



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Ingeniería Informática

Trabajo Fin de Grado

Servidor Doméstico sobre Raspberry Pi Automatizado y Configurable

Autor: Rodrigo Gutiérrez de los Reyes

Tutor: Víctor Robles Forcada

Madrid, Junio - 2022

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Servidor Doméstico sobre Raspberry Pi Automatizado y Configurable

Junio - 2022

Autor: Rodrigo Gutiérrez de los Reyes

Tutor: Víctor Robles Forcada

Arquitectura y Tecnología de Sistemas Informáticos

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

Todos los **trabajos desarrollados** durante el Grado en Ingeniería Informática, que implican el uso de un servidor, **se ejecutan en *localhost***.

Nunca supimos cómo hacer nuestro trabajo **accesible desde internet**: característica imprescindible a día de hoy para cualquier proyecto de desarrollo en el campo de la informática.

Aunque existen empresas que ofrecen la posibilidad de **pagar por el acceso a un servidor**, esto suelen ser caros para una persona que está aprendiendo. Además, es posible que no logre exprimirlo, o incluso que sea demasiado limitado. En definitiva: **no suele ser rentable**.

Este proyecto consiste la **creación de un servidor doméstico replicable a través de un script final**, obteniendo alojamiento web y acceso a base de datos a través de una API REST. Además, el servidor se hace accesible desde internet, pudiendo **eliminar la limitación del *localhost*** comentada al inicio.

Este servidor está creado para ejecutar de manera óptima sobre una **Raspberry Pi**, manteniendo un **consumo eléctrico mínimo** y, por ello, aumentando los beneficios al enfrentarlo con el pago de un servidor de terceros.

Además, el proyecto se podría **ampliar con los servicios deseados**, así como con la adquisición de múltiples equipos (para lo cual se necesitaría un mayor presupuesto de compra, desarrollo y mantenimiento), creando un **clúster** que dotase al servidor de un mayor rendimiento ante tareas más exigentes.

Por último, el trabajo se centra en ofrecer una característica muy útil e importante para un desarrollador web: la creación de un **acceso a un proyecto en desarrollo** (o no) desde internet, de manera que se pueda visualizar desde distintos tipos de dispositivos físicos.

Esto permite **comprobar el *responsive web design*** de la aplicación, es decir, cómo se adapta la aplicación a distintas dimensiones y tipos de pantallas: monitor estándar, monitor panorámico, portátil, tablet, smartphone, televisión, etc. Una característica ampliamente útil dada la enorme variedad de dispositivos electrónicos que podemos encontrar en la actualidad.

Abstract

Each **developed assignment** during the Computer Engineering Degree, which takes a server into account, **runs on *localhost***.

We never knew how to make our work **accessible from the internet**: an essential feature today for any development project in the field of computing.

Although there are companies that offer the possibility of **paying for an access to a server**, this is usually expensive for a person who is learning. Besides, you may not manage to squeeze it, or even that could be too limited. In short: **it is usually not profitable**.

This project consists of **creating a replicable home server through a final script**, obtaining web hosting and database access through a REST API. Furthermore, the server becomes accessible from the internet, being able to **eliminate the *localhost* limitation** mentioned at the beginning.

This server is created to run optimally on a **Raspberry Pi**, having a minimum **electrical consumption** and, therefore, increasing the benefits when facing it with the payment of a third-party server.

In addition, the project could be **expanded with the desired services**, as well as with the acquisition of multiple pieces of equipment (for which a larger purchase, development and maintenance budget would be needed), creating a **cluster** that would provide the server with greater performance when faced with tasks More demanding.

Finally, the work focuses on offering a very useful and important feature for a web developer: the creation of an **access to a project under development** (or not) from the internet, so that it can be viewed from different types of physical devices. .

This allows **checking *responsive web design***, that is, how the application adapts to different dimensions and types of screens: standard monitor, panoramic monitor, laptop, tablet, smartphone, television, etc. A widely useful feature given the enormous variety of electronic devices that we can find nowadays.

Tabla de contenidos

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Costes	2
1.3.1. Desarrollo	3
1.3.2. Adquisición	3
1.3.3. Mantenimiento	3
1.4. Estructura de la memoria	4
2. Planificación	5
2.1. General	5
2.2. Desarrollo	7
2.3. Memoria	8
3. Alcance	9
3.1. Aprendizaje	9
3.2. Servidor	9
3.3. Entorno de desarrollo	10
3.4. Alojamiento web	10
3.5. NAS	11
3.6. Cliente RDP	11
4. Estado del Arte	13
4.1. Actualidad	13
4.2. Novedades	13
4.3. Aclaraciones	14
5. Tecnologías	15
5.1. Hardware	15
5.1.1. Alternativas	15
5.1.2. Elección	15
5.2. Sistema Operativo	16
5.2.1. Alternativas	16
5.2.2. Elección	16
5.3. Conexión	17
5.3.1. Alternativas	17

5.3.2. Elección	17
5.4. Scripting	18
5.4.1. Alternativas	18
5.4.2. Elección	18
5.5. Servidor Web	18
5.5.1. Alternativas	18
5.5.2. Elección	18
5.6. Base de Datos	19
5.6.1. Alternativas	19
5.6.2. Elección	19
5.7. Backend	20
5.8. Frontend	20
6. Requisitos	21
6.1. Recopilación	21
6.1.1. Servidor doméstico	21
6.1.2. Automatización	21
6.1.3. Configurabilidad	21
6.1.4. Securización	21
6.1.5. Monitorización	21
6.1.6. Acceso remoto	21
6.1.7. Aplicación web	22
6.2. Análisis	22
7. Desarrollo del servidor doméstico	23
7.1. Hardware	23
7.2. Sistema operativo	24
7.3. Configuración	24
7.4. Securización	24
7.5. Servicios	25
7.6. Conexión SSH	25
7.7. Acceso desde internet	26
7.7.1. DDNS	27
7.7.2. Puertos	27
7.8. Monitorización	28
7.9. Aplicación Web	29
7.9.1. Backend	30
7.9.2. Frontend	31
7.9.3. Protocolo	32
7.9.4. Despliegue	33
7.10. Scripts de automatización	33
7.11. Documentación	34
7.12. Repositorio	34
8. Evaluación	37
8.1. Usabilidad	37
8.2. Consumo	37
8.3. Alternativas	38

TABLA DE CONTENIDOS

8.4. Rentabilidad	38
9. Conclusiones	39
9.1. Principal	39
9.2. Secundarias	39
9.3. Trabajo futuro	40
10.Impacto	41
10.1. Personal	41
10.2. Empresarial	41
10.3. Económico	42
10.4. Objetivos de Desarrollo Sostenible	42
Bibliografía	43
Anexos	49

Índice de figuras

1.1. Raspberry Pi 4 Model B 8GB	3
1.2. Meross MSS310	3
1.3. Raspberry Pi 3 Model B 1GB	3
2.1. Diagrama de Gantt: Planificación General	6
2.2. Diagrama de Gantt: Planificación de Desarrollo	7
2.3. Diagrama de Gantt: Planificación de Memoria	8
3.1. Servidores	9
3.2. Ciclo de desarrollo software	10
3.3. Servidor NAS	11
3.4. Protocolo de escritorio remoto	11
4.1. Open Media Vault	14
5.1. Raspberry Pi 4 Model B 8GB	16
5.2. Raspberry Pi OS Lite	16
5.3. Conexión Ethernet	17
5.4. Conexión Wifi	17
5.5. Conexión SSH	17
5.6. Open SSH	17
5.7. Bash Scripting	18
5.8. Nginx	19
5.9. MongoDB	19
5.10. MERN <i>stack</i>	20
5.11. Node JS	20
5.12. Express JS	20
5.13. React JS	20
7.1. Raspberry Pi 4 Model B 8GB	23
7.2. Raspberry Pi 3 Model B 1GB	23
7.3. Router	27
7.4. Menú del router	27
7.5. Formulario DDNS en el router	28
7.6. DDNS añadido al router	28
7.7. Configuración de puertos	28
7.8. Banner de inicio	28

7.9. Fragmento de respuesta <i>records</i>	30
7.10. Respuesta <i>records</i> 20/05/2022	30
7.11. Gráfico en la aplicación web de consumo diario de la Raspberry Pi	31
7.12. Gráfico en la aplicación web del precio medio diario del kWh	31
7.13. Gráfico en la aplicación web del coste diario de la Raspberry Pi	32
7.14. Tabla en la aplicación web de registros de la Raspberry Pi	32
7.15. Formulario en la aplicación web de registros de la Raspberry Pi	32
7.16. Let's Encrypt	33
7.17. Ejecución de script (1)	35
7.18. Ejecución de script (2)	35
7.19. Documentación del repositorio	35

Capítulo 1

Introducción

Este capítulo expone por qué surge el planteamiento del proyecto, cuál es la importancia de su desarrollo, y qué utilidad ofrece a una persona del ámbito de las Tecnologías de la Información y la Comunicación.

1.1. Motivación

La propuesta de este proyecto surge inicialmente por el interés de aprender cómo funciona el desarrollo de un **proyecto en el mundo laboral**, tratando de aunar gran parte del conocimiento extraído de los estudios en los últimos años.

Además, el origen del proyecto está altamente impulsado por la curiosidad y deseo de poder **acceder a una aplicación web en fase de desarrollo desde distintos dispositivos**. De este modo, sería posible comprobar el diseño web adaptativo de manera eficaz, así como poder compartir el estado del proyecto a un cliente o empresa de manera rápida y sencilla. Con este proceso, lograríamos obtener un **servidor de pruebas**, o incluso un servidor capaz alojar cualquier **servicio en producción** en la red.

Estas serían algunas de las principales utilidades ofrecidas a las personas del ámbito de las Tecnologías de la Información y la Comunicación. Pero, por otro lado, hay también una situación con alta relevancia que provoca la propuesta de este trabajo:

Durante el grado universitario, se realizan proyectos de desarrollo en distintas asignaturas que, en la mayoría de los casos, quedan bastante **lejos de una situación realista**. Se aprende acerca de Redes de Computadores, Sistemas Operativos, Administración de Sistemas Informáticos, Sistemas Orientados a Servicios, Sistemas Distribuidos, etc. pero, si el alumno no posee el interés necesario para relacionar todos los conocimientos y profundizar en ellos, permanecerá en la **superficie del aprendizaje**.

Como consecuencia de esta situación, la mayoría de los **conocimientos adquiridos** durante el grado universitario, que conllevaron años de trabajo y esfuerzo, se verán destinados a desvanecerse con el paso del tiempo, resultando **poco útiles** para el desarrollo profesional de la persona.

En un mundo globalmente conectado, los futuros informáticos reciben una formación universitaria guiada por el **conocimiento teórico** que, por lo general, está **atada al lo-**

calhost, impidiendo al alumno observar más allá de su propia máquina y **conocer el potencial y alcance real de los conocimientos obtenidos**. De este modo, resulta más complicado obtener una visión global de la informática que permita relacionar los conocimientos de cada uno de los ámbitos estudiados.

El desarrollo de este proyecto, es importante para el **aprendizaje** de quien lo lleva a cabo, pero a su vez, generará distintas **herramientas y facilidades** para cualquier persona que quiera aprender, o no, creando su propio **servidor doméstico**.

Por último, cabe destacar que la situación expuesta surge a través **sensaciones personales**, contrastadas junto a las de algunos compañeros con los que trabajé durante los estudios, y puede no representar la totalidad de casos de los estudiantes.

1.2. Objetivos

El objetivo principal del proyecto, es crear un **servidor doméstico accesible desde internet sobre el hardware de una Raspberry Pi**, ofreciendo las características comentadas al inicio del capítulo junto a ciertos requisitos que garanticen un alto nivel de calidad en lo referente al **desarrollo y aprendizaje**.

A partir de este punto, encontramos una **infinidad de posibilidades** y proyectos que se pueden realizar teniendo en cuenta el hardware utilizado. No solo en lo referente a los servicios ofrecidos por un servidor, sino de cualquier pieza de computación informática [1].

En el planteamiento del proyecto, se trató de encontrar un caso al que se pudiera **aplicar de manera práctica** cualquiera de los **conocimientos** obtenidos a lo **largo del grado**, y se escogió desarrollar las características con mayor interés personal.

De este modo, el desarrollo del proyecto busca cubrir los siguientes **objetivos**:

- **Objetivos principales:** esenciales para cubrir la finalidad principal del proyecto.
 - Creación de servidor doméstico accesible desde internet.
 - Creación de servicios de alojamiento web y base de datos.
 - Máxima securización del servidor y del acceso a sus servicios.
 - Implementación de interfaz de línea de comandos para configurar servidor.
 - Automatización de la instalación del proyecto y servicios principales.
- **Objetivos secundarios:** alternativas de ampliación del proyecto.
 - Creación de acceso al entorno de desarrollo desde dispositivos externos a la red.
 - Monitorización y análisis de consumo del servidor ejecutando servicios.
 - Creación de equipo con conexión directa a los escritorios virtuales de la UPM.

1.3. Costes

En las próximas secciones, se exponen los distintos costes que acarrea el desarrollo proyecto, desde el propio desarrollo, a la compra de las distintas herramientas de *hardware* y *software*.

Introducción

1.3.1. Desarrollo

El coste de desarrollo del proyecto, en lo referente a tareas de implementación, se establece en aproximadamente **28 jornadas de trabajo**, es decir, un total aproximado de **224 horas**.

1.3.2. Adquisición

El coste de adquisición del *hardware* previsto se establece alrededor de los **105€**:

90€ Placa de microordenador.

15€ Medidor de consumo eléctrico.



Figura 1.1: Raspberry Pi 4 Model B 8GB



Figura 1.2: Meross MSS310

Pero, finalmente el desarrollo se realizó con un hardware diferente: **Raspberry Pi 3 Model B 1GB**, con un precio de **40€**. Este es un hardware con una menor capacidad de cómputo, pero igualmente útil para realizar el trabajo definido. La causa de esta situación se comentará más a fondo en el capítulo 7, donde se desgana el desarrollo del proyecto.



Figura 1.3: Raspberry Pi 3 Model B 1GB

En cuanto al *software* utilizado, es de **uso libre** en su totalidad, por lo que no fue necesario incluir este apartado en los costes.

1.3.3. Mantenimiento

El coste de mantenimiento del proyecto, en lo que al **consumo eléctrico** se refiere, y dada la situación de costes en mayo de 2022, se establece por debajo de 1€ de manera mensual, más concretamente 0.7801761€.

Este cálculo se ha realizado con datos obtenidos de **mediciones a lo largo de una semana**, que recopiló un consumo diario cercano a 0.095kWh, junto a la media diaria del precio de la luz en el mes de abril, de 0,273746€/kWh.

Estos resultados se pondrán en contexto más detenidamente en la sección sección 8.2 del capítulo 8, referida a la evaluación del consumo, y en el 8.3 comparándolo con otras alternativas disponibles.

1.4. Estructura de la memoria

La presente memoria está estructurada en capítulos. Cada uno de ellos, posee un párrafo introductorio en el que se explica brevemente de qué trata el capítulo. Con ello, se obtiene una contextualización previa a la lectura completa.

Capítulo 2

Planificación

Este capítulo expone cómo se planificó el proyecto para realizarlo a lo largo de 17 semanas, desde el Plan de Trabajo hasta el propio desarrollo junto a la Memoria.

Esta planificación, se mostrará a través de diferentes diagramas de Gantt. Primero, se mostrará un diagrama general del conjunto de tareas al completo, y posteriormente se desgranarán las secciones de desarrollo y documentación a través de este documento.

2.1. General

Inicialmente, se estimó que las tareas establecidas conllevarían el siguiente **tiempo de dedicación** a lo largo del semestre:

<i>16h</i> Estado del arte	<i>160h</i> Implementación	<i>40h</i> Memoria
<i>16h</i> Análisis	<i>64h</i> Pruebas	<i>12h</i> Presentación
<i>24h</i> Diseño	<i>12h</i> Tareas coordinación	<i>4h</i> Defensa

De este modo, se planificó que la **distribución de tareas** se adaptarían conforme al siguiente diagrama de Gantt:

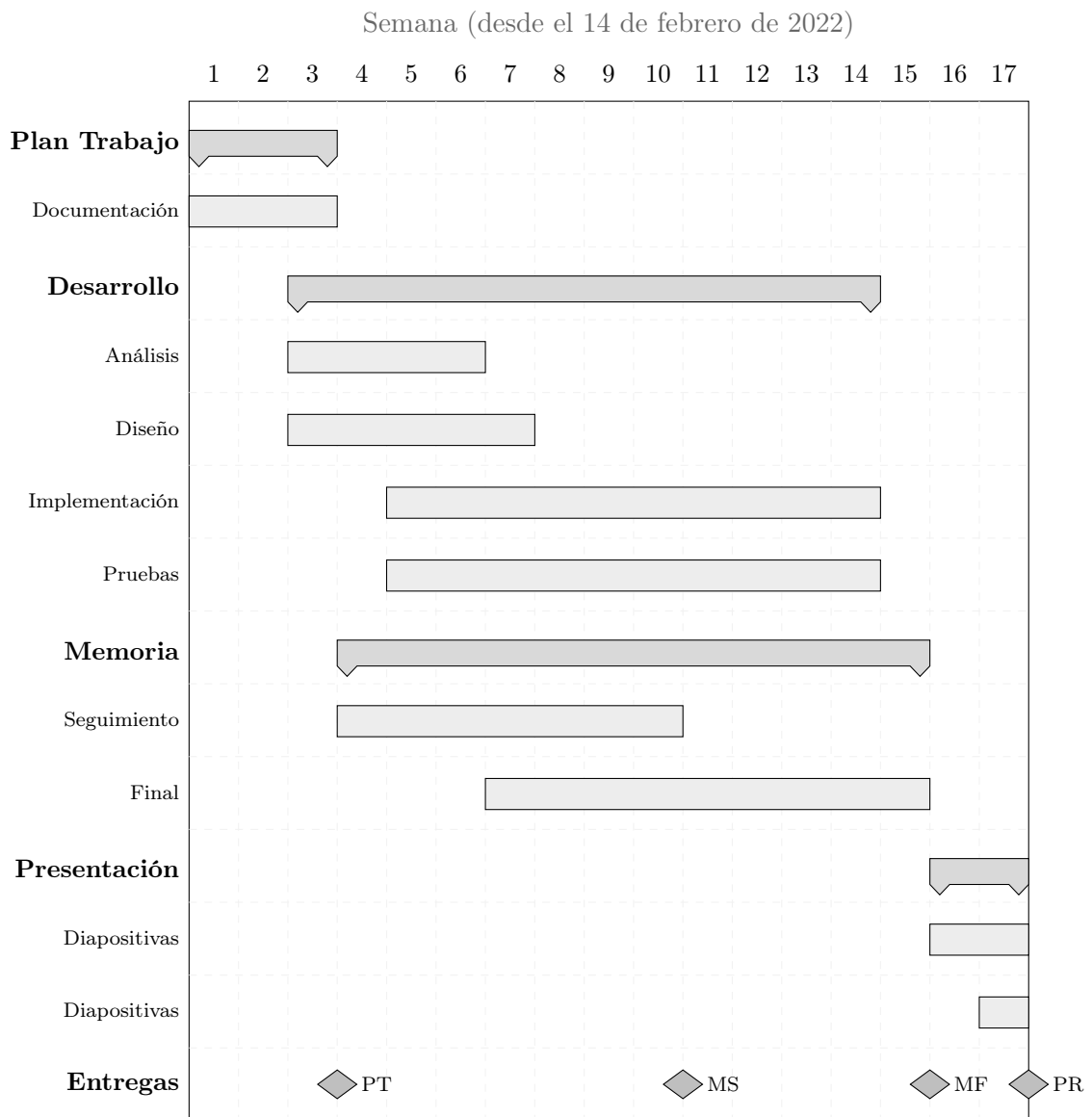


Figura 2.1: Diagrama de Gantt: Planificación General

2.2. Desarrollo

Por otro lado, realizando un desglose de las secciones de desarrollo, la planificación realizada fue la siguiente:

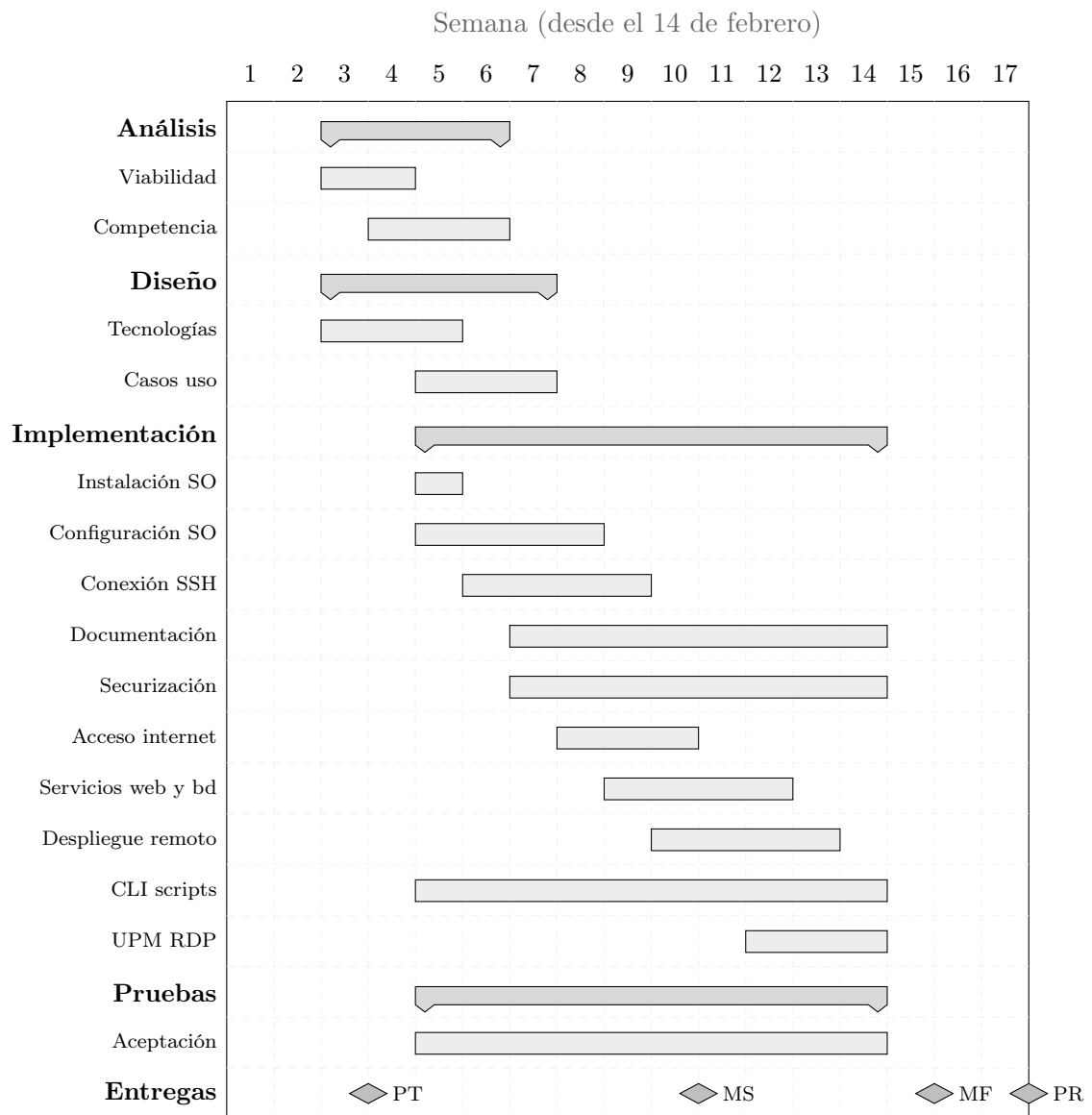


Figura 2.2: Diagrama de Gantt: Planificación de Desarrollo

2.3. Memoria

Finalmente, desglosando de las secciones de memoria, la planificación realizada se representa con el siguiente diagrama:

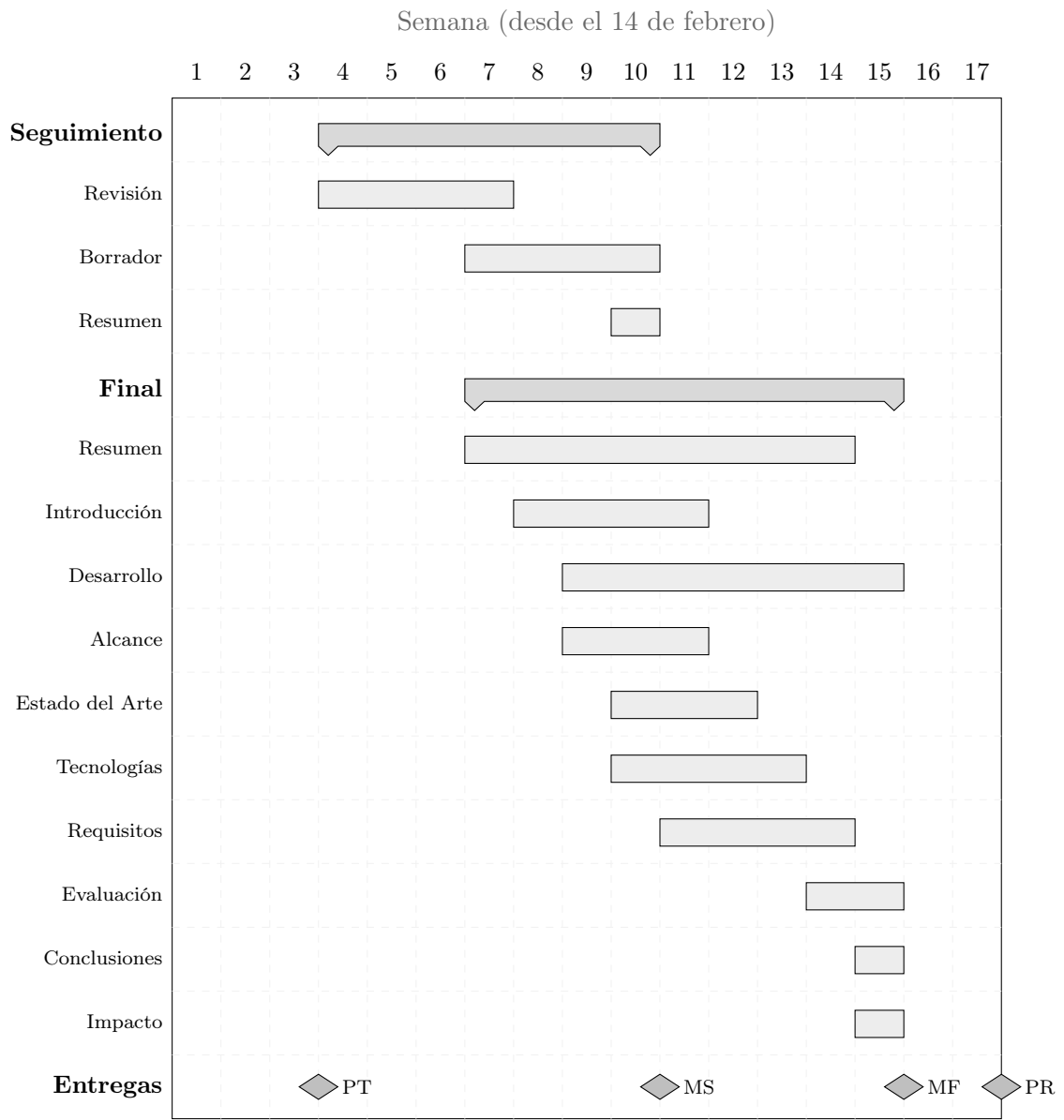


Figura 2.3: Diagrama de Gantt: Planificación de Memoria

Capítulo 3

Alcance

Este capítulo expone, de forma previa al desarrollo del trabajo, qué se espera obtener como resultado del mismo. Posteriormente, se verá complementado por el capítulo 10, correspondiente al análisis del impacto, donde se analiza qué consecuencias o beneficios puede tener el proyecto una vez finalizado.

3.1. Aprendizaje

Como se comenta en el capítulo 1, uno de los principales motivos del desarrollo del proyecto es el **aprendizaje personal**, pero también se espera obtener un **producto útil** para la mayoría de desarrolladores: un servidor doméstico.

El aprendizaje que se espera obtener, parte de la necesidad de utilizar numerosos y variados **conocimientos** que, en mayor o menor medida, se han ido obteniendo **a lo largo de la realización del grado**. Pero, también se han obtenido conocimientos muy útiles en el mundo laboral y que **no se impartieron en ninguna asignatura**.

Como consecuencia, se logrará crear una cierta **conexión entre dichos conocimientos**, permitiendo **profundizar** en ellos y **reforzarlos** a través de supuestos prácticos aplicables al desarrollo personal profesional.

3.2. Servidor



Figura 3.1: Servidores

Una vez finalizado el proyecto, como mínimo, se conseguirá obtener un **servidor doméstico de bajo consumo**, útil para una amplia gama de proyectos.

La utilidad de este servidor está limitada por la imaginación del usuario, pero, como se verá en los próximos apartados y a lo largo del documento, se ofrecerán ciertas características y servicios a modo de **demostración**. Con ello, se comprueba el correcto funcionamiento del producto y se realiza un análisis del impacto del mismo.

Pero esto no acaba aquí, ya que, a este producto, se le podrán **añadir todos los servicios y funcionalidades que se deseen**, ofreciendo inicialmente algunos de los más básicos y útiles.

3.3. Entorno de desarrollo

A la finalización del proyecto, se conseguirá una utilidad importante para la actividad laboral de un desarrollador, motivada por el interés en el **diseño web adaptativo**. Pero, el uso de esta utilidad, podrá extrapolarse a **otros ámbitos de aplicación**.

Como resultado, se podrá comprobar la **adaptación** de una aplicación web en distintos dispositivos en tiempo de desarrollo, se conseguirá acceso remoto a bases de datos, se podrá alojar una API REST, etc. Como se comentó el apartado anterior, el límite es la imaginación y necesidades del propio usuario.

Poniendo la situación en contexto, y dado un **ciclo de desarrollo** típico observable en la figura 3.2, el servidor creado puede ofrecer utilidades entre las fases 3 y 6.

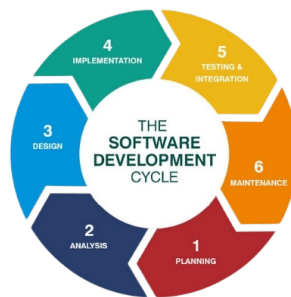


Figura 3.2: Ciclo de desarrollo software

3.4. Alojamiento web

En este caso, se encuentra otra funcionalidad y posible uso para un desarrollador: **alojamiento de su propia página o aplicación web**.

Esto, siempre y cuando no se necesite atender una ingente cantidad de peticiones (por motivos obvios, teniendo en cuenta el hardware utilizado). Pero sería más que suficiente para alojar, por ejemplo, un **portafolio**, sin necesidad de realizar una aportación monetaria mensual a una empresa de *hosting* o alojamiento web.

Además, en el contexto web, puede aprovecharse el servidor para alojar aplicaciones en **fase de pruebas** u ofrecer servicios de manera **privada**, y temporal, no indexados en motores de búsqueda.

3.5. NAS

Este apartado se corresponde con un **objetivo que se decidió eliminar** al inicio del desarrollo del proyecto, dado que se consideró que consumiría un tiempo que beneficiaría más al proyecto al dedicarse a tareas de documentación y, sobre todo, securización.

En la figura 3.3 podemos observar un **servidor NAS**. Este es un dispositivo de conectado a la red, con **funcionalidades** como el propio almacenamiento, nube privada, descargas P2P, centro multimedia, servidor web, servidor FTP, VPN, entre otras. [2].



Figura 3.3: Servidor NAS

De cualquier modo, es un objetivo que se mencionará de nuevo en el capítulo 9, concretamente en la sección 9.3, correspondiente al posible **trabajo futuro** para la ampliación y mejora del proyecto.

3.6. Cliente RDP

Una de las tareas propuestas de forma paralela al proyecto, ya que no posee relación directa con el mismo, es el uso de un **cliente RDP** para acceder a los **escritorios virtuales de la UPM**, utilizando el hardware (de bajo presupuesto) empleado en el proyecto.

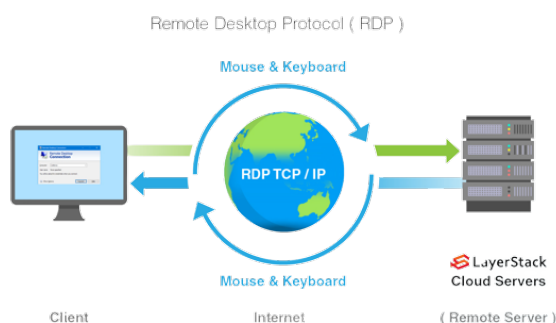


Figura 3.4: Protocolo de escritorio remoto

A pesar de ser una tarea secundaria, resulta bastante interesante de cara a alumnos que posean **escasos recursos económicos** para afrontar el curso académico.

Esta situación resulta beneficiosa, ya que se emplea un *hardware* de bajo coste con el que se puede acceder a los escritorios virtuales de la escuela.

De este modo, se estarían aprovechando **los recursos *hardware* ofrecidos por la universidad** a través de los escritorios mencionados, teniendo en cuenta el aumento en rendimiento que conlleva.

Este apartado se comentará brevemente en el “capítulo 10.4”, perteneciente al **anexo** del documento.

Capítulo 4

Estado del Arte

Este capítulo expone, en relación con la propuesta del proyecto, qué aspectos se han abordado por otras personas o entidades previamente. Se comenta además, qué opciones existen en este ámbito de forma previa al desarrollo del proyecto, destacando qué puede añadir o mejorar el proyecto al respecto.

4.1. Actualidad

En la actualidad, existen algunos proyectos de **alcance similar** al planteado con el desarrollo del trabajo. Sin embargo, estos **no llegan a cubrir todos los puntos tratados** en el desarrollo de este proyecto [3, 4, 5, 6, 7, 8].

Cada uno de los proyectos previamente investigados posee alguna **carencia** diferente, a destacar:

- | | |
|---------------------------------------|--|
| ■ Necesidad de conocimientos previos. | ■ Proyecto incompleto. |
| ■ Falta de securización. | ■ No accesibilidad desde internet. |
| ■ Falta de configuración. | ■ No automatización. |
| ■ Demasiada especificidad. | ■ Escasez de documentación. |
| ■ Demasiada transparencia. | ■ Sin cobertura de requisitos específicos. |

Es posible que, aunando todos los proyectos que se encontraron, se obtuviese un resultado bastante próximo al planteado con este proyecto. Pero, dado que uno de los objetivos principales consiste en adquirir un **alto nivel de aprendizaje** al poner en práctica y relacionar varios conocimientos del grado universitario, la mejor opción es lograr el desarrollo **comenzando desde cero**.

4.2. Novedades

Teniendo en cuenta lo mencionado en el apartado anterior, el proyecto planteado **añade ciertas funcionalidades** o características que no se pueden encontrar en las alternati-

vas ya existentes. Consecuentemente, se obtiene **un motivo más** para el desarrollo del proyecto.

Una de estas características la posibilidad de **automatizar la gran mayoría del proceso**, sin renunciar a ninguna de las características o requisitos iniciales.

Además, la **facilidad de configuración** del servidor: siendo informado acerca de los efectos de cada una de las configuraciones, ya sea a través de la interfaz de línea de comandos, o en la documentación que se provee.

4.3. Aclaraciones

Cabe destacar que, al ser el objetivo principal del proyecto nutrir el conocimiento del desarrollador, esta sección no posee un gran peso al desarrollo del mismo como puede ocurrir en otros casos.

De este modo, y yendo a un ámbito más profesional, cabe destacar el software de Open Media Vault, que posibilita la **generación de un servidor NAS** al completo en un equipo de características similares al utilizado [9].



Figura 4.1: Open Media Vault

Este se presenta como una gran opción si únicamente se requiere obtener las funcionalidades que ofrece, **sin entender el funcionamiento interno** del mismo.

Capítulo 5

Tecnologías

Este capítulo expone el conjunto de tecnologías planteadas al inicio del proyecto, así como las finalmente seleccionadas para llevar a cabo el trabajo desarrollado. Se justificará cada una de las elecciones, teniendo en cuenta las opciones descartadas, sus ventajas e inconvenientes.

5.1. Hardware

5.1.1. Alternativas

Las posibilidades de elección de hardware para este proyecto son **numerosas y variadas**, ya que **cualquier máquina** puede utilizarse como servidor y realizar el proceso planteado. Pero, teniendo en cuenta algunos de los requisitos planteados en el capítulo 6, podemos realizar un primer filtrado en el que obtendríamos las siguientes opciones [10, 11]:

- | | | |
|-----------------------|-----------------|------------------------|
| ■ Raspberry Pi 4 | ■ Rock Pi 4 | ■ PocketBeagle |
| ■ ASUS Tinker Board S | ■ Banana Pi M64 | ■ Orange Pi Plus2 |
| ■ LePotato | ■ Odroid XU4 | ■ Arduino Mega 2560 R3 |
| ■ La Frite | ■ Odroid C2 | ■ Onion Omega 2 Plus |
| ■ Rock64 Media Board | ■ UDOO x86 | ■ Nvidia Jetson Nano |

Esta es una lista de pequeños **equipos de bajo consumo**, ya que, como se verá más adelante, es uno de los principales requisitos. Esto es así, ya que, al estar conectado de manera continua, se tiene en cuenta el consumo energético del equipo.

5.1.2. Elección

El hardware seleccionado es una **Raspberry Pi 4 Model B**, en su versión de 8GB de memoria RAM. Este, se acompaña de una serie de pequeños **disipadores** para lograr un mejor mantenimiento de los elementos más proclives a las altas temperaturas.

Esta decisión se basa inicialmente en los resultados obtenidos de un análisis de la relación **calidad/precio** de los equipos mencionados. Además, la **popularidad** fue un factor

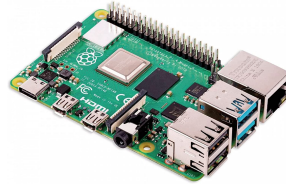


Figura 5.1: Raspberry Pi 4 Model B 8GB

determinante para asegurar la facilidad de obtener información durante el desarrollo del proyecto.

5.2. Sistema Operativo

5.2.1. Alternativas

Entre los distintos sistema operativos para la máquina que actúa de servidor, se contemplaron las **opciones más livianas de Linux**, ya que, uno de los requisitos recogidos en el capítulo 6 hace referencia al rendimiento de la misma [12].

Teniendo esto en cuenta, algunas de las distribuciones más livianas encontrados, con la menor cantidad de procesos y herramientas por omisión con en instalación, serían [13]:

- | | | | |
|-----------------|--------------|-----------|----------------|
| ■ R. Pi OS Lite | ■ PiCore | ■ RISC OS | ■ Sugar OS |
| ■ DietPi | ■ Arch Linux | ■ Raspup | ■ Alpine Linux |

Todos ellos son, en su mayoría, sistemas *headless*, es decir, sistemas que únicamente disponen de **interfaz de línea de comandos**. Esto provoca un enorme **ahorro en el procesamiento** base del sistema operativo, ya que no precisa de renderizado gráfico para el usuario.

5.2.2. Elección

El sistema operativo finalmente seleccionado fue **Raspberry Pi OS Lite**. La alternativa *headless* del sistema operativo desarrollado directamente por el equipo creador del propio hardware.

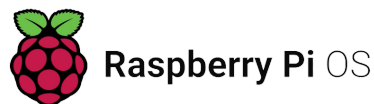


Figura 5.2: Raspberry Pi OS Lite

A pesar de no ser la opción más liviana, esta elección fue motivada por su origen, ya que es una **distribución creada por el propio equipo desarrollador del hardware adquirido**. Por este motivo, se consideró esta opción como la más optimizada para el equipo que actuará de servidor.

5.3. Conexión

5.3.1. Alternativas

Este apartado posee una alternativa de decisión doble: **interfaz de red** y **método de conexión**. En ambos casos, las opciones disponibles son bastantes más reducidas en comparación con el resto de los apartados.

La **interfaz de red** hace referencia a cómo se conecta el equipo a la red, ya sea a la red local o a internet. En este caso, las opciones que ofrece el hardware seleccionado son:

- Cableada
- Inalámbrica

Por otro lado, los **métodos de conexión** se refieren a cómo se realiza el acceso a la máquina servidora, ya sea utilizando un teclado y monitor conectado al equipo, o desde un equipo externo de manera remota [14]:

- Local
- Remota RSH
- Remota VNC
- Remota TELNET
- Remota SSH
- Remota RDP

5.3.2. Elección

En ambos casos, la opción es bastante clara, y atiende a los requisitos de **rendimiento**, **acceso remoto** y **seguridad** que se mencionan en el capítulo 6.

Siempre que sea posible, se realizará la conexión cableada a través de **ethernet**, ya que esto ofrecerá una mayor velocidad de acceso a la red y desde la red y evitará el consumo del módulo eléctrico.



Figura 5.3: Conexión Ethernet



Figura 5.4: Conexión Wifi

La opción de **conexión local** estará siempre disponible utilizando el hardware necesario. Además, se eligió ofrecer la opción de **conexión remota** utilizando el **protocolo de conexión SSH**, ya que la interfaz de la máquina es únicamente de línea de comandos, y SSH provee de la mayor seguridad y calidad de conexión entre las ofrecidas.



Figura 5.5: Conexión SSH



Figura 5.6: Open SSH

5.4. Scripting

5.4.1. Alternativas

La mayor parte del desarrollo del proyecto se realiza empleando un **lenguaje de *scripting*** que permita realizar las modificaciones necesarias en los **archivos de configuración** del sistema.

Para realizar la tarea de configuración, las alternativas disponibles son [15]:

- | | | | | |
|--------------|----------|----------|--------|-------|
| ▪ JavaScript | ▪ Python | ▪ Groovy | ▪ Lua | ▪ VBA |
| ▪ PHP | ▪ Ruby | ▪ Perl | ▪ Bash | ▪ GML |

5.4.2. Elección

El lenguaje seleccionado para llevar a cabo las tareas de desarrollo principales es **Bash**.



Figura 5.7: Bash Scripting

En este caso, el motivo de elección es bastante sencillo: la mayoría de servidores en el mundo (aproximadamente un 96,3% [12]) posee Linux como sistema operativo, el cual suele tener **de forma predeterminada una terminal con Bash**.

Esta elección nos permite que el desarrollo tenga una **alta compatibilidad** con todo tipo de distribución Linux desde los primeros momentos tras la instalación de la misma.

De este modo, si se decidiese cambiar la distribución empleada para el proyecto, el número de modificaciones necesarias sería mínimo.

5.5. Servidor Web

5.5.1. Alternativas

Dado que necesitaremos desplegar aplicaciones web, es necesario un **servidor web**.

En este caso, seleccionando las **alternativas más ligeras** para lograr el mejor rendimiento posible, disponemos de varias opciones que resultaría interesante tener en cuenta para alojar las aplicaciones web [16]:

- | | | | |
|---------|----------------|-----------------|------------|
| ▪ Nginx | ▪ Lighttpd | ▪ OpenLiteSpeed | ▪ Hiawatha |
| ▪ Caddy | ▪ MonkeyServer | ▪ Cherokee | ▪ Apache |

5.5.2. Elección

El servidor web escogido para el alojamiento de las aplicaciones es **Nginx**.



Figura 5.8: Nginx

El criterio de elección, al igual que en el resto de los casos, tiene en cuenta la **popularidad del software** para tener acceso a una buena cantidad de información. Además, el este servidor web es **utilizado de manera recurrente en el ámbito laboral**, por lo que los conocimientos adquiridos en la configuración del mismo serán también de utilidad.

5.6. Base de Datos

Dado que no son los apartados principales del proyecto, y únicamente se utilizan a modo de ejemplo, **tanto en esta sección como en la 5.7 y 5.8**, el criterio de elección se ha basado en tres aspectos principales: **popularidad, demanda laboral y posibilidad de aprendizaje**.

5.6.1. Alternativas

Atendiendo a la posibilidad de aprendizaje, el primer filtro aplicado fue mantener únicamente las bases de datos **NoSQL**. De este modo, conocería una **tecnología nueva**, ya que a lo largo del grado universitario, este tipo de bases de datos nunca se puso en práctica.

Teniendo esto en cuenta, las opciones barajadas fueron [17, 18, 19]:

- | | | | |
|-----------|-------------|------------|-----------|
| ■ MongoDB | ■ Cassandra | ■ CouchDB | ■ Riak |
| ■ Redis | ■ HBase | ■ OrientDB | ■ RavenDB |

5.6.2. Elección

El sistema de bases de datos seleccionado para el desarrollo de la persistencia del *backend* de la aplicación de ejemplo es, tanto por su **ligereza** como por el volumen de **información disponible** para el desarrollo: **MongoDB**.



Figura 5.9: MongoDB

Además, tanto esta como las próximas decisiones, fueron influenciadas por el interés de aprender un **conjunto de herramientas altamente demandado** en la actualidad del mundo laboral: **MERN**.

MERN es un acrónimo que hace referencia al uso de las tecnologías: MongoDB, Express, React y Node. Teniendo esto en cuenta, se anticipan las elecciones de los próximos apartados.

Figura 5.10: MERN *stack*

5.7. Backend

Para el desarrollo del *backend*, se seleccionó **Node** junto al *framework* de **Express**, realizando una API REST simple, pero suficiente para comprobar el correcto funcionamiento en el servidor.



Figura 5.11: Node JS



Figura 5.12: Express JS

5.8. Frontend

Por último, para el desarrollo del *frontend* se seleccionó **React**, creando con ello una **SPA**, de nuevo simple, pero igualmente suficiente para comprobar el correcto funcionamiento del servidor al alojar una aplicación web.



Figura 5.13: React JS

Capítulo 6

Requisitos

Este capítulo expone, de manera sencilla, el origen de cada una de las decisiones tomadas en el desarrollo del proyecto, así como el motivo de implementación de las principales funcionalidades.

6.1. Recopilación

6.1.1. Servidor doméstico

El principal objetivo del proyecto es crear un **servidor doméstico** que ejecute sobre el hardware de una **Raspberry Pi 4**, e idealmente en sus versiones anteriores o posteriores.

6.1.2. Automatización

La creación del servidor deberá ser fácilmente **replicable**, de forma automatizada o a través de un panel de configuración.

6.1.3. Configurabilidad

Tras la instalación, el servidor deberá ser **configurable** de manera sencilla y **documentado** adecuadamente para conocer el alcance de cada funcionalidad.

6.1.4. Securización

El servidor deberá mantener un **alto nivel de seguridad**, tanto para el propio servidor, evitando intrusiones en el mismo, como para un cliente que realice una conexión.

6.1.5. Monitorización

Con cada acceso al sistema, se mostrará de manera visual e intuitiva **información relevante para el administrador**, o cualquier usuario, acerca del sistema.

6.1.6. Acceso remoto

El servidor deberá ofrecer la posibilidad de **acceder a él de manera remota** a través de internet, manteniendo la seguridad del mismo.

6.1.7. Aplicación web

Para comprobar el correcto funcionamiento y accesibilidad desde internet, se creará una **aplicación web** sencilla que será alojada en el propio servidor.

Esta aplicación, realizará consultas a una **API REST** que, a su vez, consultará una **base de datos**, ambas **alojadas en el servidor**.

De este modo, se comprobará el correcto funcionamiento de los diferentes servicios configurados.

6.2. Análisis

Teniendo en cuenta los requisitos expuestos en la sección anterior, obtenemos las siguientes conclusiones del proyecto una vez finalizado:

Dado el hardware de una **Raspberry Pi**, el proyecto ejecutará un **script** que, como resultado, generará las **configuraciones necesarias** de un servidor que ofrezca varios servicios: **alojamiento web**, base de datos y **API REST**, todos de forma **segura** y **configurable**.

Por último, se ofrecerá una **aplicación web** simple **accesible desde internet**, que haga uso de una API REST con su correspondiente **base de datos**, demostrando el correcto funcionamiento del servidor. QQQQ

Capítulo 7

Desarrollo del servidor doméstico

Este capítulo expone el proceso de desarrollo del trabajo realizado, explicando cada una de las fases de implementación, junto a los problemas que hayan podido surgir, así como las soluciones aplicadas a cada uno de ellos.

7.1. Hardware

El primer paso para comenzar con el desarrollo, fue adquirir el hardware necesario para construir el servidor doméstico, en este caso, una **Raspberry Pi 4 Model B 8GB** (figura 7.1).

Para ello, a finales del mes de **diciembre de 2021**, una vez la propuesta del proyecto fue aceptada, **se ordenó la compra**. Pero, surgió un problema por **falta de abastecimiento**, que fue posible solucionar gracias a la intervención del tutor, Víctor Robles Forcada, quién pudo proveerme una **Raspberry Pi 3 Model B 1GB** (figura 7.2) para comenzar con el desarrollo del proyecto.

Por ello, a pesar de los inconvenientes, los plazos establecidos con el primer Plan de Trabajo, pudieron cumplirse de manera satisfactoria. Y, como consecuencia, el desarrollo del proyecto se llevó a cabo con el hardware provisto por Víctor: la Raspberry Pi 3 Model B 1GB.

Por último, tras 5 meses de retraso en la entrega, el equipo inicialmente elegido, la Raspberry Pi 4 Model B 8GB, se utilizó para realizar unas **pruebas de instalación** finales, junto a la monitorización de consumo para establecer **comparaciones** entre las distintas versiones.



Figura 7.1: Raspberry Pi 4 Model B 8GB



Figura 7.2: Raspberry Pi 3 Model B 1GB

7.2. Sistema operativo

En este caso, la **elección** del sistema operativo, finalmente **Raspberry Pi OS**, fue **más difícil** que la propia **instalación** del mismo. Además, la instalación fue una de las tareas **más sencillas** [20].

Esto sucedió porque se barajaron varias opciones de reducido tamaño y consumo, pero, a pesar de encontrar otras recomendaciones, la seguridad de utilizar una **distribución específicamente diseñada para el hardware empleado**, marcó la diferencia en la elección del mismo [13].

Además, la elegida fue la versión *headless* de la distribución: **Raspberry Pi OS Lite**. Es decir, la versión que no posee interfaz gráfica para interaccionar con el usuario, y en su lugar, se maneja a través de una **interfaz de línea de comandos**. De este modo, se **ahorra en consumo y en recursos** que podrán, en su lugar, ser dedicados al rendimiento del sistema y de los servicios empleados.

Esta modalidad de uso, se replica de servidores de grandes compañías, los cuales siguen este esquema de interfaz de línea de comandos.

Cabe destacar que, para el uso de este tipo de sistemas, asignaturas como **Programación Para Sistemas** o **Sistemas Operativos** ofrecen conocimientos valiosos que pueden ser aplicados.

7.3. Configuración

Al inicio de la ejecución del script desarrollado, y durante el proceso de configuración, **se instalan automáticamente** todos los paquetes y herramientas necesarias para la configuración y gestión del servidor.

Además, se crea un *banner* o **pantalla de bienvenida** al iniciar sesión en el servidor, ya sea de manera local o remota, así como un *prompt* amigable con información útil añadida al configurado por omisión. Estos aspectos, se visualizarán de manera gráfica en la sección 7.8 de monitorización.

De igual modo, existen otras configuraciones que se irán realizando a lo largo del **proceso de instalación**. Todas las configuraciones relevantes se comentarán en su respectiva sección para mantener el contexto.

7.4. Securización

Esta es una sección a la que prestar atención desde el inicio del proyecto. Además, debe revisarse de forma previa y posterior a la instalación de cada servicio utilizado, realizándose las revisiones correspondientes para **detectar posibles brechas de seguridad e impedir el acceso a personas no autorizadas** [21].

En especial, debe tenerse en cuenta antes de **exponer el servidor a internet**, momento en el que **cualquier persona con acceso a la red podría tratar de tomar el control sobre el servidor**, intentando obtener información protegida o produciendo daños a la infraestructura.

Este fue el apartado al que se dedicó un mayor esfuerzo, ya que, mientras se desarrollaba el *banner* de inicio para la monitorización de acceso y recursos, se detectó una **enorme cantidad de intentos de intrusión** al sistema. Este suceso, se extenderá brevemente en la sección 7.8.

Para no adelantarse en el desarrollo, y al igual que en la sección anterior, todos los aspectos de seguridad relevantes se comentarán en cada sección cuando lo requiera.

7.5. Servicios

El servidor ofrece al usuario distintos **servicios y recursos**:

- SSH para acceso remoto al servidor.
- HTTPS para acceso a aplicación web y API REST.
- Base de datos interna, para almacenamiento de datos de la API REST.

Pero, como se comentará en la sección 9.3 del capítulo 9, referente al posible trabajo futuro, **se podrían añadir otros muchos servicios**, logrando obtener un servidor NAS totalmente personalizado.

De hecho, la creación de un servidor NAS fue uno de los objetivos iniciales del proyecto, pero como se expone en la sección 3.5, esta tarea poseía una alta demanda en tiempo de dedicación, que finalmente se empleó en el apartado de **segurización** del servidor por ser más prioritaria.

7.6. Conexión SSH

Para habilitar la conexión al servidor mediante SSH, el proceso es sencillo teniendo en cuenta la distribución de Linux utilizada: basta con **generar un archivo** vacío nombrado *ssh* en la partición de arranque, bajo el directorio */boot* del sistema de ficheros [22]. De este modo, quedará habilitada la **conexión al servidor desde un equipo externo** a través de SSH.

Aunque, con esta configuración, únicamente se podrá acceder al servidor desde la **red local**. Para poder acceder desde cualquier lugar y gestionar el servidor **de manera remota a través de internet**, se establecen ciertas configuraciones detalladas en la sección 7.7 de este mismo capítulo.

Por otro lado, se configurará la **IP de la máquina de manera estática**, para evitar problemas de conexión que pueda producir el servicio de DHCP del router al que se conecte. Además, se aconseja añadir al fichero */etc/hosts*, del equipo cliente, la IP estática asignada junto a un nombre, en este caso: **raspberry** [23].

De este modo, en la fase inicial de configuración, podríamos acceder al servidor utilizando un nombre en lugar de la propia IP. En el caso actual, se estableció como dirección IP estática la **192.168.1.3**, ya que 192.168.1.1 y 192.168.1.2 pertenecen a los routers conectados a la LAN. Teniendo esto en cuenta, la entrada añadida al fichero */etc/hosts* sería la siguiente:

```
192.168.1.3    raspberry
```

Con esta configuración, utilizando el usuario por defecto, podríamos realizar la **conexión al servidor** con cualquiera de los siguientes comandos:

```
$ ssh pi@192.168.1.3
```

```
$ ssh pi@raspberrypi
```

Adicionalmente, en la configuración de este apartado, si no se poseen, se solicitará crear unas **claves SSH** para mantener un **acceso seguro** al servidor. Así como un **agente SSH** que gestione los inicios de sesión, facilitando las conexiones realizadas. Esta es una parte con una **gran influencia en la securización** del servidor [24, 25].

Por último, ampliando lo referente a los **aspectos de seguridad**, se realizan varias **tareas** en este apartado [21, 26, 27, 28]:

- Modificar el **usuario** y **contraseña** por defecto.
- Habilitar la conexión con **claves SSH**, en este caso RSA con algoritmo SHA-256.
- **Deshabilitar** la conexión como usuario *root*.
- **Deshabilitar** la conexión con usuario y contraseña.
- **Instalar y habilitar un firewall** para todos los puertos excepto los necesarios para los servicios ofrecidos, hasta el momento SSH.
- Una buena práctica es utilizar un **puerto distinto al bien conocido** para las conexiones. Es decir, usar el puerto 22022 (por ejemplo) en lugar del 22, para evitar el reconocimiento y acceso a través de ataques automatizados.

Teniendo esto en cuenta, el modo de (sección 7.8) habitual se realizaría a través de un comando del siguiente tipo:

```
$ ssh user@hostname -p PORT
```

Siguiendo las buenas prácticas de securización, el **acceso al servidor configurado** se realizaría con:

```
$ ssh rgdlr@raspberrypi -p 22022
```

7.7. Acceso desde internet

Para lograr que el servidor sea accesible desde internet, tanto en conexión remota con SSH, como para el uso de los diferentes servicios ofrecidos, es necesario realizar una serie de tareas que, si bien pueden lograr **automatizarse** en cierta medida, **no es lo más aconsejable** dados los requisitos recopilados.

Teniendo en cuenta que estaríamos utilizando un servidor doméstico, ubicado en una vivienda, y con el acceso a internet típico, las tareas necesarias son:

- Conocer la dirección **IP pública** en la que se localiza el servidor.
- Utilizar **DDNS** para manejar los posibles cambios de IP del proveedor.
- **Abrir los puertos** del router correspondientes a los servicios utilizados.



Figura 7.3: Router

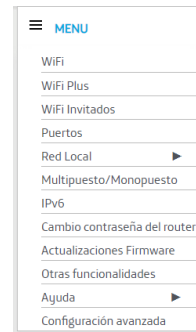


Figura 7.4: Menú del router

A pesar de intentar automatizar todo el proceso de instalación, el uso de **DDNS** y el **manejo de puertos**, es una parte que se decidió tratar **de forma manual**. En el primer caso, para eliminar carga del servidor, y en el segundo, para evitar posibles problemas de seguridad.

Además, en ambos casos, en el repositorio **se documentó el proceso** debidamente para facilitar las configuraciones manuales.

7.7.1. DDNS

Para el uso de DDNS, o **sistema dinámico de nombres de dominio**, se utilizó `https://www.noip.com`, un proveedor de servicios de dominio y *host* que cuenta con un plan gratuito. El inconveniente de este plan es que se debe confirmar la renovación cada mes para usarlo de manera continuada.

Para el uso de este servicio, se contemplaron 2 posibilidades:

1. Ejecución a través de **Docker** del servicio en el propio servidor.
2. Ejecución del servicio en el **router** [29].

Se escogió la opción 2 para **evitar añadir carga al servidor**. Como ya se comentó, esta es una parte que requiere de la actuación manual del usuario. Se debe **registrar en el proveedor** y **escoger un dominio** entre los ofrecidos para poder realizar el resto de configuraciones, tanto manuales como automatizadas (en las que se solicitará el dominio).

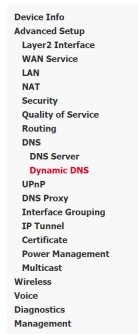
7.7.2. Puertos

Para la gestión de puertos, existen dos posibilidades:

1. **Habilitar** en el router la opción **UPnP** (*Universal Plug and Play*), para posteriormente poder automatizar la apertura y cierre de puertos [30, 31, 32].
2. Realizar en el router la **apertura y cierre de puertos de manera manual**.

Dado que ambas requieren un mayor o menor grado de interacción manual, accediendo a la configuración del router, se escogió la opción 2, ya que es una tarea realmente sencilla y puede **evitar problemas de seguridad**. Además, es útil conocer cómo realizar las distintas configuraciones de cara a posibles modificaciones futuras.

7.8. Monitorización



Add Dynamic DNS

This page allows you to add a Dynamic DNS address from DynDNS.org, No-IP.com or NowIP.com.

D-DNS provider:

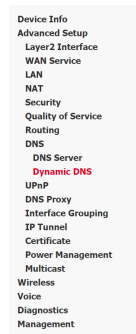
Hostname:

Interface:

No-IP Settings

Email:

Password:



Dynamic DNS

The Dynamic DNS service allows you to alias a dynamic IP address to a static hostname in any of the many domains, allowing your Broadband Router to be more easily accessed from various locations on the Internet.

Choose Add or Remove to configure Dynamic DNS.

Hostname	Username	Service	Interface	Remove
rgdlr.sytes.net	rgdlr.sytes@hotmail.com	noip	ppp0.1	<input type="checkbox"/>

Figura 7.5: Formulario DDNS en el router

Figura 7.6: DDNS añadido al router

Tabla actual de mapeo de puertos						
	Nombre	Protocolo	Puerto/Rango Externo	Puerto/Rango Interno	Dirección IP	Activar
✗	raspi-ssh	TCP	22	22	192.168.1.3	<input type="checkbox"/>
✗	raspi-sshs	TCP	22022	22022	192.168.1.3	<input checked="" type="checkbox"/>
✗	raspi-http	TCP	80	80	192.168.1.3	<input checked="" type="checkbox"/>
✗	raspi-https	TCP	443	443	192.168.1.3	<input checked="" type="checkbox"/>
✗	raspi-rest	TCP	44380	44380	192.168.1.3	<input checked="" type="checkbox"/>

Figura 7.7: Configuración de puertos

Teniendo en cuenta todos los apartados expuestos en esta sección, para realizar una **conexión mediante SSH desde el exterior de la red**, encontraríamos que el comando más habitual sería:

```
$ ssh rgdlr@rgdlr.sytes.net -p 22022
```

7.8. Monitorización

Para conocer el **estado del servidor**, se realizó un *banner* de inicio de sesión, de modo que al establecer una conexión de manera local o remota, se obtiene **información muy útil acerca del sistema** [33, 34].

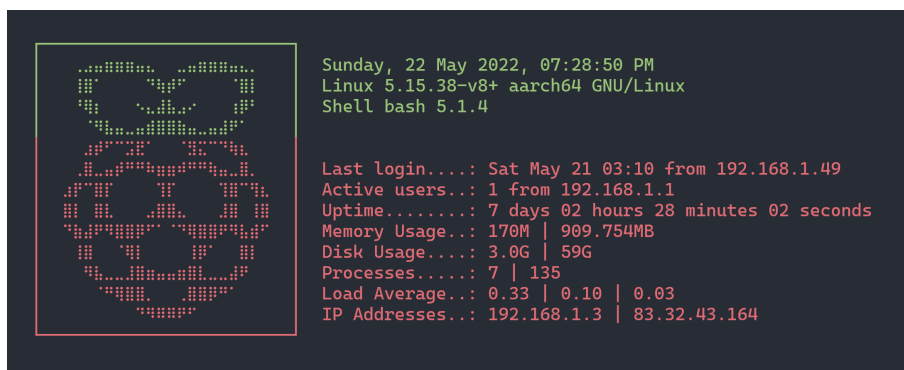


Figura 7.8: Banner de inicio

Esta información se corresponde con los siguientes datos:

- Características básicas del sistema a modo de introducción: **fecha**, versión del **kernel** y de la **shell** del sistema.
- Fecha e identificación por dirección IP de la **última conexión** realizada al servidor. Esta característica es bastante útil para detectar si ha ocurrido una intrusión no autorizada al sistema.
- **Usuarios activos** en el sistema identificados por dirección IP. Esta característica es también útil para detectar accesos al sistema.
- **Tiempo de actividad**. Esta información resulta útil para conocer el tiempo de disponibilidad del sistema y realizar estadísticas del mismo.
- **Uso de memoria principal y disco**, procesos activos a nivel de usuario y sistema, y carga media del sistema. Estas características, además de ser meramente informativas, pueden también utilizarse para detectar situaciones anormales del sistema y detectar también posibles problemas de seguridad.
- **Direcciones IP privada y pública del servidor**. Esta información es meramente informativa y se muestra de modo que se pueda consultar de manera rápida y sencilla.

Como se ha comentado, se ofrece bastante información útil e interesante del sistema, que puede ayudar a **detectar situaciones anómalas** que necesiten atención, así como **posibles intrusiones**.

Por último, ampliando el suceso expuesto en la sección de securización (7.4), cabe comentar que al realizar el proceso de monitorización, se descubrió a través de la herramienta *netstat* que había **usuarios no autorizados tratando de iniciar sesión en el servidor**.

Ante esta situación, **se detectaron ataques de fuerza bruta** con un alto número de peticiones de acceso, y se realizó un **estudio profundo en el ámbito de la seguridad** necesaria para proteger el servidor SSH. Por ello, se siguieron **buenas prácticas y políticas altamente restrictivas**.

Además, se logró observar todo el registro de intentos de inicios de sesión, observando **logs del sistema** como los almacenados en el fichero */var/log/auth.log* o los ofrecidos por herramientas como *journalctl*.

Posteriormente, también se analizaron los almacenados en */var/log/nginx/access.log*, donde se pudo registrar en un mismo día hasta un total de **1000 peticiones maliciosas** contra el servidor web instalado para ofrecer la aplicación de demostración.

7.9. Aplicación Web

Como se comentó en el capítulo 5, referente a las tecnologías empleadas, se desarrolla una aplicación web sencilla para **probar el correcto funcionamiento del servidor** a la hora de **ofrecer recursos a través de internet** [35, 36].

Para comprobarlo, se aloja la aplicación web, que consume una API REST conectada a una base de datos NoSQL. **Todos los recursos** mencionados están, de este modo, **alojados en el servidor**.

El contenido de la propia aplicación no es relevante, pero, en este caso, para relacionarla con el desarrollo del proyecto, se creó un **registro de consumo y costes del hardware utilizado**.

En ella, se puede registrar el precio medio de la electricidad y el consumo del hardware en una fecha determinada. Además, se puede **visualizar en forma de tabla o gráfico cada uno de los valores registrados**.

7.9.1. Backend


Para el desarrollo del backend, se realiza la instalación del sistema gestor de bases de datos, en este caso **MongoDB**, que empleará una base de datos NoSQL para almacenar la información de la aplicación de forma persistente.

Posteriormente, se realiza una conexión a dicha base de datos mediante una API REST, comunicándose con ella a través de peticiones HTTP. Esta, **se aloja en el servidor y se consulta a través de la URL <https://rgdlr.sytes.net/api/v1>**.

La API REST generada ofrece recursos a través de distintos *endpoints*:

- <https://rgdlr.sytes.net/api/v1/records>
 - Obtiene el registro completo de consumo de la base de datos.
 - Se puede observar la respuesta en la figura 7.9
- <https://rgdlr.sytes.net/api/v1/records/2022-05-20>
 - Obtiene el registro de consumo del día 20/05/2022.
 - Se puede observar la respuesta en la figura 7.10

De este modo, se comprueba que **el servidor ofrece los datos de manera correcta**. Además, la API REST creada se consume en el *frontend*, como se comentará en la próxima sección, y se comprueba la **integración de ambos servicios en funcionamiento**.



```

Send Request
GET https://rgdlr.sytes.net/api/v1/records

1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0
3 Date: Sat, 28 May 2022 22:06:45 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 1844
6 Connection: close
7 X-Powered-By: Express
8 Access-Control-Allow-Origin: *
9 ETag: W/"414-unAQL5d88eB5nyauSV3pr0pa8"
10
11 {
12   [
13     {
14       "_id": "6288120c6e27b2f8e0cb79e",
15       "date": "2022-05-12",
16       "consumption": "0.094",
17       "price": "0.265",
18       "cost": "0.025",
19       "__v": 0
20     },
21     {
22       "_id": "62881416e27b2f8e0cb79a",
23       "date": "2022-05-14",
24       "consumption": "0.094",
25       "price": "0.237",
26       "cost": "0.022",
27       "__v": 0
28     }
29   ]
30 }

```



```

Send Request
GET https://rgdlr.sytes.net/api/v1/records/2022-05-20

1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0
3 Date: Sat, 28 May 2022 22:10:41 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 117
6 Connection: close
7 X-Powered-By: Express
8 Access-Control-Allow-Origin: *
9 ETag: W/"75-6maXNQgu3rNGA/2rg0gmyTvaw9H"
10
11 {
12   [
13     {
14       "_id": "628819036e27b2f8e0cb78e",
15       "date": "2022-05-20",
16       "consumption": "0.092",
17       "price": "0.284",
18       "cost": "0.026",
19       "__v": 0
20     }
21   ]
22 }

```

Figura 7.9: Fragmento de respuesta *records* Figura 7.10: Respuesta *records* 20/05/2022

7.9.2. Frontend

Para el cliente, se desarrolla una aplicación web utilizando el *framework* de **ReactJS**, que ejecutará en el navegador.

Este, consume los recursos de registros mencionado en el apartado anterior, y **se accede a él través de la URL** `https://rgdlr.sytes.net`.

En esta aplicación, se ofrece la posibilidad de **añadir y modificar los registros almacenados**. Además, se puede visualizarlos toda la información en forma de gráficos dedicados a cada atributo o en forma de tabla, teniendo una entrada para cada día registrado.

Por último, se puede comprobar que el servidor aloja la aplicación ejecutando correctamente y **atendiendo a unos tiempos de carga mejores de lo esperado**. Con el resultado obtenido, se obtiene una **buena experiencia de usuario** al utilizar la aplicación de demostración.

En las próximas figuras: 7.11, 7.12, 7.13, 7.14, 7.15, se muestran unas capturas de pantalla de la **aplicación desarrollada en ejecución**.

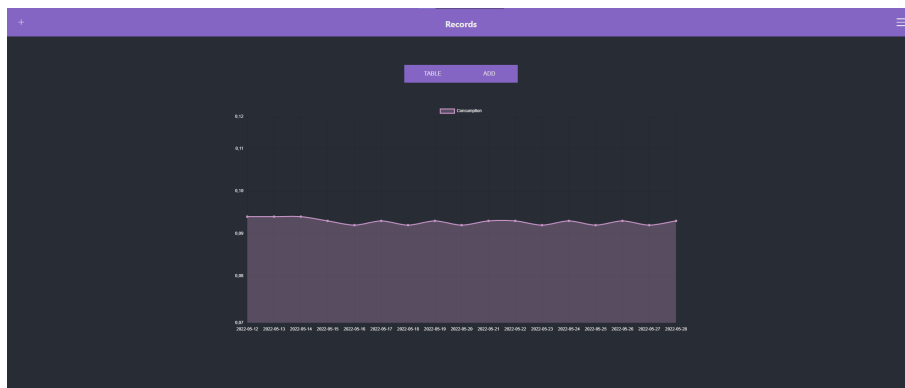


Figura 7.11: Gráfico en la aplicación web de consumo diario de la Raspberry Pi

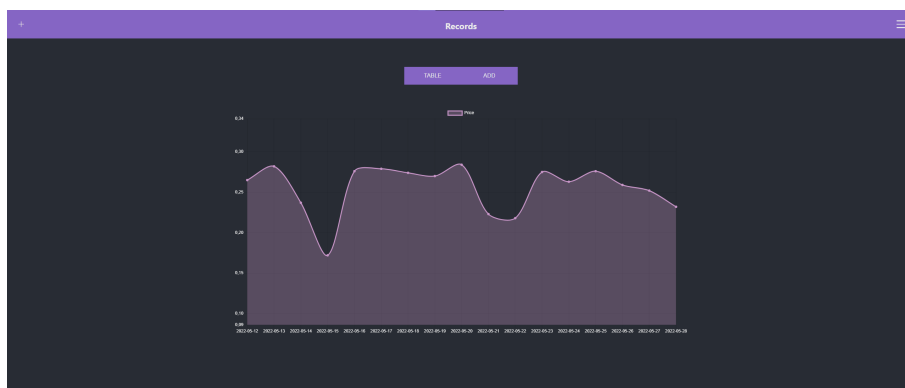


Figura 7.12: Gráfico en la aplicación web del precio medio diario del kWh

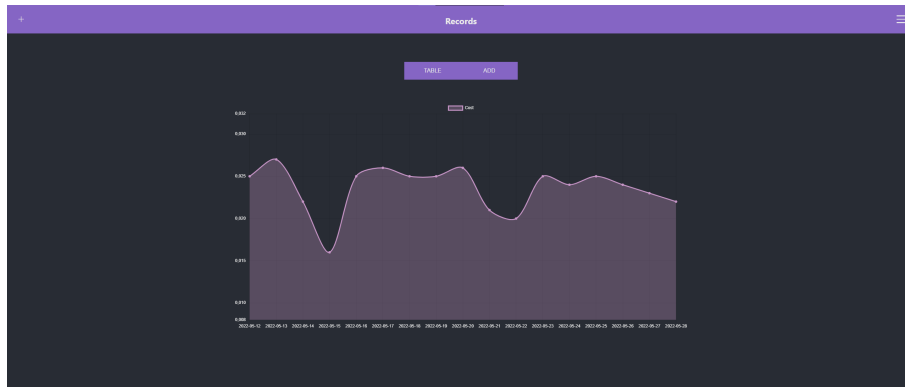


Figura 7.13: Gráfico en la aplicación web del coste diario de la Raspberry Pi

[illegible]

Figura 7.14: Tabla en la aplicación web de registros de la Raspberry Pi

Records

CHART

CANCEL

Date

05/06/2020

Consumption (kWh)

Consumption

Price (ZAR/kWh)

Price

SUBMIT

Figura 7.15: Formulario en la aplicación web de registros de la Raspberry Pi

7.9.3. Protocollo

En el caso de la aplicación web, la parte de securización atendió al **protocolo de comunicación** utilizado para acceder tanto a la propia aplicación, como a la API REST ofrecida.

Para mantener un buen nivel de seguridad desde el servidor hacia el exterior y viceversa,

se obtuvo un **certificado SSL** que permitiese el acceso a las URLs a través del protocolo HTTPS en lugar de HTTP.

De este modo, **las peticiones** realizadas entre cliente y servidor, **se cifrarán** evitando que la interceptación de una petición puedan poner en peligro la privacidad de un usuario utilizando la aplicación.

Para ello, se utilizó **Let's Encrypt**: una autoridad de certificación que proporciona certificados X.509 gratuitos para el cifrado de Seguridad de nivel de transporte (TLS) facilitando la obtención de sitios web seguros [37].



Figura 7.16: Let's Encrypt

Una vez obtenido el certificado, **se integró el proceso en el script** de automatización y configuración del servidor [38, 39].

7.9.4. Despliegue

Para el despliegue de la aplicación desde el equipo de desarrollo al servidor, se ofrece un sencillo **script que transferirá, mediante SSH, los recursos necesarios** al servidor. De este modo, se almacenarán en la localización configurada para ofrecer el servicio.

Para ello, se creó tanto el script, como la **documentación** necesaria para adaptarlo a diferentes necesidades o circunstancias.

Este script contemplará los dos **casos de despliegue** previamente comentados: *backend* y *frontend*, de modo que el proceso resulte rápido y sencillo para el usuario.

7.10. Scripts de automatización

El producto final generado con el desarrollo del trabajo es una serie de scripts que proveen de **automatización** para replicar cada apartado desarrollado en este capítulo, tanto de manera **local**, como de manera **remota**.

Estos, se encargan de realizar, de manera correcta y segura, las **configuraciones** necesarias para el uso de un servidor en una Raspberry Pi.

Inicialmente, se trató de desarrollarlo con la misma herramienta que utiliza el configurador de Raspberry para su sistema: **Whiptail** [40, 41].

Pero, finalmente se decidió desarrollar el script con **interacción a través de la propia terminal**, de modo que se pudiera **visualizar el progreso** de configuración al completo, ofreciendo la posibilidad de retroceder por si fuera necesario configurar algún dato extra.

Al final del capítulo, en la figura 7.18 se puede observar el resultado y salida de la **ejecución del script maestro** del proyecto, ejecutando de forma remota a través de SSH.

7.11. Documentación

Desde el inicio del proyecto, se consideró importante generar buena documentación para el uso de los scripts ofrecidos. De este modo, se logra **facilitar** más aún la **replicación del servidor** junto a las configuraciones establecidas.

Esta documentación, hace referencia tanto al documento actual, para un usuario que quiera conocer los **detalles más internos** en el desarrollo del trabajo, como a la documentación ofrecida en el propio repositorio que contiene el proyecto.

Al final del capítulo, en la figura 7.19, se puede observar el inicio de la **documentación ofrecida en el repositorio** para utilizar el script de configuración.

7.12. Repositorio

Por último, cabe destacar que el proyecto desarrollado será **público en GitHub a través de <https://github.com/rgdlr/raspberry-pi-server>**.

El repositorio está publicado de manera libre, bajo la **licencia *Creative Commons Zero v1.0 Universal***, de modo que cualquier persona puede acceder al trabajo, utilizarlo, e incluso proponer mejoras para el desarrollo del mismo.

Además, tras la entrega del proyecto, en lo referente al ámbito universitario, se continuará desarrollando otros aspectos fuera del alcance de esta entrega (9.3). De este modo, el aprendizaje será continuo, y se podrán aprovechar otro tipo de servicios como los de VPN, FTP, SFTP, SMTP, etc.

```
Raspberry Pi Server

Required information for server configuration
· Packages : dependencies installation
· DDNS URL : IP tracking and public access
· SSH Keys : secure access to server
· Router : public access

Install: packages
✓ Packages update
✓ Packages upgrade
✓ Full upgrade
✓ Dist upgrade
✓ Autoremove

Configuration: DDNS
? Do you have a DDNS URL? (y/n): y
? DDNS URL: rgdlr.sytes.net

Configuration: SSH keys
? Do you have a SSH key? (y/n): y
✓ Network tools install
✓ Raspberry Pi search
  User : pi
  Password : raspberry
  IP : 192.168.1.3
✓ Send SSH key to Raspberry

Configuration: SSH server
✓ SSH configuration
✓ SSH service restart
```

Figura 7.17: Ejecución de script (1)

```
Configuration: firewall
✓ Firewall install
✓ Firewall configuration

Configuration: static IP
! Default Raspberry Pi IP: 192.168.1.3
? Change Raspberry Pi IP? (y/n): n

Configuration: connectivity
! It is recommended to disable BT and WiFi for energy saving
? Would yo like to disable connectivity? (y/n): y

Install: database
✓ Database install

Install: web server
✓ Web server install
✓ Web server create
✓ Web server configuration
✓ Web server restart

Install: backend tools
✓ Backend tools install

Install: SSL certificate
✓ SSL certificate install
✓ SSL certificate configuration

🎉 Installation completed! 🎉
```

Figura 7.18: Ejecución de script (2)

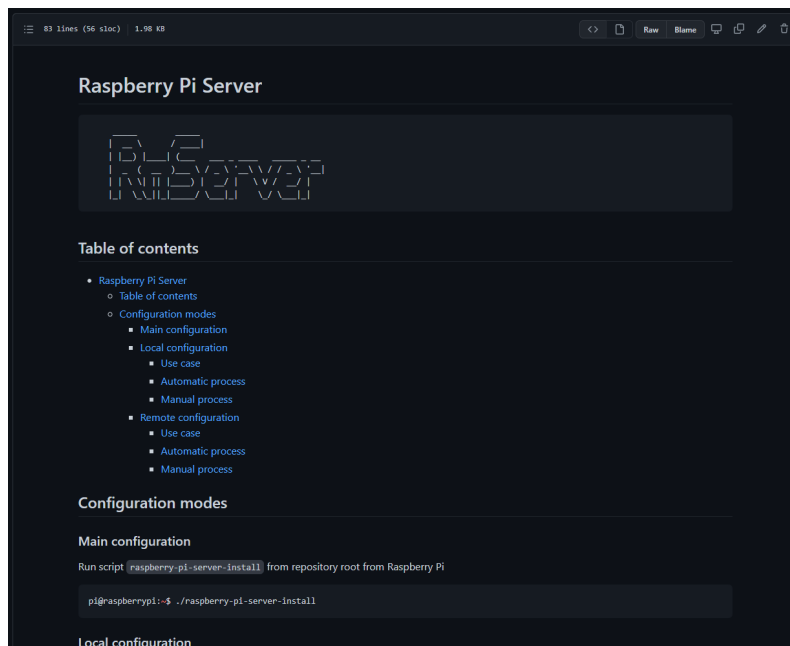


Figura 7.19: Documentación del repositorio

Capítulo 8

Evaluación

Este capítulo expone una evaluación del proyecto desde el punto de vista del usuario.

8.1. Usabilidad

La usabilidad del producto generado es uno de los aspectos más cuidados durante el desarrollo, ya que, se pretende que un **usuario con conocimientos mínimos** pueda realizar la instalación de forma segura.

Además, se trata de conseguir que un **usuario con conocimientos más avanzados** pueda entender cuáles son cada una de las configuraciones establecidas y cómo estas pueden afectar al sistema.

Para evaluar este aspecto, se propuso un **experimento** en el que, con unas bases mínimas: sistema operativo instalado y las instrucciones de instalación ofrecidas en la documentación, un **usuario sin conocimientos** en la informática lograra realizar la configuración al completo.

El **resultado** de este experimento fue parecido al **esperado**, ya que la situación tratada fue la más desfavorable posible, en la que el usuario no conocía ni siquiera qué estaba haciendo. Dada esta situación, se pudieron realizar las **configuraciones más básicas** que ofrece el script automatizado, pero en el momento de realizar las configuraciones manuales (como las correspondientes a los ajustes del router), el proceso se frenó, **no pudiendo realizar la instalación al completo con éxito**.

Sin embargo, esto no es una evaluación en contra del proyecto, ya que se requieren **conocimientos mínimos** para llevar a cabo la instalación **sin renunciar a todos los aspectos de seguridad** tratados en el mismo. De este modo, se **evita comprometer la seguridad de todo un área de red local doméstico**.

8.2. Consumo

Como se comentó en el capítulo 1, el **precio de mantenimiento** del servidor, referido únicamente al consumo del mismo en la situación actual, se adhería a los 0.7801761€.

Pero, si realizásemos los cálculos en una **situación más desfavorable**, en un mes de máximo consumo, sobre 0.125kWh, junto a altos precios, como pudo ser el mes de marzo de 2022 que alcanzó un máximo histórico de 0,399834€/kWh, obtendríamos un coste de **1.4993775€ mensuales**.

En contraposición, con una media de consumo diaria de 0.1kWh, y un precio medio diario de 0,11515€/kWh, correspondiente a los de un año atrás (marzo de 2021), obtendríamos un coste de **0.34545€ mensuales**.

8.3. Alternativas

Teniendo en cuenta el apartado anterior, cabe mencionar las alternativas existentes para un **usuario que requiera de un servidor**. En este caso, lo compararemos con una de las plataformas mejor valoradas en la primavera del año actual, 2022.

Esta plataforma recibe el nombre de **Linode**: una solución de administración en la nube que ayuda a empresas de todos los tamaños a gestionar su infraestructura informática en la nube y definir reglas de acceso a la red [42, 43].

Además, existen otros servicios de este tipo, IaaS (*Infrastructure as a Service*), como [44]:

- | | | | |
|-----------------|------------|-----------|--------------|
| ▪ Hostwinds | ▪ G2 Deals | ▪ Vultr | ▪ Amazon EC2 |
| ▪ Digital Ocean | ▪ OVHCloud | ▪ Upcloud | ▪ Google CE |

Pero, en la fecha actual, ninguno de ellos alcanza la **relación calidad/precio** ni la variedad de opciones que ofrece la alternativa comparada.

8.4. Rentabilidad

La **opción más económica** que ofrece Linode es de **5€ al mes**, siendo esta un recurso compartido con las siguientes características [45, 46]:

- | | |
|------------------------|---------------------------|
| ▪ 1 CPU Core | ▪ 1 TB Transferencia |
| ▪ 1 GB Memoria RAM | ▪ 40 Gbps Entrada de Red |
| ▪ 25 GB Almacenamiento | ▪ 1000 Mbps Salida de Red |

Esta opción, **se lograría amortizar en aproximadamente 8 meses**, teniendo en cuenta el hardware de la **Raspberry Pi 3 Model B 1GB**, con el que se ha desarrollado el proyecto.

Pero, teniendo en cuenta el hardware con el que se planteaba el proyecto al inicio, la **Raspberry Pi 4 Model B 8GB** (con un precio de aproximadamente 90€) la alternativa de Linode que más se asemeja, se situaría en los **40€ mensuales, logrando amortizar la inversión en menos de 3 meses** [46].

Capítulo 9

Conclusiones

Este capítulo expone qué resultados y conclusiones se han obtenido con el desarrollo del proyecto, teniendo en cuenta los objetivos expuestos al inicio del mismo.

Además, se expone qué posibilidades de ampliación podría tener este proyecto de cara al mantenimiento y mejoras futuras.

9.1. Principal

Como se ha comentado a lo largo del documento, uno de los principales objetivos consistía en lograr **aunar una gran parte de los conocimientos adquiridos durante el grado universitario**. Conocimientos que desde el inicio consideré interesantes de cara a la salida al mundo laboral, y que podrían ofrecerme posibilidades de desarrollo personal.

Pero, no sólo logré cumplir ese objetivo, sino que pude **ampliar mis conocimientos en lo referente a la administración de sistemas y servicios**, guiado por la securización del propio servidor y la extracción de información del mismo a través de la monitorización.

Además, esta característica de securización a la que se dedicó un especial esfuerzo, **me ha nutrido con valiosos conocimientos que considero fundamentales** y que resultarían interesantes, y sobre todo útiles, al ser impartidos en alguna asignatura del grado.

Finalmente, destacar que **el resultado final del proyecto fue muy satisfactorio**, pero sobre todo lo fue el tiempo de desarrollo del mismo.

9.2. Secundarias

Del desarrollo de este trabajo, se pueden obtener multitud de conclusiones y un **aprendizaje valioso en diversos campos de la informática**, sobre todo en cuanto parte más **práctica**. De este modo, y relacionando los temas tratados en el proyecto con ciertas asignaturas o partes del grado, podemos definir que:

- La **configuración** del servidor y su **automatización** se relaciona con asignaturas como: Sistemas Operativos, Programación Para Sistemas, Redes de Computadores, Sistemas Distribuidos o Administración de Sistemas Informáticos.

- La creación la **aplicación web** se subdivide en dos partes:
 - El **backend**, que se relaciona con asignaturas como: Programación I y II, Lógica, Algoritmos y Estructuras de Datos, Bases de Datos, Concurrencia, Ingeniería del Software I y II, Middleware, Sistemas Distribuidos o Sistemas Orientados a Servicios.
 - El **frontend**, que se relaciona muy levemente con asignaturas como Interacción Persona-Ordenador o Diseño de Aplicaciones Web.

Analizando estos resultados, podemos observar que hay una **parte práctica de la informática a la que no se le presta demasiada atención en el grado**, y que actualmente posee una **enorme oferta de trabajo**: el desarrollo *frontend*.

Cabe destacar que, en plataformas como LinkedIn, el desarrollo *frontend* posee, en términos cuantitativos, una **mayor oferta de trabajo** que el desarrollo *backend*.

Esta situación me **decepcionó bastante**, ya que siendo **uno de mis mayores intereses** en el campo, nunca tuve la oportunidad de poder dedicarle una asignatura a este apartado práctico de la Informática. No pude seleccionar la asignatura de *Diseño de Aplicaciones Web*, única asignatura del grado con relación directa con el desarrollo *frontend*, por incompatibilidad de horarios y dado su carácter optativo.

9.3. Trabajo futuro

Ya se ha comentado en el resto del documento que el producto generado tiene una gran variedad de usos y aplicaciones, y que puede ampliarse en funcionalidades altamente útiles a nivel personal.

De este modo, alguna de las ampliaciones futuras a realizar en el servidor doméstico, siguiendo el modelo de configurabilidad, automatización y securización que se ha llevado a cabo a lo largo del proyecto, son el desarrollo de servicios como **VPN**, **FTP/SFTP**, **SMTP**, etc.

Con la investigación acerca de los usos del hardware empleado, la lista de posibilidades no hace más que aumentar y volverse cada vez más interesante. Por ello, **continuaré desarrollando servicios y utilidades con la Raspberry** con la finalidad de ampliar mis conocimientos y, cuando sea posible, aportar algo a la comunidad de la informática.

Capítulo 10

Impacto

Este capítulo expone los beneficios esperados del proyecto, así como los posibles efectos adversos que pueden surgir del mismo.

Además, se harán notar aquellas decisiones tomadas a lo largo del trabajo que tienen como base la consideración del impacto.

10.1. Personal

Este es, con diferencia, el aspecto en el que el proyecto provoca un **mayor impacto**.

Como se comentó en las conclusiones del capítulo 9, el desarrollo de este proyecto ha supuesto una **gran influencia a nivel de aprendizaje personal**. En este aspecto, al inicio no era completamente consciente del alcance que podría tener este proyecto a nivel personal, ya que este **superó con creces mis expectativas**.

Además, actualmente poseo un **servidor que me permite alojar un portafolio**, entre otras muchas cosas, con los trabajos que estoy desarrollando, de modo que tenga una buena **carta de presentación de cara al mundo laboral** y mi desarrollo profesional. Incluso puedo utilizarlo para pruebas como las realizadas con todos los elementos de la aplicación web desarrollada.

Por todo esto, solo teniendo en cuenta esta sección estaría **contento con el resultado final**.

10.2. Empresarial

A pesar de que el proyecto no esté pensado desde el inicio para el ámbito empresarial, sí que había ciertas opciones que se tuvieron en cuenta y comentaron en el resumen del trabajo, así como en el capítulo introductorio 1.

En este caso, reflexionando acerca del impacto que puede tener para una empresa: el servidor desarrollado puede que **no sea la mejor opción para utilizar alojando servicios en producción**, sobre todo si necesita atender una gran cantidad de peticiones o accesos.

Pero, sí que resulta una **alternativa ideal para un proyecto en desarrollo**, incluso para una empresa de gran tamaño que necesite un **servidor de pruebas** aislado, y todo ello **por un precio ridículo** teniendo en cuenta las funcionalidades ofrecidas.

10.3. Económico

Desde el punto de vista económico, en el capítulo 8, sección 8.4, se ha podido comprobar la **rentabilidad** del proyecto desarrollado frente a las alternativas disponibles a día de hoy.

Para una persona que simplemente necesita un servidor, puede que las **alternativas expuestas en la sección 8.3** puedan resultar más **interesantes**, evitando la responsabilidad del manejo físico, así como el tiempo de dedicación necesario para llevar a cabo el proyecto. Todo esto sin tener en cuenta el proceso de automatización ofrecido y la diferencia económica de dichas alternativas.

De cualquier modo, es un proyecto interesante de replicar o utilizar por una persona que posee interés en la materia, ya que te permite conectar muchos ámbitos y temas de la informática.

10.4. Objetivos de Desarrollo Sostenible

Respecto a los Objetivos de Desarrollo Sostenible (ODS), de la Agenda 2030, el trabajo desarrollado guarda cierta relación con los siguientes objetivos [47]:

4 : educación de calidad.

7 : energía asequible y no contaminante.

12 : producción y consumo responsables.

El **desarrollo del trabajo** podría relacionarse con el **objetivo 4**. Esto resulta al ofrecerse tanto el producto como la documentación correspondiente, de manera pública y libre de acceso y uso a toda la comunidad informática a través de GitHub.

Por último, se relaciona con los **objetivos 7 y 12**, teniendo en cuenta los estudios realizados en cuanto al **consumo del servidor**. Este podría **sustituir algunos servidores domésticos que utilizan equipamiento antiguo** con un consumo que **multiplica por 30 el consumo** monitorizado con el hardware empleado.

Bibliografía

- [1] G. Halfacree, *The Official Raspberry Pi Beginner's Guide*, 4th ed. Raspberry Pi Press, 2020.
- [2] Y. Fernández. (2018) Servidores nas: qué son, cómo funcionan y qué puedes hacer con uno. [Online]. Available: <https://www.xataka.com/basics/servidores-nas-que-como-funcionan-que-puedes-hacer-uno>
- [3] PiMyLifeUp. (2022) Raspberry pi server projects. [Online]. Available: <https://pimylifeup.com/category/projects/server/>
- [4] S. Hill. (2015) Raspberry pi web server. [Online]. Available: <https://github.com/sean-hill/raspberry-pi-web-server>
- [5] R. Pi. (2022) Pi server. [Online]. Available: <https://github.com/raspberrypi/piserver>
- [6] J. Brown. (2017) Raspi 3 web server. [Online]. Available: <https://github.com/girls-whocode/RasPi3-WebServer>
- [7] E. Mazza. (2019) Raspbian server. [Online]. Available: <https://github.com/d3cod3/raspbian-server>
- [8] S. Rodriguez. (2018) Raspberry pi web tutorial. [Online]. Available: <https://github.com/seb646/raspberry-pi-web-tutorial>
- [9] D. Arduino. (2020) Cómo convertir tu raspberry pi en nas con openmediavault. [Online]. Available: <https://descubrearduino.com/convertir-tu-raspberry-pi-en-nas-con-openmediavault>
- [10] R. García. (2022) Las mejores rivales y alternativas a la raspberry pi 4. [Online]. Available: <https://www.adslzone.net/listas/gadgets/alternativas-raspberry-pi>
- [11] M. Contreras. (2022) ¿no encuentras una raspberry pi? estas son las mejores alternativas. [Online]. Available: <https://computerhoy.com/reportajes/tecnologia/mejores-alternativas-raspberry-pi-1007335>
- [12] Axarnet. (2022) Servidores linux: ¿por qué se usan más que los de windows? [Online]. Available: <https://axarnet.es/blog/servidores-linux-por-que-se-usan-mas-que-los-de-windows>
- [13] C. Cawley. (2021) The 8 best lightweight operating systems for raspberry pi. [Online]. Available: <https://www.makeuseof.com/tag/lightweight-operating-systems-raspberry-pi>

- [14] L. Garnica. (2014) Protocolos de acceso remoto. [Online]. Available: <http://servintlalo.blogspot.com/2014/05/protocolos-de-acceso-remoto.html>
- [15] İsmail Baydan. (2018) Shell scripting languages examples, bash, sh, python, powershell, msdos, php, tcl, perl. [Online]. Available: <https://www.poftut.com/shell-scripting-languages-examples-bash-sh-python-powershell-msdos-php-tcl-perl/>
- [16] Ankush. (2022) 7 servidores web de código abierto para sitios pequeños y grandes. [Online]. Available: <https://geekflare.com/es/open-source-web-servers>
- [17] T. Radius. (2022) Nosql databases overview. [Online]. Available: <https://www.trustradius.com/nosql-databases>
- [18] H. Kaur. (2020) Top 10 open-source nosql databases in 2020. [Online]. Available: <https://www.geeksforgeeks.org/top-10-open-source-nosql-databases-in-2020>
- [19] P. V. Puigcerber. (2015) How to build a raspberry pi server for development. [Online]. Available: <https://www.toptal.com/raspberry-pi/how-to-turn-your-raspberry-pi-into-a-development-server>
- [20] G. Cassibba. (2020) Raspberry pi os lite: Install, setup and configure. [Online]. Available: <https://peppe8o.com/install-raspberry-pi-os-lite-in-your-raspberry-pi>
- [21] Emmet. (2022) Improving the security of your raspberry pi. [Online]. Available: [ImprovingtheSecurityofYourRaspberryPi](https://www.improvingthesecurityofyourraspberrypi.com/)
- [22] R. Solé. (2021) Cómo activar ssh en una raspberry pi en windows 10, linux, macos, ios y android. [Online]. Available: https://profesionalreview.com/2021/08/15/activar-ssh-raspberry-pi/#Habilitar_SSH_sin_conexion
- [23] S. Campbell. (2015) How to set up a static ip on the raspberry pi. [Online]. Available: <https://www.circuitbasics.com/how-to-set-up-a-static-ip-on-the-raspberry-pi>
- [24] A. Wiki. (2018) Ssh keys. [Online]. Available: [https://wiki.archlinux.org/title/SSH_keys_\(Espa%C3%B1ol\)](https://wiki.archlinux.org/title/SSH_keys_(Espa%C3%B1ol))
- [25] UITS. (2019) About ssh-agent and ssh-add in unix. [Online]. Available: <https://kb.iu.edu/d/aeww>
- [26] Dayz. (2016) 3 steps to take to improve the security of your raspberry pi server. [Online]. Available: <http://kamilslab.com/2016/12/18/3-steps-to-take-to-improve-the-security-of-your-raspberry-pi-server>
- [27] P. Moreno. (2021) Restrict ssh login to a specific ip or host. [Online]. Available: <https://docs.rackspace.com/support/how-to/restrict-ssh-login-to-a-specific-ip-or-host>
- [28] E. Heidi. (2019) Cómo configurar un firewall con ufw en ubuntu 18.04. [Online]. Available: <https://www.digitalocean.com/community/tutorials/como-configurar-un-firewall-con-ufw-en-ubuntu-18-04-es>
- [29] Gus. (2022) Raspberry pi port forwarding dynamic dns. [Online]. Available: <https://pimylifeup.com/raspberry-pi-port-forwarding>

- [30] S. Naseer. (2015) Modify router port-forwarding rules from terminal. [Online]. Available: <https://apple.stackexchange.com/questions/208511/modify-router-port-forwarding-rules-from-terminal>
- [31] M. UPnP. (2017) Miniupnp project homepage. [Online]. Available: <http://miniupnp.free.fr>
- [32] Virgil. (2020) Open router's ports dynamically with upnp. [Online]. Available: <https://paranoiaque.fr/en/2020/05/01/open-router-ports-upnp>
- [33] J. Wachira. (2021) How to show all active ssh connections in linux. [Online]. Available: <https://www.maketecheasier.com/show-active-ssh-connections-linux>
- [34] Schkn. (2022) How to find last login on linux. [Online]. Available: <https://devconnected.com/how-to-find-last-login-on-linux>
- [35] C. Cawley. (2019) How to host your own website on a raspberry pi. [Online]. Available: <https://www.makeuseof.com/tag/host-website-raspberry-pi/>
- [36] PiMyLifeUp. (2022) Build your own raspberry pi web server. [Online]. Available: <https://pimylifeup.com/raspberry-pi-web-server>
- [37] Wikipedia. (2016) Let's encrypt. [Online]. Available: https://es.wikipedia.org/wiki/Let's_Encrypt
- [38] S. B. Rodríguez. (2019) Let's encrypt con nginx en raspberry pi (raspbian). [Online]. Available: <https://blog.tiraquelibras.com/?p=535>
- [39] Gus. (2022) Raspberry pi ssl certificates using let's encrypt. [Online]. Available: <https://pimylifeup.com/raspberry-pi-ssl-lets-encrypt>
- [40] WikiBooks. (2022) Bash shell scripting/whiptail. [Online]. Available: https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail
- [41] Atareao. (2020) Whiptail, dialogos para el terminal. [Online]. Available: <https://atareao.es/tutorial/dialogos-para-scripts/whiptail-dialogos-para-el-terminal>
- [42] Cepterra. (2021) Linode, ¿qué es? [Online]. Available: <https://www.capterra.es/software/210618/linode>
- [43] G2. (2022) Linode reviews product details. [Online]. Available: <https://www.g2.com/products/linode/reviews>
- [44] —. (2022) Top 10 linode alternatives competitors. [Online]. Available: <https://www.g2.com/products/linode/competitors/alternatives>
- [45] Linode. (2022) Linode web. [Online]. Available: <https://www.linode.com>
- [46] —. (2022) Linode pricing. [Online]. Available: <https://www.linode.com/pricing>
- [47] ONU. (2022) Objetivos de desarrollo sostenible. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible>

Anexos

Cliente RDP

En este anexo se comentará brevemente un uso secundario del hardware utilizado para el desarrollo del proyecto. Esto, consiste en utilizar la Raspberry Pi como cliente RDP de los escritorios virtuales de la UPM.

Este apartado más sencillo se incluye en el anexo, ya que, en su mayoría, únicamente trató de realizar ciertas pruebas para comprobar el funcionamiento de los mencionados escritorios virtuales en el hardware utilizado para realizar el proyecto.

Para ello, simplemente se desarrolla un script que instale el software del cliente RDP con un acceso preconfigurado a la plataforma de los escritorios virtuales de la UPM.

Con las pruebas realizadas, se ha podido comprobar que se puede hacer uso del servicio ofrecido con el hardware de la Raspberry Pi 3 Model B 1GB, pero, no ofreciendo los mejores resultados. Esto sucede ya que, a pesar de ejecutar el procesamiento en el escritorio remoto, el uso gráfico del navegador es relativamente pesado teniendo en cuenta el hardware empleado.

Por otro lado, de cara a la fluidez en el uso, la experiencia de usuario obtenida con la Raspberry Pi 4 Model B 8GB fue inigualable en comparación con el de la Raspberry Pi 3 comentada en el párrafo anterior.

De cualquier modo, la opción comprobada con el segundo caso resulta bastante útil teniendo en cuenta los resultados obtenidos dado el presupuesto empleado en el hardware.

Podría realizarse una investigación y estudio más en profundidad para este apartado, y resultaría bastante interesante de cara a alumnos que posean escasos recursos económicos, como se comentó en el capítulo 3.6. De este modo, se podría aprovechar enormemente la capacidad de cómputo ofrecida por la universidad a los alumnos.