Universidad de Valladolid

Escuela Técnica Superior de Ingenieros de Telecomunicación

Trabajo de fin de grado

Grado en Ingeniería de Tecnologías Específicas de Telecomunicación
Mención en sistemas de telecomunicación

Diseño y desarrollo de un sistema de monitorización y control del riego en una explotación agrícola con cobertura extendida

Autor
D. Miguel Campano García

Tutores
Dr. D. Manuel Rodríguez Cayetano
Dra. Dña. María Jesús González Morales

Título	**Diseño y desarrollo de un sistema de monitorización y control del riego en una explotación agrícola con cobertura extendida**
Autor	**D. Miguel Campano García**
Tutores	**Dr. D. Manuel Rodríguez Cayetano** **Dra. Dña. María Jesús González Morales**
Departamento	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

Tribunal

Presidente	Dr. D. Eduardo Gómez Sánchez
Vocal	Dr. D. Manuel Rodríguez Cayetano
Secretario	Dr. D. Juan Carlos García Escartín
Fecha	Diciembre de 2020
Calificación	

A J.J. por traerme hasta aquí

Resumen

A medida que la tecnología avanza, aumenta la necesidad de los sectores productivos por aplicar las novedades técnicas a todos sus procesos.

Uno de los campos en el que los nuevos paradigmas tecnológicos presentan una mayor proyección es el sector agrícola. La agricultura, desde sus orígenes, se ha visto beneficiada por avances técnicos que han permitido mejorar la producción, aumentar la eficiencia y reducir el tiempo invertido en las labores de campo. Ahora, inmerso el ser humano en la Era Digital, es momento de aplicar las Tecnologías de Comunicación al sector agrícola.

A lo largo de este Trabajo de Fin de Grado se desarrolla un sistema de monitorización y control de riego en una explotación agrícola con cobertura extendida.

El sistema previo, en el que se basa este trabajo, utilizaba la tecnología WiFi, lo que no permitía el despliegue en aquellas zonas sin cobertura de red, además, el alcance de una red WiFi no llega a los 100 metros en exteriores. Para decidir qué tecnología de comunicación es la encargada de soportar el sistema de cobertura ampliada, se elabora un estudio comparativo de las diferentes opciones disponibles. Una vez realizado el estudio, se escoge la tecnología LoRa, que puede llegar a ofrecer un radio de cobertura de más de 10 km y es totalmente independiente de la red de telefonía móvil.

La implementación del sistema se llevará a cabo mediante la utilización de diversas tecnologías adicionales.

Para soportar los elementos hardware propios de la comunicación LoRa se emplean placas Arduino (modelo MKR WAN 1300), el ordenador Raspberry y un concentrador ic-880a.

Con el objeto de desarrollar el software necesario para la comunicación, se ha utilizado el lenguaje Python en el *framework* Flask para adaptar las comunicaciones, el servicio web de Chirpstack para gestionar los mensajes en formato LoRa y la librería MKRWAN para la gestión de paquetes LoRa en la placa Arduino.

Una vez finalizado el proceso de diseño e implementación, se realizan las pruebas, comprobando que el sistema conforma una adaptación satisfactoria.

Abstract

As technology advances, the need for productive sectors to apply technical innovations to all their processes increases.

One of the fields in which the new technological paradigms present a greater projection is the agricultural sector.

Agriculture, from its origins, has benefited from technical advances that have made it possible to improve production, increase efficiency and reduce the time invested in field work. Now, with the human being immersed in the Digital Age, it is time to apply Communication Technologies to the agricultural sector.

Throughout this Final Degree Project, an irrigation monitoring and control system is developed in an agricultural holding with extended coverage. The previous system, on which this work is based, used WiFi technology, which did not allow deployment in those areas without network coverage, in addition, the range of a WiFi network does not reach more than 100 meters outdoors.

The system developed in this work uses (after carrying out a study of possibilities) LoRa technology, which can offer a coverage radius of more than 10 km and is totally independent from the mobile phone network. Various additional technologies are used for the implementation of the system.

To support the hardware elements of LoRa communication, Arduino boards (MKR WAN 1300 model), the Raspberry computer and an ic-880a hub are used.

In order to develop the necessary software for communication, the Python language has been used in the Flask framework to adapt communications, the Chirpstack web service to manage messages in LoRa format and the MKRWAN library for the management of LoRa packages in the Arduino board.

Once the design and implementation process is completed, tests are carried out, verifying that the system conforms to a satisfactory adaptation.

Agradecimientos

A Manuel y a Chus, por la inestimable ayuda que me han brindado.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Motivación

Para abordar este trabajo resulta conveniente comenzar reflexionando sobre el auge del Internet de las Cosas, más comúnmente conocido como *Internet of Things (IoT)*.

Una definición a grandes rasgos del concepto de *IoT* podría ser la interconexión de dispositivos y objetos a través de una red, brindando a estos elementos la posibilidad de comunicarse e interaccionar entre sí. El objetivo de esta comunicación constante es el de automatizar actividades y procesos diarios, bien sean en un ámbito personal o laboral.

Estos dispositivos podrían ser de cualquier tipo, y esto abre un amplío abanico de posibles aplicaciones entre los que destacan: la domótica (este concepto hace referencia a la capacidad de automatizar una vivienda o edificación de cualquier tipo), las ciudades inteligentes, los dispositivos *wearables* (se entiende por *wearables* aquel dispositivo electrónico que se incorpora en alguna parte de nuestro cuerpo interactuando de forma continua con el usuario y con otros dispositivos con la finalidad de realizar alguna función concreta) o la agricultura inteligente. Es preciso mencionar que estas aplicaciones sólo representan una pequeña parte de todo lo que es posible conseguir mediante la aplicación del *IoT*, hoy por hoy parece no haber límites para esta tecnología.

Este trabajo se circunscribe a la agricultura inteligente, un campo en desarrollo, pero que comienza tímidamente a crecer.

Los beneficios de aplicar *IoT* a la agricultura son prácticamente inmediatos: si es posible conocer el estado de los cultivos en tiempo real, se ahorrará una cantidad preciosa de tiempo que se puede dedicar a otras tareas como el estudio de los cultivos, por ejemplo. Además, mediante la recolección y el análisis de los datos extraídos sobre el terreno, se podrán optimizar los recursos (agua de riego, fertilizantes o abonos), y esto se traduce en una mayor eficiencia, ahorrando costes y mejorando la calidad del producto.

Por estas razones, resulta casi imperativo desarrollar sistemas que permitan controlar y monitorizar las actividades relativas a la explotación agrícola.

Es necesario, además, que la tecnología a desplegar sea capaz de ofrecer un control total al usuario. Este Trabajo de Fin de Grado se centrará en el control y programación del riego en explotaciones agrícolas, pero con importantes y necesarias mejoras.

El sistema desarrollado parte de un sistema de monitorización y control de riego que usa la tecnología WiFi para la interconexión de los circuitos controladores de las válvulas de riego y la aplicación de gestión. Dado que la tecnología WiFi tiene un alcance muy limitado (debido a las dimensiones de los cultivos), el sistema desarrollado contará con una tecnología de comunicación de largo alcance.

Cabe destacar que una vez desplegado el sistema en su totalidad, añadir otras funcionalidades será una tarea trivial, pues la complejidad reside en la implantación de la comunicación y en el desarrollo de la aplicación.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado es adaptar el sistema de monitorización y control del sistema de riego para explotaciones agrícolas basado en la tecnología WiFi, desarrollado en el trabajo de Fin de Grado de Luis Carlos Parra Rebollo [Par20], para que utilice una tecnología de comunicación de área extendida que permita su implantación en terrenos de mayores dimensiones.

Como el sistema debe ser capaz de operar en un área extendida, superando las limitaciones impuestas por tecnologías como WiFi o *Bluetooth*, se trabaja en el marco de las redes de gran área y baja potencia, también conocidas como *Low-Power Wide-Area Wetwork (LPWAN)*. Las redes *LPWAN* son un tipo de redes diseñadas para ser aplicadas en el ámbito del *IoT* y muy convenientes para el propósito de este trabajo.

Al ser tecnologías que utilizan una baja potencia de transmisión, a pesar de su largo alcance, los dispositivos que las usan pueden ser alimentados con baterías y, al mismo tiempo, se puede lograr una gran duración de las mismas (del orden de meses o años). Además, resultan ideales para extender la cobertura por la totalidad de las áreas de cultivo.

1.3. Metodología

Para decidir qué tecnología será la base de la comunicación entre los dispositivos, se realiza, en primer lugar, un estudio comparativo de las posibles tecnologías a emplear para la transmisión a los nodos finales (en este caso, las válvulas).

Este estudio es extremadamente importante ya que una mala decisión sobre la tecnología a emplear acarrearía graves problemas a la hora de implementar de manera efectiva el prototipo.

Una vez elaborado el estudio, se escoge la tecnología que conformará la base de la comunicación del nuevo sistema. A continuación, se procede a realizar una identificación de todos los elementos y procesos que son necesarios para la comunicación mediante la tecnología seleccionada.

Una vez completado ese análisis, se procede al diseño e implementación de todos los componentes previamente identificados.

Finalmente, se realiza una fase de pruebas en la que se verifica que todas las funcionalidades han sido adaptadas correctamente.

1.4. Estructura del documento

Este trabajo de Fin de Grado está compuesto por tres capítulos más esta introducción. Durante el segundo capítulo, titulado “Estado del arte”, se lleva a cabo un estudio de las posibles tecnologías que podrían soportar la comunicación dentro del sistema, una vez realizado el estudio, se escoge la tecnología LoRa y se explican los elementos del sistema desde un punto de vista descriptivo.

En el tercer capítulo, titulado “Análisis, diseño e implementación”, se detallan las fases de análisis, diseño e implementación. Para este desarrollo se compara el sistema inicial con el sistema a desarrollar, haciendo especial hincapié en las diferencias y novedades que se introducen.

Finalmente, en el cuarto y último capítulo, se describen las conclusiones y las líneas futuras, que sugieren una posible hoja de ruta para desarrollos venideros basados en este Trabajo de Fin de Grado.

Capítulo 2

Estado del arte

2.1. Tecnologías de comunicación de largo alcance

Las tecnologías de radiocomunicación se utilizan diariamente para permitir la comunicación entre dispositivos. Quizá, el ejemplo, más omnipresente de este uso constante sea la tecnología WiFi, cuya utilización ha experimentado un drástico aumento desde su origen, este estándar permite conectar dispositivos entre sí de una forma muy sencilla, pero presenta una limitación: sólo puede funcionar en distancias reducidas.

Como respuesta a los estándares de tecnologías inalámbricas *short-range*, y a la necesidad de superar los límites de distancia, nacen las tecnologías de largo alcance, que permiten expandir las fronteras de la comunicación hasta decenas de kilómetros con un bajo uso de energía, todo esto conlleva un precio: la tasa de datos debe permanecer baja.

Esto permite la creación de las redes de gran alcance y baja potencia, más conocidas como *Low-Power Wide-Area Networks (LPWAN)*. Dentro de esta clase se encuentran múltiples opciones que presentan marcadas diferencias: las hay que despliegan sus propias estaciones base, otras utilizan la infraestructura de operadores móviles, algunas usan bandas libres; y, por supuesto, todas las tecnologías *LPWAN* se distinguen y diferencian por su radio de cobertura, consumo de energía y tasa de datos permitida.

A continuación, se exponen brevemente las características más importantes a la hora de decidir qué tecnología *LPWAN* emplear en una aplicación.

La elección de la banda de frecuencia a utilizar es una cuestión muy importante, ya que el empleo de las bandas de frecuencia de telefonía celular supone una gran ventaja pues está garantizado un uso correcto de las bandas, pero, por el contrario, conlleva un coste de contratación, un menor control sobre los datos e incluso puede darse el caso de que la banda esté muy ocupada, sobre todo en áreas urbanas.

Por el contrario, el uso de bandas sin licencia *ISM* (son bandas de radio reservadas internacionalmente para el uso de energía de radiofrecuencia para fines industriales, científicos y médicos distintos de las telecomunicaciones) no asegura una utilización correcta de las frecuencias, ya que cualquiera puede emitir, y eso podría conllevar una elevada presencia de interferencias, [Dig20]. Además, se debe cumplir con las restricciones del ciclo de trabajo, que variarán en función de la potencia de transmisión y a zona geográfica.

En compensación, el uso de bandas libres implica que se puede expandir libremente la red con nuevas estaciones base y también brinda la posibilidad de crear redes privadas.

Otro factor clave es el consumo de energía de los dispositivos ya que se espera de ellos que sean capaces de funcionar de manera autónoma (normalmente, alimentados por baterías) durante largos períodos de tiempo. Es posible que los nodos estén emplazados en lugares que no permitan un cambio o recargo de batería de manera regular, por estas razones es muy interesante que el gasto energético sea el menor posible.

El radio de cobertura es también un parámetro de gran relevancia, en función del objetivo que tenga sistema a desarrollar, será necesario estudiar qué tecnología es la más apropiada para ofrecer el servicio óptimo. Es preciso señalar que existe un compromiso entre el alcance máximo y la tasa de envío de un sistema, de manera que cuanto mayor sea la distancia del nodo, menor será la cantidad de datos por segundo que se podrá transmitir.

2.1.1. Narrowband Internet of Things

Descripción general

Narrow Band IoT (NB-IoT) es la respuesta de la organización 3GPP ante las necesidades de comunicación *IoT* y la presión de otros desarrollos emergentes en el campo *IoT*, [Fly20a]. Esta tecnología se enmarca en las redes *LPWAN*, caracterizadas, como ya se ha comentado previamente, por proveer de una gran cobertura en largas distancias a los dispositivos que no necesiten una alta velocidad de transmisión de datos, garantizado así un bajo consumo energético.

NB-IoT es capaz de soportar, con un alto grado de eficiencia, las aplicaciones *IoT* y además soporta la necesidad de la comunicación masiva entre máquinas, más conocida como *massive Machine Type Communication (MTC)*.

Características

Una de las primeras cualidades a destacar sobre *NB-IoT* es que permite establecer conexiones masivas, llegando a superar los 52000 terminales por canal [PJ18].

Como los modelos de comunicación *MTC* se basan en la transmisión de paquetes de pequeño tamaño con baja periodicidad, y en donde la latencia no representa un factor crucial, se puede permitir que un gran número de usuarios utilicen una misma célula.

Además, *NB-IoT* presenta una gran flexibilidad en cuanto a su modulación: opera a 180 KHz y no permite enviar y recibir al mismo tiempo (lo que reduce su precio al no tener que implementarse un transceptor), pero se pueden fijar 12 subportadoras con un espaciado de 15 KHz, o ajustar aún más y crear 48 subportadoras con un espaciado de 3.75 KHz, [PJ18].

Para lograr este gran aprovechamiento del espectro, se definen 3 modos de operación [Boi+17]:

- *Stand alone*: se define un espacio dentro del espectro, que queda reservado para únicamente para *NB-IoT*, está es la opción que permite reutilizar el canal *GSM* y aprovechar su desempeño en frecuencia [Boi+17].
- *In Band*: Se define un bloque de recurso dentro de una portadora *LTE*, con una planificación de usuarios de recursos que puede minimizar las interferencias.
- *In Guard Band*: en este modo, se aprovechan las bandas de guarda de cada portadora *LTE*. Se ofrece un gran tasa de bajada ya que se tiene un espacio de hasta 100 KHz (el tamaño de la banda de guarda en *LTE*).

NB-IoT está diseñado para dotar de gran duración a las baterías: utiliza métodos de recepción discontinua (*eDRX*) combinados con sistemas de ahorro de energía (*PSM*) [Mar20].

En ciertos casos, se puede apreciar una mejora en el nivel de la señal si se comparan las señales de *NB-IoT* con las de *General Packet Radio Service (GPRS)* [Dur20].

Esta mejora se ha alcanzado gracias a la transmisión múltiple, mecanismo para el que se definen tres zonas en función de la distancia: CE level 0, CE level 1 y CE level 2, cada una de estas zonas tendrá definido un número de repeticiones. Además, también se observa una mejora en la relación señal a ruido [PJ18].

NB-IoT no soporta esquemas de modulación más complejos que QPSK, de esta forma no se aumenta en demasía ni la complejidad de los dispositivos, ni los costes [PJ18].

Como se puede deducir, *NB-IoT* opera en las bandas de redes, gestionadas por las compañías telefónicas, de ahí que su despliegue esté supeditado a los intereses de estas proveedoras de servicios de telecomunicación.

Cabe mencionar que el coste asociado a la implantación de esta tecnología no es muy elevado, ya que se utilizan las bandas de *GSM* y *LTE*, es muy importante recalcar que *NB-IoT* puede coexistir con las redes móviles convencionales [Boi+17].

2.1.2. *Long Term Evolution Machine-Type Communication*

Descripción general

Long Term Evolution- Machine (LTE-M) es otra de las tecnologías que se postulan para soportar la comunicación *IoT*, también está desarrollada por 3GPP [Fly20b] y de nuevo, se engloba dentro las *LPWAN*, previamente mencionadas. Es preciso diferenciar *LTE-M* de *enhanced Machine Type Communication (eMTC)* que tan solo es una subclase de *LTE-M*. No se deben utilizar los dos términos indistintamente [Hal20a].

Características

LTE-M también se denomina en ocasiones como el nuevo *GSM* para comunicación *Machine to Machine*. La mejor manera de resumir sus prestaciones es compararlo con *NB-IoT*. *LTE-M* está pensado para transmitir a una alta velocidad de datos, según las fuentes, se puede llegar hasta 10 Mbps, aunque conviene ser cautos con esta afirmación, pues es en realidad LTE CAT 1 quien permite alcanzar estas velocidades [ubl20].

Además, se garantiza una latencia tan baja que se puede hablar de comunicación en tiempo real, garantizando la movilidad. Esta posibilidad es muy interesante ya que es aplicable a soluciones para *tracking* (proceso de estimar en el tiempo la ubicación de uno o más objetos móviles).

Como característica adicional reseñable, LTE-M soporta *Voice over LTE (VoLTE)* , el servicio de voz que opera, como su propio nombre indica, sobre *LTE* [Elk20].

Esta tecnología comparte muchas características con la anteriormente mencionada *NB-IoT*, pero la más importante es la utilización de la infraestructura de *LTE*, permitiendo ser desplegada con facilidad allá donde ya exista cobertura *LTE*. El resto de diferencias y semejanzas son relativas a las cuestiones técnicas de la comunicación (tasas de envío o ancho de banda)

Los primeros pasos para el despliegue de esta red ya se han comenzado a dar, en España ha sido Orange la primera compañía que ha volcado su atención sobre esta tecnología.

2.1.3. Sigfox

Descripción general

Sigfox también es una red de *IoT* diseñada para funcionar con un bajo consumo, pero aporta una diferencia crucial con respecto a las anteriores: esta red es independiente de los despliegues de telefonía. Aunque conviene matizar que el uso de Sigfox sí conlleva la contratación de un plan de suscripción para cada dispositivo, pues la aspiración de Sigfox es convertirse en un operador global de *IoT*.

Aunque Sigfox en sus inicios fue una empresa francesa fundada en 2009, en múltiples fuentes ahora es catalogada como la primera red *IoT* dedicada [Sig20b], [Fer20]. Esta red aporta soluciones al mundo de la comunicación de máquina a máquina con su oferta de

conectividad totalmente orientada a las comunicaciones de baja velocidad. Tanto es así que Sigfox, en colaboración con otros operadores de cada país (en España Cellnex) [Fer20] ha desplegado ya su red por la geografía europea, y planean implementarla en todos los lugares del mundo [Sig20a].

Características

Sigfox utiliza las bandas libres o bandas ISM (son bandas de radio reservadas internacionalmente para el uso de energía de radiofrecuencia para fines industriales, científicos y médicos distintos de las telecomunicaciones) [Dig20], con un rango de frecuencias que va desde los 862 MHz hasta los 928 MHz (dependiendo de la región). Divide el espectro en 400 canales de 100 Hz.

Emplea una modulación diferente en subida (*DBPSK*) y en bajada (*GSFK*), trabajando siempre sobre la tecnología *Ultra Narrow Band* [Aug+16].

El tamaño de sus mensajes es muy reducido (12 bytes en subida y 8 en bajada), permitiendo la posibilidad de mandar 140 mensajes al día desde cada nodo, todos estos mensajes se gestionan a través de un *gateway* Sigfox que puede llegar a procesar hasta 1.3 millones de mensajes del día. La tasa de datos, como ya se ha comentado, es baja en comparación con la de las tecnologías mencionadas con anterioridad, cada dispositivo Sigfox puede llegar a transmitir a 100 bps [Aug+16]. Estas cualidades tan específicas permiten a los diferentes dispositivos Sigfox funcionar con un escaso empleo de energía, alargando la vida de las baterías hasta 20 años.

Para asegurar la correcta recepción se sobreponen 3 células de cobertura, cada uno perteneciente a un *gateway*. Estas células presentan un rango de entre 30 y 50 km en zonas rurales y hasta 10 km en áreas urbanas.

Todas estas características aplicadas conjuntamente conforman una tecnología totalmente dedicada al *IoT*, aunque tal vez sea ligeramente más apropiada para adquisición de datos que para escenarios de control [Aug+16].

2.1.4. *Long Range* (LoRa)

Descripción general

En el sentido estricto, el término LoRa es un técnica de modulación derivada de *Chirp Spread Spectrum (CSS*[1]*)* [RP16], patentada por Semtech. Pero el término se utiliza para denominar de manera global al sistema de comunicación de baja potencia, orientado al *IoT*. También se maneja el término LoRaWan, que hace referencia al protocolo de red que utiliza la tecnología LoRa, este protocolo se emplea para comunicar y administrar los dispositivos LoRa dentro de su red.

A grandes rasgos, este protocolo define la comunicación entre los nodos (dispositivos finales) y los *gateways* (encargados de la gestión de los nodos) [SEM20f] [Cat20].

Al igual que hace Sigfox, la tecnología LoRa también opera sobre las bandas ISM [Dig20], liberando así, la comunicación inalámbrica, del lazo de las operadoras telefónicas.

Características

Como ya se ha mencionado, LoRa utiliza el espectro libre para su comunicación. En Europa se utiliza la banda de los 868 MHz, en Estados Unidos se usa la de los 915 MHz y en Asia la de los 433 MHz.

[1] *Chirp Spread Spectrum*, es una técnica de espectro expandido que utiliza pulsos chirp modulados de frecuencia lineal de banda ancha para codificar información.

Al utilizar esta banda, resulta más sencillo para un usuario no experimentado configurar su propia red, aunque por supuesto ya hay ciertos operadores que empiezan a ofrecer redes LoRa en áreas concretas, como por ejemplo Orange en España [Ora20].

Esta tecnología comparte muchas de las características propias de las *LPWAN* ya expuestas previamente: bajo consumo de energía, largo rango de cobertura, baja tasa de datos. Pero, además, comparte ciertos elementos con Sigfox, ya que también están presentes en LoRa los *gateways* que controlan y administran la comunicación entre los nodos.

Una de las propiedades más interesantes de LoRa es el empleo del método de ajuste dinámico (*ADR*) [The20a], que permite al dispositivo final, o incluso al *gateway*, adaptar sus parámetros de potencia de emisión y tasas de transferencia en función de la distancia que separa el dispositivo final del *gateway*, teniendo en cuenta también, el tamaño del mensaje. Este ajuste permite alcanzar unas comunicaciones con alta eficiencia energética a la máxima velocidad posible.

El factor de ensanchamiento o *Spreading Factor (SF)* es también un parámetro configurable en la comunicación LoRa, de manera que se puede aumentar el ancho del pulso, aumentando a sus vez el tiempo en el aire *(time on air)* y por tanto el alcance. El precio que hay que pagar por aumentar el SF es una menor tasa de datos [SEM20d] [Ora20].

Un factor limitante es el ciclo de trabajo (*Duty Cycle*) [The20b], que hace referencia a la cantidad de tiempo que puede transmitir un único dispositivo. El *Duty Cycle* típicamente es de un 1 por ciento, lo que significa que el dispositivo puede transmitir 1 de cada 100 unidades de tiempo.

También es importante mencionar la existencia de *The Things Network*, una estructura de código abierto cuyo objetivo es proveer de cobertura LoRaWAN a todas las regiones. Esta red crece mediante la intervención de usuarios particulares o empresas que contribuyen de manera voluntaria al proyecto. La comunidad ofrece ayuda y soporte, y también una gran aporte de la estructura software necesaria para la comunicación LoRa, para que todos los usuarios puedan desplegar sus propias redes y así ampliar la cobertura, LoRa[The20c].

2.1.5. Justificación de la tecnología escogida

Una vez estudiadas las diferentes opciones, es preciso decidir cuál es la más conveniente para desarrollar el sistema de este Trabajo de Fin de Grado.

La primera opción, *NB-IoT*, podría ser una buena candidata: permite la conexión de una cantidad muy alta de dispositivos, ofrece un tamaño de paquetes más que aceptable, presenta muy baja latencia (este factor no es crucial para el sistema) y aprovecha las estructuras de red *LTE* y *GSM* preexistentes, sin interferir con las mismas.

Es esta última característica la que representa un obstáculo para la elección de *NB-IoT*, tanto las redes *LTE*, como las redes *GSM*, son administradas por compañías telefónicas, de manera que, si el sistema se desplegase sobre esta tecnología, quedaría supeditado al control y los intereses de la proveedora. Además, la cobertura *LTE* dista mucho de ser omnipresente en el territorio de la península, ocurre lo mismo (aunque en menor grado) con *GSM*, especialmente en zonas rurales donde se pretende aplicar el sistema.

Utilizar las estructuras de *GSM* tiene otra desventaja, y es que ya se lleva mucho tiempo anunciando que se va a proceder a su desmantelamiento [Nae20].

El análisis de *LTE-M* arroja unos resultados muy semejantes al de *NB-IoT*. *LTE-M* también ofrece unas cualidades técnicas más que suficientes para cumplir el propósito de nuestro sistema, incluso podría puntualizarse que son innecesarias, ya que cuenta con un ancho de banda de 1 MHz y el servicio de voz.

La razón para desestimar esta opción es la misma que en *NB-IoT*, una vez más, el sistema se vería subordinado a las decisiones de las operadoras y a la contratación de un plan de suscripción.

Sigfox presenta significativas diferencias con respecto a las anteriores opciones ofrecidas por 3GPP. Esta red, ha sido diseñada íntegramente y desde cero, para dar soporte a las redes *IoT*, por lo que está muy optimizada y podría resultar ideal, desde el punto de vista técnico, para soportar la estructura del sistema.

El escaso consumo de energía de los nodos Sigfox, combinado con el uso de la banda de frecuencias libres, hacen que sea idóneo para sistemas de agroindrustria conectada [sig20].

Podría sugerirse que la limitación en el tamaño de los paquetes (12 bytes en subida y 8 en bajada) es un gran inconveniente, pero podría resolverse con una adaptación de los sistemas de envío de paquetes dentro del sistema. Si bien, es cierto que en distintas fuentes se sugiere que Sigfox podría ser más apropiado para adquisición de datos que para escenarios de control [Aug+16].

Las razones por las que Sigfox ha sido descartado como tecnología de comunicación para el sistema son múltiples. Todas las aplicaciones que pretendan utilizar la red de Sigfox han de ajustarse a sus especificaciones, definidas en febrero de 2020 [Sig20c], pero además, todas deben pasar por un proceso de certificación antes de poder emplear la red Sigfox, esto podría llegar a representar un obstáculo adicional a la hora de desplegar el sistema. Se añade a esta circunstancia, todas las limitaciones que existen en cuanto al uso de las bandas libres en Europa.

El segundo argumento se basa en que Sigfox no deja de ser una empresa, cuyo objetivo a largo plazo es convertirse en proveedor de redes *IoT* por todo el mundo. Podría llegar el momento en que esta empresa tomase alguna decisión que afectase de manera negativa al correcto desarrollo del sistema, por ejemplo: podría empezar a cobrar por volumen de datos transmitidos.

La última candidata, y también la opción escogida, es la tecnología LoRa.

LoRa se postula como la opción más adecuada para nuestro sistema: la utilización de las bandas libres de frecuencias (accesibles en todo el territorio independientemente de los despliegues de la telefonía), el bajo consumo de energía en sus transmisiones, su rango adaptativo (que permite ajustar los parámetros de envío para optimizar la comunicación, su alto grado de configurabilidad). Todas estas características son ideales para lograr la adaptación del sistema a grandes distancias.

Además, la tecnología LoRa presenta una cualidad que la distingue sobre las demás: el usuario o desarrollador puede implementar su red de manera totalmente autónoma.

Si bien es cierto, que unirse a la red global de TTN conlleva grandes ventajas y facilidades a la hora de desarrollar una aplicación, hay múltiples opciones y una ingente cantidad de información para crear una red propia y privada, en la que todas las partes del sistema estén bajo control y supervisión del propietario. Esta última característica no la presentaban ninguna de las otras tecnologías candidatas.

Por todo lo anteriormente expuesto, se escoge LoRa como la tecnología que soportará la estructura de comunicación del sistema con rango ampliado.

2.2. Descripción detallada de la tecnología LoRa

2.2.1. Arquitectura de la comunicación LoRa

En una red LoRa se distinguen 4 elementos muy bien diferenciados: los nodos finales, los *gateways*, el servidor de red y el servidor de aplicación.

Cada una de estas partes tienen misiones y características muy concretas. En los siguientes apartados se expondrán, de manera resumida, pero ahondando en los detalles más importantes, las cualidades y funcionamiento de cada uno de los elementos.

Dispositivos LoRa

Los dispositivos LoRa representan los nodos finales de la comunicación, aquellos puntos hacia los que va dirigida la información y desde donde se mandan los datos en el enlace de subida. Actualmente hay decenas de fabricantes que incluyen la tecnología LoRa en su catálogo. Al tener tan amplia oferta de dispositivos, las posibles aplicaciones son casi infinitas: agricultura, monitorización del ambiente, *IoT* aplicado a la industria, logística o domótica son algunos ejemplos de procesos que ya tienen hardware específicamente diseñado para ellos. Todos los nodos LoRa comparten la misma estructura: un microcontrolador, un módulo radio LoRa, y una antena.

Al estudiar el funcionamiento de los dispositivos LoRa, es necesario detenerse en como estos llevan a cabo su proceso de activación, esta operación es necesaria porque de no llevarse a cabo, los nodos no podrían comunicarse.

Hay 2 posibles métodos para activar un dispositivo en una red LoRa[Sem16]:

- *Over-The-Air-Activation(OTAA)*: este es el método más utilizado porque es la manera más segura de conectar los nodos al servidor de red. A continuación, se describen de manera resumida los pasos que se llevan a cabo en la activación *OTAA*.

  1. Los nodos deben contar con 3 parámetros para realizar su activación:
     - *DevEUI*: actúa como identificador único del dispositivo, es semejante a una dirección MAC de un dispositivo de red.
     - *AppEUI*: identifica el servidor de aplicación, es similar a un número de puerto.
     - *AppKey*: clave generada para asegurar la integridad del mensaje.

     El servidor de red debe tener almacenada el mismo *AppKey*.
  2. El nodo genera el *DevNonce*, que es un número generado de manera aleatoria para evitar que otros dispositivos puedan replicar el *join-request*. Este *DevNonce* es agrupado en un mensaje junto con el *DevEUI* y el *AppEUI*, y se genera un código de integridad de mensaje con la *AppKey*. Este mensaje se manda al servidor de red.
  3. Una vez recibido el mensaje en el servidor de red, este comprueba si el *devNonce* ha sido utilizado en alguna ocasión, y, con el código de integridad autentifica al nodo, es decir: con el *DevNonce*, el *AppEUI* y el *DevEui* recibidos, genera su propio código de integridad utilizando el *AppKey*, y si este código propio coincide con el del nodo final, se autentificará el dispositivo.
  4. Una vez aceptado el dispositivo, el servidor de red genera 3 nuevos datos:
     - *DevAddr*: enlaza el *DevEUI* con una dirección de red más corta para reducir el tamaño de las cabeceras en las tramas transmitidas, es similar una dirección IP cliente.
     - *AppNonce*: es un número generado aleatoriamente.
     - *NetID*: es un identificador de red.

     Estos datos se empaquetan en un mensaje junto con unos ajustes de red (que incluyen la lista de canales de frecuencia, y el retraso entre transmisión y recepción).
  5. Sobre este mensaje, se genera el código de integridad de mensaje con el *AppKey*, y se manda el mensaje encriptado hasta el nodo.
  6. Al terminar este proceso, tanto el nodo final como el servidor de red poseerán idénticos *AppNonce* y *DevNonce*, estos serán utilizados para generar las claves (*NwkSKey* y *AppSKey*) que se enviarán al servidor de aplicación. Con este último paso queda cerrada la comunicación. Se utilizarán las claves *NwkSKey* y *AppSkey*

para comunicarse con el servidor de red y el servidor de aplicación, respectivamente.

En la figura 2.1 se representa un esquema de este proceso para facilitar su comprensión.

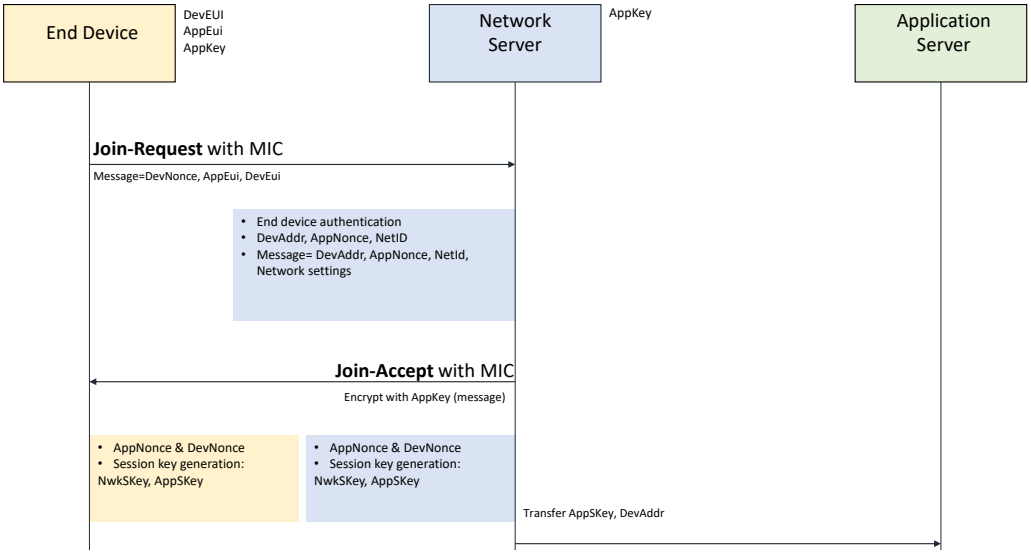

FIGURA 2.1: Esquema de la arquitectura software

- *Activation-By-Personalisation (ABP)*: Este sistema es mucho más sencillo, pero, como consecuencia de su simplicidad, es menos seguro que *OTAA*. A continuación, se describen los detalles de este procedimiento:

  1. El nodo final no guarda el *DevEUI*, ni el *AppEUI* ni el *AppKey*, el servidor de red no guarda el *AppKey*.
  2. El nodo final posee desde un primer momento su *DevAddr*, la *AppSkey* y *NwkSKey*. El servidor de red cuenta con el *DEveAddr* y su *NwkSKy*, y el servidor de aplicación conoce el *DevAddr* y el *AppSKey*.
  3. Como todas las partes conocen las claves desde el primer momento, los mensajes se encriptan y se mandan directamente.

Otro de los aspectos más importantes de los nodos LoRa es su capacidad de operar en distintas configuraciones, hay 3 posibles modos de definir el comportamiento de un nodo LoRa, a continuación, se detalla cada uno de estas configuraciones [Chi20b]:

- Nodo clase A: en este modo, el nodo siempre está dormido a menos que tenga algo que transmitir, en cuyo caso lo enviará y justo después abrirá durante unos instantes, una ventana de recepción para escuchar.

  Este mecanismo resulta muy apropiado para aquellas aplicaciones de monitorización que no reciban controles desde el *gateway* [SEM20a].

- Nodo clase B: en este caso, el nodo abre ventanas de recepción en función de unos tiempos previamente programados.

Para realizar esta operación, es preciso recibir una señal de sincronización desde el *gateway* (esta sincronización se ejecuta de manera periódica). De esta forma, el servidor de red (cuyo funcionamiento se explica en apartados posteriores) sabrá en que momentos escucha el nodo y enviará la información en el instante preciso. También mantendrá su ventana de recepción abierta justo después de transmitir [SEM20b].

- Nodo clase C: esta última configuración es la que emula una escucha continua del nodo. La ventana de recepción está siempre abierta, a excepción de aquellos momentos en los que el propio nodo esté transmitiendo.

  Este es, obviamente, el nodo que más energía consume, pero el que reduce en mayor grado la latencia en las comunicaciones [SEM20c].

Gateway LoRa

El *gateway* LoRa es el intermediario que permite a los nodos finales transmitir datos a la red.

Actúa como un enlace entre dos redes diferentes: la red LoRa sobre la que los nodos finales envían sus mensajes y la red basada en IP en la que se encuentran el servidor de red y el servidor de aplicación [Alf20].

Es conveniente aclarar que el *gateway* no traduce las tramas que le llegan, las transmite tal y como las recibe al servidor de red (a través de una interfaz con una tasa de transferencia mayor), y este se encarga de decodificar el contenido. Como no podía ser de otra forma, es el servidor de red y no el *Gateway* el encargado de generar los paquetes que se enviarán a los nodos finales.

Como se observa, el papel del *gateway* es crucial, pues él es el proveedor de cobertura para los nodos. Uno o más nodos se conectan a uno o más *gateways* conformando una red en estrella. La gran cobertura (entre 1 y 15 kilómetros) que llegan a ofrecer, posibilita que se puedan desplegar nodos en lugares muy remotos.

En el mercado hay múltiples opciones: existen *gateways* con muchos canales, lo que implica que pueden recibir simultáneamente la información de cientos de nodos, y también hay *gateways* más sencillos que sólo tienen un solo canal, que permiten recibir datos de tan solo unos pocos nodos al mismo tiempo.

Servidor de red

Conforma el corazón de la comunicación LoRa, es donde tienen lugar los procesos más complicados de tratamiento de datos.

El servidor de red se encarga del encaminamiento de mensajes recibidos desde el *gateway* a la aplicación adecuada.

Otra de sus tareas es determinar qué *gateway* es el más apropiado para reenviar cada mensaje en el enlace descendente, para tomar esta decisión, el servidor de red analiza distintos parámetros relativos a la calidad del enlace. Entre estos parámetros destacan el *RSSI* (que indica la intensidad de la señal recibida) y la *SNR* (Relación señal a ruido) de los paquetes anteriormente recibidos. Es preciso señalar que en el sistema que se desarrolla, se cuenta únicamente con un *gateway*.

Gestiona también la duplicación de mensajes: como es posible que varios *gateways* reciban información del mismo nodo en el mismo momento, el servidor de red se encarga de detectar esa duplicidad y eliminar los mensajes redundantes [Alf20].

Como ya se ha comentado en la descripción del *gateway*, el servidor de red también tiene la misión de decodificar los mensajes enviados desde los nodos finales, y realizar el proceso de encriptación para aquellos mensajes que se envían a los nodos.

Servidor de aplicación

Es aquí donde se encuentran ubicadas las aplicaciones de *IoT*. Estas son las encargadas de realizar todo el procesado útil con los datos recibidos de los nodos finales, o, dependiendo del tipo de aplicación, generar las instrucciones para controlar la acción de los dispositivos finales.

Suelen contar con una interfaz gráfica que permite al administrador controlar todas las estructuras de su red (usuarios, dispositivos, aplicaciones...) [Chi20d]. Generalmente se alojan en una nube privada o pública, e interactúan con el servidor de red a través de una interfaz que controla el servidor de red [Alf20].

2.2.2. Hardware para comunicación LoRa

Arduino

Arduino es una plataforma de desarrollo basada en una placa electrónica de *hardware* libre que incorpora un microcontrolador re-programable y una serie de pines que proporciona acceso a una serie de puertos de entrada/salida analógicos y digitales. Estos puertos permiten establecer conexiones (con objeto de intercambiar información) entre el microcontrolador y diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables dupont) [Ard20a].

Arduino MKR WAN 1300 En esta placa se combina la capacidad de la placa básica Arduino (descrita en el apartado anterior) con la funcionalidad de la conectividad LoRa, creando un conjunto capaz de conformar un nodo de comunicación LoRa. Como se puede suponer, esta placa cumple con las características previamente descritas en la sección de dispositivos LoRa [Ard20c].

Concentrador y controlador

El *gateway* está formado por dos elementos: el concentrador (cuyo cometido es únicamente intercambiar los paquetes LoRa con un dispositivo conectado a una red IP a través de un enlace de comunicaciones), y un ordenador que posea la capacidad de procesar los paquetes LoRa entrantes y salientes y de intercambiarlos con el concentrador. A continuación, se resumen las características de los componentes escogidos para desempeñar esta tarea.

Raspberry Pi 3 Model B Se elige la Raspberry como ordenador para el procesamiento de paquetes LoRa.

Resulta una muy adecuada opción, pues la Raspberry Pi ofrece todas las funcionalidades software de un ordenador, pero en un reducido tamaño y con un precio inferior a cualquier ordenador comercial orientado al público general [Fou20]. Una de las características mas interesantes de Raspberry para la creación de un *gateways* es la posibilidad de utilizar el bus de comunicaciones SPI a través de sus pines.

Para el montaje del *gateway* utilizado en este sistema se aprovecha esta funcionalidad mediante la conexión de los pines *NSS*, *SCLK* , *MISO* y *MOSI* al concentrador, tal y como se describe en la siguiente guía, desarrollada por Gonzalo Casas [Cas20].

Concentrador iC880A-SPI LoRa Concentrator Este concentrador es la parte central del *gateway* LoRa. Se trata de una de las opciones ofrecidas por IMST GmBh [Wir20] para la conectividad LoRa, esta placa está basada en el Chip SX1301 desarrollado por Semtech [SEM20e].

Ofrece la posibilidad de obtener una conexión robusta entre el *gateway* y una gran cantidad de nodos (al contar con 8 canales) a larga distancia.

Algunas de las aplicaciones que se sugieren para este concentrador son: mediciones inteligentes, automatización industrial o sensores inalámbricos.

2.2.3. Software para comunicación LoRa

Biblioteca *MKRWAN*

La librería Arduino *MKRWAN*, desarrollada por Arduino, permite al usuario controlar los procesos de comunicación LoRa. En esta librería se encuentra todo lo necesario para establecer una comunicación LoRa de manera efectiva.

Se definen los 2 métodos que un nodo puede utilizar para activarse: *OTAA* Y *ABP*. Como se ha comentado en la sección de los dispositivos LoRa, los nodos finales deben conocer su *devEui* para comunicarse con el servidor de red (y desde éste con el de aplicación), también deben contar con las distintas claves (*AppEUI*, *AppKey* en el caso de *OTAA*) para poder enviar su petición *join* al servidor de red.

La biblioteca *MKRWAN* contiene las funciones que permiten obtener el *devEUI* y realizar los procesos de activación, ya sea en modo *OTAA* o en modo *ABP*.

Esta librería también puede soportar cada una de las posibles configuraciones del nodo, brindando la posibilidad de desplegar el nodo en modo A, B o C.

Por último, contiene todo lo necesario para encriptar los mensajes en el formato preciso y enviarlos, para su posterior recepción en uno o varios *gateways*. Así mismo, podrá decodificar los paquetes recibidos, y con su contenido se podrá actuar sobre la placa Arduino. [Ard20b]

Gateway LoRa

Para la instalación del Gateway se utilizan las instrucciones y el código desarrollados por Charles Hallard, que se encuentra a disposición de todos los usuarios interesados en Github [Hal20b].

Se encuentra en esta publicación, una detallada descripción de todos los pasos a seguir para la puesta a punto de un *gateway* personal totalmente controlado por el usuario. Se explica cómo se deben configurar los distintos componentes hardware del *gateway* y cómo ejecutar la instalación.

Es preciso destacar que se deben omitir los pasos que conectan el *gateway* a *TTN*, para luego poder realizar una conexión sencilla al servidor de red propio.

Servidor de red y servidor de aplicación

Para desplegar los servidores de red y de aplicación se utilizará el servidor de código abierto de ChirpStack [Chi20a], cuyo autor es Orne Brocaar [Bro20].

Tal y como se indica en la documentación, el servidor ChirpStack está compuesto por tres elementos que se describen brevemente a continuación:

- *Chirpstack Gateway Bridge*: es el servicio responsable de convertir los paquetes en formato LoRa a un formato que el servidor de red pueda interpretar, este formato está basado en una combinación de Json [Jso20] y Protobuf [ION20].
- *Chirpstack Network Serve*r: es el encargado de realizar las funciones previamente descritas para el servidor de red (autentificación, comunicación con el servidor de aplicación, gestión de duplicados...).

- *Chirpstack Application Server*: sus labores también se han descrito con anterioridad, este segmento se encarga de procesar los datos provenientes de los nodos (o destinados a ellos). Este servidor ofrece una cómoda interfaz web para gestionar los usuarios, organizaciones, dispositivos... Y además permite integrar servicios externos a través de una API *RESTful*.

  Posee además funciones extra que lo dotan de una gran utilidad, algunos ejemplos son: la posibilidad de comprobar la cobertura de los distintos *gateway* de la zona o una interfaz donde se pueden analizar los distintos *frames* LoRa recibidos, de esta forma se puede visualizar el contenido encriptado de las tramas [Chi20c].

2.3. Tecnologías de desarrollo software

2.3.1. *Representational State Transfer (REST)*

REST es un conjunto de principios de diseño que especifican como deben ser desarrollados los sistemas distribuidos. Aporta ciertas limitaciones con el objetivo de asegurar que, si se cumplen en la aplicación distribuida, esta tendrá propiedades deseables, como pueden ser escalabilidad y modificabilidad. A continuación, se resumen brevemente los principios más importantes definidos en *REST* [Fie02]:

- La comunicación es sin estado: ninguna de los partes de la comunicación almacena información sobre la otra. Con este criterio se simplifican las comunicaciones entre cliente y servidor, y se alcanza un alto grado de escalabilidad.
- La arquitectura está basada en el modelo cliente-servidor: el objetivo subyacente es la separación clara entre las labores del cliente y el servidor, pero además se consigue una mayor portabilidad, se mejora la escalabilidad. También se posibilita que las partes puedan evolucionar de manera independiente.
- La interfaz es uniforme: al aplicar este principio se simplifica la arquitectura general y se mejora la visibilidad de las interacciones entre las partes de la comunicación.
- Los datos y las funciones se definen como recursos: toda información que pueda ser nombrada puede ser un recurso, de esta forma, todos aquellos elementos que puedan ser referenciados deben poder ajustarse en la definición de un recurso.
- Estos recursos son identificados mediante identificadores de recursos uniformes (*URI*) cada uno de estos identificadores señala de manera unívoca a un recurso. De esta forma se puede llegar a conformar un mapa de direccionamiento global. [Rod20].

2.3.2. Arquitectura modelo-vista controlador (MVC) y *Resource-oriented architecture* (ROA)

El Modelo-vista-controlador (MVC) es un patrón orientado a arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos [Uni20]:

- El modelo: contiene una representación de los datos que maneja el sistema y su lógica de negocio.
- La vista: define la interfaz de usuario, es decir, la información que se envía al cliente y los mecanismos que interaccionan con este.

- El controlador: actúa como intermediario entre las dos partes anteriores, se encarga de gestionar el flujo de información entre ellos, adaptando los datos a las necesidades de cada uno.

La metodología *Resource-Oriented Architecture* es un estilo de arquitectura de software que pretende conformar un método sistemático para desarrollar servicios en la web que se adhieran a los principios *REST*. En este Trabajo de Fin de Grado, la arquitectura *MVC* y la metodología *ROA* se combinan al añadir la capa de recursos. De esta forma, los tres componentes propios de la arquitectura *MVC* interaccionan con esta capa de recursos para lograr cumplir en todo momento con los requisitos *REST*[**TFGLuisCarlosParra**].

2.3.3. Servicios web

Un servicio Web se define un como un sistema de software designado para dar soporte a la interacción entre máquinas a través de una red. El servicio web realiza una tarea (o un conjunto de tareas) y durante su desarrollo, además del código que realiza las tareas que se esperan de él, se define lo necesario para interactuar con este (formatos de mensajes, protocolos de transporte y la ubicación) [Wor18].

Servicio web *RESTful*

Un servicio web *RESTful* es todo aquel servicio web que sigue los principios de diseño de *REST*. El servicio web tipo *RESTful* utiliza el protocolo HTTP, por esta razón sólo se soportan las operaciones definidas en este protocolo, las mas cruciales son GET, PUT, POST Y DELETE [**MRCIntrodREST**].

El *framework* FLASK

Flask es un "micro"*framework* escrito en Python y concebido para facilitar el desarrollo de aplicaciones web bajo el patrón MVC, previamente mencionado [Muñ20]. Este *framework* permite construir aplicaciones web y servicios web *RESTful* en lenguaje Python con extraordinaria facilidad. Presenta una conveniente curva de aprendizaje, de esta forma, un usuario con nula experiencia podría desarrollar, en un reducido período de tiempo, una aplicación web funcional (obviamente con mínimas prestaciones). Este *framework* sugiere estructuras y métodos, pero dota de total libertad al desarrollador, pues no obliga a utilizar ningún tipo de estructura o función fija. Es el desarrollador quien escoge las herramientas y librerías que quiere usar. Y, gracias al trabajo global de la comunidad, cada vez son más las extensiones que se pueden utilizar en este *framework*.

2.4. Válvulas de control de riego

Una electroválvula es un elemento cuya función resulta fundamental en circuitos que regulan el flujo de todo tipo de fluidos, sobre todo el agua.

Este dispositivo electromecánico, diseñado para controlar el flujo que circula por un conducto, está compuesto por dos partes básicas: el solenoide y la válvula. En algunos modelos, el primero convierte la energía eléctrica en mecánica y hace posible que se accione la válvula, mientras que otros cuentan con dos solenoides para hacer los movimientos de apertura y cierre.

El funcionamiento es sencillo: la membrana de la válvula se apoya en el cuerpo con la ayuda del muelle, y evita que el agua pase por la presión que ella misma ejerce y que está unificada tanto en la parte inferior como en la superior.

En el momento en que se envía una señal eléctrica al solenoide de la electroválvula, la bobina se imanta y levanta el émbolo. De esta manera, queda un pequeño agujero en la tapa de la válvula, por donde sale el agua de la cámara superior. Así, cambia la presión y se libera el orificio de paso general. Esto hace que se comuniquen la entrada y la salida de agua del cuerpo de la válvula.

Como se deduce, para abrir o cerrar una electroválvula es suficiente con generar un pulso eléctrico dirigido hacía el solenoide. Aunque existen diferentes modelos, en la mayoría de electroválvulas comerciales, los pulsos deben ser de unas decenas de milisegundos, y de una amplitud que varía en función del solenoide. Los valores más típicos para las dimensiones de los pulsos oscilan entre los 15 y 100 ms para la anchura, y los 9 o 12 voltios para la amplitud.

Capítulo 3

Análisis, diseño e implementación

3.1. Análisis

Como ya se ha comentado en la Introducción, este Trabajo de Fin de Grado se basa en el Trabajo de Fin de Grado realizado por Luis Carlos Parra [Par20].

Por esta razón, el sistema que se ha desarrollado comparte muchas de sus características con el sistema inicial desplegado en dicho TFG. Con el objetivo de ampliar la cobertura del sistema inicial, se ha modificado la tecnología que soporta la comunicación, sustituyendo WiFi por LoRa. Todos los resultados incluidos en la fase de análisis del sistema inicial siguen siendo válidos para el nuevo sistema, a excepción de aquellas diferencias que se indican durante las próximas secciones de esta memoria.

3.1.1. Descripción del sistema inicial

El sistema que se ha desarrollado en este Trabajo de Fin de Grado se denomina *Smartgranja*.

El objetivo de la aplicación *Smartgranja* es brindar al usuario la posibilidad de automatizar el control del riego en las explotaciones agrícolas. Es preciso mencionar que el usuario no debe tener conocimiento alguno sobre la tecnología subyacente en este sistema, el único requisito para la utilización de *Smartgranja* es disponer de un dispositivo (teléfono móvil, tablet, ordenador, etc.) con un navegador web y acceso a Internet.

Para lograr este cometido, el sistema, actuará controlando la activación y desactivación de las válvulas de riego, ofreciendo la oportunidad de actuar en todo momento sobre una o varias electroválvulas de manera inmediata, o mediante la definición de temporizadores programables según las necesidades de cada usuario. A mayores, la aplicación *Smartgranja* cuenta con un sistema de gestión de usuarios que permite limitar los accesos según que rol tenga asignado cada usuario. Los posibles roles son los siguientes:

- Rol de administrador: el usuario que ostente este rol podrá realizar cualquier operación sobre las válvulas, los temporizadores, o incluso el resto de los usuarios.
- Rol de acceso a válvulas y temporizadores: quien posea este rol podrá operar tanto sobre las válvulas como sobre los temporizadores, pero no podrá visualizar ni gestionar información sobre los demás usuarios.
- Rol de acceso a temporizadores: este rol presenta aún más limitaciones, en este caso, el usuario sólo podrá actuar sobre los temporizadores y también tendrá la posibilidad de visualizar información sobre las válvulas. No obstante, no podrá configurarlas (salvo en lo que respecta a sus temporizadores).

- Rol de lectura: este rol representa el caso de mayor restricción, el usuario sólo podrá consultar información sobre válvulas y temporizadores, pero no tendrá la opción de modificar su configuración.

3.1.2. Descripción del nuevo sistema

El nuevo sistema presenta muchas de las características y los métodos con los que ya contaba el sistema anterior. No obstante, la tecnología de comunicación deja de basarse en WiFi para desplegarse sobre LoRa, por esta razón es preciso introducir nuevos elementos adicionales en el sistema.

Estos elementos posibilitarán la comunicación LoRa, pero su introducción en el sistema no representa un obstáculo para el uso de otras tecnologías de comunicación, pues todas las decisiones han sido tomadas con el firme propósito de no interferir en el funcionamiento del sistema ya realizado. Se logra que ambas tecnologías de comunicación (LoRa y WiFi) convivan sin conflicto alguno dentro del sistema. Los elementos que se introducen podrían clasificarse en dos grupos: hardware y software. Los cambios en el hardware son la introducción del *gateway* y la sustitución de la placa Arduino MKR 1010 por la MKR 1300, estas modificaciones se detallan en secciones posteriores (3.2.2). Los cambios en el software son la introducción del servidor ChirpStack (ver sección 2.2.3) y la agregación de un servicio web intermediario, encargado de adaptar los mensajes del Servicio web *Smartgranja* a un formato apropiado para LoRa (ver sección 2.2.1).

3.1.3. Cambios en los requisitos

Todos los requisitos descritos en el Trabajo de Fin de Grado de Luis Carlos Parra [Par20] se aplican a este sistema.

Añadidos a estos requisitos ya existentes, se añade el requisito no funcional relativo a la cobertura extendida. Como este sistema debe ser capaz de cubrir áreas de tamaño elevado se utilizará una tecnología de largo alcance. Tal y como explica el estudio expuesto en el capítulo 2 (Estado del Arte), se decide que esa tecnología será LoRa.

La utilización de esta tecnología define y obliga a cumplir otros requisitos adicionales: los relativos a la ocupación de las frecuencias de las bandas libres, ya que todo sistema que haga uso de las bandas *ISM*[1] debe cumplir con las regulaciones definidas para cada región.

Esta aplicación, al no utilizar los servicios de TTN[2], no debe cumplir las regulaciones propias del servicio, pero sí debe limitar su comunicación a las normas de acceso justo establecidas por la Unión Europea. Estas limitaciones se basan en limitar el ciclo de trabajo, la potencia de transmisión o la tasa de datos [Sae+19].

Para el caso concreto de LoRa, el ciclo de trabajo debe ser menor del 1 por ciento, esto significa que de cada 100 unidades de tiempo se podrá ocupar la frecuencia durante 1. Para cumplir con esta restricción es preciso ser muy cauteloso con el tamaño de los paquetes a enviar, ya que los paquetes LoRa no deben sobrepasar los 64 bytes y deben contar con una cabecera de entre 13 y 28 bytes [Aug+16].

Por último, es preciso mencionar que en función de cuán lejos estén los nodos del *gateway*, mayor será el *spreading factor* asociado a la comunicación, por lo que el tiempo en el aire del paquete también será mayor, reduciendo el número medio de mensajes que se podrán enviar. Por suerte, existen calculadoras que ayudan a discernir los límites de una comunicación LoRa [Avb20].

[1] *Industrial, Scientific and Medical*, banda de frecuencias para usos industriales, científicos o médicos.
[2] *The Things Network*, red global y de código abierto para proveer de cobertura LoRa.

3.2. Diseño

El modelo de diseño elaborado por Luis Carlos Parra en su Trabajo de Fin de Grado [Par20] es válido para este sistema. No obstante, ciertos diagramas de casos de usos sufrirán modificaciones debido a la nueva arquitectura software que se definirá. Los diagramas de casos de usos que se verán alterados son aquellos que muestran interacción con las placas de control de válvulas y se explicarán en secciones posteriores.

3.2.1. Diseño del sistema inicial

Arquitectura hardware

La arquitectura hardware está compuesta de lo siguientes elementos:

- Un servidor cuya misión es alojar el servicio web al que accede el usuario a través de cualquier dispositivo que cuente con conexión a Internet y un navegador web.
- Los circuitos controladores serán los encargados de recibir los mensajes desde el servidor, y traducirlos en órdenes de control para actuar sobre las válvulas, ya sea inmediatamente, o con un calendario programado a través de la funcionalidad de los temporizadores.
- Las válvulas representan el nodo final de la comunicación, recibirán y ejecutarán las órdenes enviadas desde las placas controladoras y, en función de estas, permitirán o no, el flujo del agua.

En la figura 3.1, se muestran los elementos citados anteriormente unidos mediante líneas sólidas con o sin flechas.

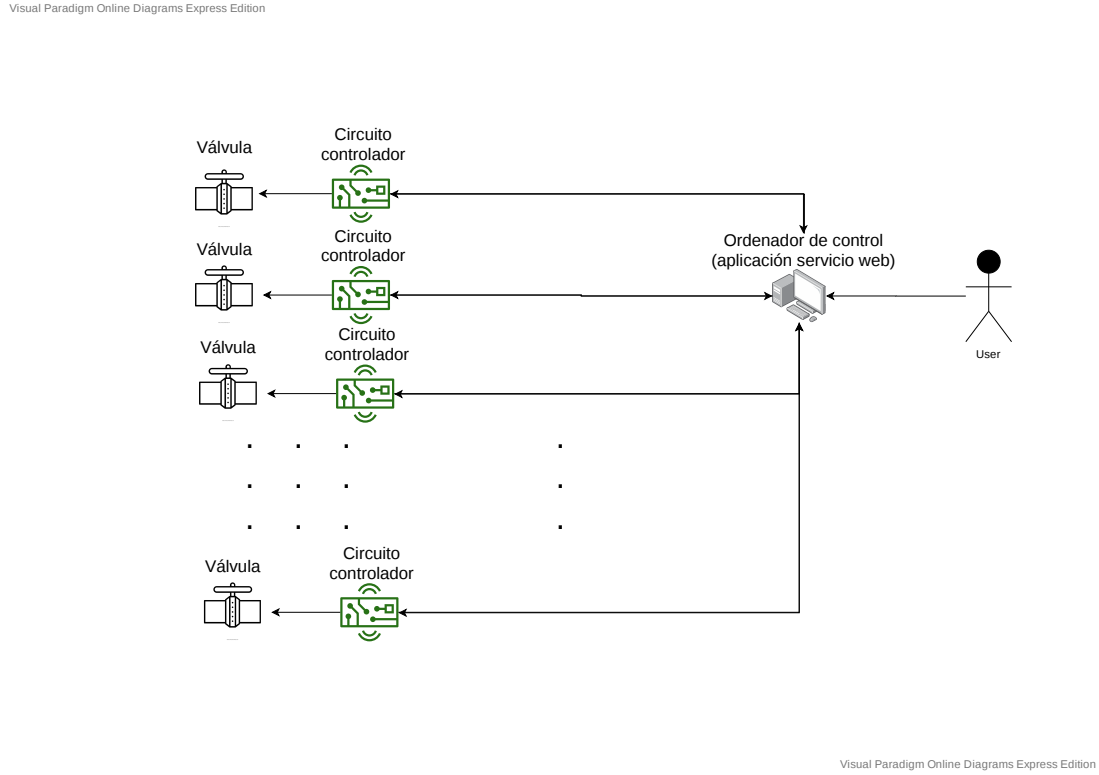

FIGURA 3.1: Esquema de arquitectura del hardware

El sentido de la flecha indica qué elemento de la arquitectura realiza las solicitudes (u operaciones sobre la otra parte) y qué componente las recibe. Por ejemplo, en la comunicación entre el circuito controlador y la válvula se puede observar cómo la flecha es de un único sentido, indicando que la comunicación siempre se produce en esa dirección y nunca en la otra. Por sencillez visual, en la imagen se muestra una válvula correspondiente a cada circuito controlador, pero se ha de tener en cuenta que cada uno de estos circuitos podrá gestionar las órdenes para una o varias válvulas.

Arquitectura software

La arquitectura software se podría definir como la descripción de los componentes de un sistema software y la relación existente entre ellos [Bus01].

Para garantizar que el acceso a la aplicación pueda realizarse desde cualquier dispositivo y desde cualquier lugar, se decide implementar una aplicación web.

Este tipo de aplicaciones son accedidas a través de un navegador web, por lo que cualquier dispositivo que tenga este tipo de aplicación de navegación garantizaría el acceso a la aplicación de control. Además, el diseño de la aplicación web ha tenido en cuenta las características especiales de los dispositivos móviles (por ejemplo, las reducidas dimensiones de su pantalla) para garantizar una correcta visualización en este tipo de dispositivos.

La aplicación web (que se ha desarrollado como un servicio web) se encuentra alojado en la máquina etiquetada como *Ordenador de control* en la figura 3.1. Este servicio web ofrece una interfaz de usuario para que este interactúe con las placas controladoras, etiquetadas en la figura 3.1 como *Circuito controlador*. El desarrollo software de este sistema inicial se divide en el despliegue de dos subsistemas, mostrados en la figura 3.2. Estos dos susbistemas son:

- Aplicación Arduino: se ejecuta en cada una de las placas controladoras y gestiona la recepción de mensajes desde el ordenador de control, para su posterior traducción a ordenes sobre la electroválvula.
- Aplicación de control *Smartgranja*: alojada en el ordenador de control y conformando un servicio web *RESTful*, es la encargada de ofrecer al usuario una interfaz de control y enviar las órdenes a las placas controladoras.

3.2.2. Cambios en el diseño del sistema inicial

Cambios en la arquitectura *hardware*

La arquitectura hardware del nuevo sistema es prácticamente igual a la presente en el Trabajo de Fin de Grado de Luis Carlos Parra [Par20], pero en esta ocasión se sustituye la placa Arduino MKR WiFi 1010 por la placa Arduino MKR WAN 1300, que cuenta, como se ha explicado anteriormente (ver sección 2.2.2), con los elementos necesarios para establecer una comunicación LoRa.

Como ambas placas pertenecen a la familia MKR (diseñadas para ser utilizadas en aplicaciones *IoT*), comparten una arquitectura muy similar, posibilitando que la sustitución de una por la otra sea tan sencilla como desconectar los pines conectados en la primera y conectarlos de la misma manera en la segunda. La única precaución que se debe tener es que la placa esté orientada correctamente al conectarla en el zócalo.

Esto demuestra que la elección de basar el nodo LoRa en la placa Arduino MKR WAN 1300 es la más conveniente para este caso, por su alto grado de portabilidad.

El circuito de control de cargas se mantiene exactamente igual que en la versión inicial.

Además, al establecerse una comunicación LoRa, un nuevo elemento aparece en la arquitectura. Se trata del *gateway* LoRa (ver sección 2.2.1) que se encarga de recibir y enviar los

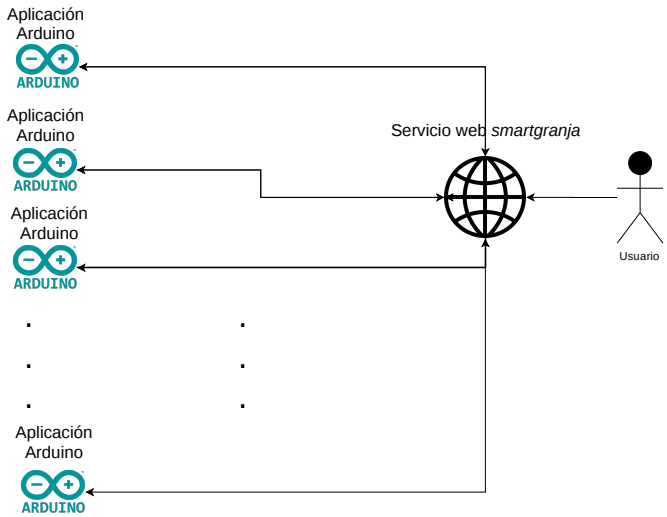

FIGURA 3.2: Esquema de la arquitectura software del sistema inicial

paquetes en formato LoRa. Este dispositivo se comunica con las placas controladoras y es responsable (entre otras tareas) de la traducción de los paquetes al formato LoRa (ver sección 2.2.3).

En figura 3.3 se puede observar la arquitectura *hardware* del nuevo sistema.

Cambios en la arquitectura *software*

Gran parte de la arquitectura software descrita en la versión del sistema anterior se mantiene, mas con el objetivo de reutilizar la aplicación de control *Smartgranja* sin tener que modificar su código, se debe generar un nuevo elemento que actúa como un intermediario en la comunicación.

Este intermediario, al que se denomina *Servicio web Intermediario*, se encargará de adaptar los mensajes enviados desde la aplicación *Smartgranja* al formato adecuado para su posterior recepción en lo aplicación Arduino (alojada en las placas Arduino que representan nodos LoRa). El servicio web Intermediario se desarrolla en el lenguaje de programación python, usando la biblioteca *flask*, y es un servicio web tipo *RESTful*.

A continuación, se describe cómo discurre el proceso de comunicación para clarificar el papel de cada uno de los elementos involucrados, el proceso es análogo independientemente de quien inicie la comunicación (un esquema de este proceso puede verse también en la figura 3.4).

1. El usuario genera una orden a través del servicio web *Smartgranja*. Esta orden se envía al servicio intermediario.

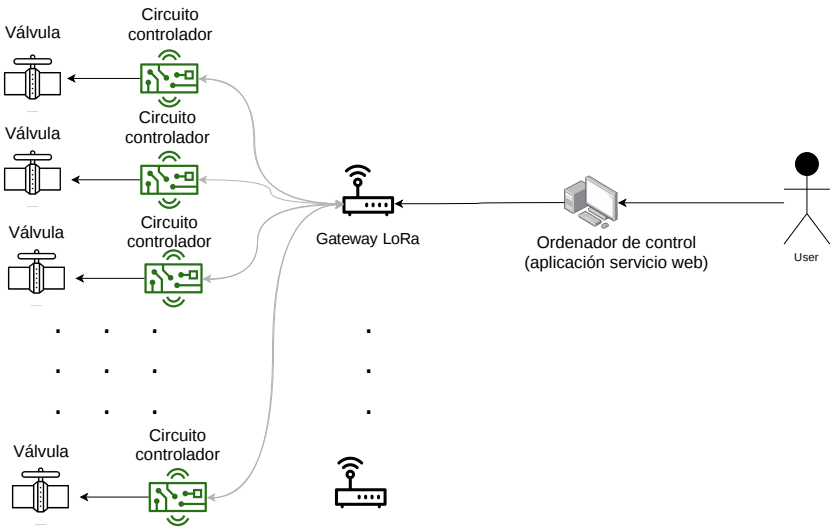

FIGURA 3.3: Esquema de la arquitectura *hardware* del nuevo sistema

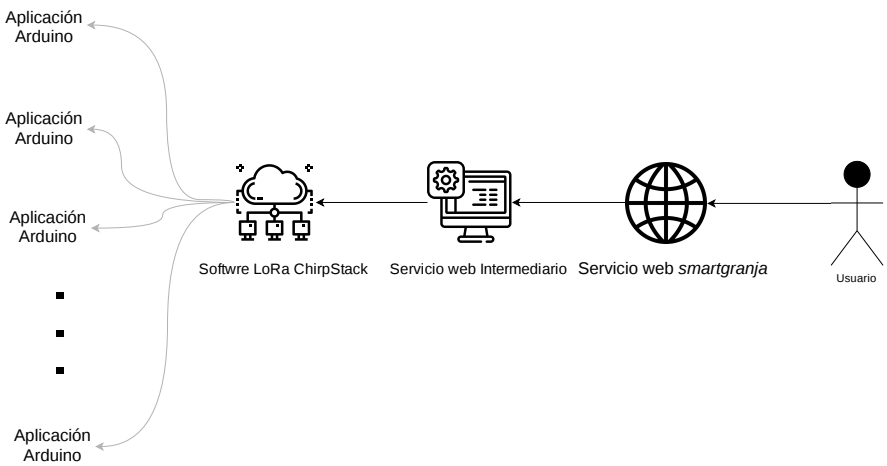

FIGURA 3.4: Esquema de la arquitectura *software* del nuevo sistema

2. El servicio intermediario recibe esa orden, la analiza y la procesa con dos objetivos: reducir el tamaño lo máximo posible, y traducir los campos para que la aplicación Arduino interprete correctamente las instrucciones.

3. El servicio intermediario envía el paquete ya procesado al servicio web ChirpStack (en formato B64)

4. El servicio web Chirpstack utiliza su API propia para enviar esa trama a la aplicación Arduino (alojada en la placa Arduino que es previamente registrada en el servicio web Chirpstack)

5. La aplicación Arduino (adaptada convenientemente para la comunicación LoRa) recibe la trama, interpreta su contenido y ejecuta las instrucciones asociadas a la operación solicitada (en algunas de las operaciones se enviará también una respuesta, que seguirá el camino inverso al de la petición).

3.2.3. Cambios en los diagramas de casos de uso de diseño

Caso de uso de creación de válvula

Este diagrama (ver figura 3.5) se ha modificado para incluir a los dos nuevos elementos presentes en la comunicación: el servicio web intermediario y el servicio web Chirpstack.

El diagrama muestra como el servicio web Chirpstack recibe los datos enviados desde la aplicación Arduino LoRa (etiquetada con un nombre que representa a la placa controladora que la contiene) ejecuta un POST sobre el servicio web intermediario para enviarle los datos generados por la aplicación Arduino.

En el diagrama se puede ver como el servicio web intermediario ejecuta un POST en el que la dirección URL contiene el campo *globalValveId*, el intermediario genera ese campo con la concatenación del *boardValveId* y del *valveId* recibidos desde el servicio web ChirpStack.

El servicio web *Smartgranja* actualiza la base de datos mediante el método *CreateValve* y devuelve un OK al intermediario. Se puede observar que el OK no regresa hasta la aplicación Arduino LoRa, esta es una decisión meditada pues es necesario reducir las comunicaciones debido a las limitaciones de la tecnología LoRa.

Caso de uso de modificación de válvula

Como se puede observar en el diagrama 3.6, se han incluido los dos nuevos elementos en la comunicación. Este caso es distinto al anterior, pues representa el sentido de comunicación más habitual: desde el servicio web *Smartgranja* hacia la aplicación Arduino LoRa.

En este caso, se ejecuta un PUT sobre la clase recurso *Valve* del servicio web intermediario, este método identifica la operación (modifyValve en este caso) que se ejecutará en la aplicación Arduino LoRa.

Se empaqueta el identificador del método en su versión reducida "mv", el identificador de válvula, y el campo *open* que nos indica a que estado debe cambiar la válvula. Con este formato comprimido se ha conseguido reducir de manera drástica el tamaño del mensaje para adecuarlo a las necesidades de LoRa.

Este paquete se envía al servicio web Chirpstack (utilizando una vez más el *boardId* obtenido tras el procesado del *globalValveId*) y, finalmente, el servicio ChirpStack envía la trama a la aplicación Arduino LoRa.

Por último, si todo es correcto, se envía la notificación del OK hasta el usuario (con la inserción previa del nuevo estado de la válvula en la base de datos), si falla la ejecución, el usuario recibe la notificación del error.

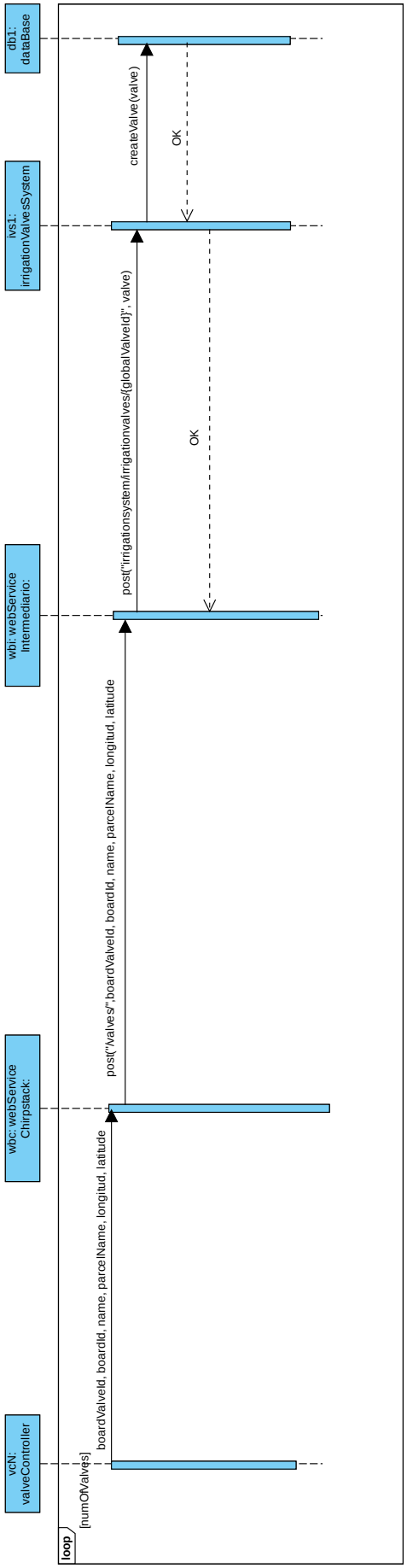

FIGURA 3.5: Caso de uso de creación de la válvula

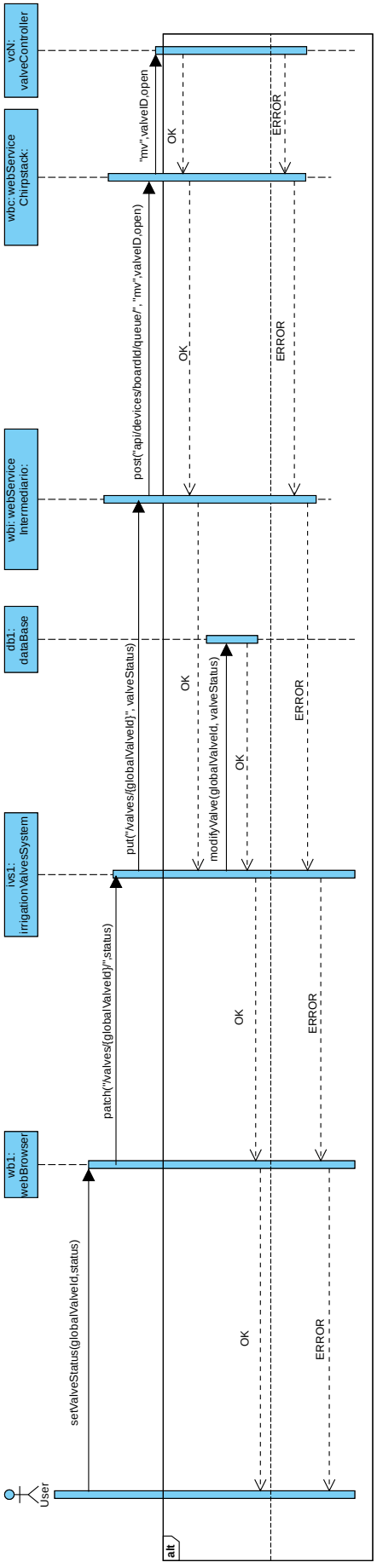

FIGURA 3.6: Caso de uso de modificación de válvula

Caso de uso de borrado de válvula

Este caso es muy parecido al anterior, el proceso se realiza de manera análoga con la excepción de que, una vez borrada la válvula de la base de datos, se pide la lista de temporizadores asociados a esa válvula para proceder a su eliminación (ver figura 3.7).

Una vez terminado este proceso de borrado, se notifica al usuario el éxito en el borrado.

Caso de uso de creación de temporizador

Este caso de uso refleja un procedimiento muy parecido a los dos anteriores.

Se incluyen, una vez más, los dos nuevos elementos de la comunicación (ver figura 3.8). Los procesos de envío de mensajes, comprobación de su contenido y reducción de su tamaño son análogos a los anteriormente descritos.

En este caso, la aplicación Arduino LoRa, devuelve un identificador de temporizador, que será guardado en la base de datos, tras completar este proceso, se mandará la notificación de OK hasta el usuario, en caso de que todo se haya realizado correctamente.

Caso de uso de modificación de temporizador

Este diagrama (ver figura 3.9) representa la modificación de un temporizador, el procedimiento es análogo al del esquema anterior.

Caso de uso de borrado de temporizador

Una vez más, los procesos implicados en este caso de uso (mostrados en la figura 3.10) son similares a los ya mencionados anteriormente.

3.3. Implementación y pruebas

3.3.1. Servicio web intermediario

En el sistema anterior, la placa Arduino contaba con conexión a la red a través de WiFi, y gracias a esta característica, la aplicación web de Arduino se desarrolló como un servicio web *RESTful* (dado que la cantidad de información intercambiada con la placa no estaba limitada).

En este sistema, la comunicación con la aplicación Arduino ser realiza mediante la tecnología LoRa, por lo que no se puede implementar un servicio web de ese tipo dadas las limitaciones en cuanto a tamaño de los datos y velocidad de transmisión motivadas por las restricciones en el uso de canal que impone LoRa.

No obstante, con el objetivo de no modificar el funcionamiento del servicio web Smartgranja y de adaptar la aplicación Arduino a la comunicación LoRa con el menor número de cambios posibles en el software existente, se crea un servicio web intermediario.

Este servicio web intermediario es un servicio tipo *RESTful*, y actúa como intérprete entre el servicio web *Smartgranja* y la aplicación Arduino LoRa. Es preciso recalcar que este servicio web intermediario no traduce la información en el formato empleado por el protocolo IP al formato utilizado por la tecnología LoRa (esa es una de las misiones del servidor de red, según se explica en la sección 2.2.3), pero sí procesa los mensajes intercambiados entre estos dos elementos para ajustar su contenido a los requisitos de la tecnología LoRa.

Uno de los cometidos del servicio intermediario es reducir el tamaño de los mensajes dirigidos a la aplicación Arduino LoRa: como ya se ha comentado, una de las limitaciones más importantes de la tecnología LoRa es que los paquetes intercambiados deben tener el menor tamaño posible, por eso, el intérprete se encarga de recibir los mensajes enviados desde

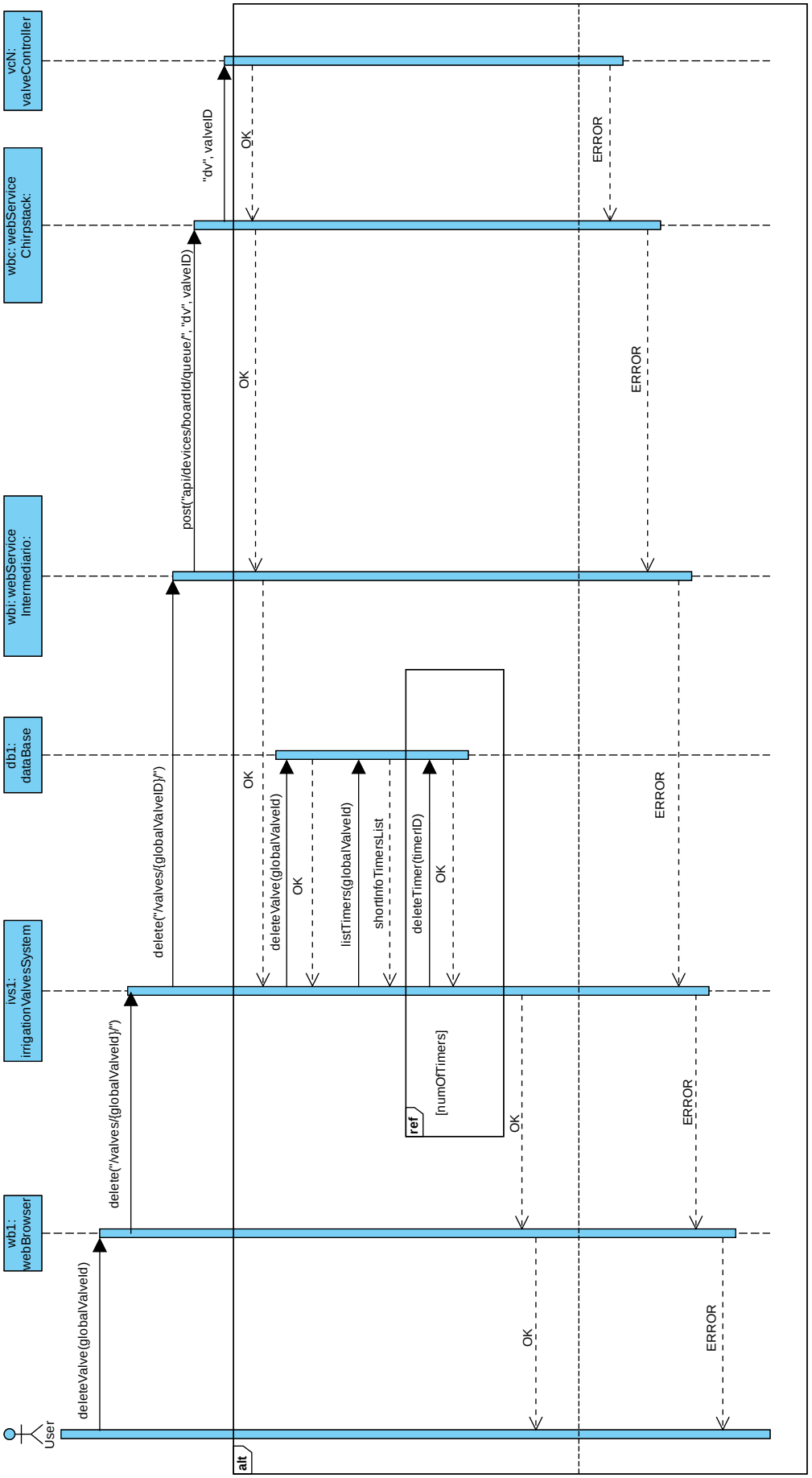

FIGURA 3.7: Caso de uso de borrado de válvula

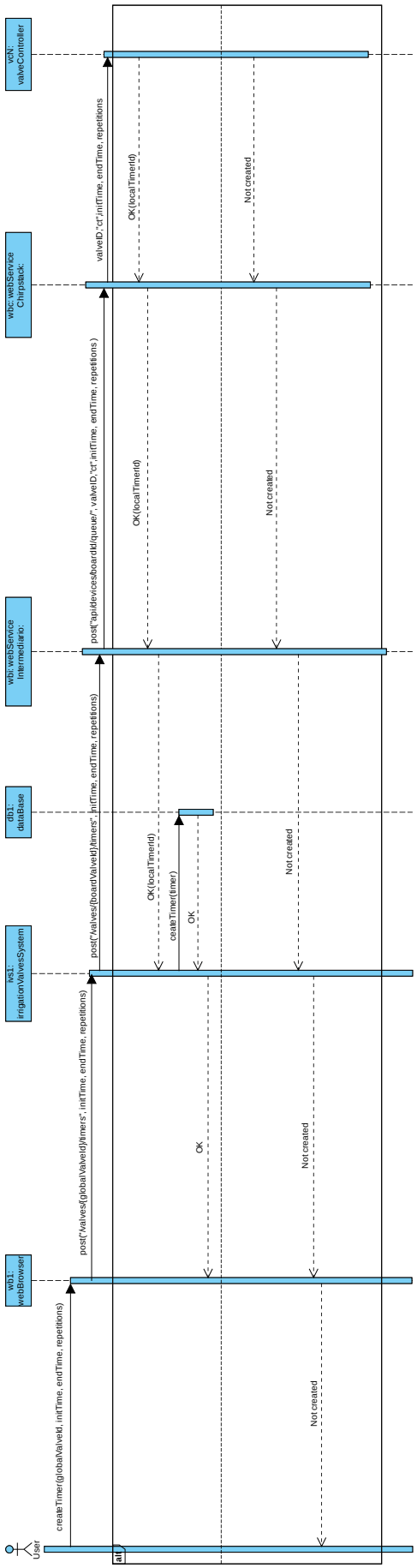

FIGURA 3.8: Caso de uso de creación de temporizador

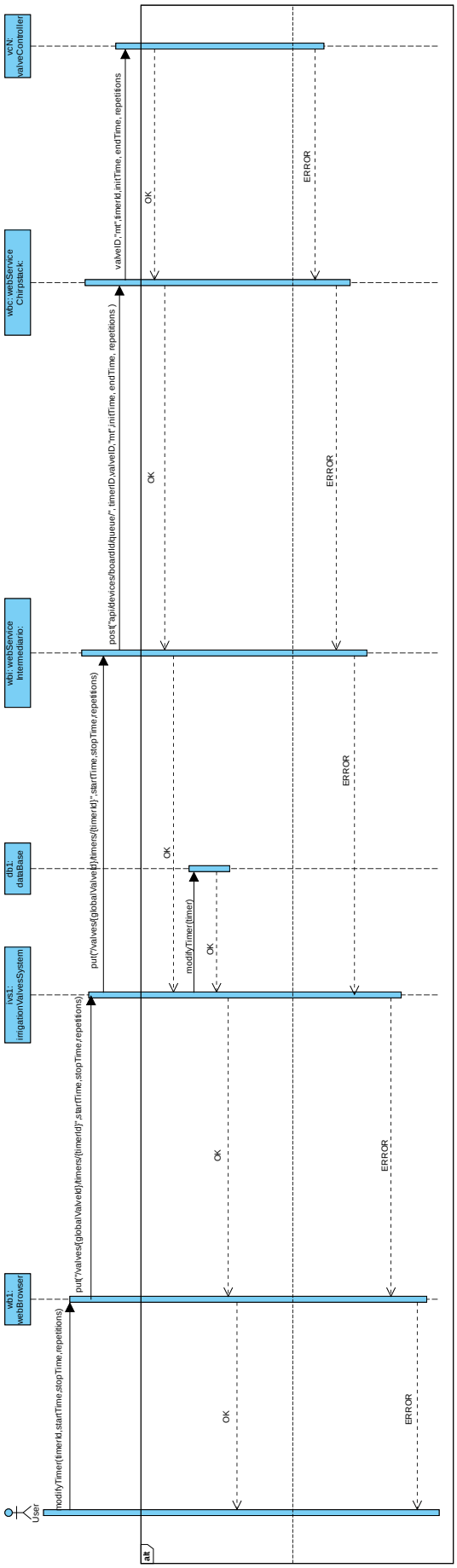

FIGURA 3.9: Caso de uso de modificación de temporizador

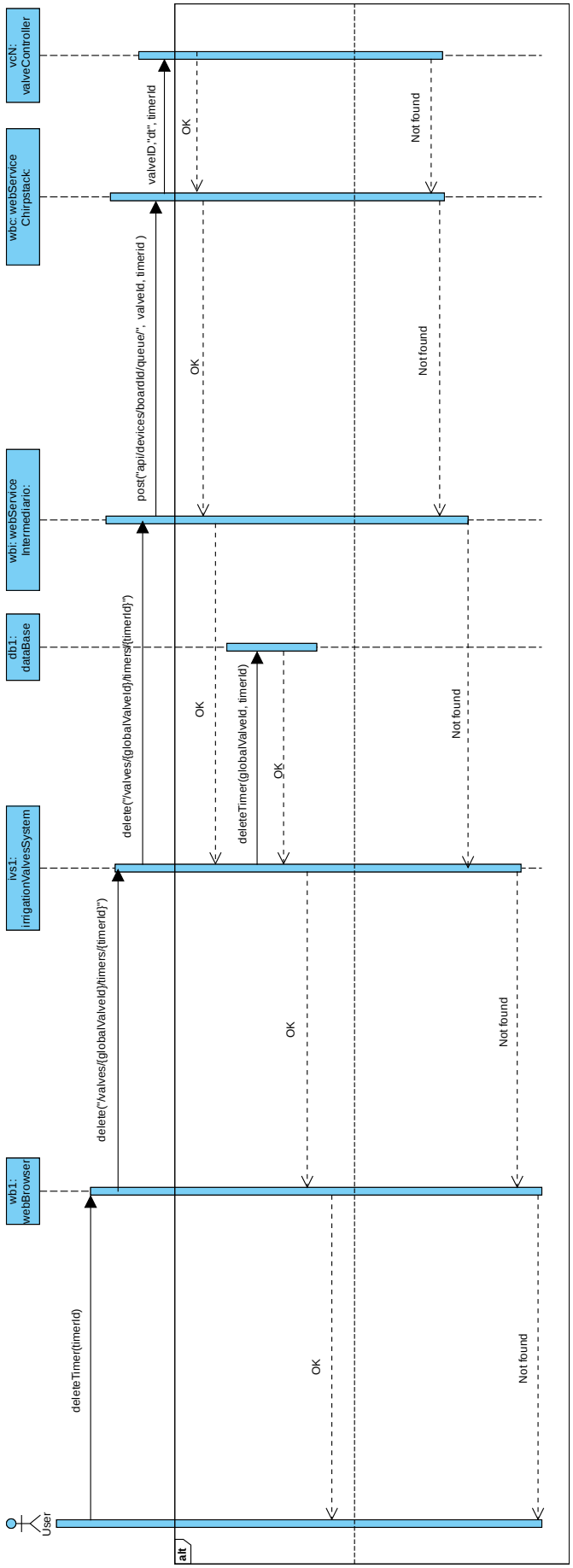

FIGURA 3.10: Caso de uso de borrado de temporizador

el servicio web *Smartgranja*, interpretar su contenido y, en función de esa interpretación, generar un nuevo mensaje con unas dimensiones menores y que pueda ser interpretado por la aplicación Arduino LoRa.

El procedimiento es análogo para ciertos mensajes que se mandan desde la aplicación Arduino LoRa al servicio *Smartgranja*: se da un caso en el que el mensaje que el servicio web *Smartgranja* debe recibir tiene unas dimensiones mayores de las que LoRa puede soportar, por esta razón, el contenido del mensaje se manda dividido en paquetes de tamaño igual o inferior al máximo tamaño soportado por LoRa, y se reensambla en el servicio web intermediario para su posterior envío a *Smartgranja*. Este proceso se describe en la sección 3.3.2.

No conviene olvidar que en estos dos procesos de comunicación también está implicado el servicio web Chirpstack, tal y como se indica en la sección 3.2.2.

A continuación, se describen las clases y métodos que componen el servicio web intermediario, se incluye el diagrama de recursos para facilitar su comprensión (ver figura).

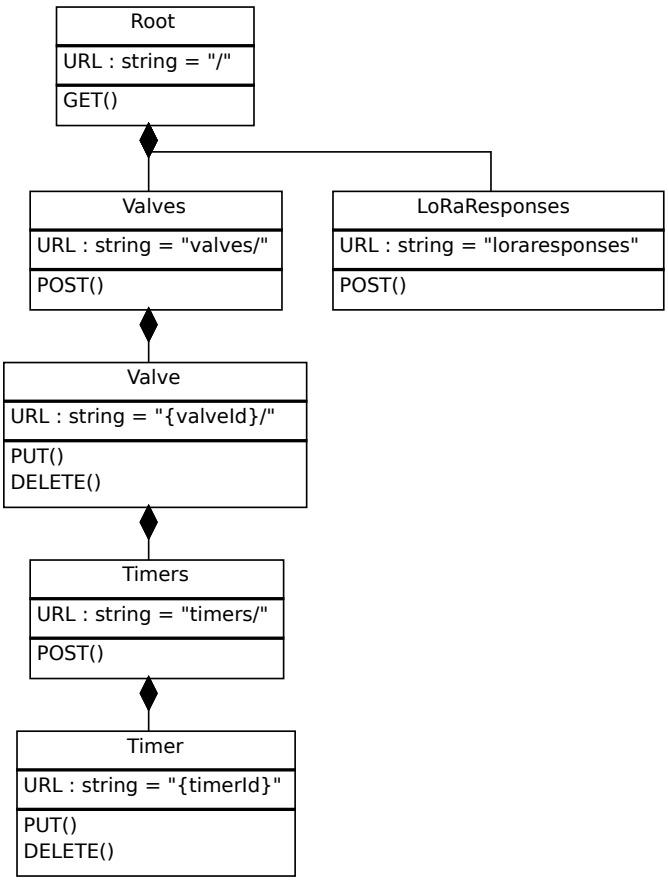

FIGURA 3.11: Diagrama de recursos del servicio web intermediario

- En primer lugar, existe una clase raíz del árbol puesto que hay dos recursos que no guardan relación entre sí: *Valves* y *LoRaResponses*. Esta clase cuenta con un método GET que devuelve los nombres de los recursos hijos directos de ellas y sus URLs, tal y como debe operar un recurso raíz en un modelo de recursos para garantizar la característica de navegabilidad de REST.
- El recurso *Valves* posee un método POST, que permite la creación de nuevas válvulas.

- El recurso *LoRaResponses* tiene un método POST que se utiliza para recoger las respuestas de la aplicación Arduino LoRa. Estas respuestas podrían corresponder a solicitudes sobre cualquiera de los recursos del servicio intermediario, por lo que lo más conveniente es la definición de un recurso que recoja todas estas solicitudes generadas en la aplicación Arduino LoRa.
- Del recurso *Valves*, nace un recurso hijo *Valve*, que contiene los métodos PUT y DELETE, estos métodos permiten la modificación de válvulas y el eliminado de las mismas.
- El recurso *Valve* es padre a su vez del recurso *Timers*, que soporta el método POST para la creación de nuevos temporizadores.
- Y finalmente, el recurso *Timer* (hijo del recurso *Timers*), tiene los métodos PUT y DELETE para la modificación y borrado de temporizadores.

A continuación, se describe cómo se desarrollarían las tareas de un método concreto de una clase recurso, el recurso *Timers* recibirá una petición POST para crear un nuevo temporizador asociado a una válvula:

1. Se recibe desde *Smartgranja* una petición POST sobre el URL `valves/{globalValveId}/timers/` cuyo cuerpo contiene los siguientes datos en formato JSON:

   ```
   {"startTime":"10:00:00", "stopTime":"11:15:00", "repeatDays": 5}
   ```

   (el formato usado para la hora de apertura y cierre de válvula es HH:MM:SS, donde HH son los dos dígitos de la hora, MM los dos dígitos de los minutos, SS los dos dígitos de los segundos. Es preciso aclarar que el último valor de los datos en formato JSON, representa una secuencia de bits que indica en qué días de la semana se repite el intervalo de temporización, representando el bit 0 al lunes.

2. El método realiza un tratamiento sobre `globalValveId` para separar el identificador de placa (`boardValveId`) del identificador de válvula local dentro de una placa (`valveId`).

3. Prepara un paquete cuyo contenido es

   ```
   "ct",valveId,"10:00:00","11:15:00",5
   ```

4. Lo envía hacía el nodo LoRa a través del servidor Chirpstack, para eso emplea el método POST de la api de Chirpstack, con la siguiente URL `/api/devices/{boardValveId}/queue/`.

5. La aplicación Arduino LoRa recibe el paquete, lo procesa (tal y como se describe en la sección 3.3.2) y manda la respuesta hacia el servicio Chirpstack.

6. El servicio Chirpstack reenvía la respuesta hacía el servicio intermediario, la recepción se realiza a través del método POST del recurso *LoRaResponses*, que enviará la respuesta hacia el servicio web *SmartGranja*

3.3.2. Aplicación arduino basada en LoRa

En la sección precedente se explica que la aplicación Arduino del sistema anterior se comportaba como un servicio web *RESTful*, se detalla que ese comportamiento no es posible (o al menos no de manera directa) ya que la placa Arduino LoRa solamente tiene acceso directo a una red LoRa donde existen limitaciones mucho más restrictivas en el tamaño de los paquetes y ancho de banda disponible que en una red IP.

Igual que se crea un servicio web intermediario para procesar y adaptar los mensajes a los requisitos de LoRa, es preciso realizar unas modificaciones en la aplicación Arduino para dotarla de la capacidad de procesar y entender los mensajes recibidos. Además, debe ser capaz de transmitirlos usando el formato de mensajes reducidos que se ha diseñado para la comunicación entre el servicio web intermediario y la propia aplicación Arduino.

El formato implementado para reducir el tamaño de los paquetes en la comunicación se ha diseñado teniendo siempre presente la premisa de enviar única y exclusivamente el contenido necesario. Todos los mensajes que se reciben en la aplicación Arduino LoRa presentan la misma estructura:

- El primer parámetro señala la operación que se va a ejecutar, con tan sólo dos sencillos caracteres se pueden identificar todos los métodos disponibles en la aplicación Arduino LoRa. De esta forma, para ejecutar una operación sobre las válvulas se podrá recibir: mv (modificar el estado de la válvula) o dv (borrar una válvula). De manera análoga, las operaciones sobre los temporizadores podrán ser: ct (crear un temporizador), dt (borrar un temporizador) o mt (modificar un temporizador).
- El segundo parámetro es el identificador de la válvula siempre. Es un número entero.
- Los siguientes campos siempre representarán los datos necesarios para llevar a cabo las operaciones. Por ejemplo, para crear un temporizador se recibirán los campos: hora de apertura de válvula, hora de cierre (ambas en el formato HH:MM:SS) y el número que indica en qué días se repite esa temporización.

Todos estos campos siempre se reciben en el mismo orden y separados por comas, de esta forma, es sencillo para la aplicación Arduino LoRa separar e identificar cada elemento.

Para que la aplicación Arduino LoRa pueda recibir mensajes, debe, en primer lugar, realizar ciertos procesos que le permitan configurarse como un nodo LoRa, a continuación se describen los pasos necesarios para lograr esta configuración:

- Se utilizan las funciones ofrecidas por la biblioteca MKRWAN (ver sección 2.2.3) para que el nodo se active en modo OTAA (ver sección 2.2.1).
- Se configura el nodo en modo C (ver sección 2.2.1) para que escuche de manera continua.
- Se activa la característica ADR (ver sección 2.1.4).

Una vez configurada la placa para que se comporte como un nodo LoRa, es preciso adaptar el proceso de recepción e interpretación de los paquetes recibidos:

- Utilizando una vez más las funciones ofrecidas por la librería MKRWAN, se traduce el contenido de los paquetes (codificación B64 a codificación ASCII).
- A continuación, se procesan los mensajes recibidos (que siempre presentan el formato descrito anteriormente), identificando tanto el método de la aplicación Arduino sobre el que se ejecuta la petición, como sus argumentos.

- Una vez ejecutado el método, la aplicación Arduino LoRa inserta la respuesta en un paquete LoRa y lo envía al servicio intermediario (a través del servicio Chirpstack) para que este transmita la respuesta al usuario a través del servicio *Smartgranja*.

Hay un caso en el que es preciso incidir, se trata del arranque de válvulas: cuando se enciende la placa, bien porque sea necesario sustituir la batería o bien porque se instale una nueva electroválvula, esta debe quedar registrada en el servicio *Smartgranja*.

Para lograrlo, la aplicación Arduino debería enviar los datos de la válvula en un paquete LoRa. Mas no es posible integrar todos los datos en un único paquete (ya que excedería el tamaño máximo soportado por LoRa), así que la aplicación LoRa envía un único dato en cada paquete, y repite esta operación hasta completar la transmisión de todos los datos necesarios para registrar una válvula en la aplicación web *Smartgranja*.

Tal y como se ha comentado en la sección anterior, es el servicio web intermediario el encargado de reensamblar los fragmentos para recuperar el mensaje completo y enviarlo, asegurando su integridad, a *Smartgranja*.

Se puede observar, tras el análisis del servicio web intermediario y de la aplicación Arduino LoRa, que con estas adaptaciones se consigue adecuar de manera satisfactoria la comunicación a la tecnología LoRa, y además, este proceso es totalmente transparente para el sevicio web SmartGranja (que inicialmente solamente soportaba placas Arduino con comunicación mediante WiFi), por lo que los dos sistemas podrían coexistir en una misma explotación de manera transparente.

Capítulo 4

Conclusiones y líneas futuras de trabajo

4.1. Conclusiones

En primer lugar, se ha completado un estudio exhaustivo de las diferentes tecnologías candidatas a ser el soporte de la comunicación del nuevo sistema, gracias a este proceso, la decisión de basar las comunicaciones en LoRa está totalmente justificada, además se ha comprobado que es la tecnología óptima para lograr los objetivos de este Trabajo de Fin de Grado.

En segundo lugar, tras haber llevado a cabo las fases de análisis, diseño e implementación, se ha conseguido adaptar de manera satisfactoria el sistema inicial, esta adaptación se ha realizado teniendo en cuenta las limitaciones de LoRa y con la firme objetivo de que el sistema basado LoRa no interfiera con el sistema basado en WiFi, estos dos objetivos se han cumplido y los dos sistemas podrían funcionar, en un mismo entorno, de manera simultánea y sin interferencias.

4.2. Líneas futuras de trabajo

A continuación, se proporcionan sugerencias para la elaboración de un estudio posterior en respuesta a este Trabajo de Fin de Grado.

- Sería posible, con unas adaptaciones mínimas, añadir sensores de humedad y temperatura en el terreno. Los datos recogidos por estos sensores se utilizarían para automatizar aún más este sistema, permitiendo al usuario definir los riegos en función de los valores de humedad y temperatura.

  De esta forma, se ofrecerían dos opciones: un modo programado (a través de los temporizadores), y un modo automático (en función de los valores de humedad y temperatura).

- Podría añadirse al sistema un módulo GPS, cuyos datos permitirían al usuario conocer la posición de las válvulas en todo momento, esta opción resultaría muy interesante para aquellos dispositivos de riego móviles (como pueden ser los *pivots*), ya que se podría llegar incluso a reflejar su avance en un mapa, mostrando de manera gráfica el recorrido realizado por el dispositivo de riego móvil.

- Una opción interesante sería aumentar aún más la autonomía de las baterías, se podrían añadir placas solares que recarguen paulatinamente las baterías durante las horas de luz.

Apéndice A

Documentos de diseño

A.1. Requisitos funcionales

A.1.1. Definición de actores

- User: Actor que representa a la persona que vaya a utilizar el sistema.
- Arduino: Actor que representa a una de las partes del sistema (controlador terminal). Este actor actúa como tal únicamente contra el servicio web.

A.1.2. Diagrama de casos de uso

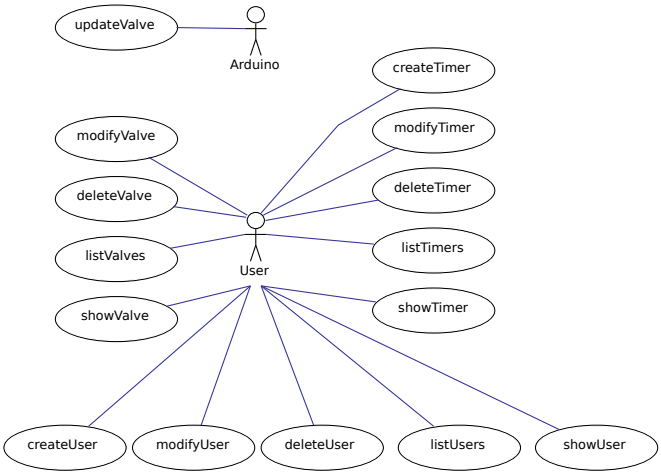

Figura A.1: Diagrama de casos de uso del sistema

A.1.3. Casos de uso del sistema

Creación de usuario

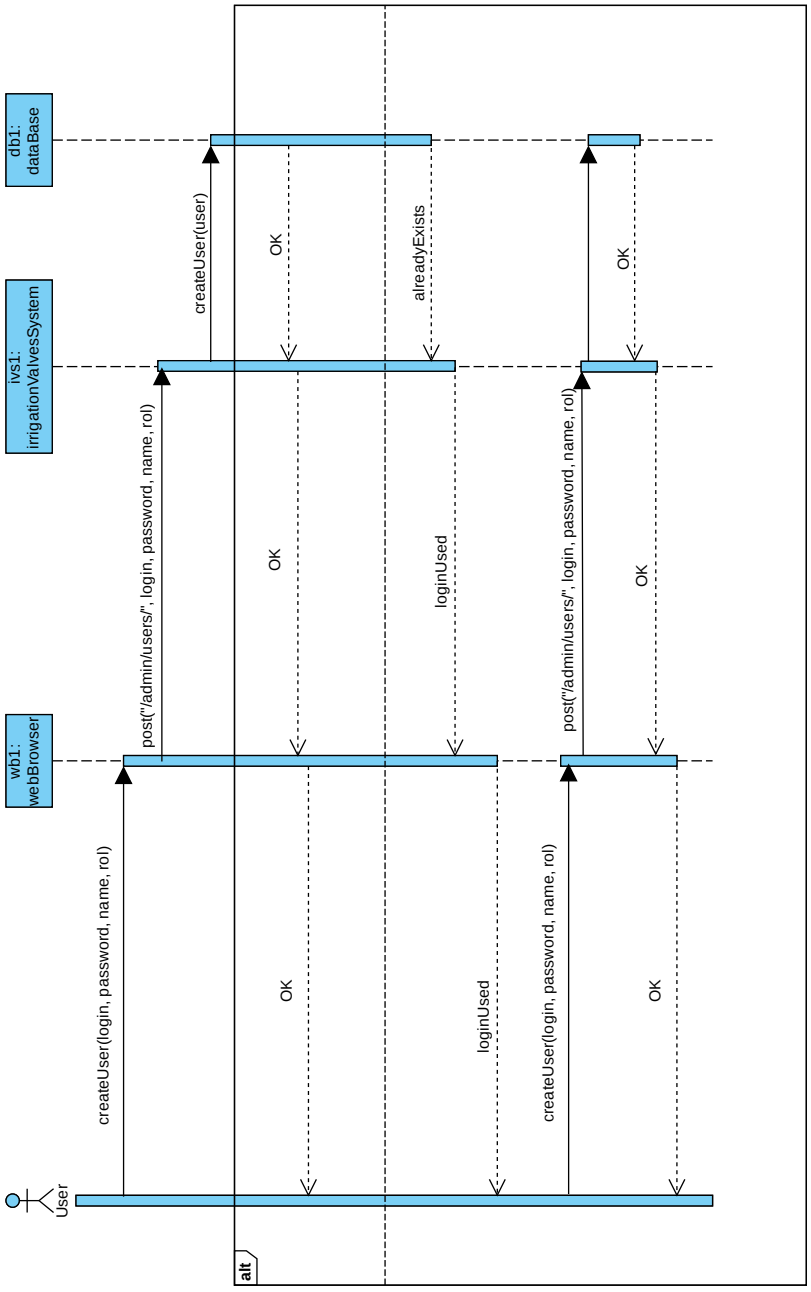

FIGURA A.2: Caso de uso de creación de los usuarios

Listado de usuarios de la aplicación

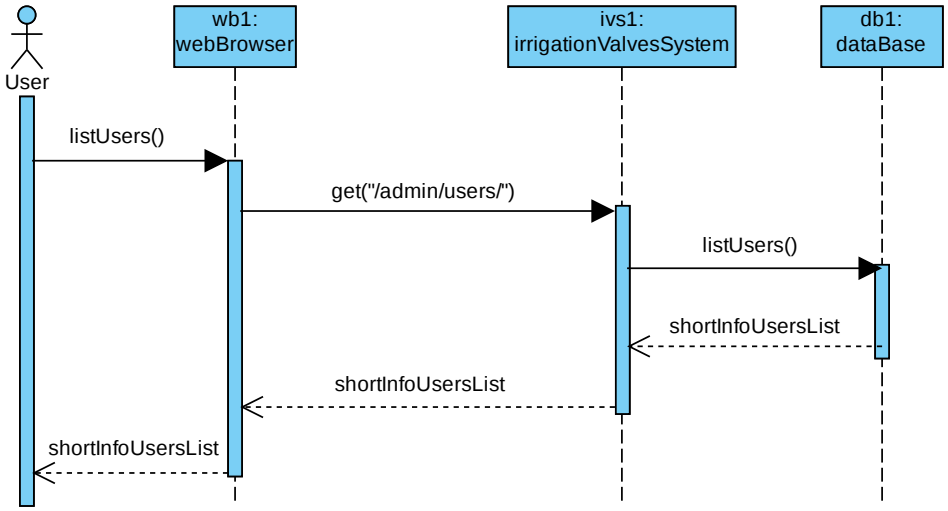

FIGURA A.3: Caso de uso de listado de los usuarios

Visualización de usuario

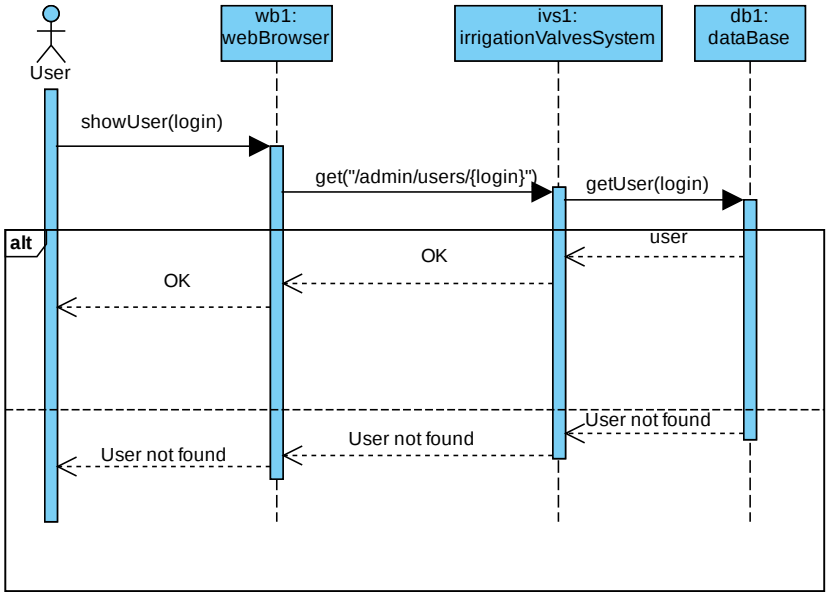

FIGURA A.4: Caso de uso de visualización de los usuarios

Modificación de usuario

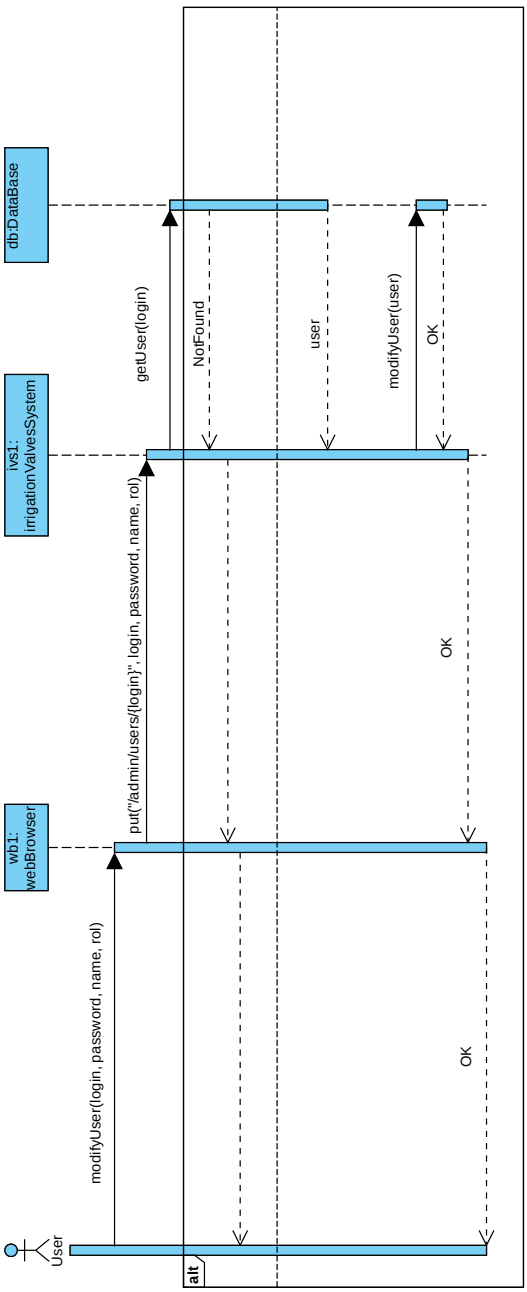

FIGURA A.5: Caso de uso de modificación de un usuario

Borrado de usuario

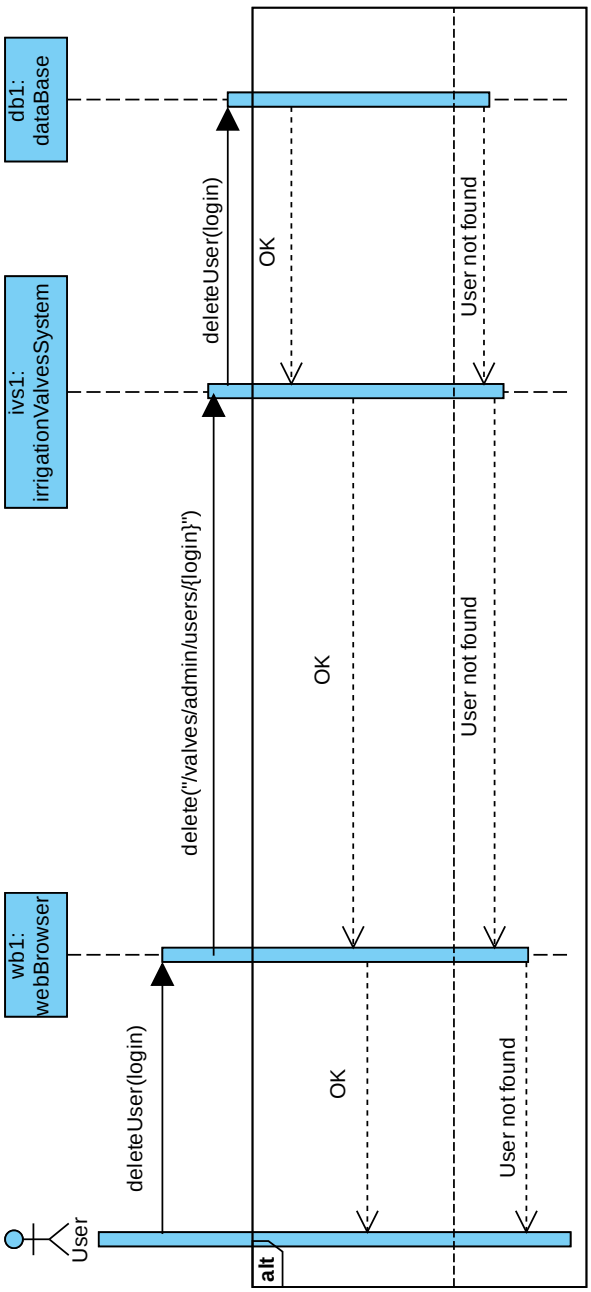

FIGURA A.6: Caso de uso de borrado de los usuarios

Creación de válvula

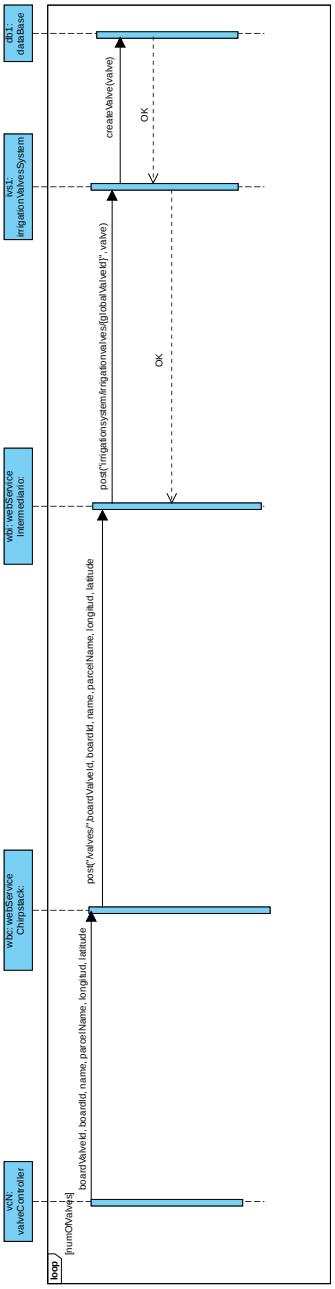

FIGURA A.7: Caso de uso de creación de la válvula

Listado de válvulas

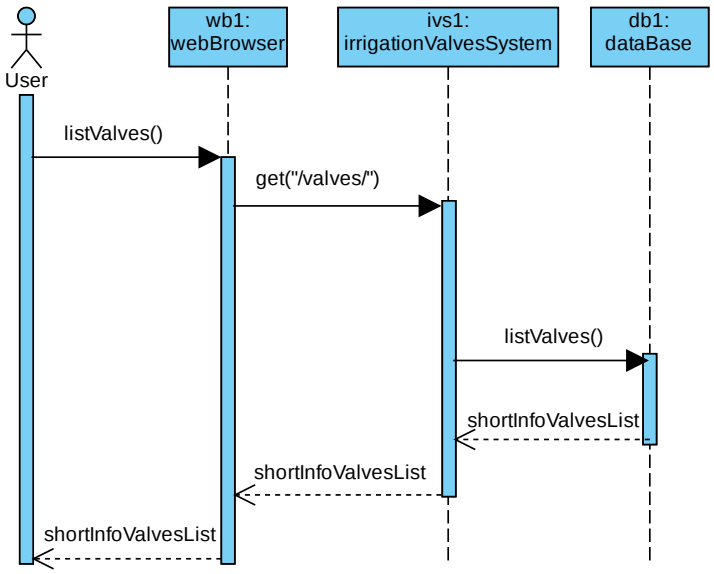

FIGURA A.8: Caso de uso de listado de las válvulas

Visualización de la válvula

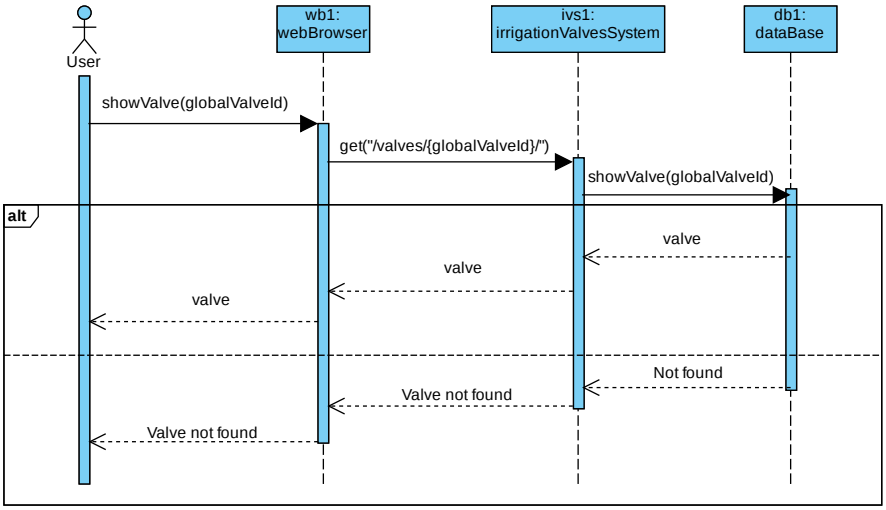

FIGURA A.9: Caso de uso de visualización de la válvula

Modificación de válvula

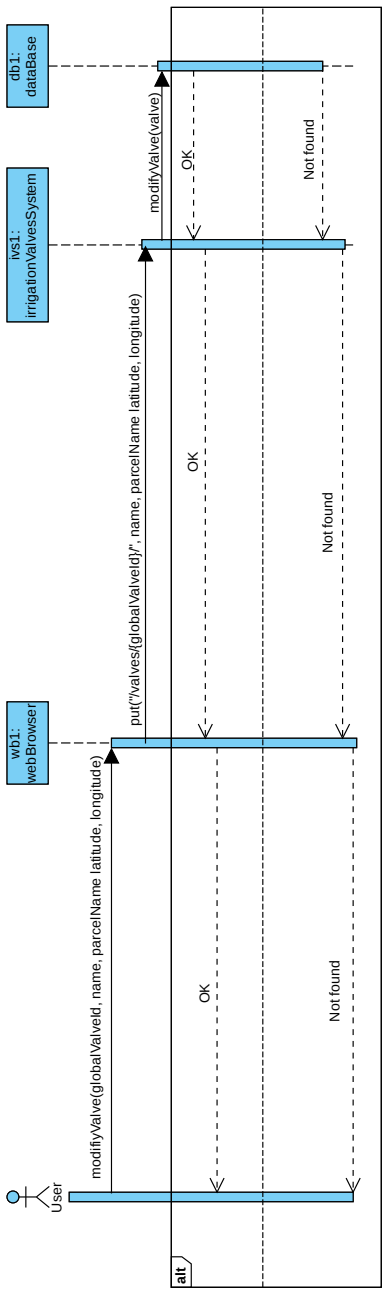

FIGURA A.10: Caso de uso de modificación de válvula

Apertura y cierre de válvula

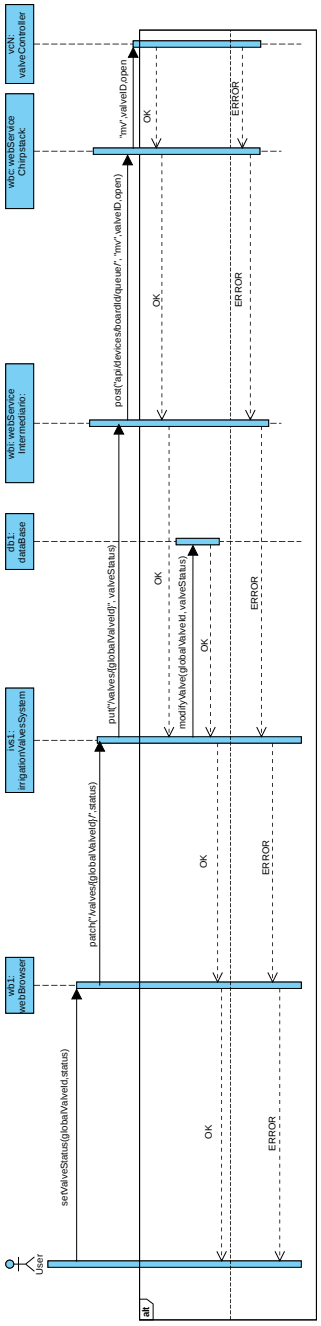

FIGURA A.11: Caso de uso de apertura y cierre de válvula

Borrado de válvula

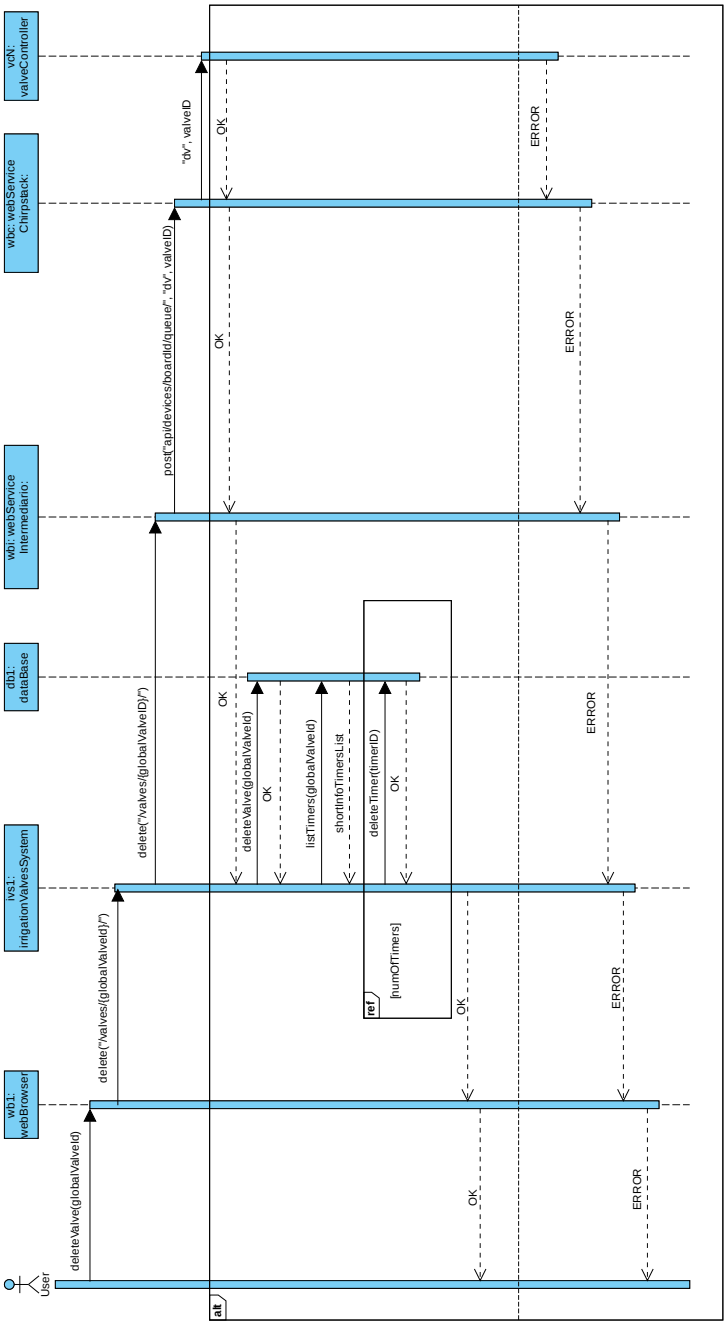

FIGURA A.12: Caso de uso de borrado de válvula

Vencimiento del temporizador

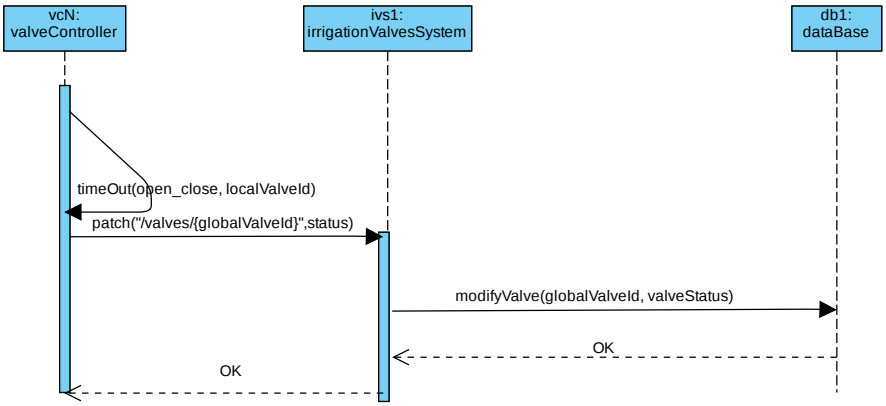

FIGURA A.13: Caso de uso de vencimiento del temporizador

Creación de temporizador

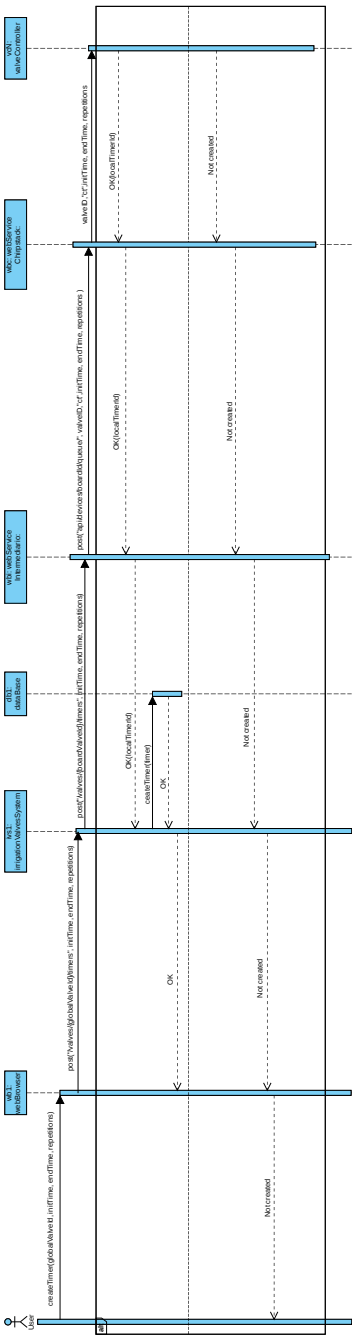

FIGURA A.14: Caso de uso de creación de temporizador

Listado de temporizadores

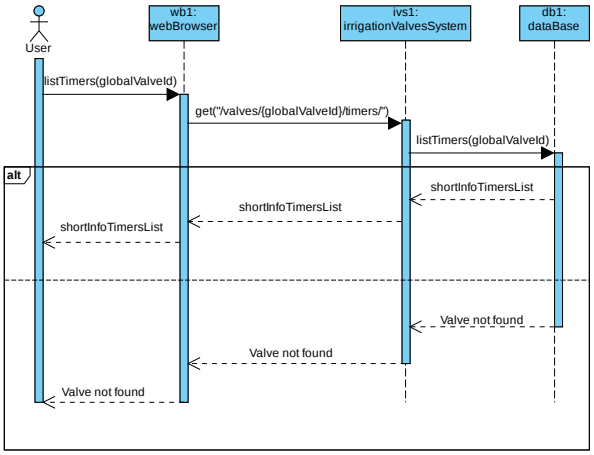

FIGURA A.15: Caso de uso de listado de temporizadores

Visualización de temporizador

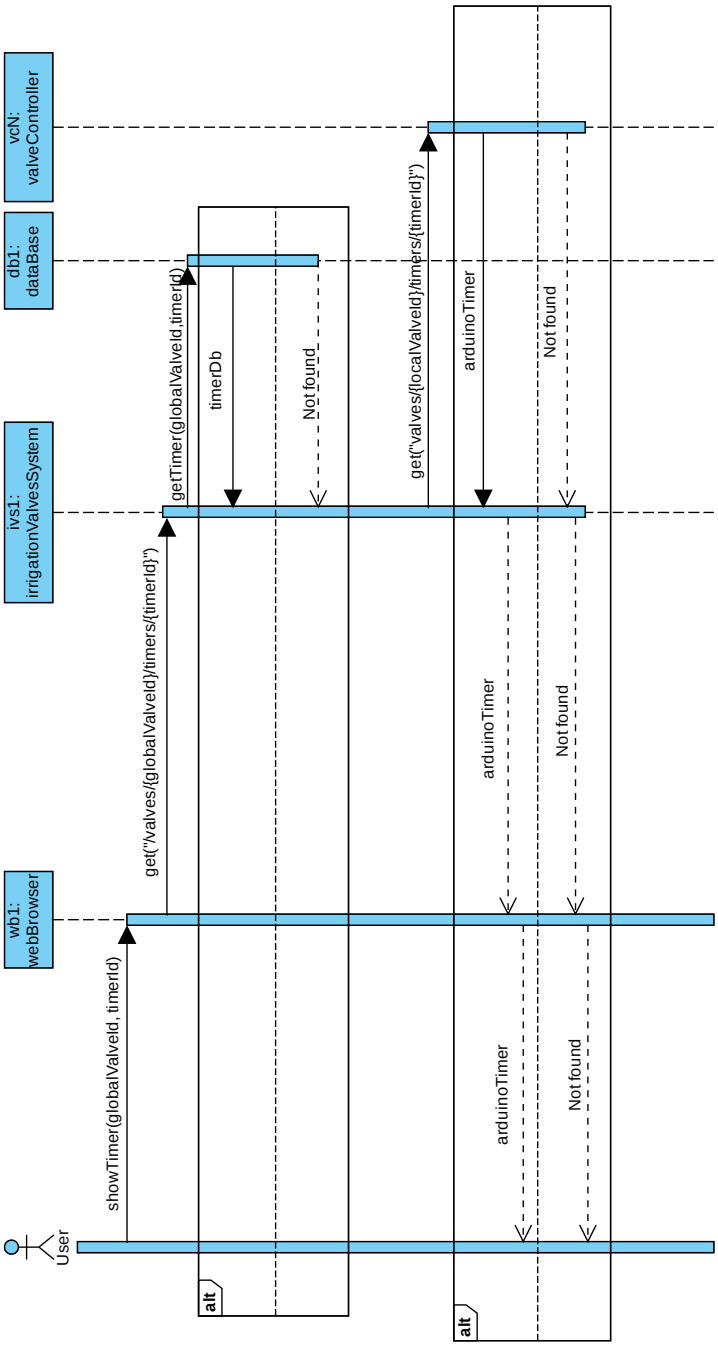

FIGURA A.16: Caso de uso de visualización del temporizador

Modificación de temporizador

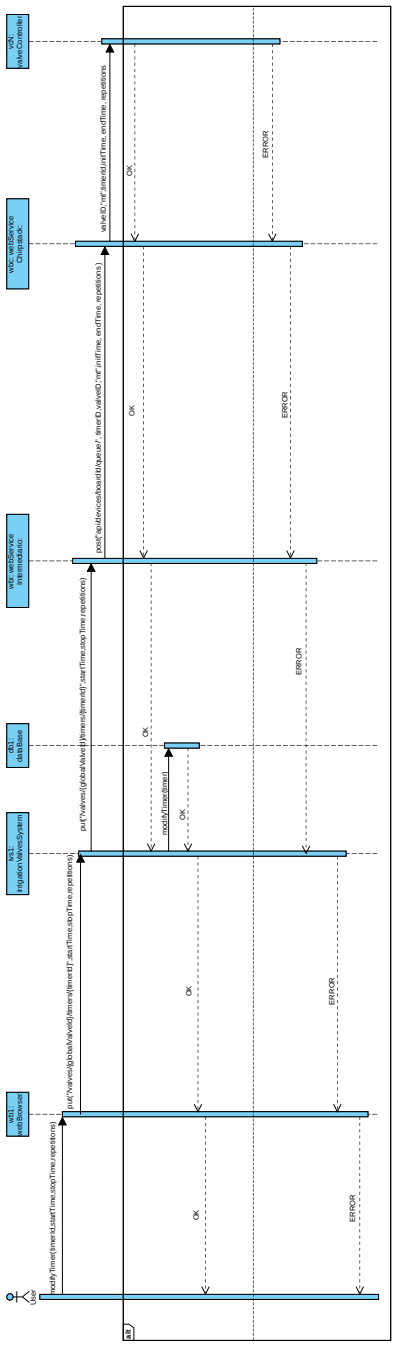

FIGURA A.17: Caso de uso de modificación de temporizador

Borrado de temporizador

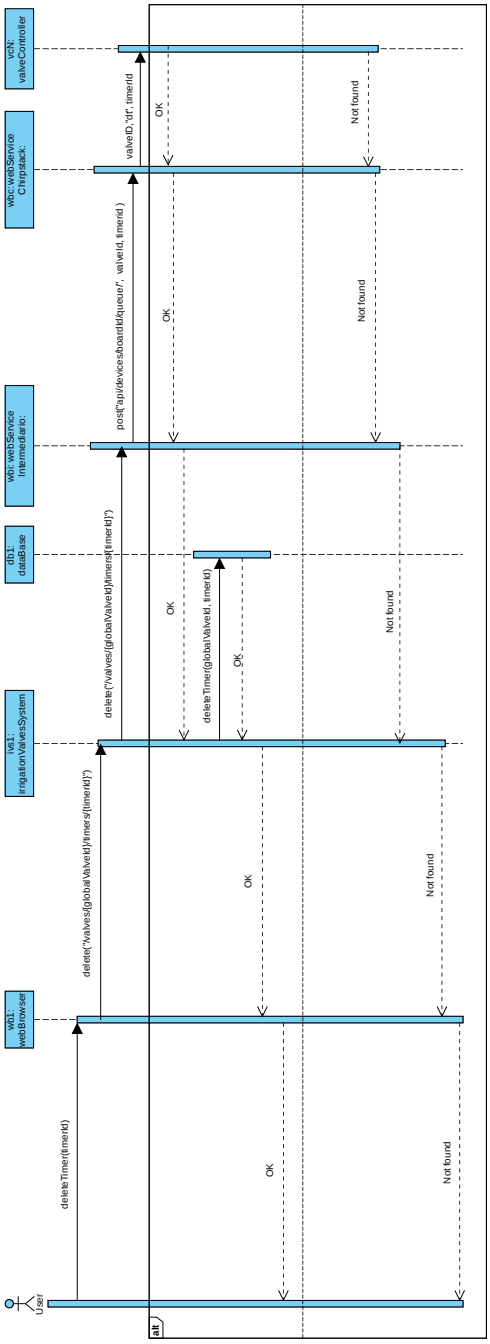

FIGURA A.18: Caso de uso de borrado de temporizador

A.2. Servicio web *Smartgranja*

A.2.1. Modelo de recursos del servicio web

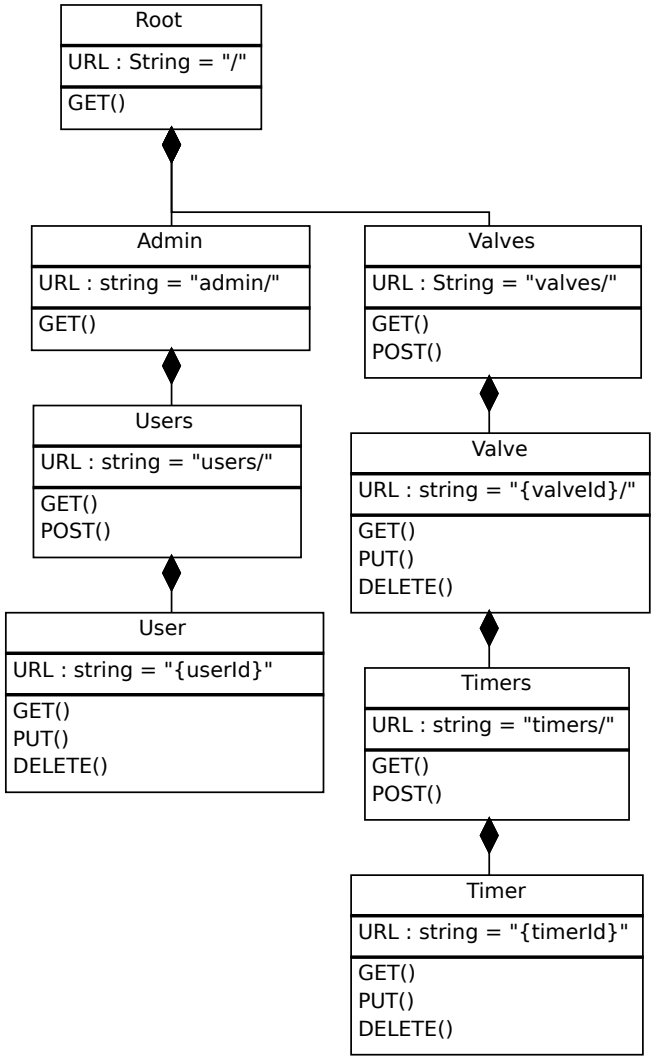

FIGURA A.19: Diagrama de recursos de la aplicación java

A.2.2. Autorización de usuarios

	Recursos		
Roles	**Person(s)**	**Valve(s)**	**Timer(s)**
Admin	R/W	R/W	R/W
ResourcesWrite	—	R/W	R/W
TimersWrite	—	R	R/W
ReadOnly	—	R	R

CUADRO A.1: Roles de usuarios y operaciones permitidas sobre recursos (R/W: lectura y escritura; R: solo lectura; —: sin permisos)

A.3. Servicio web intermediario

A.3.1. Modelo de recursos del servicio web

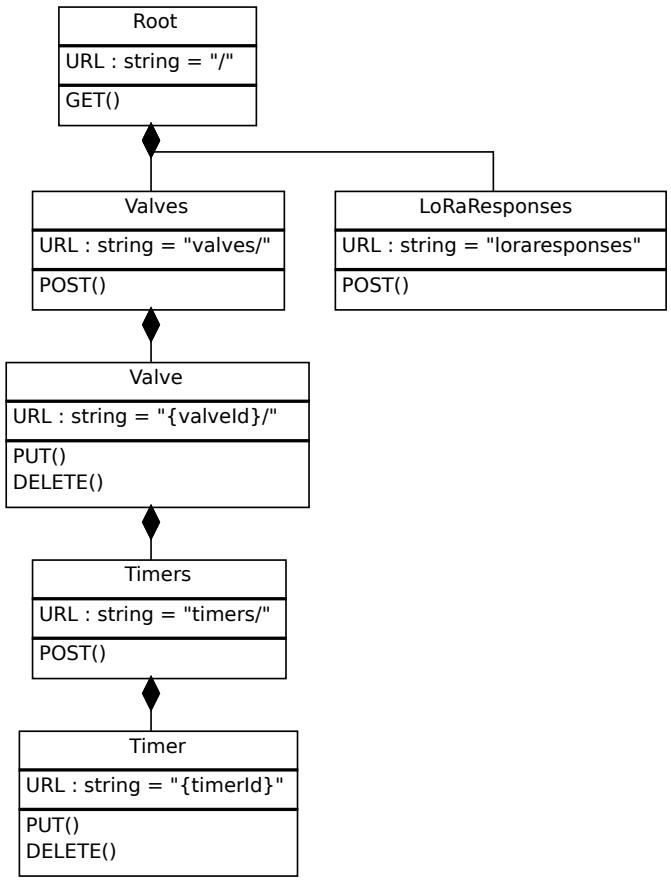

FIGURA A.20: Diagrama de recursos del servicio web intermediario

Bibliografía

[Alf20] *El protocolo LoRaWAN*. AlfaIoT. 6 de diciembre de 2020. URL: https://alfaiot.com/blog/ultimas-noticias-2/post/el-protocolo-lorawan-6 (visitado el 6 de diciembre de 2020).

[Ard20a] *¿Que es Arduino? | Arduino.cl - Compra tu Arduino en Línea*. Arduino. 6 de diciembre de 2020. URL: https://arduino.cl/que-es-arduino/ (visitado el 7 de diciembre de 2020).

[Ard20b] *Arduino - MKRWAN*. Arduino AG. 6 de diciembre de 2020. URL: https://www.arduino.cc/en/Reference/MKRWAN (visitado el 6 de diciembre de 2020).

[Ard20c] *Arduino Blog » Introducing the Arduino MKR WAN 1300 and MKR GSM 1400!* Arduino. 7 de diciembre de 2020. URL: https://blog.arduino.cc/2017/09/25/introducing-the-arduino-mkr-wan-1300-and-mkr-gsm-1400/ (visitado el 7 de diciembre de 2020).

[Aug+16] Aloÿs Augustin y col. «A Study of LoRa: Long Range and Low Power Networks for the Internet of Things». En: *Sensors* 16 (octubre de 2016), pág. 1466. DOI: 10.3390/s16091466.

[Avb20] Arjan Avbentem. *Airtime calculator for LoRaWAN*. 6 de septiembre de 2020. URL: https://avbentem.github.io/airtime-calculator/ttn/eu868/51 (visitado el 7 de diciembre de 2020).

[Boi+17] Rubbens Boisguene y col. «A survey on NB-IoT downlink scheduling: Issues and potential solutions». En: junio de 2017, págs. 547-551. DOI: 10.1109/IWCMC.2017.7986344.

[Bro20] Orne Brocaar. *Orne Brocaar, freelance software engineer (Go / Golang, Python and Django, IoT, LoRaWAN)*. 13 de febrero de 2020. URL: http://www.brocaar.com/ (visitado el 6 de diciembre de 2020).

[Bus01] Frank Buschmann. *Pattern-oriented software architecture : a system of patterns*. Ed. por John Wiley y Sons. 2001.

[Cas20] Gonzalo Casas. *Home · ttn-zh/ic880a-gateway Wiki · GitHub*. 8 de diciembre de 2020. URL: https://github.com/ttn-zh/ic880a-gateway/wiki (visitado el 8 de diciembre de 2020).

[Cat20] *Tecnología LoRA y LoRAWAN - Catsensors*. Catsensors. 3 de diciembre de 2020. URL: https://www.catsensors.com/es/lorawan/tecnologia-lora-y-lorawan (visitado el 3 de diciembre de 2020).

[Chi20a] *ChirpStack open-source LoRaWAN®Network Server*. ChirpStack. 24 de noviembre de 2020. URL: https://www.chirpstack.io/ (visitado el 6 de diciembre de 2020).

[Chi20b] *Device-classes - ChirpStack open-source LoRaWAN®Network Server*. ChirpStack. 24 de noviembre de 2020. URL: https://www.chirpstack.io/network-server/features/device-classes/ (visitado el 6 de diciembre de 2020).

[Chi20c] *Frame logging - ChirpStack open-source LoRaWAN®Network Server*. ChirpStack. 24 de noviembre de 2020. URL: https://www.chirpstack.io/application-server/use/frame-logging/ (visitado el 6 de diciembre de 2020).

[Chi20d] *Introduction - ChirpStack open-source LoRaWAN®Network Server*. ChirpStack. 24 de noviembre de 2020. URL: https://www.chirpstack.io/application-server/ (visitado el 6 de diciembre de 2020).

[Dig20] *Banda ISM ICM radiofrecuencia Industrial Cientifica y Medica*. Digital Micro Devices S.L. 3 de diciembre de 2020. URL: https://d3.xlrs.eu/banda-ism/ (visitado el 3 de diciembre de 2020).

[Dur20] Alain Durand. *RFC 3574 - Transition Scenarios for 3GPP Networks*. 29 de noviembre de 2020. URL: https://tools.ietf.org/html/rfc3574 (visitado el 3 de diciembre de 2020).

[Elk20] Ed Elkin. *The Secret Value of VoLTE*. 3 de diciembre de 2020. URL: http://blog.tmcnet.com/next-generation-communications/2014/04/the-secret-value-of-volte.html (visitado el 3 de diciembre de 2020).

[Fer20] Vicent Ferrer. *Qué es Sigfox - ¿Como funciona esta red IOT? Usos y casos de éxito*. 3 de diciembre de 2020. URL: https://vicentferrer.com/sigfox/ (visitado el 3 de diciembre de 2020).

[Fie02] Thomas Ray Fielding. *Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)*. 15 de marzo de 2002. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (visitado el 7 de diciembre de 2020).

[Fly20a] Kevin Flynn. *Standardization of NB-IOT completed*. 3 de diciembre de 2020. URL: https://www.3gpp.org/news-events/1785-nb_iot_complete (visitado el 3 de diciembre de 2020).

[Fly20b] Kevin Flynn. *Standards for the IoT*. 3 de diciembre de 2020. URL: https://www.3gpp.org/news-events/1805-iot_r14 (visitado el 3 de diciembre de 2020).

[Fou20] The Raspberry Pi Foundation. *Buy a Raspberry Pi 3 Model B+ ? Raspberry Pi*. Raspberry Pi. 8 de diciembre de 2020. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/?resellerType=home (visitado el 8 de diciembre de 2020).

[Hal20a] *eMTC (LTE Cat-M1) - Halberd Bastion*. Halberd Bastion. 3 de diciembre de 2020. URL: https://halberdbastion.com/technology/iot/iot-protocols/emtc-lte-cat-m1 (visitado el 3 de diciembre de 2020).

[Hal20b] Charles Hallard. *GitHub - ch2i/LoraGW-Setup: SX1301 Lora Concentrator Raspberry PI based gateway setup*. 6 de diciembre de 2020. URL: https://github.com/ch2i/LoraGW-Setup (visitado el 6 de diciembre de 2020).

[ION20] *Protobuf: ¿Qué es Protocol Buffers? + Tutorial - IONOS*. IONOS España. 6 de diciembre de 2020. URL: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/protocol-buffers/ (visitado el 6 de diciembre de 2020).

[Jso20] *JSON*. Json org. 5 de diciembre de 2020. URL: https://www.json.org/json-en.html (visitado el 6 de diciembre de 2020).

[Mar20] Vanja Samuelsson Markus Pihl. *Los modos de ahorro de energía de NB-IoT y Cat-M | DigiKey*. 3 de diciembre de 2020. URL: https://www.digikey.es/es/articles/how-to-enable-power-saving-modes-of-nb-iot-and-cat-m (visitado el 3 de diciembre de 2020).

[Muñ20] José Domingo Muñoz. *Qué es Flask y ventajas que ofrece | OpenWebinars*. 7 de diciembre de 2020. URL: https://openwebinars.net/blog/que-es-flask/ (visitado el 7 de diciembre de 2020).

[Nae20] *Que sucede con el apagado de redes 2G/3G Nae*. Nae. 4 de diciembre de 2020. URL: https://nae.global/es/que-sucede-con-el-apagado-de-redes-2g-3g/ (visitado el 4 de diciembre de 2020).

[Ora20] *LoRaWAN: la tecnologia que puede impulsar IoT | Orange Business Services*. Orange Business Services 2020. 3 de diciembre de 2020. URL: https://www.orange-business.com/es/blogs/lorawan-tecnologia-que-puede-impulsar-internet-las-cosas (visitado el 3 de diciembre de 2020).

[Par20] Luis Carlos Parra Rebollo. «Diseño y desarrollo de un sistema de monitorización y control del riego en una explotación agrícola». Trabajo de Fin de Grado. Universidad de Valladolid, noviembre de 2020.

[PJ18] Sakshi Popli y Sanjeev Jain. «A Survey on Energy Efficient Narrowband Internet of Things (NBIoT): Architecture, Application and Challenges». En: *IEEE Access* PP (noviembre de 2018), págs. 1-1. DOI: 10.1109/ACCESS.2018.2881533.

[Rod20] Manuel Rodríguez Cayetano. *Paradigmas y tecnologías de procesamiento distribuido: Introducción al estilo de arquitectura REST (Representational State Transfer)*. E.T.S.I. de Telecomunicación. Universidad de Valladolid. Febrero de 2020.

[RP16] Brecht Reynders y S. Pollin. «Chirp spread spectrum as a modulation technique for long range communication». En: noviembre de 2016, págs. 1-5. DOI: 10.1109/SCVT.2016.7797659.

[Sae+19] Martijn Saelens y col. «Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs : an overview and future challenges». En: *EURASIP JOURNAL ON WIRELESS COMMUNICATIONS AND NETWORKING* 2019.1 (2019), 219:1-219:32. ISSN: 1687-1472. URL: http://dx.doi.org/10.1186/s13638-019-1502-5.

[Sem16] N. Sornin (Semtech). *LoRaWAN Specification*. 1 de julio de 2016.

[SEM20a] *An In-depth look at LoRaWAN® Class A Devices | DEVELOPER PORTAL*. SEMTECH. 6 de diciembre de 2020. URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-a-devices/ (visitado el 6 de diciembre de 2020).

[SEM20b] *An In-depth Look at LoRaWAN® Class B Devices | DEVELOPER PORTAL*. SEMTECH. 6 de diciembre de 2020. URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-b-devices/ (visitado el 6 de diciembre de 2020).

[SEM20c] *An In-depth Look at LoRaWAN® Class C Devices | DEVELOPER PORTAL*. SEMTECH. 6 de diciembre de 2020. URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-c-devices/ (visitado el 6 de diciembre de 2020).

[SEM20d] *LoRa and LoRaWAN: Technical overview | DEVELOPER PORTAL*. SEMTECH. 3 de diciembre de 2020. URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/ (visitado el 3 de diciembre de 2020).

[SEM20e] *SX1301 | Digital Baseband Chip LoRaWAN macro gateways | Semtech*. SEMTECH. 7 de diciembre de 2020. URL: https://www.semtech.com/products/wireless-rf/lora-gateways/sx1301 (visitado el 7 de diciembre de 2020).

[SEM20f] *What is LoRa? | Semtech LoRa Technology | Semtech*. SEMTECH. 3 de diciembre de 2020. URL: https://www.semtech.com/lora/what-is-lora (visitado el 3 de diciembre de 2020).

[Sig20a] *Coverage | Sigfox*. Sigfox. 3 de diciembre de 2020. URL: https://www.sigfox.com/en/coverage (visitado el 3 de diciembre de 2020).

[Sig20b] *Our story | Sigfox*. Sigfox. 3 de diciembre de 2020. URL: https://www.sigfox.com/en/sigfox-story (visitado el 3 de diciembre de 2020).

[Sig20c] *Sigfox Device Radio Specifications | Sigfox build*. Sigfox build. 4 de diciembre de 2020. URL: https://build.sigfox.com/sigfox-device-radio-specifications (visitado el 4 de diciembre de 2020).

[sig20] *Gracias a su estacion agro-meteorologica conectada a Sigfox, la nueva empresa Sencrop recauda 10 millones de euros - Sigfox Espana*. sigfox España. 3 de diciembre de 2020. URL: https://www.sigfox.es/blogs/post/gracias-a-su-estacion-agro-meteorologica-conectada-a-sigfox-la-nueva-empresa-sencrop-recauda-10-mill (visitado el 4 de diciembre de 2020).

[Tal19] *Restlet Framework | Overview*. Talend. 2019. URL: https://restlet.talend.com/ (visited on January 31, 2020).

[The20a] *Adaptive Data Rate | The Things Network*. The Things Network. 24 de noviembre de 2020. URL: https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate.html (visitado el 3 de diciembre de 2020).

[The20b] *Duty Cycle The Things Network*. The Things Network. 24 de noviembre de 2020. URL: https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html (visitado el 3 de diciembre de 2020).

[The20c] *The Things Network*. The Things Network. 3 de diciembre de 2020. URL: https://www.thethingsnetwork.org/ (visitado el 3 de diciembre de 2020).

[ubl20] *LTE Cat 1 | u-blox*. u-blox. 3 de diciembre de 2020. URL: https://www.u-blox.com/en/technologies/lte-cat-1 (visitado el 3 de diciembre de 2020).

[Uni20] *Modelo vista controlador (MVC). Servicio de Informática ASP.NET MVC 3 Framework*. Universidad de Alicante. 7 de diciembre de 2020. URL: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html (visitado el 7 de diciembre de 2020).

[Wir20] *iC880A-SPI LoRa® Concentrator - Wireless Solutions*. Wireless Solutions by IMST GmbH. 7 de diciembre de 2020. URL: https://wireless-solutions.de/products/lora-solutions-by-imst/radio-modules/ic880a-spi/ (visitado el 7 de diciembre de 2020).

[Wor18] *Web Services Glossary*. World Wide Web Constortion (W3C). 9 de octubre de 2018. URL: https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice (visitado el 7 de diciembre de 2020).