UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Sistema Domótico Inteligente
Documentación Técnica

Presentado por David Colmenero Guerra
en Universidad de Burgos — 5 de enero de 2021
Tutor: Álvar Arnaiz-González
Tutor: Alejandro Merino Gómez

Índice general

Índice general I

Índice de figuras III

Apéndice A Plan de Proyecto Software 1
A.1. Introducción 1
A.2. Planificación temporal 1
A.3. Estudio de viabilidad 10

Apéndice B Especificación de Requisitos 19
B.1. Introducción 19
B.2. Objetivos generales 19
B.3. Catálogo de requisitos 19
B.4. Especificación de requisitos 22

Apéndice C Especificación de diseño 29
C.1. Introducción 29
C.2. Diseño de datos 29
C.3. Diseño procedimental 29
C.4. Diseño arquitectónico 29

Apéndice D Documentación técnica de programación 31
D.1. Introducción 31
D.2. Estructura de directorios 31
D.3. Manual del programador 35
D.4. Compilación, instalación y ejecución del proyecto 35
D.5. Pruebas del sistema 35

Apéndice E Documentación de usuario **37**
E.1. Introducción 37
E.2. Requisitos de usuarios 37
E.3. Instalación 37
E.4. Manual del usuario 37

Bibliografía **39**

Índice de figuras

A.1. Gráfico Burndown sprint1. 3
A.2. Gráfico Burndown sprint2. 3
A.3. Gráfico Burndown sprint3. 4
A.4. Gráfico Burndown sprint4. 5
A.5. Gráfico Burndown sprint5. 5
A.6. Gráfico Burndown sprint6. 6
A.7. Gráfico Burndown sprint7. 7
A.8. Gráfico Burndown sprint8. 7
A.9. Gráfico Burndown sprint9. 8
A.10.Gráfico Burndown sprint10. 9
A.11.Gráfico Burndown sprint11. 9
A.12.Compatibilidad entre licencias. 16

B.1. Casos de uso 1/2. 23
B.2. Casos de uso 2/2. 23

D.1. Partes del proyecto. 32

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En cualquier proyecto de este tipo es relevante incluir cierta información sobre la planificación, viabilidad y análisis de costes en el desarrollo del mismo con la finalidad de conseguir una visión de cómo ha ido evolucionando el proyecto y los ciclos de trabajo que se han llevado a cabo.

He de señalar que en la última tercera parte de los sprints, no siempre he contado con una línea de datos con la que poder replicar datos en GitHub, por lo que se han generado menos commits de los que se hubiera hecho si hubiera contado con medios a mi alcance. En cualquier caso, se ha intentado maximizar el número de puntos de salvado para que quedase constancia de la evolución del proyecto. Además, al redactar el proyecto en LaTeX desde la plataforma de Overleaf no refleja el tiempo real invertido puesto que no se realizan cambios en el repositorio de forma automática, hasta que compilo, descargo y actualizo el repositorio manualmente.

En el primer apartado trataremos la **planificación temporal**, donde podremos ver como han evolucionado los tiempos de trabajo durante las semanas en las que se ha trabajado en el proyecto. En el segundo apartado se reflejará un **estudio de viabilidad** sobre el proyecto, que a su vez incluye dos partes, la parte económica y la parte legal.

A.2. Planificación temporal

Al comenzar el proyecto se determinó que se utilizaría una metodología ágil para hacer el desarrollo del proyecto pero esta decisión ha terminado

siendo un hándicap producido por la falta de documentación y la dudosa credibilidad de muchas páginas web y documentación que he encontrado, y se ha traducido en una gran inversión en tiempo.

Ante todo, se ha intentado seguir un mínimo de pautas **Scrum** [15] pese a no existir un grupo de trabajo real con unas tareas diarias y roles definidos:

- Las tareas fueron siempre semanales en forma de «sprints».
- Al finalizar el sprint se hace la entrega del trabajo elaborado en la semana y se determinan las próximas tareas.
- Tras determinar las tareas se definen los «milestones» y los «issues».
- Para hacer el seguimiento de las tareas se ha utilizado en tablero **Kanban** de **ZenHub**.
- Tras finalizar el sprint se puede comprobar el trabajo mediante los *gráficos burndown*.

Las reuniones de trabajo han sido consensuadas y planificadas para realizarlas los jueves de cada semana, con el siguiente resultado:

Sprint 00 - 01/10/2020 - 08/10/2020

En la primera reunión nos reunimos Álvar Arnaiz González y yo. En ella hice la propuesta de temática del TFG y, además expuse diferentes ideas y enfoques sobre el proyecto en la que se determinó a grandes rasgos la posible viabilidad del proyecto. Los objetivos de este «sprint» introductorio fueron la búsqueda de repositorios y otros proyectos para tomar ideas o si era posible hacer algún fork. También, el estudio la estructura del proyecto y herramientas software e interfaces a utilizar.

Sprint 01 - 09/10/2020 - 15/10/2020

Tras determinar el enfoque final de proyecto, se consensuó que los objetivos de este sprint fueron la búsqueda de información relevante, otros proyectos y recursos que puedan servir de utilidad como pueden ser APIS. Se buscaron algunas páginas web para realizar pruebas de extracción de datos con «web scraping» [1] y se descompusieron las tareas en unidades pequeñas de trabajo para poder afrontarlas durante sprints.

[1] Se trata brevemente en el punto 4 de la memoria

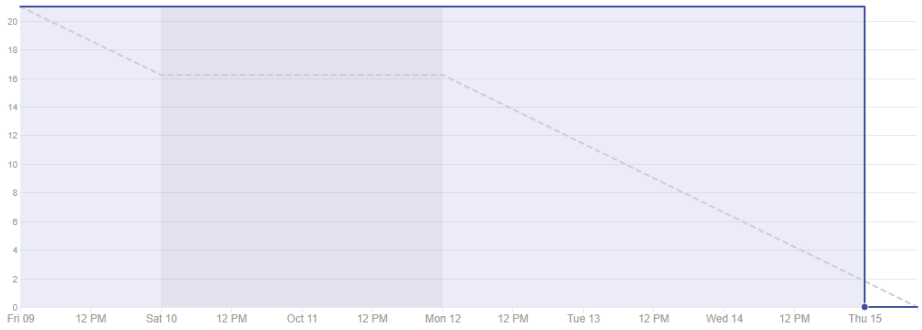


Figura A.1: Gráfico Burndown sprint1.

En este sprint también generé el repositorio en GitHub para poder trabajar contra él y podemos ver las líneas de código utilizadas en el primer *issue*.

Sprint 02 - 16/10/2020 - 22/10/2020

Los objetivos de este sprint fueron el buscar repositorios, APIS, servicios y tecnologías con las que darle forma al proyecto. Se decidió generar algún tipo de aplicación desde la que poder controlar la instalación haciéndola más amigable y con mejor capacidad de interacción. Se investigan opciones.

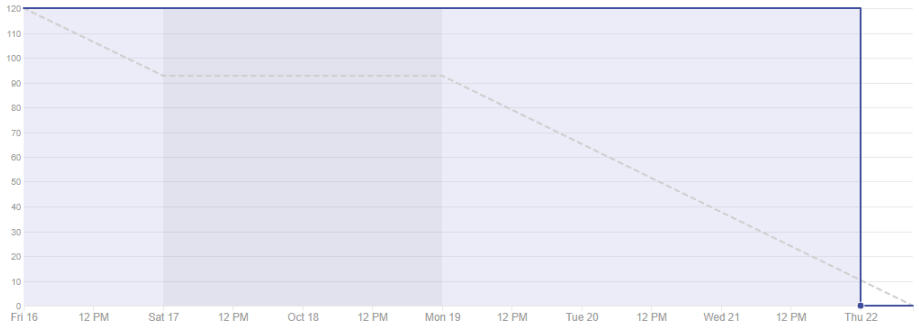


Figura A.2: Gráfico Burndown sprint2.

En el issue 2 se hizo una búsqueda por la web para buscar repositorios y proyectos sobre domótica

En el issue 3 buscamos algunas APIS para orientar el proyecto hacia el uso de APIS destinadas a ofrecer información que pueda servirnos y se realizaron algunas pruebas.

En el issue 4 se realizó un pequeño estudio de las tecnologías que podríamos utilizar, desde el procesador de textos, que finalmente se utilizó LaTeX, como la revisión de las normativas electrotécnicas que debía seguir en el proyecto y también se determinó la utilización de Telegram para interactuar con el Sistema Domótico.

Sprint 03 - 23/10/2020 - 29/10/2020

Los objetivos de este sprint fueron determinar los componentes hardware necesarios para completar el proyecto y se realiza la compra de material. En todos los «issues» del «milestone 3» podemos ver múltiples enlaces, comparativas, justificaciones e imágenes en las que se ha basado toda la instalación física posterior.

Figura A.3: Gráfico Burndown sprint3.

Sprint 04 - 30/10/2020 - 05/10/2020

Se incorpora D.Alejandro Merino Gómez al que se le presenta el proyecto y se le concede acceso a los repositorios para que pueda co-tutorizar el proyecto.

Los objetivos de este «milestone» fueron todos de la parte física de la instalación:

Podemos ver en el issue 15 que finalicé la tirada de cable junto a un diagrama explicativo del funcionamiento de un pulsador de 3 posiciones (Posición 1, Reposo y Posición 2. Teniendo en cuenta que las dos posiciones que dan continuidad al circuito tienen invertida la polaridad).

También vemos en el issue 16 la presentación de la Raspberry Pi que utilizaremos, los conectores del tipo JST-XH que se han utilizado en el

crimpado de los cables electrónicos, los cables finalizados con sus conectores y la disposición final de la Raspberry Pi en su ubicación final de la instalación.

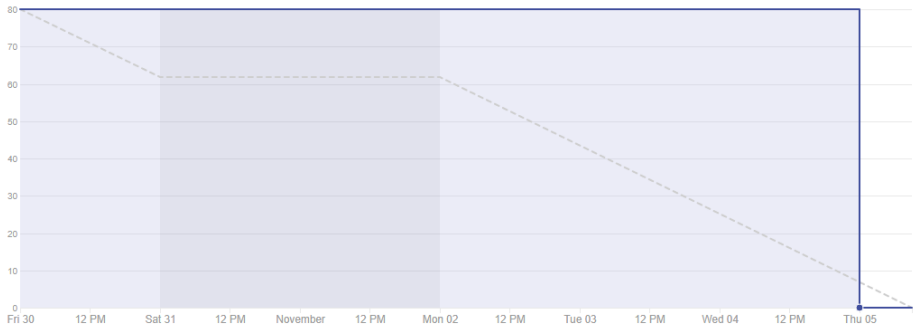


Figura A.4: Gráfico Burndown sprint4.

Sprint 05 - 06/11/2020 - 12/11/2020

Se repasan los cambios propuestos y se incluyen otros nuevos sobre la parte física de la instalación. Se hace el seguimiento de la instalación y se comentan las fotos y el estado de la instalación. Los objetivos de este sprint son todos aquellos que sean necesarios para la adecuación del software básico para comenzar el proyecto como la actualización del sistema operativo o instalación de software adicional. Además se deben valorar opciones para controlar los GPIO.

Vemos en los issues 21 y 19 las configuraciones básicas realizadas a nuestro Raspbian. Para explicar este proceso grabé dos vídeos, titulados Primeros Pasos Raspberry Pi y Actualización de Raspberry Pi y configuración básica, respectivamente.

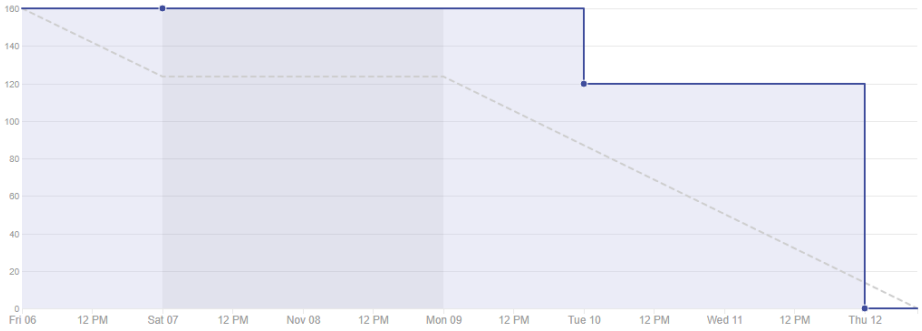


Figura A.5: Gráfico Burndown sprint5.

Sprint 06 - 13/11/2020 - 19/11/2020

Se revisan los puntos anteriores y se compone un tablero de pruebas de forma que se puede controlar el encendido de una bombilla desde nuestra Raspberry Pi. También se graba un vídeo explicativo y se diseñan unos diagramas explicativos.

Podemos ver en el issue 22 que se presenta un diagrama con la lista de los componentes que vamos a utilizar en el tablero de pruebas y un diagrama de como se compondrá el tablero. En este issue también grabé un vídeo para explicar el tablero de pruebas y decidí utilizar un efecto más conservador de cara al final del proyecto.

En el issue 23 he explicado como se controlan los GPIO de nuestra Raspberry Pi desde su distribución Linux [2] desde Bash y desde Python.

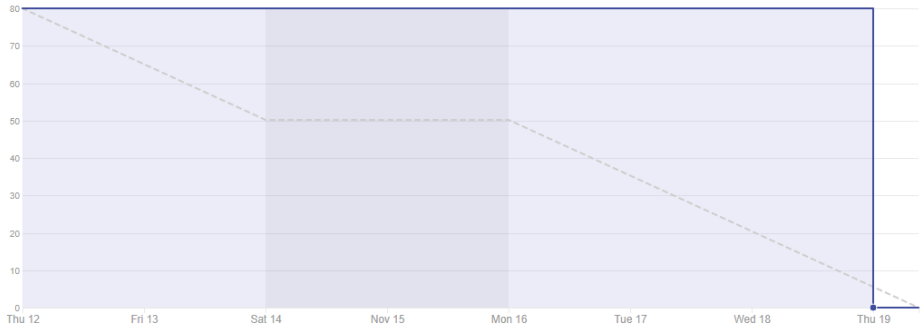


Figura A.6: Gráfico Burndown sprint6.

Sprint 07 - 20/11/2020 - 26/11/2020

Se realiza una investigación sobre como se puede implantar el bot en nuestro proyecto, se realiza el primer código de pruebas y se presentan las primeras pruebas con el bot. Podemos ver en el issue 13 un resumen de las pruebas que estuve realizando con los diferentes teclados de que dispone Telegram.

También, se proponen correcciones en la redacción.

[2] Raspbery pi (Raspbian OS).

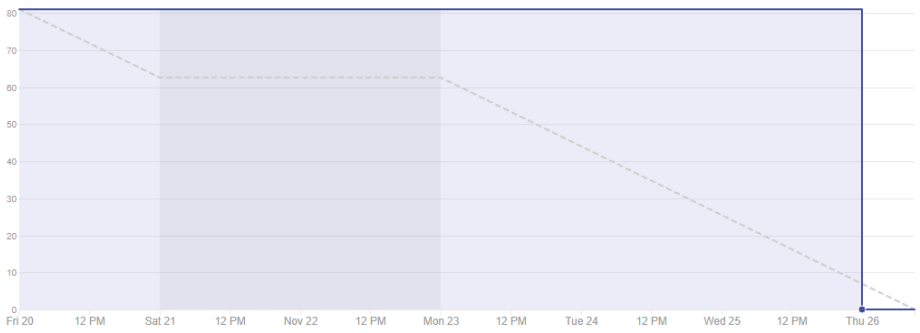


Figura A.7: Gráfico Burndown sprint7.

Sprint 08 - 27/11/2020 - 03/12/2020

En el milestone 8, podemos ver que se incluyen varios issues:

- Creación de scripts Bash para controlar el conjunto del Sistema Domótico inteligente.
- Obtención de los datos de geolocalización y cálculo de temperaturas.
- Generación del nuevo Cron tras integrar los scripts existentes.

De esta manera, se genera la primera automatización completa de la parte automática del proyecto utilizando CRON y lo pasamos del entorno de pruebas a un entorno de producción en fase experimental. En este momento únicamente tenemos control sobre la máquina de forma física o por VNC. Por último, se presentan las primeras funcionalidades del bot, se proponen cambios y mejoras.

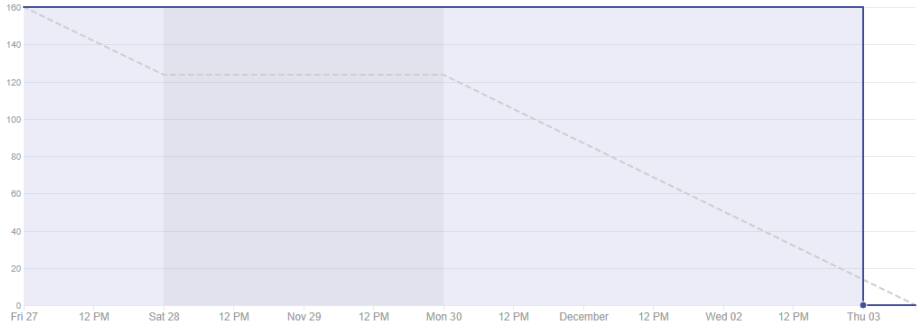


Figura A.8: Gráfico Burndown sprint8.

Sprint 09 - 04/12/2020 - 10/12/2020

En el milestone 9 se proponen revisan las funcionalidades del Bot [11] de Telegram [12] que utilizaremos, se proponen cambios y mejoras en el código preexistente en fase de pruebas.

Finalmente se integra un bot de Telegram completamente funcional desde el que podremos lanzar órdenes a nuestro sistema domótico con los scripts existentes.

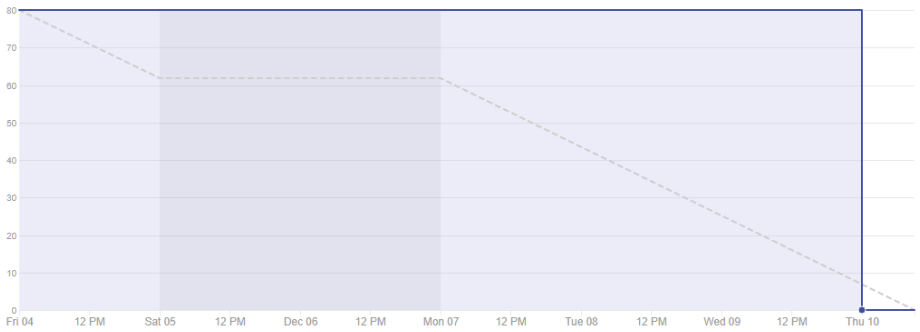


Figura A.9: Gráfico Burndown sprint9.

Sprint 10 - 11/12/2020 - 17/12/2020

En el milestone 10 se ha aprovechado para optimizar las rutas de los archivos a los que se invoca desde el código en ejecución y se han modificado algunas salidas para que tengan un aspecto visual más atractivo, incluyendo tablas, emoticonos desde Unicode [16].

Se comprueban las funcionalidades proponiendo cambios funcionales y de estilos de forma que se pueda interactuar contra el bot y se muestren los mensajes en un formato más atractivo. También se propone unificar las rutas de trabajo de los diferentes scripts.

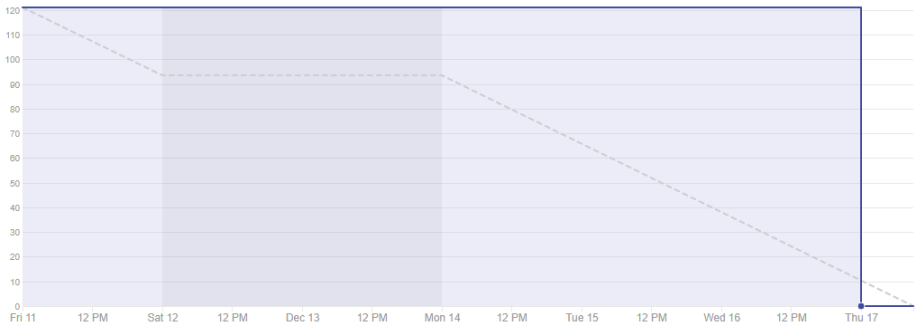


Figura A.10: Gráfico Burndown sprint10.

Sprint 11 - 18/12/2020 - 24/12/2020

Se propone dividir el código por funcionalidades (obtención de información de la web, y por otro lado, modificación de archivos del sistema y se proponen cambios en la redacción a implementar en el sprint 12. Finalmente,

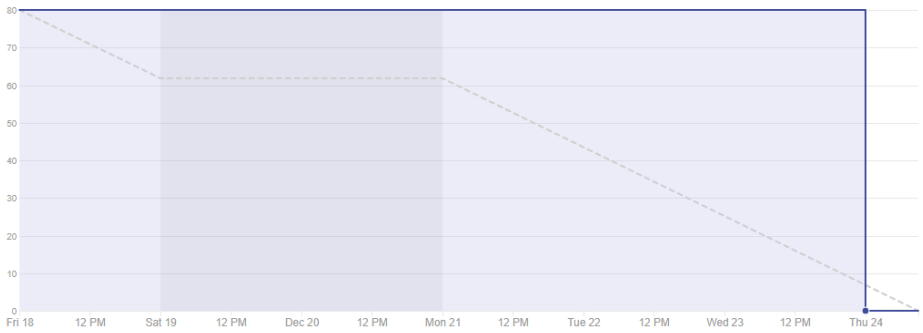


Figura A.11: Gráfico Burndown sprint11.

se ha conseguido generar la división del código en 3 archivos principales:

1. Toma de datos de la web y guardado en json.
2. Lectura de parámetros personalizados.
3. Grabado de CRON conforme a esas preferencias.

De esta manera conseguimos minimizar las comunicaciones con el exterior y poder relanzar el código separado por funcionalidad lógica de forma que podamos relanzarlo de forma aislada en caso de necesitarlo. Un ejemplo puede ser cuando se modifica la hora de subida de las persianas, el módulo

de generación de CRON leería el archivo de parámetros personalizados y generaría la configuración a partir de éstos.

Sprint 12 - 25/12/2020 - 31/01/2021

Se finaliza la redacción de la memoria, antes de la revisión final del texto, y se comienza con los anexos. Además se produce depurado del código y se utiliza SonarCloud [4].

Sprint 13 - 01/12/2020 - 07/01/2021

Redacción de los anexos, corrección de la memoria y limpieza de código.

Sprint 14 - 08/01/2020 - 14/01/2020

Corrección de los anexos y grabado de vídeos.

Sprint 15 - 15/01/2020 - 20/01/2020

Edición de vídeos y entrega del proyecto.

A.3. Estudio de viabilidad

El estudio de viabilidad es inherente a cualquier proyecto y determina si éste puede ser aplicable o no, y también sirve para aclarar algunos puntos importantes para ayudar a la toma de decisiones ejecutivas:

- Señalar las características únicas propias del proyecto en cuestión que permitan poder realizar un estudio de riesgos.
- Enmarcar legalmente el proyecto a realizar y determinar el espacio de mercado encaja.
- Poder ajustar el presupuesto del proyecto a las necesidades existentes.

Viabilidad económica

El primer punto que se tratarán los detalles económicos del proyecto, que son una parte fundamental para saber si el proyecto se puede asumir en un proyecto con carácter comercial desde el ámbito empresarial con carácter profesional.

En esta sección se explica cada uno de los costes asociados al proyecto teniendo en cuenta que se ha intentado minimizar los costes en todos y cada uno de los puntos de la instalación y material.

Coste de personal

Aún habiendo sido desarrollado por una única persona, el apartado de costes de personal es el que contiene un gasto mayor con respecto al resto de apartados económicos.

Se ha estimado un sueldo de 22.000€ brutos anuales en 12 pagas para realizar el cálculo en el intervalo entre el 01 de octubre de 2020 y el 20 de enero de 2021. Si hacemos el cálculo de estos 111 días que ha durado el proyecto, vemos que se corresponde a 3,7 meses de trabajo por lo que el gasto total en personal durante el proyecto asciende a 8.818,33€. El desglose mensual podemos verlo en la tabla A.1 y el coste total en la tabla A.2.

Concepto	**Porcentaje**	**Coste**
Sueldo Bruto Mensual		1833,33 €
Contingencias Comunes	23,60 %	432,67 €
Tipo general	5,50 %	100,83 €
Fogasa	0,20 %	3,67 €
FP	0,70 %	12,83 €
Total Gasto Mensual (Suma)		**2.383,33 €**

Tabla A.1: Coste de personal mensual

Concepto	**Coste**
Número de meses	3.7 meses
Gasto mensual	2.383,33 €
Total Gasto Proyecto	**8.818,33 €**

Tabla A.2: Coste total en personal durante el proyecto

Para realizar el cálculo me he apoyado en la documentación oficial que tiene a disposición del ciudadano la Seguridad Social en su página web [10]. En ella podemos ver que hay que aportar un 23,60 % en concepto

de Contingencias comunes, un 5,50 % en concepto de desempleo de Tipo general, un 0,20 % a FOGASA y un 0,70 % para FP.

Coste hardware

Aunque se ha procurado ahorrar en costes de hardware, y sabiendo que este proyecto se puede realizar íntegramente desde la misma máquina Raspberry Pi que se ha utilizado para desarrollar y que puede se trabajar conectado a la máquina incluiré un equipo portátil de gama corporativa desde el que trabajar cómodamente.

Tras valorar las características de las diferentes Raspberry Pi, y sabiendo la potencia que necesita el proyecto, adquiriría la versión 3B+ por disponer de mayor potencia y mejor conectividad por una inversión razonable aunque para el proyecto 'reciclaré' una RbP 2B que tengo en desuso. Por ello, en el apartado económico figurará la versión 3B a un precio de 34,30€. El estudio preliminar se realizó en el milestone 3 [3] y puede observarse online mientras que el coste final se representa en la tabla A.3

Para calcular la amortización del material en el tiempo transcurrido he determinado un periodo de amortización de 5 años, de los cuales han transcurrido 3,7 meses.

Concepto	Coste	Coste amortizado
Portátil Lenovo Thinkpad T50	1.427,86 €	88,05 €
Raspberry Pi 2B	34,30€	2,12€
Tarjeta SD	14,08 €	0,87 €
Cable UTP5e	14,41 €	0,89 €
Caja Raspberry Pi	15,28 €	0,94 €
Placa Board	3,50 €	0,22 €
Dongle WiFi	6,95 €	0,43 €
Relés	38,35 €	2,36 €
Cables electrónica	6,59 €	0,41 €
Conectores de cables electrónicos	10,70 €	0,66 €
Crimpadora JST	12,00 €	0,74 €
Guía pasacables	2,00 €	0,12 €
Total	**1.586,02 €**	**97,80 €**

Tabla A.3: Coste total en hardware

Coste software

Como el único programa que no es online lo podemos correr sobre Linux, utilizaremos Linux para ahorrar en costes. El único gasto es la licencia del programa Fritzing, que tiene un costo total de 8€ [6].

Costes varios

Únicamente incluiré una tarifa de datos y una de móvil por un precio de 50€/mes, al ser casi 4 meses el precio asciende a 200€. El motivo es que se necesita un número de teléfono móvil para poder hacer pruebas con Telegram aunque lo ideal es utilizar dos números de teléfono para hacer algunas pruebas.

Costes Total

Si sumamos los costes anteriormente descritos, vemos que la suma asciende a **10.612,35€** que han quedado descritos en la tabla A.4.

Concepto	Coste
Personal	8.818,33€
Hardware	1.586,02€
Software	8€
Caja Raspberry Pi	200€
Total	**10.612,35 €**

Tabla A.4: Tabla de costes totales

Beneficios

En un principio el proyecto está pensado para ser utilizado de forma libre pero, en el caso de cambiar de idea se puede monetizar de varias formas. Quizás, la más sencilla sea un pago por licencia con carácter anual pudiendo poner una suscripción de 1€ o 2€ por cada mes de uso en cada instalación de forma que se podría rentabilizar rápidamente como se ha plasmado en la tabla A.5.

Tiempo	1 Instalación	2 Instalaciones	3 Instalaciones
1 año	12€	24€	36€
2 años	22€	45€	50€

Tabla A.5: Cuotas

Viabilidad legal

Tanto en el proyecto como en los anexos se ha hecho alusión a algunas licencias como «Creative Commons [2]» y «GNU [7]». En este apartado se detalla el marco legal del proyecto, pero para poder entenderlo debemos entender que es una licencia:

> *Una licencia de software es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribución) y el licenciatario (usuario consumidor, profesional o empresa) del programa informático, para utilizarlo cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas, es decir, es un conjunto de permisos que un desarrollador le puede otorgar a un usuario en los que tiene la posibilidad de distribuir, usar o modificar el producto bajo una licencia determinada.* [14]

Para poder determinar la licencia más adecuada para nuestro proyecto debemos hacer un estudio de las licencias que se utilizan en todo el proceso ya que hemos utilizado licencias a tener en cuenta.

Frontend

En el proyecto se ha utilizado un bot de Telegram como FrontEnd del sistema domótico. Podemos ver las licencias del sistema que utilizamos en la tabla A.6.

Nombre	Licencia	Descripción
Telegram Code	MIT	Código interno de Telegram
Telegram App [12]	GPLv2 y post	Aplicaciones móviles
Telegram Api [11]	GPLv3 excepto OpenSSL	API pública de Telegram

Tabla A.6: Dependecncias en el FrontEnd.

Backend

En la parte de BackEnd nos hemos servido de otros códigos o «dependencias» que también cuentan con sus propias licencias, como podemos ver en la tabla A.7. Al realizar un pequeño estudio podemos determinar compatibilidad de las licencias de las dependencias utilizadas en nuestro proyecto y que podemos ver en la imagen A.12.

Nombre	Version	Licencia	Descripción
Raspbian	5.4	GPL	Sistema Operativo de Raspberry Pi
Python [9]	3.7	PSF	Lenguaje de programación interpretado
urllib3	1.24.1	MIT	Cliente HTTP
datetime	-	ZPL	Módulo para trabajar con fechas y horas
pandas	1.1.4	BSD	Herramienta de manipulación de datos de alto nivel
matplotlib	3.3.3	PSF based	Biblioteca para la generación de gráficos
telebot	0.0.4	MIT	Biblioteca que encapsula todas las llamadas a la API
time	-	MIT	Biblioteca que ofrece funciones relacionadas con el tiempo
simplejson	3.16.0	GPL	Paquete para trabajar con objetos json [5]
requests	2.21.0	Apache 2.0	Biblioteca HTTP
pycurl	7.43.0.6	LGPL/MIT	Cliente HTTP
telegram	0.0.1	GPLv2 to LGPLv3	Cliente para Telegram
sys	-	MIT	Para acceder a funciones del sistema operativo
math	-	MIT	Para ealizar operaciones matemáticas
os	-	MIT	Para acceder a funciones del sistema operativo
pip	20.2.4	MIT	Administrador de paquetes de Python

Tabla A.7: Dependecncias en el BackEnd.

Tras ordenar las compatibilidades de las licencias de las dependencias de nuestro proyecto vemos que la licencia más compatible es GPL3 [8], y para evistar interferir de algún modo con las licencias de alguna de ellas, nuestro software cuenta con dicha licencia GPL3.

El liberar el código bajo GPL3 [8] resulta beneficioso ya que al ser de libre disposición cualquiera puede beneficiarse de éste, del mismo modo que puede mejorarlo y licenciarlo con una licencia libre compatible y poder aprovechar también dichas mejoras en futuros desarrollos.

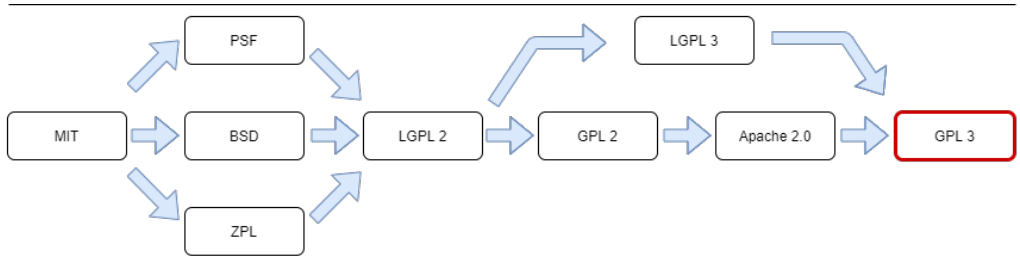


Figura A.12: Compatibilidad entre licencias.

Por ello, para entender que permite la licencia GPL3 [8] he confeccionado un breve resumen. Aunque la tabla A.8 no representa la casuística completa de la licencia si nos permite tener una idea global del funcionamiento de ésta.

Permitido	Limitaciones	Obligaciones
Uso Comercial y privado.	Se prohíbe la concesión de sublicencias.	Se debe incluir el software original.
El código se puede modificar.	Responsabilidad limitada.	Debe incluir los cambios en el software.
El código se puede distribuir.	No dispone de garantía.	El código debe divulgarse bajo una licencia compatible con GPL 3.0.
Se puede otorgar garantía al código con licencia.		Debe incluir la licencia.

Tabla A.8: GPL3

Podemos ampliar la información en la página oficial de GNU [8].

Documentación

Para generar la documentación se han utilizado dos licencias, la referente a LaTeX y a la aplicación Fritzing. Los diagramas electrónicos del proyecto han sigo generados con esta licencia Creative Commons by-sa [1]. La documentación del proyecto tiene la misma cobertura CC by-sa, que permite el uso comercial y distribución tanto de esta obra como derivadas de ésta siempre que se cuente con la misma licencia que regula la obra original. Podemos ver los derechos adquiridos de la obra así como las obligaciones en la página oficial [1].

Nombre	Licencia	Descripción
LaTeX [13]	LPPL	Procesador de textos
Fritzing	CC by-sa	Diagramas electrónicos

Tabla A.9: Dependecncias en la documentación.

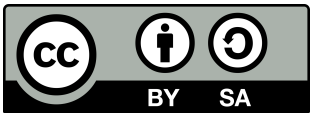
Resumen del licenciamiento del presente proyecto

Para facilitar la búsqueda y comprensión de las licencias con que cuenta el proyecto he generado la tabla resúmen A.10

Recurso	Licencia
Código Fuente	GPL3
Documentación	CC-BY-SA-3.0
Imágenes	CC-BY-SA-3.0
Vídeos	CC-BY-SA-3.0

Tabla A.10: Licencias que protegen el proyecto.

Se pueden ver las licencias en los enlaces correspondientes de GPL3 [8] y CC-BY-SA-3.0 [1].



Apéndice B

Especificación de Requisitos

B.1. Introducción

En este punto se detallan los requisitos con que debe cumplir el proyecto y que determinarán restricciones, funcionalidades y comportamiento de éste.

B.2. Objetivos generales

Los objetivos generales enmarcan la funcionalidad del proyecto, que se detallan a continuación:

- Crear una plataforma que sirva que consiga aumentar confort, comodidad y seguridad dentro del domicilio.
- Generar un Sistema Domótico que nos permita automatizar parámetros de la vivienda así como poder ordenar acciones en tiempo real.
- Crear un Sistema Simulador de Presencia Domiciliaria de funcionamiento autónomo.
- Debe contar con una interfaz de comunicación multiplataforma para interactuar con el software.

B.3. Catálogo de requisitos

En este punto se desglosan los requisitos con que debe cumplir el proyecto, derivando de los anteriormente expuestos.

Requisitos funcionales

- **RF-1 Obtención de datos:** El Sistema debe ser capaz de obtener datos necesarios para su correcto funcionamiento.
  - **RF-1.1 Obtención de ubicación:** Debe ser capaz de determinar su ubicación geográfica.
  - **RF-1.2 Obtención de posición solar:** Debe poder obtener información sobre la posición solar con respecto del planeta según su ubicación geográfica.
  - **RF-1.3 Obtención de temperaturas:** Debe poder obtener información meteorológica de las próximas horas en la ubicación en la que se encuentra el Sistema.
  - **RF-1.4 Datos bajo demanda:** El usuario debe poder solicitar la obtención de los datos bajo demanda.
- **RF-2 Control de periféricos:** El Sistema debe poder controlar los periféricos que tiene conectados.
  - **RF-2.1 Control automático:** El sistema debe poder controlar de forma autónoma los dispositivos conectados.
  - **RF-2.2 Control instantáneo:** El usuario debe poder controlar los diferentes periféricos a placer.
  - **RF-2.3 Control planificado:** El sistema debe poder planificar el control de los periféricos conectados.
  - **RF-2.4 Parametrización:** El usuario debe poder parametrizar la automatización de los periféricos.
- **RF-3 Información:** La información de que dispone el Sistema siempre estará a disposición del usuario.
  - **RF-3.1 Información de sistema:** El usuario podrá obtener información básica sobre el sistema.
  - **RF-3.2. Información de planificación:** El usuario debe poder obtener información sobre la planificación de la automatización de los periféricos.
  - **RF-3.3. Información parametrización:** El usuario debe poder obtener información sobre la parametrización de la automatización de los periféricos.

    - **RF-3.4 Diagramas:** El usuario debe poder obtener un diagrama de temperaturas del día u otros días anteriores que estén archivados en la máquina a modo de registro.

- **RF-4 Control máquina:** Se deben poder ejecutar algunas tareas básicas sobre la máquina que soporta el Sistema.
    - **RF-4.1 Operaciones bajo demanda:** El usuario podrá apagar y reiniciar la máquina siempre que lo desee.
    - **RF-4.2 Operaciones automáticas:** El sistema se reiniciará, al menos, una vez al día.
- **RF-5 Gestión de la máquina:** La máquina que soporta el sistema debe poder ser accesible por parte del usuario.
    - **RF-5.1 Gestión local:** El usuario podrá gestionar el Sistema Operativo de la máquina que soporta el Sistema de forma local.
    - **RF-5.2 Gestión remota:** El usuario podrá gestionar el Sistema Operativo de la máquina que soporta el Sistema de forma remota vía VNC.
- **RF-6 Comunicación entre BackEnd y FrontEnd:** Debe existir un filtro en las comunicaciones.
    - **RF-6.1 Selección de usuarios:** El propio usuario debe poder gestionar los usuarios que pueden interactuar con el Sistema.
    - **RF-6.2 Rechazo de peticiones:** Los usuarios que no estén admitidos en la lista no podrán acceder a la comunicación con el Sistema.
    - **RF-6.3 Interacción multiplataforma:** Debe poder interactuar mediante las aplicaciones móviles existentes de Telegram y desde cualquier navegador popularmente extendido como pueden ser Chrome, Edge o Firefox entre otros.

Requisitos no funcionales

- **RNF-1 Escalabilidad:** Debe poder ampliarse fácilmente.
- **RNF-2 Eficiencia:** Debe minimizar la carga computacional para minimizar también el consumo energético del Sistema, además de permitir que la temperatura exterior incida en menos medida en las condiciones deseadas optimizando el consumo de recursos.

- **RNF-3 Rendimiento:** El sistema debe ser fluido y evitar cargas innecesarias.
- **RNF-4 Usabilidad:** El FrontEnd debe ser lo más usable posible, es decir, fácil de utilizar y aprender e intuitivo, y adaptado a las necesidades que pretende cubrir.
- **RNF-5 Disponibilidad:** El Sistema debe estar siempre en correcto funcionamiento.
- **RNF-6 Durabilidad:** El software debe poder funcionar correctamente durante un tiempo relativamente largo.
- **RNF-7 Capacidad:** Debe poder obtener la información necesaria y actuar conforme a lo que se espera de él.
- **RNF-8 Documentación:** Debe existir la suficiente documentación para poder implementar e interactuar con el Sistema.
- **RNF-9 Operabilidad:** Debe permitir manejar y controlar el Sistema según los requisitos funcionales.
- **RNF-10 Mantenibilidad:** Debe desarrollarse de tal manera que el mantenimiento sea lo más fácil y rápido posible.
- **RNF-11 Seguridad:** Todas las operaciones desde el FrontEnd deben ser seguras y estar cifradas.
- **RNF-12 Legibilidad:** El software debe ser fácilmente legible.
- **RNF-13 Extensibilidad:** El código debe ser facilmente adaptable y reutilizable.
- **RNF-14 Liberación de código:** Debe disponer de un Sistema Operativo GNU y el código debe tener algún tipo de licencia GNU.
- **RNF-15 Respaldo documental:** Toda la instalación debe realizarse conforme a los estándares legales vigentes.

B.4. Especificación de requisitos

La especificación de requisitos contempla los casos de uso contra nuestro código. Además, también podemos ver el diagrama de los casos de uso, aunque lo he dividido en los diagramas «1/2» B.1 y «2/2» B.2

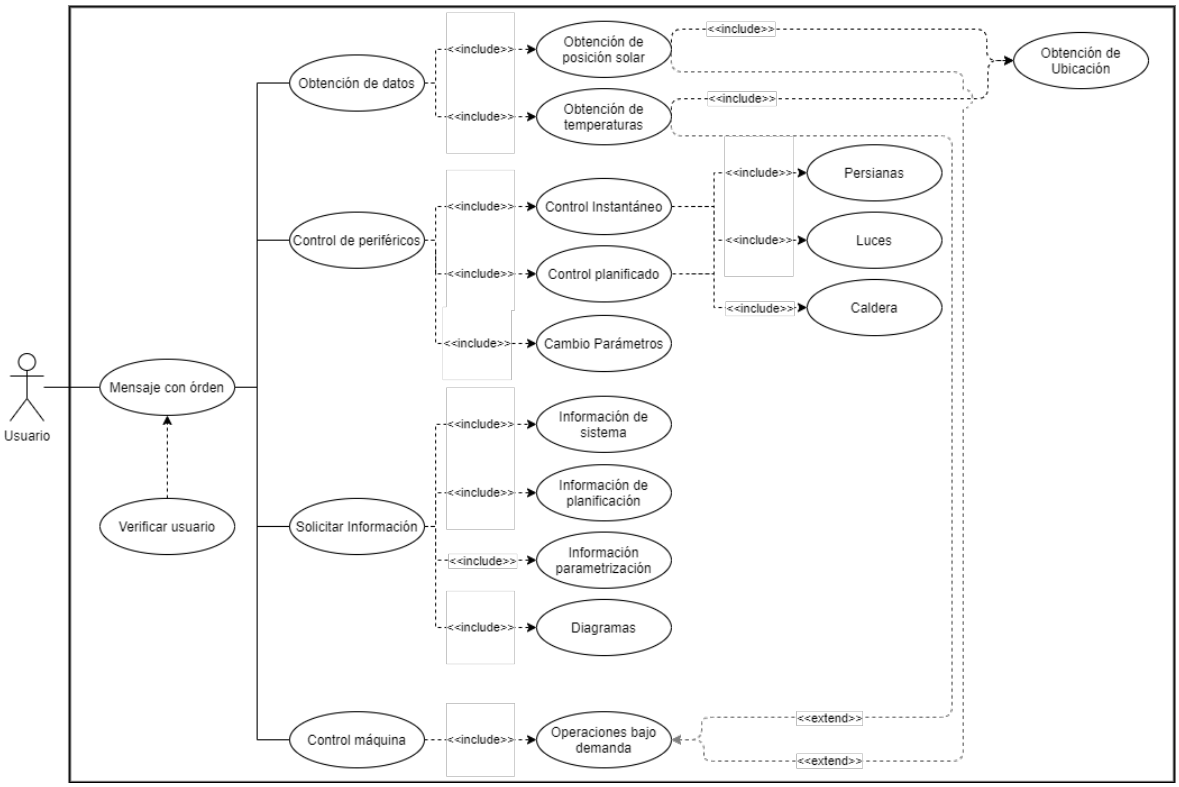


Figura B.1: Casos de uso 1/2.

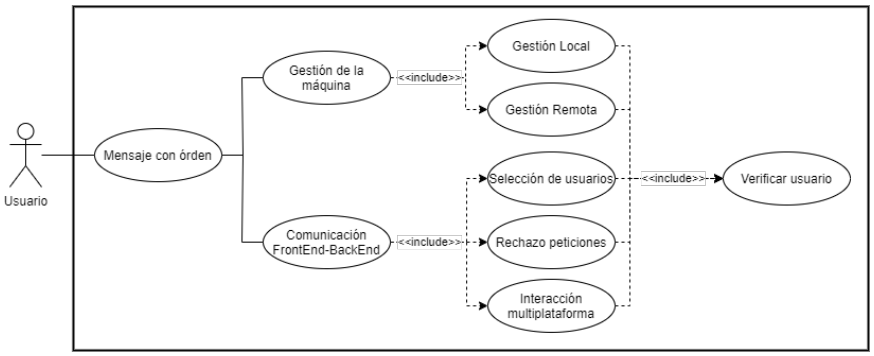


Figura B.2: Casos de uso 2/2.

Actores

Los actores serán cada uno de los usuarios del Sistema que controlará el sistema desde el FrontEnd.

CU-01	**Obtención de posición solar**
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-1, RF-1.1, RF-1.2, RF-1.3, RF-1.4
Descripción	Permite al usuario obtener los datos necesarios para parametrizar el Sistema Domótico Inteligente.
Precondición	▪ Al ser un proceso principalmente automático, únicamente se requiere una línea de datos con acceso a Internet. ▪ En el caso de hacer la petición el usuario, también necesita ser uno de los usuarios autorizados.
Acciones	▪ El usuario solicita la obtención inmediata de los datos necesarios para realizar la próxima programación. ▪ El programa llama a las APIS de geolocalización, astrológicos y meteorológicos.
Postcondición	El bot lanza los scripts de recopilación de datos y almacena los datos.
Excepciones	Si no puede recopilar los datos envía mensaje al usuario.
Importancia	Alta

Tabla B.1: CU-01 - Obtención de datos

CU-02	**Control de periféricos**
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-2, RF-2.1, RF-2.2, RF-2.3, RF-2.4
Descripción	Permite controlar los periféricos
Precondición	▪ Deben existir periféricos y estar configurados en el archivo al efecto. ▪ El usuario debe estar acreditado.
Acciones	▪ El usuario puede ordenar controlar un elemento.
Postcondición	El periférico cambia de estado.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Media

Tabla B.2: CU-02 - Control de periféricos.

CU-03	**Parametrización de periféricos**
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-2, RF-2.4
Descripción	Permite parametrizar el control automático
Precondición	▪ Deben existir periféricos y estar configurados en el archivo al efecto. ▪ El usuario debe estar acreditado. ▪ Deben introducirse los parámetros correctos.
Acciones	El usuario parametriza la automatización de un periférico.
Postcondición	El periférico cambiará de estado.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Alta

Tabla B.3: CU-03 - Parametrización de periféricos

CU-04	**Solicitar Información**
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-3, RF-3.1, RF-3.2, RF-3.3, RF-3.4
Descripción	Permite obtener información del sistema, de automatización, de la parametrización.
Precondición	▪ El usuario debe estar acreditado. ▪ Debe existir la información.
Acciones	Lectura de información.
Postcondición	Se envía información del sistema.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Baja

Tabla B.4: CU-04 Solicitar Información

CU-05	**Solicitar Información de sistema**
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-3, RF-3.1
Descripción	Permite obtener información del sistema
Precondición	▪ El usuario debe estar acreditado. ▪ Debe existir el diagrama.
Acciones	Se envía información del sistema.
Postcondición	Se enviará la información solicitada.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Baja

Tabla B.5: CU-05 Solicitar Información de sistema

CU-06	Solicitar Información de planificación
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-3, RF-3.1
Descripción	Permite obtener información sobre la planificación del funcionamiento de los periféricos.
Precondición	▪ El usuario debe estar acreditado. ▪ Debe existir el diagrama.
Acciones	Lectura de información.
Postcondición	Se envía información del sistema
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Baja

Tabla B.6: CU-06 Solicitar Información de planificación

CU-07	Solicitar Información de parametrización
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-3, RF-3.1
Descripción	Permite obtener información sobre los parámetros personalizados introducidos por el usuario.
Precondición	▪ El usuario debe estar acreditado. ▪ Debe existir el diagrama.
Acciones	Lectura de información.
Postcondición	Se envia información de parametrización.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Baja

Tabla B.7: CU-07 Solicitar Información de parametrización

CU-08	Solicitar Diagramas informativos
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-3, RF-3.1
Descripción	Permite obtener información sobre los parámetros personalizados introducidos por el usuario.
Precondición	▪ El usuario debe estar acreditado. ▪ Debe existir el diagrama.
Acciones	Busca el diagrama elegido.
Postcondición	Envía el diagrama.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Baja

Tabla B.8: CU-08 Solicitar Diagramas informativos

CU-09	Operaciones bajo demanda
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-4, RF-4.1
Descripción	Permite ejecutar apagado y reinicio de la máquina a placer.
Precondición	El usuario debe estar acreditado.
Acciones	Apaga o Reinicia el Sistema.
Postcondición	La máquina ejecuta la órden enviada.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Baja

Tabla B.9: CU-09 Operaciones bajo demanda

CU-10	Gestión local de la máquina
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-5, RF-5.1
Descripción	Permite administrar la máquina que soporta el sistema de forma local
Precondición	El usuario existir en el sistema.
Acciones	Administrar el sistema.
Postcondición	Si conoce las credenciales podrá administrar el sistema.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Alta

Tabla B.10: CU-10 Gestión local de la máquina

CU-11	Gestión remota de la máquina
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-5, RF-5.2
Descripción	Permite administrar la máquina que soporta el sistema de forma remota
Precondición	El usuario debe existir en el sistema, VNC debe estar funcionando y conocer las credenciales.
Acciones	Administrar el sistema.
Postcondición	Si conoce las credenciales podrá administrar el sistema.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Alta

Tabla B.11: CU-11 Gestión remota de la máquina

CU-12	Selección de usuarios
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-6, RF-6.1, RF-6.2
Descripción	Se deben poder escoger los usuarios que podrán interactuar con el bot, el resto quedan descartados.
Precondición	Los usuarios seleccionados deben existir en el archivo creado para ello.
Acciones	Permite interactuar con el bot o rechaza las peticiones.
Postcondición	Si está acreditado podrá interactuar con el sistema.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Alta

Tabla B.12: CU-12 Selección de usuarios

CU-13	Interacción Multiplaforma
Versión	1.0
Actor	Usuario
Requisitos asociados	RF-6, RF-6.1, RF-6.2, RF-6.3
Descripción	Debe poder interactuar desde un usuario existente y desde cualquier plataforma.
Precondición	El usuario debe existir y estar acreditado.
Acciones	Permite interactuar con el bot o rechaza las peticiones.
Postcondición	Podrá interactuar con el sistema.
Excepciones	Si no puede realizar la acción envía mensaje al usuario.
Importancia	Alta

Tabla B.13: CU-13 Interacción Multiplaforma

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

Hay que conocer las diferentes partes del proyecto y tener en cuenta que son completamente diferentes aunque en este proyecto se han integrado para poder hacer uso del sistema domótico. Podemos ver una primera distribución general en la imagen D.1. En ella podemos diferenciar tres partes principales:

Tuve un problema generalizado en el código puesto que obtenía las rutas a los archivos de forma unitaria desde el archivo en ejecución. Este es el código utilizado para obtener la ruta en Python:

```
0 {
1     #!/usr/bin/env python3
2     #Calculamos ruta
3     ruta=os.getcwd().split('/')
4 }
```

Éste es el código utilizado para obtener la ruta en Bash:

```
0 {
1     #!/bin/bash
2     path=$(pwd)
3 }
```

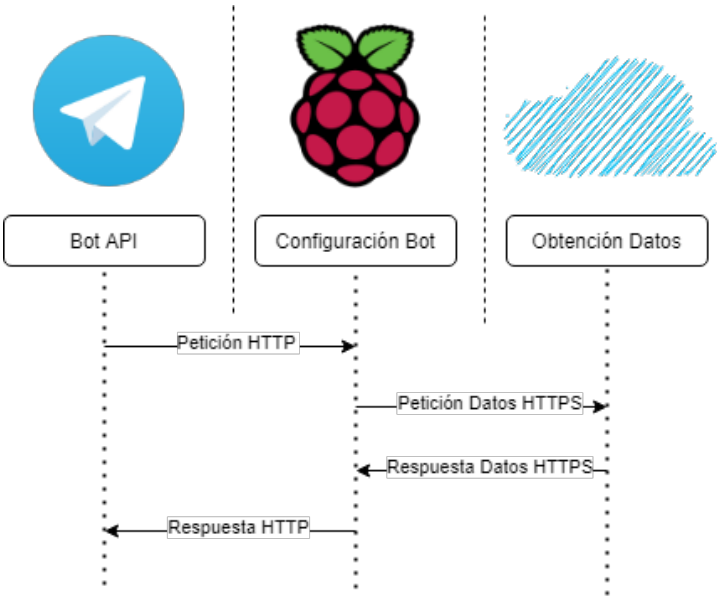


Figura D.1: Partes del proyecto.

- La parte de toma de datos.
- La parte de mensajería.
- La parte física de nuestra instalación domótica, incluyendo la Raspberrry Pi.

Por ello, el código se ha estructurado en tres partes:

- La carpeta **auto**, que permite obtener la información de las APIs seleccionadas de Internet.
- La carpeta **bot**, donde almacenamos la configuración de nuestro bot.
- La carpeta **control**, donde almacenamos los scripts de control de la instalación domótica.

En todas ellas, los scripts, tanto de Bash como de Python llevan la cabecera específica para el tipo de código contenido en el archivo:

```
0 {
1     #!/usr/bin/env python3
2 }
```

Éste es el código utilizado para obtener la ruta en Bash:

```
{
    #!/bin/bash
}
```

Para los archivos Python, la sentencia «!/usr/bin/env python3» y para los archivos Bash, la sentencia «!/bin/bash». He introducido esta cabecera porque he tenido problemas al lanzar las instrucciones desde otro script.

Carpeta auto

Al comienzo del proyecto esperé encontrar alguna web desde la que poder recoger la información haciendo webscraping pero tras valorarlo detenidamente en las tutorías prefería a buscar una API (ver concepto **??**). Uno de los motivos es que una web es más susceptible de sufrir cambios que una API que se ha predispuesto para una función. Finalmente, dispuse el proyecto para poder funcionar con dos APIs públicas y gratuitas para dotar a nuestro sistema domótico de autonomía. Las siguientes fases de la parte de toma de datos, se albergan en la carpeta auto ya que es un proceso automatizado que permite al sistema obtener los datos de forma desatendida y continua. El proceso de automatización comprende varias fases:

1. La fase de recopilación de datos.
2. La fase de lectura local de parámetros.
3. El cocinado de los datos.
4. El grabado de datos en el sistema local.

La fase de recopilación de datos llama a la API de geolocalización y nos devuelve, entre otros, los parámetros de nuestra ubicación en formato de latitud y longitud. Posteriormente, estos datos de ubicación son necesarios para incorporarlos en la consulta de la siguiente API **??** y obtener la información de la salida y puesta del sol así como la información de las temperaturas del día siguiente en dicha ubicación. Con esta recopilación de datos tenemos la base para poder determinar el comportamiento autónomo que necesitamos por lo que almacenaremos esta información en un archivo de datos.

El siguiente paso es leer las preferencias del usuario, que se almacenan en el archivo de condicionantes. Y, una vez tengamos estos datos, podemos pasar a generar el Cron para controlar de forma automática toda la instalación.

Estas fases se han confeccionado al final del proyecto puesto que al principio se desarrolló de forma unitaria pero surgió el problema de que, cada vez que queríamos hacer una modificación en el sistema domótico de cualquier índole, teníamos que hacer una petición nueva a las APIs y generar de nuevo un archivo Cron con los parámetros del usuario «al vuelo» para conseguir regenerar la automatización con los parámetros nuevos. Este sistema funcionaba correctamente pero era poco eficiente puesto que que se multiplican las operaciones en función de las veces que se quiera modificar algún parámetro. Únicamente he permitido realizar dos salidas .ªl vuelo"tras obtener los datos, estos son: el grabado de los datos recién obtenidos y la gráfica de las temperaturas conforme a dichos datos.

He tenido algunos problemas al realizar varias peticiones a las APIS ya que, por falta de permisos, no podía sobreescribir los archivos ni las imágenes. Para subsanarlo, incluí una restricción para modificar los permisos de los archivos de forma que únicamente pueden acceder el propietario y el grupo.

Carpeta control

La carpeta control alberga aquellos scripts bash que controlan los elementos domóticos. Este punto únicamente me dio problemas a la hora de implantarlo desde el bot, que se subsanó con la librería os.

Carpeta bot

La carpeta bot contiene varios archivos, desde el archivo de control del bot hasta otros que contienen la mayoría de funcionalidades de éste. En primera instancia, se dispuso todo el código dentro del archivo de control del bot, pero por legibilidad y facilidad de mantenimiento se extrajeron las funciones. En este punto, lo que más problemas me ha dado han sido los teclados. Como comportamiento de los teclados es diferente entre ellos tendría que dividir las opciones del bot entre dos tipos de teclados diferentes. Finalmente, opté por no incluir estos teclados ya que no incluye funcionalidad alguna y disminuirían la legibilidad y facilidad de uso del bot. Además, para aumentar la usabilidad del bot, se ha dispuesto un menú que podemos ver al introducir el carácter /.ªdemás de la posibilidad de utilizar la primera letra del comando siempre que no tengamos que acompañarlo de parámetros.

Configuración del bot como servicio

El bot se puede lanzar como si fuera un script más pero en este caso es necesario incorporarlo como servicio. Esta decisión tiene beneficios como la

posibilidad de que se inicie con el sistema o que podamos lanzarlo o pararlo desde cualquier ubicación del SO sin tener que recordar la ubicación con la sentencia al efecto de «**sudo service bot start/stop**». Pero, una vez que el demonio está corriendo, tenemos que tener en cuenta de que ya está levantada la instancia del bot, por lo que tendremos que detenerla a la hora de hacer algún tipo de modificación así como recordar que debemos levantarlo una vez terminemos.

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] Creative Commons. *Creative Commons by-sa 3.0*. [Página Oficial de la licencia].

[2] Creative Commons. Norma creative commons. [Página web Oficial, Noviembre de 2020].

[3] David. Presupuesto preliminar en milestone 3.

[4] David. Sonarcloud.

[5] ECMA. Javascript object notation. [Página web Oficial, Noviembre de 2020].

[6] Fritzing. Fritzing. [Página Oficial].

[7] GNU. *Free Software Foundation, Inc.* [Página web Oficial, Noviembre de 2020].

[8] GNU. *Licencia GPL3*. [Página Oficial de la licencia].

[9] Python. Python software foundation. [Página web Oficial].

[10] SS. *Información General sobre cotización*, 2020. [Información General, Octubre de 2020].

[11] Telegram. Telegramapi. [Página Oficial].

[12] Telegram. Telegramapp. [Página Oficial].

[13] Wikipedia. Latex — wikipedia, la enciclopedia libre. [Página web, Noviembre de 2020].

[14] Wikipedia. Licencias software.

[15] Wikipedia. *Metodología Scrum*. [Página web, Noviembre de 2020].

[16] Wikipedia. Unicode-wikipedia.