

UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

“HomePi, Plataforma para la gestión de objetos inteligentes”

Vº Bº del Director del
Proyecto

DIRECTOR: Vicente García Díaz

CODIRECTOR: Cristian González García

AUTOR: Santiago Martín Agra

Agradecimientos

Me gustaría agradecer primero a mi familia y amigos por apoyarme y darme fuerza durante el desarrollo de este proyecto, tampoco me quiero olvidar de la gente de Simbiosys que sin los días libres que me han dejado hubiera sido imposible.

Resumen

Se ha desarrollado un sistema con el objetivo de controlar diferentes objetos inteligentes, en este caso los objetos inteligentes son sensores y actuadores que combinados o de forma independiente nos permiten crear accesorios para controlar puertas de garaje y luces o para obtener información sobre la humedad y temperatura.

Este proyecto tiene tres partes:

HomePi Server, programada con TypeScript en NodeJS, se ejecuta en una Raspberry Pi y es el puente de comunicación entre la aplicación móvil y los sensores y actuadores colocados en casa.

HomePi Android, programada en Android, es una aplicación que permite consultar el estado de todos los accesorios, así como crear nuevos o cambiar la configuración de los existentes. También permite la recepción de notificaciones.

HomePi Wemos, programados en SimpleC, se instalan en placas Wemos. Si es la primera vez se instala o no se conecta a ninguna red Wi-Fi, se crear una red Wi-Fi propia para permitir configurar la red Wi-Fi a la que se quiere conectar, entonces ya estará preparado para recibir peticiones HTTP consultando el estado de sensores o para modificar el estado de un actuador.

Para realizar la comunicaciones entre **HomePi Server** y **HomePi Android** se utiliza la base de datos en tiempo real de Firebase.

Palabras Clave

Internet de las Cosas, Domótica, DiY, Firebase, Android, iOS, NodeJS, Java, TypeScript, Home Automation.

Abstract

A system has been developed with the objective of controlling different intelligent objects, in this case the intelligent object are sensors and actuators which combined or independently allow us to create accessories to control garage doors or home lights or get information about temperature or humidity.

This project has three parts:

HomePi Server, programmed with TypeScript in NodeJS, which run on a Raspberry Pi and is the bridge between the Android app and the sensors and actuators placed at home.

HomePi Android, programmed with Java, is an application that allows us to interact or check the status of all accessories, as well as create or change any setting of an existing accessory. It also allows the reception of notifications from HomePi Server.

HomePi Wemos, programmed in SimpleC, run on a Wemos board. If we run it for the first time or it can not connect to any WiFi network, HomePi Wemos will create its own WiFi network to allow us to configure with WiFi network use, then it will be prepared to receive HTTP requests to send the sensor actual status or to change the actuator status.

The communications between HomePi Server and HomePi Android are performed through the Firebase real time database.

Keywords

Internet of Things, DiY, Firebase, Android, iOS, NodeJS, Java, TypeScript, Home Automation.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO	15
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	15
1.2 RESUMEN DE TODOS LOS ASPECTOS	16
1.2.1 <i>Introducción</i>	16
1.2.2 <i>Análisis y diseño</i>	16
1.2.3 <i>Implementación</i>	16
1.2.4 <i>Planificación y presupuesto</i>	17
1.2.5 <i>Pruebas</i>	17
CAPÍTULO 2. INTRODUCCIÓN	18
2.1 JUSTIFICACIÓN DEL PROYECTO	18
2.2 OBJETIVOS DEL PROYECTO	19
2.3 ESTUDIO DE LA SITUACIÓN ACTUAL	20
2.3.1 <i>PiDome</i>	20
2.3.2 <i>Blynk</i>	21
2.4 EVALUACIÓN DE ALTERNATIVAS HARDWARE	22
2.4.1 <i>Microordenador</i>	23
2.4.2 <i>Microcontroladores</i>	24
2.5 EVALUACIÓN DE ALTERNATIVAS SOFTWARE.....	26
2.5.1 <i>Sistema operativo Raspberry Pi</i>	26
2.5.2 <i>Domótica para dispositivos móviles</i>	28
2.5.3 <i>Lenguaje de programación Raspberry Pi</i>	30
2.5.4 <i>Sistema de gestión de base de datos</i>	32
CAPÍTULO 3. ASPECTOS TEÓRICOS	36
3.1 INTERNET DE LAS COSAS O IoT	36
3.2 DIY o DO IT YOURSELF	36
3.3 NOTIFICACIONES PUSH	36
3.4 FIREBASE.....	36
3.4.1 <i>Base de datos en tiempo real</i>	37
3.4.2 <i>Firebase Cloud Messaging</i>	37
CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTO INICIALES.....	38
4.1 PLANIFICACIÓN INICIAL.....	38
4.2 PRESUPUESTO INICIAL.....	45
4.2.1 <i>Desarrollo de Presupuesto Detallado (Empresa)</i>	45
4.2.2 <i>Desarrollo de Presupuesto Simplificado (Cliente)</i>	48
CAPÍTULO 5. ANÁLISIS	49
5.1 DEFINICIÓN DEL SISTEMA.....	49
5.1.1 <i>Determinación del Alcance del Sistema</i>	49
5.2 REQUISITOS DEL SISTEMA	51
5.2.1 <i>Obtención de los Requisitos del Sistema</i>	51
5.2.2 <i>Identificación de Actores del Sistema</i>	53
5.2.3 <i>Especificación de Casos de Uso</i>	53

5.3	IDENTIFICACIÓN DE LOS SUBSISTEMAS EN LA FASE DE ANÁLISIS	54
5.3.1	<i>Descripción de los Subsistemas</i>	55
5.3.2	<i>Descripción de los Interfaces entre Subsistemas</i>	56
5.4	DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS.....	57
5.4.1	<i>Diagrama de Clases</i>	57
5.4.2	<i>Descripción de las Clases</i>	57
5.5	ANÁLISIS DE INTERFACES DE USUARIO	60
5.5.1	<i>Descripción de la Interfaz</i>	60
5.6	ESPECIFICACIÓN DEL PLAN DE PRUEBAS.....	62
5.6.1	<i>Pruebas unitarias</i>	62
5.6.2	<i>Pruebas de integración</i>	62
5.6.3	<i>Pruebas de sistema</i>	62
5.6.4	<i>Pruebas de usabilidad</i>	62
CAPÍTULO 6. DISEÑO DEL SISTEMA.....		64
6.1	ARQUITECTURA DEL SISTEMA	64
6.1.1	<i>Diagramas de Paquetes</i>	64
6.1.2	<i>Diagramas de Despliegue y Componentes</i>	66
6.2	DISEÑO DE CLASES	70
6.2.1	<i>Diagrama de Clases</i>	70
6.3	DIAGRAMAS DE INTERACCIÓN.....	75
6.3.1	<i>Interactuar con accesorio</i>	75
6.4	DISEÑO DE LA BASE DE DATOS.....	76
6.4.1	<i>Descripción del SGBD Usado</i>	76
6.4.2	<i>Integración del SGBD en Nuestro Sistema</i>	76
6.4.3	<i>Modelo de datos</i>	76
6.5	DISEÑO DE LA INTERFAZ	77
6.5.1	<i>Pantalla inicial</i>	77
6.5.2	<i>Listado de accesorios</i>	78
6.5.3	<i>Configuración del accesorio</i>	79
6.6	ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS.....	80
6.6.1	<i>Pruebas Unitarias</i>	80
6.6.2	<i>Pruebas de Integración y del Sistema</i>	83
6.6.3	<i>Pruebas de Usabilidad</i>	83
6.6.4	<i>Pruebas de Rendimiento</i>	86
CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA		87
7.1	ESTÁNDARES Y NORMAS SEGUIDOS	87
7.1.1	<i>HomePi Server</i>	87
7.1.2	<i>HomePi Andriod</i>	88
7.1.3	<i>JSON</i>	88
7.2	LENGUAJES DE PROGRAMACIÓN.....	89
7.2.1	<i>Android SDK + Java</i>	89
7.2.2	<i>TypeScript</i>	90
7.2.3	<i>Simple C</i>	91
7.3	HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO	92
7.3.1	<i>WebStorm</i>	92
7.3.2	<i>npm</i>	92
7.3.3	<i>gulp.js</i>	93
7.3.4	<i>nodemon</i>	93

7.3.5	<i>Mocha</i>	93
7.3.6	<i>Chai</i>	94
7.3.7	<i>David-DM</i>	94
7.3.8	<i>Travis CI</i>	95
7.3.9	<i>Android Studio</i>	95
7.3.10	<i>Gradle</i>	96
7.3.11	<i>VisualStudio Code</i>	96
7.3.12	<i>Git</i>	96
7.3.13	<i>GitHub</i>	97
7.3.14	<i>GitHub Desktop</i>	97
7.3.15	<i>Arduino IDE</i>	98
7.3.16	<i>Microsoft Word</i>	98
7.3.17	<i>Microsoft Excel</i>	99
7.3.18	<i>Microsoft Project</i>	99
7.3.19	<i>Google Drive</i>	99
7.3.20	<i>Draw.io</i>	100
7.3.21	<i>Flat Tomato</i>	100
7.4	CREACIÓN DEL SISTEMA	102
7.4.1	<i>Problemas Encontrados</i>	102
7.4.2	<i>Descripción Detallada de las Clases</i>	103
CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS		104
8.1	PRUEBAS UNITARIAS.....	104
8.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA	107
8.2.1	<i>Pruebas de integración</i>	107
8.2.2	<i>Pruebas de sistema</i>	107
8.3	PRUEBAS DE USABILIDAD.....	108
8.3.1	<i>Usuarios con sistemas de domótica</i>	108
8.3.2	<i>Usuarios con ganas de implantar un sistema de domótica</i>	112
8.4	PRUEBAS DE RENDIMIENTO.....	117
CAPÍTULO 9. MANUALES DEL SISTEMA.....		118
9.1	MANUAL DE INSTALACIÓN	118
9.1.1	<i>Crear proyecto en Firebase</i>	118
9.1.2	<i>Configurar HomePi Server</i>	118
9.1.3	<i>Configurar HomePi Android</i>	120
9.1.4	<i>Configurar HomePi Wemos</i>	121
9.2	MANUAL DE EJECUCIÓN	124
9.2.1	<i>Ejecutar</i>	124
9.2.2	<i>Detener</i>	124
9.3	MANUAL DE USUARIO	125
9.3.1	<i>Gestión usuarios</i>	125
9.3.2	<i>Crear accesorio</i>	125
9.3.3	<i>Configurar accesorios</i>	127
9.3.4	<i>Cambiar proyecto de Firebase</i>	127
9.3.5	<i>Recuperar contraseña</i>	128
9.4	MANUAL DEL PROGRAMADOR.....	130
9.4.1	<i>HomePi Server</i>	130
9.4.2	<i>HomePi Android</i>	130

CAPÍTULO 10. CONCLUSIONES Y AMPLIACIONES.....	132
10.1 CONCLUSIONES.....	132
10.2 AMPLIACIONES	133
10.2.1 <i>Integración con HomeKit de Apple.</i>	133
10.2.2 <i>Integración con asistentes.....</i>	133
10.2.3 <i>NFC</i>	133
10.2.4 <i>Awareness API de Google.....</i>	133
10.2.5 <i>Macros o acciones grabadas o automáticas</i>	134
10.2.6 <i>Roles</i>	134
10.2.7 <i>Compatibilidad con Amazon Dash.....</i>	134
CAPÍTULO 11. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTO FINALES	136
11.1 PLANIFICACIÓN FINAL	137
11.2 PRESUPUESTO FINAL.....	143
11.2.1 <i>Desarrollo de Presupuesto Detallado (Empresa).....</i>	143
CAPÍTULO 12. REFERENCIAS BIBLIOGRÁFICAS	145
12.1 LIBROS Y ARTÍCULOS.....	145
12.2 REFERENCIAS EN INTERNET	147
CAPÍTULO 13. APÉNDICES	148
13.1 GLOSARIO Y DICCIONARIO DE DATOS	148
13.2 CONTENIDO ENTREGADO EN EL ARCHIVO ADJUNTO	149
13.2.1 <i>Contenidos generales</i>	149
13.2.2 <i>Subsistemas.....</i>	149
13.3 CÓDIGO FUENTE	151

Índice de Figuras

Tabla 4-1. Coste detallado por tareas.....	45
Tabla 4-2. Coste detallado hardware	46
Tabla 4-3. Coste detallado software.....	46
Tabla 4-4. Gastos mensuales fijos	46
Tabla 4-5. Presupuesto total del proyecto	47
Tabla 4-6. Presupuesto del cliente	48
Tabla 11-1. Coste final detallado: tareas	143
Tabla 11-2. Coste final detallado: hardware.....	143
Tabla 11-3. Coste final detallado: Software.....	144
Tabla 11-4. Gastos finales mensuales fijos	144
Tabla 11-5. Presupuesto total final del proyecto.....	144
Ilustración 1. Logotipo PiDome	20
Ilustración 2. Interfaz de control web de PiDome	20
Ilustración 3. Logotipo de Blynk	21
Ilustración 4. Capturas de la aplicación de Blynk para móviles.....	22
Ilustración 5. Logotipo Raspberry Pi.....	23
Ilustración 6. Placa Orange Pi Plus2	24
Ilustración 7. Logotipo Arduino	25
Ilustración 8. Placa Wemos mini pro	25
Ilustración 9. Logotipo Raspbian	27
Ilustración 10. Logotipo Windows 10 IoT	27
Ilustración 11. Logotipo iOS	28
Ilustración 12. Logotipo HomeKit (iOS)	29
Ilustración 13. Logotipo Android	29
Ilustración 14. Google Home, dispositivo.....	30
Ilustración 15. Logotipo Java	31
Ilustración 16. Logotipo NodeJS	31
Ilustración 17. Logotipo Firebase	33
Ilustración 18. Logotipo PostgreSQL.....	34
Ilustración 19. Logotipo MySQL	34
Ilustración 20. Vista general de la planificación del proyecto	39
Ilustración 21. Planificación tarea: Inicio	40
Ilustración 22. Planificación tarea: Análisis	41
Ilustración 23. Planificación tarea: Diseño del sistema	42
Ilustración 24. Planificación tarea: Implementación del sistema	43
Ilustración 25. Planificación tarea: Final de documentación	44
Ilustración 5.1. Casos de uso	53
Ilustración 27. Subsistemas HomePi	55
Ilustración 28. Diagrama de clases preliminar	57
Ilustración 29. Interfaz pantalla inicial Android.....	60
Ilustración 30. Interfaz pantalla configuración Android	61
Ilustración 31. Diagrama de paquetes HomePi Android	64
Ilustración 32. Diagrama de paquetes HomePi Server.....	65
Ilustración 33. Diagrama de despliegue y componentes	67
Ilustración 34. Diagrama de clases HomePi Android	71

Ilustración 35. Diagrama de clases HomePi Server	73
Ilustración 36. Diagrama de interacción. Caso de uso: Interactuar con accesorio.....	75
Ilustración 37. Pantalla final HomePi Android: inicial.....	77
Ilustración 38. Pantalla final HomePi Android: Listado de accesorios.....	78
Ilustración 39. Pantalla final HomePi Android: Configuración del accesorio.....	79
Ilustración 40. Etiqueta Commitizen friendly	87
Ilustración 41. Etiqueta Semantic release	87
Ilustración 42. Logotipo ButterKnife	89
Ilustración 43. Ejemplos de MaterialDialog.....	90
Ilustración 44. Logotipo TypeScript	90
Ilustración 45. Logotipo WebStorm.....	92
Ilustración 46. Logo npm.....	92
Ilustración 47. Logotipo de gulp	93
Ilustración 48. Logotipo nodemon	93
Ilustración 49. Logotipo Mocha.....	94
Ilustración 50. Logotipo chai	94
Ilustración 51. Etiqueta David DM	94
Ilustración 52. Logotipo de Travis CI.....	95
Ilustración 53. Logotipo Android Studio	95
Ilustración 54. Logotipo Gradle	96
Ilustración 55. Logotipo VisualStudio Code	96
Ilustración 56. Logotipo git	97
Ilustración 57. Logotipo GitHub	97
Ilustración 58. Logotipo GitHub Desktop.....	97
Ilustración 59. Logotipo Arduino IDE.....	98
Ilustración 60. Logotipo Microsoft Word.....	98
Ilustración 61. Logo Microsoft Excel.....	99
Ilustración 62. Logo Microsoft Project	99
Ilustración 63. Logo Google Drive	100
Ilustración 64. Logotipo Draw.io	100
Ilustración 65. Logotipo Flat Tomato.....	101
Ilustración 66. Sensor DHT11	102
Ilustración 67. Sensor de DHT22	102
Ilustración 68. Mensaje de error en HomePi Android	107
Ilustración 69. Consola de Firebase	118
Ilustración 70. Configuración proyecto Firebase	119
Ilustración 71. Generación de la clave privada de Firebase	119
Ilustración 72. Vista general Consola de Firebase	120
Ilustración 73. Pantalla inicial de configuración de HomePi Android.....	120
Ilustración 74. Pantalla para configurar el proyecto de Firebase en HomePi Android	121
Ilustración 75. Google-services.json	121
Ilustración 76. Wemos con shield instalada	122
Ilustración 77. Menú configuración WiFi HomePi Wemos.....	122
Ilustración 78. Lista de redes disponibles HomePi WiFi	123
Ilustración 79. Consola de Firebase Authentication	125
Ilustración 80. Pantalla inicial HomePi Android	125
Ilustración 81. Elegir accesorio HomePi Android	126
Ilustración 82. Configurar nuevo accesorio HomePi Android	126
Ilustración 83. Cambio de configuración accesorio HomePi Android.....	127
Ilustración 84. Menú HomePi Android	128

Ilustración 85. Pantalla de contraseña HomePi Android	128
Ilustración 86. Aplicación prototipo vs HomePi Android	132
Ilustración 87. Amazon Dash.....	134
Ilustración 88. Amazon Dash IoT	135
Ilustración 89. Vista general planificación final	137
Ilustración 90. Plantificación final tarea: Inicio	138
Ilustración 91. Planificación final tarea: Análisis	139
Ilustración 92. Planificación final tarea: Diseño del sistema	140
Ilustración 93. Planificación final tarea: Implementación del sistema	141
Ilustración 94. Planificación final tarea: Final de documentación.....	142

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La motivación del principal del proyecto es construir un sistema de domótica basado en IoT con una orientación DIY, do it yourself, es decir que cualquiera pueda construir su propia versión del sistema y/o modificarla a su antojo para cubrir sus necesidades.

La base de este proyecto ha sido un sistema que permitía controlar una puerta del garaje desde el móvil (iOS y Android), creada durante las primeras semanas del curso.

Con esta base, dado el bajo coste de distintos: sensores, actuadores, microordenadores y microcontroladores y vista la gran comodidad que ofrecía se decía ampliar este sistema para incluir nuevas funcionalidades.

Como el objetivo final del proyecto sería demasiado amplio para tiempo que se dispone para realizar este Trabajo de Fin de Grado en el prototipo que se implementara compatibilidad con:

- Sensores:
 - Humedad y temperatura.
- Actuadores:
 - Relé.

Con esta pequeña lista de sensores y actuadores se permitirá:

- Controlar:
 - Puerta de garaje.
 - Luces.
 - Termostato.
- Obtener información sobre:
 - Humedad.
 - Temperatura.

1.2 Resumen de Todos los Aspectos

1.2.1 Introducción

A finales de verano tuvimos un problema en casa, los mandos para controlar la puerta del garaje dejaron de funcionar. En lugar de comprar unos nuevos, decidí que podía aprovechar la oportunidad para hacerlo mejor.

¿Cómo? Pues en lugar de utilizar un mando tradicional, con un alcance determinado y que muchas veces no funciona, utilizar el teléfono móvil.

Una vez desarrollado el prototipo que solo permite controlar la puerta del garaje y comprobar lo cómodo que resulta y lo baratos que son los componentes, pensé que si podía controlar el garaje porque no las luces de casa o un termostato.

Es por ello que se decide desarrollar este proyecto que permitirá controlar y configurar nuevos tipos de objetos: luces, termostato y la puerta del garaje y obtener información de sensores para conocer la humedad y temperatura de casa desde una aplicación móvil.

1.2.2 Análisis y diseño

Una de las partes más críticas de este proyecto es hacer funcionar de forma conjunta diferentes dispositivos: un microordenador como la Raspberry Pi, microcontroladores compatibles con Arduino y una aplicación móvil para Android que permitan conocer y modificar el estado del sistema en cualquier momento.

También se debe tratar de simplificar la forma de incorporar compatibilidad con nuevos sensores y actuadores.

1.2.3 Implementación

El proyecto consta de varios componentes, el primero será el encargado de mantener la comunicación entre los diferentes sensores y actuadores y la aplicación móvil, para ello se utilizará Firebase.

El primer componente funciona en una Raspberry Pi 3, pero al utilizar un lenguaje multiplataforma, como NodeJS, es también compatible con Windows y Mac.

Otro de los componentes serán los diferentes microcontroladores que serán los encargados de controlar los sensores y actuadores disponibles en el sistema.

Por último, tenemos la aplicación Android que se conectara con Firebase para conocer y modificar el estado de los distintos sensores y actuadores, así como para crear, editar o borrar nuevos accesorios.

1.2.4 Planificación y presupuesto

Se ha planificado que el desarrollo del proyecto se lleva a cabo en **360 horas** comenzando el 1 mayo de 2017 y terminando el 3 de julio de 2017 y el presupuesto estimado ha sido de **25.156,03 €.**

1.2.5 Pruebas

Las pruebas son una parte importante de todos los proyectos, ya que sirven para detectar errores y así evitar que lleguen a los usuarios finales.

Con el objetivo de corregir el mayor número de errores posible se realizarán pruebas de diferente tipo: unitarias, de integración, de sistema, de usabilidad y accesibilidad.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Varios han sido los motivos que han llevado a la realización de este proyecto para construir un sistema de domótica con IoT y orientación DIY.

- Una vez construido el sistema para controlar la puerta del garaje y comprobar lo cómodo que resultaba se empieza a pensar: ¿Y si ahora controlo la calefacción también? ¿Y si ahora controlo las luces del salón? ¿Y si ...?
- El bajo precio de micrонтroladores, sensores y actuadores, así como la gran comunidad que hay al rededor de ellos que siempre son una fuente de inspiración.
- En cuanto al enfoque DIY, cada vez que hablaba a algún conocido sobre el sistema todos me hacían las mismas preguntas: ¿Y eso es muy difícil de hacer? ¿Y si quiero hacerlo yo que tengo que hacer?
- IoT es un área interesante que está en continuo crecimiento y casi cada día están saliendo nuevos dispositivos, programas... para utilizarlo en distintos ámbitos de nuestra sociedad.
- El sistema tiene que permitir interactuar a varias personas de forma simultánea con los diferentes accesorios, ya que una de las motivaciones para la realización de este proyecto es la implementación en mi propia casa.
- También se implementará control remoto desde fuera la casa, para permitir por ejemplo encender la calefacción al salir del trabajo o simular la presencia encendiendo y apagando luces.

2.2 Objetivos del Proyecto

Los objetivos más relevantes a nivel técnico son:

- Desarrollar un sistema de domótica basado en IoT que pueda ser utilizado diariamente y por varias personas de forma simultánea.
- Hacer uso de diferentes sensores: temperatura, humedad... y actuadores: relés.
- Hacer trabajar de forma conjunta diferentes dispositivos microordenadores como Raspberry Pi, microcontroladores como Arduino y la aplicación móvil que permite controlar el sistema.
- Construir un sistema que permita añadir compatibilidad con nuevos sensores y actuadores de la forma más cómoda posible para el desarrollador.
- Documentar todos los pasos para facilitar a otras personas la construcción de un sistema basado en el que se va a desarrollar que cubra sus necesidades.

También es objetivo del proyecto mejorar o adquirir conocimientos en lenguajes de programación, creación de arquitecturas, administración de sistemas, técnicas para el desarrollo de proyectos...

2.3 Estudio de la Situación Actual

En esta sección se expondrán distintos sistemas similares al que se pretende desarrollar. Alguna de estas alternativas no cubriría todas las características del sistema que se va a desarrollar, pero todas tienen alguna funcionalidad común con el nuevo sistema.

2.3.1 PiDome

PiDome es una plataforma de Open Source de desarrollo de domótica diseñada especialmente para poder construir con una Raspberry Pi un sistema completo de domótica.



Ilustración 1. Logotipo PiDome

Al igual que el sistema que se pretende desarrollar PiDome está diseñado tanto para usuarios expertos, ya que permite crear y añadir compatibilidad con nuevos dispositivos, y para usuarios inexpertos, que fácilmente pueden añadir funciones creadas por otros usuarios.



Ilustración 2. Interfaz de control web de PiDome

2.3.1.1 Ventajas

- Ya está desarrollado.
- Posibilidad de crear macros para definir comportamientos conjuntos.
- Compatibilidad con otros dispositivos de domótica (como Philips Hue).
- Monitorización de los valores de los sensores.

2.3.1.2 Inconvenientes

- No tiene integración ni aplicación para iOS.
- No tiene compatibilidad nativa con micrcontroladores (ejemplo: Arduino)
- Necesidad de abrir puertos para controlarlo desde fuera de casa, con los consecuentes problemas de seguridad.

2.3.2 Blynk

Blynk es una plataforma para controlar Raspberry Pis y Arduinos mediante Internet que cuenta con aplicaciones para iOS y Android.



Ilustración 3. Logotipo de Blynk

Blynk ofrece un sistema que permite crear nuestro propio dashboard para controlar los diferentes sensores y actuadores que tengamos conectados a nuestras microordenadores o microcontroladores.

Por un lado, tenemos la aplicación para dispositivos móviles, que nos permite definir los sensores y actuadores que tenemos conectados a nuestros controladores: eligiendo tipo: led, motor, sensor de temperatura... y en qué pin o pines lo hemos contactado.

También ofrece una librería para que los desarrolladores puedan hacer compatibles nuevos sensores o actuadores que encajen con los definidos dentro de la aplicación, así como ejemplos de código para las diferentes plataformas que soporta para que solo sea necesario que nuestro controlador ejecute ese código automáticamente podamos controlarlo desde Blynk.

Por último, tenemos el servidor de Blynk que es el que encarga de las comunicaciones entre la aplicación móvil y los diferentes controladores.

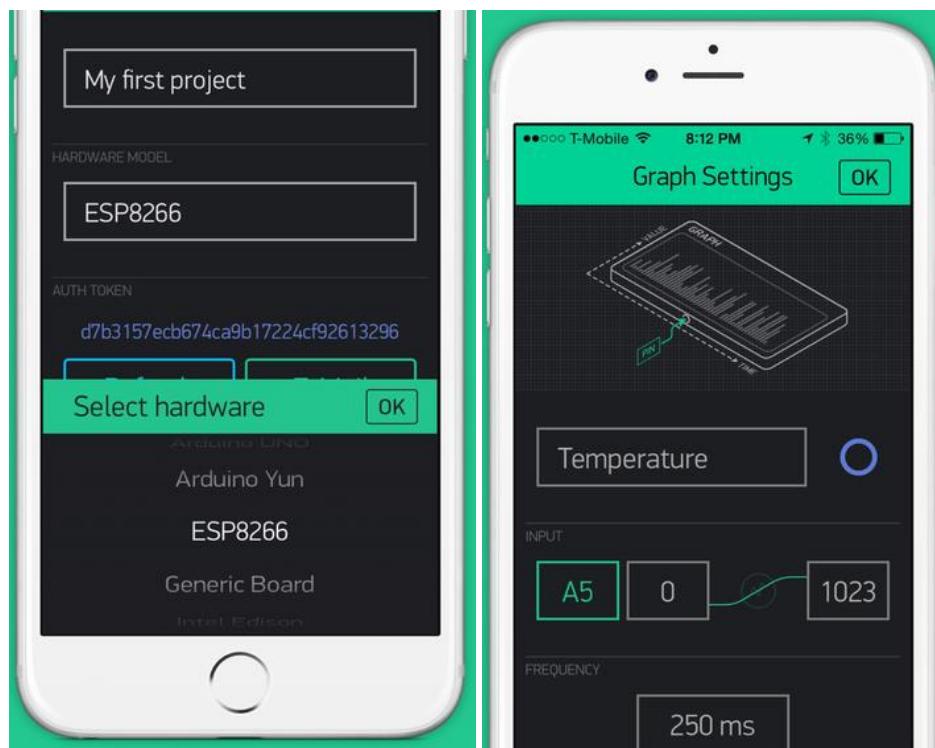


Ilustración 4. Capturas de la aplicación de Blynk para móviles.

2.3.2.1 Ventajas

- Ya está desarrollado.
- Aplicaciones para iOS y Android
- Gran cantidad de ejemplos y documentación.
- Facilidad de uso.

2.3.2.2 Inconvenientes

- Necesidad de abrir puertos para controlarlo desde fuera de casa, con los consecuentes problemas de seguridad.
- Limitado los dispositivos disponibles desde la aplicación de Blynk.
- No ofrece compatibilidad con HomeKit.

2.4 Evaluación de Alternativas Hardware

En esta sección se describen las distintas alternativas a nivel de hardware evaluadas para la realización del proyecto. De cada una de las alternativas se detallan las ventajas e inconvenientes.

2.4.1 Microordenador

Es el elemento principal del sistema, ya que será el encargado de comunicarse con las distintas aplicaciones que controlaran el sistema y los diferentes sensores y actuadores.

2.4.1.1 *Raspberry Pi 3*

Raspberry Pi es un computador de placa reducida, computador de placa única o computador 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.

En concreto la Raspberry Pi 3 fue lanzada en febrero de 2016 con una gran mejora en capacidad de procesamiento respecto al modelo anterior, así como la primera en incluir Wi-Fi y Bluetooth.



Ilustración 5. Logotipo Raspberry Pi

2.4.1.1.1 Ventajas

- La más popular de este tipo de ordenadores, por tanto, es el más soporte tiene.
- Cuenta con Wi-Fi y Bluetooth.
- Multitud de posibilidad a la hora de instalar un sistema operativo: Windows 10 IoT, Raspbian,...
- Multitarea.
- Compatibilidad con múltiples lenguajes de programación.

2.4.1.1.2 Inconvenientes

- Si se quisiera utilizar para controlar actuadores o leer datos de sensores solo cuenta con pines digitales.

2.4.1.2 Orange Pi Plus2



Ilustración 6. Placa Orange Pi Plus2

2.4.1.2.1 Ventajas

- Compatibilidad con Sistemas Operativos de Raspberry Pi
- Mayor RAM
- Multitarea.
- Cuenta con Wi-Fi y Bluetooth.

2.4.1.2.2 Inconvenientes

- No hay mucho soporte.

2.4.1.3 Alternativa seleccionada microordenador.

La Orange Pi Plus 2 técnicamente es superior a la Raspberry Pi 3, pero debido a que la Raspberry Pi es más popular y por tanto pose mayor soporte, así como que el autor posee una Raspberry Pi 3 Model B, por qué no haría falta comprar otra.

2.4.2 Microcontroladores

Dado que este proyecto pretende poder controlar distintos sensores y/o actuadores colocados en distintas partes de una casa/oficina/etc, se ha pensado utilizar microcontroladores que ya su precio es menor al de una Raspberry Pi.

2.4.2.1 Arduino UNO R3

Arduino, es una compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas computadora de desarrollo de hardware y software, compuesta respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programa cada placa.



Ilustración 7. Logotipo Arduino

2.4.2.1.1 Ventajas

- La más popular de este tipo de placas, por tanto, es la que más soporte tiene.

2.4.2.1.2 Inconvenientes

- No incorpora Wi-Fi o conexión ethernet en su versión básica.

2.4.2.2 Wemos D1 mini pro

Wemos D1 mini pro es un microcontrolador totalmente compatible con Arduino, lo que nos permite usar cualquier librería de Arduino. Está basada en el chip ESP-8266EX lo que hace que cuente con conexión Wi-Fi y Bluetooth integrada.

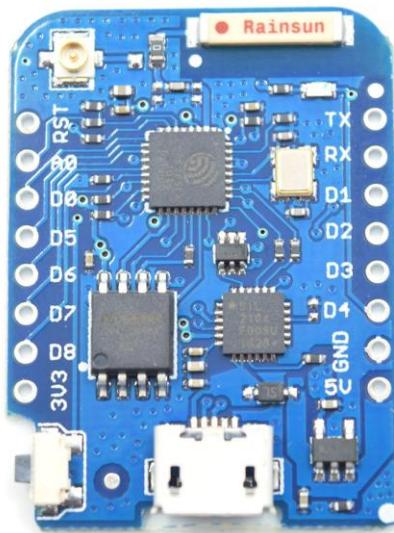


Ilustración 8. Placa Wemos mini pro

Su pequeño tamaño la hace idea para colocar en cualquier lugar de la casa para poder permitir controlar interruptores o colocar diferentes sensores de temperatura.

También Wemos ofrece una serie de sensores y actuadores diseñados para funcionar con la Wemos D1 mini que son del mismo tamaño de la placa y permiten crear dispositivos muy compactos.

2.4.2.2.1 Ventajas

- Compatibilidad total con Arduino.
- Wi-Fi integrado.
- Muy pequeña.
- Muy barata.

2.4.2.2.2 Inconvenientes

- Hay mucha información sobre el chip ESP-8266EX, pero en concreto sobre la placa Wemos hay bastante menos.

2.4.2.3 Alternativa seleccionada microntrolador

Dado que la idea es que estos microcontroladores estén en diferentes puntos de la casa de debido, su chip Wi-Fi integrado, su menor precio y la compatibilidad total con Arduino se ha elegido como microcontrolador la placa Wemos mini pro.

2.5 Evaluación de Alternativas Software

En esta sección se describen las distintas alternativas a nivel de software evaluadas para la realización del proyecto. Al igual que en caso de las alternativas hardware de cada alternativa se detallan las ventajas e inconvenientes.

2.5.1 Sistema operativo Raspberry Pi

2.5.1.1 Raspbian

Raspbian es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en Debian Wheezy (Debian 7.0) para la placa computadora Raspberry Pi.

Inicialmente era un port no oficial de Debian Wheezy adaptado para la arquitectura de la CPU de la Raspberry Pi. Actualmente debido a su popularidad la Raspberry Fundation colabora en su desarrollo y se actualiza a la casi a la par con la versión de escritorio de Debian.

Gracias a la popularidad de esta distribución cada vez existen más paquetes compilados de forma que sea compatible con la arquitectura de la Raspberry Pi, por lo que es prácticamente igual que la versión de Debian para ordenadores de escritorio.

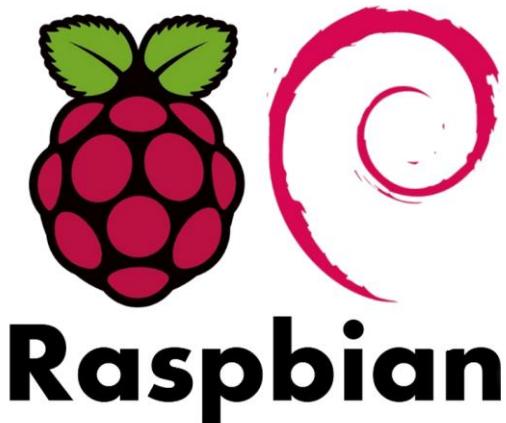


Ilustración 9. Logotipo Raspbian

2.5.1.1.1 Ventajas

- Es el sistema operativo oficial para la Raspberry Pi.
- Compatibilidad con la mayoría de paquetes de Debian.
- Puede controlarse solamente mediante la terminal.

2.5.1.1.2 Inconvenientes

- Sin una correcta configuración cualquiera podría acceder a él.

2.5.1.2 Windows 10 IoT Core

Windows 10 IoT Core es una versión de Windows optimizada para ser instalada en microordenadores, como la Raspberry Pi, que pueden poseer pantalla o no.



Ilustración 10. Logotipo Windows 10 IoT

El motivo de la existencia de Windows 10 IoT es permitir ejecutar las aplicaciones UWP (Universal Windows Platform) en estos dispositivos en un esfuerzo de Microsoft de aumentar la popularidad de estas aplicaciones, pero carece de una interfaz como la Windows 10 y de muchas de sus características.

La última actualización de Windows 10 IoT ha incluido la posibilidad utilizar a Cortana, el asistente virtual de Windows, para los proyectos que se desarrollen en él.

2.5.1.2.1 Ventajas

- Diseñado específicamente para IoT.
- Compatibilidad con Raspberry Pi
- Puede controlarse solamente mediante la terminal.
- Asistente virtual integrado.

2.5.1.2.2 Inconvenientes

- Se necesita Windows 10 para poder configurarlo.

2.5.1.3 *Alternativa seleccionada: Sistema operativo Raspberry Pi*

A pesar de Windows IoT es una opción interesante ya que es específico para IoT, dado que el sistema operativo principal del autor no es Windows y que desarrollarlo para Raspbian, dado que se utilizarán componentes compatibles con Debian u otras distribuciones no dejaría al sistema desarrollador dependiente del sistema operativo en el que se utilice.

2.5.2 Domótica para dispositivos móviles

Una parte importante a la hora de construir un sistema de estas características es como se va controlar, dado que hoy en día todo el mundo posee de un smartphone esta puede ser una las mejores opciones para controlarlo.

Por ello se ha realizado una pequeña investigación sobre como los dos sistemas operativos móviles más importantes, iOS y Android, se están preparando para la entrada de la domótica en nuestras vidas.

2.5.2.1 iOS

iOS es un sistema operativo móvil de la multinacional Apple Inc. Originalmente desarrollado para el iPhone (iPhone OS), después se ha usado en dispositivos como el iPod touch y el iPad. No permite la instalación de iOS en hardware de terceros.



Ilustración 11. Logotipo iOS

En iOS 8 Apple introduce su framework para el control de dispositivos de domótica HomeKit. En las siguientes versiones de iOS se añade compatibilidad con nuevos tipos de accesorios y con iOS se añade la aplicación Casa, para mejorar la gestión de estos accesorios.

2.5.2.1.1 Ventajas

- Integración total en el Sistema Operativo.
- Apple se encarga de la seguridad al utilizarlo fuera de casa.

2.5.2.1.2 Inconvenientes

- No se puede crear todo tipo de accesorios.
- Se necesita permanentemente en casa un dispositivo con iOS, Apple TV o iPad, para poder utilizarlo fuera de casa.



Ilustración 12. Logotipo HomeKit (iOS)

2.5.2.2 Android

Al contrario que en Apple en iOS, Google aún no ha añadido ningún tipo de framework de domótica a Android.



Ilustración 13. Logotipo Android

En los últimos meses parece que está preparando el terreno con la aplicación de Google Home, que sirve para controlar los Google Chromecast, dispositivo que conectado a la televisión permite enviar música o video desde nuestros teléfonos u ordenadores, y Google Home su dispositivo para hablar con el asistente de Google.



Ilustración 14. Google Home, dispositivo.

También han lanzado una primera versión de Android Things, que es una versión de Android optimizada para IoT que ofrece compatibilidad con todos los servicios de Google y muchas de las librerías disponibles para Android.

2.5.2.3 *Alternativa seleccionada: Domótica para dispositivos móviles*

Dado que se quiere ofrecer compatibilidad con ambos sistemas operativos móviles, se desarrollará el sistema de forma que desde iOS se pueda utilizar con HomeKit y para Android se desarrollará una aplicación que permita también controlar el sistema sin problemas.

2.5.3 Lenguaje de programación Raspberry Pi

En la actualidad debido a la popularidad de la Raspberry Pi y a que es capaz de ejecutar multitud de sistemas operativos, de los cuales muchos una versión adaptada de una versión de ordenadores de escritorio, la mayoría de los lenguajes de programación populares se pueden utilizar para desarrollar software que funcione en ella.

2.5.3.1 *Java*

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

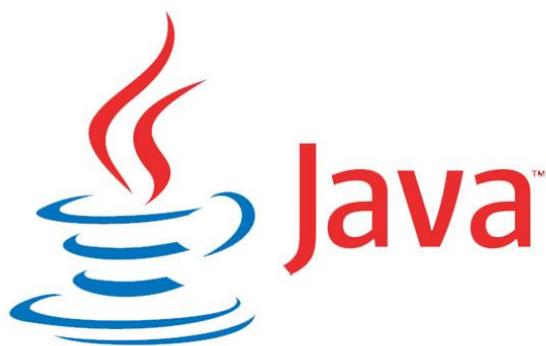


Ilustración 15. Logotipo Java

2.5.3.1.1 Ventajas

- Permite compatibilidad con varias plataformas.
- Hay multitud de librerías para utilizar las características de las Raspberry Pi.
- Es el lenguaje que mayoritariamente he utilizado durante la carrera.
- Hay librería para ofrecer compatibilidad con HomeKit.

2.5.3.1.2 Inconvenientes

- Excesivo código para tareas sencillas.

2.5.3.2 NodeJS

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web.



Ilustración 16. Logotipo NodeJS

2.5.3.2.1 Ventajas

- Permite compatibilidad con varias plataformas.
- Hay multitud de librerías para utilizar las características de las Raspberry Pi.
- Se puede añadir código externo de forma sencilla.
- Interés del autor en aumentar su conocimiento en este framework.
- Hay librería para ofrecer compatibilidad con HomeKit.

2.5.3.2.2 Inconvenientes

- Muchas formas de programar, no hay una guía como pueden ser las Java Code Coventions.
- El autor no posee un conocimiento tan avanzado como en Java.

2.5.3.3 *Alternativa seleccionada: Lenguaje de programación Raspberry Pi*

Debido que una de las ideas fundamentales del sistema es permitir añadir compatibilidad con nuevos sensores y actuadores, se ha elegido como lenguaje de programación en la Raspberry Pi, NodeJS ya que facilita en gran medida el cumplimiento de este objetivo.

2.5.4 Sistema de gestión de base de datos

Como en todo proyecto se necesita alguna forma de almacenar los datos que maneje el sistema sobre los usuarios, sensores y actuadores, para ello se utilizan las bases de datos. Para realizar este proyecto se han buscado las mejores alternativas, tanto SQL como NoSQL.

2.5.4.1 *Firebase*



Ilustración 17. Logotipo Firebase

Firebase es un producto de Google que ofrece un conjunto de herramientas para facilitar el desarrollo de aplicaciones, tanto móviles como web.

Entre estas herramientas se encuentran la base de datos en tiempo real, que es la que se analiza en este punto, pero también otras como sistema de envío de notificaciones push, hosting...

2.5.4.1.1 *Ventajas*

- Plan gratuito, con limitaciones.
- Base de datos en tiempo real.
- Seguridad.
- Librerías para su uso multiplataforma.

2.5.4.1.2 *Inconvenientes*

- Es un sistema no relacional, por lo que de entrada puede resultar confuso.

2.5.4.2 *PostgreSQL*

PostgreSQL es un sistema de gestión de bases de datos relacional libre y gratuito, cuyo desarrollo lo realiza una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por diferentes organizaciones.

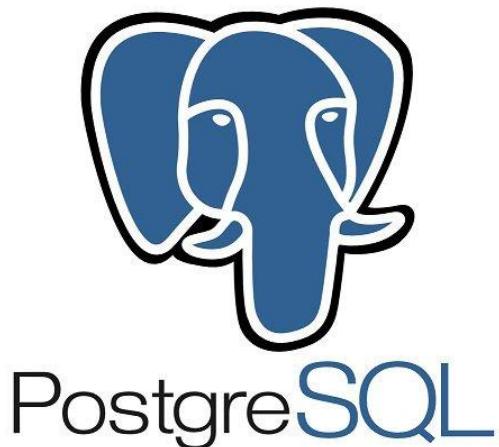


Ilustración 18. Logotipo PostgreSQL

2.5.4.2.1 Ventajas

- Totalmente gratuito.
- Sistema relacional.
- Librerías para su uso multiplataforma.
- Alta concurrencia.
- Multitud de tipos nativos: texto ilimitado, arrays, direcciones IPs.

2.5.4.2.2 Inconvenientes

- Habría que mantener un servidor con acceso desde Internet, con las implicaciones a nivel de seguridad que eso implicaría.

2.5.4.3 MySQL

Al igual que PostgreSQL, MySQL es un sistema de gestión de bases de datos relacional desarrollado por Oracle.



Ilustración 19. Logotipo MySQL

MySQL es considerado uno de los proyectos de bases de datos más populares del mundo, por ello es utilizado en multitud de proyectos.

2.5.4.3.1 Ventajas

- Gratis.
- Sistema de base de datos relacional.
- Librerías de uso multiplataforma.
- Mantenido por una gran empresa como es Oracle.
- Posibilidad de elegir diferentes motores de almacenamiento.
- Es uno de los más utilizados.

2.5.4.3.2 Inconvenientes

- La necesidad de gestionarlo en un servidor conectado a Internet, lo que puede provocar un fallo de seguridad si está mal configurado.

2.5.4.4 Alternativa seleccionada: Sistema de gestión de base de datos

Por motivos principalmente de seguridad se ha decidido utilizar Firebase como sistema de gestión de bases de datos. Debido a que Android no pose sistema de domótica integrado y hay que se quiere garantizar el acceso al sistema desde fuera de red local de la casa, es la mejor opción.

Otro punto a favor de Firebase es su sistema de base de datos en tiempo real, ya que utilizando sus librerías para los distintos dispositivos y lenguajes se mantiene todos los dispositivos sincronizados con el mínimo esfuerzo.

Capítulo 3. Aspectos Teóricos

En esta sección se describen brevemente aquellos conceptos, herramientas y tecnología empleadas para la realización de este proyecto.

3.1 Internet de las Cosas o IoT

Internet de las Cosas o *IoT* (en siglas en inglés) es un término propuesto por Kevin Ashton en el Auto-ID Center del MIT en 1999 que hace referencia a la interconexión de objetos cotidianos, como puede ser la nevera, la puerta del garaje..., con Internet.

Actualmente estamos dentro de una fase de inicial en la que cada empresa trata de que su estándar o visión se imponga sobre el resto. Hablando de números en 2017 se estima que habrá 8,4 mil millones de dispositivos IoT conectados, lo que significa un 31% más que en 2016.

3.2 DiY o Do it Yourself

DiY (Do it Yourself) o es español hazlo tú mismo hace referencia a un producto o servicio que tú mismo fábricas, ya sea para ahorrar dinero, aprender, entretenerte o un poco de todo lo anterior.

Gracias a internet y el bajo coste de microcontroladores, sensores y actuadores si nos interesamos por algo como IoT, encontramos varios páginas como hackter.io o instructables.com donde cualquiera puede subir sus proyectos, explicando paso a paso como realizarlo y compartirlos con el resto de comunidad.

3.3 Notificaciones push

Las notificaciones push son mensajes que se envían desde un servidor directamente a un dispositivo móvil, estos mensajes suelen ser utilizados para generar notificaciones en el dispositivo móvil que lo recibe.

En la actualidad son usadas por miles de aplicaciones como WhatsApp o Telegram para notificarnos de que algún contacto nos ha enviado un mensaje o por la aplicación de nuestro periódico favorito para enviarnos una notificación con las noticias de última hora.

3.4 Firebase

Firebase es una plataforma de Google que ofrece a los desarrolladores de aplicaciones web y móviles una serie de servicios con posibilidad de escalarlos en función del uso que se haga de ellos con diferentes planes de precios, empezando por uno gratuito.

Entre los servicios que ofrece Firebase destacan:

- **Analíticas**, nos da todo tipo de información sobre los usuarios que utilizan nuestra aplicación: tiempo de uso, datos demográficos..., además permite configurar eventos personalizados para entender mejor como nuestros usuarios utilizan nuestra aplicación.
- **Cloud Functions**, sin tener un servidor Firebase ofrece la opción de ejecutar código que realice operaciones complejas.
- **Hosting**, para subir nuestra página web o almacenar archivos de los usuarios de nuestra aplicación.
- **Remote Config**, permite modificar partes de nuestra aplicación sin necesidad de lanzar una nueva versión.
- **App Indexing**, al igual que ocurre con las páginas webs, se puede indexar el contenido de nuestra aplicación para aparecer en las búsquedas de Google.
- **AdMob**, la plataforma de anuncios para móviles de Google tiene integración con las analíticas de Firebase lo que permite a los desarrolladores aumentar sus ingresos con mejores campañas de anuncios.
- **Crash Reporting**, cada vez que ocurre un error en nuestra aplicación Firebase nos proporciona un informe con multitud de información sobre ese fallo.

A continuación, se detallan en profundidad aquellos servicios utilizados en este proyecto.

3.4.1 Base de datos en tiempo real

Firebase ofrece una base de datos orientada a objetos, es decir, no tiene un esquema predefinido y se puede guardar en ella cualquier tipo de objeto.

Además, esta base de datos ofrece la posibilidad a la hora de realizar una consulta de suscribirse a un evento de forma que cualquier cambio que se realice en la base de datos sea notificado a todos los suscriptores del evento.

3.4.2 Firebase Cloud Messaging

Antiguamente conocido como Google Cloud Messaging, es un servicio que permite enviar notificaciones push de forma ilimitadas y gratuita, que funciona tanto en iOS como en Android.

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

4.1 Planificación Inicial

En este apartado se presenta la planificación inicial del proyecto. Para realizar esta planificación se ha tenido en cuenta la documentación que se tiene que realizar previamente al desarrollo del sistema y posteriormente al desarrollo.

Se ha realizado utilizando la herramienta Microsoft Project, que permite obtener diferentes tipos de diagramas e información de la planificación realizada.

En Capítulo 11 se incluye la planificación final del proyecto con la que se comparara esta planificación inicial del proyecto, ya que seguramente factores externos al proyecto han sufrido retrasos y cambios en la misma.

La duración prevista para el proyecto es de 360 horas empezando el 1 de mayo de 2017 y terminando el 3 de julio de 2017, trabajando 3 horas de lunes a viernes para poder compatibilizar este proyecto con las clases y las prácticas en empresa.

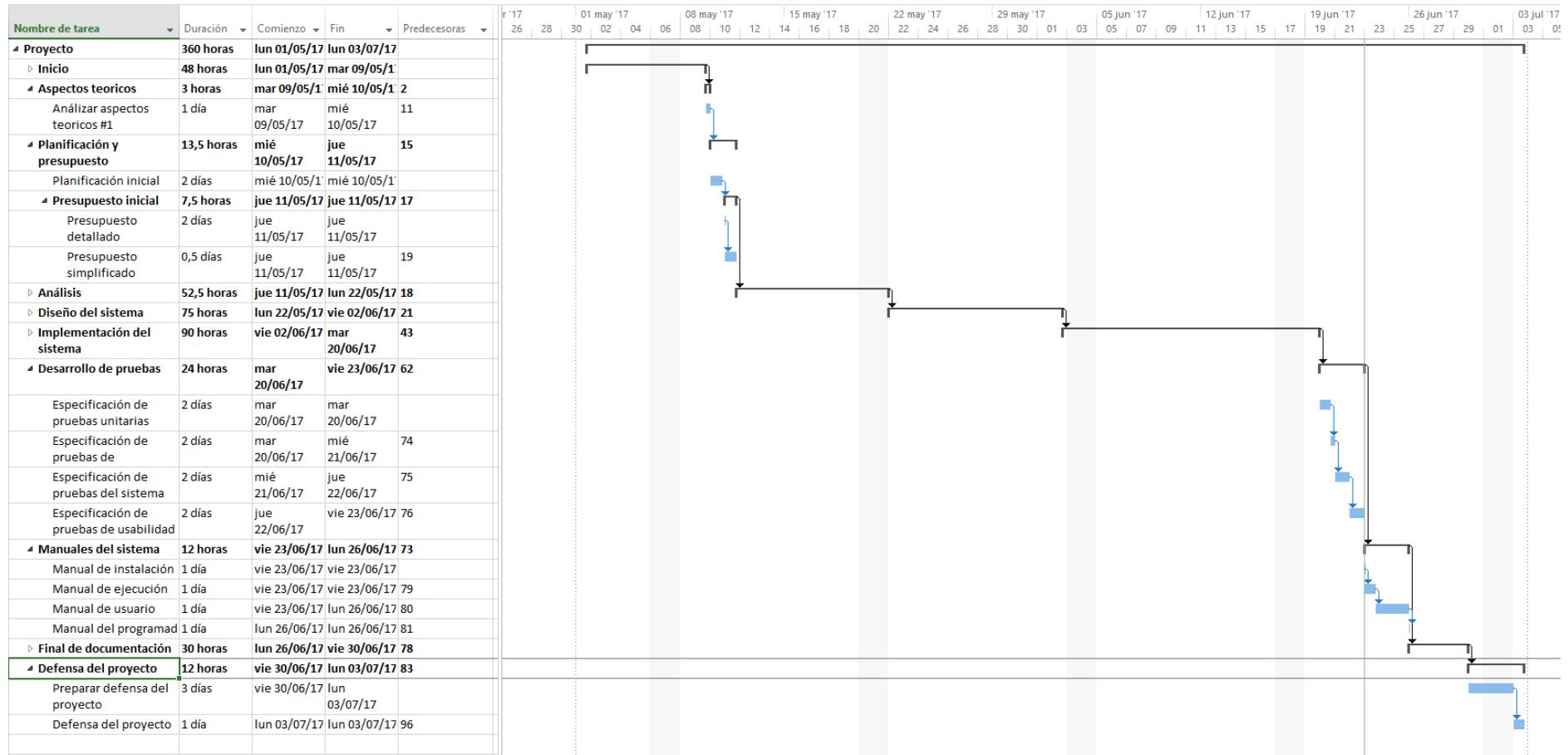


Ilustración 20. Vista general de la planificación del proyecto

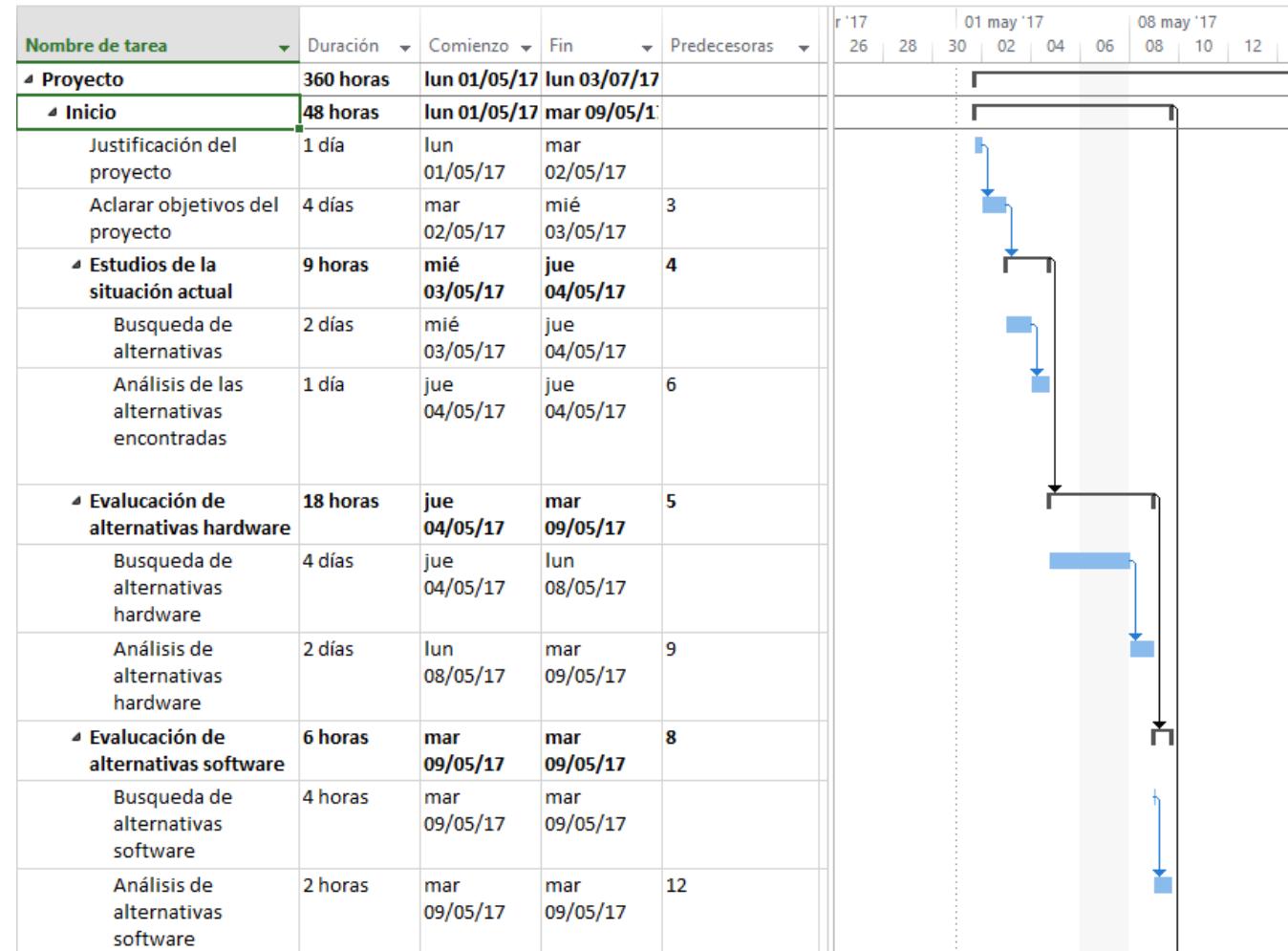


Ilustración 21. Planificación tarea: Inicio

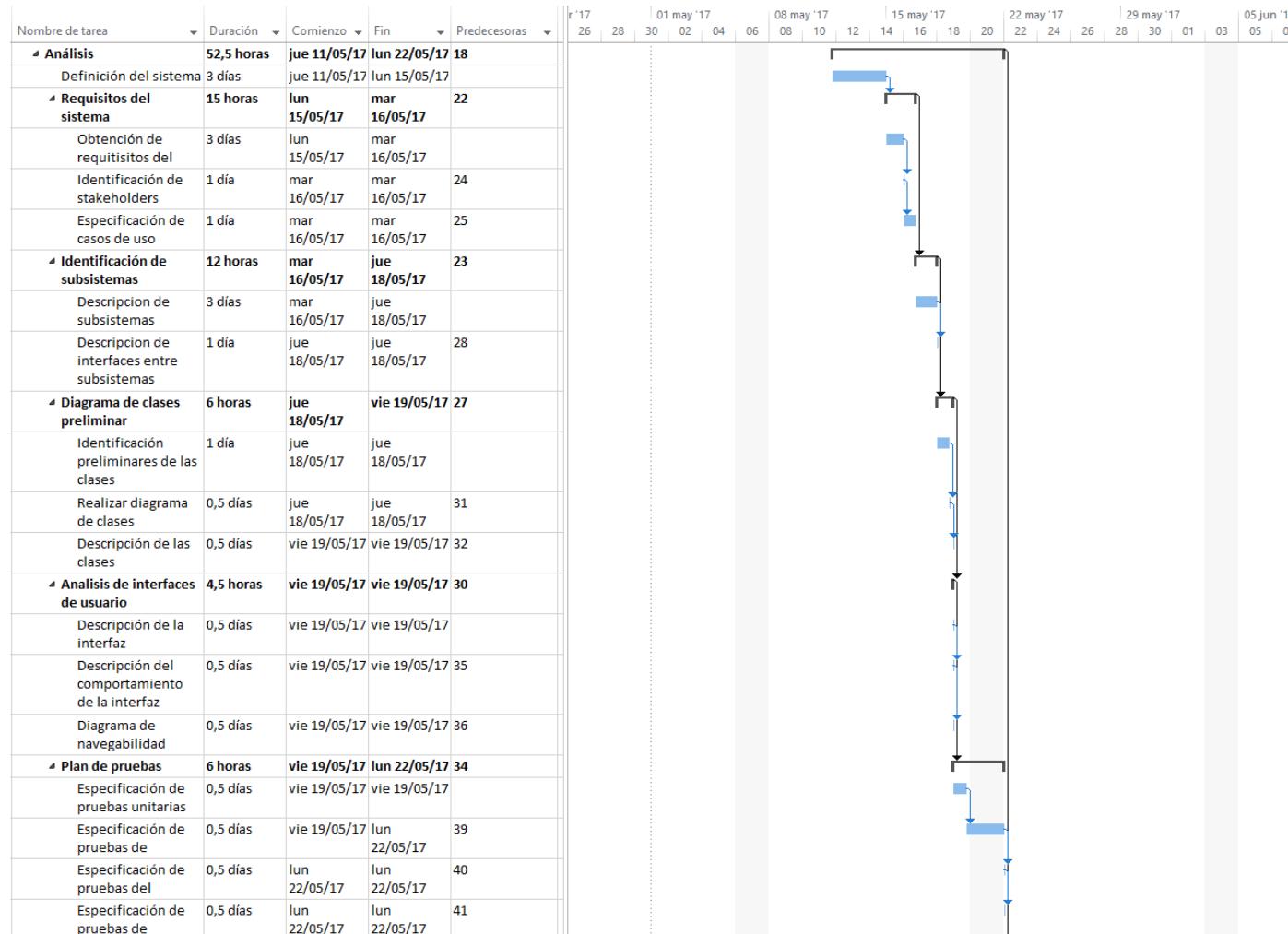


Ilustración 22. Planificación tarea: Análisis

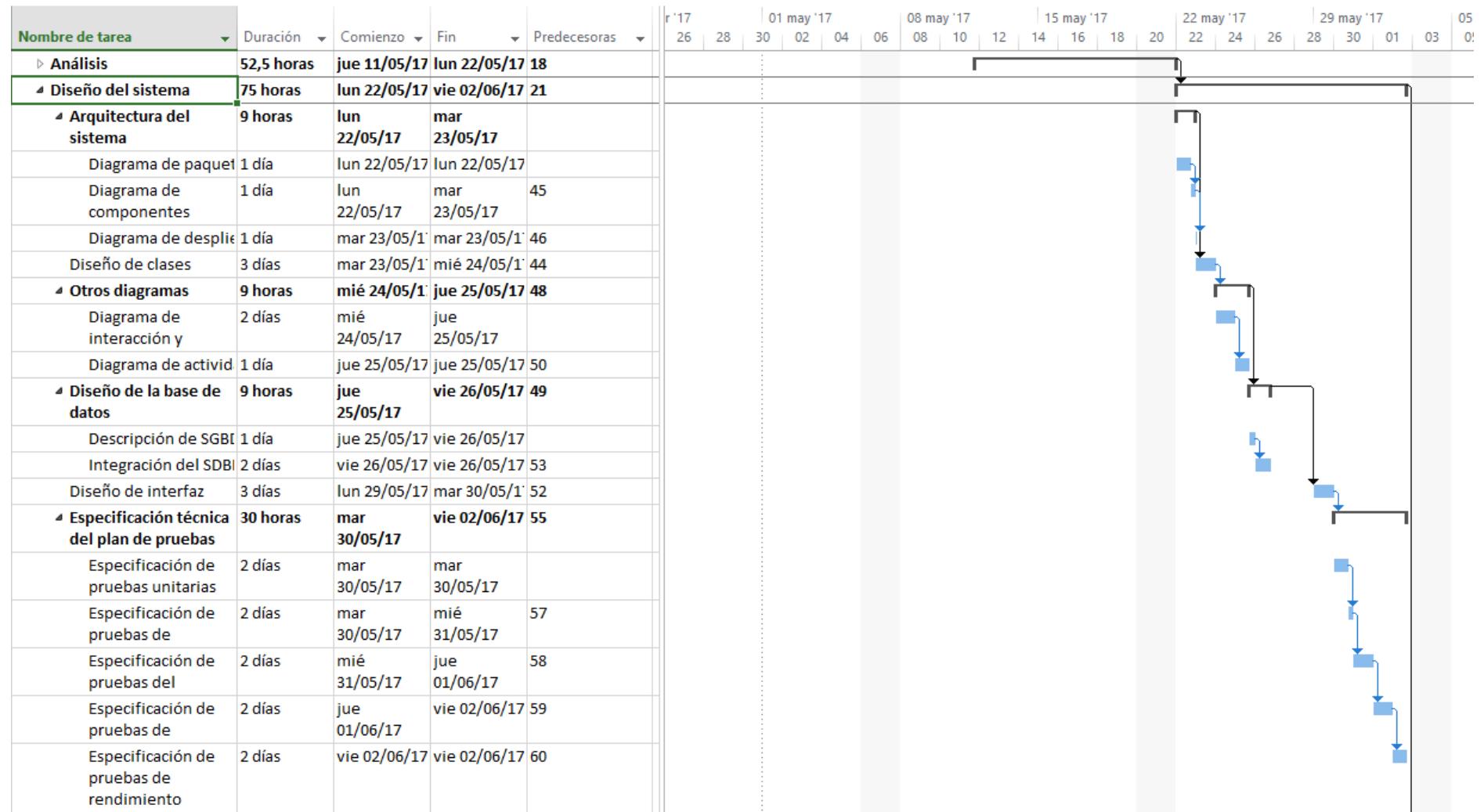


Ilustración 23. Planificación tarea: Diseño del sistema

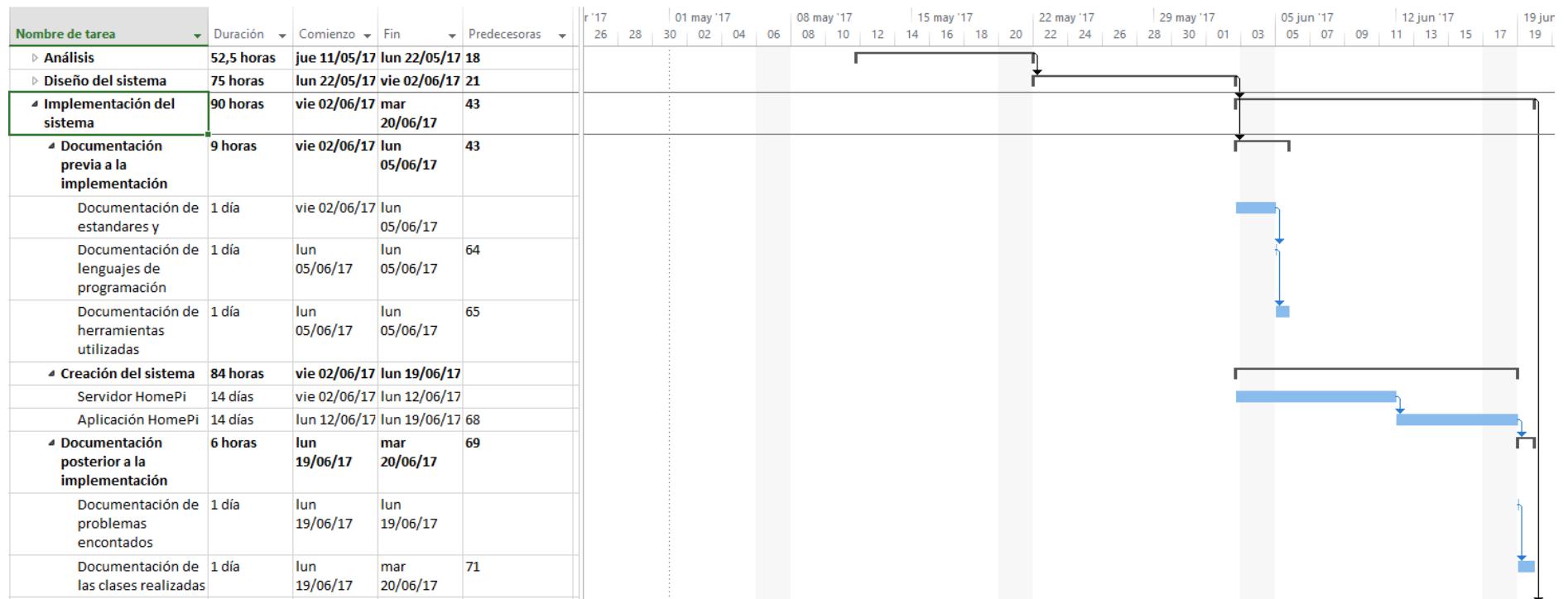


Ilustración 24. Planificación tarea: Implementación del sistema

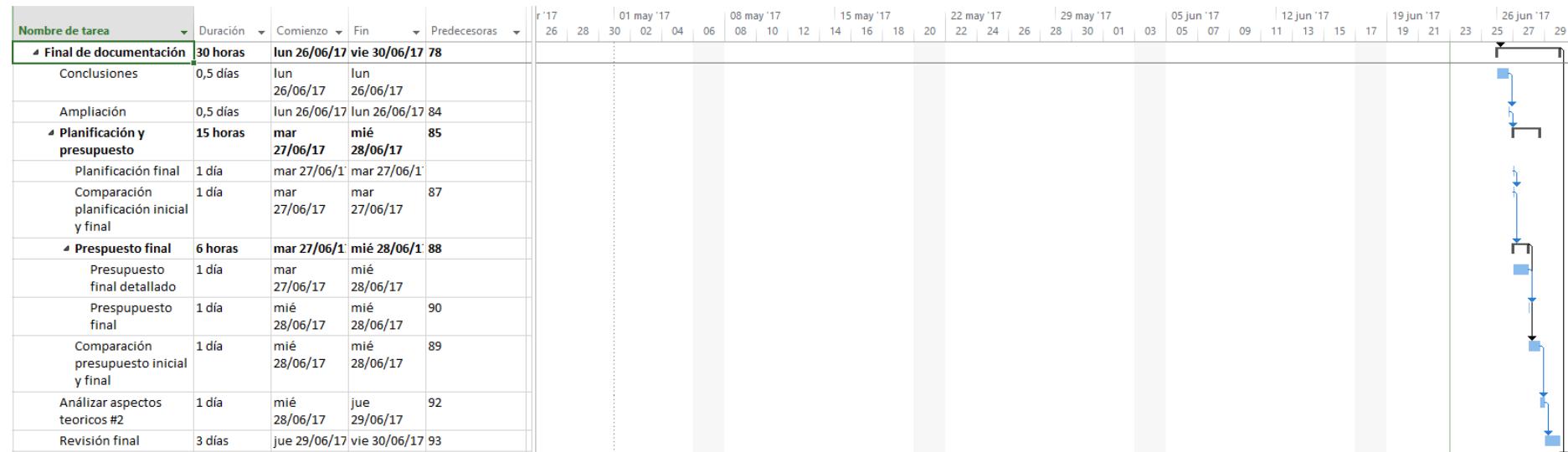


Ilustración 25. Planificación tarea: Final de documentación

4.2 Presupuesto Inicial

Al igual que en el caso de la planificación en el Capítulo 11 se incluye el presupuesto final del proyecto y se compara con este presupuesto inicial.

El tipo de trabajo que recoge este presupuesto son 360 horas y contempla la realización de toda la documentación y el desarrollo del proyecto en si.

4.2.1 Desarrollo de Presupuesto Detallado (Empresa)

Este presupuesto se realiza a nivel interno de la empresa para tener una idea clara de donde se realizan los gastos del proyecto. Se incluyen los costes de las diferentes tareas de la planificación, hardware, software, materiales, etc.

Dado que este proyecto va a ser desarrollado en su totalidad por una sola persona, el autor de este documento solo se considera un precio por hora.

Tarea	Horas	Precio/Hora	Total
Inicio	48,00	50,00 €	2.400,00 €
Aspectos teoricos	3,00	50,00 €	150,00 €
Planificación y presupuesto	13,50	50,00 €	675,00 €
Análisis	52,50	50,00 €	2.625,00 €
Diseño	75,00	50,00 €	3.750,00 €
Implementación	90,00	50,00 €	4.500,00 €
Pruebas	24,00	50,00 €	1.200,00 €
Manuales	12,00	50,00 €	600,00 €
Fin del proyecto	42,00	50,00 €	2.100,00 €
Total horas:	360,00	Total precio:	18.000,00 €

Tabla 4-1. Coste detallado por tareas

En la tabla anterior se encuentran todas las tareas de máximo nivel en las que se ha dividido el proyecto en la planificación.

Dado que inicialmente se ha planificado el proyecto para solo trabajar de lunes a viernes, sin incluir festivos, el precio por hora no sufre ningún cambio en ninguna de las tareas.

Hardware	Precio unitario	Cantidad	Vida útil (meses)	Coste para el proyecto
Portátil	1.500,00 €	1	60	50,00 €
Raspberry Pi 3	40,00 €	1		40,00 €
Wemos D1 mini pro	5,01 €	3		15,03 €
Sensor temperatura DHT11	2,28 €	1		2,28 €
Rele	1,92 €	1		1,92 €
Sensor luminosidad	4,50 €	1		4,50 €
Sensor distancia	4,50 €	1		4,50 €
			Total precio:	118,23 €

Tabla 4-2. Coste detallado hardware

En la tabla anterior se ha calculado el precio del hardware necesario para el proyecto. En el caso de que el hardware se use para más de un proyecto en la columna vida útil aparecerá un número que servirá para calcular el coste para el proyecto teniendo en cuenta que este dura solo dos meses. Si no aparece nada en esta columna es que el hardware se entrega con el proyecto.

Software	Precio unitario	Cantidad	Vida útil (meses)	Coste para el proyecto
Firebase	0,00 €	1		0,00 €
macOS Sierra	0,00 €	1		0,00 €
Microsoft Windows 10 Pro	279,00 €	1	30	18,60 €
Microsoft Office 360	69,00 €	1	12	11,50 €
Microsoft Project	119,00 €	1	30	7,93 €
Raspbian	0,00 €	1		0,00 €
Google Drive	14,00 €	1	2	14,00 €
GitHub	0,00 €	1		0,00 €
WebStorm	59,00 €	1	12	9,83 €
Arduino IDE	0,00 €	1		0,00 €
VisualStudio Code	0,00 €	1		0,00 €
Android IDE	0,00 €	1		0,00 €
			Total precio:	61,87 €

Tabla 4-3. Coste detallado software

En la tabla anterior se detallan los costes del software necesario para realizar este proyecto, al igual que la tabla del hardware, si en la columna de vida no aparece nada es que ese software es necesario para el proyecto o que es gratuito.

Gastos mensuales fijos	Meses	Precio/Mes	Total
Conexión a Internet	2	60,00 €	120,00 €
Electricidad	2	150,00 €	300,00 €
Calefacción	2	50,00 €	100,00 €
Agua	2	50,00 €	100,00 €
Cuota de autónomo	2	50,00 €	100,00 €
		Total precio:	720,00 €

Tabla 4-4. Gastos mensuales fijos

Se añade también un desglose de los gastos mensuales fijos para el desarrollo de este proyecto: como son: conexión a internet, electricidad, etc. La persona que desarrolla el proyecto se da de alta como autónomo para poder recibir el pago del proyecto de forma legal, por lo que las cuotas necesarias también se incluyen en el presupuesto del proyecto.

Subtotal	18.900,10 €
Beneficio (10%)	1.890,01 €
IVA (21%)	4.365,92 €
Total	25.156,03 €

Tabla 4-5. Presupuesto total del proyecto

Por último, se incluye el coste total del proyecto considerando un beneficio del 10% y aplicando el IVA correspondiente, lo que cada un total de **25.156,03€**.

4.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

En esta sección se incluye el presupuesto del cliente, reducido a 3 partidas: Desarrollo que incluye los costes del desarrollo del proyecto, los gatos mensuales y los beneficios del proyecto, el coste del software externo que se necesita para el proyecto y por último el hardware, el cual será entregado con el proyecto.

Concepto	Coste Total Concepto
Desarrollo	20.610,01 €
Software y licencias	61,87 €
Hardware	118,23 €
<i>Subtotal</i>	20.790,11 €
<i>IVA (21%)</i>	4.365,92 €
TOTAL	25.156,03 €

Tabla 4-6. Presupuesto del cliente

Capítulo 5. Análisis

Este apartado contiene toda la especificación de requisitos y toda la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

Como se menciona en la introducción la idea esencial es construir un sistema de domótica que sea seguro y fácil de expandir y usar.

5.1.1 Determinación del Alcance del Sistema

Este proyecto tiene como objetivo el desarrollo de un sistema que permita controlar todo tipo de accesorios mediante IoT que se pueden considerar dentro de un sistema de domótica.

Las elecciones en temas de tecnologías y diseño se han realizado con dos puntos fundamentales en mente:

El primero la seguridad, es una parte esencial del proyecto ya que va a permitir controlar la casa de los usuarios, por lo que se tiene que garantizar, en la medida de lo posible, que nadie que no tenga acceso va a poder controlar el sistema.

Por este motivo se toma la decisión de utilizar varios de los servicios de Firebase, ya que el Plan Spark es gratuito e incluye:

- 100 conexiones simultaneas a la base de datos en tiempo real.
- 10GB de transferencia mensual.
- Notificaciones push ilimitadas.
- Funciona bajo HTTPS.
- Entre otras cosas.

El segundo que sea sencillo de instalar, modificar y expandir para ello por una parte se utilizan componentes baratos y en lo máximo de lo posible con un gran soporte por parte de la comunidad como pueden ser Raspberry Pi y Arduino.

También este es otro de los motivos por lo que se ha decidido utilizar Firebase por que nos permite iniciar la comunicación desde dentro de la red local del usuario a internet, sin necesidad de abrir puertos o configuración adicional.

5.1.1.1 Elementos desarrollados

HomePi Server, totalmente multiplataforma diseñado y programado para funcionar en Mac, Windows y Linux, aunque lo ideal sería una Raspberry Pi ya que tiene que estar funcionando para permitir interactuar con los distintos accesorios.

También es el encargado de reaccionar a los cambios en la base de datos en tiempo real de Firebase, actualizando con las lecturas de los sensores o activando/desactivando algún actuador.

En esta primera versión se han creado implementaciones de sensores y actuadores que funcionan con llamadas HTTP, pensados para funcionar con las placas Wemos, que gracias a su pequeño tamaño se pueden colocar sin problema en cualquier parte de la casa y gracias a que integran WiFi pueden comunicarse con **HomePi Server** mediante la red local.

Se opta por la implementación mediante HTTP ya que también permitiría compatibilidad con plataformas como <https://thingspeak.com/> que permiten recolectar datos para su posterior análisis.

En cuanto a accesorios se han desarrollado: Control de luces, puerta de garaje, termostato y lectura de datos de sensores de temperatura y humedad.

HomePi Android App es una aplicación móvil para Android, permite configurar con que aplicación Firebase va a interactuar desde la propia aplicación.

Permite la creación, configuración y borrado de los distintos accesorios disponibles, también realizar las acciones disponibles en función del accesorio.

Tanto **HomePi Server** como **HomePi Android**, serán diseñados de forma que permitan a todos los usuarios utilizar el sistema de forma simultánea, permitiendo activar y desactivar las notificaciones cuando otro usuario realiza algún cambio en un accesorio.

HomePi Wemos, software que se ha desarrollado el software para permitir a las placas Wemos, mediante peticiones HTTP, leer datos de un sensor de temperatura y humedad DHT11 o DTH22, activar y desactivar un rele, leer un sensor de luminosidad para comprobar el estado de las luces y uno de distancia para el estado de la puerta del garaje.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Requisitos funcionales

Como el proyecto va a tener como resultado varios subsistemas se van a tratar los requisitos de cada uno de ellos de forma separada.

5.2.1.1.1 HomePi Server

Código	Nombre Requisito	Descripción del Requisito
RS1	Configurar Firebase	Se debe permitir al usuario configurar los datos del servidor de firebase que va a utilizar para HomePi mediante el archivo de configuración proporcionado por firebase.
RS2	Envío de notificaciones push	El servidor tiene que poder enviar notificaciones push a los usuarios mediante Firebase Cloud Messaging
RS3	Compatibilidad con las placas Wemos	El sistema tiene que ofrecer compatibilidad con los sensores y actuadores instalados en las placas wemos que se especifican en los requisitos RW3 al RW6 mediante llamadas HTTP.
RS4	Creación de accesorios	El servidor tiene que ser capaz de crear los accesorios dinámicamente atendiendo a los cambios en la base de datos de firebase.
RS5	Interactuar con los sensores y actuadores	El servidor tiene que ser capaz de interactuar con los sensores y actuadores que componen los accesorios respondiendo a los cambios que se produzcan en la base de datos de firebase.

5.2.1.1.2 HomePi Android App

Código	Nombre Requisito	Descripción del Requisito
RA1	Configurar Firebase	Se debe permitir al usuario configurar los datos del servidor de firebase que va a utilizar para HomePi
RA2	Inicio de sesión	Se debe permitir al usuario iniciar sesión con su correo electrónico y contraseña.
RA3	Recuperación de contraseña	Se debe permitir al usuario recuperar la contraseña a través de su correo electrónico.
RA4	Accesos disponibles	
RA4.1	Acceso “Garaje”	Permitirá abrir y cerrar una puerta de garaje.
RA4.2	Acceso “Luz”	Permitirá encender y apagar una luz, lámpara...
RA4.3	Acceso “Temperatura”	Permitirá obtener el valor de temperatura de un sensor de temperatura.
RA4.4	Acceso “Humedad”	Permitirá obtener el valor de humedad de un sensor de humedad.

RA4.5	Accesorio “Termostato”	Permitirá fijar una temperatura objetivo, encender y apagar la calefacción.
RA5	Creación de accesorios	Se debe permitir al usuario crear accesorios, especificados en RA4.
RA6	Modificación de accesorios	Se debe permitir al usuario cambiar la configuración de los creados accesorios, especificados en RA4.
RA7	Borrado de accesorios	Se debe permitir al usuario borrar los accesorios creados previamente. Accesos específicos en RA4.
RA8	Múltiples usuarios simultáneos	Se debe permitir interactuar con todos los accesorios disponibles a múltiples usuarios simultáneos.
RA9	Notificaciones push	La aplicación debe permitir la recepción de notificaciones push enviadas mediante Firebase Cloud Messaging
RA9.1	Configuración de notificaciones	Se debe permitir que el usuario elija para qué accesorios quiere recibir notificaciones.

5.2.1.1.3 HomePi Wemos

Código	Nombre Requisito	Descripción del Requisito
RW1	Conexión con la red Wi-Fi	Se debe permitir conectar la placa wemos con la red wifi.
RW2	Servidor HTTP	Se debe crear un servidor capaz de contestar a peticiones HTTP.
RW3	Compatibilidad con sensor de temperatura y humedad.	Se debe permitir leer los datos de un sensor de temperatura tipo DHT11 y DHT22 y devolverlos en una respuesta HTTP.
RW4	Compatibilidad con rele	Se debe permitir controlar un rele con dos peticiones HTTP, una para encender el rele y otra para apagarlo.
RW5	Compatibilidad con sensor de luminosidad	Se debe permitir leer los datos de un sensor de luminosidad y devolverlos en una respuesta HTTP.
RW6	Compatibilidad con sensor de distancia	Se debe permitir leer los datos de un sensor de distancia y devolverlos en una respuesta HTTP.

5.2.1.2 Requisitos no funcionales

Código	Nombre Requisito	Descripción del Requisito
RN1	Seguridad	Se debe garantizar que ningún usuario no autorizado podrá acceder al sistema y manipular los accesorios.
RN2	Facilidad de instalación	Se debe desarrollar un sistema sencillo de instalar que además esté acompañado de un buen manual de instalación.

RN3	Facilidad de expansión	Se debe proporcionar una estructura de clase e interfaces que permitan la creación de nuevos accesorios, sensores y actuadores.
RN4	Cuenta de Google	El sistema requiere de una cuenta de Google para poder crear el servidor en Firebase.
RN5	Conexión a internet	Tanto HomePi Server como HomePi Android necesitan una conexión a internet para poder conectarse con Firebase.
RN6	Disponibilidad	El sistema tiene que estar disponible 24h 7 días a la semana.

5.2.2 Identificación de Actores del Sistema

Por un lado, en sistema tenemos al usuario/s que lo utilizan mediante la aplicación para Android, por otro lado, al servidor **HomePi Server**, que está pensado para ser ejecutado en un Raspberry Pi, pero que es compatible con Mac, Windows y Linux.

También tenemos las distintas placas Wemos repartidas por la casa a las que **HomePi Server** se conecta para interactuar con los sensores y actuadores.

Por último, tenemos a Firebase que sirve de puente de comunicación entre la aplicación de Android y el **HomePi Server**.

5.2.3 Especificación de Casos de Uso

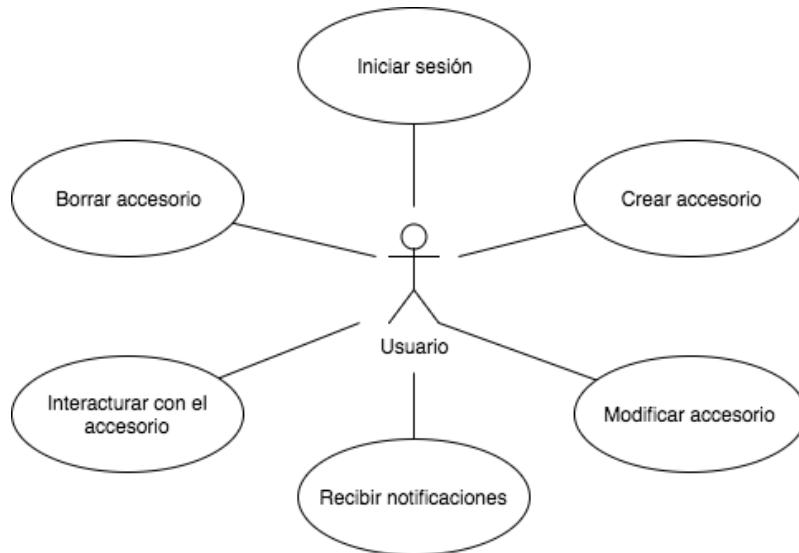


Ilustración 5.1. Casos de uso

<u>Nombre del Caso de Uso</u>
Iniciar sesión
Descripción

Cuando el usuario inicie la aplicación deberá introducir el usuario y contraseña creados mediante Firebase para poder visualizar la lista de accesorios disponibles.

Nombre del Caso de Uso
Crear accesorio

Descripción

Desde la lista de accesorios en cualquier momento el usuario podrá dar de alta un nuevo accesorio, de cualquiera de los tipos disponibles, añadiendo toda la configuración necesaria para que este funcione correctamente.

Nombre del Caso de Uso
Modificar accesorio

Descripción

En cualquier momento el usuario podrá cambiar cualquiera de los parámetros de configuración de cualquiera de los accesorios que haya creado previamente, por ejemplo, para corregir errores.

Nombre del Caso de Uso
Borrar accesorio

Descripción

En cualquier momento el usuario podrá borrar cualquiera de los accesorios creados previamente.

Nombre del Caso de Uso
Interactuar con el accesorio

Descripción

En la lista de accesorios el usuario podrá interactuar con cualquiera de los accesorios creados, se entiende por interactuar: actualizar el valor de un sensor, activar/desactivar la calefacción, ...

Nombre del Caso de Uso
Recibir notificaciones

Descripción

En cualquier momento el usuario podrá recibir en su aplicación notificaciones que informen del cambio en accesorio, por ejemplo, otro usuario ha abierto la puerta del garaje, o de algún error al realizar una acción, por ejemplo, se ha intentado abrir la puerta pero no se ha abierto.

5.3 Identificación de los Subsistemas en la Fase de Análisis

El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.

5.3.1 Descripción de los Subsistemas

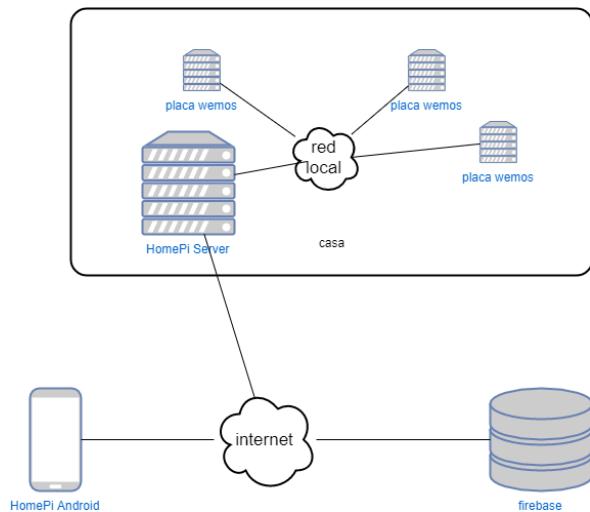


Ilustración 2. Subsistemas HomePi

5.3.1.1 Subsistemas HomePi Server

Este subsistema engloba todo el software necesario para la comunicación entre Firebase y los diferentes sensores y actuadores desplegados en la red local de la casa donde se realice la instalación. Dentro de este subsistema podemos encontrar los siguientes módulos:

Módulo de actuadores. Este subsistema será el encargado de transmitir las órdenes a los distintos accesorios compatibles con HomePi.

Módulo de sensores. Este subsistema será el encargado de leer la información de los sensores compatibles con HomePi.

Módulo de accesorios. Este subsistema será el encargado de reaccionar a los cambios, en la base de datos de Firebase, realizados en los distintos accesorios creados. Cada uno de los accesorios está formado por distintos sensores o actuadores.

5.3.1.2 Subsistemas HomePi Android

Este subsistema engloba la aplicación Android creada para permitir al usuario conectarse con Firebase para controlar, configurar y borrar los accesorios que tenga creados. Podríamos descomponer este subsistema en los siguientes módulos:

Módulo de accesorios. Este subsistema será el encargado de reaccionar a los cambios, en la base de datos de Firebase, realizados en los distintos accesorios creados.

Módulo de configuración. Este subsistema será el encargado de realizar cambios en la configuración de los accesorios ya creados o en la creación de nuevos accesorios.

Módulo de notificaciones. Este subsistema será el encargado de mostrar al usuario las notificaciones recibidas mediante Firebase Cloud Messaging.

5.3.2 Descripción de los Interfaces entre Subsistemas

Entre diferentes sistemas, es decir, entre **HomePi Server** y **HomePi Android** las comunicaciones se realizan mediante cambios en la base de datos proporcionada por Firebase.

Entre **HomePi Server** y los diferentes sensores y actuadores la comunicación depende de la implementación que se realiza de las interfaces. En el caso de la implementación que acompaña al proyecto realizado estas comunicaciones se realizan mediante peticiones HTTP en la red local de la casa donde está instalado.

5.4 Diagrama de Clases Preliminar del Análisis

En esta sección se mostrar un diagrama preliminar de clases, así como la explicación de cada una de ellas. Este diagrama servirá como base para la fase de diseño.

5.4.1 Diagrama de Clases

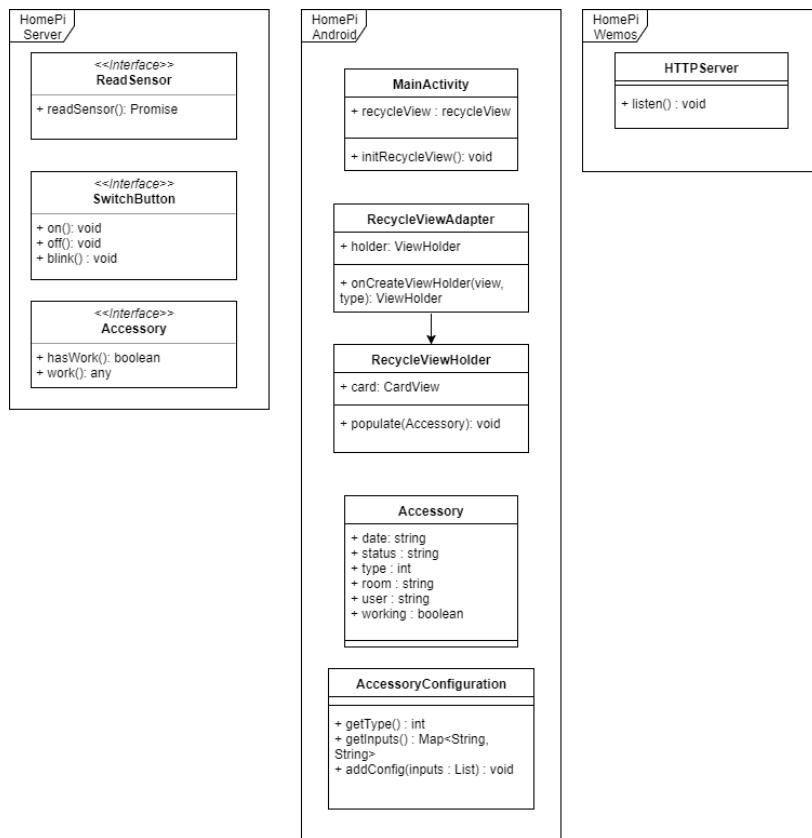


Ilustración 3. Diagrama de clases preliminar

5.4.2 Descripción de las Clases

Las clases deberían estar organizadas por los subsistemas identificados en anteriormente, rellenando una tabla como la siguiente por cada clase del mismo:

5.4.2.1 HomePi Server

<u>Nombre de la Interfaz</u>
ReadSensor
Descripción
Interfaz con un método que permite obtener los datos de un sensor.
Responsabilidades
Obtener los datos de un sensor
Métodos Propuestos

readSensor: este método será el que cualquier sensor compatible con HomePi tendrá que redefinir para devolver los datos del sensor que se ha leído. Para permitir un mejor funcionamiento concurrente se devolverá un Promise asíncrona.

Nombre de la Interfaz
SwitchButton
Descripción
Interfaz con los métodos necesarios para controlar un botón.
Responsabilidades
Transmitir las instrucciones al botón.
Métodos Propuestos
on: enciende el botón
off: apaga el botón
blink: enciende y apaga el botón rápido.

5.4.2.2 HomePi Android

Nombre de la Clase
MainActivity
Descripción
Activity principal para mostrar la información de los accesorios
Responsabilidades
Mostrar una lista con la información de los distintos accesorios.
Atributos Propuestos
recycleView: componente Android para mostrar una lista de tarjetas con la información de cada accesorio
Métodos Propuestos
initRecyclerView: método que inicializara el recycleView con la información de firebase

Nombre de la Clase
RecycleViewAdapter
Descripción
Adatador para mostrar información en el recycleView
Responsabilidades
Adaptar la información para construir el ViewHolder correcto para cada accesorio.
Métodos Propuestos
onCreateViewHolder: método que según el tipo del accesorio creara el ViewHolder correspondiente.

Nombre de la Clase
RecycleViewHolder
Descripción
Clase que tiene los datos del accesorio para mostrarlos al usuario.
Responsabilidades
Asignar los datos del accesorio a los elementos de la interfaz correspondiente.
Métodos Propuestos
populate: método asigna los datos del accesorio al layout de android.

Nombre de la Clase
Accessory
Descripción
Clase con los datos de los accesorios obtenidos de Firebase.
Responsabilidades
Servir como contenedor para los datos de Firebase.
Atributos Propuestos
date: fecha de la última interacción con el accesorio. Status: información del estado del accesorio. Type: información sobre el tipo de accesorio. Room: información de la habitación en la que se encuentra el accesorio. User: información del ultimo usuario que ha realizado una interacción con el accesorio. Working: información sobre si el servidor está trabajando actualmente en el accesorio.

Nombre de la Clase
AccessoryConfiguration
Descripción
Clase que permite configurar un accesorio.
Responsabilidades
Informar a la vista que campos se tiene que crear para configurar el accesorio y transmitir la información introducida en los campos a Firebase.
Métodos Propuestos
getType: método para obtener el del elemento del accesorio que se va a crear. getInputs: método que devuelve un map con el place holder y el valor actual del input para la configuración addConfig: método que recibe el valor de los inputs y los guarda o actualiza en Firebase para cambiar la configuración.

5.4.2.3 HomePi Wemos

Nombre de la Clase
HTTPServer
Descripción
Clase encargada de crear un servidor HTTP accesible desde la red local.
Responsabilidades
Responder a las peticiones HTTP de HomePi Server.
Métodos Propuestos
listen: método encargado de recibir y procesar las peticiones HTTP

5.5 Análisis de Interfaces de Usuario

Dado que de los distintos subsistemas el único que va a tener una interfaz va a ser la aplicación Android en este apartado únicamente se va a tratar ese tema.

5.5.1 Descripción de la Interfaz

5.5.1.1 Pantalla inicial



Ilustración 4. Interfaz pantalla inicial Android

En esta pantalla se mostrará la información de los diferentes accesorios creados previamente, por ejemplo, en el diseño vemos los sensores de temperatura y humedad, la puerta del garaje y una luz del salón.

La información mostrada dependerá del accesorio, pero siempre habrá una información común que será la habitación en la que se encuentra el accesorio y la fecha a la que se realiza el último cambio.

Cada accesorio tendrá uno o varios botones que permitirán interactuar con él. Por ejemplo, los sensores tendrán un botón para obtener el valor actual de temperatura, humedad..., en el caso de la puerta del garaje el botón servirá para abrir o cerrar la puerta, en función del estado actual, en el caso de las luces el botón servirá para encender o apagarla y por último, en el caso del termostato tendrá un botón para encender o apagar la calefacción acompañado de dos botones para ajustar la temperatura objetivo.

5.5.1.2 Pantalla de configuración

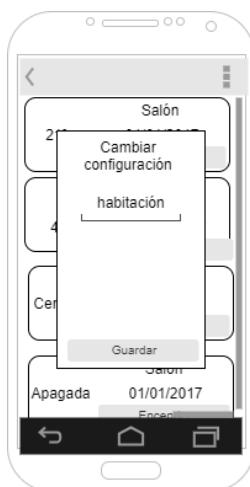


Ilustración 5. Interfaz pantalla configuración Android

Al igual que ocurre con la información que se muestra sobre el accesorio, la configuración también variara en función del accesorio.

Para acceder a la configuración de un accesorio bastara con hacer click sobre él.

Como todos los accesorios tienen una información en común la habitación en la que se encuentra se podrá configurar eso en todos.

Dependiendo del accesorio aparecerán opciones para configurar sensores o actuadores o ambos, por ejemplo, para la temperatura y humedad necesitaremos configurar un sensor.

Desde esta pantalla también se podrá configurar, si el accesorio lo permite, si se quieren recibir notificaciones sobre él.

5.6 Especificación del Plan de Pruebas

5.6.1 Pruebas unitarias

Las pruebas unitarias sirven para comprobar que una función o modulo funcionan de manera correcta.

En este proyecto se van a realizar pruebas unitarias sobre las implementaciones de los diferentes accesorios en HomePi Server, debido a que es la parte más crítica del sistema, si estas implementaciones fallan no funcionaría nada.

En concreto se harán pruebas para comprobar que el método “hasToWork” funciona siempre que se necesita, en todos los estados del accesorio en los que se puede llamar.

De igual modo se realizarán pruebas sobre el método “work” para comprobar que realiza los cambios necesarios en el accesorio para cumplir con las instrucciones del usuario.

5.6.2 Pruebas de integración

Este tipo de pruebas son utilices para identificar problemas en cada uno de los subsistemas de forma independiente.

En este caso se realizarán pruebas de integración con HomePi Android comprobando que si el usuario introduce datos incorrectos a la hora de crear o cambiar la configuración de un accesorio estos no producen ningún error del que el sistema no puede recuperarse, además que se proporciona el feedback adecuado para que usuario corrija esos errores en la configuración.

5.6.3 Pruebas de sistema

Las pruebas de integración son útiles para proyectos divididos en varios subsistemas y/o módulos y comprobar que todos funcionan correctamente trabajando unos con otros.

Dado que este proyecto se divide en varios subsistemas se realizarán pruebas para comprobar de forma exhaustiva que tanto HomePi Server como HomePi Android responden correctamente a los cambios que se realizan en la base de datos de Firebase, dejando siempre los datos de los accesorios en un estado consistente.

5.6.4 Pruebas de usabilidad

Dado que uno de los subsistemas de HomePi es una aplicación móvil que esta destina a ser utilizada por diferentes tipos de usuarios se tomara una muestra representativa de ellos y se observaran sus reacciones a los prototipos de interfaz diseñados.

Una vez realizada la interacción se le someterá a un pequeño cuestionario para obtener feedback, tanto de la interfaz, como de si les interesaría implantar HomePi en sus hogares.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

En este capítulo encuentra la arquitectura de todo el proyecto, en ella se puede ver tanto la estructura de cada uno de los subsistemas que componen el proyecto como la forma de intercomunicarse.

6.1.1 Diagramas de Paquetes

6.1.1.1 HomePi Android

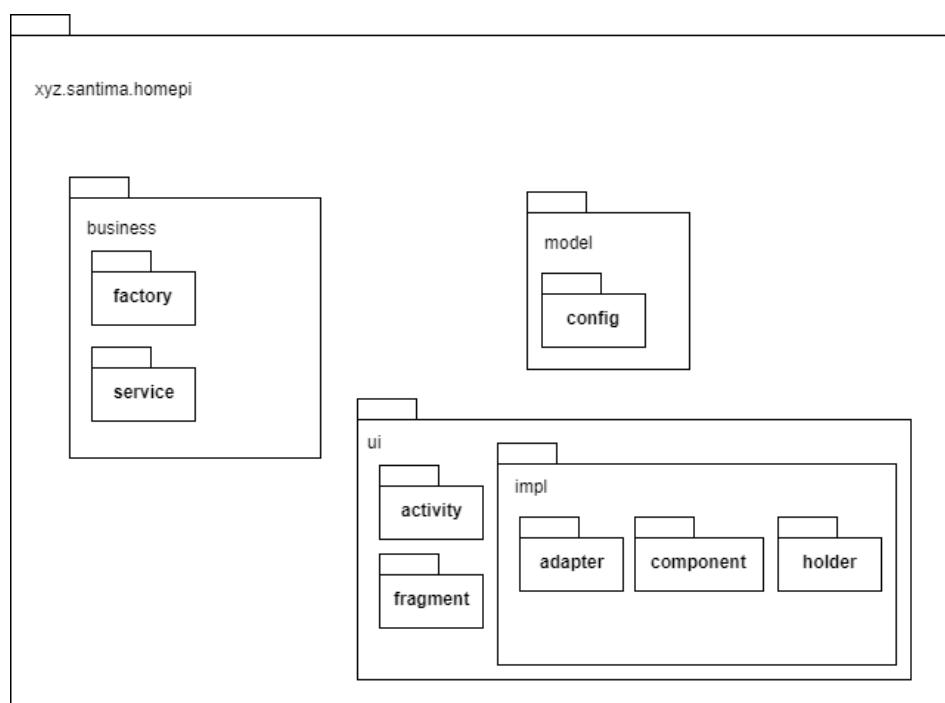


Ilustración 6. Diagrama de paquetes HomePi Android

6.1.1.1.1 business

Contiene dos paquetes, en el paquete Factory se encuentra la Factoría encargada de crear la lista de sensores y actuadores disponibles.

En el paquete Service, se encuentran los dos servicios que necesarios para poder recibir las notificaciones push desde Firebase Cloud Messaging.

6.1.1.1.2 model

En este paquete se encuentran las clases base para cargar la información de los accesorios desde Firebase, así como la clase para guardar la configuración de Firebase en SQLite.

También tenemos el paquete config, en el que se encuentra la jerarquía creada para poder crear las configuraciones de los diferentes sensores y actuadores.

6.1.1.1.3 ui

En este paquete se encuentran organizados en subpaquetes los elementos de interfaz de Android.

En los paquetes activity y fragment se encuentran las activitys y fragments utilizados por la aplicación.

El paquete impl incluye las implementaciones propias de elementos de interfaz de Android realizadas para la aplicación, también organizado con subpaquetes según se trate de adapters, holders u otros componentes.

6.1.1.2 HomePi Server

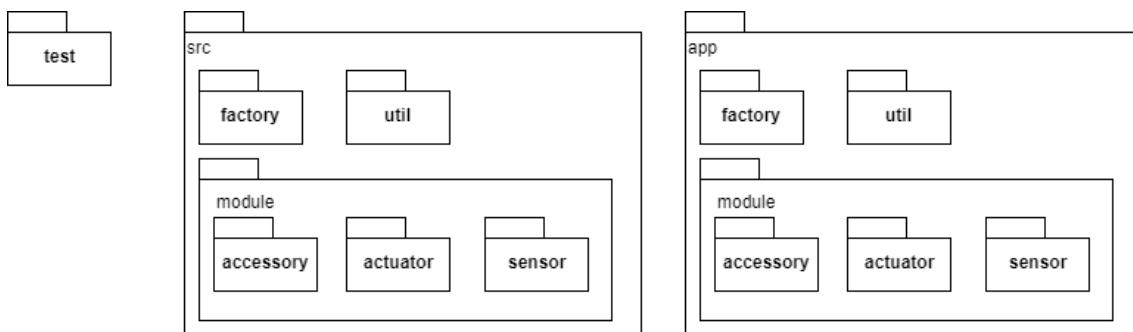


Ilustración 7. Diagrama de paquetes HomePi Server

6.1.1.2.1 test

Contiene las clases con los test unitarios para los distintos accesorios.

6.1.1.2.2 src

Contiene todo el código de HomePi Server escrito en TypeScript, se divide en diferentes subpaquetes.

6.1.1.2.2.1 factory

Contiene las factorías que se encargan de crear los diferentes accesorios, sensores y actuadores a partir de la configuración que se recibe mediante Firebase.

6.1.1.2.2.2 Util

Contiene varias clases de utilidad que se utilizan en los diferentes accesorios, sensores y actuadores.

6.1.1.2.2.3 Module

Esta subdivido en paquetes donde se encuentran los diferentes sensores, actuadores y accesorios con sus correspondientes interfaces e implementaciones.

6.1.1.2.3 app

Contiene exactamente la misma estructura que el paquete src, solo que en este caso contiene código JavaScript que procede de la salida del compilador de TypeScript.

6.1.2 Diagramas de Despliegue y Componentes

En esta sección se han juntado el despliegue de despliegue y el de componentes, ya que juntos dan una mejor visión del sistema.

El diagrama de despliegue nos permite ver todos los subsistemas que componen este proyecto y como se comunican entre ellos.

Para el despliegue y correcto funcionamiento del proyecto se necesita un proyecto de Firebase, ya que se utilizan varios de los servicios que ofrece.

La comunicación entre los distintos subsistemas se realiza utilizando diferentes protocolos, el más adecuado para cada una de las situaciones.

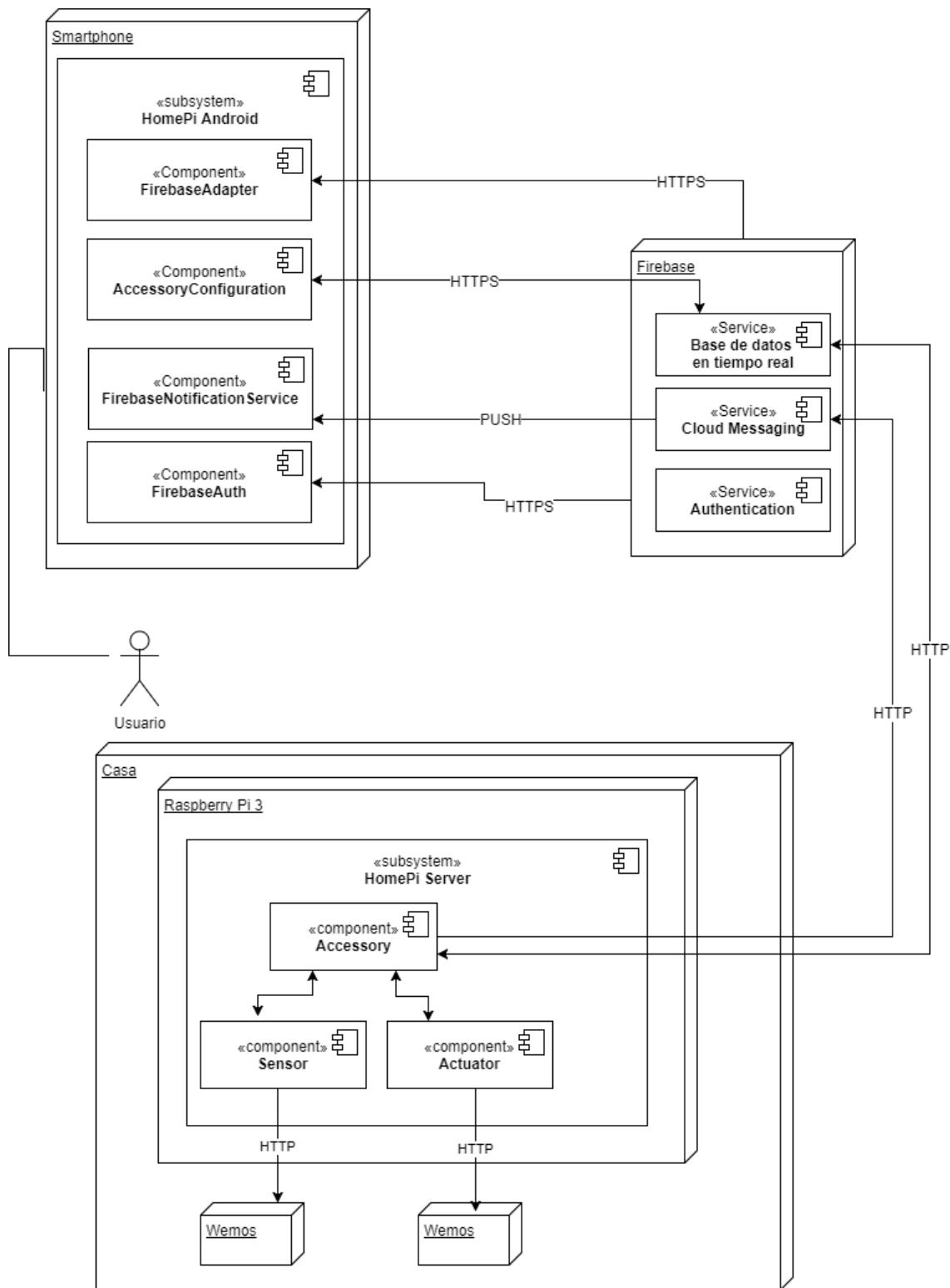


Ilustración 8. Diagrama de despliegue y componentes

6.1.2.1 HomePi Android

La aplicación para Android es la forma que tiene el usuario de interactuar con los accesorios que ha creado o creará en HomePi.

6.1.2.1.1 FirebaseAdapter

Se conecta a la base de datos en tiempo real de Firebase para recibir y enviar que se realizan sobre la lista de accesorios.

Todas estas comunicaciones se realizan de manera segura sobre HTTPS.

6.1.2.1.2 AccessoryConfiguration

De forma similar al componente FirebaseAdapter, se conecta a la base de datos en tiempo real de Firebase para enviar los cambios en la configuración de los accesorios.

6.1.2.1.3 FirebaseNotificationService

Es el componente encargado de recibir las notificaciones de Firebase Cloud Messaging, para lo que se utiliza el protocolo de Notificaciones Push, y transformar la información las notificaciones que se muestran al usuario.

6.1.2.1.4 FirebaseAuth

Es el componente encargado de iniciar la sesión del usuario en el servicio de FirebaseAuth. También permite recuperara la contraseña mediante el envío de un correo electrónico al usuario.

6.1.2.2 HomePi Server

En este diagrama aparece ejecutándose en una Raspberry Pi 3, aunque es compatible tanto con Windows, Mac y Linux.

Este subsistema hace de intermediario entre las acciones del usuario que se reciben mediante cambios en la base de datos en tiempo de real de Firebase y los diferentes actuadores y sensores instalados en la casa.

6.1.2.2.1 Accessory

Es el componente encargado de comunicarse con Firebase, para recibir y enviar cambios a la base de datos en tiempo real, mediante HTTP para garantizar que las comunicaciones son seguras y nadie puede escucharlas.

También se encarga de enviar las notificaciones a Firebase mediante HTTPS para que su servicio de Cloud Messaging las envie a los dispositivos necesarios.

6.1.2.2.2 Sensor y Actuator

Son los componentes encargados de la comunicación con los sensores y actuadores, que se hace mediante HTTP, ya que se encuentran dentro de la red local de la casa del usuario y además no se transmite ningún tipo de información sensible.

6.1.2.3 *Wemos*

Son las encargadas de leer los sensores o cambiar el estado de los actuadores respondiendo a las diferentes llamadas HTTP que tienen disponibles.

Se utilizan ya que, gracias a su pequeño tamaño, bajo precio y conexión WiFi se pueden colocar en multitud de lugares.

6.1.2.4 *Firebase*

Dentro de los diferentes planes que ofrece Firebase para este proyecto es suficiente el plan “Spark”, que es gratuito.

6.2 Diseño de Clases

En este apartado se incluye el diagrama de clases diseñado para el proyecto, como uno de los objetivos es que se puedan añadir nuevos sensores, actuadores y accesorios al sistema se diseña un utilizando interfaces para permitir fácilmente nuevas implementaciones.

Se han utilizado patrones de diseño en aquellos elementos en los que se ha considerado oportuno, por ejemplo, para los accesorios se utiliza el patrón composite, ya que un accesorio es un conjunto de sensores y/o actuadores.

6.2.1 Diagrama de Clases

6.2.1.1 HomePi Android

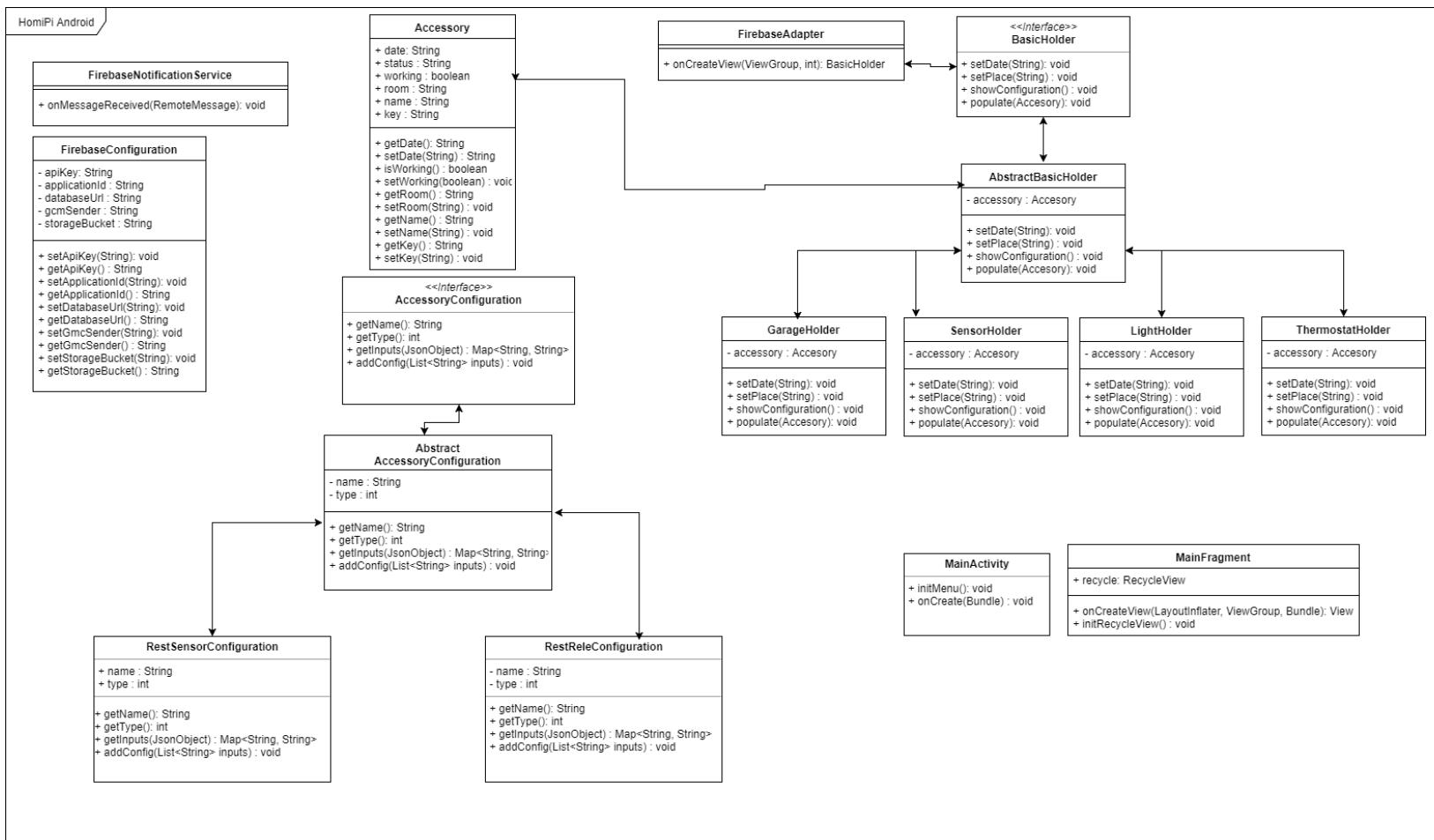


Ilustración 9. Diagrama de clases HomePi Android

6.2.1.1.1 RecycleView, Adapter y Holder

Para poder mostrar los accesorios dentro del componente de Android RecycleView se necesita crear una clase que implemente los métodos necesarios, en este caso además se quiere utilizar como fuente de datos Firebase, que incluye su propia implementación del RecycleView.

El problema es que la implementación de Firebase está pensada para que todos los elementos del RecycleView sean iguales por lo que al final será necesario crear nuestra propia versión del adapter.

Una vez tenemos el adapter, en este caso será el “FirebaseAdapter”, necesitamos crear clases que extiendan el ViewHolder del RecycleView, que será el que pinte los datos en la interfaz.

Como cada accesorio puede tener su propia interfaz, se crea una interfaz para que un solo adapter pueda tratar con diferentes implementaciones del ViewHolder, como es posible que gran parte del código sea común se piensa en introducir una clase abstracta.

6.2.1.1.2 FirebaseConfiguration

Es el objeto que con Realm almacenara en SQLite los datos de configuración del proyecto de Firebase.

6.2.1.1.3 AccesoryConfiguration

Al igual que ocurre con los accesorios en sí mismo los diferentes sensores y actuadores tendrán su propia configuración.

Como se planea que sea sencillo añadir nuevos en el futuro y siguiendo el esquema ideado durante la fase de análisis se crean interfaz y clase abstracta, con el código común, así como implementaciones concretas para los sensores y actuadores implementados.

6.2.1.1.4 Activity y Fragment

Es una recomendación a la hora de programar en Android tener una activity e ir cambiando el fragment interno para mostrar diferentes contenidos.

Esta elección permite que sea más sencillo añadir nuevas pantallas, si durante el desarrollo o mediante el feedback de los usuarios o simplemente en el futuro se ve que son necesarias, o realizar un diseño adaptado para tablets en el futuro.

6.2.1.2 HomePi Server

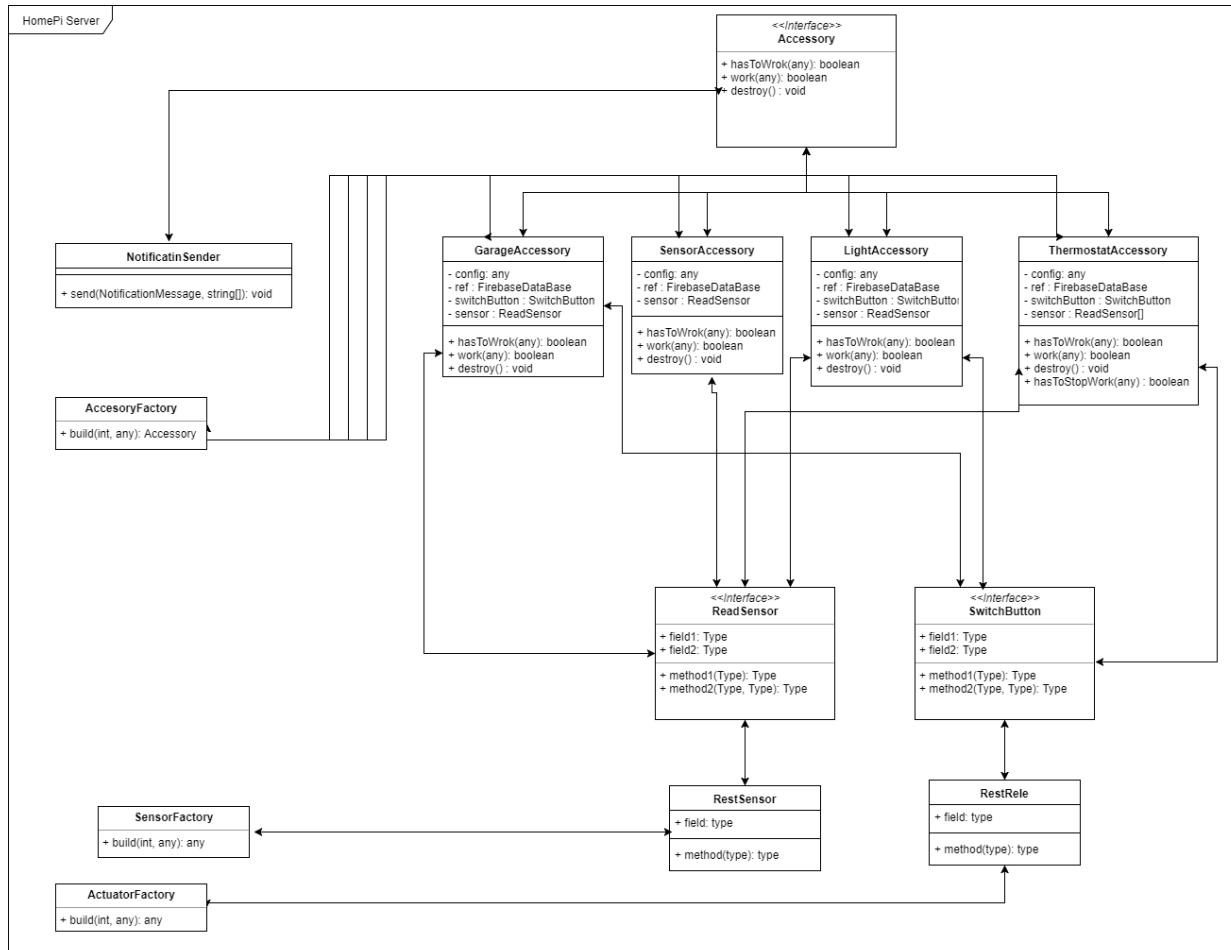


Ilustración 10. Diagrama de clases HomePi Server

6.2.1.2.1 Accesorios

En la fase de análisis simplemente se identificó la interfaz que implementaran los accesorios, ahora mismo podemos ver todos los accesorios que se pretende crear.

Todos comparte implementan únicamente los métodos de la interfaz, a excepción del accesorio termostato que se ha pensado que necesita un método adicional para determinar si se tiene que apagar la calefacción, con el nombre de “hasToStopWork”.

6.2.1.2.2 Sensores

En la fase de análisis somo se identificó la interfaz necesaria para leer los datos de un sensor, ahora mismo se incluye en el diagrama la clase que permitirá leer los datos del sensor con peticiones HTTP.

6.2.1.2.3 Actuadores

En la fase de análisis somo se identificó la interfaz necesaria para interactuar con un botón, ahora mismo se incluye en el diagrama la clase que permitirá interactuar mediante peticiones HTTP con ese botón.

6.2.1.2.4 Factorías

Para permitir crear los accesorios, sensores y actuadores con la configuración almacenada en Firebase se ha optado por crear una factoría para cada tipo de objeto.

Estas factorías solo tienen un método estático que recibe el tipo y la configuración del accesorio, sensor o actuador que creara.

6.2.1.2.5 NotificationSender

Tiene como único cometido enviar las notificaciones push mediante Firebase Cloud Messaging.

6.3 Diagramas de Interacción

Los diagramas de interacción que se muestran a continuación corresponden con algunos de los casos de uso del Capítulo 5.2.3.

Este tipo de diagramas representan todas las llamadas necesarias para realizar a cabo una acción del usuario.

6.3.1 Interactuar con accesorio

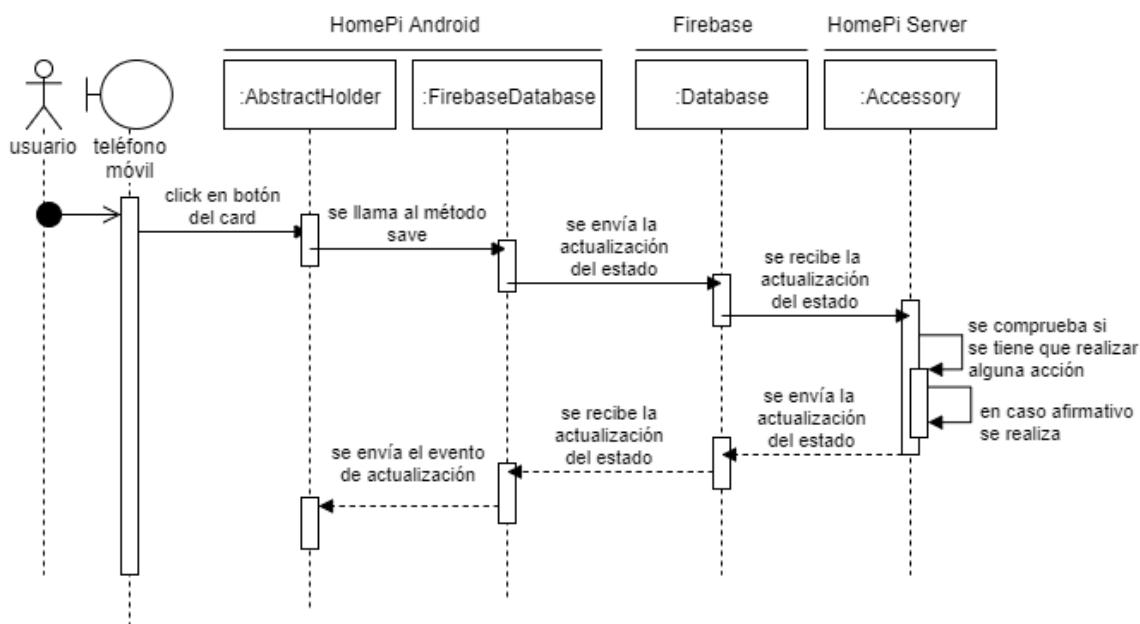


Ilustración 11. Diagrama de interacción. Caso de uso: Interactuar con accesorio.

En este caso el usuario utilizaría la aplicación para Android y realizaría cualquier interacción con un accesorio, por ejemplo: encender una luz, abrir la puerta del garaje...

Esta acción llamaría al método correspondiente del “AbstractHolder”, si no ha sido sobreescrito, lo que produciría una actualización en la base de datos de Firebase, que sería inmediatamente enviada a HomePi Server.

Una vez en HomePi Server, se comprobaría si este cambio implica que el accesorio tiene que realizar alguna acción, si no es necesario, se terminaría aquí.

Si fuera necesario, se realizaría la acción y se actualizaría la base de datos de Firebase con el resultado. Esta actualización se recibiría en HomePi Android que actualizaría el estado del accesorio que se muestra al usuario.

6.4 Diseño de la Base de Datos

6.4.1 Descripción del SGBD Usado

Para este proyecto se va a utilizar la base de datos en tiempo real de Firebase, que es una base de datos NoSQL sin esquema que almacena objetos JSON.

Una de las características más interesantes de esta base de datos es que permite suscribirse a eventos para ser notificado cuando se produce una actualización sobre los datos que nos interesa.

6.4.2 Integración del SGBD en Nuestro Sistema

Para integrar la base de datos en el proyecto se han utilizado las librerías oficiales de Firebase tanto para Android como para NodeJS.

6.4.3 Modelo de datos

Como la base de datos de Firebase carece de esquema, cada uno de los objetos almacenados puede poseer su propia estructura, pero dado que la integración con Android es más sencilla si se recuperan de Firebase objetos con esquema similar para los objetos que representan el estado los accesorios se ha seguido el siguiente esquema común:

```
{  
    "name" : "Interruptor 1",  
    "room" : "Mi habitación"  
    "date" : "2017-04-12T19:20:20.123"  
    "key" : "kjfasdkjfh",  
    "working" : false,  
    "user" : "HomePi Server"  
    "type" : 1  
}
```

En el caso de las configuraciones de los accesorios, tanto en HomePi Server con NodeJS como en HomePi Android con Java se trabaja con objetos JSON para no limitar las opciones de configuración.

6.5 Diseño de la Interfaz

Dado que no se han realizado más prototipos que los detallados en 5.5, en este apartado se incluyen las capturas finales de la interfaz de HomePi Android.

6.5.1 Pantalla inicial

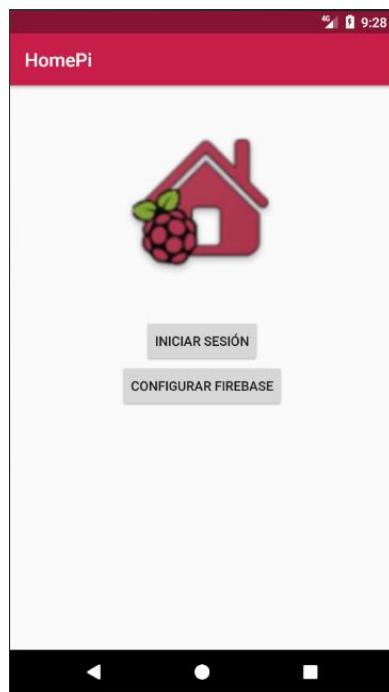


Ilustración 12. Pantalla final HomePi Android: inicial

Desde esta pantalla se puede iniciar sesión o cambiar la configuración del proyecto Firebase que utiliza HomePi.

6.5.2 Listado de accesorios



Ilustración 13. Pantalla final HomePi Android: Listado de accesorios

Se ve una lista con la información de cada uno de los accesorios creados, pudiéndose interactuar con ellos directamente desde esta pantalla o cambiar la configuración sin tocamos sobre uno de ellos.

También desde el botón con el “+” en verde, podremos añadir nuevos accesorios.

6.5.3 Configuración del accesorio



Ilustración 14. Pantalla final HomePi Android: Configuración del accesorio

Esta pantalla es muy similar a la de creación del accesorio, solo cambia la última opción, en lugar de borrar aparece la opción de salir.

Es diferente para cada accesorio, por ejemplo, en los que admiten notificaciones aparece el botón para activarlas/desactivarlas.

6.6 Especificación Técnica del Plan de Pruebas

Las pruebas descritas en el apartado 5.6, se realizarán de forma automática las pruebas unitarias y el resto se realizarán de manera manual.

6.6.1 Pruebas Unitarias

Estas pruebas se realizan sobre HomePi Server, para comprobar que los métodos de los accesorios “hasToWork” y “work” de los accesorios Garaje, Luces y Sensores y para los métodos “hasToWork” y “hasToStopWork” del accesorio termostato funcionan correctamente en cada uno de los estados posibles.

Como framework de pruebas para NodeJS se utiliza Mocha (7.3.5) con la librería Chai (7.3.6) que incluye funciones de aserto recomendable para TDD, ya que permite crear pruebas leíbles:

```
expect(x).to.be.throw();
expect(x).to.be.equals(1);
```

Estas pruebas se ejecutan tanto en local como de forma automática en el servidor de integración continua Travis CI (7.3.8), esto junto con commitizen (7.1.1.1), semantic-release (7.1.1.2) y la utilización de diferentes ramas permite crear versiones de forma automática.

Accesorio	Método	Entrada	Resultado esperado
GarageAccessory	hasToWork	working: true user:app status:OPENNING	false
		working: true user:app status:CLOSSING	false
		working: true user:app status:OPEN	true
		working: true user:app status:CLOSE	true
		working: false user:app status:OPENNING	false
		working: false user:app status:CLOSSING	false
		working: false user:app status:OPEN	false
		working: false user:app status:CLOSE	false

		working: true user:server status:OPENNING	false
		working: true user:server status:CLOSSING	false
		working: true user:server status:OPEN	false
		working: true user:server status:CLOSE	false
LightAccessory	hasToWork	Working: true User:app Status:OPEN	CLOSSING, CLOSE
		Working: true User:app Status:CLOSE	OPENNING, OPEN
	work	Working: true User:app Status:false	true
		Working: true User:app Status:true	true
		Working: false User:app Status:false	false
		Working: false User:app Status:true	false
		Working: true User:server Status:false	false
		Working: true User:server Status:true	false
		Working: true User:app Status:false	true
		Working: true User:app Status:true	false
		Working: true User: app	""
		Working: true User: server	
SensorAccessory	hasToWork	Working: true User: app	true
		Working: true User: server	false
	work	Working: false User: server	false
		Working: true User: app	""
ThermostatAccessory	hasToWork	Working: true	true

		User:app Status:21 timeOut: null	
		Working: true User:app Status:21 timeOut: {}	false
		Working: false User:app Status:21 timeOut: null	false
		Working: false User:app Status:21 timeOut: {}	false
		Working: true User:server Status:21 timeOut: {}	false
		Working: true User:server Status:21 timeOut: false	false
hasToStopWork		Working: true User:app Status:21 timeOut: null	true
		Working: true User:app Status:21 timeOut: {}	false
		Working: false User:app Status:21 timeOut: null	false
		Working: false User:app Status:21 timeOut: {}	true
		Working: true User:server Status:21 timeOut: {}	false
		Working: true User:server Status:21 timeOut: false	false
		Working: true User:server Status:21 timeOut: {}	false

		Working: true User:server Status:21 timeOut: false	false
--	--	---	-------

6.6.2 Pruebas de Integración y del Sistema

La realización de estas pruebas será manual, como parte de las pruebas de integración se comprobará que, si se introducen datos incorrectos en HomePi Android, la aplicación responde correctamente, por ejemplo, campos vacíos en la configuración de Firebase.

En cuanto a las pruebas de sistema se realizarán de forma manual, para comprobar que se reciben y envían correctamente los cambios tanto en HomePi Android como en HomePi Server.

Acción	Resultado esperado
Pulsar botón para abrir puerta del garaje.	La puerta pasa a estado abierto hasta que está abierta.
Actualizar temperatura.	Se muestra la temperatura actual leída del sensor.
Recibir notificaciones.	Se recibe la notificación.

6.6.3 Pruebas de Usabilidad

El objetivo principal de las pruebas de usabilidad es obtener información de cliente potenciales del sistema para mejorarlo y acercarse a las necesidades que estos poseen.

Estas pruebas se van a realizar a dos grupos de personas:

- **Usuarios con sistemas de domótica implantados:** hace referencia a personas que cuentan en su casa con cualquier sistema de domótica.
- **Usuarios con ganas de implantar un sistema de domótica:** hace referencia a personas interesadas en implantar un sistema de domótica en sus viviendas.

6.6.3.1 Diseño de Cuestionarios

6.6.3.1.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un teléfono móvil frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca

¿Qué tipo de actividades realiza con el teléfono móvil?

1. Es parte de mi trabajo o profesión
2. Lo uso básicamente para ocio
3. Solo empleo aplicaciones estilo Office
4. Únicamente leo el correo y navego ocasionalmente
5. Para todas las anteriores

¿Ha usado alguna vez software como el de esta prueba?

1. Sí, he empleado software similar
2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares
3. No, nunca

¿Qué busca Vd. Principalmente en un programa?

1. Que sea fácil de usar
2. Que sea intuitivo
3. Que sea rápido
4. Que tenga todas las funciones necesarias

6.6.3.1.2 Actividades guiadas

Se realizarán estas pruebas con una versión HomePi que tenga accesorios creados (temperatura, humedad, puerta del garaje y termostato), para permitir al usuario interactuar con ellos, así como cambiar configuraciones o crear nuevos accesorios.

Pregunta	Dificultad	Éxito
¿Cuál es la temperatura en el salón?	Baja	
¿Cuál es la humedad en el salón?	Baja	
¿Podrías abrir la puerta del garaje?	Baja	
¿Podrías activar la calefacción para tener la casa en 21º?	Media	
(Dentro del menú de configuración del accesorio) ¿Cómo activarías las notificaciones para saber cuándo otra persona modifica la puerta del garaje?	Baja	
(Dentro del menú de configuración del accesorio) ¿Cómo cambiarias el nombre de la habitación en la que se encuentra este accesorio?	Baja	

(Con un papel con los datos del sensor de temperatura) ¿Serías capaz de crear un accesorio para controlar la temperatura del salón?	Alta	
---	------	--

6.6.3.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Una vez el usuario ha interactuado con la aplicación se recoge su satisfacción con el siguiente cuestionario:

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
<i>¿Sabe dónde está dentro de la aplicación?</i>				
<i>¿Le resulta sencillo el uso de la aplicación?</i>				
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
<i>¿Funciona cada tarea como Vd. Espera?</i>				
<i>¿El tiempo de respuesta de la aplicación es muy grande?</i>				
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
<i>El tipo y tamaño de letra es</i>				
<i>Los iconos e imágenes usados son</i>				
<i>Los colores empleados son</i>				
Diseño de la Interfaz		Si	No	A veces
<i>¿Le resulta fácil de usar?</i>				
<i>¿El diseño de las pantallas es claro y atractivo?</i>				
<i>¿Cree que el programa está bien estructurado?</i>				
Observaciones				
Cualquier comentario del usuario				

6.6.3.1.4 Cuestionario para el responsable de las Pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	
<i>Tiempo en realizar cada tarea</i>	
<i>Errores leves cometidos</i>	
<i>Errores graves cometidos</i>	

6.6.4 Pruebas de Rendimiento

El proyecto ha sido desarrollado teniendo en cuenta las limitaciones de hardware de una Raspberry Pi., por lo que en todo momento se comprobará que es posible ejecutarlo en ella.

Así como que plan de Firebase elegido soporta los usuarios planeados, 4. Que serían 3 personas, mis padres y yo, y el servidor HomePi Server.

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

7.1.1 HomePi Server

7.1.1.1 Commitizen

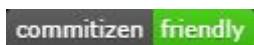


Ilustración 15. Etiqueta Commitizen friendly

Commitizen, <https://github.com/commitizen/cz-cli/>, es una herramienta compatible con npm que permite estructurar los mensajes que acompañan a los commit dentro de un repositorio git.

La herramienta se integra con el comando git de la consola y nos hace una serie de preguntas para construir el mensaje de cada commit.

La primera es el tipo de cambio: una nueva funcionalidad, corrección de algún bug, cambios en la documentación, cambios por estilo, refactorización, mejoras de rendimiento, cambios en los test...

Luego se nos pregunta sobre que parte de nuestro hemos realizado esos cambios, por ejemplo, si lo tenemos repartido en módulos, sobre cuáles.

Lo siguiente es una descripción corta de los cambios, a continuación, una descripción larga y para acabar si ese commit esta relacionado con alguna issue.

7.1.1.2 Semantic release



Ilustración 16. Etiqueta Semantic release

Semantic release, <https://github.com/semantic-release/semantic-release>, es otra herramienta compatible con npm que utilizada junto con Commitizen permite generar releases, siguiendo el estándar de Semantic Versioning, <http://semver.org/>.

Su funcionamiento es muy sencillo, cada vez que realizamos un commit a la rama principal de nuestro repositorio, si se pasan todos los test, semantic release analiza todos los commits desde la última release, en función del tipo de cambio de cambio que hayamos especificado en

commitizen modifica el número de versión tantas veces como sea necesario, como se menciona anteriormente siguiendo el estándar Semantic Versioning.

Para acabar publica un release en GitHub con el número de versión adecuado y con la lista de los todos los cambios, clasificados por tipo, desde la última versión publicada.

7.1.2 HomePi Andrid

7.1.2.1 Java Code Coventions

Para el desarrollo de la aplicación en Java para Android se han seguido las JCC.

7.1.3 JSON

JSON(JavaScript Object Notation) es un formato de texto utilizado para el intercambio de datos. Nació como parte del lenguaje JavaScript, pero debido a su cada vez más extendido uso de forma independiente como alternativa a XML se empieza a considerar un lenguaje independiente.

En el caso de este proyecto se utiliza para:

- La base de datos en tiempo real ya almacena objetos en este formato.
- Las respuestas de la HomePi Wemos van codificadas en JSON.
- La configuración de los accesorios, tanto desde HomePi Android como desde HomePi Server se realiza en JSON.

```
{  
    "status" : 21  
    "working" : false  
    "config" : {  
        "notifications" : ["3esfdsafdas4"]  
    }  
}
```

7.2 Lenguajes de Programación

En esta sección de la documentación se describirán los distintos lenguajes de programación utilizados durante el desarrollo de este proyecto.

De cada uno de especificar la versión utilizada, así como algunas de las librerías externas más importantes.

7.2.1 Android SDK + Java

Java es un lenguaje de programación basado en clases y orientado a objetos. Actualmente en la versión que se usa en este proyecto de Android Studio es el único lenguaje oficial para programar en Android.

Este ha sido el lenguaje de programación que más hemos utilizado durante toda la carrera. Además, gracias a un proyecto reciente había adquirido bastante experiencia a la hora de utilizarlo con las características propias de Android.

Para este proyecto se ha utilizado el SDK de Android en su versión 25, en concreto la versión 25.0.3 de las herramientas de construcción.

7.2.1.1 Principales librerías utilizadas

7.2.1.1.1 ButterKnife



Ilustración 17. Logotipo ButterKnife

ButterKnife es un conjunto de anotaciones que permite facilitar la obtención de referencias a elementos de Android como pueden ser: componente de las vistas, strings del archivo de localización o la asignación de métodos a eventos.

En este proyecto se ha utilizado la versión 8.6.0

7.2.1.1.2 Realm

Realm es uno de los múltiples ORM disponibles para Android, permite mapear objetos en un base de datos relacionar como SQLite a objetos de Java.

Se ha decidido utilizar esta librería sobre las otras alternativas disponibles debido que según la mayoría de benchmarks consultados es el más rápido de todos.

Aunque solo se utiliza para guardar los ajustes de conexión con Firebase, se ha decidido una librería de ORM debido a que trabajar directamente con SQLite es bastante incómodo.

Para este proyecto se ha utilizado la versión 3.3.2

7.2.1.1.3 MaterialDialog

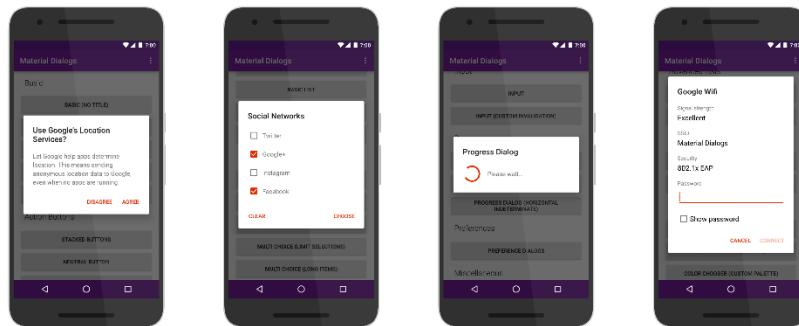


Ilustración 18. Ejemplos de MaterialDialog

Es una librería que ofrece una API sencilla para crear diálogos en Android siguiendo la especificación de MaterialDesign de Google.

7.2.2 TypeScript

TypeScript es un lenguaje de programación libre y código abierto, mantenido por Microsoft, que añade a JavaScript nuevas características como tipado estático y objetos basados en clases (que hasta la última especificación de JavaScript no estaba disponible).

Al estar construido sobre JavaScript, por lo que TypeScript es compatible con cualquier código JavaScript existente, además el compilador de TypeScript lo traduce a código JavaScript original.



Ilustración 19. Logotipo TypeScript

Se decide utilizar TypeScript sobre JavaScript debido a permite utilizar de una forma más amplia que el estándar actual de JavaScript conceptos teóricos aprendidos durante carrera para Java.

Para el desarrollo de este proyecto se utiliza la versión 2.3.4 del compilador de TypeScript.

7.2.3 Simple C

Las placas Wemos son compatibles con el lenguaje creado por Arduino para sus productos, que está basado en el lenguaje de alto nivel Processing y soporta todas las funciones estándar de C y algunas de C++, llamado Simple C.

Se ha utilizado la versión incluida en la última versión del Arduino IDE.

7.3 Herramientas y Programas Usados para el Desarrollo

Descripción de todas las herramientas de desarrollo, sistemas adicionales existentes, complementos y otros productos software utilizados para la implementación del proyecto.

7.3.1 WebStorm

WebStorm es un IDE, entorno de desarrollo integrado, desarrollado por JetBrains para el desarrollo de proyectos con tecnologías web: HTML5, CSS, SASS, JavaScript, Aplicaciones hibridas...



Ilustración 20. Logotipo WebStorm

Entre las características destacadas encontramos:

- Integración con múltiples frameworks y herramientas.
- Completado de código inteligente.
- Integración con git, gulp, NodeJS, npm y TypeScript.

Para el desarrollo de este proyecto se ha utilizado la versión 2017.1.4 para el desarrollo de HomePi Server.

7.3.2 npm

npm es el gestor de dependencias utilizado para programar en NodeJS, ya que se instala automáticamente al instalar NodeJS.



Ilustración 21. Logo npm

Se ejecuta desde la línea de comandos. Aparte de gestionar las dependencias de nuestro proyecto permite incluir información sobre el proyecto en su archivo de configuración, así como establecer una serie de comandos, como puede ser la ejecución de test.

Para el desarrollo de este proyecto se ha utilizado la versión 4.2.0.

7.3.3 gulp.js

gulp.js es un gestor de tareas útiles durante el desarrollo de proyectos. Es gratuito, open-source y está completamente estricto en JavaScript.



Ilustración 22. Logotipo de gulp

Para este proyecto se ha utilizado para automatizar la conversión del código en TypeScript a código JavaScript siguiendo la especificación “es5”, para así hacer que HomePi Server puede ser ejecutado con NodeJS directamente sin necesidad de instalar el compilador de TypeScript.

Para el desarrollo de este proyecto se ha utilizado la versión 3.9.1

7.3.4 nodemon

nodemon es una herramienta para el desarrollo de aplicaciones en NodeJS que monitoriza los cambios en el código y automáticamente recarga la aplicación con todos los nuevos cambios.



Ilustración 23. Logotipo nodemon

Durante el desarrollo de este proyecto se ha utilizado la versión 1.11.0.

7.3.5 Mocha

Mocha es un framework para la ejecución de pruebas en NodeJS o en el navegador.



Ilustración 24. Logotipo Mocha

En este proyecto se ha utilizado la versión 3.4.2 para la ejecución de las pruebas unitarias en HomePi Server.

7.3.6 Chai

Chai es una librería para facilitar el desarrollo de proyectos siguiendo BDD o TDD, es 100% compatible con Mocha.



Ilustración 25. Logotipo chai

La característica principal chai es que sus métodos para realizar los test son muy expresivos y permiten realizar test legibles. También existe multitud de paquetes que añaden nuevas funciones a chai.

Para este proyecto se ha utilizado la versión 4.0.2.

7.3.7 David-DM

David DM, <https://david-dm.org/>, es un servicio que permite comprobar que todas las dependencias que utilizamos para nuestro proyecto se encuentran en la última versión.



Ilustración 26. Etiqueta David DM

Es un servicio muy útil ya que de una forma visual podemos comprobar si tenemos alguna dependencia desactualizada, lo que puede probar problemas de seguridad.

Funciona solo para proyectos en NodeJS.

7.3.8 Travis CI



Ilustración 27. Logotipo de Travis CI

Travis CI es un servidor de integración continua que funciona perfectamente junto con GitHub.

Para este proyecto se ha utilizado Travis para la ejecución automática de las pruebas unitarias en HomePi Server evitando que semantic release publique nuevas versiones si hay algún problema con las pruebas unitarias.

Travis CI es gratuito para proyecto open-source o para estudiantes como parte del pack de estudiantes de GitHub.

7.3.9 Android Studio



Ilustración 28. Logotipo Android Studio

Android Studio es el IDE oficial de Google para el desarrollo de aplicaciones en Android, esta basado en el IntelliJ de JetBrains, por lo que comparte muchas de las características generales de los IDEs de JetBrains.

Características destacadas:

- Gestión de dependencias con gradle.
- Completado inteligente de código.
- Emulador Android.
- Integración con git.

Para el desarrollo de este proyecto se ha utilizado la versión 2.3.3.

7.3.10 Gradle

Grade es un herramienta open-source para automatizar la tarea de construcción de nuestros proyectos, en este caso se ha utilizado para el desarrollo del proyecto Android.



Ilustración 29. Logotipo Gradle

Gradle también permite controlar las diferentes dependencias de nuestro proyecto.

En este proyecto se ha utilizado la versión 2.3.3 de Gradle para Android.

7.3.11 VisualStudio Code

Editor ligero de código libre y gratuito desarrollado por Microsoft, ofrece compatibilidad con multiples lenguajes, frameworks... mediante plug-ins que se pueden descargar desde la tienda integrada.



Ilustración 30. Logotipo VisualStudio Code

Para el desarrollo de este proyecto se ha utilizado la versión 1.13.1 y se ha utilizado para editar los archivos JSON exportados/importados desde/a Firebase.

7.3.12 Git

Git es un software de control de versión diseñado con dos ideas en mente que sea ligero y tenga un gran rendimiento.



Ilustración 31. Logotipo git

En este proyecto para la realización de commit mediante Commitizen se ha utilizado la versión de consola de git.

7.3.13 GitHub

Entre las diferentes plataformas alternativas que ofrecen la posibilidad de alojar proyecto mediante el sistema de control de versiones git para los tres repositorios con componen HomePi se ha utilizado GitHub.



Ilustración 32. Logotipo GitHub

Por citar alguno de los motivos:

- Es una de las plataformas más utilizadas para el control de versiones.
- Compatibilidad con multitud de servicios externos, como Travis.
- Wiki para cada proyecto.

7.3.14 GitHub Desktop



Ilustración 33. Logotipo GitHub Desktop

GitHub Desktop es la aplicación de escritorio de GitHub para manejar los repositorios de git, es compatible con cualquier servicio que utilice git, no solo GitHub.

Se ha decidido utilizar una versión de escritorio conjunto con la versión de consola, ya que resulta mucho más cómodo comparar los cambios con una interfaz gráfica que con el comando en consola.

7.3.15 Arduino IDE

Arduino IDE es el entorno de desarrollo oficial para placas Arduino, aunque en este caso se utilizan placas del fabricante Wemos como se explica anteriormente estas son compatibles con Arduino.



Ilustración 34. Logotipo Arduino IDE

Este IDE permite compilar y enviar el código a las diferentes placas compatibles, además de tener un sistema para la gestión de las diferentes librerías que se utilizan.

Para este proyecto se ha utilizado la versión 1.8.3.

7.3.16 Microsoft Word

Microsoft Word es el programa de procesamiento de texto de Microsoft y forma parte de su suite ofimática Office 360.



Ilustración 35. Logotipo Microsoft Word

Se ha utilizado este software para la realización de la documentación del proyecto. En concreto se ha utilizado la versión 2016, en sus versiones para Windows y Mac.

7.3.17 Microsoft Excel

Microsoft Word es el programa de hora de cálculo de Microsoft y forma parte de su suite ofimática Office 360



Ilustración 36. Logo Microsoft Excel

En este proyecto se ha utilizado para realizar todos los cálculos referentes a los presupuestos, en concreto la versión de 2016 para Windows.

7.3.18 Microsoft Project

Microsoft Word es el programa para gestión de proyecto de Microsoft, no forma parte de su suite ofimática Office 360, pero si está dentro de la familia de Office.



Ilustración 37. Logo Microsoft Project

Permite realizar planificaciones de proyectos, sin importar el tamaño, controlar los recursos asignados, el calendario...

Se ha utilizado la versión de 2013 para Windows para realizar las planificaciones de este proyecto.

7.3.19 Google Drive

Google Drive es el servicio de alojamiento de Google, gratuitamente ofrece 15GB de almacenamiento.



Ilustración 38. Logo Google Drive

Ofrece integración con múltiple herramientas y servicios como: la suite ofimática de Google y otras de terceros como draw.io.

En este proyecto se ha utilizado para realizar una copia de seguridad de la documentación y los diferentes diagramas.

7.3.20 Draw.io

Es una herramienta gratuita con integración en Google Drive que permite la creación de diferentes tipos de diagramas: clases, despliegue, paquetes...



Ilustración 39. Logotipo Draw.io

También permite la realización de prototipos de aplicaciones Android.

7.3.21 Flat Tomato

Flat Tomato es una aplicación para iOS/Android para gestionar el uso del tiempo siguiendo la técnica de pomodoros.

Esta técnica consiste en trabajar en periodos cortos, normalmente de 25 minutos, y entre cada uno de estos periodos realizar un pequeño descanso, normalmente de 5 minutos. Con estas pausas frecuentes se trata de mejorar la agilidad mental y ofrecer un mayor rendimiento durante los períodos de trabajo.



Ilustración 40. Logotipo Flat Tomato

Se ha utilizado para tratar de llevar un control del tiempo dedicado a este proyecto.

7.4 Creación del Sistema

7.4.1 Problemas Encontrados

7.4.1.1 Sensores DHT11 y DHT22 imprecisos

Como sensores para obtener la información de temperatura y humedad se había pensado en un principio en el DHT11, debido a que es más barato que la versión superior DHT22.

Se conocía que este sensor DHT11 era más impreciso, pero no fue hasta la hora de la implementación cuando se descubrió que daba valores excesivamente incorrectos con +- 10 grados en la temperatura y con valores de humedad de 80 o 0.

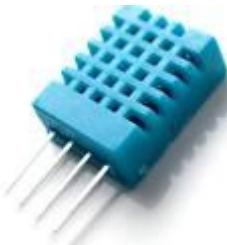


Ilustración 41. Sensor DHT11

Debido a que la idea es tener el sistema en funcionamiento se procedido a adquirir la versión superior el DHT22.



Ilustración 42. Sensor de DHT22

En este caso el sensor daba valores de temperatura basta realistas, pero siempre devolvía 1% de humedad, lo que parece imposible encontrándonos en Asturias.

Debido a que la función principal del sensor iba a ser obtener valores de temperatura y el DHT22 cumplía no se buscaron más alternativas.

7.4.1.2 Integración con HomeKit

Inicialmente el sistema iba a ofrecer compatibilidad con el framework de domótica diseñado por Apple HomeKit, por el medio de una librería intermedia para NodeJS llamada Homebridge.

El problema fue cuando en el prototipo, solo con la puerta del garaje, que sirve como base de este proyecto sin ningún cambio en el código se empezaron a producir problemas al ser utilizado simultáneamente con Android y desde iOS.

Tras una investigación sobre el origen de esta incompatibilidad repentina sin llegar a ninguna conclusión, se decide abandonar esta integración para la primera versión del proyecto.

7.4.2 Descripción Detallada de las Clases

Dado que la documentación generada sobre el detalle de las clases implementadas es demasiado grande como para incluirla en el documento, en el apartado 13.2 se explica su ubicación en el archivo adjunto a esta documentación.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

Para la realización de estas pruebas se han creado sensores y actuadores falsos, para evitar errores ya que durante la ejecución en Travis CI no estarán disponibles.

En el parámetro usuario, app significa que se le pasa un usuario como el que enviaría la app y server significa que se le pasa el usuario que dejaría HomePi Server después de actualizar el estado del accesorio.

Accesorio	Método	Entrada	Resultado esperado	Resultado obtenido
GarageAccessory	hasToWork	working: true user:app status:OPENNING	false	false
		working: true user:app status:CLOSSING	false	false
		working: true user:app status:OPEN	true	true
		working: true user:app status:CLOSE	true	true
		working: false user:app status:OPENNING	false	false
		working: false user:app status:CLOSSING	false	false
		working: false user:app status:OPEN	false	false
		working: false user:app status:CLOSE	false	false
		working: true user:server status:OPENNING	false	false
		working: true user:server status:CLOSSING	false	false

		status:CLOSE		
	work	Working: true User:app Status:OPEN	CLOSSING, CLOSE	CLOSSING, CLOSE
		Working: true User:app Status:CLOSE	OPENNING, OPEN	OPENNING, OPEN
LightAccessory	hasToWork	Working: true User:app Status:false	true	true
		Working: true User:app Status:true	true	true
		Working: false User:app Status:false	false	false
		Working: false User:app Status:true	false	false
		Working: true User:server Status:false	false	false
		Working: true User:server Status:true	false	false
	work	Working: true User:app Status:false	true	true
		Working: true User:app Status:true	false	false
		Working: true User: app	""	""
		Working: false User: server	false	false
SensorAccessory	hasToWork	Working: true User: app	true	true
		Working: true User: server	false	false
		Working: false User: server	false	false
	work	Working: true User: app	""	""
ThermostatAccesory	hasToWork	Working: true User:app Status:21 timeOut: null	true	true
		Working: true User:app Status:21 timeOut: {}	false	false
		Working: false User:app Status:21 timeOut: null	false	false

		Working: false User:app Status:21 timeOut: {}	false	false
		Working: true User:server Status:21 timeOut: {}	false	false
		Working: true User:server Status:21 timeOut: false	false	false
hasToStopWork		Working: true User:app Status:21 timeOut: null	true	true
		Working: true User:app Status:21 timeOut: {}	false	false
		Working: false User:app Status:21 timeOut: null	false	false
		Working: false User:app Status:21 timeOut: {}	true	true
		Working: true User:server Status:21 timeOut: {}	false	false
		Working: true User:server Status:21 timeOut: false	false	false
		Working: true User:server Status:21 timeOut: {}	false	false
		Working: true User:server Status:21 timeOut: false	false	false
		Working: true User:server Status:21 timeOut: false	false	false
		Working: true User:server Status:21 timeOut: false	false	false

Las 45 pruebas diseñadas pasan correctamente, por tanto, los accesorios se comportan tal y como se espera en todas las posibles combinaciones.

8.2 Pruebas de Integración y del Sistema

Para la realización de estas pruebas se ejecuta HomePi Server en el equipo de desarrollo y la aplicación en el emulador de Android Studio.

8.2.1 Pruebas de integración

Se ha intentado cambiar la configuración de Firebase dejando alguno de los campos sin completar y se muestra el mensaje esperado al usuario.

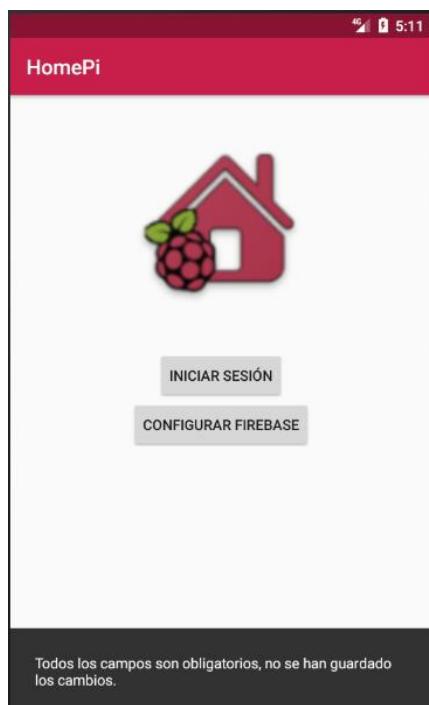


Ilustración 43. Mensaje de error en HomePi Android

8.2.2 Pruebas de sistema

Acción	Resultado esperado	Éxito
Pulsar botón para abrir puerta del garaje.	La puerta pasa a estado abriendo hasta que está abierta.	Si
Actualizar temperatura.	Se muestra la temperatura actual leída del sensor.	Si
Recibir notificaciones.	Se recibe la notificación.	Si

8.3 Pruebas de Usabilidad

Para realizar las pruebas de usabilidad se realizan los cuestionarios definidos en el apartado 5.6.4, sobre diferentes usuarios pertenecientes a los grupos definidos.

8.3.1 Usuarios con sistemas de domótica

En este caso se realizan a mis padres, ya que en casa está implantado en este momento el prototipo para controlar el garaje y ambos los utilizan con frecuencia.

8.3.1.1 Usuario 1

8.3.1.1.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un teléfono móvil frecuentemente?
<ul style="list-style-type: none">1. <u>Todos los días</u>2. Varias veces a la semana3. Ocasionalmente4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el teléfono móvil?
<ul style="list-style-type: none">1. Es parte de mi trabajo o profesión2. Lo uso básicamente para ocio3. Solo empleo aplicaciones estilo Office4. Únicamente leo el correo y navego ocasionalmente5. <u>Para todas las anteriores</u>
¿Ha usado alguna vez software como el de esta prueba?
<ul style="list-style-type: none">1. <u>Sí, he empleado software similar</u>2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
<ul style="list-style-type: none">1. Que sea fácil de usar2. <u>Que sea intuitivo</u>3. Que sea rápido4. Que tenga todas las funciones necesarias

8.3.1.1.2 Actividades guiadas

Se realizarán estas pruebas con una versión HomePi que tenga accesorios creados (temperatura, humedad, puerta del garaje y termostato), para permitir al usuario interactuar con ellos, así como cambiar configuraciones o crear nuevos accesorios.

Pregunta	Dificultad	Éxito
¿Cuál es la temperatura en el salón?	Baja	Si
¿Cuál es la humedad en el salón?	Baja	Si
¿Podrías abrir la puerta del garaje?	Baja	Si
¿Podrías activar la calefacción para tener la casa en 21º?	Media	Si
(Dentro del menú de configuración del accesorio) ¿Cómo activarías las notificaciones para saber cuándo otra persona modifica la puerta del garaje?	Baja	No, el botón de notificaciones refleja el estado objetivo en lugar del estado actual.
(Dentro del menú de configuración del accesorio) ¿Cómo cambiarias el nombre de la habitación en la que se encuentra este accesorio?	Baja	Si
(Con un papel con los datos del sensor de temperatura) ¿Serías capaz de crear un accesorio para controlar la temperatura del salón?	Alta	Si

8.3.1.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Una vez el usuario ha interactuado con la aplicación se recoge su satisfacción con el siguiente cuestionario:

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?		X		
¿Le resulta sencillo el uso de la aplicación?	X			
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?	X			

<i>¿El tiempo de respuesta de la aplicación es muy grande?</i>	X			
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
<i>El tipo y tamaño de letra es</i>	X			
<i>Los iconos e imágenes usados son</i>	X			
<i>Los colores empleados son</i>	X			
Diseño de la Interfaz		Si	No	A veces
<i>¿Le resulta fácil de usar?</i>		X		
<i>¿El diseño de las pantallas es claro y atractivo?</i>		X		
<i>¿Cree que el programa está bien estructurado?</i>		X		
Observaciones				
<i>El botón de las notificaciones debería estar del revés.</i>				

8.3.1.1.4 Cuestionario para el responsable de las Pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Si, tras una pequeña explicación de la app, el usuario la ha utilizado con rapidez.
<i>Tiempo en realizar cada tarea</i>	-
<i>Errores leves cometidos</i>	-
<i>Errores graves cometidos</i>	-

8.3.1.2 Usuario 2

8.3.1.2.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un teléfono móvil frecuentemente?
<ol style="list-style-type: none"> 1. <u>Todos los días</u> 2. <u>Varias veces a la semana</u> 3. <u>Ocasionalmente</u> 4. <u>Nunca o casi nunca</u>
¿Qué tipo de actividades realiza con el teléfono móvil?
<ol style="list-style-type: none"> 1. <u>Es parte de mi trabajo o profesión</u> 2. <u>Lo uso básicamente para ocio</u> 3. <u>Solo empleo aplicaciones estilo Office</u> 4. <u>Únicamente leo el correo y navego ocasionalmente</u> 5. <u>Para todas las anteriores</u>
¿Ha usado alguna vez software como el de esta prueba?

<ol style="list-style-type: none"> 1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
<p>¿Qué busca Vd. Principalmente en un programa?</p> <ol style="list-style-type: none"> 1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

8.3.1.2.2 Actividades guiadas

Se realizarán estas pruebas con una versión HomePi que tenga accesorios creados (temperatura, humedad, puerta del garaje y termostato), para permitir al usuario interactuar con ellos, así como cambiar configuraciones o crear nuevos accesorios.

Pregunta	Dificultad	Éxito
¿Cuál es la temperatura en el salón?	Baja	Si
¿Cuál es la humedad en el salón?	Baja	Si
¿Podrías abrir la puerta del garaje?	Baja	Si
¿Podrías activar la calefacción para tener la casa en 21º?	Media	Si
(Dentro del menú de configuración del accesorio) ¿Cómo activarías las notificaciones para saber cuándo otra persona modifica la puerta del garaje?	Baja	No, el botón de notificaciones refleja el estado objetivo en lugar del estado actual.
(Dentro del menú de configuración del accesorio) ¿Cómo cambiarías el nombre de la habitación en la que se encuentra este accesorio?	Baja	Si
(Con un papel con los datos del sensor de temperatura) ¿Serías capaz de crear un accesorio para controlar la temperatura del salón?	Alta	Si

8.3.1.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Una vez el usuario ha interactuado con la aplicación se recoge su satisfacción con el siguiente cuestionario:

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
<i>¿Sabe dónde está dentro de la aplicación?</i>	X			
<i>¿Le resulta sencillo el uso de la aplicación?</i>	X			
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
<i>¿Funciona cada tarea como Vd. Espera?</i>	X			
<i>¿El tiempo de respuesta de la aplicación es muy grande?</i>	X			
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
<i>El tipo y tamaño de letra es</i>	X			
<i>Los iconos e imágenes usados son</i>	X			
<i>Los colores empleados son</i>	X			
Diseño de la Interfaz		Si	No	A veces
<i>¿Le resulta fácil de usar?</i>		X		
<i>¿El diseño de las pantallas es claro y atractivo?</i>		X		
<i>¿Cree que el programa está bien estructurado?</i>		X		
Observaciones				
<i>El botón de las notificaciones debería estar del revés.</i>				

8.3.1.2.4 Cuestionario para el responsable de las Pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Si, tras una pequeña explicación de la app, el usuario la ha utilizado con rapidez.
<i>Tiempo en realizar cada tarea</i>	-
<i>Errores leves cometidos</i>	-
<i>Errores graves cometidos</i>	-

8.3.2 Usuarios con ganas de implantar un sistema de domótica

Se realizan sobre amigos/familiares/conocidos interesados en implantar un sistema de domótica.

Para cuando se realizaron las pruebas a usuarios dentro de este grupo, ya se había corregido el problema con el botón de las notificaciones.

8.3.2.1 Usuario 1

8.3.2.1.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un teléfono móvil frecuentemente?	<ol style="list-style-type: none"> 1. <u>Todos los días</u> 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el teléfono móvil?	<ol style="list-style-type: none"> 1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente 5. <u>Para todas las anteriores</u>
¿Ha usado alguna vez software como el de esta prueba?	<ol style="list-style-type: none"> 1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. <u>No, nunca</u>
¿Qué busca Vd. Principalmente en un programa?	<ol style="list-style-type: none"> 1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. <u>Que tenga todas las funciones necesarias</u>

8.3.2.1.2 Actividades guiadas

Se realizarán estas pruebas con una versión HomePi que tenga accesorios creados (temperatura, humedad, puerta del garaje y termostato), para permitir al usuario interactuar con ellos, así como cambiar configuraciones o crear nuevos accesorios.

Pregunta	Dificultad	Éxito
¿Cuál es la temperatura en el salón?	Baja	Si
¿Cuál es la humedad en el salón?	Baja	Si, quizás faltaría alguna mejor identificación de que es lo que indica este accesorio, se descubrió por descarte

¿Podrías abrir la puerta del garaje?	Baja	Si
¿Podrías activar la calefacción para tener la casa en 21º?	Media	Si
(Dentro del menú de configuración del accesorio) ¿Cómo activarías las notificaciones para saber cuándo otra persona modifica la puerta del garaje?	Baja	Si
(Dentro del menú de configuración del accesorio) ¿Cómo cambiarias el nombre de la habitación en la que se encuentra este accesorio?	Baja	Si
(Con un papel con los datos del sensor de temperatura) ¿Serías capaz de crear un accesorio para controlar la temperatura del salón?	Alta	Si

8.3.2.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Una vez el usuario ha interactuado con la aplicación se recoge su satisfacción con el siguiente cuestionario:

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	X			
¿Le resulta sencillo el uso de la aplicación?	X			
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?	X			
¿El tiempo de respuesta de la aplicación es muy grande?			X	
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es			X	
Los iconos e imágenes usados son	X			
Los colores empleados son	X			
Diseño de la Interfaz	Si	No	A veces	
¿Le resulta fácil de usar?	X			
¿El diseño de las pantallas es claro y atractivo?	X			

<i>¿Cree que el programa está bien estructurado?</i>	X		
Observaciones			
<p>El usuario mostro un gran interés por instalar el sistema en su casa.</p> <p>Por otro lado, en el móvil del usuario todos los textos de la aplicación se vean excesivamente pequeños, se tendrá que investigar el motivo.</p>			

8.3.2.1.4 Cuestionario para el responsable de las Pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Si, tras una pequeña explicación de la app, el usuario la ha utilizado con rapidez.
<i>Tiempo en realizar cada tarea</i>	-
<i>Errores leves cometidos</i>	-
<i>Errores graves cometidos</i>	-

8.3.2.2 Usuario 1

8.3.2.2.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un teléfono móvil frecuentemente?
<ol style="list-style-type: none"> 1. <u>Todos los días</u> 2. <u>Varias veces a la semana</u> 3. <u>Ocasionalmente</u> 4. <u>Nunca o casi nunca</u>
¿Qué tipo de actividades realiza con el teléfono móvil?
<ol style="list-style-type: none"> 1. <u>Es parte de mi trabajo o profesión</u> 2. <u>Lo uso básicamente para ocio</u> 3. <u>Solo empleo aplicaciones estilo Office</u> 4. <u>Únicamente leo el correo y navego ocasionalmente</u> 5. <u>Para todas las anteriores</u>
¿Ha usado alguna vez software como el de esta prueba?
<ol style="list-style-type: none"> 1. <u>Sí, he empleado software similar</u> 2. <u>No, aunque si empleo otros programas que me ayudan a realizar tareas similares</u> 3. <u>No, nunca</u>
¿Qué busca Vd. Principalmente en un programa?
<ol style="list-style-type: none"> 1. <u>Que sea fácil de usar</u> 2. <u>Que sea intuitivo</u> 3. <u>Que sea rápido</u> 4. <u>Que tenga todas las funciones necesarias</u>

--

8.3.2.2.2 Actividades guiadas

Se realizarán estas pruebas con una versión HomePi que tenga accesorios creados (temperatura, humedad, puerta del garaje y termostato), para permitir al usuario interactuar con ellos, así como cambiar configuraciones o crear nuevos accesorios.

Pregunta	Dificultad	Éxito
¿Cuál es la temperatura en el salón?	Baja	Si
¿Cuál es la humedad en el salón?	Baja	Si
¿Podrías abrir la puerta del garaje?	Baja	Si
¿Podrías activar la calefacción para tener la casa en 21º?	Media	Si
(Dentro del menú de configuración del accesorio) ¿Cómo activarías las notificaciones para saber cuándo otra persona modifica la puerta del garaje?	Baja	Si
(Dentro del menú de configuración del accesorio) ¿Cómo cambiarias el nombre de la habitación en la que se encuentra este accesorio?	Baja	Si
(Con un papel con los datos del sensor de temperatura) ¿Serías capaz de crear un accesorio para controlar la temperatura del salón?	Alta	Si

8.3.2.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Una vez el usuario ha interactuado con la aplicación se recoge su satisfacción con el siguiente cuestionario:

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?		X		
¿Le resulta sencillo el uso de la aplicación?	X			
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca

<i>¿Funciona cada tarea como Vd. Espera?</i>	X			
<i>¿El tiempo de respuesta de la aplicación es muy grande?</i>	X			
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
<i>El tipo y tamaño de letra es</i>	X			
<i>Los iconos e imágenes usados son</i>	X			
<i>Los colores empleados son</i>	X			
Diseño de la Interfaz	Si	No	A veces	
<i>¿Le resulta fácil de usar?</i>	X			
<i>¿El diseño de las pantallas es claro y atractivo?</i>	X			
<i>¿Cree que el programa está bien estructurado?</i>	X			
Observaciones				
Le pareció muy cómodo de utilizar, pero no demostró gran interés en instalarlo en su casa.				

8.3.2.2.4 Cuestionario para el responsable de las Pruebas

Aspecto Observado	Notas
<i>El usuario comienza a trabajar de forma rápida por las tareas</i>	Si, tras una pequeña explicación de la app, el usuario la ha utilizado con rapidez.
<i>Tiempo en realizar cada tarea</i>	-
<i>Errores leves cometidos</i>	-
<i>Errores graves cometidos</i>	-

8.4 Pruebas de Rendimiento

Las pruebas de rendimiento han consistido en ejecutar HomePi Server en la Raspberry Pi y comprobar con las pruebas de integración y de sistema que funciona sin problemas.

La Raspberry Pi 3, está conectada en Internet mediante Wi-Fi, utiliza Raspbian como sistema operativo, sin cambiar nada de la configuración predeterminada, con la versión 7 de NodeJS instalada.

En HomePi hay 4 accesorios creados: puerta del garaje, sensor de temperatura, sensor de humedad y termostato. Por otra parte, para se ha comprobado que el plan “Spark” de Firebase es suficiente para el uso que se ha planeado, instalado en una casa, en mi caso donde vivimos 3 personas. Ya que, si es usado de forma simultánea por todos, 3 personas más HomePi Server, serían 4 conexiones a la base de datos cuando el plan permite hasta 100

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

9.1.1 Crear proyecto en Firebase

- Crear una cuenta en Google, si no se dispone de una.
- Ir a <https://console.firebaseio.google.com/> para entrar en la consola de Firebase
- Elegir la opción “Añadir proyecto”

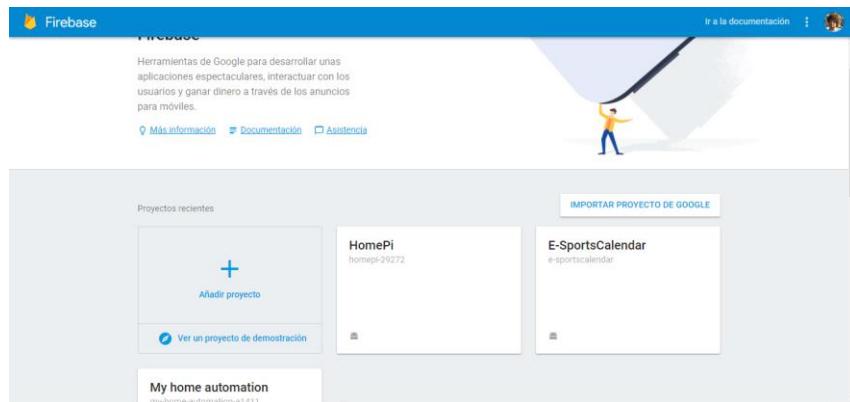


Ilustración 44. Consola de Firebase

- Rellenar los datos que se piden: nombre del proyecto y región (se recomienda elegir Europa).

9.1.2 Configurar HomePi Server

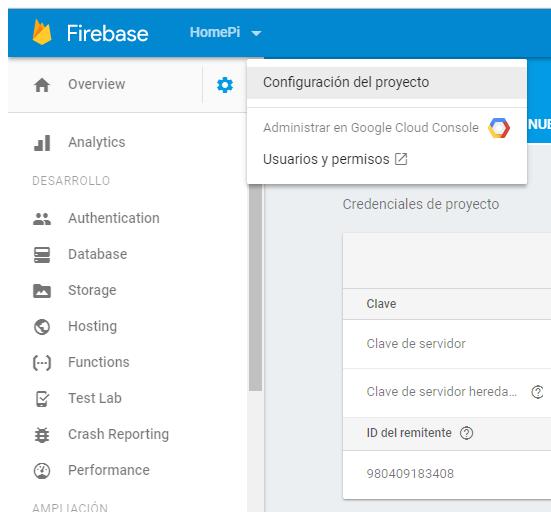
- Instalar NodeJS
 - Windows o Mac o Linux
 - Visitar la página oficial: <https://nodejs.org/es/download/>
 - Descargar el instalador adecuado a la versión del sistema operativo en la que quiere instalar.
 - Raspberry Pi
 - Conectarse por ssh o desde la terminal del escritorio para ejecutar los siguientes comandos:

```
curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
sudo apt-get install nodejs
```

- En la carpeta con el código de HomePi Server ejecutar para instalar todas las dependencias:

```
npm install
```

- Volver a la consola de Firebase



- Entrar en “Configuración del proyecto”.
- Elegir la pestaña “Cuentas de Servicio”.

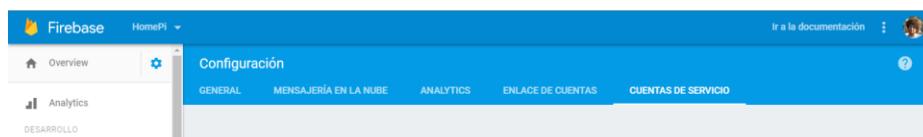


Ilustración 45. Configuración proyecto Firebase

- Seleccionar NodeJS entre los distintos lenguajes disponibles y pulsar sobre “Generar nueva clave privada”.

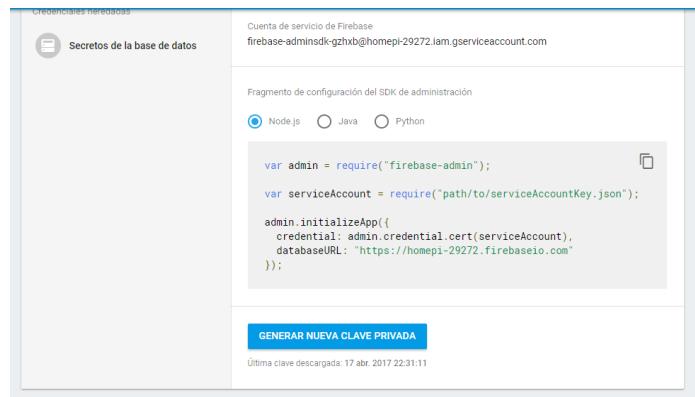


Ilustración 46. Generación de la clave privada de Firebase

- Descargar el fichero .json que se genera en dentro de la carpeta de HomePi Server.
- En la pestaña de “Mensajería en la nube” copiar la clave del servidor.
- Completar el fichero .env que se encuentra en la carpeta de HomePi Server.

```
CONFIGURATION_FILE=/<ruta_al_fichero .json>
DATABASE_URL=<aparece en la pantalla en la que se genera el fichero .json como
databaseURL>
SENDER_ID=<copiar la clave del servidor copiada de la pestaña de mensajería en la nube>
```

9.1.3 Configurar HomePi Android

- Ir a la consola de Firebase, <https://console.firebaseio.google.com/>

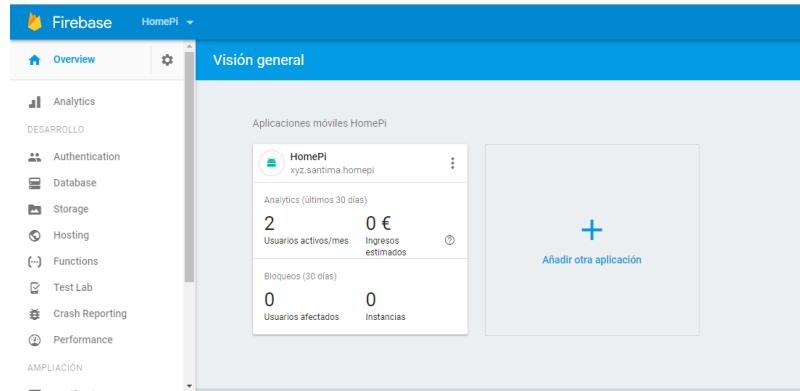


Ilustración 47. Vista general Consola de Firebase

- Elegir “Android” y llenar los campos:
 - Nombre de paquete de Android: xyz.santima.homepi
 - (opcional) Apodo de la aplicación: homePi
- Pulsar sobre “Registrar la aplicación” y en la siguiente ventana pulsar sobre “Descargar Google-services.json”.
- Ir a Google Play, <https://play.google.com/store/apps/details?id=xyz.santima.homepi>, para descargar la aplicación.

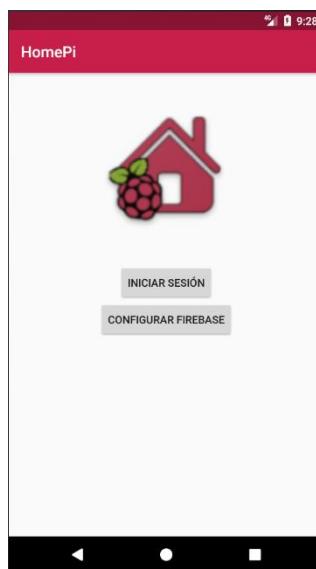


Ilustración 48. Pantalla inicial de configuración de HomePi Android

- Pulsar sobre “Configurar Firebase” y llenar los datos que se solicitan.

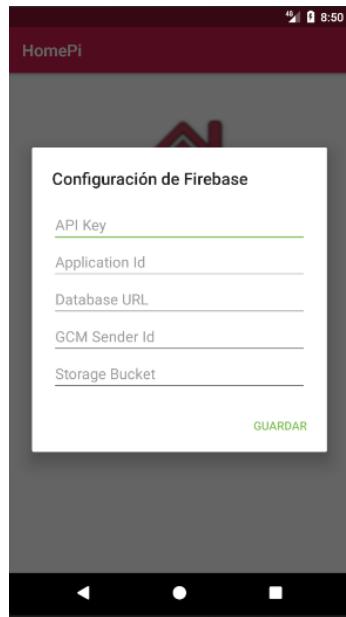


Ilustración 49. Pantalla para configurar el proyecto de Firebase en HomePi Android

- Para encontrar los datos que se nos solicitan se debe consultar el archivo “google-services.json” recién descargado.
 - El “GCM Sender ID” que se encuentra en la pestaña “Mensajería en la nube” dentro de la configuración del proyecto de Firebase (118 Ilustración 44 e Ilustración 45).

Ilustración 50. Google-services.json

9.1.4 Configurar HomePi Wemos

- Instalar IDE Arduino

- Copiar el contenido de la carpeta librerías en la carpeta libraries del IDE de Arduino y el contenido de la carpeta hardware en la carpeta hardware del IDE de Arduino, en el apartado 13.2 se explica donde se encuentran en el entregable, las carpetas del IDE de Arduino se encuentran en documentos o donde se elija en la instalación.
- Conectar sensor DHT22 o relé, si se compran las shields de wemos bastara con hacer coincidir los nombres de los pines.

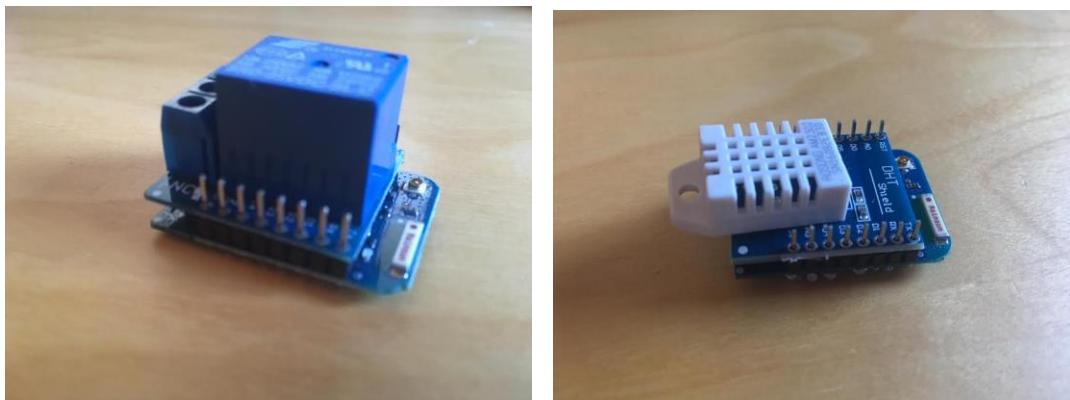


Ilustración 51. Wemos con shield instalada

- Abrir en IDE de Arduino el proyecto correspondiente al relé o DHT22.
- Pulsar sobre el botón de subir.
- Conectarse a la red WiFi sin contraseña que ha creado la placa wemos, el nombre cambiara en función del código que se cargue en la placa.

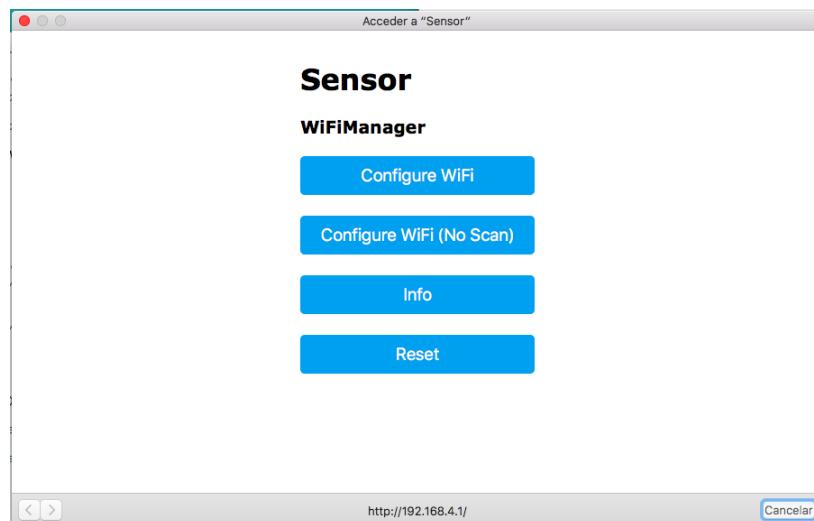


Ilustración 52. Menú configuración WiFi HomePi Wemos

- Entrar nuestro navegador y visitar <http://102.168.4.1/> donde pulsaremos en la opción de “Configure WiFi”.

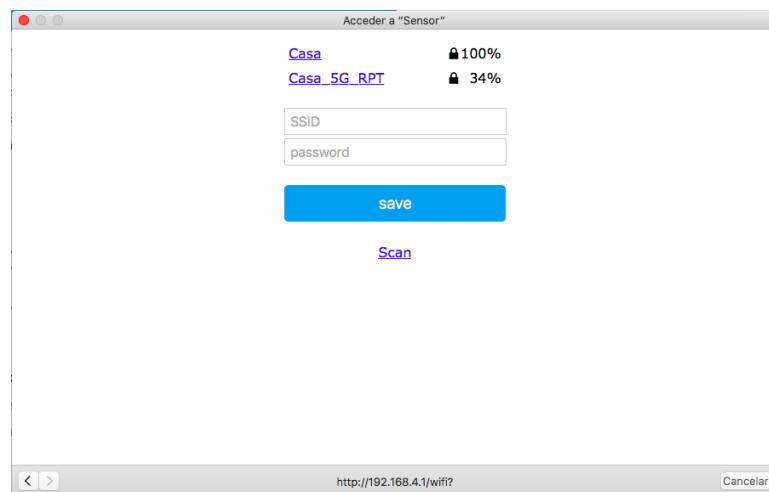


Ilustración 53. Lista de redes disponibles HomePi WiFi

- Elegir la red WiFi a la que queramos conectarnos, introducir la contraseña y pulsar el botón “save”.
- Ya podemos obtener información del sensor o controlar el relé.

9.2 Manual de Ejecución

9.2.1 Ejecutar

- En la carpeta donde se encuentra HomePi Server ejecutar.

```
npm run start
```

9.2.2 Detener

- En Windows
 - “ctrl” + “c”
 - a la pregunta: “¿Desea terminar el trabajo por lotes (S/N)?” responder pulsando “s”-
- En Mac
 - “Cmd” + “c”
- En Linux y Raspberry Pi
 - “Ctrl” + “c”

9.3 Manual de Usuario

9.3.1 Gestión usuarios

- Lo primero será visitar la consola de Firebase y entrar en nuestro proyecto creado anteriormente.
- Elegiremos la pestaña de “Authentication” del menú lateral.

The screenshot shows the Firebase Authentication console. On the left, there's a sidebar with 'Overview', 'Analytics', 'DESARROLLO' (Development) section containing 'Authentication' (selected), 'Database', 'Storage', 'Hosting', 'Functions', 'Test Lab', and 'Crash Reporting'. The main area has tabs for 'USUARIOS', 'MÉTODO DE INICIO DE SESIÓN', and 'PLANTILLAS'. It includes a search bar, a blue 'AÑADIR USUARIO' button, and a table with columns: Identificador, Proveedores, Fecha de creación, Inicio de sesión, and UID de usuario. One row is visible: 'santiagomartinagra@gmail.c...' with a mail icon, '12 jun. 2017', '22 jun. 2017', and 'IPRGNq0i7SVHCB6eewsrJPMWn5...'. Below the table are pagination controls: 'Filas por página: 50', '1-1 de 1', and arrows.

Ilustración 54. Consola de Firebase Authentication

- Para **crear un nuevo usuario**, pulsaremos sobre el botón de “Añadir usuario” e introduciremos el correo electrónico y contraseña deseada.
- Para **borrar usuario o cambiar contraseña**, colocaremos el ratón encima del usuario y al final nos aparecerán dos puntos, pulsando sobre ellos nos parecerán ambas opciones.

9.3.2 Crear accesorio



Ilustración 55. Pantalla inicial HomePi Android

- Pulsaremos el botón verde con el símbolo del más que hay en la pantalla inicial.

- Luego nos aparece la lista de accesorios que podemos crear, elegiremos el que queramos y pulsamos el botón crear.

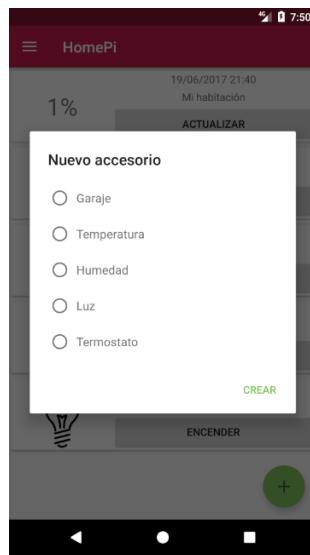


Ilustración 56. Elegir accesorio HomePi Android

- En función del accesorio que elegimos tendremos distintas opciones para configurar, para cualquiera podremos ponerle un nombre y decir en que habitación se encuentra.



Ilustración 57. Configurar nuevo accesorio HomePi Android

- Una vez completada toda la configuración pulsaremos el botón de volver, donde antes se nos preguntara si queremos guardar el accesorio.

9.3.3 Configurar accesorios

- Pulsando sobre cualquier accesorio en la pantalla principal accederemos a su configuración, donde podremos cambiar todos los ajustes de cuando lo creamos, borrarlo o activar/desactivar las notificaciones si el accesorio dispone de ellas.



Ilustración 58. Cambio de configuración accesorio HomePi Android

9.3.4 Cambiar proyecto de Firebase

- Pulsando el botón de menú que se encuentra arriba a la derecha de la ventana principal podemos acceder a la configuración de la aplicación.

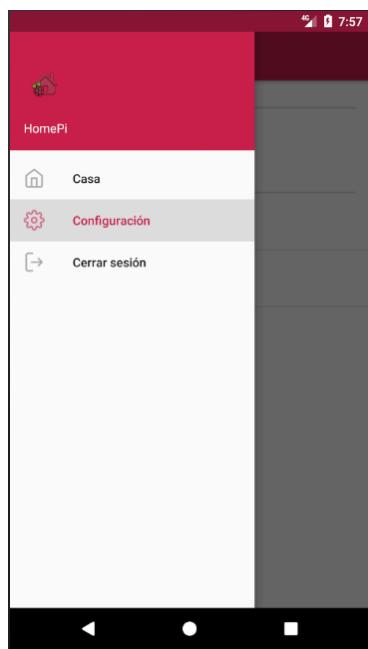


Ilustración 59. Menú HomePi Android

- Una vez en la configuración, podemos corregir o cambiar cualquiera de los parámetros de configuración del proyecto de Firebase.

9.3.5 Recuperar contraseña

- Si en algún momento olvidamos nuestra contraseña para acceder a HomePi, podemos pulsar el botón de “¿Algún problema al iniciar sesión?” que aparece en la pantalla donde introducimos la contraseña.

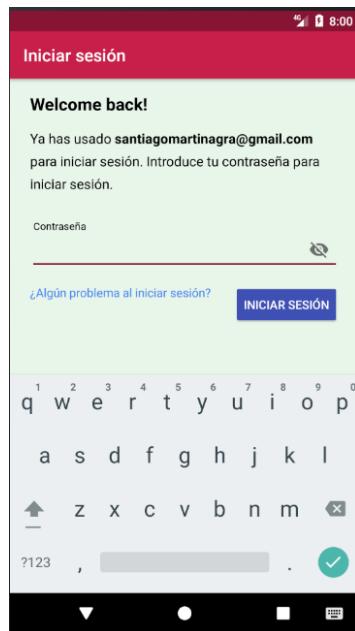


Ilustración 60. Pantalla de contraseña HomePi Android

- Con esto se nos enviara un correo a nuestra dirección de correo electrónico para que podamos restablecer la contraseña.

9.4 Manual del Programador

9.4.1 HomePi Server

El principal motivo para modificar HomePi Server será añadir compatibilidad con nuevos sensores, actuadores o accesorios.

Para cualquiera de los casos anteriores bastaría con crear una clases o clases con la lógica necesaria, siguiendo la estructura de paquetes que se detallara más adelante.

Todos los sensores y actuadores incluidos implementan su correspondiente interfaz para permitir una futura compatibilidad de los accesorios implementados con nuevas formas de leer sensores o manejar actuadores.

9.4.2 HomePi Android

Al igual que ocurre con HomePi Server, el principal motivo para modificar HomePi Android será añadir compatibilidad con nuevos sensores, actuadores y accesorios.

9.4.2.1 Nuevos accesorios

Si los layouts actuales que se encuentran en el directorio de layout de Android con los nombres: thermostat_card, text_status_card y img_status_card, no son adecuados se tendrá que crear uno nuevo con un CardView como base.

Lo siguiente será crear un nuevo ViewHolder para ese accesorio, los actuales se encuentran en ui.impl.holder, se incluye una interfaz, con métodos necesarios para que funcione sin necesidad de tocar nada más, y una clase abstracta con una implementación común de la mayoría de esos métodos.

Luego dentro de la clase ui.impl.adapter.OwnFirebaseRecycleViewAdapter en el método onCreateViewHolder se tendrá que añadir el nuevo ViewHolder.

Para permitir al usuario crear nuevos accesorios sería necesario añadir el nombre al array de accesorios del método OnClickFab del MainFragment que se encuentra en ui.fragment.

El siguiente paso sería modificar los layouts de configuración que se cargan desde la carpeta xml dentro de los recursos de Android.

Por ultimo en la clase ui.fragment.AccessoryConfigurationFragment en el método onCreatePreferenceFix, tendríamos que añadir el accesorio creado y utilizando el método initConfigSelectorDialog podremos crear los diálogos para elegir los sensores y actuadores necesarios.

9.4.2.2 *Nuevos sensores o actuadores*

Para la configuración en model.config se encuentran todas las interfaces y clases utilizadas para crear los sensores y actuadores disponibles, al igual que el caso de los accesorios con una jerarquía de interfaz y clase abstracta que facilita la implementación de nuevos.

Una vez creada la clase con la configuración tendríamos que añadirla dentro del paquete business.factory a la factoría adecuada según se trata de un sensor o un actuador.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Dado que uno de los principales objetivos era ampliar el prototipo inicial que solo contaba con el control de la puerta del garaje podría decir que se cumplió con creces.

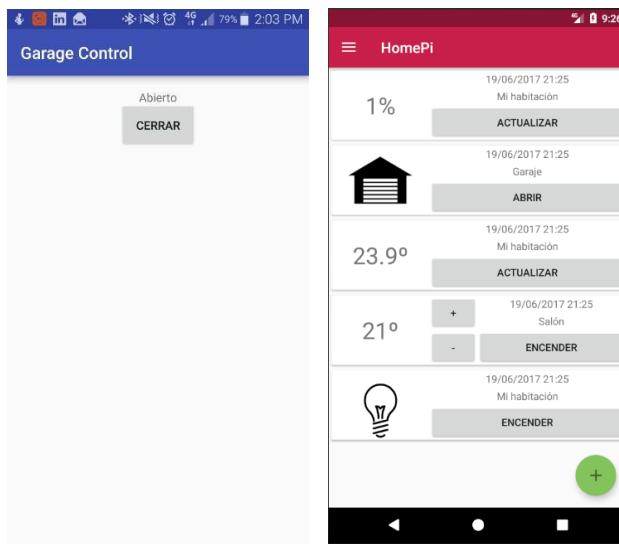


Ilustración 61. Aplicación prototípico vs HomePi Android

Ahora permite controlar puertas del garaje, luces, un termostato y obtener información de sensores de temperatura y humedad. Además, no solo controlar sino también gestionar estos accesorios permitiendo crear nuevos y cambiar configuraciones de los existentes.

Actualmente en casa están funcionando tanto los sensores de temperatura y humedad como el sistema para controlar la puerta del garaje que es utilizado diariamente. En el verano se aprovechará para instalar el termostato, porque no será necesaria la calefacción y porque se pretende diseñar e imprimir algún tipo de caja en 3D para poder instalarlo en la pared.

El sistema está preparado para admitir nuevos tipos de sensores, actuadores y accesorios de forma sencilla.

Se ha ampliado el conocimiento en multitud de herramientas utilizadas durante el desarrollo como pueden ser commitizen y semantic release, que hacen que el seguimiento del desarrollo por terceras personas sea más sencillo, así como consultar commits anteriores y entender fácilmente qué cambios se realizaron.

También se ha adquirido mayor conocimiento tanto en Java para Android como en TypeScript para NodeJS.

10.2 Ampliaciones

En este apartado se van a indicar posibles ampliaciones que podrían mejorar el proyecto y que quedan pendientes para el futuro.

10.2.1 Integración con HomeKit de Apple

Tal y como se ha diseñado el sistema, con los sensores y actuadores siendo controlados por HTTP, se puede utilizar gracias al proyecto homebridge, <https://github.com/nfarina/homebridge>, que existen varios plugin creados por la comunidad que permiten crear accesorios comunicándose por HTTP con los sensores y actuadores.

No obstante en un futuro se podría integrar directamente el framework de HomeKit en HomePi para poder controlarlo todo desde una única aplicación.

10.2.2 Integración con asistentes

Debido a popularidad que están ganando dispositivos como el Amazon Echo para usar Alexa y Google Home para usar Google Assistant y dado que ambos asistentes poseen de APIs para desarrolladores se podría crear un nuevo módulo de HomePi que permita controlar los accesorios con la voz.

10.2.3 NFC

Muchos smartphones poseen lector NFC y se podría aprovechar para expandir las posibilidades de HomePi.

Se podría añadir la posibilidad de configurar nuevos accesorios o la configuración inicial de aplicación con Firebase simplemente leyendo una etiqueta NFC en la que se han guardado previamente esos datos, lo que facilitaría la configuración.

También se pueden utilizar para automatizar tareas, por ejemplo: con una etiqueta NFC en la mesilla de noche que al colocar el teléfono sobre ella se baje la calefacción y se apaguen todas las luces.

10.2.4 Awareness API de Google

En Android está disponible la Awareness API de Google que permite reaccionar a eventos basados en la ubicación, por ejemplo, el irse de casa para apagar la colección o al estar cerca mostrar una notificación con un acceso rápido a abrir la puerta del garaje o directamente abrir la puerta del garaje.

10.2.5 Macros o acciones grabadas o automáticas

Poder crear conjuntos de acciones que se puedan ejecutar de manera sencilla, por ejemplo: una para antes de irnos a dormir que baje la calefacción y apague todas las luces o la inversa para cuando nos levantemos.

Pero no solo el dar la opción de ejecutar estos conjuntos de tareas sino el poder automatizar su ejecución a una hora concreta o con las ampliaciones anteriores referentes al NFC y API Awareness.

10.2.6 Roles

Poder establecer roles dentro de los usuarios de HomePi, limitando desde el uso de algunos accesorios por horas o directamente no permitiendo el control del mismo.

También estaría bien la opción de ocultar accesorios para que no puedan ser vistos por el resto de usuarios.

10.2.7 Compatibilidad con Amazon Dash

Amazon Dash son unos botones creados por Amazon para permitir comprar productos de forma rápida, la idea es colocar estos botones cerca de donde tengamos el producto que van a comprar para así cuando nos quedemos sin el poder comprarlo pulsando el botón.



Ilustración 62. Amazon Dash

Estos botones funcionan por WiFi y con muy baratos, solo 5€, y dado que existen librerías para utilizarlos de forma personalizada y no para comprar en Amazon, se podrían utilizar para controlar cualquier accesorio de HomePi sin necesidad del móvil.

También existe una versión de los Amazon Dash preparada para IoT con Amazon Web Services, pero tiene el inconveniente de ser bastante más cara, 25€, que la preparada para comprar.



Ilustración 63. Amazon Dash IoT

Capítulo 11. Planificación del Proyecto y Presupuesto finales

La planificación y presupuesto que se encuentra en este capítulo refleja la evolución sufrida por la planificación y presupuesto iniciales a lo largo del proyecto.

Los principales cambios han sido la reducción de algunas tareas, debido principalmente a la falta de tiempo que ha provocado que de las 360 horas planificadas en 2 meses se realicen solo 300 horas en poco más de un mes.

Los cambios sufridos en la planificación se deben a varios factores:

- **Clases de 4º.** Aunque en el último semestre de 4º solo hay dos asignaturas, están requieren mucho tiempo, debido a las múltiples entregar que se realizan de manera semanal.
- **Prácticas en Simbiosys.** Lo que empezó como unas prácticas curriculares de 5 horas al día de lunes a viernes durante 1 mes y 1 semana se amplió a prácticas extracurriculares con el mismo horario hasta agosto, donde se transformara en un contrato indefinido.
- **Participación en el programa e².** El programa desarrollado por la colaboración de la Universidad de Oviedo, el CISE (Centro Internacional Santander Emprendimiento) y Oviedo Emprende tenía como objetivo crear grupos de 4 estudiantes de diferentes titulaciones, un alumno tutor y un empresario de la región para desarrollar un proyecto. En mi caso fui elegido como alumno tutor lo que conllevo 40 horas de formación previa más las 150 horas de trabajo en el equipo para desarrollar el proyecto.
- **Desarrollo de proyecto personal.** Junto con un compañero de la carrear decidimos embarcarnos en el desarrollo de una aplicación para Android, E-Sports Calendar. La idea de la aplicación es reunir la información de diferentes competiciones de E-Sports, o deportes electrónicos, en una sola aplicación al igual que otras aplicaciones hacen con deportes tradicionales. El proyecto evoluciono de una sola aplicación a una API con su correspondiente gestor de contenidos que permite manejar la información que se muestra en la aplicación y ahora mismo nos encontramos en busca de financiación para poder dejar de ser un desarrollo por hobby a algo más serio.

En el caso del presupuesto final se verá afectado por la reducción de horas, así como la necesidad de adquirir un nuevo sensor debido a que el presupuestado originalmente no daba valores correctos.

11.1 Planificación Final

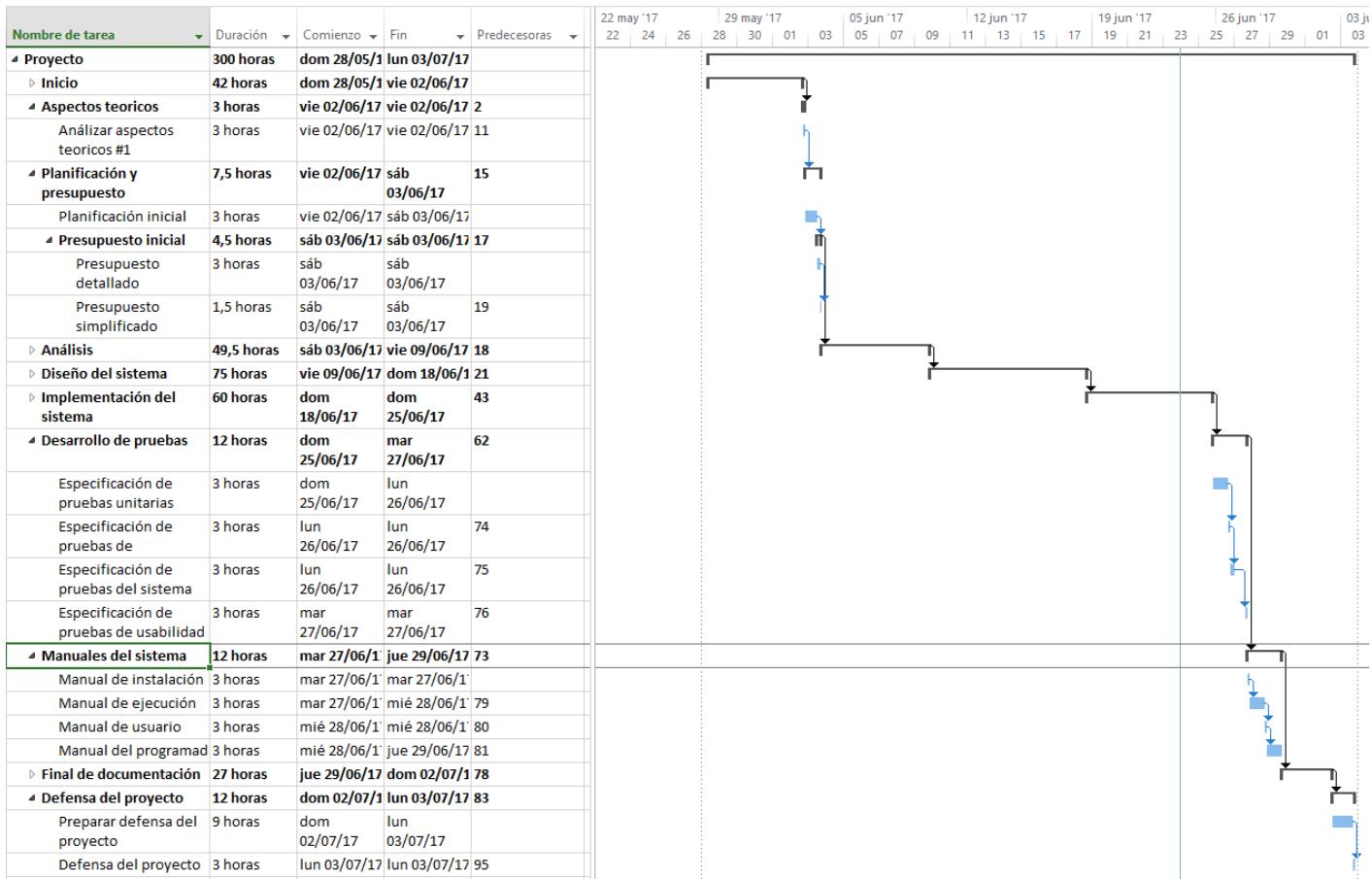


Ilustración 64. Vista general planificación final

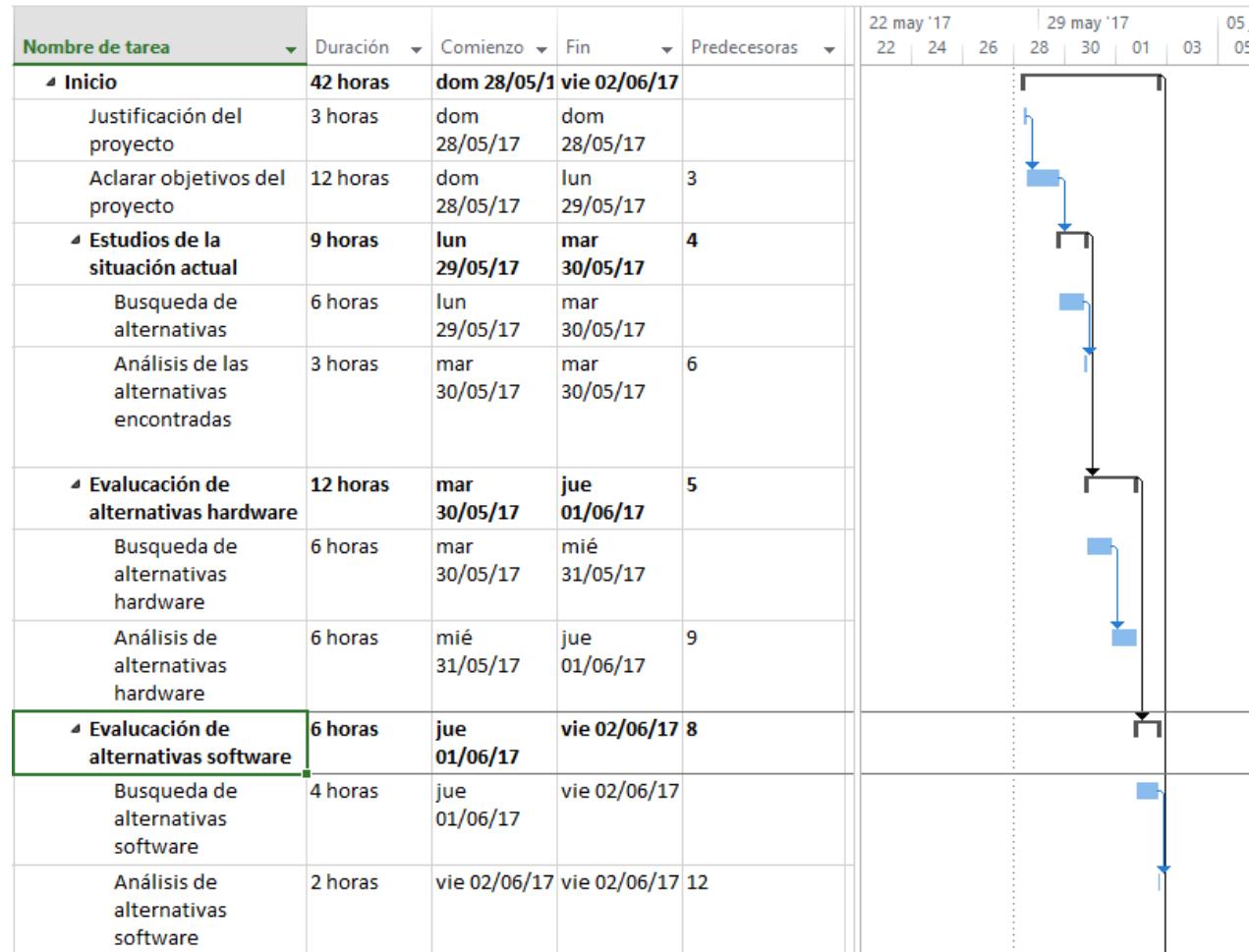


Ilustración 65. Plantificación final tarea: Inicio

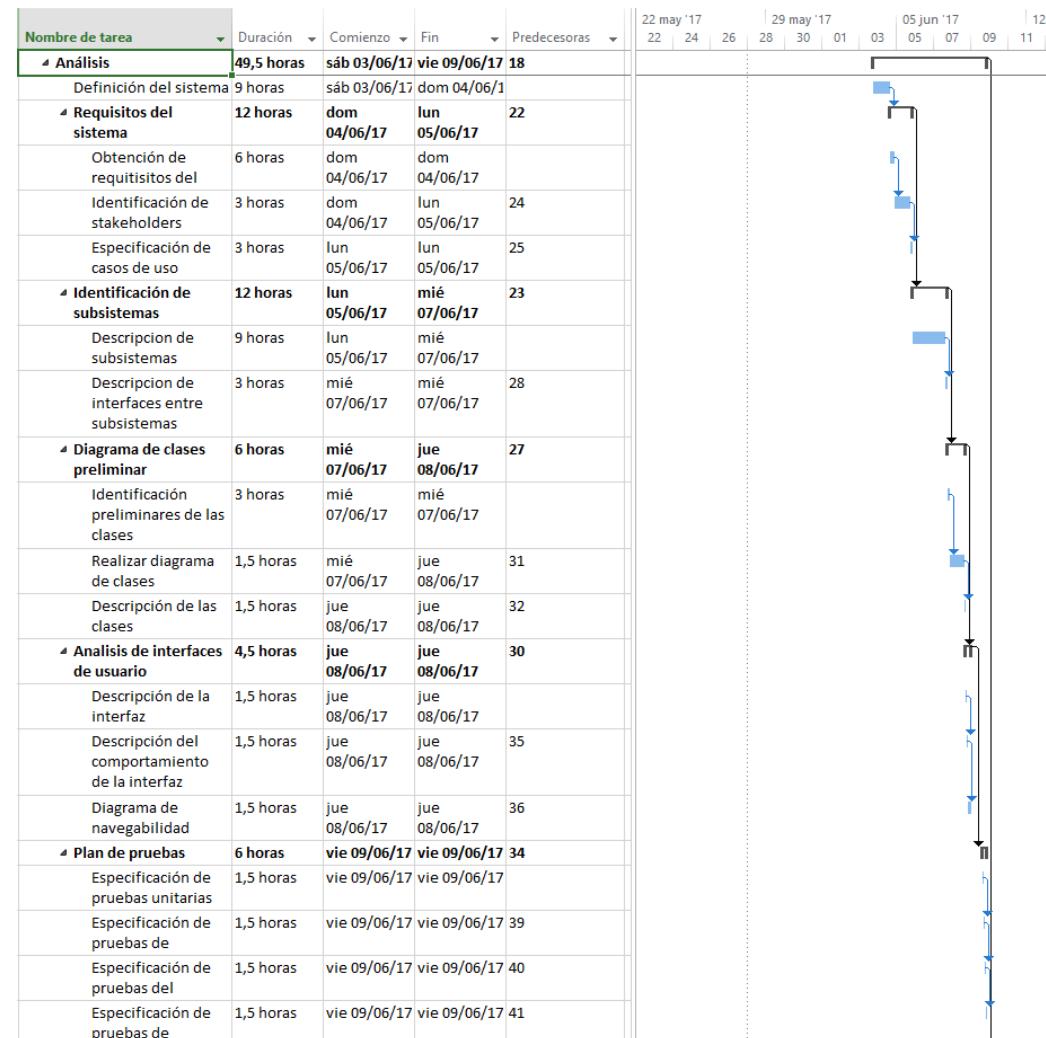


Ilustración 66. Planificación final tarea: Análisis

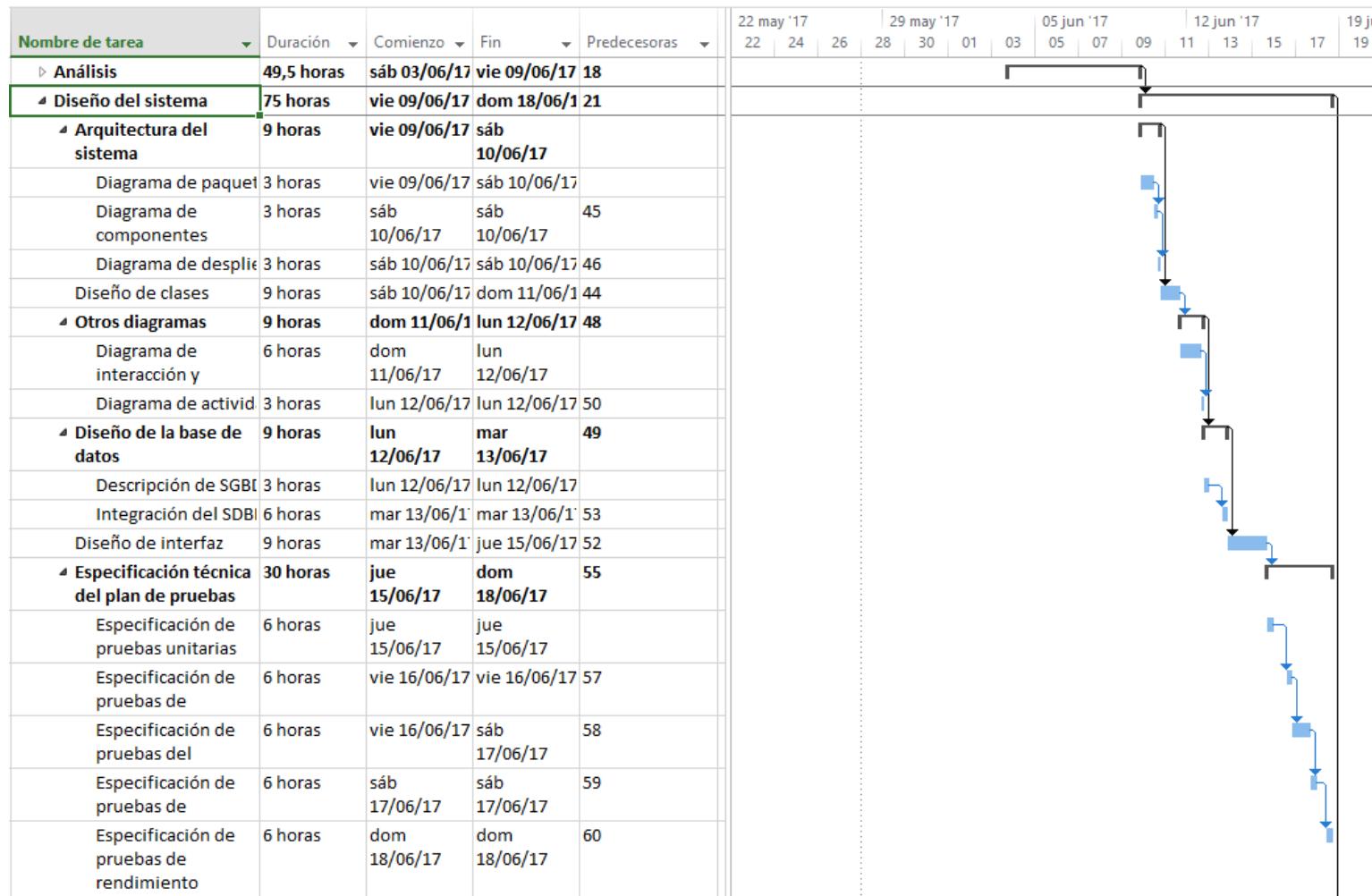


Ilustración 67. Planificación final tarea: Diseño del sistema

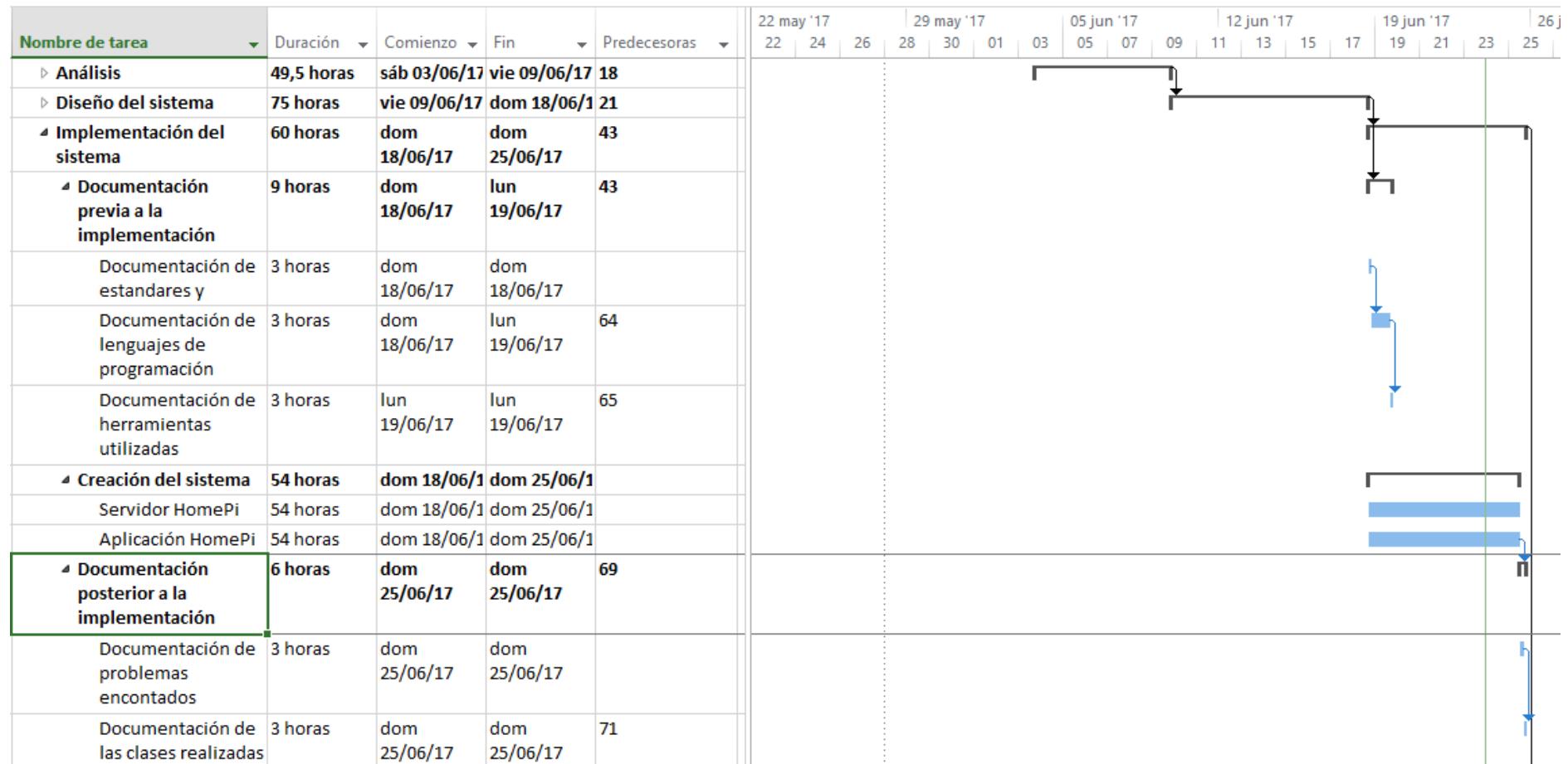


Ilustración 68. Planificación final tarea: Implementación del sistema

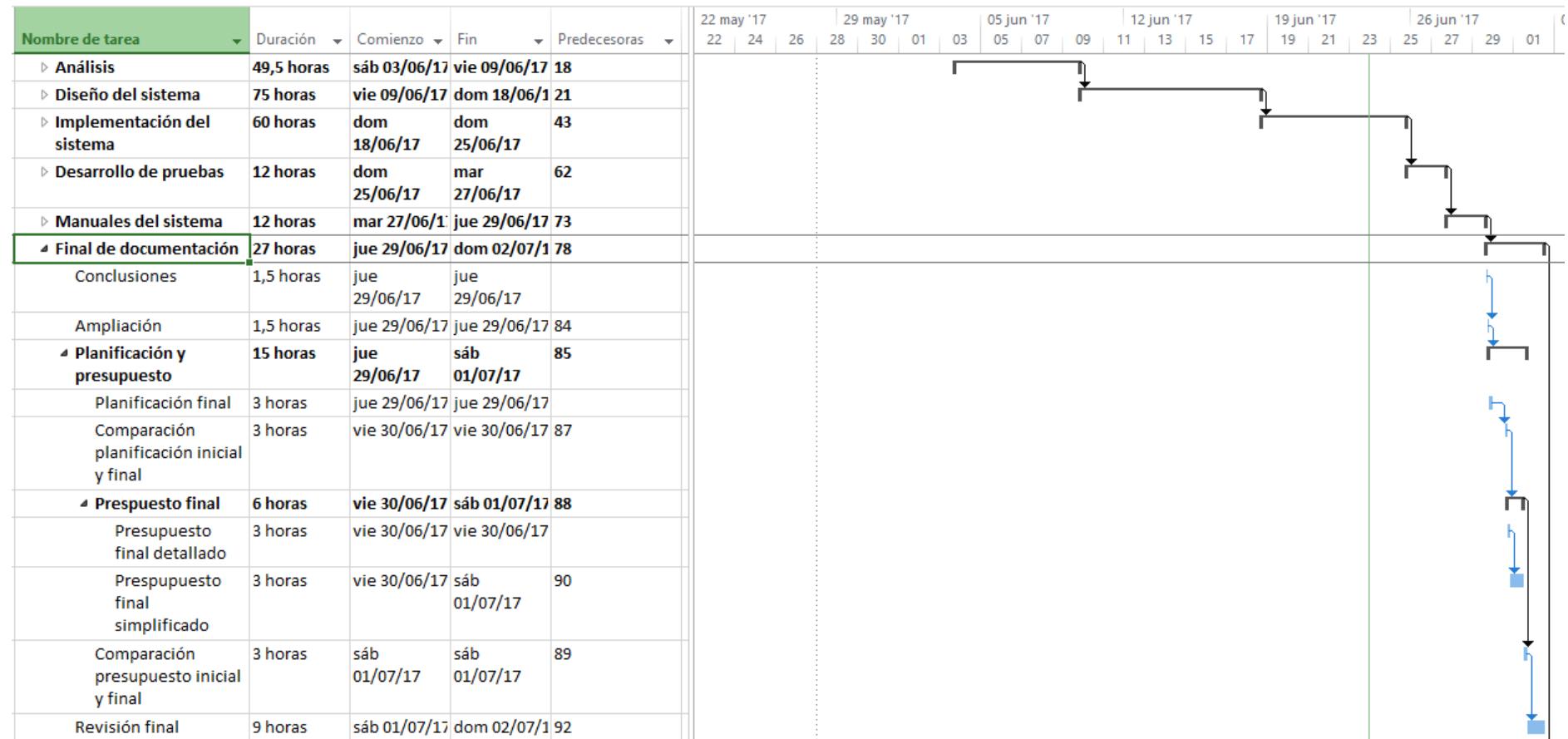


Ilustración 69. Planificación final tarea: Final de documentación

11.2 Presupuesto Final

Debido a la reducción de horas del trabajo el presupuesto se ha reducido de 25.156,03€ a 21.168,46€.

Aunque en el presupuesto inicial no se contemplaba la compra de un segundo sensor de temperatura, por fallos en el primero, la reducción de horas produce una diferencia mayor entre ambos presupuestos que el coste del sensor, por lo que al final el proyecto continúa siendo rentable con 1.590,42€ de beneficio.

11.2.1 Desarrollo de Presupuesto Detallado (Empresa)

Tarea	Horas	Precio/Hora	Total
Inicio	42,00	50,00 €	2.100,00 €
Aspectos teóricos	3,00	50,00 €	150,00 €
Planificación y presupuesto	7,50	50,00 €	375,00 €
Análisis	49,50	50,00 €	2.475,00 €
Diseño	75,00	50,00 €	3.750,00 €
Implementación	60,00	50,00 €	3.000,00 €
Pruebas	12,00	50,00 €	600,00 €
Manuales	12,00	50,00 €	600,00 €
Fin del proyecto	39,00	50,00 €	1.950,00 €
Total horas:		300,00	Total precio: 15.000,00 €

Tabla 11-1. Coste final detallado: tareas

Hardware	Precio unitario	Cantidad	Vida útil (meses)	Coste para el proyecto
Portátil	1.500,00 €	1	60	50,00 €
Raspberry Pi 3	40,00 €	1		40,00 €
Wemos D1 mini pro	5,01 €	3		15,03 €
Sensor temperatura DHT11	2,28 €	1		2,28 €
Sensor temperatura DHT22	4,08 €	1		4,08 €
Rele	1,92 €	1		1,92 €
Sensor luminosidad	4,50 €	1		4,50 €
Sensor distancia	4,50 €	1		4,50 €
			Total precio:	122,31 €

Tabla 11-2. Coste final detallado: hardware

Sofware	Precio unitario	Cantidad	Vida útil (meses)	Coste para el proyecto
Firebase	0,00 €	1		0,00 €
macOS Sierra	0,00 €	1		0,00 €
Microsoft Windows 10 Pro	279,00 €	1	30	18,60 €
Microsoft Office 360	69,00 €	1	12	11,50 €
Microsoft Project	119,00 €	1	30	7,93 €
Raspbian	0,00 €	1		0,00 €
Google Drive	14,00 €	1	2	14,00 €
GitHub	0,00 €	1		0,00 €
WebStorm	59,00 €	1	12	9,83 €
Arduino IDE	0,00 €	1		0,00 €
VisualStudio Code	0,00 €	1		0,00 €
Android IDE	0,00 €	1		0,00 €
			Total precio:	61,87 €

Tabla 11-3. Coste final detallado: Software

Gastos mensuales fijos	Meses	Precio/Mes	Total
Conexión a Internet	2	60,00 €	120,00 €
Electricidad	2	150,00 €	300,00 €
Calefacción	2	50,00 €	100,00 €
Agua	2	50,00 €	100,00 €
Cuota de autonomo	2	50,00 €	100,00 €
		Total precio:	720,00 €

Tabla 11-4. Gastos finales mensuales fijos

Subtotal	15.904,18 €
Beneficio (10%)	1.590,42 €
IVA (21%)	3.673,87 €
Total	21.168,46 €

Tabla 11-5. Presupuesto total final del proyecto

El coste total final del proyecto considerando un beneficio del 10%, aplicando el IVA correspondiente y ajustando los cambios de horas queda en un total de **21.168,46€**.

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

International Telecommunication Union, Overview of the Internet of things, (2012) 14. <https://www.itu.int/rec/T-REC-Y.2060-201206-I>.

L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Comput. Networks. 54 (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.

K. Ashton, That “Internet of Things” thing, RFID J. 22 (2009) 97–114. <http://www.itrco.jp/libraries/RFIDjournal-That Internet of Things Thing.pdf>.

D. McFarlane, S. Sarma, J.L. Chirn, C.Y. Wong, K. Ashton, Auto ID systems and intelligent manufacturing control, Eng. Appl. Artif. Intell. 16 (2003) 365–376. doi:10.1016/S0952-1976(03)00077-0.

C. González García, MIDGAR: Interoperabilidad de objetos en el marco de Internet de las Cosas mediante el uso de Ingeniería Dirigida por Modelos, University of Oviedo, 2017. doi:10.13140/RG.2.2.26332.59529.

G. Sánchez-Arias, C. González García, B.C. Pelayo G-Bustelo, Midgar: Study of communications security among Smart Objects using a platform of heterogeneous devices for the Internet of Things, Futur. Gener. Comput. Syst. 74 (2017) 444–466. doi:10.1016/j.future.2017.01.033.

C. González García, J.P. Espada, MUSPEL: Generation of Applications to Interconnect Heterogeneous Objects Using Model-Driven Engineering, in: V.G. Díaz, J.M.C. Lovelle, B.C.P. García-Bustelo (Eds.), Handb. Res. Innov. Syst. Softw. Eng., IGI Global, 2015: pp. 365–385. doi:10.4018/978-1-4666-6359-6.ch15.

C. González García, D. Meana-Llorián, B.C.P. G-Bustelo, J.M.C. Lovelle, A review about Smart Objects, Sensors, and Actuators, Int. J. Interact. Multimed. Artif. Intell. 4 (2017) 7–10. doi:10.9781/ijimai.2017.431.

C. González García, J.P. Espada, Using Model-Driven Architecture Principles to Generate Applications based on Interconnecting Smart Objects and Sensors, in: V.G. Díaz, J.M.C. Lovelle, B.C.P. García-Bustelo, O.S. Martinez (Eds.), Adv. Appl. Model. Eng., IGI Global, 2014: pp. 73–87. doi:10.4018/978-1-4666-4494-6.ch004.

C. González García, MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos IoT, University of Oviedo, 2013. doi:10.13140/RG.2.1.1466.4727.

C. González García, C.P. García-Bustelo, J.P. Espada, G. Cueva-Fernandez, Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios, Comput. Networks. 64 (2014) 143–158.
doi:10.1016/j.comnet.2014.02.010.

12.2 Referencias en Internet

[Microsoft] Microsoft. Información sobre TypeScript. <https://www.typescriptlang.org/>. 2017 [Consultada por última vez el 25/07/2017]

[Google] Google. Información sobre Firebase. <https://firebase.google.com/>. 2017 [Consultada por última vez el 25/07/2017]

[Google] Google. Ejemplos Firebase. <https://github.com/firebase/quickstart-nodejs>. 2017 [Consultada por última vez el 25/07/2017]

[Google] Google. Android Things. https://developer.android.com/things/preview/index.html#flash_android_things. 2017 [Consultada por última vez el 15/07/2017]

[Google] Google. Documentación sobre Android. <https://developer.android.com/>. 2017 [Consultada por última vez el 25/07/2017]

[Raspberry Pi Foundation] Raspberry Pi Foundation. <https://www.raspberrypi.org/>. 2017

[StackOverflow] StackOverflow. Página de preguntas y respuestas para programadores. <https://stackoverflow.com>

[Wemos] Wemos. Página oficial de las placas Wemos. <https://www.wemos.cc/>. 2017

[Wemos] Wemos. Ejemplos. https://github.com/wemos/D1_mini_Examples. 2017 [Consultada por última vez el 14/07/2017]

[Wikipedia] Wikipedia. <https://es.wikipedia.com/>. 2017

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

- **Accesorio:** Conjunto de uno o más sensores y/o actuadores preparados para realizar una tarea, por ejemplo: encender una luz, leer un sensor o controlar la calefacción.
- **Actuador:** dispositivo capaz de generar un efecto en su entorno, por ejemplo: relé, motor...
- **Sensor:** Dispositivo que capta información sobre su entorno.

13.2 Contenido Entregado en el Archivo adjunto

13.2.1 Contenidos generales

Directorio	Contenido
<i>./ Directorio raíz del Archivo adjunto</i>	Contiene un fichero leeme.txt explicando toda esta estructura. Se puede incluir autorun e icono del proyecto si existe.
<i>./documentacion</i>	Contiene toda la documentación asociada al proyecto.
<i>./documentacion/uml</i>	Contiene todas las imágenes de los diagramas utilizados en la documentación.
<i>./presentacion</i>	Directorio que contiene la presentación en Powerpoint o equivalente utilizada el día de la defensa del proyecto, si está disponible en el momento de realizar el Archivo adjunto.
<i>./HomePi</i>	Contiene los 3 subsistemas que componen HomePi.

13.2.2 Subsistemas

13.2.2.1 HomePi Android

Se encuentra en la ruta “HomePi/HomePi Android”.

Directorio	Contenido
<i>./doc</i>	Contiene la documentación del subsistema.
<i>./build</i>	Contiene los archivos generados por Android Studio al realizar una compilación.
<i>./app</i>	Contiene el código de la aplicación Android con la estructura de Android Studio.

13.2.2.2 HomePi Server

Se encuentra en la ruta “HomePi/HomePi Server”.

Directorio	Contenido
<i>./docs</i>	Contiene la documentación del subsistema.
<i>./src</i>	Contiene el código TypeScript que compone el sistema.
<i>./test</i>	Contiene el código de los test unitarios realizados sobre los accesorios.
<i>./app</i>	Contiene el código de HomePi Server compilado según el estándar ES5 para ser ejecutado con NodeJS.
<i>./.env</i>	El archivo que contiene parte de la configuración de HomePi Server

13.2.2.3 HomePi Wemos

Se encuentra en la ruta “HomePi/HomePi Wemos”.

Directorio	Contenido
<code>./librerias</code>	Contiene las librerías necesarias para poder compilar y subir el código a las placas wemos.
<code>./hardware</code>	Contiene los drivers de hardware y librerías para poder utilizar las placas wemos.
<code>./humidity_temperature_server</code>	Contiene el código para subir a una placa wemos y que mediante peticiones HTTP lea los datos de un sensor de temperatura y humedad DHT22.
<code>./rele_server</code>	Contiene el código para subir a una placa wemos y que mediante peticiones controle un relé.
<code>./obstacle_server</code>	Contiene el código para subir a una placa wemos para obtener las lecturas de un sensor de obstáculos IR
<code>./photoresistor_server</code>	Contiene el código para subir a una placa wemos para obtener las lecturas de un fotorresistor.

13.3 Código Fuente

El código fuente se incluirá en el archivo adjunto que se ha de subir. En esta sección se indicará la ruta donde se puede encontrar el código fuente y se describirá brevemente su estructura por carpetas y que contiene cada una.