UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Gestión desde Android de un simulador de control domótico con Raspberry Pi

Presentado por Mario Juez Castrillo
en Universidad de Burgos — 6 de junio de 2018
Tutor: César Represa Pérez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Para llevar acabo este proyecto he seguido una metodología llamada *Top-down*. En el modelo *Top-down* se formula un resumen del sistema, sin especificar detalles. Cada parte del sistema se refina diseñando con mayor detalle. Cada parte nueva es entonces redefinida, cada vez con mayor detalle, hasta que la especificación completa es lo suficientemente detallada para validar el modelo [5]. Teniendo como base lo anterior, la planificación que se ha seguido ha sido marcar por parte de mi tutor desde un principio los requisitos generales tanto funcionales como no funcionales que debería tener el sistema. Y a partir de esto, yo por mi parte crear requisitos a más bajo nivel según trataba de implementar sus requisitos.

A.2. Planificación temporal

En cuanto a la planificación temporal de este proyecto, no ha sido realizada por semanas, sino de la siguiente manera:

1. El primer día quedé con mi tutor, y el me explicó que se esperaba del proyecto y que debería tener de cara a su exposición. De esta manera elaboramos unos requisitos generales.

2. A partir de estos requisitos generales, siguiendo la metodología *Top-down*, yo mismo cree requisitos más detallados y más pequeños que me llevarían a la implementación de los requisitos generales.

3. Cada vez que se consiguiese un requisito general de los elaborados, se realizaría una reunión en la que se mostraría el contenido en cuanto a código y funcionalidad de dicho requisito.

4. En dicha reunión, se exponen mejoras estéticas o de funcionalidad para dicho requisito, ya que a primera vista son muy generales, y nos enfocamos hacia el siguiente.

Durante la realización del proyecto se han ido siguiendo los pasos anteriores, y no se ha realizado una planificación semanal, ya que cada reunión dependía de cada requisito. Además es posible que un requisito pudiese estar resuelto en una semana, pero otro en dos, o incluso tres semanas, dependiendo de la complejidad.

A partir de los objetivos fijados, como planificación *Top-down* elaboré una serie de tareas más sencillas para ir realizando y conseguir cumplir dichos objetivos. Estas son las tareas que me propuse para cumplir los objetivos y mantener cierta planificación temporal en el proyecto.

- **Definir una habitación y los elementos a controlar.**

  1. Crear una *Activity* o ventana principal que contendrá las estancias [1].
  2. Crear un botón para poder crear las estancias dinámicamente.
  3. Buscar una estructura para almacenar estancias en la que se permita la inserción, modificación y eliminación de ellas.
  4. Implementar la funcionalidad del botón.
  5. Crear una estructura personalizada para adaptarla a nuestras necesidades.
  6. Diferenciar a la hora de la inserción que tipo de estancia estamos insertando mediante una lista de opción única.
  7. Añadir imágenes de las diferentes estancias para que sea más visual al usuario saber que tipo de estancias ha creado.
  8. Añadir un menú contextual en la estructura que permita modificar el nombre de la estancia o su eliminación.

- **Conexión con la estación.**

[1] estancias: posibles lugares de la casa: *Habitación*, *Salón*, *Cocina*, *Baño*

1. Buscar información sobre como realizar la conexión.
2. Implementar una conexión saliente mediante *Sockets*.
3. Implementar dicha conexión en una tarea asíncrona para mejorar su funcionalidad.
4. Añadir dicha funcionalidad cuando se creen nuevas estancias o se modifiquen las ya existentes de alguna manera.

- **Incremento de la funcionalidad de la aplicación.**

1. Crear una *Activity* o ventana secundaria donde se representará la iluminación.
2. Crear un botón para poder crear las bombillas dinámicamente.
3. Buscar una estructura para almacenar dichas bombillas en la que se permite la inserción, modificación y eliminación de ellas.
4. Implementar la funcionalidad del botón.
5. Crear una estructura personalizada para adaptarla a las necesidades de la bombilla.
6. Implementar la funcionalidad del interruptor para cambiar el estado de la bombilla.
7. Añadir un menú contextual en la estructura que contiene las bombillas que permite modificar el nombre de ellas y su eliminación.
8. Permitir que se realice una conexión a la estación con la creación de nuevas bombillas o que se modifiquen las ya existentes de alguna manera.
9. Añadir flecha de retroceso que nos devuelva a la ventana principal.
10. Permitir que al pulsar sobre una estancia nos dirija a su ventana secundaria con su iluminación correspondiente.

- **Desarrollo de la interfaz en la Raspberry Pi.**

1. Crear una ventana básica vacía.
2. Buscar una estructura en la que poder almacenar de forma visual las estancias y su iluminación.
3. Crear la estructura donde se almacenarán.
4. Añadir funcionalidad a la estructura para que permita la modificación y eliminación de las estancias y su iluminación.
5. Crear el servidor socket para escuchar los mensajes de la aplicación.

6. Crear un método que interprete los mensajes que llegan de la app.
7. Separar el servidor socket en un hilo independiente.
8. Añadir texto e imágenes en la parte de iluminación.
9. Añadir iconos a la estancias para diferencias cada tipo.
10. Añadir persistencia de los elementos.
11. Añadir la introducción de la IP del servidor de forma gráfica.
12. Modificar el servidor para que permita la conexión de más de un usuario simultáneamente.

- **Mejora de utilización de la aplicación.**

1. Crear una *Activity* o ventana donde se representará la *ip*, el *puerto* y el *estado* del servidor.
2. Permitir la conexión y desconexión del servidor desde esta ventana mediante un botón.
3. Mantener la persistencia de los elementos de la aplicación mediante una base de datos local.
4. Permitir borrar la base de datos desde el menú desplegable de la ventana principal.
5. Permitir acceder a la ventana del Servidor desde el menú desplegable de la ventana principal.
6. No permitir realizar cambios de ningún tipo sin estar conectado al servidor.
7. Crear un hilo independiente que escuche los cambios que nos manda el servidor.
8. Actualizar la vista con cada cambio que nos llegue del servidor.
9. Actualizar la base de datos local con la base de datos del servidor cada vez que nos conectemos a él.

- **Lanzamiento de la primera versión completamente funcional.**

1. Realizar pruebas manuales de todos los aspectos de la aplicación.
2. Poner a prueba que su funcionamiento es correcto con 3 usuarios conectados a la vez.
3. Realizar test instrumentales de la base de datos en la parte de la aplicación.

A.3. Estudio de viabilidad

Para el estudio de la viabilidad he decidido crear un análisis *DAFO*, que es una herramienta de estudio de la situación de una empresa, institución, proyecto o persona, analizando sus características internas (Debilidades y Fortalezas) y su situación externa (Amenazas y Oportunidades) en una matriz cuadrada [4].

Debilidades

- Es un simulador de una vivienda inteligente, no está llevado a la realidad.
- No es posible que funcione si la Raspberry y el terminal Android están conectado a una red diferente.
- Solo hemos tratado el tema de la iluminación de la vivienda, quedan muchos aspectos de ella sin tratar.

Fortalezas

- Código Python reutilizable en cualquier otro sistema operativo, ya que python es un lenguaje multiplataforma.
- Permite el uso de varios usuarios simultáneamente.
- El hardware utilizado es barato.

Amenazas

- Ya existen viviendas inteligentes en la realidad.
- Actualmente, empresas como Xiaomi están sacando bombillas inteligentes que suprimirían el intermediario del servidor. La conexión sería directa con la bombilla porque contiene su propia conexión a Internet.

Oportunidades

- El mercado de la domótica aún está comenzando y tal vez no sea tan complicado hacerse un hueco como en otros mercados.

- Están sacando al mercado placas con mayor potencia que la Raspberry Pi, como por ejemplo, la **ODROID-XU4** [1].
- Podemos mejorar la aplicación añadiendo más elementos de la vivienda que podemos controlar y centralizarlos todos desde la misma aplicación.

Viabilidad económica

En cuanto a la viabilidad económica vamos a diferencia entre dos aspectos:

Coste Personal

Este proyecto ha sido desarrollado por un único desarrollador realizando aproximadamente 20 horas semanales de trabajo. Suponiendo que el sueldo de un desarrollador a tiempo parcial es de 15 euros por hora.

```
4 semanas/mes x 15 euros/hora = 1200 euros por mes
```

Teniendo en cuenta que el proyecto ha sido realizado durante los meses *Marzo*, *Abril* y *Mayo*:

```
3 meses x 1200 euros/mes = 3600 euros
```

El coste total a pagar al desarrollador del proyecto serían 3600 €.

Para el cálculo del salario del desarrollador no se ha tenido en cuenta ningún tipo de retención porque el proyecto no ha sido realizado en una empresa. En cambio, debemos añadir costes de vivienda como son la luz y el internet. El gasto de vivienda al residir en un piso compartido de tres personas se reduce a la tercera parte.

El gasto de luz es:

```
3 meses x (60 euros/mes / 3 personas) = 60 euros
```

El gasto de internet es:

```
3 meses x (35 euros/mes / 3 personas) = 35 euros
```

El gasto total del coste personal siguiendo la tabla A.1 es:

```
Sueldo desarrollador + internet + luz = 3600 + 35 + 60 = 3695 euros
```

Nombre	Precio (€)
Luz	60
Internet	35
Desarrollador	3600

Tabla A.1: Coste personal.

Coste Hardware

Para este proyecto, no he necesitado comprar ningún hardware adicional porque ya disponía de él. Por ello mostraré un precio aproximado del coste de estos elementos.

El coste total del hardware siguiendo la tabla A.2 es 1083 €.

Hardware	Precio (€)
Raspberry Pi	35
Tarjeta Micro SD 64 GB Sandisk	22
Rii Mini i8 (Teclado + Ratón)	14
Ordenador Portátil	1000
Enchufe + Cable USB-Micro USB	7
Carcasa Raspberry Pi	5

Tabla A.2: Coste aproximado del Hardware.

El coste total del proyecto ha sido:

```
Coste personal + Coste Hardware = 3695 + 1083 = 4778 euros
```

Viabilidad legal

Las herramientas utilizadas en este proyecto son *JetBrains Pycharm* y *Android Studio*.
Android Studio está basado en el software *IntelliJ IDEA* de **JetBrains** y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0.

JetBrains PyCharm Community Edition ha sido publicado gratuitamente a través de la licencia Apache, aunque en este proyecto he usado **Jetbrains PyCharm Professional Edition** a través de una licencia de

estudiante de manera gratuita también.
No ha sido necesario dar crédito o pagar por ninguna de las librerías utilizadas en este proyecto.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado vamos a hablar sobre la planificación seguida en este trabajo, sus objetivos generales y sus requisitos.

B.2. Objetivos generales

El objetivo principal del trabajo era trabajar sobre la idea de la domótica y la posibilidad de manejar tu casa desde tu teléfono. Partiendo de esa base y realizando una estrategia *Top-down*, entre mi tutor y yo realizamos unos objetos generales que este proyecto debería cumplir.

1. Definir una habitación y los elementos a controlar.
2. Conexión con la estación.
3. Incremento de la funcionalidad de la aplicación.
4. Desarrollo de la interfaz en la Raspberry Pi.
5. Mejora de utilización de la aplicación.
6. Lanzamiento de la primera versión completamente funcional.
7. Documentación: Escribir memoria y anexos.
8. Diseñar el póster de la presentación.

B.3. Catalogo de requisitos

A partir de dichos objetivos y entrando más en detalle, se han obtenido los requisitos funcionales más generales según las ventanas con las que puede interactuar el usuario en la aplicación y en la Raspberry Pi.

Requisitos funcionales para la aplicación

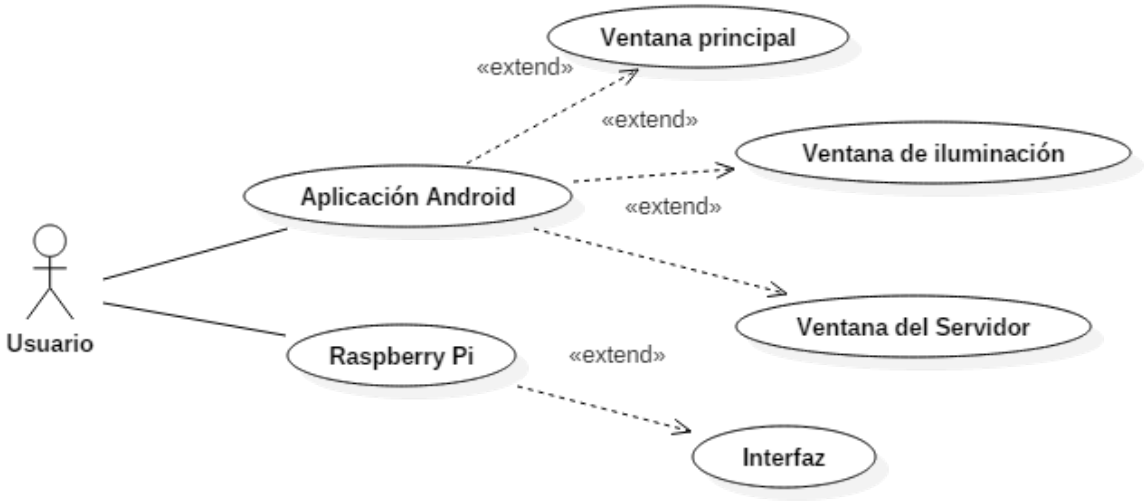

Figura B.1: Diagrama de casos de uso general.

Ventana principal

1. Crear estancia.
2. Modificar nombre.
3. Eliminar estancia.
4. Borrar base de datos.
5. Ir a la ventana del Servidor.

Ventana de Iluminación

1. Crear bombilla.
2. Modificar nombre.
3. Eliminar bombilla.
4. Encender bombilla.
5. Apagar bombilla.
6. Volver a la ventana principal.

Ventana del Servidor

1. Modificar IP
2. Modificar Puerto.
3. Conectar al servidor.
4. Desconectar del servidor.
5. Volver a la ventana principal.

Interfaz Raspberry

1. Actualizar IP.

B.4. Especificación de requisitos

En este apartado, una vez hemos visto el diagrama de caso de uso general B.1, vamos a desglosarlo en diagramas de caso de uso más pequeños, B.2, B.3, B.4 y B.5 que además contienen los requisitos funcionales nombrados en el punto anterior. Además, en las tablas B.1, B.2, B.3, B.4, B.5, B.6, B.7, B.8, B.9, B.10, B.11, B.12, B.13, B.14, B.15, B.16, B.17 se puede ver explicado cada caso de uso con más detalle.

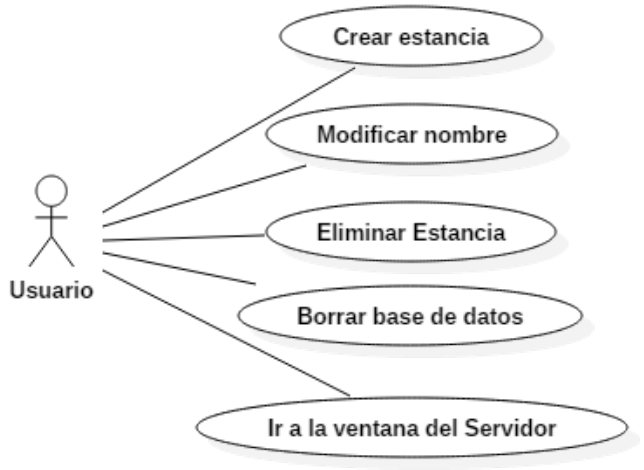

Figura B.2: Funciones que puede realizar el usuario en la ventana principal.

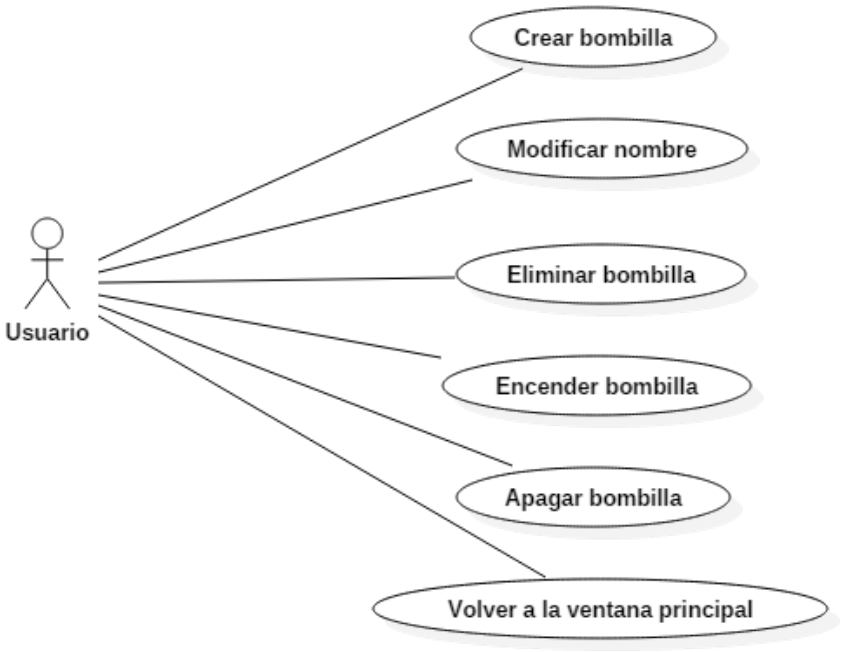

Figura B.3: Funciones que puede realizar el usuario en la ventana de iluminación.

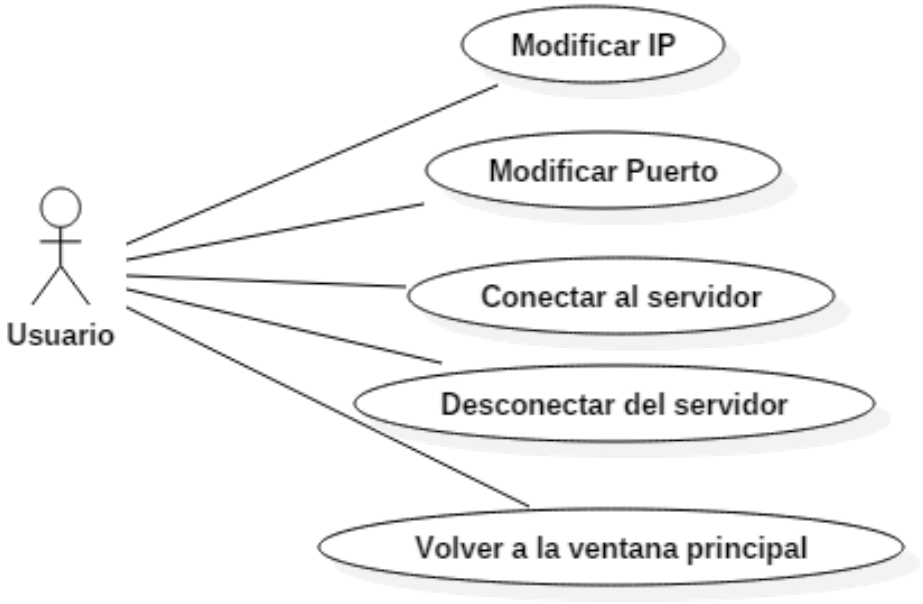

Figura B.4: Funciones que puede realizar el usuario en la Ventana del Servidor.

Figura B.5: Funciones que puede realizar el usuario en la interfaz Python.

Caso de uso ventana principal: *Crear estancia*.		
Descripción	Permite al usuario crear una estancia nueva.	
Requisitos	RF-1	
Precondiciones	El usuario debe encontrarse en la ventana principal de la aplicación. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa el botón con símbolo + que se encuentra en la parte superior derecha.
	2	Marca una opción de la lista de estancias en la ventana emergente.
	3	Elige un nombre para esta estancia.
	4	Pulsa sobre el botón *Aceptar* de la ventana emergente.
Postcondiciones	Se ha creado una nueva estancia de ese tipo en la ventana principal.	
Excepciones	Tratar de crear una estancia sin seleccionar un tipo o escribir un nombre.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.1: Caso de uso ventana principal: *Crear estancia*.

Caso de uso ventana principal: *Modificar nombre.*		
Descripción	Permite modificar el nombre de una estancia ya existente.	
Requisitos	RF-2	
Precondiciones	El usuario debe encontrarse en la ventana principal de la aplicación. Debe existir al menos una estancia para poder modificar su nombre. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Mantener pulsado brevemente sobre la estancia hasta que se muestre el menú contextual.
	2	Seleccionar la opción *Cambiar nombre* del menú contextual.
	3	Escribir el nuevo nombre en la ventana emergente que se nos muestra.
	4	Pulsar sobre el botón *Aceptar* de la ventana emergente.
Postcondiciones	El nombre de la estancia sobre la que estábamos trabajando se ha actualizado.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.2: Caso de uso ventana principal: *Modificar nombre.*

Caso de uso ventana principal: *Eliminar estancia*.		
Descripción	Permite eliminar una estancia ya existente.	
Requisitos	RF-3	
Precondiciones	El usuario debe encontrarse en la ventana principal de la aplicación. Debe existir al menos una estancia para poder eliminar. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Mantener pulsado brevemente sobre la estancia hasta que se muestre el menú contextual.
	2	Seleccionar la opción *Borrar* del menú contextual.
	3	Pulsar sobre el botón *Aceptar* de la ventana emergente de confirmación.
Postcondiciones	La estancia debe desaparecer de la vista actual	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.3: Caso de uso ventana principal: *Eliminar estancia*.

Caso de uso ventana principal: *Borrar base de datos*.		
Descripción	Permite borrar tanto la base de datos local como la del servidor.	
Requisitos	RF-4	
Precondiciones	El usuario debe encontrarse en la ventana principal de la aplicación. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Pulsar sobre las tres líneas blancas horizontales de la parte superior izquierda para mostrar el menú desplegable.
	2	Seleccionar la opción *Borrar base de datos* del menú desplegable.
	3	Pulsar sobre el botón *Aceptar* de la ventana emergente de confirmación.
Postcondiciones	Debe borrarse la base de datos local y del servidor. Deben desaparecer todas las estancias que hubiese en la vista.	
Excepciones		
Importancia	Media	
Urgencia	Media	

Tabla B.4: Caso de uso ventana principal: *Borrar base de datos*.

Caso de uso ventana principal: *Ir a la ventana del Servidor*.		
Descripción	Permite cambiar desde la ventana principal a la ventana del Servidor.	
Requisitos	RF-5	
Precondiciones	El usuario debe encontrarse en la ventana principal de la aplicación.	
Secuencia normal	Paso	Acción
	1	Pulsar sobre las tres líneas blancas horizontales de la parte superior izquierda para mostrar el menú desplegable.
	2	Seleccionar la opción *Servidor* del menú desplegable.
Postcondiciones	Deberá encontrarse en la ventana del Servidor	
Excepciones		
Importancia	Alta	
Urgencia	Media	

Tabla B.5: Caso de uso ventana principal: *Ir a la ventana del Servidor*.

Caso de uso ventana de iluminación: *Crear bombilla.*		
Descripción	Permite al usuario crear una bombilla nueva.	
Requisitos	RF-1	
Precondiciones	El usuario debe encontrarse en la ventana de iluminación de la aplicación. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa el botón con símbolo + que se encuentra en la parte superior derecha.
	2	Marca la única opción existente en la ventana emergente.
	3	Elige un nombre para esta bombilla.
	4	Pulsa sobre el botón *Aceptar* de la ventana emergente.
Postcondiciones	Se ha creado una nueva bombilla en la ventana de iluminación.	
Excepciones	Tratar de crear una bombilla sin seleccionarla en la lista o escribir un nombre.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.6: Caso de uso ventana de iluminación: *Crear bombilla.*

Caso de uso ventana de iluminación: *Modificar nombre*.		
Descripción	Permite modificar el nombre de una bombilla ya existente.	
Requisitos	RF-2	
Precondiciones	El usuario debe encontrarse en la ventana de iluminación de la aplicación. Debe existir al menos una bombilla para poder modificar su nombre. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Mantener pulsado brevemente sobre la bombilla hasta que se muestre el menú contextual.
	2	Seleccionar la opción *Cambiar nombre* del menú contextual.
	3	Escribir el nuevo nombre en la ventana emergente que se nos muestra.
	4	Pulsar sobre el botón *Aceptar* de la ventana emergente.
Postcondiciones	El nombre de la bombilla sobre la que estábamos trabajando se ha actualizado.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.7: Caso de uso ventana de iluminación: *Modificar nombre*.

Caso de uso ventana de iluminación: *Eliminar bombilla.*		
Descripción	Permite eliminar una bombilla ya existente.	
Requisitos	RF-3	
Precondiciones	El usuario debe encontrarse en la ventana de iluminación de la aplicación. Debe existir al menos una bombilla para poder eliminar. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Mantener pulsado brevemente sobre la bombilla hasta que se muestre el menú contextual.
	2	Seleccionar la opción *Borrar* del menú contextual.
	3	Pulsar sobre el botón *Aceptar* de la ventana emergente de confirmación.
Postcondiciones	La bombilla debe desaparecer de la vista actual.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.8: Caso de uso ventana de iluminación: *Eliminar bombilla.*

Caso de uso ventana de iluminación: *Encender bombilla.*		
Descripción	Permite cambiar el estado de la bombilla a *Encendida*	
Requisitos	RF-4	
Precondiciones	El usuario debe encontrarse en la ventana de iluminación de la aplicación. Debe existir al menos una bombilla para poder encenderla. El interruptor deben estar apagado. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Pulsar el interruptor de la bombilla para cambiar su estado y encenderla.
Postcondiciones	La bombilla debe encenderse, cambiando su imagen y su estado.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.9: Caso de uso ventana de iluminación: *Encender bombilla.*

Caso de uso ventana de iluminación: *Apagar bombilla.*		
Descripción	Permite cambiar el estado de la bombilla a *Apagada*	
Requisitos	RF-5	
Precondiciones	El usuario debe encontrarse en la ventana de iluminación de la aplicación. Debe existir al menos una bombilla para poder apagarla. El interruptor deben estar encendido. Debe estar conectado con el servidor.	
Secuencia normal	Paso	Acción
	1	Pulsar el interruptor de la bombilla para cambiar su estado y apagarla.
Postcondiciones	La bombilla debe apagarse, cambiando su imagen y su estado.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.10: Caso de uso ventana de iluminación: *Apagar bombilla.*

Caso de uso ventana de iluminación: *Volver a la ventana principal.*		
Descripción	Permite ir desde la ventana de iluminación a la ventana principal.	
Requisitos	RF-6	
Precondiciones	El usuario debe encontrarse en la ventana de iluminación de la aplicación.	
Secuencia normal	Paso	Acción
	1	Pulsar sobre la flecha que se encuentra en la parte superior izquierda.
Postcondiciones	Nos encontramos en la ventana principal.	
Excepciones		
Importancia	Media	
Urgencia	Media	

Tabla B.11: Caso de uso ventana de iluminación: *Volver a la ventana principal.*

Caso de uso ventana del Servidor: *Modificar IP*.		
Descripción	Actualiza el valor asignado a la IP del servidor.	
Requisitos	RF-1	
Precondiciones	El usuario debe encontrarse en la ventana de servidor de la aplicación.	
Secuencia normal	Paso	Acción
	1	Pulsar sobre el campo de texto de la IP.
	2	Escribir la nueva IP que se quiera utilizar.
	3	Pulsar sobre el botón *Actualizar*.
Postcondiciones	La IP ha sido guardada en la base de datos.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.12: Caso de uso ventana del Servidor: *Modificar IP*.

Caso de uso ventana del Servidor: *Modificar Puerto*.		
Descripción	Actualiza el valor asignado a la IP del servidor.	
Requisitos	RF-2	
Precondiciones	El usuario debe encontrarse en la ventana de servidor de la aplicación.	
Secuencia normal	Paso	Acción
	1	Pulsar sobre el campo de texto del Puerto.
	2	Escribir el nuevo Puerto que se quiera utilizar.
	3	Pulsar sobre el botón *Actualizar*.
Postcondiciones	El Puerto se guarda en la base de datos.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.13: Caso de uso ventana del Servidor: *Modificar Puerto*.

Caso de uso ventana del Servidor: *Conectar al servidor*.	
Descripción	Crea una conexión con el Servidor.
Requisitos	RF-3
Precondiciones	El usuario debe encontrarse en la ventana de servidor de la aplicación. El campo IP debe contener una IP válida. El campo Puerto debe contener un puerto válido.
Secuencia normal	Paso Acción 1 Pulsar el botón *Conectar*.
Postcondiciones	El estado del servidor pasa a *Conectado*.
Excepciones	
Importancia	Alta
Urgencia	Alta

Tabla B.14: Caso de uso ventana del Servidor: *Conectar al servidor*.

Caso de uso ventana de servidor: *Desconectar del servidor*.	
Descripción	Cierra una conexión existente con el Servidor.
Requisitos	RF-4
Precondiciones	El usuario debe encontrarse en la ventana del Servidor de la aplicación. Debe existir una conexión con el Servidor actualmente.
Secuencia normal	Paso Acción 1 Pulsar el botón *Desconectar*.
Postcondiciones	El estado del servidor pasa a *Desconectado*.
Excepciones	
Importancia	Alta
Urgencia	Alta

Tabla B.15: Caso de uso ventana del Servidor: *Desconectar del servidor*.

Caso de uso ventana del Servidor: *Volver a la ventana principal.*		
Descripción	Permite ir desde la ventana del Servidor a la ventana principal.	
Requisitos	RF-5	
Precondiciones	El usuario debe encontrarse en la ventana del Servidor de la aplicación.	
Secuencia normal	Paso	Acción
	1	Pulsar sobre la flecha que se encuentra en la parte superior izquierda.
Postcondiciones	Nos encontramos en la ventana principal	
Excepciones		
Importancia	Media	
Urgencia	Media	

Tabla B.16: Caso de uso ventana del Servidor: *Volver a la ventana principal.*

Caso de uso Interfaz Python: *Actualizar IP.*		
Descripción	Permite introducir una IP nueva.	
Requisitos	RF-1	
Precondiciones	El usuario debe encontrarse en la Interfaz Python del Servidor tras arrancar el Servidor. La IP debe estar vacía o incorrecta.	
Secuencia normal	Paso	Acción
	1	En el error que se muestre, pulsar el botón *Aceptar.*
	2	En la nueva ventana que se abrirá, escribir la nueva IP en el cuadro de texto correspondiente.
	3	Pulsar el botón *Actualizar.*
Postcondiciones	El socket estará escuchando correctamente en esa IP.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.17: Caso de uso Interfaz Python: *Actualizar IP.*

Apéndice C

Especificación de diseño

C.1. Introducción

En este apéndice vamos a hablar sobre los diferentes aspectos de diseño.

C.2. Diseño de datos

En este proyecto se trabaja con muchos datos, pero la mayoría de ellos son del mismo tipo. Se trabaja con gran cantidad de Strings o cadenas de caracteres. La lógica de trabajar con cadenas de caracteres frente a otros tipos de datos radica en la facilidad para ser leídos a través de un *buffer* o incluso de ser tratados. Además, hay que tener en cuenta que ciertos tipos de datos no pueden ser almacenados o enviados. Por ejemplo, en nuestra base de datos querríamos almacenar el estado de las luces mediante una variable de tipo `boolean`, pero nuestra base de datos **Sqlite** no permite alamcenarlos. En su defecto, el estado de las luces ha tenido que ser guardado mediante una variable de tipo `Integer` que podrá contener los valores *0* o *1*.

En nuestra base de datos, ya sea en la parte de Raspberry Pi o de Android, solo almacenaremos cadenas de caracteres (`Strings`) o números (`Integer`). A través de la conexión entre clientes y servidor solo se manda cadenas de caracteres a través de un *buffer* para realizar cambios en cualquiera de las partes. Solamente se ha hablado de estos dos tipos de datos porque son los dos únicos tipos que o se almacenan para su tratamiento o son tratados en el momento.

C.3. Diseño procedimental

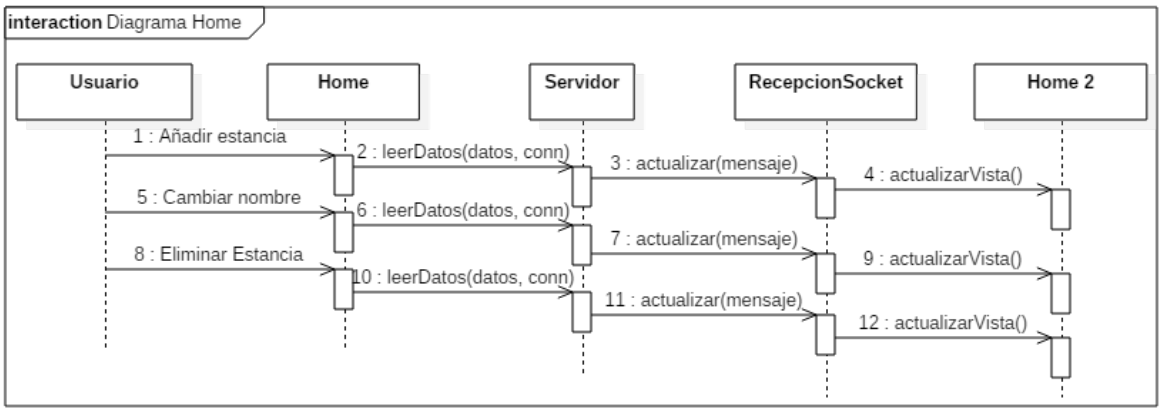

Figura C.1: Diagrama de acciones sobre una estancia en la clase **Home**.

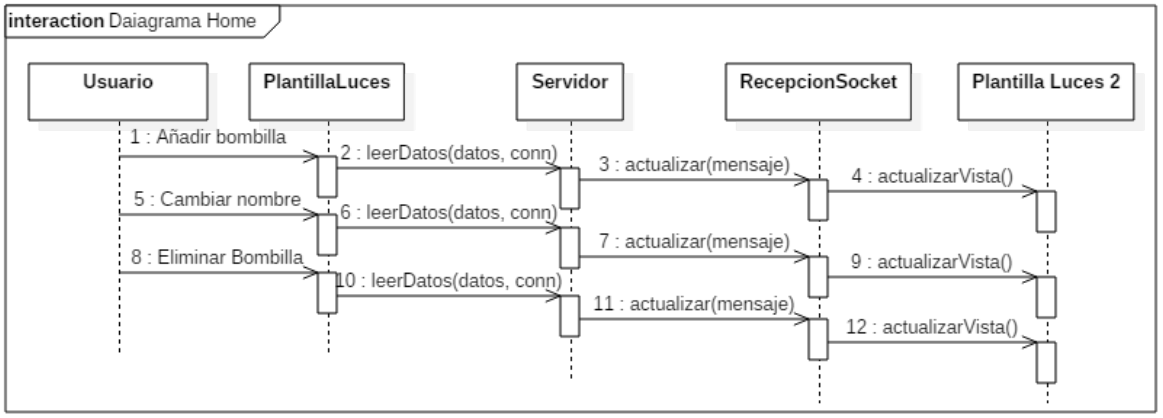

Figura C.2: Diagrama de acciones sobre una bombilla en la clase **Plantilla-Luces**.

C.4. Diseño arquitectónico

En este apartado hablaremos del diseño arquitectónico de ambas partes del proyecto.

Raspberry Pi

En esta parte del proyecto no existen paquetes que agrupen los ficheros y los archivos se encuentran en la raíz del proyecto. Vamos a hablar brevemente de los métodos y características de estos ficheros.

Servidor.py

En este fichero se encuentra la implementación de la clase *Servidor* que contiene los siguientes métodos.

- `__init__(self)`: se encarga de inicializar las variables, iniciar la interfaz y arrancar el servidor en un hilo independiente.
- `iniciarInterfaz(self)`: se encarga de inicializar todas la variables y estructuras de la interfaz y arrancarla.
- `centrarVentana(self, ventana, widht, height)`: centra la ventana pasada por parámetro con un tamaño igual a *widht* y *height*.
- `agregarIp(self)`: se encarga de mostrar una nueva ventana desde la que poder actualizar la IP.
- `actualizarIp(self)`: se encarga de recoger el valor de la IP que se ha introducido en la ventana creada por `agregarIp(self)`.
- `leerDatos(self, datos, conn)`: se encarga de interpretar todos los mensajes que llegan al servidor y actuar en medida de lo que se ha leído.
- `clienteThread(self, conn)`: se encarga de crear un nuevo hilo independiente para cada conexión de un cliente con el servidor.
- `iniciarServidor(self)`: se encarga de inicializar todas las variables y estructuras del servidor y arrancarlo.
- `main()`: se encarga de realizar la ejecución principal del archivo.

Database.py

En este fichero se encuentra la implementación de la clase *Database* que contiene los siguiente métodos

- `__init__(self)`: se encarga de crear las tablas de la base de datos si no estuviesen ya creadas.

- `insertarEstancia(self, id, nombre)`: se encarga de insertar una nueva estancia en la base de datos.
- `insertarElemento(self, estancia, nombre, estado)`: se encarga de insertar un nuevo elemento en la base de datos.
- `actualizarEstancia(self, nombreViejo, nombreNuevo)`: se encarga de actualizar el nombre de una estancia de la base de datos.
- `actualizarElemento(self, estancia, nombreViejo, nombreNuevo)`: se encarga de actualizar nombre de un elemento de una estancia de la base de datos.
- `actualizarEstado(self, estancia, nombre, estado)`: se encarga de actualizar el estado de un elemento de una estancia de la base de datos.
- `actualizarIP(self, ip)`: se encarga de actualizar la IP de la base de datos.
- `eliminarEstancia(self, estancia)`: se encarga de eliminar una estancia de la base de datos.
- `eliminarElemento(self, estancia, index)`: se encarga de eliminar un elemento de una estancia de la base de datos.
- `recuperarEstancia(self, index)`: se encarga de devolver una estancia de la base de datos.
- `recuperarElemento(self, estancia, index)`: se encarga de devolver un elemento de la base de datos.
- `recuperarIp(self)`: se encarga de devolver la ip almacenada en la base de datos.
- `numeroEstancias(self)`: se encarga de devolver el número de estancias en la base de datos.
- `numeroElementos(self, estancia)`: se encarga de devolver el número de elementos de una estancia de la base de datos.
- `cerrarBD(self)`: se encarga de cerrar la conexión con la base de datos.
- `borrarBD(self)`: se encarga de borrar todas las tablas de la base de datos.

Aplicación Android

En esta parte del proyecto no existen paquetes para diferenciar unas clases de otras, todas se encuentran en el directorio **Simulador Domotica Raspberry/app android/app/src/main/java/mario/app android**. La función de cada una de las clases la podemos encontrar en la sección D.3. Aquí simplemente vamos a visualizar el diagrama de clases de cada uno de los ficheros *.java* de ese directorio.

1. `BDLocal`: C.3

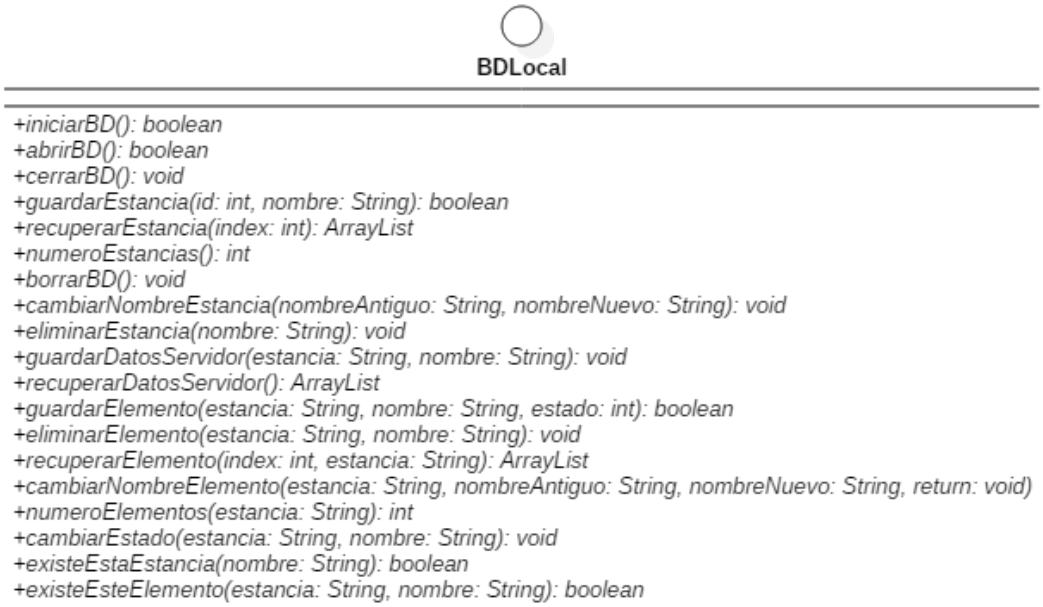

Figura C.3: Diagrama de la clase **BDLocal**.

2. `SQLite`: C.4
3. `Conexion`: C.5
4. `CustomAdapterEstancia`: C.6
5. `CustomAdapterLuz`: C.7
6. `Habitacion`: C.8

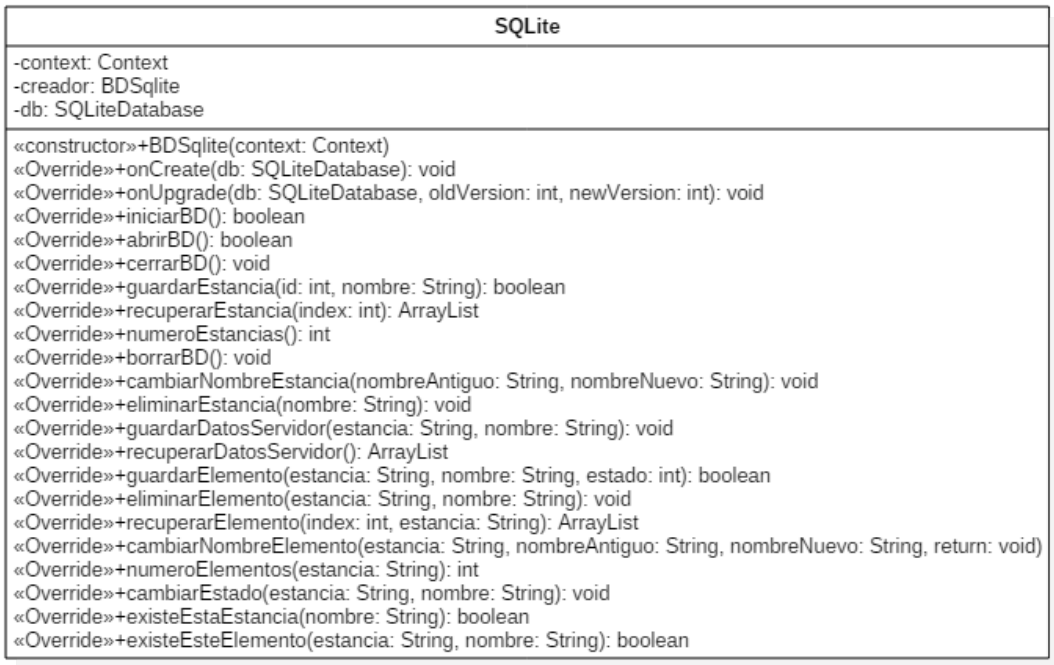

Figura C.4: Diagrama de la clase **SQLite**.

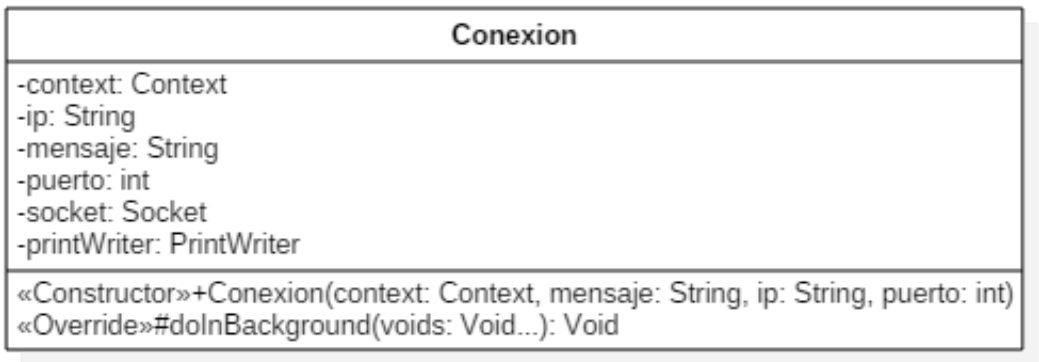

Figura C.5: Diagrama de la clase **Conexion**.

7. `Home`: C.9

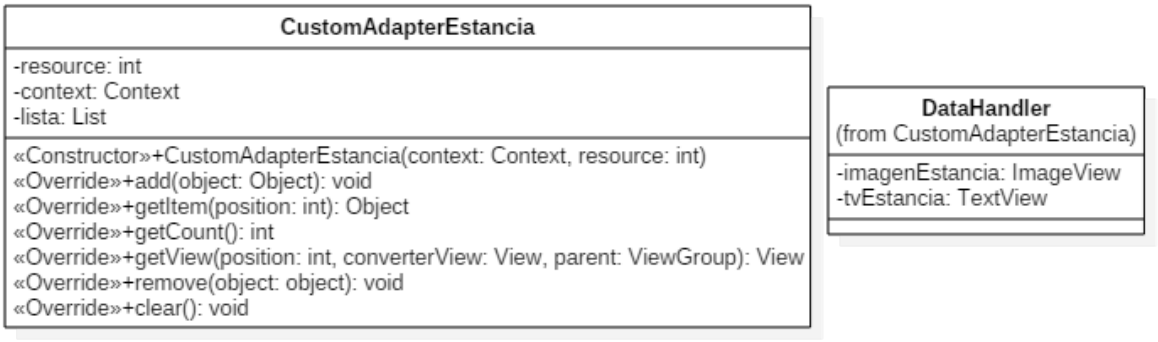

Figura C.6: Diagrama de la clase **CustomAdapterEstancia**.

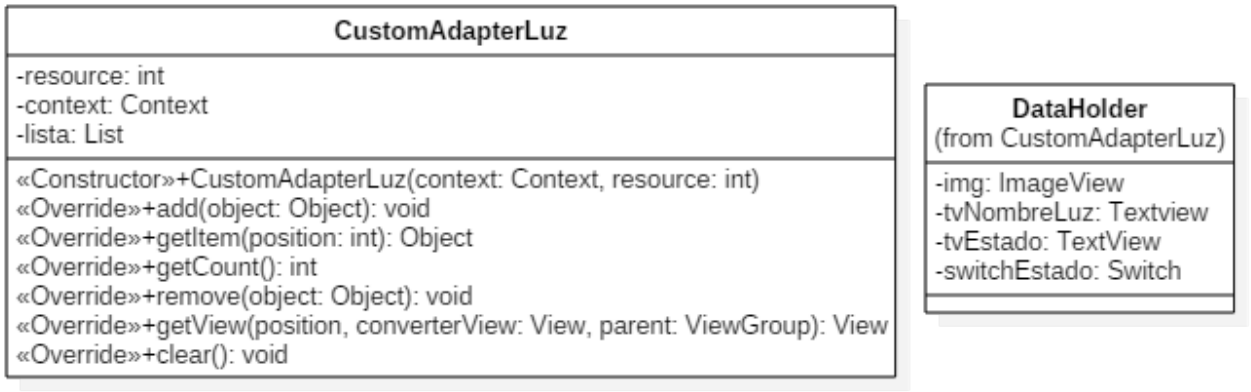

Figura C.7: Diagrama de la clase **CustomAdapterLuz**.

8. `Luz`: C.10
9. `PlantillaLuces`: C.11
10. `RecepcionSocket`: C.12
11. `Servidor`: C.13

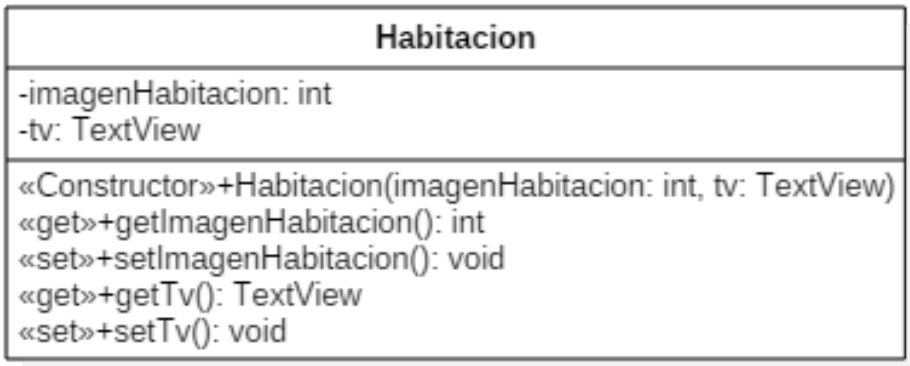

Figura C.8: Diagrama de la clase **Habitacion**.

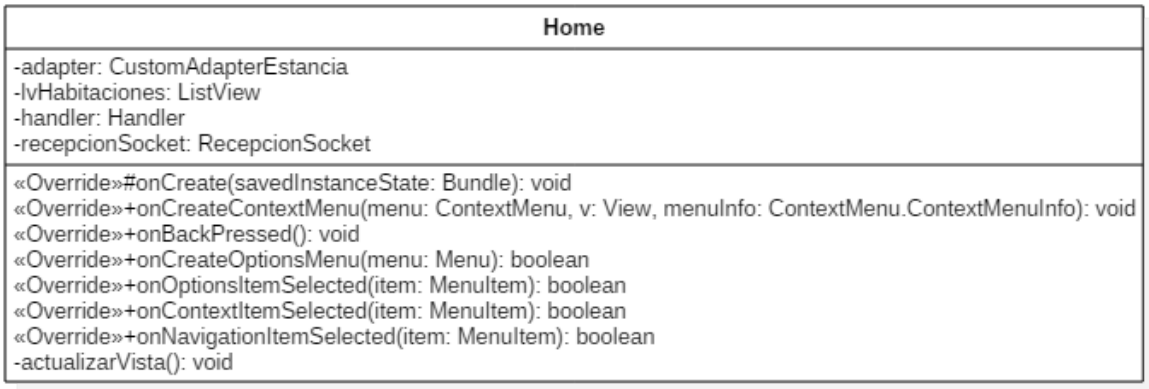

Figura C.9: Diagrama de la clase **Home**.

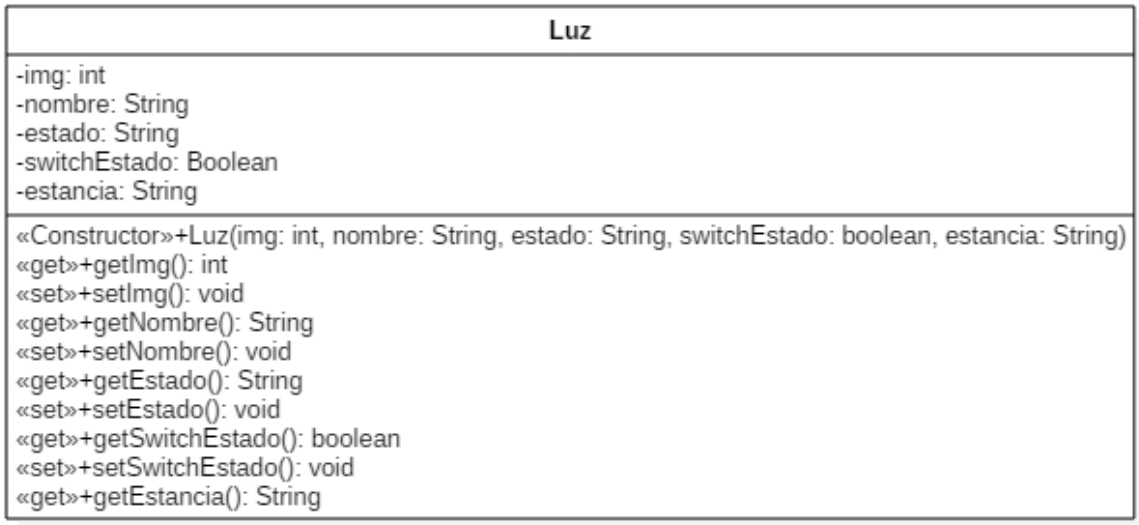

Figura C.10: Diagrama de la clase **Luz**.

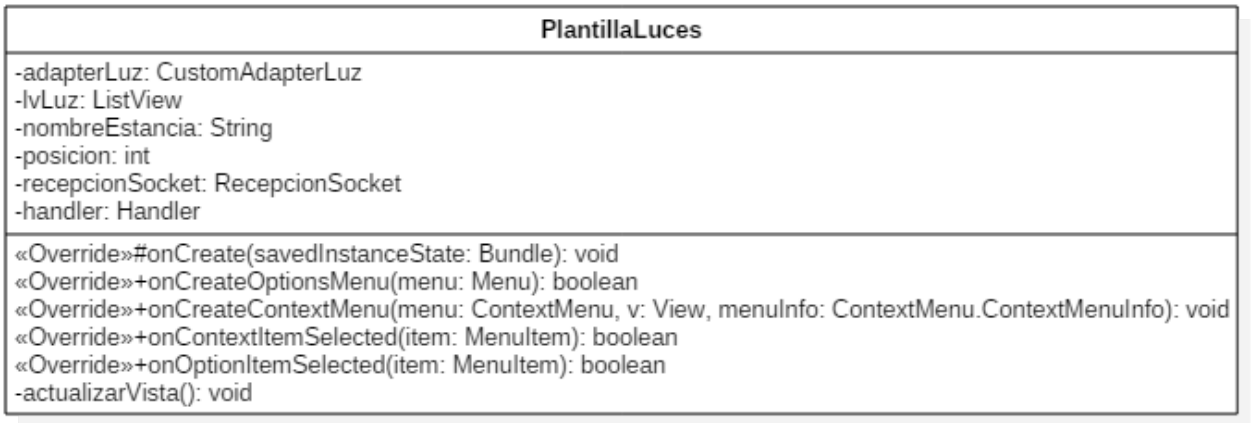

Figura C.11: Diagrama de la clase **PlantillaLuces**.

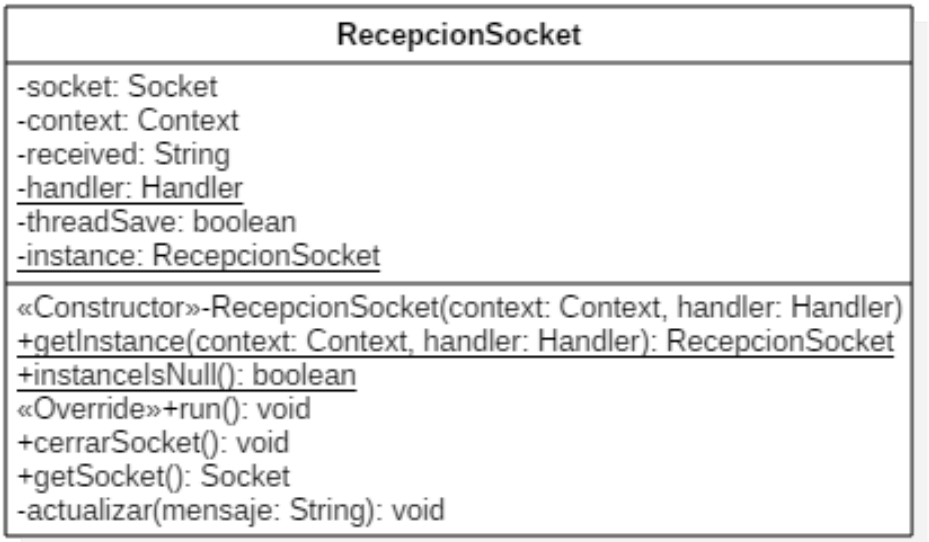

Figura C.12: Diagrama de la clase **RecepcionSocket**.

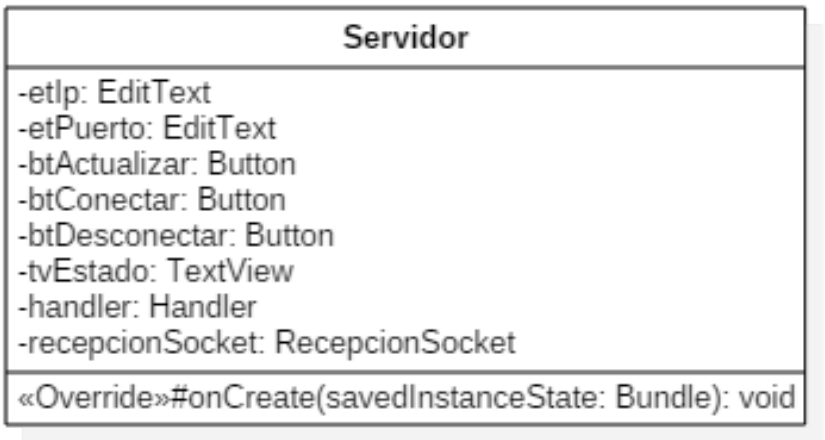

Figura C.13: Diagrama de la clase **Servidor**.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice voy a tratar de explicar todos los detalles técnicos necesarios para que las personas que desean trabajar con este proyecto o incluso continuarlo puedan hacerlo de manera sencilla.

D.2. Estructura de directorios

La siguiente estructura de directorios es la utilizada para la realización del proyecto y se puede encontrar en el siguiente repositorio https://github.com/MJ0T4/Simulador_Domotica_Raspberry.

- **'/':** directorio raíz del proyecto. En el se encuentra el fichero *README.md* y los siguientes directorios:
  - **apk:** contiene la apk para realizar la instalación de la aplicación
  - **app android:** contiene todo el proyecto referente a la parte de *Android* con Android Studio.
    - **app:** contiene el código fuente del proyecto.
      - **src:** contiene el código fuente del proyecto.
        1. **androidTest/java/mario/app android:** contiene los test instrumentales.

2. **main**: contiene todas las clases java y todos los recursos del proyecto.
3. **test/java/mario/app android:** contiene los test unitarios.

- **diseño:** contiene el archivo *diseño.mdj* que contiene a su vez todos los diagramas realizado para este proyecto.
- **doc:** contiene la documentación del proyecto.
  - **img:** contiene todas las imágenes utilizadas para la documentación de la memoria y anexos.
  - **tex:** contiene cada uno de los ficheros correspondientes a la memoria y anexos.
- **servidor raspberry:** contiene todo el proyecto referente a la parte de *Python 3* con Jetbrains PyCharms.
  - **imagenes:** contiene todas la imágenes utilizadas por la interfaz.

D.3. Manual del programador

En este apartado voy a hacer una breve explicación técnica del funcionamiento del proyecto.

Raspberry Pi

En la Raspberry Pi ejecutaremos el código desarrollado en Python 3. Dentro del directorio del proyecto de Python tenemos dos ficheros fundamentales.

- El fichero `Servidor.py` contiene toda la lógica de la interfaz y el servidor.
- El fichero `Database.py` contiene todos los métodos que permiten realizar cambios en la base de datos.

Dentro del fichero `Servidor.py` solo se encuentra una clase que controla todo. Dentro de ella existen métodos para la creación de la interfaz y su tratamiento y métodos para la inicialización del servidor y el tratamiento de mensajes. El hilo de ejecución de ambos se realiza a la vez en la llamada a la clase, pero cada uno tiene su hilo independiente.

Aplicación Android

En este apartado sobre hablaré brevemente sobre los ficheros .java que se encuentran en el directorio **Simulador Domotica Raspberry/app android/app/src/main/java/mario/app android**. Dentro de este directorio nos encontramos con los siguientes ficheros.

- `BDLocal.java`: interfaz creada para declarar, pero no implementar, los métodos que se usarán para realizar cambios en la base de datos.
- `BDSqlite.java`: implementa la interfaz `BDLocal.java` y desarrolla los métodos declarados por ella.
- `Conexion.java`: implementa una tarea asíncrona para mandar cambios al servidor.
- `CustomAdapterEstancia.java`: adaptador personalizado para representar las filas de estancias en una *ListView*.
- `CustomAdapterLuz.java`: adaptador personalizado para representar las filas de bombillas en una *ListView*.
- `Habitacion.java`: implementa los elementos que irán en cada fila del adaptador `CustomAdapterEstancia.java`.
- `Home.java`: implementa la ventana principal donde se representarán las estancias de nuestra vivienda.
- `Luz.java`: implementa los elementos que irán en cada fila del adaptador `CustomAdapterLuz.java`.
- `PlantillaLuces.java`: implementa la *Activity* en la que se mostrarán las bombillas de una estancia.
- `RecepcionSocket.java`: implementa la escucha de mensajes del servidor en un `Thread` implementado la interfaz `Runnable`.
- `Servidor.java`: implementa la modificación de datos del servidor, su conexión y desconexión.

D.4. Compilación, instalación y ejecución del proyecto

Para comenzar con la compilación del proyecto es necesario descargarlo a través de *GitHub* https://github.com/MJ0T4/Simulador_Domotica_Raspberry. Una vez tenemos el proyecto descomprimido comenzamos a trabajar con él.

Para realizar la compilación del proyecto Android, tendremos que instalar previamente la herramienta de desarrollo **Android Studio**. Para ello, podemos descargar el ejecutable desde el siguiente enlace:

https://developer.android.com/studio/

Una vez descargado e instalado, lo ejecutamos. Dentro de la ventana principal nos digirimos al menú superior de opciones. En el seguiremos los siguientes pasos:

1. Pulsar sobre la opción `File` del menú de opciones.
2. Pulsar sobre la opción `New` del menú que se ha desplegado.
3. Pulsar sobre la opción `Import project` del menú que se ha desplegado nuevamente.
4. Se abrirá un explorador de archivos en el que debemos buscar el directorio `app_android` del proyecto que hemos descargado.
5. Una vez seleccionado lo cargamos y ya tendremos el proyecto en nuestro Android Studio.

Con el proyecto ya cargado en la herramienta Android Studio solo debemos pulsar sobre el icono de un martillo verde y el proyecto comenzará a compilarse.

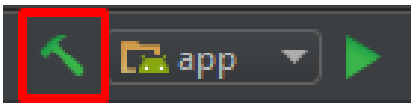

Figura D.1: Icono para la compilación del proyecto.

La ejecución del proyecto es posible realizarla sobre un emulador virtual de Android que viene incluido con la herramienta o sobre un terminal físico. Para la ejecución mediante el emulador virtual solo debemos pulsar sobre el

botón con símbolo *Play* de color verde que se puede ver en la imagen D.1. Si por el contrario deseamos realizar la instalación de la apk debemos mirar la sección E.3. Para conocer los posibilidades que tiene esta aplicación tanto en el emulador virtual como en el terminal físico mirar la sección E.4.

La compilación del proyecto de la parte de Raspberry Pi no es necesaria ya que simplemente con la ejecución del fichero `Servidor.py` se realiza la compilación. Para realizar la instalación del proyecto de Python en la Raspberry Pi mirar la sección E.3. Y para su ejecución y funcionamiento mirar la sección E.4.

D.5. Pruebas del sistema

Durante la realización de este proyecto, las pruebas sobretodo de la parte de Android han sido realizadas manualmente a través de un emulador de Android que proporciona la herramienta Android Studio. Solo existía una parte del proyecto que no se podía visualizar y era la parte de la base de datos. Por ello, creí conveniente realizar un test instrumental que comprobase el correcto funcionamiento de los métodos de la base de datos sobre el emulador virtual de Android.
En la parte de la Raspberry Pi, las pruebas también se han realizado manualmente, además de que su código no es muy extenso. Para comprobar el correcto funcionamiento de la base de datos, como en la parte de Android, he utilizado el programa *DB Browser for Sqlite*. A través de este programa he podido visualizar las tablas y el contenido del archivo *database*. Este programa se puede descargar desde el siguiente repositorio:

https://github.com/sqlitebrowser/sqlitebrowser/releases

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado hablaremos sobre todo lo necesario para realizar la instalación y la puesta a punto del sistema.

E.2. Requisitos de usuarios

Todos los elementos necesarios para poder ejecutar este proyecto son los siguientes.

- Terminal basado en Android con una versión igual a Lollipot 5.0 o superior.
- Raspberry Pi con conexión a internet por Wifi o Ethernet. En nuestro caso usaremos la Raspberry Pi 3 Modelo B conectada por Wifi.
- Una micro SD de al menos 8 gigabytes (el sistema ocupa más de 4 gigabytes).
- Es necesario que el terminal y la Raspberry Pi estén conectados a la misma red de Internet.

E.3. Instalación

Raspberry Pi

Comenzaremos con los pasos necesarios para instalar un nuevo sistema operativo en nuestra Raspberry Pi.
El primer paso es conectar nuestra micro SD a un ordenador a través de un adaptador y formatearla como podemos ver en E.1

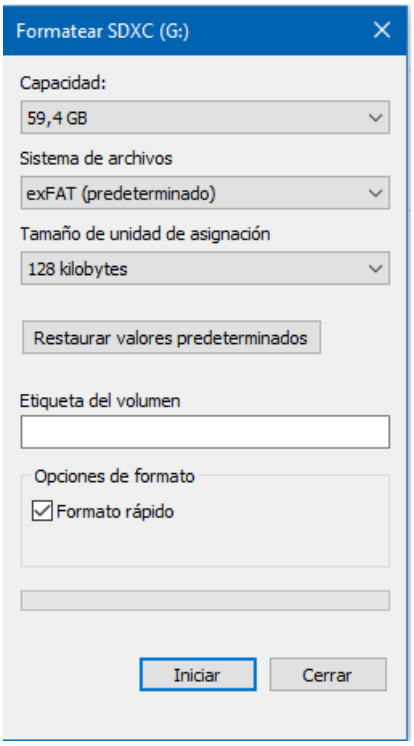

Figura E.1: Formateo en exFAT por ser una SD de gran tamaño.

Después descargaremos la imagen del sistema operativo que queremos utilizar en nuestra Raspberry. En este caso usaremos *Raspbian* que es el más común para este dispositivo.
Podremos descargar la imagen desde la página oficial de Raspberry [2].
Una vez dentro de la página de descargas de Raspberry y tras dirigirnos a la opción de *Raspbian*, tenemos dos opciones.

- *Raspbian Stretch with desktop*, que es una imagen con escritorio basado en Debian Stretch.

- *Raspbian Stretch Lite*, que es una versión reducida de la anterior en la que no contaremos con escritorio.

Para nuestro caso, utilizaremos la primera opción ya que necesitamos un escritorio en el que podamos visualizar el estado de nuestra casa mediante la interfaz. Una vez elegimos el sistema con escritorio se nos descargará un archivo *.zip* que contendrá nuestra imagen del sistema.

Ahora que tenemos nuestra imagen del sistema, necesitamos un programa especial para grabar esa imagen en nuestra micro SD. El programa que yo recomiendo y he usado para la grabación de la imagen se llama *Etcher*. *Etcher* es un programa para la grabación de imágenes en SD, que existe en varias plataformas y además es de software libre. Puede ser descargado desde su página oficial [3].

Una vez tenemos la imagen descargada, el programa instalado y la micro SD conectada al ordenador, vamos a grabar el sistema operativo. Para ello abrimos el programa *Etcher* y seguimos los siguientes pasos:

1. Pulsamos sobre *Select image* y mediante el explorador de archivos buscamos la imagen que previamente habíamos descargado.

2. Pulsamos sobre *Select drive* y marcamos nuestra micro SD.

3. Finalmente pulsamos sobre *Flash* y comenzará a funcionar.

Antes comenzar a grabar la imagen, el aspecto que debería tener el programa es el siguiente:

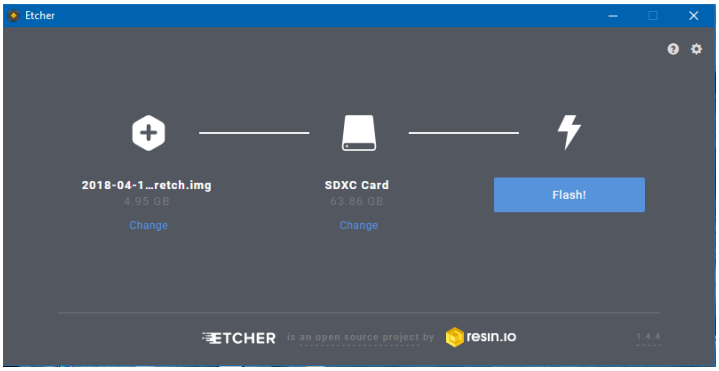

Figura E.2: Ventana principal del programa Etcher con la imagen cargada.

Tras finalizar la grabación de la imagen en la micro SD, Windows no reconocerá la tarjeta y nos dirá que existe un problema con esa unidad, insistiendo en que debemos formatearla. Esto es debido al formato de archivos que utiliza *Raspbian*, que no es posible ser leído en Windows.
Ahora mismo, ya tenemos nuestro sistema operativo preparado para arrancar, asi que solo nos queda insertar la micro SD en la Raspberry Pi y encenderla.

Tras arrancar nuestra Raspberry Pi, deberíamos ver un escritorio como el que se muestra en la imagen E.3.

Figura E.3: Escritorio de Raspbian Stretch en Raspberry Pi.

Lo primero que recomiendo es actualizar el sistema y todos sus paquetes a las versiones más actuales, y activar además las opciones de VNC y SSH desde la configuración de la Raspberry Pi. Una vez hecho todo lo anterior, antes de empezar debemos comprobar que ya venga por defecto preinstalada una versión de Python 3 en nuestro sistema.
Para comprobarlo debemos abrir una terminal, ya sea mediante el icono de la barra de tareas o la combinación de teclas Ctrl + Alt + T.
Una vez dentro de la terminal, hay que tener en cuenta que el sistema contiene a la vez una versión de Python 2 y una de Python 3. Para diferenciar cuando se utiliza cada una, se sigue la siguiente norma: Siempre que un comando comience por *python*, se usará la versión 2 y cuando el comando comience por *python3* se usará la 3.
Teniendo en cuenta esta aclaración, vamos a ejecutar un comando con Python 3 para conocer la versión instalada y a su vez, comprobar que realmente exista una versión de Python 3 instalada. El comando a utilizar es el siguiente:

```
python3 --version
```

La salida por pantalla de la terminal debería parecerse a la que se muestra en la imagen E.4.

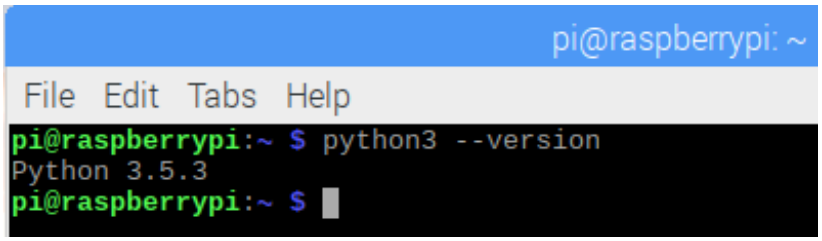

Figura E.4: Salida por pantalla del comando python3 –version.

Sabiendo que nuestro sistema ya tiene una versión preinstalada de Python 3, no tenemos nada más que pasar los archivos *Servidor.py*, *Database.py* y la carpeta *imagenes* a la Raspberrry y ejecutarlos. Podemos pasarlos a través de Internet o simplemente introduciendo una memoria USB en la Raspberry con ellos. Para que todo quede más ordenado he creado una carpeta llamada *Servidor Raspberry* que contendrá todos los ficheros necesarios. Debería verse como en la imagen E.5.

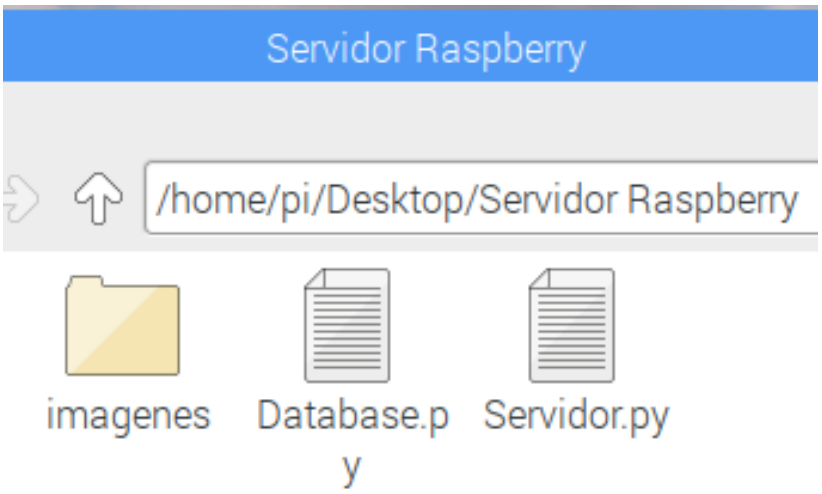

Figura E.5: Directorio *Servidor Raspberry* que contiene los ficheros especificados.

Android

Para comenzar a utilizar la aplicación, lo primero que tenemos que hacer es instalar la apk. Para poder poder instalarla, al ser una aplicación externa

que no proviene de ninguna market o tienda, es necesario activar la opción de orígenes desconocidos que permite instalar aplicaciones de terceros. Para activar dicha opción tenemos que seguir los siguientes pasos:

1. Entrar en el menú *Ajustes* de nuestro terminal Android.
2. Dirígete a la opción *Seguridad* dentro de la categoría *Personal*.
3. Y ahora de nuevo dirígete a la sección *Administración de dispositivos* y la segunda opción *Orígenes desconocidos* es la que debemos activar.
4. Una vez intentemos activarla, nos mostrará un mensaje de confirmación que deberemos aceptar.

La apk se puede descargar directamente desde el repositorio del proyecto (https://github.com/MJOT4/Simulador_Domotica_Raspberry).

E.4. Manual del usuario

Es esta sección vamos a explicar detalladamente como conseguir ejecutar la parte de *Raspberry Pi* y la parte de *Android* conjuntamente.

Raspberry Pi

Para comenzar con la parte del servidor, he optado por introducir la IP sobre la que se creará el servidor de forma gráfica para que sea más fácil para el usuario. Lo primero es saber como ejecutar el servidor y para ello vamos a seguir los siguientes pasos:

1. Abrimos una terminal desde el icono de la barra de tareas o con la combinación de teclas Ctrl + Alt + T.
2. Necesitamos dirigirnos al *Escritorio* que es donde habíamos guardado previamente los ficheros. Para ello escribimos:

   ```
   cd Desktop/
   ```

3. Ahora que ya nos encontramos en el *Escritorio* debemos ir a la carpeta que donde tengamos los ficheros, en mi caso *Servidor Raspberry*.

   ```
   cd Servidor\ Raspberry/
   ```

4. Por seguridad comprobaremos que realmente los ficheros están ahí mediante el comando `ls`.

5. Una vez hemos comprobado que tenemos los ficheros *Servidor.py*, *Database.py* y el directorio *imagenes*, ya podemos ejecutar el servidor. Para ello utilizamos el siguiente comando:

   ```
   python3 Servidor.py
   ```

En la imagen E.6 se puede observar una terminal con los comandos que hemos mencionado antes.

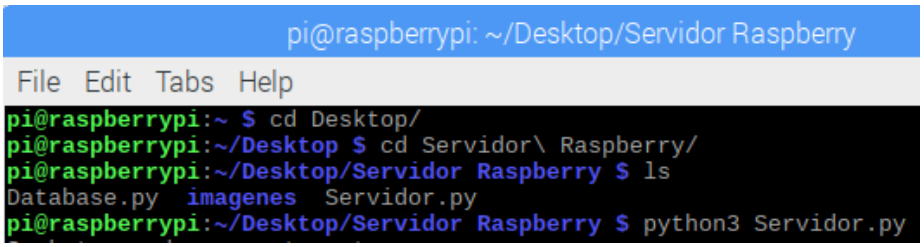

Figura E.6: Terminal con los comandos necesarios para ejecutar el servidor.

Después del último comando se abrirá una nueva ventana que será la interfaz que nos mostrará los elementos de nuestra casa que iremos creando y modificando mediante la aplicación de Android.
Por defecto, si la aplicación acaba de ser arrancada por primera vez, nos mostrará un error diciéndonos que no ha sido posible montar el servidor sobre esa ip. El error será igual al de la imagen E.7.

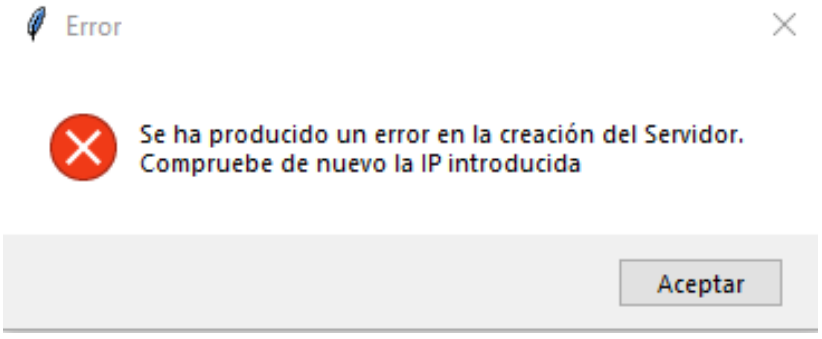

Figura E.7: Error en la creación del servidor esa una IP.

Cuando pulsemos el botón *Aceptar*, se abrirá una ventana emergente en la que podremos escribir nuestra IP y que será almacenada en la base datos.

Por tanto, una vez que hemos introducido nuestra IP correctamente, el resto de veces que carguemos el servidor de nuevo utilizará esta IP.
La ventana emergente se verá como la imagen E.8.

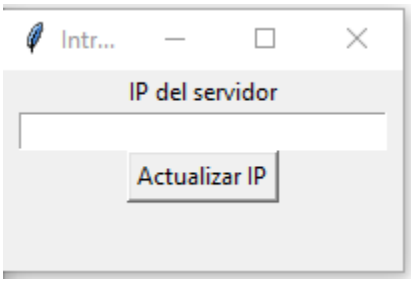

Figura E.8: Error en la creación del servidor esa una IP.

Una vez introducida la IP y pulsado el botón *Actualizar IP*, la ventana se cerrará y se intentará crear de nuevo el servidor. Si la IP introducida es incorrecta, volverá a mostrarse el error y de nuevo la ventana emergente. Si la IP es correcta, el servidor se creará y no saltarán más errores. Si por alguna razón debiésemos reiniciar el servidor, no será necesario introducir de nuevo la IP porque ha sido guardada en la base de datos.
Si quisiéramos introducir una nueva IP válida, solo tendríamos que borrar la base de datos manualmente o con la opción existente en la App móvil.

Android

Con la aplicación ya instalada, vamos a hablar sobre las posibilidades que nos ofrece la App y sus ventanas.

Ventana Principal

Cuando se abre la aplicación nos encontramos con la ventana que se muestra en la imagen E.9. En ella se mostrarán nuestras estancias, es decir, las diferentes salas de estar que podemos crear. Las salas que podemos crear son ***Habitaciones***, ***Salones***, ***Cocinas*** y ***Baños***, y cada una de ellas tiene una imagen asignada.
En la parte superior, contamos con un menú desplegable que podemos mostrar pulsando sobre las tres líneas blancas horizontales, el título y un botón desde el que podemos desplegar opciones.
Pinchando sobre el botón con icono verde y símbolo + se mostrará una ventana emergente con diferentes opciones. En dicha ventana E.10, tendremos

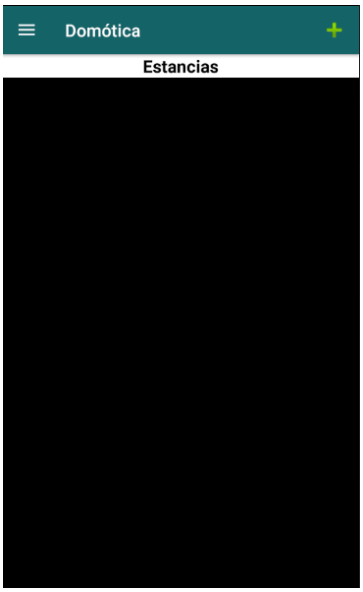

Figura E.9: Ventana inicial y principal de esta aplicación.

la opción de crear cualquier tipo de las cuatro estancias pulsando sobre su nombre y además elegir un nombre para ella.

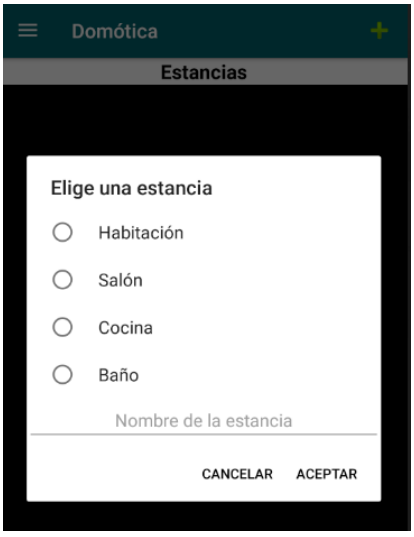

Figura E.10: Ventana emergente para la creación de nuevas estancias.

Tras seleccionar una de las opciones y escribir un nombre para ella, pulsamos sobre el botón *Aceptar* y se creará. Para realizar un ejemplo,

crearemos una *Habitación* con el nombre *Habitación*, que podremos ver en la imagen E.11.

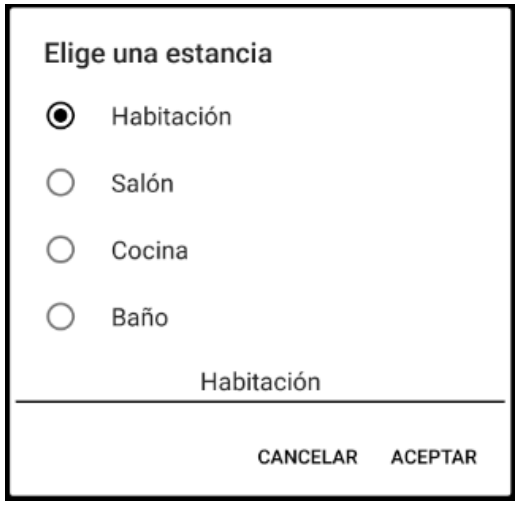

Figura E.11: Creación de una nueva *Habitación* mediante la ventana emergente.

Una vez pulsado el botón *Aceptar* se nos creará la habitación y la ventana principal se nos quedará de la siguiente manera E.12.
Con está habitación ya creada, contamos con las opciones de eliminarla o cambiar su nombre.

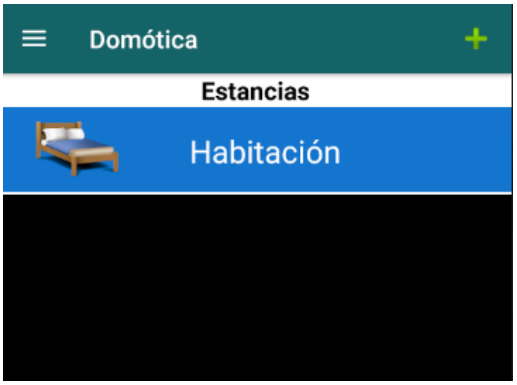

Figura E.12: Creación de una nueva *Habitación* mediante la ventana emergente.

Para poder borrar o cambiar el nombre de la *Habitación* creada anteriormente debemos acceder a su menú contextual. Este menú aparecerá tras

pulsar poco más de un segundo sobre ella. Se puede ver en la imagen E.13.

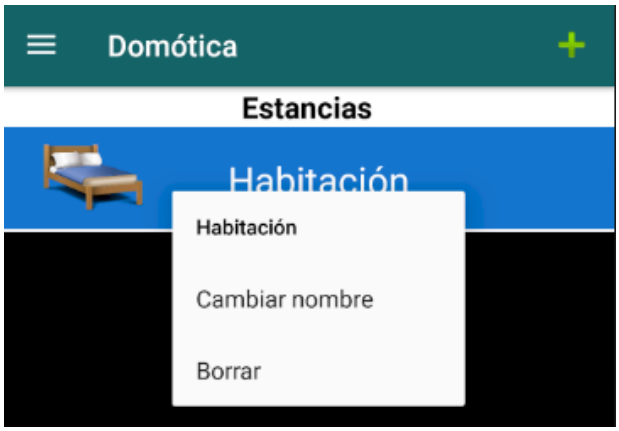

Figura E.13: Menú contextual de una estancia

Si pulsamos sobre la opción *Cambiar nombre* se mostrará una nueva ventana emergente en la que debemos insertar el nombre y pulsar el botón *Aceptar*. Dicha ventana emergente se verá como la imagen E.14

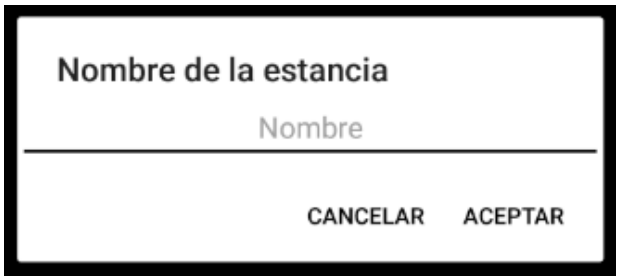

Figura E.14: Opción *Cambiar nombre* del menú contextual de una *Habitación*.

En cambio, si pulsamos sobre la opción *Borrar* se mostrará un mensaje emergente de confirmación para la realización de dicha acción. El mensaje sería igual al de la imagen E.15.

Para poder realizar todas las acciones a anteriores es necesario estar conectado al servidor, sino se nos mostrará una ventana emergente E.16 con un botón que nos dirigirá a la ventana del Servidor.

Si pulsamos sobre las tres líneas blancas horizontales en la barra superior, se desplegará un menú. En este menú tenemos dos opciones:

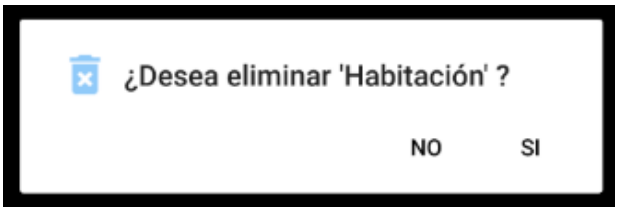

Figura E.15: Opción *Borrar* del menú contextual de una *Habitación*.

Figura E.16: Ventana emergente en caso de no estar conectado al servidor.

- **Servidor**, que nos redirigirá a la misma ventana que el botón de la imagen E.16.
- **Borrar base de datos**, que borrará tanto los datos locales como los del servidor.

El menú desplegable podemos verlo en la imagen E.17.

Figura E.17: Menú principal que se mostrará al pulsar en las tres líneas blancas horizontales

Ventana Servidor

Esta es otra de las ventanas de la aplicación desde la que podemos acceder mediante el menú desplegable que se encuentra en la ventana principal(E.17) o mediante el botón de la ventana emergente que se nos muestra cuando queremos realizar una acción y no estamos conectados al servidor (E.16).

En la ventana servidor (E.18) nos encontramos con dos campos para rellenar:

Figura E.18: Ventana *Servidor* de la aplicación.

- **IP**, este campo viene vacío por defecto y en él debemos escribir la *IP* del servidor al que queremos conectarnos.

- **Puerto**, este campo viene por defecto como **8888** y es el puerto que se usará para realizar la conexión, aunque permite la posibilidad de ser cambiado.

Una vez hemos escrito la **IP** y el **Puerto** al que deseamos conectarnos, solo debemos presionar el botón *Actualizar*. Cuando el botón es pulsado, estos datos son guardados en la base de datos para futuras conexiones, de manera que solo deberíamos realizar este paso una vez.
Tras actualizar los datos, ya podemos conectarnos al servidor, así que pulsamos sobre el botón *Conectar* que se encuentra en la inferior. Si la conexión se ha realizado con éxito, nuestra ventana del Servidor se debería ver como la imagen E.19, sino se seguiría viendo como la imagen E.18.

Figura E.19: Conexión exitosa con el servidor.

Esta acción de conectarse al servidor solo deberá realizarse una vez, ya que por defecto en la ventana principal E.9 tratará de conectarse al servidor con la **IP** que hayamos guardado en la base de datos desde la ventana del Servidor E.18.

Una vez estamos conectados y deseamos volver a la ventana principal, solo debemos pulsar sobre la flecha que se encuentra en la parte superior izquierda de la ventana. De nuevo en ventana principal, vamos a hablar sobre la última ventana.

Ventana iluminación

La ventana de iluminación es una ventana desde la cuál podemos crear, modificar y eliminar *Bombillas* para las estancias creadas en la ventana principal. Para acceder a ella, solo debemos encontrarnos en la ventana principal y pulsar sobre la estancia que queremos. Tras esto, nos dirigirá a una nueva ventana como la de la imagen E.20

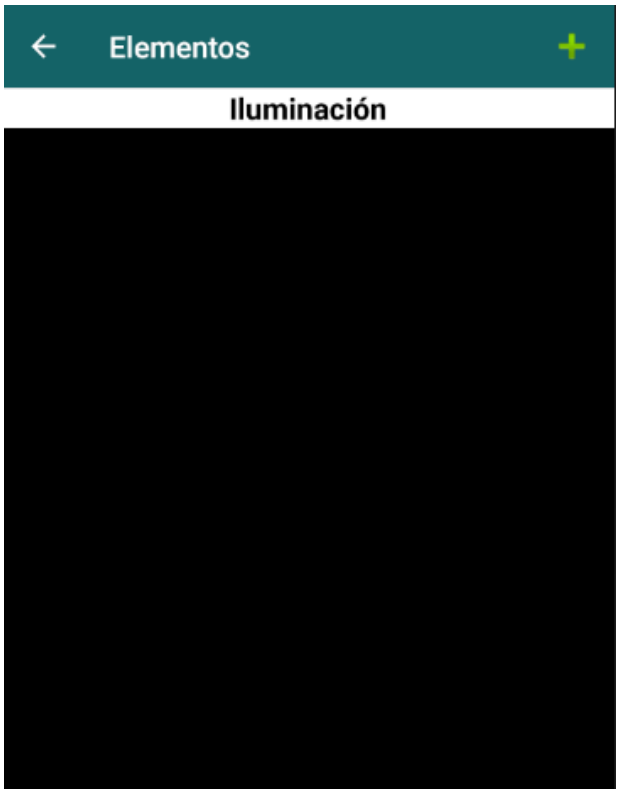

Figura E.20: Ventana iluminación.

Pinchando sobre el botón con icono verde y símbolo + que se encuentra en la parte superior derecha, se mostrará una ventana emergente E.21 para la creación de bombillas.

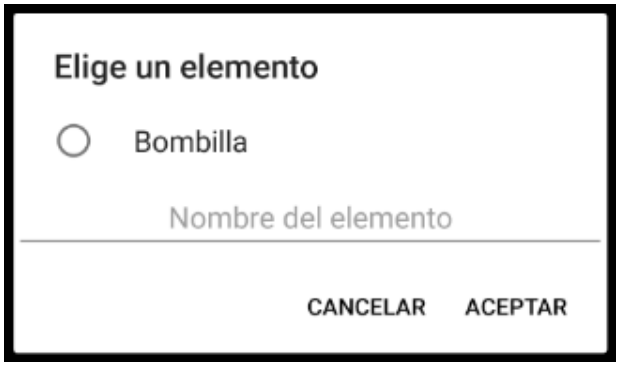

Figura E.21: Ventana emergente para la creación de una bombilla.

Debemos pinchar sobre la única opción posible, especificar un nombre para ella y pulsar sobre el botón *Aceptar*. Tras hacerlo, se nos creará una nueva *Bombilla* y nuestra ventana de iluminación se verá como la de la imagen E.22.

Figura E.22: Ventana iluminación tras crear una *Bombilla*.

En esta nueva *Bombilla* contamos con una imagen, que nos muestra su estado, su nombre, un texto que también que nos indica su estado y un interruptor para encenderla y apagarla. Para cambiar su estado, es tan sencillo como pulsar el interruptor y ver como se ilumina la bombilla, tanto aquí como en la parte de Python. La imagen E.23 muestra como sería el resultado de pulsar el interruptor.

Figura E.23: Ventana iluminación con una *Bombilla* encendida.

Además de todo esto, cada *Bombilla* que creemos, contará con su propio menú contextual E.24 con las mismas opciones que las estancias (*Cambiar nombre* y *Borrar*) y que se mostrará de la misma manera.

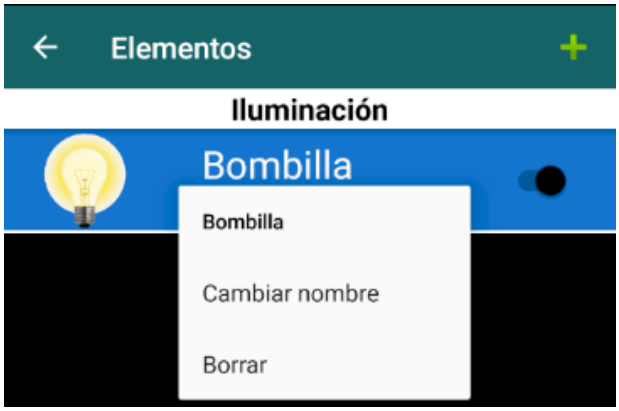

Figura E.24: Menú contextual de una *Bombilla*.

Bibliografía

[1] ARM® big.LITTLE™ technology. Odroid-xu4 — http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825. [Internet; descargado 30-mayo-2018].

[2] Raspberry Pi Foundation. https://www.raspberrypi.org/downloads/. [Online; accedido 22-mayo-2018].

[3] Resin.io. https://etcher.io/. [Online; accedido 22-mayo-2018].

[4] Wikipedia. Análisis dafo — wikipedia, la enciclopedia libre — https://es.wikipedia.org/w/index.php?title=An%C3%A1lisis_DAFO&oldid=108233925, 2018. [Internet; descargado 30-mayo-2018].

[5] Wikipedia. Top-down y bottom-up — wikipedia, la enciclopedia libre — https://es.wikipedia.org/w/index.php?title=Top-down_y_bottom-up&oldid=107948796, 2018. [Internet; descargado 30-mayo-2018].