

SISTEMA DE MONITOREO DEL SUMINISTRO ELÉCTRICO LIBRE Y DE BAJO COSTE



**UNIVERSIDAD
DE GRANADA**

TRABAJO FIN DE GRADO

Autor

GONZALO DE LA TORRE MARTÍNEZ

Tutor

SERGIO ALONSO BURGOS



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicaciones.

Departamento de Lenguajes y Sistemas Informáticos.

GRANADA, 8 DE JULIO DE 2022

AGRADECIMIENTOS

A mi familia, por ofrecerme la oportunidad y motivarme para estudiar algo que me llena. A mi tutor, por guiarme de la mejor forma posible y ofrecerme independencia a la hora de llevar el proyecto. A mis amigos y compañeros de piso, por acompañarme en lo que ha sido una de las mejores etapas de mi vida.

Yo, **Gonzalo de la Torre Martínez**, alumno de la titulación **GRADO EN INGENIERÍA INFORMÁTICA** de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 45926744E, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Gonzalo de la Torre Martínez.

D. Sergio Alonso Burgos, Profesor del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado “Sistema de monitoreo del suministro eléctrico libre y de bajo coste”, ha sido realizado bajo su supervisión por Gonzalo de la Torre Martínez, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada, a 8 de julio de 2022

Sergio Alonso Burgos:



ÍNDICE DE CONTENIDOS

INTRODUCCIÓN	7
RESUMEN	11
ABSTRACT	12
OBJETIVOS	13
ORGANIZACIÓN	14
METODOLOGÍA	14
Repositorio	14
Modelo en Cascada	14
Modelo Iterativo/Ágil	15
PLANIFICACIÓN	17
PRESUPUESTO	18
Mano de obra	18
Dispositivo Arduino	19
Software	19
Servidor y dominio	19
ANÁLISIS	21
ESPECIFICACIÓN DE REQUISITOS	21
Requisitos funcionales.	21
Fase 1 (Dispositivo Monitor).	21
Fase 2 (App Móvil)	22
Fase 3 (API)	22
Fase 4 (Página Web)	23
Requisitos NO funcionales.	24
Requisitos de información.	24
CASOS DE USO	25
Diagrama de casos de uso	25
Descripción de casos de uso.	27
DISEÑO	35
VISIÓN GENERAL DEL SISTEMA	35
MODELADO DE LA BASE DE DATOS	36
Base de datos relacional (SQL)	36
Base de datos no relacional (NoSQL)	36
Ventajas SQL	37
Ventajas NoSQL	37
Conclusión (elección final)	37
Diagrama entidad-relación de la base de datos No Más Cortes	38
ARQUITECTURA DEL SISTEMA	40
Fase 1: Diseño de prototipo monitor mediante Arduino.	40

Fase 2: aplicación móvil	43
Fase 3: API REST	46
Fase 4: página web	48
INTERFACES DE USUARIO	50
MockUps App No Más Cortes.	50
MockUps Web No Más Cortes.	51
Paleta de colores	52
IMPLEMENTACIÓN	53
BASE DE DATOS	53
ARDUINO	54
Comandos AT	55
Paso de mensajes	55
APP MÓVIL	57
API REST	58
Prevención de inyección SQL	58
PÁGINA WEB	59
Diseño final	60
PRUEBAS	62
Fase 1: Arduino	62
Fase 2: aplicación móvil	63
Fase 3: API REST	64
Fase 4: página web	65
CONCLUSIONES	66
BIBLIOGRAFÍA	68

INTRODUCCIÓN

Desde hace muchos años, la zona norte de Granada sufre un gran problema. Son constantes los cortes de energía eléctrica que se producen en los hogares de los vecinos del lugar. Es por esto que, los vecinos del distrito Norte de la ciudad, en declaraciones institucionales al ayuntamiento de Granada, exigen el fin de los reiterados cortes de energía eléctrica.

Lo que se pide en estas declaraciones institucionales, son diferentes inversiones que mejoren las condiciones de la infraestructura allí instalada. Además, piden que el suministro energético se convierta en un derecho de las familias. Con estos escritos pretenden acabar con el calvario de tener que vivir en sus casas con cortes constantes que les hagan la vida imposible.

Diferentes figuras políticas del ayuntamiento de Granada han mostrado su apoyo públicamente a los vecinos del distrito mediante declaraciones en rueda de prensa.

Por ejemplo, una noticia muy reciente del periódico Granada Hoy cuenta lo siguiente: "El Pleno del Ayuntamiento de Granada de marzo incluirá una declaración institucional en la que todos los grupos políticos con representación municipal solicitan de "forma urgente e inmediata y de manera integral" una plan de actuación "concreto y estructural" por parte de las administraciones competentes y de la empresa suministradora del servicio eléctrico que permita "acabar con la lacra de los cortes de luz que desde hace diez años sufre el distrito Norte de la ciudad" "[1].

En la misma noticia, se destaca lo siguiente, el 75% de la población paga sus recibos de la luz y un 20% cuenta con una situación de pobreza energética.

Por otra parte, la empresa líder del sector eléctrico español, Endesa, mediante su director provincial en Granada, José Luis Pérez Mañas, declara que, "la red no da para más y es técnicamente imposible solventar el problema" [2]. Sugiriendo así, que en esta zona de Granada se realizan diferentes actividades ilegales relacionadas con plantaciones de marihuana que provocan que su red colapse.

Estas declaraciones chocan con los datos ofrecidos por la siguiente noticia digital de IDEAL Granada, que afirma que, en 2019, sólo el 6,25% de los enganches ilegales proviene de las plantaciones de marihuana.[3]

La Asociación Pro Derechos Humanos de Andalucía es una organización sin ánimo de lucro, privada, cuyo propósito es defender los Derechos Humanos en el territorio andaluz. Mediante esta asociación se busca acabar con las irregularidades en lo que a Derechos humanos se refiere y ayudar a posibles víctimas.

Este grupo se hizo eco del problema que existe en la zona Norte y ha emprendido varias acciones para intentar revertir esta situación.

“El derecho a la energía (luz), vivienda, alimentación, sanidad y educación, son derechos mínimos para vivir con dignidad, que en este caso vienen siendo vulnerados sistemáticamente por todas las administraciones públicas implicadas que, repetidamente, miran hacia otro lado y no se responsabilizan de que los derechos se respeten también en esta parte de la ciudad”[4].

El fragmento anterior, perteneciente al periódico Granada Digital, destaca la inacción de las administraciones públicas y locales ante el problema. Además, también denuncia el mal hacer de la empresa de energía (Endesa), por priorizar su beneficio económico ante el problema de los vecinos.

Por lo tanto la Asociación Pro Derechos Humanos de Andalucía (APDHA) insta a Endesa a intervenir “donde y como sea necesario” y a las administraciones a cumplir con su compromiso con la ciudadanía y empezar a poner soluciones.

Vistas las incongruencias de las instituciones locales, nacionales y de la energética Endesa, nace la organización No Más Cortes. Este es un proyecto donde colaboran los siguientes grupos:

- Anaquerando,
- la Parroquia de la Paz,
- el Centro Socioeducativo Lestonnac,

- la Asociación Pro Derechos Humanos de Andalucía en Granada (APDHA),
- la Asociación Andaluza de Barrios Ignorados y su delegación en Granada y
- la Delegación Pastoral Gitana Granada.

Gracias a los grupos mencionados arriba y al tutor de este trabajo de fin de grado, Sergio Alonso Burgos, se desarrolla la plataforma en forma de página web llamada No Más Cortes.

La organización y todas las personas que la componen trabajan y aportan su granito de arena sin ninguna intención de lucro, todo por la mejora de las condiciones de esta zona de Granada.

La finalidad de esta página web es mostrar los cortes de luz que se producen en la zona, con el objetivo de dar una mayor visibilidad al problema y así poder ejercer más presión si cabe a las instituciones y a la propia Endesa.

El modo de funcionamiento de esta página web era el siguiente:

1. Un vecino detecta un corte de luz.
2. Llama a un teléfono proporcionado para informar sobre la calle y las horas en las que se ha producido dicho corte.
3. El administrador que ha recibido la llamada introduce la información facilitada.
4. Finalmente ese corte de luz es visible en la página web.

Teóricamente parecía una buena idea y la página web estuvo activa desde inicios de 2020 hasta enero de 2021 registrando diariamente cortes de luz. Pero poco a poco fueron apareciendo problemas que empezaron a hacer dudar de la viabilidad del proyecto.

En primer lugar, dado que este problema (cortes de luz constantes) ha dejado de ser excepcional para convertirse en la normalidad de muchas personas, los vecinos olvidaban apuntar las horas a las que se producían los cortes o los recordaban con inexactitud. Por lo tanto, muchas veces se presentaban problemas de integridad de datos porque no eran del todo correctos.

También, debido al volumen de cortes, los administradores de la página web, no tenían el tiempo suficiente para introducir toda la información que les llegaba. Hay que aclarar que eran personas que no cobraban por esa tarea, simplemente querían ayudar a la causa.

Por lo tanto, dados todos estos problemas, se decidió parar con la monitorización manual de los cortes para pensar una solución más automatizada y viable.

La solución que se ha pensado es la que presenta este trabajo de fin de grado. Esta nueva solución propone un dispositivo que detecte automáticamente todos los cortes de luz, a continuación, estos datos se transfieran a un servidor y finalmente se muestren en una página web.

RESUMEN

El proyecto que se presenta consta de 4 partes que explicaré a continuación. En primer lugar, construiremos un dispositivo prototipo mediante una placa Arduino UNO y varios módulos. Este dispositivo, conectado a un enchufe de la instalación eléctrica de una vivienda, se encargará de detectar cambios de estado en el suministro de energía. Es decir, será nuestro monitor de datos.

Por otro lado, crearemos una página web prototipo donde cualquier usuario podrá consultar los cortes de luz producidos en la zona, junto con un mapa que muestre de zona más gráfica las últimas muestras tomadas por el monitor construido con Arduino. Los usuarios administradores tendrán acceso a otras funcionalidades extra como la descarga de datos en bruto o el filtrado de los mencionados cortes de luz. Para esta parte utilizaremos algunos componentes del framework de programación Symfony, como es Twig (gestor de plantillas), que está respaldado por el lenguaje de programación PHP.

Para la transmisión de datos entre el servidor y el dispositivo monitor hemos decidido ayudarnos de una aplicación Android realizada desde cero a modo de prototipo. Esta aplicación recibirá mediante Bluetooth las medidas del monitor y se encargará de enviarlas al servidor. También tendrá un apartado protegido para la configuración del dispositivo Arduino por parte de los administradores. Hemos decidido implementar la aplicación mediante el IDE de programación Android Studio y el lenguaje de programación Java.

Finalmente, desarrollaremos una API REST que nos permitirá realizar el manejo de datos necesario para comunicar los diferentes elementos del sistema. Para la realización de esta parte se ha decidido usar Node.js junto al paquete Express que nos facilita el trabajo.

ABSTRACT

The project presented consists of 4 parts that I will explain below. First, we will build a prototype device using an Arduino UNO board and different modules. This device, connected to a plug in the electrical installation of a home, will be responsible for detecting changes in the state of the energy supply. That is, it will be our data monitor.

On the other hand, we will create a prototype web page where any user can consult the power outages produced in the area, along with a map that shows the latest samples taken by the monitor built with Arduino. Administrator users will have access to other extra features such as downloading raw data or filtering the aforementioned power outages. For this part we will use some components of the Symfony programming framework, such as Twig (template manager), which is supported by the PHP programming language.

For the transmission of data between the server and the monitor device we have decided to use an Android application made from scratch as a prototype. This application will receive the measurements from the monitor via Bluetooth and will send them to the server. It will also have a protected section for the configuration of the Arduino device by administrators. We have decided to implement the application using the Android Studio programming IDE and the Java programming language.

Finally, we will develop a REST API that will allow us to perform the necessary data management to communicate the different elements of the system. To carry out this part, it has been decided to use Node.js together with the Express package that makes our work easier.

OBJETIVOS

El objetivo principal de este trabajo es demostrar que se puede desarrollar un sistema de **bajo coste** para el monitoreo de los cortes de luz.

A continuación se presentan los objetivos específicos que debe cumplir nuestro proyecto. También se especifican objetivosopcionales.

1. Construir un prototipo de dispositivo Arduino que realice lo siguiente:
 - 1.1. Captar cambios de estado en el suministro eléctrico.
 - 1.2. Almacenar los cambios de estado detectados.
 - 1.3. Mantener la información del dispositivo.
 - 1.4. Transmitir datos recogidos mediante Bluetooth.
2. Desarrollar una aplicación móvil prototipo que:
 - 2.1. Reciba información de las medidas captadas por el dispositivo Arduino.
 - 2.2. Configure el dispositivo Arduino.
 - 2.3. Actualice el dispositivo Arduino.
 - 2.4. Envíe las medidas recogidas al servidor.
3. Hacer y desplegar una página web prototipo que cumpla los siguientes requisitos:
 - 3.1. Ofrecer los datos de los cortes de luz registrados para descargar.
 - 3.2. Mostrar los cortes de luz recogidos.
 - 3.3. Mostrar gráficamente con un mapa los cortes de luz que se han producido (opcional).
 - 3.4. Poner a disposición de los administradores un espacio exclusivo.
 - 3.5. Filtrar por fecha y dispositivo los cortes de luz (opcional).
 - 3.6. Ordenar los cortes de luz mediante diferentes criterios (duración, calle...) (opcional).

ORGANIZACIÓN

METODOLOGÍA

Repository

Se ha creado un enlace a un repositorio de GitHub donde se guardará y mantendrá toda la información, archivos fuente y documentación de este trabajo de fin de grado:

<https://github.com/gonzalodelatorree/trabajo-fin-grado>

Se procederá a explicar la metodología que se va a utilizar para desarrollar este proyecto. En el desarrollo de software tradicional es común el uso del modelo en cascada. En cambio, en los últimos años han cogido mucha fuerza los modelos iterativos/ágiles, por lo tanto voy a compararlos y ver qué tiene más sentido en este contexto.

Modelo en Cascada [5]

La característica principal del modelo en cascada es que sigue una secuencia lineal. Esta secuencia lineal consta de 5 etapas que definen los pasos a seguir para conseguir un buen software.

1. Análisis: consiste en realizar una definición detallada de los requisitos que tiene que cumplir el sistema, además de analizar los costes, rentabilidad del proyecto, viabilidad...
2. Diseño: en esta fase definimos la arquitectura del sistema, componentes, entornos de trabajo. Es decir, buscamos una solución específica a las exigencias del punto anterior.
3. Implementación: transformamos toda la documentación generada en las fases anteriores en código, esto nos da nuestra primera versión de nuestro proyecto.
4. Pruebas: integramos nuestro desarrollo en un entorno para comprobar su correcto funcionamiento y buscar mejoras.
5. Servicio/Mantenimiento: última fase en la que nuestra aplicación pasa al entorno para el que se ha creado, no podemos olvidarnos una vez despleguemos nuestro desarrollo, debemos mantenerlo para estar atentos en caso de errores.



Figura 1 [6]. Velázquez Camacho, J. (2012, 29 noviembre). Modelo Cascada [Ilustración].

Modelo Iterativo/Ágil [5]

Por su parte, el modelo iterativo/ágil destaca por su búsqueda del desarrollo de una manera más dinámica. Está enfocado a la productividad, en vez de seguir ciertos pasos a rajatabla como en el modelo en cascada.

Podríamos definirlo como una forma de trabajo en la que un proyecto se divide en varias partes que tienen que completarse y entregarse en breves períodos de tiempo.

Con este modelo, nuestro objetivo es tener una comunicación muy buena con el cliente, para poder adaptarnos a sus necesidades y entregar pequeñas mejoras del software a corto plazo.

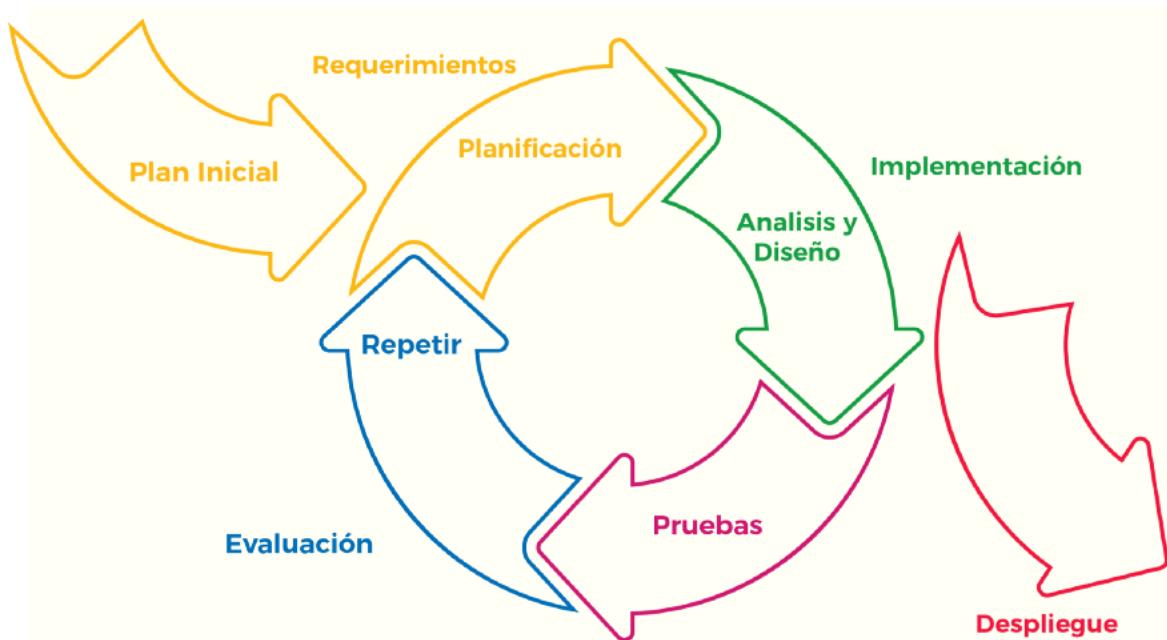


Figura 2 [7]. Jerez, N. (2019, 3 junio). Metodología Iterativa [Ilustración].

Por lo tanto, en nuestro caso, haremos uso de una mezcla de estas dos formas de trabajo. Por una parte, seguiremos el modelo tradicional (cascada) para definir y aclarar los pasos a seguir al realizar nuestro proyecto. Aunque a su vez, también nos guiaremos por el método ágil porque nuestro proyecto está lleno de pequeños objetivos que pueden ir completándose individualmente en periodos pequeños de tiempo.

Por una parte, cogeremos la ventaja de la estructuración y análisis del método en cascada para, por ejemplo, definir el intercambio de mensajes entre dispositivos, API, etc.

Por otro lado, a la hora de implementar las diferentes funcionalidades del sistema, iremos moviéndonos de una fase a otra. De esta manera podremos empezar y finalizar una funcionalidad completa (como hace el método iterativo) del sistema transversalmente en todas las fases.

PLANIFICACIÓN

El proyecto puede dar comienzo el día 21 de Marzo, y debe estar completado el 8 de Julio, por lo tanto se disponen de 14 semanas (excluyendo semanas no laborables). Para compatibilizar el desarrollo del proyecto junto a las obligaciones académicas, se le dedicarán 10 horas a la semana.

Como se ha comentado en el apartado anterior, vamos a dividir el proyecto en 4 fases, por lo tanto el reparto de trabajo a cada parte será el siguiente.

FASE	SEMANAS (5 días/semana)	DÍAS (2h/día)
1: Arduino	6	30
2: App móvil	4	20
3: API + BBDD	1	5
4: Web	3	15
TOTAL:	14	70

Hay que tener en cuenta que la parte de Arduino es la más compleja logísticamente, ya que después de la fase de análisis hay que conseguir las piezas, en este caso se piden mediante tiendas online. Hay que tener en cuenta que los plazos de entrega no tienen porqué ser exactos y hay que dar un pequeño margen de tiempo. Esta es la principal razón por la que la fase 1 es la más extensa en tiempo.

En la siguiente tabla se mostrará la planificación por semanas del proyecto, vemos en que se van a centrar los esfuerzos en esa semana. Cada color especifica una fase y cada letra una etapa del desarrollo software. Para más simplicidad, he agrupado las fases del desarrollo software en 4:

1. Análisis.
2. Diseño.
3. Implementación.
4. Pruebas.

	A	A	A	A	D
Semana 1					
Semana 2	A	D	D	D	I
Semana 3	D	D	D	I	I
Semana 4	D	I	I	I	A
Semana 5	I	I	I	A	A
Semana 6	I	I	P	A	D
Semana 7	I	P	A	D	D
Semana 8	I	D	D	I	A
Semana 9	P	D	I	I	D
Semana 10	P	I	I	I	I
Semana 11	I	P	I	P	A
Semana 12	P	A	A	D	D
Semana 13	D	I	I	I	I
Semana 14	I	I	P	P	Despliegue

Fase 1 Fase 2 Fase 3 Fase 4

A -> Análisis
D -> Diseño
I -> Implementación
P -> Pruebas

PRESUPUESTO

Mano de obra

Para concretar la parte de mano de obra humana, tenemos que tener en cuenta la cantidad de horas que se van a necesitar para finalizar el proyecto. Además una remuneración de 10€ por hora.

TIEMPO (DÍAS)	TIEMPO (HORAS)	PRECIO / HORA (€)	TOTAL (€)
70	140	13	1820.00

Dispositivo Arduino

Para poder construir el dispositivo mediante arduino necesitamos varios módulos y piezas, a continuación se muestra el desglose:

ARTÍCULO	PRECIO (€)	ENLACE COMPRA
Placa Arduino UNO	3.5	Aliexpress
Módulo RTC DS3231 + pila	3.2*	Amazon
Módulo lector microSD	1.7*	Amazon
Tarjeta microSD	2.1*	Amazon
Módulo Bluetooth HC05	5.5*	Amazon
Módulo Cargador de Batería (con protección)	0.7*	Amazon
Pilas litio AA 3000mA	1.4*	Amazon
Adaptador de corriente microUSB	2.3	Aliexpress
Módulo step-up 5V	1.4*	Amazon
Caja Estanca	5.5*	Amazon
Fungibles (cables, estaño...)	4	
TOTAL:		31.3€

(Se muestra el precio aproximado por la fluctuación de precio en el mercado, ya que un producto puede variar un poco dependiendo del momento de compra.)

Precio al comprar varias unidades*

Software

En este apartado, el coste será de 0€ ya que se utilizarán herramientas de software libre y gratuito.

Servidor y dominio

En el ámbito de servidor y dominio podemos encontrar varias opciones.

Hostinger[8]. Este proveedor de hosting y dominio tiene un plan, llamado *Hosting Premium*, que ofrece la adquisición del dominio de forma gratuita, además certificado SSL y protección

DDoS. Nos da la opción de acceder mediante SSH. El precio sería de 2€ al mes durante un año y si posteriormente queremos renovar, serían 5€/mes.

WebEmpresa[9]. Con la oferta de lanzamiento, su *Plan Mini* ofrece dominio gratuito durante el primer año y características de potencia del servidor parecidas al proveedor anterior. Es decir, recursos suficientes como para alojar nuestro proyecto sin problema. El precio sería de 49.50€ el primer año y 99€ los años posteriores.

Por lo tanto, Hostinger parece una muy buena opción por su oferta del primer año y la renovación de años posteriores. Además ofrece recursos suficientes para cumplir con las necesidades de nuestro proyecto.

Entonces, podríamos tener cubierta la parte de servidor y dominio por (2€/mes * 12 meses) 24€ el primer año y alrededor de 60€ en años posteriores.

Como conclusión de este apartado, podemos ver que, teniendo en cuenta que trabajaremos con herramientas de software gratuitas, y que construir el dispositivo de monitorización cuesta poco más de 30€, parece factible completar el objetivo principal de este trabajo, la demostración de que se puede desarrollar un sistema de **bajo coste** para el monitoreo de los cortes de luz.

ANÁLISIS

ESPECIFICACIÓN DE REQUISITOS

La especificación de requisitos es un proceso por el cual definimos las funcionalidades que debe cumplir un sistema, es importante destacar que en este proceso solo vamos a definir lo que el sistema debe hacer, no cómo lo va a hacer.

La definición de estos requisitos forma parte de un proceso de comunicación entre el cliente y los desarrolladores. Es muy importante que los objetivos queden claramente definidos entre las dos partes en un documento, esto se puede usar en forma de contrato entre cliente y desarrollador por posibles malentendidos en el futuro sobre el funcionamiento de la aplicación.

En este trabajo, que está definido en 4 fases, vamos a definir una serie de requisitos funcionales, no funcionales y de información en cada una de las fases.

Un requisito funcional [10] es aquel que define una funcionalidad específica que un sistema debe cumplir, puede describir entradas, salidas y comportamientos.

Un requisito no funcional define la calidad del software, es decir, genera restricciones que debe cumplir el sistema a nivel de latencia, seguridad, fiabilidad, forma de desarrollo, etc.

Finalmente, definimos el último requisito que vamos a usar en nuestro sistema, los requisitos de información. Estos son los que nos dicen qué información debe guardar y gestionar el sistema para su correcto funcionamiento.

Requisitos funcionales.

1. Fase 1 (Dispositivo Monitor).

- 1.1. El dispositivo debe ser capaz de detectar los cambios de estado en la corriente de un enchufe.

- 1.2. Cada vez que se encienda el dispositivo debe almacenar un mensaje de inicialización en la tarjeta microSD.
- 1.3. Cada cambio de estado de la corriente se debe almacenar en la tarjeta microSD.
- 1.4. El dispositivo constantemente estará observando si le llega algún mensaje de los siguientes:
 - 1.4.1. Petición de datos de configuración.
 - 1.4.2. Petición de actualización de datos de configuración.
 - 1.4.3. Petición de actualizar hora del reloj.
 - 1.4.4. Petición de enviar información de medidas (cambios de estado).
 - 1.4.4.1. Borra registro de medidas.
 - 1.4.4.2. Mantiene registro de medidas actual.
- 1.5. Deben enviarse las medidas recogidas mediante Bluetooth a un dispositivo externo.
- 1.6. El dispositivo debe ser capaz de actualizar su configuración mediante un mensaje de entrada por Bluetooth.
- 1.7. Debe ser capaz de actualizar su reloj interno mediante un mensaje de entrada por Bluetooth.
- 1.8. El dispositivo enviará sus datos de configuración por Bluetooth.

2. Fase 2 (App Móvil)

Los siguientes requisitos precisan de una previa conexión Bluetooth al dispositivo sobre el que se quiere actuar.

- 2.1. La aplicación móvil deberá proveer un apartado para usuarios estándar.
 - 2.1.1. Deberá recoger las medidas recogidas por el dispositivo de la fase 1, seguidamente deberá enviar esas medidas al servidor.
- 2.2. También deberá proveer un apartado para administradores del sistema.
 - 2.2.1. Un administrador podrá inicializar un dispositivo nuevo.
 - 2.2.2. Tendrá la opción de actualizar la información de un dispositivo ya existente.
 - 2.2.3. Podrá actualizar la hora del reloj del dispositivo al que está conectado.

3. Fase 3 (API)

La API REST deberá proveer las siguientes peticiones

- 3.1. GET (consulta de datos)
 - 3.1.1. Retornar información de todos los dispositivos.
 - 3.1.2. Retornar información de un dispositivo concreto.
 - 3.1.3. Retornar solo los ID de todos los dispositivos en el sistema.
 - 3.1.4. Retornar información de todas las medidas.
 - 3.1.5. Retornar información de una medida especificando su ID.
 - 3.1.6. Retornar todas las medidas registradas por un dispositivo.
 - 3.2. POST (añadir nuevos datos)
 - 3.2.1. Insertar nuevo dispositivo
 - 3.2.2. Insertar nuevas medidas.
 - 3.3. PUT (actualizar datos existentes)
 - 3.3.1. Actualizar dispositivo dado un ID.
-
- 4. **Fase 4 (Página Web)**
 - 4.1. La página web deberá proveer un apartado para cualquier usuario.
 - 4.1.1. Mostrar una página principal con información que explique brevemente el proyecto
 - 4.1.2. Mostrar una segunda página con información sobre cortes de luz, inicialmente el servidor debe convertir medidas en bruto en cortes de luz con un inicio y un fin.
 - 4.1.2.1. Mostrar un mapa con puntos donde se han producido los cortes de luz.
 - 4.1.2.2. Mostrar una tabla donde aparezcan los cortes de luz de la última semana.
 - 4.2. Ofrecer una forma de iniciar sesión como administrador.
 - 4.3. También deberá proveer un apartado para administradores del sistema en la página de información de cortes de luz.
 - 4.3.1. El administrador podrá filtrar los cortes de luz mediante los siguientes parámetros.
 - 4.3.1.1. Fecha de inicio y finalización del corte de luz.
 - 4.3.1.2. Id de dispositivo.
 - 4.3.2. El administrador podrá descargar un archivo .csv con todas las medidas en el rango de fecha seleccionado.

4.3.3. El administrador podrá descargar un archivo .csv con todas las medidas en el rango de fecha seleccionado de un dispositivo concreto.

Requisitos NO funcionales.

1. La transferencia de datos entre los dispositivos de la fase 1 y 2 debe de ser menor a 3 segundos.
2. La transferencia de datos entre dispositivos de las fases 2, 3 y 4 debe ser menor a 1.5 segundos
3. Todas las mediciones, inserciones e intercambio de fechas entre dispositivos, deben realizarse mediante el horario UTC, aunque en la página web, la hora se muestre en formato hora local del servidor.
4. El intercambio de datos entre la app móvil y la API REST y la página web debe realizarse mediante estructura JSON.
5. El sistema debe estar disponible los 365 días del año.
6. El dispositivo monitor (fase 1) estará vigilando cambios de estado cada segundo.

Requisitos de información.

1. Debe haber un registro de dispositivos en la base de datos. Un dispositivo almacenará la siguiente información:
 - a. Id del dispositivo.
 - b. Nombre de la persona que tiene instalado el dispositivo en casa.
 - c. Dirección donde está instalado.
 - d. Email de la persona de contacto.
 - e. Teléfono de la persona de contacto.
 - f. Longitud.
 - g. Latitud.
2. Se deben almacenar todas las medidas registradas por los dispositivos en la base de datos ligadas al dispositivo que las detecta. Las medidas tendrán información sobre:
 - a. Fecha y hora en la que se produce.
 - b. Tipo de medida (inicialización, subida de tensión, bajada de tensión).
 - c. Id del dispositivo que la capta.

CASOS DE USO

Los casos de uso son descripciones de una actividad o proceso, es decir, una funcionalidad independiente que nos ofrece un sistema para un fin concreto. Estos procesos son realizados por los llamados actores. Un actor es el nombre que se le da a la persona o entidad que realiza dicha acción.

Diagrama de casos de uso

Por otra parte, tenemos lo que se llama diagrama de casos de uso. Esto es la representación de un sistema por medio de relaciones entre casos de uso y actores. A continuación, se muestra el sistema dividido en tres subsistemas, dependiendo en qué ámbito actúe ese caso de uso. Tendremos un subsistema formado por el dispositivo monitor y la app móvil. Después, un subsistema formado por el dispositivo monitor de medidas, la app móvil y la API y finalmente un último subsistema formado por la API y la página web.

Definiremos tres tipos de actores, en primer lugar, un usuario general, que puede ser cualquier persona que quiera visitar la página web. En segundo lugar, tendremos a un usuario colaborador, que será aquel que tenga en casa un dispositivo de monitorización. El usuario colaborador heredará las funciones del usuario general. Nuestro último actor será el administrador. Este actor heredará las funciones de los dos actores anteriores y además, tendrá relación con sus casos de uso específicos.

Los casos de uso que definiremos en el sistema son los siguientes.

- Ajustar hora del dispositivo.
- Obtener datos de configuración del dispositivo.
- Transmitir medidas.
- Inicializar dispositivo.
- Actualizar dispositivo.
- Login/Logout.
- Mostrar cortes de la última semana.
- Mostrar cortes con filtro.
- Descargar CSV con todas las medidas.
- Descargar CSV con medidas de un dispositivo

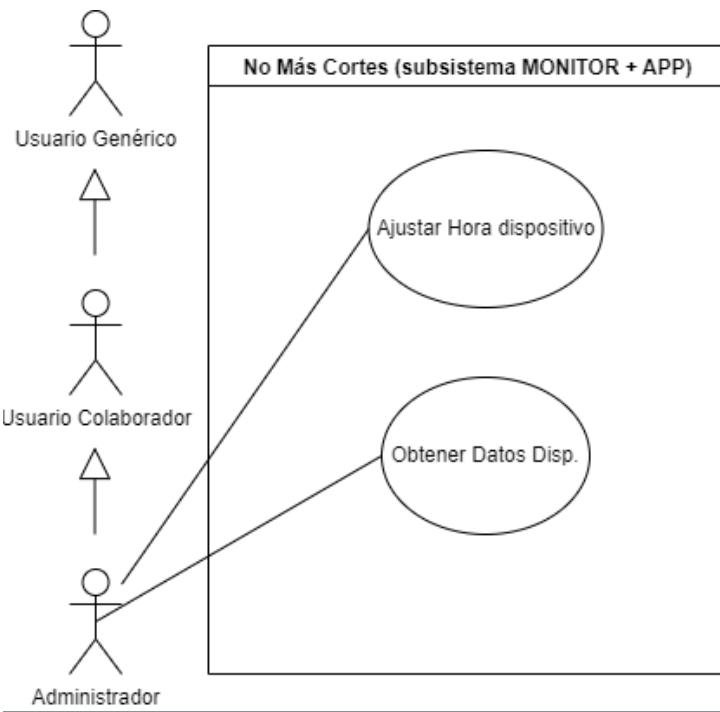


Figura 3. Diagrama de casos de uso del subsistema formado por el monitor y la aplicación móvil.

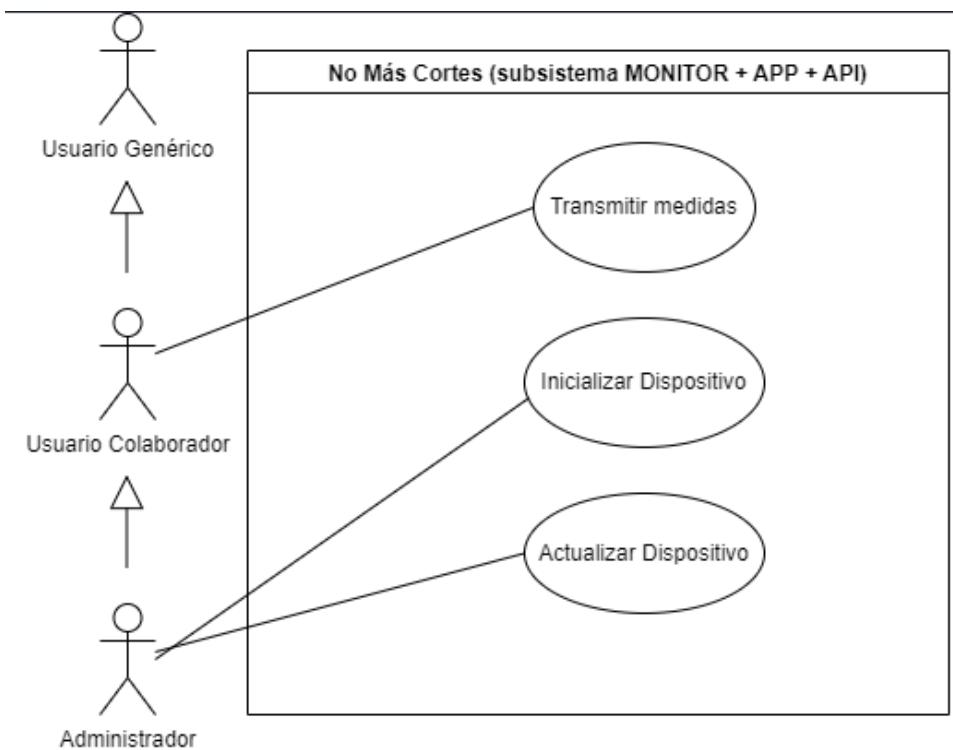


Figura 4. Diagrama de casos de uso del subsistema formado por el monitor, la aplicación móvil y la API.

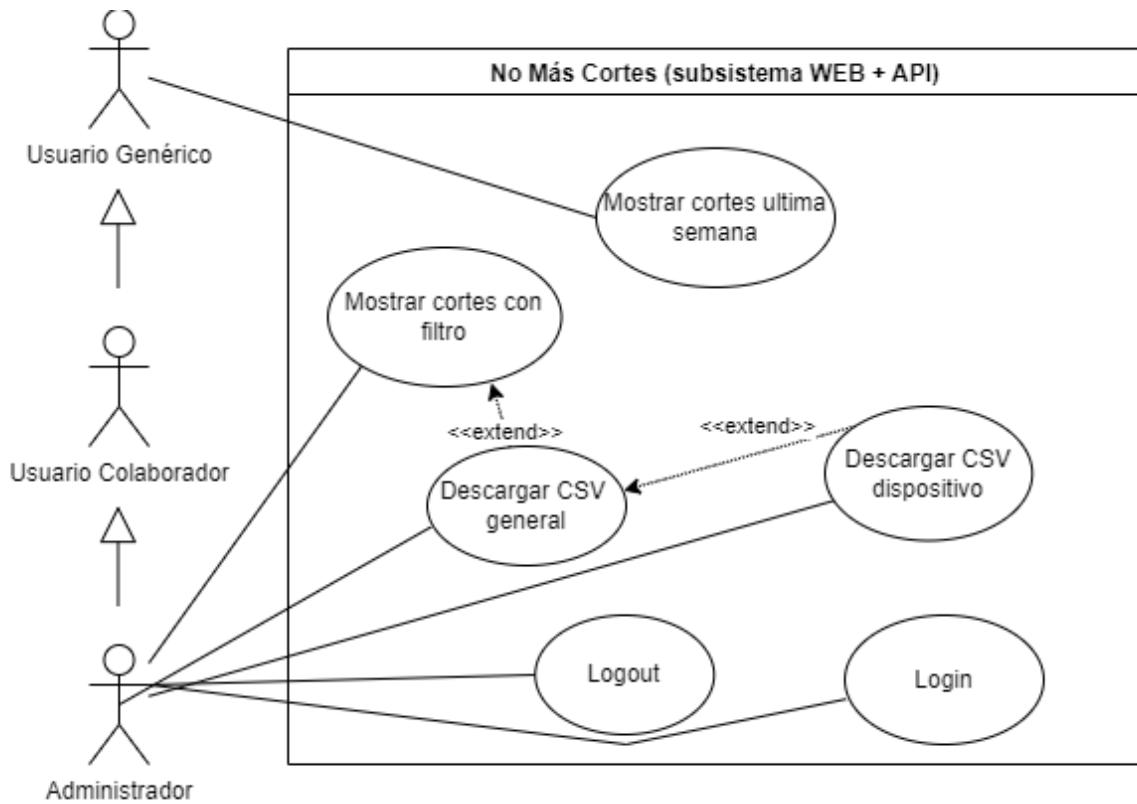


Figura 5. Diagrama de casos de uso del subsistema formado por la página web y la API.

Descripción de casos de uso.

En este apartado nos centraremos en describir más a fondo cada uno de los casos de uso que tenemos en nuestro sistema. Además acompañaremos cada caso de uso con un diagrama de flujo que lo represente.

Caso de uso	Ajustar hora del dispositivo.
Actores	Administrador.
Precondiciones	-
Postcondiciones	La hora del reloj en el dispositivo monitor es actualizada.
Propósito	Actualizar la hora del reloj para que sea lo más fiel posible al horario real.
Resumen	Desde el dispositivo móvil se envía un mensaje al dispositivo monitor junto a una fecha y hora para que el reloj del dispositivo se actualice.

Cada color en los diagramas de flujo representa una parte del sistema:

- Marrón = aplicación móvil.
- Azul = dispositivo de monitorización.
- Verde = API REST.
- Rojo = página web.

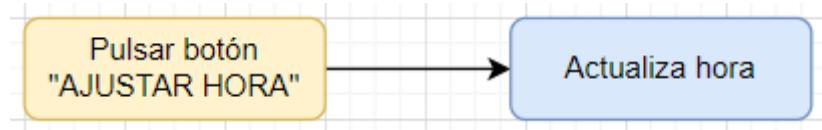


Figura 6. Flujo de datos del caso de uso “Ajustar hora del dispositivo”.

Caso de uso	Obtener datos de configuración del dispositivo.
Actores	Administrador.
Precondiciones	El dispositivo monitor está inicializado.
Postcondiciones	Llega un mensaje con la información del dispositivo.
Propósito	Obtener la información de configuración del dispositivo.
Resumen	El administrador desea saber la información de contacto y configuración del dispositivo, por lo tanto, pulsa el botón preparado para ello. La app móvil manda un mensaje para que le devuelva la información requerida al dispositivo monitor. Finalmente, la información se muestra en la pantalla del dispositivo móvil.

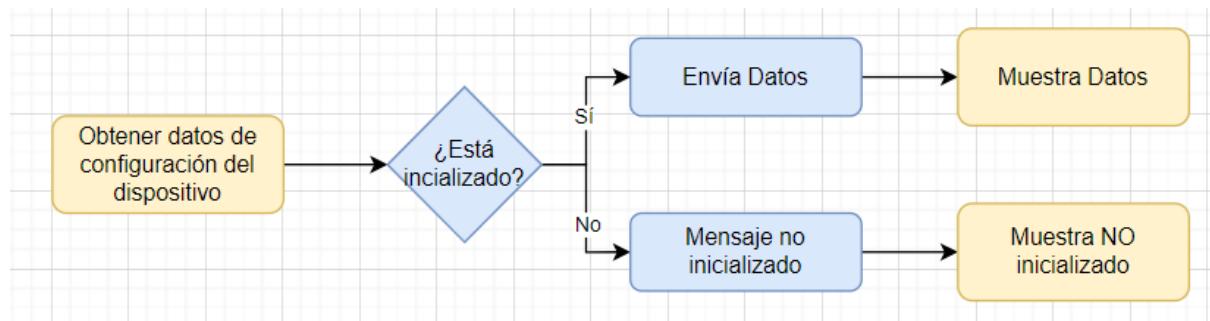


Figura 7. Flujo de datos del caso de uso “Obtener datos de configuración del dispositivo”.

Caso de uso	Transmitir medidas.
Actores	Usuario Colaborador.
Precondiciones	El dispositivo monitor está inicializado.
Postcondiciones	Se almacenan en la base de datos las medidas captadas.
Propósito	Guardar la información de las medidas registradas permanentemente en la base de datos.
Resumen	Un usuario que tiene un dispositivo en su casa pulsa el botón de la aplicación móvil para la transmisión de medidas entre el dispositivo y la base de datos. El dispositivo de monitoreo manda un mensaje con todas las medidas registradas y la aplicación móvil manda una petición a la API para almacenarlas. En caso de que se inserten correctamente, el dispositivo de monitoreo elimina de su registro todas las medidas para almacenar las nuevas que se produzcan. En caso de que haya algún error en la inserción, el dispositivo monitor no borrará los registros que haya acumulado.

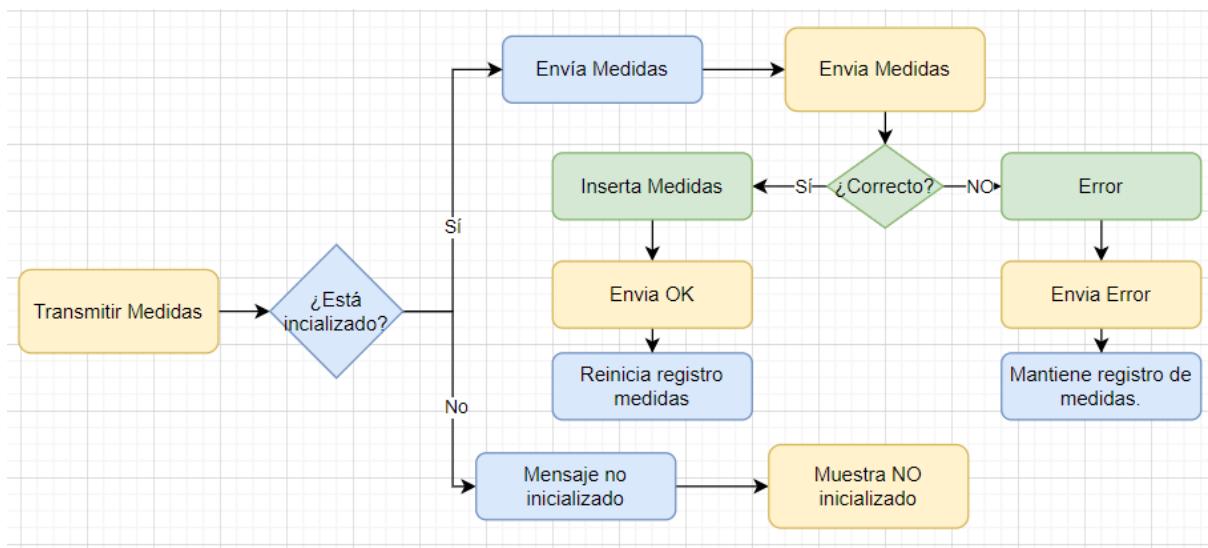


Figura 8. Flujo de datos del caso de uso “Transmitir medidas”.

Caso de uso	Inicializar dispositivo.
Actores	Administrador.
Precondiciones	El dispositivo de monitoreo no está inicializado.
Postcondiciones	El dispositivo almacena la información de configuración que se le proporciona.
Propósito	Cada dispositivo debe saber su propia información en caso de que tenga que ser consultada y para realizar las operaciones necesarias.
Resumen	Desde el dispositivo móvil, se rellenan los datos del nuevo dispositivo a insertar. Seguidamente, se envía una petición a la API para guardar un nuevo dispositivo en la base de datos, el nuevo ID insertado se devuelve a la aplicación móvil. Finalmente, mediante un mensaje enviado desde el dispositivo móvil, el dispositivo de monitoreo guarda su información de configuración.

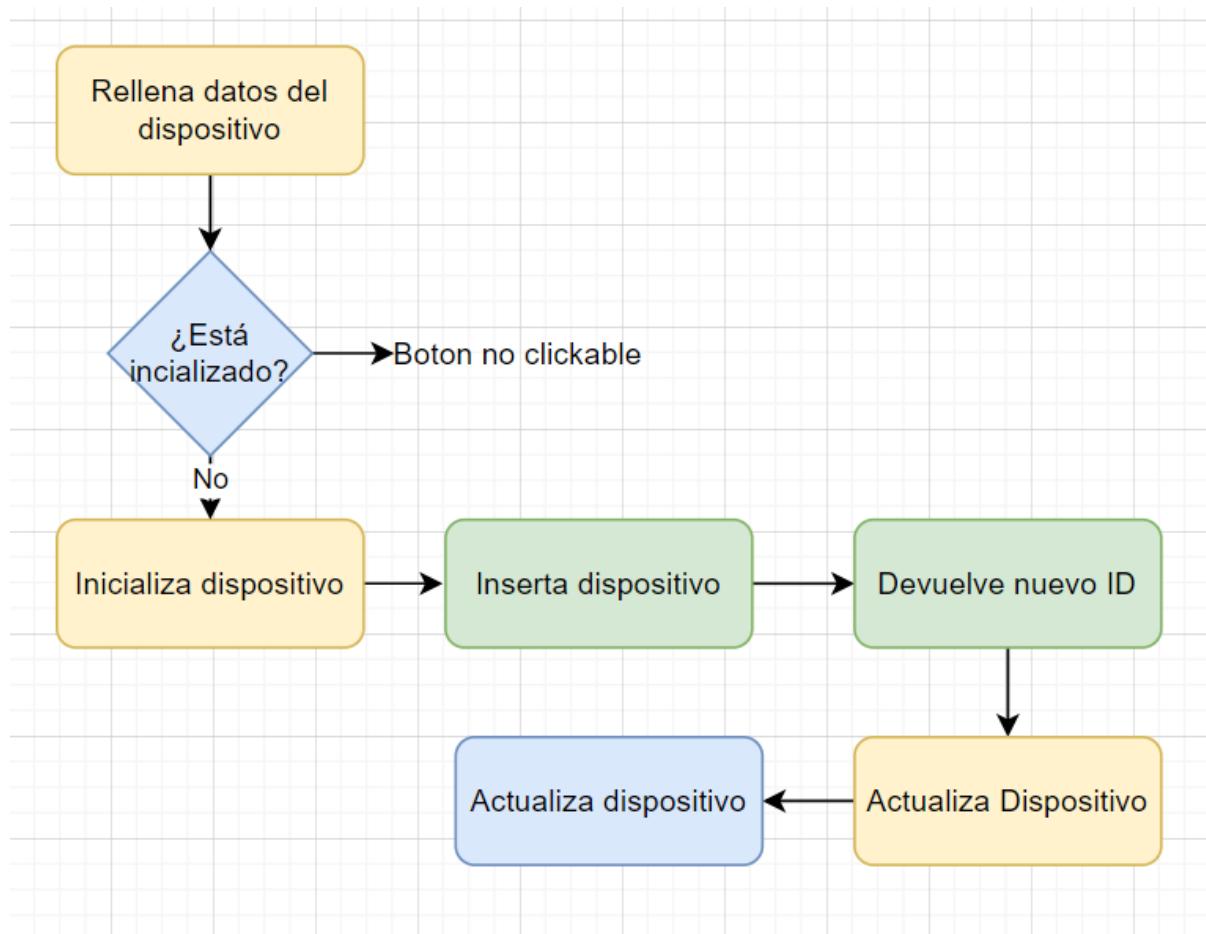


Figura 9. Flujo de datos del caso de uso “Inicializar dispositivo”.

Caso de uso	Actualizar dispositivo.
Actores	Administrador.
Precondiciones	El dispositivo de monitoreo está inicializado.
Postcondiciones	Queda actualizada la información del dispositivo en todas las partes del sistema.
Propósito	Tener la opción de actualizar la información del dispositivo en caso de que sea necesario cambiarla en algún momento.
Resumen	Desde el dispositivo móvil, se rellenan los datos de forma que estén actualizados. Seguidamente, se envía una petición a la API para guardar un nuevo dispositivo en la base de datos, el nuevo ID insertado se devuelve a la aplicación móvil. Finalmente, mediante un mensaje enviado desde el dispositivo móvil, el dispositivo de monitoreo guarda su información de configuración.

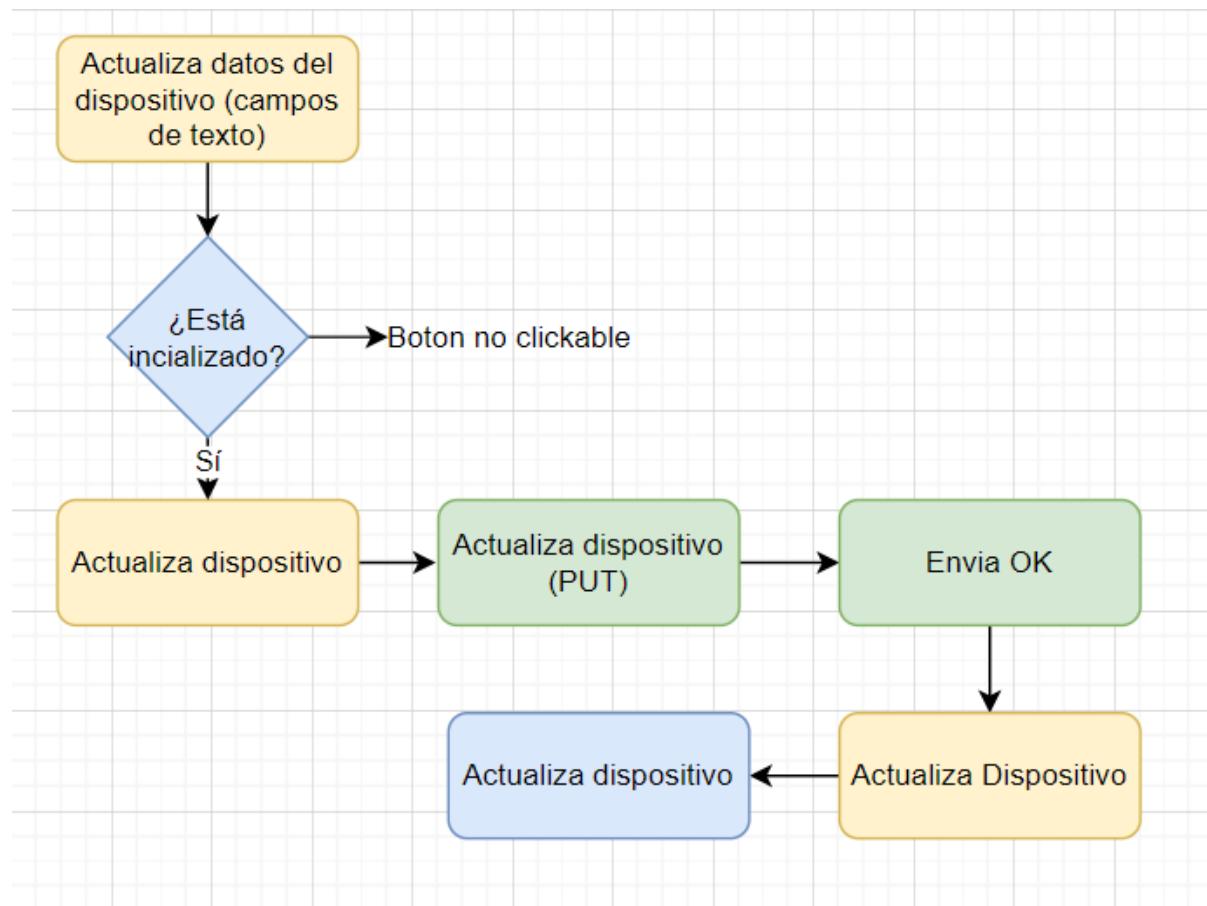


Figura 10. Flujo de datos del caso de uso “Actualizar dispositivo”.

Caso de uso	Login.
Actores	Administrador.
Precondiciones	No se ha iniciado sesión.
Postcondiciones	Se inicia sesión en la página web como administrador.
Propósito	El administrador debe tener acceso a funciones exclusivas dentro del sistema.
Resumen	El administrador envía un formulario con un nombre de usuario ‘admin’ y una contraseña y accede a la página web en modo administrador.

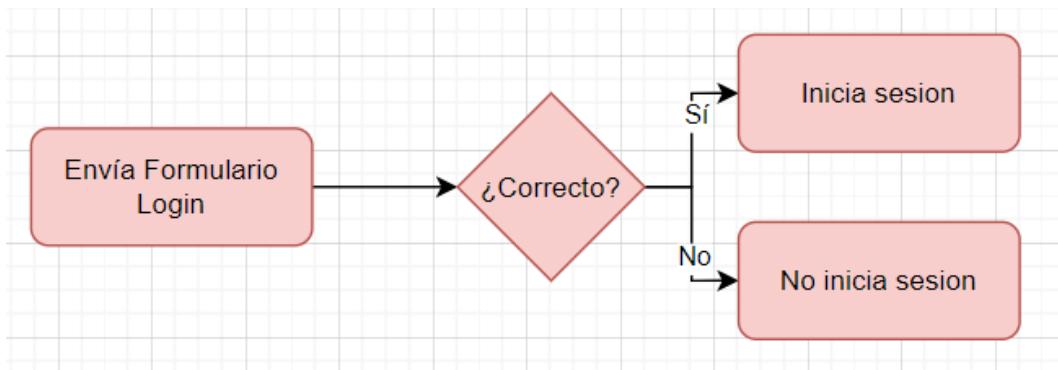


Figura 11. Flujo de datos del caso de uso “Login”.

Caso de uso	Logout
Actores	Administrador
Precondiciones	Hay una sesión iniciada como administrador.
Postcondiciones	La sesión abierta es eliminada.
Propósito	Dar la opción de volver al modo usuario general.
Resumen	El administrador pulsa el botón de cerrar sesión en la página web y se cierra su sesión como administrador.

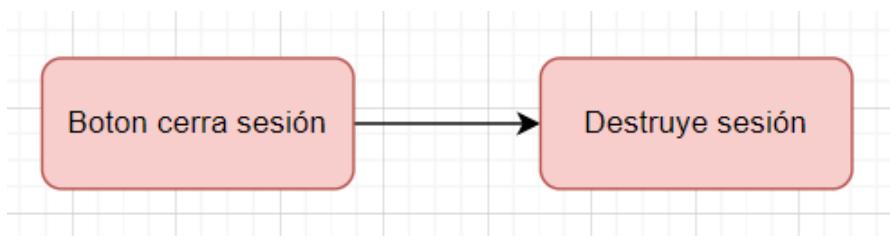


Figura 12. Flujo de datos del caso de uso “Logout”.

Caso de uso	Mostrar cortes de la última semana.
Actores	Usuario General
Precondiciones	-
Postcondiciones	Se muestran gráficamente los cortes de luz de la última semana.
Propósito	Proveer de herramientas de consulta a los usuarios externos que se interesen en el proyecto.
Resumen	El usuario entra en la página web y va al apartado de Mapa/Datos donde se le mostrará un mapa y una tabla con los cortes de luz que se han producido en la última semana.

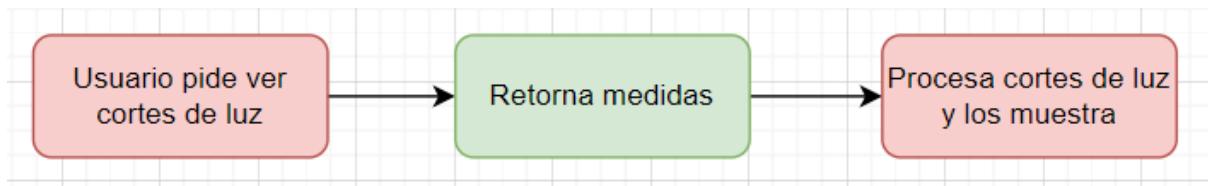


Figura 13. Flujo de datos del caso de uso “Mostrar cortes de la última semana”.

Caso de uso	Mostrar cortes con filtro.
Actores	Administrador
Precondiciones	Se ha iniciado sesión como administrador.
Postcondiciones	Se muestran gráficamente los cortes de luz que cumplen con los filtros seleccionados
Propósito	Proveer una herramienta a los administradores para que sean capaces de filtrar datos y así obtener diferentes consultas.
Resumen	El administrador elige una fecha de inicio y de fin. Entonces, al pulsar el botón de filtrar aparecen solamente los cortes de luz producidos en ese rango de tiempo. También puede ser que el administrador desee sólo los cortes de luz de un dispositivo, por lo tanto, debe seleccionar de nuevo un intervalo de tiempo y seleccionar el dispositivo que desea. Finalmente se le mostrarán los cortes de luz que cumplan las condiciones.

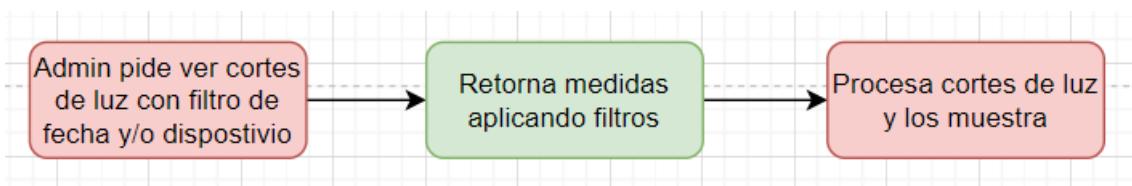


Figura 14. Flujo de datos del caso de uso “Mostrar cortes con filtro”.

Caso de uso	Descargar CSV con todas las medidas.
Actores	Administrador.
Precondiciones	Extiende “Mostrar cortes con filtro.”
Postcondiciones	Descarga automáticamente un archivo .csv con todas las medidas registradas en la base de datos.
Propósito	Que se puedan obtener los datos de medidas en bruto.
Resumen	Una vez que el usuario haya realizado el caso de uso “Mostrar cortes con filtro”, aparecerá un botón que, al pulsarlo, generará un archivo .csv en las descargas del navegador. Este archivo contendrá todas las medidas registradas en el sistema.

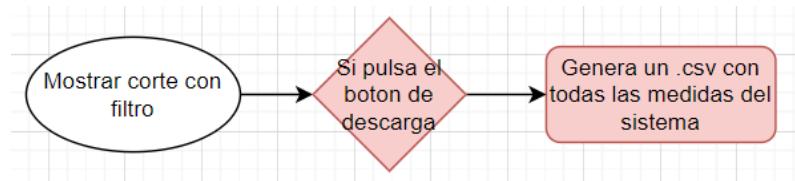


Figura 15. Flujo de datos del caso de uso “Descargar CSV con todas las medidas”.

Caso de uso	Descargar CSV con medidas de un dispositivo.
Actores	Administrador.
Precondiciones	Extiende “Descargar CSV con todas las medidas” mientras que esté activado el filtro de selección de un dispositivo concreto
Postcondiciones	Descarga automáticamente un archivo .csv con todas las medidas registradas en la base de datos por un dispositivo determinado.
Propósito	Que se puedan obtener los datos de medidas en bruto de un solo dispositivo.
Resumen	Este caso realizará la misma función que el caso de uso anterior, la diferencia es que se activará un botón que permitirá descargar solamente las medidas de un único dispositivo en un archivo .csv.

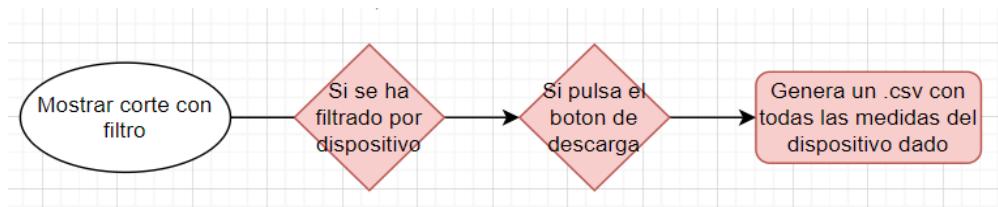


Figura 16. Flujo de datos del caso de uso “Descargar CSV con medidas de un dispositivo”.

DISEÑO

Una de las fases necesarias en el desarrollo software es el diseño. Es una etapa muy importante porque va a definir una solución que cumpla todos los requisitos propuestos en el análisis, la fase anterior.

Hay que tener muy en cuenta que esta fase de diseño es bastante crítica, ya que un cambio cuando hayamos pasado de fase puede ser fatal y sumar muchos cambios y retrasos.

En este caso vamos a dividir el diseño en tres partes: en primer lugar deberemos decantarnos por un modelo de base de datos y definir su estructura. La segunda parte será ver la arquitectura que tendrán cada una de las partes de nuestro software. Finalmente, una parte esencial, será definir las diferentes interfaces de usuario, tanto de la página web como de la aplicación móvil, teniendo en cuenta que entre el público de nuestra aplicación móvil puede haber gente de avanzada edad.

VISIÓN GENERAL DEL SISTEMA

En la siguiente ilustración podemos observar cómo va a estar confeccionado el sistema. Habrá una comunicación Bluetooth entre el prototipo de monitorización y la aplicación móvil. Por la otra parte, las peticiones y respuestas HTTP se encargarán de comunicar la aplicación móvil y el servidor que desplegará la API REST y la aplicación web.

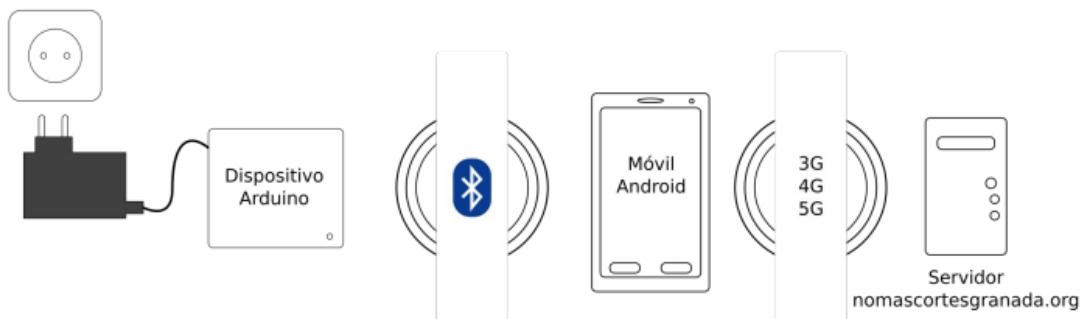


Figura 17. Visión general del sistema.

MODELADO DE LA BASE DE DATOS

Comenzaremos nuestra parte de diseño con el modelado de la base de datos. Como sabemos, desde hace unos años, el mundo de las bases de datos se divide en dos: bases de datos relacionales y no relacionales. Lo primero que tenemos que hacer es responder la siguiente pregunta: ¿Necesitamos una base de datos SQL o NoSQL?

Para ello, debemos saber primero qué ofrece cada una y qué ventajas aporta una sobre la otra y viceversa.

Base de datos relacional (SQL)

Las bases de datos relacionales son un tipo de base de datos que almacena y proporciona acceso a datos relacionados entre sí. Están basadas en un modelo de relación entre los datos que, de forma intuitiva se puede representar en tablas. Cada registro de una tabla se puede almacenar mediante un identificador y los campos que se consideren necesarios.

El modelo relacional ha sido capaz de proporcionar una forma estandarizada para representar y consultar los datos de una o varias aplicaciones. Por lo tanto, una base de datos implementada de manera relacional puede ser consultada por muchas aplicaciones diferentes, ya que el estándar es el mismo.

Este estándar que se comenta es SQL (Structured Query Language), está basado en álgebra relacional, es decir, provee de un lenguaje matemático consistente que ayuda a la hora de hacer consultas en la base de datos de tipo relacional.

Base de datos no relacional (NoSQL)

Las bases de datos no relacionales destacan por la no estricta organización en tablas de los sistemas relacionales tradicionales. No hay una sola definición de lo que es NoSql (Not only SQL), ya que se puede dividir en varios tipos.

Por ejemplo, existen bases de datos basadas en clave valor, las más simples, en la que cada elemento se almacena con una clave y un valor asociado a ella. También tenemos las basadas en grafos, donde se guarda la información encima de redes de datos.

Otro tipo serían las bases de datos documentales, que almacenan información como si de un documento se tratara, gracias a JSON o XML. Este tipo de almacenamiento extiende el de las bases de datos basadas en clave valor, ya que ofrece esa funcionalidad y además, consultas más avanzadas sobre el contenido del documento. Como último ejemplo, hablaremos de las bases de datos orientadas a objetos, donde la información se representa, como bien indica su nombre, mediante objetos, de la misma forma que en los lenguajes de programación orientados a objetos.

Ventajas SQL

- Es más ventajoso cuando hay un claro esquema de datos en el sistema, ya que podemos hacer muchas consultas diferentes y complejas.
- Más consistente, es capaz de garantizar integridad de datos.
- Mayor soporte, por la cantidad de años acumulados en el mercado como estándar.
- Tipo y verificación de datos

Ventajas NoSQL

- NoSQL es más útil cuando hay grandes picos de peticiones por parte de usuarios y aplicaciones.
- Es mejor para un volumen de datos muy grande, por su rapidez al indexar.
- Se pueden distribuir los datos más fácilmente por localización geográfica.
- Versatilidad, ya que podemos insertar registros con campos diferentes a inserciones anteriores.
- Para proyectos pequeños no necesitan de gran capacidad computacional en el servidor.

Conclusión (elección final)

Teniendo en cuenta todos los puntos anteriores y las necesidades de nuestro proyecto, nos decantamos por seguir una estructura de datos relacional (SQL). Las razones son las siguientes:

1. Nuestro proyecto tiene que almacenar la información de dispositivos y medidas. Además los campos de cada una de estas entidades son fijos. Esto se traduce en un esquema bastante simple que encaja muy bien dentro de un entorno relacional con dos tablas.

2. El volumen de datos que manejará nuestra base de datos no será muy grande y crecerá muy progresivamente, por lo tanto no es necesario la potencia de un sistema NoSQL.
3. El sistema estará formado por un solo servidor, no será necesaria la distribución de datos, otra razón más para decantarse por SQL.
4. Finalmente, y no menos importante, como desarrollador del proyecto, he trabajado en diferentes ocasiones con tecnología SQL, por lo tanto es otro motivo que ayuda a la decisión final.

Diagrama entidad-relación de la base de datos No Más Cortes

Como se ha comentado anteriormente, necesitaremos almacenar la información de dispositivos y medidas detectadas, por lo tanto generamos la entidad dispositivo y la entidad medida. Cada una de estas entidades, para cumplir con la especificación de requisitos, incluirá los siguientes campos:

- Dispositivo
 - Identificador.
 - Dirección.
 - Persona de contacto (usuario colaborador).
 - Teléfono de contacto.
 - Email de contacto.
 - Latitud.
 - Longitud.
- Medida
 - Identificador.
 - Tipo de medida.
 - Fecha.
 - Identificador del dispositivo que la capta.

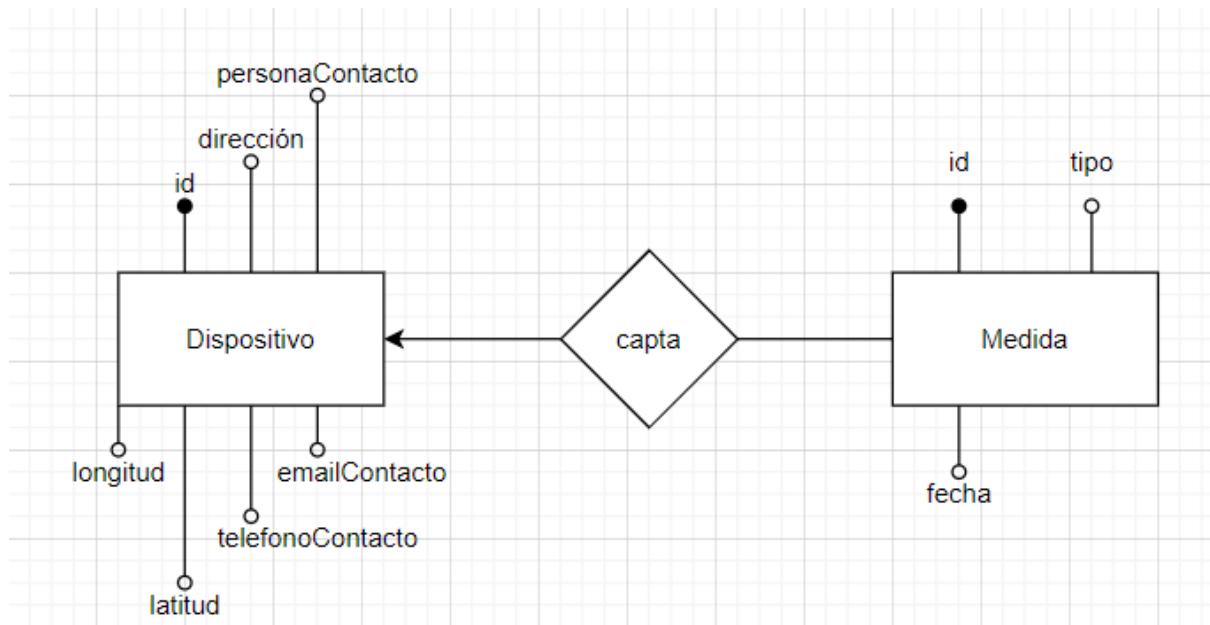


Figura 18. Modelo entidad relación de la base de datos.

En la fase de análisis se habla constantemente del usuario administrador. Pero como vemos, no aparece en el modelo entidad relación de la base de datos. La respuesta a esto es la siguiente, no se ha necesitado guardar la información de administradores. Directamente, en la fase de implementación, se usarán herramientas que permitan la identificación de un usuario como administrador sin la necesidad de guardar su información en la base de datos.

ARQUITECTURA DEL SISTEMA

Fase 1: Diseño de prototipo monitor mediante Arduino.

En esta fase, vamos a definir cómo vamos a diseñar el prototipo monitor de cortes de luz. En primer lugar, sabemos que existen diferentes tecnologías en la actualidad que nos podrían ser útiles en el desarrollo de esta tarea, por lo tanto el primer paso será ver qué opciones tenemos y cuál elegiremos.

Opción 1: Arduino



Arduino es una plataforma donde podremos crear nuestros proyectos de electrónica. Es de código abierto y está basado en hardware y software libre. Es por esto que podemos comprar en el mercado diferentes módulos, placas, aplicaciones, etc de otras empresas y desarrolladores.

Arduino provee la herramienta Arduino IDE, es una entorno de programación donde escribiremos el código que nuestro prototipo ejecutará. Este proyecto se inició en 2003 por estudiantes de una universidad Italiana, con el propósito de acercar la electrónica a otros estudiantes con pocos recursos económicos.

Opción 2: Raspberry Pi



Raspberry Pi es, a fin de cuentas, un pequeño ordenador muy simple, reducido y de bajo coste. Es suficientemente potente como para poder programar y compilar programas que se ejecuten en estos dispositivos.

Es un producto muy versátil, con más potencia de cálculo que Arduino. Nació para desarrollar sobre él pequeños prototipos y para la formación de alumnos en electrónica en los colegios e institutos.

Su sistema operativo es de código abierto, pero a diferencia de Arduino, la parte de hardware sí que se mantiene oculta al público general. Cuenta con Wifi y Ethernet integradas por defecto en la placa.

Aunque su finalidad no sea esta, la comunidad ha estado aprovechando constantemente estos dispositivos de Raspberry Pi para pequeños proyectos.

Opción 3: Microcontrolador controlado por micro Python: NodeMCU

He añadido esta opción ya que puede parecer interesante, NodeMCU tiene en el mercado una placa versión V3 que tiene alguna característica que nos puede interesar. Para los dos opciones anteriores necesitamos un módulo para insertar una memoria microSD, pero la placa NodeMCU contiene una memoria flash de un total de 4MB que podría ser suficiente para nuestro propósito. Esta placa, diseñada para la conexión con otros dispositivos, trae de fábrica un chip para la conexión Wifi.

Para la elección final, he elegido como criterios, la facilidad para conseguir los kits de trabajo necesarios, las características que ofrecen las placas finalmente, el precio de cada una de ellas.

Entonces, la opción 1, una placa Arduino parece la opción más adecuada. En primer lugar, en la biblioteca del centro (Escuela Técnica Superior de Ingenierías Informática y de Telecomunicaciones) existen varios kits de desarrollo de Raspberry Pi y de Arduino a modo de préstamo, pero no de NodeMCU, por lo tanto, en primera instancia descartamos la opción 3, aunque pueda parecer interesante, y no se descarta su utilización en una versión posterior del proyecto.

Finalmente, la elección entre Raspberry Pi y Arduino es más simple, ya que Arduino nos ofrece muchísima versatilidad de componentes y de documentación por parte de la comunidad, además Raspberry Pi es varias veces más caro que Arduino y si llegara el momento de implementar el sistema en un entorno real, su precio subiría considerablemente.

Se puede ver en la siguiente ilustración cómo se organizarían los módulos y el cableado de Arduino para su definir su comportamiento posteriormente en el Arduino IDE.

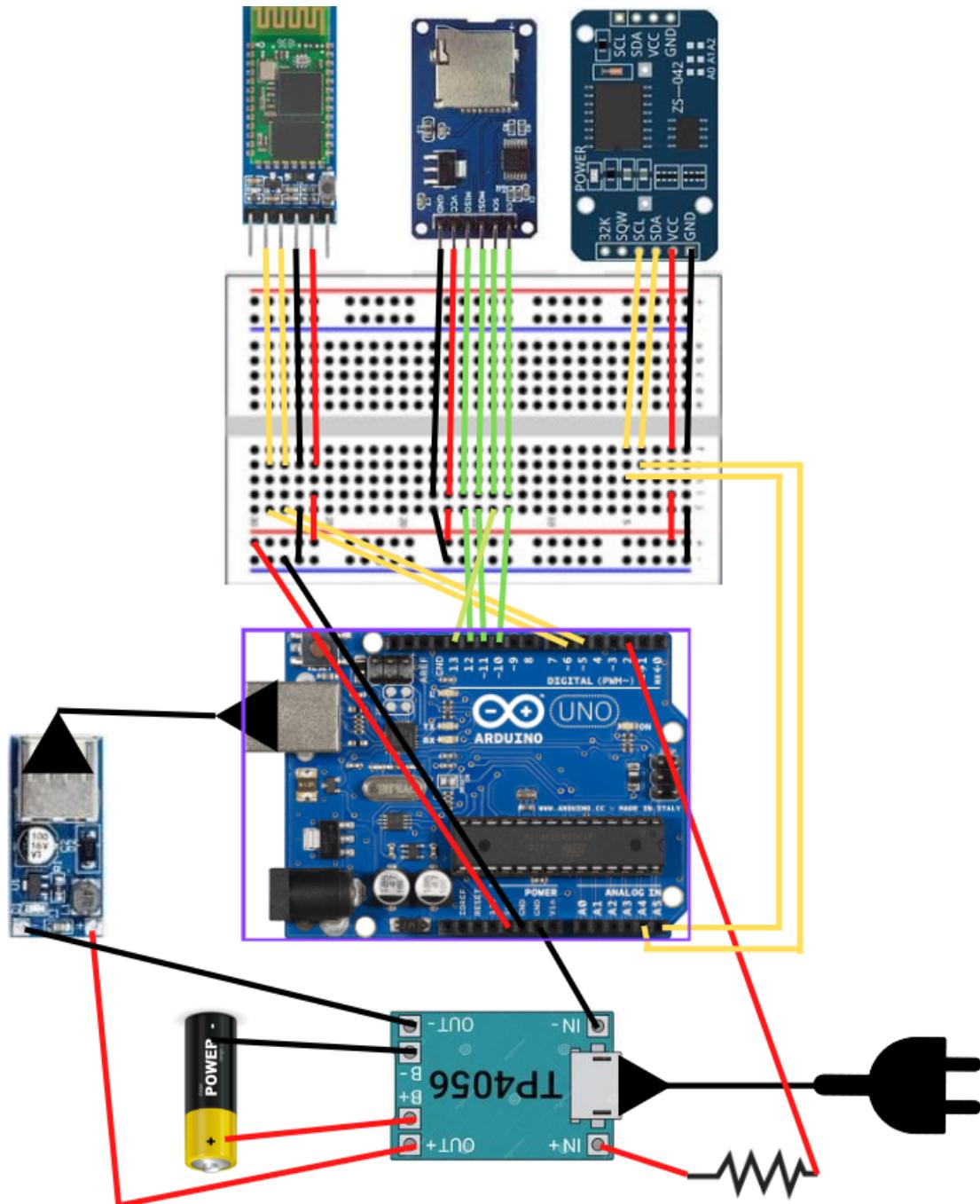


Figura 19. Esquema de conexiones de Arduino y módulos.

Fase 2: aplicación móvil

Pasaremos, en esta fase del diseño, a confeccionar la arquitectura de nuestra aplicación móvil. En primer lugar debemos señalar que vamos a desarrollar la aplicación en el entorno de desarrollo Android Studio, junto al lenguaje de programación Java, para construir una aplicación exclusivamente para Android. Las razones que me han llevado a ello han sido las siguientes.

1. Principalmente, he estado trabajando con Java durante gran parte de mi tiempo desde que inicié mis estudios, por lo tanto conozco el lenguaje y puedo aplicar fácilmente mis conocimientos para implementar los patrones y metodologías necesarias.
2. También estoy acostumbrado a manejar el IDE Android Studio.
3. Obtendremos gran cantidad de documentación y desarrollo de la comunidad.
4. Según los datos del artículo citado [12], el 72.5% del tráfico móvil en el mundo, viene de dispositivos Android, en contraposición del 26.9% de IOS. Además si hacemos una media de precio entre los dispositivos que usan Android y los que usan IOS, el dispositivo IOS es 2.7 veces más caro [13]. Por lo tanto, teniendo en cuenta que, el distrito norte de Granada no destaca por ser de las zonas más ricas de la capital, hay una alta probabilidad de que en una vivienda haya al menos 1 dispositivo Android que pueda usar nuestra aplicación para la transferencia de datos.

Patrón Singleton [14]

Explicaremos la utilidad del patrón Singleton, que será muy útil durante el desarrollo de nuestra aplicación, ya que necesitaremos hacer uso de él varias veces.

En nuestro caso, necesitaremos generar una conexión bluetooth y una conexión a una API, estas dos conexiones generan mensajes mediante un receptor (por ejemplo el adaptador bluetooth del teléfono móvil). Si tenemos en cuenta que sólo tenemos un receptor bluetooth y que éste solo podrá realizar operaciones de una en una, qué sentido tendría tener varias instancias de una misma conexión. Es por esto que Singleton servirá de ayuda.

El patrón Singleton es una de las formas más simples de patrón de diseño que existe. La intención del patrón es hacer que un objeto de una clase sea la única instancia en un sistema.

Para conseguir eso, podemos comenzar donde los clientes lo instancian. Así que tenemos que bloquear cada intento de creación de un objeto con un método que permita únicamente una instancia de la clase del objeto en la salida. Ese método debe ser un método estático (método de clase), ya que no tendría sentido permitir que una instancia de la clase cree otra instancia. Además, nuestra clase tiene que tener como atributo una instancia estática que refiera a la instancia única disponible en el sistema.

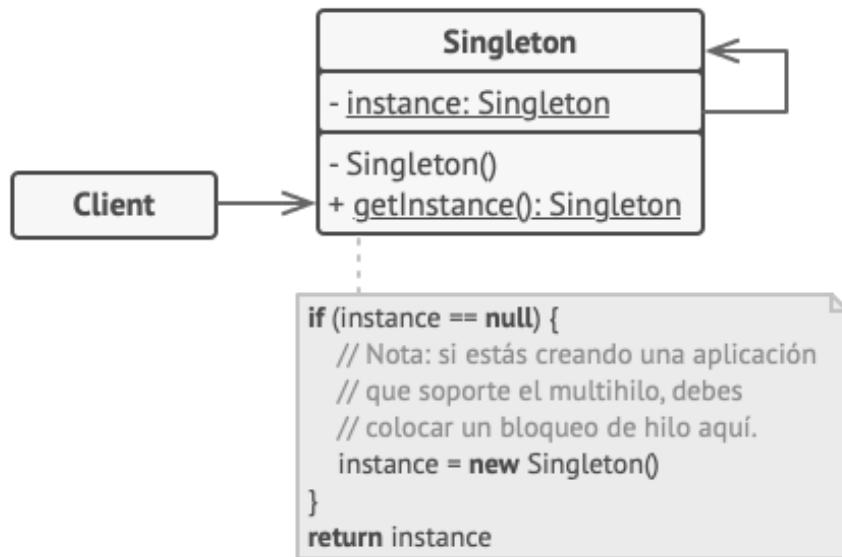


Figura 20 [15]. Zhart, D. (s.f). Patrón Singletón [Ilustración].

Diagrama de clases

A continuación se presenta el diagrama de clases que definirá el esquema de nuestra aplicación móvil. Tendremos dos pantallas, la primera, muy sencilla para la transmisión de medidas por parte de un usuario colaborador, y la segunda, para que maneje información del dispositivo el administrador.

Por otra parte, vemos las tres clases más importantes de nuestro esquema, que implementarán el patrón Singleton. La razón de implementar el patrón es que solo debe haber una conexión bluetooth simultánea, otra que actúe sobre la API y además, solo podrá haber una instancia de la clase dispositivo, ya que la conexión bluetooth solo la haremos hacia un dispositivo.

Finalmente, tendremos una hebra constantemente escuchando posibles mensajes provenientes del dispositivo monitor mediante Bluetooth.

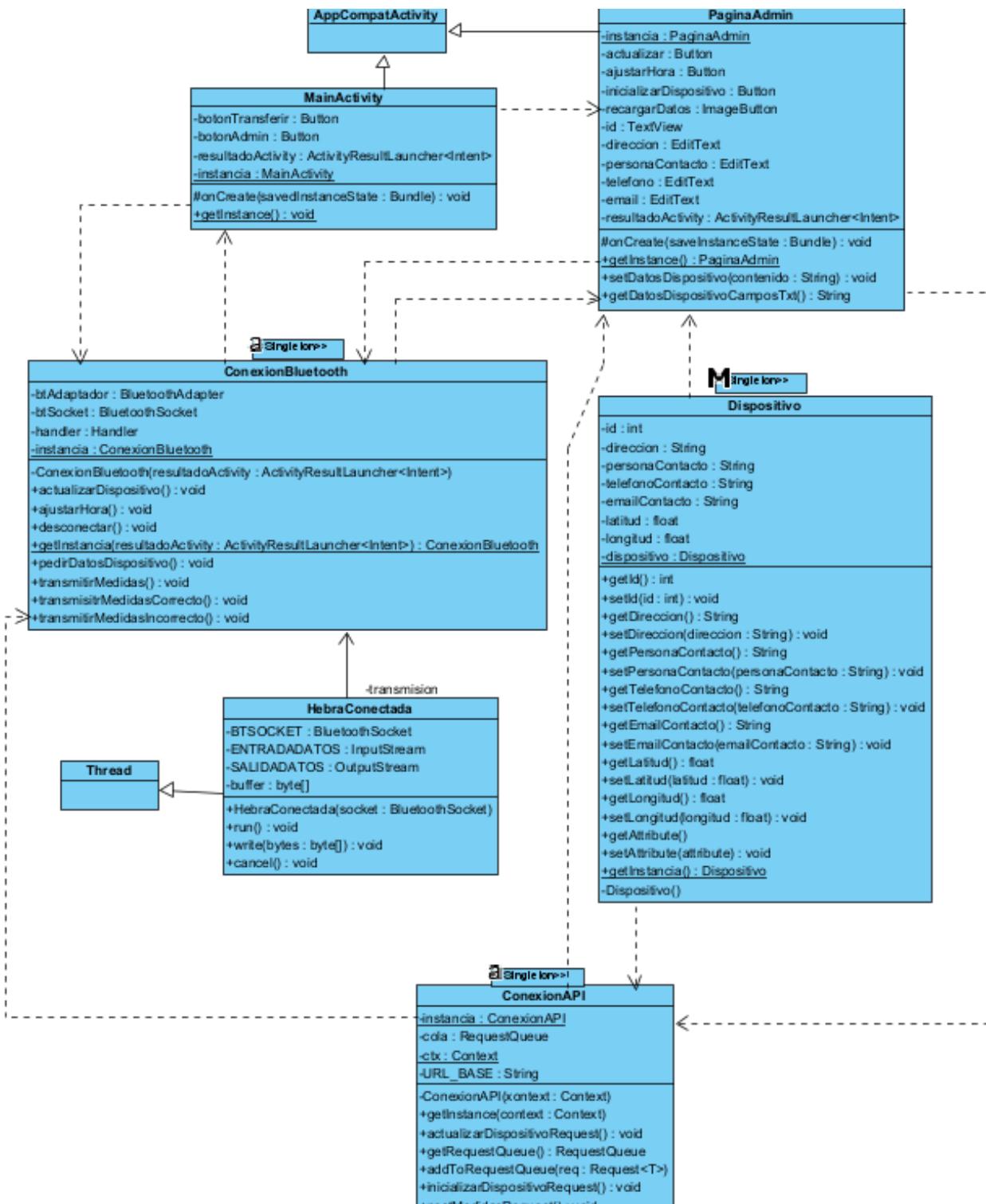


Figura 21. Diagrama de clases de la aplicación No Más Cortes.

Fase 3: API REST

Una vez está definida la estructura que tendrá nuestra base de datos, podemos pasar a definir nuestra API REST y las peticiones que nos ofrecerá para intercambio de datos en todo nuestro sistema.

Una API de REST es, según RedHat[16], una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. Es decir, una herramienta que nos permite interactuar con un computador para obtener datos o ejecutar una función. En nuestro caso, va a servir como una plataforma donde ofreceremos los datos que queremos mostrar, sin comprometer la seguridad y control de nuestro sistema.

Utilizaremos una estructura cliente-servidor, donde el tráfico de información se sustentará bajo el protocolo HTTP, en el formato JSON. Es conveniente saber que la comunicación RESTful es *stateless*, con esto nos referimos a que no se guarda información de una petición a otra, las peticiones HTTP son independientes.

Para nuestro sistema definiremos los siguientes tres tipos de peticiones: [17]

1. GET: este método solicita un recurso especificado. Las solicitudes que usan GET solo deben usarse para recuperar datos (no deben incluir datos).
2. POST: este método envía datos al servidor. El tipo del cuerpo de la solicitud es indicada por la cabecera “Content-Type”. En nuestro caso servirá para la adición de dispositivos o medidas.
3. PUT: este método crea un nuevo elemento o reemplaza una representación del elemento de destino con los datos de la petición. En nuestro caso servirá para la actualización de dispositivos.

Recordemos que vamos a necesitar hacer consultas y modificaciones sobre dispositivos y medidas. Por lo tanto vamos a tener una URL base de la que partirá la API:

$$URL\ BASE = \textit{https://}<<\textit{dominio}>>/\textit{api}$$

A partir de aquí definiremos las siguientes rutas para dispositivos y medidas.

Dispositivos

1. GET

- a. Consultar dispositivo por su identificador.

https://<<dominio>>/api/dispositivos/:id

- b. Consultar todos los dispositivos.

https://<<dominio>>/api/dispositivos

- c. Consultar todos los identificadores de dispositivos.

https://<<dominio>>/api/dispositivosID

2. POST

- a. Método de inserción de dispositivo (añadiendo al body de la petición un JSON con la información del dispositivo).

https://<<dominio>>/api/dispositivos

3. PUT

- a. Método de actualización de dispositivo (añadiendo al body de la petición un JSON con la información nueva del dispositivo).

https://<<dominio>>/api/dispositivos/:id

Medidas

4. GET

- a. Consultar medida por su identificador.

https://<<dominio>>/api/medidas/:id

- b. Consultar todas las medidas.

https://<<dominio>>/api/medidas

- c. Consultar todas las medidas de un dispositivo.

https://<<dominio>>/api/medidas/dispositivo/:id

5. POST

- a. Método de inserción de medida(añadiendo al body de la petición un JSON array con la información de las nuevas medidas a insertar).

https://<<dominio>>/api/medidas

Fase 4: página web

Una vez en la fase 4 de diseño, la página web, debemos tener en cuenta que, por una lado, tenemos que gestionar la información de nuestra base de datos y a su vez, ser capaz de mostrarla de una manera ordenada e intuitiva para los visitantes del sitio. Por lo tanto aplicaremos el patrón arquitectónico MVC (modelo, vista y controlador).

MVC [18] es un tipo de arquitectura que define lo siguiente: los datos, la interfaz de usuario y el procesamiento de datos se van a estructurar independientemente y se relacionarán entre ellos para hacer funcionar una aplicación (la página web en este caso).

El comportamiento es como explicaremos ahora. La vista es el componente que se encarga de generar la interfaz de usuario, muestra la información de manera gráfica para su captación de forma fácil por parte del usuario. La función del modelo será gestionar los datos del sistema, muchas veces el modelo puede interactuar con la vista para mandarle la información que necesita, aunque generalmente esta información pasa antes por el controlador. El controlador actúa de intermediario entre los dos componentes anteriores. Gestiona las acciones del usuario, realizando peticiones y actualizaciones en el modelo para ofrecer actualizaciones de información a la vista del usuario.

La forma en la que vamos a diseñar nuestra página web seguirá el siguiente esquema, donde la vista y el modelo no interactúan sin una mediación del controlador.

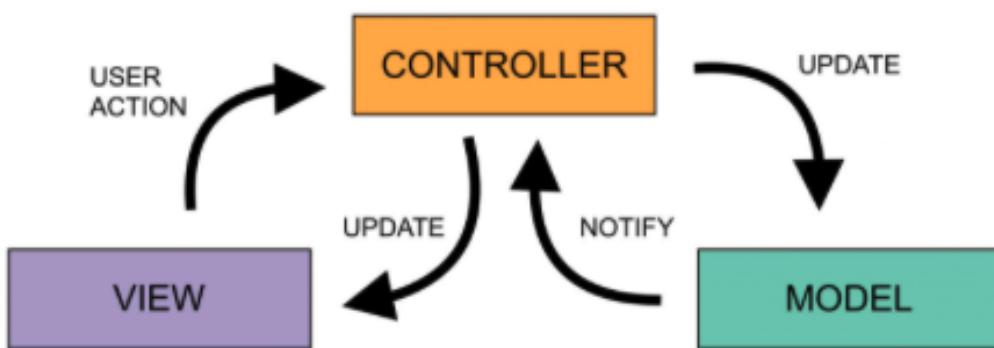


Figura 22 [19]. Tan, K. (2021, 8 junio). Model View Controller [Ilustración].

A continuación, se mostrará un pequeño diagrama de cómo estará compuesta la página web.

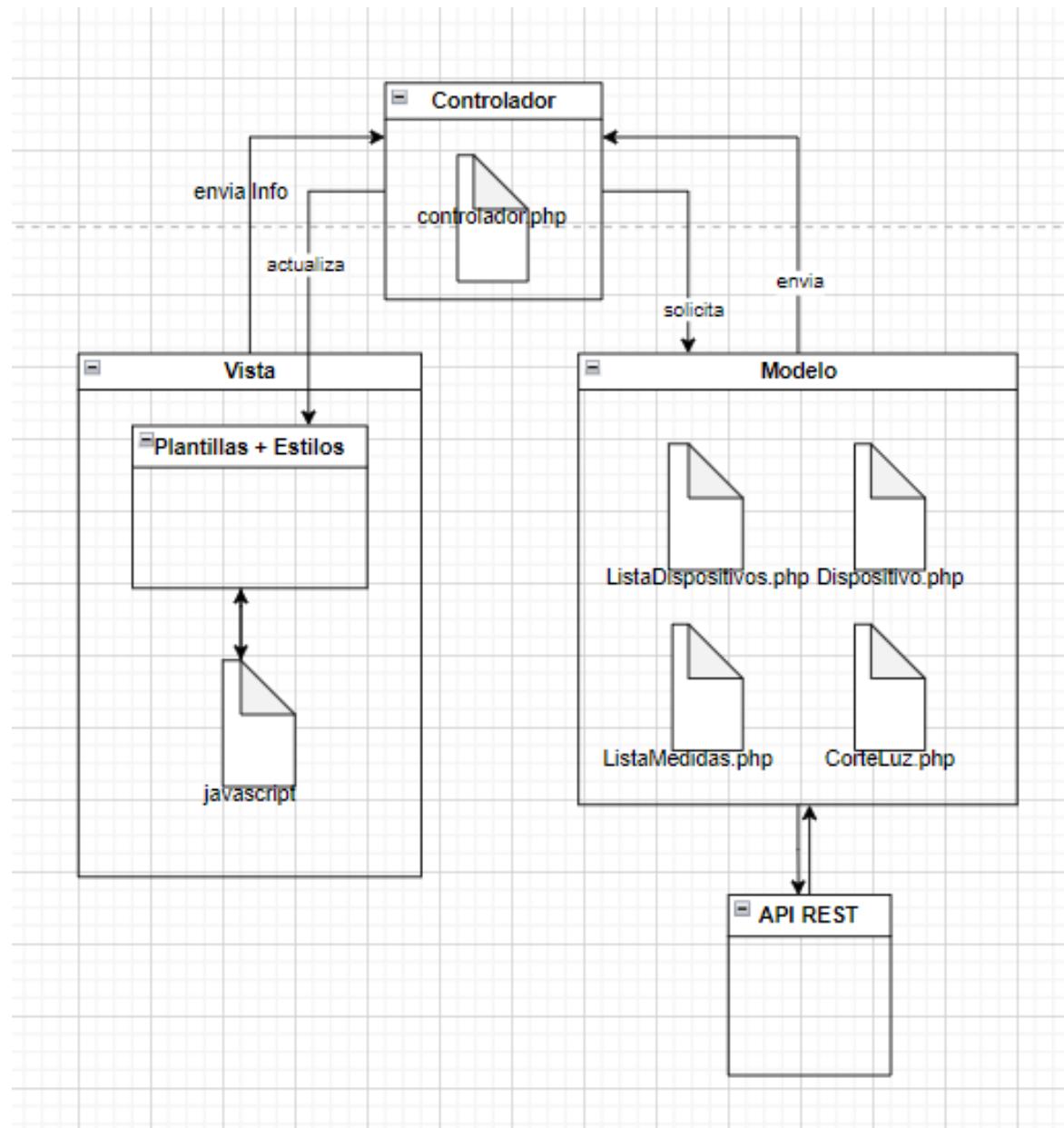


Figura 23. Diagrama del sistema Web No Más Cortes.

Para poder llevar a cabo el sistema propuesto en la figura 21, utilizaremos el gestor de plantillas Twig, que explicaremos en la parte de implementación.

INTERFACES DE USUARIO

Finalmente, para acabar con la fase de diseño, tendremos que dar una primera idea de cómo serán nuestras interfaces de usuario. El diseño de la interfaz de usuario es un método asociado al diseño industrial y prioriza maximizar la usabilidad y la experiencia de usuario. El objetivo final del diseño de la interfaz de usuario es hacer que la interacción entre el usuario y el sistema sea tan simple y eficaz como sea posible. Es a lo que llamamos diseño centrado en el usuario.

MockUps App No Más Cortes.

Al hablar con Sergio, tutor de este TFG, me advirtió que algunas personas que podrían usar la aplicación móvil del sistema No Más Cortes no son nativos tecnológicos, es decir, puede ser gente mayor o que no está acostumbrada a las nuevas tecnologías. Entonces debemos enfocar el desarrollo de la aplicación hacia la simplicidad y la usabilidad.

- Para las funciones de usuario colaborador hemos desarrollado el siguiente MockUp de la izquierda, la parte del administrador la tenemos a la derecha.

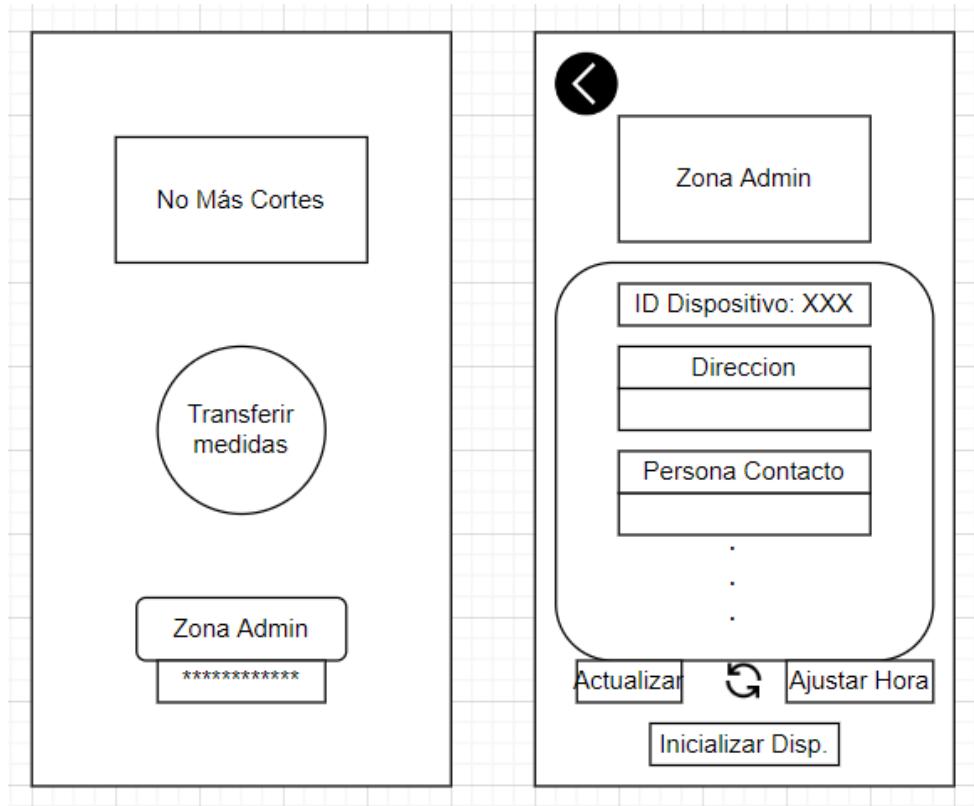


Figura 24. MockUp App No Más Cortes.

MockUps Web No Más Cortes.

Para este apartado presentamos dos tipos de MockUp, los primeros, para la visualización en pantallas de más de 800px de ancho, y los segundos para dispositivos móviles o tabletas con menos de 800px de ancho de pantalla.



Figura 25. MockUp landing page Web No Más Cortes.

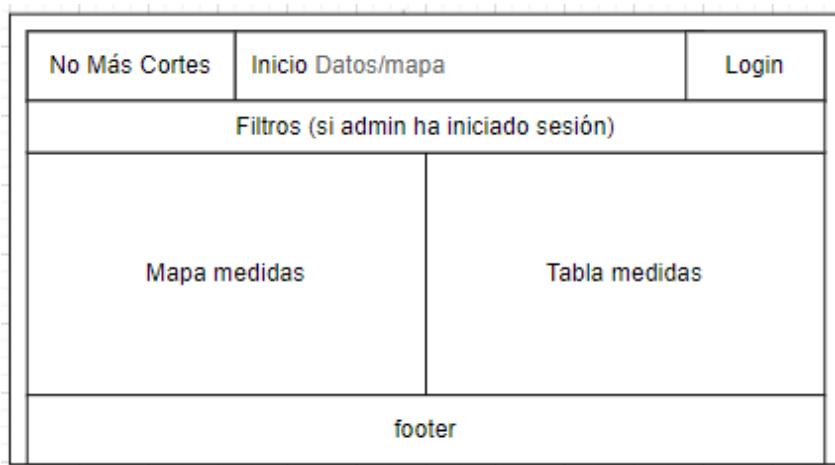


Figura 26. MockUp datos/mapa Web No Más Cortes.

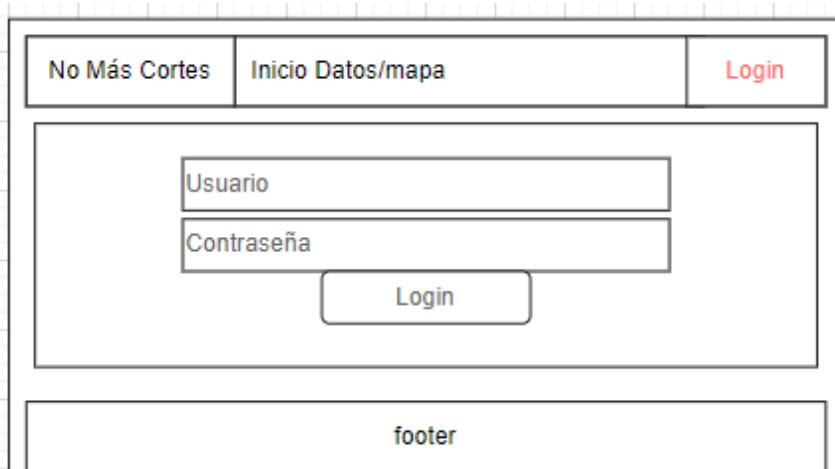


Figura 27. MockUp login Web No Más Cortes.



Figura 28. MockUp responsive de la Web No Más Cortes.

Paleta de colores

Otra parte que no podemos obviar del diseño de las interfaces de usuario es, la paleta de colores. Utilizaremos los siguientes tonos para la interfaz de la página web y la aplicación móvil:

Código de color	Color	Observaciones
#333333		Según algunos estudios, los colores generan diferentes emociones en las personas, por lo tanto, hemos elegido tonos negros para mostrar elegancia en nuestra aplicación.
#111111		Además, también se ha escogido el naranja, que genera en las personas confianza y alegría. Por lo tanto, intentaremos que aflore en nuestros usuarios, al recordar nuestra aplicación, esos sentimientos de connotación positiva.
#F15838		
#fda594		
#d5d5d5		

IMPLEMENTACIÓN

Entramos ya en la fase de implementación, donde se definirán las técnicas utilizadas y las partes críticas y no triviales del desarrollo de nuestra aplicación.

BASE DE DATOS

A continuación, se muestra el código de creación de las dos tablas que compondrán el modelo, de esta forma se podrá visualizar claramente qué tipos de datos usaremos para cada atributo. Los identificadores estarán definidos mediante AUTO INCREMENT, para que se asigne a la hora de la inserción por el Sistema Gestor de Base de Datos.

Haremos uso del SGBD MySQL, ya que es un muy buen producto, con mucha documentación y aporte de la comunidad y además tengo experiencia de trabajar con este en proyectos anteriores.

```
CREATE TABLE dispositivo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    direccion VARCHAR(40) NOT NULL,
    personaContacto VARCHAR(50) NOT NULL,
    emailContacto VARCHAR(30),
    telefonoContacto VARCHAR(15),
    latitud DECIMAL(8,6),
    longitud DECIMAL(9,6)
);

ALTER TABLE dispositivo AUTO_INCREMENT = 100;
```

Figura 29. Implementación de la tabla dispositivo.

```
CREATE TABLE medida (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tipo CHAR(1) NOT NULL,
    fecha TIMESTAMP NOT NULL,
    dispositivo INT NOT NULL,
    FOREIGN KEY (dispositivo) REFERENCES dispositivo(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,

    CONSTRAINT tipo_cnstr CHECK(tipo in('S', 'U', 'D'))
);
```

Figura 30. Implementación de la tabla medida.

ARDUINO

Hemos desempeñado toda la creación de código para el comportamiento de nuestro prototipo Arduino mediante la herramienta Arduino IDE.

Como sabemos , Arduino está basado en el lenguaje de programación C++, ya tenemos conocimientos de este lenguaje, y además nos ayudaremos de las siguientes bibliotecas y documentación ofrecida por Arduino para manejar el comportamiento de los componentes montados sobre la placa.

1. Lectura y escritura mediante la biblioteca SD [20]. Hay que tener en cuenta que la tarjeta microSD debe estar formateada previamente en FAT o FAT32. El módulo de tarjeta microSD nos ayudará mediante un archivo para guardar información de la configuración del dispositivo y además contendrá, en otro archivo diferente, todas medidas registradas.
2. Manejo del tiempo con la librería RTCLib para el reloj de tiempo real [21]. Este módulo nos servirá para que nos indique la hora en el momento que el dispositivo lo necesite (por ejemplo para almacenar una medida nueva).
3. El control de los mensajes Bluetooth para el módulo HC-05 se realizará mediante la biblioteca SofwareSerial [22], la misma que se usa para la depuración en el Arduino IDE.
4. El módulo TP4056 será útil para cargar la batería del dispositivo. Este módulo contiene una propiedad de carga con protección, para proteger a nuestra batería.
5. Pila de litio. La necesitaremos para, en los momentos que no haya energía eléctrica en la casa, que el dispositivo se mantenga encendido y pueda detectar correctamente las subidas y bajadas de tensión.
6. En cuanto al módulo step UP, será necesario para que la tensión que llegue al dispositivo arduino ronde siempre los 5V, de esta manera nos aseguraremos un correcto funcionamiento del dispositivo a nivel de potencia.

Comandos AT

Hay que destacar que, el módulo Bluetooth HC-05 es modificable mediante los llamados comandos AT. Surge la necesidad de, al generar un prototipo que se va a usar en un proyecto futuro, poder editar los parámetros de información de nuestro dispositivo. Por lo tanto, sería conveniente hacer una cambio en el PIN de acceso para la conexión bluetooth y el nombre del dispositivo para que no sea el genérico por defecto.

Entonces, debemos entrar en la configuración del dispositivo Bluetooth (presionando el botón del módulo mientras lo vamos a encender). Seguidamente, mediante el Serial de comunicación le mandaremos el siguiente comando para cambiar el nombre. También, con el segundo comando que se muestra, podemos cambiar el PIN de acceso.

- AT+NAME=NoMasCortes
- AT+PSWD="5555"

Un último apunte sobre el prototipo Arduino, es que estará monitoreando cambios de estado de corriente cada segundo. Pero, ¿Cómo se detecta un cambio de estado? Bien, para ello necesitaremos una variable que en todo momento tendrá almacenado un 0 o 1 (detecta corriente o no). Además, otra variable contendrá el estado (0/1) del instante anterior de la corriente. Por lo tanto, sabiendo el valor de estas dos variables, podemos deducir si ha habido una subida de corriente o una bajada de corriente.

Paso de mensajes

Vamos a definir una estructura para el intercambio de mensajes vía Bluetooth entre el prototipo Arduino y la aplicación móvil.

Estos mensajes, estarán divididos en 3 partes. La primera será donde se especifique el tipo de mensaje que estamos pasando, mediante un carácter, el tipo de mensaje puede ser uno de los siguientes.

- S => (Send): se utilizará para mensajes que contengan medidas detectadas por Arduino.

- E => (Error): una vez se hayan enviado las medidas, la aplicación devolverá este mensaje por si ha habido algún error en su procesamiento.
- C => (Completed) Una vez se hayan enviado las medidas, la aplicación devolverá este mensaje confirmando que el proceso ha finalizado correctamente.
- T => (Time): se utilizará cuando la aplicación solicite actualización de la hora del Real Time Clock.
- I => (Initialize): será útil para inicializar el dispositivo Arduino.
- U => (Update):será útil para actualizar la información del dispositivo Arduino.

El formato de los mensajes será definido por esta estructura:

Mensaje completo		
Tipo mensaje (char)	Información mensaje	Fin de mensaje (#)

Un primer campo para la definición del mensaje, un segundo campo con la información a transferir y por último, un campo que indica el final del mensaje (representado por una almohadilla).

En la siguiente imagen se puede ver el primer prototipo montado del dispositivo monitor basado en Arduino.

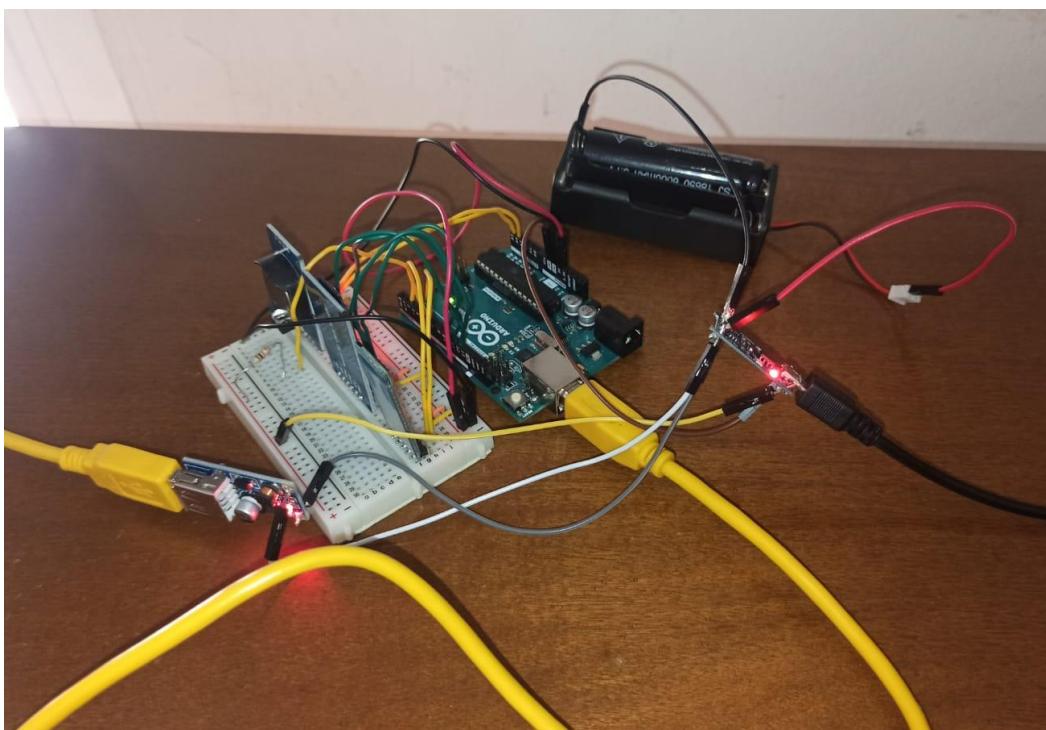


Figura 31. Primer prototipo del dispositivo monitor (Arduino).

APP MÓVIL

Una vez entramos en el desarrollo de la aplicación móvil, se explicará rápidamente el punto más importante y para el que se ha construido la aplicación, el intercambio de datos de medidas entre el prototipo monitor y la base de datos.

En primer lugar, el usuario solicita las medidas mediante un mensaje SEND, declarado en el punto anterior. En ese momento, la hebra que se encarga de escuchar los mensajes que provienen de Bluetooth registra el mensaje al completo con toda la información de medidas. Es en este punto, cuando el mensaje se manda a un Handler definido en la clase ConexiónBluetooth. Este Handler recibe el mensaje, lo identifica y hace una llamada a la función correspondiente de la API para que se haga un POST de las medidas en la API REST.

La API REST se encargará de devolver un mensaje de confirmación o de error dependiendo de cómo haya finalizado el proceso. Finalmente, una vez recibido el mensaje desde la API, la aplicación Bluetooth envía un mensaje (de tipo E, error o C, completed) mediante Bluetooth al prototipo Arduino.

Recordemos que es un prototipo y que la versión final, en caso de que se hiciera un proyecto real, estaría más cuidada estéticamente.

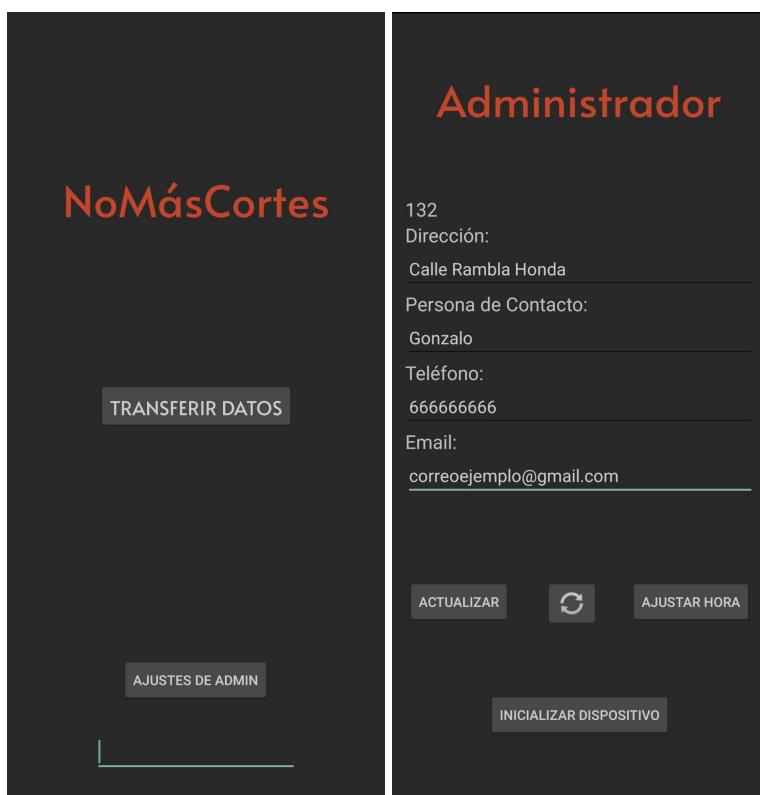


Figura 32. Versión final de la APP Android.

API REST

Llegamos a la fase implementación de la API REST, en este punto debemos elegir qué herramienta vamos a utilizar para desarrollarla. Tras barajar la opción de hacerlo en el lenguaje PHP (como la página web) sin ningún framework que nos ayude, finalmente se ha decidido implementar la API mediante Node.js, con su paquete *Express*.

Express se encargará de ayudarnos a dividir nuestro código en dos partes, en primer lugar la definición y configuración de la aplicación que contiene la API REST y por otro lado, la configuración y definición de las rutas accesibles por los clientes.

Prevención de inyección SQL

Para que nuestro sistema goce de la seguridad necesaria tendremos que implementar las consultas a la base de datos teniendo en cuenta usuarios malintencionados que pueden hacer técnicas de inyección SQL.

La inyección SQL es una técnica que intenta vulnerar la estructura de una base de datos insertando sentencias SQL dentro de la una URL. Un ejemplo sería, añadiendo dentro de una variable de la URL lo siguiente:

```
http://<dominio>/tiempo?ciudad=Madrid; DROP TABLE ciudades;
```

En el caso de una sentencia SQL tratada como una cadena de caracteres a la hora de hacer la consulta en el servidor, esta quedaría de la siguiente manera

```
SELECT * FROM ciudades WHERE nombre='Madrid'; DROP TABLE ciudades;
```

Por lo tanto, se ejecutará la primera sentencia y después la segunda, causando una pérdida de datos considerable.

Para evitar esto, en el sistema No Más Cortes, hemos implementado las consultas a la base de datos mediante sentencias preparadas. Es muy sencillo, dentro de la sentencia SQL se coloca el carácter ‘?’ donde se espera un valor variable, y en un array de datos se especificará la variable que dará el valor a la posición ‘?’.

```
conexion.query("SELECT * FROM medida WHERE id=?", [req.params.id], (error, resultado) =>{
    if(error) return res.send(error);
    res.json(resultado);
});
```

Figura 33. Ejemplo de query ante inyección SQL.

PÁGINA WEB

Finalmente, queda por implementar la página web, para ello, como ya se ha explicado, utilizaremos el patrón Modelo, Vista, Controlador. Pero lo que no se ha explicado aún es, la forma en la que vamos a aplicar este patrón.

Twig es un motor de plantillas desarrollado sobre el lenguaje de programación PHP. Su objetivo principal es el de facilitar a los desarrolladores de aplicaciones web que utilizan la arquitectura MVC el trabajo con la parte de las vistas. Es parte del framework de desarrollo web Symfony.

La clave del éxito de Twig es su facilidad de aprendizaje y de uso. Nos provee herramientas de desarrollo de vistas que nos permiten escribir variables provenientes de php de forma muy sencilla, herencia de plantillas para reducir y segmentar el código de manera muy eficiente. También ofrece estructuras propias de lenguajes de programación como son bloques condicionales o bucles for.

Una parte a destacar de la página web, es que es la encargada de generar los pares de medidas que conforman un corte de luz. Un corte de luz se define con una fecha de inicio y una fecha de fin. Entonces, cuando hacemos una consulta de medidas de un dispositivo, se nos devuelven, por orden, las medidas captadas por este dispositivo. Por lo tanto, y según lo pactado con el tutor, Sergio, haremos una búsqueda de pares de medias tal que la primera sea de tipo ‘D’ (bajada de tensión) y la segunda sea de tipo ‘U’ (subida de tensión) y estén consecutivas. No vamos a tener en cuenta cortes de luz que no tengan definido el inicio y/o fin del corte.

Nos queda pendiente nada más la explicación del método de inicio de sesión. Bien, lo implementaremos de la siguiente manera. El administrador deberá introducir el usuario *admin* además de la contraseña. Al validar el formulario, dentro de nuestro servidor, mediante una encriptación con el algoritmo md5, se procederá a verificar que los campos se hayan introducido correctamente, si es así generamos una nueva sesión de administrador.

Diseño final

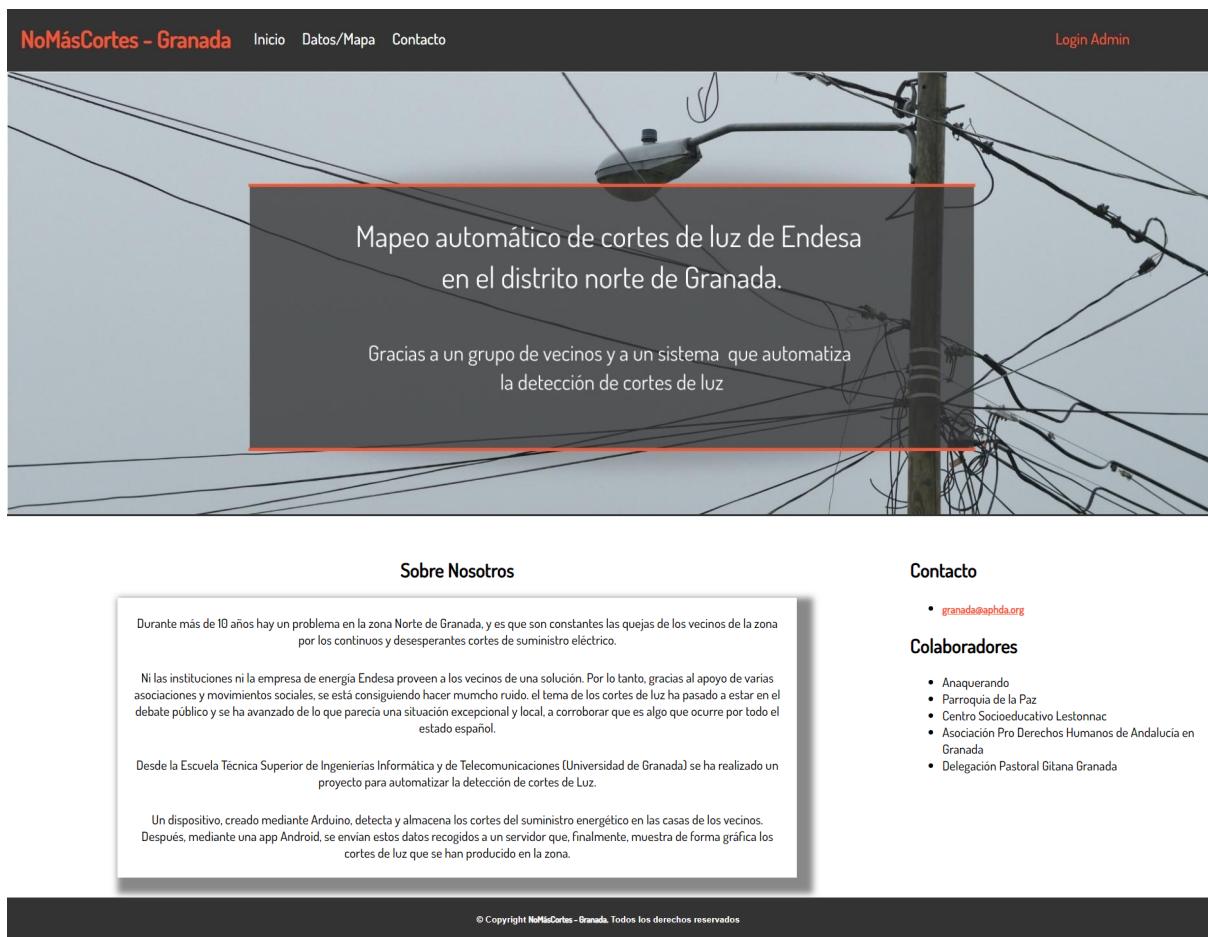


Figura 34. Landing page de la página web.

Inicio de sesión solo disponible para administradores.

The login form is a simple interface with a light gray background. It contains two input fields: one for "Usuario" (User) and one for "Contraseña" (Password), both outlined in red. Below these fields is a large orange button labeled "Login".

Figura 35. Login de administradores de la página web.

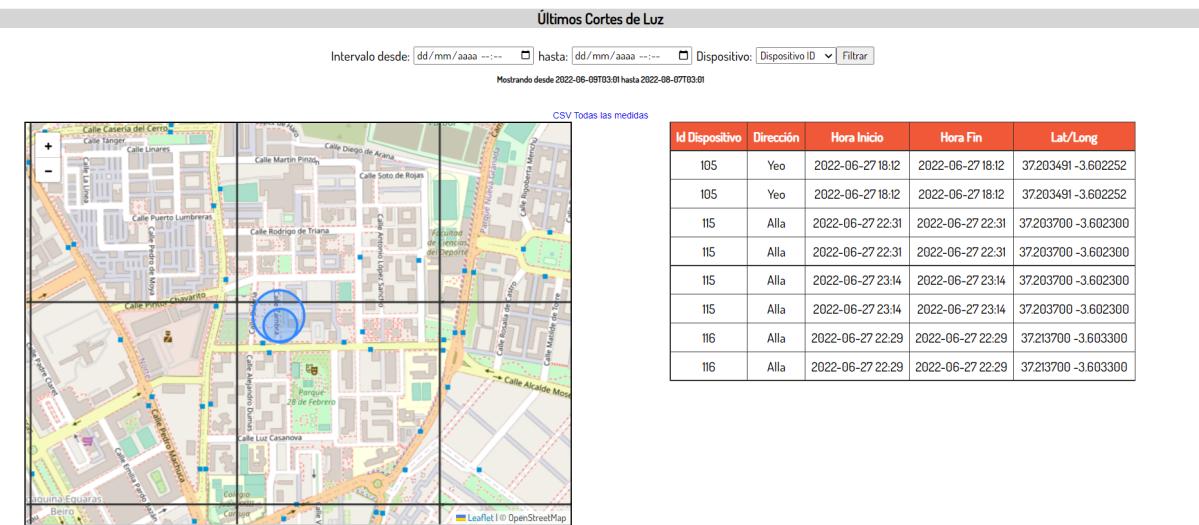


Figura 36. Página que muestra cortes de luz de la página web.

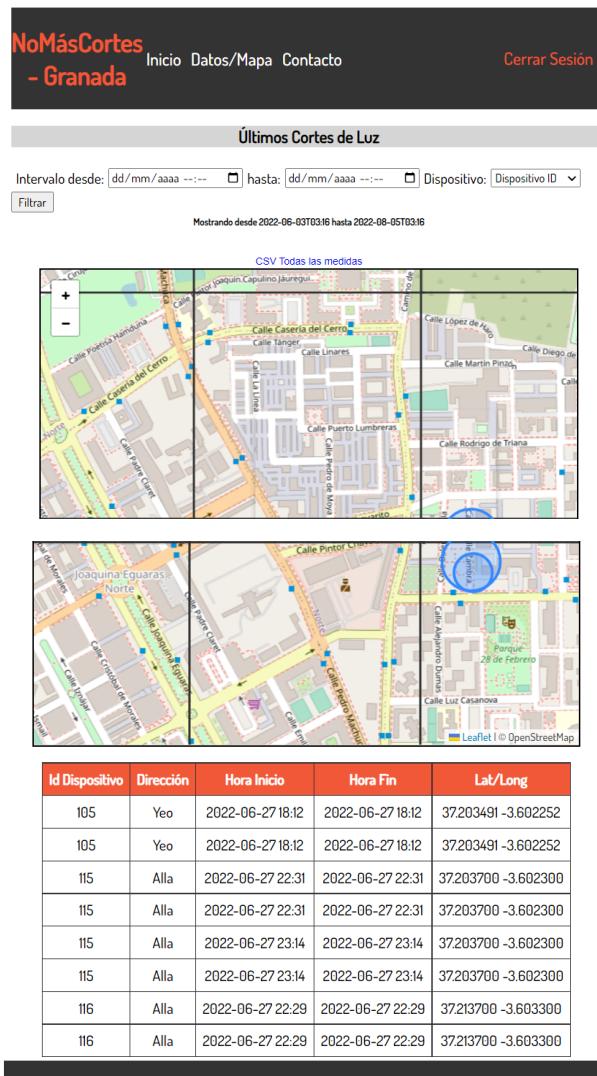


Figura 37. Diseño de página de datos de la página web responsive.

PRUEBAS

Vamos a entrar en la última fase de nuestro proyecto, que es la realización de pruebas de nuestro sistema para verificar su correcto funcionamiento. Explicaremos qué herramientas o técnicas hemos utilizado para ir haciendo pruebas durante el desarrollo del sistema. Para la esta fase se necesita generar, al menos, una prueba para cada uno de los casos de uso de nuestro sistema, pero en el apartado pondremos algunas pruebas de ejemplos que hemos realizado.

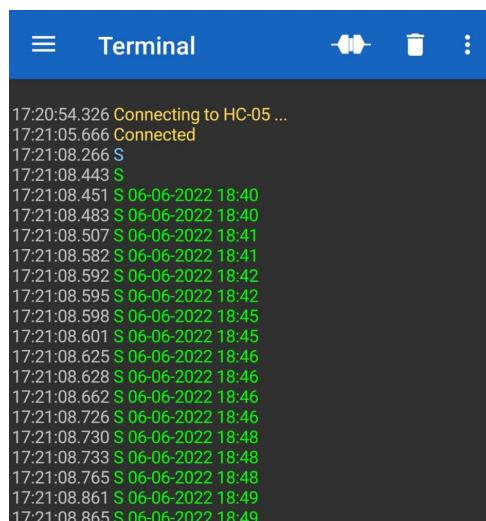
Además, durante el desarrollo del proyecto se han ido realizando pruebas de aceptación, es decir, se le ha mostrado al cliente, en este caso el tutor del trabajo, las funcionalidades desarrolladas para su visto bueno.

Fase 1: Arduino

En este caso, la primera y más básica herramienta de pruebas de Arduino es el puerto USB que muestra en la pantalla del PC donde se está escribiendo el código las líneas utilizadas para debuggear el código. En una primera instancia, nos servirá para asegurarnos de que funcionan bien las detecciones de cambio de estado y las escrituras/lecturas de los archivos de la memoria microSD.

Para comprobar la funcionalidad general del prototipo y de los mensajes bluetooth vamos a hacer uso de una aplicación móvil externa llamada Serial Bluetooth Terminal. Podemos encontrar la aplicación en Google Store a través de este [enlace](#).

Gracias a esta aplicación somos capaces de mandar mensajes al dispositivo que esté conectado (módulo bluetooth en este caso) mediante cadenas de caracteres. De esta manera es posible enviar un mensaje de los definidos en nuestra implementación y que el prototipo nos responda. Un ejemplo es el siguiente para la transmisión de medidas.



```
17:20:54.326 Connecting to HC-05 ...
17:21:05.666 Connected
17:21:08.266 S
17:21:08.443 S
17:21:08.451 S 06-06-2022 18:40
17:21:08.483 S 06-06-2022 18:40
17:21:08.507 S 06-06-2022 18:41
17:21:08.582 S 06-06-2022 18:41
17:21:08.592 S 06-06-2022 18:42
17:21:08.595 S 06-06-2022 18:42
17:21:08.598 S 06-06-2022 18:45
17:21:08.601 S 06-06-2022 18:45
17:21:08.625 S 06-06-2022 18:46
17:21:08.628 S 06-06-2022 18:46
17:21:08.662 S 06-06-2022 18:46
17:21:08.726 S 06-06-2022 18:46
17:21:08.730 S 06-06-2022 18:48
17:21:08.733 S 06-06-2022 18:48
17:21:08.765 S 06-06-2022 18:48
17:21:08.861 S 06-06-2022 18:49
17:21:08.865 S 06-06-2022 18:49
```

Figura 38. Prueba de funcionalidad de Arduino con Serial Bluetooth Terminal.

Fase 2: aplicación móvil

Para realizar pruebas sobre la aplicación Android, utilizaremos el Logcat de Android Studio. Gracias a esta herramienta podemos imprimir los valores de la variable que queramos en un momento del programa determinado. De esta manera seremos capaces de ver cómo está tratando el flujo de datos nuestra aplicación.

En este ejemplo se muestra como la aplicación trata el caso de uso de transmitir medidas, recordemos: petición de medidas a Arduino y inserción en la base de datos por medio de la API.

Utilizaremos un dispositivo móvil físico para realizar las pruebas de la aplicación, por lo tanto en ese dispositivo hay que activar la opción de desarrollador, esto nos permite instalar y manipular la aplicación desde el dispositivo físico y que las banderas de debug se muestren en el Logcat de Android Studio.

Para comprobar que el dispositivo móvil puede acceder a la API, como esta está desplegada en local para el desarrollo, tenemos que desactivar el firewall del PC (servidor) para que las peticiones del dispositivo puedan recibirse.

```
2022-07-08 20:59:01.274 10912-11286/com.example.nomascortes D/TAG_DEBUG_: S
132
S 08-07-2022 19:45
#
2022-07-08 20:59:01.275 10912-10912/com.example.nomascortes D/TAG_DEBUG_H: 132
S 08-07-2022 19:45
2022-07-08 20:59:01.315 10912-10912/com.example.nomascortes D/TAG_DEBUG: Generando petición API
2022-07-08 20:59:01.431 10912-10912/com.example.nomascortes D/TAG_DEBUG_Response: [{"respuesta":"Filas insertadas correctamente!"}]
2022-07-08 20:59:08.552 10912-11286/com.example.nomascortes D/TAG_DEBUG_: S
132
#
2022-07-08 20:59:08.552 10912-10912/com.example.nomascortes D/TAG_DEBUG_H: 132
2022-07-08 20:59:08.552 10912-10912/com.example.nomascortes D/TAG_DEBUG: Generando petición API
2022-07-08 20:59:08.641 10912-10912/com.example.nomascortes D/TAG_DEBUG_Response: [{"respuesta":"Filas insertadas correctamente!"}]
```

Figura 39. Pruebas con la APP Android..

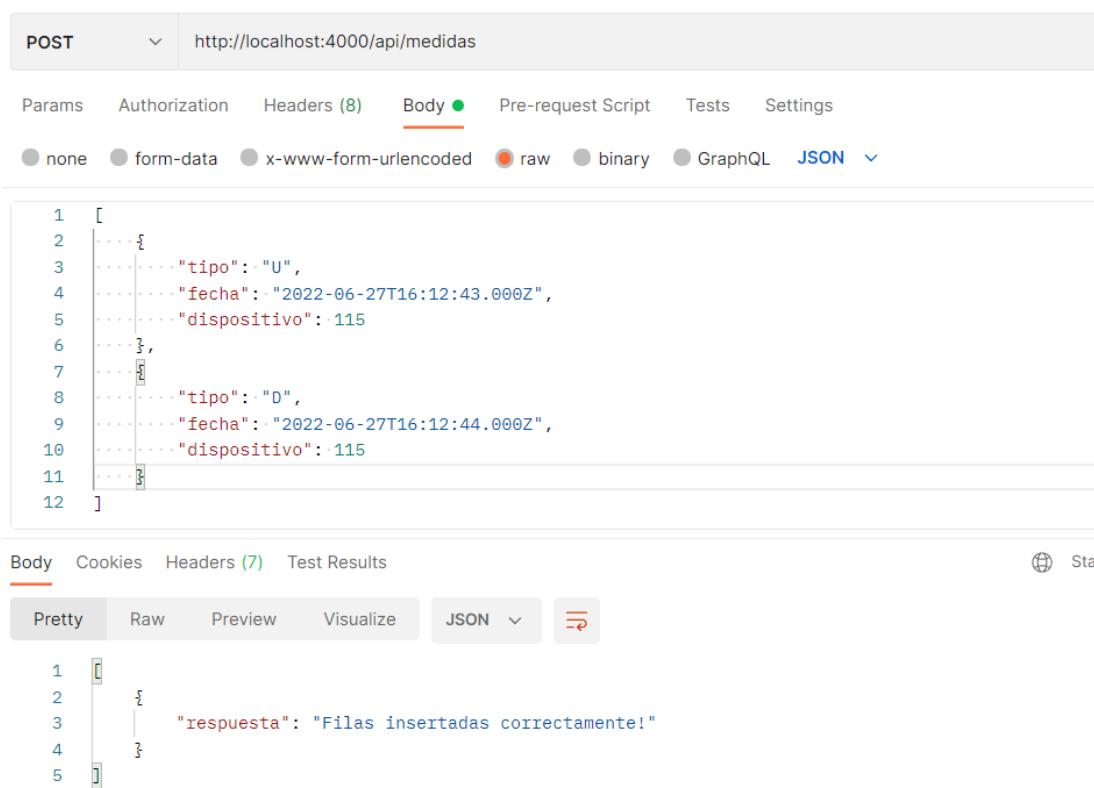
Vemos que las medidas llegan correctamente, también llega el ID del dispositivo y con esos datos podemos realizar la petición a la API que inserta correctamente las nuevas medidas

Fase 3: API REST

Existe una aplicación de escritorio, que ya conocía de otros proyectos, llamada Postman. Postman nació como una extensión para navegadores que nos permite realizar peticiones HTTP y testear APIs. Postman ha evolucionado hasta convertirse en una aplicación de escritorio disponible para la mayoría de sistemas operativos. Postman ofrece más servicios a parte del mencionado para testear APIs, pero nos centraremos en ese ya que es donde el que nos sirve de ayuda.

Lanzaremos el servidor de forma local en nuestro PC para comprobar la api en nuestro dominio localhost.

Aquí podemos ver un ejemplo de POST de una medida realizada sobre nuestra API.



The screenshot shows the Postman interface with a POST request to `http://localhost:4000/api/medidas`. The `Body` tab is selected, displaying the following JSON payload:

```
1 [  
2   {  
3     "tipo": "U",  
4     "fecha": "2022-06-27T16:12:43.000Z",  
5     "dispositivo": 115  
6   },  
7   {  
8     "tipo": "D",  
9     "fecha": "2022-06-27T16:12:44.000Z",  
10    "dispositivo": 115  
11  }  
12 ]
```

The `Response` tab shows the following JSON response:

```
1 {  
2   "respuesta": "Filas insertadas correctamente!"  
3 }
```

Figura 40. Pruebas mediante Postman.

Fase 4: página web

Finalmente, solo nos queda la validación de la página web. Para ello, vamos a lanzar el servidor en local en nuestro PC. Dado que nuestra aplicación está construida sobre PHP, nos dirigiremos a la localización donde está instalado y en el archivo php.ini debemos activar la impresión de warnings y errores. De esta forma, todos los errores y los warnings nos saldrán impresos en el navegador al hacer peticiones al servidor.

Por otra parte, durante el desarrollo de la aplicación se han utilizado herramientas de php como la función *var_dump()* que especifica el tipo, longitud y contenido de la variable.

En primer lugar, hay que valorar que los datos llegados desde las peticiones a la API estén correctos y bien estructurados, después, hay que verificar que, al procesar esos datos mantengan la consistencia y finalmente, mediante las vistas, debemos comprobar los mismos correctamente formateados en la tabla y mapa de la página web.

CONCLUSIONES

Una vez entrados en este apartado, debemos hacer una valoración objetiva del trabajo realizado y los objetivos alcanzados. En relación a los objetivos, podemos decir que se han superado con éxito los objetivos que teníamos previstos. Hemos sido capaces de diseñar e implementar un prototipo capaz de monitorear y mostrar cortes de luz de un sistema eléctrico, junto a sus objetivos específicos. Además se ha cumplido 1 de los 2 objetivos opcionales, uno era el objetivo de mostrar gráficamente los cortes de luz en un mapa. En contraposición, no hemos sido capaces de desarrollar, por razones de tiempo, el objetivo opcional de ordenar, en la página web, los cortes de luz por criterios de duración, dirección, etc.

En líneas generales, considero que se ha realizado un buen trabajo, y dado que, el producto que se entrega, es un prototipo, hay muchas partes que están preparadas para una extensión de funcionalidades.

Por otra parte, me he dado cuenta que, al realizar un producto software con tantas implicaciones a nivel de tecnologías y conocimientos, es necesario tener una muy buena base a la hora de realizar las fases de análisis y diseño, para no tener que dar un giro de 180º en medio de la implementación y que los plazos de tiempo no se vean muy incrementados forzosamente. También me he dado cuenta de la importancia de los equipos de trabajo, para poder repartir las tareas, ya que en este proyecto, probablemente hubiera sido necesario un pequeño equipo de 3 o 4 personas para que su desarrollo acabe con un punto extra de calidad.

En el aspecto de planificación, he visto como ha tenido que ser modificada levemente, sin consecuencias graves. Se han tenido que compensar algunas horas de realización de la página web, en favor del desarrollo de la aplicación móvil, ya que el desarrollo móvil a priori, no parecía muy complicado, pero a la hora de la verdad, se necesitaban usar muchas tecnologías diferentes como Bluetooth, conexiones con la API, etc. y el proceso de coordinación de las mismas no es trivial.

Como se ha visto y repetido continuamente, este proyecto toca diferentes tecnologías que por suerte, he tenido la oportunidad de haber visto previamente durante gran parte del grado. Por lo tanto gracias a las diferentes asignaturas he podido adquirir conocimientos suficientes para desarrollar el proyecto.

Por ejemplo, para la parte de análisis, diseño y desarrollo de patrones de diseño, me han sido de ayuda asignaturas como Ingeniería del Software o Desarrollo de Software. Para el desarrollo de la base de datos ha sido muy útil las asignaturas relacionadas como Fundamentos de Bases de Datos o Diseño y Desarrollo de Sistemas de Información. La asignatura Diseño de Interfaces de Usuario es muy útil para hacer interfaces de usuario fáciles e intuitivas como las de este prototipo y por supuesto asignaturas básicas de programación para hacer un código limpio y estructurado.

Por lo tanto, es un hecho que el grado nos prepara para desarrollar individualmente ese pensamiento abstracto necesario para hacer productos de calidad.

BIBLIOGRAFÍA

1. G., R. (2022, 24 marzo). El Ayuntamiento de Granada pedirá a Gobierno, Junta y Endesa «acabar con la lacra de los cortes de luz en el distrito Norte». Granada Hoy.
https://www.granadahoy.com/granada/Ayuntamiento-Granada-pedira-acabar-lacra-cortes-luz-distrito-norte_0_1668134069.html
2. Troyano, R., & Árbol, E. (2016, 26 enero). «Almanjáyar, el polígono industrial de la droga en Andalucía». Cadena SER.
https://cadenaser.com/emisora/2016/01/26/radio_granada/1453815559_569792.html
3. Hueso, S. G. (2020, febrero 3). Solo el 6% de los enganches detectados en Norte en 2019 tenían relación con la maría. Ideal Granada.
<https://www.ideal.es/granada/solo-enganches-detectados-granada-20200203184919-nt.html>
4. Pascual, M. (2014, 17 diciembre). La Asociación Pro Derechos Humanos de Andalucía contra los cortes de luz en la zona norte de la capital de Granada. GranadaDigital.
<https://www.granadadigital.es/la-asociacion-pro-derechos-humanos-de-andalucia-contra-los-cortes-de-luz-en-la-zona-norte-de-la-capital-de-granada/>
5. González González, F., Calero Castañeda, S. L., Loaiza Buitrago, D. F. (s. f.). Comparación de las metodologías cascada y ágil para el aumento de la productividad en el desarrollo de software. Universidad Santiago de Cali.
6. Velázquez Camacho, J. (2012, 29 noviembre). Modelo Cascada [Ilustración].
<https://www.northware.mx/blog/desarrollo-en-cascada-waterfall-vs-desarrollo-agile-scrum/>
7. Jerez, N. (2019, 3 junio). Metodología Iterativa [Ilustración].
<https://medium.com/sue%C3%B1os-graficos/dise%C3%B1o-iterativo-la-metodolog%C3%A1-Da-que-perfeccionar%C3%A1-tus-proyectos-21034b0d277e>
8. Hostinger. (2022). Una Plataforma De Alojamiento Web Intuitiva.
https://www.hostinger.es/?utm_medium=affiliate&utm_source=aff27522&utm_campaign=211&session=1022def4e2f0992bcd5ff51ea0ace
9. webempresa. (2022, 4 junio). Lanzamiento Hosting 2022.
https://www.webempresa.com/promo/lanzamiento-hosting-wordpress-2022.html?utm_medium=affiliate&utm_source=1295&utm_campaign=Afiliados
10. Monferrer Agut, R. (2000). Especificación de Requisitos Software según el estándar de IEEE 830. Universitat Jaume I.

11. Moreira Resabala, W. M. (2016). Estudio comparativo entre SQL y NoSQL basado en el rendimiento, seguridad, e integridad de los datos. Universidad Técnica Estatal de Quevedo.
<https://repositorio.uteq.edu.ec/handle/43000/4036>
12. Shum, Y. M. (2021, 14 marzo). Situación Global Mobile o Móvil 2021. Yi Min Shum Xie.
<https://yiminshum.com/mobile-movil-mundo-2021/>
13. Chowdhry, A. (2015, 3 febrero). Average iPhone Price Increases To \$687 And Android Decreases To \$254, Says Report. Forbes.
<https://www.forbes.com/sites/amitchowdhry/2015/02/03/average-iphone-price-increases-to-687-and-android-decreases-to-254-says-report/?sh=543c8b0c539e>
14. Chung, C. (2011). Singleton. In: Pro Objective-C Design Patterns for iOS. Apress.
https://doi.org/10.1007/978-1-4302-3331-2_7
15. Zhart, D. (s.f). Patrón Singletón [Ilustración]. Refactoring Guru.
<https://refactoring.guru/es/design-patterns/singleton>
16. RedHat. (2020, 8 mayo). ¿Qué es una API de REST?
<https://www.redhat.com/es/topics/api/what-is-a-rest-api>
17. Mozilla Developers. (2021, 22 junio). Métodos de petición HTTP - HTTP | MDN.
<https://developer.mozilla.org/es/docs/Web/HTTP/Methods>
18. García, I. (2021, 23 diciembre). Qué es Modelo-Vista-Controlador. Blog SEO, Diseño Web & Gráfico | Caronte Web Studio Vitoria-Gasteiz.
<https://carontestudio.com/blog/que-es-modelo-vista-controlador/>
19. Tan, K. (2021, 8 junio). Model View Controller [Ilustración]. Medium.com.
<https://medium.com/geekculture/model-view-controller-70cf248455d>
20. Arduino. (2021, 18 noviembre). Guide to Arduino & Secure Digital (SD) Storage. Arduino Documentation. <https://docs.arduino.cc/learn/programming/sd-guide>
21. Adafruit. (2022). RTCLib: RTC_DS3231 Class Reference. RTC_DS3231 Class Reference.
https://adafruit.github.io/RTCLib/html/class_r_t_c _d_s3231.html
22. Arduino. (s.f). SoftwareSerial Library. Arduino Documentation.
<https://docs.arduino.cc/learn/built-in-libraries/software-serial>
23. TWIG. (2022). Home - Twig - The flexible, fast, and secure PHP template engine. The Flexible, Fast, and Secure Template Engine for PHP. <https://twig.symfony.com/>