hasta las pruebas finales.

El desarrollo tendrá lugar en sprints de una o dos semanas de duración. Al inicio de cada sprint, se llevará a cabo una revisión y priorización de la lista de trabajo pendiente. Al final de cada sprint, se llevará a cabo una revisión del trabajo con el objetivo de identificar posibles errores y establecer mejoras para el trabajo en el próximo sprint. Con esta iteración continua del desarrollo, se logrará un crecimiento incremental de las mejoras y una adaptación a los cambios necesarios para la ejecución exitosa del proyecto.

5 DESARROLLO

5.1 Interfaz de usuario

La interfaz de usuario es una parte muy importante del desarrollo de cualquier aplicación, sobre todo a la hora de la instalación de sistemas de seguridad para hogares, ya que esta debe ser muy efectiva y muy fácil de utilizar por cualquier persona. En el caso de esta aplicación, la UI se diseñará de manera clara, sencilla e intuitiva para que cualquier usuario pueda operar fácilmente con ella.

Habrá un menú principal de la aplicación a través del cual se accederá a todas las otras funciones. Las cámaras disponibles se listarán en el menú principal ocupando la mayor parte de la pantalla dado su importancia, mostrando tanto el nombre de la cámara como su dirección IP, para que la transmisión en directo de vídeo de cada cámara sea la parte más accesible de acceder. En lo que respecta a añadir nuevas cámaras, habrá un botón flotante con el signo "+", usaremos esta letra ya que es muy común en el diseño de la interfaz de usuario y mejorará su visibilidad y uso.

El otro elemento de la interfaz de usuario será un botón de alarma en la parte superior derecha, en forma de un icono de alarma. Este botón activará las funciones de seguridad de Raspberry Pi activando un pin de éste, si el usuario cree que encuentra algún movimiento sospechoso. Habrá un botón de configuración para que los usuarios establezcan la dirección IP de la Raspberry Pi y para que puedan eliminar las cámaras configuradas.

Las interfaces iniciales se han creado con un software conocido por hacer prototipos llamado Marvel App [9]. El prototipo, como veremos más adelante, sufrirá algunos cambios para garantizar la experiencia de usuario.

Fig. 1: Prototipo del menú de inicio

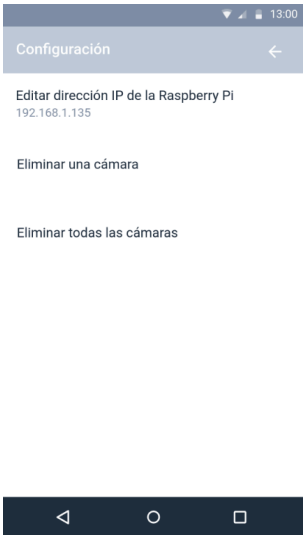

Fig. 2: Prototipo del menú de configuración

5.2 Bases del desarrollo

Para comenzar con el desarrollo, es necesario establecer e instalar los elementos que vamos a utilizar para configurar nuestro entorno de trabajo. Para ello, se instalará el sistema operativo Raspberry Pi OS en una tarjeta microSD, que luego se insertará en el Raspberry Pi. Por otro lado, se instalará y configurará el entorno de Android Studio para desarrollar la aplicación para comenzar el proyecto que nombraré como "Camera Manager".

Desde el sitio oficial de Raspberry Pi, se descargará la imagen de "Raspberry Pi OS with desktop" (64 bits) y se instalará en la microSD. El mencionado sistema operativo es un sistema operativo especializado que se ejecuta en la Raspberry Pi con un excelente rendimiento y proporcionando un escritorio con herramientas básicas esenciales para el desarrollo.

Se debe tener una configuración de red donde la Raspberry Pi y las cámaras IP estén en la misma red, ya que en esto se basará nuestro servidor. La IP fija en ambos es una buena práctica en lugar de cualquiera de las prácticas DHCP para no encontrar complicaciones en la conectividad y para facilitar la configuración de la futura aplicación.

En la parte del backend, la Raspberry Pi actuará como un intermediario entre la aplicación Android y las cámaras IP, se desarrollará una API REST en ella. Esta API permitirá registrar nuevas cámaras, obtener la lista de cámaras registradas, recibir video en vivo y activar la alarma si es necesario.

En la parte del frontend, se creará un nuevo proyecto en Android Studio llamado "Camera Manager" que contendrá todas las funcionalidades requeridas para la gestión de cámaras y alarmas. La interfaz de las aplicaciones se diseñarán pudiendo observar el resultado en vivo gracias a Android Studio. La usabilidad y las consideraciones estéticas se tendrán en cuenta para que el producto sea atractivo y fácil de usar.

Finalmente, envolviendo todo el proyecto abriremos un github para poder tener en cuenta todos los cambios que se van introduciendo en el proyecto, además de tener el proyecto en un lugar seguro.

5.3 Desarrollo del frontend

En cuanto a la parte del frontend las funcionalidades específicas que tendrá nuestra aplicación serán las siguientes:

Gestión de cámaras:

- **Agregar Cámara**: Este módulo permitirá al usuario agregar una nueva cámara con un nombre y dirección IP en concreto.
- **Editar Cámara**: El usuario podrá modificar tanto la dirección IP como el nombre de las cámaras.
- **Eliminar Cámara**: El usuario podrá eliminar una o más cámaras.

Vista de Transmisiones en Vivo: El usuario podrá ver las transmisiones de las cámaras en la aplicación.

Reproducción de Video: La reproducción de video por parte de la aplicación evita que la pantalla se apague y también detecta eventos durante la reproducción, como el búfer y los errores para mostrar una pantalla de carga mientras se carga el stream, por ejemplo.

Modo inmersivo: Mientras se reproduce vídeo en horizontal, la aplicación no mostrará barras de estado y navegación para dar una experiencia de pantalla completa.

Detección de Movimiento:

- **Servicio de detección de movimiento**: Tendremos un servicio de detección de movimiento es un servicio en segundo plano que escuchará constantemente el servidor de la rpi para recibir notificaciones de detección de movimiento.
- **Notificaciones**: La aplicación notificará al usuario cuando detecte algún movimiento con información sobre la cámara en concreto dónde se ha detectado el movimiento.

Configuración:

- **Cambiar la Dirección IP del Servidor**: Se le proporcionará al usuario la opción de cambiar la dirección IP del servidor.
- **Eliminar Todas las Cámaras**: El usuario tendrá la opción de eliminar todas las cámaras que el usuario ha registrado en el sistema utilizando una llamada a la API de Raspberry Pi.

Alarma: La aplicación tendrá la funcionalidad de alarma, la cual hará que se active un pin en la Raspberry Pi con el propósito de que se realice la opción de seguridad que el usuario desee.

Para implementar todos estos elementos, haremos uso de unas librerías para facilitar la implementación de las funcionalidades principales de la aplicación:

- **Retrofit**: Se utilizará para hacer solicitudes a la API del servidor, consiguiendo así que la Raspberry Pi haga lo que el usuario desee.
- **LibVLC**: Librería de VLC para el control de la reproducción de transmisiones de video en tiempo real.
- **WebSockets**: Serán utilizados para recibir notificaciones en tiempo real sobre la detección de movimiento por parte del servidor.

- **SharedPreferences**: Se utilizará para guardar las preferencias del usuario dentro de la aplicación android, en este caso lo usaremos para guardar la dirección ip de la raspberry pi como servidor.

- **Notificaciones**: Utilizaremos NotificationCompat para notificar al usuario sobre la detección de movimiento.

Para desarrollar el frontend de una aplicación que gestiona cámaras y alarmas domóticas, se utilizó el lenguaje de programación de Kotlin, contando con una arquitectura basada en varias clases que funcionan en conjunto para ofrecer una experiencia de usuario eficiente y fluida.

La clase ***MainActivity*** actúa como el punto de entrada y control principal. Al iniciarse, configura la interfaz de usuario, incluyendo la barra de herramientas y la vista principal donde se muestra la lista de cámaras registradas. Aquí se inicializan las preferencias compartidas que almacenan la dirección IP del servidor, crucial para mantener la conectividad. Si no hay una IP almacenada, la aplicación pide al usuario que la ingrese al comenzar. La *MainActivity* también gestiona eventos de la interfaz, como los botones para agregar una nueva cámara y refrescar la lista existente. Además, inicia un servicio de detección de movimiento tras obtener la IP del servidor, asegurando que el sistema de seguridad esté siempre activo.

Otra de las clases sería ***AddCameraActivity***, que permite al usuario añadir nuevas cámaras o editar las existentes. Al iniciarse, configura la interfaz con campos de texto para el nombre y la dirección IP de la cámara, y un botón para confirmar la acción. Si se está editando una cámara, los campos se rellenan automáticamente con la información correspondiente y el botón cambia a "Actualizar Cámara". Al confirmar, la actividad valida los datos y llama a la API para agregar o actualizar la cámara según sea necesario.

AlarmReceiver y ***BootReceiver*** son fundamentales para mantener el servicio de detección de movimiento activo. *AlarmReceiver* reinicia el servicio cuando recibe una alarma del sistema, utilizando la IP almacenada en las preferencias compartidas. Si no encuentra la IP, registra un error. *BootReceiver* escucha el evento de arranque del dispositivo y reinicia el servicio de detección de movimiento con la IP del servidor, asegurando que el servicio se inicie automáticamente tras un reinicio del dispositivo.

La comunicación con el servidor se maneja mediante ***ApiClient*** y ***ApiInterface***. ApiClient configura Retrofit con la URL base del servidor de la Raspberry Pi, mientras que *ApiInterface* define los endpoints de la API, permitiendo recuperar la lista de cámaras, añadir, actualizar y eliminar cámaras, así como controlar las alarmas. Esta estructura facilita la interacción eficiente de la aplicación con el servidor para realizar todas las operaciones necesarias.

La clase ***Camera*** modela una cámara IP y es Parcelable, lo que permite pasar sus datos fácilmente entre los componentes de Android. ***CameraAdapter*** es el adaptador de la vista que muestra la lista de cámaras y gestiona la interacción del usuario, manejando eventos de pulsación y pulsación larga para iniciar transmisiones de video y editar o eliminar cámaras respectivamente. También realiza solicitudes a la API para iniciar y detener transmisiones.

MotionDetectionServic escucha eventos de detección de movimiento enviados por la Raspberry Pi a través de un *WebSocket*. Mantiene el servicio activo configurando una alarma que se dispara regularmente. Establece una conexión con el servidor para recibir notificaciones de movimiento y alerta al usuario mediante notificaciones en el teléfono.

SettingsActivity permite al usuario cambiar la configuración de la aplicación, como la dirección IP del servidor y eliminar todas las cámaras configuradas. Proporciona una interfaz clara para estas acciones, incluyendo cuadros de diálogo para confirmar cambios importantes.

VideoPlayerActivity se encarga de mostrar las transmisiones en vivo de las cámaras seleccionadas. Para ello utilizamos *LibVLC* y *MediaPlayer* para gestionar el flujo de vídeo, ofreciendo una experiencia de usuario inmersiva. Al finalizar la transmisión, la actividad se cierra y realiza una llamada a la API para detener la transmisión en el servidor. Al inicio de la aplicación, en vez de utilizar *LibVLC* se intentó utilizar *Exoplayer*. Sin embargo, tras muchas pruebas, el vídeo de las cámaras no aparecía en la aplicación, investigando más al respecto decidí utilizar *LibVLC* ya que era compatible con más tipos de vídeos que *Exoplayer*, permitiendo aumentar al máximo el número de cámaras que funcionen con este sistema.

Este desarrollo integrado y bien organizado del frontend garantiza una gestión eficiente de las cámaras y alarmas domóticas, ofreciendo al usuario una herramienta poderosa y fácil de usar.

5.3 DESARROLLO DEL BACKEND

El backend de la aplicación está programado en Python utilizando la biblioteca **Flask**. Se aloja un servidor **RESTful** en una Raspberry Pi cuyas funciones principales son la gestión de datos de la cámara de seguridad, la activación de la alarma y la detección de movimiento. Notifica a los clientes que están conectados a través de caso de un evento de detección de movimiento.

El servidor utiliza Flask para los endpoint HTTP y Flask-Sock para los **WebSockets**. Se definen unas rutas para controlar las cámaras, iniciar y detener el flujo de video en tiempo real, y la activación de la alarma. La detección de movimiento se realiza procesando las transmisiones de vídeo con **OpenCV.**

Al iniciar el servidor, si está presente, carga las cámaras desde un archivo JSON con los datos de las cámaras que

los usuarios han ido registrando en el servidor (cameras.json). Si no se encuentra este archivo, se cargará una lista de cámaras predefinida. El GPIO se configura para la activación de una alarma física que esté conectada a la Raspberry Pi.

En cuanto a las rutas para los endpoint HTTP tenemos las siguientes:

- **/camaras (GET):** Muestra la lista de cámaras.
- **/cameras (POST)**: Registra una nueva cámara.
- **/cameras/int:camera_id/start (POST)**: Comienza el flujo en vivo de una cámara.
- **/cameras/int:camera_id/stop (POST):** Detiene el stream en vivo de una cámara.
- /**cameras (DELETE):** Elimina todas las cámaras de la lista.
- **/alarm (POST)**: Enciende una alarma física en la Raspberry Pi.
- **/cameras/int:camera_id (DELETE)**: Elimina una cámara en concreto.
- **/camaras/int:id_de_camara (PUT)**: Actualiza la información de una cámara.

Además de estas, tenemos la ruta **/ws** que maneja las conexiones WebSocket. Los clientes se conectan automáticamente para ser notificados en tiempo real si se descubre algún movimiento. El servidor mantiene una lista de clientes que están actualmente conectados, y notifica a todos los clientes activos si se descubre algún movimiento.

Para la **detección de movimiento** capturamos el primer segundo de video. Continuamos saltando algunos frames iniciales ya que esto ayuda a estabilizar la captura y evita no detectar demasiado cambio entre dos frames, por ello examinamos la detección cada dos frames. Con estos, se selecciona una parte específica de la imagen, que es la parte central, dejando márgenes arriba y abajo. Esto ayuda a evitar falsas alarmas de movimiento en los bordes de la imagen, ya que como podemos ver en la *figura 3*, en mi caso y en muchos otros, las cámaras ip suelen registrar la fecha actual tanto en la parte de arriba como en la de abajo, para evitar que esto sea un inconveniente, focalizamos la parte central.

Una vez tenemos los dos frames claves, se calcula la diferencia entre dos imágenes y se convierte a blanco y negro para simplificar el análisis. Luego, se aplica un desenfoque para reducir el ruido y las pequeñas variaciones que no representan un movimiento real.

Después, se convierte la imagen en una versión binaria, donde las áreas blancas indican movimiento y las negras no. Esta imagen se procesa para rellenar posibles huecos y conectar áreas de movimiento cercanas.

Fig. 3: Captura de la cámara con la fecha actual

Se detectan y analizan los contornos de las áreas blancas. Los contornos muy pequeños se ignoran porque probablemente no representen un movimiento significativo. Para los contornos más grandes, se verifica si están dentro de la parte seleccionada de la imagen.

Si se detecta un movimiento significativo en la parte central de la imagen, se registra. Este método asegura que solo se detecten movimientos importantes, minimizando las falsas alarmas y mejorando la eficiencia del sistema.

Durante todo este proceso, las imágenes se actualizan continuamente para permitir la detección en tiempo real. Se introduce un pequeño retraso entre cada iteración para no sobrecargar el sistema y asegurar que las diferencias detectadas representen un movimiento continuo.

Por la parte del **streaming de la cámara**, cada cámara tiene un ID y una URL de stream RTSP únicos en la misma red. Al iniciar un stream, ffmpeg toma el stream de la cámara correspondiente y lo transmite nuevamente en RTMP con una calidad reducida, esto se hace para que la raspberry pi pueda soportar el stream y se vea de manera fluida. Se reduce tanto la calidad inicial de la cámara, como el bitrate. Las operaciones de ffmpeg se realizan con el módulo subprocess, que permite transmisiones en vivo personalizadas y la guardamos en una estructura de datos para que pueda ser posible recuperar el proceso de streaming y pararlo si es necesario.

El pin GPIO en la Raspberry Pi es controlado por la librería RPi.GPIO. Cuando se activa **la alarma**, el pin se sube alto durante 3 segundos, luego se baja. El pin que activamos es de 5V, consiguiendo que el usuario pueda activar cualquier sistema personalizado de alarma des de la propia aplicación.

Por último, para asegurar que ningún proceso queda ejecutándose si la aplicación experimenta un fallo al ejecutarse, hacemos uso de atexit para registrar un manejador que apagará todos los procesos de ffmpeg y el bucle de detección de movimiento al salir del servidor.

6 RESULTADOS

Para finalizar, ahora entraremos a los resultados finales, es decir, en la aplicación en sí. Por ello seguiremos el flujo de uso de la aplicación para mostrar sus funcionalidades, así como su interfaz final.

Al **inicio** de la aplicación, lo primero que saldrá es un popup para que el usuario introduzca la ip de la Raspberry Pi, una vez introducida se listarán todas las cámaras que se reciban de la api. Si pulsamos cualquiera de ellas podremos acceder al streaming de esta. También podemos observar dos iconos en la barra de tareas, el primero de ellos nos permitirá activar la alarma mientras que el segundo nos llevará a la pantalla de configuración:

Fig. 4: Popup para introducir la dirección IP del servidor

Fig. 5: Pantalla principal de la aplicación

En cuanto a la **gestión de las cámaras**, el usuario podrá agregar nuevas cámaras (presionando el botón "+" visible en la figura 5), editarlas o eliminarlas según sea necesario. Para editar o borrar una cámara, se deberá dejar pulsado sobre la cámara deseada, en ese momento le saldrán las opciones mencionadas. Por otro lado, si pulsa el botón de añadir cámara, encontrará un formulario similar en el que podremos introducir los datos.

Podemos observar en la figura 6 la pantalla para añadir una cámara y en la figura 7 la pantalla para editar una de ellas:

Fig. 6: Pantalla para añadir una cámara

Fig. 7: Pantalla para editar una cámara

El usuario también puede cambiar la dirección IP del servidor y eliminar todas las cámaras desde la pantalla de **configuración**.

Fig. 8: Pantalla de configuración

Por último, como habíamos mencionado, al pulsar cualquiera de las cámaras se abrirá el reproductor del stream que tras un momento de carga nos mostrará el vídeo en directo de la cámara que hemos seleccionado, contamos también con un botón de atrás si queremos volver a la pantalla anterior.

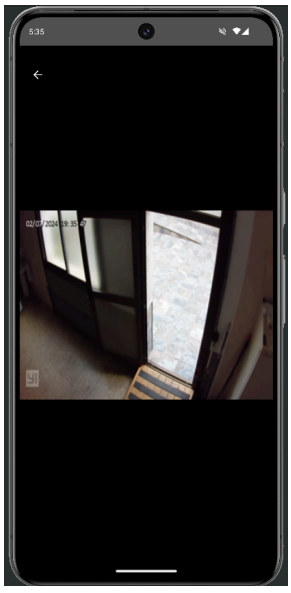

Fig. 8: Pantalla de reproducción de vídeo

7 Conclusión

En este apartado tendremos en cuenta todos los elementos mencionados en el trabajo y los expondremos de manera lógica y ordenada puntualizando los elementos que finalmente no hayamos tratado y posibles extensiones futuras del propio trabajo.

Se ha creado una aplicación para gestionar cámaras y alarmas domóticas haciendo uso de Android Studio para su uso en teléfonos Android. La aplicación permite la sencilla visualización en directo de cámaras de vigilancia y la activación de alarmas. A su vez, se ha creado un ecosistema por el cual utilizamos la Raspberry Pi como servidor para acceder a las propias cámaras que tengamos conectadas a la red domótica, así como la activación de uno de los pines de la Raspberry Pi en caso de que el usuario active la alarma. Finalmente, se ha diseñado una interfaz intuitiva, simple y amigable para que el usuario no tenga ningún problema a la hora de interactuar con la aplicación

Los elementos que no se han podido tratar seria la compatibilidad con otros sistemas. Al fin y al cabo, la aplicación final se ha desarrollado exclusivamente teniendo en cuenta los usuarios de Android, sin considerar la compatibilidad con iOS. Esto podría limitar el alcance a aquellas personas que utilicen dispositivos de Apple. Además, no se ha podido dar soporte a todos los tipos de cámaras posibles, nos hemos centrado tan solo en las cámaras IP que puedan streamear su contenido a través del protocolo RTSP.

7 Extensiones del trabajo realizado

Como último apartado presentaré algunas mejoras que podrían hacer para mejorar la experiencia del usuario o bien ampliar el número de usuarios interesados:

- **Compatibilidad de múltiples plataformas**: extender la compatibilidad de la aplicación para hacer posible su uso tanto en dispositivos Android como en iOS.

- **Implementación de control PTZ**: integrar, para las cámaras que sean compatibles, una manera de controlar su movimiento a través de estos controles, para así poder ajustar desde la aplicación parámetros como el ángulo y el zoom de las cámaras.
- **Ampliación de protocolos de cámaras**: extender la aplicación para poder ser compatible con más tipos de cámaras, mejorando la flexibilidad y utilidad de la propia aplicación para los usuarios
- **Mejoras en seguridad**: implementar mejoras de seguridad para que los usuarios puedan usar la aplicación fuera de casa sin ninguna preocupación de que haya alguna brecha en los datos y sean expuestos

Agradecimientos

Quiero agradecer a mi tutor, Jordi Serra, por darme libertad a la hora de hacer el trabajo, así como los consejos dados para mejorar la aplicación.

También quiero agradecer a mi familia por darme apoyo en todo momento durante la realización del trabajo.

Bibliografía

[1] Salgado, A. G., & Peñaloza, F. G. (2019) Sistema domótico con aplicación móvil en android.
[2] Utitiaja Mayancha, J. K. (2019). Aplicación móvil para video vigilancia (Bachelor's thesis).
[3] Admin. (2023, 17 diciembre). Qué tipos de cámaras de vigilancia existen? Domotica En Casa. https://sudomotica.com/que-tipos-de-camaras-de-vigilancia-existen/
[4] Ortiz, Ó. (2023, 20 diciembre). Tipos de cámaras de vigilancia ¿Cuál es mejor? Eligenio. https://eligenio.com/es/blog/tipos-de-camaras-de-vigilancia/
[5] De Luz, S. (2024, 11 enero). ¿Qué protocolo es mejor? TCP vs UDP, descubre cuándo usar cada uno. RedesZone. https://www.redeszone.net/tutoriales/internet/tcp-udp-caracteristicas-uso-diferencias/
[6] Klyushkov, M. (2022, 25 enero). ¿Qué es RTSP y por qué es necesario? Flussonic. https://flussonic.com/es/blog/news/about-rtsp/
[7] fcisistemas.com. (2022, 16 mayo). Cámaras IP con protocolo ONVIF, ¿qué es? - Sistemas de videovigilancia Argos. Sistemas de videovigilancia Argos. https://argos.red/protocolo-onvif/
[8] Presta, M. (2021, 13 octubre). Las 25 mejores herramientas de desarrollo móvil. Back4App Blog. https://blog.back4app.com/es/las-25-mejores-herramientas-de-desarrollo-movil/
[9] Marvel. (s. f.). Marvel - The design platform for digital products. Get started for free. https://marvelapp.com/