UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

DomoCamera

Entorno domótico para la monitorización de cámaras desde Android.

Presentado por Francisco Martín Vargas
en Universidad de Burgos — 19 de septiembre de 2021
Tutor: César Represa Pérez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. César Represa Pérez, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Expone:

Que el alumno D. Francisco Martín Vargas, con DNI 71704736M, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "DomoCamera — Entorno domótico para la monitorización de cámaras desde Android".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 19 de septiembre de 2021

Vº. Bº. del Tutor:

D. César Represa Pérez

Resumen

El auge de la domótica y los teléfonos móviles ha tenido un impacto en la sociedad. Por un lado, la domótica facilita una gestión más eficiente en el hogar y hace más segura una vivienda. Por otro lado, la telefonía móvil pone a nuestro alcance cientos de herramientas tecnológicas que mejoran la calidad de vida de las personas.

Este proyecto muestra la unión perfecta entre estos dos sectores, poniéndolos además al alcance de todo el mundo para poder gestionar la seguridad del hogar desde nuestro teléfono móvil. Para ello se pretende crear un entorno que nos permita monitorizar cámaras a través de nuestro dispositivo móvil Android.

En el presente documento se muestra la realización de un servidor implementado en una Raspberry Pi a través del lenguaje de programación Python, así como una aplicación desplegada a través de Android desde la cual podremos gestionar todo el entorno. Para llevarlo a cabo, se han utilizado los entornos de desarrollo *Visual Studio Code* (Servidor) y *Android Studio* (aplicación).

El resultado de este trabajo se puede encontrar a través del repositorio de GitHub: DomoCamera.

Descriptores

Android, Servidor, Python, Domótica, Raspberry Pi.

Abstract

The boom of home automation and mobile phones has had an impact on society. On the one hand, home automation facilitates a more efficient home management and makes it safer. On the other hand, mobile telephony puts hundreds of technological tools at our disposal to improve people's quality of life.

This project shows the perfect match between these two sectors, putting them affordable for all, to be able to manage home security from our mobile phone. The project aim is to create an environment that allows us to monitor cameras through our Android mobile device.

This document shows the creation of a server implemented in a Raspberry Pi through the Python programming language, as well as an application deployed through Android from which we can manage the entire environment. To carry it out, the development environments that have been used are *Visual Studio Code* (Server) and *Android Studio* (application).

The result of this work can be found through the GitHub repository: DomoCamera.

Keywords

Android, Server, Python, Home automation, Raspberry Pi.

Índice general

Índice de figuras

Índice de tablas

Introducción

Hoy en día en los hogares cada vez estamos más rodeados de tecnologías, tanto móviles, como aparatos del hogar, los cuales disponen de sistemas informáticos muy avanzados y automatizados que nos proporcionan herramientas para ser manejados a distancia. Hablamos de lavadoras, frigoríficos, enchufes, interruptores, termostatos, e incluso aspiradores inteligentes, todos ellos monitorizables a distancia.

La domótica es el conjunto de todas las tecnologías que agrupan el control y la automatización inteligente de una vivienda, aportando servicios de gestión energética, bienestar, comunicación y seguridad. Gracias a ella llegaremos a gestionar de una manera más eficiente nuestros recursos, además de aportarnos más seguridad y un extra de confort.

Pero no sólo ha ayudado a las personas, gracias a la eficiencia energética y de recursos, también colabora con el cambio climático. Gracias a este sector, se reducen los consumos en calefacción, en energía para la vivienda, en gasto de cantidad de agua, etc, todo ello contribuye a detener el cambio climático además de a la población.

Es en la seguridad en lo que me he basado para la realización de este proyecto, crear un entorno que ayude en la gestión de la seguridad en la vivienda. Todo ello gestionado a través de un dispositivo móvil, debido a que hoy en día es indispensable el uso de un dispositivo móvil y todas las tecnologías se están desarrollando en base a ellos, por ejemplo el NFC para pagar sin necesidad de una tarjeta.

Además, el sector domótico está creciendo a gran velocidad hoy en día y ha evolucionado notablemente recientemente. Gracias a esta evolución y crecimiento, podemos obtener más funcionalidad, facilidad de uso e instalación, variedad de producto, además de mayor calidad y mayor oferta.

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** Descripción e introducción del proyecto. También podremos encontrar la estructura de la memoria, de los anexos y de los materiales adjuntos.
- **Objetivos del proyecto:** explicación de los objetivos que se pretenden cumplir.
- **Conceptos teóricos:** breve introducción a los conceptos teóricos clave para la comprensión del desarrollo del proyecto.
- **Técnicas y herramientas:** conjunto de técnicas metodológicas y herramientas empleadas durante el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo del proyecto:** exposición de los aspectos más relevantes en el desarrollo del proyecto.
- **Trabajos relacionados:** pequeña presentación y comparación con algunos trabajos relacionados con el presente trabajo.
- **Conclusiones y Líneas de trabajo futuras:** conclusiones derivadas tras la realización del proyecto, así como posibles mejoras futuras del resultado del desarrollo del proyecto.

1.2. Estructura de los anexos

Los anexos siguen la siguiente estructura:

- **Plan del proyecto Software:** desarrollo de la planificación temporal y el estudio de viabilidad del proyecto.
- **Especificación de Requisitos:** requisitos derivados de los objetivos del proyecto.
- **Especificación de diseño:** descripción del diseño del sistema con los consiguientes diagramas.
- **Documentación técnica de programación:** explicación de los recursos necesarios para trabajar con el proyecto (entornos de desarrollo, lenguajes, etc).

- **Documentación de usuario:** guía para usuario final, en ella se expone como se debe usar el producto final.

1.3. Materiales adjuntos

Contenido del CD

El CD que se ha entregado junto con la memoria se estructura de la siguiente manera:

- **Memoria:** se incluye la última versión de la memoria en formato pdf.
- **Anexos:** se incluye la última versión de los anexos en formato pdf.
- **Código:** se incluye la última versión del código tanto de la aplicación como del servidor.
- **Vídeo:** se incluye una demo vídeo del funcionamiento del producto final.

Enlaces de los materiales del proyecto

- Repositorio del proyecto.
- Vídeo demostración (enlace a YouTube).

Objetivos del proyecto

El objetivo principal que persigue este proyecto es el desarrollo de un entorno domótico que permita la visualización de cámaras en el hogar a través de una aplicación Android, todo ello a través de la comunicación entre un servidor que realice la gestión y el propio dispositivo Android.

El servidor debe establecer la conexión con las cámaras y ser capaz de retransmitir la imagen al dispositivo Android, conectado a la misma red WiFi.

2.1. Objetivos generales

- Desarrollar una aplicación para el entorno Android que permita la monitorización de cámaras dispuestas en un hogar.
- Desarrollar y desplegar un servidor que gestione el entorno domótico (la conexión con las cámaras y con la aplicación Android).
- Facilitar al usuario la monitorización mediante una interfaz fácil y sencilla.
- Utilizar dispositivos de bajo consumo.

2.2. Objetivos técnicos

- Desarrollar un servidor con el lenguaje Python que gestione toda la complejidad.
- Desarrollar una aplicación en el entorno Android con soporte para API 16 y superiores.
- Emplear la herramienta Gradle para la automatización del proceso de construcción y compilación de software de la aplicación Android.

- Utilizar Git como sistema de control de versiones distribuido junto con la plataforma GitHub.
- Aprovechar las posibilidades que ofrecen las herramientas de integración continua como SonarQube en el repositorio.

2.3. Objetivos personales

- Mostrar la capacidad de los conocimientos adquiridos durante el periodo universitario.
- Aprender a manejar metodologías y herramientas innovadoras que son utilizadas en el mercado laboral.
- Mejorar mi habilidad en el desarrollo de aplicaciones en el entorno Android.
- Profundizar en los conceptos aprendidos sobre Python en mi formación universitaria.

Conceptos teóricos

Para comprender el marco teórico del desarrollo de este proyecto, debemos conocer previamente algunos conceptos en los que se basa el mismo.

3.1. Domótica

La domótica [3] según la Asociación Española de Domótica e Inmótica (CEDOM) es el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema. La domótica aporta servicios de gestión energética, seguridad, bienestar y comunicación.

En este proyecto se centra en dar un servicio de seguridad en el que vamos a controlar cámaras en un vivienda y en futuras versiones podrían añadirse otros elementos inteligentes del hogar.

3.2. Android

Android [4] es un sistema operativo (OS) que inicialmente fue desarrollado para dispositivos móviles pero que hoy en día engloba ordenadores, relojes, televisores, tablets, coches, etc. Este entorno ha sido desarrollado por Google cuyo obetivo fue fomentar el uso de una plataforma abierta, gratuita, multiplataforma y muy segura, y es por ello que está basado en Linux (Linux es un núcleo sistema operativo (OS) open source).
Este sistema permite programar aplicaciones empleando una variación de Java llamada Dalvik (o ART a partir de su version 5.0) y proporciona todas

las interfaces necesarias para desarrollar fácilmente aplicaciones que acceden a las funciones del dispositivo (como pudiera ser el GPS, o la memoria entre otros).
Y es por todo esto que es el sistema operativo más usado el mundo.

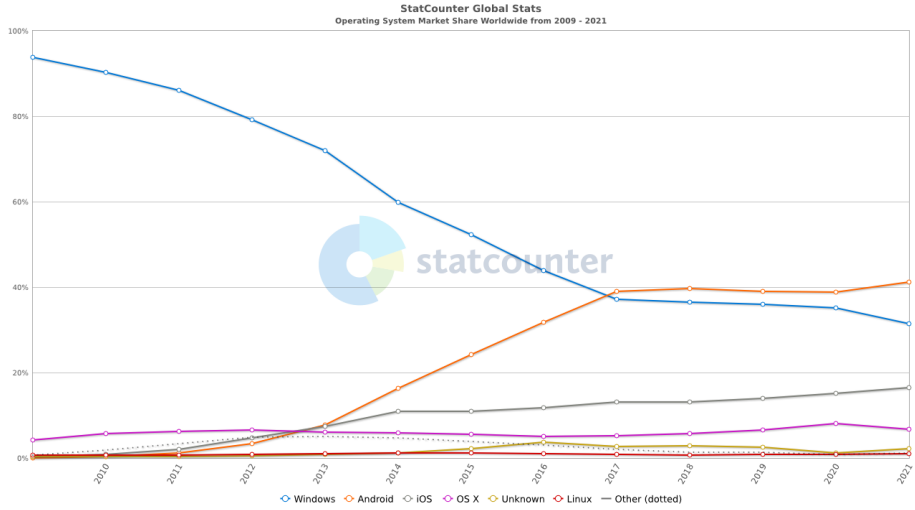

Figura 3.1: Sistemas operativos mas usados desde 2009 a 2021 [1]

3.3. Servidor

Un servidor es una máquina que está integrada en una red informática que se encarga de realizar tareas para los clientes. Su funcionamiento se basa en el modelo Cliente-Servidor (figura 3.2), un servidor provee un servicio y los clientes solicitan recursos mediante peticiones y el servidor provee los recursos con respuestas.

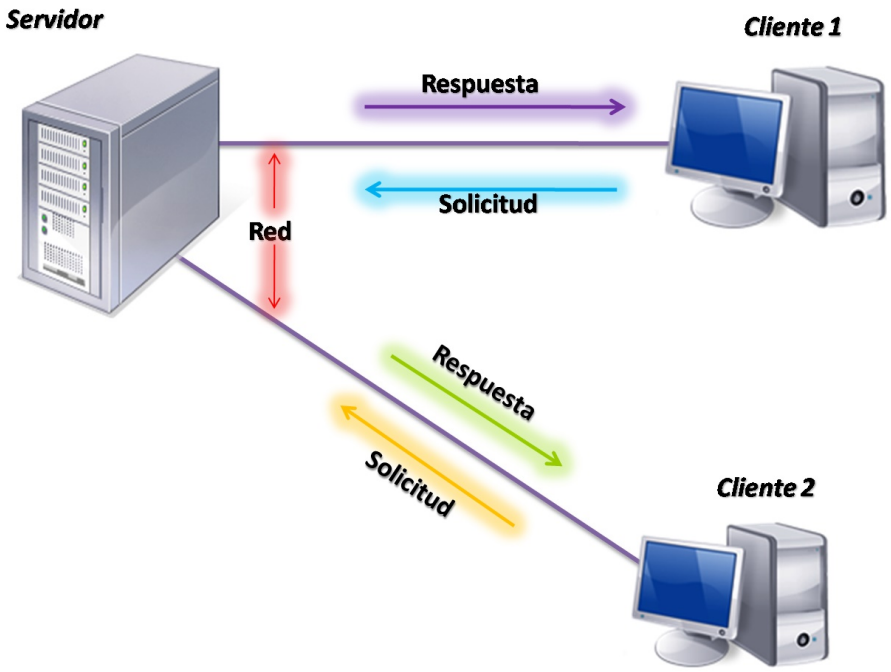

Figura 3.2: Arquitectura Cliente-Servidor

Existen muchos tipos de servidores, tanto servidores web, como FTP, proxy o inlcuso de juegos online. En este proyecto se va a desarrollar un servidor proveedor de servicios, en este caso monitorización de cámaras.

3.4. Python

Python [5] es un lenguaje interpretado, interactivo y orientado a objetos, que incorpora módulos, excepciones, tipado dinámico, tipos de datos de muy alto nivel y clases. Está muy extendido y nos permite realizar cualquiera de nuestros propósitos gracias a su gran cantidad de librerías y programadores. Fue creado a principios de la década de 1990 por Guido van Rossum en Stichting Mathematisch Centrum (CWI) en los Países Bajos como sucesor de un idioma llamado ABC [6].
Este lenguaje [5] nos proporciona una gran biblioteca estándar que abarca áreas como procesamiento de cadenas (ya sea tanto expresiones regulares, Unicode, o incluso cálculo de diferencias entre archivos), protocolos de Internet (como son HTTP, FTP, SMTP, XML-RPC, POP, IMAP,o programación CGI entre otros), ingeniería de software (desde pruebas unitarias, registro,

creación de perfiles, hasta análisis del propio código Python) e interfaces del sistema operativo (tales como llamadas al sistema, sistemas de archivos, incluidos sockets TCP / IP).

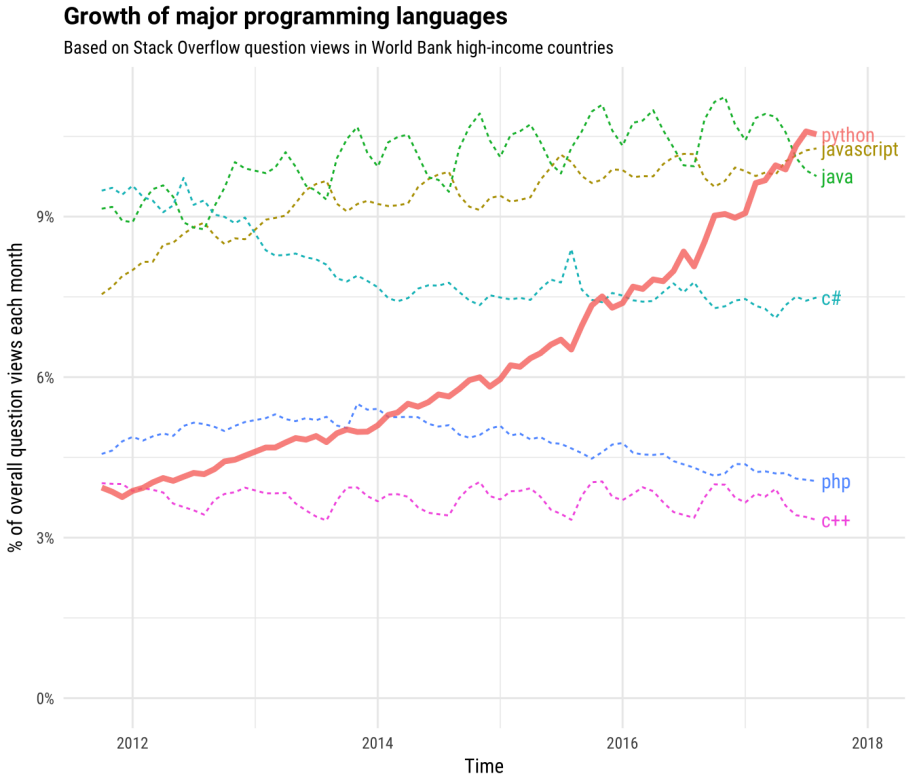

Figura 3.3: lenguajes de programación mas usados desde 2012 a 2018 [2].

3.5. Dispositivos de bajo consumo

Hoy en día los dispositivos de bajo consumo están en auge. Con las nuevas normativas debido al cambio climático las empresas están dedicando sus recursos a diseñar dispositivos de bajo consumo. En este proyecto pensé en la Raspberry Pi ya que tengo una en casa y he realizado algún proyecto ya con ella.

Raspberry Pi

La Raspberry Pi [7] es un ordenador de bajo coste, del tamaño de una tarjeta de crédito. Es un pequeño dispositivo capaz de permitir a perso-

nas de todas las edades explorar la informática y aprender a programar en lenguajes como Scratch y Python. Es capaz de hacer todo lo que se espera de un ordenador de sobremesa, desde navegar por Internet y reproducir vídeo de alta definición, hasta hacer hojas de cálculo, procesar textos y jugar. La Raspberry Pi, sin ventilador y de bajo consumo, funciona de forma muy silenciosa y consume mucha menos energía que otros ordenadores.

En artículo *"Power consumption of the Raspberry Pi: A comparative analysis"* [8] compara los distintos consumos que puede realizar los distintos dispositivos respecto de una Raspberry Pi, la cual es mucho más eficiente que un PC de sobremesa en el orden de consumir hasta 10 veces menos.

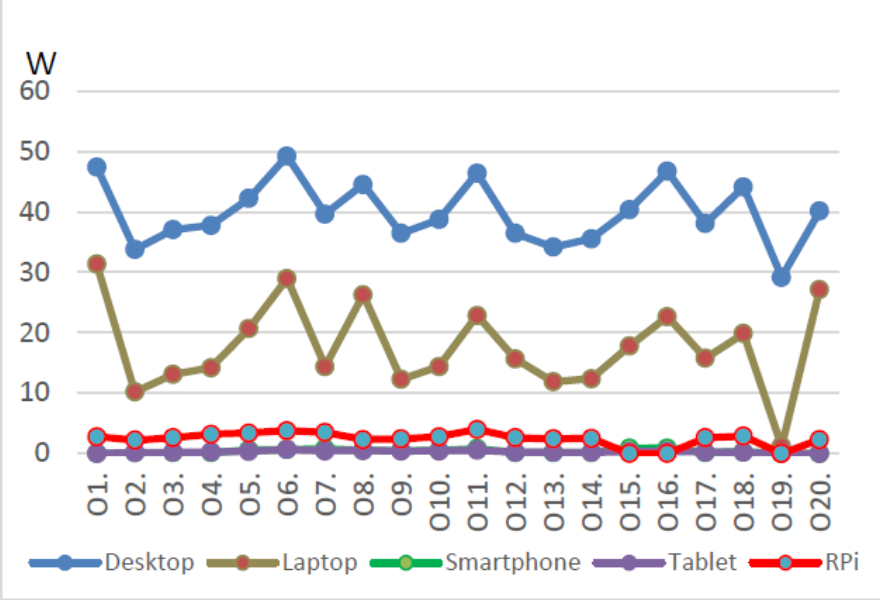

Figura 3.4: Comparación de consumo en vatios (watts) de diferentes dispositivos en distintas pruebas.

La Raspberry Pi es un proyecto de la *Raspberry Pi Foundation*.

Técnicas y herramientas

4.1. Herramientas de control de versiones

Github

GitHub es una plataforma que nos permite gestionar y organizar nuestros proyectos y está basada en la nube, incorporando las funciones de control de versiones que proporciona Git. Esta herramienta posibilita a los desarrolladores poder almacenar y administrar su código, realizar un registro y control de los cambios sobre el código almacenado. Es una de las herramientas más populares entre los desarrolladores, cuenta con más de 100 millones de repositorios, y la mayoría de ellos son de código abierto.
He utilizado GitHub para el alojamiento de mi proyecto en el repositorio "https://github.com/fmv1001/LocalStream".

Git Bash

Git Bash es una aplicación para Windows para la emulación de la línea de comandos de Git a través de una shell. Una shell es un intérprete de comandos que provee una interfaz de usuario para la comunicación con el sistema operativo.
Esta aplicación me ha servido para subir cambios al respositorio de GitHub a través de los comandos git.

4.2. Herramientas de gestión de proyectos

ZenHub

ZenHub es una plataforma para la gestión ágil de proyectos que se integra con github, funcionando como aplicación nativa en su interfaz. Te ayuda a planificar tu proyecto dentro de GitHub, automatiza el flujo de trabajo. Más del 75 % de los desarrolladores que usan ZenHub en sus proyectos dicen que ZenHub mejora su enfoque y ayuda en el envío de un mejor software en un menor periodo de tiempo, y el 65 % informan de proyectos con un mejor alcance.
ZenHub me ha permitido gestionar mi proyecto, ayudándome con la planificación del mismo, para cumplir los tiempos de entrega del mismo.

4.3. Metodologías

Desarrollo por fases

El desarrollo por fases o etapas es un modelo de software en el que las especificaciones no son conocidas desde el inicio, se van desarrollando simultáneamente en las distintas versiones del código, esta metodología es la que se ha llevado a cabo en este proyecto.

4.4. Patrones de diseño

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de diseño de software en una red informática en la que se clasifica a los dispositivos implicados. Un cliente que solicita información (demandante) y el servidor se la proporciona (proveedor de recursos).

Patrón Modelo-Vista-Controlador (MVC)

El patrón MVC es un patrón arquitectónico que separa el sistema en tres capas, los datos, la lógica de la aplicación y la interfaz de usuario. El modelo es la capa de datos, la vista es la interfaz de usuario y el controlador, la lógica de la aplicación.

4.5. Herramientas de evaluación de código

SonarQube

SonarQube es una plataforma de código abierto desarrollada en Java que nos permite realizar análisis de código con diferentes herramientas de forma automatizada. Usa diferentes herramientas de análisis estático de código fuente como pueden ser Checkstyle, PMD o FindBugs para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.
En este proyecto se ha usado la herramienta Sonar-Scanner de SonarQube, para la evaluación del código tanto de la aplicación Android en java como del servidor desarrollado en python.

4.6. Herramientas de documentación

LaTeX

LaTeXes un sistema de software libre de composición tipográfica de alta calidad, orientado a la producción de documentación técnica y científica. LaTeXes el estándar de facto para la comunicación y publicación de documentos científicos gracias a sus características, posibilidades y calidad profesional.
He usado LaTeXpara el desarrollo de tanto este documento como de los anexos.

TEXMAKER

Texmaker es un editor LaTeX, avanzado y multiplataforma. Integra las diferentes herramientas necesarias para desarrollar documentos con LaTeX, todo ello en una sola aplicación. Incluye compatibilidad para unicode, autocompletado, corrección ortográfica, plegado de código, modo de vista continua y tiene integrado un visor de pdf. Texmaker es fácil de usar y de configurar, y está publicado bajo la licencia GPL (open source).

Draw.io

Draw.io es una herramienta de diagramación que nos permite realizar cualquier tipo de diagramas (diagrmas de flujo, de secuencia, de red, UML, etc). Es muy interesante ya que proporciona elementos UML, muy necesarios en proyectos software.

4.7. Lenguajes de programación

Python

Python es un lenguaje de programación ampliamente utilizado por empresas de casi todo el mundo para la construcción de aplicaciones web, análisis de datos, automatización de operaciones y creación de aplicaciones empresariales con alta fiabilidad y escalabilidad.
Para desarrollar el servidor he utilizado la versión 3.0 del lenguaje Python.

Java

Java es un lenguaje de programación orientado a objetos y basado en clases que se usa para el desarrollo de aplicaciones, diseñado para tener la menor cantidad de dependencias de implementación posibles. Además es uno de los lenguajes de programación utilizados para el desarrollo de aplicaciones para el sistema operativo Android. El código compilado de Java puede ejecutarse en cualquier plataforma que tenga instalada la máquina virtual java, sin la necesidad de volver a ser compilado.
Java es el lenguaje que he escogido para la construcción de la aplicación Android debido a mi familiaridad con este lenguaje.

SQL

SQL (Structured Query Language) es un lenguaje declarativo que ha sido diseñado para la administración y recuperación de información de sistemas de gestión de bases de datos relacionales. Este lenguaje es conservado por el organismo ANSI (American National Standards Institute).
El lenguaje SQL ha sido muy útil en este proyecto a la hora de respaldar al información necesaria tanto en el servidor como en la aplicación.

4.8. Entornos de desarrollo integrado (IDE)

Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para el sistema operativo Android y está basado en IntelliJ IDEA (IDE de Java desarrollado por JetBrains). Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, entre otras tenemos: un sistema

de compilación flexible basado en Gradle o la integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra.
Android Studio es el IDE oficial para desarrollar aplicaciones en Andorid y por eso lo he escogido para mi proyecto.

Visual Studio Code

Visual Studio Code es un entorno de desarrollo desplegado por Microsoft para todas las plataformas (Windows, Linux y macOS). Incluye soporte para depuración, resaltado de sintaxis, control integrado de Git, finalización inteligente de código, fragmentos y refactorización de código. Visual Studio Code es un proyecto open source aunque la descarga de forma oficial es bajo software privativo de Microsoft incluyendo algunas de sus características personalizadas.
Para la fase de desarrollo del servidor he utilizado esta aplicación open source de Microsoft.

4.9. Herramientas de automatización de compilación del código

Gradle

Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los scripts de compilación de Gradle se escriben utilizando Groovy o Kotlin DSL (Domain Specific Language). Gradle tiene una gran flexibilidad y nos deja hacer usos otros lenguajes y no solo de Java, también cuenta con un sistema de gestión de dependencias muy estable. Además es el sistema de compilación oficial para Android y cuenta con soporte para diversas tecnologías y lenguajes.

4.10. Protocolos de comunicación

UDP

El protocolo de datagramas de usuario (User Datagram Protocol o UDP) es un protocolo de la capa de transporte basado en el intercambio de datagramas (un datagrama es un paquete de datos que compone el mínimo

bloque de información en una red). Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, gracias a la incorporación suficiente de información de direccionamiento en su cabecera. Una de sus ventajas es que no implementa acuse de recibo.
Gracias a este protocolo el servidor envía las imágenes al dispositivo móvil Android sin que este confirme que lo ha recibido (no implementa acuse de recibo), perdiendo rendimiento, y el envío es no bloqueante para el hilo que envía.

TCP / IP

TCP es un protocolo del nivel de transporte empleado por aplicaciones que requieren entrega garantizada (acuse de recibo). Se trata de un protocolo que implementa control de flujo para gestionar tiempos de espera, retransmisiones, entre otros. TCP establece un enlace virtual full duplex entre dos puntos finales. Cada punto final está formado por una dirección IP y un puerto TCP.
Para que el servidor y la aplicación se comuniquen a la hora de realizar alguna tarea, este protocolo ha permitido que en ningún momento se pierda la conexión y en ese caso los dos serán conscientes y actuaran para salvaguardar la información y mantener un sistema coherente.

Sockets

Los sockets hacen posible la comunicación entre conectores de procesos, cada uno de ellos tiene asociada una dirección, un puerto, y un protocolo asociado (UDP o TCP). Los sockets nos permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los procesos que se denomina programa "cliente". El segundo proceso espera a que otro inicie la comunicación, por este motivo se denomina programa "servidor". Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.
He usado los sockets junto a los protocolos mencionados en esta misma sección para la comunicación entre el servidor y el cliente (dispositivo Android).

RTSP

El Protocolo de Transmisión en Tiempo Real (Real Time Streaming Protocol o RTSP) es una tecnología de vídeo de probada eficacia. Es un

protocolo usado en la capa de aplicación para la transferencia de datos de medios de vídeo en tiempo real. Permite el control en la transmisión de audio y vídeo entre puntos finales. También facilita el envío de contenidos en streaming de baja latencia vía Internet. El puerto que usa por defecto RTSP es el 554.
En este proyecto se ha usado este protocolo para establecer la conexión con las cámaras, a través de RTSP OpenCV obtiene el control.

4.11. Librerías

SQLAlchemy

SQLAlchemy es el conjunto de herramientas SQL para Python además de ser el mapeador relacional de objetos que ofrece toda la potencia y flexibilidad del lenguaje SQL. Aporta un complejo conjunto de patrones de persistencia muy conocidos, diseñados para un acceso eficiente y de alto rendimiento a la base de datos, adaptados a un lenguaje pitónico y sencillo.

OpenCV

OpenCV es una biblioteca de código abierto para visión artificial, aprendizaje automático y procesamiento de imágenes. Es compatible con gran variedad de lenguajes de programación como Python, C++, Java, entre otros.
OpenCV me ha permitido mantener una conexión con las cámaras disponibles y a su vez obtener las imágenes de estas.

SQLite

SQLite es una biblioteca que hace uso de lenguaje C para implementar un pequeño motor de base de datos SQL haciendo que sea rápido, autónomo, manteniendo una alta fiabilidad y con todas las funciones necesarias para una base de datos. SQLite es el motor de base de datos más utilizado en el mundo. La biblioteca *SQLite* está incluida en todos los teléfonos móviles además de en la mayoría de los ordenadores, y viene integrado en un número inmenso de aplicaciones de uso diario. Además, SQLite es la API para utilizar una base de datos en Android.

Android Support Library

El paquete Android Support Library es un conjunto de bibliotecas que proporcionan versiones compatibilidad con versiones anteriores de APIs de la estructura Android, así como algunas características que únicamente están disponibles por medio de las API de esta biblioteca. Cada biblioteca del paquete es compatible con una versión específica de la API de Android.

AndroidX

AndroidX es el proyecto de open source que Android utiliza para desarrollar, probar, empaquetar, versionar y liberar bibliotecas dentro de Jetpack. AndroidX ha mejorado y sustituido al paquete Android Support Library.

JavaFx

JavaFX a través de paquetes gráficos y multimedia nos permite diseñar, crear, probar, depurar e implantar apps clientes que funcionaran de coherentemente en las distintas plataformas. El SDK de JavaFX-Android posee una implementación de JavaFX para ser ejecutado en Android, junto con otras herramientas para la construcción de paquetes en Android.

JCodec

JCodec es un librería Java que implementa códecs de vídeo y audio. Actualmente JCodec tiene soporte en Android, y es muy útil si manejas APIs Android de versiones antiguas. Esta librería me ha permitido poder crear un archivo de vídeo sobre las imágenes que llegan desde el servidor.

Material Design

Material Design es un lenguaje de diseño creado por Google orientado a Android, que admite movimientos y pulsaciones táctiles en pantalla gracias a funciones y movimientos que imitan objetos del mundo real. Material design es una guía completa para el diseño visual, de movimiento y de interacción en todas las plataformas y dispositivos.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación.

5.1. Elección del tema

Siempre me ha gustado el hecho de desarrollar una aplicación Android y cuando estuve mirando tema para mi proyecto, me llamó la atención el hecho de desarrollar algo que me sirviera en mi día a día. Así que decidí realizar una aplicación para el sistema Android dado lo extendido que está. También decidí darle un enfoque domótico y hacer un servidor dado que una asignatura que he cursado en este último curso trataba el tema y me llamó mucho la atención. Y todo esto me llevó a realizar el proyecto que expongo a continuación.

5.2. Comienzo del proyecto

Una vez escogido el tema, tocaba diseñar la estructura del proyecto. Decidí que la mejor opción sería separar el servidor en un dispositivo y la aplicación en otro, ya que así el dispositivo no cargaría con toda la carga computacional que supondría para él, dado que estos no disponen de mucha batería, aplicando el modelo cliente-servidor.

Ahora tocaba decidir donde desarrollar el servidor y en que lenguaje. En este paso, me decanté por el lenguaje Python ya que está muy extendido

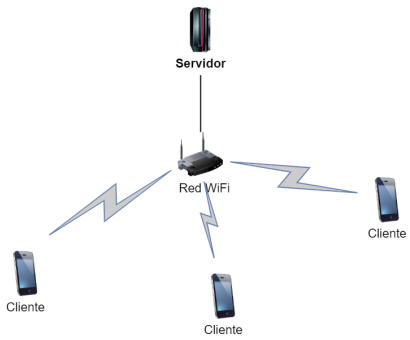

Figura 5.1: Arquitectura modelo Cliente-Servidor.

(llegando a ser uno de los lenguajes de programación mas usados del mundo [2]) y estaba muy familiarizado con él debido a que lo he usado en numerosas ocasiones.
Después tenía que asegurarme de poder realizar la conexión con una cámara a través de Python. Tras una búsqueda intensa por internet encontré la librería OpenCV. Para la realización de este proyecto me compré una cámara IP, pero tras unos días no conseguía acceder a ella desde OpenCV con python. Para no perder demasiado tiempo contacté con la empresa que me había vendido la cámara y les comenté mi problema. Rápidamente me contestaron y me proporcionaron la dirección que debía usar, y gracias a ello pude realizar una conexión python-cámara. En la figura 5.2, podemos ver los diferentes intentos realizados en la conexión Servidor-Cámara. Algunos intentados bajo el protocolo http y otros bajo el protocolo rtsp, el que finalmente funcionó y se ha usado.

ipcam.py

```
import cv2


print("1")
cap = cv2.VideoCapture('rtsp://192.168.18.17:8080/h264_ulaw.sdp')
cap = cv2.VideoCapture('rtsp://192.168.18.17:8080/h264_pcm.sdp')
VideoCapture:cap.open('rtsp://admin:1@192.168.18.51/user=admin_password=tlJwpbo6_channel=1_stream=0.sdp')

cap = cv2.VideoCapture('http://192.168.18.17:8080/onvif/device_service')
cap = cv2.VideoCapture("http://192.168.18.17:8080/browserfs.html")
cap = cv2.VideoCapture("http://ff972057153fbd6c:1234@192.168.18.51:8088/mjpeg.cgi?user=ff972057153fbd6c&password=1234&channel=0&.mjpg")

#while (True):
print("2")
ret, frame = cap.read()
#print("3")
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
#print("4")
cv2.imwrite('C:/Users/franm/Desktop/TFG/ProyectoGit/RaceStream/app/app/src/main/res/img.png',gray)
#print("5")
cap.release()
```

Figura 5.2: Primeras conexiones con la cámara.

Una vez realizada dicha conexión ya podía empezar con el siguiente punto del proyecto.

5.3. Desarrollo del servidor

A la hora de desarrollar el servidor pensé que debía de implementarlo en algún dispositivo de bajo consumo, ya que iba a estar operativo las 24 horas del día. Por ello pensé en la Raspberry Pi, que ya había realizado algún proyecto personal con ella y se puede ejecutar python en ella.

Figura 5.3: Raspberry Pi 4.

Ahora ya podía empezar a programar. Inicialmente realicé una conexión mediante Sockets con el servidor esperando a la conexión TCP / IP de la aplicación. En este momento cogí ideas del libro *Foundations of Python Network Programming* [9]. Poco a poco fui añadiendo más funcionalidad, como por ejemplo, permitir añadir las cámaras o eliminarlas, o poder para el servidor entre otras. Para poder mantener persistencia en la información, decidí crear un ORM para la base de datos de la cámara (en la tabla 5.1 podemos ver ejemplos de instancias en dicha base de datos, exactamente igual a la tabla 5.2 de la aplicación).

Un ORM o Object Relational Mapping (Mapeador de Objetos Relacionales) es una estructura que permite convertir (mapear) datos de objetos en el formato necesario para una base de datos, vinculando los mismos (el objeto con los datos en la base de datos). Este mapeo puede realizarse a

Id	Ip Address	Name	Port
0	192.168.0.30:8080	Cámara 1	9999
1	192.168.0.31:8080	Cámara 2	9998
2	192.168.0.32:8080	Cámara 3	9997

Tabla 5.1: Ejemplo de la base de datos de *cámaras* en el servidor.

cualquier base de datos.
Para ello usé **SQLAlchemy**, debido a que esta librería dispone de su propio ORM, el cual es capaz de mapear tablas a clases Python y convertir automáticamente las llamadas a funciones dentro de dichas clases a sentencias en lenguaje SQL.

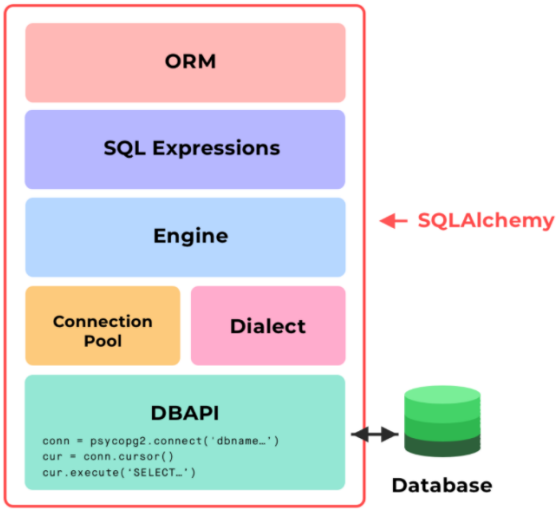

Figura 5.4: Diagrama de la librería SQLAlchemy.

Para el envío de imágenes de que se obtenían desde la conexión con la cámara que mantenía el servidor gracias a la librería OpenCV, una vez obtenía el frame, realizamos la serialización del mismo con la funcion *imencode*. Esta función comprime la imagen y la almacena en el búfer de memoria redimensionándola para ajustarla al resultado. Tras su serialización enviamos el frame mediante un socket UDP, para que en caso de que se pierda un paquete en el envío no se pierda fluidez en la imagen con respecto a la consistencia. Todo ello se realiza desde un hilo en segundo plano que es controlado por el hilo principal del servidor.

Por temas de consumo energético se tomó la decisión de mantener la conexión de la cámara pero sólo obtener la imagen cuando la aplicación lo requiere (cuando el usuario pincha en el icono para ver la cámara).

5.4. Desarrollo de la aplicación

El desarrollo de la aplicación Android fue lo que más trabajo me ha costado y es en este donde más problemas me he encontrado.

Antes de comenzar a escribir código, realicé un cursillo para Android Studio, a continuación dejo el enlace: *Curso de programación Android desde cero*.

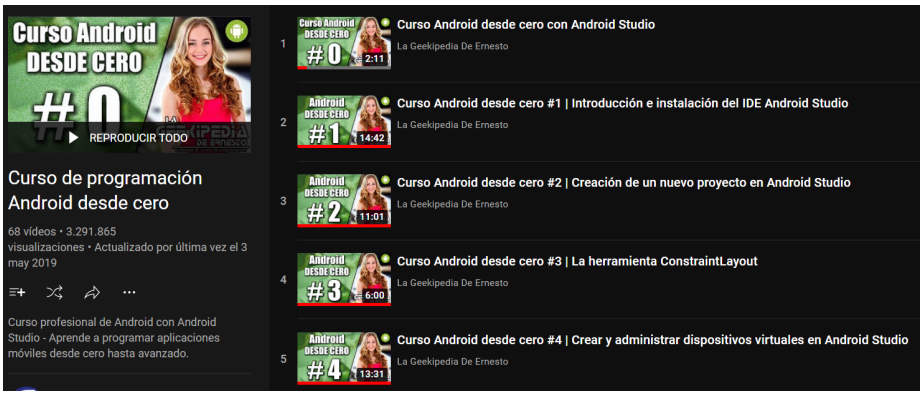

Figura 5.5: *Curso de programación Android desde cero.*

En un principio creé una única actividad en la que utilizaba un botón para conectarme al servidor y recibir un pequeño texto. Es aquí donde me encontré el primer problema, ya que al enviar datos desde un socket de python y otro Java, los datos no se serializan igual. Para solucionarlo tuve que convertir a bytes el texto en python y recibirlo en Java en un buffer de bytes y convertirlo a String.

Una vez conseguido enviar texto desde las dos partes, y recibirla con éxito, comencé con la parte en que enviamos imágenes a la aplicación. Y aquí tuve muchos problemas, ya que OpenCV desde python codificaba las imágenes en un formato que luego desde Java no conseguía descifrar. Tras mucho tiempo buscando la solución, encontré la forma adecuada, el paquete *android.graphics.Bitmap*. Primero creé un *DatagramPacket* con un buffer de

bytes para la entrada de los datos, y con la clase *BitmapFactory* y la función *decodeByteArray* pude mostrar la imagen enviada desde el servidor, convirtiéndola en un *BitMap*. También hubo que usar un manejador o Handler para permitir que el hilo secundario actualice la interfaz.

Cuando la aplicación recibe una imagen del servidor la actualiza sobre un *ImageView*. Intenté que la imagen se visualizara sobre un VideoView pero no aceptaba la imagen que llegaba de un socket, requería un documento de vídeo o una conexión de un vídeo por HTTP.

Después de conseguir la conexión de vídeo, diseñé la interfaz de usuario. Para ello usé el paquete *androidx.navigation* con un menú desplegable. En este momento encontré mas problemas, dado que el menú desplegable utiliza Fragments de Android y no se pueden comunicar directamente con la actividad principal. Para ello use la clase *ViewModel* del paquete *androidx.lifecycle*, para la comunicación entre fragments y con la actividad principal.

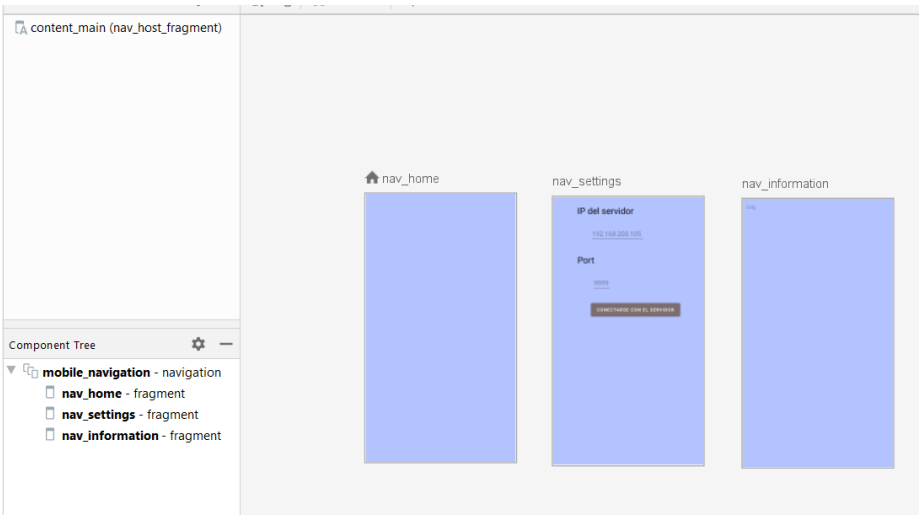

Figura 5.6: Archivo navigation.

Tras conseguir implementar el diseño, se fue añadiendo funcionalidad a la aplicación (podemos ver el resultado en la figura 5.7):

1. Añadir las cámaras.

2. Eliminar las cámaras.

3. Parar el servidor.
4. Desconectarse del servidor.
5. Grabación de un pequeño fragmento de vídeo.
6. Visualización del log del sistema.

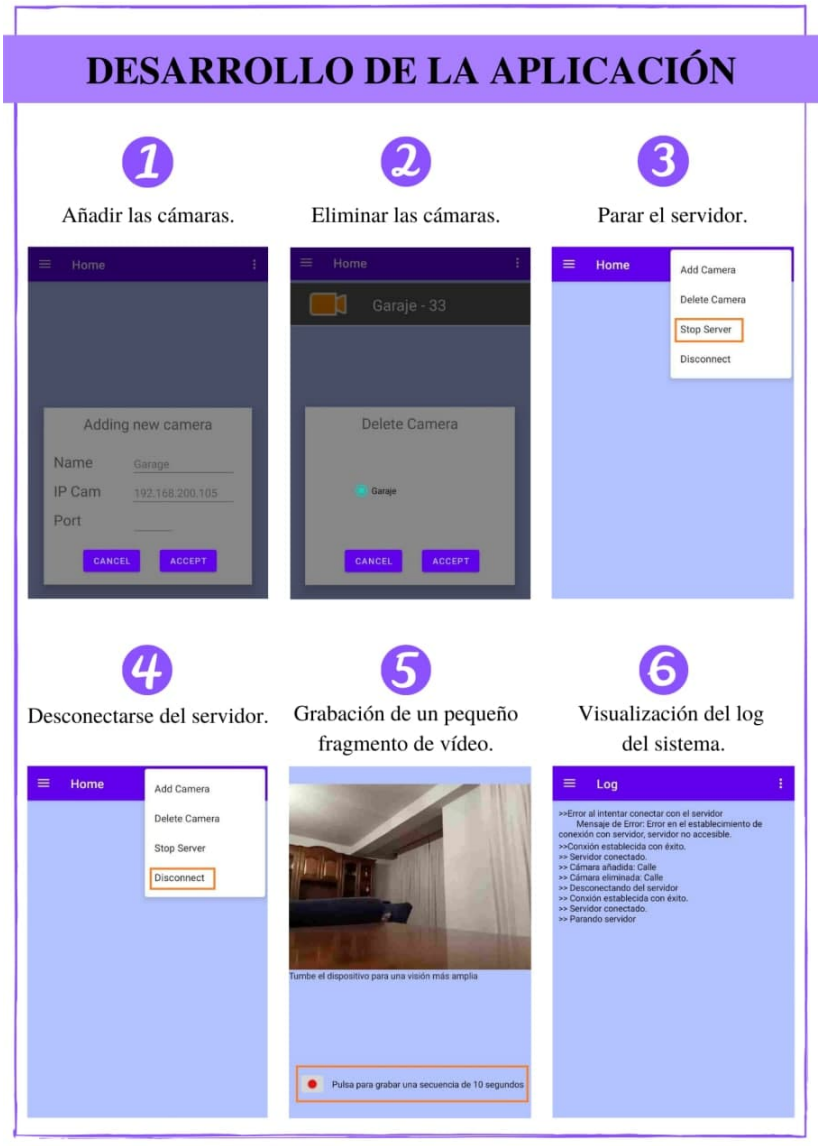

Figura 5.7: Pantallas de la aplicación según la funcionalidad.

En cuanto a la implementación de la grabación de un pequeño fragmento de vídeo, debo comentar que encontré algunos problemas. Como la aplicación da soporte a la gran mayoría de los dispositivos Android (99 %), el API de android que se utiliza (*API 16*) es antiguo y no dispone de muchas librerías compatibles. Buscando librerías encontré JCodec, que daba soporte para Android y lograba realizar la acción deseada. Añadirla al proyecto fue fácil y gracias a esta librería logré realizar la grabación. Para ello usé su clase *AndroidSequenceEncoder*, que permite la creación de vídeos a partir de imágenes, o en nuestro caso de *Bitmaps*.

Persistencia en la aplicación

Para lograr la persistencia en Android debemos mantener la información en una base de datos y **SQLite** es la API en Android el uso de bases de datos.
En la app hemos creado dos tablas:

- Para las cámaras la tabla 5.2. En dicha tabla tenemos 4 comunas:
  - *Id* para el identificador y *primary key*.
  - *Ip Address* con la ip de la cámara.
  - *Name* para el nombre adjudicado a la cámara.
  - *Port* con el puerto por el que el cliente va a recibir sus imágenes.
- Para la IP del servidor la tabla 5.3. En esta tabla tenemos 1 columna, *Ip Address* con la ip del servidor.

Id	Ip Address	Name	Port
0	192.168.0.30:8080	Cámara 1	9999
1	192.168.0.31:8080	Cámara 2	9998
2	192.168.0.32:8080	Cámara 3	9997

Tabla 5.2: Ejemplo de la base de datos de *cámaras* en la app.

Para esta base de datos he creado una interfaz para el acceso a la misma, para encapsular dicha responsabilidad, con la ayuda de la clase *SQLiteOpenHelper*.

Ip Address
192.168.0.30

Tabla 5.3: Ejemplo de la base de datos de *dirección ip* del servidor en la app.

5.5. Evaluación del código

Cuando ya disponía prácticamente de una versión final del código, realicé un análisis del mismo con la herramienta SonarQube. Esta herramienta analiza tanto fragmentos de código sensibles a la seguridad, como *code smells*, duplicaciones de código, vulnerabilidades, bugs, y posibles errores entre otros.
Inicialmente analicé por separado la aplicación y el servidor. Aquí podemos ver unas imágenes que ilustran el resultado (figuras 5.8 y 5.9):

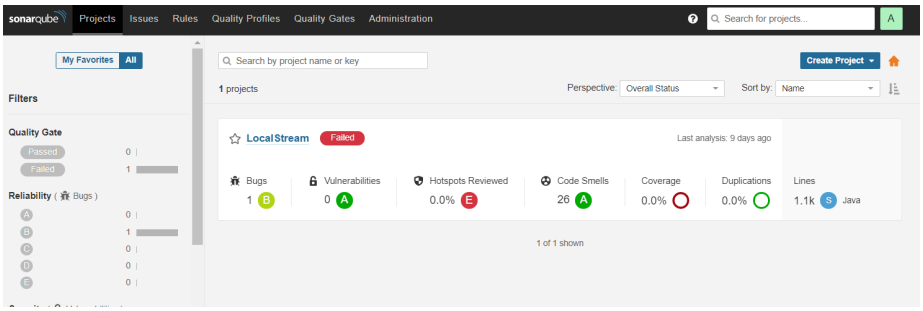

Figura 5.8: Resultado del escaneo del código de la aplicación con SonarQube.

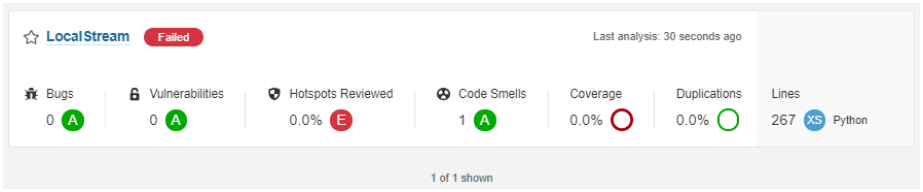

Figura 5.9: Resultado del escaneo del código del servidor con SonarQube.

Podemos apreciar que al analizar el código las primeras veces obtendremos una calificación global "*failed*", debido a que algunos de los campos que se han comprobado no es apto (podemos ver en la figura 5.8 el campo "*Hotspots Reviewed*" tiene una calificación "*E*"). Tras refactorizar el código,

ya lo analicé todo junto, obteniendo la calificación más alta (A) en todos los campos y la calificación global *Passed* (figura 5.10).

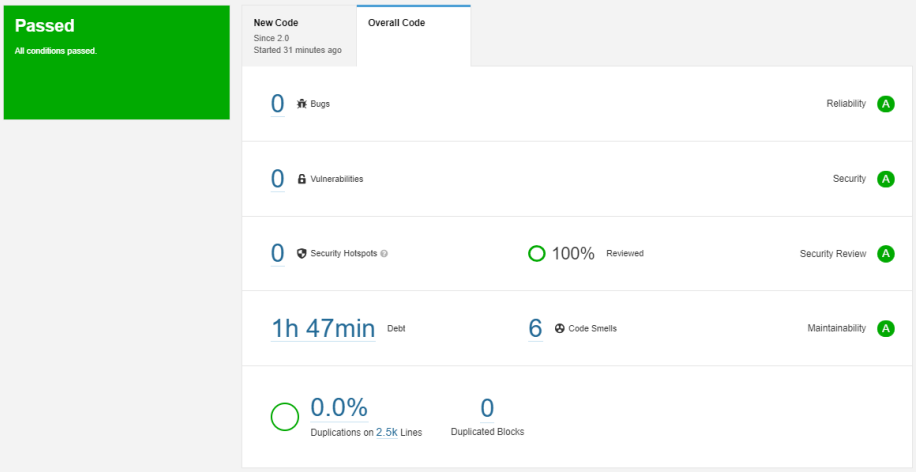

Figura 5.10: Resultado del escaneo del código con SonarQube.

Gracias a esta herramienta el proyecto ha obtenido una calidad de código superior, siendo menos vulnerable y más seguro para el usuario final.

5.6. Documentación

Para la documentación he utilizado el editor de textos offline *TEXMAKER*, que es el recomendado por la universidad en el repositorio con la plantilla de LaTeX. Gracias a este editor y a la plantilla proporcionada por la universidad, he podido desarrollar la memoria y anexos con más facilidad.

Cabe destacar que a la hora de hacer las referencias bibliográficas, se me presentó el problema de no visualizarse las URLs de las mismas. Para ello me dirigí a la página del repositorio de la plantilla utilizada [10], y en el apartado de *Issues* vi que aún estaba abierto un *issue* (*No aparecen las url en las referencias bibiográficas*, el 18) que trataba este problema. En él se explicaba como solucionar el problema:

1. Cambiar en el archivo *memoria.text* el estilo de la bibliografía.
2. Y por último descargar el archivo *IEEEtran.bst* y dejarlo en la misma carpeta que el archivo de la memoria.

Tras realizar estos cambios, ya se aparecían las URLs, y con ello ya disponía de la memoria y los anexos terminados y conformes a los estándares.

Logo App

También se ha diseñado un logo para el despliegue de la aplicación. Podemos verlo en la siguiente imagen (figura 5.11):

Figura 5.11: Logo de la aplicación.

Trabajos relacionados

6.1. IP Cam Viewer Lite

IP Camera Viewer Lite (figura 6.1) es una aplicación android que permite acceder y controlar remotamente su cámara IP, grabador de vídeo digital, grabador de red y/o cámara web. Es un proyecto realizado por *hit-mob.com* con la colaboración de *WordPress*.

Figura 6.1: Logo de la aplicación *IP Cam Viewer Lite*.

Esta aplicación es muy parecida a la que en este proyecto se ha desarrollado pero es muy compleja de utilizar, y requiere tener ciertos conocimientos sobre la red de internet y sus protocolos y conexiones.

Puedes encontrar dicha aplicación en el siguiente enlace: *IP Cam Viewer Lite*.

6.2. tinyCam Monitor PRO

tinyCam Monitor (figura 6.2) es una aplicación desarrollada para el sistema Android diseñada con el fin de poder vigilar, controlar y grabar remotamente vídeo digital para cámaras de red o IP (ya sean privadas o públicas), DVRs y/o codificadores de vídeo. Esta aplicación ha sido desarrollada por *Tiny Solutions LLC*.

Figura 6.2: Logo de la aplicación *tinyCam Monitor PRO*.

tinyCam Monitor PRO tiene una interfaz de usuario mucho más simple y fácil de usar que IP Cam Viewer Lite pero es de pago.

Puedes encontrar dicha aplicación en el siguiente enlace: *tinyCam Monitor PRO*.

6.3. Ventajas y debilidades respecto a la competencia

Características	AppFran	IP Cam Viewer Lite	tinyCam Monitor PRO
Servidor en local	✓	×	×
Interfaz de usuario	✓	×	✓
Gratuito	✓	✓	×
Log	✓	×	×
Plataformas	Android	Android	Android

Tabla 6.1: Comparativa de las características de los proyectos.

Las principales fortalezas del proyecto son:

- **Servidor en local**: en este proyecto no se envían las imágenes a ningún servidor remoto, no sale de nuestro router la información. Esto es una ventaja ya que muchas empresas dedicadas a la monitorización de cámaras primero envían las imágenes a sus servidores que pueden estar en cualquier lugar del mundo (normalmente china) y desde ese servidor envían las imágenes a nuestra aplicación conectada a Internet, con la consiguiente pérdida de rendimiento.
- **Interfaz de usuario**: la interfaz de usuario de la aplicación es muy intuitiva, sencilla y con una curva de aprendizaje rápida y accesible a todo el mundo, incluso sin conocimiento previo del tema.
- **Aplicación gratuita**.

Las principales debilidades son:

- Tiene menos funcionalidades que la competencia
- Menor calidad de imagen.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Procedo a mostrar las conclusiones que se derivan del desarrollo del presente proyecto:

- El objetivo general del proyecto se ha cumplido con éxito. Se ha desarrollado un entorno en el que se pueden monitorizar cámaras dispuestas en un hogar, manteniendo una interfaz de usuario sencilla y con una corta curva de aprendizaje, llegando a un gran número de dispositivos (99,8 % de los dispositivos Android).
- Se ha conseguido desplegar el servidor sobre un dispositivo de bajo consumo (Raspberry Pi).
- Gracias a la realización del proyecto he profundizado en el conocimiento sobre conexiones con sockets, aplicándolo a un entorno real.
- Este proyecto ha abarcado una buena parte de los conocimientos que he obtenido durante mi estancia en la universidad. Pero no sólo he hecho uso de ellos, si no que también he estudiado y adquirido nuevos conocimientos requeridos para la realización del proyecto. Entre ellos se encuentran: Android, OpenCV, LaTeX, SonarQube o RTSP.
- Manejar la plataforma SonarQube para el análisis del código me ha ayudado a la detección temprana fallos, defectos y vulnerabilidades en el software, generando un código de mayor calidad y evitando que exista un comportamiento indefinido que pueda afectar a los usuarios finales.

- Gracias a la investigación que ha requerido la realización del proyecto, se ha aprendido a realizar rápidas y eficientes búsquedas bibliográficas.
- Durante el periodo de desarrollo de este proyecto se han manejado un gran numero de tecnologías y herramientas. Todas ellas han ayudado a mejorar la calidad del mismo. No obstante, algunas de ellas han conllevado una sobrecarga de trabajo importante. Pese a ello, el conocimiento adquirido será de gran utilidad en el futuros en otros proyectos.

7.2. Líneas de trabajo futuras

A continuación vamos a comentar una a una las posibles mejoras o continuaciones aplicables al presente trabajo.

Servidor remoto

Podría desarrollarse un servidor en la nube para poder acceder a las cámaras desde cualquier parte del mundo con sólo una conexión a Internet.

Vista previa

Podría implementarse sobre la aplicación para que antes de entrar en la cámara se vea en pequeño la imagen.

Detección de movimiento

Detección de movimiento cuando algo se mueva en el rango que se este visualizando.

Notificaciones

Notificaciones en la aplicación.

Añadir otros dispositivos del hogar para controlar y/o monitorizar

Dado que este proyecto tiene un pequeño enfoque domótico, se podrían añadir fácilmente otros elementos para ser controlados y monitorizados, ya sea la calefacción (poder programarlas, encenderla o apagarla desde el teléfono), la iluminación, una lavadora, las persianas (programar cuando

subirlas y bajarlas para que el sol caliente la casa por la mañana en invierno, pero no entre demasiado el frío por la noche), u otros dispositivos inteligentes del hogar.

Implementar motorización desde PC

Se podría implementar una herramienta en Windows o Linux para monitorizar el sistema, a la vez que en android.

Bibliografía

[1] Statcounter, "Operating system market share worldwide," 2021. [Online]. Available: https://gs.statcounter.com/os-market-share#yearly-2009-2021

[2] D. Robinson, "The incredible growth of python," 2017, [Internet; accedido 01-septiembre-2021]. [Online]. Available: https://stackoverflow.blog/2017/09/06/incredible-growth-python/

[3] A. E. de Domótica e Inmótica CEDOM, "Qué es domótica," 2020. [Online]. Available: http://www.cedom.es/sobre-domotica/que-es-domotica

[4] D. Robledo, *Desarrollo de aplicaciones para Android I.* Ministerio de Educación, Cultura y Deporte, 2016.

[5] P. S. Foundation, "Preguntas frecuentes generales sobre python," 2021. [Online]. Available: https://docs.python.org/es/3/faq/general.html#why-was-python-created-in-the-first-place

[6] ——, "Historia y licencia," 2021. [Online]. Available: https://docs.python.org/3/license.html

[7] R. P. FOUNDATION, "What is a raspberry pi?" 2021. [Online]. Available: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

[8] G. Bekaroo and A. Santokhee, "Power consumption of the raspberry pi: A comparative analysis," *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pp. 361–366, 2016.

[9] B. Rhodes and J. Goerzen, *Foundations of Python Network Programming*. Apress, 2014.

[10] U. de Burgos, "Plantilla latex," 2017. [Online]. Available: https://github.com/ubutfgm/plantillaLatex