Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

APLICACIÓN MÓVIL DE VIGILANCIA DE UN AMBIENTE INTELIGENTE MEDIANTE SERVICIOS

Alumno: Juan Antonio Pérez Cruz

Tutor: Dra. Dª. Macarena Espinilla Estévez
Dr. D. Pedro José Sánchez Sánchez
Dpto: Informática

Septiembre, 2016

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Doña Macarena Espinilla Estévez, tutora del Proyecto Fin de Carrera titulado: Aplicación móvil de vigilancia de un ambiente inteligente mediante servicios, que presenta Juan Antonio Pérez Cruz, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2016

El alumno: Los tutores:

Juan Antonio Pérez Cruz Macarena Espinilla Estévez

Pedro José Sánchez Sánchez

Índice

1. Introducción ... 8
1.1. Introducción al trabajo ... 8
1.2. Motivación ... 8
1.3. Objetivos ... 9
1.4. Resumen del documento ... 10
2. Planificación ... 11
2.1. Introducción ... 11
2.2. Especificación de tareas ... 11
2.3. Cronograma Gantt ... 13
2.4. Presupuesto ... 14
3. Sistemas de vigilancia ... 15
3.1. Introducción a los sistemas de vigilancia ... 15
3.2. Ejemplos de sistemas de vigilancia en el hogar ... 17
3.3. Dispositivos de bajo coste ... 22
3.3.1. Arduino ... 22
3.3.2. Raspberry Pi ... 24
4. Análisis del sistema ... 28
4.1. Estudio y justificación de los componentes elegidos ... 28
4.1.1. Placa ... 28
4.1.2. Cámara ... 28
4.1.3. Sensor de movimiento ... 29
4.1.4. Tarjeta SD ... 30
4.1.5. Alimentación ... 31
4.1.6. Módulo Wifi (opcional) ... 31
4.1.7. Carcasa (opcional) ... 31
4.1.8. Cables ... 31
4.1.9. Resumen del pedido ... 31
4.2. Análisis del sistema ... 32
4.2.1. Análisis de requisitos ... 32
4.2.2. Perfiles de usuario ... 33
4.2.3. Casos de uso ... 33
4.3. Especificación de requisitos funcionales y no funcionales ... 39
4.3.1. Requisitos funcionales ... 39
4.3.2. Requisitos no funcionales ... 42

5. Diseño del sistema 44
5.1. Introducción 44
5.2. Servicios Web 44
5.3. Estructura de los datos 47
5.4. Raspberry Pi 51
5.5. Diseño de la interfaz 53
5.5.1. Guía de estilo y metáforas 53
5.5.2. Storyboard 59
5.6. Arquitectura del software 64
6. Implementación y pruebas del sistema 65
6.1. Introducción 65
6.2. Desarrollo del sistema de vigilancia 66
6.2.1. Raspberry Pi 66
6.2.2. Servidor 66
6.2.3. Aplicación Android 73
6.2.4. Herramientas de desarrollo 75
6.3. Casos de prueba 76
7. Conclusiones y futuras mejoras 84
Anexo 1: Manual de instalación 87
A.1. Raspberry Pi 87
A.2. Servidor 93
Anexo 2: Manual de usuario 95
Bibliografía 110

Índice de ilustraciones

Ilustración 1: Diagrama de Gantt 13
Ilustración 2: Sistema de vigilancia Myfox 17
Ilustración 3: Sistema de vigilancia Piper 19
Ilustración 4: Sistema de vigilancia Foscam 21
Ilustración 5: IDE Arduino 22
Ilustración 6: Pack Arduino 23
Ilustración 7: Arduino YUN 24
Ilustración 8: Raspberry Pi 27
Ilustración 9: Sensor de puerta 30
Ilustración 10: Sensor PIR 30
Ilustración 11: Diagrama frontera 34
Ilustración 12: Componentes modelo Entidad-Relación 48
Ilustración 13: Modelo Entidad-Relación 49
Ilustración 14: Tablas de la base de datos 49
Ilustración 15: Componentes del sistema de la cámara 51
Ilustración 16: Sistema de la cámara montado 52
Ilustración 17: Sistema operativo Raspbian 52
Ilustración 18: Paleta de colores usada en la aplicación Android 55
Ilustración 19: Flat Button en la aplicación Android 55
Ilustración 20: Floating Action Button en la aplicación Android 55
Ilustración 21: Toast en la aplicación Android 56
Ilustración 22: Menú en la aplicación Android 56
Ilustración 23: Alert en la aplicación Android 56
Ilustración 24: Toolbar en la aplicación Android 56
Ilustración 25: Card en la aplicación Android 57
Ilustración 26: Floating label en la aplicación Android 57
Ilustración 27: User input error en la aplicación Android 57
Ilustración 28: Text field en la aplicación Android 57
Ilustración 29: Notificación Big Style en la aplicación Android 58
Ilustración 30: Navigation Drawer en la aplicación Android 59
Ilustración 31: Swipe to refresh en la aplicación Android 59
Ilustración 32: Iniciar sesión y nuevo usuario del Storyboard 60
Ilustración 33: Fotos y foto del Storyboard 61
Ilustración 34: Menú de foto del Storyboard 61

Ilustración 35: Cerrar sesión del Storyboard 62
Ilustración 36: Mis cámaras del Storyboard 62
Ilustración 37: Mi cuenta y Acerca de del Storyboard 63
Ilustración 38: Diagrama de funcionamiento del sistema 65
Ilustración 39: Diagrama de funcionamiento de Google Cloud Messaging 74

Índice de ilustraciones del manual de instalación

Ilustración A1 1: Conexión sensor PIR y cámara 87
Ilustración A1 2: Instalación de Raspbian 88
Ilustración A1 3: Conexión de Wifi, SD y alimentación del sistema de la cámara 88
Ilustración A1 4: Obtención de la MAC 91
Ilustración A1 5: Configuración del Wifi en Raspbian 92

Índice de ilustraciones del manual de usuario

Ilustración A2 1: inicio de sesión 95
Ilustración A2 2: Nuevo usuario 96
Ilustración A2 3: Inicio de sesión con datos 96
Ilustración A2 4: Sin fotos para mostrar 97
Ilustración A2 5: Menú principal 98
Ilustración A2 6: Sin cámaras para mostrar 99
Ilustración A2 7: Nueva cámara 99
Ilustración A2 8: Cámara registrada correctamente 100
Ilustración A2 9: Borrar cámara 101
Ilustración A2 10: Notificación 102
Ilustración A2 11 Mis fotos con fotos 103
Ilustración A2 12: Una foto seleccionada 103
Ilustración A2 13: Menú foto seleccionada 104
Ilustración A2 14: Permiso para guardar foto 105
Ilustración A2 15: Eliminar foto 106
Ilustración A2 16: Modificar datos de la cuenta 107
Ilustración A2 17: Eliminar cuenta 107
Ilustración A2 18: Acerca de 108
Ilustración A2 19: Cerrar sesión 109

Índice de tablas

Tabla 1: Comparativa Arduino YUN y Raspberry PI 2 28
Tabla 2: Comparativa cámara Waveshare y Camera pi.....29
Tabla 3: Resumen de los componentes pedidos 31
Tabla 4: Comparativa SOAP y REST 46
Tabla 5: Comparativa acciones de HTTP con otros sistemas.....47
Tabla 6: Herramientas de desarrollo empleadas 75

1. Introducción

1.1. Introducción al trabajo

Este proyecto abarca multitud de campos de desarrollo e implementación de ingeniería informática. Se requieren conocimientos de asignaturas del Grado tales como Arquitectura de Computadores, Programación Orientada a Objetos, Estructuras de Datos, Bases de Datos, Interacción Persona Ordenador, Desarrollo de Aplicaciones Móviles, Diseño e Implementación de Servidores, Redes e Infraestructuras de Comunicaciones, entre otras.

La idea del proyecto de cara al usuario es, sin embargo, bastante simple. Se trata de poder vigilar un lugar deseado de forma remota. Para ello se utiliza obligatoriamente una conexión a Internet aceptable y de calidad (con un caudal mínimo de 1 o 2MB) que permita interactuar entre un dispositivo móvil, un servidor con las fotos y una cámara capaz de hacer dichas fotos al detectar movimiento. Todo ello, empleando componentes de bajo coste.

1.2. Motivación

La simple idea de poder vigilar un lugar deseado de forma remota y a bajo coste es hoy en día una realidad gracias al auge de dispositivos con potencia y rendimiento más que suficientes para contribuir al 'Internet de las Cosas'. Un sensor de movimiento y una cámara (también con precio contenido) conectados a este tipo de dispositivos que controlan y procesan la información de estos actuadores nos vale para poder empezar.

¿Y qué es el Internet de las Cosas? Más dispositivos conectados a la red que personas. Dispositivos cotidianos como lámparas, termostatos, cámaras, etc. En palabras de la reconocida compañía de comunicaciones CISCO: "*El internet de las cosas es la próxima evolución de Internet que lo cambia todo" […] "Ya están en marcha proyectos que prometen cerrar la brecha entre ricos y pobres"* [1]. Hoy día se puede ver que esto es ya posible, y con este Trabajo fin de Grado se pretende contribuir como un proyecto más a este movimiento revolucionario.

Por otro lado, el uso de los teléfonos inteligentes o 'smartphones' es hoy día una realidad totalmente estandarizada. Incluso en personas de edad más avanzada. Esto

permite entre otras cosas que los usuarios puedan interactuar con multitud de aplicaciones y además de una forma sencilla y hoy día muy natural. De hecho, el porcentaje de usuarios con smartphones se incremente cada año. En nuestro país, un 80% de la población española cuenta con algún tipo de smartphone. Ditrendia en su informe Mobile en España y el mundo, elevaba la cifra a mediados de 2015 hasta el 87% [2].

Así que, con la posibilidad de desarrollar una aplicación móvil que llegue prácticamente a cualquier usuario para que pueda manejar la información que llega desde una cámara en un lugar remoto, el poder implementar el sistema de dicha cámara, y además poder almacenar toda esa información en una base de datos en un servidor, ¿por qué no ponerse 'manos a la obra'? Todo ello sin perder la perspectiva de usar un sistema con un coste bajo.

1.3. Objetivos

Los objetivos de este Trabajo Fin de Grado son los siguientes:

- Elegir e implementar un sistema compuesto de una cámara un sensor de movimiento y una placa de procesamiento programable de bajo coste que sirva como actuador para la vigilancia.
- Desarrollar e implementar un servidor Web con una base de datos que garantice la persistencia de la información requerida del cliente.
- Emplear servicios Web para la interacción entre la cámara-servidor-cliente.
- Desarrollar un prototipo de aplicación móvil para poder visualizar las fotos tomadas por la cámara, con notificaciones push y con acceso restringido a los datos por usuario.

1.4. Resumen del documento

En el primer apartado se explica brevemente una introducción al proyecto a realizar.

En el segundo, se planifica todo el trabajo a realizar.

El tercer capítulo expone los sistemas de vigilancia de forma detallada.

El cuarto capítulo analiza el sistema.

El quinto, explica el diseño del sistema.

En el sexto capítulo tiene la implementación y pruebas del sistema.

En el séptimo, se exponen las conclusiones finales y las mejoras que se pueden realizar en el futuro.

Por último, tienen lugar los anexos con el manual de instalación y de usuario y la bibliografía.

2. Planificación

2.1. Introducción

En este apartado se detallará la planificación del proyecto para proporcionar un marco de trabajo y con ello estimar los tiempos de cada una de las fases que lo compone.

Para ello, se especifican las tareas y el cronograma Gantt a lo largo de una línea temporal acotada.

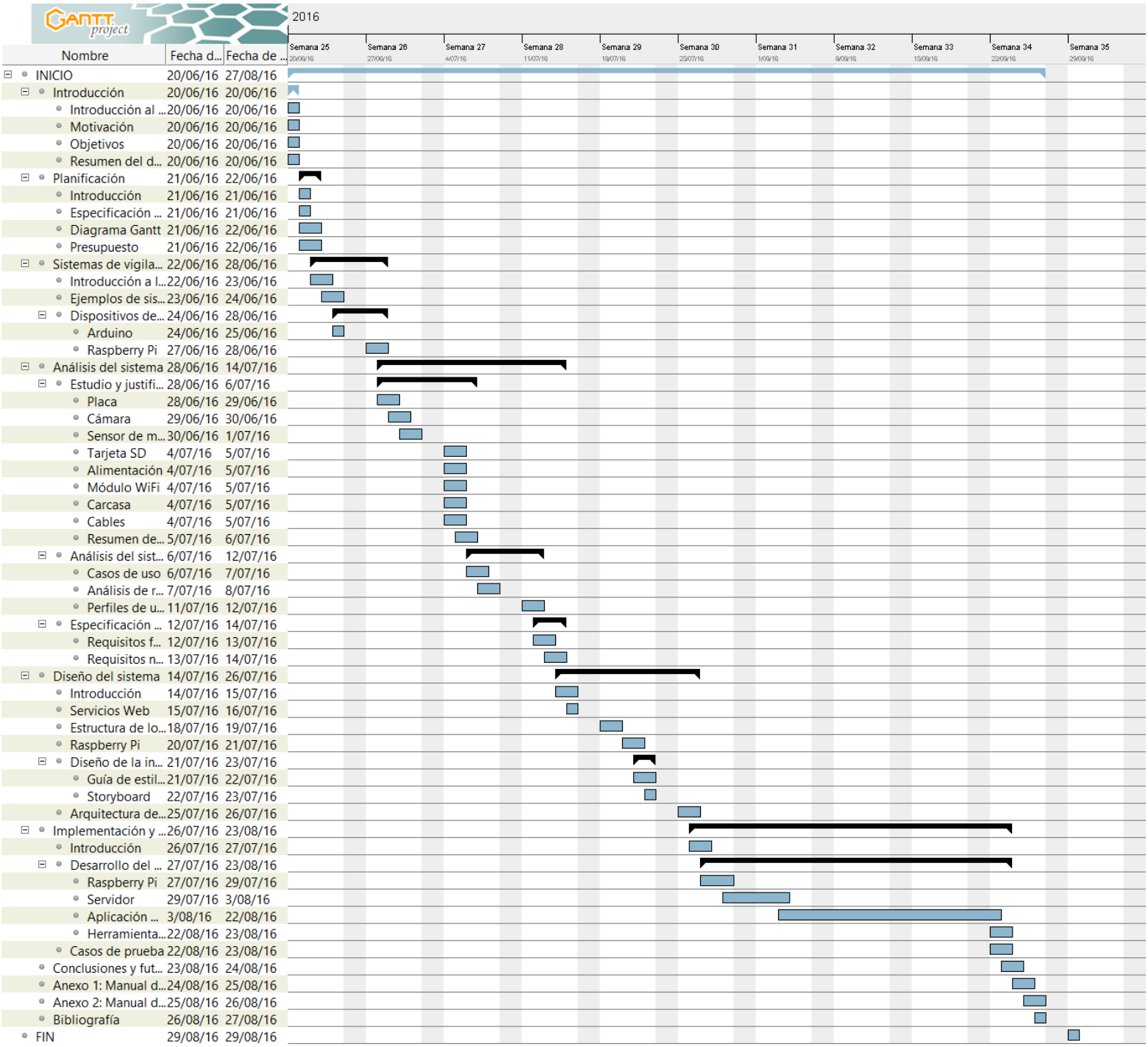
2.2. Especificación de tareas

Las tareas a llevar a cabo para desarrollar el proyecto son:

- **Introducción:** aquí se explica brevemente de qué se trata el proyecto, la motivación para realizarlo, los objetivos marcados para lograrlo y un resumen de este documento.
- **Planificación:** se establece un plazo de tiempo para cada una de las tareas a realizar en el proyecto mediante un diagrama de Gantt y un presupuesto para adquirir los componentes.
- **Sistemas de vigilancia:** una breve introducción a los sistemas de vigilancia, ejemplos reales de los que se pueden encontrar en el hogar y los principales dispositivos de bajo coste que puedan valer para este proyecto.
- **Análisis del sistema:** se hace un estudio y justificación de los componentes que se van a emplear y un detallado análisis del sistema: análisis de requisitos, perfiles de usuario, casos de uso y especificación de requisitos funcionales y no funcionales.
- **Diseño del sistema:** se expone aquí una breve introducción de lo que se refiere al diseño del sistema, una explicación de los servicios web y cuál se escoge y por qué, la estructura de los datos (la base de datos) escogida para garantizar la persistencia de los datos, la placa de bajo coste elegida para controlar la cámara, el desarrollo detallado del diseño de la interfaz y la arquitectura software empleada.

- **Implementación y pruebas del sistema:** una breve introducción a este apartado, el desarrollo del sistema de vigilancia, explicando las tecnologías utilizadas para la cámara, el servidor y la aplicación móvil y los casos de prueba para garantizar el correcto funcionamiento de la aplicación.
- **Conclusiones y futuras mejoras:** aquí se hace un breve análisis personal de lo que ha conllevado la realización del proyecto y posibles puntos donde se puede mejorar el sistema de cara al futuro.

2.3. Cronograma Gantt



Nombre	Fecha d...	Fecha de ...
INICIO	20/06/16	27/08/16
Introducción	20/06/16	20/06/16
Introducción al ...	20/06/16	20/06/16
Motivación	20/06/16	20/06/16
Objetivos	20/06/16	20/06/16
Resumen del d...	20/06/16	20/06/16
Planificación	21/06/16	22/06/16
Introducción	21/06/16	21/06/16
Especificación ...	21/06/16	21/06/16
Diagrama Gantt	21/06/16	22/06/16
Presupuesto	21/06/16	22/06/16
Sistemas de vigila...	22/06/16	28/06/16
Introducción a l...	22/06/16	23/06/16
Ejemplos de sis...	23/06/16	24/06/16
Dispositivos de...	24/06/16	28/06/16
Arduino	24/06/16	25/06/16
Raspberry Pi	27/06/16	28/06/16
Análisis del sistema	28/06/16	14/07/16
Estudio y justifi...	28/06/16	6/07/16
Placa	28/06/16	29/06/16
Cámara	29/06/16	30/06/16
Sensor de m...	30/06/16	1/07/16
Tarjeta SD	4/07/16	5/07/16
Alimentación	4/07/16	5/07/16
Módulo WiFi	4/07/16	5/07/16
Carcasa	4/07/16	5/07/16
Cables	4/07/16	5/07/16
Resumen de...	5/07/16	6/07/16
Análisis del sist...	6/07/16	12/07/16
Casos de uso	6/07/16	7/07/16
Análisis de r...	7/07/16	8/07/16
Perfiles de u...	11/07/16	12/07/16
Especificación ...	12/07/16	14/07/16
Requisitos f...	12/07/16	13/07/16
Requisitos n...	13/07/16	14/07/16
Diseño del sistema	14/07/16	26/07/16
Introducción	14/07/16	15/07/16
Servicios Web	15/07/16	16/07/16
Estructura de lo...	18/07/16	19/07/16
Raspberry Pi	20/07/16	21/07/16
Diseño de la in...	21/07/16	23/07/16
Guía de estil...	21/07/16	22/07/16
Storyboard	22/07/16	23/07/16
Arquitectura de...	25/07/16	26/07/16
Implementación y ...	26/07/16	23/08/16
Introducción	26/07/16	27/07/16
Desarrollo del ...	27/07/16	23/08/16
Raspberry Pi	27/07/16	29/07/16
Servidor	29/07/16	3/08/16
Aplicación ...	3/08/16	22/08/16
Herramienta...	22/08/16	23/08/16
Casos de prueba	22/08/16	23/08/16
Conclusiones y fut...	23/08/16	24/08/16
Anexo 1: Manual d...	24/08/16	25/08/16
Anexo 2: Manual d...	25/08/16	26/08/16
Bibliografía	26/08/16	27/08/16
FIN	29/08/16	29/08/16

Ilustración 1: Diagrama de Gantt

2.4. Presupuesto

Con un presupuesto de menos de 100€ se eligen adecuadamente los componentes que formen el sistema de vigilancia.

La estimación de esta cantidad se ha determinado como se verá más adelante en base a los precios de estos componentes. Una de las premisas de este proyecto es que el sistema de vigilancia sea de bajo coste y la idea de que el sistema a realizar por menos de esa cantidad sea tal se determina comparando con otros sistemas del mercado actual y teniendo en cuenta que el sistema debe llevar una placa o mini ordenador que permita programar el comportamiento de la cámara y del sensor de movimiento que la active.

3. Sistemas de vigilancia

3.1. Introducción a los sistemas de vigilancia

Cuando hablamos de sistemas de vigilancia, nos referimos hoy día a vigilancia IP. "*El Video IP, también conocido como Video Over IP (del inglés) es uno de los grandes resultados que nos trajo la era digital con la globalización de la información; y es el término que se ha utilizado para nombrar la Tecnología que sorprendió al mundo al capturar, comprimir y convertir las secuencias de imágenes en movimiento (video) en un flujo de datos que puede ser trasmitido por redes de computadoras, también conocidas como Redes IP (LAN / WLAN / WAN / Internet)"*. [3]

El transporte de vídeo por redes de datos ha sido posible gracias a tres puntos principales:

- Los avances en las técnicas de digitalización y compresión de imágenes.
- El crecimiento en las redes de datos (IP Networks)
- El desarrollo y comercialización cada vez mayor de equipos de vídeo digitales que han sido de interés mundial (Satélites, Televisión digital por cables, DVD, etc.)

La industria ha desarrollado hardware y software suficientemente potentes y eficientes como para realizar todas las funciones de los sistemas analógicos tradicionales, y superarlos ampliamente con la incorporación de funciones "inteligentes" que eran solo un sueño hace pocos años. Aquí nos referimos ya a los sistemas de vigilancia IP.

Los sistemas de vigilancia IP son aquellos en que imágenes y/o audio son capturados por las cámaras y micrófonos, se comprimen y se transmiten por una red de datos Local o Internet (LAN/WAN) y pueden ser accesibles desde uno o varios puntos en cualquier lugar del mundo mediante computadores convencionales, capaces de descomprimir esos datos, visualizarlos, analizarlos, grabarlos y hasta de generar acciones de manera automática en respuesta a diferentes eventos pre-definidos o a voluntad de un operador.

Entre los elementos que pueden componer un sistema de vigilancia IP encontramos:

- Las cámaras IP
- Servidores de vídeo
- Decodificadores de vídeo
- Grabadoras Digitales de Red
- Software Inteligente para centrales de monitoreo

En la realización de este proyecto, y como se detallará más adelante, se ha optado por el uso de una cámara conectada a una Raspberry PI. Esto, junto con el script y el software configurado, actúa como una cámara IP.

¿Y en qué sectores se puede usar la Vigilancia IP? Puede utilizarse en infinidad de situaciones, desde aplicaciones sencillas residenciales hasta en sistemas profesionales de magnitud gubernamental o multinacional. Entre los campos a utilizar se encuentran:

- Educación
- Transporte
- Bancos
- Gobierno
- Comercios minoristas/Plazas y Centros Comerciales
- Industrial
- Casinos/Entretenimiento
- Hogar

En nuestro caso nos centraremos en el uso de la vigilancia en el hogar, dado la complejidad requerida y el uso que se le dará al sistema de este TFG.

3.2. Ejemplos de sistemas de vigilancia en el hogar

A continuación, se detallarán tres sistemas de vigilancia basados en cámaras IP que se pueden adquirir y usar hoy en día para nuestro hogar:

- Myfox

  Se trata de una cámara de vigilancia con tapa que cerramos a demanda. Esto sirve para cuando se está en la casa asegurar la privacidad cerrando dicha tapa. Graba vídeo en calidad fullHD, posee visión nocturna y detector de movimiento para enviar alarmas al smartphone.

  Desde su página web se puede observar cómo ofrecen distintos módulos además de la cámara. Como mínimo es necesaria dicha cámara y su precio es de 199€.

  Parece un servicio de calidad y se ofrece una aplicación para Android e iOS para poder manejar la cámara. [4]

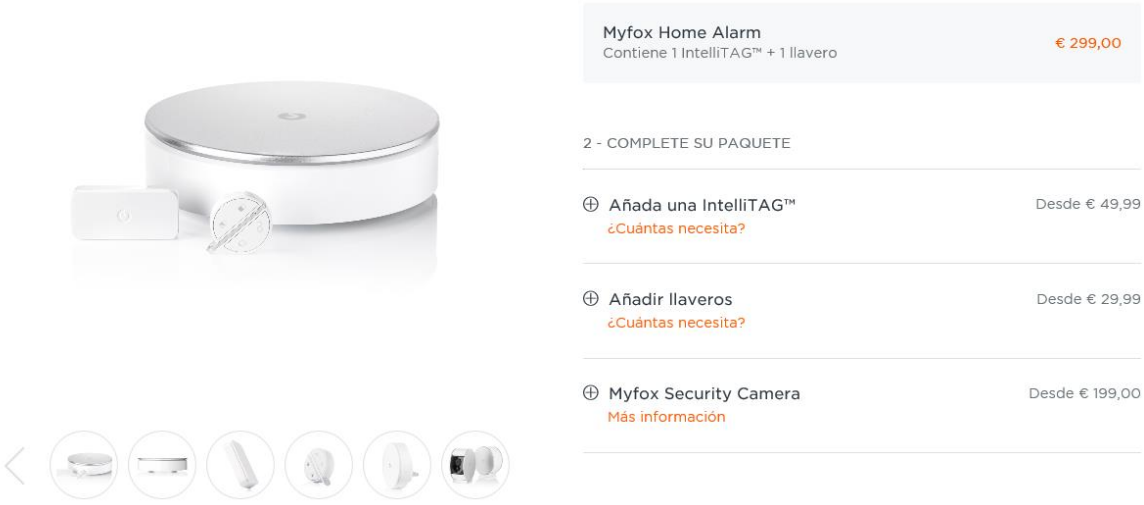


Ilustración 2: Sistema de vigilancia Myfox

- Piper

  Es un sistema de vigilancia que aúna un control domótico compatible con el estándar Z-Wave. Es una idea muy interesante dado que permite conectar dispositivos extras que sean compatibles con Z-Wave como podrían ser interruptores inteligentes, sistemas de control de apertura y cierre de puertas, enchufes inteligentes, etc.

  La cámara posee un ángulo de visión de 180º, con micrófono y sirena incorporados. Posee sensores de temperatura, humedad, movimiento, luz y sonido, así como conexión inalámbrica. El precio: sobre 225€.

  Como se puede ver en su página web, se ofrece también una aplicación para acceso remoto en las principales plataformas móviles [5]:

Ilustración 3: Sistema de vigilancia Piper

- Foscam

  Este quizás sea el que más interese para nuestro propósito. Se trata de un sistema que ofrece un servicio más simple que los dos anteriores y posee un precio más atractivo.

  En la página web principal de la empresa se pueden adquirir distintos tipos de cámaras IP a buen precio y por medio de una aplicación (al igual que los dos anteriores) éstas se manejan. El precio medio para poder adquirirlo sería en torno a unos 80€.

  Éste servicio seria competencia directa del sistema desarrollado en este TFG, puesto que el precio que se ofrece y el precio es acorde a lo que se puede determinar como un sistema de vigilancia ‘low cost’.

  Aquí una captura de su página web [6]:

Ilustración 4: Sistema de vigilancia Foscam

3.3. Dispositivos de bajo coste

Dado que uno de los objetivos de este proyecto consiste en diseñar e implementar un sistema de vigilancia, el uso de cámaras IP no nos valen. Éstas cámaras ya vienen con su lógica programada y preparada para su uso. Incluso ya te ofrecen una aplicación de forma gratuita.

Llegados a este punto, es el momento de explicar qué placas programables o microcontroladores hoy día existen para poder montar todo el sistema y con un precio bajo.

3.3.1. Arduino

Arduino es una plataforma de electrónica 'open-source' o de código abierto cuyos principios son contar con un software y hardware fáciles de usar. La idea es que cualquiera con mínimos conocimientos de programación se capaz de realizar proyectos interactivos.

Centrándonos en el software, éste incluye un IDE (entorno de desarrollo) para casi todas las plataformas (Windows, Linux, Mac). La idea es poder escribir nuestras aplicaciones, descargarlas al Arduino y ejecutarlas. El entorno de desarrollo es gratuito y se programa en Wiring (un framework para microcontroladores), muy parecido a c++.

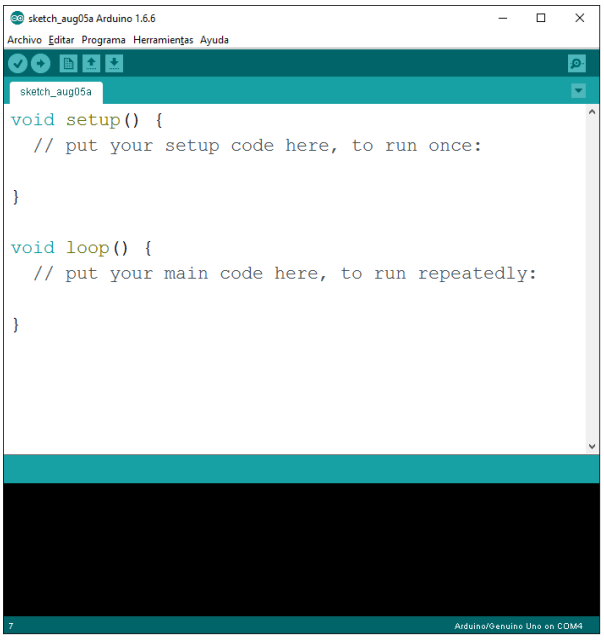


Ilustración 5: IDE Arduino

En cuanto al hardware, hay muchas placas basadas en Arduino. Al ser 'Open-source', cualquiera que quiera hacer una placa puede hacerlo. El hardware más sencillo consiste en una placa con un microcontrolador y una serie de puertas de entrada y salida. Los microcontroladores AVR más usados son el Atmega168, Atmega1280 y Atmega8 por su sencillez y bajo conste que permiten el desarrollo de múltiples proyectos. Todos ellos incluyen conexiones para añadir y programar sensores muy baratos e interesantes como detectores de humo, sensores de humedad, de temperatura, de ultrasonidos, pantallas, etc. Y todos estos sensores son compatibles con muchas placas del marcado. Por tanto, se puede decir que son universales. Aquí una imagen con algunos de estos sensores que se venden en packs con los microcontroladores Arduino:

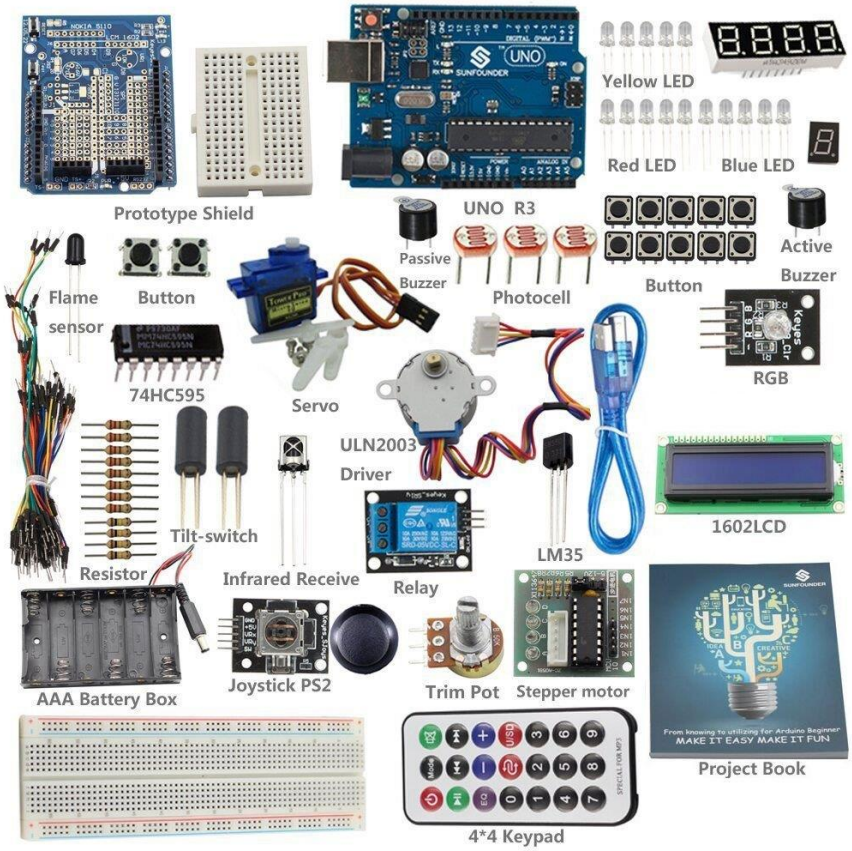


Ilustración 6: Pack Arduino

Cabe destacar que el microcontrolador que suelen incluir estos packs es el Arduino UNO R3, uno de los más básicos. El precio de estos packs suele rondar los 60€. Adquirir el microcontrolador aparte son unos 20€.

https://www.amazon.es/SunFounder-Starter-Learning-Arduino-Principiante/dp/B00E59L71S/ref=sr_1_1?ie=UTF8&qid=1470346953&sr=8-1&keywords=arduino

Quizás la placa que más interesa para el sistema que se va a desarrollar sea el Arduino YUN, dado que incluye el módulo Wifi, un puerto USB, lector de tarjetas con sistema operativo Linux y su microcontrolador es el ATmega32U4. Por el puerto USB se podría conectar una cámara para poder montar el sistema de vigilancia [7]:

Ilustración 7: Arduino YUN

El precio de esta placa ronda los 75€.

3.3.2. Raspberry Pi

Raspberry Pi es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Aunque no se indica expresamente si es hardware libre o con derechos de marca, en su sección de preguntas y respuestas frecuentes (FAQs) explican que disponen de contratos de distribución y venta con dos empresas, pero al mismo tiempo

cualquiera puede convertirse en revendedor o redistribuidor de las tarjetas Raspberry Pi: ver punto 7 (en inglés) de https://www.raspberrypi.org/help/faqs/#buyingWhere, por lo que se entiende que es un producto con propiedad registrada, pero de uso libre. De esa forma mantienen el control de la plataforma, pero permitiendo su uso libre tanto a nivel educativo como particular.

Tampoco deja claro si es posible utilizarlo a nivel empresarial u obtener beneficios con su uso, asunto que se debe consultar con la fundación.

En cambio, el software sí es open source, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian.

El modelo más actual es Raspberry Pi 2 model B, lanzado en 2015. Dicho modelo cuenta con un SoC Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB). La CPU es quad-core a 900MHz, con arquitectura ARM Cortex A7. La GPU admite resoluciones y reproducción de hasta 1080p y la memoria RAM (SDRAM) es de 1GB compartidos con la GPU. Son características más que decentes para un mini ordenador propiamente dicho (ya que posee su sistema operativo) que pueda valernos para el proyecto. Además, sus conexiones son:

- **Micro USB**: Es el sistema de alimentación de la Raspberry, con un cargador de móvil micro USB (5V) común podemos darle corriente. Es recomendable que sea de al menos 2A para un funcionamiento estable con periféricos como teclados, ratones, discos duros o adaptadores Wifi.
- **GPIO**: Estos puertos son una de las cosas que diferencia a la Raspberry de un PC clásico. Mediante estas entradas y salidas de propósito general podremos hacer que la Raspberry interactúe con actuadores abriendo y cerrando contactos, encendiendo LEDs, conociendo el estado de un interruptor, etc. Son un total de 40 puertos de los cuales 26 se pueden usar como entradas/salidas. Estos puertos sirven de la misma forma que en Arduino. De hecho, se pueden usar los mismos sensores que para Arduino dado que son compatibles. Con la ayuda del IDE correspondiente podremos programar el comportamiento de cada sensor acoplado a un puerto determinado.

- **USB**: La versión 2 B cuenta con 4 puertos USB para lo que lo necesitemos. Lo más común es ocupar 1 o 2 para teclado y ratón y otro para un adaptador Wifi. Los otros pueden servir para conectar memorias externas como discos duros, pendrives, o bien un módulo Bluetooth entre otras cosas.
- **Micro SD**: Situada en la parte trasera, nos vale una tarjeta micro SD para poder almacenar el sistema operativo. Es recomendable que sea una tarjeta de lectura/escritura rápida y con un espacio recomendado de 4GB. Una buena opción es elegir una de clase 10.
- **HDMI**: A través de un cable HDMI podemos conectar la Raspberry a la TV u otro monitor para poder uso de su sistema operativo con interfaz gráfica o bien si lo preferimos por medio de consola.
- **Audio 3,5mm**: Por si queremos conectar unos cascos, altavoces u otro tipo de dispositivo de audio.
- **Ethernet**: Puerto más que recomendado para utilizar la conexión a Internet. Más seguro y fiable que la conexión Wifi, aunque nos obliga a depender de la longitud del cable hasta el rúter.
- **Display DSI**: Existen pequeñas pantallas táctiles con conector de este tipo que podemos acoplar a la Raspberry y hacernos una pseudo-tablet.
- **Camera CSI**: un puerto al que le podemos conectar una cámara y en la que podremos programar su comportamiento gracias a las librerías en la distribución oficial Raspbian de su sistema operativo.

En el lado del software se puede decir que cuenta con un montón de distribuciones para poder instalarlas en la tarjeta SD, entre los cuales están: Raspbian (oficial), Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IoT, Osmc, Libreelec, Pinet, RISC OS, etc. En el caso de elegir un sistema basado en Linux, el IDE para poder ejecutar scripts o programas para el manejo de sus puertos GPIO será Python, aunque también se podrían realizar programas en c++.

Lo más destacable además es su precio: alrededor de 40€. Se puede ver que, por ese dinero, una tarjeta SD para el sistema operativo y un cargador de móvil podemos tener un verdadero ordenador con distribuciones basadas en Linux entre otras. Es decir, un sinfín de servicios muy útiles como Apache, MySQL o Tomcat y

con la posibilidad de interactuar con sensores de coste muy bajo a través de sus puertos GPIO [8].

Aquí se puede ver en una imagen cómo es la placa:

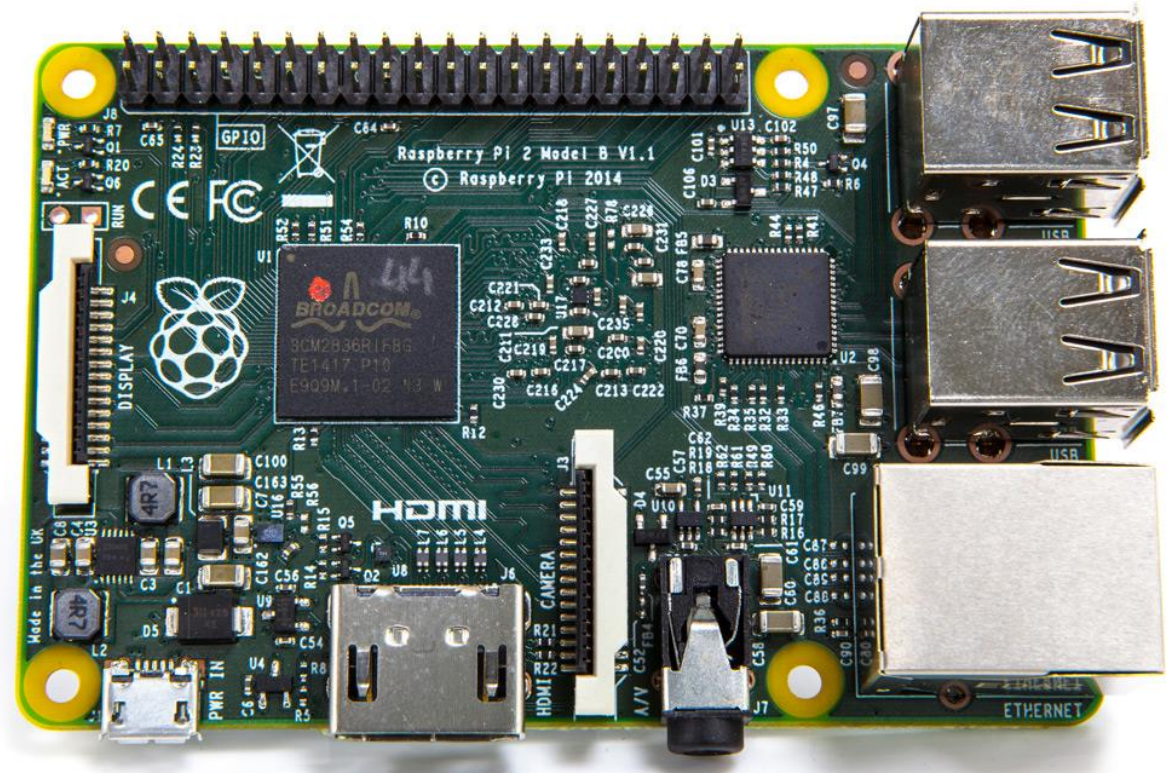


Ilustración 8: Raspberry Pi

4. Análisis del sistema

4.1. Estudio y justificación de los componentes elegidos

4.1.1. Placa

Lo primero de todo, se van a comparar y elegir entre las 2 principales placas que se ajustan a los requerimientos de este proyecto.

Placa	Arduino YUN	Raspberry Pi 2 modelo B
Ethernet	Sí	Sí
WiFi integrado	Sí	No
Sistema Operativo	Sí	Sí
Puertos para sensores	Sí	Sí
Módulo para cámara	No	Sí
Potencia suficiente	Sí	Sí
Precio	~75€	~40€
Enlace (www.amazon.es)	https://goo.gl/xsBjW4	https://goo.gl/YsAEVq

Tabla 1: Comparativa Arduino YUN y Raspberry PI 2

Nota: Los precios detallados en la tabla varían a razón de días de acuerdo a la política de precios de las tiendas y las empresas mencionadas.

Se puede concluir que la elección adecuada es la Raspberry Pi. El precio es prioridad en este proyecto. El módulo Wifi es opcional, pero se necesita aparte la alimentación de la placa, una tarjeta SD, cables de conexión, un sensor de movimiento para que se active la cámara y la cámara.

4.1.2. Cámara

Una vez elegida la placa, el siguiente paso es elegir la cámara. Hay 3 modelos principales que funcionan con las librerías 'PiCamera'. Estas librerías se utilizan en el IDE de Python para poder controlar la cámara. Cabe destacar que para poder usar estas librerías lo recomendable es usar la última versión disponible de Raspbian (el sistema operativo de la Raspberry). En una pequeña tabla comparativa se puede ver las características de las 2. Interesa que sean infrarrojos, para que el sistema de vigilancia funcione bien de noche.

Cámara	Waveshare	Pi camera
Calidad	5MP, vídeo 1080p	5MP, vídeo 1080p
Infrarrojos	Sí (LEDS incluidos)	Sí (sin LEDS)
Alimentación requerida junto con la Raspberry	3A	2A
Precio	~25€	~30€
Enlace (www.amazon.es)	https://goo.gl/XsNDZl	https://goo.gl/F7vK20

Tabla 2: Comparativa cámara Waveshare y Camera pi

Nota: Los precios detallados en la tabla varían a razón de días de acuerdo a la política de precios de las tiendas y las empresas mencionadas.

Nos decantamos por la cámara de la marca Waveshare, ya que nos interesa ajustar el precio y que se vean las imágenes en la oscuridad. Muy a tener en cuenta que la alimentación que se necesita para poder utilizar la cámara con la Raspberry es de 3A. Esto no se especifica de ninguna forma en la página oficial del producto, pero leyendo comentarios en la propia página de compra (Amazon) aseguran que debido a los 2 LEDS de iluminación infrarroja que se incluyen con la cámara, la potencia necesaria asciende de 2 a 3A. En cualquier caso, los LEDS se pueden desmontar de la cámara o dejar sólo uno para que el consumo sea más contenido.

4.1.3. Sensor de movimiento

Hay múltiples sensores que se pueden usar en este apartado. Uno de los primeros a tener en cuenta es un sensor de puerta. El mecanismo es sencillo, un juego de imanes activa un sensor cada vez que uno de los imanes que lo componen se separa. Este sensor se conecta a un puerto GPIO de la Raspberry y se identificaría cuando se ha abierto o cerrado la puerta. Es un sensor que ronda los 5€. El problema de este sensor, es que limita el uso del sistema de vigilancia de forma que sólo se puede usar en puertas. Una imagen de cómo es este sensor:

Ilustración 9: Sensor de puerta

El más apropiado sería un sensor PIR. Este tipo de sensor es barato (entre 3 y 10€) y funciona realmente bien en Raspberry. Se trata de un sensor de infrarrojos pasivo. Si el sensor detecta por medio de señales infrarrojas que el entorno ha cambiado a una distancia de entre 1 y 7 metros, el voltaje que emite en la placa pasa a ser de alto (3.3V) a bajo (0V). Se puede regular la distancia a la que se requiere que funcione y la sensibilidad a la hora de actuar a través de dos resistencias que incorpora. En la imagen de la izquierda se puede ver qué conexiones en los puertos GPIO de la Raspberry se debe usar el sensor.

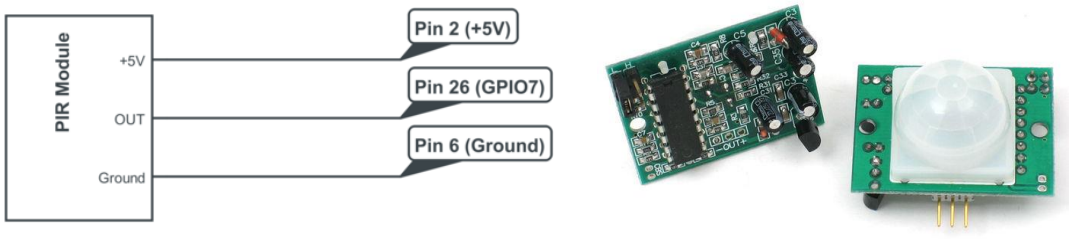


Ilustración 10: Sensor PIR

4.1.4. Tarjeta SD

Para la tarjeta SD nos vale cualquiera que tenga un mínimo de 4GB de capacidad y sea de clase 10. De la capacidad se puede deducir que el sistema operativo (en nuestro caso Raspbian) ocupa en torno a 1GB, por tanto, debemos dejar algo de margen para poder instalar algún servicio, librerías, etc. La razón por la que debe ser clase 10 es debido a la velocidad de lectura/escritura que asciende hasta los 10MB/s. Esto hará que el sistema vaya más rápido y para poder capturar vídeo e imágenes en HD. El precio de una tarjeta de estas características ronda los 5 o 6€ dependiendo de la capacidad [9].

4.1.5. Alimentación

Como ya se ha mencionado anteriormente, con un cargador de tablet o móvil de 2A se podría utilizar el sistema (el cable de alimentación debe ser micro USB). El problema es que debido a los LEDs de la cámara para la visión nocturna se requieren 3A. El precio de este tipo de cargador ronda los 12 o 13€.

4.1.6. Módulo Wifi (opcional)

Este módulo no es estrictamente necesario gracias a la entrada Ethernet de Raspberry, pero se ha optado por incorporar uno. Esto es gracias a su bajo coste de unos 7€ y a que permite poder ubicar el sistema de vigilancia en casi cualquier parte del hogar. Hoy en día prácticamente en todos los hogares existe un rúter con conexión Wifi, por lo que es una buena forma de aprovecharlo.

4.1.7. Carcasa (opcional)

Para que la placa esté cubierta frente a polvo y otros elementos, es mejor incorporarla a una tapa a medida. El valor de este componente es de unos 4€

4.1.8. Cables

Para poder establecer comunicación entre el sensor PIR y la Raspberry es necesario el uso de unos pequeños conectores. Su valor es de aproximadamente 1€.

4.1.9. Resumen del pedido

En esta tabla se puede ver detalladamente el pedido con el precio detallado:

Componente	Precio	Enlace
Raspberry PI 2 B	40€	https://goo.gl/YsAEVq (Amazon)
Cámara WaveShare	25€	https://goo.gl/XsNDZl (Amazon)
Sensor de mov. PIR	2,19€	http://goo.gl/aJLMTu (eBay)
Tarjeta SD clase 10 16GB	4,63€	https://goo.gl/BaFrqA (Amazon)
Cargador 3A	9,95€	https://goo.gl/SJyEzH (Amazon)
WiFi USB	8,67€	https://goo.gl/aGPXuS (Amazon)
Caja Rasp.Pi 2 B	3,45€	https://goo.gl/cRs8ps (Amazon)
Cables jumper H-H	1,70€	http://goo.gl/L7rjq3 (eBay)

Tabla 3: Resumen de los componentes pedidos

Precio total (gastos de envío de Amazon incluidos): 98,60€

Nota: Los precios detallados en la tabla varían a razón de días de acuerdo a la política de precios de las tiendas y las empresas mencionadas. También se debe mencionar que el uso del adaptador Wifi o la carcasa son opcionales dependiendo del lugar donde se vaya a ubicar el sistema de vigilancia.

4.2. Análisis del sistema

4.2.1. Análisis de requisitos

"Lo más difícil en la construcción de un sistema software es decidir precisamente qué construir."

"No existe tarea con mayor capacidad de lesionar al sistema, cuando se hace mal."

"Ninguna otra tarea es tan difícil de rectificar a posteriori."

[10]

"El 45% de los errores tienen su origen en los requisitos y en el diseño preliminar."

[11]

"El 56% de los errores que tienen lugar en un proyecto software se deben a una mala Especificación de Requisitos"

[12]

Lo cierto es que en esta fase del proyecto es importante definir qué tenemos que hacer. La mayoría de los errores se encuentran en esta fase. Si no se hace un buen análisis de lo que el usuario quiere (en este caso el tutor de este TFG) y cómo lo va a llevar el programador, es frecuente que se acumulen fallos imprevistos.

Para conseguir un buen estudio de estos requisitos se hace uso de los perfiles de usuario, casos de uso y una explicación detallada de los requisitos funcionales y no funcionales del proyecto.

4.2.2. Perfiles de usuario

En este apartado se diferencian quiénes van a ser los usuarios que van a utilizar la aplicación Android y la cámara de vigilancia y qué requisitos deben cumplir para saber manejar dicho sistema. En nuestro caso hay un único perfil: el de usuario convencional. Los requisitos de éste para poder utilizar la vigilancia son:

- Habilidad con el manejo de una aplicación Android. Teniendo en cuenta que la aplicación es sencilla y posee una interfaz cuidada y acorde con los estándares marcados en Android no supone mayor problema de cara a un usuario móvil. Sin embargo, utiliza funcionalidades como iniciar y cerrar sesión, notificaciones o bien eliminar una foto.
- Habilidad con el manejo de Linux. Dado que el identificador de cada cámara será único y se ha optado por la MAC del dispositivo Ethernet de cada Raspberry, se necesita que el usuario por medio de un comando en la terminal (en el sistema operativo Raspbian) conozca dicho identificador. Esto se realiza para poder añadir una cámara de vigilancia a un usuario. También se debe saber instalar el script correspondiente y configurar el acceso a Wifi (se realiza cómodamente desde la interfaz gráfica).

4.2.3. Casos de uso

Los diagramas de casos de uso describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario. Desde esta perspectiva es capaz de marcar unos límites al sistema y poseer una perspectiva más clara de su funcionamiento. Se debe expresar en lenguaje natural.

Los casos de uso describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él. El actor representa cualquier agente que intercambia información con el sistema (persona o máquina) y el caso de uso es una secuencia de transacciones que se encuentran relacionadas con su comportamiento. Se debe documentar:

- Una lista enumerada de los pasos que sigue el actor para interactuar con el sistema.
- Una lista de alternativas que aparecen durante la ejecución del caso de uso.

Entre las relaciones que unen al actor con el caso de uso se encuentran las de tipo:

- Extensión (extends): ocurre cuando un caso de uso extiende a otro (pasa a formar parte de otro).
- Inclusión (include): ocurre cuando un caso de uso incluye a otro (incorpora el comportamiento de otro).

Aquí se muestra el diagrama frontera:

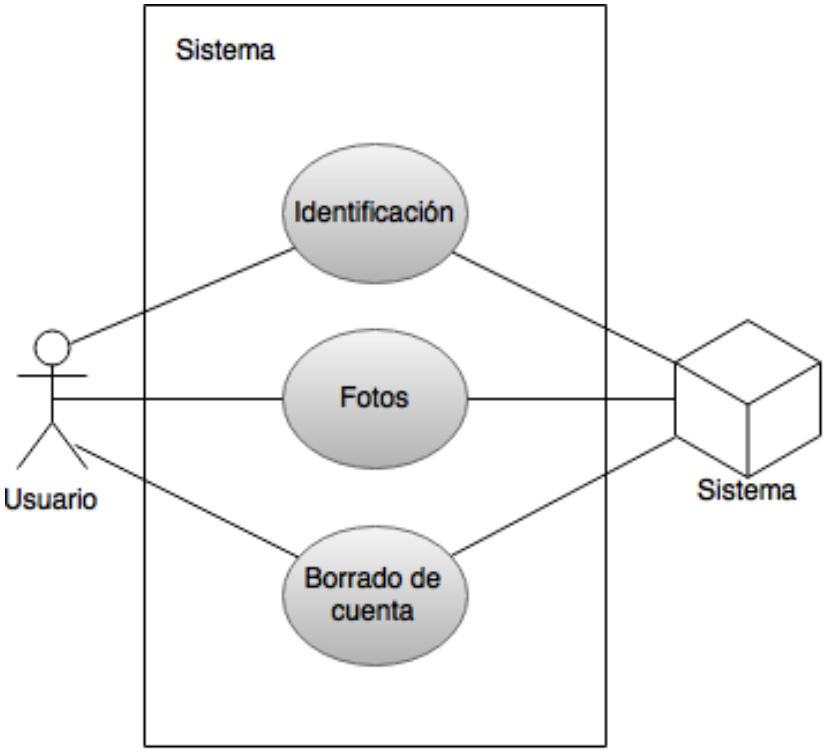


Ilustración 11: Diagrama frontera

Ahora se da paso a las funcionalidades del sistema en los diferentes casos de uso:

1. **Caso de uso: Registro**

    - **Actores participantes:** Usuario desconocido
    - **Sistema:** Aplicación Android
    - **Condiciones previas:** Ninguna
    - **Operaciones:** 1) El usuario introduce sus datos y pulsa el botón de Registrar
    - **Alternativas:**

        1) Algún campo no ha sido completado o es incorrecto y se avisa de ello
        2) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

2. **Caso de uso: Identificación**

    - **Actores participantes:** Usuario
    - **Sistema:** Aplicación Android
    - **Condiciones previas:** Ninguna
    - **Operaciones:** 1) El usuario introduce sus datos y pulsa el botón de Iniciar Sesión
    - **Alternativas:**

        1) No coinciden esos datos con los registrados en la base de datos
        2) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

3. **Caso de uso: Cerrar sesión**

    - **Actores participantes:** Usuario
    - **Sistema:** Aplicación Android
    - **Condiciones previas:** Estar identificado
    - **Operaciones:** 1) El usuario pulsa el botón de Cerrar sesión para terminar con la sesión iniciada

- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

4. **Caso de uso: Visualizar todas las fotos**

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado
- **Operaciones:** 1) El usuario pulsa el botón de Mis fotos en el menú lateral para terminar visualizar las fotos de las cámaras que tiene asociadas
- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

5. **Caso de uso: Visualizar una única foto**

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado, haber seleccionado la foto
- **Operaciones:**

    1) El usuario pulsa el botón de Guardar en el teléfono en el menú superior para guardar la foto en la memoria del teléfono
    2) El usuario pulsa el botón de Eliminar para eliminar la foto del servidor
    3) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

6. **Caso de uso: Visualizar cámaras registradas**

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado
- **Operaciones:**

1) El usuario pulsa el botón de Añadir nueva cámara para añadir una nueva cámara
2) El usuario pulsa el botón de Borrar una cámara existente

- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

7. Caso de uso: Añadir nueva cámara

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado, estar en la vista Mis cámaras
- **Operaciones:**

1) El usuario introduce los datos de la nueva cámara y pulsa Aceptar
2) El usuario cancela la operación

- **Alternativas:**

1) Algún campo no ha sido completado o es incorrecto y se avisa de ello
2) La cámara introducida no se puede registrar porque ya estaba registrada anteriormente por ese mismo usuario o por otro distinto y se avisa de ello

8. Caso de uso: Eliminar una cámara existente

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado, estar en la vista Mis cámaras
- **Operaciones:**

1) El usuario pulsa el botón de Eliminar cámara entre la lista de cámaras asociadas
2) El usuario cancela la operación

- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

9. Caso de uso: Modificar datos de usuario

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado
- **Operaciones:**

  1) El usuario pulsa el botón de Guardar para actualizar los datos modificados
  2) El usuario pulsa el botón de Eliminar cuenta para eliminar su cuenta y todas las fotos y cámaras asociadas a ella

- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

10. Caso de uso: Eliminar cuenta

- **Actores participantes:** Usuario
- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado, estar en la vista Mi cuenta
- **Operaciones:**

  1) El usuario pulsa el botón de Eliminar cuenta para borrar todos los datos de usuario con fotos y cámaras asociadas a ella y se avisa de ello
  2) El usuario cancela la operación

- **Alternativas:** 1) La conexión a Internet del teléfono está desactivada y se muestra en un mensaje

11. Caso de uso: Visualizar créditos de la aplicación

- **Actores participantes:** Usuario

- **Sistema:** Aplicación Android
- **Condiciones previas:** Estar identificado
- **Operaciones:** 1) El usuario pulsa el botón de Acerca de en el menú lateral
- **Alternativas:** Ninguna

4.3. Especificación de requisitos funcionales y no funcionales

Los requisitos se pueden subdividir en dos tipos:

4.3.1. Requisitos funcionales

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Los requerimientos de comportamiento para cada requerimiento funcional se muestran en los casos de uso. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación.

Como se define en la ingeniería de requisitos, los requisitos funcionales establecen los comportamientos del sistema.

Típicamente, un analista de requisitos genera requisitos funcionales después de realizar los casos de uso. Sin embargo, esto puede tener excepciones, ya que el desarrollo de software es un proceso iterativo y algunos requisitos son previos al diseño de los casos de uso. Ambos elementos (casos de uso y requisitos) se complementan en un proceso bidireccional.

Un requisito funcional típico contiene un nombre y un número de serie único y un resumen. Esta información se utiliza para ayudar al lector a entender por qué el requisito es necesario, y para seguir al mismo durante el desarrollo del producto.

El núcleo del requisito es la descripción del comportamiento requerido, que debe ser clara y concisa. Este comportamiento puede provenir de reglas organizacionales o del negocio, o ser descubiertas por interacción con usuarios, inversores y otros expertos en la organización [13].

Estos requisitos entonces son:

- **Registrarse:** el sistema debe permitir que nuevos usuarios puedan hacer uso de la aplicación.
- **Iniciar sesión:** el sistema debe permitir que usuarios ya registrados inicien sesión.
- **Cerrar sesión:** el sistema debe permitir que los usuarios que ya hayan iniciado sesión, puedan cerrarla.
- **Visualizar las fotos:** se debe poder visualizar las fotos tomadas por las cámaras asociadas a la cuenta de un usuario.
- **Notificaciones push:** el sistema debe ser capaz de alertar al usuario de que una nueva foto ha sido tomada por alguna de las cámaras asociadas a su cuenta.
- **Visualizar una única foto:** cada usuario debe poder ver cada foto de forma detallada, con la hora, la fecha y la cámara por la que ha sido tomada.
- **Guardar una foto:** cada usuario debe poder guardar la(s) foto(s) deseadas en la memoria del teléfono.
- **Eliminar una foto:** cada usuario debe poder eliminar la(s) foto(s) deseadas de la aplicación.
- **Añadir cámara:** se debe poder añadir una nueva cámara a la cuenta del usuario para que ésta sea capaz de enviar las fotos que tome.
- **Eliminar cámara:** se debe poder eliminar una cámara previamente asociada a la cuenta del usuario, junto con las fotos tomadas por dicha cámara.
- **Modificar usuario:** se debe poder modificar datos como el nombre, apellidos y contraseña de un usuario ya registrado.
- **Eliminar usuario:** el usuario debe poder eliminar su cuenta con toda la información relativa a ella: usuario, cámaras y fotos asociadas.

- **Visualizar créditos de la aplicación:** el usuario debe poder visualizar un apartado donde se indique la autoría de la aplicación.

Todos los requisitos mencionados se pueden clasificar atendiendo al grado de importancia en la aplicación.

Existen 3 grados de prioridad:

- **De implantación obligatoria:** son aquellos requisitos que son necesarios para poder satisfacer las necesidades del sistema:
    - Registrarse
    - Iniciar sesión
    - Cerrar sesión
    - Visualizar las fotos
    - Añadir cámara
    - Eliminar cámara
    - Eliminar usuario
- **De implantación deseable:** son aquellos requisitos cuya implantación es importante, pero cuya existencia no influye en el funcionamiento básico de la aplicación. Añaden mayor funcionalidad:
    - Notificaciones push
    - Visualizar una única foto
    - Guardar una foto
    - Eliminar una foto
    - Modificar usuario
- **De implantación opcional:** son aquellos requisitos cuya implantación es opcional. No añaden una funcionalidad extra, pero hacen que la aplicación sea aún más completa:
    - Visualizar créditos de la aplicación

4.3.2. Requisitos no funcionales

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento [14].

Algunos de estos requisitos son:

- **Rendimiento:** El sistema en general debe poder ser capaz de funcionar de forma rápida y eficaz. Para ello, se hace uso de un servidor potente con capacidades hardware que permitan el acceso de varios usuarios a la base de datos y la comunicación de múltiples cámaras.
- **Seguridad:** El sistema debe hacer uso de acceso a los datos de cada usuario por medio de una contraseña. Dicha contraseña debe almacenarse en la base de datos de forma cifrada.
- **Disponibilidad:** El servidor debe estar disponible 24 horas al día los 7 días de la semana.
- **Interfaz:** La característica más importante a tener en cuenta en este apartado es la usabilidad. Para que un sistema interactivo cumpla sus objetivos tiene que ser usable y accesible para la mayor parte de la población humana. La usabilidad es la medida en la que un producto puede ser usado por determinados usuarios para conseguir unos objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso dado. A continuación, se detallan los principios generales de la usabilidad:
    - **Facilidad de aprendizaje**: el tiempo requerido desde el no conocimiento de una aplicación hasta su uso productivo debe ser mínimo. Para ello el sistema debe ser:
        - **Sintetizable**: cuando una operación produce un cambio en el sistema, el usuario debe poder captarlo.

- **Familiar**: debe existir una correlación entre los conocimientos que posee el usuario y los conocimientos requeridos para la interacción en un sistema nuevo.

- **Flexibilidad**: es la multiplicidad de maneras en que el usuario y el sistema pueden intercambiar información. Algunos parámetros que miden la flexibilidad son:

  - **Control de usuario**: el usuario es quien conduce la interacción (cancelar acciones, alertar sobre operaciones que puedan conllevar a borrados no deseados, etc.).
  - **Adaptabilidad**: es la adecuación automática de la interfaz al usuario.

- **Consistencia**: un sistema es consistente si todos los mecanismos que se utilizan son siempre usados de la misma manera, siempre que se utilicen y sea cual sea el momento en que se haga. Lo mejor es seguir una guía de estilo.
- **Robustez**: el sistema debe permitir al usuario conseguir sus objetivos sin problemas.
- **Recuperabilidad**: el sistema debe permitir al usuario corregir una acción una vez ésta ha sido reconocida como errónea.
- **Tiempo de respuesta**: es el tiempo que necesita el sistema para expresar los cambios de estado al usuario. Los tiempos de respuesta deben ser soportables para el usuario.
- **Adecuación de las tareas**: el sistema debe permitir todas las tareas que el usuario quiere hacer y en la forma en que éste las quiere hacer.
- **Disminución de la carga cognitiva**: debe favorecerse en los usuarios el reconocimiento sobre el recuerdo. Los usuarios no deben tener que recordar abreviaturas y códigos complicados.

5. Diseño del sistema

5.1. Introducción

Esta fase proporciona la funcionalidad de un sistema a través de sus componentes.

El diseño de sistemas es el arte de definir la arquitectura de hardware y software, componentes, módulos y datos de un sistema de cómputo, a efectos de satisfacer ciertos requerimientos. Es la etapa posterior al análisis de sistemas.

A lo largo de todo este proceso de diseño se hace hincapié en los servicios Web como método de intercambio de información entre los componentes que forman el sistema, la estructura de los datos utilizada para garantizar la persistencia de estos, el sistema de la cámara en la placa Raspberry Pi, el diseño de la interfaz para la aplicación Android dirigida a los usuarios y la arquitectura del software.

5.2. Servicios Web

El consorcio W3C define los Servicios Web como sistemas software diseñados para soportar una interacción interoperable maquina a máquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja.

Las características de un Servicio Web son:

- Un servicio debe poder ser accesible a través de la Web. Para ello debe utilizar protocolos de transporte estándares como HTTP, y codificar los mensajes en un lenguaje estándar que pueda conocer cualquier cliente que quiera utilizar el servicio.
- Un servicio debe contener una descripción de sí mismo. De esta forma, una aplicación podrá saber cuál es la función de un determinado Servicio Web, y cuál es su interfaz, de manera que pueda ser utilizado de forma automática por cualquier aplicación, sin la intervención del usuario.
- Debe poder ser localizado. Deberemos tener algún mecanismo que nos permita encontrar un Servicio Web que realice una determinada función.

De esta forma tendremos la posibilidad de que una aplicación localice el servicio que necesite de forma automática, sin tener que conocerlo previamente el usuario

Se pueden diferenciar dos tipos de Servicios Web:

- **SOAP (Simple Object Access Protocol)**: es un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP. Está basado en XML, esto facilita la lectura, pero también los mensajes resultan más largos y, por lo tanto, considerablemente más lentos de transferir.

  Existen múltiples tipos de modelos de mensajes en SOAP, pero de lejos, el más común es el RPC (Remote Procedure Call), en donde un nodo de red (el cliente) envía un mensaje de solicitud a otro nodo (el servidor) y el servidor inmediatamente responde el mensaje al cliente.

  Los mensajes SOAP, son independientes del sistema operativo, y pueden transportarse en varios protocolos de internet como SMTO, MIME y HTTP.

- **REST (Representational State Transfer**): es un estilo de arquitectura de software dirigido a sistemas hipermedias distribuidos como lo es la Web y se refiere específicamente a una colección de principios (los cuales resumen la forma en que los recursos son definidos y diseccionados) para el diseño de arquitecturas en red.

  Este término es utilizado en su mayoría para describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como lo hace SOAP, en tal sentido éstos dos significados pueden chocar o incluso solaparse.

  Es importante entender que es posible diseñar un sistema software de gran tamaño de acuerdo con esta arquitectura propuesta sin utilizar HTTP o sin interactuar con la Web, así como también diseñar una simple interfaz XML+HTTP que no sigue los principios REST, y en cambio seguir un modelo RPC.

Es importante remarcar el hecho de que REST no es un estándar, ya que es tan solo un estilo de arquitectura, pero también está basado en los siguientes estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG/…
- Tipos MIME: text/xml, text/html.

En esta tabla se pueden ver algunas de las diferencias entre SOAP y REST:

SOAP	REST
Es un protocolo	Es un estilo de arquitectura software
SOAP puede utilizar además el protocolo HTTP o SMTP	REST puede usar servicios web basados en SOAP porque es puede usar protocolos como HTTP o SOAP
SOAP define estándares que deben ser estrictamente cumplidos	REST requiere menos ancho de banda y recursos que SOAP
SOAP define su propia seguridad	REST hereda las medidas de seguridad de los protocolos que usa (como HTTP)
SOAP permite el formato de datos XML únicamente	REST permite diferentes tipos de formato como texto plano, HTML, XML o JSON

Tabla 4: Comparativa SOAP y REST

Los servicios web se van a implementar usando el protocolo HTTP, y puesto que personalmente me siento más cómodo usando formatos como JSON y la información que se envía por dicho protocolo no requiere que sea muy grande, la elección escogida será la de REST. Las imágenes que enviará la cámara serán codificadas a una cadena (string) para enviarlas por dicho protocolo hasta llegar al servidor, donde

se decodificará para convertirla de nuevo a imagen y almacenarla. De esta forma se ayuda a que la transmisión de datos sea lo más rápida posible.

Aquí se pueden ver las operaciones que se pueden realizar desde el protocolo HTTP y sus equivalencias con otras tecnologías :

Acción	HTTP	SQL	Copy & Paste	Unix Shell
Create	PUT	Insert	Pegar	>
Read	GET	Select	Copiar	<
Update	POST	Update	Pegar después	>>
Delete	DELETE	Delete	Cortar	Del/rm

Tabla 5: Comparativa acciones de HTTP con otros sistemas

[15]

5.3. Estructura de los datos

En este apartado se explica la estructura que tendrán los datos almacenados en el servidor. Antes de nada, se explicará qué tipo de base de datos se ha elegido.

Las bases de datos relacionales están muy estandarizadas hoy día, aunque también están despegando las no relacionales. El principal inconveniente que supone usar una base de datos relacional como SQL es la flexibilidad. Con un diseño orientado a propiedades como puede ser OWL (base de datos no relacional) se consiguen diseños mucho más libres, donde al añadir una propiedad no se tiene que modificar ninguna estructura ni ninguna tabla. Esto es gracias a que la estructura que usa internamente OWL es un grafo.

Debido a que las relaciones y las entidades requeridas para este proyecto no son de gran magnitud y el sistema se queda algo acoplado debido a la tecnología que se va a usar (placa base Raspberry Pi), no se espera que haya una modificación futura que pueda resultar catastrófica para la base de datos y se deba modificar a gran escala. Por tanto, se ha optado por una base de datos relacional como SQL. Además, en el lado del servidor es muy fácil y usable la tecnología MySQL para gestionar la base de datos.

Al igual que en OWL se usan grafos para representar la información, en SQL se utiliza el modelo Entidad-Relación.

El modelo Entidad-Relación es una técnica especial de representación gráfica que incorpora información relativa a los datos y la relación existente entre ellos para darnos una visión del mundo real. Entre las características del modelo E-R:

- Refleja tan sólo la existencia de los datos, no lo que se hace con ellos.
- Se incluyen todos los datos del sistema en estudio y, por tanto, no es orientado a aplicaciones particulares.
- Es independiente de las bases de datos y sistemas operativos concretos.
- No se tienen en cuenta restricciones de espacio, almacenamiento ni tiempo de ejecución.
- Está abierto a la evolución del sistema.

En el modelo E-R se describen las entidades como conceptos u objetos, atributos como unidades básicas e indivisibles de información acerca de una entidad que sirven para identificarla o describirla y las relaciones entre las entidades. Aquí se puede ver cómo se representan cada uno de estos elementos:

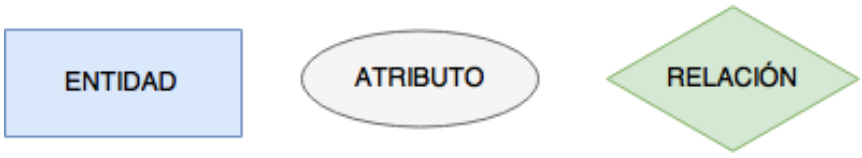


Ilustración 12: Componentes modelo Entidad-Relación

Las relaciones pueden ser de distintos tipos:

- **Relación 1-1**: Es aquella relación en la que una instancia de una entidad le corresponde otra instancia de la otra entidad.
- **Relación 1-N**: Es aquella relación en la que una instancia de una entidad le corresponden muchas instancias de la otra entidad.
- **Relación N-M**: Es aquella relación en la que muchas instancias de una entidad corresponden con muchas instancias de la otra entidad. Esta relación da lugar a una nueva tabla, que será la unión de las dos entidades, más atributos propios (si los tuviera).

A continuación, se representa el modelo E-R. Debido a que no existe una relación de muchos a muchos entre entidades, es el modelo final de datos.

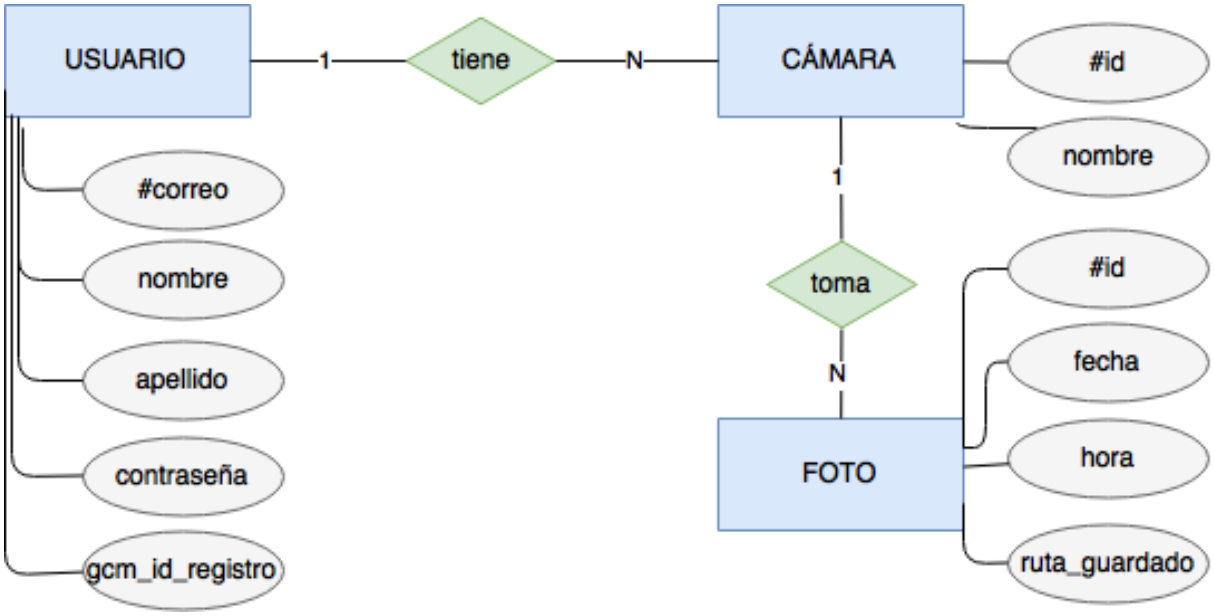


Ilustración 13: Modelo Entidad-Relación

Ahora es el turno de la estructura de cómo quedan las tablas ya creadas en la base de datos del servidor. La base de datos se denomina 'vigilancia':

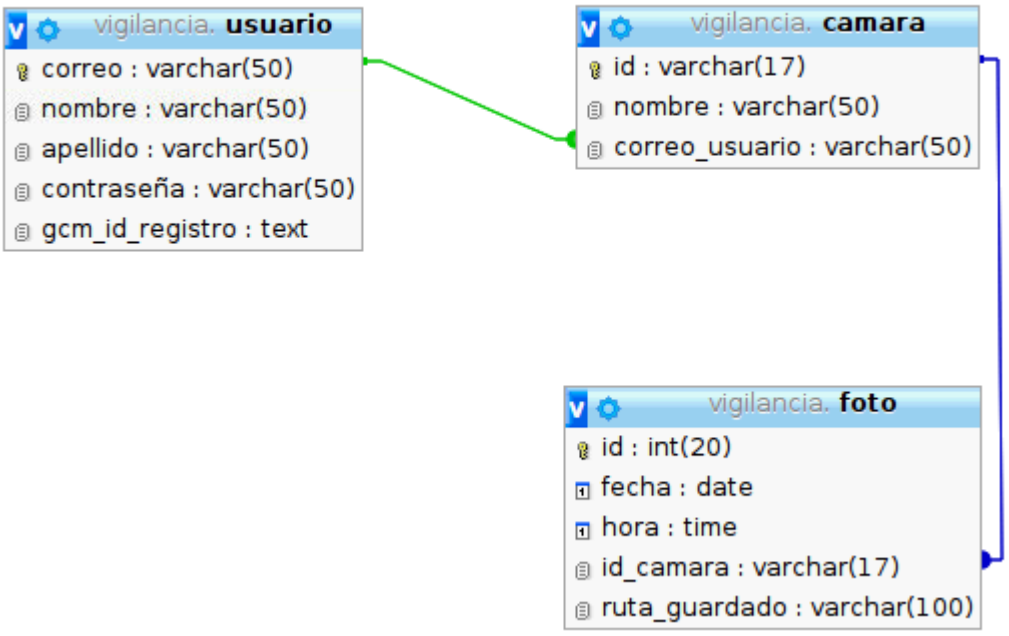


Ilustración 14: Tablas de la base de datos

- Tabla **usuario**: esta tabla representa a la persona que manejará la aplicación móvil. Sus atributos son:

    - ***correo***: es la clave primaria de la tabla. Identifica a cada usuario de forma única. Es una forma de identificación muy común en las aplicaciones de hoy en día.
    - ***nombre***: el nombre del usuario.
    - ***apellido***: el primer apellido del usuario.
    - ***gcm_id_registro***: es una id o token que se utiliza para poder mandar notificaciones push a cada usuario.

- Tabla **camara**: esta entidad representa cada una de las cámaras que puede tener un usuario. Una cámara solo puede ser única y estar asociada a un único usuario. El usuario, sin embargo, puede tener varias cámaras. Sus atributos son:

    - ***id***: es la clave primaria de la cámara. Éste identificador posee 17 campos puesto que será la dirección física o MAC del dispositivo Ethernet de cada cámara separado por '-'.
    - ***nombre***: un nombre asociada a la cámara.
    - ***correo_usuario***: es la clave foránea de la tabla usuario. Es la resultante de la relación entre usuario y cámara descrita anteriormente.

- Tabla **foto:** en esta tabla se almacenan los datos relativos a cada foto. Una foto se corresponde con una única cámara, pero una cámara puede realizar varias fotos. Sus atributos son:

    - ***Id:*** es la clave primaria de la foto.
    - ***fecha:*** este atributo almacena la fecha en la que fue tomada la foto por una cámara.
    - ***hora:*** almacena la hora en la que fue tomada una foto por una cámara.

- ***id_camara:*** es la clave foránea de la tabla canara. Es la resultante de la relación entre la cámara y la foto descrita anteriormente.
- ***ruta_guardado:*** se almacena la ruta donde la foto se ha almacenado en el sistema de archivos del servidor.

5.4. Raspberry Pi

En este apartado se expone con fotos cómo queda el sistema de vigilancia compuesto por la Raspberry Pi, la cámara y el sensor de movimiento PIR.

Ilustración 15: Componentes del sistema de la cámara

Ilustración 16: Sistema de la cámara montado

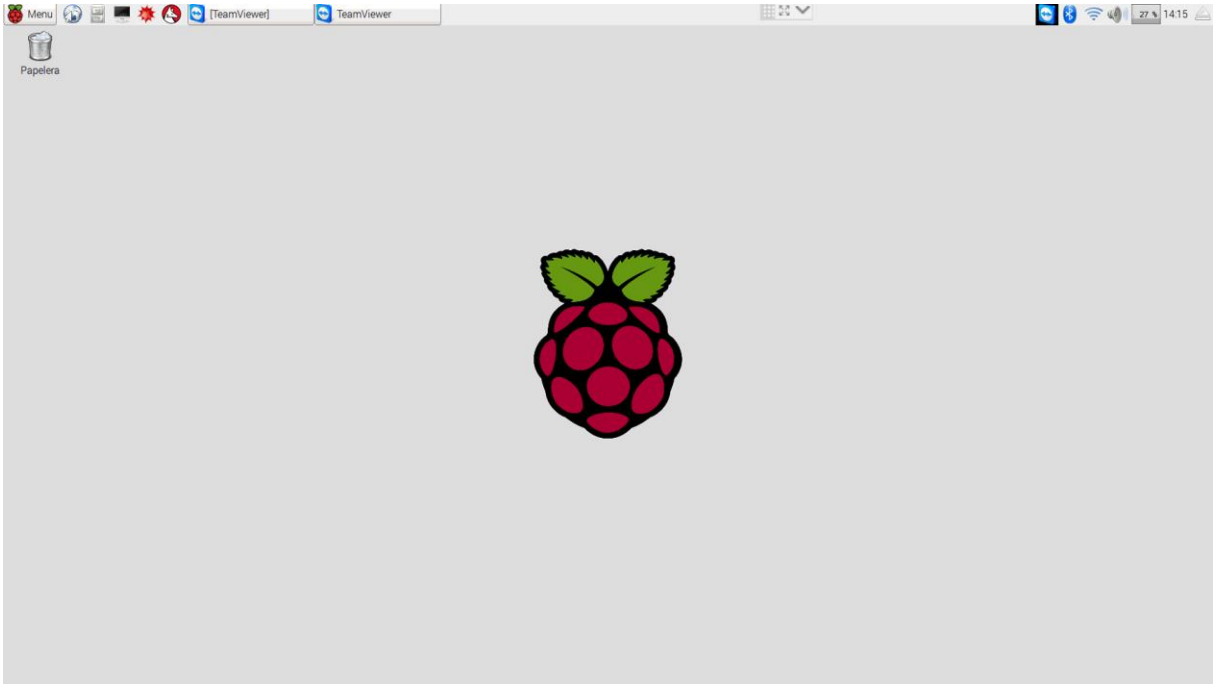


Ilustración 17: Sistema operativo Raspbian

5.5. Diseño de la interfaz

El diseño de interfaz de usuario o ingeniería de la interfaz es el diseño de computadoras, aplicaciones, máquinas, dispositivos de comunicación móvil, aplicaciones de software, y sitios web enfocado en la experiencia de usuario y la interacción.

Normalmente es una actividad multidisciplinar que involucra a varias ramas, es decir, al diseño y el conocimiento como el diseño gráfico, industrial, web, de software y la ergonomía; y está implicado en un amplio rango de proyectos, desde sistemas para computadoras, vehículos hasta aviones comerciales.

Su objetivo es que las aplicaciones o los objetos sean más atractivos y además, hacer que la interacción con el usuario sea lo más intuitiva posible, conocido como el diseño centrado en el usuario. En este sentido las disciplinas del diseño industrial y gráfico se encargan de que la actividad a desarrollar se comunique y aprenda lo más rápidamente, a través de recursos como la gráfica, los pictogramas, los estereotipos y la simbología, todo sin afectar el funcionamiento técnico eficiente [16].

¿Y qué es una interfaz? Es una superficie de contacto entre dos entidades. Así pues, en una bicicleta las interfaces son los pedales, sillín y manillar o en una puerta es el pomo para abrir o cerrarla. En informática se pueden definir las interfaces como las partes del sistema con las que el usuario entra en contacto física y cognitivamente. El usuario realiza dos tipos de interacciones:

- Interacción física (teclado, ratón, pantalla…)
- Interacción cognitiva (lo que se presenta al usuario debe ser comprensible para él)

Para conseguir una buena interfaz se hace uso entonces de guías de estilo, Storyboards y metáforas.

5.5.1. Guía de estilo y metáforas

En este apartado se hace hincapié en la guía de estilo utilizada para desarrollar la aplicación móvil. El estilo seguido es el estipulado por Google para las aplicaciones Android. Dicho estilo se denomina ‘Material Design’.

Material Design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma. Fue desarrollado por Google y anunciado en la conferencia Google I/O celebrada el 25 de junio de 2014.

Material se integró en Android Lollipop como reemplazo de Holo, anteriormente utilizado desde Android 4 y sucesores. La filosofía también se aplicó en Google Drive y Google Docs, Sheets y Slides, y se irá extendiendo progresivamente a todos los productos de Google (incluyendo Google Search, Gmail y Google Calendar), proporcionando una experiencia consistente en todas las plataformas. Google también lanzó APIs para que los desarrolladores externos incorporaran Material Design a sus aplicaciones [17].

La idea principal de Material Design es servir de lenguaje de diseño común en un amplio rango de dispositivos y, por tanto, de experiencias de usuario. Desde el móvil hasta la Web, pasando por las tabletas, podrán servirse de este lenguaje de diseño. Está diseñado sobre todo para ser utilizado en dispositivos táctiles, pero no deja de lado otros métodos de entrada como el teclado, el ratón o la entrada por voz.

Todos los elementos de una interfaz que siga este lenguaje de diseño pueden tener relieve. El mundo real no es plano, por lo que la representación del mundo real que Material Design nos permitiría crear tampoco lo sería. De igual modo, un objeto en el mundo real sigue una serie de leyes físicas; cada objeto dentro de una interfaz de Material Design también debería seguirlas.

Lo principal es entender las aplicaciones y los elementos que forman parte de ellas, porque su interacción será idéntica a la del mundo real. La tipografía también tiene mucho más sentido en este nuevo lenguaje de diseño: distintos tamaños y cuerpos permiten crear jerarquías. Tienen un sentido, más allá de ser atractivas a la vista. Las animaciones también buscan imitar la interacción de los objetos en el mundo real (60 frames por segundo, incluso en la Web).

Para conseguir un estilo en la aplicación acorde a los estándares de Google se detallan a continuación algunos de los elementos a destacar en la aplicación:

- **Paleta de colores:** El rojo para las alertas. Como bien es sabido el rojo es un color para captar la atención del usuario y no se debe emplear de forma abusiva.

  Para los colores principales de la aplicación se ha empleado el azul y naranja. El azul y el naranja son colores complementarios en el círculo cromático, por lo que quedan estéticamente muy bien.

Ilustración 18: Paleta de colores usada en la aplicación Android

- **Fuente:** La fuente utilizada es la que trae por defecto Android en la versión Marshmallow.
- **Flat Button:** es un botón compuesto por “tinta” que muestra reacciones cuando se presiona y no se levanta.

  Se ha empleado para acciones como iniciar sesión, registrarse, eliminar cuenta, etc.

Ilustración 19: Flat Button en la aplicación Android

- **Floating Action Button:** estos botones son usados para promocionar una acción. Se distinguen por un icono circular flotando sobre la interfaz gráfica y tiene comportamientos de movimiento como transformaciones, lanzamientos y transferencias a un punto de anclaje.

  El empleado en la aplicación representa la acción de añadir.

Ilustración 20: Floating Action Button en la aplicación Android

- **Toasts:** usados para mostrar mensajes del sistema. Los toasts son similares a los snackbars a diferencia de que no contienen acciones y no pueden ser eliminadas. Estos desaparecen tras un breve período de tiempo. Aquí un ejemplo de uso de toasts:

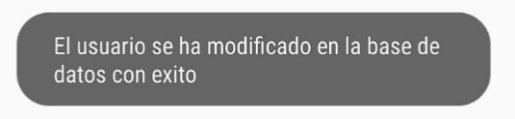


Ilustración 21: Toast en la aplicación Android

- **Menú:** los menús permiten a los usuarios realizar acciones siendo seleccionadas de una lista de opciones que abren temporalmente una nueva hoja de 'material':

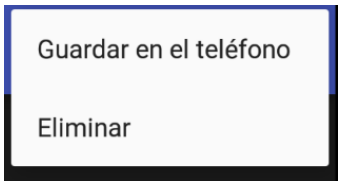


Ilustración 22: Menú en la aplicación Android

- **Alerts:** Los alerts son interrupciones urgentes e informan al usuario acera de una situación. Requieren de aceptación o cancelación.

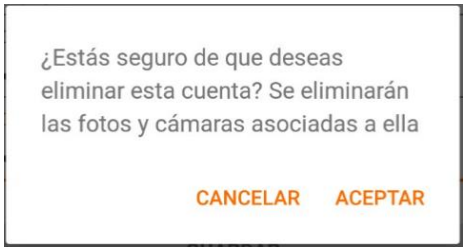


Ilustración 23: Alert en la aplicación Android

- **Toolbar:** los toolbars aparecen un paso por encima de la 'hoja de papel' y es afectada por las acciones que se realicen en este.

Ilustración 24: Toolbar en la aplicación Android

- **Cards:** una card o carta es un pedazo de papel donde se detalla información. Un ejemplo del uso de este widget:

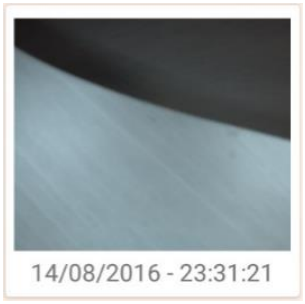


Ilustración 25: Card en la aplicación Android

- **Floating labels:** cuando el usuario interactúa con el campo de entrada de texto, el texto que indica qué debe escribir en dicha línea de texto se desplaza hasta la parte superior.

Ilustración 26: Floating label en la aplicación Android

- **User input errors:** esto es usado para indicar al usuario que algo en el campo de texto es erróneo:

Ilustración 27: User input error en la aplicación Android

- **Text field:** son los campos de texto donde el usuario introduce un texto.

Ilustración 28: Text field en la aplicación Android

- **Notificaciones Bigstyle:** El sistema de notificaciones permite a los usuarios mantenerse informados sobre eventos relevantes y oportunos de

su aplicación, como nuevos mensajes de chat de un amigo o un evento del calendario.

Se debe pensar en las notificaciones como un canal de noticias que alerta a los usuarios sobre eventos importantes a medida que se producen. Las notificaciones aparecen sobre la barra de estado.

Puedes decidir cuántos detalles mostrarán las notificaciones de tu aplicación. Las notificaciones pueden mostrar las primeras líneas de un mensaje o la vista previa de una imagen más grande.

A través de la información adicional, se proporciona más contexto al usuario y, en algunos casos, se puede permitir que el usuario lea todo el mensaje, en mi caso, que el usuario visualice la nueva foto.

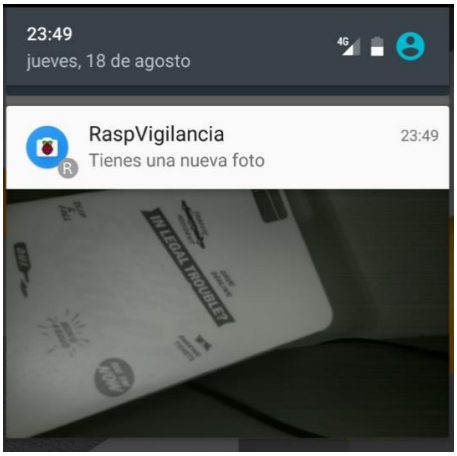


Ilustración 29: Notificación Big Style en la aplicación Android

- **Navigation drawer:** es un componente que se desliza desde la izquierda para mostra contenido. Es un patrón muy común en aplicaciones de Google y sigue los mismos estándares que las listas.

  Normalmente constituye la navegación principal de la aplicación. Se encuentra escondida casi siempre, pero es mostrada cuando el usuario desliza el dedo desde el borde izquierdo de la pantalla, o cuando se pulsa el icono del Toolbar.

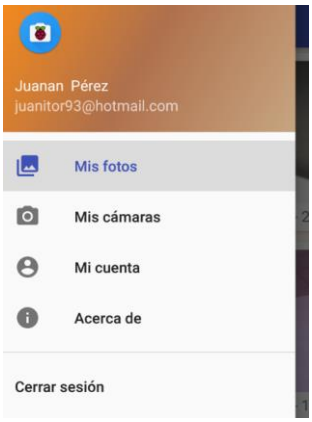


Ilustración 30: Navigation Drawer en la aplicación Android

- **Swipe to refresh:** Swipe to refresh es un gesto de deslizamiento que está disponible al comienzo de la listas, grids y colecciones de cards donde aparece el contenido más reciente. Esto se ha utilizado para sincronizar de forma manual las fotos que el usuario puede visualizar con las que se encuentran en el servidor.

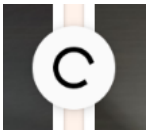


Ilustración 31: Swipe to refresh en la aplicación Android

[18] [19]

5.5.2. Storyboard

Storyboard o guión gráfico es un conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender una historia, pre-visualizar una animación o seguir la estructura de una película antes de realizarse o filmarse. El storyboard es el modo de pre-visualización que constituye el modo habitual de pre-producción en la industria del cine.

La creación del storyboard se atribuye a Georges Méliès. El proceso de storyboarding, en la forma que se conoce hoy, fue desarrollado por Webb Smith en el estudio de Walt Disney durante principios de los años 1930, después de varios años de procesos similares que fueron empleados en Disney y en otros estudios de animación. El storyboarding se hizo popular en la producción de películas de acción durante principios de los años 1940 [20].

De cara a nuestro propósito en esta fase del proyecto (que no es otro que el diseño de una aplicación móvil), el storyboard nos sirve de prototipo para realizar un diseño centrado al usuario para recoger sus necesidades y mejorar la utilización.

Un prototipo es un documento, diseño o sistema que simula o tiene implentadas partes del sistema final. Es una herramienta muy útil para hacer participar al usuario en el diseño y poder evaluarlo ya en las primeras fases del proyecto.

En mi caso, se ha supuesto que un cliente es un compañero junto con la persona que tutoriza este proyecto, para obtener distintas opiniones y una retroalimentación.

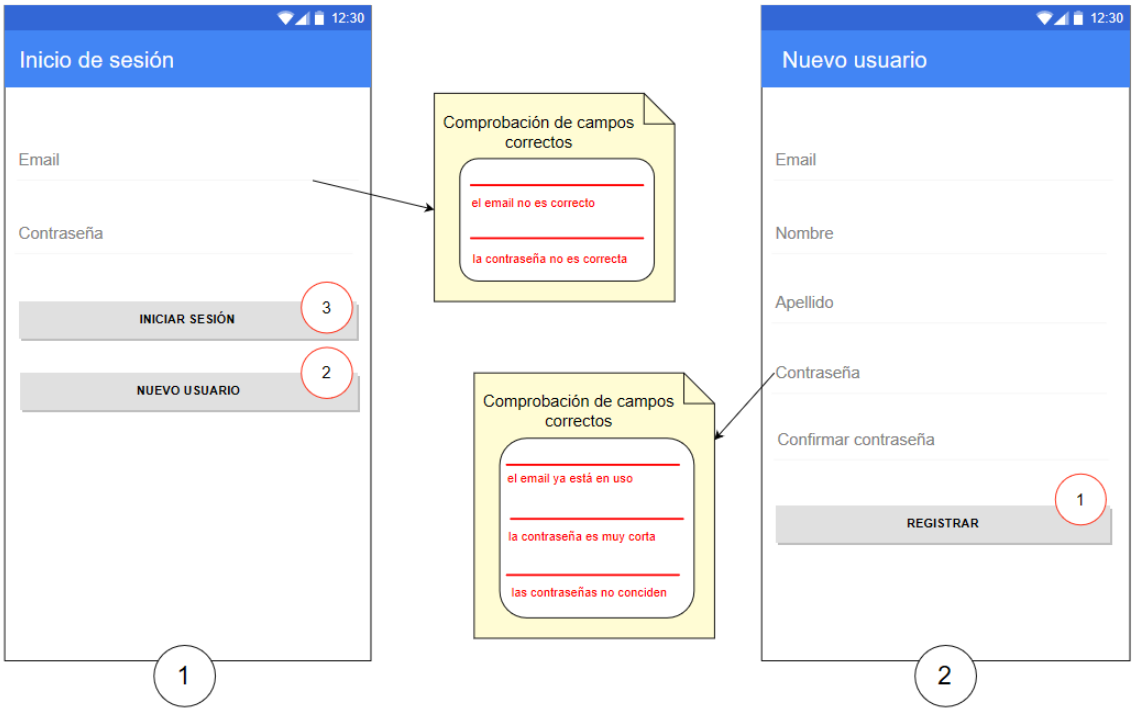


Ilustración 32: Iniciar sesión y nuevo usuario del Storyboard

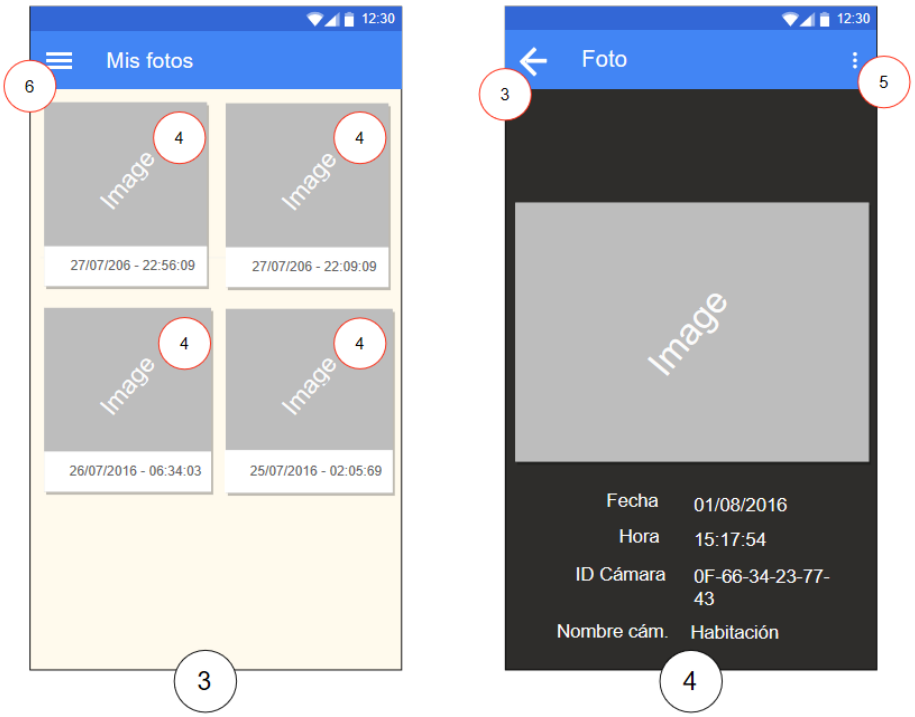


Ilustración 33: Fotos y foto del Storyboard

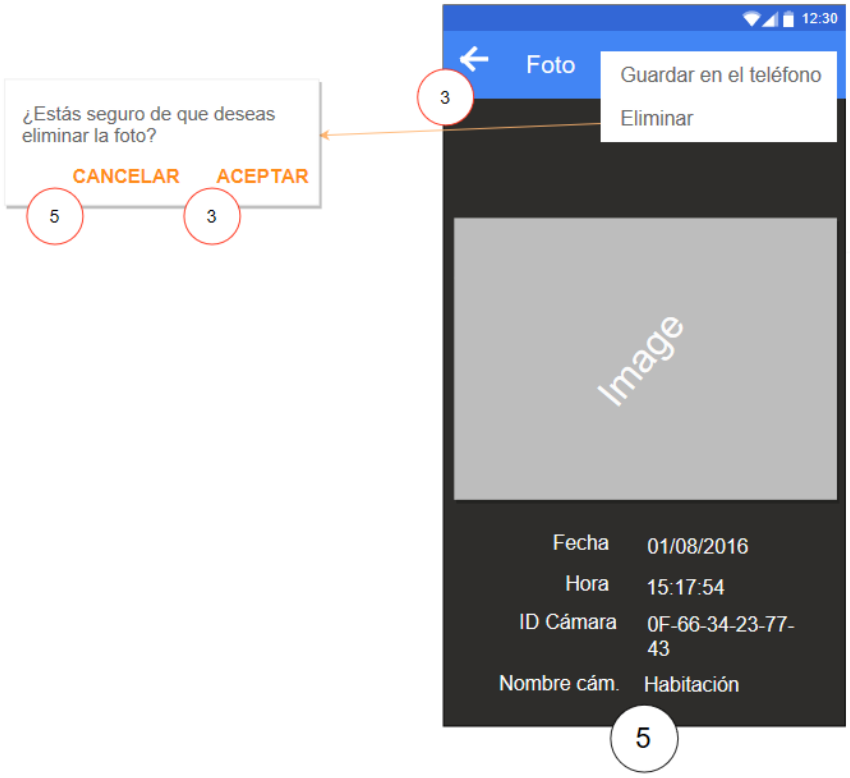


Ilustración 34: Menú de foto del Storyboard

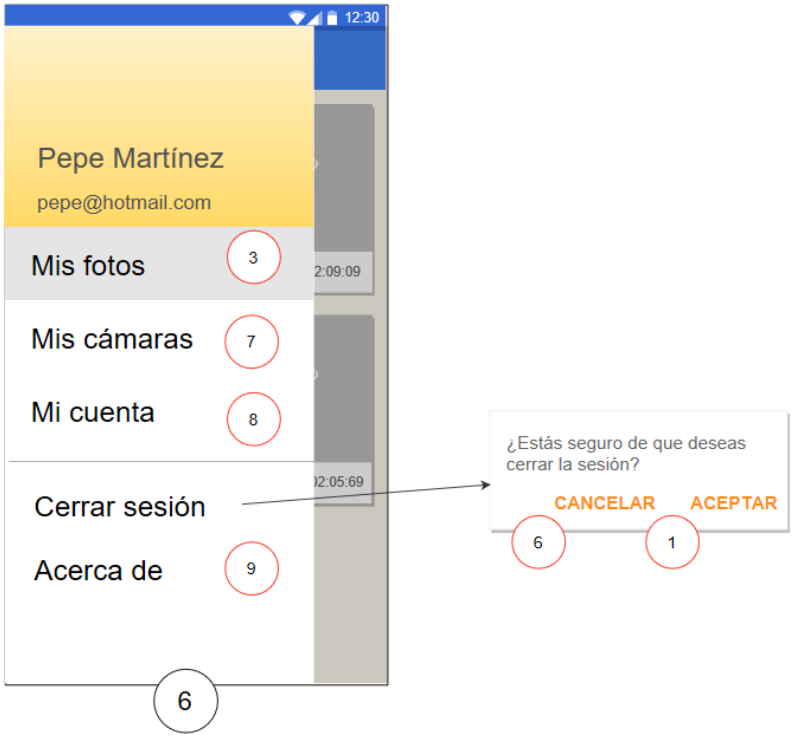


Ilustración 35: Cerrar sesión del Storyboard

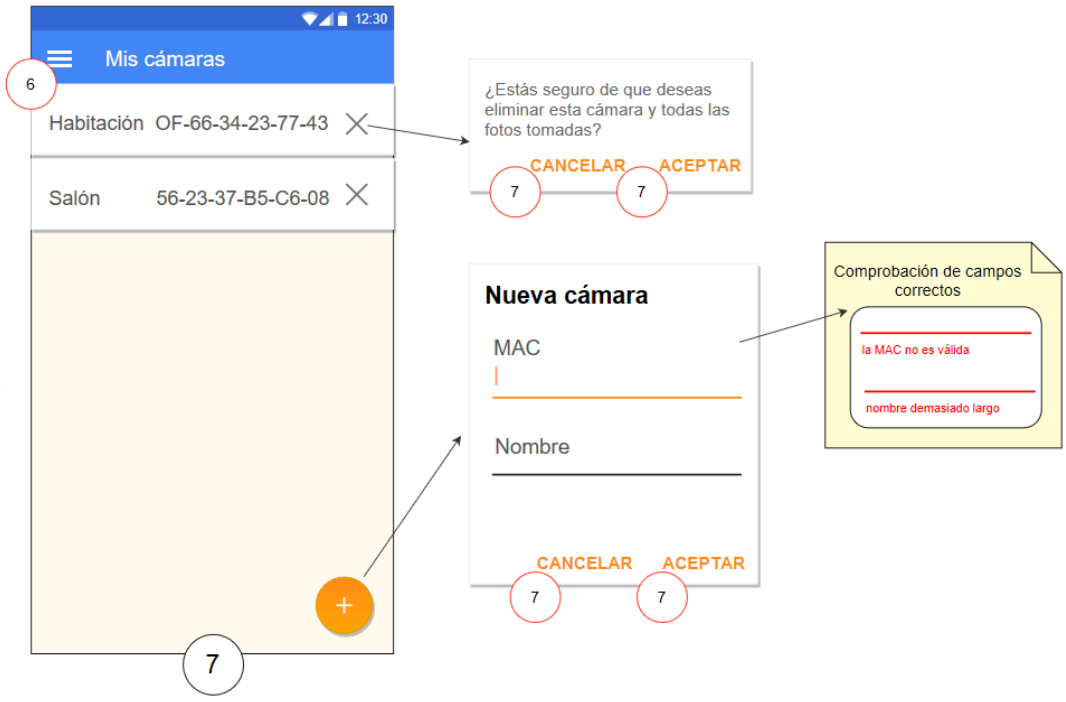


Ilustración 36: Mis cámaras del Storyboard

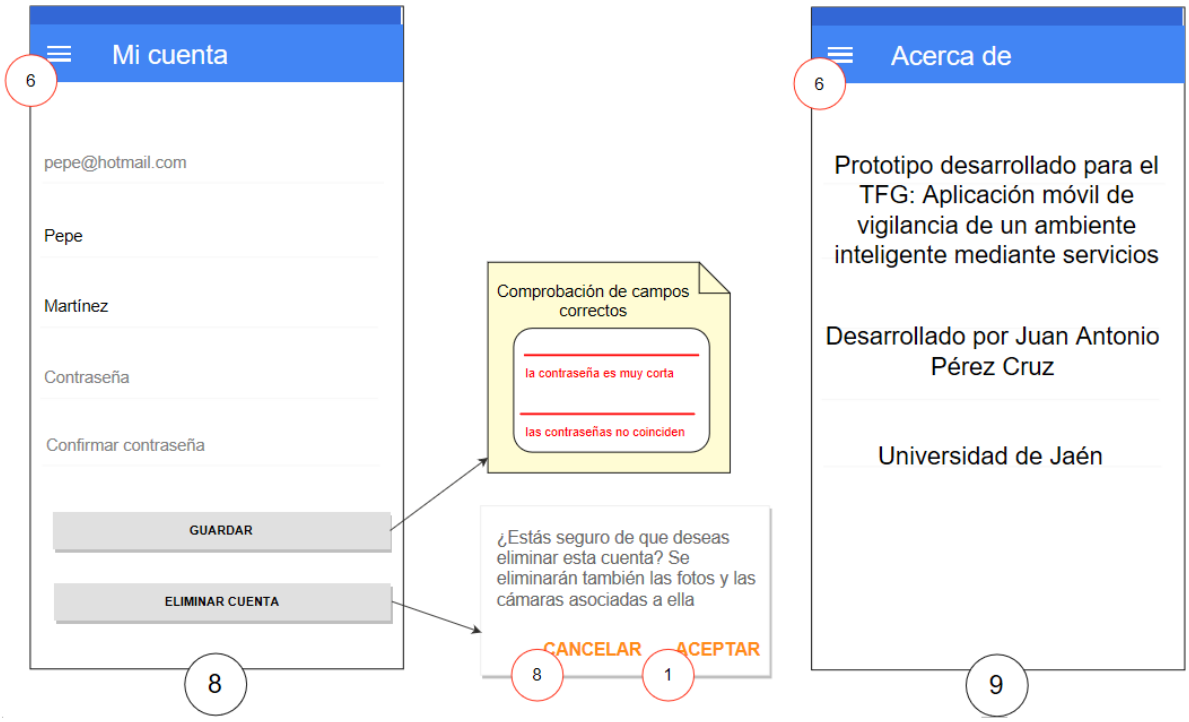


Ilustración 37: Mi cuenta y Acerca de del Storyboard

5.6. Arquitectura del software

Tal y como se verá posteriormente en la implementación del sistema, la interacción entre la cámara y el servidor usa una arquitectura Cliente-Servidor.

El sistema de la cámara envía una petición al servidor empleando servicios web (REST) con la foto tomada e información sobre esta. El servidor atiende esa petición y almacena los datos en la base de datos.

En cuanto al servidor se utiliza un patrón Modelo-Controlador. Esto es debido a que el modelo contiene los métodos que tratan los datos para manejarlos en la base da datos y el controlador contiene los métodos que atienden las peticiones que llegan mediante los servicios web (ya sea desde la cámara o bien desde la aplicación Android) para más adelante interactuar con el modelo y que éste gestione los datos. La Vista no existe como tal, ya que es la aplicación móvil la que contiene la interfaz y la interacción con el usuario.

6. Implementación y pruebas del sistema

6.1. Introducción

En este apartado se explica de forma detallada el proceso de programación de cada componente que forma el sistema. El sistema está formado por la cámara, el servidor y el prototipo de aplicación móvil. No hay que olvidar otros dos principales puntos muy a tener en cuenta: los servicios web y los servicios Google Cloud Messaging (GCM) para poder enviar notificaciones push a la aplicación. De cada uno se indican las principales tecnologías utilizadas y por qué han sido elegidas.

También se incluyen las pruebas realizadas al sistema para comprobar su correcto funcionamiento y corregir posibles errores.

En este diagrama se puede ver el funcionamiento básico del sistema:

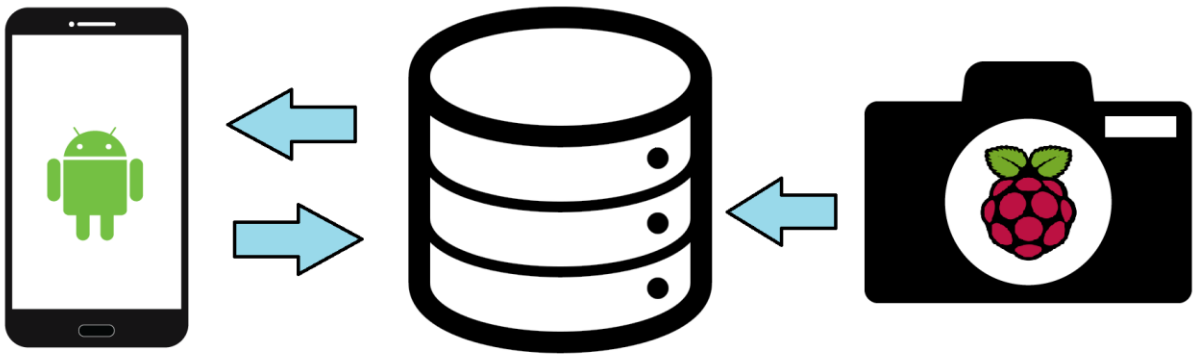


Ilustración 38: Diagrama de funcionamiento del sistema

El sistema de la cámara está controlado por la placa Raspberry Pi y su sistema operativo Raspbian OS, que está basado en Debian (Linux) y permite interactuar con los módulos de la cámara y del sensor de movimiento. Cuando la cámara consigue una foto la envía al servidor mediante servicios web.

El servidor almacena en la base de datos las fotos que le llegan de la cámara y los datos de los usuarios. Él es el cerebro de todo el sistema, pues es quien acepta las peticiones de los servicios web empleados tanto por parte de la cámara como por parte de la aplicación móvil.

La aplicación móvil interactúa con el usuario y el servidor para mostrar, almacenar, borrar e incluir nuevos datos.

6.2. Desarrollo del sistema de vigilancia

6.2.1. Raspberry Pi

Con la cámara y el sensor de movimiento PIR detallados anteriormente conectados a la placa, es el momento de programar el comportamiento del sistema de vigilancia.

El estado del sensor varía en función de que haya detectado movimiento o no. Cuando el movimiento haya sido detectado, la cámara toma una foto y la envía al servidor mediante una petición POST del protocolo HTTP.

Todo esto se hace de forma simple y rápida gracias a la librería 'request' que posee Python. Se especifica el método en la dirección URL del servidor y se envían los parámetros. Antes de nada, para poder tomar una foto se necesita la librería de la cámara denominada 'picamera' que se puede encontrar en el sistema operativo Raspbian instalado en la placa.

Para garantizar una transmisión rápida y con un consumo de recursos en la red mínimo se hace uso de una librería denominada 'base64'. Aquí se codifica una imagen y pasa a ser un dato de tipo string (texto plano). Este codificador/decodificador se encuentra disponible también en PHP, por lo que en la parte del servidor se decodifica la imagen en una sola línea para poder almacenarla.

6.2.2. Servidor

Se ha optado por elegir un host propio con sistema operativo basado en Linux. El servidor utiliza el servicio Apache.

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que alguien quería que tuviese la connotación de algo que es firme y enérgico, pero no agresivo, y la tribu

Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además, Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, a patchy server (un servidor "parcheado") suena igual que Apache Server.

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Además, tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Jugó un papel fundamental en el desarrollo fundamental de la World Wide Web y alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo, ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft3). En 2009 se convirtió en el primer servidor web que alojó más de 100 millones de sitios web.

Entre sus ventajas encontramos:

- Es modular
- De código abierto
- Multiplataforma
- Extensible
- Popular (fácil conseguir ayuda o soporte)

[21]

El lenguaje de programación empleado para el servidor ha sido PHP.

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue

uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy, lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico, como Facebook, para optar por el mismo como tecnología de servidor.

Fue creado originalmente por Rasmus Lerdorf en 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP. Este lenguaje forma parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término PHP.

Entre sus características encontramos:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de php arrays.
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.

- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.
- Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones WEB de manejo de contenido, y es su uso principal [22].

Puesto que se van a utilizar servicios web, se ha optado por escoger un framework que facilite la implementación de las peticiones HTTP como GET y POST, así como el manejo del modelo de datos. Este framework es Slim framework.

Slim framwework es un micro framework para PHP que nos permite escribir rápidamente aplicaciones web y APIs. Funciona de la misma forma que Laravel o Code Igniter pero con funciones más limitadas. Para la implementación del servidor del proyecto es más que suficiente. Está preparado para emplear el Modelo-Vista-Controlador, aunque en este caso la vista no se trataría de ninguna forma puesto que

no se encuentra embebida en el sistema web, sino que se encuentra en la aplicación Android. Entre sus características se encuentran:

- Creador de rutas bastante potente:
    - Soporta métodos HTTP standard y personalizados.
    - Parámetros de ruta con comodines y condiciones.
    - Redirecciones de rutas, paros y saltos.
- Renderizado de plantillas y vistas personalizadas.
- Mensajes Flash.
- Encriptación segura de cookies con AES-256.
- Caché HTTP.
- Logging de accesos personalizado.
- Gestión de errores.
- Configuración sencilla

A continuación, se detallan los métodos empleados para interactuar con la cámara de vigilancia y con la aplicación Android por medio de los servicios web REST:

Almacenar una foto:

URL: http://localhost/webservice/uploadPhoto

Parámetros: id_camara, image, time, date

Método: POST

Iniciar sesión:

URL: http://localhost/webservice/login

Parámetros: email, password

Método: POST

Registrar usuario:

URL: http://localhost/webservice/userRegister

Parámetros: email, password, name, surname

Método: POST

Registrar cámara:

URL: http://localhost/webservice/registerCam

Parámetros: id, name, email

Método: POST

Obtener las fotos:

URL: http://localhost/webservice/getPhotos

Parámetros: email

Método: GET

Borrado de foto:

URL: http://localhost/webservice/deletePhoto

Parámetros: id_camara, time, date

Método: POST

Obtener cámaras:

URL: http://localhost/webservice/getCameras

Parámetros: email

Método: GET

Borrado de cámara:

URL: http://localhost/webservice/deleteCamera

Parámetros: id_camara

Método: POST

Modificar usuario:

URL: http://localhost/webservice/modifyUser

Parámetros: email, name, surname, password

Método: POST

Borrado de usuario:

URL: http://localhost/webservice/deleteUser

Parámetros: email

Método: POST

Registro del id para Google Cloud Services:

Parámetros: email

Método: PUT

Por último, toca hablar de phpMyAdmin.

phpMyAdmin es una herramienta de Software Libre escrita en PHP creada para la manipulación de la base de datos MySQL a través de navegadores de Internet. Soporta una amplia gama de operaciones, siendo efectuadas de modo gráfico a través de una interfaz gráfica, las más frecuentes que realiza el interfaz, son el manejo de bases datos, tablas, campos, relaciones, índices, usuarios, permisos, etc. También proporciona la capacidad de ejecutar directamente sentencias SQL. Entre sus características se encuentran:

- Interfaz sobre web intuitiva.

- Proporciona herramientas de gestión de la base de datos:
- Edición, creación, modificación y eliminación de bases de datos, tablas, vistas, campos, relaciones e índices.
- Mantenimiento de usuarios y sus privilegios.
- Mantenimiento de procedimientos almacenados.
- Importación de datos desde CSV y SQL.
- Exportación a varios formatos: CSV, SQL, XML, PDF, SO/IEC 26300 - OpenDocument Text y Spreadsheet, Word, LATEX y otros.
- Administración de múltiples servidores.
- Creación del despliegue de la base de datos en un gráfico exportado a PDF.
- Creación de consultas complejas haciendo uso QBE (Query By Example) [23].

6.2.3. Aplicación Android

Puesto que uno de los objetivos de este proyecto incluyen la realización de un prototipo de aplicación móvil, se ha elegido al sistema operativo Android como plataforma. La otra candidata sería iOS.

Primeramente, en una asignatura de tercer curso del grado se vió la programación de aplicaciones móviles en el sistema de Apple. Su sistema operativo para móviles llamado iOS me pareció muy intuitivo e interesante de cara al desarrollo de aplicaciones, pero tenía un gran inconveniente: el ecosistema Apple. Es bien sabido que la marca promueve la compatibilidad entre sus dispositivos de una forma muy exclusiva, ya sea entre televisor, ordenador, móvil, etc. Tal es así, que para poder desarrollar una aplicación móvil necesitas un ordenador de la marca. Puesto que no poseo ningún ordenador para desarrollar de la marca de la manzana, se descartó desde primer momento la realización de la aplicación para iOS. También he de reconocer que nunca había desarrollado ninguna aplicación para el sistema operativo Android, por lo que la idea de utilizar este sistema de Google creció.

No obstante, en ambos sistemas se encuentran disponibles las notificaciones push. Las notificaciones push son un sistema de alertar al usuario de que un evento ha ocurrido de forma instantánea. Para el sistema requerido sería fundamental que el

usuario pueda ser notificado al momento de que la cámara de vigilancia ha detectado algo en el ambiente. Para lograr esto en el sistema Android se hace uso del servicio llamado Google Cloud Messaging (GCM).

Google Cloud Messaging (GCM) para Android es un servicio gratuito que nos permite enviar información desde nuestro servidor a un dispositivo Android. El servicio GCM controla todo lo relacionado con el almacenamiento en cola de los mensajes y su entrega a las aplicaciones. Estas son sus características básicas:

- Para que una aplicación reciba mensajes, no es necesario que ésta se encuentre ejecutándose, sino que el sistema la despertará cuando el mensaje llegue.
- GCM pasa la información recibida directamente a la aplicación, la cual tiene control total sobre ella.
- Requiere dispositivos con Android 2.2 o superior que tengan instalada la aplicación de Google Play Store o un emulador con Google APIs.
- Hace uso de una conexión existente a los servicios de Google. Es necesario que los usuarios tengan configurada su cuenta de Google en el dispositivo. Esta cuenta de Google no es un requisito para los dispositivos con Android 4.0.4 o superior.

Aquí se puede ver un esquema de como funcionan estos servicios:

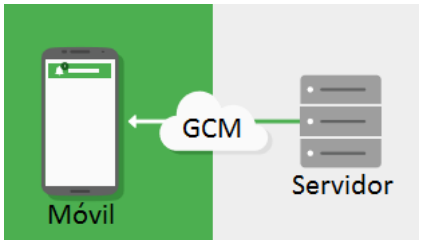


Ilustración 39: Diagrama de funcionamiento de Google Cloud Messaging

6.2.4. Herramientas de desarrollo

En este apartado se van a enumerar las herramientas y librerías empleadas para cada parte del sistema:

Tabla 6: Herramientas de desarrollo empleadas

Sistema	Nombre	Explicación
Raspberry Pi	Raspbian	Raspbian es el sistema operativo basado en Debian(Linux) usado en la placa
	Python3	Es el lenguaje de programación empleado para programar el sensor y la cámara
	NotePad++	Es un editor de múltiples lenguajes. Ha sido el IDE usado para programar en Python
	Cron	Cron es un servicio de Linux que se utiliza para añadir tareas para que puedan ser iniciadas junto con el sistema
Servidor	PHP5	Es el lenguaje empleado para programar el comportamiento del servidor
	Apache	Es el servicio que pone en marcha el servidor
	Slim framework	El framework que facilita el manejo del MVC y los servicios web
	phpMyAdmin	El gestor de la base de datos. Utiliza bases de datos MySQL.
	NotePadqq	Editor (IDE) de PHP en Linux
Android	Android Studio	Es el IDE de desarrollo oficial para Android
	Picaso	Es una librería para cargar imágenes de una URL dada
	Volley	Es una librería para realizar peticiones HTTP al servidor
	GCM	Es la librería empleada para poder utilizar las notificaciones push en Android por medio del Google Cloud Messaging de Google

6.3. Casos de prueba

Una vez implementado todo el sistema, incluyendo el código de la cámara, el del servidor y el de la aplicación móvil, se da paso a las pruebas de funcionamiento de este. Para ello, se ha probado minuciosamente cada apartado de la aplicación móvil, tal y como un usuario final haría con la aplicación.

Iniciar sesión

- **Caso de prueba 1:**
  - **Entrada:** correo electrónico en blanco, contraseña en blanco.
  - **Salida:** el sistema muestra dos mensajes indicando que ambos campos están vacíos y deben rellenarse.
  - **Rol:** Usuario sin identificar.
- **Caso de prueba 2:**
  - **Entrada:** correo electrónico, contraseña en blanco.
  - **Salida:** el sistema muestra dos mensajes indicando que el campo de la contraseña debe rellenarse.
  - **Rol:** Usuario sin identificar.
- **Caso de prueba 3:**
  - **Entrada:** correo electrónico en blanco, contraseña.
  - **Salida:** el sistema muestra dos mensajes indicando que el campo del correo debe rellenarse.
  - **Rol:** Usuario sin identificar.
- **Caso de prueba 4:**
  - **Entrada:** correo electrónico, contraseña.
  - **Salida:** el sistema muestra un mensaje indicando que el usuario o la contraseña son incorrectos.
  - **Rol:** Usuario sin identificar.

- **Caso de prueba 5:**

    - **Entrada:** correo electrónico, contraseña.
    - **Salida:** el sistema identifica correctamente al usuario y este entra en la aplicación.
    - **Rol:** Usuario sin identificar.

Nuevo usuario

- **Caso de prueba 1:**

    - **Entrada:** correo electrónico en blanco, nombre en blanco, apellido en blanco, contraseña en blanco, confirmar contraseña en blanco.
    - **Salida:** el sistema muestra mensajes indicando qué campos están vacíos y deben rellenarse.
    - **Rol:** Usuario sin identificar.

- **Caso de prueba 2:**

    - **Entrada:** correo electrónico en blanco, nombre en blanco, apellido en blanco, contraseña en blanco, confirmar contraseña en blanco.
    - **Salida:** el sistema muestra mensajes indicando qué campos están vacíos y deben rellenarse.
    - **Rol:** Usuario sin identificar.

- **Caso de prueba 3:**

    - **Entrada:** correo electrónico, nombre en blanco, apellido en blanco, contraseña en blanco, confirmar contraseña en blanco.
    - **Salida:** el sistema muestra mensajes indicando qué campos están vacíos y deben rellenarse antes de dar paso al registro.
    - **Rol:** Usuario sin identificar.

- **Caso de prueba 4:**

    - **Entrada:** correo electrónico, nombre, apellido, contraseña, confirmar contraseña.

- **Salida:** el sistema muestra mensajes indicando qué campos están vacíos y deben rellenarse.
- **Rol:** Usuario sin identificar.

- **Caso de prueba 5:**

  - **Entrada:** correo electrónico, nombre, apellido, contraseña, confirmar contraseña.
  - **Salida:** el sistema muestra mensajes indicando que el correo introducido ya existe.
  - **Rol:** Usuario sin identificar.

- **Caso de prueba 6:**

  - **Entrada:** correo electrónico, nombre, apellido, contraseña, confirmar contraseña.
  - **Salida:** el sistema muestra un mensaje indicando que las contraseñas no coinciden.
  - **Rol:** Usuario sin identificar.

- **Caso de prueba 7:**

  - **Entrada:** correo electrónico, nombre, apellido, contraseña, confirmar contraseña.
  - **Salida:** el sistema muestra un mensaje indicando que la contraseña es demasiado corta.
  - **Rol:** Usuario sin identificar.

- **Caso de prueba 8:**

  - **Entrada:** correo electrónico, nombre, apellido, contraseña, confirmar contraseña.
  - **Salida:** el sistema valida el registro con éxito y vuelva a la pantalla de iniciar sesión.
  - **Rol:** Usuario sin identificar.

Mis fotos

- **Caso de prueba 1:**
    - **Entrada:** ninguna.
    - **Salida:** el sistema muestra un mensaje indicando que no hay conexión a Internet en el teléfono.
    - **Rol:** Usuario identificado.
- **Caso de prueba 2:**
    - **Entrada:** ninguna.
    - **Salida:** el sistema muestra las fotos de las cámaras asociadas al usuario.
    - **Rol:** Usuario identificado.

Ver una foto

- **Caso de prueba 1:**
    - **Entrada:** pulsar sobre una foto.
    - **Salida:** el sistema muestra la foto escogida a tamaño completo con información sobre ésta.
    - **Rol:** Usuario identificado.
- **Caso de prueba 2:**
    - **Entrada:** pulsar sobre el menú superior derecho.
    - **Salida:** el sistema muestra el menú con la opción de guardar la foto en el teléfono y bien eliminarla.
    - **Rol:** Usuario identificado.
- **Caso de prueba 3:**
    - **Entrada:** pulsar sobre el botón de eliminar del menú superior derecho.

- **Salida:** el sistema muestra un mensaje indicando que se va a eliminar la foto.
- **Rol:** Usuario identificado.

- **Caso de prueba 3:**

  - **Entrada:** pulsar sobre el botón de guardar del menú superior derecho.
  - **Salida:** el sistema no hace nada.
  - **Rol:** Usuario identificado.

  Corregir poder pedir permisos de almacenamiento para que el usuario pueda guardar la foto

Mis cámaras

- **Caso de prueba 1:**

  - **Entrada:** ninguna.
  - **Salida:** el sistema muestra un mensaje indicando que no hay conexión a Internet en el teléfono.
  - **Rol:** Usuario identificado.

- **Caso de prueba 2:**

  - **Entrada:** ninguna.
  - **Salida:** el sistema muestra un listado con las cámaras del usuario.
  - **Rol:** Usuario identificado.

Añadir cámara

- **Caso de prueba 1:**

  - **Entrada:** la MAC de la cámara en blanco y el nombre que se le va a poner en blanco.
  - **Salida:** el sistema muestra un mensaje indicando qué campos están vacíos y deben rellenarse.

- **Rol:** Usuario identificado.

- **Caso de prueba 2:**

  - **Entrada:** la MAC de la cámara y el nombre que se le va a poner.
  - **Salida:** el sistema muestra un mensaje indicando que la MAC no es válida.
  - **Rol:** Usuario identificado.

- **Caso de prueba 3:**

  - **Entrada:** la MAC de la cámara y el nombre que se le va a poner.
  - **Salida:** el sistema muestra un mensaje indicando que esa MAC ya está en uso y no se puede registrar.
  - **Rol:** Usuario identificado.

- **Caso de prueba 4:**

  - **Entrada:** la MAC de la cámara y el nombre que se le va a poner.
  - **Salida:** el sistema muestra un mensaje indicando que el nombre es demasiado largo.
  - **Rol:** Usuario identificado.

- **Caso de prueba 5:**

  - **Entrada:** la MAC de la cámara y el nombre que se le va a poner.
  - **Salida:** el sistema muestra un mensaje indicando que la cámara se ha registrado con éxito.
  - **Rol:** Usuario identificado.

Eliminar cámara

- **Caso de prueba 1:**

  - **Entrada:** pulsar botón eliminar.
  - **Salida:** el sistema muestra un mensaje indicando que se borrará la cámara elegida y todas las fotos que se han tomado con ella.

- **Rol:** Usuario identificado.

Corregir que el servidor borre todas las fotos de dicha cámara. Se borra la información correctamente de la base de datos, pero no las fotos del sistema de archivos

Modificar cuenta

- **Caso de prueba 1:**
  - **Entrada:** nombre, apellidos, contraseña en blanco y confirmar contraseña en blanco.
  - **Salida:** el sistema muestra un mensaje indicando qué campos se encuentran vacíos.
  - **Rol:** Usuario identificado.
- **Caso de prueba 2:**
  - **Entrada:** nombre, apellidos, contraseña y confirmar contraseña.
  - **Salida:** el sistema modifica los datos correctamente.
  - **Rol:** Usuario identificado.

Corregir que el servidor identifique que las contraseñas coincidan antes de validar la modificación de los datos

Ver acerca de

- **Caso de prueba 1:**
  - **Entrada:** ninguna.
  - **Salida:** el sistema muestra los créditos de la aplicación.
  - **Rol:** Usuario identificado.

Cerrar sesión

- **Caso de prueba 1:**
  - **Entrada:** ninguna.

- **Salida:** el sistema muestra un mensaje indicando que se cerrará la sesión actual.
- **Rol:** Usuario identificado.

7. Conclusiones y futuras mejoras

La realización de todo este Trabajo Fin de Grado ha resultado muy interesante.

Tener un sistema de vigilancia hoy día es muy común dada la cantidad de cámaras IP que se ofrecen por un precio bajo y la utilización de aplicaciones móviles capaces de manejar e interactuar con ellas. Sin embargo, desde un punto de vista de desarrollador informático resulta muy satisfactorio poder seleccionar un hardware de bajo coste que pueda ser programado para que se comporte como se requiera, configurar una base de datos en un servidor remoto o poder hacer una aplicación móvil.

Personalmente me gusta poder realizar un sistema donde se tienen muy en cuenta tanto la parte de software como la de hardware. Es un proyecto donde verdaderamente se reúnen las cualidades de un Ingeniero Informático en la especialidad Tecnologías de la Información.

Ha sido una aventura poner la guinda del pastel al último paso del Grado en Ingeniería Informática aprendiendo y utilizando tecnologías como los servicios web, Android o la programación de Raspberry Pi. Ha quedado demostrado aquello que dicen de que en informática nunca se deja de aprender y, en mi caso, de superarse a uno mismo.

No obstante, nada es perfecto, y tras la realización de todo el proyecto ha habido puntos que por falta de tiempo o no haber planificado correctamente los requisitos se han quedado en el aire. Todo el sistema es un prototipo, pero ya se puede vislumbrar algunas de las mejoras que se pueden realizar en el futuro:

- **Internacionalización**. Es el proceso de diseñar una aplicación de tal manera que pueda adaptarse a diferentes lenguajes y regiones sin necesidad de cambiar el código.

  Un programa internacionalizado no tiene elementos dependientes de la lengua o del contexto cultural de un país o región en el propio código.

Los elementos textuales, como los mensajes o las etiquetas de los componentes, no están en el código. Están fuera y se toman dinámicamente.

Entre sus ventajas:

- El mismo ejecutable funciona en todo el mundo
- El mercado es mayor
- No hace falta hacer un desarrollo internacional del producto una vez acabada la primera versión
- Se utilizan los recursos más eficientemente
- El mantenimiento del código y la inclusión de nuevas localizaciones es menos costoso

Esto se realizaría en la aplicación Android, dado que es la parte del sistema con la que interactúa el usuario. En Android esto se resuelve implementando varios archivos strings.xml (uno por cada idioma a implementar). Dependiendo del idioma en la que esté configurado el terminal que va a ejecutar la aplicación se escoge un archivo u otro.

Para que la internacionalización se ejecute correctamente es necesario que todas las cadenas o strings que aparecen en la aplicación se identifiquen mediante un id único. Así, al referenciarlas en el código se alude a un id específico el cual está siendo implementado en cada archivo strings.xml en un idioma distinto. La aplicación cuando ya esté instalada, al identificar el idioma del teléfono de forma automática escogerá un archivo u otro. Puede parecer un poco tedioso tener que identificar a todas las cadenas que van a aparecer en la aplicación, pero es la única forma de garantizar que en un futuro se puedan añadir más idiomas a parte del original de forma sencilla y rápida.

- **Videovigilancia en tiempo real:** poder visualizar desde la aplicación móvil qué está ocurriendo en cada cámara instalada del usuario es una función muy a tener en cuenta. Muchas cámaras IP del mercado actual lo implementan ya.

  En el caso de este proyecto no resulta muy descabellado pensar que la cámara de la Raspberry se acople a un pequeño motor que la haga girar unos pocos grados a izquierda o derecha. Gracias a la interfaz GPIO de la placa se podría

programar para que el usuario desde la aplicación pueda rotar la visión y observar qué está ocurriendo en ese momento en el lugar que está vigilando.

Además, a esta función se le podría añadir la posibilidad de tomar una foto o grabar o vídeo de corta duración.

Anexo 1: Manual de instalación

A.1. Raspberry Pi

Es el momento de configurar la cámara. Para ello lo primero de todo es conectar el módulo PIR y la cámara como se indica en la imagen:

Ilustración A1 1: Conexión sensor PIR y cámara

Lo siguiente es instalar el sistema operativo Raspbian en la tarjeta SD. Para ello, se descarga su última versión disponible en:

https://www.raspberrypi.org/downloads/raspbian/

Para poder instalarlo en la SD se puede emplear un ordenador con Windows, con Linux o con Mac. En este caso se hará desde un ordenador con Windows.

Para ello necesitamos el programa Win32DiskImager que se descarga aquí:

https://www.raspberrypi.org/documentation/installation/installing-images/windows.md

Una vez descargado se elige el archivo .img con la última versión de Raspbian y la letra donde se ubica la tarjeta SD. Pulsamos sobre Write.

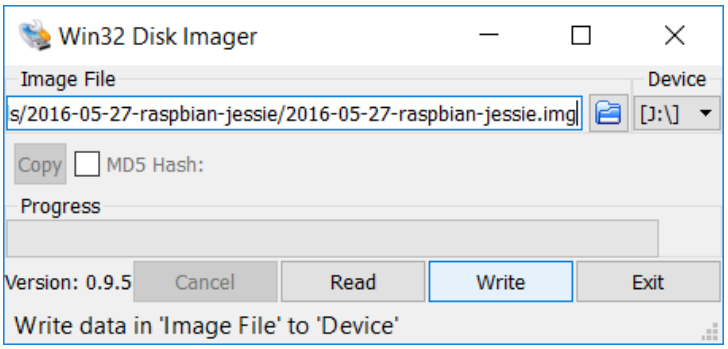


Ilustración A1 2: Instalación de Raspbian

Una vez terminado el proceso se inserta la tarjeta SD y el módulo Wifi. El Wifi es opcional, ya que se puede conectar la Raspberry por medio de un cable Ethernet. También se debe conectar a la corriente eléctrica mediante su cargador:

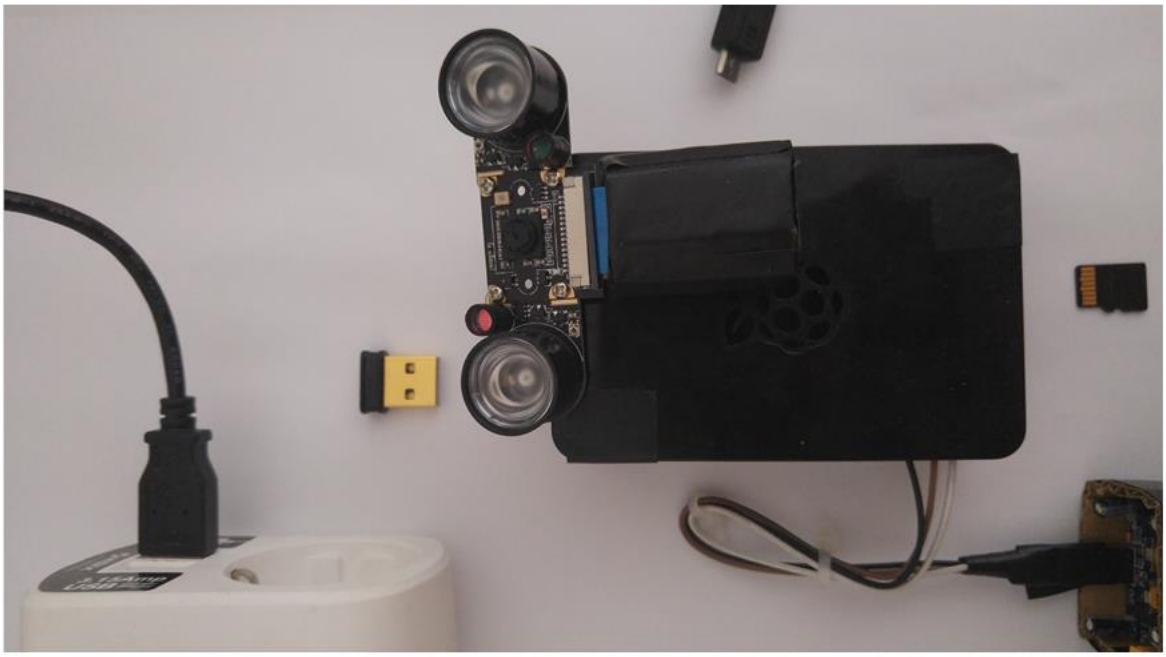


Ilustración A1 3: Conexión de Wifi, SD y alimentación del sistema de la cámara

Para poder interactuar con el sistema operativo es recomendable enchufar un teclado, ratón y pantalla con puerto HDMI a la Raspberry. En la configuración inicial es fundamental habilitar el módulo de la cámara y establecer la hora local para que las fotos tomen los datos de hora y fecha correctos. También se puede configurar más tarde esto desde la interfaz gráfica en Menú -> Preferencias -> Configuración de Raspberry Pi.

Lo siguiente es la instalación de los script para que la cámara se active al detectar movimiento:

1. Desde un pendrive, por ejemplo, se ubica el script “camera.py” en la ruta /home/pi/Documents/camera/. La carpeta “camera” se debe crear manualmente.
2. Crear la carpeta “captures” dentro de Documents. La ruta sería: /home/pi/Documents/captures/
3. Abrir la terminal en Menú -> Accesorios -> LXTerminal
4. Lo siguiente es hacer posible que el sistema arranque el script de la cámara de forma automática cuando se encienda la Raspberry. Para ello se utiliza el servicio llamado “cron” en el que se añaden tareas para ser arrancadas al inicio. Desde la terminal se ejecuta el comando:

   sudo crontab -e

   y se elige la opción 2. Es el editor denominado “nano” para poder editar el fichero

5. Ahora se añade la línea:

   @reboot python3 /home/pi/Documents/camera/camera.py &

   Y se guardan los cambios. Esto ejecuta el programa python3 y el script dado al inicio

6. Para poder controlar el cierre o apertura del proceso que vamos a lanzar se crea un nuevo script dentro de la carpeta init.d. Para ello ejecutamos el comando:

   sudo nano /etc/init.d/camera

   Y se inserta el siguiente código:

```
#!/bin/bash

# /etc/init.d/camera

### BEGIN INIT INFO

# Provides: camera Required-Start: $remote_fs $syslog Required-Stop:

# $remote_fs $syslog Default-Start: 2 3 4 5 Default-Stop: 0 1 6

# Short-Description: Example initscript Description: This service is

# used to manage PIR sensor and camera

### END INIT INFO

case "$1" in

      start)

            echo "Starting camera"

            python /home/pi/Documents/camera/camera.py

            ;;

       stop)

             echo "Stopping camera"

             killall python

             ;;

       *)

              echo "Usage: /etc/init.d/camera start|stop"

              exit 1

              ;;

esac

exit 0
```

Y se guardan los cambios

7. Ahora se deben añadir permisos de ejecución. Para ello:

   sudo chmod +x /etc/init.d/camera

8. Ahora cada vez que esté conectada la Raspberry y queramos arrancar o parar el proceso del script de la cámara basta con ejecutar:

sudo /etc/init.d/camera stop para pararlo

sudo /etc/init.d/camera start para volverlo a arrancar

Para obtener la dirección MAC del módulo Ethernet para poder registrar la cámara desde la aplicación Android se ejecuta el comando:

Ifconfig

La dirección física o MAC que se debe utilizar es la de la interfaz eth0:

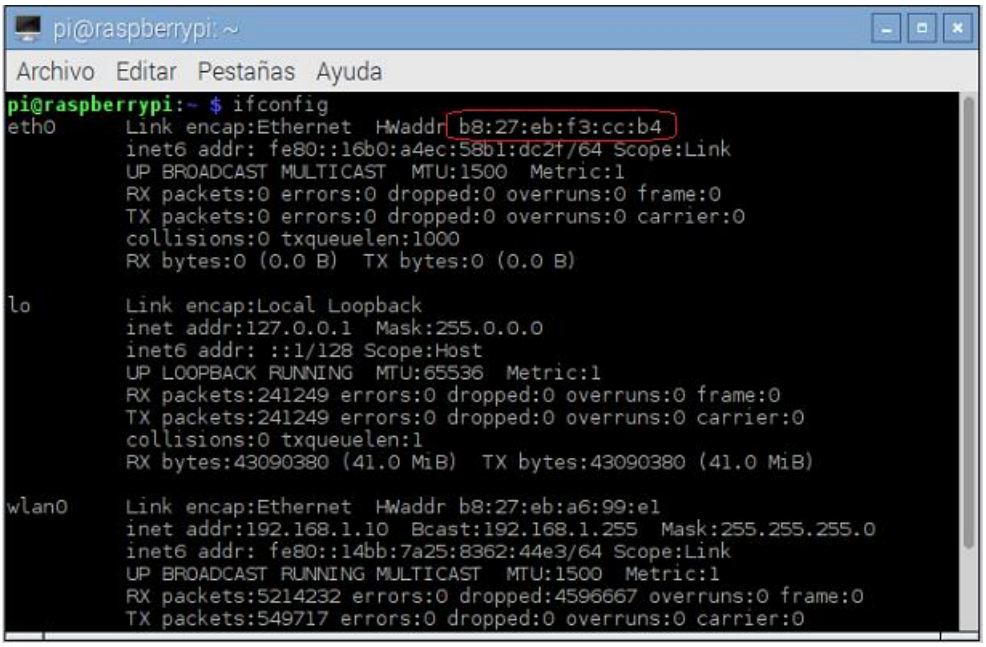


Ilustración A1 4: Obtención de la MAC

En la aplicación Android se debe usar como separador de caracteres el guion “-“ en vez de los puntitos “:”

Por último, si se desea conectar la Raspberry por Wifi es necesario configurarlo:

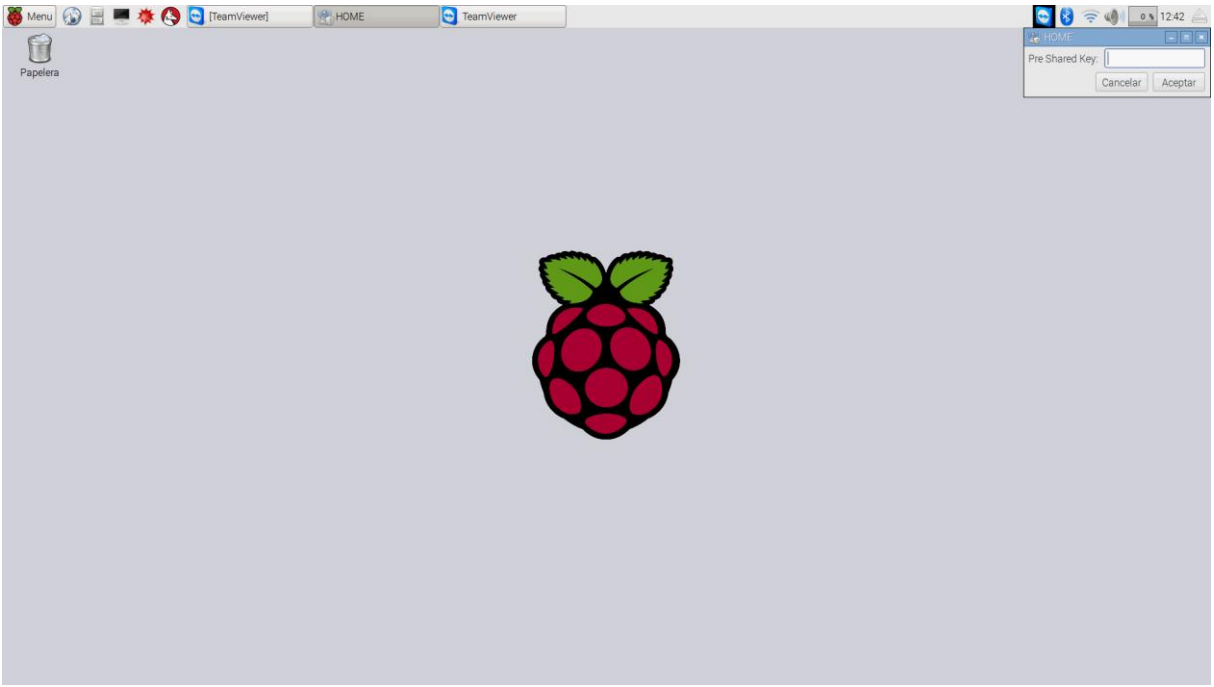


Ilustración A1 5: Configuración del Wifi en Raspbian

Si en cambio se prefiere conectar el cable Ethernet no se debe configurar nada.

A.2. Servidor

El sistema que actúa de servidor para éste proyecto es otra placa Raspberry con el sistema operativo Raspbian también. Es una distribución basada en Debian (Linux).

Lo primero es instalar el servicio Apache. Desde la terminal se comprueban si hay actualizaciones:

sudo apt-get update

sudo apt-get upgrade

Ahora se instala el paquete Apache:

sudo apt-get install apache2 -y

Lo segundo es instalar PHP:

sudo apt-get install php5 libapache2-mod-php5 -y

Lo siguiente es MySQL:

sudo apt-get install mysql-server php5-mysql -y

Por último, es el turno de phpMyAdmin:

sudo apt-get install phpmyadmin

Falta el framework llamado Slim framework que es el encargado de utilizar los servicios web y facilitar el modelo-vista-controlador para la llamada a los métodos que se utilicen por medio de la URL:

1. Copiar el contenido llamado “webservice” adjunto en la ruta del servidor: /var/www/html/
2. Habilitar el módulo rewrite. Desde la terminal:

   sudo A2enmod rewrite

3. Ahora ejecutar

sudo nano /etc/apache2/apache2.conf

4. Añadir al final:

```
<Directory /var/www/html/webservice/>

        Options Indexes FollowSymLinks

        AllowOverride All

         Require all granted

 </Directory>
```

Y se guarda el archivo.

5. Se establece el nombre de la base de datos y los parámetros para poder usar los servicios GCM para las notificaciones push en el archivo /var/www/html/webservice/include/Constants.php:

```
<?php

//Constants to connect with the database

define('DB_USERNAME', 'root');

define('DB_PASSWORD', 'root');

define('DB_HOST', 'localhost');

define('DB_NAME', 'vigilancia');

define("GOOGLE_API_KEY", "AIzaSyDf3V0IF9kGrC7OfvG742qlIr_gtPk12QU");

// push notification flags

define('PUSH_FLAG_RASPVIGILANCIA', 1);

define('PUSH_FLAG_USER', 2);

?>
```

6. Por último, se debe abrir el puerto 80 en el rúter para poder utilizar la IP pública del servidor.

Anexo 2: Manual de usuario

En este manual se detalla qué acciones puede realizar el usuario en la aplicación móvil y cuáles son las respuestas de ésta.

Para poder disfrutar del sistema de vigilancia se necesita una cuenta de usuario. Para ello se pulsa sobre el botón NUEVO USUARIO de la pantalla principal.

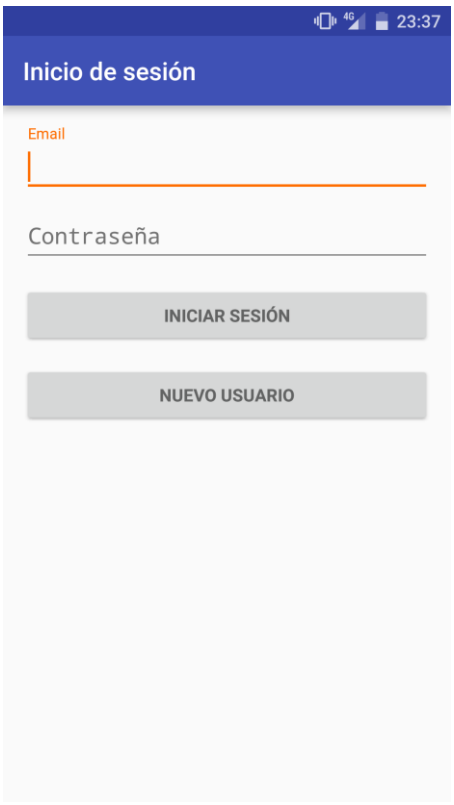


Ilustración A2 1: inicio de sesión

A continuación, se deben rellenar todos los campos para que el usuario quede registrado en el sistema. Cabe destacar que para que el registro se complete con éxito se debe estar conectado a Internet (ya sea por Wifi o por datos). También es importante recalcar que el email introducido no debe estar registrado con anterioridad en la base de datos y que las contraseñas deben coincidir. La aplicación alertará sobre cualquiera de estos posibles casos. Para poder continuar se pulsa el botón REGISTRAR

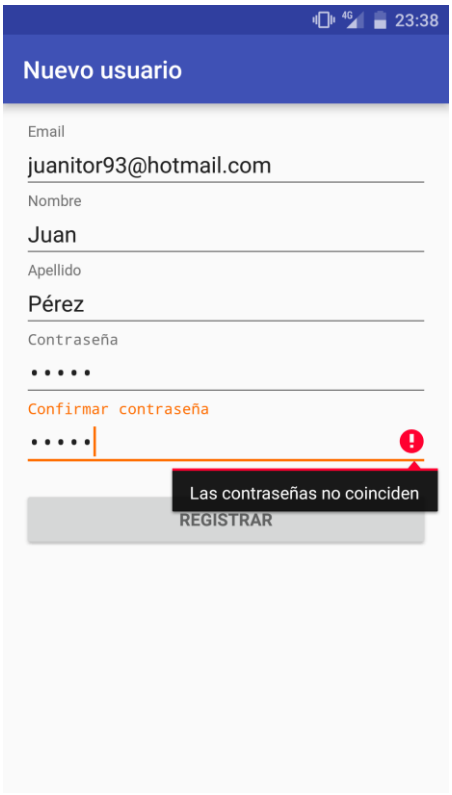


Ilustración A2 2: Nuevo usuario

Una vez completado el registro con éxito es el momento de iniciar sesión. Para ello se pulsa sobre el botón INICIAR SESIÓN

Ilustración A2 3: Inicio de sesión con datos

La pantalla principal muestra las fotos que se obtienen de las cámaras instaladas y registradas de ese usuario. Al principio no habrá ninguna.

Ilustración A2 4: Sin fotos para mostrar

Al pulsar sobre el botón situado en la esquina superior izquierda se puede visualizar el menú principal

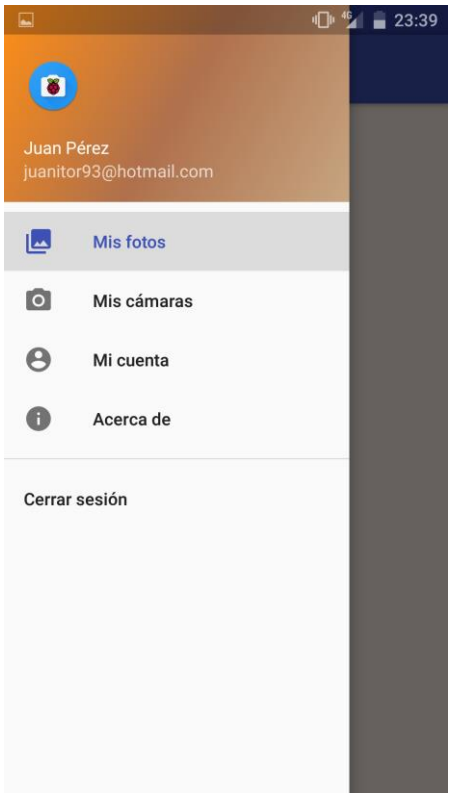


Ilustración A2 5: Menú principal

En el menú se pueden encontrar las principales funciones de la aplicación. Para poder empezar a recibir fotos de la cámara instalada debemos registrarla. Se pulsa sobre la opción Mis cámaras:

Ilustración A2 6: Sin cámaras para mostrar

Es el momento de añadir una cámara. Para ello se pulsa sobre el botón “+”:

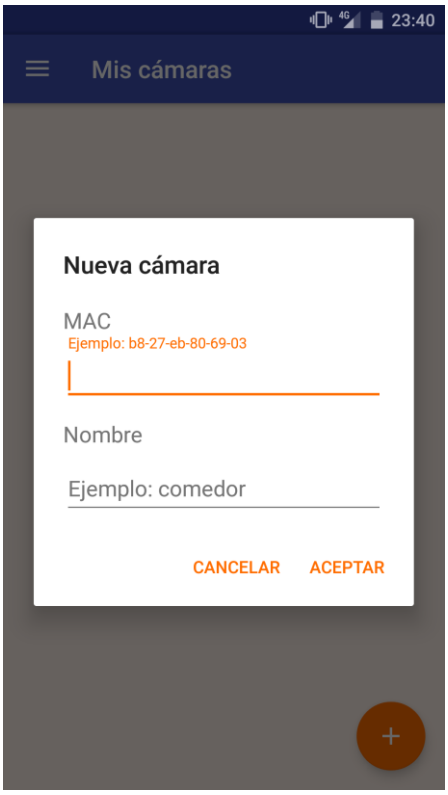


Ilustración A2 7: Nueva cámara

Aquí se debe introducir la dirección MAC del módulo Ethernet de la cámara. Para poder obtener dicha dirección es necesario conectar la placa Raspberry Pi a un monitor, teclado y ratón para ejecutar el comando “ifconfig” desde la terminal. La dirección que se necesita es la que se encuentra en el apartado “eth0”.

Una vez obtenida la dirección ésta se debe rellenar en el campo correspondiente. El sistema validará si la dirección no es correcta o si ésta ya está en uso. Con esta dirección que es única para cada dispositivo de la cámara se garantiza que una cámara pueda ser controlada desde una única cuenta.

Si los campos han sido correctos y se ha pulsado en ACEPTAR, la cámara aparecerá en la lista de las cámaras registradas

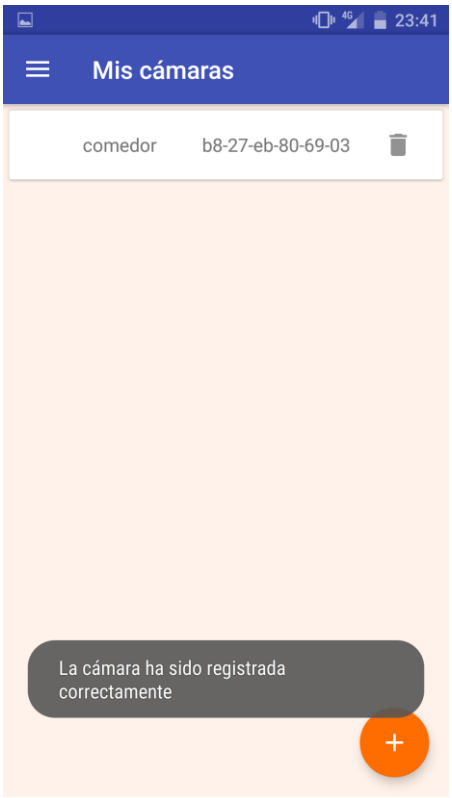


Ilustración A2 8: Cámara registrada correctamente

Se pueden borrar las cámaras registradas que se deseen pulsando sobre el icono de la papelera situado a la derecha de cada una de las cámaras. No obstante, se alertará sobre las consecuencias de su borrado, ya que si se elimina una cámara determinada, las fotos tomadas con dicha cámara y que no estén guardadas en la memoria del teléfono serán eliminadas también

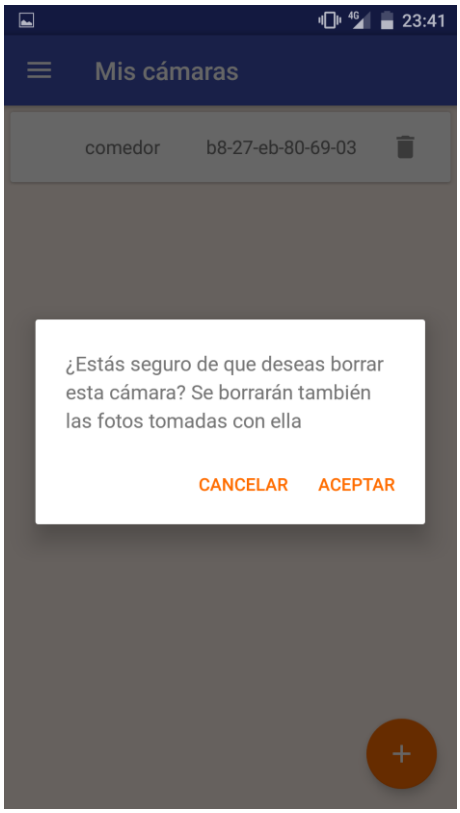


Ilustración A2 9: Borrar cámara

Una vez registrada la cámara instalada en alguna parte del hogar, el usuario es capaz de obtener las fotos que ésta realiza cuando detecta movimiento. Si no se está mirando la aplicación, el sistema envía una notificación al usuario:

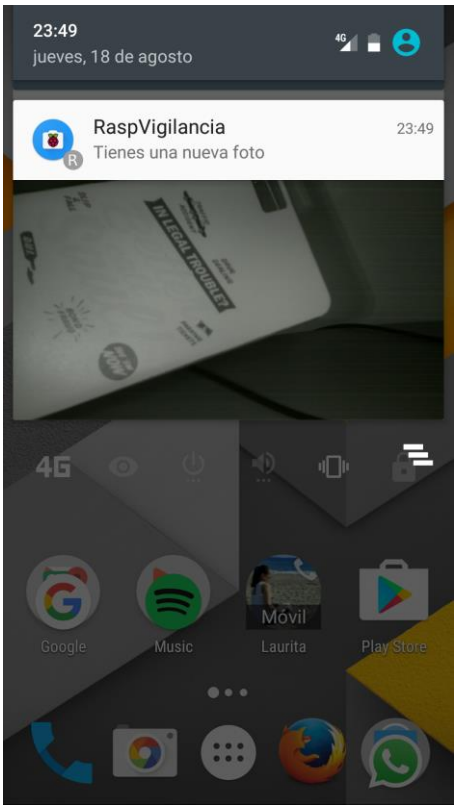


Ilustración A2 10: Notificación

En el caso de que se encuentre el usuario interactuando con la aplicación se añadirán las nuevas fotos a la pantalla principal Mis fotos:

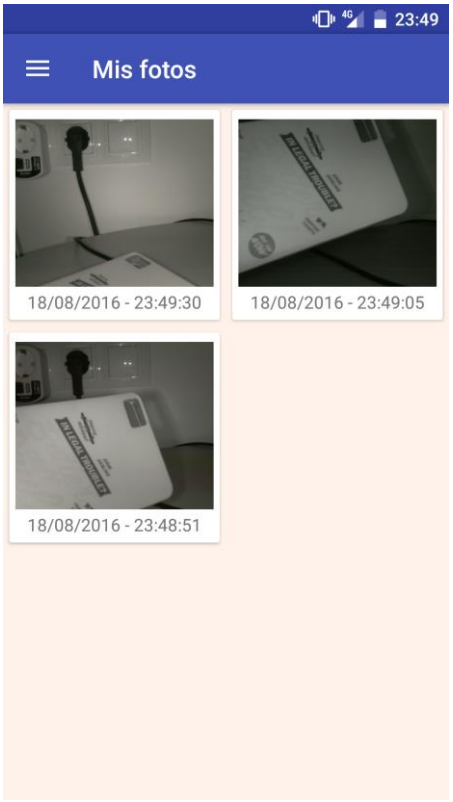


Ilustración A2 11 Mis fotos con fotos

Para poder visualizar mejor una foto, se pulsa sobre ésta:

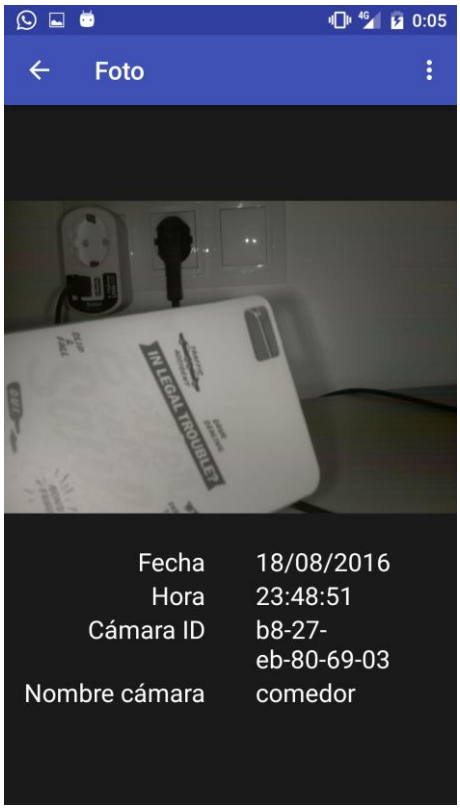


Ilustración A2 12: Una foto seleccionada

Al pulsar sobre el menú superior derecho se da paso a 2 opciones:

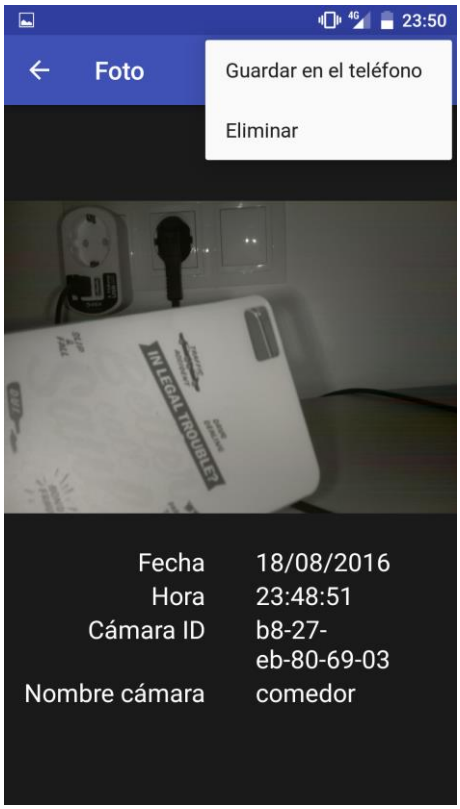


Ilustración A2 13: Menú foto seleccionada

Para poder guardar la foto elegida en la memoria del teléfono se debe aprobar el permiso de escritura en el dispositivo. Esto ocurrirá si el dispositivo móvil maneja Android 6.0 o superior. Esta alerta ocurrirá una única vez siempre que el usuario apruebe el permiso de almacenamiento.

Las fotos se guardan en una carpeta llamada RaspFotos en la raíz del sistema de ficheros del terminal

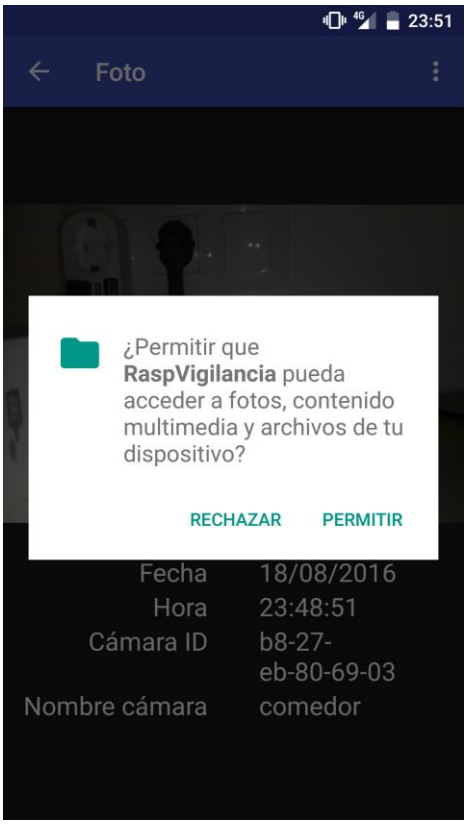


Ilustración A2 14: Permiso para guardar foto

Si se desea eliminar una foto se debe pulsar sobre la opción Eliminar. Este borrado se alerta previamente y conlleva a la eliminación de la foto en la aplicación. La foto no se borrará de la carpeta RaspFotos si ésta se ha guardado ahí con anterioridad.

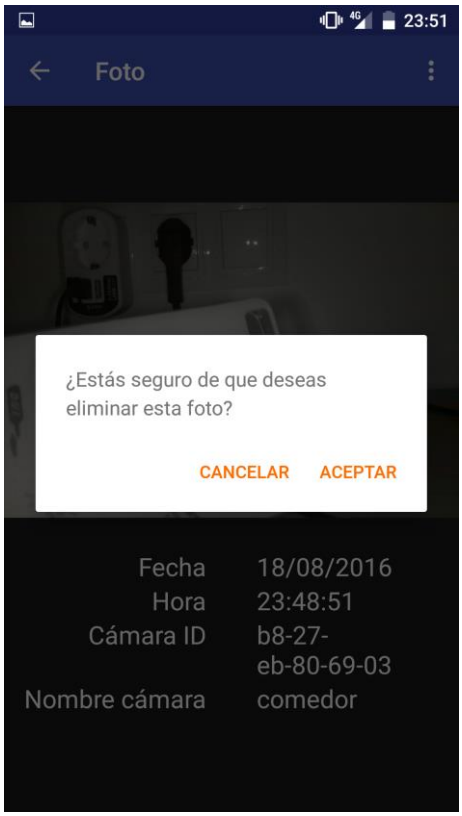


Ilustración A2 15: Eliminar foto

El usuario puede modificar su nombre, apellido y su contraseña si lo desea. Para ello basta con pulsar sobre la opción del menú principal lateral Mi cuenta.

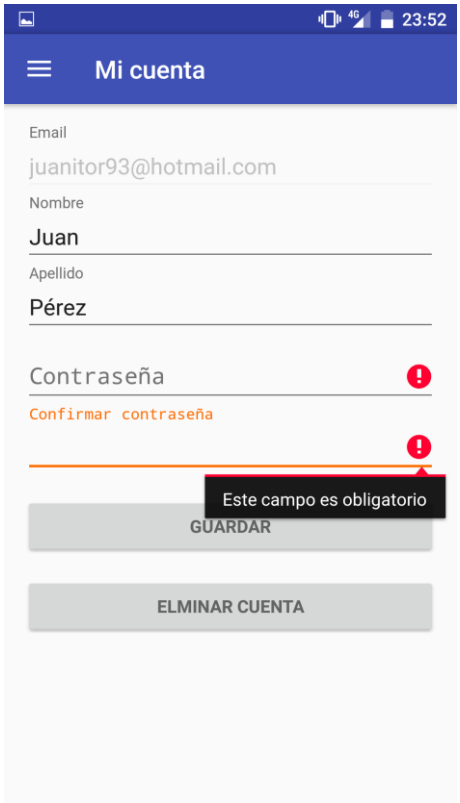


Ilustración A2 16: Modificar datos de la cuenta

Para guardar los datos basta con pulsar sobre el botón GUARDAR.

También es posible eliminar la cuenta y todos los datos que conllevan: cámaras registradas, fotos y demás datos de usuario. Se alerta sobre sus consecuencias:

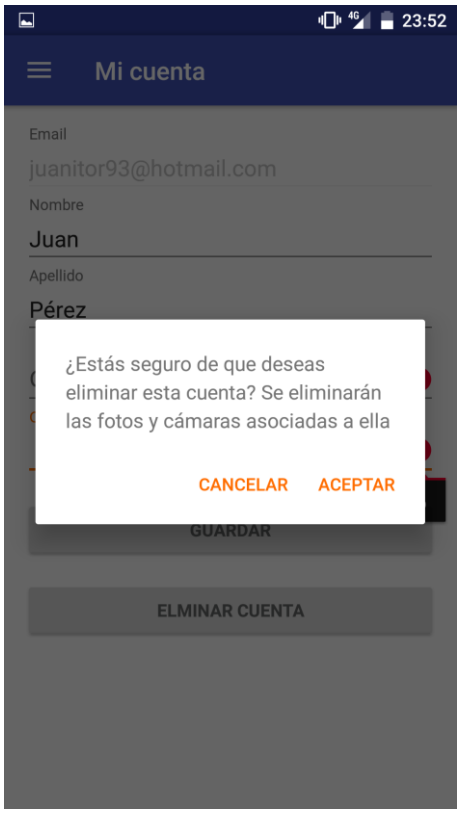


Ilustración A2 17: Eliminar cuenta

Para visualizar los créditos de la aplicación se pulsa sobre el botón Acerca de del menú principal lateral:

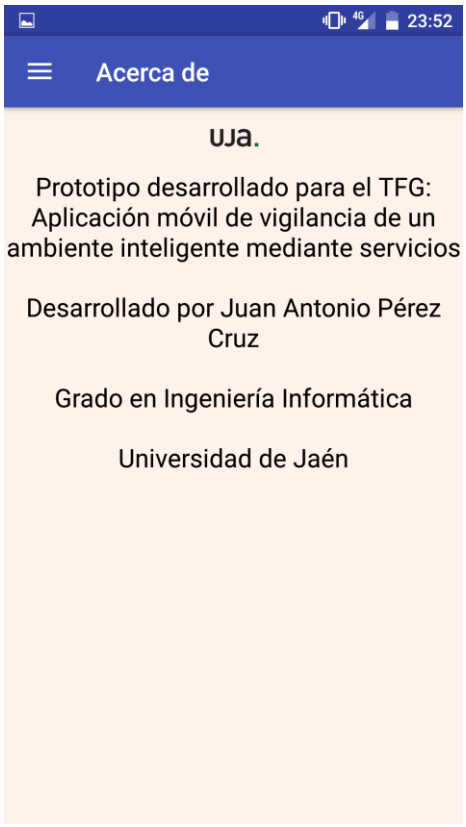


Ilustración A2 18: Acerca de

Por último, se puede iniciar sesión desde otra cuenta. Para ello se pulsa sobre el botón Cerrar sesión del menú principal lateral. Se alerta sobre la acción que se va a realizar:

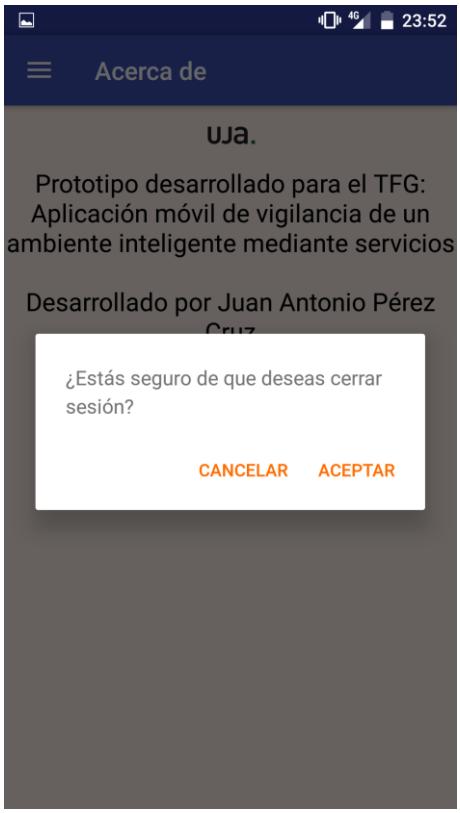


Ilustración A2 19: Cerrar sesión

Bibliografía

[1] Evans, D. (2011). *Internet de las cosas como la próxima evolución de Internet lo cambia todo.*

[2] “Informe Mobile en España y el mundo” Informe Ditrendia. (2015).

[3] “Introducción a los Sistemas de Vigilancia IP.” www.midisec.com. (2008).

[4] “Proteja su hogar y su privacidad”. https://www.getmyfox.com/es/camara-de-vigilancia.html. (2016).

[5] “Get Piper”. www.getpiper.com. (2016).

[6] “Cámaras IP: vigilancia fácil y económica”. www.foscam.es. (2016).

[7] “Todo lo que necesitas saber sobre Arduino”. http://www.xataka.com/especiales/guia-del-arduinomaniaco-todo-lo-que-necesitas-saber-sobre-arduino. (2015).

[8] “Raspberry PI2”. www.raspberrypi.org. (2016).

[9] “SD clase 10”. https://www.sdcard.org/developers/overview/speed_class/. (2016).

[10] Cita de Brooks, F. (1987).

[11] Cita de Boehm. (1975).

[12] Cita DeMarco. (1985).

[13] "Requisito funcional". https://es.wikipedia.org/wiki/Requisito_funcional. (2016).

[14] "Requisito no funcional". https://es.wikipedia.org/wiki/Requisito_no_funcional. (2016). [15] Marset, R. N. (2006). *REST vs Web Services.*

[16] "Diseño de interfaz de usuario". ttps://es.wikipedia.org/wiki/Dise%C3%B1o_de_interfaz_de_usuario. (2016).

[17] "Material Design". https://es.wikipedia.org/wiki/Material_design. (2016).

[18] "Material Design". https://developer.android.com/design/index.html. (2016).

[19] "Material Desgin en español". http://www.materialdoc.es/fab/. (2016).

[20] "Storyboard". https://es.wikipedia.org/wiki/Storyboard. (2016).

[21] "Servidor HTTP Apache". https://es.wikipedia.org/wiki/Servidor_HTTP_Apache. (2016).

[22] "PHP". https://es.wikipedia.org/wiki/PHP. (2016).

[23] "phpMyAdmin". https://es.opensuse.org/PhpMyAdmin. (2011).