UNIVERSIDAD DE CASTILLA-LA MANCHA

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

Protocolo de mensajería ligero inspirado en MQTT para redes ESP-NOW

Rubén Gómez Villegas

Mes, 2024

UNIVERSIDAD DE CASTILLA-LA MANCHA

TRABAJO FIN DE GRADO

Departamento de Tecnologías y Sistemas de Información

Tecnología Específica de Sistemas de Información

Protocolo de mensajería ligero inspirado en MQTT para redes ESP-NOW

Autor: Rubén Gómez Villegas

Tutor Académico: Rubén Cantarero Navarro

Cotutor Académico: Ana Rubio Ruiz

Mes , 2024

*D*edicado a mi familia y a todos
aquellos ...

Declaración de Autoría

Yo, Rubén Gómez Villegas con DNI 02318379W declaro que soy el único autor del trabajo fin de grado titulado “Protocolo de mensajería ligero inspirado en MQTT para redes ESP-NOW” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Talavera de la Reina, a…..

Fdo: ……

Resumen

Esta plantilla puede modificarse para adaptarse a las particularidades de cada Proyecto, tanto en contenido como en formato, siempre y cuando se respete las directrices básicas indicadas en la guía de estilo y formato para la elaboración de TFG del Grado en Ingeniería Informática de la Facultad de Ciencias Sociales y Tecnologías de la Información de Talavera de la Reina.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Agradecimientos

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Índice general

Índice de figuras

Índice de Tablas

Índice de Listados

Acrónimos

ACK Acknowledgement, Acuse de recibo

AIoT Artificial Intelligence of Things, Inteligencia Artificial de las Cosas

AMQP Advanced Message Queuing Protocol, Protocolo Avanzado de Colas de Mensajes

ARPANET Advanced Research Projects Agency Network, Red de la Agencia de Proyectos de Investigación Avanzada

CBCF Congressional Black Caucus Foundation, Fundación del Caucus Negro del Congreso de los Estados Unidos

CoAP Constrained Application Protocol, Protocolo de Aplicación Restringida

DDS Data Distribution Service, Servicio de Distribución de Datos

DTLS Data Transport Layer Security, Protocolo de Seguridad de la Capa de Transporte

HTML HyperText Markup Language, Lenguaje de Marcado de Hipertexto

HTTP HyperText Transfer Protocol, Protocolo de Transferencia de Hipertexto

HTTPS HyperText Transfer Protocol Secure, Protocolo Seguro de Transferencia de Hipertexto

IoT Internet of Things, Internet de las Cosas

IP Internet Protocol, Protocolo de Internet

ITU International Telecommunication Union, Unión Internacional de Telecomunicaciones

MIB Management Information Base, Base de Información Gestionada

MQTT Message Queue Telemetry Transport, Transporte de Telemetría en Colas de Mensajes

OASIS Organization for the Advancement of Structured Information Standards, Organización para el Avance de Estándares de Información Estructurada

QoS Quality of Service, Calidad de Servicio

REST Representational State Transfer, Transferencia de Estado Representacional

RFID Radio Frequency Identification, Identificación por Radiofrecuencia

SNMP Simple Network Management Protocol, Protocolo Simple de Gestión de Redes

SSL Secure Sockets Layer, Capa de Puertos Seguros

TCP Transmission Control Protocol, Protocolo de Control de Transmisión

TLS Transport Layer Security, Seguridad de la Capa de Transporte

TFG Trabajo Final de Grado

UDP User Datagram Protocol, Protocolo de Datagramas de Usuario

VoIP Voice over Internet Protocol, Voz sobre Protocolo de Internet

XML Extensible Markup Language, Lenguaje de Marcado Extensible

XMPP Extensible Messaging and Presence Protocol, Protocolo Extensible de Mensajería y Comunicación de Presencia

Capítulo 1

Introducción

El capítulo de Introducción debe describir el problema que se pretende resolver con el desarrollo del Trabajo Fin de Grado (TFG). Debe dar respuesta al qué sin especificar cómo se va a realizar, para lo cual se usarán el resto de los capítulos del documento. El lector de este documento debe tener claro el alcance del proyecto habiendo leído únicamente el capítulo de Introducción.

1.1. Motivación

Esta sección aborda la motivación del trabajo. Se trata de señalar la necesidad que lo origina, su actualidad y pertinencia. Puede incluir también un estado de la cuestión (o estado del arte) en la que se revisen estudios o desarrollos previos y en qué medida sirven de base al trabajo que se presenta.

En este capítulo debería introducirse el contexto disciplinar y tecnológico en el que se desarrolla el trabajo de modo que pueda entenderse con facilidad el ámbito y alcance del TFG. Puesto que un TFG no tiene que ser necesariamente un trabajo con aportes novedosos u originales, solo es necesario la inclusión de estado del arte cuando este contribuya a aclarar aspectos clave del TFG o se desee justificar la originalidad del trabajo realizado. Si la sección estado del arte es muy extensa, considera la opción de introducirla como un capítulo independiente.

1.2. Redacción de la Memoria

Durante la realización de la memoria del TFG es importante tener presente respetar la guía de estilo de la institución. Por tanto, el empleo de plantillas para un sistema de procesamiento de textos (por ejemplo, Word o LaTeX) puede requerir su adaptación cuando la plantilla mencionada no haya sido suministrada en la institución a la que se dirige el trabajo.

Para redactar un trabajo académico de modo efectivo se deben tener presentes una serie de normas que

ayuden a conseguir un resultado final que sea claro y de fácil lectura.

A la hora de redactar el texto se debe poner especial atención en no cometer plagio y respetar los derechos de propiedad intelectual. En particular merece gran atención la inclusión de gráficos e imágenes procedentes de Internet que no sean de elaboración propia. En este sentido se recomienda consultar el manual de la Universidad de Cantabria [1] en el que se explica de modo conciso cómo incluir imágenes en un trabajo académico.

1.3. Estructura del Documento

Este capítulo suele incluir una sección que indica la estructura (capítulos y anexos) del documento y el contenido de cada una de las partes en que se divide. Por tanto, las secciones que suelen acompañar este capítulo son:

- Motivación. Responde a la pregunta sobre la necesidad o pertinencia del trabajo.
- Objetivo. Determina de modo claro el propósito del trabajo descrito que puede desglosarse en subobjetivos cuando el objetivo principal se puede descomponer en módulos o componentes. Es muy importante definir el objetivo de modo apropiado. El Capítulo Objetivos de esta guía explica cómo definir el objetivo.
- Antecedentes o Contexto disciplinar/tecnológico. También puede denominarse Estado del Arte cuando se trata de comentar trabajos relacionados que han abordado la cuestión u objetivo que se plantea.
- Estructura del documento. Resumen de los capítulos y anexos que integran el documento.

1.4. Abreviaturas y Acrónimos

Un TFG que utiliza muchas abreviaturas y acrónimos puede añadir esta sección dónde se muestra el conjunto de abreviaturas y acrónimos y su significado.

1.5. Objetivos

Para hacer un planteamiento apropiado de los objetivos se recomienda utilizar la Guía para la elaboración de propuestas de TFG en la que se explica cómo definir correctamente los objetivos de un TFG.

1.5.1. Objetivo General

Introduce y motiva la problemática (i.e ¿cuál es el problema que se plantea y por qué es interesante su resolución?).

[1] Guía de Imágenes: https://web.unican.es/buc/Documents/Formacion/guia_imagenes.pdf

Debe concretar y exponer detalladamente el problema a resolver, el entorno de trabajo, la situación y qué se pretende obtener. También puede contemplar las limitaciones y condicionantes a considerar para la resolución del problema (lenguaje de construcción, equipo físico, equipo lógico de base o de apoyo, etc.). Si se considera necesario, esta sección puede titularse Objetivos del TFG e hipótesis de trabajo. En este caso, se añadirán las hipótesis de trabajo que el/la estudiante pretende demostrar con su TFG.

Una de las tareas más complicadas al proponer un TFG es plantear su Objetivo. La dificultad deriva de la falta de consenso respecto de lo que se entiende por objetivo en un trabajo de esta naturaleza.

En primer lugar, se debe distinguir entre dos tipos de objetivo:

- La finalidad específica del TFG que se plantea para resolver una problemática concreta aplicando los métodos y herramientas adquiridos durante la formación académica. Por ejemplo, Desarrollo de una aplicación software para gestionar reservas hoteleras on-line.
- El propósito académico que la realización de un TFG tiene en la formación de un graduado. Por ejemplo, la adquisición de competencias específicas de la intensificación cursada.

En el ámbito de la memoria del TFG se tiene que definir el primer tipo de objetivo, mientras que el segundo tipo es el que se añade en el Capítulo de Conclusiones y que justifica las competencias específicas de la intensificación alcanzadas y/o reforzadas con la realización del trabajo.

La categoría del objetivo planteado justifica modificaciones en la organización genérica de la memoria del trabajo. Así en el caso de estudios y validación de hipótesis el apartado de resultados y conclusiones debería incluir los resultados de experimentación y los comentarios de cómo dichos resultados validan o refutan la hipótesis planteada.

1.5.2. Objetivos Específicos

Generalmente, el objetivo general puede ser descompuesto en varios objetivos más específicos que se pretenden alcanzar. En esta sección se enumeran y describen cada uno de ellos.

Junto con la definición de estos objetivos se puede especificar los requisitos que debe satisfacer la solución aportada. Estos requisitos especifican características que debe poseer la solución y restricciones que acotan su alcance. En el caso de un trabajo cuyo objetivo es el desarrollo de un artefacto los requisitos pueden ser funcionales y no funcionales.

Al redactar el objetivo de un TFG se debe evitar confundir los medios con el fin. Así es habitual encontrarse con objetivos definidos en términos de las acciones (verbos) o tareas que será preciso realizar para llegar al verdadero objetivo. Sin embargo, a la hora de planificar el desarrollo del trabajo si es apropiado descomponer todo el trabajo en hitos y estos en tareas para facilitar dicha planificación.

La categoría del objetivo planteado justifica modificaciones en la organización genérica de la memoria

del trabajo. Así en el caso de estudios y validación de hipótesis el apartado de resultados y conclusiones debería incluir los resultados de experimentación y los comentarios de cómo dichos resultados validan o refutan la hipótesis planteada.

Capítulo 2

Estado del arte

2.1. Internet de las Cosas

Dado que una de las principales motivaciones de este Trabajo de Fin de Grado es el aporte que puede ofrecer a la facilidad de implementación del Internet de las Cosas, es importante conocer el joven concepto que nos rodea en la actualidad.

El siglo XX dio lugar al desarrollo de una cantidad de inventos que permitieron una revolución y un avance ágil en la sociedad, inventos nacidos de ideas creativas de sus desarrolladores y que en la actualidad son indispensables debido al frecuente uso y la manera en la que facilita la vida humana. Entre estos inventos se encuentra una de las herramientas más importantes y que mejoró la comunicación, el Internet. Nacido en 1969, ha permitido enlazar a personas con personas y tener acceso a información, pero hoy en día, no solo las personas están conectadas a Internet, también millones de objetos cuyas funciones son dependientes de la red.

Internet de las Cosas o *Internet of Things* es un término cuyo origen está disputado. El primer uso de este término data del discurso que realizó en septiembre de 1985 Peter T. Lewis, cofundador de la primera compañía de telefonía móvil de Estados Unidos, Cellular One. En tal discurso realizado en la Conferencia Legislativa Anual de la CBCF en Washington D.C., Lewis comentó de manera acertada “Predigo que no sólo los humanos, sino también las máquinas y otras cosas se comunicarán interactivamente a través de Internet.” **(mencionar al podcast)**. Por otro lado, el mismo término fue acuñado en 1999 cuando Kevin Ashton, directivo de Procter & Gamble, tuvo la iniciativa de investigar el uso de etiquetas de identificación por radiofrecuencia (RFID) y otros sensores en los productos de la cadena de suministro, y para presentar el proyecto tuvo que idear un título llamativo para la presentación. Esta presentación le permitió encontrar financiación además de cofundar y dirigir el laboratorio Auto-ID Center del Instituto de Tecnología de Massachusetts, en el cuál construyó la base del Internet de las cosas.

Es esencial tener una clara comprensión del significado del Internet de las Cosas, en mayor parte por la confusión inherente al término en sí y por las aplicaciones cotidianas de esta tecnología. A primera vista, "Internet de las Cosas" podría dar la impresión de ser un término moderno para referirse a "conectar a Internet algo para controlarlo", una definición bastante simple para alguien que por ejemplo simplemente controla las luces de su hogar desde su teléfono móvil. Es tanta la confusión que no existe una definición formal única, debido a que hay sistemas que implementan IoT y no cumplen todas las definiciones, por lo tanto, no se ha llegado a un consenso para formalizar una definición única, y a la hora de intentar conocer su significado se encuentran una disparidad de definiciones. En 2012, la ITU (Unión Internacional de Telecomunicaciones) recomendó una aclaración del términio, definiendo el Internet de las Cosas como una infraestructura global que permite ofrecer servicios avanzados a todo tipo de aplicaciones conectando objetos entre sí e interoperando tecnologías de la información y comunicación, aprovechando capacidades de identificación, obtención de datos, procesamiento y comunicación y cumpliendo con requisitos de seguridad y privacidad. Sin embargo, por lo general, podemos entender que el Internet de las Cosas trata de dotar de capacidades de comunicación además de procesamiento, captura y/o análisis de datos a distintos tipos de entes, como dispositivos físicos, objetos, edificaciones, terrenos, sistemas, hardware, software, e incluso contextos y situaciones, ya sea añadiéndoles dispositivos o integrando las capacidades en los propios objetos. Estos entes pueden estar compuestos de sensores, que recopilan datos, o actuadores, que controlan otros objetos, y a través de redes, privadas o públicas, pueden intercambiar información con otros dispositivos, recopilar la información en un mismo dispositivo y transferir órdenes, además que existe la posibilidad de formar una agrupación de dispositivos para identificarlos como un único sistema que trata los datos. Estas redes de conexión de dispositivos IoT pueden utilizar la información transferida para poder automatizar sus comportamientos, pero el nodo final de la conexión suelen ser las personas, ya que utilizan los objetos como fuentes de información, para a su vez utilizar los datos recibidos como descubrimiento de oportunidades de negocio y nuevos servicios, monitorización y evaluación del estado y el comportamiento de los objetos y el entorno, y la toma de decisiones sobre los propios objetos manipulándolos remotamente y programándolos. Dependiendo del entorno en que se implemente, estos sistemas deben cumplir requisitos de seguridad y privacidad, evitando manipulaciones y accesos indebidos o incluso daños en los propios objetos y en el entorno que los rodea.

Un ejemplo concreto de la aplicación de Internet de las Cosas es un sistema de riego por aspersión inteligente. Como se puede comprobar en la figura **todo: insertar figura**, existe un jardín dividido en zonas, cada una con un dispositivo compuesto por sensores de temperatura y humedad y actuadores que activan y desactivan los aspersores que riegan la vegetación. Estos dispositivos están conectados y se comunican con un único gateway o puerta de enlace, teniendo la capacidad de recibir información de los dispositivos y mandar órdenes a estos. El gateway está comunicado a través de Internet con un servidor compuesto por una base de datos, donde se almacenan los datos históricos y registros, y un servicio que le permite ser controlado desde otro dispositivo conectado a Internet en cualquier sitio y momento. Este último dispositi-

vo, denominado cliente, puede ser un teléfono móvil o un ordenador, y puede utilizarse para ver el estado del jardín y controlar los dispositivos manualmente desde una interfaz. La siguiente arquitectura, abstraída completamente de las limitaciones y los problemas que puede ofrecer su implementación, podrá dar lugar a dos casos de uso:

- El usuario quiere activar los aspersores. El usuario, desde la interfaz el dispositivo cliente como puede ser un botón, enviará la orden de activar los aspersores al servicio, que a su vez se lo enviará al gateway. El gateway enviará una orden compatible a los dispositivos instalados en el jardín, haciendo que los aspersores comiencen a funcionar. Esta secuencia de ejecución será parecida si el usuario desea desactivar los aspersores.
- Los aspersores funcionan cuando la temperatura es muy elevada y la humedad es baja. El usuario previamente, desde la interfaz del cliente, ha establecido que los aspersores funcionen de manera automática cuando, por ejemplo, el ambiente supere una temperatura de 42ºC y la humedad sea considerada baja. Estos parámetros los recibe el servidor y se los pasa al gateway, el cual los recordará. A partir de ese momento, el gateway irá recibiendo de los sensores de cada zona lecturas de temperatura y humedad, y las comparará con los parámetros establecidos. En el caso de que se superen la temperatura y humedad, se activarán los aspersores de la zona. Una vez activados, se mantendrá la lectura de temperatura y humedad, y cuando las lecturas sean inferiores, se desactivarán los aspersores. Además, con cada activación de los aspersores, el gateway notificará al servicio, que a su vez notifica al usuario, especificando la zona activada.

/*TODO: Dibujar una figura con 3 zonas: - Jardin con sensores y aspersores - Gateway, al que se conecta el jardin, y conectada a servicio internet y a bbdd - Movil del usuario, app conectado a servicio internet* /

Cabe recalcar que el mencionado ejemplo puede volverse más complejo, por ejemplo añadiendo comprobaciones de previsión meteorológica en los próximos días, sensores de luz, control del caudal del agua, pero igualmente cumple con los requisitos para ser un sistema que implementa el Internet de las Cosas, ya que dota a un jardín de capacidad de comunicación, captura y análisis de datos, y se comunica con otros dispositivos y con personas para informar del estado y recibir órdenes.

/*TODO: hablar ventajas, desventajas y cosas a tener en cuenta (bateria, limitaciones de computación, etc)*/ /*TODO: hablar de la importancia y el valor de los datos*/ /*TODO: hablar de usos de IoT*/

2.1.1. Primeros ejemplos de IoT

El primer dispositivo IoT que se conoce fue una máquina de Coca-Cola conectada a ARPANET a principios de la década de los 80 en la Universidad Carnegie Mellon de Pittsburgh, Pensilvania. Desde los años 70, el departamento de Ciencias de la Computación se encargaba de mantener la máquina de refrescos, que era cargada por alumnos en horarios erráticos y cuyos precios eran los más baratos (unos 10 centavos menos)

no solo del campus sino también de la zona de Pittsburgh, lo que provocó una gran popularidad en ventas. A mediados de la misma década, debido a la expansión del departamento, se tuvieron que desplazar las oficinas lejos de la máquina que se encontraba en la tercera planta del edificio, lo que provocaba molestias a los programadores al ir en busca de su dosis de cafeína, ya que al bajar a la máquina y gastar el dinero que tanto les había costado conseguir, se encontraban con que o bien estaba vacía o que la Coca-Cola conseguida estaba recién cargada y, por lo tanto, desagradablemente caliente. Cansados de este problema, un grupo de personas se reunió para idear una solución. Instalaron microinterruptores en la máquina de Coca-Cola para poder detectar cuántas botellas había en cada una de las 6 columnas disponibles. Estos interruptores estaban conectados, a través de una interfaz de red instalada en la máquina, al ordenador principal del departamento, el PDP-10 (denominado CMUA). Para este ordenador, diseñaron un programa, que apoyándose en un servidor programado para registrar la hora de la última transacción de cada columna, mostraba uno de los siguientes datos por cada columna de la máquina de Coca-Cola:

- "EMPTY", si la columna estaba vacía.
- El tiempo que llevaba la nueva botella cargada, por ejemplo, "1h 3m".
- "COLD", si la columna contenía una botella fría, es decir, que llevaba 3 horas cargada.

Para completar la idea, permitieron el acceso a esta información desde fuera del CMUA. Modificaron el servidor Finger ya existente en el CMUA, el cual era utilizado para obtener información sobre los usuarios conectados al sistema, para añadirle el usuario "coke", para que cada vez que se hiciese la petición a Finger con dicho usuario se ejecutase el programa de estado de la máquina de Coca-Cola. Debido a que las peticiones Finger formaban parte de los protocolos estándares de ARPANET, se podía descubrir el estado de la máquina de Coca-Cola tanto desde cualquier ordenador conectado en la Universidad Carnegie Mellon como desde cualquier parte del mundo conectada a la red, simplemente utilizando el comando `finger ↪ coke@cmua`.

Esta innovación se mantuvo en pie durante más de una década, pasando por una reescritura del código cuando se tuvo que migrar del PDP-10 al más moderno Unix Vaxen a principios de los 80, hasta que Coca-Cola descontinuó las botellas antiguas y las nuevas no eran compatibles con la máquina de refrescos existente, lo que provocó que fuera sustituida por completo.

La máquina de Coca-Cola de los 80 inspiró a que en 1992 un estudiante y un miembro del departamento de Ciencias de la Computación de la misma universidad instalasen a la nueva maquina y la de M&M que se encontraba cercana aisladores ópticos caseros para identificar la presencia del producto, y las conectasen a Internet para poder consultar el estado ejecutando un programa llamado "jf" ("junk food", comida basura).

Incluir imágenes: - Xerox Alto, ordenador fabricado en 1973, usado en la CMU para ver el estado de la maquina

- Máquina de cocacola https://engineercom.wpenginepowered.com/wp-content/uploads/2016/02/IIot1.png

- PDP-10

Posteriormente apareció otro notable objeto conectado a un Internet más similar al actual pero con menos personas (rondando las 3 millones), y fue una tostadora en 1990. El origen de este objeto ocurrió en la edición de 1989 de Interop, una conferencia anual que reunía a profesionales de las tecnologías de la información, ingenieros de redes, desarrolladores y proveedores de tecnología, en la que se podía poner a prueba la interoperabilidad de dispositivos informáticos, electrónicos y productos relacionados con Internet. En dicha edición, el organizador Dan Lynch retó a John Romkey a poner un dispositivo en línea y presentarlo en la siguiente edición. Romkey aceptó, ya que no desconocía la tecnología de la época que le permitiría conseguir dicho reto. John Romkey fue cocreador de la primera pila de protocolos de Internet TCP/IP, PC/IP del Instituto Tecnológico de Massachusetts (MIT) para MS-DOS en el IBM PC, además de proveer pilas TCP/IP a través de la empresa FTP Software, la cual fundó.

Por aquella época, Romkey trabajaba en el Protocolo Simple de Gestión de Redes (SNMP), junto a Karl Auerbach y David Bridgham, de Epilogue Technology, y Simon Hackett, de Internode. SNMP es un protocolo sencillo que permite a los usuarios leer y escribir variables en un agente remoto y que hasta ese momento se había utilizado únicamente para gestionar dispositivos como routers, para inspeccionar recuentos de paquetes y tablas de enrutamiento, borrar contadores y habilitar o deshabilitar interfaces de red. John Romkey quería aprovechar la oportunidad para demostrar que, dado que SNMP podía controlar dispositivos físicos, también podría ampliar su alcance para gestionar salas completas que albergaran equipos de red y ordenadores, y consideró que controlar una tostadora sería un buen punto de partida. La tostadora, al ser un electrodoméstico sencillo, si su implementación del protocolo funcionaba y podía demostrarlo, el público podría entender que la automatización es posible y se podría llevar a niveles industriales. También, la implementación de Internet en la tostadora podría plantear problemas de seguridad a tener en cuenta, ya que habría que gestionar quién puede tostar el pan y quién puede consultar los hábitos de uso del electrodoméstico, además de tener el factor ridículo de poner una tostadora en Internet.

El primer paso que tomaron fue buscar una tostadora fácil de controlar físicamente, ya que la mayoría requería de bajar una palanca para también bajar el pan, algo fácil e intuitivo para el ser humano pero difícil de controlar por ordenadores sin hacer uso de la robótica. Con la ayuda de Two Guys and a Vax, una empresa amiga de Epilogue, descubrieron una tostadora Sunbeam Radiant Control, la cual no necesitaba ninguna palanca. Si la tostadora tiene energía y un pan insertado, el pan se baja automáticamente y se empieza a tostar, y finaliza cuando el pan se ha tostado hasta el punto deseado o se corta la corriente. Ese modelo era ideal para ser controlado por ordenador, ya que era más fácil de controlar la alimentación de energía que llegaba al dispositivo que una palanca mecánica de una tostadora.
Con la tostadora ideal elegida, el siguiente paso fue construir hardware capaz de controlar la energía. A principios de la década de los 90, la sociedad se encontraba en un entorno pre-web y no existían las placas de desarrollo actuales como Arduino, ni Wi-Fi, y mucho menos dispositivos conectables de manera inalám-

brica a través de Wi-Fi, por lo que este paso era obligatorio. El primer intento de John Romkey fue crear un relé simple controlable a través del puerto paralelo de un ordenador portátil, el cual funcionó brevemente, pero terminó tostando el puerto del portátil debido a que el relé consumía más energía de la que el puerto podía suministrar. El segundo intento, mucho más exitoso y resultando en una manera fácil de controlar la tostadora mediante software, fue usar un interruptor de menor consumo conectado directamente al puerto paralelo del portátil para controlar un relé más grande que, a su vez, controlaba la alimentación de la tostadora.

El último paso fue la implementación del software para controlar la tostadora por Internet, es decir, programar la parte SNMP del proyecto que indicaba cómo de oscuro tostar el pan y el tipo de pan insertado. El lenguaje de SNMP permite especificar que controlar en un agente remoto, ya que cada dispositivo gestionado puede tener su propia base de información gestionada (MIB por sus siglas en inglés). Dentro del MIB se pueden especificar objetos, y por cada uno el tipo de objeto, el acceso permitido, el estado y la descripción, y en el caso de la tostadora, se habían especificado en el código como objetos el fabricante, modelo, los controles de la tostadora para subir y bajar la rebanada (o iniciar y terminar el tostado), cómo de hecha se deseaba la tostada y el tipo de pan. Como ejemplo, la siguiente porción de código controlaba el inicio del tostado:

```
toasterControl OBJECT-TYPE
    SYNTAX INTEGER {
        up(1),
        down(2)
    }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION"
        This variable controls the current state of the toaster.
        To begin toasting, set it to down(2).
        To abort toasting (maybe in the event of an emergency), set it to up(2)".
    :: = {toaster 3}
```

(codigo tomado del articulo)

Gracias a los objetos especificados, se podía identificar qué tostadora se estaba controlando, además de establecer el inicio, el tipo de pan y el nivel de tostado final del pan al modificar los objetos, todo a través del agente SNMP conectado al hardware que controlaba la alimentación de la tostadora, permitiendo un control remoto del electrodoméstico. Por ejemplo, si un usuario a través de SNMP modificaba la variable adecuada para indicar que quiere iniciar la tostadora, la tostadora recibía alimentación e iniciaba el proceso de tostado.

La tostadora de Internet fue demostrada finalmente en el expositor de Epilogue en la convención de Interop 1990, mostrando un uso alternativo de SNMP, fuera de la gestión de routers, y aprovechando la ocasión para buscar licenciar su implementación SNMP. Como curiosidad, no se podía realizar la demostración de forma legal, ya que tostar pan se consideraba preparación de alimentos, algo permitido solo al sindicato, y para evitar este problema, Epilogue Technology negoció con el sindicato y acordaron tostar repetidamente la misma rebanada de pan para que el público no se la comiera. Además, en la posterior edición de 1991, el hardware de la tostadora evolucionó para añadir un pequeño brazo robótico de LEGO, también controlado por Internet, capaz de tomar una rebanada de pan e insertarla en la tostadora, consiguiendo automatizar el sistema por completo. En la actualidad, John Romkey aún posee esta tostadora de Internet, la cual sigue activa de vez en cuando en Portland, Oregón.

- Incluir imágenes tostadora (sacarla de url)

Como último ejemplo, también de la misma época de Internet, está la cafetera del laboratorio de informática de la Universidad de Cambridge. Desde los inicios de la computación, la Universidad de Cambridge ha jugado un papel fundamental, contando incluso los primeros ordenadores útiles de uso general, la EDSAC y la EDSAC 2. Además, ha albergado pioneros en ciencias de la computación, incluyendo al creador de las subrutinas, desarrolladores de los primeros procesadores ARM y de varios lenguajes de programación. Esto da lugar a que, por el entorno y por el talento que se encontraba en el campus, era natural que surgieran proyectos de informática. Uno de estos fue la máquina de café conectada a Internet, cuya primera versión fue creada por Quentin Stafford-Fraser y Paul Jardetzky. En 1991, mientras se implantaba la World Wide Web en el Centro Europeo de Investigación Nuclear (CERN), la cual no tenía la popularidad ni la presencia actual, los autores del objeto eran investigadores en el grupo de sistemas del campus, y se encontraban junto a entre 10 y 13 compañeros trabajando la mayoría en una sala del antiguo laboratorio de informática conocida como la Sala Trojan, en cambio otros estaban dispersos por la universidad. Debido a que eran académicos con recursos limitados, solo disponían de una única máquina de café de goteo-filtro para todos, que se encontraba en el pasillo junto a la salida de la Sala Trojan. Aunque la cafeína era esencial para mantener la investigación en informática (de hecho, por la necesidad las jarras de café no duraban mucho), el café de esta máquina era poco inspirador y considerado horrible, pero tolerable si se podía conseguir recién hecho, algo que pocas personas tenían suerte de disfrutar, como Stafford-Fraser que se sentaba al lado de la máquina, mientras otras personas que trabajaban en plantas distintas tenían que recorrer todo el edificio solo para encontrar los posos en el fondo de la jarra, café de muy mala calidad o descubrir que les tocaba rellenar la máquina. Este problema causó cierta angustia en el grupo, por lo que decidieron solucionarlo mediante la tecnología.

En el laboratorio se disponía de cámaras de video económicas excedentes resultado de un proyecto, así como de varios racks de ordenadores simples utilizados para pruebas de redes. Así, en noviembre de 1991, en un lluvioso día, prepararon el sistema XCoffee. Primero fijaron una de las cámaras a un soporte de labo-

ratorio, instalándola para que apuntara a la jarra de la cafetera ubicada en el pasillo. Los cables conectados a la cámara se extendieron por debajo del suelo hasta una tarjeta capturadora de fotogramas de vídeo, instalada en un ordenador Acorn Archimedes situado en el rack de la Sala Trojan. El software de este sistema constaba de dos partes: la parte servidor, escrita por Paul Jardetzky, que capturaba imágenes de la cafetera cada pocos segundos y en diversas resoluciones; y la parte cliente, escrita por Quentin Stafford-Fraser, ejecutable por cualquier usuario conectado a través de Ethernet en la misma red local que el servidor, y que mostraba una imagen obtenida del servidor del tamaño de un icono en una ventana en la esquina de la pantalla. Este sistema operaba utilizando el protocolo X Window System, donde XCoffee empleaba llamadas a procedimientos remotos en modo de transferencia asíncrona a través de Ethernet para obtener las imágenes de la cafetera que se mostraban al usuario, con una resolución de 768x576 píxeles y 8 bits de profundidad, en escala de grises y actualizada tres veces por minuto, una configuración no molesta ya que la jarra se llenaba despacio y para distinguir el tono del café bastaba con comprobar como de gris estaba la imagen.

Teniendo este sistema, los investigadores del departamento, desde la comodidad de sus oficinas, podían comprobar la cafetera, su luz de encendido, el nivel y la oscuridad del café, y si consideraban oportuno bajaban a por su dosis de cafeína, solucionando finalmente el esfuerzo físico de comprobar la cafetera y evitando la angustia emocional de llegar y encontrarla vacía. Esta solución solo ayudaba a quienes estaban conectados a la red informática interna del laboratorio de Cambridge, mientras que a los que no y no tenían la posibilidad de ejecutar el software de la cámara de la cafetera, como el Dr. Martyn Johnson, el problema les seguía afectando. Influenciado por esta limitación y por la evolución de las páginas web, que pasaron en marzo de 1993 de ser páginas simples de texto que solo se les podía cambiar los colores y la fuente, a adquirir la capacidad de incluir imágenes en el código HTML y mostrarlas, junto a Daniel Gordon, ambos crearon la primera webcam de la historia.

El 22 de noviembre de 1993, Gordon y Johnson, tras estudiar las capacidades de la web y analizar el código del servidor, y considerando que la tarea sería relativamente sencilla, tomaron el sistema de XCoffee y lo desplegaron a la World Wide Web. Johson desarrolló un script, cuya primera versión únicamente ocupaba unas 12 líneas de código, que, junto a algunas modificaciones del software original, hacía que, tras recibir una solicitud HTTP, el servidor web desplegado solicitase el fotograma más reciente capturado por la tarjeta capturadora, y generara y sirviese la página que mostraba esta imagen, en otras palabras, una página con una imagen distinta cada vez. Esta implementación sirvió como prueba para el desarrollo de páginas web dinámicas que, en lugar de contener imágenes estáticas, presentaran imágenes que cambiasen constantemente. Además, ampliaron el acceso para poder acceder no solo desde la red local, sino también desde Internet.

La posibilidad de consultar la cámara directamente desde el navegador web, sin necesidad de ejecutar software ni utilizar protocolos de red especiales, tuvo como efecto secundario, que, aunque no fuese muy

útil para personas a varios kilómetros de distancia de la Sala Trojan, todo el mundo podía ver el estado de la cafetera, así ocurrió que cientos de miles de personas han mirado la cafetera, ganando notoriedad internacional y convirtiéndola en la cafetera más famosa del mundo. Esta fama llevó a la instalación de una lámpara apuntando a la jarra para permitir consultar su estado incluso por la noche.

Tras una década en funcionamiento y habiendo pasado por varias cámaras y cafeteras, el grupo de investigación de encontraba frente a un nuevo desafío. Con el software volviéndose cada vez más difícil de mantener y la necesidad de trasladar el laboratorio a las nuevas instalaciones en el Edificio William Gates, se planteaba la urgencia de pasar página. Este software de investigación demostraba no tener la calidad óptima, como suele ocurrir con otro software del mismo estilo, y a su vez la imposibilidad de migrar el sistema a las nuevas máquinas del laboratorio complicaba aún más la situación. Finalmente, tomaron una decisión que tuvo gran cobertura mediática y no fue apoyada en las protestas nostálgicas de los aficionados a las webcams de todo el mundo: apagar la cámara y la cafetera. El 22 de agosto de 2001, a las 09:54 UTC, se observó en Internet la última imagen capturada por las cámaras que apuntaban a la máquina de café de la Sala Trojan, que muestra los dedos de Daniel Gordon, Martyn Johnson y Quentin Stafford-Fraser pulsando el interruptor de apagado del Acorn Archimedes que había capturado las imágenes durante estos años. En el mismo mes, una de las cuatro o cinco últimas máquinas de café de la Sala Trojan vistas en Internet, la que más tiempo estuvo en servicio, una Krups ProAroma 305, se puso en subasta en Internet a través del portal de subastas y comercio electrónico eBay con el fin de recaudar fondos para las instalaciones de café del nuevo laboratorio. Fue adquirida por 3.350 libras, siendo el mayor postor el sitio web alemán de noticias Der Spiegel. Posteriormente, la cafetera fue restaurada de manera gratuita por los empleados de Krups, y fue encendida de nuevo en la oficina de redacción de la revista. Desde el verano de 2016, la cafetera se encuentra en préstamo permanente en el museo de informática Heinz Nixdorf Museums Forum de Paderborn, Alemania.

2.1.2. Redes IoT, comunicación entre dispositivos y roles

- Sensores
- Actuadores
- Smart objects
- PAN, WAN, etc

Aunque Internet de las Cosas se mencionase por primera vez a finales del siglo XX, no se comenzó a usar hasta bien entrados los 2010s *Incluir grafica: https://trends.google.com/trends/explore?date=all&q= %2Fm %2F02vnd10&hl=es*

https://www.nougir.com/index.php/blog-3/item/13-que-es-iot-o-internet-de-las-cosas-y-sus-aplicaciones https://informationmatters.net/internet-of-things-definitions/ (DOCUMENTO) https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=11559&lang=es (LIBRO) https://www.taylorfrancis.com/chapters/oa-edit/10.1201/9781003337584-3/internet-things-cognitive-transformation-technology-research-trends-

applications-ovidiu-vermesan-markus-eisenhauer-harald-sundmaeker-patrick-guillemin-martin-serrano-elias-tragos-javier-vali %C3 %B1o-arthur-van-derwees-alex-gluhak-roy-bahr (LIBRO) https://books.google.es/books/about/La_R (DOCUMENTO) https://cdn.ihs.com/www/pdf/IoT_ebook.pdf (DOCUMENTO) https://www.researchgate.net/publication/35254 https://www.microsoft.com/insidetrack/blog/transforming-microsoft-buildings-with-iot-technology-and-indoor-mapping/ (LIBRO) https://www.se.com/us/en/download/document/998-20233517/ (DOCUMENTO) https://www.researchgate.net/publication/320135661_Efficient_IoT-based_Sensor_BIG_Data_Collection-Processing_and_Analysis_in_Smart_Buildings (DOCUMENTO) https://www2.deloitte.com/content/dam/Deloitte/nl/Documents estate/deloitte-nl-fsi-real-estate-smart-buildings-how-iot-technology-aims-to-add-value-for-real-estate-companies.pdf https://innovayaccion.com/blog/aplicando-el-internet-de-las-cosas-a-las-empresas-2 (DOCUMENTO) https://ieeexplore.ieee.org/document/7879243 https://www.intel.com/content/www/us/en/internet-of-things/overview.html https://www.fundacionbankinter.org/ftf-informes/internet-de-las-cosas/ https://www.statista.com/statistic number-of-connected-devices-worldwide/ (VIDEO) https://www.youtube.com/watch?v=RnasX1bFBh8 (LIBRO) https://www.amazon.com/IoT-Fundamentals-Networking-Technologies-Protocols/dp/1587144565 (LIBRO) https://www.ra-ma.es/libro/internet-de-las-cosas_93304/ https://es.wikipedia.org/wiki/Internet_de_las_cosas https://www.eexcellence.es/expertos/kevin-ashton-un-tecnologo-visionario https://www.redhat.com/es/topics/internet-of-things/what-is-iot https://at3w.com/blog/iot-internet-of-things-tecnologia-proteccion-contra-rayo-tomas-tierra/ https://www.itop.es/blog/item/iot-origen-importancia-en-el-presente-y-perspectiva-de-futuro.html https://www.linkedin.com/pulse/el-origen-del-internet-de-las-cosas-iot-comnet-s-a-/ https://www.ui1.es/blog-ui1/historia-y-origen-del-iot https://blog.avast.com/es/kevin-ashton-named-the-internet-of-things https://www.datacenterdynamics.com/en/podcasts/zero-downtime/episode-18-the-origin-of-the-internet-of-things-with-peter-lewis/ https://swifttalk.net/2021/10/06/the-concept-of-iot-internet-of-things/ https://www.cs.cmu.edu/~coke/ https://www.cs.cmu.edu/~coke/history_long.txt https://www.ibm.com/docs/es/aix/7.1?topic=prot namefinger-protocol https://www.livinginternet.com/i/ia_myths_toast.htm https://en.wikipedia.org/wiki/John_Romkey https://ieeexplore.ieee.org/document/7786805 Romkey, J. (2017). Toast of the IoT: The 1990 Interop Internet Toaster. IEEE Consumer Electronics Magazine, *6*(1), 116–119. doi:10.1109/mce.2016.2614740 https://romkey.com/about/ https://blog.avast.com/the-internets-first-smart-device https://en.wikipedia.org/wiki/Trojan_Room_cof https://www.cl.cam.ac.uk/coffee/qsf/coffee.html https://www.cl.cam.ac.uk/coffee/coffee.html https://www.youtube.com/watch?v= https://www.bbc.com/news/technology-20439301 https://www.historyofinformation.com/detail.php?id=1507 https://quentinsf.com/coffeepot/metcalfe/ https://www.cl.cam.ac.uk/coffee/qsf/switchoff.html https://owl.museum-digital.de/object/3761

/*TODO: apartados restantes*/

2.2. Protocolos de mensajería usados en IoT

El uso más común de IoT es desplegar una arquitectura compuesta por varios dispositivos IoT. En mayor parte, estos dispositivos se designarán simplemente como dispositivos IoT, ya sean sensores o actuadores, mientras que habrá pocos dispositivos (al menos uno) con el rol de centro de mensajería. Dependiendo del uso que se le dé a la arquitectura, estos objetos se comunicarán, y dependiendo de quién sea el emisor y el receptor, dan lugar a estos escenarios:

- Un mensaje ha llegado al centro de mensajería IoT. Este mensaje es procesado y se actúa en consecuencia, por ejemplo, enviando la información necesaria a ciertos dispositivos IoT o almacenándola a una base de datos.
- Un dispositivo IoT ha generado datos. Estos datos son procesados y luego enviados al centro de mensajería.
- Un dispositivo IoT ha recibido datos del centro de mensajería. Estos datos son procesados y se actúa en consecuencia.

Estos dispositivos, por sí solos, no son capaces de intercambiar esos grandes volúmenes de datos que tratan. Por ello, tras escoger una tecnología para conectar los dispositivos entre sí, es esencial en el desarrollo de aplicaciones IoT disponer de un protocolo de mensajería.

Un protocolo de comunicación permite que los dispositivos se comuniquen y transmitan mensajes entre los dispositivos IoT y el centro de mensajería. Además, proporciona cierta fiabilidad a la comunicación, ya que permite que los mensajes lleguen y sus datos sean entendidos y procesados correctamente. Esta comunicación ocurre sobre TCP, o incluso sobre abstracciones de mayor nivel como HTTPS.

La elección del protocolo se basa en cómo se adecua al escenario en el que se quiere implementar, considerando requisitos a tener en cuenta como la ubicación, las limitaciones físicas, el consumo, la batería y el coste. Por lo general, no cualquier protocolo de comunicación es apropiado. Los protocolos que se mencionan en este apartado se adecuan a la mayoría de escenarios IoT debido a su rapidez y su fácil implementación, y es posible escoger aquel que se adapte mejor a los requisitos.

2.2.1. MQTT

El protocolo Message Queue Telemetry Transport es uno de los más populares en el ámbito del Internet de las Cosas. Diseñado para ser ligero y adecuado para redes con ancho de banda limitado y dispositivos con pocos recursos, este estándar del comité técnico OASIS permite el transporte bidireccional de mensajes con datos entre múltiples dispositivos.

MQTT utiliza el patrón de comunicación publicación-suscripción. En este patrón, los publicadores categorizan los mensajes, y los suscriptores recibirán mensajes de las categorías de su interés, a diferencia de la

comunicación tradicional en la que los publicadores envían los mensajes directamente a los suscriptores. En el caso de MQTT, el patrón está basado en temas o topics, siendo posible que los suscriptores se interesen por uno o varios. Este patrón permite utilizar una red única para transmitir datos entre dispositivos y servidores, habilitando el control remoto de una gran cantidad de dispositivos a través de Internet.

En una red MQTT, se definen dos roles principales: el broker o intermediario de mensajes y los clientes. El broker MQTT es un servidor comparable a una oficina de correos, que recibe todos los mensajes publicados por los clientes y los dirige a los clientes de destino apropiados. Por otra parte, un cliente es cualquier dispositivo conectado al broker a través de una red, y puede producir y recibir datos al publicar y suscribirse respectivamente. Este mecanismo es útil para compartir datos, controlar y gestionar dispositivos. Por ejemplo, un dispositivo cliente puede publicar datos de sensores y además recibir información de configuración o comandos de control. La enrutación de mensajes que realizada por el broker proporciona transparencia y desacoplamiento en el espacio, ya que el publicador no necesita conocer ni la cantidad ni las direcciones de los suscriptores, y los suscriptores no necesitan conocer al publicador, ambos interactúan únicamente con el broker.

Los mensajes están organizados en una jerarquía de temas o topics. Al publicar un mensaje, se publica en un tema específico, y en el caso de querer publicar a varios se deben realizar varias publicaciones. En cambio, un suscriptor puede suscribirse a un tema específico o a varios simultáneamente y recibirá una copia de todos los mensajes compatibles con los temas suscritos. La manera de indicar varios temas es mediante el uso de los siguientes caracteres comodín:

- Comodín de un nivel '+': coincide con un nivel de tema y puede utilizarse más de una vez en la especificación del tema.
- Comodín de varios niveles '#': coincide con cualquier número de niveles y debe ser el último carácter en la especificación del tema.

Como ejemplo de uso de los temas, cuando se publica un mensaje en el tema "edificioA/sensor1/temperatura", el broker enviará una copia del mensaje los clientes suscritos a los temas "edificioA/sensor1/temperatura", "edificioA/+/temperatura" y "edificioA/#", pero no a un cliente suscrito a "edificioB" o a "edificioA/+/humedad".

La transmisión de mensajes se realiza de forma asíncrona, sin detener la ejecución de ambos componentes a la hora de publicar o recibir, y se puede realizar una comunicación uno a muchos (un publicador y varios suscriptores), muchos a uno (varios publicadores y un suscriptor), uno a uno (un publicador y un suscriptor, menos común) y muchos a muchos (varios publicadores y varios suscriptores).

En el caso de que el broker reciba una publicación de un tema en el cual no hay nadie suscrito, el broker por defecto descarta el mensaje. Es posible activar la retención de mensajes configurando un campo en el mensaje para evitar esto, consiguiendo así que el broker almacene el último mensaje retenido de cada

tema y lo distribuya inmediatamente a cualquier nuevo cliente suscrito, permitiendo así que el suscriptor reciba el valor más reciente en lugar de esperar a una nueva publicación, y además añadiendo soporte a una comunicación desacoplada en el tiempo, donde publicadores y suscriptores no necesitan estar conectados simultáneamente.

El protocolo soporta un mecanismo de limpieza de sesión. Por defecto, un cliente tras desconectarse y volverse a conectar no recibe los mensajes publicados durante su desconexión y el broker olvida las suscripciones del mismo cliente. Pero al desactivar dicho mecanismo, el broker mantiene tanto las relaciones de suscripción como los mensajes offline, enviándolos al cliente al momento de reconectarse, lo cual es útil para dispositivos que se conectan y desconectan constantemente, común en redes IoT. Además, MQTT enfrenta la inestabilidad de la red con un mecanismo Keep Alive que, al transcurrir un prolongado periodo sin interacción, ocurre un ping entre el cliente y el broker para evitar la desconexión. Si el ping falla y se identifica el cliente como desconectado, aplicará un mecanismo Last Will, que publica un último mensaje a un tema específico debido a una desconexión anormal, en el caso de estar configurado.

MQTT dispone de 14 tipos de mensajes diferentes, la mayoría utilizados para mecanismos internos y flujos de mensajes:

- CONNECT: establece una conexión con el broker, y si está configurado, se debe proporcionar un usuario y contraseña.
- DISCONNECT: finaliza una sesión MQTT enviando este mensaje para cerrar la conexión. Esta desconexión se denomina “graceful shutdown” o “apagado elegante”, porque está la posibilidad de conectarse al broker con la misma sesión y reanudar el progreso.
- PINGREQ/PINGRESP: una operación de ping utilizada para saber si está viva la conexión y mantenerla.
- PUBLISH: contiene un mensaje para publicarlo en un tema específico.
- SUBSCRIBE: utilizado por los clientes para suscribirse a un tema específico y recibir las actualizaciones de este.
- UNSUBSCRIBE: mensaje que utiliza un cliente para indicar la pérdida de interés y anular la suscripción a un tema específico
- LWT: este mensaje “last will and testament” (última voluntad y testamento) se configura en un cliente para publicarse automáticamente si ocurre una desconexión inesperada. El broker mantiene un temporizador, y si comprueba que recientemente el cliente no ha publicado ni ha mandado un PINGREQ, se publica el mensaje LWT especificado notificando así a los suscriptores.

El diseño de MQTT se basa en la simplicidad y en minimizar el ancho de banda, dejando la interpretación de los mensajes en manos del desarrollador. Los mensajes transmitidos a través de la red tienen la posibilidad de configurar el QoS o calidad de servicio por cada tema, asociados con distintas garantías de entrega y cómo se entregan los mensajes. Aunque MQTT depende de TCP, el cual tiene su propia garantía de

entrega, históricamente los niveles QoS eran necesarios para evitar la pérdida de datos en redes antiguas y poco fiables, una preocupación válida para las redes móviles actuales. Estos son los siguientes tipos de QoS:

- QoS 0, a lo sumo una vez: los mensajes se envían y no se tiene en cuenta si llegan o no. Está la posibilidad de la pérdida de mensajes y no se hacen retransmisiones.
- QoS 1, al menos una vez: el receptor recibe el mensaje por lo menos una vez. El receptor debe enviar un acuse de recibo al emisor en cuanto reciba el mensaje, y si este ACK nunca llega (ya sea debido a que el mensaje nunca llegó o que el ACK se perdió), el emisor retransmitirá el mensaje, pudiendo producirse mensajes duplicados.
- QoS 2, exactamente una vez: asegura que el mensaje llegue exactamente una vez, manejado mediante la sobrecarga en la comunicación y el envío de una serie de acuses de recibo, y es la mejor opción cuando no se acepta ni la pérdida ni la duplicidad de mensajes.

La transmisión de datos se realiza principalmente sobre la capa TCP/IP, pero existe la posibilidad de operar encima de otros protocolos de red que ofrezcan conexiones ordenadas, sin pérdidas y bidireccionales, y se transmiten en un tamaño reducido de paquetes de datos, estructurado por los siguientes campos:

- Cabecera fija, en la que se especifica el tipo de mensaje, si el mensaje es un duplicado, el QoS, si es un mensaje que retener y el tamaño del paquete.
- Cabecera variable, no siempre presente en los mensajes, y puede transportar información adicional de control.
- Payload o carga útil.

Por defecto, este protocolo envía credenciales y mensajes en texto plano sin medidas de seguridad, pero admite utilizar conexiones TLS/SSL protegidas por certificado, nombre de usuario y contraseña para cifrar y proteger la información transferida contra la intercepción, modificación o falsificación. Además, un broker MQTT tiene soporte para conectar dispositivos IoT a escala masiva, un factor tenido en cuenta a la hora de su diseño y que resulta en una alta capacidad de concurrencia, rendimiento y escalabilidad, características útiles en una red IoT. Implementaciones como EMQX y HiveMQ han alcanzado hitos notables, con 100 y 200 millones de conexiones respectivamente, y un pico de 1 millón de mensajes gestionados por segundo. A esta escalabilidad se le puede sumar la capacidad de implementar múltiples brokers, para así compartir la carga de clientes y tratar la redundancia y la copia de seguridad de los mensajes en caso de fallo.

2.2.2. AMQP

Protocolo de la capa de aplicación de estándar abierto para middleware orientado a mensajes Protocolo binario diseñado para soportar una amplia gama de aplicaciones de mensajería y patrones de comunicación No está diseñado específicamente para soluciones IoT, pero funciona muy bien para las comunicaciones

de mensajes que incluyen muchos escenarios IoT Protocolo binario de capa de aplicación Puede utilizarse para mensajería punto a punto y pub-sub Amplia compatibilidad con escenarios de mensajería Admite el cifrado de mensajes extremo a extremo Mensajes enviados sobre TCP y UDP

Cons: alto consumo de recursos (energía y memoria) Orientación de mensajes, colas, enrutamiento, fiabilidad y seguridad AMQP define como deben comportarse tanto el proveedor del servicio del mensajería como el cliente. Permite que implementaciones de diferentes vendedores puedan funcionar juntoas

AMQP es un protocolo a nivel de cable, que describe el formato de los datos que se envían a través de la red como un flujo de bytes. Cualquier herramienta que pueda crear e interpretar mensajes que se ajusten a este formato de datos puede interoperar con cualquier otra herramienta que lo cumpla, independientemente del lenguaje de implementación

Es un protocolo de mensajería avanzado Permite la comunicación entre servicios y aplicaciones Protocolo abierto que funciona a nivel de capa de aplicación Define la creación de mensajes, encolamiento, enrutamiento de los mensajes producidos y exactitud para entregar los mensajes a los consumidores

Compuesto de un broker de mensajería. Este internamente posee exchanges, donde se conectan los productores de mensajes. También posee colas, vinculadas a exchanges a través de distintos criterios. Los consumidores de los datos se conectan a las colas para extraer los mensajes que producen los publicadores.

El protocolo establece el comportamiento del servidor de mensajería y de los clientes que se conectan al broker, de manera que las implementaciones de distintos proveedores son interoperables AMQP proporciona la unidad de trabajo necesarioa para intercambiar información

Es un protocolo orientado a crear un “cable” entre aplicaciones conectadas Incluye funcionalidad para entregar mensajes de manera fiable, representar los datos a través de diferentes formatos, flexixbilidad para definir datos, preparado para escalabilidad y capacidad de definir varias topologías

El objetivo de diseño de AMQP era conseguir la creación de un protocolo que permita interoperar mediante mensajes, tanto dentro de la misma organizacaión como entre organizaciones, mediante la creación de un mensaje con estándar abierto para realizar transacciones de negocio

Entidades AMQP: - Broker: servidor de mensajería al que los clientes se conectan usando el protocolo AMQP y que se encarga de distribuir los mensajes - Usuario - Conexión: conexión física sobre algún protocolo de transporte como TCP/IP o SCTP (ligada a un usuario) - Canal: conexión lógica ligada a una conexión

Entidades usadas para transferencia de mensajes entre aplicaciones son declaradas dentro de un canal, que garantiza la creación lógica de los elementos necesarios para la comunicación, como la creación de un exchange, una cola a la cual enviar mensajes y la vinculación entre las entidades.

Entidades que definen el protocolo para la creación y consumo de mensajes mediante AMQP: - Exchanges:

entidades a las que los productores envían mensajes, y desde donde se envían los datos a las diferentes colas. Los mensajes se envían a una o más colas según el exchange y la clave de enrutamiento/routing key con que se publican. Tipos de exchange: - Topic exchange: cuando se publica un mensaje, el determina qué colas lo recibirán comparando la routing key del mensaje con el patrón de enrutamiento que vinculan la cola con el exchange - Direct exchange: envía mensajes a un receptor concreto, TODO: seguir

2.2.3. XMPP

Otra manera de comunicar varios dispositivos IoT se logra mediante el Extensible Messaging and Presence Protocol, anteriormente conocido como Jabber. Este protocolo se basa en la transmisión de datos estructurados en formato XML dentro de una red de arquitectura cliente-servidor, en la cual los dispositivos están identificados por un Jabber ID, cuyo formato es similar al de una dirección de correo electrónico (por ejemplo, “abc@example.com”). En esta red, el cliente establece una conexión TCP/IP con el servidor, la cual permanece abierta. Posteriormente, el cliente se autentica con el servidor, y tras una autenticación exitosa, se habilita la posibilidad de enviar y recibir mensajes.

Una característica notable de XMPP es que no existe un servidor central que gestione todas las comunicaciones; el intercambio de mensajes entre clientes y servidores está descentralizado, permitiendo a los usuarios y organizaciones operar su propio servidor XMPP. Esto no restringe a los usuarios a conectarse únicamente con otros usuarios en el mismo servidor, ya que, al ser un estándar abierto regido por la Internet Engineering Task Force, los desarrolladores disponen de un protocolo bien documentado y fiable. De este modo, es posible interoperar entre diferentes implementaciones de XMPP a través de Internet, independientemente del proveedor. En el caso de querer comunicarse con otro servidor, ambos servidores XMPP se intercambian la información necesaria, habilitando un modelo federado.

Este protocolo está diseñado para ofrecer mensajería instantánea o casi en tiempo real a través de la red, sin importar la distancia entre los dispositivos, uno de los problemas más comunes en IoT. Además, permite obtener información de presencia sobre los usuarios conectados y mantener una lista de contactos para cada usuario. XMPP también admite extensibilidad, permitiendo a los desarrolladores añadir características y funcionalidades personalizadas, ofreciendo más allá de la mensajería tradicional y adaptando XMPP a necesidades específicas de aplicaciones, como la transmisión de señales VoIP, video, ficheros, chat grupal, conferencias multiusuario, suscripción de presencia para conocer cuándo alguien está conectado a la red, y comunicación publicación-suscripción para recibir actualizaciones sobre temas específicos de interés.

En los mensajes XMPP se utiliza el formato stanzas XML para estructurar y transportar los datos. Existen 3 tipos principales de stanzas:

- Stanza de mensaje (`message`): utilizado para enviar mensajes instantáneos entre clientes. Contiene los campos remitente, destinatario, cuerpo del mensaje y otros metadatos opcionales. Después de

recibir el mensaje, el servidor utiliza el campo de destinatario para enrutar el propio mensaje. Ejemplo de uso de esta stanza:

```
<message from='abc@example.com'
   to='xyz@example.com'
   type='chat'>
   <body>Hemos tenido una velada encantadora.</body>
</message>
```

- Stanza de presencia (`presence`): permite a las entidades conocer el estado y la disponibilidad online/offline de otros cliente. También puede transportar información adicional, como la actividad del cliente o su ubicación. Cuando un cliente se conecta o desconecta del servidor, envía una stanza de presencia para notificar a otros clientes de su lista de contactos. Ejemplo de uso de esta stanza:

```
<presence from="abc@example.com">
   <show>away</show>
   <status>Paro para comer.</status>
   <priority>5</priority>
</presence>
```

- Stanza de IQ o info/query (`iq`): se usa para consultar al servidor, gestionar suscripciones o intercambiar datos estructurados entre clientes y servidores. Funciona de manera similar a los métodos HTTP GET y POST, siguiendo un patrón de petición-respuesta, en el cual un cliente envía una petición al servidor y este responde con la información solicitada o con una confirmación. Ejemplo de uso de esta stanza:

```
<iq to="user@example.com" type="get" id="314">
   <query xmlns="http://jabber.org/protocol/disco#items" />
</iq>
```

El protocolo XMPP es altamente escalable debido a su capacidad de manejar multitud de conexiones y mensajes simultáneos. Además, al ser descentralizado, permite implementar fácilmente más servidores para gestionar el aumento de usuarios y altos picos de uso. En cuanto a seguridad, XMPP es compatible con cifrado de extremo a extremo mediante TLS o SSL, garantizando así la confidencialidad de los mensajes. Por último, cuenta con una amplia comunidad de usuarios, diversas implementaciones y guías que facilitan a los desarrolladores la creación de aplicaciones que integren este protocolo.

2.2.4. DDS

Diseñado para ser usado en sistemas en tiempo real, y es un estándar máquina-máquina del Object Management Group (OMG) Objetivos DDS: permitir intercambios de datos fiables, de alto rendimiento, interoperables, en tiempo real y escalabes usando un patrón de mensajería publish subscribe Protocolo DDS diseñado para responder a las necesidades específicas de aplicaciones Diseñado para sistemas en tiempo real Conecta dispositivos directamente entre ellos Elimina el requisito del servidor intermediario: mayor velocidad y eficacia en el intercambio de datos Mejor protocolo de aplicación segura, por su naturaleza sin servidor Capaz de transmitir millones de mensajes a multitud de receptores de forma instantánea Entrega en tiempo real, útil para cualquier sistema que lo requiera Demuestra notable capacidad de adaptación, responde fácilmente a las neesidades únicas de cada ámbito de aplicaci´´on Prioriza entrega de un intercambio de datos fiable, escalable, de alta velocidad e interoperable

Caract: - Diseñado para sistemas en tiempo real - Facilita comunicación directa entre dispositivos - Minimiza sobre carga de comunicación

Ventajas: - Arquitectura flexible y adaptable - Baja sobrecarga, permite integración perfecta con sistemas de alto rendimiento - Garantiza una entrega de datos determinista - Altamente escalable para adaptarse a las crecientes demandas del sistema - Optimiza la utilización del ancho de banda de transporte

//TODO: buscar más cosas de DDS (wikipedia o info)

2.2.5. CoAP

El Constrained Application Control es un protocolo de la capa de aplicación que permite a dispositivos con recursos limitados, como los que se encuentran en una red IoT, comunicarse entre sí. Funciona en un marco cliente-servidor, en el cual el cliente realiza una solicitud a un punto de comunicación del dispositivo servidor, y este responde, permitiendo la interoperabilidad entre los dispositivos uno a uno.

Este protocolo opera sobre el protocolo de transporte UDP, que, a diferencia de TCP, no requiere que los dispositivos establezcan una conexión de datos previa al envío de datos. Esto trae tanto consecuencias positivas como negativas. La consecuencia negativa radica en la poca fiabilidad en la comunicación de base, ya que el protocolo UDP no garantiza la entrega de los mensajes, sino que esta garantía se gestiona desde la implementación de CoAP. Es posible establecer acuses de recibo (ACK), de manera que, por cada mensaje enviado, el dispositivo espera un acuse de recibo y, en caso de no recibirlo en un tiempo determinado, el mensaje se retransmite. La consecuencia positiva del uso de UDP es la posibilidad de funcionar en redes con pérdidas o inestables, adecuado para redes IoT, ya que suelen operar en entornos de red difíciles, y la rapidez en la comunicación, pues no requiere una conexión de datos previa, enviando directamente el mensaje.

Esta comunicación utiliza una arquitectura RESTful, en la cual los datos y las funcionalidades se consideran recursos a los que se accede mediante una interfaz estándar y uniforme. Estos recursos se acceden y se interactúa con ellos mediante métodos HTTP estándar (GET, POST, PUT, DELETE, que realizan las funciones "obtener", "crear", "actualizar" y "eliminar" recursos, respectivamente), permitiendo una interoperabilidad sencilla entre distintos tipos de dispositivos y facilitando a los desarrolladores la creación de aplicaciones que usan protocolo. No es necesario que los recursos de la red sean conocidos por el dispositivo que vaya a utilizarlos, ya que CoAP implementa un mecanismo de descubrimiento integrado, útil en redes IoT en las que los dispositivos constantemente se conectan y desconectan. Esta función de descubrimiento trata de consultar un recurso conocido "núcleo" en la red, el cual provee la lista de los recursos de los dispositivos en la red. Es decir, si un dispositivo IoT quiere interactuar con los recursos de otro, puede consultar al núcleo y comprobar qué recursos hay disponibles y cómo puede interactuar con ellos.

El intercambio de mensajes CoAP entre dispositivos es asíncrono, lo que significa que un dispositivo puede enviar una solicitud a otro y continuar ejecutando otras tareas mientras que la respuesta puede recibirla en cualquier momento. Esto se logra mediante un id en los mensajes, permitiendo al dispositivo relacionar peticiones con respuestas, asegurando un alto nivel de fiabilidad en el intercambio de mensajes. Esta comunicación asíncrona es crucial en redes IoT, ya que los dispositivos pueden no estar siempre conectados o disponibles para responder en el momento de la solicitud.

CoAP se basa en el intercambio de mensajes compactos codificados en un formato binario simple. El tamaño de estos mensajes no puede superar al necesario para encapsularlos dentro de un datagrama IP, y tienen distintos campos:

- Versión de CoAP.
- Tipo de mensaje.
- Longitud del Token.
- Código, en formato "c.dd", siendo "c" la clase indicando si es una solicitud, una respuesta satisfactoria, un error del cliente o un error del servidor, y "dd" el detalle.
- ID de mensaje.
- Token, usado para correlacionar solicitudes y respuestas.
- Opciones.
- Payload o carga útil.

Los distintos tipos de mensajes que se pueden transmitir son los siguientes:

- Mensajes confirmables (CON): utilizados cuando se necesita asegurar que el mensaje llegue al destinatario. Contienen un temporizador y un mecanismo de retroceso. Al transmitir una petición CON, se espera recibir un mensaje ACK con el mismo ID de la petición o una respuesta en un mensaje CON y con un ID distinto.

- Mensajes no confirmables (NON): son mensajes menos fiables, usados para enviar información no crítica, que no requieren un acuse de recibo (ACK). En el caso de enviarse una solicitud como un mensaje NON, la respuesta también se recibirá como un mensaje NON (en el caso que el servidor tenga la información necesaria para responder).
- Mensajes de acuse de recibo (ACK): son transmitidos para reconocer que ha llegado un mensaje confirmable específico identificado por su ID de transacción. Estos mensajes pueden tener su propio payload y algunas opciones para detallar la recepción.
- Mensaje de reinicio (RST): cuando al receptor le falta información para procesar una solicitud, transmite un mensaje RST. Esto ocurre cuando el receptor se ha reiniciado y no ha persistido adecuadamente la petición recibida anteriormente, o cuando cancela una transacción.

Además es un protocolo diseñado para requerir poca energía en la transferencia (tiene bajo consumo de recursos), y permite transferir tanto datos como archivos, utilizar el protocolo DTLS para aumentar la seguridad de las transferencias, y extender la implementación del protocolo con funcionalidades adicionales. Por el contrario, es un protocolo menos maduro y menos adoptado que sus alternativas, resultando en una menor cantidad de recursos, guías y herramientas, además de una compatibilidad reducida con otros dispositivos IoT.

https://webbylab.com/blog/mqtt-vs-other-iot-messaging-protocols-detailed-comparison/ https://www.techtarget.com/iotagenda/ti
12-most-commonly-used-IoT-protocols-and-standards https://build5nines.com/top-iot-messaging-
protocols/ https://www.a3logics.com/blog/iot-messaging-protocols/ https://en.wikipedia.org/wiki/MQTT
https://en.wikipedia.org/wiki/Publish %E2 %80 %93subscribe_pattern https://www.emqx.com/en/blog/what-
is-the-mqtt-protocol https://www.gotoiot.com/pages/articles/mqtt_intro/index.html https://dzone.com/refcardz/getting-
started-with-mqtt https://www.hivemq.com/resources/achieving-200-mil-concurrent-connections-with-
hivemq/ http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/ https://en.wikipedia.org/wiki/Advanced_Mess
https://www.gotoiot.com/pages/articles/amqp_intro/index.html https://en.wikipedia.org/wiki/XMPP
https://www.pubnub.com/guides/xmpp/ https://blazeclan.com/blog/xmpp-for-dummies-part-3-stanzas-in-
detail/ https://slixmpp.readthedocs.io/en/latest/api/stanza/presence.html https://slixmpp.readthedocs.io/en/latest/api/stanza/iq.htm
https://www.emqx.com/en/blog/coap-protocol0 https://www.gotoiot.com/pages/articles/coap_intro/index.html
(DOCUMENTO) https://datatracker.ietf.org/doc/html/rfc7252

2.3. Espressif y sus dispositivos

Espressif Systems es una multinacional pública de semiconductores sin fábrica fundada en 2008, que opera como líder mundial en el ámbito del Internet de las Cosas y está comprometida en proporcionar a millones de usuarios algunos de los mejores dispositivos y plataformas de software de la industria, junto con una variedad de soluciones IoT seguras.

La empresa se identifica como una empresa compuesta por especialistas, ingenieros y científicos dedicados al desarrollo de soluciones de vanguardia de bajo consumo que aprovechan la comunicación inalámbrica, el IoT y la inteligencia artificial de las cosas (AIoT). Estas soluciones se caracterizan por su seguridad, robustez, eficiencia energética, versatilidad, asequibilidad y enfoque código abierto.

Con el surgimiento de la inteligencia artificial y la evolución del IoT, la demanda de productos con conectividad inalámbrica segura ha ido creciendo considerablemente, y Espressif Systems ha respondido a este desafío desarrollando soluciones adaptadas a las necesidades del mercado. Espressif emplea los nodos de tecnología avanzada, la informática de bajo consumo, la comunicación inalámbrica, así como la tecnología de malla, para crear conjuntos de chips y módulos de alto rendimiento, que son más inteligentes, adaptables y versátiles.

El compromiso de esta empresa china con el código abierto se refleja en su oferta de una variedad de frameworks y herramientas de desarrollo para construir aplicaciones en diferentes ámbitos, como IoT, audio, malla, conectividad de dispositivos, reconocimiento facial y asistentes de voz. Gracias a las tecnologías y soluciones abiertas de Espressif, permiten acercar el IoT a sus clientes, comerciales y no comerciales, y que desarrolladores de todos los ámbitos puedan utilizarlas a nivel mundial y construir sus propios dispositivos inteligentes conectados y soluciones a algunos problemas del presente. Con este acercamiento a sus clientes, estos últimos pueden efectuar una conectividad inalámbrica de sus productos optimizando el rendimiento de estos y en un reducido tiempo de desarrollo. A su vez, apoyan activamente proyectos de código abierto en la comunidad de makers, creyendo en la accesibilidad de la tecnología como motor para el desarrollo de la sociedad AIoT del futuro.

Además de su enfoque en el código abierto y en la tecnología accesible, Espressif se destaca por el compromiso con la sostenibilidad, la responsabilidad social corporativa y por la inversión en investigación y desarrollo de tecnologías ecológicas. A lo largo de su trayectoria, ha respaldado activamente soluciones que reducen el consumo de energía y el desperdicio de materiales.

Los productos de Espressif se utilizan ampliamente en productos como electrodomésticos, altavoces, bombillas, cámaras, tabletas, TV boxes, terminales de pago y otros dispositivos de electrónica de consumo, y son especialmente útiles en contextos donde es necesario el Internet de las Cosas, comunicación inalámbrica e Inteligencia Artificial. Espressif es conocido por sus populares series de chips, módulos y placas de desarrollo ESP8266, ESP32 y ESP32-S, ESP32-C y ESP32-H, los cuales se analizarán en el siguiente apartado.

https://www.espressif.com/en/company/about-espressif https://www.eurotronix.com/es/fabricantes/espressif https://www.digikey.es/es/supplier-centers/espressif-systems

Capítulo 3

Metodología

En este apartado se deben indicar las metodologías empleadas para planificación y desarrollo del TFG, así como explicar de modo claro y conciso cómo se han aplicado dichas metodologías.

3.1. Guía Rápida de las Metodologías de Desarrollo del Software

A continuación, se incluye una guía rápida que puede ser de gran utilidad en la elaboración de este capítulo.

3.1.1. Proceso de Desarrollo de Software

El proceso de desarrollo de software se denomina también ciclo de vida del desarrollo del software (SDLC, Software Development Life-Cycle) y cubre las siguientes actividades:

- Obtención y análisis de requisitos (requirements analysis). Es la definición de la funcionalidad del software a desarrollar. Suele requerir entrevistas entre los ingenieros de software y el cliente para obtener el ‘qué’ y ‘cómo’. Permite obtener una especificación funcional del software.
- Diseño (SW design). Consiste en la definición de la arquitectura, los componentes, las interfaces y otras características del sistema o sus componentes.
- Implementación (SW construction and coding). Es el proceso de codificación del software en un lenguaje de programación. Constituye la fase en que tiene lugar el desarrollo de software.
- Pruebas (testing and verification). Verificación del correcto funcionamiento del software para detectar fallos lo antes posible. Persigue la obtención de software de calidad. Consisten en pruebas de caja negra y caja blanca. Las primeras comprueban que la funcionalidad es la esperada y para ello se verifica que, ante un conjunto amplio de entradas, la salida es correcta. Con las segundas se comprueba la robustez del código sometiéndolo a pruebas cuya finalidad es provocar fallos de software.

Esta fase también incorpora las pruebas de integración en las que se verifica la interoperabilidad del sistema con otros existentes.

- Documentación (documentation). Persigue facilitar la mejora continua del software y su mantenimiento.
- Despliegue (deployment). Consiste en la instalación del software en un entorno de producción y puesta en marcha para explotación. En ocasiones implica una fase de entrenamiento de los usuarios del software.
- Mantenimiento (maintenance). Su propósito es la resolución de problemas, mejora y adaptación del software en explotación.

3.1.2. Metodologías de Desarrollo Software

Las metodologías son el modo en que las fases del proceso software se organizan e interaccionan para conseguir que dicho proceso sea reproducible y predecible para aumentar la productividad y la calidad del software.

Una metodología es una colección de:

- Procedimientos: indican cómo hacer cada tarea y en qué momento,
- Herramientas: ayudas para la realización de cada tarea, y
- Ayudas documentales.

Cada metodología es apropiada para un tipo de proyecto dependiendo de sus características técnicas, organizativas y del equipo de trabajo. En los entornos empresariales es obligado, a veces, el uso de una metodología concreta (p. ej. para participar en concursos públicos). El estándar internacional ISO/IEC 12270 describe el método para seleccionar, implementar y monitorear el ciclo de vida del software.

Mientras que unas intentan sistematizar y formalizar las tareas de diseño, otras aplican técnicas de gestión de proyectos para dicha tarea. Las metodologías de desarrollo se pueden agrupar dentro de varios enfoques según se señala a continuación.

- Metodología de Análisis y Diseño de Sistemas Estructurados (SSADM, Structured Systems Analysis and Design Methodology). Es uno de los paradigmas más antiguos. En esta metodología se emplea un modelo de desarrollo en cascada (waterfall). Las fases de desarrollo tienen lugar de modo secuencial. Una fase comienza cuando termina la anterior. Es un método clásico poco flexible y adaptable a cambios en los requisitos. Hace hincapié en la planificación derivada de una exhaustiva definición y análisis de los requisitos. Son metodologías que no lidian bien con la flexibilidad requerida en los proyectos de desarrollo software. Derivan de los procesos en ingeniería tradicionales y están

enfocadas a la reducción del riesgo. Emplea tres técnicas clave:

- Modelado lógico de datos (Logical Data Modelling),
- Modelado de flujo de datos (Data Flow Modelling), y
- Modelado de Entidades y Eventos (Entity EventModelling).

- Metodología de Diseño Orientado a Objetos (OOD, Object-Oriented Design). Está muy ligado a la OOP (Programación Orientada a Objetos) en que se persigue la reutilización. A diferencia del anterior, en este paradigma los datos y los procesos se combinan en una única entidad denominada objetos (o clases). Esta orientación pretende que los sistemas sean más modulares para mejorar la eficiencia, calidad del análisis y el diseño. Emplea extensivamente el Lenguaje Unificado de Modelado (UML) para especificar, visualizar, construir y documentar los artefactos de los sistemas software y también el modelo de negocio. UML proporciona una serie diagramas de básicos para modelar un sistema:

  - Diagrama de Clases (Class Diagram). Muestra los objetos del sistema y sus relaciones.
  - Diagrama de Caso de Uso (Use Case Diagram). Plasma la funcionalidad del sistema y quién interacciona con él.
  - Diagrama de secuencia (Sequence Diagram). Muestra los eventos que se producen en el sistema y como este reacciona ante ellos.
  - Modelo de Datos (Data Model).

- Desarrollo Rápido de Aplicaciones (RAD, Rapid Application Developmnent). Su filosofía es sacrificar calidad a cambio de poner en producción el sistema rápidamente con la funcionalidad esencial. Los procesos de especificación, diseño e implementación son simultáneos. No se realiza una especificación detallada y se reduce la documentación de diseño. El sistema se diseña en una serie de pasos, los usuarios evalúan cada etapa en la que proponen cambios y nuevas mejoras. Las interfaces de usuario se desarrollan habitualmente mediante sistemas interactivos de desarrollo. En vez de seguir un modelo de desarrollo en cascada sigue un modelo en espiral (Boehm). La clave de este modelo es el desarrollo continuo que ayuda a minimizar los riesgos. Los desarrolladores deben definir las características de mayor prioridad. Este tipo de desarrollo se basa en la creación de prototipos y realimentación obtenida de los clientes para definir e implementar más características hasta alcanzar un sistema aceptable para despliegue.

- Metodologías Ágiles. "[…] envuelven un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo. Cada itera-

ción del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación. Teniendo gran importancia el concepto de"Finalizado" (Done), ya que el objetivo de cada iteración no es agregar toda la funcionalidad para justificar el lanzamiento del producto al mercado, sino incrementar el valor por medio de "software que funciona" (sin errores). Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. […]"}[1]

3.2. Proceso de Testing

Se debe indicar qué tipo de pruebas se han realizado, por ejemplo las siguientes:

- Pruebas modulares (pruebas unitarias). Su propósito es hacer pruebas sobre un módulo tan pronto como sea posible. Las pruebas unitarias que comprueban el correcto funcionamiento de una unidad de código. Dicha unidad elemental de código consistiría en cada función o procedimiento, en el caso de programación estructurada y cada clase, para la programación orientada a objetos. Las características de una prueba unitaria de calidad son: automatizable (sin intervención manual), completa, reutilizable, independiente y profesional.
- Pruebas de integración. Pruebas de varios módulos en conjunto para comprobar su interoperabilidad.
- Pruebas de caja negra.
- Beta testing.
- Pruebas de sistema y aceptación.
- …

3.3. Marco Tecnológico

En esta sección se enumeran las tecnologías y herramientas utilizadas en la elaboración del TFG. A continuación, se citan algunos ejemplos.

3.3.1. Herramientas CASE (Computer Aided Software Engineering)

Las herramientas CASE están destinadas a facilitar una o varias de las tareas implicadas en el ciclo de vida del desarrollo de software. Se pueden dividir en la siguientes categorías:

- Modelado y análisis de negocio.
- Desarrollo. Facilitan las fases de diseño y construcción.
- Verificación y validación.
- Gestión de configuraciones.
- Métricas y medidas.

[1] Fuente: Wikipedia

- Gestión de proyecto. Gestión de planes, asignación de tareas, planificación, etc.

3.3.2. IDE (Integrated Development Environment)

- Notepad++: https://notepad-plus-plus.org/
- Visual Studio Code: https://code.visualstudio.com/
- Atom: https://atom.io/
- GNU Emacs: https://www.gnu.org/s/emacs/
- NetBeans: https://netbeans.org/
- Eclipse: https://eclipse.org/
- QtCreator: https://www.qt.io/ide/
- jEdit: http://www.jedit.org/

3.3.3. Depuración

- GNU Debugger: https://www.gnu.org/s/gdb/

3.3.4. Repositorios y control de versiones

- Git: https://git-scm.com/
- Mercurial: [https://www.mercurial-scm.org/
- Github: https://github.com/
- Bitbucket: https://bitbucket.org/
- SourceTree: https://www.sourcetreeapp.com/

3.3.5. Documentación

- LaTeX: https://www.latex-project.org/
- Markdown: https://markdown.es/
- Doxygen: https://www.doxygen.nl/
- DocGen: http://mtmacdonald.github.io/docgen/docs/index.html
- Pandoc: http://pandoc.org/

3.3.6. Gestión y Planificación de Proyectos

- Trello: https://trello.com/
- Jira: https://es.atlassian.com/software/jira
- Asana: https://asana.com/
- Slack: https://slack.com/

- Basecamp: https://basecamp.com/
- Teamwork Proyects: https://www.teamwork.com/project-management-software
- Zoho Projects: https://www.zoho.com/projects/

Capítulo 4

Resultados

En los que se describen cómo se ha aplicado el método de trabajo para el caso concreto del TFG, incluyendo aquellos elementos (modelos, diagramas, especificaciones, etc.) más importantes y relevantes que se quieran hacer notar.

4.1. Resultados del TFG

Este apartado debe explicar cómo el empleo de la metodología permite satisfacer tanto el objetivo principal como los específicos planteados en el TFG así como los requisitos exigidos (según exposición en capítulo Objetivos).

Capítulo 5

Conclusiones

En este capítulo se debe incluir el juicio crítico y discusión sobre los resultados obtenidos. Si es pertinente deberá incluir información sobre trabajos derivados como publicaciones o ponencias, así como trabajos futuros, solo si estos están planificados en el momento en que se redacta el texto. Incluirá obligatoriamente la justificación de las competencias de la tecnología específica cursada por el estudiante que se han adquirido durante el desarrollo del TFG

5.1. Revisión de los Objetivos

En esta sección se deberá revisar en qué grado se han completado los objetivos fijados al principio del proyecto. Se deberá también indicar las posibles desviaciones de los objetivos fijados, así como de la planificación, y tratar de justificar tales desviaciones.

5.2. Presupuesto

Si el TFG consiste en el desarrollo e implementación de un prototipo, la memoria debe incluir el coste del prototipo considerando tanto el hardware como los recursos humanos necesarios para su desarrollo.

Cuando se tiene en cuenta la puesta en marcha de un proyecto de ingeniería, la planificación y presupuesto que se realizan de modo previo a su ejecución son críticos para gestionar los recursos que permitan alcanzar los objetivos de calidad, temporales y económicos previstos para el proyecto. Es muy importante que todas las justificaciones aportadas se sustenten no solo en juicios de valor sino en evidencias tangibles como: historiales de actividad, repositorios de código y documentación, porciones de código, trazas de ejecución, capturas de pantalla, demos, etc.

5.3. Competencias Específicas de Intensificación Adquiridas y/o Reforzadas

Se deberán listar aquellas competencias de la intensificación que hayan sido adquiridas y/o reforzadas con el desarrollo de este TFG, incluyendo su justificación.

Bibliografía

[1] BibTeX format description. https://www.bibtex.org/Format/.

[2] PlantUML de un vistazo. https://plantuml.com/es/.

Anexo A. Uso de la Plantilla

A continuación se presenta la estructura y utilización de esta plantilla.

A.1. Configuración

La configuración de la plantilla se realiza a través del fichero *config.yaml*. Este es un fichero *yaml*, cuyos campos se describen en la Tabla A.1.

Tabla A.1: Valores del fichero *config.yaml*

clave	valor
Cite	Citas bibliográficas a incluir en la bibliografía no referenciadas en el texto
Cotutor	Nombre y apellidos del co-tutor académico
Csl	Fichero csl con el formato de las referencias
Department	Departamento del tutor académico
Language	Lenguaje de la plantilla [english|spanish]
Month	Mes de defensa del TFG
Name	Nombre del autor del TFG
Technology	Tecnología específica
Title	Título del TFG
Tutor	Nombre del tutor académico
Year	Año de defensa del TFG

Un ejemplo de configuración de este fichero se muestra en el Listado A.1.

Listado A.1: Ejemplo de fichero de configuración config.yaml

```
1 Cite: @Gutwin2010GoneBut, @Rekimoto1997PickDrop
2 Cotutor: John Deere
```

```
3  Csl: input/resources/csl/acm-sig-proceedings.csl
4  Department: Ciencias de la Computación
5  Language: spanish
6  Month: Agosto
7  Name: Johny Anston
8  Technology: Sistemas de Información
9  Title: Una Aplicación para Resolver Problemas Genéricos
10 Tutor: Adam Smith
11 Year: 2023
```

A.2. Estructura de Directorios

La plantilla tiene la estructura mostrada en la Figura A.1.

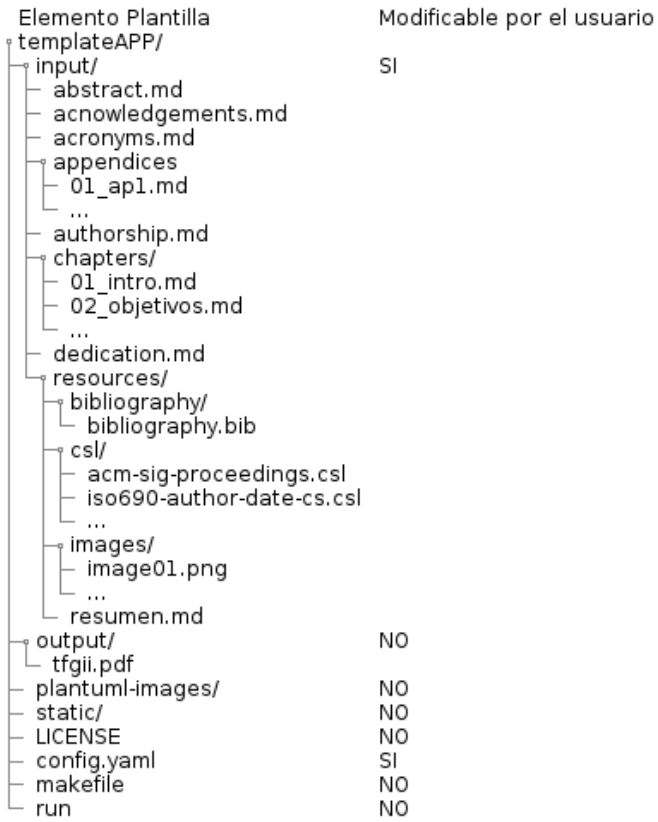

Figura A.1: Estructura de directorios de la plantilla del TFG

El documento generado por la plantilla es **output/tfgii.pdf**. Este documento se sobreescribe cada vez que se compila la plantilla. Es este documento, compilar se refiere a procesar los elementos de la plantilla para

generar el documento *pdf*.

Las carpetas y ficheros modificables por el usuario y que contienen la información propia del TFG son las siguientes (marcadas como SI en la figura A.1):

- input
- config.yaml

El resto de ficheros y directorios no deben de ser modificados por el usuario para el correcto funcionamiento de la plantilla.

A.2.1. Carpeta input

La carpeta *input* contiene los ficheros que corresponden con las partes del TFG.

En primer lugar están los ficheros correspondientes al **abstract**, **acknowledgements**, **acronyms**, **authorship** y **dedication**. En estos, a excepción del **acronyms**, sólo hay que introducir el texto correspondiente.

El fichero **acronyms** tiene un formato yaml. En el, hay que introducir los pares *acrónimo* y *valor* separados por ":" y un espacio. Un ejemplo sería el del Listado A.2.

Listado A.2: Ejemplo de definición de acrónimos

```
1 HTML: Lenguaje de marcas de hypertexto
2 HCI: Human Computer Interaction
```

También se encuentran en el directorio *input* los directorios **appendices** y **chapters**. Contienen, respectivamente, los apéndices y los capítulos del *TFG*, que corresponden con ficheros en *markdown*. En estos ficheros hay que tener en cuenta lo siguiente:

- El fichero debe comenzar con un encabezado de primer nivel
- Los ficheros se ordenan en el documento según su orden alfabético
- Sólo se tienen en cuenta los ficheros que comienzan con dos dígitos numéricos, p.ej. 01
- Si un fichero no comienza con dos dígitos numéricos, es ignorado y no se incluye a la hora de generar el documento

A.2.2. Carpeta resources

El directorio *resources* contiene recursos de utilidad para la generación del documento, como imágenes o bibliografía.

La bibliografía se recoge en el fichero **bibliography/bibliography.bib**. Es un fichero en formato *bibtex* [1] que contiene los elementos bibliográficos. Por ejemplo, en el Listado A.3 se encuentra definido el

elementos bibliográfico *bibtex*.

Listado A.3: Ejemplo de definición de elemento bibliográfico

```
@misc{bibtex,
  title = {BibTeX Format Description},
  howpublished = {\url{https://www.bibtex.org/Format/}},
  note = {Accessed: 2023-09-28}
}
```

El directorio **csl** contiene ficheros con diferentes formatos de bibliografía. Finalmente, el directorio **images** contiene las imágenes a utilizar en el documento.

A.3. Elementos del Documento

Si bien la plantilla permite utilizar cualquier elemento de *Markdown* y *pandoc*, hay una serie de elementos que pueden ser de especial utilidad de cara a la realización de un TFG:

- Referencias bibliográficas
- Figuras
- Tablas
- Listados de código
- Notas al pie de página
- Acrónimos
- PlantUML
- Referencias dentro del Documento

A.3.1. Citas Bibliográficas

La plantilla utiliza la bibliografía que se recoge en el fichero correspondiente. Para incluir una cita a un elemento bibliográfico, hay que añadir en el documento su referencia precedida del símbolo @. Por ejemplo, para añadir la cita al elemento biliográfico *bibtex* se haría de la forma indicada en el Listado A.4.

Listado A.4: Ejemplo de cita de elemento bibliográfico

```
Es un fichero en formato *bibtex* @bibtex.
```

A.3.2. Figuras

Para incluir figuras lo hacemos añadiendo una imagen en la cual especificamos el elemento *label*, que sirve para referenciarla. Por ejemplo, el código del Listaqdo A.5 produce como resultado la Figura A.2.

Listado A.5: Ejemplo de definición de una figura

```
1 ![Logo de HTML5\label{anexo:ejemploFiguraResultado}](HTML5_sticker.png){width
    ↪ =50%}
```

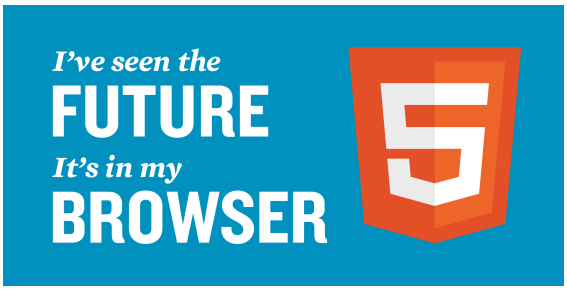

Figura A.2: Logo de HTML5

A.3.3. Tablas

Para definir una tabla se utiliza la sintaxis mostrada en el Listado de Código A.6, cuyo resultado se muestra en la Tabla A.2.

Listado A.6: Ejemplo de definición de una tabla

```
1 | Right | Left | Default | Center |
2 |------:|:-----|---------|:------:|
3 | 12 | 12 | 12 | 12 |
4 | 123 | 123 | 123 | 123 |
5 | 1 | 1 | 1 | 1 |
```

Tabla A.2: Tabla de ejemplo

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

A.3.4. Notas al Pie de Página

Para la definición de notas al pie de página se utiliza el código mostrado en el Listado A.7, cuyo resultado se muestra al pie de esta página[1].

[1] Ejemplo de nota al pie de página.

Listado A.7: Ejemplo de definición de una nota a pie de página

```
cuyo resultado se muestra al pie de esta página[^anexo:ejemploNotaPie].

[^anexo:ejemploNotaPie]: Ejemplo de nota al pie de página.
```

A.3.5. Acrónimos

Para hacer referencia a un acrónimo, por ejemplo HTML, hay que utilizar el código mostrado en el Listado A.8. Previamente, el acrónimo debe de haber sido definido, tal y como se muestra en el Listado A.2.

Listado A.8: Ejemplo de definición de referencia a un acrónimo

```
Para hacer referencia a un acrónimo, por ejemplo [HTML](#HTML), hay ...
```

A.3.6. PlantUML

Esta plantilla permite la inserción de diagramas en *PlantUML* [2]. En el Listado A.9 se muestra un ejemplo básico de un diagrama de objetos en *PlantUML*, cuyo resultado se muestra en la Figura A.3.

Listado A.9: Ejemplo de código plantuml básico

```
@startuml
skinparam dpi 300
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 .. Object08 : some labels
@enduml
```

También se puede dar formato, colores y formas, a los diagramas generados con PlantUML, tal y como se muestra en el Listado A.10. El resultado se muestra en la Figura A.4.

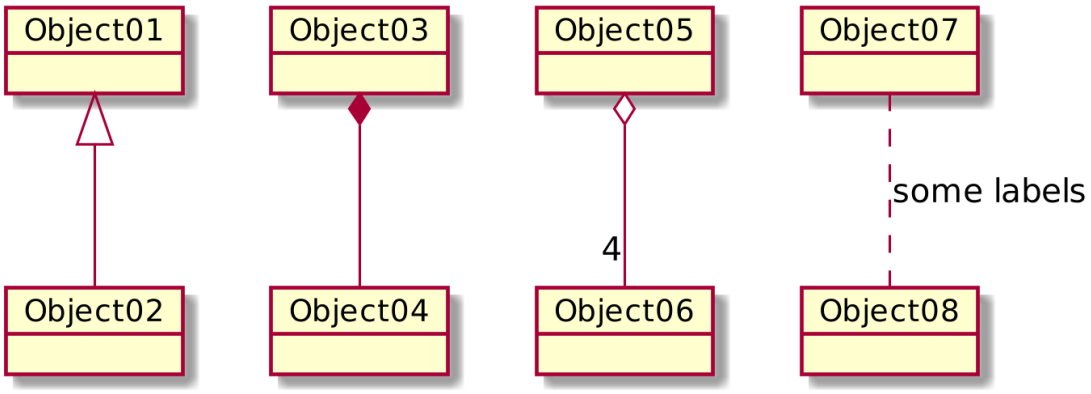

Figura A.3: Ejemplo de plantuml básico

Listado A.10: Ejemplo de código plantuml con formato

```
@startmindmap
skinparam dpi 300
<style>
mindmapDiagram {
  node {
    BackgroundColor white
    FontColor #B30033
  }
  :depth(1) {
    BackgroundColor white
    FontColor #B30033
  }
  .uclm {
    BackgroundColor white
    FontColor #B30033
  }
  .active {
    BackgroundColor #B30033
    FontColor white
  }
}
</style>
* Requisitos
** Requisitos de usuario <<active>>
```

```
*** Requisitos de dominio <<active>>
*** Requisitos de negocio <<active>>
*** Requisitos de usuario final <<active>>
** Requisitos de sistema
** Requisitos de SW/HW

@endmindmap
```

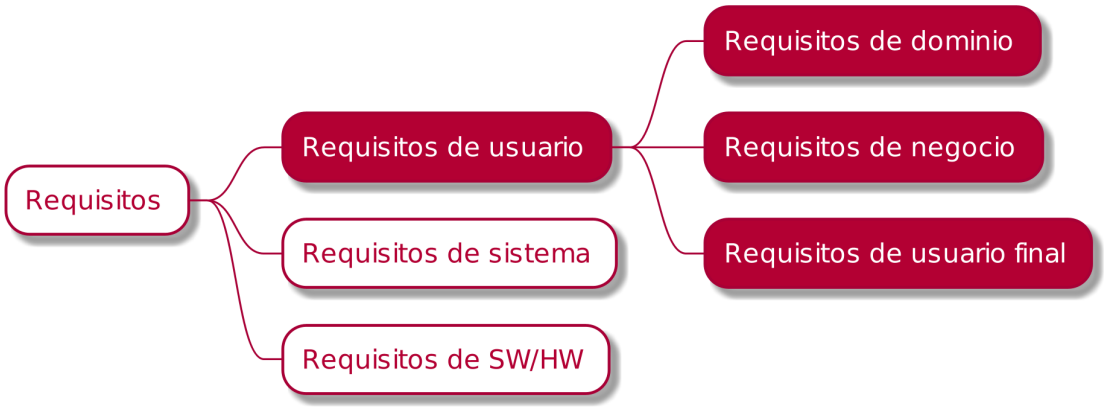

Figura A.4: Ejemplo de plantuml con formato

A.3.7. Referencias Dentro del Documento

Para hacer una referencia a un capítulo dentro del documento se indica entre llaves el texto del enlace, seguido entre paréntesis del nombre del capítulo precedido del carácter *'#'*. El nombre del capítulo se indica en minúsculas y se sustituyen los espacios por el carácter *'-'*. Por ejemplo, si se quiere referenciar al capítulo *"Estructura de Directorios"* se utilizará el siguiente código mostrado en la Figura A.11.

Listado A.11: Referencia dentro del documento

```
Referencia a la [estructura de directorios](#estructura-de-directorios)
```

A.4. Creación del Documento

La creación del documento se puede realizar utilizando un contenedor en *Docker* o de forma nativa en *Linux*[2]. La Figura A.5 muestra la estructura del directorio raíz de la plantilla.

A continuación, se explica cómo funciona cada una de estas aproximaciones, así como sus requisitos.

[2]Se ha probado que funciona correctamente en la distribución *Ubuntu 22.04.1*.

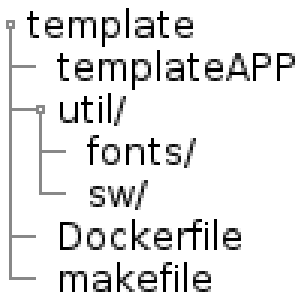

Figura A.5: Estructura de directorios raíz de la plantilla

A.4.1. Ubuntu

Ejecutar 'make' en el directorio raíz de la plantilla.

Hay que tener instalado el siguiente software[3]:

- texlive-latex-base
- texlive-lang-spanish
- texlive-xetex
- make
- pandoc-plantuml-filter

También hay que instalar las fuentes adecuadas. Para ello crear el directorio (si no existe), copiar las fuentes y actualizarlas según el código de la Figura A.12.

Listado A.12: Instalación de las fuentes para el documento

```
$ sudo mkdir -p /usr/share/fonts/truetype/msttcorefonts/
$ sudo cp util/fonts/* /usr/share/fonts/truetype/msttcorefonts/
$ sudo fc-cache -f -v
$ sudo texhash
```

Finalmente, hay que instalar pandoc, pero en su versión 2.19.2-1. Para ello, ejecutar la orden de la Figura A.13 en el directorio raíz de la plantilla. Si se utiliza cualquier otra versión es muy probable que se tengan problemas, por ejemplo a la hora de generar imágenes con *plantuml*.

Listado A.13: Instalación de Pandoc en su versión 2.19.2-1

```
$ sudo dpkg -i util/sw/pandoc-2.19.2-1-amd64.deb
```

[3] Para instalar el software en Ubuntu hay que ejecutar *apt install nombre-paquete*

A.4.2. Docker

Ejecutar ‘make docker’ en el directorio raíz de la plantilla.

En el caso de utilizar Docker, el único prerequisito es tenerlo instalado[4].

[4] https://docs.docker.com/engine/install/ubuntu/

Anexo B. Título del Anexo II

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh

sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

B.1. Una sección

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu

eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.