UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Open Data in Smart Cities

Presentado por Alejandro de Castro Arnaiz
en Universidad de Burgos — 8 de julio de 2024
Tutor: Bruno Baruque Zanon y Héctor Cogollos Adrián

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Bruno Baruque Zanon, profesor del departamento de Digitalización, área de Ciencia de la Computación e Inteligencia Artificial. D. Héctor Cogollos Adrián, profesor del departamento de Ingeniría informática, área de Lenguajes y Sistemas Informáticos. Exponen:

Que el alumno D. Alejandro de Castro Arnaiz, con DNI 71792120N, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Open Data in Smart Cities de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 8 de julio de 2024

Vº. Bº. del Tutor:

BARUQUE ZANON, BRUNO (FIRMA)

Firmado digitalmente por BARUQUE ZANON, BRUNO (FIRMA)
Fecha: 2024.07.08 17:35:38 +02'00'

D. Bruno Baruque Zanón

Vº. Bº. del co-tutor:

D. Héctor Cogollos Adrián

Resumen

Durante los últimos años, el acceso a datos en tiempo real gracias a sensores ha incrementado exponencialmente, dejando así, una versión nueva del Internet de las cosas (IOT). Esto permite que estén disponibles datos como aforos, horarios, precios y disponibilidad entre otros muchos datos de diferentes edificios y servicios en las ciudades. La gran cantidad de organismos que ofrecen datos abiertos provoca que haya multitud de formatos y estructuras de datos diferentes, lo que dificulta unificarlas en aplicaciones funcionales. Es por ello que hace unos años se lanzó *FIWARE*, una plataforma europea que cuenta con el apoyo con de las principales empresas tecnológicas y que propone un estándar para dichos datos. El proyecto consiste en el desarrollo de una aplicación que permita al usuario una experiencia más cómoda en una ciudad, mediante el tratamiento de datos abiertos incorporando *FIWARE*. Se ofrece al usuario diversas funcionalidades con los datos obtenidos de su ciudad (aforos, localizaciones, teléfonos, horarios...), como la consulta, registro de datos periódicamente, entre otras funcionalidades. Se ha desarrollado de manera interactiva, dejando así al usuario la capacidad de añadir nuevos conjuntos de datos.

Descriptores

fiware, iot, ngsi, datos en tiempo real, conjunto de datos, datos abiertos

Abstract

During the last few years, access to real-time data thanks to sensors has increased exponentially, leaving a new version of the Internet of Things (IOT). This allows data such as gauges, schedules, prices and availability among many other data from different buildings and services in cities to be available. The large number of organizations that offer open data means that there are many different formats and data structures, which makes it difficult to unify them into functional applications. This is why FIWARE, a European platform supported by leading technology companies, was launched a few years ago to propose a standard for such data. The project consists of the development of an application that allows the user a more comfortable experience in a city, through the processing of open data incorporating FIWARE. It offers the user various functionalities with the data obtained from their city (gauges, locations, telephones, schedules...), such as consultation, data registration periodically, among other functionalities. It has been developed in an interactive way, giving the user the ability to add new data sets.

Keywords

open data, fiware, iot, ngsi, real-time data, data set

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En la última década, la demanda de acceso instantáneo a datos en tiempo real ha crecido exponencialmente. Esta necesidad se debe en parte a la proliferación de dispositivos conectados y al avance de la tecnología de la información y la comunicación (TIC). Los seres humanos, tanto a nivel individual como colectivo, requieren información rápida y precisa para tomar decisiones informadas, mejorar la eficiencia operativa y optimizar la gestión de recursos [17, 20, 16].

Para acceder a la información en tiempo real, es fundamental el uso de dispositivos equipados con sensores y conexión a Internet. Estos dispositivos, que pueden variar desde teléfonos inteligentes y relojes inteligentes hasta sensores ambientales y cámaras de seguridad, son capaces de recopilar, procesar y transmitir datos de manera continua. Este flujo constante de datos permite a los sistemas y aplicaciones reaccionar instantáneamente a las condiciones cambiantes y proporcionar información actualizada a los usuarios.

El concepto de Internet de las Cosas (*IoT*, por sus siglas en inglés) se refiere a la interconexión de dispositivos y objetos cotidianos a través de Internet, permitiéndoles enviar y recibir datos. Según Gubbi et al. (2013), IoT es ”una visión en la que ’cosas’ (objetos físicos) están integradas con electrónica, software, sensores y conectividad de red para recopilar e intercambiar datos” [16]. Esto crea un ecosistema en el que los datos pueden ser utilizados para mejorar la eficiencia, productividad y calidad de vida.

En el contexto de las *Smart Cities*, el *IoT* juega un papel crucial. Los sensores y dispositivos IoT distribuidos por toda la ciudad pueden monitorizar el estado de diferentes infraestructuras, precios, disponibilidad, y otros parámetros esenciales en tiempo real.

Este trabajo propone una aplicación que facilita al usuario el acceso a todos estos datos, permitiendo así, a cualquier persona acceder a los datos de la ciudad que desee en tiempo real, de manera rápida y precisa.

Existen otras aplicaciones que permiten el acceso a dichos datos, pero muchas de ellas suelen estar enfocadas en un conjunto de datos concreto, como pueden ser los *parkings*, en una ciudad específica, o simplemente son *APIs* de datos que ofrecen los datos en crudo mediante ficheros *JSON*, *CSV* u otros formatos determinados. En esta aplicación, el usuario puede acceder al conjunto de datos que desee, de la ciudad que desee, y tener una visión gráfica y personalizada de los distintos conjuntos de datos. Además, esta aplicación tiene la ventaja de estar desarrollada de manera web, esto permite el acceso rápido a cualquier usuario y en cualquier dispositivo que tenga acceso a un navegador web, sin necesidad de descargarse ninguna aplicación.

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve introducción al problema a resolver y la solución propuesta en este trabajo. Estructura de la memoria y listado de materiales entregados.
- **Objetivos del proyecto:** objetivos que persigue el desarrollo del proyecto. Tanto los objetivos marcados por los requisitos del software y los objetivos de carácter técnico.
- **Conceptos teóricos:** explicación de los conceptos teóricos más importantes para la comprensión del proyecto.
- **Técnicas y herramientas:** técnicas y herramientas utilizadas durante las diferentes etapas del proyecto y una explicación y/o comparación con otras que finalmente no se han utilizado.
- **Aspectos relevantes del desarrollo:** aspectos más importantes del proyecto, como sus etapas, metodología, el proceso de aprendizaje...
- **Trabajos relacionados:**
- **Conclusiones y líneas de trabajo futuras:** conclusiones del proyecto, conclusiones técnicas, conclusiones personales y futuras mejoras.

La memoria contiene los siguientes anexos:

- **Plan del proyecto software:** planificación temporal del proyecto, y viabilidad económica y legal del mismo.

- **Especificación de requisitos del software:** se detallan los objetivos, los requisitos funcionales, así como los casos de uso de la aplicación.
- **Especificación de diseño:** se detallan las guías del diseño empleadas para el desarrollo del proyecto, tales como diagrama de secuencias, modelado de datos o diagrama de interfaces.
- **Manual del programador:** se recoge la información más relevante relacionada el código de la aplicación, así como las herramientas clave para su desarrollo, compilación y ejecución. También se incluyen distintas pruebas que se han realizado para asegurar el correcto funcionamiento del mismo.
- **Manual de usuario:** guía para el usuario donde se incluyen los requisitos para el uso de la aplicación y un manual para su uso.
- **ODS:** se incluye una reflexión personal sobre los aspectos de la sostenibilidad que se abordan en el trabajo.

2. Objetivos del proyecto

El objetivo principal del proyecto es el desarrollo de una aplicación que permita consultar los datos abiertos de *Smart Cities* utilizando la plataforma FIWARE [10].

2.1. Objetivos marcados por los requisitos del *software*

- Permitir a cualquier usuario el acceso a los distintos conjuntos de datos, que los servicios de *IT* hayan publicado en abierto, de manera visual, rápida y precisa.
- Ofrecer al usuario una experiencia sencilla y cómoda, enfocada a cualquier tipo de público. También se busca que esta experiencia sea personalizada, pudiendo personalizar la visualización de cada conjunto por cada usuario.
- Permitir al usuario guardar un registro de los conjuntos de datos que desee para un posterior análisis.
- Facilitar al usuario la incorporación de nuevos conjuntos de datos, disponibles para todos los usuarios.

2.2. Objetivos de carácter técnico

- Investigar sobre la implantación del protocolo de *IoT FIWARE* y generar una aplicación sencilla de publicación de datos y ponerlo en marcha.
- Aprender sobre el desarrollo de aplicaciones web.

- Ampliar y consolidar conocimientos sobre análisis y tratamiento de datos.
- Usar *GitHub* para el control de versiones.
- Implementar una base de datos en la aplicación para guardar los diferentes datos necesarios.
- Usar una metodología ágil *Scrum* para el desarrollo del proyecto.
- Aplicar y ampliar los conocimientos sobre *Python*, desarrollo web, bases de datos, gestión de proyectos, estructuras de datos y análisis de *software* adquiridos en el grado.
- Aplicar y ampliar los conocimientos adquiridos durante el grado.
- Ampliar los conocimientos de la herramienta de desarrollo de documentos LaTeX.

3. Conceptos teóricos

Para entender correctamente el proyecto, procedo a explicar los conceptos fundamentales que servirán de cimientos para el desarrollo del proyecto.

En primer lugar se explica la plataforma FIWARE. Posteriormente el modelo de información NGSI-LD que servirá de estándar y el Orion Context Broker, el encargado de almacenar y gestionar toda la información.

3.1. Introducción a *IoT*

El Internet de las Cosas (*IoT*, por sus siglas en inglés) es un concepto que se refiere a la interconexión de dispositivos y objetos físicos a través de internet. Estos dispositivos, equipados con sensores, software y otras tecnologías, pueden recopilar y compartir datos con otros dispositivos y sistemas a través de redes. *IoT* permite la automatización y el control remoto de procesos y sistemas, lo que resulta en una mayor eficiencia, ahorro de costos y mejora en la calidad de vida [18, 1, 15].

Componentes Clave de *IoT*

- **Dispositivos y Sensores**: Recogen datos del entorno o del sistema en el que están integrados.
- **Conectividad**: Permite la transmisión de datos a través de redes (Wi-Fi, Bluetooth, 5G, etc.).
- **Plataformas de Datos**: Almacenan y procesan los datos recogidos por los dispositivos.

- **Interfaz de Usuario**: Permite a los usuarios interactuar con el sistema *IoT*, controlando dispositivos y visualizando datos.

3.2. *Fiware*

Fiware es una iniciativa que ofrece un conjunto de estándares abiertos para el manejo y compartición de datos, impulsada por la Unión Europea para el desarrollo y despliegue global de aplicaciones *IoT*. Proporciona *APIs* abiertas y componentes para gestionar información de contexto en soluciones inteligentes, gemelos digitales y espacios de datos [10]. En resumen, *Fiware* es una base tecnológica que permite crear soluciones innovadoras en áreas como las Ciudades Inteligentes, Agricultura Inteligente, Energía Inteligente, Industria Inteligente y Gestión del Agua.

3.3. *NGSI-LD* y su aplicación en Ciudades Inteligentes

NGSI-LD es un modelo de información y *API* para editar, consultar y suscribirse a información de contexto, definido dentro de la plataforma *Fiware*. Está destinado a facilitar el intercambio abierto y la compartición de información estructurada entre diferentes partes interesadas [6]. Se utiliza en diversos ámbitos de aplicación, como Ciudades Inteligentes, Industria Inteligente, Agricultura Inteligente, y más generalmente para el Internet de las Cosas, Sistemas Ciberfísicos y Gemelos Digitales [5].

El modelo *NGSI-LD* se compone de entidades, atributos y metadata.

Las entidades de definen como objetos *JSON* y representan cada uno de los objetos físicos o lógicos, como sensores o estancias. En ese proyecto serían *parkings*, bibliotecas, puntos de carga, etc). Todas las entidades deben poseer:

- **identificador** para poder relacionar cada entidad con la información que proporciona
- **tipo**, el cual indicará la naturaleza del objeto, si es una *parking*, un tipo de sensor, etc.

Las entidades poseen también otras propiedades denominadas atributos, el segundo componente de este modelo. Los atributos aportan información sobre la entidad y tienen la siguiente estructura:

- **Nombre:** define el atributo. Por ejemplo, "plazas libres".
- **Tipo:** representa el tipo de atributo, no corresponde con los tipos de JSON ya que *NGSI* tiene sus propios valores. Por ejemplo, "*Number*".
- **Valor:** contiene el valor del atributo. Opcionalmente aquí puede haber metadatos, como la unidad o marca de tiempo.

Los metadatos tienen la misma estructura que los atributos: nombre, tipo y valor. Estos nos proporcionan valores adicionales sobre el atributo, como puede ser las unidades del valor.

Ejemplo modelo *NGSI-LD*

Imagina que estamos desarrollando una aplicación para monitorizar el tráfico en una ciudad inteligente. Queremos obtener información sobre el estado actual de los semáforos en una intersección específica.

1. **Entidad (Entity)**: Representa un semáforo en una intersección. Cada semáforo tiene un ID único y se encuentra en una ubicación específica.
2. **Atributos (Attributes)**: Los atributos de un semáforo podrían incluir:
   - **Estado (State)**:
     - **type**: `Property`
     - **value**: Puede ser “rojo”, “verde” o “amarillo”.
   - **Tiempo restante (Remaining Time)**:
     - **type**: `Property`
     - **value**: El tiempo restante en segundos para cambiar de estado.
     - **unitCode**: `SEC`
   - **Ubicación (Location)**:
     - **type**: `GeoProperty`
     - **value**: Las coordenadas geográficas del semáforo (latitud, longitud).
3. **Contexto (Context)**: En nuestro modelo, el contexto sería la información sobre los semáforos en la ciudad. Por ejemplo:

- **Semáforo 1**:
  - **ID**: `TrafficLight:sem1`
  - **type**: `Semáforo`
  - **Estado**:
    - **type**: `Property`
    - **value**: "rojo"
  - **Tiempo restante**:
    - **type**: `Property`
    - **value**: 30
    - **unitCode**: `SEC`
  - **Ubicación**:
    - **type**: `GeoProperty`
    - **value**: (40.7128, -74.0060)
- **Semáforo 2**:
  - **ID**: `TrafficLight:sem2`
  - **type**: `Semáforo`
  - **Estado**:
    - **type**: `Property`
    - **value**: "verde"
  - **Tiempo restante**:
    - **type**: `Property`
    - **value**: 15
    - **unitCode**: `SEC`
  - **Ubicación**:
    - **type**: `GeoProperty`
    - **value**: (40.7127, -74.0059)

4. **Consulta (Query)**: Podemos consultar el estado actual de un semáforo específico utilizando su ID. Por ejemplo:
   - "¿Cuál es el estado actual del semáforo con ID 'TrafficLight:sem1'?"
5. **Respuesta (Response)**: La aplicación recibiría la información del semáforo solicitado y podría mostrarla al usuario.

En resumen, *NGSI-LD* nos permite modelar y acceder a información de contexto de manera estructurada, lo que es fundamental para construir aplicaciones inteligentes y conectadas [8]. Las entidades y atributos definidos en *NGSI-LD* se utilizan para representar datos en aplicaciones *Fiware*.

3.4. *Orion Context Broker*

El componente principal y esencial de cualquier plataforma o solución desarrollada con *Fiware* es el *Orion Context Broker (OCB)*, que desempeña una función crucial en cualquier solución inteligente: gestionar, consultar y actualizar la información de contexto.

El OCB permite que las entidades, conocidas como proveedores de contexto (como los sensores), publiquen información de contexto para que esté disponible para otras entidades, conocidas como consumidores de contexto, que están interesadas en procesar dicha información. Por ejemplo, una aplicación que desea consultar el estado de los semáforos para calcular la ruta más corta en un momento dado es un consumidor de contexto. Los proveedores y consumidores de contexto pueden ser cualquier aplicación o incluso otros componentes dentro de la plataforma *Fiware* [9].

El *OCB* es un servidor que implementa una *API* basada en el modelo de información *NGSI*, permitiendo varias operaciones:

- **Registrar aplicaciones de proveedores de contexto**: por ejemplo, un sensor de temperatura en una habitación.
- **Actualizar información de contexto**: por ejemplo, enviar actualizaciones sobre la temperatura.
- **Ser notificado de cambios en la información de contexto**: por ejemplo, cuando la temperatura cambia, o a intervalos regulares, como obtener la temperatura cada minuto.
- **Consultar información de contexto**: Orion almacena la información de contexto actualizada desde las aplicaciones, lo que permite que las consultas se resuelvan basándose en esta información.

El servidor *OCB* siempre está escuchando en un puerto que generalmente es el 1026. el *OCB* utiliza la base de datos es *MongoDB* para almacenar el estatus actual de las entidades, no se almacena información histórica de sus cambios. Para este propósito se debe utilizar una base de datos externa al *OCB* [11], como es el caso de *Cygnus*[1] , componente el cual se estudió su

[1] *Cygnus* es un componente de la plataforma *Fiware* que actúa como un conector de datos y tiene la función principal de persistir la información de contexto gestionada por el *Orion Context Broker* en diferentes sistemas de almacenamiento a largo plazo. *Cygnus* recibe notificaciones de *Orion* cuando hay actualizaciones en la información de contexto y las almacena en bases de datos u otros sistemas de almacenamiento seleccionados por el usuario [7].

implementación en este proyecto como se detalla en el anexo A, apartado de *Planificación temporal Sprint 5: Simulador (14/02/2024 - 14/03/2024)*.

En la imagen 3.1 extraída de la documentación oficial[2] se puede apreciar el esquema de funcionamiento del *OCB*:

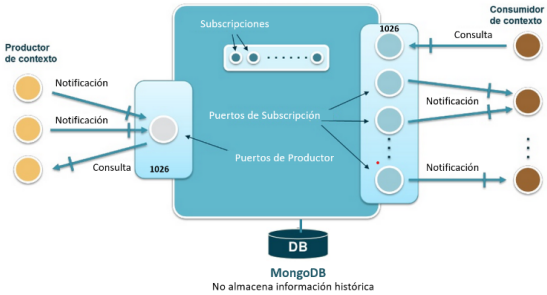


Figura 3.1: Diagrama de *OCB*

[2]Imagen extraída de https://fiware-training.readthedocs.io/es-mx/latest/ecosistemaFIWARE/ocb/

4. Técnicas y herramientas

4.1. Técnicas metodológicas

Scrum

Scrum es un marco de trabajo ágil para la gestión y desarrollo de proyectos complejos, especialmente en el ámbito del software. Se basa en iteraciones cortas y regulares llamadas *sprints*, que típicamente duran entre una y cuatro semanas. El objetivo de *Scrum* es proporcionar un producto potencialmente entregable al final de cada sprint, lo que permite realizar ajustes y mejoras continuas basadas en la retroalimentación entre el equipo. [22]

Se decidió usar la metodología ágil *Scrum* con *sprints* de 2 semanas, durante este periodo apuntaba posibles mejoras o ideas, y realizaba los cambios o añadidos establecidos para este periodo. Transcurrido el plazo, se realizaba una reunión donde se trataban dichos puntos y se establecían los siguientes. Para ello nos ayudábamos de las herramientas de Gestión de tareas 4.2.

Mock Server

Un *mock server* es un servidor que imita el comportamiento de un servidor real, proporcionando respuestas predefinidas a solicitudes específicas. Se utiliza principalmente durante el desarrollo y las pruebas de aplicaciones para simular interacciones con *APIs* externas o componentes del sistema que aún no están disponibles o completamente desarrollados.

Estas son las razones por las que se decidió usar esta técnica en el proyecto:

- **Desarrollo Aislado:** Permite trabajar en partes de la aplicación que dependen de servicios externos sin necesidad de esperar a que esos servicios estén disponibles.
- **Pruebas predecibles:** Proporciona respuestas consistentes y controladas, lo que facilita la realización de pruebas unitarias y de integración.
- **Evitar Límites y Costos:** Ayuda a evitar límites de tasa o costos asociados con el uso de servicios en tiempo real durante el desarrollo y las pruebas.

Un *mock server* funciona de la siguiente manera, recibe solicitudes *HTTP* (*GET*, *POST*, *PUT*, *DELETE*, etc.) y devuelve respuestas predefinidas, que pueden incluir códigos de estado *HTTP*, encabezados y cuerpos de respuesta. En este caso se ha usado la herramienta *Postman* (ver 4.2) para el desarrollo de esta técnica.

4.2. Herramientas

Las herramientas usadas para el proyecto son muy diversas y por ello se han clasificado según su funcionalidad. Quedan destalladas a continuación

Gestión de tareas

- Microsoft Planner[3]: permite gestionar las tareas del proyecto entre los diferentes *sprints*, fue utilizada desde el inicio del proyecto hasta finales de abril. Se escogió porque es una herramienta muy útil para la gestión de tareas y era la que usaba en ese momento en el trabajo, por lo que tenía ya conocimientos previos sobre ella. Un punto calve para su elección fue el acceso gratuito con la cuenta educativa de la universidad. Después dejó de usarse, ya que GitHub ofrecía una alternativa similar con *issues*, y permitía enlazar las diferentes tareas con los cambios realizados en el código.
- GitHub[4]: es una plataforma de desarrollo colaborativo, la cual incluye control de versiones y gestión de proyectos gracias a los *issues*, que eran utilizados para marcar las tareas y su estado durante los distintos *sprints*.

[3] https://tasks.office.com
[4] https://github.com

- ZenHub[5]: es una herramienta de gestión de tareas muy útil ya que combina características de las dos anteriores y tiene sincronización con GitHub, por lo que el control de versiones del código puede ser muy completo. No fue usada finalmente por ser de pago.

Editor de texto

LaTeX: es un sistema de composición de documentos basado en el procesador de textos TeX, utilizado para el desarrollo de la memoria y los anexos. Su elección se basa principalmente en el objetivo de aplicar y ampliar los conocimiento adquiridos en la carrera.

Gestión bibliográfica

Para la realización de una bibliografía completa se ha usado la herramienta *JabRef*[6]. Es un gestor de referencias bibliográficas de código abierto que facilita la organización y gestión de citas y bibliografías para documentos académicos.

Repositorio

Como repositorio se escogió *GitHub*, debido a que es una herramienta muy popular con una gran integración con la mayoría de editores de código como Visual Studio Code, el usado en este proyecto. Además de tener control de versiones, muy útil para proyectos complejos como este.

Editores de código

Para este proyecto se ha usado únicamente *Visual Studio Code 2022*, debido a ser un editor gratuito que posee extensiones que facilitan la programación en distintos lenguajes, y con una buena integración **GitHub**.

Docker

Docker es una plataforma de software que permite crear, probar y desplegar aplicaciones rápidamente. Utiliza contenedores, que son entornos aislados que incluyen todo lo necesario para ejecutar una aplicación, como el código, las bibliotecas y las dependencias. Esto garantiza que las aplicaciones

[5] https://www.zenhub.com/
[6] https://www.jabref.org

funcionen de manera consistente en diferentes entornos, desde el desarrollo hasta la producción [19, 3].

En este proyecto se ha utilizado *Docker* para la implementación del *Orion Context Broker*, el *mock server* que se ha utilizado para el tratamiento de datos *NGSI-LD*.

Base de datos y APIs

En este proyecto, la gestión de los datos es muy importante, ya que, se trabajan con muchos datos. Para ello, se han usado una serie de herramientas.

XAMPP

XAMPP es una distribución gratuita y de código abierto que facilita la instalación y el uso de un entorno de desarrollo web en un sistema local. Su nombre es un acrónimo que representa los componentes principales que incluye:

- **X** (Cross-platform): Indica que es multiplataforma, compatible con Windows, Linux, y macOS [13].
- **A** (Apache): Incluye el servidor web Apache, que es uno de los servidores web más populares del mundo.
- **M** (MariaDB/MySQL): Proporciona el sistema de gestión de bases de datos MariaDB (anteriormente MySQL).
- **P** (PHP): Incluye el lenguaje de scripting PHP, que es ampliamente utilizado para el desarrollo web del lado del servidor.
- **P** (Perl): También incluye Perl, otro lenguaje de programación.

XAMPP simplifica la instalación y configuración de estos componentes, lo que lo hace ideal para desarrolladores que desean crear, probar y depurar aplicaciones web localmente antes de desplegarlas en un servidor en producción [14].

MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) que utiliza el lenguaje SQL (Structured Query Language) para gestionar y manipular datos. Es uno de los sistemas de bases de datos más utilizados

en el mundo, especialmente en aplicaciones web [2], como es el caso de este proyecto. Algunas características clave de MySQL incluyen:

- **Alto Rendimiento**: Optimizado para rendimiento y escalabilidad, capaz de manejar grandes cantidades de datos y transacciones.
- **Confiabilidad**: Proporciona características avanzadas como recuperación ante fallos y replicación para garantizar la integridad y disponibilidad de los datos.
- **Flexibilidad**: Soporta múltiples motores de almacenamiento (InnoDB, MyISAM, etc.) que permiten elegir el más adecuado según las necesidades específicas.
- **Compatibilidad**: Es compatible con múltiples plataformas y puede integrarse con una amplia variedad de lenguajes de programación y herramientas de desarrollo.

MySQL es utilizado por muchas aplicaciones y sitios web de alto tráfico, como Facebook, Twitter, y YouTube, debido a su eficiencia y robustez. Es una elección común para desarrolladores que buscan una solución de base de datos relacional confiable y de alto rendimiento [4], ambas cualidades claves en os objetivos marcados en este proyecto.

Postman

Postman es una herramienta de desarrollo de *API*[7] que facilita la creación, prueba, documentación y monitorización de APIs. En este caso, para hacer consultas y escrituras en el servidor de Orion Context Broker. Ofrece una interfaz amigable que permite realizar peticiones HTTP y ver las repuestas del servidor, como podemos en las imágenes 5.1 y 5.1.

Bocetos de ventanas

Se modeló un diseño de ventanas previo al diseño estético o *front-end* de la aplicación para poder analizar el diseño ***UX***[8] y ***UI***[9] de la aplicación, objetivos clave en este proyecto como se ha remarcado en la sección de objetivos2.1. Para ello se utilizó **Pencil**[10], una aplicación de código abierto

[7] Interfaz de Programación de Aplicaciones)
[8] User Experience
[9] User Interface
[10] https://pencil.evolus.vn

utilizada para la creación de prototipos y el diseño de interfaces de usuario (***UI***).

Gitbook

Esta herramienta ha sido empleada para el desarrollo de una guía de usuario visual y fácil de seguir gracias a la esquematización y división en apartados de la misma.

Lenguajes de programación y librerías

Python

Python es uno de los lenguajes de programación más usados en todo el mundo en la programación orientada a objetos (POO). Posee estructuras de datos de alto nivel, sintaxis legible y tipado dinámico, lo cual lo convierten en un lenguaje ideal para el desarrollo de aplicaciones. Es por ello que se escogió este lenguaje.[12]

Otras opción que se planteó fue Java, y aunque ambas opciones poseen bibliotecas para el desarrollo web y siendo Java un lenguaje del cual poseía más conocimientos, se consideró que Python podría ser mejor opción debido a su simpleza y velocidad a la hora de desarrollar aplicaciones.

Flask

Flask es un *microframework* para *Python* que se utiliza para desarrollar aplicaciones web. Proporciona las herramientas y funcionalidades básicas necesarias para crear sitios web y aplicaciones web dinámicas, como manejo de rutas (URL routing), renderizado de plantillas HTML, manejo de formularios, y conexión a bases de datos, entre otras cosas [21]. Se decidió usar *Flask* por cumplir con varios de los objetivos principales del proyecto especificados en la sección de objetivos generales 2.1, permitiendo al usuario un acceso rápido a cualquier conjunto, solo introduciendo la *url* del conjunto al que desea acceder. También se garantiza un acceso cómodo, ya que al ser mediante una dirección web, no es necesario descargar ningún tipo de aplicación para su uso.

Librerías

La tabla 4.1 contiene la información de las principales librerías usadas en el desarrollo del proyecto, así como la versión y una breve descripción.

Librería	Versión	Descripción
`Flask`	`3.0.3`	Un microframework para aplicaciones web en Python, diseñado para ser simple y escalable.
`Flask-MySQLdb`	`2.0.0`	Extensión para Flask que proporciona conexión a bases de datos MySQL.
`Flask-WTF`	`1.2.1`	Integración de Flask con WTForms para renderizar formularios, validación y protección CSRF.
`mysql-connector-python`	`9.0.0`	Conector de MySQL escrito en Python que implementa la especificación DB API v2.0.
`dash`	`2.17.1`	Framework para construir aplicaciones web interactivas de ciencia de datos y ML.
`pandas`	`2.2.2`	Biblioteca para manipulación y análisis de datos, proporcionando estructuras de datos y operaciones para manipular tablas numéricas y series temporales.
`xmltodict`	`0.13.0`	Biblioteca para convertir datos XML en diccionarios de Python de manera sencilla.

Tabla 4.1: Tabla de librerías

Otros lenguajes

Las diversas vistas de la aplicación han sido meticulosamente desarrolladas utilizando *HTML5*, lo que garantiza una estructura semántica y moderna. Para complementar y embellecer estas vistas, se ha empleado *CSS3*, permitiendo así un estilo visual atractivo y personalizado que mejora la experiencia del usuario. Además, en ciertos apartados específicos, se ha integrado la potente herramienta *Bootstrap*[11], la cual facilita la creación de diseños responsivos y consistentes.

Es importante destacar que, para la implementación de funcionalidades interactivas y dinámicas, se han desarrollado *scripts* en *JavaScript*. Estos *scripts* no solo añaden interactividad, sino que también mejoran la usabilidad y la eficiencia de la aplicación, proporcionando una experiencia de usuario más rica y fluida.

Revisión de código

Para garantizar la calidad del código, se ha utilizado la herramienta *SonarCloud*. *SonarCloud* es una plataforma de código abierto que permite la inspección continua de la calidad del código. Esta herramienta realiza

[11] https://getbootstrap.com

revisiones automáticas mediante análisis estático del código, detectando errores y vulnerabilidades.

5. Aspectos relevantes del desarrollo del proyecto

5.1. Etapas del proyecto

Investigación

Durante la primera etapa del proyecto, se investigó acerca de *FIWARE* y sus distintas aplicaciones, y se decidió enfocar la aplicación a usar el modelo información *NGSI-LD* 3.3. Esta decisión fue impulsada por el reciente crecimiento de la plataforma *FIWARE* y el modelo *NGSI-LD*, gracias a la colaboración de las empresas más significativas en el desarrollo *IOT*, como Telefónica [12].

Despliegue del *Orion Context Broker* y *MongoDB*

Tras un proceso de búsqueda e investigación de *OpenData* reales que usaran el modelo *NGSI-LD*, por ser este un estándar de uso libre, se descubrió que el número de conjuntos de datos que siguiesen este modelo de información era muy excaso aún. Por lo que se decidió crear un *mock server* que usara NGSI-LD, en lugar de los distintos formatos que están en uso a día de hoy (*JSON*, *GEOJSON*, *CSV*, *XML*...), estandarizando así el uso de un mismo formato para todos los conjuntos de datos. Esto favorecería el desarrollo y mantenimiento de todas las aplicaciones que consuman este tipo de datos, no solo para las *Smart Cities* sino en muchos otro ámbitos como *Industria 4.0*.

[12]Proyecto de Telefónica de *fiware-orion*, el cual lleva 6 años en marcha y a día de hoy se sigue ampliando. https://github.com/telefonicaid/fiware-orion/tree/master

Se comenzó con la instalación del *Orion Context Broker* para la gestión de la información en tiempo real y, *MongoDB*, que actúa como la base de datos subyacente para almacenar esta información de contexto. Todo ello se desplegó en un *Docker*.

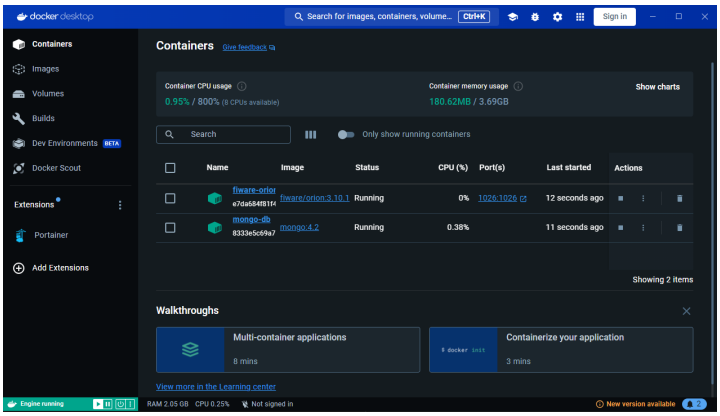


Figura 5.1: *Orion Context Broker* y la base de datos *MongoDB* en funcionamiento

Posteriormente se elaboró un *script* que permitiese al usuario gestionarlo de manera más rápida, ya que, el *Orion Context Broker* se gestiona mediante comandos, y para una mayor reusabilidad y mejor mantenimiento. Con este mini programa se permite al usuario:

- Iniciar y para el servidor, es decir encender el *docker* de del *Orion Context Broker* y el de la *MongoDB*.
- Mostrar y borrar las entidades almacenadas.
- Iniciar y para el simulador. Explicaré su funcionamiento más adelante en 5.1.
- Instalar y desinstalar los *dockers*.

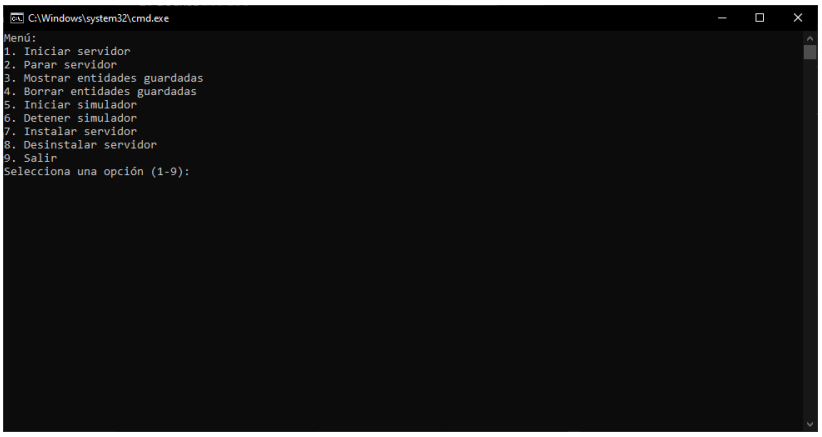


Figura 5.2: Programa gestor del *OCB*

En esta etapa mediante la herramienta *Postman*[13], se hicieron pruebas con el *Orion Context Broker*, creando, consultando, actualizando y borrando entidades el servidor. Así se podía verificar que el servidor funcionaba correctamente y estaba preparado para almacenar las distintas entidades. Las cuales posteriormente serían consultadas, actualizadas o eliminadas desde la aplicación con peticiones *GET*[14], *PATCH*[15] o *DELETE*[16] de *HTTP* y poder consultadas con peticiones *GET* de *HTTP* en el puerto 1026, que es el usado por el servidor.

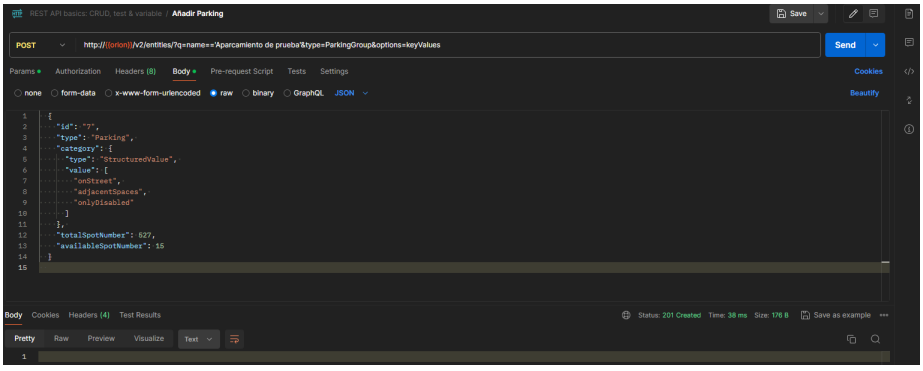


Figura 5.3: Ejemplo de uso de POST en *Postman*

[13]Herramienta para probar y gestionar *APIs*
[14]Se utiliza para solicitar datos de un recurso específico.
[15]Se utiliza para hacer actualizaciones parciales a un recurso existente.
[16]Se utiliza para eliminar un recurso específico.

Simulador de conjunto de datos usando el modelo *NGSI-LD*

Con las primeras cargas de datos se pudo observar que el modelo información *NGSI-LD* todavía no está siendo muy usado por las diferentes *APIs* de datos, por lo que se decidió crear un generador de datos que usara este modelo. Este es usado a modo de simulador, el cual genera de un conjunto de datos de *parkings* de la ciudad de Burgos simulando recibir datos de sensores instalados en dichos *parkings* y lo adapta al modelo *NGSI-LD*.

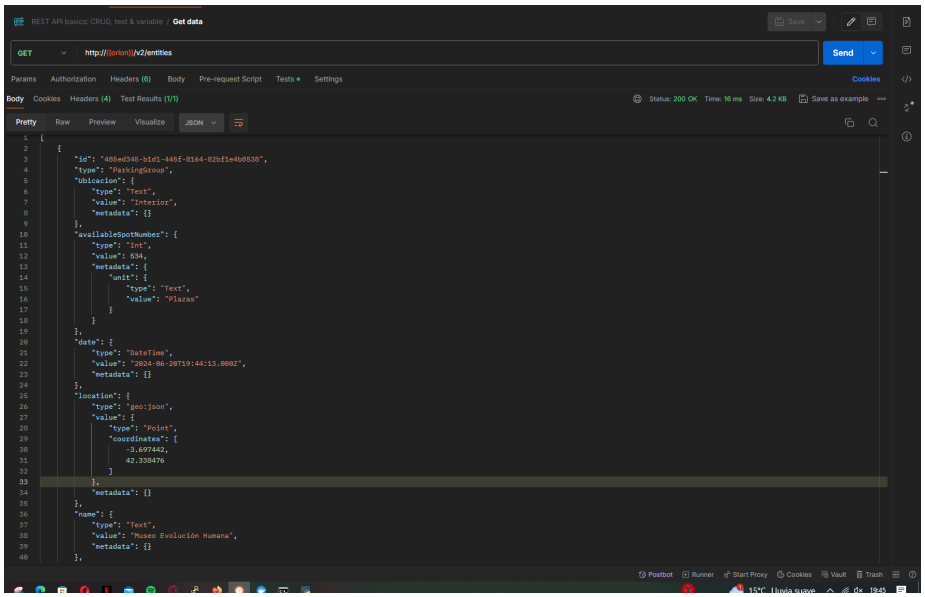


Figura 5.4: Ejemplo de uso de GET en Postman

Con esto se pudo probar que el *OCB*[17] estaba funcionando correctamente y pudiendo así leer y escribir datos de manera correcta. También se introdujeron algunas entidades diferentes, no solo *parkings*, para probar que los distintos conjuntos de datos podían convivir sin problema en el mismo *OCB*[17].

Para un mejor entendimiento de la estructura del proyecto, se adjunta una imagen con un esquema de la estructura 5.5.

[17] Orion Context Broker

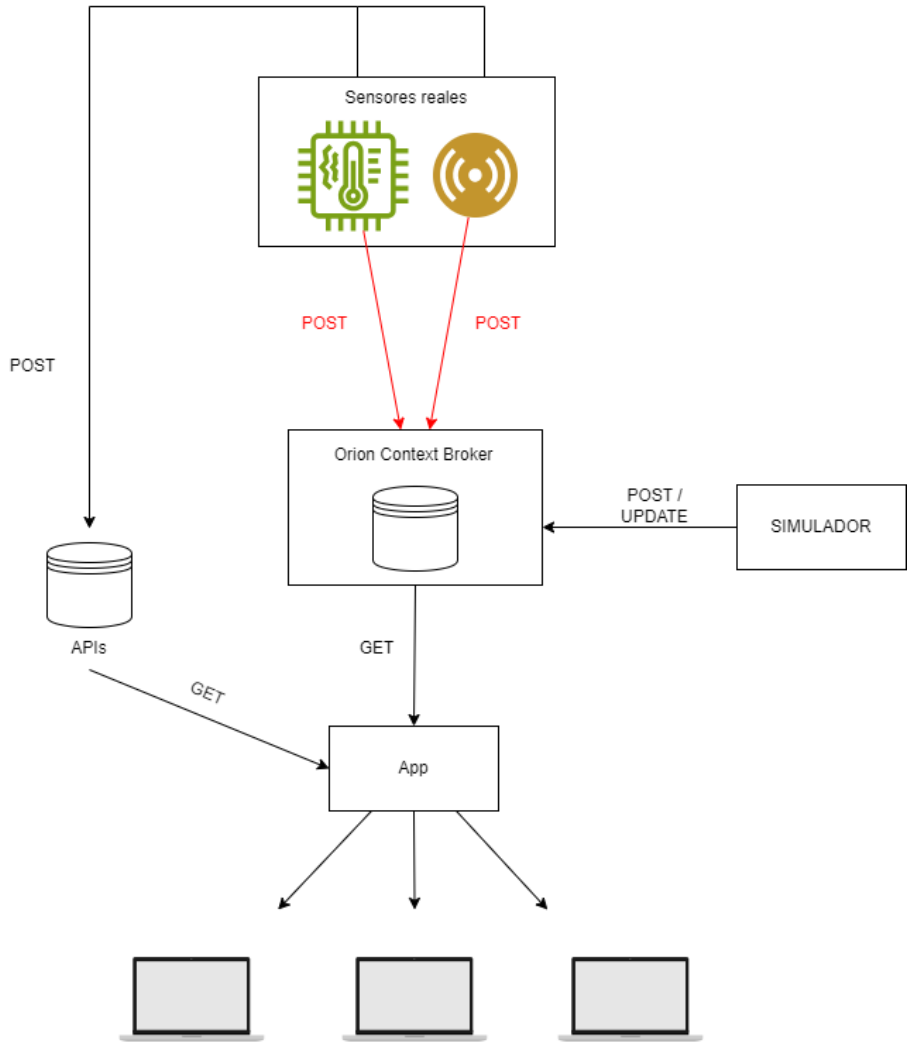


Figura 5.5: Estructura del proyecto

Desarrollo de la aplicación

Como el objetivo del proyecto es realizar una aplicación funcional y, el número de conjuntos de datos disponibles con el modelo de información*NGSI-LD* aún es mínimo, se comenzó con el desarrollo de la aplicación con los formatos tradicionales que se están usando a día de hoy (*JSON*, *GEOJSON*, *CSV*, *XML*...) y se dejó el *OCB*[17] preparado para que cuando el número de conjuntos que utilicen el modelo de información*NGSI-LD* sea mayor, se pueda utilizar.

Para el desarrollo de la aplicación se hizo un análisis de los posibles perfiles de usuarios que tendría y sus necesidades. Se clasificaron en dos grupos:

- **Usuario estándar:** usaría la aplicación para acceder a un conjunto de datos en concreto en un momento puntual. Su principal objetivo sería obtener información acerca de un conjunto de manera rápida y precisa. Por ejemplo, un usuario quiere decidir a qué biblioteca de su ciudad acudir, por ello, consulta en conjunto y analiza las posibles opciones en función de los parámetros que más le interesen (aforo, horario, proximidad...).
- **Usuario avanzado:** este usuario sería un perfil más próximo a un analista de datos, que desearía consultar uno o varios conjuntos, durante un periodo de tiempo determinado con una frecuencia establecida.

Una vez establecidos los perfiles y las necesidades a cubrir, se empezó con el análisis y diseño de la aplicación. Y posteriormente su desarrollo, el cuál, ha sido complejo ya que requería de cubrir las necesidades de los usuarios actuales y de usuarios futuros con un modelo de información diferente, el modelo *NGSI-LD*, mencionado en el apartado 3.3.

Para su desarrollo se ha tenido en cuenta que entre los objetivos generales, se encuentren objetivos tales como permitir al usuario tener una experiencia sencilla, cómoda y rápida. Es por ello que un usuario puede marcar que parámetros desea visualizar de un conjunto, o marcar varios conjuntos como favoritos y poder acceder desde el menú de inicio de manera rápida.

Las funcionalidades principales de dicha aplicación, basándose en los perfiles de los usuarios potenciales y en los objetivos y necesidades a cubrir son:

- **Consulta de conjuntos de datos en tiempo real:** esta sería la principal funcionalidad de la aplicación, permite al usuario consultar cualquier conjunto de datos de los que se encuentren incorporados en la aplicación.

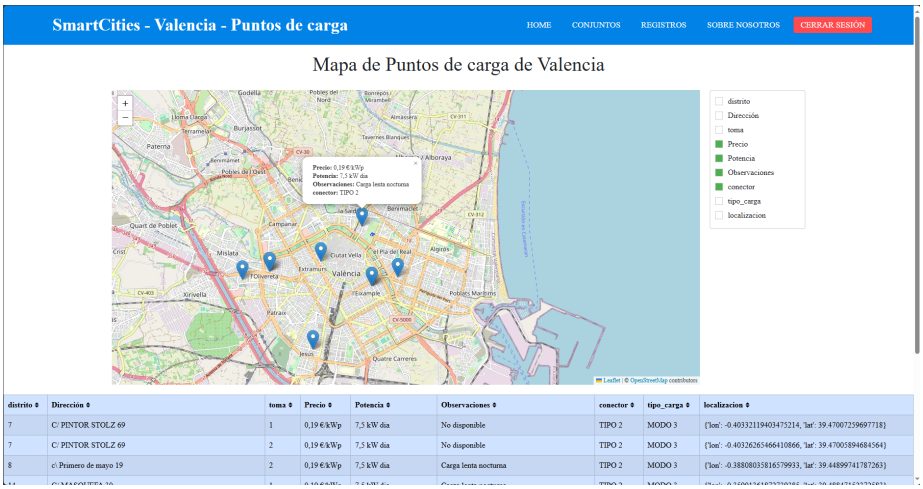
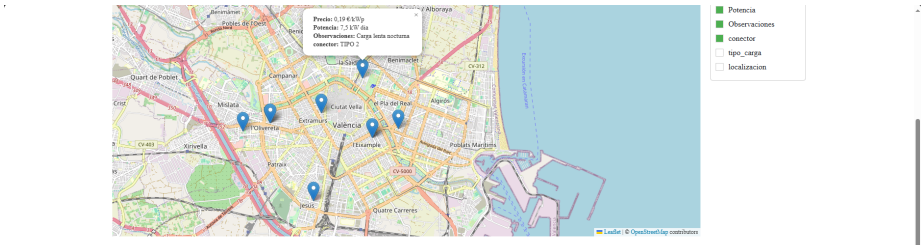


Figura 5.6: Ejemplo de consulta de datos en tiempo real

- **Añadir conjuntos de datos:** esta funcionalidad permite a los usuarios añadir nuevos conjuntos de datos, los cuales estarán visibles al resto de usuarios.
- **Añadir registros:** cualquier usuario puede comenzar a registrar un conjunto de datos con una frecuencia determinada, si la frecuencia es mayor con la que se actualizan en la *API* de datos de donde se extraen, se notifica al usuario para que corrija su elección de periodicidad.
- **Gestionar registros:** un usuario que tenga uno más o registros guardándose para él, puede descargar o eliminar lo registros que desee.
- **Favoritos:** a la hora de consultar un conjunto de datos, un usuario puede marcar o desmarcar de favoritos un conjunto, esto modificará su ventana de *home*, donde se muestran los conjuntos favoritos del usuario para poder acceder a ellos de una manera más rápida.
- **Personalizar vista de conjuntos de datos:** en los mapas, el usuario puede modificar los *tooltips*, para poder mostrar los parámetros que él desee. También puede ordenar la tabla por orden alfabético según el campo que desee.



distrito	Dirección	toma	Precio	Potencia	Observaciones	conector	tipo_carga	localizacion
7	C/ PINTOR STOLZ 69	1	0,19 €/kWp	7,5 kW dia	No disponible	TIPO 2	MODO 3	{'lon': -0.40332119403475214, 'lat': 39.47007259697718}
7	C/ PINTOR STOLZ 69	2	0,19 €/kWp	7,5 kW dia	No disponible	TIPO 2	MODO 3	{'lon': -0.40326265466410866, 'lat': 39.47005894684564}
8	c\ Primero de mayo 19	2	0,19 €/kWp	7,5 kW dia	Carga lenta nocturna	TIPO 2	MODO 3	{'lon': -0.38808035816579933, 'lat': 39.44899741787263}
14	C/ MASQUEFA 30	1	0,19 €/kWp	7,5 kW dia	Carga lenta nocturna	TIPO 2	MODO 3	{'lon': -0.35901361872739385, 'lat': 39.48847152372583}
2	c\ Conde Altea, 28	1	0,19 €/kWp	7,5 kW dia	Carga lenta nocturna	TIPO 2	MODO 3	{'lon': -0.3676063707870176, 'lat': 39.46612648701756}
6	C\ Santa Rosa, 4	1	0,19 €/kWp	7,5 kW dia	Carga lenta nocturna	TIPO 2	MODO 3	{'lon': -0.35848213577781678, 'lat': 39.468562113882296}
7	C\ Santiago de Lés	2	0,07€/kWp	Según produccion FV	Carga depende de la disponibilidad del sol	TIPO 2	MODO 3	{'lon': -0.4128996996922673, 'lat': 39.467890502269044}
2	c\ Conde Altea, 28	2	0,19 €/kWp	7,5 kW dia	Carga lenta nocturna	TIPO 2	MODO 3	{'lon': -0.3675769558813438, 'lat': 39.466135676343775}
3	Mercado Rojas Clemente c\ Simon Ortiz (en frente del nº1)	1	0,07€/kWp	Según produccion FV	Carga depende de la disponibilidad del sol	TIPO 2	MODO 3	{'lon': -0.38542695495544 5, 'lat': 39.47281510000963}
5	C\ Pintor Vilar, 1	1	0,19 €/kWp	7,5 kW dia	Carga lenta nocturna	TIPO 2	MODO 3	{'lon': -0.37096353521586 34, 'lat': 39.482238543131196}

Figura 5.7: Ejemplo de consulta de datos en tiempo real 2

Otras funcionalidades principales que solo tienen los administradores son:

- **Gestionar conjuntos:** un administrador puede eliminar conjuntos. Esto permite que en caso de que un administrador detecte o se le comunique que alguno de los enlaces está roto o los datos están corruptos, podrá quitar ese conjunto de la base de datos.
- **Gestionar usuarios:** un administrador puede consultar todos los usuarios que se encuentran en la aplicación, cambiar el rol de algún usuario o eliminarlo.
- **Gestionar traducciones:** un puede añadir traducciones que se emplean para hacer que la lectura de datos por parte del usuario sea más cómoda y legible. Por ejemplo, un atributo que sea "PlLibr.[en] un conjunto de datos, se vea traducido como "Plazas libres"

Revisión de código

El código ha sido revisado utilizando *SonarQube*, una herramienta de análisis estático de código que permite identificar errores, vulnerabilidades y problemas de calidad. Después de realizar el análisis, se corrigieron los principales problemas detectados para asegurar que el código cumpla con los estándares de calidad y seguridad requeridos.

6. Trabajos relacionados

En este apartado quiero mencionar algunos trabajos relacionado con el proyecto, los cuales han proporcionado información de calidad al proyecto.

6.1. TFM "Desarrollo de un entorno de visualización de datos de smart-city basados en NGSI" de Mario Núñez Callejo

Este proyecto contribuye a la evolución de las ciudades inteligentes al proporcionar una herramienta que facilita la interpretación y el uso de datos urbanos complejos, apoyando así la toma de decisiones informadas y la gestión eficiente de los recursos urbanos.

En este proyecto también se elabora una herramienta para la interpretación de datos NGSI, solo que solo abarca datos de Cantabria, lo cual limita los usos de la herramienta, aunque se comparten algunos objetivos comunes como mejorar la interoperabilidad de los datos de ciudades inteligentes mediante la estandarización y el uso de NGSI.

https://repositorio.unican.es/xmlui/bitstream/handle/10902/21941/434715.pdf

6.2. TFG"Plataforma IoT basada en FIWARE con cuadro de mandos integral y Business Intelligence para una Smart City" de Ismael Martínez fajardo

Este proyecto contiene una gran cantidad de información acerca de *Smart Cities* y de *IOTs*. Sobretodo hace mucho hincapié en la plataforma *Fiware* y explica muchos ejemplos que han ayudado a la comprensión de esta rama. Se exponen gran cantidad usos con *Fiware* y se detalla en profundidad sobre estructura y composición de la misma.

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwivrJr925aHAxWFRKQEHfW7AnoQFnoECBwQAQ&url=https%3A%2F%2Fopenaccess.uoc.edu%2Fhandle%2F10609%2F145822%3Flocale%3Des&usg=AOvVaw2uPoiBiShp42ZNQso7doBX&opi=89978449

7. Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

Tras finalizar el proyecto se han obtenido diferentes conclusiones, las cuales he clasificado en tres grupos.

- **Conclusiones del proyecto**
- **Conclusiones técnicas**
- **Conclusiones personales**

Conclusiones del proyecto

El proyecto ha concluido con éxito, cumpliendo todos y cada uno de los objetivos marcados al comienzo del mismo. Se ha conseguido realizar una aplicación funcional, rápida y accesible; cubriendo las necesidades de los usuarios de manera personalizada y un costo de recursos mínimo.

Conclusiones técnicas

- Es escalable y fácilmente mantenible. Permite añadir nuevos formatos de manera rápida y simple.
- Tiene una interfaz amigable con el usuario, permitiendo que pueda personalizar los conjuntos que desea consultar y añadir nuevos.

- La aplicación es robusta, tiene un tratamiento de excepciones preparado para cubrir todos los errores.
- Posee cifrado para que las contraseñas estén siempre seguras.
- Su implementación y uso es simple.

Conclusiones personales

Uno de los objetivos de este proyecto es aplicar y ampliar los conocimientos adquiridos durante el grado, con ello, he podido desarrollar nuevas habilidades de gestión de proyectos, análisis y diseño de sistemas, tratamientos de datos en tiempo real, administración de bases de datos, programación orientada a objetos, estructuras de datos, interacción hombre-maquina, entre otros.

Gracias a este proyecto he podido mejorar mucho la capacidad de adaptación a nuevos retos de diversa índole. También he podido trabajar la capacidad de resolución de problemas, ya que, como se ha detallado con anterioridad, en este proyecto, se ha llegado a puntos en los que los obstáculos encontrados eran muy superiores a la capacidad de recursos y tiempo que se tenía. Aún así, se ha sabido sacar el proyecto adelante con gran éxito.

7.2. Posibles mejoras futuras

A continuación se detalla una lista con posibles mejoras a implementar en futuras versiones. Dichas mejoras complementarían la aplicación de una manera muy positiva, ofreciendo al usuario nuevas funcionalidades o mejoras de las ya existentes:

- Como se ha detallado con anterioridad la aplicación posee acceso a multitud de conjuntos de datos, pero todos ellos con en formatos estándares actuales. En este proyecto se ha ido más allá, y se ha buscado poder tener una solución futura al nuevo *IOT*[18], con el nuevo modelo de información NGSI-LD, el cual ya ha sido modelado e integrado, como se puede ver en el simulador 5.1.
- Incorporación de la herramienta *Cygnus*[19] de *FIWARE* para la gestión del historial de entidades y poder mostrar así al usuario la evolución de las mismas.

[18] Internet Of Things

[19] https://fiware-cygnus.readthedocs.io/en/latest/

- Internacionalizar la aplicación es un punto bastante importante, ya que *FIWARE* es una plataforma europea, la cual es usada en muchos países.

Estas mejoras no se han incluido en el proyecto por falta de tiempo o ser muy avanzadas para el estado actual de la plataforma *FIWARE*.

Bibliografía

[1] Kevin Ashton. That 'internet of things' thing. *RFID Journal*, 2009.

[2] Oracle Corporation. Mysql :: Mysql documentation, 2023.

[3] Docker Inc. What is docker? https://www.docker.com/what-docker, 2023. Accessed: 2024-06-30.

[4] Paul Dubois. *MySQL Cookbook*. O'Reilly Media, 2014.

[5] ETSI. Ngsi-ld and the internet of things, 2020.

[6] ETSI Industry Specification Group (ISG). Ngsi-ld specification, 2018.

[7] FIWARE. Cygnus overview. https://fiware-tutorials.rtfd.io/en/latest/cygnus/. Consultado el 23 de junio de 2024.

[8] FIWARE Foundation. Ngsi-ld: A powerful model for context information management, 2020.

[9] FIWARE Foundation. *FIWARE Orion Context Broker*, 2023.

[10] FIWARE Foundation. Fiware: Una plataforma impulsada por la unión europea para el desarrollo y despliegue global de aplicaciones de internet del futuro. https://www.fiware.org, 2024. Consultado el 15 de junio de 2024.

[11] FIWARE Foundation. Orion context broker documentation. https://fiware-orion.readthedocs.io, 2024. Consultado el 15 de junio de 2024.

[12] Python Software Foundation. The python tutorial. https://docs.python.org/es/3/tutorial/. Consultado el 23 de junio de 2024.

[13] Apache Friends. Xampp - apache friends, 2023.

[14] Apache Friends. Xampp documentation, 2023.

[15] Gartner. What is the internet of things (iot)? https://www.gartner.com/en/information-technology/glossary/internet-of-things, 2023. Accessed: 2024-06-30.

[16] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[17] Sima Abolhassani Khajeh, Morteza Saberikamarposhti, and Amir Masoud Rahmani. Real-time scheduling in iot applications: A systematic review. *Sensors*, 20(12):3456, 2020.

[18] Adrian McEwen and Hakim Cassimally. *Designing the Internet of Things*. Wiley, 2013.

[19] Dirk Merkel. *Docker: Docker Up & Running*. O'Reilly Media, 2015.

[20] Keyur K Patel and Sunil M Patel. Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application future challenges. *International Journal of Engineering Science and Computing*, 6(5):6122–6131, 2016.

[21] Pallets Projects. Flask (a python microframework). https://flask.palletsprojects.com/. Consultado el 19 de junio de 2024.

[22] Ken Schwaber and Jeff Sutherland. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2020. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf.