SISTEMA DE MONITOREO Y SUPERVISIÓN FISIOLÓGICA Y FÍSICA

Trabajo de Fin de Grado

Grado en Ingeniería Informática

VNiVERSiDAD D SALAMANCA

Julio de 2021

Autor

José Pizzano Martín

Tutores

Sara Rodríguez González

Jorge Herrera Santos

Sergio Márquez Sánchez

Dra. Sara Rodríguez González, profesora titular de la Universidad de Salamanca en el área de ciencias de la computación e inteligencia artificial junto con D. Sergio Márquez Sánchez y D. Jorge Herrera Santos del grupo de investigación BISITE de la Universidad de Salamanca en calidad de co-tutores.

Hacen constar que:

El trabajo el trabajo titulado: “Sistema de monitoreo y supervisión fisiológica y física” ha sido realizado por D. José Pizzano Martín, con DNI 70963493Y garantizando la originalidad de la memoria de trabajo para la superación de la asignatura Trabajo de Fin de Grado de la titulación del grado de ingeniería informática de la Universidad de Salamanca.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a martes, 6 de julio de 2021

Firmado por MARQUEZ SANCHEZ SERGIO - 70891085W el día 06/07/2021 con un certificado emitido por AC FNMT Usuarios

RODRIGUEZ GONZALEZ SARA - 70864126E

Firmado digitalmente por RODRIGUEZ GONZALEZ SARA - 70864126E
Fecha: 2021.07.06 21:03:27 +02'00'

HERRERA SANTOS JORGE - 70938872H

Firmado digitalmente por HERRERA SANTOS JORGE - 70938872H
Fecha: 2021.07.07 00:15:33 +02'00'

Resumen

En la sociedad actual se está estableciendo cada vez más un uso continuado de los dispositivos electrónicos gracias al desarrollo de invenciones como los teléfonos inteligentes. A lo largo de nuestra historia, esta conducta de dependencia tecnológica ha ido adaptándose al día a día de las personas donde encontramos desde el paso de uso de impresos al uso de bases de datos para recolectar y guardar datos como el uso de la comba o el balón a los videojuegos y redes sociales para el entretenimiento.

Una de las tecnologías emergentes del momento es la tecnología vestible o comúnmente conocida como "wearables" que permiten introducir dispositivos electrónicos inteligentes de recolección de datos mediante sensores con el objetivo de monitorizarlos y mostrárselos al usuario como es el caso de los relojes inteligentes.

Esto ha llevado a idear diferentes conceptos a los que aplicar la idea de los "wearables" en los que encontramos el uso de estos para la prevención de riesgos laborales cuyo objetivo principal siempre será siempre garantizar la seguridad del usuario ya sea evitando los riesgos o pudiendo acudir los más rápido posible en caso de que hubiese un accidente. Por ello, se propone un sistema de monitorización de datos capaz de avisar al usuario y a terceras personas en los casos que fuese preciso.

Para ello, se realiza un proyecto fundamentado en tres partes principales. La primera consta de los sensores capaces de recoger datos del entorno de forma cíclica y los dispositivos de comunicación para transmitir dichos datos.

La segunda parte será el receptáculo de esos datos que utilizará el mismo usuario para la visualización de los mismos. Esto se gestiona mediante una aplicación móvil para dispositivos Android capaz de recibir los datos vía Bluetooth y mostrarlos por pantalla para uso del propio usuario.

Por último, la tercera de las partes será un servidor web basado en JavaScript y Node.js donde los distintos dispositivos móviles enviarán los datos que reciban de forma periódica para poder realizar un estudio en caso de que fuese necesario, así como avisos de forma instantánea si alguno de los sensores determinase que ha aparecido algún problema pudiendo actuar instantáneamente.

Todo el proceso que gestiona el sistema no precisa en gran medida la interacción directa con el usuario garantizando así una respuesta rápida otorgándole de observar todos los datos recogidos por la aplicación, así como al servidor web mostrar los cambios en los dispositivos de forma dinámica e intuitiva.

Palabras clave: prevención de riesgos, monitorización de datos, dispositivos vestibles.

Abstract

On nowadays society is getting stablished a habitual usage of electronic devices thanks to the development of inventions such as smartphones. In quality of time, this behaviour has been adopted by humans in a matter of decades where we have gone from printed matter to using databases for data collection or the use of a ball and skipping rope to videogames and social media for entertainment.

One of the emerging technologies of the moment is the wearable technology also known as "wearables" which let us incorporate smart electronic devices using sensors with the objective of monitoring data and show it to the user as is the case of smart watches.

This idea has led to devise different concepts where we can apply the concept of "wearable". Between these ideas we can find those for prevention of occupational hazards which main objective will always be to ensure safety to the user either avoiding risks or being able to go as fast as possible in case of emergency. Therefore, it is proposed a monitoring system capable of alerting the user and third parties in case of need.

For this, a project based on three main parts is carried out. The first consists of the sensors capable of cyclically collecting data from the environment and communication devices to transmit said data.

The second part will be the receptable for that data which will be used by the same user to view them. This is managed through a mobile application for Android devices capable of receiving data via Bluetooth and displaying them on the screen for the user´s own use.

Finally, the third of the parts will be a web server based on JavaScript and Node.js where the different mobile devices will send their data periodically to carry out a study, if necessary, as well as notifications instantly if any of the sensors determine that a problem has appeared and can act instantly.

The entire process that the system manages does not require direct interaction with the user to a great extent, thus guaranteeing a quick response, allowing them to observe all the data collected by the application, as well as the web server showing the changes in the devices in a dynamic and intuitive way.

Key words: risk prevention, data monitoring, wearable devices.

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción y antecedentes

Tras casi dos décadas desde la aparición del "IBM Personal Communicator", considerado el primer smartphone de la historia, la evolución de los dispositivos electrónicos ha sido más que exponencial. En este primer modelo encontrábamos funcionalidades como un reloj, una agenda de eventos o el correo electrónico lo que consideramos funcionalidades básicas de nuestros teléfonos móviles en la actualidad.

Esta corriente de teléfonos inteligentes es el pilar fundamental que hizo cambiar la visión de que los ordenadores solo se podían utilizar frente un escritorio a un mundo más independiente pasando de la idea de teclado mecánico y pequeña pantalla a algo completamente táctil capaz del reconocimiento facial lo que ha otorgado un gran abanico de ideas donde el usuario podría gestionar y visualizar diferentes tipos de datos con la comodidad de transportarlo en su bolsillo sin ocupar demasiado espacio.

Una de estas ideas se ha desarrollado y adaptado perfectamente a la sociedad actual como ha sido el caso de los dispositivos "wearables" que han imitado a la perfección ese concepto de ocupar poco espacio y otorgar facilidades al usuario imitando a dispositivos de uso diario de las personas. El caso más conocido es el de los relojes inteligentes los cuales sustituyen a los relojes convencionales ya que ofrecen al usuario la oportunidad de conocer algo más que la hora como el ritmo cardíaco y las horas de sueño ya que muchos poseen sensores para dichas tareas junto con características como la posibilidad de ver fotografías, asistentes virtuales e incluso recibir mensajes y llamadas.

Dado el auge de dichos dispositivos, cada vez son encontramos más ámbitos de aplicación de los "wearables" como es el caso del campo de la medicina donde se han logrado desarrollar sensores capaces de medir de forma precisa parámetros como la presión arterial, los paso dados o la liberación de agentes bioquímicos con el simple hecho de llevarlos puestos de forma habitual.

No es de extrañar que uno de los campos de desarrollo de los "wearables" haya sido en ámbitos de salud ya que garantizar la seguridad de una persona sin la necesidad de realizar estudios continuos llevados a cabo por personas, facilita y mejora de manera única la integridad del individuo, así como de aislarlo sin exponer a un tercero a contagios en caso de que esto fuese necesario. Dichos estudios se llevan siempre a cabo sobre la persona y su condición, pero también se puede garantizar su seguridad midiendo valores externos al individuo que se pueden encontrar en su entorno los cuales, en caso de no concordar con las medias habituales, podrían ser un claro indicador de que la persona en cuestión esté en una emergencia y deba ser socorrida.

Este concepto de previsión ambiental mediante sensores se observa en un proyecto llevado a cabo por el *Centro de Investigación Aplicada "Tecnalia"* junto con la *Fundación Sueskola* que presentaban un traje de bomberos inteligente que, además de medir las constantes vitales del usuario, poseía sensores térmicos que preveían si el usuario podía sufrir una situación de estrés térmico o un golpe de calor sumándole detectores de gases tóxicos que eran notificados si aparecían.

Por esto, se plantea una idea con un objetivo similar al traje de bomberos donde encontramos una chaqueta que, mediante una disposición de diferentes sensores, es capaz de determinar si el usuario se encuentra en peligro y notificar en caso necesario.

Para gestionar este problema, se plantea un sistema de comunicación de datos que consta de tres partes fundamentales en las que se dividen las tareas de recopilación de datos ambientales que rodean el entorno del usuario, visualización de estos en cualquier tipo de dispositivo y finalmente su gestión y posible estudio junto con la actuación en casos de emergencia para conseguir lograr los objetivos de prevención e inmediatez.

A este sistema se la ha llamado "*Prevention Monitor System: sistema de monitoreo y supervisión fisiológica y física*" del cual se puede conocer su funcionamiento a lo largo de este documento.

La estructura del proyecto se basará en la estructuración estándar para un trabajo de fin de grado de la Universidad de Salamanca donde se recogen los objetivos que lleva a cabo este trabajo, las metodologías utilizadas junto con las herramientas para el desarrollo, los resultados obtenidos junto con todas las tablas, figuras y código necesarios para llevar a cabo la correcta comprensión del producto.

2. Objetivos del proyecto

El objetivo principal del proyecto es diseñar un sistema que direccione los datos para su correcta visualización mediante la obtención de estos mediante sensores adaptados a un chip controlador que los enviará mediante Bluetooth a una aplicación móvil para su visualización y que finalmente, en caso de emergencia, se mande a un servidor web para acciones emprendidas por terceros garantizando así la seguridad mediante la prevención de riesgos laborales. Este sistema debe ser ofrecer escalabilidad para poder agregar nuevos dispositivos, ya sea por la parte de los sensores, como conectividad de distintas chaquetas a la aplicación móvil, así como el registro de las mismas en el servidor web. Por ello, se establecen los siguientes objetivos:

- Obtención de datos: el sistema debe ser capaz de obtener los diferentes datos ofrecidos mediante los sensores conectados al mismo para poder enviarlos de forma periódica para su visualización. Este objetivo se dividirá en dos:
    - Análisis de la trama de datos: debido a que los sensores mandarán los datos de forma automática al chip controlador, se debe establecer un orden lógico para obtener aquello que se reciba por parte de los sensores.
    - Envío de los datos: una vez obtenidos los datos, se deben generar paquetes que se envíen mediante una conexión por Bluetooth a la aplicación móvil.
- Visualización de datos: se debe generar una aplicación móvil basada en este caso en sistemas Android capaz de recoger esos datos por Bluetooth y presentarlos por pantalla en caso de que el usuario precise verlos.
    - Envío de datos: se vuelve a sumar el subobjetivo de enviar datos que, en este caso, se mandan los avisos necesarios junto con los datos recogidos de forma periódica en formato JSON a un servidor web.
        - Avisos: se mandarán de forma automática en cuanto se detecten para garantizar la inmediatez de reacción ante posibles adversidades presentes en el medio.
        - Datos: se nadarán los datos recibidos en la aplicación de forma periódica creando así pequeñas bases de datos en caso de necesitar recogerlos con los fines para los que se precisen.
- Gestión de datos y avisos: por la parte del servidor web se habilitan dos métodos para la recepción de datos y visualización:
    - Solicitudes GET: utilizadas por el usuario para la visualización de las páginas web donde encontrará las chaquetas conectadas, así como tablas respectivas a los datos.
    - Solicitudes POST: utilizadas por la aplicación móvil para mandar tanto los avisos como los datos en formato JSON.

Se plantean, además, objetivos secundarios que no garantizan la funcionalidad del sistema, pero deben ser contemplados ante la posibilidad de que el producto finalice siendo un producto real disponible en el mercado. Estos objetivos son:

- Garantizar la seguridad: el sistema posee dos puntos de unión clave donde se gestionan los datos los cuales se deben verificar para asegurar que el origen de estos y a donde se envían.
- Escalabilidad: al tratarse de un proyecto aplicable a varios dispositivos, este sistema debe ser escalable por las tres partes que lo componen:
    - El chip ESP32 debe permitir la conexión de diferentes sensores en futuros proyectos.

- La aplicación móvil debe permitir conexiones a chaquetas distintas en caso de que la original se cambie o se tenga más de una.
- El servidor web debe permitir el acceso al mismo por parte de varias chaquetas que actualicen sus datos y sus avisos.

Un prototipo de arquitectura inicial que resume todos los puntos anteriores se presenta en la siguiente imagen en la que se puede observar como la chaqueta puede recoger los diferentes sensores que se comunicarían con el dispositivo móvil del usuario. Una vez recibidos, el teléfono móvil se encargaría de manera automática de enviar los datos recibidos al servidor web que, a su vez, los mostraría por pantalla en un dispositivo de monitorización abierto de tal forma que el usuario final determine donde quiere comprobar y estudiar esos datos.

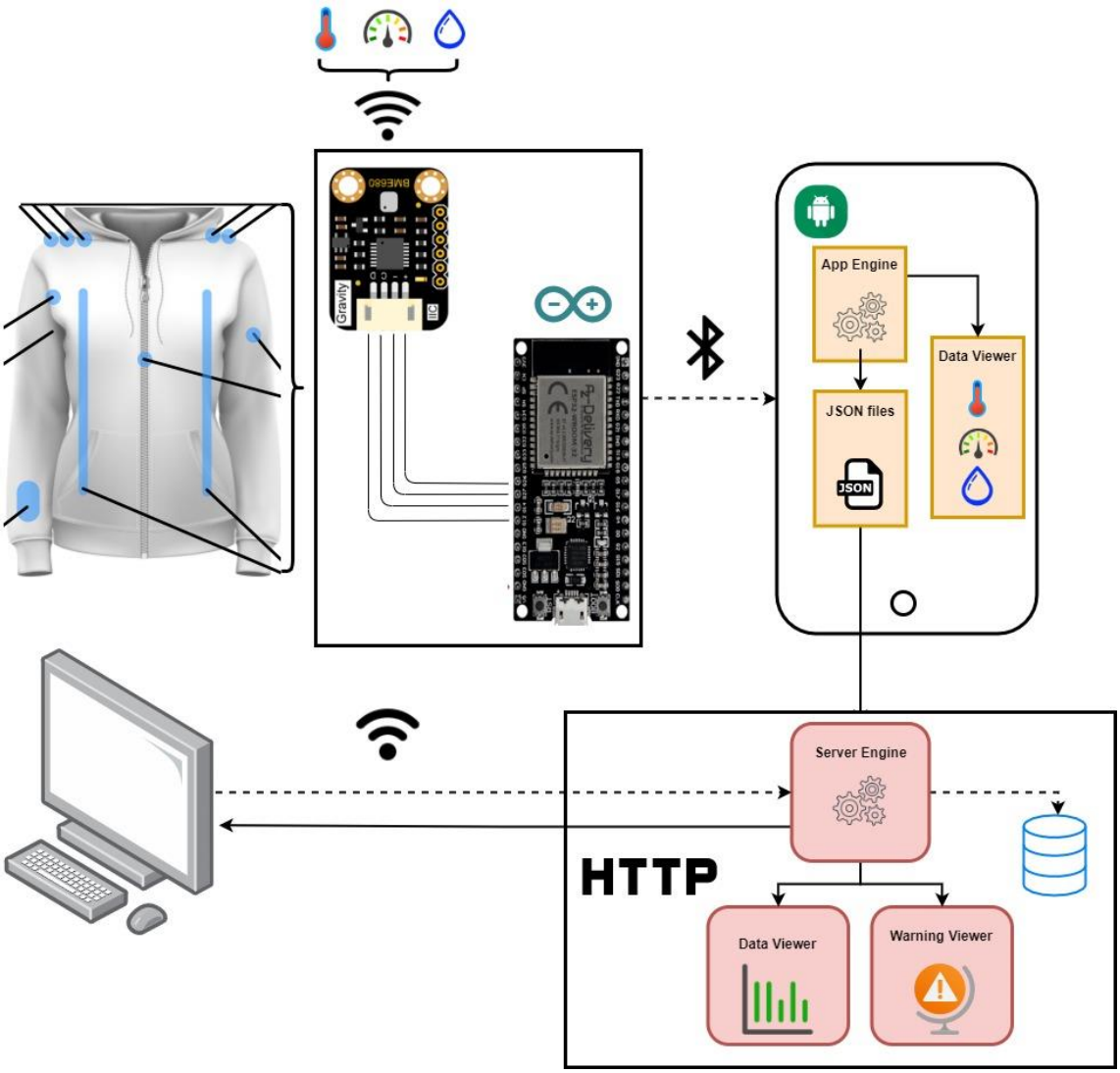


Ilustración 1: Diagrama de arquitectura general

3. Estado del arte

El estado del arte es la técnica mediante la que se puede determinar la situación de una determinada tecnología dentro de ámbitos de la investigación técnica, científica e industrial. Para realizar un correcto estado del arte se debe realizar un análisis exhaustivo de tipo documental que genere el contexto actual en el que se encuentra aquello que se desea estudiar.

Habiendo hecho un análisis de las partes de las que se fundamenta este proyecto es necesario contextualizar cuatro partes referentes a las cuatro iteraciones para conocer de una forma clara como gestionar cada sistema.

3.1. Wearables en ámbitos de prevención de riesgos laborales

Los dispositivos wearables, como se ha ido indicando en apartados anteriores, han ido ganando popularidad en los últimos años facilitando la toma de datos personales en usuarios que lo precisen y en esta idea es donde se fundamenta el proyecto, en tomar datos, en este caso datos ambientales del entorno del usuario, para garantizar su seguridad y generar una prevención en casos de necesidad de forma totalmente automática. Según el colectivo *HealthWorks*, este mercado estima un crecimiento de los 58.3 billones de dólares para el año 2025 considerando a esta tecnología una de las más relevantes e importantes del futuro próximo (Yin Chau, y otros, 2019).

La aplicación de estas herramientas incluye desde tecnologías que aplicadas en campos de la educación donde se aplicó el uso de las Google Glasses como herramienta de enseñanza con el objetivo de transmitir diferentes procedimiento directamente a un teléfono evitando así los costosos dispositivos de reproducción presentes en las clases actuales (Knight, Gajendragadkar, & Bokhari, 2015). Pero sin duda, donde más han conseguido impactar este tipo de tecnologías, es en el ámbito médico donde se ha logrado una reducción de los costes sanitarios, así como su eficiencia.

Dentro del ámbito médico, el uso y tratamiento de los datos es susceptible a que los usuarios desconfíen de su uso dentro de este ámbito. Esto se vuelca en un estudio realizado por el "*International Journal of Medical Informatics*" que muestra la aceptación de los dispositivos wearables con este objetivo mediante un estudio empírico de la privacidad. El modelo conceptual se fundamenta en datos recopilados mediante una encuesta presentada a un total de 333 usuarios que utilizan dispositivos portátiles para el cuidado de la salud como es la aplicación del procesamiento de señales avanzadas y dispositivos wearable para controlar el ritmo cardíaco de los fetos durante el embarazo garantizando seguridad durante el proceso (Li, Wu, Gao, & Shi, 2016).

Los resultados revelan tres hallazgos:

1. Las decisiones de las personas por utilizar wearables en ámbitos de la salud están determinadas por su análisis en riesgo-beneficio, es decir, si el riesgo en términos de privacidad recibido por el usuario es mayor que el beneficio a obtener, este no los empleará.
2. El riesgo que constituye la investigación se fundamenta en la sensibilidad de la información de salud, la innovación personal, la protección legislativa y el prestigio percibido.
3. El beneficio percibido por las personas está determinado en la información que reciben, sus propias limitaciones y la dirección que lleva la investigación.

Conociendo esto es necesario conocer a los usuarios a los que irá dirigido el proyecto y en que ámbitos laborales se desarrolla su trabajo.

Un ámbito de desarrollo de dispositivos wearables es la ropa inteligente donde, en forma de prendas combinadas con diferentes sensores, se logra combinar el principal objetivo de estos productos. Dentro del ámbito de este proyecto se busca la aplicación de estas prendas inteligentes para la prevención en ambientes laborales donde no se han desarrollado muchos prototipos.

3.1.1. Casos de aplicación de dispositivos wearable

Durante el año 2019, la empresa *TECNALIA* en colaboración con la *Plataforma Tecnológica Española de Seguridad Industrial* (PESI), organizaron una jornada extensiva de aplicación de los dispositivos wearables en ámbitos de la salud y prevención de riesgos laborales en el parque tecnológico de Gipuzkoa durante la cual se presentó un concepto de aplicación de exoesqueletos en el ámbito laboral.

Durante la ponencia de Juantxu Martín (Martín, 2019) sobre la aplicación de los exoesqueletos para la rehabilitación y prevención de trastornos se explica el uso de un dispositivo vestible externo con el objetivo de ayudar a sus usuarios en ámbitos médicos, como rehabilitaciones o actividades empresariales como la carga de objetos pesados. Uno de los dispositivos más conocidos mundialmente es el exoesqueleto "Hank" el cual se sitúa en el tren inferior del usuario para facilitar la rehabilitación de pacientes con heridas o enfermedades neurodegenerativas e incluso de accidentes que imposibiliten el anda.

Se puede deducir que en el caso anterior no existe una aplicación software de por medio que facilite la prevención. Por ello, Pablo Callejo de la empresa *STT Systems* (Callejo, 2019) exponía en esa misma conferencia como lograr una aplicación de los wearables utilizando los exoesqueletos para la obtención y estudio de los datos. En este caso, se buscaba mejor los dolores musculares de las actividades empresariales mediante el uso de diferentes sensores dispuestos estratégicamente capaces de generar un estudio que determine si el usuario posee alguna mala postura que pueda corregirse para evitar futuros dolores. Todo ello se logra mediante la transmisión mediante redes locales a las que se conecta el exoesqueleto para enviar sus datos los cuales son recogidos por un servidor que actúa como monitor calculando, según el software establecido, la actividad del usuario.

De estas ideas de monitorización surge a nivel nacional un prototipo planteado para un traje de bomberos por el *Centro de Investigación Aplicada "Tecnalia"* junto con la *Fundación Sueskola* que planteaban el control de constantes ambientales como la temperatura ambiente o un detector de gases para comprobar la reducción de oxígeno en sala junto con el control de constantes físicas como el ritmo cardíaco, la respiración y la temperatura corporal.

A pesar de que el cuerpo de bomberos está capacitado para gestionar situaciones de riesgo, otorgar medidas de seguridad siempre será una forma de garantizar la seguridad de los empleados ya que a nivel nacional se realizan alrededor de 350.000 intervenciones solo del equipo de bomberos (Basque Research, 2013).

Esta invención no solo se plantea para contrarrestar accidentes e incendios, sino que también es una herramienta que podría facilitar el trabajo en ámbitos como la minería o de la industria química donde se exponen a altos niveles de toxicidad.

En este caso de ejemplo no se presenta ninguna interfaz con el usuario que emplea el traje, sino que se envían directamente a una centralita por lo que no es un trabajo centrado en la

interfaz sino en la inmediatez de la respuesta y correcta actuación del sistema. Por ello, a la hora de aplicarlo a este proyecto se seguirá este concepto otorgando siempre un concepto modular e iterativo para gestionar posibles mejoras.

3.2. Desarrollo en Arduino

La comunidad de Arduino, basada en el desarrollo de software libre ha ido ganando popularidad a lo largo de los años gracias a la unificación del método de desarrollo de software para módulos controladores. Esto ha facilitado la gestión de grandes proyectos en ámbitos de chips evitando el uso de diversas herramientas según el producto que se estuviera utilizando.

Basando todos sus proyectos en lenguaje C++ se logra que un programador experimentado logre desarrollar un software sin gran esfuerzo debido al uso lógico que se aplica junto con el gran abanico de opciones y librerías que se ofrecen para facilitar las tareas.

En un principio la IDE de Arduino solo utilizaba sus propios controladores como es el caso del Arduino UNO pero se ha permitido la conexión de otros módulos a su plataforma para el desarrollo de software. Uno de estos casos es el del módulo ESP32, creado por una empresa ajena a Arduino pero que mediante una estandarización de las librerías ha permitido, como a muchos otros módulos, la conexión del ESP32 a la plataforma con un correcto funcionamiento.

Gracias a esto la IDE de Arduino se ha convertido en la mejor plataforma de desarrollo garantizando al programador que independientemente de la elección que realice del chip, este será soportado por la plataforma.

3.3. Desarrollo en Android Studio

En este punto del estudio del estado del arte surge una división de qué camino tomar ya que el desarrollo de aplicaciones para dispositivos móviles tiene dos grandes comunidades.

Por un lado, se encuentra el desarrollo en sistemas iOS de la empresa *Apple* para sus dispositivos iPhone basados en lenguaje Swift que facilita las tareas del programador ya que este se basa en la sencillez y un avanzado sistema de detección de errores. La gestión de estos proyectos se realiza mediante Appcode o Xcode 8, la IDE más común, ofreciendo al programador distintas herramientas.

Por otro lado, está el desarrollo de aplicaciones en sistemas Android basadas en lenguaje Java y lenguaje Kotlin el cual va ganando popularidad entre desarrolladores de este tipo de software. La IDE más conocida para la gestión de estos proyectos es Android Studio.

Conociendo esto, es necesario determinar para que sistemas se desea desarrollar el software por lo que se toma como referencia la popularidad de uso de los sistemas Android y iOS en el mercado actual.

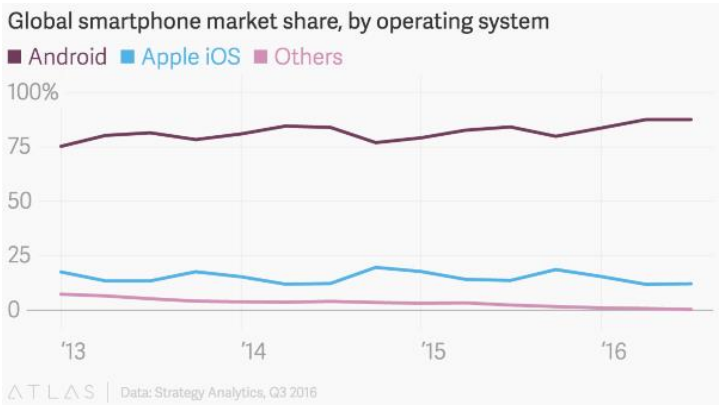


Ilustración 2: Comparación de mercado de Android y iOS

Se puede ver que, a pesar de la popularidad de Apple en el mundo de los teléfonos, Android sigue manejando el mercado de una forma aplastante de entorno a tres cuartas partes de todos los dispositivos móviles del mundo desde el 2013 al 2016. Por ello, para garantizar una mayor aceptación en el mercado y maximizar los beneficios junto con la experiencia del equipo en entornos Java, se decide utilizar el desarrollo para dispositivos Android.

3.4. Desarrollo en Visual Studio y Node.js

El boom de internet ha sido una de las invenciones que ha generado uno de los mayores impactos en la sociedad moderna cambiando de forma radical la forma de vida y desarrollo de actividades empresariales. Por ello, el lenguaje JavaScript ha ido ganando popularidad a lo largo de los años debido a la facilidad que presenta para poder aplicarse a ámbitos de servidores web.

No es de extrañar que cada vez son más las páginas web que precisan una interfaz interactiva con el cliente en vez de aquellas que se han presentado estáticas combinando HTML y CSS y aquí es donde entra el JavaScript para otorgar dinamismo a estos métodos. Esto lo logra de forma certera la herramienta de Node.js, capaz de gestionar códigos ligeros de construcción rápida y escalable en navegadores web permitiendo un gran número de conexiones simultaneas y garantizando el servicio a ese número de clientes.

Node.js se fundamenta en la IDE de Visual Studio como otras muchas herramientas de desarrollo como proyectos basados en C# para aplicaciones de escritorio. Esto vuelve a generar el mismo impacto que tiene la IDE de Arduino, unificar varios métodos de desarrollo para que el programador no se vea en la obligación de aprender a utilizar diferentes herramientas.

4. Conceptos teóricos

Una vez que se conocen los objetivos y hecho el resumen es fácil comprobar que este proyecto se ha divido en tres partes fundamentales donde encontramos el chip controlador de la chaqueta junto con los sensores que facilitan los datos, la aplicación móvil y finalmente el servidor web de los cuales se debe realizar un análisis teórico previo para comprender todo su funcionamiento.

4.1. Módulo ESP32

Los módulos ESP32, creados originalmente por la compañía *Espressif Systems* en el año 2016, son chips que se encuentran dentro de la familia de los SoC (*System on a Chip*). El diseño de estos sistemas integra módulos capaces de obtener el comportamiento de un ordenador abaja escala mediante el uso de un único circuito integrado. En el caso particular del ESP32 se ofrecen unas características únicas debido a su bajo coste de producción donde podemos encontrar:

- Bajo consumo de energía: garantizando una duración mayor de baterías integradas.
- Robusto diseño: consiguen trabajar en rangos extremos de temperatura desde los cuarenta grados centígrados bajo cero hasta los ciento veinticinco grados centígrados.
- Interconectividad: el módulo presenta la opción de conexión mediante Wi-Fi y Bluetooth para la comunicación con otros dispositivos a través de diferentes interfaces.

Para conocer cómo se estructuran estos módulos, es necesario conocer la arquitectura general de los SoC.

4.1.1. Arquitectura SoC

Los sistemas en un chip o SoC están constituidos por una serie de componentes que garantizan su funcionamiento independiente según el programa que determine el desarrollador. Estas partes son:

- Microcontrolador: posee la CPU del módulo con dos núcleos.
- Procesador de señales digitales (DSP): mediante el cual se garantiza de forma óptima el análisis de operaciones numéricas bajo altas velocidades. Esto es necesario para los diferentes sensores que envían los datos.
- Módulos de memoria: los módulos presentan memorias de tipo SRAM, ROM y Flash que variarán en capacidad dependiendo de la compañía que los comercialice.
- Contadores: para la gestión de recepción de datos de forma síncrona desde los sensores que lleve conectados.
- Controladores de interfaces analógicas: para la gestión de datos de errores y programas cargados desde herramientas de desarrollo.
- Reguladores de voltaje: normalmente de 3 y 5 voltios para garantizar la alimentación de los sensores.

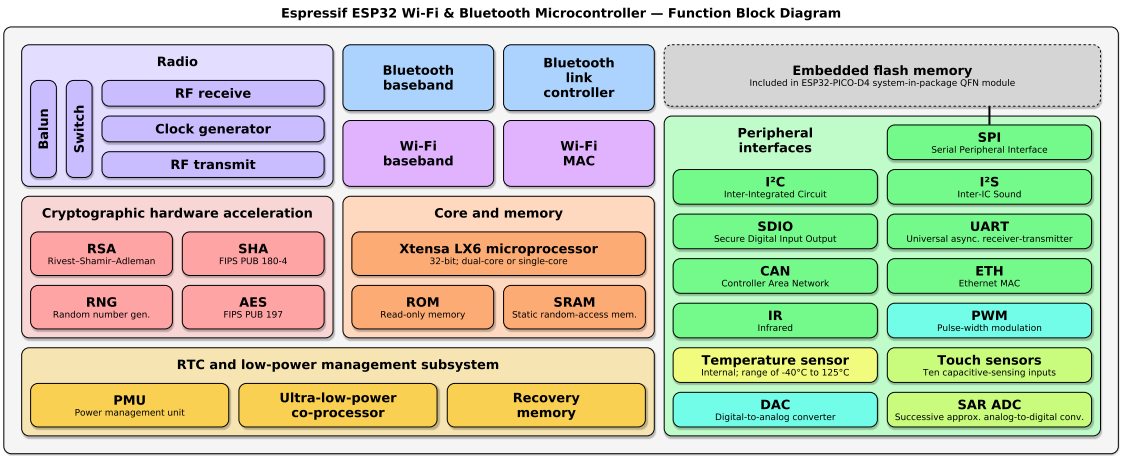


Ilustración 3: Diagrama de bloques: ESP32

4.1.2. Conectividad Bluetooth

Desarrollado por el grupo *Bluetooth Special Interés Group* en 1989, el Bluetooth es una especificación industrial para conexiones de tipo inalámbrico de área personal (WPAN) que permite la transmisión de datos en corto alcance mediante dos dispositivos vinculados mediante un canal.

Para el módulo ESP32 se posibilitan dos métodos de conexión Bluetooth en su versión 4.2:

- Bluetooth Seria: se trata de la conexión Bluetooth clásica que mantiene los conceptos básicos del protocolo.
- Bluetooth Low Energy (BLE): es la novedad de esta versión del Bluetooth que incorpora una pila de protocolo que, mediante una conexión simple pero rápida, permite a los dispositivos de baja potencia gestionar de forma óptima las conectividades Bluetooth.

4.1.3. Conectividad de los pines

El sensor ESP32 se fundamenta en un diseño general de los pines de conexión el cual se presenta en la siguiente ilustración:

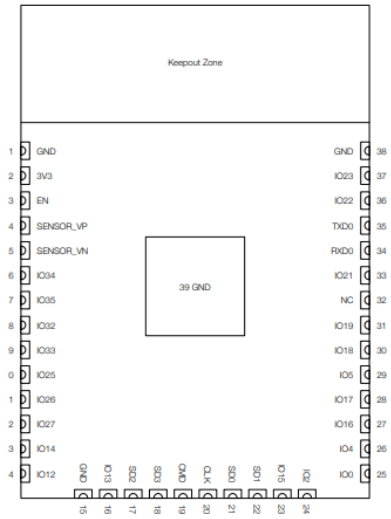


Ilustración 4Diseño de pines del ESP32

Se puede observar un total de 38 pines que permiten la gestión de los datos que gestiona el módulo ESP32 los cuales pueden ser configurados según la empresa de desarrollo. Los pines se agrupan en distintos tipos según su funcionalidad, para más detalle se incluye la siguiente tabla.

Nombre	Número	Tipo	Funcionalidad
GND	1	P	Toma a tierra
3V3	2	P	Fuente de alimentación
EN	3	I	Señal de habilitación de módulo. Activo en alto
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Toma a tierra
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/LCK	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1	22	I/0	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0

NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Toma a tierra

Tabla 1: Definición de los pines del módulo ESP32

Se puede determinar que la mayoría de los pines presentan pines de tipo GPIO, del inglés *General Purpose Input/Output* cuyo comportamiento se determina por el usuario en tiempo de ejecución. A esto se le suman los pines de GND de toma a tierra para los componentes y los pines de alimentación como el pin número 2 que ofrece 3.3 voltios.

Cabe destacar la funcionalidad de los pines 17, 18, 21 y 22 que se emplean para la gestión de la memoria flash, el pin 19 que se emplea para el CMD del sistema y el pin 20 que ofrece la señal de reloj para posibles sincronizaciones.

En el caso de este proyecto se ha empleado un módulo ESP32 diseñado por la compañía AZ-Delivery la cual ofrece el siguiente diagrama de salida de los pines:

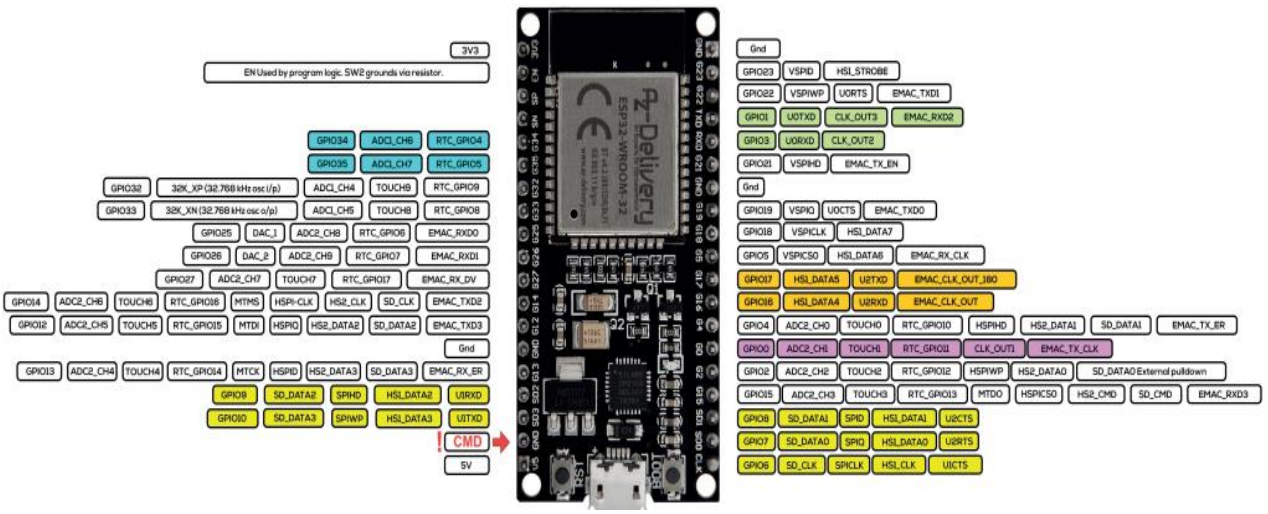


Ilustración 5: Diagrama de pines del ESP32 AzDelivery

En la leyenda de los pines se puede observar la clasificación de estos mediante cinco colores:

- Amarillo: empleados para la memoria flash interna por lo que se recomienda no emplearlos para otras finalidades.
- Azul: se utilizan como método de entrada único, pero no permiten el arranque o apagado interno del módulo.
- Verde: se utilizan por la funcionalidad USB/REPL.
- Naranja: se emplean en módulos ESP32-WROVER-KIT para el acceso a una SPI RAM externa.
- Blanco: uso libre para la gestión de datos del módulo ESP32.

4.2. Sensor BME680

Para obtener un ejemplo de sensor de obtención de datos, en este proyecto se ha optado por la integración de un sensor medioambiental basado en la arquitectura del BME680 de la empresa *DFRobot Gravity*.

Este sensor ofrece la posibilidad de obtener datos ambientales referentes a la temperatura, la presión, la humedad y la resistencia al gas cuyo objetivo principal es determinar la calidad del aire. Todos estos factores se determinan mediante la integración de un sensor de recepción de componentes orgánicos volátiles, un sensor de temperatura, un barómetro y un sensor de humedad. El dispositivo presenta conectividad tipo I2C para la obtención directa de los datos a la que se le suma una conexión SPI para posibles proyectos en los que se requiera una expansión.

Se ha optado por este sensor debido a que el método más utilizado para la detección de Covid-19 en el ambiente es la de determinar la calidad del aire y presencia de dióxido y monóxido de carbono en el ambiente pudiendo así prever de forma inmediata este factor y evitar poner en riesgo a las personas que usen la chaqueta.

Desde un punto técnico, el sensor BME680 es interesante de usar debido también a su bajo coste de producción, la integración de tecnologías MEMS ("*Microelectromechanical Systems*") lo que proporciona un tamaño reducido efectivo para dispositivos wearable, así como la posibilidad de conexión a alimentación tanto de 5 como de 3 voltios otorgando una mayor flexibilidad.

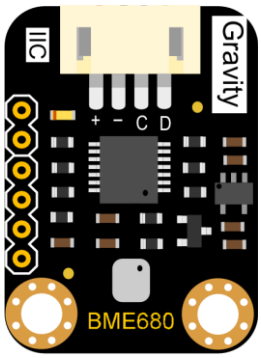


Ilustración 6: Diagrama del sensor BME680

4.2.1. Conectividad de los pines

El sensor BME680 posee cuatro pines de conexión que garantizan la correcta transmisión de los datos los cuales se recogen y explican en la siguiente tabla:

Símbolo	Nombre	Funcionalidad
+	VCC	Fuente de alimentación (admite desde 3,3 a 5 voltios)
-	GND	Toma a tierra
C	SCL	Señal de reloj tipo I2C
D	SDA	Señal de datos tipo I2C

Tabla 2: Definición de pines del sensor BME680

4.3. Aplicaciones móviles para Android

Los sistemas operativos Android, presentados originalmente por la compañía *Android Inc.* y adquirido posteriormente en 2005 por la compañía *Google,* es uno de los sistemas más utilizados a día de hoy no solo para software de dispositivos móviles, sino para tabletas, relojes inteligentes y otros dispositivos wearables. Este software está basado en núcleos Linux de software libre que ofrece al usuario una interfaz intuitiva para el uso del mismo.

Dado que se trata de software libre, el desarrollo de aplicaciones para estos sistemas operativos es más sencillo ya que existe gran cantidad de información y librerías respecto a muchos módulos dependiendo que software se quiera generar lo que le otorga un éxito rotundo en el mercado.

Sí es cierto que el desarrollo para aplicaciones en dispositivos iOS también está permitido pero la comunidad que ofrece Android para desarrolladores es mucho más amplia que la mencionada por lo que garantiza seguridad a la hora de generar nuevos proyectos sobre todo para programadores que se acaban de iniciar en este mundo.

4.4. Servidores web

Los servidores web o servidores de protocolo HTTP son un tipo de software que gestiona peticiones del tipo cliente – servidor hospedado en un navegador web donde un cliente, ya sea usuario o programa, realiza solicitudes al servidor bajo una IP establecida.

Los servidores web poseen diferentes tipos de métodos para garantizar la comunicación entre la parte del cliente y la parte del servidor. Los más utilizados son:

- Método GET: el método GET se emplea para devolver al cliente la representación de los datos solicitados bajo la url indicada en diferentes formatos. El más común es formato HTML que ofrece una vista acomodada al navegador web.
- Método POST: el método POST emplea el envío de datos en el mismo cuerpo de la petición HTTP. Dichos datos se recogen mediante formularios y se codifican en la misma url.
- Método PUT: el método PUT se emplea con el fin de actualizar los contenidos de un determinado fichero y crearlos si fuese necesario. Estos cambios realizados no generan cambios en los ficheros originales generando un método idempotente.
- Método DELETE: este método permite eliminar un recurso identificado en una url.
- Método HEAD: el método HEAD tiene una funcionalidad idéntica al método GET, pero devuelve las cabeceras HTTP en vez del contenido.

Todos estos métodos están definidos dentro del servidor web los cuales serán utilizados según corresponda la funcionalidad que se requiera para garantizar el correcto uso del mismo.

4.4.1. Conectividad HTTP

El protocolo HTTP, de sus siglas "*Hypertext Transfer Protocol*", es un sistema de petición de datos en medios cliente – servidor que se envían mediante una entidad que lo más común suele ser un servidor web. Cada petición es individual por cliente y se envía al servidor el cual gestiona la solicitud y responde con lo correspondiente.

HTTP es un protocolo con sesiones, pero sin estados, es decir, no se guardan los datos entre dos peticiones dentro de la misma sesión. Estas conexiones no se gestionan por el protocolo HTTP ya que quedan dentro de la capa de transporte en el modelo OSI, pero se fundamenta

en el uso del protocolo TCP dado su mayor fiabilidad frente a protocolos UDP. A lo largo del desarrollo de este protocolo se han podido incorporar controladores para distintos elementos como:

- Caché: se puede indicar en el HTTP como se desean los elementos en caché.
- Autentificación: el protocolo HTTP permite proteger las páginas web mediante una autentificación.
- Proxies y tunneling: las peticiones HTTP utilizan proxies para acceder a los servidores.
- Sesiones: el almacenamiento de cookies permite al protocolo HTTP ofrecer un mayor dinamismo al servidor ofreciéndole datos para cargar las sesiones anteriores de los usuarios.

Toda esta gestión de datos se gestiona mediante dos tipos de mensajes:

- Peticiones: se define la acción que quiere realizar el cliente, normalmente GET o POST junto con la url del recurso. A esto se le suma la versión y cabeceras del HTTP, así como parámetros opcionales para tramitar correctamente la solicitud.
- Respuestas: las respuestas están comprendidas por la versión de protocolo HTTP que se está empleando, un código mostrando si se ha logrado o no la acción solicitada, un mensaje de estado y las cabeceras al igual que en las peticiones. A esto se le puede incluir parámetros opcionales como el recurso solicitado.

5. Metodología, técnicas y herramientas

Conociendo todos los conceptos teóricos que se precisan para desarrollar el proyecto, es momento de pasar a las herramientas y técnicas que hay que utilizar para obtener un producto funcional que albergue mediante los lenguajes de programación elegidos, todo el sistema.

En este apartado se encuentran recogida la metodología seguida para el desarrollo del proyecto, tras lo cual se muestran las técnicas y técnicas utilizadas para la gestión de los módulos, la aplicación móvil y el servidor web en las que se profundiza en los lenguajes de programación utilizados, así como las herramientas y librerías utilizadas para obtener el producto.

5.1. Metodología

La metodología fundamental del proyecto se basa en la metodología del proceso unificado. Esta metodología se caracteriza por el uso de casos de uso para su direccionamiento en la que se centrará al proyecto en la arquitectura del mismo buscando siempre la modularidad iterativa e incremental. Se decide utilizar este modelo debido a la limitación de tiempo presente de un total de cuatro meses y que dicho proyecto se gestiona únicamente por una persona lo cual dificulta el desarrollo de este.

El ciclo de vida del proyecto del proceso unificado se basa en la realización de iteraciones dentro de las etapas clásicas (modelado de negocio, requisitos, análisis, diseño, implementación, pruebas y despliegue) en las cuales se irá puliendo la idea principal hasta poder obtener un producto software final.

Para llevarlo a cabo, se separará el proyecto en las siguientes iteraciones:

- Iteración inicial: conformada por el estudio del estado del arte de cada una de las partes principales del proyecto para poder consensuar un prototipo inicial del producto en la que se recogen los requisitos, el análisis del sistema y un diseño de la arquitectura.
- Iteración de captación de datos: en esta iteración nos centramos en el diseño del software del módulo ESP32 para su futuro uso de una forma efectiva buscando la optimización de la extracción realizando un previo estudio, así como la investigación del uso de envío de esos datos por conexión Bluetooth.
- Iteración de muestra de datos: el objetivo será realizar la captación de los datos enviados por bluetooth al dispositivo móvil y establecer una interfaz que los muestre de forma ordenada por pantalla. Esto se debe gestionar realizando un estudio previo de desarrollo de aplicaciones móviles en plataformas Android que garanticen un correcto funcionamiento de la aplicación.
- Iteración de presentación de avisos: como última iteración, se realiza un estudio de como poder gestionar un servidor web que pueda recibir los avisos de las chaquetas, así como la trama de datos para casos de estudio. Para esto y al igual que en las dos iteraciones anteriores, es necesario realizar un estudio previo para conocer cómo llevar a cabo un servidor web.

Conociendo las iteraciones principales del proyecto, se generan los hitos del mismo que se descomponen en etapas y tareas. Para gestionar todo esto, se genera un proyecto de organización mediante la herramienta de *Microsoft Project* del cual se recogerán todos los detalles en el Anexo I donde se explica el proceso con mayor detalle.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Inicio	**21,38 días**	**lun 01/02/21**	**mar 02/03/2**		
Modelado de negocio	**5 días**	**lun 01/02/21**	**vie 05/02/21**		
Investigación del estado del arte	2 días	lun 01/02/21	mar 02/02/21		José Pizzano Martín
Implementaciones existentes	1 día	mié 03/02/2	mié 03/02/2	3	José Pizzano Martín
Investigación de herramientas de desarrollo	2 días	jue 04/02/21	vie 05/02/21	4	José Pizzano Martín
Requisitos	**1 día**	**lun 08/02/21**	**lun 08/02/21**		
Modelo de requisitos inicial	1 día	lun 08/02/21	lun 08/02/21	2	José Pizzano Martín
Análisis	**4 días**	**mar 09/02/2**	**vie 12/02/21**		
Análisis del sistema Arduino	1 día	mar 09/02/2	mar 09/02/2	6	José Pizzano Martín
Análisis de la App Android	2 días	mié 10/02/2	jue 11/02/21	6	José Pizzano Martín
Análisis del servidor web	1 día	vie 12/02/21	vie 12/02/21	6	José Pizzano Martín
Diseño	**10,75 días**	**lun 15/02/21**	**lun 01/03/21**		
Diseño inicial de arquitectura	6 horas	lun 15/02/21	lun 15/02/21	8	José Pizzano Martín
Diseño inicial de sistema Arduino	6 horas	mar 16/02/21	mié 17/02/21	9	José Pizzano Martín
Diseño inicial de App Android	6 horas	lun 15/02/21	mar 16/02/2	10	José Pizzano Martín
Diseño inicial de servidor web	6 horas	lun 01/03/21	lun 01/03/21	11	José Pizzano Martín
Implementación	**0,5 días**	**lun 01/03/21**	**mar 02/03/2**		
Preparación del entorno de trabajo	2 horas	lun 01/03/21	lun 01/03/21	12	José Pizzano Martín
Instalación de herramientas de trabajo	2 horas	mar 02/03/21	mar 02/03/21	18	José Pizzano Martín
Pruebas	**0,13 días**	**mar 02/03/2**	**mar 02/03/2**		
Pruebas de correcto funcionamiento del software de desarrollo	1 hora	mar 02/03/21	mar 02/03/21	17	José Pizzano Martín
Hito fin de iteración inicial	0 días	mar 02/03/2	mar 02/03/2	1	José Pizzano Martín

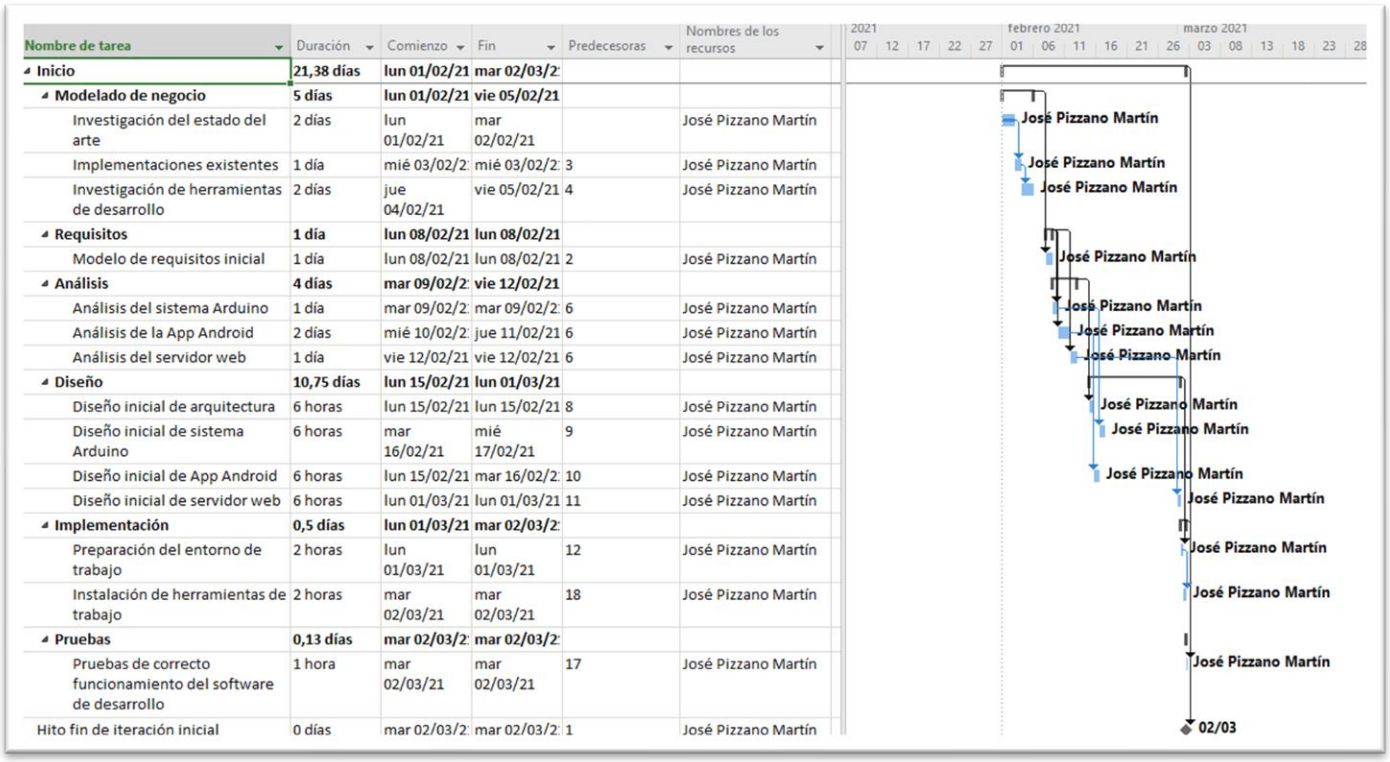


Ilustración 7: Diagrama de Gannt: Iteración inicial

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Iteración de captación de datos	**17 días**	**mar 02/03/2**	**jue 25/03/21**		
Modelo de negocio	**9 días**	**mar 02/03/2**	**lun 15/03/21**		
Investigación sobre módulo ESP32	1 día	mar 02/03/21	mié 03/03/21	22	José Pizzano Martín
Investigación sobre sensor BME680	2 horas	mar 09/03/21	mar 09/03/21	22	José Pizzano Martín
Investigación de conectividad Bluetooth con ESP32	1 día	vie 12/03/21	lun 15/03/21	22	José Pizzano Martín
Requisitos	**1 día**	**lun 15/03/21**	**mar 16/03/2**		
Ajuste de requisitos del sistema Arduino	1 día	lun 15/03/21	mar 16/03/21	24	José Pizzano Martín
Análisis	**1 día**	**mar 16/03/2**	**mié 17/03/2**		
Ajuste del análisis del sistema Arduino	1 día	mar 16/03/21	mié 17/03/21	28	José Pizzano Martín
Diseño	**1 día**	**mié 17/03/2**	**jue 18/03/21**		
Ajuste del diseño del sistema Arduino	1 día	mié 17/03/21	jue 18/03/21	30	José Pizzano Martín
Implementación	**4 días**	**jue 18/03/21**	**mié 24/03/2**		
Implementación del sistema Arduino	2 días	jue 18/03/21	lun 22/03/21	32	José Pizzano Martín
Integración y despliegue del sistema Arduino	2 días	lun 22/03/21	mié 24/03/21	35	José Pizzano Martín
Pruebas	**1 día**	**mié 24/03/2**	**jue 25/03/21**		
Pruebas del sistema Arduino	1 día	mié 24/03/2	jue 25/03/21	34	José Pizzano Martín
Hito fin de iteración de captación de datos	0 días	jue 25/03/21	jue 25/03/21	23	José Pizzano Martín

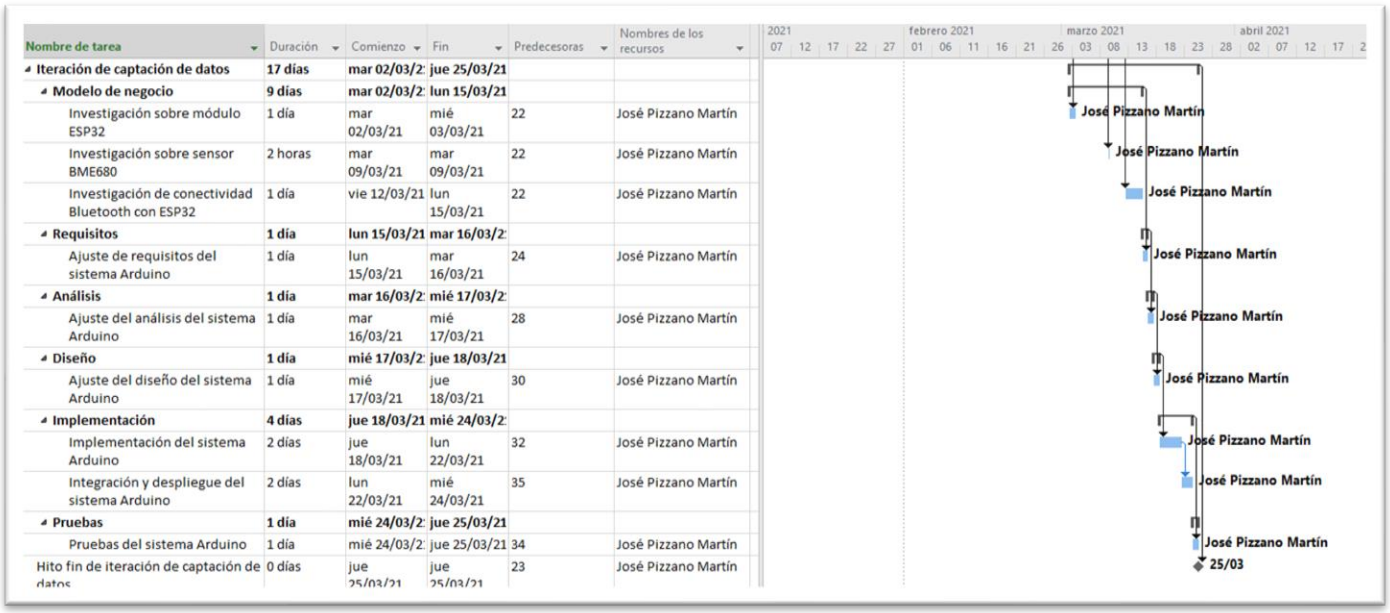
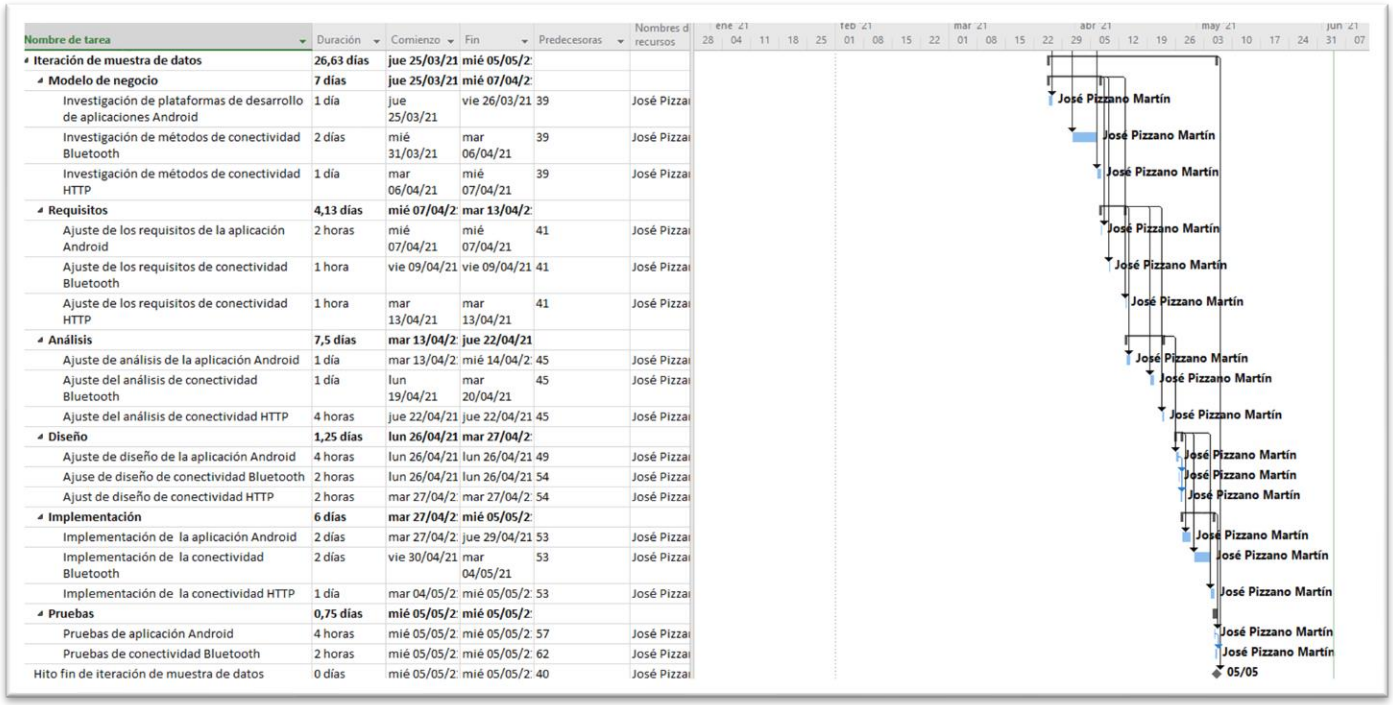
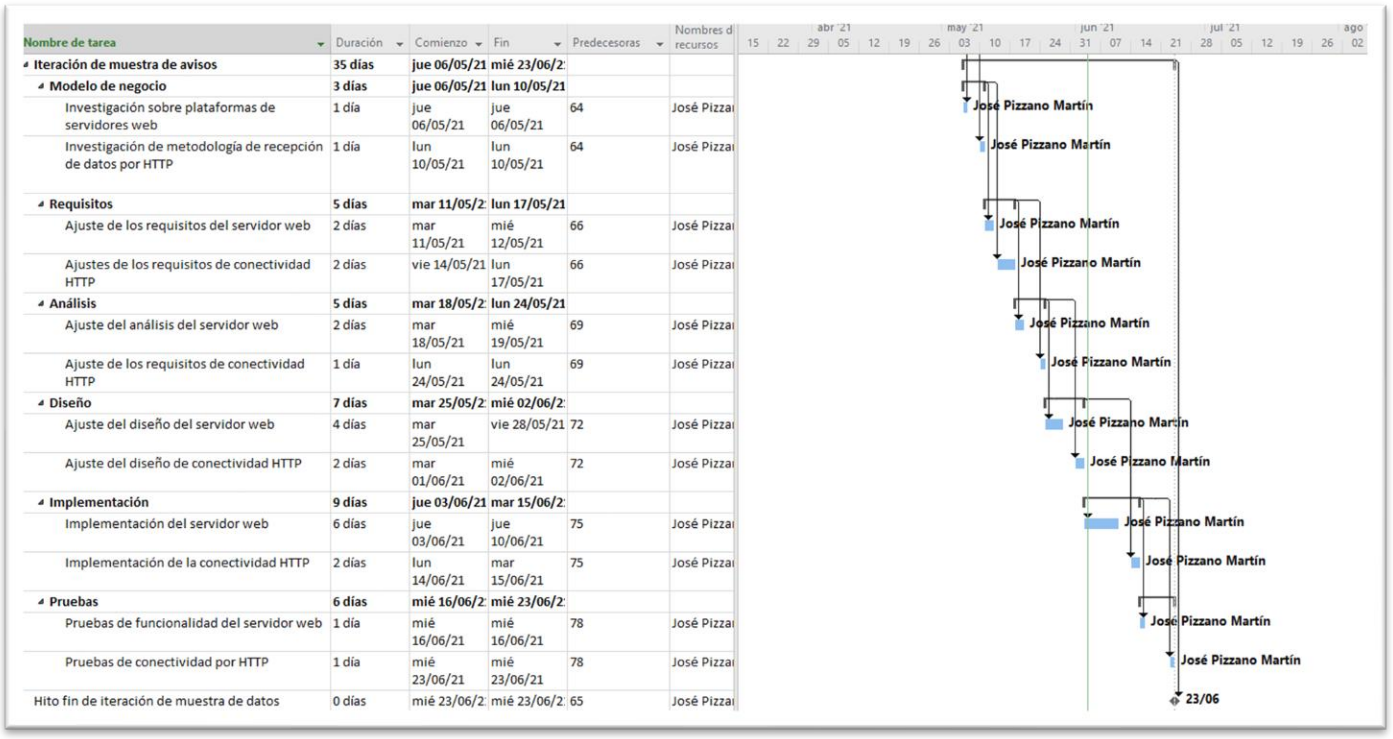


Ilustración 8: Diagrama de Gannt: Iteración de captación de datos



Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de recursos
Iteración de muestra de datos	**26,63 días**	**jue 25/03/21**	**mié 05/05/2**		
Modelo de negocio	**7 días**	**jue 25/03/21**	**mié 07/04/2**		
Investigación de plataformas de desarrollo de aplicaciones Android	1 día	jue 25/03/21	vie 26/03/21	39	José Pizza
Investigación de métodos de conectividad Bluetooth	2 días	mié 31/03/21	mar 06/04/21	39	José Pizza
Investigación de métodos de conectividad HTTP	1 día	mar 06/04/21	mié 07/04/21	39	José Pizza
Requisitos	**4,13 días**	**mié 07/04/2**	**mar 13/04/2**		
Ajuste de los requisitos de la aplicación Android	2 horas	mié 07/04/21	mié 07/04/21	41	José Pizza
Ajuste de los requisitos de conectividad Bluetooth	1 hora	vie 09/04/21	vie 09/04/21	41	José Pizza
Ajuste de los requisitos de conectividad HTTP	1 hora	mar 13/04/21	mar 13/04/21	41	José Pizza
Análisis	**7,5 días**	**mar 13/04/2**	**jue 22/04/21**		
Ajuste de análisis de la aplicación Android	1 día	mar 13/04/2	mié 14/04/2	45	José Pizza
Ajuste del análisis de conectividad Bluetooth	1 día	lun 19/04/21	mar 20/04/21	45	José Pizza
Ajuste del análisis de conectividad HTTP	4 horas	jue 22/04/21	jue 22/04/21	45	José Pizza
Diseño	**1,25 días**	**lun 26/04/21**	**mar 27/04/2**		
Ajuste de diseño de la aplicación Android	4 horas	lun 26/04/21	lun 26/04/21	49	José Pizza
Ajuse de diseño de conectividad Bluetooth	2 horas	lun 26/04/21	lun 26/04/21	54	José Pizza
Ajust de diseño de conectividad HTTP	2 horas	mar 27/04/2	mar 27/04/2	54	José Pizza
Implementación	**6 días**	**mar 27/04/2**	**mié 05/05/2**		
Implementación de la aplicación Android	2 días	mar 27/04/2	jue 29/04/21	53	José Pizza
Implementación de la conectividad Bluetooth	2 días	vie 30/04/21	mar 04/05/21	53	José Pizza
Implementación de la conectividad HTTP	1 día	mar 04/05/2	mié 05/05/2	53	José Pizza
Pruebas	**0,75 días**	**mié 05/05/2**	**mié 05/05/2**		
Pruebas de aplicación Android	4 horas	mié 05/05/2	mié 05/05/2	57	José Pizza
Pruebas de conectividad Bluetooth	2 horas	mié 05/05/2	mié 05/05/2	62	José Pizza
Hito fin de iteración de muestra de datos	0 días	mié 05/05/2	mié 05/05/2	40	José Pizza

Ilustración 10: Diagrama de Gannt: Iteración de muestra de datos



Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de recursos
Iteración de muestra de avisos	**35 días**	**jue 06/05/21**	**mié 23/06/2**		
Modelo de negocio	**3 días**	**jue 06/05/21**	**lun 10/05/21**		
Investigación sobre plataformas de servidores web	1 día	jue 06/05/21	jue 06/05/21	64	José Pizza
Investigación de metodología de recepción de datos por HTTP	1 día	lun 10/05/21	lun 10/05/21	64	José Pizza
Requisitos	**5 días**	**mar 11/05/2**	**lun 17/05/21**		
Ajuste de los requisitos del servidor web	2 días	mar 11/05/21	mié 12/05/21	66	José Pizza
Ajustes de los requisitos de conectividad HTTP	2 días	vie 14/05/21	lun 17/05/21	66	José Pizza
Análisis	**5 días**	**mar 18/05/2**	**lun 24/05/21**		
Ajuste del análisis del servidor web	2 días	mar 18/05/21	mié 19/05/21	69	José Pizza
Ajuste de los requisitos de conectividad HTTP	1 día	lun 24/05/21	lun 24/05/21	69	José Pizza
Diseño	**7 días**	**mar 25/05/2**	**mié 02/06/2**		
Ajuste del diseño del servidor web	4 días	mar 25/05/21	vie 28/05/21	72	José Pizza
Ajuste del diseño de conectividad HTTP	2 días	mar 01/06/21	mié 02/06/21	72	José Pizza
Implementación	**9 días**	**jue 03/06/21**	**mar 15/06/2**		
Implementación del servidor web	6 días	jue 03/06/21	jue 10/06/21	75	José Pizza
Implementación de la conectividad HTTP	2 días	lun 14/06/21	mar 15/06/21	75	José Pizza
Pruebas	**6 días**	**mié 16/06/2**	**mié 23/06/2**		
Pruebas de funcionalidad del servidor web	1 día	mié 16/06/21	mié 16/06/21	78	José Pizza
Pruebas de conectividad por HTTP	1 día	mié 23/06/21	mié 23/06/21	78	José Pizza
Hito fin de iteración de muestra de datos	0 días	mié 23/06/2	mié 23/06/2	65	José Pizza

Ilustración 9: Diagrama de Gannt: Iteración de muestra de avisos

5.2. Técnicas

En este apartado se muestran las técnicas teóricas necesarias para elaborar el sistema completo.

5.2.1. Elicitación de requisitos

La metodología de la elicitación de requisitos en un método de desarrollo fundamental para el desarrollo de proyectos basados en software que generan un gran impacto en el diseño y en las fases del ciclo de vida de las iteraciones.

Para conseguir los objetivos de esta técnica es necesario:

- Identificar objetivos del sistema.
- Identificar los requisitos funcionales, de la información y no funcionales.
- Priorizar los objetivos del proyecto.
- Preparar etapas de fijación de objetivos.

5.3. Herramientas

5.3.1. Arduino IDE

El IDE o entorno de desarrollo integrado de la compañía Arduino es una herramienta fundamental para el desarrollo de software enfocado en chips y módulos controladores.

Ilustración 11: Logo de Arduino

Este software basado en lenguaje de programación Java permite generar archivos en lenguaje C y C++ para su posterior compilación en un gran catálogo de módulos. Eso es gracias a la incorporación de una biblioteca propia del entorno denominada “Wiring” la cual permite una estandarización del sistema de compilación para los distintos controladores y una detección automática de la misma facilitando la gestión de proyectos basados en controladores.

Antes de comenzar a programar, es necesario asegurar la correcta funcionalidad del módulo que se desea utilizar mediante la pestaña de herramientas donde se recogen en una lista las características fundamentales de los controladores. En caso de que el módulo requerido no apareciese, se le permite al programador buscar el modelo para instalar las bibliotecas correctas respecto a esta para que el entorno sea capaz de generar un programa leíble por el chip.

La funcionalidad del IDE de Arduino es similar a cualquier otra plataforma de desarrollo software donde se deben indicar las bibliotecas necesarias para la gestión de la funcionalidad del software a la que se le suma la declaración de variables necesarias para la gestión del software.

Seguido de esto aparecen las dos funciones principales que harán que nuestro sistema funcione correctamente.

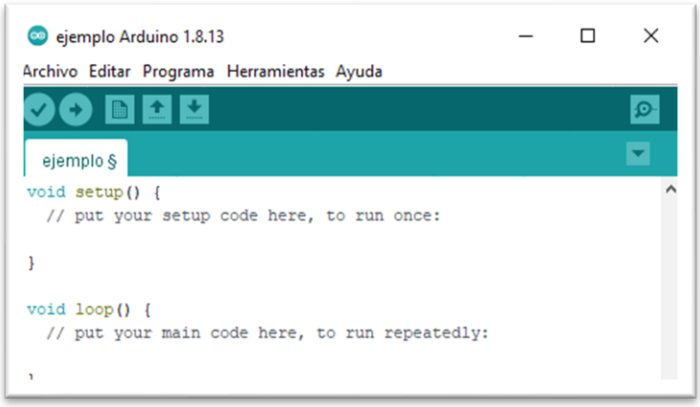


Ilustración 12: Ejemplo de programa en Arduino

En el ejemplo de la ilustración 12 se pueden ver las funciones principales de todo programa basado en la IDE de Arduino, las cuales son:

- Función "setup": en esta función de configuración se insertarán todos los elementos correspondientes a la inicialización de los módulos conectados, así como los sensores o funcionalidades secundarias que se puedan encontrar en este módulo.
- Función "loop": está segunda función está fundamentada en un bucle infinito que no finalizará hasta que no se apague o reinicie el dispositivo que tenga instalado el software. Esto permite la ejecución constante del programa facilitando el intercambio de datos y conexiones.
  Aquí es donde se almacenarán todas las operaciones relacionadas con la modificación de los datos que interactúen con el programa.

Una vez que se han realizado todas las modificaciones al software que se desee ejecutar, la IDE de Arduino nos ofrece la opción de verificar el código lo que compilará el programa para garantizar que no existen errores dentro de este y la opción de subir el código que genera un programa que el controlador conectado será capaz de identificar y ejecutar de forma correcta.

5.3.2. Android Studio

EL IDE o entorno de desarrollo integrado "Android Studio" es la herramienta oficial desarrollada por *Google* en 2014 para el desarrollo de aplicaciones en dispositivos Android. La aplicación permite la gestión de software en lenguajes de Java, C++ y Kotlin.

Ilustración 13: Logo de Android Studio

Cabe indicar que este último es un intento por parte de la comunidad Android de estandarizar el desarrollo de este tipo de software. Este lenguaje está fuertemente influenciado por otros lenguajes como Java, Scala, Groovy y C# el cual se fundamenta en la programación orientada a objetos con completa interoperabilidad con sistemas Java.

Este entorno de desarrollo posee una interfaz similar a otros programas como Visual Studio donde nos divide los proyectos en los ficheros de datos y los de interfaz en los que el programador podrá asignar la funcionalidad y la estética, en este caso de la aplicación móvil.

Al igual que la IDE de Arduino, Android Studio ofrece la posibilidad de realizar una compilación previa del programa para depurar los errores y se le suma un conjunto de emuladores de dispositivos móviles facilitando el desarrollo de las aplicaciones junto con la posibilidad de la ejecución en dispositivos reales si se encuentran conectados al ordenador lo que permite al desarrollador la interacción directa con la interfaz de la aplicación viendo su respuesta en tiempo real.

5.3.3. Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma diseñado en 2009 por la compañía *Node.js Developers Joyent* que se basa en el motor V8 del navegador Google Chrome para la gestión de servidores web mediante lenguaje JavaScript.

Ilustración 14: Logo de Node.js

Este entorno se presenta dentro de la IDE de Visual Studio que permite al programador acceder a todos los elementos de la herramienta en la que encontramos ficheros JavaScript donde se alberga el comportamiento de las solicitudes web que recibe el servidor y archivos HTML que presentan la parte visual de este último.

Para facilitar el uso de la comunicación entre los archivos JavaScript y HTML existen una herramienta con funcionalidad back-end de software libre denominada Express.

5.3.3.1. Express

La herramienta Express está diseñada para facilitar al diseñador el desarrollo de sus aplicaciones web de una forma minimalista y flexible garantizando que no es necesario un gran conocimiento previo para su uso.

Ilustración 15: Logo Express

Dentro de Express existen muchos motores de plantilla dependiendo de la funcionalidad que se necesite presentar en el sistema donde las más conocidas son: *Pug*, *Haml.js* y *EJS* la cual se introduce ahora.

EJS

Uno de los motores de plantilla más utilizados es EJS, una extensión para Express que permite insertar código en plantillas HTML de forma dinámica y rápida ofreciendo una mayor optimización.

La sintaxis de EJS para la apertura es: "< % [opción]" donde entre las opciones se encuentra:

- "_": el guion bajo elimina todos los espacios en blanco antes de ella.
- "=": el símbolo igual envía los valores a la plantilla HTML.
- "-": el guion envía el valor sin escape a la plantilla HTML.
- "#": el asterisco indica la etiqueta de un comentario sin ejecución.
- "%": si se quiere insertar un símbolo "%" literal.

La sintaxis de EJS para el cierre es: "[opción] % >" donde entre las opciones se encuentra:

- "-": el guion indica el recorte después de una nueva línea.
- "_": el símbolo de guion bajo elimina todos los espacios en blanco después de ella.

5.3.4. Lenguajes de programación empleados

Como se ha ido indicando en los apartados anteriores, este proyecto se ha fundamentado en tres lenguajes de programación necesarios para el desarrollo de los sistemas del proyecto junto con las librerías necesarias para la gestión de los proyectos.

5.3.4.1. C++

El lenguaje de programación C++, diseñado por el grupo *Bjarne* Stroustrup, es un lenguaje que se fundamenta en C con el objetivo de estar orientado a objetos extendiendo la funcionalidad de este último.

Ilustración 16: logo de C++

Las características principales de C++ son:

- Es un lenguaje abierto estandarizado por modelos ISO.
- Es un lenguaje compilable de forma nativa por la máquina garantizando mayor velocidad.
- Es un lenguaje inseguro por lo que se espera que el programador tenga un conocimiento previo de las tareas que esté realizando.
- Admite la declaración de datos estáticos y dinámicos en tiempos de compilación agregando un grado más de flexibilidad.
- Es portátil y ejecutable en un alto número de dispositivos.
- Es capaz de compilar todo el código escrito en C ya que proviene de este.
- Tiene un gran número de librerías facilitando las tareas de programación.

Aplicado a la interfaz de Arduino para el desarrollo de este sistema, es necesario el uso de cuatro bibliotecas:

- "Wire.h": con esta librería se consigue conectar con los dispositivos basados en I2C/TWI que contienen línea de reloj y línea de datos.
- "SPI.h": la librería permite comunicarse con dispositivos fundamentados en arquitecturas SPI.
- "BluetoothSerial.h": se trata de una biblioteca sencilla enfocada a la conexión mediante Bluetooth de tipo serial para los módulos ESP32.
- "DFRobot_BME680_I2C.h": librería ofrecida por la compañía Gravity para la gestión del sensor BME680 y los parámetros que otorga.

5.3.4.2. Java

El lenguaje de programación Java es un proyecto planteado por la empresa *Sun Microsystems* durante 1995 que ha ido ganando importancia a lo largo de los años hasta convertirse en una plataforma informática fundamental para el uso y funcionamiento de la mayoría de los dispositivos electrónicos. Esta herramienta es de uso completamente gratuito y se fundamenta en estar orientado a objetos.

Este lenguaje derivado de los lenguajes de programación C y C++, pero siendo este un lenguaje de alto nivel.

Ilustración 17: Logo de Java

Para el desarrollo de la aplicación móvil se han utilizado los siguientes paquetes que se agrupan en:

Paquete Android

- "android.bluetooth": esta librería perite el acceso a la conectividad Bluetooth para aplicaciones Android con otros dispositivos y la transferencia de datos entre estos.
- "android.os": proporciona operaciones básicas para los servicios del sistema. Dentro de esta biblioteca se han seleccionado tres:
    - "android.os.Build": otorga información respecto a la compilación extraída desde las propiedades del sistema.
    - "android.os.Bundle": genera un mapeado a partir de campos String de varios valores parseables.
    - "android.os.Handler": permite enviar y procesar mensajes y objetos ejecutables asociados al hilo "MessageQueue".
    - "android.os.Message": define un mensaje que contiene una descripción y datos seleccionados que pueden enviarse a un Handler.

- "android.widget": la librería otorga elementos visuales para la interfaz de la aplicación. Dentro de esta biblioteca se han utilizado dos clases:
    - "android.wirget.TextView": elemento visual de la interfaz que presenta texto por pantalla.
    - "android.widget.Toast": presenta un texto al usuario como un mensaje por pantalla.

Paquete Java

- "java.io": proporciona a la aplicación entrada y salida de paquetes de datos, serialización y ficheros de sistema. Dentro de este paquete sean seleccionado tres clases:
    - "java.io.IOException": ofrece una señal de tipo I / O debido a alguna excepción que haya surgido.
    - "java.io.InputStream": clase abstracta que representa el input de los datos.
    - "java.io.OutputStream": clase abstracta que representa el output de los datos.
    - "java.io.UnsuportedEncodingException": error emergente en caso de que la codificación de caracteres no esté soportada por el sistema.
- "java.net": proporciona las clases para la gestión de aplicaciones que implementan usos de red. De este paquete se han elegido dos clases:
    - "java.net.InetAddress": representa el protocolo de direcciones IP para conexiones a internet.
    - "java.net.UnknownHostException": lanza un error en caso de que el host solicitado no se logre encontrar.
- "java.util.UUID": del paquete útil solo se ha utilizado la clase UUID que sirve para presentar una clase como inmutable, universal y con un identificador único.

Paquete Volley

Volley es una biblioteca que permite gestionar las conexiones vía HTTP de una forma más sencilla para el programador agilizando así el uso de redes en el desarrollo de aplicaciones. Dentro de todas las clases presentes de esta biblioteca, se han utilizado para el proyecto un total de siete:

- "com.android.volley.AuthFailureError": genera un error indicando que ha surgido un error de autentificación durante la solicitud.
- "com.android.volley.Request": solicitud estándar para todas redes.
- "com.android.volley.RequestQueue": cola de solicitudes de diferentes hilos.
- "com.android.volley.Response": encapsula las respuestas para un envío.
- "com.android.volley.VolleyError": excepción que encapsula los errores de la biblioteca Volley.
- "com.android.volley.toolbox.StringRequest": genera una solicitud para la recepción de un cuerpo en una URL determinada como un String.
- "com.android.volley.toolbox.Volley": caché estándar de la carpeta.

Paquete Gson

Gson, al igual que la biblioteca anterior, es una biblioteca para Java que se utiliza para la conversión de ficheros JSON en objetos java y el proceso inverso. En este caso no se utilizan clases más que las de parsear datos a formato JSON.

5.3.4.3. JavaScript

El lenguaje JavaScript, diseñado en 1995 por la compañía *Netscape Communications* junto con la *Fundación Mozilla*, es un lenguaje de programación ligero y de alto basado en Java que se compila en el momento de la solicitud y se emplea en la mayoría de los navegadores web los cuales han desarrollado su motor a partir de este lenguaje.

Ilustración 18: Logo de JavaScript

Muchos entornos de desarrollo de servidores web emplean este lenguaje para el desarrollo de sus proyectos como es el caso de Node.js, mencionado en apartados anteriores en el que se han indicado todas las modificaciones para el uso de este lenguaje ya que no presenta bibliotecas.

JavaScript es un lenguaje muy utilizado debido a su fácil aprendizaje, así como su versatilidad logrando una alta adaptación a diferentes medios web.

5.3.4.4. HTML

El lenguaje HTML, de las siglas en inglés HyperText Markup Language, se fundamenta en realizar la estructuración de los documentos mediante el uso de etiquetas que se emplean para el desarrollo de páginas web.

Ilustración 19: Logo de HTML

Compaginado con JavaScript, el lenguaje HTML es uno de los elementos principales a la hora de generar páginas web donde en este caso nos ofrece la estructuración de estas sin necesidad del uso de datos más que los del propio lenguaje.

Cualquier elemento externo a este como por ejemplo las imágenes, no se incorporan directamente en el texto, sino que serán referenciadas para poder presentarse.

Al igual que en el caso del lenguaje anterior, HTML no presenta bibliotecas de donde obtener diferentes herramientas, pero si cuenta con un gran número de plantillas y métodos de inclusión de código generando archivos de texto dinámicos.

5.3.4.5. CSS

El CSS o Cascading Style Sheets, es un lenguaje que se emplea para elaborar estilos de diseño gráfico para las páginas web ya que suele ir vinculado al HTML mediante referencia como se ha indicado en el apartado anterior o incluso incrustado en el propio HTML indicando así su estilo.

5.3.4.6. Bootstrap

Una de las herramientas de plantillas básicas para el uso de HTML es la herramienta de Bootstrap que proporciona un gran número de archivos con diferentes estilos y categorías facilitando al programador el diseño de su sitio web.

5.3.5. Git y Github

El programa Git es un software libre enfocado en el control de versiones pensado en aumentar la eficiencia de las versiones del software facilitando la compatibilidad de este con los dispositivos.

Por otro lado, GitHub es una plataforma web popularmente conocida que utiliza el sistema de versiones de Git para almacenar diferentes proyectos software y sus diferentes versiones garantizando la difusión de software libre y soporte entre programadores.

5.3.6. Draw.io

La plataforma web de draw.io es un sistema online basado en el almacenamiento en la nube que consta con una herramienta destinada a la creación de diagramas explicativos de forma gratuita.

El objetivo de uso de este programa es el poder representar de una forma económica y rápida los sistemas creados, así como el proyecto general.

5.3.7. Visual Paradigm

Visual Paradigm es una aplicación de escritorio comúnmente utilizada para el desarrollo de diagramas de UML lo cual facilita la elaboración de proyectos dentro del ámbito de la ingeniería.

5.3.8. Microsoft Project y Microsoft Word

Dos herramientas comúnmente conocidas de las cuales la primera ya se ha presentado para el desarrollo de la planificación temporal gracias a la posibilidad de insertar las iteraciones y actividades del proyecto y asignarles un intervalo de tiempo de desarrollo y se presenta finalmente un diagrama de Gannt.

Por otro lado, la herramienta de Word permite la gestión de documentos de texto y facilita generar distintos estilos para poder desarrollar sus documentos.

5.3.9. EzEstimate

Herramienta CASE necesaria para la elaboración de estimaciones de tiempo para proyectos donde se insertan los casos de uso de un sistema y estima el tiempo que llevaría el completo desarrollo de este.

5.3.10. PostMan

Se trata de una plataforma enfocada en el desarrollo de APIs permitiendo simular peticiones de clientes fictios a las aplicaciones de forma local para lograr una depuración íptima del software que se está desarrollando.

5.3.11. Doxygen

Generador automático de documentación de diferentes entornos de programación mediante los comentarios presentes en el código. Este software es capaz de recoger la documentación de diferentes lenguajes de programación.

La herramienta proporciona archivos HTML que presentan la información relevante del software desarrollado en forma de repositorio.

6. Aspectos relevantes del desarrollo

Los aspectos relevantes del desarrollo del proyecto se fundamentan en la explicación del proceso de desarrollo del mismo desde un punto de vista de la ingeniería del software siguiendo las iteraciones escogidas para la gestión del mismo agrupándose en categorías comunes como son: el modelo de requisitos, el análisis y el diseño. A estas partes se le agregará siempre una parte final de implementación y testeo de los sistemas desarrollados.

Una vez finalizado, se debe gestionar una visión a alto nivel del sistema siguiendo la arquitectura planteada para todo el proceso cuyas partes habrá que profundizar para una explicación completa de los puntos donde se aplican las herramientas explicadas en puntos anteriores.

6.1. Comprensión del dominio

Es preciso comenzar con todos los datos referentes al dominio del problema para poder comprender la trama de datos a obtener y en qué puntos es necesario enfocarse para el correcto desarrollo del proyecto.

Para gestionar este primer punto se ha contado con la ayuda del departamento de electrónica del grupo de investigación BISITE de Salamanca por parte de Sergio Márquez Sánchez y Jorge Herrera Santos quienes ofrecieron la idea principal para la Universidad de Salamanca como un posible trabajo de fin de grado. Ellos han ofrecido al equipo de desarrollo la idea principal que tenían para poder gestionar esta chaqueta inteligente y se han ofrecido los medios materiales para ello como es el caso de un módulo de desarrollo ESP32 y un sensor BME680 para realizar el trabajo.

Por parte de la Universidad de Salamanca, se ha contado con la ayuda de la profesora Sara Rodríguez González perteneciente al departamento de ciencias de la computación de la Universidad de Salamanca y también miembro de grupo de investigación BISITE que ha ofrecido la visión de desarrollo del software donde se decide realizar una aplicación móvil.

Definiendo esto se genera una idea clara del objeto de estudio, así como su finalidad que se introducen a continuación.

6.1.1. Objeto de estudio

Se define que el objeto de estudio sea generar una trama de datos a partir de los sensores acoplados en el módulo ESP32 para posteriormente presentarlos por pantalla y gestionarlos vía servidor web.

6.1.2. Finalidad de estudio

La finalidad del estudio es generar un sistema capaz de la obtención de los datos, su transmisión y visibilidad en diferentes dispositivos por lo que se deberán realizar diferentes subsistemas que garanticen estos objetivos.

Al sistema no es necesario ofrecerle ningún tipo de dato, pero si es necesario comprender como los genera por parte de los sensores para garantizar un correcto funcionamiento a la hora del envío y registro.

Una vez planteados todos puntos se genera un posible diagrama de requisitos que servirá como apoyo para el desarrollo del proyecto el cual se muestra a continuación.

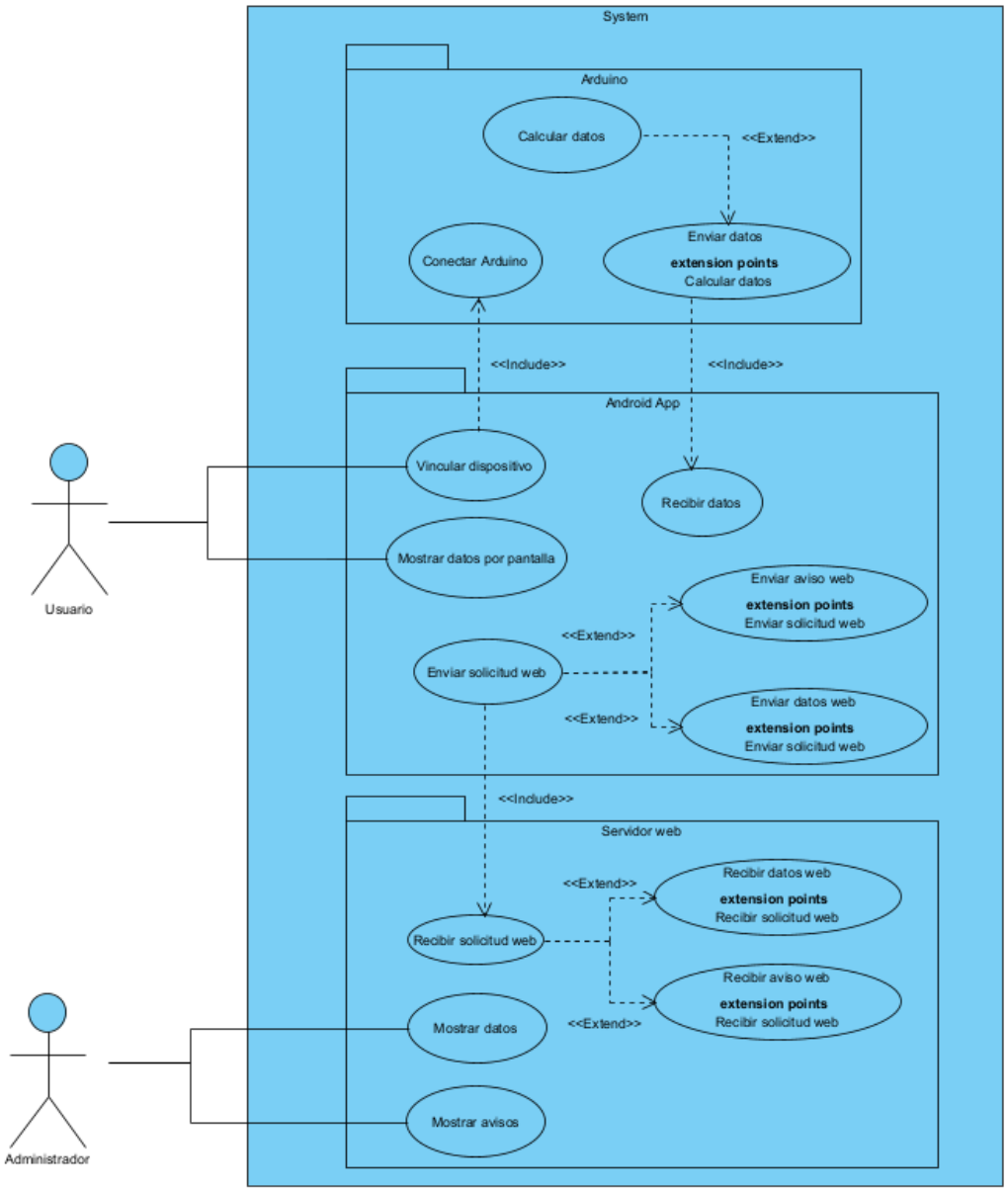


Ilustración 20: Diagrama de requisitos inicial

Una vez planteado de forma simple como se quieren desarrollar los puntos del sistema, hay que profundizar un poco en ello para poder garantizar todos los requisitos, tanto funcionales como no para albergar toda la funcionalidad del sistema.

Iterando sobre el diagrama inicial se busca ampliar el concepto de este llegando a un resultado cuarenta requisitos funcionales y cinco requisitos no funcionales los cuales se muestran en las ilustraciones diecisiete, dieciocho y diecinueve.

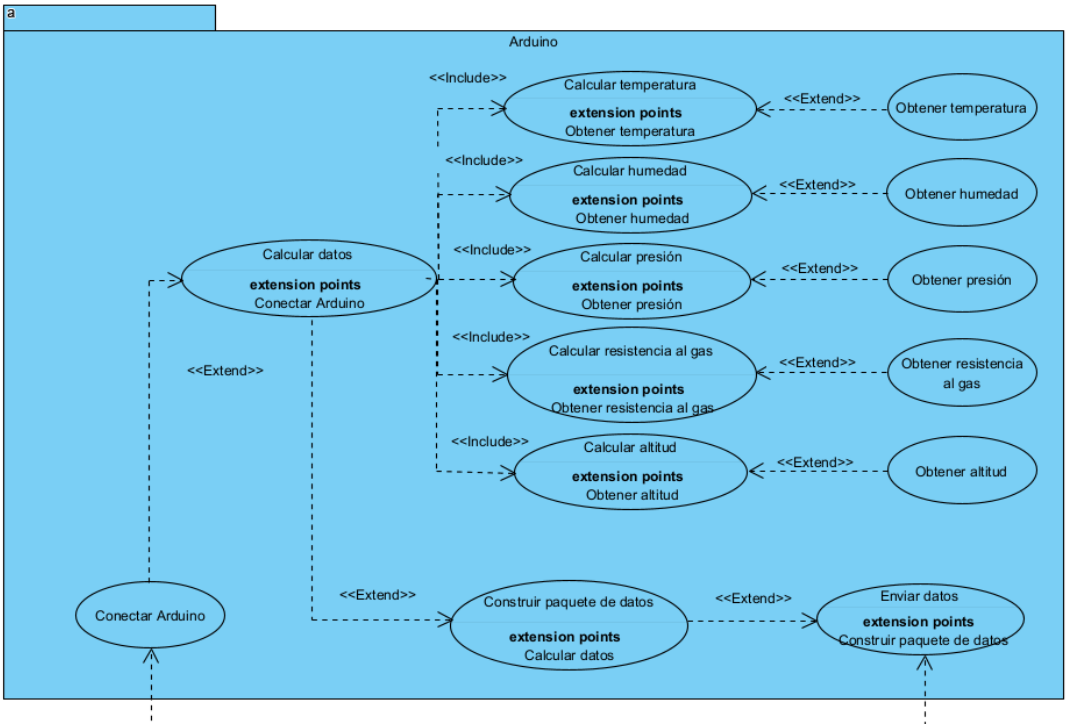


Ilustración 21: Diagrama de requisitos: Arduino

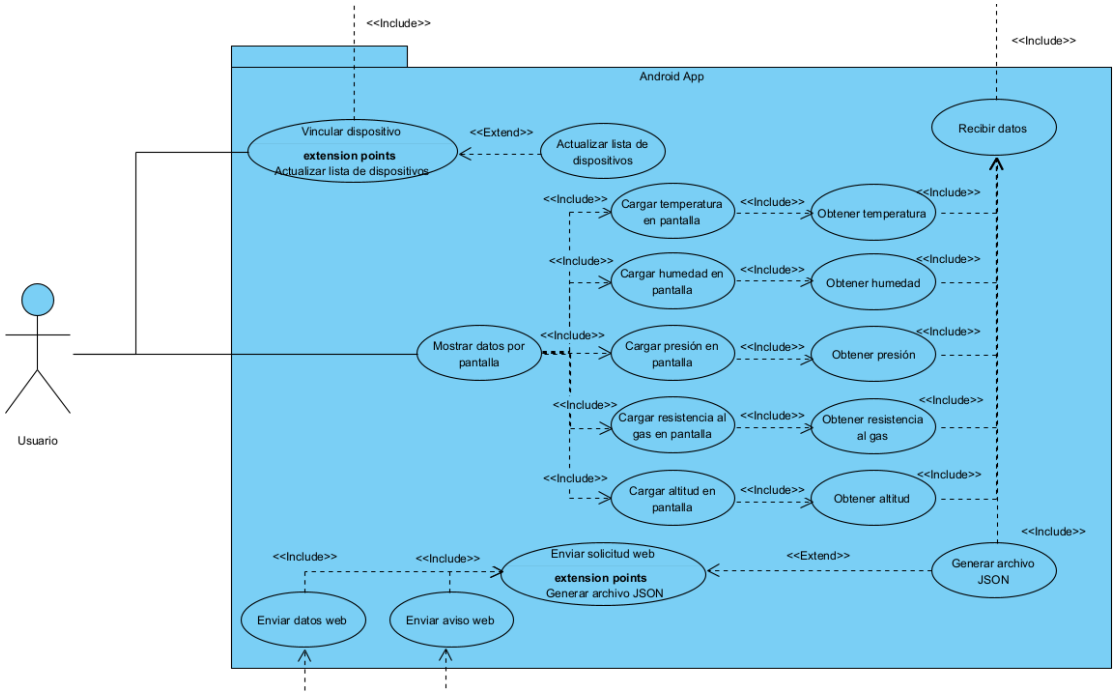


Ilustración 22: Diagrama de requisitos: Android app

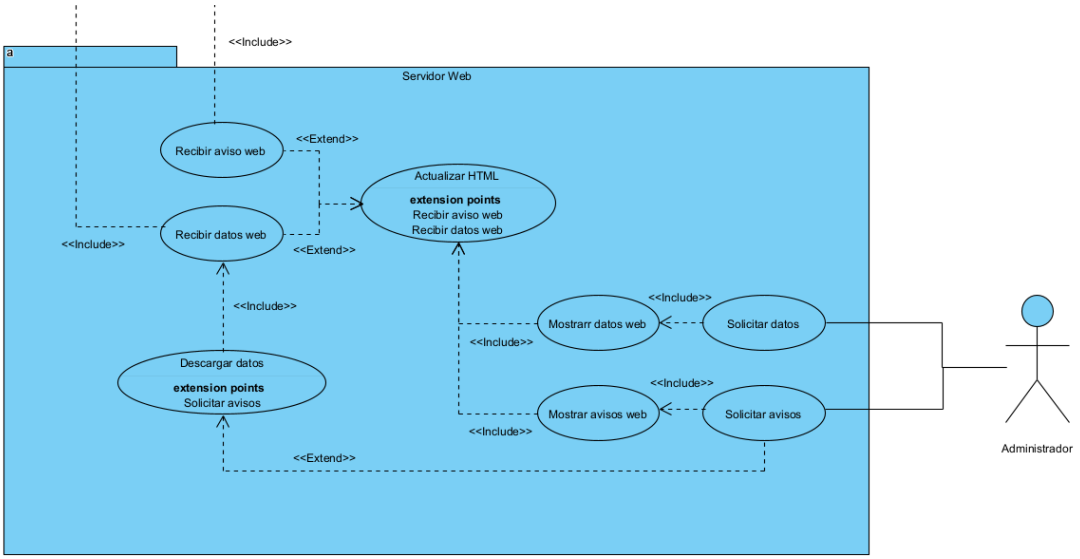


Ilustración 23: Diagrama de requisitos: Servidor web

6.2. Análisis y diseño

Continuando con la iteración de inicio, es momento de plantear un análisis del sistema, así como un diseño inicial que se irá modificando según se desarrollen los módulos de las siguientes iteraciones. Por ello, en este apartado se muestra solamente los diagramas referentes al diseño inicial y se adjunta al final del documento el Anexo III donde se muestra el desarrollo completo de todos los modelos de análisis y diseño.

Para el análisis y desarrollo se han planteado cuatro tipos de diseño: diseño de datos, diseño arquitectónico, diseño de interfaces y diseño procedimental.

En este punto solamente se explica el diseño de los datos ya que, para el diseño arquitectónico, dada su complejidad, se facilita un

6.2.1. Diseño de datos

El diseño de datos corresponde a todo lo referente sobre el modelo de dominio del proyecto donde se define como se recoge la información y el tratamiento de esta.

Dada la naturaleza y la velocidad de desarrollo del proyecto, no se ha seguido el método tradicional de desarrollo donde cabe destacar:

- La definición de los datos puede ser modificada a lo largo del proyecto en caso de que el prototipo funcional inicial sea modificado determinando así el método de obtención de datos y la manipulación de estos para su correcto entendimiento.
- Los tipos de datos planteados son susceptibles a cambios y puede que no sigan una dinámica fiel a la planteada debido al uso de los diferentes lenguajes de programación y su interpretación de los datos.

Tras plantear esto, se define el modelo de dominio inicial que se puede visualizar en la siguiente imagen:

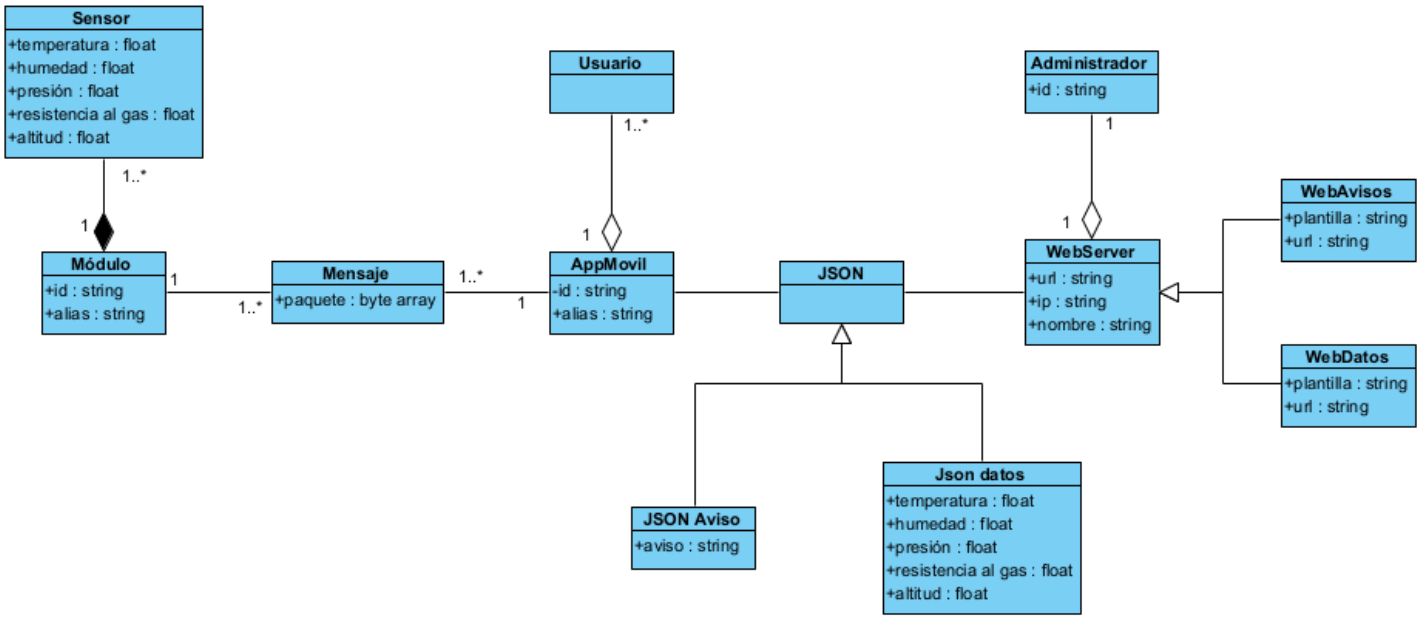


Ilustración 24: Modelo del dominio

Para lograr un mayor entendimiento del modelo del dominio, se adjunta la siguiente tabla que explica de forma breve las diferentes partes que lo conforman.

Clase	Descripción
Sensor	Representa el sensor conectado al módulo ESP32 que obtiene los diferentes valores necesarios para el funcionamiento del subsistema como son la temperatura, la humedad, la presión, la resistencia al gas y la altitud. No se esperan campos extra de los sensores
Módulo	Representa el módulo ESP32 del subsistema de obtención de datos que poseerá un identificador único para diferenciarse de otros dispositivos, así como un alias. El módulo incluye todos los datos del sensor para construir un mensaje con una estructuración predefinida.
Mensaje	Representa un array de datos cuyos campos corresponderán a todos los datos ofrecidos por el sensor mediante una estructura predefinida para facilitar la comprensión del resto de subsistemas y de posibles cambios en un futuro.
Temperatura	Representa el valor de la temperatura devuelto por el Sensor. Carece de identificación debido al cambio constante por su actualización.
Humedad	Representa el valor de la humedad devuelto por el Sensor. Carece de identificación debido al cambio constante por su actualización.
	Representa el valor de la presión devuelto por el Sensor.

Presión	Carece de identificación debido al cambio constante por su actualización.
ResistenciaAlGas	Representa el valor de la resistencia al gas devuelto por el Sensor. Carece de identificación debido al cambio constante por su actualización.
Altitud	Representa el valor de la altitud devuelto por el Sensor. Carece de identificación debido al cambio constante por su actualización.
AppMovil	Representa el subsistema generado por la aplicación móvil que posee un identificador único para determinar su funcionalidad dentro del servidor web, así como el mensaje recibido mediante conexión Bluetooth y el archivo JSON a enviar mediante conexión HTTP.
JSON	Representa el bloque formado por datos o avisos de la aplicación móvil en formato de archivo JSON.
Aviso	Representa un archivo de tipo JSON formado por un único campo de información respectiva a avisos.
Bloque	Representa un archivo de tipo JSON formado por todos los campos referentes al mensaje.
WebServer	Representa el servidor web donde se almacena toda la información referente a las vistas de las páginas web. En este servidor encontramos dos campos formados por la vista de los avisos y la vista de los datos.
Usuario	Representa al usuario que accede a la aplicación móvil. No posee ningún campo.
Administrador	Representa al usuario que accede al servidor web para controlar los datos.

Tabla 3: Descripción de las clases del modelo de análisis

6.2.2. Diseño arquitectónico

El diseño referente a la arquitectura del sistema incluye los apartados referentes a los elementos y mecanismos principales del proyecto visualizados desde un aspecto a alto nivel.

El sistema se agrupa en tres sistemas principales y tres auxiliares que generan el tránsito de datos de manera correcta los cuales son:

- "arduino": responsable de la obtención de los datos de los diferentes sensores, así como el encargado de generar el paquete de datos que requiere el sistema para tratar los datos obtenidos.
- "androidApp": subsistema encargado de la recepción y tratamiento de datos mediante bluetooth que genera un primer método visual para el usuario.
- "serverWeb": responsable de la correcta visualización de todos los datos provenientes de las diferentes chaquetes con el objetivo de monitorizar el sistema y garantizar los objetivos del proyecto como la seguridad y la inmediatez.

Los sistemas auxiliares son:

- "bluetooth": referente al canal de transmisión fundamentado en tecnología bluetooth por donde se transmite la trama de datos desde el módulo ESP32 al dispositivo móvil al que esté vinculado.
- "json": referente al canal de transmisión que utiliza solicitudes HTTP desde la aplicación móvil al servidor web establecido para la recepción de los datos mediante archivos JSON.
- "usuarios": contiene el aspecto relacionado con la interacción con los usuarios donde se diferencia entre el usuario que utiliza la aplicación móvil con el objetivo de visualizar los datos y un usuario administrador capacitado con el acceso al servidor web para poder monitorizar las distintas chaquetas que se encuentren conectadas.

Dada esta organización, se plantea la siguiente arquitectura del sistema:

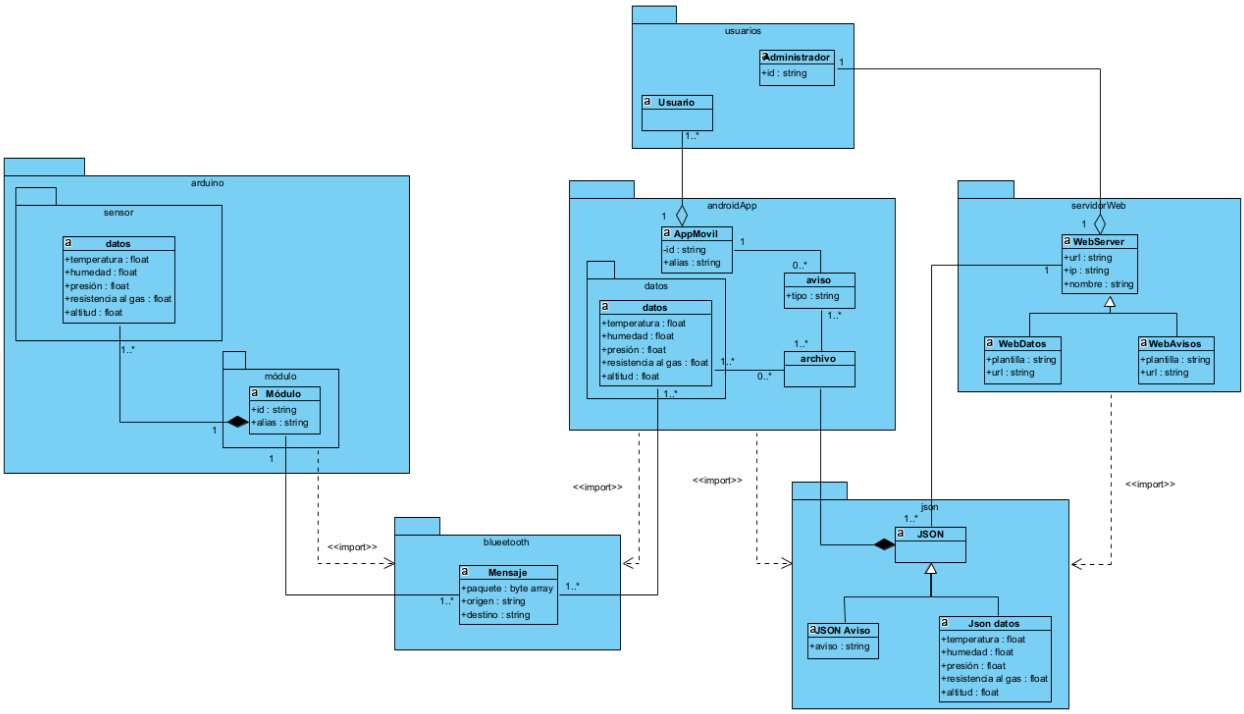


Ilustración 25: Diseño arquitectónico

Para observar con mayor detalle cómo se conforma la arquitectura del sistema, se recomienda consultar el Anexo III, Diagrama de clases de diseño donde se amplía la información referente al diseño arquitectónico que presenta el proyecto.

6.2.2.1. ***Arquitectura ETL***

El fundamento teórico en el que se basa el sistema es el modelo ETL, del inglés *Extract Transfrom Load,* la cual se fundamenta en el tratamiento y gestión de los datos que se emplea en este sistema debido a la gran opción de obtención de datos debido a los distintos sensores, las diferentes forma de transformarlos para el correcto entendimiento del sistema y la carga de estos en pantalla para su correcta visualización.

Los modelos basados en ETL requieren una correcta estructuración de los datos y se suelen aplicar en proyectos correctamente estructurados y asentados por lo que, en este, dado que los casos de estudio pueden estar sujetos a cambios, no se podrá seguir completamente. Este punto puede entrar en conflicto directo con el sistema y los lenguajes de programación que implementa ya que, por ejemplo, Java se fundamenta en establecer el esquema de datos previamente frente a la funcionalidad del sistema lo que puede afectar a la escalabilidad del proyecto.

Otro problema que aparece a la hora de aplicar estas estructuras se da en que este proyecto no se fundamenta en tres etapas de desarrollo como se sugiere, sino que se sigue el método del proceso unificado donde se presentan un total de cinco iteraciones sujetas a sus respectivas modificaciones por lo que aparecen distintas dificultades:

- Puede aparecer el conocido efecto de "cuello de botella" en aspectos de desarrollo de los subsistemas donde surjan más problemáticas a la hora de implementar correctamente todos los módulos.
- La escalabilidad del sistema puede verse comprometida debido a la fluctuación de los tipos de sensores o conexiones al servidor web por lo que se debería garantizar una forma de arbitrar estas tareas.
- Debido al punto anterior la carga que puede presentar el sistema también se ve afectada ya que habrá puntos en los que sea necesario gestionar una gran cantidad de solicitudes de diferentes datos.

Las arquitecturas ETL tradicionales utilizan el framework de Hadoop con el objetivo de la programación distribuida de las actividades que dificulta la escalabilidad del sistema ya que recae en la mano del desarrollador conocer las diferentes limitaciones de las herramientas de desarrollo, así como del código que se implementa donde debería de poder de garantizarse de forma automática.

Para evitar esto, es necesario determinar un flujo de datos que garantice la correcta comunicación del sistema obteniendo diferentes propiedades beneficiosas:

- Alta cohesión: cada módulo tiene un objetivo claro determinado garantizando que se puedan reutilizar en otro proyectos y validarlos ante fallos.
- Acoplamiento mínimo: todo módulo presente en el sistema garantiza su independencia respecto a los otros por lo que no son dependientes de los otros.
- Mantenimiento simple: es necesario garantizar que, ante los cambios de los módulos, se pueda actualizar el sistema de una forma simple.
- Paralelismo: se debe garantizar que todo el sistema permita acceder a varios módulos a este y que se garantice el correcto funcionamiento.

6.2.2.2. Arquitectura MVC y frontend

Para determinar la funcionalidad de los sistemas de proyecto, se plantea una arquitectura fundamentada en el modelo vista controlador o comúnmente conocida como MVC la cual se puede subdividir en dos apartados principales relevantes a la interfaz:

- Una aplicación móvil desarrollada en Android Studio mediante lenguaje Java la cual interacciona con el usuario final en forma de vista recibiendo datos por parte del dispositivo o chaqueta y finalmente mandando esos datos al servidor web.
- Un servidor web basado en Node.js mediante lenguaje JavaScript que interacciona de forma de vista con el usuario final con el objetivo de monitorización de los datos.

6.3. Módulo para la gestión del ESP32

Este módulo se desarrolla gracias al uso de la plataforma del IDE de Arduino mediante lenguaje C++ cuya función principal es la obtención de los datos de los diferentes sensores que tenga conectados para su posterior envío mediante conexión Bluetooth.

Esta aplicación posee un único endpoint que será el envío del paquete con los datos recibidos y gestionados por el módulo ESP32 de forma cíclica y actúa de una forma completamente estática que podrá modificarse dependiendo de los sensores que se encuentren en el sistema.

El procedimiento del módulo es el más sencillo de comprender y se demuestra en la ilustración siguiente donde se puede observar la obtención de datos con el prototipo desarrollado mediante el uso de un sensor BME680 donde se le solicita al sensor la obtención de cada uno de sus parámetros.

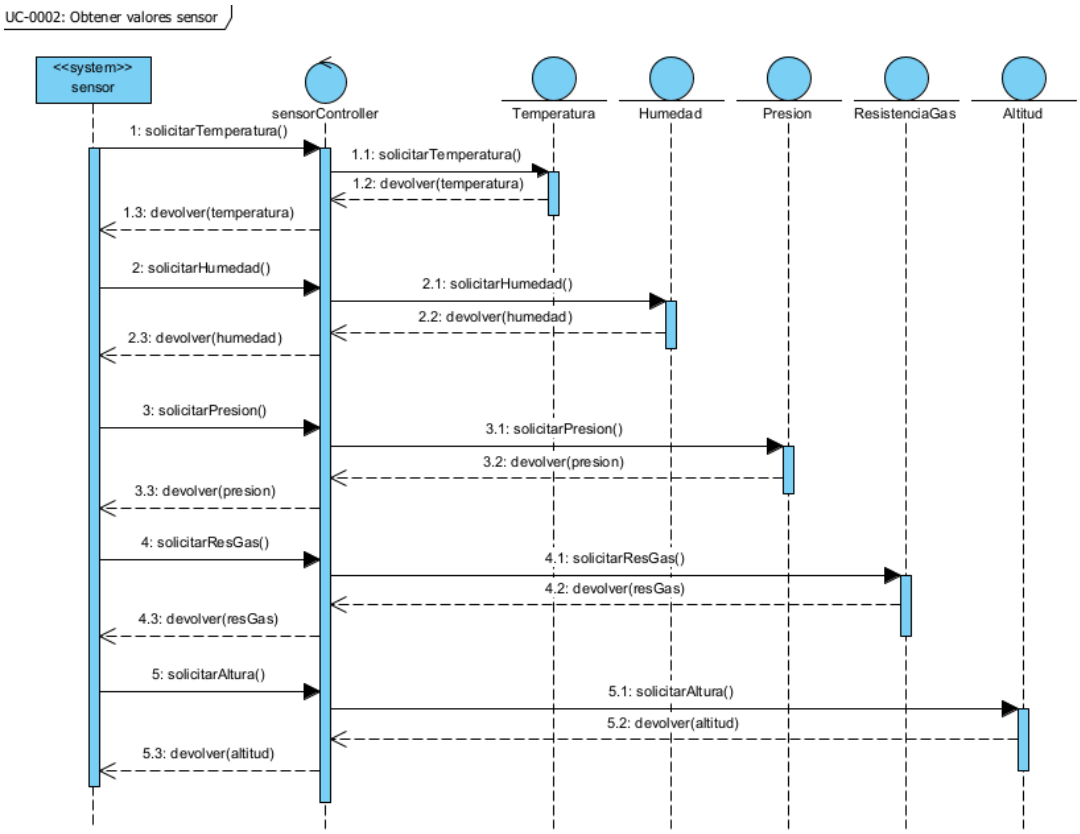


Ilustración 26: Obtener valores de sensor

6.4. Módulo de la aplicación móvil

La aplicación móvil como ya se ha indicado, se desarrolla en la plataforma de desarrollo de Android Studio mediante lenguaje Java que se fundamenta en la visión de los diferentes datos recibidos por Bluetooth.

Este módulo se fundamenta en dos apartados de trata de datos las cuales son las siguientes:

- Recepción mediante conectividad Bluetooth del paquete de datos enviado por el módulo ESP32.
- Envío de datos mediante protocolo HTTP al servidor web.

Por otro lado, los datos se visualizan en la pantalla del dispositivo móvil donde el usuario puede comprobar los parámetros ambientales del entorno.

Para la gestión de los datos es necesario conocer el tamaño del paquete que se establece para el envío por parte del módulo anterior para garantizar una correcta recepción y descomponerlo en las diferentes variables del sistema para mostrarlas por pantalla. En el caso del prototipo se trata de un array de veintiún caracteres donde:

- Del primero al cuarto pertenecen al valor de la temperatura.
- Del quinto al noveno pertenecen al valor de la presión.
- Del décimo al decimotercero pertenecen al valor de la humedad.
- Del decimocuarto al decimoctavo pertenecen al valor de la resistencia al gas.
- Del decimonoveno a vigésimo primero pertenecen al valor de la altitud.

6.4.1. Interfaz de la aplicación móvil

Una vez que se logra descomponer los datos del mensaje recibido, se presentan por pantalla de la siguiente forma:

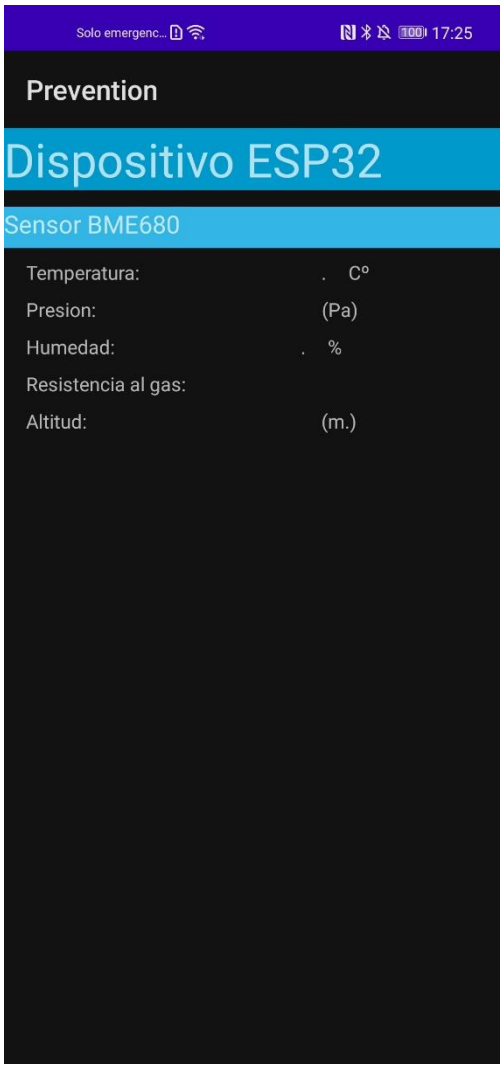


Ilustración 27: Aplicación móvil: Pantalla principal

Esta pantalla se irá actualizando de forma periódica y completamente automática con los valores recibidos.

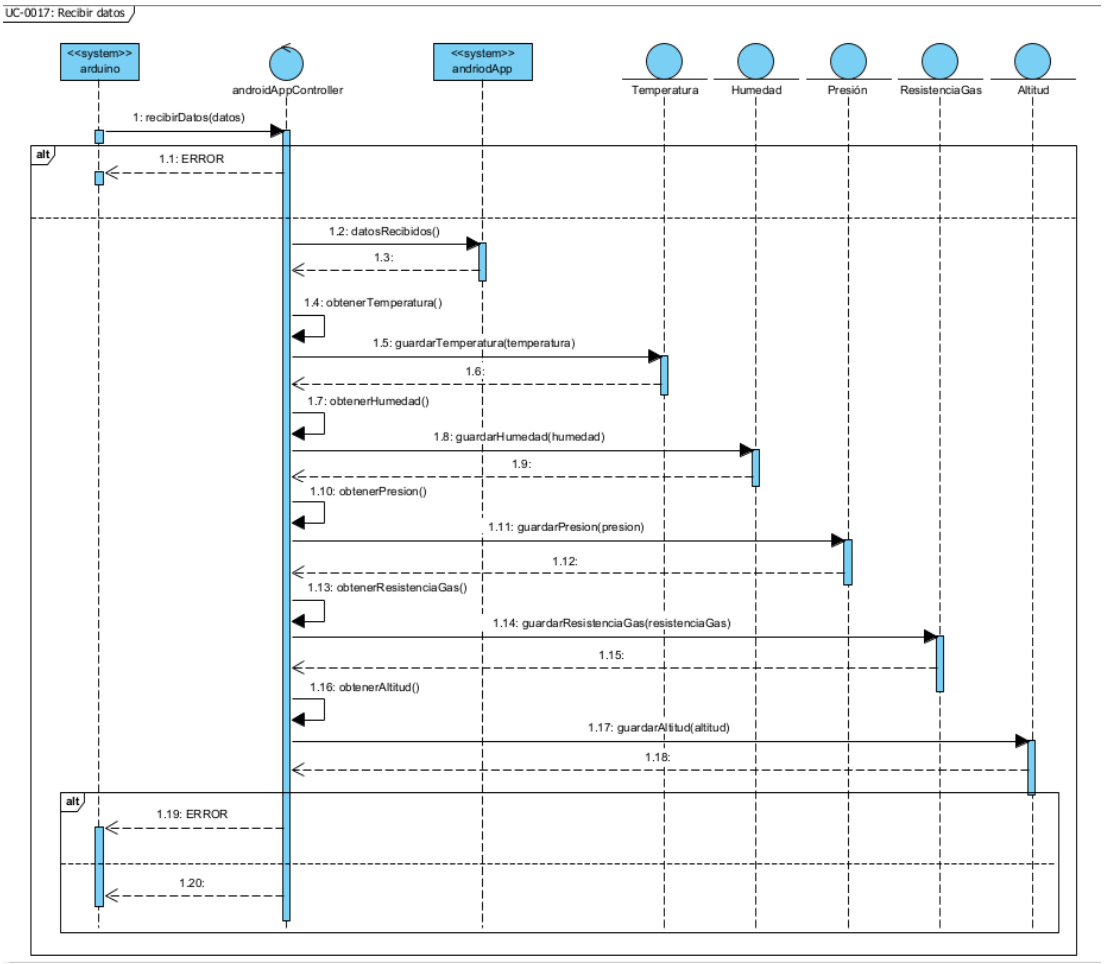


Ilustración 28: Recepción de datos en aplicación móvil

Al construir las variables, estas se deben guardar para formar los elementos necesarios para la gestión de los archivos JSON para enviarlos al servidor web. Para ello, se utiliza la biblioteca de Google denominada "Gson" que facilita la creación de este tipo de archivos de manera completamente automática.

Es necesario comparar entre dos tipos de archivos JSON formulados:

- JSON de datos: son todos los archivos JSON que contienen los datos recibidos por Bluetooth y tratados por la aplicación de forma tal que se garantice la integridad a la hora de construir los archivos.
- JSON de avisos: todos los archivos JSON que se generan a partir de eventos programados según datos de referencia que supongan un problema para el usuario que utiliza la chaqueta.

Dependiendo del tipo de archivo se enviará a un punto a otro del servidor web garantizando así que el tratamiento de los datos se garantiza de forma correcta.

La realización de obtención de archivos JSON se muestra en la ilustración 29.

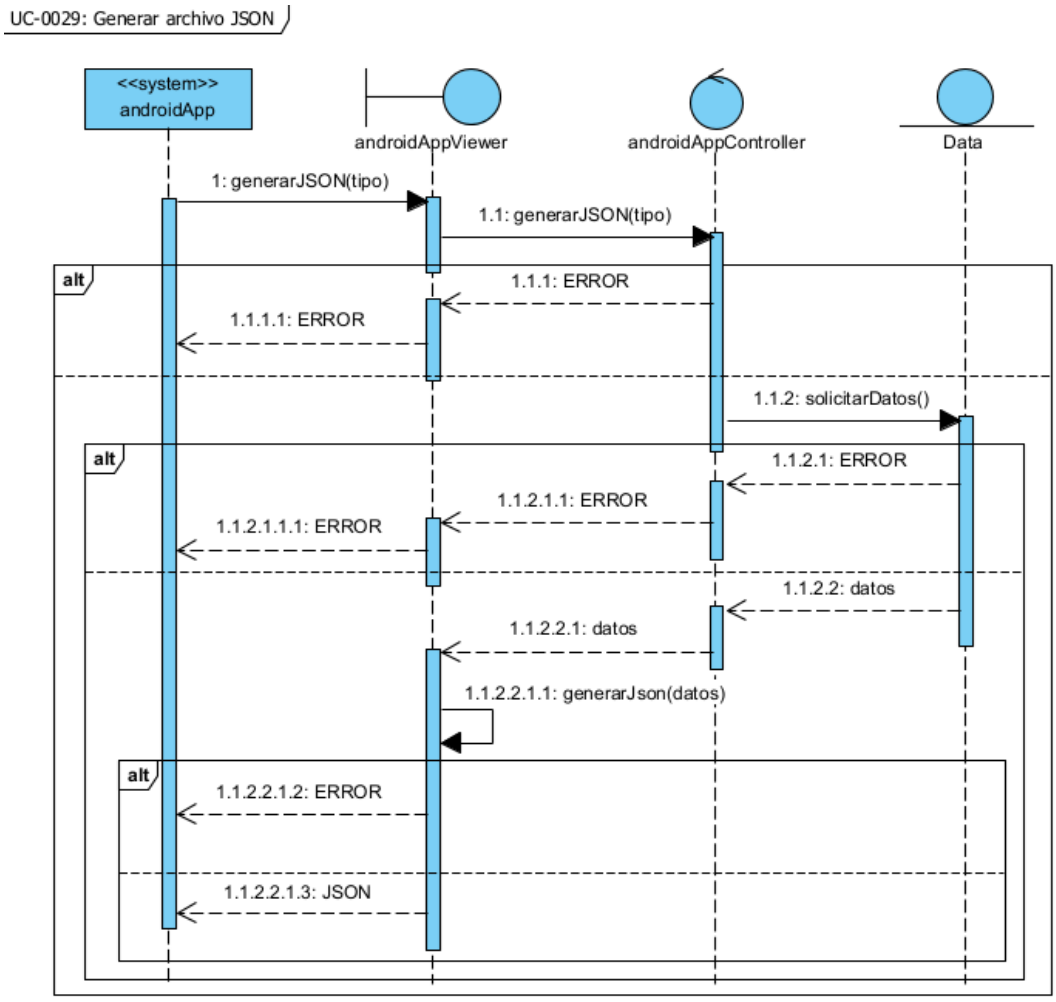


Ilustración 29: Generar archivo JSON

6.5. Módulo del servidor web

El servidor web se diseña bajo la herramienta de Node.js mediante lenguaje JavaScript cuyo objetivo principal será la recepción de los datos de los distintos dispositivos móviles mediante conectividad HTTP.

Para ello, se configuran dos páginas de recepción y muestra de datos que atienden peticiones de tipo GET y POST. Las peticiones GET son gestionadas por los usuarios que quieren visualizar las páginas web del servidor y, por otro lado, las solicitudes POST serán realizadas por los dispositivos móviles que mandan datos y avisos de forma periódica.

Las solicitudes POST se muestra en la siguiente ilustración.

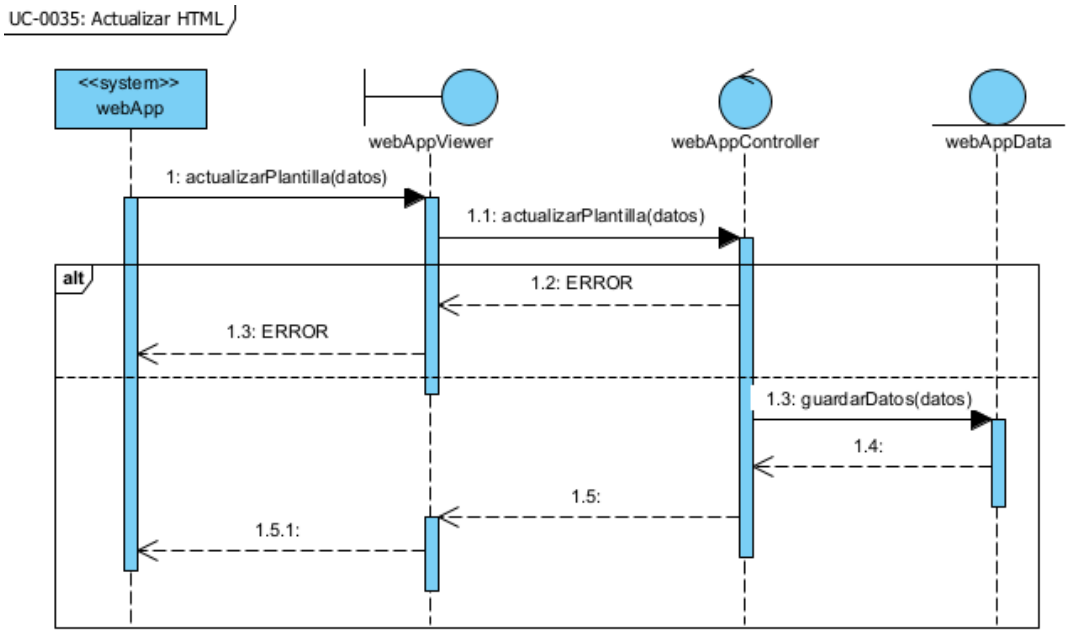


Ilustración 30: Actualizar HTML

6.5.1. Interfaz del servidor web

Las páginas web obtenidas se muestran en las ilustraciones 30 y 31.

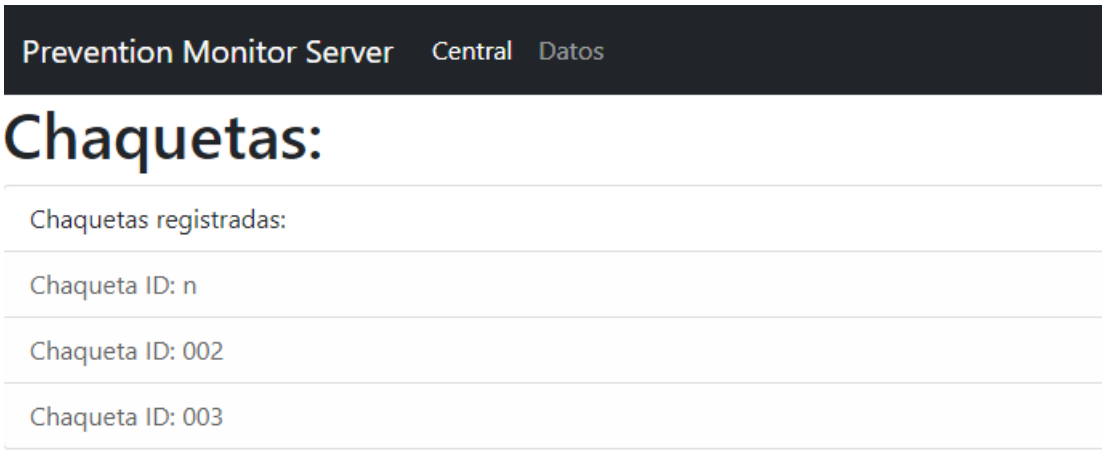


Ilustración 31: Página web de monitorización

Prevention Monitor Server Central Datos

#	Temperatura	Humedad
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0

Ilustración 32: Página web de datos

A estas dos páginas de funcionalidad principal se añade una página de soporte para la gestión del error 404 para lograr encerrar al usuario en caso de mal uso del servidor y se utiliza la herramienta de BootStrap que ofrece diferentes plantillas para el desarrollo de archivos HTML completamente configurables.

En el caso de la página 404 del servidor se utiliza una plantilla animada para HTML de la página “Free Frontend” la cual se presenta en la imagen 33.

Ilustración 33: Página web 404

Por último, se debe generar un sistema de descarga de los datos que se han ido recibiendo en el sistema el cual, para el prototipo funcional, solo se habilitará para una de las chaquetas, pero podrá ser ampliado en el caso de generar más dispositivos conectados. El sistema de descarga se explica en la ilustración 29.

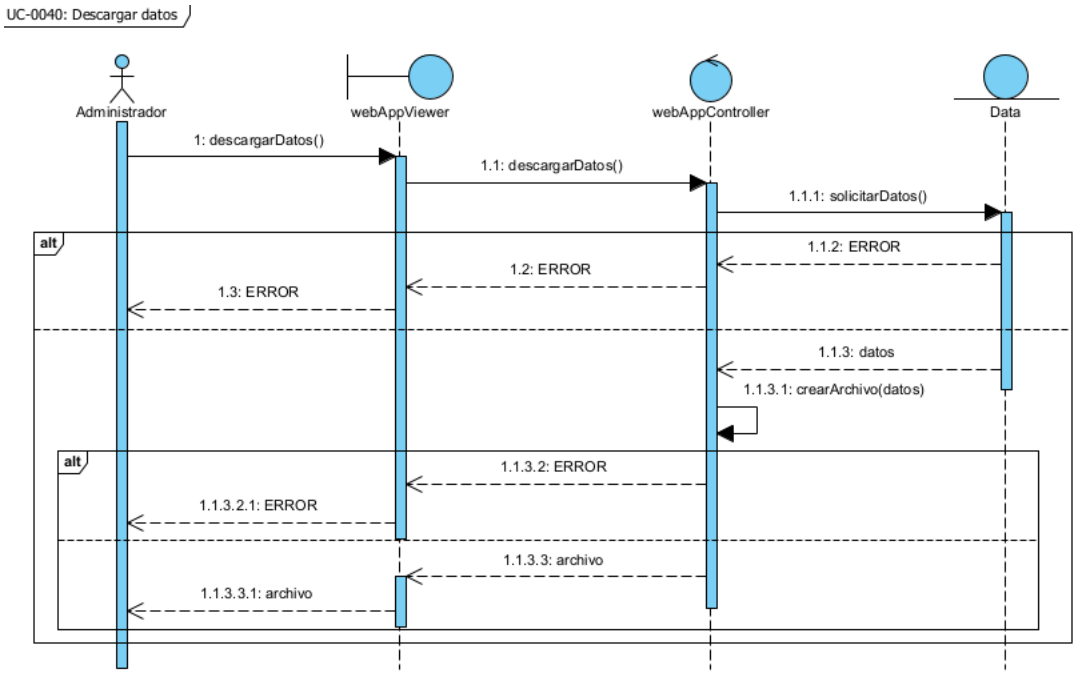


Ilustración 34: Descargar datos

6.6. Aspectos relevantes al paquete de datos

Durante el desarrollo del prototipo se ha enfocado el tratamiento de los datos de tal forma que se garantizase la integridad de aquellos que ofrece el sensor BME680: temperatura, presión, humedad, resistencia al gas y altitud. Por ello, en la funcionalidad entregada se realiza un tratamiento de un array de un total de 21 caracteres referentes a todos los campos y la gestión de archivos JSON de tal estructura.

A la hora de ampliar el número de sensores y por ende, la cantidad de campos tanto del paquete enviado como los campos de los archivos JSON deben ser ampliados también.

Esto es preciso conocerlo ya que se ofrece un prototipo como base funcional que se pueda modificar dependiendo de la naturaleza de las actividades a las que se enfoque el proyecto donde convenga utilizar un tipo de sensor u otro.

Para garantizar la funcionalidad, habría que:

1. Determinar cuántos caracteres son necesarios guardar en el array para garantiza el envío por Bluetooth sin su pérdida.
2. Determinar cuántos campos variables se utilizan en la aplicación móvil para crear los archivos JSON y conocer sus campos para su tratamiento en el servidor web.

6.7. Aspectos relevantes a la seguridad

El único punto vulnerable respecto a la seguridad es la conexión entre el dispositivo móvil y el servidor web ya que no se precisa ningún tipo de confirmación de conectividad al tratarse de envíos a una dirección web.

Para este proyecto y a nivel conceptual se ha determinado que al tratarse de servidores que se encuentran en redes cerradas sin acceso a internet se puede evitar así el acceso de terceras personas que puedan corromper u obtener los datos referentes al sistema.

Por otro lado, el acceso a la aplicación móvil debe ser limitante a dispositivos móviles que pertenezcan a la empresa que desarrolle la actividad ya que el envío al servidor web se gestiona de manera automática por lo que se podría saturar el sistema desde este apartado si no se tiene en cuenta.

6.8. Aspectos relativos al despliegue

Al tratarse de módulos independientes que no están sujetos a la funcionalidad del resto, la integridad de los subsistemas no se ve comprometida si el resto no funciona, aunque el objetivo final del proyecto no se cumpliría.

Aun así, existe un orden claro que precisa el sistema para garantizar la integridad de los datos el cual es:

1. Iniciar el servidor web para gestionar la recepción de todos los datos recibidos de los diferentes dispositivos móviles.
2. Iniciar el módulo ESP32 para comenzar la recepción de los datos de los sensores conectados.
3. Iniciar la aplicación móvil que será necesario vincular al módulo ESP32 deseado que posteriormente enviará los datos de forma automática al servidor web.

Por todo esto, se realiza un diagrama de despliegue referente a todos estos puntos el cual se puede observar con más detalle en el anexo III en el apartado de “Plan de desarrollo” y en la ilustración 34.

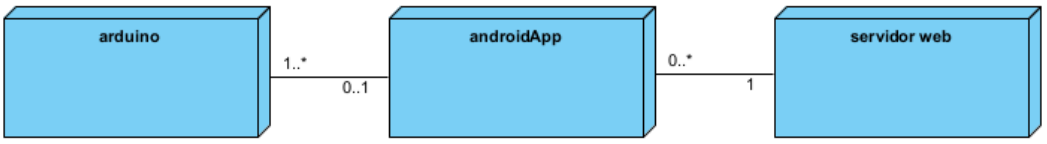


Ilustración 35: Diagrama de despliegue

Dentro de estos paquetes se puede observar con más detalle los puntos de despliegue en las tres ilustraciones siguientes.

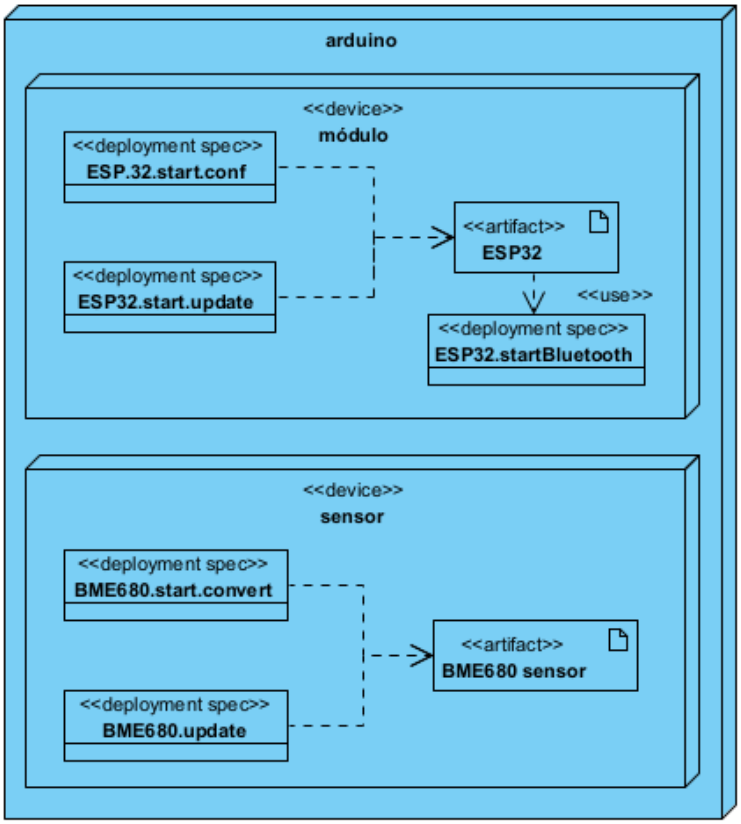


Ilustración 36: Diagrama de despliegue: Módulo arduino

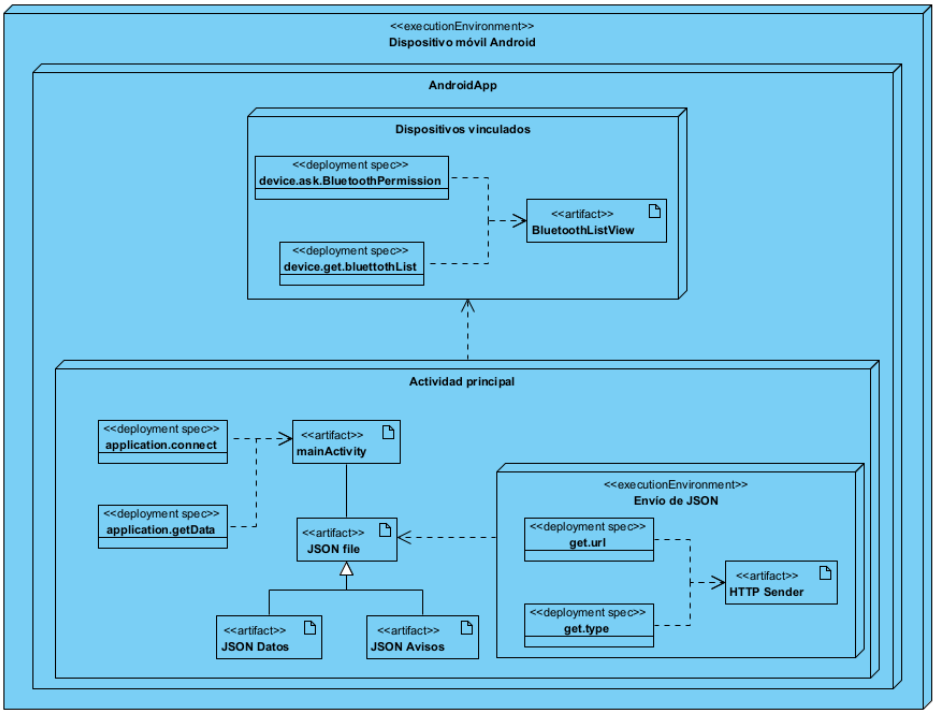


Ilustración 37: Diagrama de despliegue: Módulo de la aplicación móvil

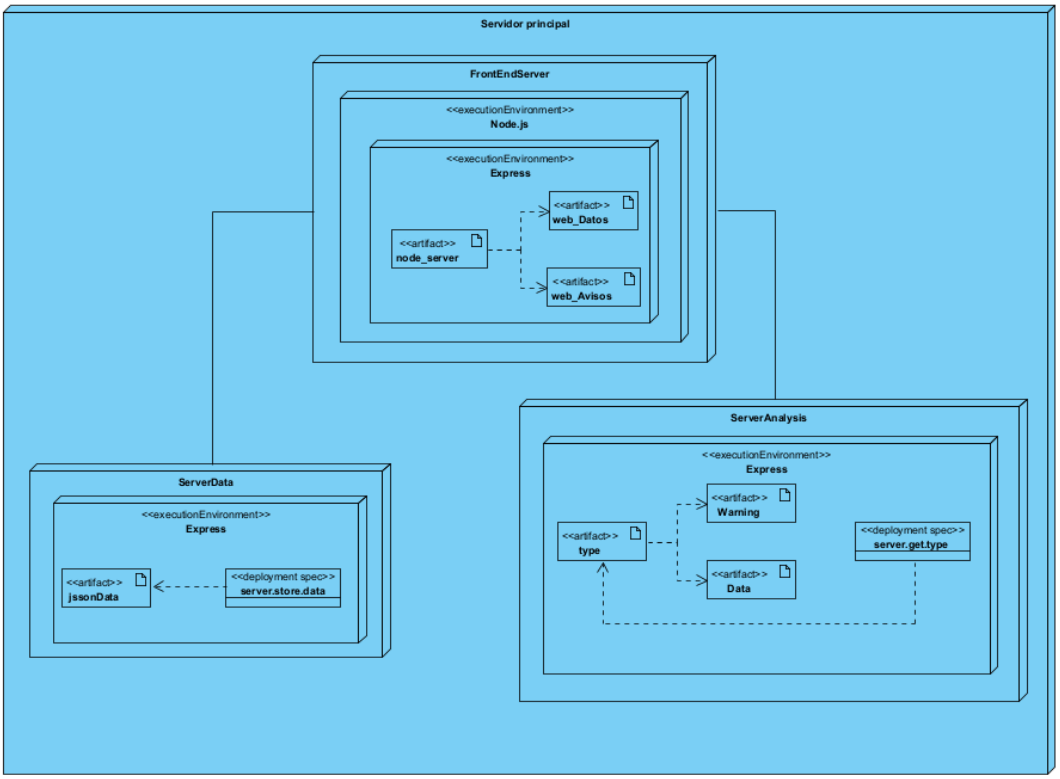


Ilustración 38: Diagrama de despliegue: Módulo del servidor web

7. Resultados y conclusiones

Tras el transcurso del desarrollo del proyecto y siguiendo la planificación pactada al comienzo de este se puede concluir que se han logrado los objetivos de desarrollo principales del trabajo que son la obtención y tratamiento de datos del medio y notificación de estos.

Para gestionar estos objetivos, el proyecto se desglosa en los tres sistemas requeridos para eelo los cuales se han dividido en el software para la obtención de datos en arduino, la aplicación mócil que los vuelca por pantalla y finalmente el servidor web de monitorización de estos. Para ello, se ha tenido siempre en cuenta el objetivo fundamental de buscar la inmediatez del sistema para conseguir la prevención de riesgos y por otro lado, la sensillez para facilitar al usuario el entendimiento de los datos.

Como resultado se obtiene un sistema capaz de:

- Gestionar la obtención de datos mediante el módulo ESP32.
- Utilizar la conexión por Bluetooth para un paquete concreto de datos.
- Representar los datos recibidos por pantalla.
- Gestionar el envío y la recepción de datos mediante HTTP en formato JSON.
- Visualizar el estado de los dispositivos.
- Descargar los datos recolectados.

El resultado del proyecto concluye con la obtención de los tres sistemas introducidos garantizando la obtención de los datos por parte de los sensores y correcto envío y tratamiento para la visualización pior parte de la aplicación móvil y el servidor web.

En el caso específico de este sistema se consigue desarrollar y realizar pruebas de una aplicación móvil conectada a una chaqueta la cual será la única que se comunique con el servidor web pero se habilita la capacidad de procesamiento para variois dispositivos móviles con el servidor web. Por ello, se da por conseguido el objetivo planteado por el proyecto de volcado y obtención de datos planteándolo como una base teórica para futuros proyectos de dispositivos werables efocados en la prevención de riesgos laborales.

Como conclusión, la gestión y puesta a punto del proyecto me ha servido para afianzar todos los conocimientos obtenidos a lo largo de estos cuatro años de grado de ingeniería informática en la Universidad de Salamanca a nivel de ingeniería, gestión de proyectos junto con mis habilidades de programador y diseñador generando nuevos intereses y objetivos que conseguir en el futuro. Sobre todo, uno de los aspectos conseguidos más importantes es la confianza adquirida y la satisfacción de conseguir desarrollar un proyecto desde cero.

7.1. Aplicaciones comerciales

Dentro del alcance del proyecto se debe plantear a que nicho de mercado se puede aplicar el prototipo final obtenido.

Debido a las características que ofrece el sensor BME680 de obtención de parámetros ambientales como la temperatura, la presión, la humedad, la resistencia al gas y la altitud, se puede enfocar a un número de clientela muy amplio desde empresas que precisen determinar la presencia de temperaturas extremas como determinar los gases existentes en el entorno.

Un ejemplo clave y de actualidad durante el año 2021 ha sido la determinación de ambientes de COVID-19 en espacios cerrados que, debido al escaso desarrollo de sensores de presencia

del virus y de su excesivo precio, se han desarrollado planes alternos y aplicables en el ámbito laboral. Para ello, se determina la presencia de dióxido de carbono en sala como método de prevención de riesgos garantizando que, en un lugar cerrado ventilado, se reduce la cantidad de dióxido de carbono y por ende, de presencia de COVID-19 por lo que se le otorga al prototipo una funcionalidad clave para el uso en actividades centradas en solucionar este tipo de problema ya sea en ámbitos de sanidad como en todos aquellos que requieren un control exhaustivo de ello.

Dada la versatilidad de los sensores del mercado, el proyecto se puede adaptar a cualquier actividad que requiera un mínimo de prevención para asegurar la seguridad de los empleados. Por tanto, se determinan posibles proyectos futuros en el siguiente apartado.

7.2. Proyectos futuros

Este trabajo de fin de grado de ingeniería informática por parte de la Universidad de Salamanca se centra en realizar un prototipo funcional de un sistema de monitorización de parámetros ambientales con el objetivo de monitorizarlos y asegurar la seguridad mediante la prevención.

Por ello, se plantea como una base teórica abierta a diferentes proyectos que puedan necesitar una estructura fundamentada en este trabajo e implantar mejoras para conseguir mayor estabilidad o adaptabilidad a la actividad empresarial en la que se centre. Esto se logra realizando un estudio previo de los factores de riesgo y que sensores se adaptarían mejor para la obtención de los datos. Cualquier modificación de los sensores debe ser comprobada a la hora de programación para actualizar las vistas tanto de las aplicaciones móviles como del servidor web y actualizar el tamaño de la trama de datos que se envían

Otro de los puntos clave para mejorar sería mejorar el servidor web para asegurar todas las comunicaciones de las aplicaciones móviles para evitar cuellos de botella, así como pérdida de datos ya que, a la hora de escalar a más aplicaciones se deben comprobar la correcta recepción de los datos.

A nivel de interfaz gráfica sería interesante poder añadir elementos visuales que faciliten la organización tanto de la aplicación como del servidor web. Por un lado, si se aumentase el número de sensores conectados, se podría agregar un menú lateral que los recoja para acceder a los datos interesados, así como una opción para visualizar la lista y conectarse a otros dispositivos. Por el lado del servidor una mejora interesante sería mostrar las chaquetas conectadas para poder seleccionar que datos ver por pantalla, así como establecer una lógica de estos garantizando un orden además de poder descargar aquellos datos que se desee ya que actualmente genera un documento con todos ellos.

Finalmente, y como apartado de mejora final, habría que realizar una mejora en el sistema de alertas introduciendo algún tipo de aviso sonoro y que genere un mayor impacto en el usuario o que incluso se gestionase de manera automática activando el protocolo de seguridad establecido por la empresa.

Todos estos cambios y mejoras otorgan un gran abanico de posibilidades al proyecto para satisfacer las necesidades de futuros proyectos fundamentados en la prevención de riesgos y en actividades como la minería o cuerpos de seguridad como bomberos o policía.

7.3. Agradecimientos

Mi especial agradecimiento al equipo de investigación del BISITE de Salamanca y en especial a Sergio Márquez Sánchez, Jorge Herrera Santos y Sergio Alonso Rollán quienes estuvieron siempre a mi disposición en caso de necesidad. Por otro lado, agradecer a Sara Rodríguez González por parte de la Universidad de Salamanca quien ha estado siempre a mi disposición para gestionar los documentos del proyecto y dudas de nivel informático.

Por otro lado, dar las gracias a Héctor Acosta García por la aportación del dispositivo de desarrollo para la aplicación móvil sin el cual se habría dificultado la programación del software que, a pesar de que existen emuladores, se obtiene una respuesta mucho más clara y concisa de la interacción entre el usuario y el sistema.

Finalmente, siempre agradeceré a mi familia por estar ahí y ayudarme en el ámbito de desarrollo y fijación de la interfaz los cuales han sido los usuarios de evaluación para lograr los conceptos.

8. Bibliografía

Android. (1 de Mayo de 2012). *Philosophy and Goals*. Obtenido de https://web.archive.org/web/20120501080416/http://source.android.com/about/philosophy.html

Android Developers. (27 de Diciembre de 2019). *Introducción general a Bluetooth*. Obtenido de https://developer.android.com/guide/topics/connectivity/bluetooth

Android Developers. (3 de Junio de 2020). *Descripción general de Volley*. Obtenido de https://developer.android.com/training/volley?hl=es

Android Developers. (6 de Junio de 2020). *Paquetes de bibliotecas de compatibilidad*. Obtenido de https://developer.android.com/topic/libraries/support-library/packages?hl=es

Android Developers. (9 de Junio de 2021). *Android.os*. Obtenido de https://developer.android.com/reference/android/os/package-summary

Android Developers. (21 de Abril de 2021). *Android.widget*. Obtenido de https://developer.android.com/reference/android/widget/package-summary

APR. (2006). *¿Qué es y para qué sirve HTML? El lenguaje más importante para crear páginas webs.* Obtenido de https://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=435:ique-es-y-para-que-sirve-html-el-lenguaje-mas-importante-para-crear-paginas-webs-html-tags-cu00704b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192

Arduino. (24 de Diciembre de 2019). *SPI Library*. Obtenido de https://www.arduino.cc/en/reference/SPI

Arduino. (12 de Diciembre de 2019). *Wire Library*. Obtenido de https://www.arduino.cc/en/reference/wire

AZ Delivery. (s.f.). *AZ Delivery: ESP32 description*. Obtenido de https://www.az-delivery.de/es/products/esp32-developmentboard#description

Banzi, M., & Shiloh, M. (2014). *Getting started with Arduino.* Sebastopol.

Basque Research. (28 de Junio de 2013). *Un traje inteligente incrementa la seguridad de los bomberos*. Obtenido de https://www.agenciasinc.es/Noticias/Un-traje-inteligente-incrementa-la-seguridad-de-los-bomberos

Bergmmanm, R., & Davidson, J. (11 de Abril de 2018). *AuthFailureError*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/AuthFailureError.java

Byous, J. (1998). *Java Technology: The early years.* Sun Developer Network.

C Plus Plus. (2000). Obtenido de https://www.cplusplus.com/

Callejo, P. (20 de Noviembre de 2019). *Tecnalia Research & Innovation*. Obtenido de https://es.slideshare.net/TECNALIA/wearables-para-la-salud-y-prevencin-de-riesgos-laborales-195541159

Capan, T. (s.f.). *¿Por qué demonios usaría Node.js? Un tutorial caso por caso*. Obtenido de https://www.toptal.com/nodejs/por-que-demonios-usaria-node-js-un-tutorial-caso-por-caso

Davidson, J. (11 de Abril de 2018). *Volley Error*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/VolleyError.java

Davidson, J., & Eum, J. (27 de Marzo de 2019). *Volley String Request*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/toolbox/StringRequest.java

Davidson, J., Hager, U., Eum, J., Smith, P., & Biral, J. (1 de Abril de 2021). *Volley JSON Request*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/Request.java

Davidson, J., Morey, J., & Liu, E. (26 de Marzo de 2020). *Volley Response*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/Response.java

Davidson, J., Zhao, Y., & Faraj, K. (1 de Junio de 2021). *Volley*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/toolbox/Volley.java

Davison, J., Liu, E., Hager, U., & Eum, J. (21 de Agosto de 2020). *Volley Request Queue*. Obtenido de https://github.com/google/volley/blob/master/src/main/java/com/android/volley/RequestQueue.java

DFRobot. (s.f.). *Gravity I2C BME680 Environmental Sensor VOC, Temperature, Humidity, Barometer SKU SEN0248*. Obtenido de https://wiki.dfrobot.com/Gravity__I2C_BME680_Environmental_Sensor__VOC,_Temperature,_Humidity,_Barometer__SKU__SEN0248

Eernisse, M. (2012). *EJS. Embedded JavaScript templating.* Obtenido de https://ejs.co/

Espressif . (6 de Marzo de 2017). *Espressif Systems Datasheet*. Obtenido de https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Espressif. (2021). *Espressif*. Obtenido de https://www.espressif.com/en/products/modules/esp32

Espressif. (2021). *Espressif*. Obtenido de https://www.espressif.com/en/products/modules/esp32espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

Express. (2017). *Express*. Obtenido de https://expressjs.com/

Express. (2017). *Template engines*. Obtenido de https://expressjs.com/en/resources/template-engines.html

Google. (2020). *volley Library*. Obtenido de https://github.com/google/volley

Google. (27 de Mayo de 2021). *GSON Library*. Obtenido de https://github.com/google/gson

Gravity. (s.f.). *I2C BME680*. Obtenido de https://www.dfrobot.com/product-1697.html

Handy, A. (24 de Junio de 2011). *Node.js pushes JavaScript to the server-side*. Obtenido de https://sdtimes.com/javascript/node-js-pushes-javascript-to-the-server-side/

Ibernstone. (2018). *Bluetooth Serial Library*. Obtenido de https://github.com/espressif/arduino-esp32/tree/master/libraries/BluetoothSerial

Java. (s.f.). *¿Qué es la tecnología Java y para qué la necesito?* Obtenido de https://www.java.com/es/download/help/whatis_java.html

JavaScript: The Definitive Guide, 6th Edition. (2011). O'Reilly Media, Inc.

Knight, H. M., Gajendragadkar, P. R., & Bokhari, A. (12 de Mayo de 2015). *Wearable technology: using Google Glass as a teaching tool*. Obtenido de https://pubmed.ncbi.nlm.nih.gov/25969411/

Kreinin, Y. (13 de Octubre de 2009). *Defective C++*. Obtenido de http://yosefk.com/c++fqa/defective.html

Li, H., Wu, J., Gao, Y., & Shi, Y. (17 de Abril de 2016). *Examining individuals' adoption of healthcare wearable devices: An empirical study from privacy calculus perspective*. Obtenido de https://pubmed.ncbi.nlm.nih.gov/26878757/

Luján Mora, S. (2001). *Programación en Internet: Clientes Web.* Club Universitario.

Luján Mora, S. (2002). *Programación de aplicaciones web: historia, principios básicos y clientes web.* Club Universitario.

luoyunfenglalala. (2017). *BME680 Library*. Obtenido de https://github.com/DFRobot/DFRobot_BME680/blob/master/DFRobot_BME680_I2C.h

Martín, J. (20 de Noviembre de 2019). *Tecnalia Research & Innovation*. Obtenido de https://es.slideshare.net/TECNALIA/wearables-para-la-salud-y-prevencin-de-riesgos-laborales-195541019

MDN Contributors. (23 de Junio de 2021). *Generalidades del protocolo HTTP*. Obtenido de https://developer.mozilla.org/es/docs/Web/HTTP/Overview

MDN Web Docs. (21 de Mayo de 2021). *About Javascript.* Obtenido de https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript?redirectlocale=en-US&redirectslug=JavaScript%2FAbout_JavaScript

MDN Web Docs. (23 de Junio de 2021). *JavaScript*. Obtenido de https://developer.mozilla.org/es/docs/Web/JavaScript

MDN Web Docs. (14 de Abril de 2021). *Learn to style HTML using CSS*. Obtenido de https://www.w3.org/TR/CSS21/syndata.html#q10

Node.js. (s.f.). *About Node.js*. Obtenido de https://nodejs.org/en/about/

Normas APA. (s.f.). *¿Qué es el estado del arte?* Obtenido de https://normasapa.net/que-es-el-estado-del-arte/

Oracle. (2020). *Package Java.io*. Obtenido de https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html

Oracle. (2020). *Package Java.net*. Obtenido de https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html

Oracle. (2020). *Package Java.util.Description*. Obtenido de https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html#:~:text=util%20Description,%2C%20and%20a%20bit%20array)

Palau, G. (8 de Enero de 2019). *Desarrollo de Apps: Diferencias entre Android (Google) e iOS (Apple)*. Obtenido de https://pickaso.com/2019/desarrollo-apps-diferencias-android-vs-ios

Prime Faraday Technology Watch. (Enero de 2002). *An Introduction to MEMS (Micro-electromechanical Systems).* Obtenido de https://www.lboro.ac.uk/microsites/mechman/research/ipm-ktn/pdf/Technology_review/an-introduction-to-mems.pdf

Signorini, M. G., Fanelli, A., & Magenes, G. (30 de Enero de 2014). *Monitoring fetal heart rate during pregnancy: contributions from advanced signal processing and wearable technology*. Obtenido de https://pubmed.ncbi.nlm.nih.gov/24639886/

Sruostrup, B. (1997). *The C++, Programming Language.* Adison-Wesley.

W3C. (s.f.). *Syntax and Basic data types for CSS*. Obtenido de https://www.w3.org/TR/CSS21/syndata.html#q10

WebEmpresa. (s.f.). *¿Qué es GITHub y para qué sirve?* Obtenido de https://www.webempresa.mx/hosting/que-es-github.html

Yin Chau, K., Sum Lam, M. H., Chueng, M. L., Ho Tso, E. K., Broom, D. R., Flint, S. W., . . . Yiu Lee, K. (24 de Septiembre de 2019). *Smart technology for healthcare: Exploring the antecedents of adoption intention of healthcare wearable technology*. Obtenido de https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6767842/

9. Glosario

- **API** (Interfaz de Programación de Aplicaciones)**:** conjunto de subrutinas, funciones y procedimientos que se emplean para desarrollar e integrar el software de las aplicaciones otorgándoles una capa de abstracción.
- **API REST:** API con filosofía del tipo REST (Representational State Transfer) que emplea protocolos sin estado mediante un conjunto de operaciones CRUD como operaciones POST, GET, PUT o DELETE.
- **CASE** (Computer Aided Software Engineering): conjunto de aplicaciones informáticas destinadas a aumentar el balance en el desarrollo de software.
- **CSS** (Style Sheet Cascade): lenguaje de diseño gráfico para la presentación de páginas basadas en lenguaje HTML.
- **CRUD** (Create, Retrieve, Update and Delete)**:** conjunto de operaciones básicas para la gestión de llamadas web.
- **DELETE:** solicitud HTTP con el objetivo de eliminar un recurso específico de un servidor.
- **ETL** (Extract Transfrom Load)**:** procedimiento de copia de datos desde uno o varios puntos a un destino determinado.
- **GET:** solicitud HTTP utilizada para solicitar datos desde un recurso específico.
- **HTML** (Hyper Text Markup Language)**:** lenguaje estándar para el desarrollo de páginas web.
- **HTTP** (HyperText Transfer Protocol): protocolo de comunicación entre dispositivos aplicado al WWW (World Wide Web) para el envío y recepción de información.
- **HTTPS** (HyperText Transfer Protocol Secure): protocolo basado en el HTTP para el envío de información de forma segura.
- **IDE** (Integrated Development Enviroment): entorno de desarrollo integrado con el objetivo de facilitar las tareas de programación.
- **IP** (Internet Protocol): la dirección IP es un número asignado que identifica de forma única una interfaz en red de cualquier dispositivo conectado a ella.
- **IOT** (Internet Of Things)**:** concepto empleado al conjunto de objetos cotidianos interconectados a la red con el objetivo de recopilar datos.
- **JSON** (JavaScript Object Notation)**:** formato de documento de texto para la representación de datos de forma estructurada para entornos JavaScript.
- **MEMS** (Micro-ElectroMechanical Systems)**:** proceso tecnológico mediante el cual se desarrollan dispositivos integrados de tamaño reducido.
- **MVC** (Modelo Vista Controlador): estilo de arquitectura empleado en el desarrollo de software que separa los datos, la interfaz del usuario y la lógica de control.
- **POST:** solicitud HTTP utilizada para enviar datos a un servidor con el objetivo de crear o actualizar un recurso de este.
- **PUT:** solicitud HTTP utilizada para enviar datos de manera idempotente a un servidor para crear o actualizar un recurso.
- **SoC:** tipo de arquitectura estandarizada para microchips.
- **TCP** (Transmission Control Protocol)**:** protocolo de red que permite la transmisión de datos entre dos puntos conectados sin errores.
- **UDP** (User Datagram Protocol)**:** protocolo mínimo de envío de datos orientado a mensajes.
- **UML** (Unified Modelling Language)**:** lenguaje estándar empleado para la modelación de sistemas en ámbitos de ingeniería del software.

- **URL** (Uniform Resource Locator): dirección asignada a un recurso con acceso a la red con el objetivo de ser localizado e ientificado.