



ugr | Universidad
de **Granada**

TRABAJO FIN DE MÁSTER
INGENIERÍA INFORMÁTICA

Sistema de videovigilancia con una Raspberry PI

(SIVIRA)

Autor

Jonathan Martín Valera (alumno)

Directores

Juan Julián Merelo Guervós (tutor)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, septiembre de 2019

Agradecimientos

Después de todo este periodo de formación académica, puedo mirar hacia atrás y hacer una retrospectiva de todo lo sucedido en estos años.

Ha sido una etapa bastante dura e intensiva; una etapa donde se ha tenido que superar grandes obstáculos y emplear muchísima dedicación y esfuerzo. Puedo decir con satisfacción que ha valido la pena. Considero que en esta etapa he adquirido una gran madurez mental, en la que no solo he aprendido algunos conceptos específicos, sino que que puedo decir que he aprendido a pensar y a ser autocrítico con el trabajo.

Gracias al grado y máster de ingeniería informática, veo la realidad de otra forma, con capacidad de poder realizar cualquier reto que me pueda proponer. Si tuviera que resumir en una frase todo lo aprendido en estos años, sería: '*No esperes a que nadie te diga como hacer las cosas, simplemente investiga, experimenta y saca tus propias conclusiones*'.

Este trabajo fin de máster es un ejemplo de ello, una idea hipotética, en la que he investigado, aprendido lo necesario para llevarla a cabo e implementado con éxito.

Quiero agradecer a todos los profesores que he tenido a lo largo de estos años, con especial atención a mi tutor Juan Julián Merelo Guervós, que me ha enseñado en este último año a investigar y conocer una gran cantidad de herramientas actuales que me están siendo de gran utilidad en mi día a día laboral.

Finalmente, agradecer a mis padres la ayuda que, a su forma cada uno, me han proporcionado en todo este periodo, a mis tíos por apoyarme, motivarme y confiar siempre en mí, y especialmente a mi novia Nerea, que me ha estado animando todo este tiempo y dando apoyo moral además de ser fuente de inspiración y superación en mí.

Sistema de videovigilancia con una Raspberry PI

Jonathan Martín Valera (alumno)

Estudiante de Informática en Escuela Técnica Superior Informática y Telecomunicación, Universidad de Granada, 18071, Granada, España

Resumen

Palabras clave: videovigilancia, sistema de seguridad, raspberry PI.

Este proyecto se basa en el desarrollo e implementación de un sistema de videovigilancia de bajo coste, haciendo uso de una raspberry PI y sus componentes. La idea es tener una raspberry PI equipada con una cámara y un sensor de movimiento para poder utilizarla como sistema de seguridad, controlada a través de un dispositivo móvil usando la aplicación de telegram.

La aplicación consta de una serie de modos (manual, automático y streaming) con el que poder vigilar la zona que deseas, y envía alertas automáticas a tu móvil en el caso de detectar alguna actividad sospechosa, capturar imágenes o grabar vídeos de forma manual, o visualizar la retransmisión en directo de la visualización de la cámara, y todo esto se puede controlar desde cualquier localización.

Abstract

Keywords: video surveillance, security system, raspberry PI.

This project is based on the development and implementation of a low-cost video surveillance system, using a raspberry PI and its components. The idea is to have a PI raspberry equipped with a camera and a motion sensor to be used as a security system, controlled through a mobile device using the telegram application.

The application consists of a series of modes (manual, automatic and streaming) with which you can monitor the area you want, and send automatic alerts to your mobile in the event of detecting any suspicious activity, capture images or record videos manually, or view the live broadcast of the camera display, and all this can be controlled from any location.

Yo, **Jonathan Martín Valera**, alumno de la titulación máster de ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77144272-H, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jonathan Martín Valera

A handwritten signature in blue ink, reading "Jonathan", is enclosed within a blue oval. The oval is roughly drawn with a computer program and has a slightly irregular shape.

Granada a 24 de agosto de 2019.

D. Juan Julián Merelo Guervos (tutor), Profesor del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado sistema de videovigilancia con una Raspberry PI, ha sido realizado bajo su supervisión por Jonathan Martín Valera (alumno), y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 24 de agosto de 2019.

El director:

A handwritten signature consisting of the letters "JJ" above a horizontal line, with three wavy lines underneath.

1 Introducción

1.1 Motivación

El número de robos y hurtos ha ido creciendo en estos últimos años, y por ello, es necesario la aplicación de medidas preventivas y correctoras que intenten reducir este número.

Hoy en día vivimos en una sociedad informatizada y tecnológica, donde cada uno tiene la capacidad de poder encontrar casi cualquier información allá donde esté, gracias a los dispositivos móviles y a las redes de comunicaciones que usa internet.

¿Quién no lleva un smartphone con conexión de datos en su bolsillo? En España el 85 % de la población hace uso de su smartphone a diario. Estos dispositivos móviles tienen una gran cantidad de usos y aplicaciones, y se han convertido en una herramienta imprescindible en nuestro día a día.

Uniendo tecnología y aplicación de medidas de seguridad basada en videovigilancia, surge la idea de este proyecto; la idea de construir un sistema de videovigilancia de bajo coste que sea fácil de usar, mantener, y lo más importante, que pueda ser controlado desde cualquier parte.

Partiendo de esta idea, se ha investigado y construido una aplicación que cumple con estas expectativas y más, ya que el uso que se le quiera dar puede ir más allá del concepto de videovigilancia para la seguridad. Por ejemplo, se puede utilizar para controlar entradas y salidas en una zona, control parental . . . , aunque eso sí, dentro de un uso permitido y responsable.

Otro de los principales aspectos, objetivos y motivación de este proyecto, es que su uso no sea de forma privada, es decir, que pueda ser usado por todas las personas que lo deseen. Por este motivo, se ha intentado reducir todo el coste posible del hardware, y se ha liberado todo el código fuente, junto con las instrucciones necesarias para la instalación y despliegue de la aplicación.

Toda la información relacionada se puede consultar en el repositorio de github [1].

1.2 Objetivos

El objetivo principal de este proyecto es el de poder construir un sistema de seguridad basado en la videovigilancia, de bajo coste, y accesible a todo el mundo.

Objetivos generales

- Conseguir rebajar el coste de los sistemas de videovigilancia habituales.
- Alertar a un usuario ante un evento de movimiento generado dentro de una zona controlada.
- Almacenar y obtener pruebas (fotos y/o vídeo) tras la detección de cualquier intruso.
- Monitorizar y controlar el estado de un entorno.
- Lograr aumentar el nivel de seguridad de cualquier tipo de entorno.

Objetivos específicos

- Desarrollar un sistema accesible para todo el mundo y fácil de usar.
- Permitir el acceso y gestión del sistema a través de un dispositivo móvil.
- Investigar y conocer el uso de los bots de telegram como medio de interacción entre el usuario y la aplicación back-end desarrollada en este proyecto.
- Estudiar el funcionamiento de una Raspberry PI y sus principales componentes para su aplicación en el ámbito del proyecto.
- Conocer, diseñar y hacer uso de una arquitectura basada en microservicios que permita el uso de servicios independientes, ágiles y escalables.
- Usar mecanismos de gestión de colas de mensajes y tareas asíncronas para evitar posibles esperas al interaccionar con la aplicación.
- Utilizar la inteligencia artificial para poder filtrar y evitar falsos positivos en las alertas generadas.

1.3 Estructura del documento

Este documento está estructurado de la siguiente forma:

- 1. Introducción:** Motivos y objetivos por los cuales se ha desarrollado este proyecto.
- 2. Planificación del proyecto:** Muestra información detallada acerca de las diferentes etapas del proyecto, junto con la planificación prevista.
- 3. Análisis de mercado:** Estudio comparativo acerca de software similar al propuesto en este proyecto.
- 4. Tecnologías y herramientas utilizadas:** Breve descripción sobre las tecnologías y herramientas utilizadas para desarrollar este proyecto.

2 Planificación

Como en todo proyecto, realizar una buena planificación es la clave para no fracasar en el desarrollo del proyecto. Por ello, se ha realizado una planificación detallada del proyecto, utilizando el concepto de *iteraciones* y *sprints* para ir distribuyendo el trabajo a lo largo de las semanas, durante las cuales se han ido desarrollando una serie de tareas que cumplen un objetivo específico.

Estos conceptos surgen de los llamados métodos de desarrollo ágiles, en los cuales se intenta descomponer una aplicación o proyecto por funcionalidades, y el objetivo es ir desarrollando y testeando cada una de forma independientes para poder integrarla con el resto.

La planificación que se ha previsto para el desarrollo del proyecto es la siguiente:

Número de semanas	11
Tiempo total estimado (h)	341h
Fecha de inicio del proyecto	01-07-2019
Fecha de fin del proyecto	15-09-2019

2.1 Fase preparatoria

En esta fase se comenzará el desarrollo del proyecto. Tiene un esfuerzo estimado de **36.5 horas**. Se realizará un estudio de viabilidad de la idea del proyecto, así como un estudio de mercado, una investigación acerca de las posibles herramientas a usar

El objetivo de esta primera fase es tener claro que el proyecto es viable tanto en recursos, tiempo y presupuesto, para que posteriormente comience su desarrollo e implementación.

Iteración 1: Análisis y estudio de la aplicación.

En esta iteración se describirán los principales objetivos de la aplicación, se estudiará sus posibles casos de uso y se elaborará un plan de iteraciones para planificar el desarrollo

del proyecto a lo largo del tiempo que se dispone.

Para esta iteración, se ha establecido la siguiente planificación:

Semana	1
Tiempo total previsto (h)	36.5h
Fecha de inicio	01-07-2019
Fecha de fin	06-07-2019

Esta iteración consta de los siguientes sprints:

Sprint 1: Descripción de la aplicación

Product backlog	Descripción	Semana	Tiempo (h)
Objetivos	Descripción de los objetivos	1	0.5
Funcionalidades	Descripción de las funcionalidades	1	1.5
Motivaciones	Motivaciones personales acerca de la temática del proyecto	1	0.5
Usuarios y escenarios	Descripción de los posibles casos de uso de la aplicación propuesta	1	2

Sprint 2: Investigación y preparación del entorno.

Product backlog	Descripción	Semana	Tiempo (h)
Estudio y análisis de mercado	Búsqueda de proyectos similares y estudio sobre sus componentes	1	5
Tecnologías y herramientas	Estudio de tecnologías y herramientas necesarias para el desarrollo del proyecto	1	5
Entorno de trabajo	Creación del repositorio de github y entorno de desarrollo del proyecto	1	4

Sprint 3: Planificación y plan de iteración.

Product backlog	Descripción	Semana	Tiempo (h)
Descomposición de tareas	Identificación de funcionalidades y descomposición en tareas	1	4
Plan de iteración	Elaboración de un plan de iteración para distribuir todas las tareas a lo largo del tiempo previsto para el desarrollo del proyecto	1	10
Documento de planificación	Documento donde se describe el plan de iteración y planificación	1	4

2.2 Fase de implementación y pruebas

Una vez se tienen claros los objetivos del proyecto, se ha visto que el proyecto es viable, y se han preparado las principales herramientas a utilizar, comienza la fase de implementación del proyecto.

El objetivo de esta fase es llevar a cabo la idea general del proyecto, es decir, implementar un sistema que sea capaz de realizar las acciones deseadas.

Esta fase de implementación se ha distribuido en varias iteraciones, con el fin de organizar y realizar una parte de la funcionalidad de la aplicación en cada iteración, donde se implementa, prueba y verifica el estado del software.

Al final de esta fase, se habrá obtenido un software implementado incrementalmente, donde se ha testeado cada módulo, tanto con pruebas unitarios como de integración y validación.

Para esta fase, se ha previsto un total de **221 horas**, y se ha distribuido en las siguientes iteraciones:

Iteración 2: Implementación de los módulos principales.

El objetivo de esta iteración es la implementación de los principales módulos de los que va a constar la aplicación. Estos módulos interaccionarán directamente con el hardware, y/o serán utilizados por los agentes y la API.

Para esta iteración, se ha establecido la siguiente planificación:

Semana	2
Tiempo total previsto (h)	45h
Fecha de inicio	07-07-2019
Fecha de fin	14-07-2019

Product backlog	Descripción	Semana	Tiempo (h)
Formación	Estudio de las bibliotecas de Python necesarias: <i>PiCamera, logging ...</i>	2	7
Módulo cámara	Implementación del módulo que va a conectar y utilizar la cámara de la raspberry.	2	7
Módulo vídeo	Implementación del módulo vídeo para conectar y utilizar la cámara de la raspberry.	2	8
Módulo logging	Implementación del módulo logging para almacenar todos los logs producidos por módulos, agentes y la API.	2	8
Test unitarios	Implementación de los test unitarios de los módulos desarrollados en esta iteración y depuración necesaria.	2	15

Iteración 3: Implementación de la API principal.

El objetivo de esta iteración es la implementación de la API que conectará los distintos módulos de la aplicación y responderá a todas las peticiones que reciba. Adicionalmente, se implementarán tareas asíncronas para evitar posibles esperas y poder atender al máximo número de peticiones simultáneamente.

Para esta iteración, se ha establecido la siguiente planificación:

Semana	3-4
Tiempo total previsto (h)	50h
Fecha de inicio	15-07-2019
Fecha de fin	24-07-2019

Product backlog	Descripción	Semana	Tiempo (h)
Formación	Estudio de las bibliotecas de Python necesarias: <i>flask</i> , <i>celery</i> , <i>requests</i> , servicio <i>rabbit mq</i> ...	3	8
API	Implementación de las funciones principales de la <i>API</i> .	3	12
Tareas asíncronas	Implementación de las tareas asíncronas de la <i>API</i>	3	6
Test unitarios	Implementación de los test unitarios para la <i>API</i> .	3	12
Test de integración	Implementación de los test de integración <i>API</i> .	4	12

Iteración 4: Implementación del motion agent y streaming server.

El objetivo de esta iteración es la implementación del módulo de streaming y el agente que sea capaz de detectar el movimiento para generar una alerta en consecuencia.

Para esta iteración, se ha establecido la siguiente planificación:

Semana	4-5
Tiempo total previsto (h)	36h
Fecha de inicio	25-07-2019
Fecha de fin	31-07-2019

Product backlog	Descripción	Semana	Tiempo (h)
Formación	Estudio de las bibliotecas de Python necesarias: <i>RPi.GPIO</i> ...	4	3
Módulo streaming	Investigación e implementación del módulo de streaming.	4	15
Agente detector de movimiento	Implementación del agente detector de movimiento.	4	10
Test unitarios	Implementación de los test unitarios del streaming server y motion agent.	5	8

Iteración 5: Implementación del agente de reconocimiento de objetos.

El objetivo de esta iteración es la implementación de un agente que sea capaz de identificar a una persona en una foto para poder evitar falsos positivos en las alertas generadas por el agente sensor de movimiento.

Para esta iteración, se ha establecido la siguiente planificación:

Semana	5-6
Tiempo total previsto (h)	44h
Fecha de inicio	01-08-2019
Fecha de fin	08-08-2019

Product backlog	Descripción	Semana	Tiempo (h)
Formación	Estudio de las bibliotecas necesarias: <i>Tensorflow, OpenCV ...</i>	5	8
Agente detector de objetos	Implementación del agente detector de objetos.	5	20
Test unitarios	Implementación de los test unitarios del detector de objetos.	6	6
Test de integración	Implementación de los test de integración del detector de objetos.	6	10

Iteración 6: Implementación del bot de telegram

El objetivo de esta iteración es la implementación de un bot de telegram como medio de interacción entre el usuario y la *API* desarrollada. Dicho bot servirá para poder controlar la aplicación desde cualquier parte y en cualquier dispositivo, gracias a la *API* y aplicación multiplataforma proporcionada por telegram.

Para esta iteración, se ha establecido la siguiente planificación:

Semana	6-7
Tiempo total previsto (h)	46h
Fecha de inicio	09-08-2019
Fecha de fin	18-08-2019

Product backlog	Descripción	Semana	Tiempo (h)
Formación	Estudio de las bibliotecas necesarias y API de telegram:	6	8
Funcionalidades del bot de telegram	Implementación del bot de telegram.	6	20
Integración del bot con la API	Integración de las funcionalidades del bot con la <i>API</i> desarrollada.	7	3
Interfaz de usuario	Desarrollo de una interfaz de botones e iconos en el bot de telegram.	7	15

2.3 Fase de documentación

Tras haber realizado y probado la implementación del proyecto, es hora de documentar todo el proceso realizado, cómo instalar y usar la aplicación ...

En este caso, se ha dividido la documentación en dos partes. La primera de ellas se corresponde con la documentación en el repositorio de github. Esta documentación tiene como objetivo mostrar para qué sirve, cómo poder instalarla y desplegarla y por último cómo usarla. Cualquier usuario que acceda al repositorio, puede ver toda esta información como en cualquier otro proyecto de software libre.

La otra documentación realizada es la del proyecto. En este caso, se realizará una documentación más enfocada al ámbito del proyecto: planificación, investigación, desarrollo, conclusiones ... (en lugar de la aplicación en concreto).

Para esta fase, se ha previsto un total de **83.5 horas**, y se ha distribuido en las siguientes iteraciones:

Iteración 7: Documentación de la aplicación en github

El objetivo de esta iteración es la documentación de la aplicación en el repositorio público de github. En dicha documentación se mostrará una descripción sobre la aplicación, una guía de instalación, guía de uso ...

Para esta iteración, se ha establecido la siguiente planificación:

Semana	8
Tiempo total previsto (h)	25h
Fecha de inicio	19-08-2019
Fecha de fin	25-08-2019

Product backlog	Descripción	Semana	Tiempo (h)
Readme	Documentación general de la aplicación.	8	6
API	Documentación de la API	8	6
Módulos	Documentación de los módulos.	8	5
Instalación y hardware	Documentación sobre la instalación y hardware necesario.	8	3
Guía de usuario	Documentación de una guía de usuario para poder utilizar la aplicación.	8	5

Iteración 8: Documentación del proyecto

El objetivo de esta iteración realizar la documentación del proyecto explicando objetivos, planificación, herramientas utilizadas, descripción de la aplicación ...

Para esta iteración, se ha establecido la siguiente planificación:

Semana	10-11
Tiempo total previsto (h)	58.5h
Fecha de inicio	01-09-2019
Fecha de fin	15-09-2019

Product backlog	Descripción	Semana	Tiempo (h)
Crear archivo LaTeX	Generación de plantilla, paquetes, resumen y portada.	10	2.5
Introducción	Motivaciones, objetivos ...	10	2
Planificación	Planificación del proyecto.	10	3
Ánalisis de mercado	Ánalisis de software similares al propuesto.	10	3
Tecnologías y herramientas usadas	Tecnologías y herramientas usadas en este proyecto.	10	5
Descripción de la aplicación	Descripción módulos y componentes de la aplicación.	10	20
Presupuesto	Presupuesto estimado del proyecto.	11	1
Guía de usuario	Guía de instalación y uso.	11	15
Conclusiones	Conclusiones generales del proyecto.	11	2
Diagramas de diseño	Graficar los diagramas de diseño que se han diseñado para este proyecto.	11	3
Otros	Diagrama de Gantt ...	11	2

3 Análisis de mercado

Antes de comenzar con el desarrollo del proyecto, se ha realizado un estudio de mercado para comprobar si hay algún tipo de sistema similar ya implementado, qué características tiene, y qué posibles diferencias habría con el que se tiene pensado implementar.

Es cierto que la raspberry PI tiene infinidades de posibles aplicaciones, y que en la web se puede encontrar todo tipo de tutoriales e ideas para poder explotar al máximo su uso, pero ¿existirá algún tipo sistema de seguridad que sea funcional, fácil de instalar y utilizar?. Para responder a esta pregunta, se ha realizado una búsqueda intensiva en la web y youtube, ya que suelen ser los principales medios donde se publica este tipo de información.

Investigando, se ha descubierto que hay múltiples páginas y vídeos que hacen referencia al uso de una raspberry PI con una cámara, y su posible aplicación para poder visualizar en tiempo real la grabación realizada por la cámara.

A continuación se van a detallar los principales aspectos a destacar de los sistemas encontrados.

3.1 Proyecto: Build a Raspberry Pi Security Camera Network

Este es un proyecto [2] que se basa en el uso de una distribución Linux llamada `motionEyeOS` y de código abierto [3] que convierte una raspberry PI en un sistema de videovigilancia.

Si leemos la descripción de este proyecto [2], podemos observar como se hace una pequeña descripción de los componentes necesarios y una guía de instalación del sistema `motionEyeOS`.

Entre sus principales características, podemos encontrar las siguientes:

- **Acceso restringido:** Consta de unas credenciales (usuario y contraseña) para poder acceder a la visualización y configuración de las cámaras.
- **Sistema multicámara:** Es un sistema que soporta la visualización e interacción

con múltiples cámaras simultáneamente conectadas en la misma red.



Figura 1: Sistema de seguridad con 3 cámaras

- **Conexión via wifi:** Este sistema soporta la conexión via wifi. Será necesario añadir el SSID y clave de la red.
- **Configuración de la cámara:** El sistema consta de un menú para poder configurar la cámara. Los parámetros configurables son:
 - Nombre de la cámara: Nombre para poder identificar la cámara.
 - Filtrado de luz: Filtro para evitar falsos positivos ante el encendido y apagado de una luz.
 - Brillo automático: Configuración automática de brillo y contraste de luz.
 - Resolución de la cámara: Configuración para poder modificar la resolución de la cámara.
 - Rotación de la cámara: Rotación de la imagen de vídeo o cámara.
 - FPS: Configuración del número de frames por segundo (imágenes por segundo).
- **Almacenamiento:** Es posible almacenar los archivos de imagen o vídeo.
- **Fecha y hora:** Posibilita la opción de mostrar fecha y hora durante la grabación de un vídeo o la captura de una foto.
- **Vídeo streaming:** Permite la visualización del vídeo capturada por las cámaras en tiempo real.
- **Detección de movimiento:** El sistema permite la detección de movimiento y en consecuencia la grabación de un vídeo o la captura de una foto.

- **Alertas:** Es posible generar alertas via email o webhook cuando se detecta movimiento.

3.2 Proyecto: Raspberry Pi As Low-cost HD Surveillance Camera

Este es un proyecto [4] bastante simple y centrado en la vigilancia y basado en **Motion detection software** [5]. Este es un software de código abierto y disponible en los repositorios de raspbian para poder monitorizar las señales de vídeo para varios tipos de cámaras (entre ellas la PiCamera).

Algunas de las características de este software son:

- Grabación de vídeos y/o capturas de fotos.
- Visualización de vídeo en tiempo real (streaming).
- Gestión de actividades y activación de scripts tras dichos eventos.
- Registro de eventos en bases de datos.
- Plantillas personalizables para detección de movimiento.
- Soporte completo de TLS (https) con autenticación y control web para streaming.

En este proyecto, se hace una descripción de los componentes necesarios para poder montar este sistema, e incluso se hace una estimación del precio de dichos componentes y se proporciona un enlace de compra.

A continuación se proporciona una guía básica de instalación y configuración, mostrando los siguientes resultados:



(a) Colocación de la cámara

(b) Streaming

Figura 2: Sistema de videovigilancia

3.3 Proyecto: sensor de movimiento y cámara con envío de imágenes a correo

Este es un proyecto descrito en un vídeo de youtube [6] que explica cómo construir un pequeño sistema de seguridad basado en la detección de movimiento, captura de una foto y envío de la foto capturada tras haberse activado la detección de movimiento.

En este vídeo se explica qué componentes son necesarios, cómo conectarlos y se proporciona un script de Python que se encarga de la detección movimiento y envío de la imagen al correo gmail, utilizando librerías de Python como *PiCamera* y *smtplib*.

Este proyecto no tiene complejidad alguna, ya que es muy simple y todo está explicado para cualquier usuario, y es un buen comienzo para empezar a investigar y probar el mundo de la videovigilancia basada en bajo coste.

3.4 Conclusión

Investigando por la web, he descubierto que hay muchos tipos de proyectos similares a los mencionados, unos más simples y otros más complejos, pero en general todos comparten el objetivo de poder vigilar una zona a través de una cámara y una raspberry PI.

Otro de los principales aspectos a destacar, es que la mayoría hacen el uso del software libre `motionEyeOS` o `motion agent detection` que se han mencionado en las secciones [3.1](#) y [3.2](#).

El objetivo de este proyecto no es 'volver a reinventar la rueda' sino hacer de este un proyecto innovador y con un valor añadido respecto al resto.

Por ello, se ha diseñado y propuesto un sistema que cumpla con la mayoría de características de proyectos similares, combinando lo mejor de cada uno, y añadiendo nuevas funcionalidades adicionales que hagan que sea una de las mejores opciones a la hora de plantearse implementar un sistema de videovigilancia de bajo coste.

Según he podido observar, hay muchos proyectos en los que la documentación es escasa y no muy concisa, además de realizar instalaciones y configuraciones a bajo nivel. Esto hace que cualquier usuario no tenga acceso a este tipo de aplicaciones, ya que requieren un usuario medio-avanzado en conocimientos informáticos.

Por ello, se ha tenido en cuenta la simplicidad del software y su fácil gestión y uso, además de intentar cubrir todos los principales aspectos de una videovigilancia.

En las siguientes secciones, se mostrará más en detalle qué funcionalidades tiene esta aplicación, pero para mostrar sus principales aspectos clave y diferenciadores respecto al resto se va a realizar una comparación, tal y como se puede observar en la siguiente tabla.

Observando la tabla [1](#), se puede comprobar como la aplicación propuesta cumple con la mayoría de las características del resto de proyectos actuales, y mejora en otros aspectos.

Como aspecto a destacar, decir que el proyecto propuesto consta de una aplicación backend y una aplicación multiplataforma de usuario, además de ser fácil de usar y estar documentada. Estos aspectos principales aportan una gran versatilidad a este proyecto, siendo sin duda la mejor opción hasta el momento desarrollada en el ámbito de la videovigilancia de raspberry PI.

También, como trabajos futuros, se desea implementar la funcionalidad multicámara, haciendo posible la interacción con más de una cámara simultáneamente.

Proyectos	Proyecto 1	Proyecto 2	Proyecto 3	Mi propuesta
Captura de imagen	✓	✓	✓	✓
Grabación de vídeo	✓	✓	✗	✓
Streaming	✓	✓	✗	✓
Alertas	✓	✓	✓	✓
Filtrado de alertas inteligente	✗	✗	✗	✓
Aplicación de usuario	✗	✗	✗	✓
Soporte multicámaras	✓	✓	✗	✗
Fácil instalación y uso	✗	✗	✓	✓
Cámara configurable	✓	✗	✗	✓
Documentación	✗	✗	✗	✓
Almacenamiento de eventos	✗	✓	✗	✓

Tabla 1: Comparativa de los distintos proyectos

4 Tecnologías y herramientas utilizadas

Para el desarrollo de este proyecto se han utilizado una gran cantidad de herramientas y tecnologías de código abierto. A continuación se muestra cada una de ellas.

4.1 Diseño

Para el diseño de este proyecto, se han realizado prototipos y diagramas para poder diseñar la arquitectura y principales componentes de la aplicación, además de la interacción de los componentes con el resto. Para realizar esto, se ha utilizado el siguiente software:

Draw.io

Draw.io [7] es una aplicación para diseño y diagramas de aplicaciones. Es un servicio gratis de Google y de código abierto.

Este software se ha utilizado para diagramas de diseño de interfaz.



Visual paradigm

Visual paradigm [8] es un software de modelado UML que nos permite analizar, diseñar, codificar, probar y desplegar. Dibuja todo tipo de diagramas UML, genera código fuente a partir de dichos diagramas y también posibilita la elaboración de documentos.



Este software se ha utilizado para la generación del diagrama de clases e interacción entre componentes.

4.2 Sistema control de versiones

Los sistemas de control de versiones [9] son programas que tienen como objetivo controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado a cualquiera de sus piezas, las personas que intervinieron en ellos ...

El control de versiones es una de las tareas fundamentales para la administración de un proyecto de desarrollo de software en general. Surge de la necesidad de mantener y llevar control del código que vamos programando, conservando sus distintos estados. Es absolutamente necesario para el trabajo en equipo, pero resulta útil incluso a desarrolladores independientes.

Por este motivo, no he dudado en utilizar un sistema de control de versiones. El sistema control de versiones me ha facilitado mucho la gestión y seguimiento del proyecto. El sistema que he utilizado ha sido **git**, y como medio de alojamiento para este proyecto he utilizado **github**.

Git

Git [10] es un sistema de control de versiones distribuido, de código libre y multiplataforma que permite la gestión de repositorios software a través de una línea de comandos [9].

Para este proyecto, se ha creado un repositorio donde se ha almacenado todo el código fuente de la aplicación junto con sus cambios a lo largo de su desarrollo.



Github

Github [11] es un servicio para alojamiento de repositorios de software gestionados por el sistema de control de versiones **Git**. En definitiva, **Github** es un sitio web pensado para

hacer posible el compartir el código de una manera más fácil y al mismo tiempo darle popularidad a la herramienta de control de versiones en sí, que es **Git**.

Este proyecto se ha ido alojando en el repositorio de github https://github.com/jmv74211/TFM_security_system_PI.



4.3 Lenguajes de programación y lenguajes de marcas

La implementación de este proyecto está desarrollada básicamente en **Python**. También se ha hecho uso de otros lenguajes de programación para alguna tarea en particular.

A continuación se muestran los lenguajes de programación utilizados.

Python

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general. Este lenguaje de programación viene instalado predeterminadamente en casi todos los sistemas operativos, y consta de un gran número de bibliotecas para realizar una gran cantidad de tareas.

Python se ha utilizado como lenguaje de programación base, con aproximadamente un 90 % del código fuente total. Básicamente he elegido este lenguaje ya que tiene Raspberry PI posee bibliotecas para interaccionar con Python, y con el resto de componentes involucrados.



HTML, CSS y Javascript

Para poder visualizar el streaming de vídeo a través del navegador, ha sido necesario implementar una pequeña página web utilizando **HTML** y **CSS**. Respecto a la parte de conexión entre la cámara y el navegador se ha utilizado **Python** para crear el servidor HTTP y la conexión websocket, y una biblioteca de funciones de **Javascript** para la transmisión de datos.



BASH

Bash es un intérprete de comandos que ejecuta, una por una, las instrucciones introducidas por el usuario o contenidas en un script y devuelve los resultados. En otras palabras, actúa como interfaz entre el kernel Linux y los usuarios o programas del modo texto. Además, incorpora numerosas utilidades de programación y mejoras sobre sh, su shell predecesora. Debido a que es una herramienta desarrollada por GNU, suele ser utilizada por defecto en las distros actuales.



En este proyecto se ha programado un script en **bash** para iniciar o parar la aplicación.

4.4 Bibliotecas y frameworks externos

Para la implementación de este proyecto, se han utilizado un gran número de bibliotecas para poder interaccionar entre sus distintos componentes.

A continuación se describen brevemente las bibliotecas y frameworks utilizados.

PiCamera

PiCamera [12] es una biblioteca de Python que proporciona una gran cantidad de funciones para interaccionar directamente con la cámara de la Raspberry PI.

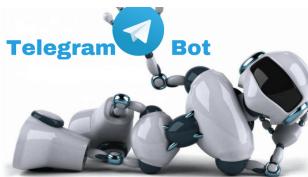
Esta biblioteca ha sido utilizada en los módulos que interactúan directamente con el recurso hardware de la cámara.



pyTelegramBotAPI

pyTelegramBotAPI [13] es una biblioteca de funciones de alto nivel que hace uso de la API de telegram.

En este proyecto se ha utilizado para crear el bot de telegram, que será la aplicación con la que el usuario interaccionará, y éste a su vez con la API desarrollada en la aplicación backend.



Flask

Flask [14] es un framework ligero de aplicaciones web WSGI. Está diseñado para hacer que el despliegue de una aplicación sea rápida y fácil, con la capacidad de escalar a aplicaciones complejas.

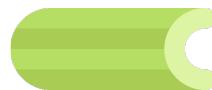
En este proyecto se ha utilizado para implementar varios servicios web, como la API principal que interactúa con el resto de módulos y componentes.



Celery

Celery [15] es una implementación de una cola de tareas para aplicaciones web de Python que se utiliza para ejecutar tareas de forma asíncrona.

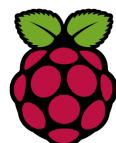
En este proyecto se ha utilizado principalmente en la API y en el agente de movimiento. El objetivo es poder realizar peticiones, y que la API atienda al mayor número de ellas, devolviendo la respuesta tras la ejecución de las tareas, sin producir ningún tipo de espera entre las comunicaciones (Más información en la sección X).



RPi.GPIO

RPi.GPIO [16] es un paquete que provee de una clase para controlar las conexiones GPIO de la Raspberry PI.

En este proyecto es usado para recibir la señal del sensor de movimiento, de forma que es posible saber cuándo se ha activado.



Tensorflow

Tensorflow es una plataforma de código abierto para aprendizaje automático. Cuenta con un ecosistema completo y flexible de herramientas, bibliotecas y recursos que permiten a los investigadores impulsar el estado del arte en machine learning y a los desarrolladores construir y desplegar fácilmente aplicaciones potenciadas por el machine learning.

En este proyecto se ha utilizado para construir un agente que sea capaz de reconocer objetos en una foto utilizando un modelo ligero preentrenado (más información en la sección X).



Requests

Requests [18] es una librería HTTP con licencia Apache2, escrita en Python. Permite enviar peticiones HTTP/1.1, además de añadir cabeceras, datos de formulario, archivos y parámetros con diccionarios Python simples, y acceder a los datos de respuesta de la misma manera.

En este proyecto ha sido utilizado para la comunicación entre los servicios a través de sus API's.



4.5 Documentación

LaTeX

Para elaborar la documentación se ha utilizado LaTeX [19] que es un sistema de elaboración de documentos para una composición tipográfica de alta calidad. Se utiliza con mayor frecuencia para documentos técnicos o científicos de mediano a gran tamaño, pero se puede utilizar para casi cualquier forma de publicación.

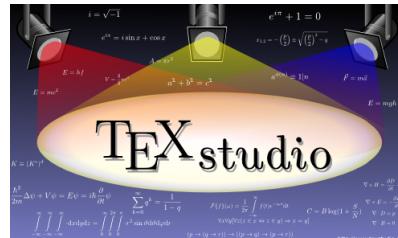
Para este proyecto se ha utilizado LaTeX, utilizando la distribución TeXLive [20] que proporciona un sistema TeX completo con binarios para la mayoría de las versiones de Unix, incluyendo GNU/Linux, macOS, y también Windows. Incluye todos los principales programas relacionados con TeX, paquetes de macros y fuentes que son software libre, incluyendo soporte para muchos idiomas en todo el mundo.



TeXstudio

TeXstudio [21] es un entorno integrado para crear documentos LaTeX. El objetivo es hacer que escribir LaTeX sea lo más fácil y cómodo posible.

Para este proyecto se ha utilizado la versión *2.12.14*.



4.6 Entorno de desarrollo

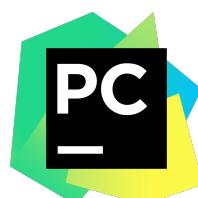
El entorno de desarrollo ha sido un aspecto muy importante para la implementación del proyecto, ya que el código se está ejecutando en una Raspberry PI que tiene recursos muy limitados, y los editores o entornos que puede utilizar no son los más cómodos.

Por este motivo, he estado investigando si se puede realizar un desarrollo de la aplicación vía remota. Leyendo foros, observé que el entorno de desarrollo **pycharm** tiene la opción de poder crear y desarrollar un proyecto vía SSH.

Pycharm

Pycharm [22] es un entorno de desarrollo profesional de **Python**. Gracias a este entorno de desarrollo he podido gestionar el proyecto de forma local, ya que los cambios generados se han sincronizado y ejecutado en la Raspberry PI a través de SSH.

En este proyecto se ha utilizado la versión *2019.1.3 Professional edition*, licencia académica que he obtenido gracias a la UGR.



4.7 Otros servicios

Rabbit MQ

RabbitMQ [23] es un broker de mensajes que acepta y reenvía mensajes. Un broker de mensajes actúa como plataforma intermediaria a la hora de procesar la comunicación entre dos aplicaciones.

En este proyecto ha sido utilizado para crear dos colas de procesos, utilizados por las tareas asíncronas desarrolladas con `celery`.



Telegram bot API

Los bots de Telegram [24] son aplicaciones de terceros que se ejecutan dentro de Telegram. Los usuarios pueden interactuar con los bots enviándoles mensajes, comandos y peticiones. Se pueden controlar estos bots usando peticiones HTTPS a la API de bot de Telegram.

En este proyecto se ha utilizado para crear un bot de telegram que interactúe con el usuario y con la aplicación backend desarrollada.



5 Descripción de la aplicación: SIVIRA

El sistema de videovigilancia con una Raspberry PI (**SIVIRA**) es una aplicación que nos permite construir un sistema de seguridad basado en la videovigilancia de bajo coste utilizando una Raspberry PI , una cámara y un sensor de movimiento.

La idea es poder controlar el sistema a través de una aplicación móvil, con la que el usuario pueda conectarse desde cualquier parte y tener acceso a su información y sistema de seguridad. Las funciones principales de este sistema son las siguientes:

- **Sistema automático** de alertas generadas al capturar el movimiento.
- **Sistema manual** para grabar vídeo o capturar una foto instantáneamente.
- **Sistema de streaming** para visualizar la imagen en tiempo real.

Además, podemos realizar configuraciones personalizadas a la cámara, como por ejemplo, cambiar la resolución, rotación... y la opción de poder activar un agente que filtre las alertas automáticas, generando únicamente alertas cuando se detecta a una persona en una imagen (evitar falsos positivos).

5.1 Arquitectura de la aplicación

Esta aplicación consta de una **arquitectura basada en microservicios**. Una arquitectura de microservicios [25] consta de una colección de servicios autónomos y pequeños. Los servicios son independientes entre sí y cada uno debe implementar una funcionalidad de negocio individual.

En cierto modo, los microservicios son la evolución natural de las arquitecturas orientadas a servicios aunque con ciertas diferencias.

¿Por qué se ha utilizado esta arquitectura?

En primer lugar, se ha realizado una descomposición de los principales componentes necesarios para construir la aplicación, y se ha observado que hay funcionalidades independientes que se pueden comunicar entre sí para integrarse en la aplicación. Esto aporta una gran serie de ventajas respecto a un diseño monolítico [26] como las siguientes:

- **Implementaciones independientes:** Es posible actualizar un servicio sin volver a implementar toda la aplicación y revertir o poner al día una actualización si algo va mal. Las correcciones de errores y las publicaciones de características son más fáciles de administrar y entrañan menos riesgo, por lo tanto, facilitan el mantenimiento de este software.
- **Desarrollo independiente:** Cada microservicio se ha ido desarrollando a lo largo de una iteración y de forma independiente al resto. Esto ha agilizado bastante la tarea, ya que las modificaciones y errores no se retropropagaban.
- **Fácil escalabilidad:** Cada microservicio puede ser escalado de forma independiente al resto. Por ejemplo, en el caso de que sea necesario escalar la detección de objetos en imágenes, bastaría con replicar el microservicio de detección de objetos y balancear las peticiones entre ellos.
- **Fácil integración y alta cohesión:** Un sistema de microservicios se puede integrar y adaptar a casi cualquier sistema. Por ejemplo, en el caso de que quisiera añadir un sistema multicámara a la aplicación, bastaría con añadir una nueva capa superior de abstracción sobre el software desarrollado y se integraría perfectamente.

Basándome en esta arquitectura basada en microservicios, se ha diseñado el siguiente sistema (ver figura 3).

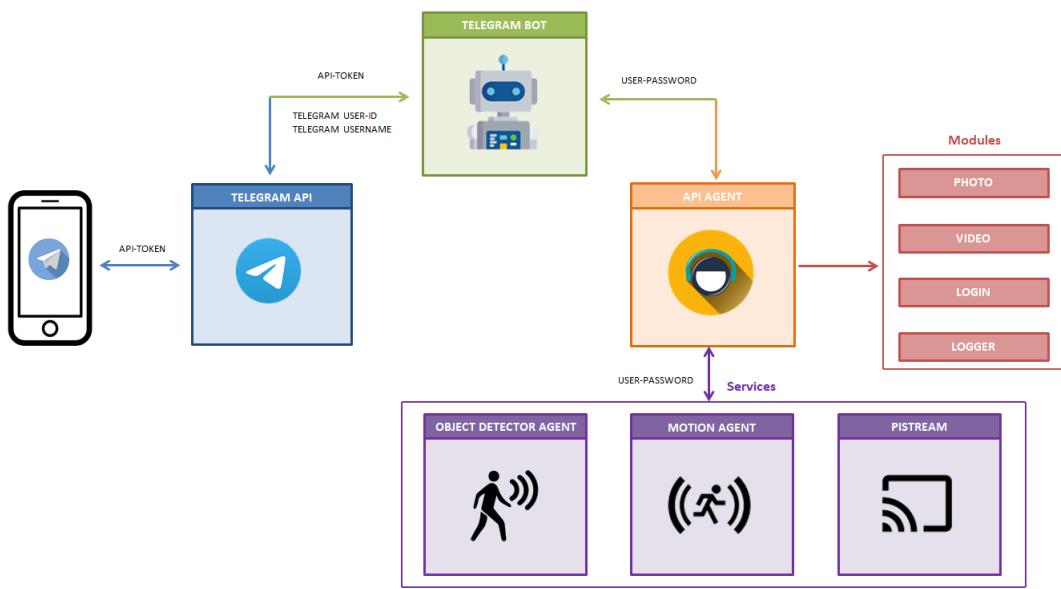


Figura 3: Arquitectura de la aplicación

En primer lugar tenemos los **módulos** de la aplicación (ver figura 4) que implementan un conjunto de funcionalidades para comunicarse con el módulo hardware de la cámara de la Raspberry PI, autenticación y logs.

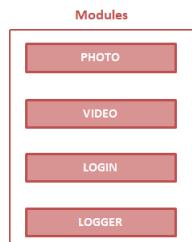


Figura 4: Módulos de la aplicación

A continuación tenemos el **API agent** (ver figura 5). Este es un servicio web que inicia la API que conecta con el resto de módulos de la aplicación.



Figura 5: API agent

Esta API importa y hace uso de los módulos de la aplicación.

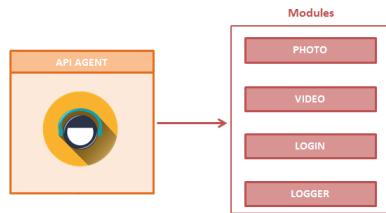


Figura 6: Conexión entre la API y los módulos

Por otra parte, tenemos un conjunto de servicios que se inician de forma independiente y proporcionan las funcionalidades de detección de objetos en imágenes, detección de movimiento y servicio de streaming.

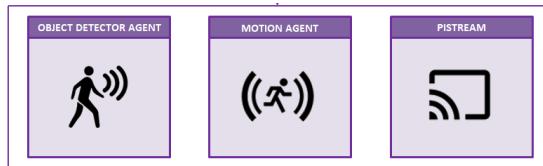


Figura 7: Servicios de la aplicación

La API está conectada con todo este conjunto de servicios y tiene la capacidad de poder gestionarlos.

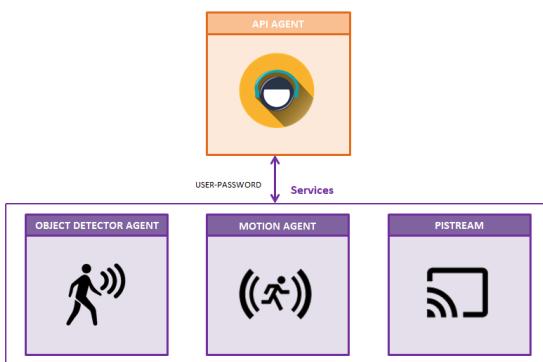


Figura 8: Conexión entre los servicios y la API

En la figura 9 podemos ver, como quedan unidos todos estos componentes entre sí.

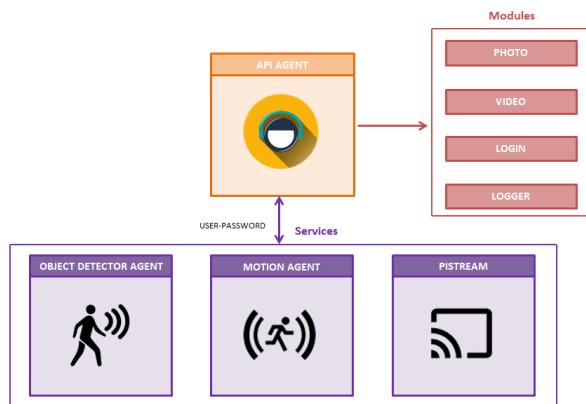


Figura 9: Conexión de módulos y servicios con la API

El siguiente componente es el **bot de Telegram**. Este bot es un proceso que se ejecuta junto a la API, cuyo objetivo es conectar la API de telegram con la API de la aplicación.

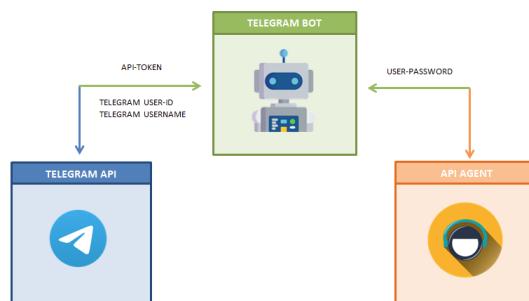


Figura 10: Conexión entre el bot de telegram y la API

Finalmente, el usuario puede interactuar con la aplicación (**SIVIRA**), haciendo uso de la aplicación multiplataforma **Telegram**, usando el bot que se ha creado.

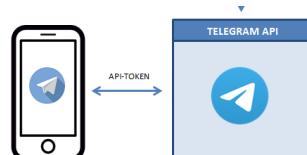


Figura 11: Conexión entre el usuario y la aplicación

5.2 Descripción de los componentes

La aplicación (SIVIRA) desarrollada en este proyecto está compuesta por los siguientes componentes:

5.2.1 Módulos

Los módulos son un conjunto de funciones cuyo objetivo es encapsular funcionalidades que después serán utilizadas por el resto de componentes de la aplicación (ver figura 4). A continuación se realiza una descripción de cada uno de ellos.

Logger

Módulo cuyo objetivo es poder capturar y almacenar los logs de la aplicación en ficheros independientes para monitorizar el estado de la aplicación. Su implementación parte de una clase abstracta base, cuyo objetivo es definir las propiedades y métodos que tendrán todas las clases que implementarán los logs del sistema.

PONER AQUÍ DIAGRAMA DE CLASES DE LOGS

Por defecto, los logs son mostrados y almacenados de la siguiente forma:

- **Consola:** Corresponde a la salida `stdout`. Todo logs por encima de un nivel determinado de prioridad será mostrado por la salida de consola.
- **Archivo de módulo:** Cada módulo y microservicio tendrán un fichero de log independiente para poder monitorizar su comportamiento a lo largo de su ejecución.
- **Archivo de la aplicación:** La aplicación consta de un archivo de logs donde se almacenan todos los logs de módulos y microservicios de forma conjunta.
- **Archivo de errores:** Todos los errores que surjan en cualquier módulo o microservicio es almacenado en este fichero para poder comprobar el estado de la aplicación.

Los niveles de logs que se han establecido son los siguientes:

- **DEBUG:** Nivel destinado a la depuración del código. Este nivel solo genera log en la salida por pantalla, y no se almacena en ningún fichero.
- **INFO:** Nivel destinado a mostrar un mensaje informativo sobre alguna acción llevada a cabo. Estos logs son almacenados en el fichero de módulo correspondiente, en el fichero de logs de la aplicación y mostrados por pantalla.
- **WARNING:** Nivel destinado a mostrar un mensajes de advertencia por posible mal uso de las llamadas a funciones, o simplemente porque algunas de ellas están 'deprecated'. Estos logs son almacenados en el fichero de módulo correspondiente, en el fichero de logs de la aplicación y mostrados por pantalla.
- **ERROR:** Nivel destinado a mostrar mensajes de error, que no son críticos para el sistema, y por lo tanto la aplicación sigue ejecutándose a pesar de producirse dichos error. Estos logs son almacenados en el fichero de módulo correspondiente, en el fichero de logs de la aplicación, en el fichero de errores de la aplicación.
- **CRITICAL:** Nivel destinado a mostrar mensajes de error críticos que provocan el fallo de la aplicación. Estos logs son almacenados en el fichero de módulo co-

rrespondiente, en el fichero de logs de la aplicación, en el fichero de errores de la aplicación.

El formato por defecto de los logs es el siguiente:

- **Formato del archivo de errores:** [Nivel:FechaHora:NombreArchivo:Función:Línea] Mensaje. Por ejemplo:

```
[ERROR:2019-08-15 13:10:11,597:logger.py:error:112x = set] Error while
trying to send an alert to Detector API agent with address
http://192.168.1.100:11000.¿It is running?
```

- **Formato del resto de logs:** [Nivel:FechaHora] Mensaje. Por ejemplo:

```
[INFO:2019-08-13 18:35:03,278] A 5 seconds video is being
```

Todos los ficheros de logs son almacenados en el directorio llamado `logs`. Dentro de dicho directorio, nos encontraremos los siguientes ficheros de logs:

- **API_agent.log:** Logs correspondientes a la API.
- **detector_object_agent.log:** Logs correspondientes al agente detector de objetos.
- **photo_module.log:** Logs correspondientes al módulo de fotos.
- **video_module.log:** Logs correspondientes al módulo de vídeo.
- **motion_agent.log:** Logs correspondientes al agente detector de movimiento.
- **security_system_PI_app_ERROR.log:** Logs correspondientes a los errores obtenidos por cualquier módulo o microservicio.
- **security_system_PI_app.log:** Logs correspondientes a los generados por módulos y microservicios a lo largo de su ejecución.
- **telegram_bot_agent.log:** Logs correspondientes al bot de telegram.

La implementación de este módulo puede comprobarse en este [enlace](#).

Photo

Módulo que implementa la clase **Photo** con la que se pretende administrar el recurso de la cámara de la Raspberry PI para capturar fotografías, además de establecer los diferentes parámetros de la cámara como resolución, rotación ...

Esta clase añade una capa de abstracción a la biblioteca PiCamera [12], para permitir conectarse con el recurso de la cámara de forma personalizada.

Las funcionalidades que aporta este módulo son las siguientes:

- Capturar una fotografía.
- Realizar una captura secuencial de fotografías en un intervalo de tiempo
- Establecer la configuración de la cámara: Rotación, resolución, giro horizontal o giro vertical.

La implementación de este módulo puede comprobarse en este [enlace](#).

Video

Módulo que implementa la clase **Video** con la que se pretende administrar el recurso de la cámara de la Raspberry PI para realizar grabaciones de vídeo, además de establecer los diferentes parámetros de la cámara como resolución, rotación ...

Esta clase añade una capa de abstracción a la biblioteca PiCamera [12], para permitir conectarse con el recurso de la cámara de vídeo de forma personalizada.

Las funcionalidades que aporta este módulo son las siguientes:

- Realizar una grabación de vídeo.
- Conversión de formato de vídeo .h264 a .mp4 (compatible con telegram).
- Establecer la configuración: Rotación, resolución, giro horizontal, giro vertical y mostrar hora y fecha durante la grabación de vídeo. .

La implementación de este módulo puede comprobarse en este [enlace](#).

Authentication

Módulo implementado para realizar una autenticación en el sistema. Esta autenticación es solicitada por la API, ya que junto a la petición deberá de ir las credenciales de acceso que se han definido en la configuración de la aplicación.

Este módulo básicamente implementa una función para poder comprobar si la autenticación es correcta. Su implementación puede comprobarse en este [enlace](#).

5.2.2 Servicios

Los servicios son procesos destinados a realizar alguna tarea en concreto y enviar una petición o respuesta con los resultados a la API. A continuación se realiza una descripción de cada uno de ellos.

Motion agent

Este servicio se encarga de controlar el sensor de movimiento y detectar cuando es activado para poder manejar dicho evento y enviar una alerta a la API. Las tareas realizadas por este servicio son las siguientes:

- Controlar el estado del sensor.
- Capturar fotos o vídeos en caso de detectar algún tipo de movimiento.
- Enviar una petición para procesar la imagen capturada, con objetivo de poder detectar si hay alguna persona en ella.
- Generar una alerta en la API en caso detectar a una persona en la foto.
- Mover la foto generada por la alerta al directorio de `false_positive` en caso de no detectar ninguna persona en la foto.

Para más información, en la sección X se detallará como funciona e interacciona este servicio con el resto de elementos.

La implementación de este servicio puede comprobarse en este [enlace](#).

Object detector agent

Este servicio se encarga de procesar una foto y devolver una lista de objetos detectadas en dicha foto. Este servicio es usado por el `motion agent`, ya que cuando detecta movimiento (si la funcionalidad de detección de personas está activada), envía una petición a este servicio, y se le responde con la lista de objetos.

Esta funcionalidad ha sido implementada gracias al uso de la biblioteca de aprendizaje automático `Tensorflow` [17], ya que se ha utilizado para cargar el modelo preentrenado `ssdlite_mobilenet_v2_coco` [27] para poder predecir una lista de objetos que aparecen en una imagen.

El motivo por el cual se ha seleccionado ese modelo, es porque es el modelo más ligero, y el único viable para este proyecto en la Raspberry PI (debido a sus bajos recursos hardware). Antes de escoger este modelo, se hicieron pruebas con otros un poco más pesados y los resultados eran abrumadores. Utilizando otros modelos, se han obtenido una media de espera de más de 100 segundos para procesar la imagen (obviamente inviable).

Utilizar el modelo `ssdlite_mobilenet_v2_coco` junto con una resolución de imagen media (1280x720) ha permitido obtener tiempos de procesamiento comprendidos entre unos 5 y 10 segundos, tiempo de espera que es aceptable.

El uso de este servicio es optativo, es decir, puede ser deshabilitado en las opciones o mediante la interfaz de usuario, y la aplicación puede funcionar normalmente. Es optativo por el hecho de su uso implica una serie de ventajas e inconvenientes que puede hacer pensar si realmente vale la pena utilizar este servicio o no.

Personalmente, yo recomendaría usar este servicio, ya que hay casos en los que se quiere controlar una zona y puede que haya demasiadas falsos positivos en las alertas por motivos como: animales domésticos, alta sensibilidad del sensor ...

A continuación se mencionan las posibles ventajas e inconvenientes de utilizar este servicio.

Ventajas

- Evita posibles falsos positivos en las alertas generadas.
- Puede aumentar el grado de eficacia del sistema de seguridad.

Desventajas

- Añade sobrecarga de procesamiento en la Raspberry PI, calentamiento
- Añade latencia (5 a 10 segundos) en el momento de generar alertas, es decir, aumenta el tiempo desde que se produce un evento hasta que se envía la alerta.
- Dado que el modelo de predicción es muy ligero, no es efectivo al 100% y puede cometer algunos fallos.
- Dificulta el proceso de instalación de la aplicación.

Por estos motivos, y porque la instalación de este agente puede resultar un poco tediosa (aunque todo está explicado en el repositorio del proyecto [1]) se ha decidido entre realizar dos tipos de instalaciones de la aplicación, una en la que se utiliza otro servicio para filtrar eventos y reducir el número de falsos positivos en las alertas, y la otra en la que se prescinde totalmente de este servicio, y se genera una alerta tras la detección de cualquier movimiento.

La implementación de este servicio puede comprobarse en este [enlace](#).

Referencias

- [1] Jonathan Martín Valera, TFM Security system PI github repository, disponible en
https://github.com/jmv74211/TFM_security_system_PI
- [2] Gus, 26 Jul 2019, Build a Raspberry Pi Security Camera Network, disponible en
<https://pimylifeup.com/raspberry-pi-security-camera/>
- [3] ccrisan, 3 Sep 2019, A Video Surveillance OS For Single-board Computers , disponible en <https://github.com/ccrisan/motioneyeos>
- [4] scavix, instructables.com, Raspberry Pi As Low-cost HD Surveillance Camera disponible en <https://www.instructables.com/id/Raspberry-Pi-as-low-cost-HD-surveillance-camera/>
- [5] Motion-project, 19 Ago 2019, Camera video monitor, disponible en <https://github.com/Motion-Project/motion>
- [6] Humberto Higinio, 14 Oct 2018, Sistema de Seguridad Raspberry Pi-Sensor de movimiento y cámara HD con envío de imágenes a correo, disponible en <https://www.youtube.com/watch?v=rK6uLwMRtIs>
- [7] Google, DrawIO, diseño y diagramas de aplicaciones open source, disponible en <https://www.draw.io/>
- [8] Visual paradigm, diseño de diagramas UML, disponible en <https://www.visual-paradigm.com/>
- [9] Israel Alcázar, 03 Jun de 2014, Introducción a Git y Github, disponible en <https://desarrolloweb.com/articulos/introduccion-git-github.html>
- [10] Git, Sistema de control de versiones distribuido, multiplataforma y de código abierto, disponible en <https://git-scm.com/>
- [11] Github, Servicio de alojamiento para el desarrollo de software utilizando Git , disponible en <https://git-scm.com/>
- [12] PiCamera documentación, Interfaz de cámara Raspberry PI para python.

- [13] pyTelegramBotAPI, Bibliotecas de funciones para la conexión con la API de telegram, disponible en <https://github.com/eternnoir/pyTelegramBotAPI>.
- [14] Flask, framework para el despliegue ágil de aplicaciones sencillas, disponible en <https://flask.palletsprojects.com/en/1.1.x/>.
- [15] Celery, Cola de tareas distribuidas, disponible en <http://www.celeryproject.org/>.
- [16] RPi.GPIO, Biblioteca para controlar los pines GPIO de la Raspberry PI, disponible en <https://pypi.org/project/RPi.GPIO/>.
- [17] Tensorflow, An end-to-end open source machine learning platform, disponible en <https://www.tensorflow.org/>.
- [18] Requests, An end-to-end open source machine learning platform, disponible en <https://www.tensorflow.org/>.
- [19] Latex-project, An introduction to LaTeX, disponible en <https://www.latex-project.org/about/>.
- [20] TeX Live, 06 Jul de 2019, Introduction to TeX Live, disponible en <https://www.tug.org/texlive/>.
- [21] TeXstudio.org, Welvome to TeXstudio, disponible en <https://www.texstudio.org/>.
- [22] Jetbrains, Pycharm, The Python IDE for Professional Developers, disponible en <https://www.jetbrains.com/pycharm/>.
- [23] RabbitMQ, Most widely deployed open source message broker, disponible en <https://www.rabbitmq.com/>.
- [24] Telegram bot, Bots: An introduction for developers, disponible en <https://core.telegram.org/bots>.
- [25] Olprod, 13 Nov 2018, Estilo de arquitectura de microservicios, disponible en <https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices>.

- [26] Chakray, 18 Jul 2018, DevOps: Arquitectura monolítica vs Microservicios, disponible en <https://www.chakray.com/es/devops-arquitectura-monolitica-vs-microservicios/>
- [27] Tensorflow models ,Tensorflow detection model zoo, disponible en https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md