Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android y web para la gestión de flotas mediante el uso de FIWARE, React y React Native

Autor: Fernando Aragón Urrea
Tutor: María Teresa Ariza Gómez

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android y web para la gestión de flotas mediante el uso de FIWARE, React y React Native

Autor:
Fernando Aragón Urrea

Tutor:
María Teresa Ariza Gómez
Profesor titular

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Trabajo de Fin de Grado: Aplicación Android y web para la gestión de flotas mediante el uso de FIWARE, React y React Native

Autor: Fernando Aragón Urrea

Tutor: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

Antes de comenzar la memoria, quiero agradecer a toda mi familia por el apoyo y confianza que han depositado en mí todos estos años, aportándome el coraje y valor para continuar los estudios y no desistir de ellos. Además de darme todos los medios y facilidades posibles para poder seguir en los estudios.

También agradecer a todos y cada uno de los profesores de la Escuela Técnica Superior de Ingeniería, cuyo esfuerzo ha servido para aumentar mi conocimiento y experiencia de cara al futuro. En especial a la tutora de este proyecto, María Teresa Ariza Gómez, por su tiempo y esfuerzo para orientarme en la realización de este proyecto, el cual no podría haber logrado sin ella.

Por último, agradecer a mis compañeros de carrera y amigos, con los cuales he pasado malos y grandes momentos, pero siempre estuvieron ahí.

Fernando Aragón Urrea

Sevilla, 2022

Resumen

Gracias al constante desarrollo en las Telecomunicaciones, el descubrimiento de tecnologías como el 5G ha fomentado la aparición y auge de los dispositivos móviles, así como la creación de conceptos como el *Internet of Things* (IoT), que permite conectarse e intercambiar datos a un gran número de dispositivos. Sin embargo, como estas tecnologías aún no se han desarrollado lo suficiente y a corto plazo no parece que vaya a haber un gran número de vehículos conectados a la red.

Este proyecto seguirá las líneas de los proyectos realizados por *Leopoldo Ángulo Gallego y Sergio Mellado Contioso*. Se ha creado una solución capaz de monitorizar la información de una flota de vehículos de una empresa de repartos. Sector que también ha ganado muchísima importancia en la última década. Además, también sería posible implementar la solución para el control de tráfico por parte de una organización estatal, como por ejemplo la Dirección General de Tráfico (DGT).

Se usará una sencilla aplicación React Native para la recogida y envío de la información de contexto, para el almacenamiento se utilizará la plataforma gratuita de software abierta FIWARE, y sus componentes Orion Context Broker (OCB) y Cygnus. El OCB se usará para la recogida de los datos enviados por la app, pero no aportará persistencia. Esa será la tarea de Cygnus, que los almacenará en una Base de Datos PostgreSQL.

Se diseñarán, además, un servicio web REST usando Express y nodejs que será el encargado de recibir la información del dispositivo móvil y enviársela al Orion Context Broker, las actualizaciones se podrían enviar directamente hacia el OCB, sin embargo, se envían al servicio REST, ya que este es necesario para acceder a la información de las BBDD MySQL y PostgreSQL, se utiliza para centralizar el tráfico de peticiones HTTP. El servicio dispone de los medios suficientes para implementar una función de Login, gracias a una BBDD MySQL y recoger los datos de la BBDD PostgreSQL.

Por último, se creará también un cliente web con React, para que aquellos encargados o gestores de la empresa puedan acceder y controlar la información de sus empleados.

Abstract

Thanks to the constant development in Telecommunications, the appearance of technologies like 5G has encouraged the apparition and rise of smartphones as well as the creation of concepts like the Internet of Things (IoT), that allows a great number of devices to connect and exchange data. However, these technologies have not been developed that much and in short term there will not be too many vehicles connected to the net.

This project will follow the lines established by the projects done by *Leopoldo Ángulo Gallego and Sergio Mellado Contioso*. It has been created a solution able to monitor the information of a vehicles fleet of a delivery company. Sector whose importance has increased in the last decade. Besides this, it would also be possible implementing this solution for the traffic control by a state organism, like *Dirección General de Tráfico* (DGT).

It will be used a simple React Native App the collecting and dispatching the context information, for the data storage it will be used FIWARE and its components Orion Context Broker (OCB) and Cygnus. OCB will collect the data sent by the mobile app without persistency. That's a task for cygnus that will store the data in a PostgreSQL database.

Also, a REST web service it is going to be designed using Express and nodejs, this web service will be in charge of gathering the information from the app and sending it to the OCB, updates could be directly sent to the OCB, however as the REST service is needed to collect the data from the MySQL and PostgreSQL databases, it will be used to centralize the traffic of HTTP requests. The service will be also able to incorporate a Login function thanks to the MySQL database and collecting the data from a PostgreSQL database.

Lastly, a React web client will be created, so a manager of the company can access and control his employees' information.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

IoT	Internet of Things
BBDD	Base de datos
OSM.	OpenStreetMaps
JSON	JavaScript Object Notation
OCB	Orion Context Broker
REST	Representational State Transfer
RAM	Random Access Memory
SSD	Solid-State Drive
GPS	Global Positioning System
SO	Sistema Operativo
DevTools	Developer Tools
IDE	Integrated Development Environment
AVD	Android Virtual Device
VM	Virtual Machine
HTTP	Hypertext Transfer Protocol
UML	Unified Modeling Language
JSX	JavaScript XML
XML	Extensible Markup Language
MongoDB	Mongo Database
npm	Node Packet Manager
SQL	Structured Query Language
API	Application Programming Interface
IP	Internet Protocol
CORS	Cross-Origin Resource Sharing
DIR	Dirección
JS	JavaScript
JDK	Java Development Kit
SDK	Software Development Kit
cmd	Command
USB	Universal Serial Bus

1 INTRODUCCIÓN

Trade isn't about goods. Trade is about information. Goods sit in the warehouse until information moves them.

- C. J. Cherryh -

El gesto de llevarse la mano al bolsillo para sacar el teléfono móvil se ha convertido en algo frecuente en la última década, un reciente estudio publicado por comScore sitúa en un 46.7% el porcentaje de europeos con teléfono inteligente en la actualidad [1]. Gracias a la proliferación del smartphone y espoleado por la reciente situación sanitaria, las empresas de servicio a domicilio y de transporte de mercancías aumentaron su volumen de negocio y ventas. Las ventas a domicilio sufrieron un alza del 70% en comparación a 2019, de acuerdo con un estudio de Euromonitor. Mientras que las empresas de transporte de mercancías resultaron vitales a la hora de mantener la cadena de suministros.

1.1 Motivación

El principal impulso del Proyecto es el de concederles a las empresas del sector de transporte las herramientas necesarias para poder mantener y gestionar de forma óptima una flota de vehículos: camiones, motos… con la información obtenida, y el posible uso de otras técnicas como el análisis masivo de datos (Big Data), para lograr maximizar los beneficios de la compañía.

Debido a que el coche conectado no parece que vaya a convertirse en una realidad tangible a corto plazo, será el uso del smartphone del conductor, gracias al que se obtendrá la información necesaria para controlar la ruta y velocidad de los vehículos entre otros datos. Para lograr esto, se diseñará una aplicación móvil usando React Native, un framework[1] Javascript de código abierto creado por Facebook que permite el diseño de aplicaciones híbridas para Android e iOS.

Se usará la plataforma FIWARE, una iniciativa de código abierto desarrollado por Telefónica y Vodafone, para el desarrollo de soluciones Internet of Things (IoT) y otros componentes que se detallarán posteriormente para la recolección y almacenamiento de la información mencionada anteriormente.

Asimismo, se ha pensado en la creación de un cliente web para que encargados u otros trabajadores de la empresa puedan visualizar los datos y actuar de acuerdo con los mismos.

Como ya se mencionó antes, se busca la realización de un proyecto que pueda ser útil para un sector que goza de un papel de gran importancia en los últimos tiempos, además se profundizará en el uso de tecnologías como el IoT y las aplicaciones móviles híbridas, relativamente nuevas y con gran proyección a futuro.

[1] Framework: estructura, conformada por un conjunto básico de librerías, compiladores... que se usan como base en el desarrollo software.

1.2 Antecedentes

Este Trabajo se ha inspirado en los proyectos realizados por Leopoldo Ángulo Gallego y Sergio Mellado Contioso, en 2019 y 2020 respectivamente, llamados: *Control y Monitorización de hábitats para animales mediante Raspberry-Pi, Fiware, Spring y Android* [2] *y Aplicación Android y Servicio Web Spring para la monitorización de datos obtenidos en un vehículo haciendo uso de la plataforma FIWARE* [3]

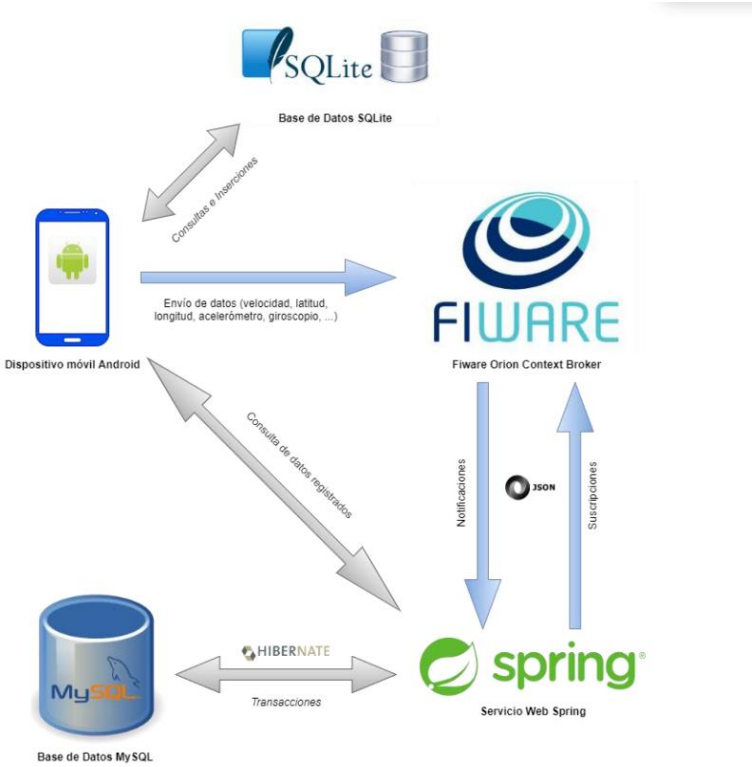

Ilustración 1: arquitectura del sistema de Sergio Mellado Contioso

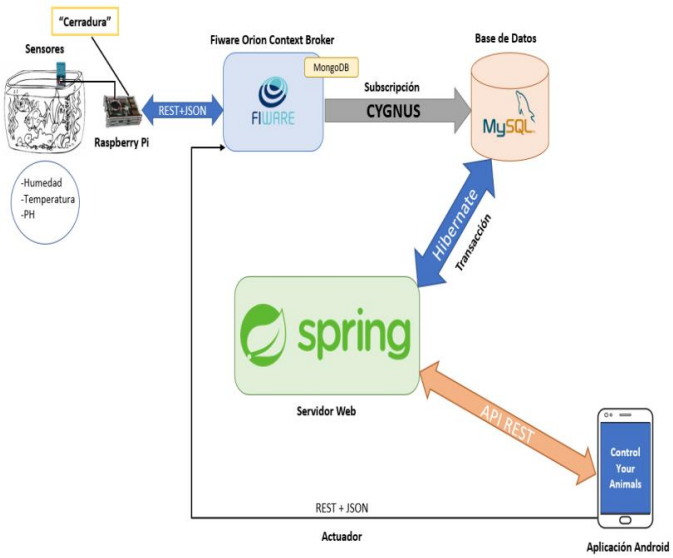

Ilustración 2: arquitectura del sistema de Leopoldo Ángulo gallego

Tomando como base el trabajo de Sergio Mellado Contioso, se añadirá la parte de Cygnus para almacenar el histórico, además se usará un cliente web, que usará la tecnología Node, que podrá ser utilizado por los encargados de la empresa para visualizar de una forma más visible e interactiva los datos de los trabajadores. El servicio web usará la tecnología Nodejs, otro framework para la creación de servicios REST, en vez de spring. Además, la aplicación estará desarrollada en React Native, que permite el desarrollo de aplicaciones

híbridas para Android e iOS.

1.3 Objetivos

El objetivo del Proyecto es que una empresa del sector de transporte, de reparto o de transporte de mercancías, sea capaz de controlar los movimientos de los vehículos y conductores que trabajen para ella. Para el envío de datos se utilizará una aplicación para smartphone que recogerá los datos de velocidad, posición etc. Cabe destacar que en el modelo de datos de FIWARE de "Vehicle", que será el utilizado para crear las entidades asignadas a cada vehículo de la empresa se pueden recoger otros muchos tipos de datos, como la capacidad y nivel del depósito de este, aunque para poder utilizar estos campos sería necesario la instalación de sensores en los vehículos.

FIWARE y algunos de sus componentes que se detallarán posteriormente, se usarán para almacenar mediante el uso del modelo publicador/suscriptor los datos obtenidos en una base de datos PostgreSQL.

Asimismo, se ha pensado en la creación de un cliente web para que encargados u otros trabajadores de la empresa puedan visualizar los datos y actuar de acuerdo con los mismos.

También se diseñará un servicio web REST. Encargado de la comunicación entre todos los componentes del sistema, gestión de información, suscripciones y usuarios.

1.4 Descripción de la solución

En este apartado se realizará un esbozo del proyecto para cumplir los objetivos establecidos anteriormente.

1.4.1 Objetivos específicos

- Identificación de usuario: Cada usuario dispondrá de una entidad "Vehicle", con número de matrícula y nombre. Por eso los trabajadores deberán identificarse con nombre de usuario y contraseña al abrir la aplicación.
- Obtención de datos del smartphone: los datos extraídos provendrán de los sensores de este, también se extraen y visualizan datos que no se envían ya que no existen en el modelo de datos de "Vehicle", como los de aceleración.
- Visualización de los datos del propio usuario: el usuario de la aplicación móvil podrá acceder a la ruta que ha realizado desde que se identificó, además gracias a la conexión con Open Street Maps comparará su propia velocidad con la máxima velocidad permitida en la vía, si existiese.
- Envío de la información recogida: se envía al Context Broker de FIWARE, que está ejecutándose en un contenedor Docker junto con Cygnus y PostgreSQL. El envío se realiza mediante un mensaje HTTP, en cuyo cuerpo los datos estarán en formato JSON. Cygnus está suscrito al Orion Context Broker de Fiware, por lo que cuando se actualice la entidad, Fiware notificará el cambio a Cygnus, que almacenará la información en una Base de Datos (BBDD) Postgresql.
- Consulta y visualización de los datos: se obtienen los datos almacenados en la BBDD PostgreSQL y se representan en el servicio web de varias formas posibles, se muestran las últimas posiciones registradas, la última ruta etc.
- Borrado de la información de trabajadores y/o los trabajadores: se eliminan los últimos registros de información de los usuarios, y también se podría borrar estos mismos. Se usa esta función para cambiar el vehículo asignado a los conductores.
- Registro de usuarios: los administradores, desde el cliente web, podrán añadir usuarios administradores y conductores, aunque no podrán eliminar a estos primeros. Tras rellenar los campos de información necesarios.

1.4.2 Funcionalidades

Las funcionalidades del Sistema serán las siguientes:

- Posibilidad de que un determinado usuario haga log in a la aplicación mediante la información de usuario guardada en una base de datos MySQL.
- Obtener los datos: velocidad, coordenadas, bearing[2]… de los sensores del dispositivo móvil.
- Envío de los datos con el uso de una tarea en segundo plano de la aplicación.
- Posibilidad de ver la ruta recorrida y la velocidad máxima de la vía por la que se está circulando en la app y mediciones del acelerómetro y giroscopio.
- Comparar la velocidad de la vía, si existiese, con la del vehículo.
- Actualización de la información de los conductores, que están guardados en la base de datos mongoDB del Orion Context Broker.
- Notificación a Cygnus de los cambios en las entidades mencionadas en el anterior punto.
- Almacenamiento de los nuevos datos en la base de datos PostgreSQL por Cygnus.
- Acceso a los datos almacenados anteriormente mediante el uso de un cliente web de uso exclusivo para los administradores.
- Permitir a los administradores registrar a nuevos trabajadores y administradores.
- Borrar los trabajadores y/o su información de ruta, además de asignarles un nuevo vehículo.
- Visualizar la información de los usuarios de distintas formas: última ubicación, ruta…

1.4.3 Esquema de la arquitectura

En la ilustración 6 se muestra el funcionamiento del sistema diseñado.

[2] Bearing: ángulo horizontal de un objeto con respecto al norte magnético.

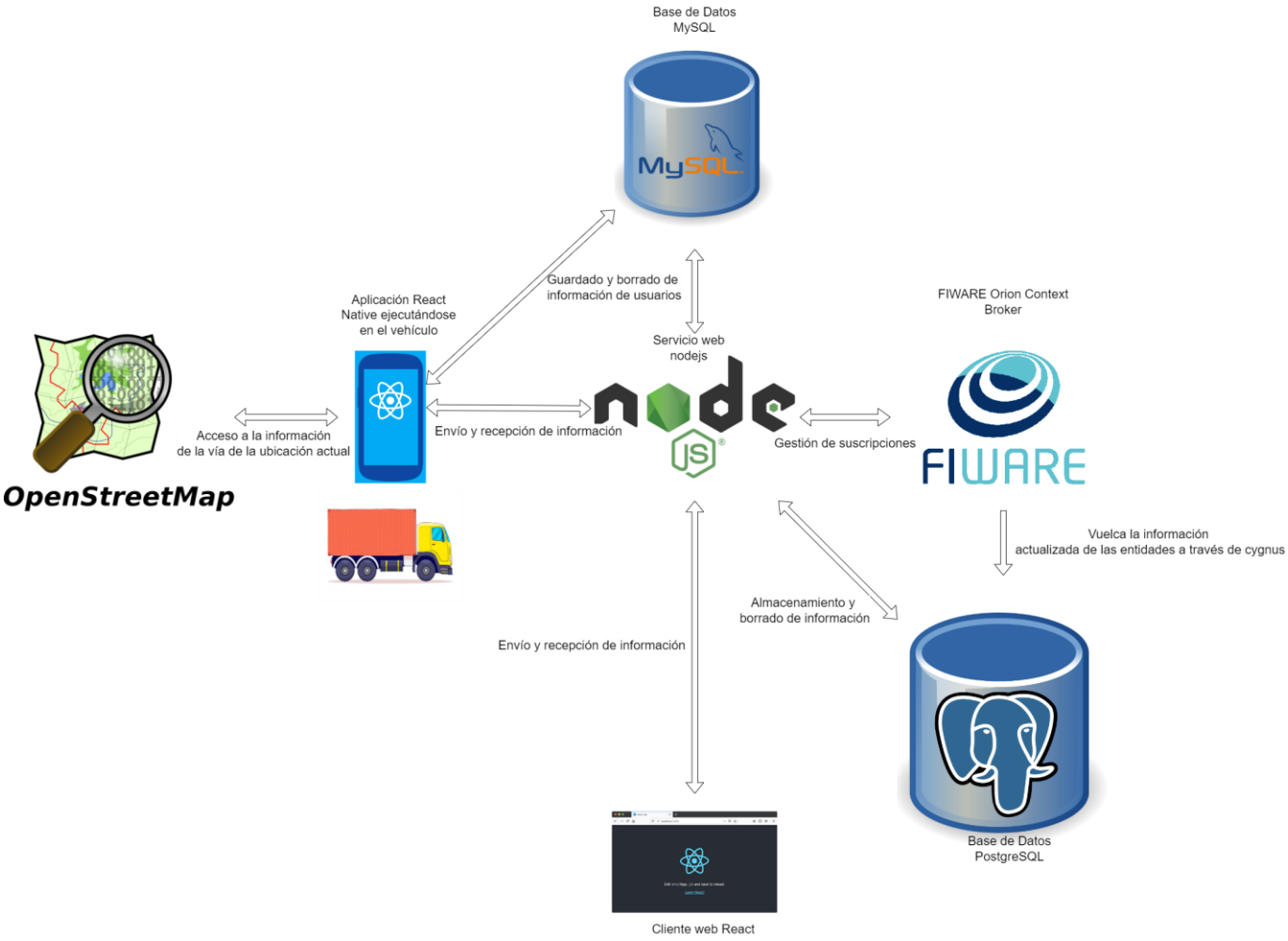

Ilustración 3: esquema del sistema

El smartphone, mediante la aplicación, realizará una llamada a la API de OpenStreetMap, enviándole sus coordenadas geográficas, y obtendrá si estuviese disponible la velocidad máxima de la carretera para compararla con la del vehículo. Asimismo, enviará el usuario y contraseña introducidos a la BBDD MySQL para iniciar sesión además de los datos obtenidos de sus sensores al servicio web nodejs, que reenviará los datos al Orion Context Broker de FIWARE, que, mediante el uso de Cygnus, un conector que usa el patrón suscripción/notificación, los almacenará en una BBDD PostgreSQL. En los trabajos en los que se ha inspira este proyecto, las peticiones de actualización al OCB, se enviaban directamente a este, sin embargo, al utilizar la BBDD MySQL, la cual dispone de la información de todos los conductores de la compañía, y al tener la necesidad de realizar operaciones sabiendo estos datos, resultaba mejor crear un servicio REST que crease una sola conexión, en el momento de ejecutarse, a la BBDD, y que realizase las operaciones de consulta, actualizado y borrado accediendo a los datos de esta BBDD.

Por último, el cliente web, a través del servicio web ya mencionado, será capaz de borrar los datos almacenados por FIWARE en la BBDD PostgreSQL y registrar nuevos usuarios, bien conductores o administradores del sistema, además de eliminar los primeros.

1.5 Estructura de la memoria

1. **Introducción**: se describe brevemente la solución descrita en la memoria, y se comentan la motivación, los antecedentes y objetivos de este, además de incluirse la estructura de la memoria.
2. **Recursos utilizados:** se enuncian y explican brevemente aquellos recursos hardware, software y online que se han utilizado en el proyecto.
3. **Tecnologías utilizadas:** se enuncian y explican las tecnologías empleadas en el desarrollo del

proyecto.

4. **Servidor REST y FIWARE:** se explica el servicio REST y cómo se ha empleado la plataforma FIWARE.
5. **Cliente web:** se muestra y explica la funcionalidad del cliente web React diseñado.
6. **Aplicación móvil:** se muestra y explica la funcionalidad de la aplicación móvil diseñada.
7. **Conclusiones y líneas futuras:** reflexión sobre el proyecto y sobre cómo podría ser mejorado en el futuro.

- Anexo A: instrucciones para la instalación y configuración de React Native
- Anexo B: instrucciones para la instalación y configuración de Visual Studio Code.
- Anexo C: instrucciones para la instalación y configuración de React.
- Anexo D: instrucciones para configurar VMware para que se pueda usar en este proyecto.
- Anexo E: instrucciones para la instalación, creación y configuración del servicio web.
- Anexo F: instrucciones para la instalación, configuración y ejecución de MySQL.
- Anexo G: instrucciones para la instalación de Docker y para ejecutar OCB y Cygnus en Ubuntu.

2 RECURSOS UTILIZADOS

Every successful hardware has a software behind

- Thiru Voona -

Van a detallarse los recursos utilizados en la realización del proyecto y las subsiguientes pruebas. Podría utilizarse otro tipo de hardware que sea capaz de ejecutar el sistema completo, ya que solo se describe el entorno de desarrollo empleado.

2.1 Recursos hardware

En este apartado se va a describir el hardware usado en el proyecto.

2.1.1 Ordenador portátil Intel Core i7-105010U

En un ordenador portátil con las siguientes características se ha ejecutado, en Windows: el cliente web y es necesario instalar la aplicación en el móvil vía ventana de comandos. Se ha usado la máquina virtual Ubuntu para ejecutar todas las BBDDs, MySQL y PostgreSQL, el servicio web y el Orion Context Broker junto con Cygnus.

- Memoria RAM: 8 Gb
- Almacenamiento: 512 Gb SSD
- Sistema operativo: Windows 10 Pro

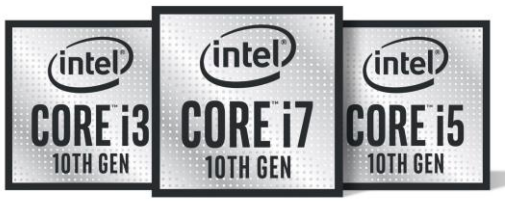

Ilustración 4: Intel Core 10º generación.

2.1.2 Smartphone

En el proyecto podría utilizarse cualquier otro tipo de dispositivo Android o iOS, aunque el usado en el desarrollo del proyecto ha sido el siguiente:

- Modelo: Samsung Galaxy S10 Lite.
- Sistema operativo: Android 12.
- Procesador: Qualcomm SM8150 Snapdragon 855 octa-core 2.84 Ghz.
- Memoria RAM: 8 Gb.
- Almacenamiento: 128 Gb.
- Procesador gráfico: Adreno 640.
- GPS: GPS con soporte A-GPS, BDS y Galileo.
- Sensores: proximidad, luz ambiente, giroscopio, brújula digital y acelerómetro

Ilustración 5: imagen Samsung Galaxy S10 Lite

2.2 Recursos software

2.2.1 Sistema Operativo Windows 10 Pro.

Sistema operativo (SO) creado por Microsoft que supone de facto el estándar para ordenadores domésticos y de negocios. Además del uso del cmd que viene integrado en el mismo, y de algunos paquetes instalables como nodejs y npm, se utilizarán los siguientes recursos no nativos a Windows:

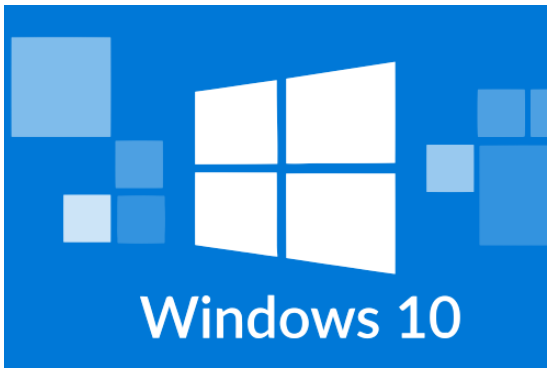

Ilustración 6: logo de Windows 10

2.2.1.1 Navegador Google Chrome

Navegador empleado a la hora de visualizar el cliente web creado. Sobre todo, se ha utilizado las Chrome Devtools, mediante el atajo ctrl + shft + J, que constituye un conjunto de herramientas gratuitas proporcionadas por Google para el desarrollo web. [4]

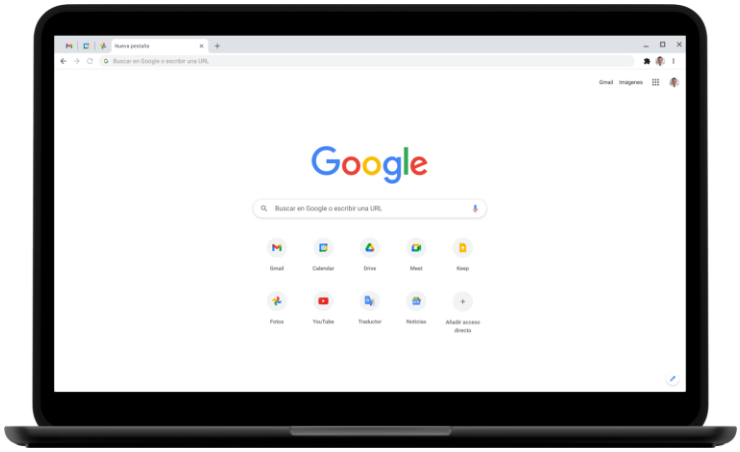

Ilustración 7: navegador Google Chrome

2.2.1.2 Visual Studio Code 1.68.1

IDE[3] desarrollado por Windows para Microsoft y Linux que se ha utilizado a la hora de codificar en REACT y REACT NATIVE, se han instalado varias extensiones para ambos lenguajes para hacer la codificación de forma más cómoda. Aunque en su instalación por defecto no cuente con todas las herramientas necesarias para ser considerado un IDE (depurador, compilador…), mediante la instalación de extensiones puede conseguirse esa funcionalidad. IDE más popular de acuerdo con la encuesta de popularidad de IDEs en Stack Overflow en 2021 [5].

[3] IDE: entorno de desarrollo integrado, herramienta software que aporta distintas herramientas para editar código, compilar, depurar errores etc.

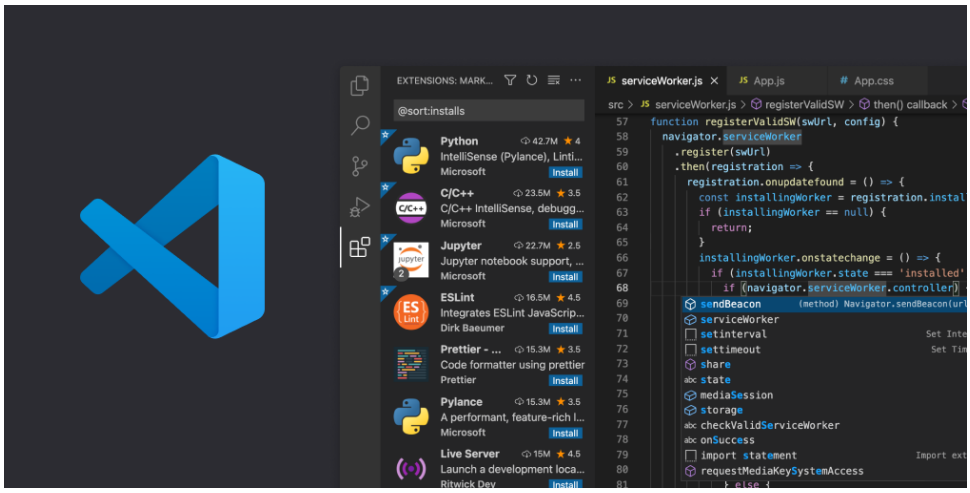

Ilustración 8: imagen y logo de Visual Studio Code

2.2.1.3 Android Studio 2020.3.1

IDE oficial para el desarrollo de aplicaciones Android, aunque en este caso solo se ha utilizado su emulador para crear un AVD[4] y para la selección del dispositivo móvil para la depuración de la aplicación, ya que como se comentó anteriormente se usó Visual Studio Code para la codificación.

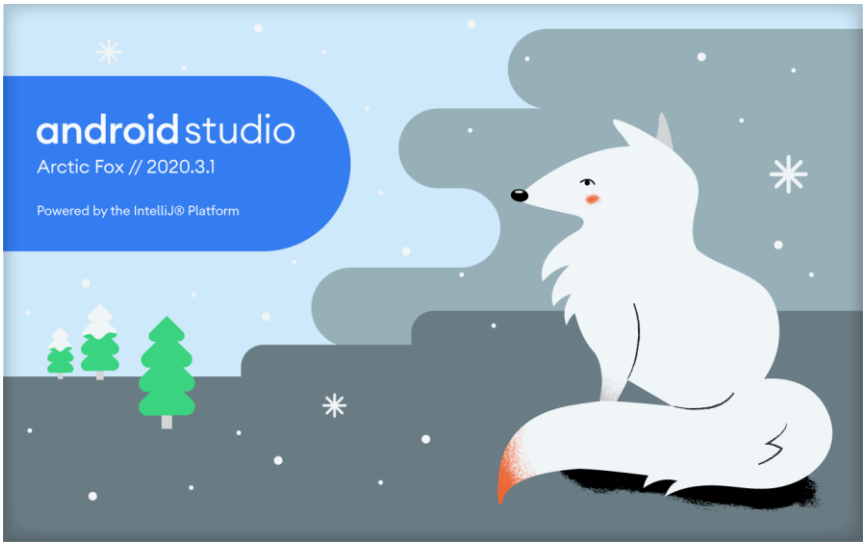

Ilustración 9: logo de Android Studio 2020.3.1 (Arctic Fox)

2.2.2 Sistema Operativo Ubuntu 20.04 LTS.

Se ha utilizado el sistema operativo Ubuntu 20.04 LTS, mediante el uso de la máquina virtual (VM) proporcionada por el departamento de Ingeniería Telemática de la Universidad de Sevilla. Se ha usado VMWARE Worskstation 16 player para virtualizar la misma, dentro de esta se han utilizado las siguientes herramientas:

[4] AVD: dispositivo virtual de Android, dispositivo Android simulado en el emulador de Android Studio.

Ilustración 10: logo Ubuntu 20.04

2.2.2.1 Postman

Herramienta que permite el envío de peticiones HTTP a una dirección y puertos a elegir, además de visualizar la respuesta enviada por el servidor. Se ha utilizado en la depuración del servicio web y de FIWARE, ya que en el proyecto son las propias aplicaciones web y móvil la que realizan estas peticiones, con su uso se pueden ver las entidades guardadas en FIWARE, borrarlas, añadirlas… y otras muchas funciones.

Ilustración 11: logo Postman

2.2.2.2 Wireshark

Uno de los analizadores de paquetes más populares, gratuitos y disponible en multitud de plataformas: Windows, Linux, Android… Usado en la depuración para ver el contenido y cabeceras de los paquetes HTTP intercambiados entre el servicio web y/o las aplicaciones móviles y web.

Ilustración 12: logo Wireshark

2.2.2.3 Eclipse

IDE empleado en la programación del servicio web REST con Nodejs y Express. Uno de los IDEs más populares en la actualidad.

Ilustración 13: logo Eclipse

2.3 Recursos online

Conjunto de herramientas online y gratuitas, aunque algunas requieren de registro para el dibujo de esquemas del proyecto.

2.3.1 app.diagrams.net

Web que permite dibujar diagramas variados, UML, esquemas etc.

2.3.2 online.visual-paradigm.com

Herramienta utilizada únicamente para dibujo de casos de uso.

2.3.3 dbdiagram.io

Herramienta que permite la realización de esquemas de BBDD.

3 TECNOLOGÍAS UTILIZADAS

Technology is best when it brings people together

- Matt Mullenweg -

3.1 Node.js

Nodejs fue ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, diseñado para la creación de aplicaciones webs escalables. Por cada conexión recibida por Node.js se activa un callback[5], sin el uso de hilos. HTTP es elemento destacado es Node.js, ya que está diseñado para la transmisión de operaciones con streaming[6] y latencia baja [6]. Es necesario tenerlo instalado para la creación de la aplicación móvil React-Native y además Se usará Express.js, el framework estándar de facto para el desarrollo mediante Node.js, para el desarrollo del servicio web.

Ilustración 14: logo node.js

3.1.1 Express.js

Express.js o Express, es un framework para Node.js minimalista y flexible para el desarrollo de aplicaciones móviles y web, aportando métodos de utilidad HTTP y middleware[7] a disposición del usuario [7]. Debido a la cantidad y utilidad de los métodos, ya mencionados anteriormente, provistos por express, se usará para la creación de un servicio web que siga la arquitectura REST, es decir, que use mensajes HTTP para el envío y uso de datos, para ser llamado etc.

[5] Callback: referencia a código ejecutable pasada a otra parte del código.
[6] Streaming: flujo constante de datos desde otro dispositivo, con poco o ningún almacenamiento de datos.
[7] Software que reside entre el Sistema Operativo y las aplicaciones ejecutadas por éste.

Ilustración 15: logo express.js

3.1.2 Npm

Mayor registro de software del mundo y gestor de paquetes por defecto de Node.js [8], se usará para instalar todos y cada uno de los paquetes necesarios para el desarrollo de las aplicaciones web y móvil. Se puede usar para la instalación de funciones o etiquetas JSX que se usarán en las aplicaciones.

Ilustración 16: logo npm

3.2 FIWARE

FIWARE es un software de código abierto que combina componentes y arquitecturas para permitir la conexión de información de contexto IoT junto con servicios de Big Data[8]. FIWARE usa unos modelos de datos estandarizados de facto en su programa "Smart Data Models", para garantizar la compatibilidad entre plataformas. El propósito de FIWARE es que sea fácil de conectar y usar con soluciones de terceros [9]. En el proyecto se usarán los siguientes componentes de FIWARE.

Ilustración 17: logo FIWARE

3.2.1 Orion Context Broker

Principal y único componente obligatorio de cualquier plataforma o solución desarrollada con FIWARE. El Orion Context Broker (OCB) aporta una función fundamental en cualquier solución inteligente: administrar la información de contexto, consultarla y actualizarla.

El OCB permite la publicación de información de contexto por entidades productoras de contexto, como por

[8] Big data: conjunto de información tan grande como para que el software tradicional no pueda lidiar con él.

ejemplo sensores, de manera que la información se mantenga disponible para otras entidades, consumidoras de contexto, como una aplicación. Las entidades pueden ser cualquier tipo de aplicación o incluso otro componente dentro de la plataforma FIWARE.

El OCB permite distintos tipos de funcionalidades, registrar entidades creadoras de contexto, actualizar los datos de estos en un determinado periodo, o consultar la información guardada.

Cabe destacar que el OCB está siempre escuchando en un puerto, generalmente 1026, y almacena la última información en una BBDD MongoDB, por lo que carece del histórico de datos de la información [10].

Se utilizará para recoger los datos del GPS del móvil y almacenarlos en entidades independientes para cada usuario.

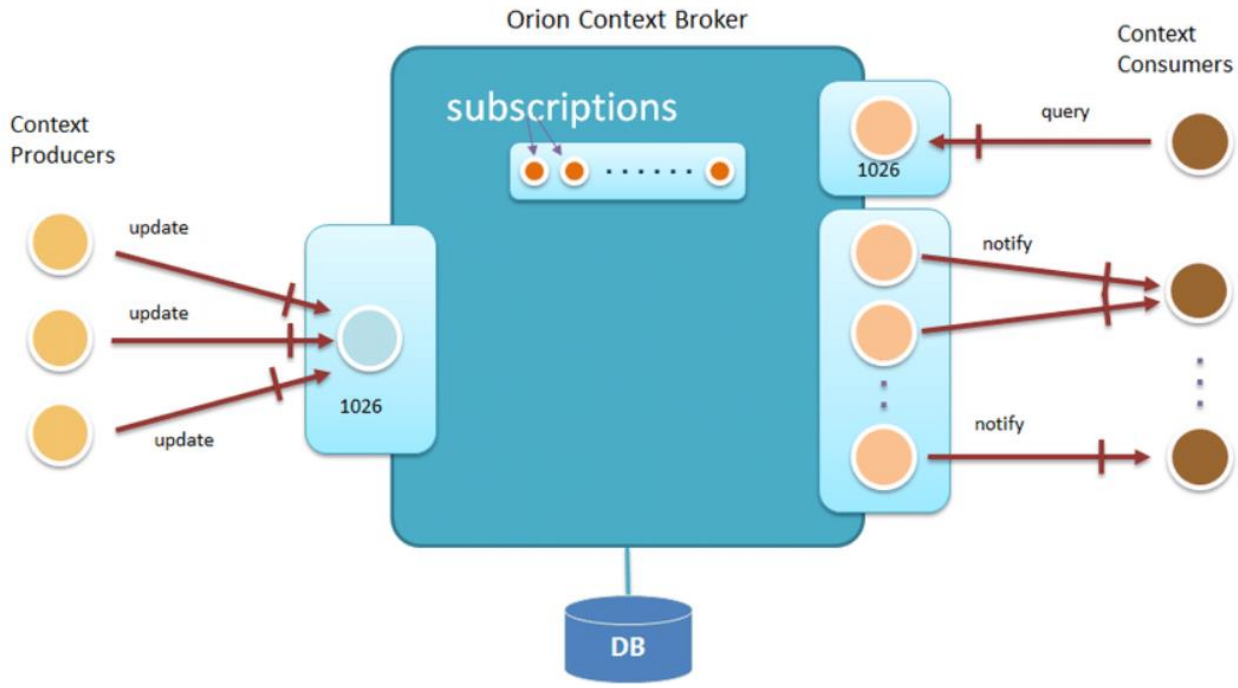

Ilustración 18: diagrama de funcionamiento de OCB

3.2.2 Cygnus

Como ya se ha mencionado anteriormente, el OCB carece del histórico de datos de sus entidades, para esto se debe usar una BBDD externa al mismo.

Cygnus es un módulo parte de FIWARE, del ecosistema Cosmos. Es un conector a cargo de ciertas fuentes de información, en almacenamientos de terceros. Cygnus está basado en Apache Flume, tecnología que aborda el diseño y ejecución de colecciones de datos y agentes de persistencia. Un agente se compone de un listener o fuente a cargo de recibir la información, un canal donde la fuente coloca la información después de convertirlo a un evento Flume y un sumidero que recibe el evento Flume con el fin de que el histórico persista en un almacenamiento de terceros. Cygnus actualmente puede almacenar su información en numerosas BBDD, aunque en este caso se ha utilizado PostgreSQL. [11]

En el proyecto, el componente Cygnus, siguiendo el patrón suscriptor-publicador, se suscribirá a todos y cada uno de los cambios de todas las entidades de un determinado contexto de OCB, por lo que cada actualización de estas se guardará en la BBDD PostgreSQL por Cygnus.

En la siguiente imagen se muestra el funcionamiento general de una implementación de Cygnus y el OCB.

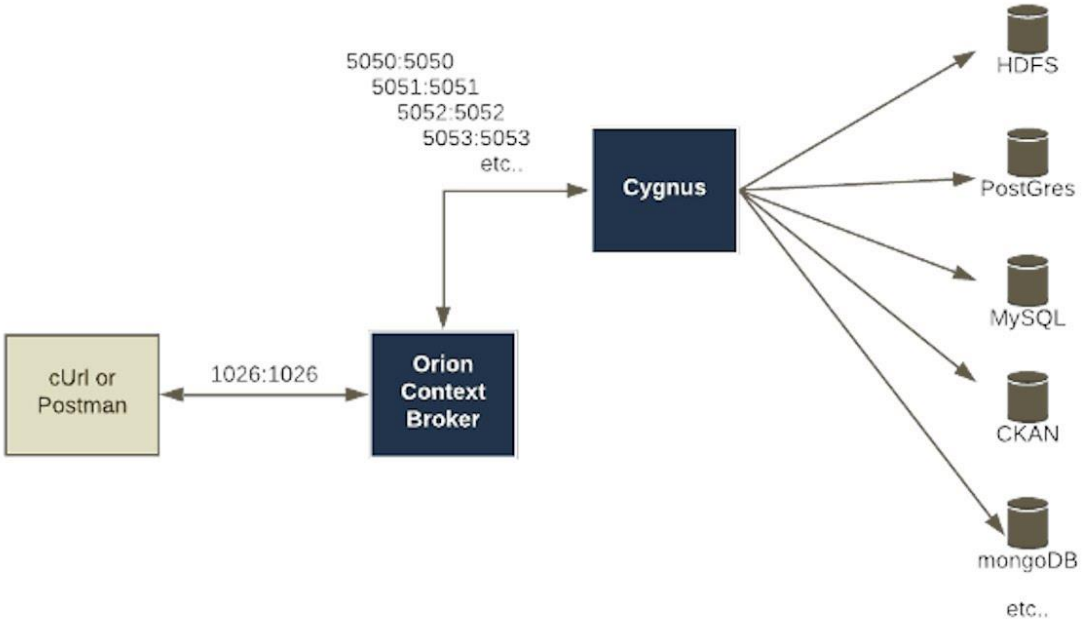

Ilustración 19: arquitectura de sistema con Cygnus implementado.

3.3 Bases de datos

A continuación, se detallarán las BBDD que han intervenido en el proyecto.

3.3.1 MySQL

BBDD relacional de código abierto [12] que se encargará de almacenar los datos de información de usuario y administrador, documentos de identificación, nombres y apellidos, matrículas de los vehículos etc. Es posible borrar o alterar sus datos a partir del cliente web.

3.3.2 MongoDB

BBDD no relacional que almacena el último valor de cada entidad registrada en el OCB en formatos flexibles y similares a JSON, es de uso público y licencia AGPL[9]. [13]

3.3.3 PostgreSQL

BBDD relacional gratuita y de código abierto [14] que se utilizará para almacenar los datos históricos de las entidades productoras de contexto. De su configuración, ejecución y del almacenamiento en esta será el propio Cygnus el que se encargue. Para que Cygnus sea el que lidie con esta tarea, habrá que configurarlo via Docker.

3.4 Docker

Plataforma que permite la ejecución en un entorno controlado y aislado el cual se llamará container. El aislamiento entre containers permite la ejecución de varios containers al mismo tiempo. Los containers son ligeros y contienen todo lo necesario para la ejecución de este, sin depender de ningún tipo de software integrado en el equipo que ejecute Docker. [15]

En el trabajo se ejecutará Docker en la VM de Ubuntu, a través de un fichero el cual estará configurado

[9] AGPL: licencia derivada de la GPL que obliga a la distribución del código cuando este sea usado para dar un servicio.

debidamente para permitir la instalación y ejecución de los containers de Cygnus, en OCB y PostgreSQL, así como la conexión entre los mismos. Para conectarlos es necesario el uso de reglas en las que se indique a qué container y en qué puerto será la conexión. En la siguiente ilustración se muestra el funcionamiento de la plataforma Docker.

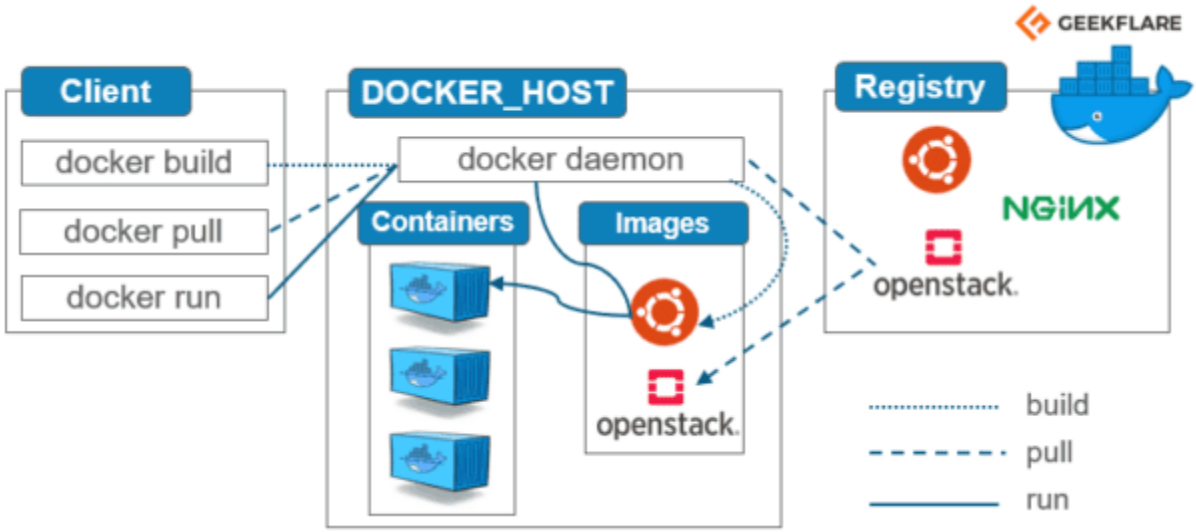

Ilustración 20: diagrama de funcionamiento de Docker

3.5 React

Biblioteca de JavaScript de código abierto para la creación de interfaces creado por Facebook. React diseña vistas para cada estado de la aplicación y actualiza y renderiza la aplicación de forma eficiente. Combina los componentes JSX junto con el código, aunque no es necesario. Puede renderizarse desde el servidor usando el ya mencionado Node.js.

3.6 Metro

Metro es un bundler, o empaquetador, para Javascript, es decir una herramienta que agrupa un fichero Javascript y sus dependencias para devolver un único fichero Javascript, su principal propósito es hacer más fácil la experiencia del desarrollador. Las 3 fases del proceso seguido por el bundler son:

- Resolución: Metro obtiene un gráfico con todos los módulos necesarios para el empaquetado.
- Transformación: todos los módulos son convertidos a un formato entendible por la plataforma deseada, en este caso, React Native,
- Serialización: tras transformar los módulos se combinan en un único fichero JavaScript.

Ilustración 21: logo de Metro

3.7 React Native

Framework basado en Javascript para aplicaciones móviles que permite crear aplicaciones móviles para iOS y Android usando el mismo código. Creado por Facebook y basado en React, React Native no renderiza componentes JSX como React, sino que renderiza componentes nativos a las aplicaciones móviles como vistas. Aun así, el código sigue estando basado en JavaScript, pero en este caso será empaquetado por Metro.

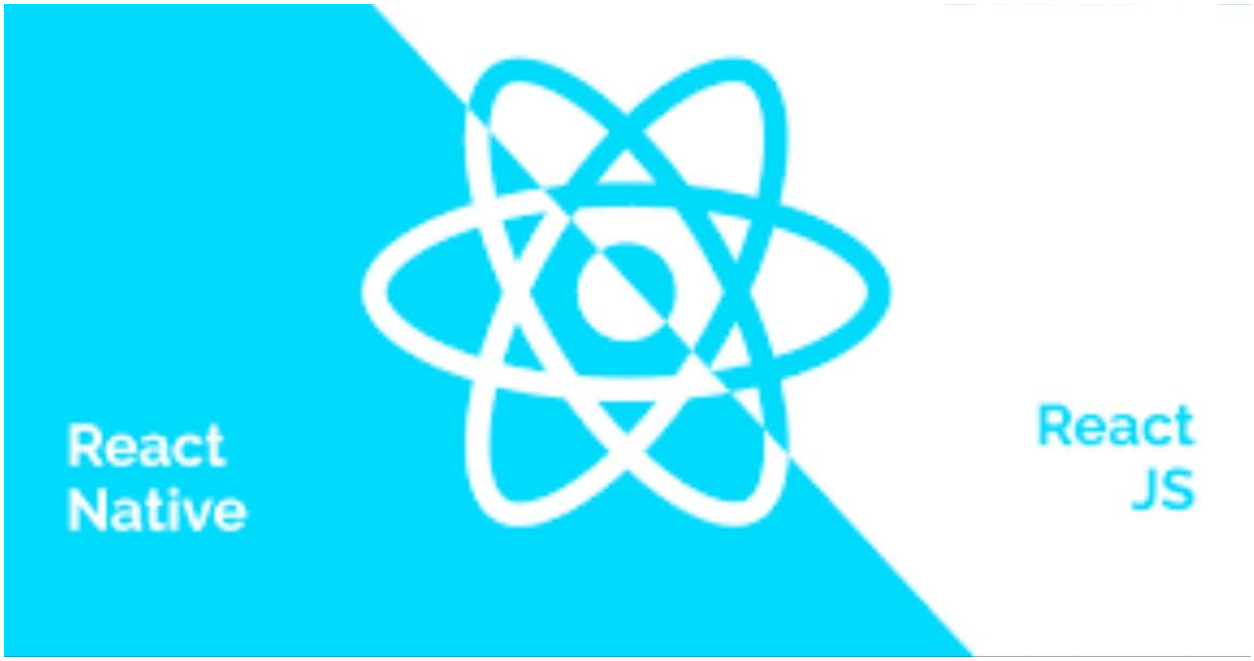

Ilustración 22: logos de React y React Native

3.8 OpenStreetMap

La mayor colección de mapas gratuitos del mundo. OpenStreetMap (OSM), es un proyecto colaborativo en el que son los propios usuarios los que obtienen y añaden información cartográfica de todo el planeta, también incluye información de carreteras, negocios, imágenes aéreas… y otras funcionalidades como cálculo de rutas óptimas. OSM usa una estructura de datos topológica en la que a cada ubicación geográfica se le asigna un nodo, estos nodos pueden agruparse en relaciones, vías y ser etiquetados. OSM proporciona 3 herramientas para usar y aprovechar sus datos:

- Leaflet: biblioteca de JavaScript que permite incluir mapas deslizantes en aplicaciones y páginas webs, que puede ser usada por React y React Native. No es propia de OSM, sino que es desarrollada por un tercero. Usada para dibujar los mapas en React.

- API[10] de Overpass: API de solo lectura que permite el acceso a todos los datos de OSM, utiliza su propio lenguaje de consultas (Overpass QL), basado en XML, el cual se encapsula en un mensaje HTTP. Overpass QL usa todo tipo de operadores y bloques básicos. Se utiliza en la aplicación móvil para, enviando las coordenadas del dispositivo, obtener la información de la vía si estuviera disponible.
- Planet.osm: herramienta para obtener toda la información almacenada en OSM, en torno a 740 Gb. Única herramienta no usada en el proyecto.

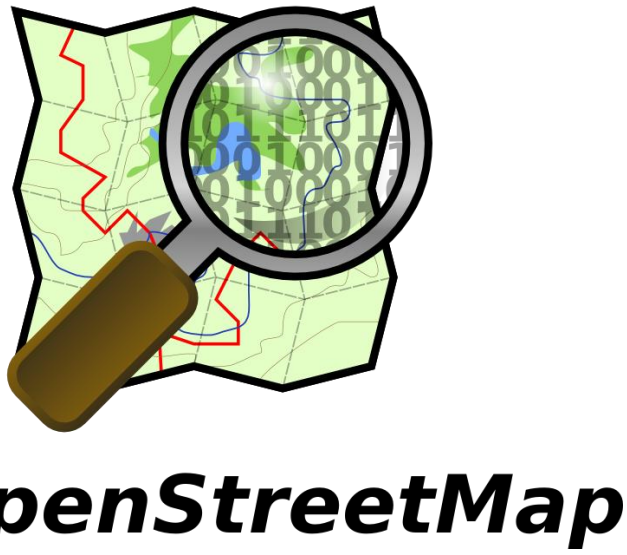

Ilustración 23: logo OpenStreetMap

[10] API: conjunto de protocolos diseñados para ser utilizados fácilmente desde otro software.

4 SERVIDOR REST Y FIWARE

When you reduce the cost of failure, you see a big eruption of innovation

- Gary Hauthier -

En este capítulo se describirán los elementos del lado servidor del proyecto. Los componentes serán: FIWARE y sus componentes, las bases de datos del sistema y el servicio web REST.

4.1 SERVICIO WEB REST

El servicio web REST diseñado será el de múltiples tareas, que se pueden generalizar en dos tipos. Aquellas tareas en las que el servicio accede y/o altera una BBDD, enviando una respuesta al cliente web o a la aplicación móvil, y las de actualizar los datos almacenados en el Orion Context Broker y Cygnus.

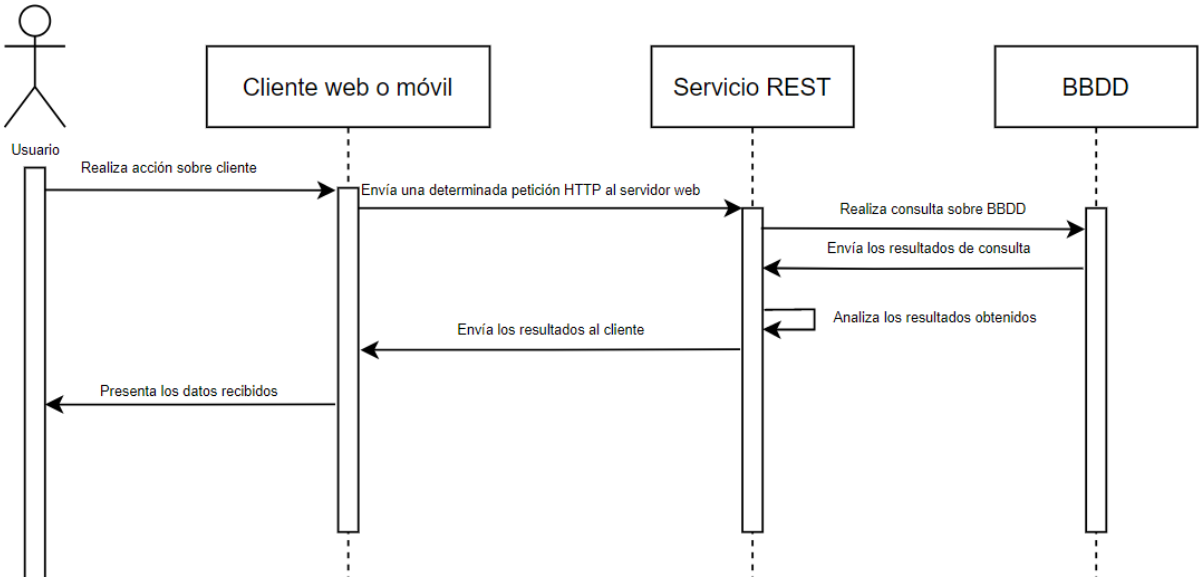

Ilustración 24: diagrama de secuencia de consulta a BBDD

En el diagrama de secuencia de la ilustración 24 se observa sobre líneas generales el funcionamiento del sistema cuando el usuario desea realizar una consulta (selección, eliminación…) a la BBDD.

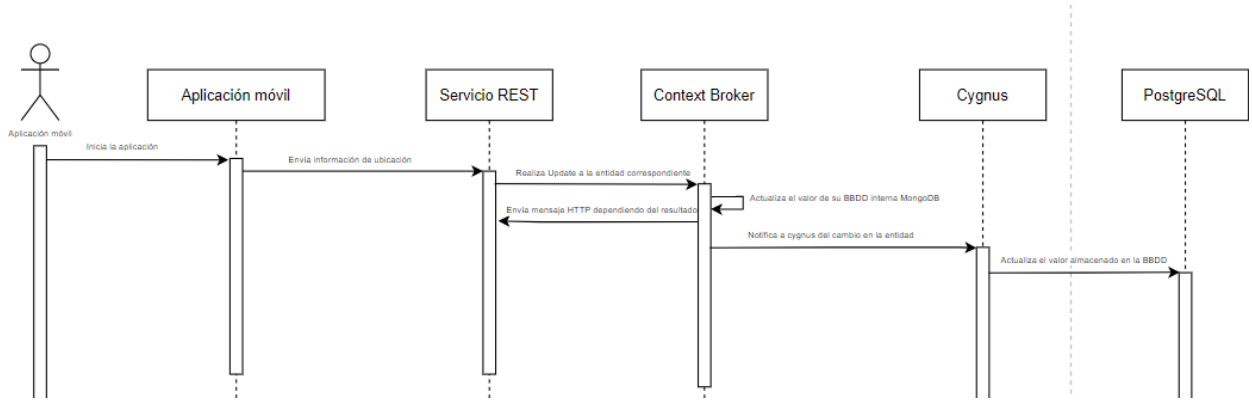

Ilustración 25: diagrama de secuencia de un envío de los datos de ubicación de un conductor

En este otro diagrama de secuencia de la ilustración 25 se describen las interacciones ante una actualización de los datos de un conductor.

Como todo servicio REST, dependiendo de la ruta a la que se dirija el mensaje HTTP y su tipo, la tarea del servicio cambiará. Ahora se procederá a describir todas las funcionalidades y rutas del servicio. El servicio web está localizado en una dirección IP propia privada: 192.168.1.61:3000, la configuración de la máquina virtual se mostrará en el anexo.

Url	**Método**	**Parámetro**	**Funcionalidad**
IP_máquina:puerto/loginadmin	POST	Nickname usuario y contraseña.	Permite entrar a la plataforma a un conductor.
IP_máquina:puerto/loginusu	POST	Nickname administrador y contraseña.	Permite entrar a la empresa a un gesto de la empresa.
IP_máquina:puerto/registeradmin	POST	Nickname, nombre, apellidos, documento de identificación y contraseña de usuario.	Registra un administrador.
IP_máquina:puerto/registerusu	POST	Nickname, nombre, apellidos, documento de identificación, matrícula, sede y contraseña de usuario.	Registra un usuario.
IP_máquina:puerto/registersede	POST	Nombre de la sede, latitud, longitud.	Registra una sede.
IP_máquina:puerto/deletesede	POST	Nombre de la sede.	Elimina una sede.
IP_máquina:puerto/deleteusu	POST	Nickname usuario.	Elimina un usuario.
IP_máquina:puerto/deleteinfo	POST	Nickname usuario.	Elimina un usuario y toda su información.
IP_máquina:puerto/updateentity	POST	Objeto location y nickname de usuario.	Actualiza la información del usuario.
IP_máquina:puerto/getubis	GET	Ninguno.	Obtiene las últimas ubicaciones de todos

			los usuarios
IP_máquina:puerto/getroutes	POST	Nickname usuario.	Obtiene la ruta de un usuario.
IP_máquina:puerto/getroutesdate	POST	Nickname usuario, fecha inicio y fecha final.	Obtiene las rutas entre 2 fechas elegidas.
IP_máquina:puerto/ususlist	GET	Ninguno	Obtiene toda la lista de usuarios.
IP_máquina:puerto/deletedates	POST	Nickname usuario, fecha inicio y fecha final	Borra la ruta de usuario entre 2 fechas elegidas.

Tabla 1: API del servicio web desarrollado

Como apunte, Nodejs usa CORS[11], por lo tanto, es posible que al usar un navegador a este no se le permita acceder a los recursos del recurso web. Por eso, para el funcionamiento del servicio web es necesario añadir las siguientes líneas.

```
const cors = require('cors')

app.use(function(req, res, next) {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
  next();
});
const corsOptions = {
  origin: true,
  credentials: true
}

app.options('*', cors(corsOptions));
```

En este código se especifica que se deben aceptar todo tipo de orígenes y cabeceras HTTP, además de incluir una ruta en caso de que la aplicación reciba un mensaje HTTP de tipo OPTIONS, el cual ocurrirá cuando se use CORS y el navegador "pregunte" al servicio si acepta una conexión.

Las variables del sistema que no deban estar dentro del código como las solicitudes para el OCB o la información de la BBDD, se encuentra en un fichero aparte que se importa en el código del servicio.

4.2 FIWARE

En el trabajo se han utilizado dos componentes de FIWARE, Orion Context Broker y Cygnus, en cuyo

[11] CORS, Cross-Origin Resource Sharing: mecanismo que se usa para indicar de qué dominios aparte del propio, se aceptan las peticiones y el tipo de estas.

funcionamiento se profundizará a continuación.

4.2.1 Orion Context Broker

Como ya se mencionó en la sección de tecnologías utilizadas, el Orion Context Broker (OCB), almacena los últimos valores de entidades publicadoras de contexto en su BBDD propia, una MongoDB, mientras que envía la información a entidades consumidoras de contexto, en este caso Cygnus, que será la encargada de almacenar el histórico de datos de cada entidad.

Para almacenar la información de las entidades, FIWARE dispone de un conjunto de modelos de datos inteligentes a seguir que usan el esquema JSON. El propósito de estos modelos es que todas las soluciones que usen FIWARE sean interoperables, aunque da la opción a cambiarlos e incluso a publicarlos para que se conviertan en parte del estándar de modelos en el futuro. En nuestra solución se usará el modelo de datos Vehicle, aunque en vez de representar los vehículos en sí, lo hará con los conductores. En el caso de que un conductor cambie de vehículo, el número de matrícula almacenado cambiará. Tampoco se utilizarán todos y cada uno de los campos del modelo, solo los más relevantes. Entre los campos existentes están localización, altitud, velocidad… [16].

Los datos de las entidades se envían via HTTP, en cuyo body se almacenan en el formato anteriormente comentado la información. Sin embargo, en las cabeceras del mensaje, debe especificarse que el contenido enviado y aceptado es de formato JSON, además, el OCB permite multi tenancy[12], esto se logra mediante los campos en la cabecera Fiware-Service y Fiware-Servicepath, cuando una entidad se crea, actualiza, o consulta, el OCB solo afectará a las entidades que compartan los campos anteriormente descritos. En nuestro ejemplo Fiware-Service y Fiware-ServicePath tienen los valores “cygnusexample” y “/” respectivamente.

El OCB acepta las peticiones HTTP de tipo GET, PUT, POST, DELETE, OPTIONS, HEAD, TRACE y CONNECT. Aunque en el Proyecto desarrollado las usadas serán las presentes en la tabla 1:

POST	Tipo de mensaje utilizado para la creación de una entidad.
PATCH	Tipo de mensaje usado para la actualización de los valores de una entidad.
GET	Tipo de mensaje usado para la obtención de los últimos valores de cierta entidad, o de las sucripciones del OCB.
DELETE	Tipo de mensaje usado para eliminar una entidad o suscripción.

Tabla 2: tipos de mensajes HTTP y acciones sobre OCB

Cabe destacar que, aunque se usen estos 4 tipos de mensajes, la acción que haga el OCB ante estos variará dependiendo de la ruta usada, pudiendo alterar suscripciones, entidades etc.

Como se comentó anteriormente, Cygnus es una entidad consumidora de contexto, es decir debe recibir la información que se almacene en el OCB, por lo tanto la mejor opción es usar el patrón publicador/suscriptor, esto es tan fácil como realizar una petición HTTP mediante POSTMAN, a la dirección: http://localhost:1026/v2/subscriptions/, con las cabeceras elegidas para que el Fiware-Service y Fiware-ServicePath sean los deseados y estos valores en el cuerpo:

[12] Multi tenancy: tipo de arquitectura software en el que una sola instancia de una o varias aplicaciones pueden usarse para varios usuarios distintos.

```
{
  "description": "Notify Cygnus of all context changes",
  "subject": {
    "entities": [
      {
        "idPattern": ".*"
      }
    ]
  },
  "duration": “P1M”,
  "notification": {
    "http": {
      "url": "http://cygnus:5055/notify"
    },
    "attrsFormat" : "legacy"
  },
  "throttling": 5
}
```

Este cuerpo indica al OCB de que debe notificar a Cygnus de los cambios de contexto de las entidades con cualquier patrón de id, con una duración de 1 mes, la dirección a la que notificar, el formato de los atributos soportados y los segundos que deben pasar entre notificaciones, en este caso 5 segundos, que es un periodo menor del tiempo en el que la aplicación móvil envíe las actualizaciones. Se añade una descripción de la suscripción que se verá cuando se acceda a esta mediante un GET.

4.2.2 Cygnus

El funcionamiento de Cygnus ya se ha explicado en las secciones anteriores. Tras ser desplegado, este se encargará de acuerdo con su configuración de almacenar el histórico de datos en una BBDD a elegir, en este caso, MYSQL.

No obstante, durante la realización de un proyecto se ha encontrado un bug a la hora de usar Cygnus junto con el OCB.

La primera ocasión que se crea una entidad en el OCB, este notifica a Cygnus de la creación de esta y Cygnus crea una tabla propia para esa entidad. Sin embargo, cuando la entidad es eliminada, la BBDD no es alterada, y si se borrase manualmente la tabla y posteriormente se crease una nueva entidad en el OCB con la misma id que la eliminada anteriormente, OCB intentaría actualizar los datos de la tabla que ya se ha eliminado de la BBDD PostgreSQL, lo cual causaría un error en la misma.

4.3 Bases de datos

En esta sección se presentarán diagramas de las BBDDs y sus relaciones, además de comentar el propósito y funcionamiento de cada uno. Asimismo, se obviará la BBDD MongoDB pues es propia de OCB y no se accede o controla esta durante el proyecto.

4.3.1 MySQL

Ilustración 26: diagrama de la base de datos MySQL

En el diagrama de bases de datos de la ilustración 26 se muestra el esquema de la BBDD MySQL, se muestran 3 tablas simples en las que se guardan la información de los administradores, usuarios y las sedes. Las claves de las tablas admins y usus son las identificaciones de cada uno, y de las sedes su nombre, además también existe una relación entre la tabla usus y las sedes en la que el nombre de la sede es clave externa de la tabla usus con una relación una a muchos, es decir el nombre de una sede puede ser múltiples sedes de un usuario.

MySQL soporta multi tenancy, ya que con una sola instancia de la aplicación pueden crearse y usarse varias bases de datos completamente independientes en paralelo. La única restricción es que debe indicarse el nombre de la BBDD a usar en la conexión de MySQL. En este caso, la BBDD usada será “my_database” y la conexión se realizará en el inicio del servicio REST.

4.3.2 PostgreSQL

Ilustración 27: diagrama de la base de datos PostgreSQL

En el diagrama de la ilustración 27 se muestra cómo son las tablas para cada usuario en Cygnus, usux es un nombre cualquiera de usuario. Se crea una entrada en la misma para cada uno de los atributos presentes, localización, nombre, velocidad… los que se actualizarán gracias a la información de los sensores móviles. Además, también se incluyen en la tabla el fiwareservicepath, además del fiwareservice, que está presente en el nombre de esta.

5 CLIENTE WEB

The right server for the right job

- Bob Muglia -

En esta sección serán descritas las funcionalidades del cliente web desarrollado para el proyecto. El cliente está orientado a los administradores de la empresa, para que puedan visualizar y borrar los datos de los conductores, además también podrá borrar y añadir nuevas sedes, usuarios etc.

5.1 Diagrama de casos de uso

En la ilustración 28 se mostrará un diagrama de casos de uso de la solución web.

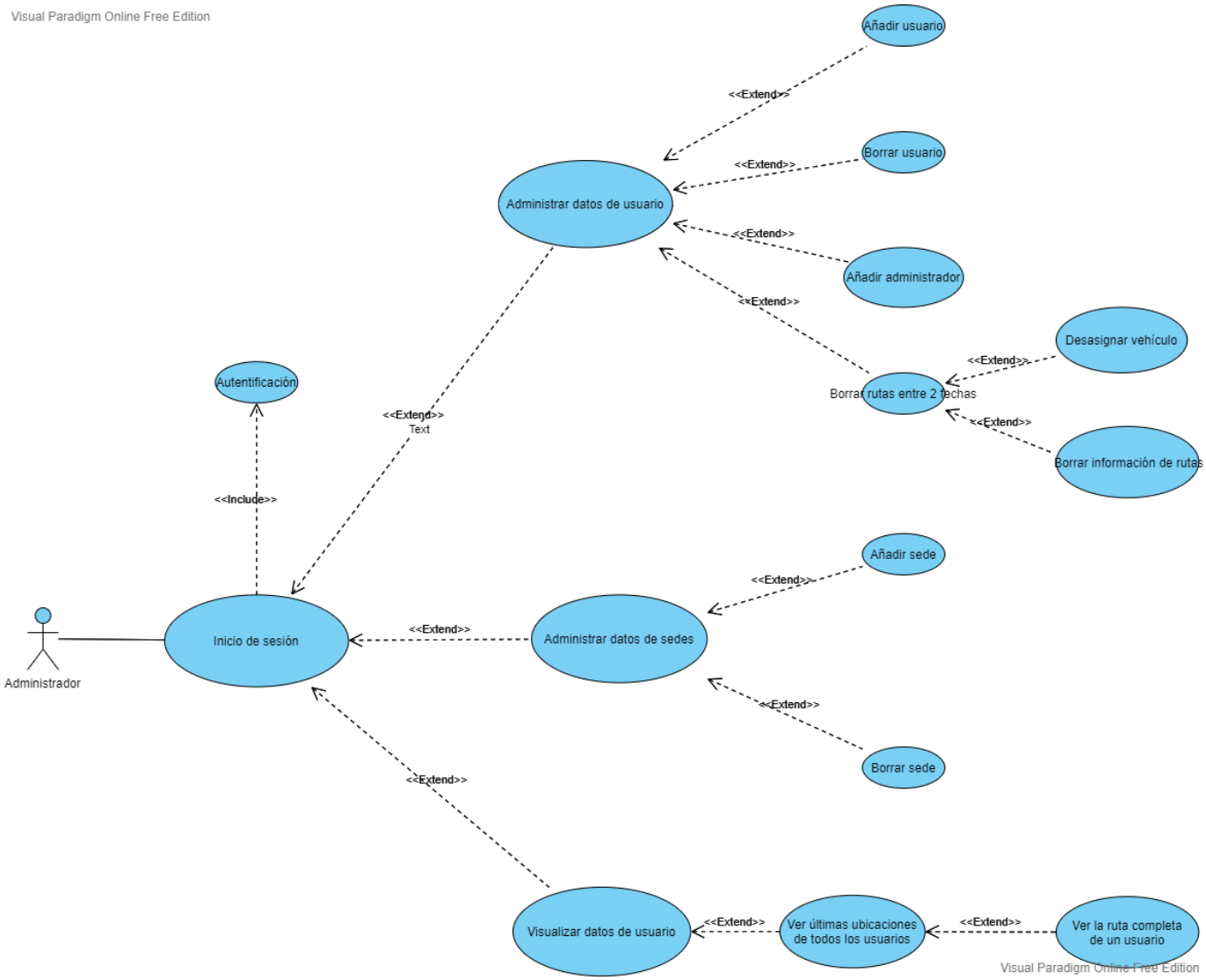

Ilustración 28: diagrama de casos de uso de cliente React

El usuario administrador representa un encargado de la empresa, por lo tanto, es necesario que sea capaz de añadir conductores y además de ser capaz de controlar las rutas de estos y eliminarlos si fuese necesario. La funcionalidad completa del cliente web se detallará más adelante.

5.2 Cliente web React

Como se ha comentado en secciones anteriores se ha utilizado la biblioteca Javascript React, mantenida y desarrollada por Facebook para la creación del cliente web. A la hora de desarrollar en React se pueden usar componentes de clase o componentes funcionales, sin embargo, desde la versión 18.6 de React, en la cual se introdujeron los hooks (que se usarán en el proyecto), React ha instado al uso de los componentes funcionales en vez de los de clase, incluso reescribiendo la documentación [17]. La estructura de ficheros desarrollada es la siguiente.

5.2.1 Index

El fichero index.js, es en el cual se define la estructura de rutas del cliente web. Se importa la función correspondiente a cada fichero Javascript y además se define una ruta para la función importada en concreto.

5.2.2 Data

Carpeta donde se encuentran todas las variables globales al sistema en el fichero global.js. Se almacenan direcciones IP, nombres, etc. Las variables se exportan en este archivo y debe ser importado en cada archivo que desee usarlas.

5.2.3 Misc

Directorio no usado en el que se deberían de encontrar recursos como imágenes o logos.

5.2.4 Pages

Directorio en el que se encuentran todos los ficheros Javascript que se traducen a webs distintas. Se corresponden cada uno con una función, estas se exportan para que index.js pueda importarlos y designar su ruta. Son los siguientes:

- Administrador.js

Portal que se muestra en la ilustración 29 que permite elegir si añadir administradores o usuarios, además de borrar los últimos o las rutas.

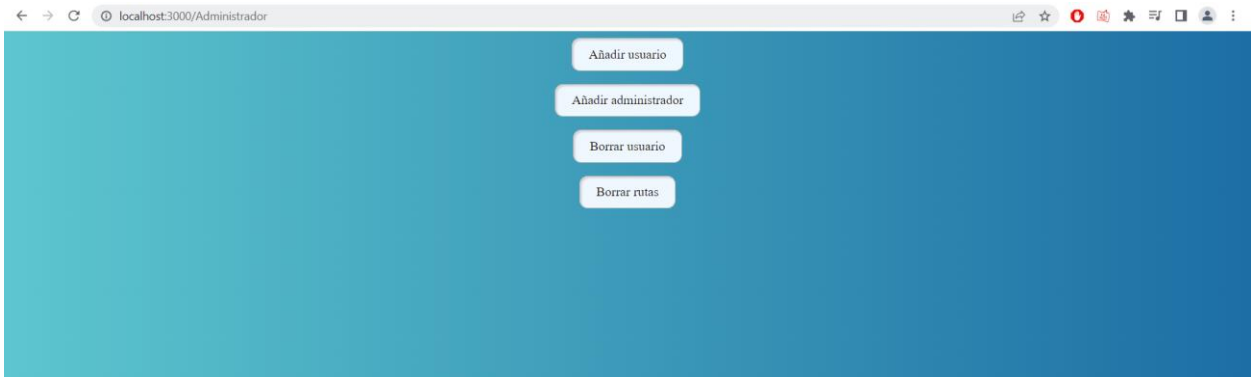

Ilustración 29: captura de pantalla de la web DIR_IP:Puerto/Administrador

- AdministradorSedes.js

Portal que permite elegir qué hacer, añadir o borrar sedes, se muestra en la imagen 30.

Ilustración 30: captura de pantalla de la web DIR_IP:Puerto/AdministradorSedes

- AnadirAdministrador.js

Página web de la ilustración 31 en la que se puede añadir otro usuario administrador, se necesita nombre y apellidos, número de identificación y alias junto con su contraseña.

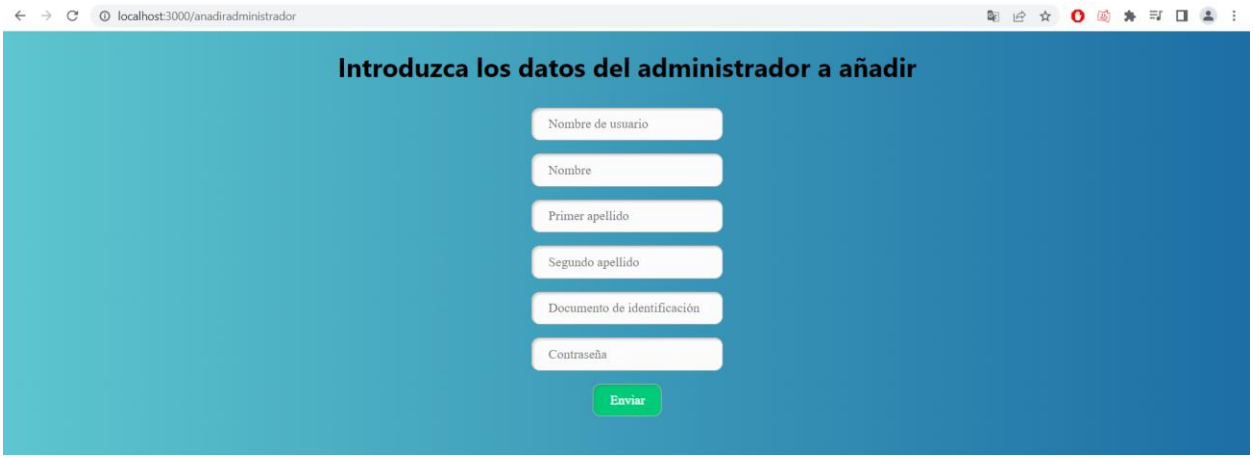

Ilustración 31: captura de pantalla de la web DIR_IP:Puerto/AnadirAdministrador

- AnadirSede.js

Página de la ilustración 32 que permite añadir y borrar sedes junto a su ubicación geográfica, mediante el uso de sus coordenadas.

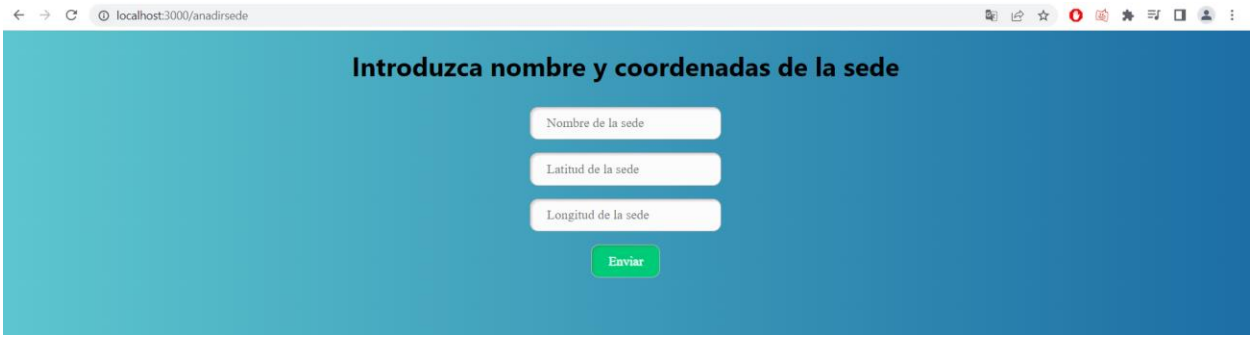

Ilustración 32: captura de pantalla de la web DIR_IP:Puerto/AnadirSede

- AnadirUsuario.js

Página web que se muestra en la ilustración 33 en la que se puede añadir otro conductor, se necesita nombre y apellidos, número de identificación, alias y matrícula del vehículo, junto con su contraseña.

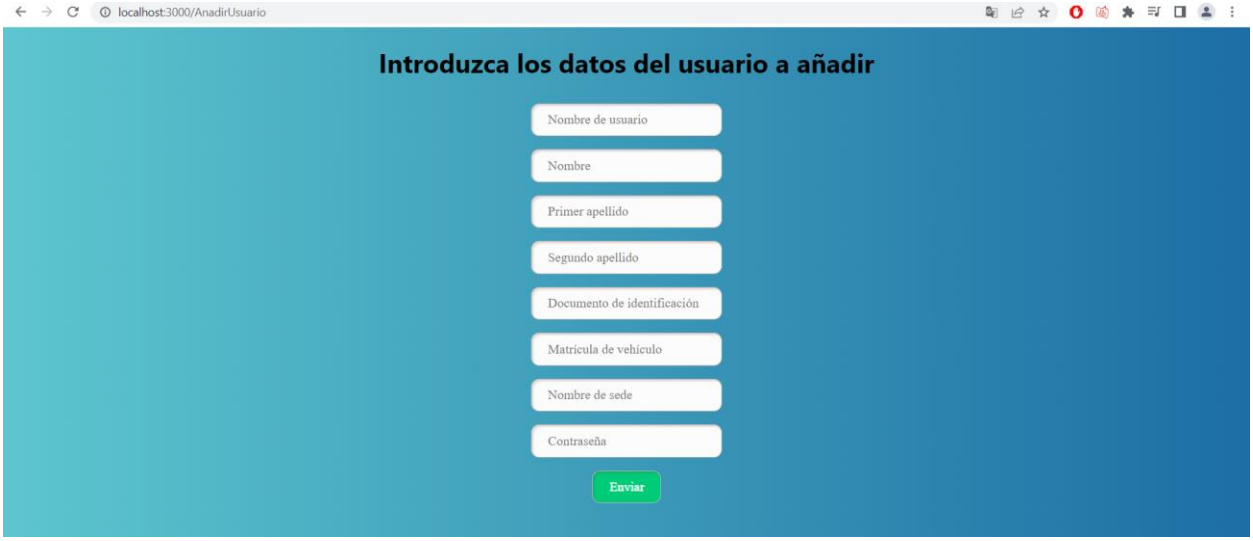

Ilustración 33: captura de pantalla de la web DIR_IP:Puerto/AnadirUsuario

- BorrarRutas.js

Página web en la que aparece un datepicker, como se muestra en la ilustración34, pequeño calendario en

el que se puede elegir un intervalo entre 2 fechas. Al seleccionar un intervalo eliminará todas las rutas guardadas entre esas dos fechas.

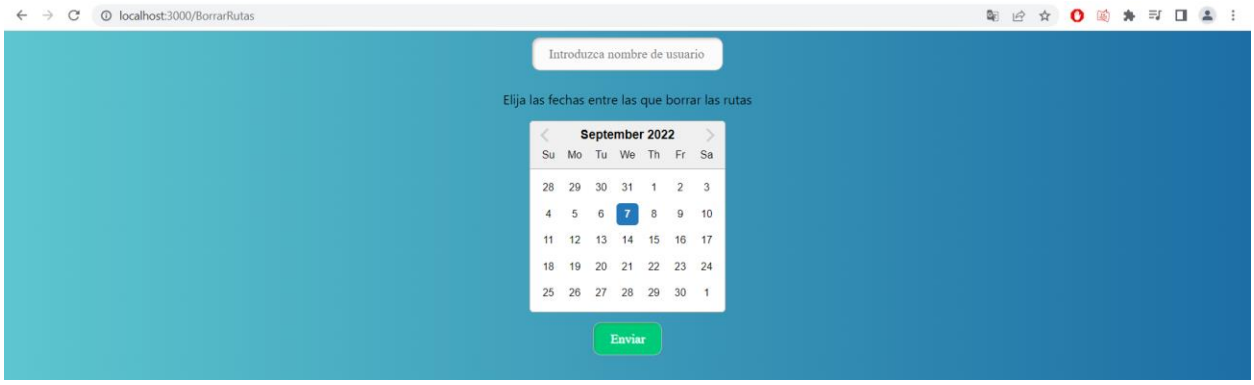

Ilustración 34: captura de pantalla de la web DIR_IP:Puerto/BorrarRutas

- BorrarSede.js

Portal que se enseña en la ilustración 35, en la que al introducir el nombre de la sede la eliminará de la BBDD.

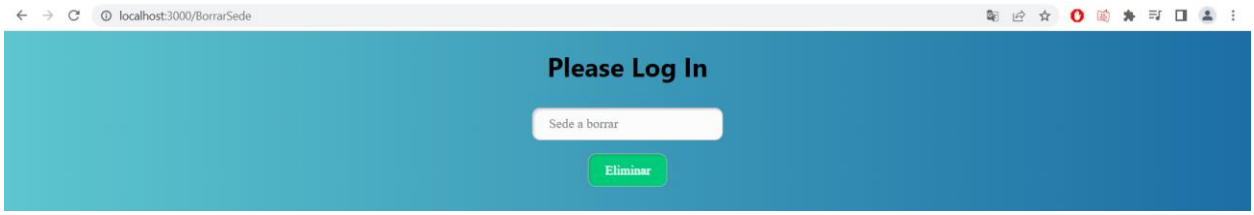

Ilustración 35: captura de pantalla de la web DIR_IP:Puerto/BorrarSede

- BorrarUsuario.js

Web de la imagen 36, en la que al introducir el nombre del usuario lo eliminará de la BBDD.

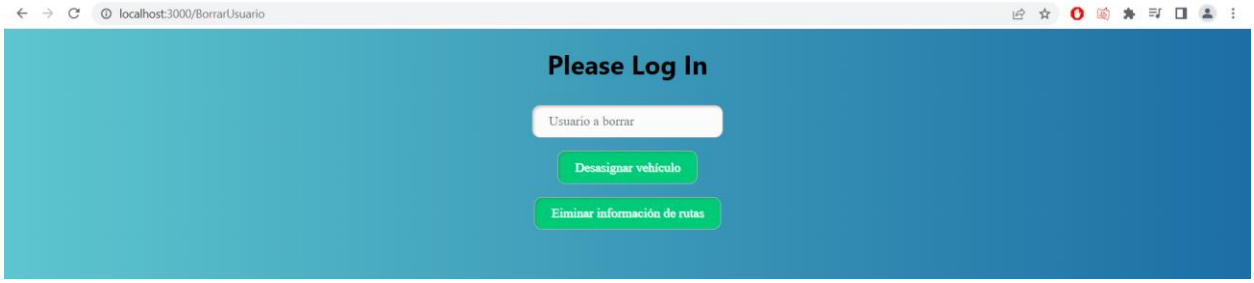

Ilustración 36: captura de pantalla de la web DIR_IP:Puerto/BorrarUsuario

- CalendarioMapa.js

Web que se enseña en la ilustración 37, en la que aparece un mapa además del datepicker mencionado anteriormente, al seleccionar un intervalo mostrará las rutas comprendidas en ese intervalo.

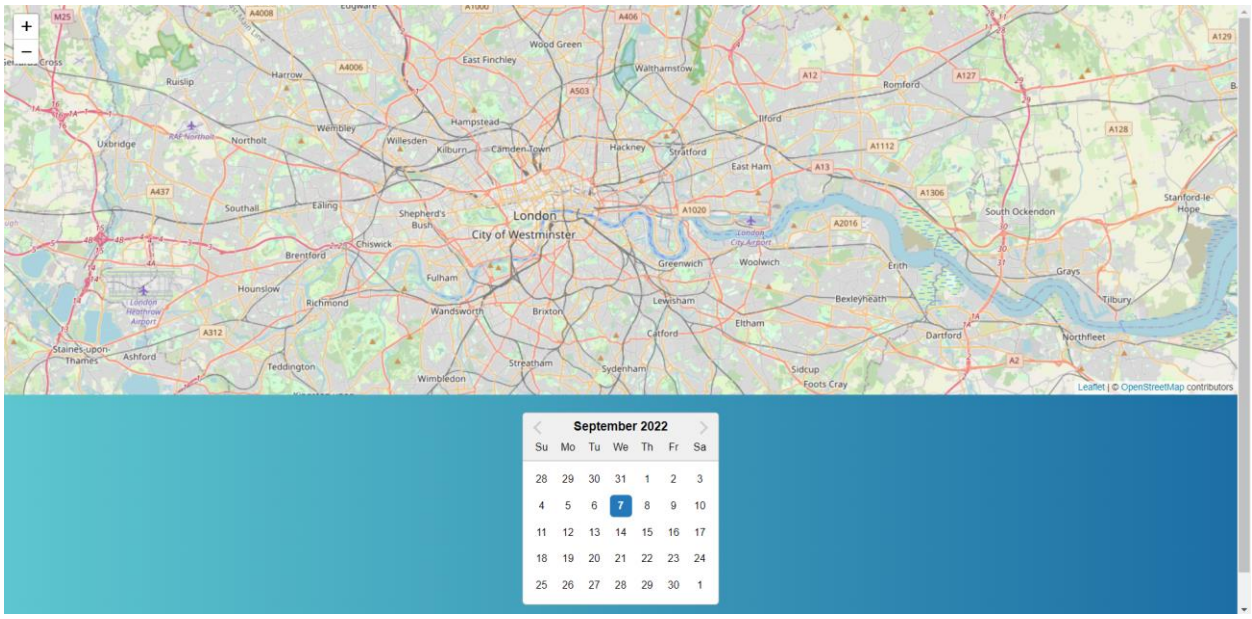

Ilustración 37: captura de pantalla de la web DIR_IP:Puerto/CalendarioMapa

- Datos.js

Mapa de la imagen 38, en el que aparecen las últimas ubicaciones de todos los conductores. Cuando se clica en una de ellas aparece un popup[13] en la ubicación junto con un hipervínculo, al hacer clic en este llevará a Ruta.js.

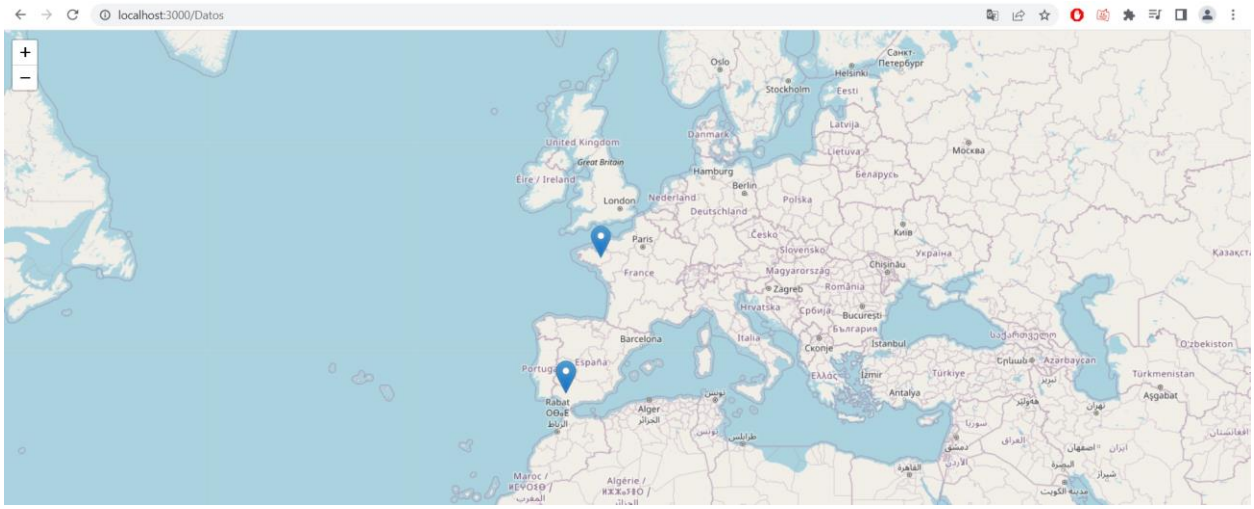

Ilustración 38: captura de pantalla de la web DIR_IP:Puerto/Datos

- Home.js

Portal inicial de la imagen 39, que permite el inicio de sesión del administrador.

[13] Popup: ventana emergente que aparece tras clicar en el marcador.

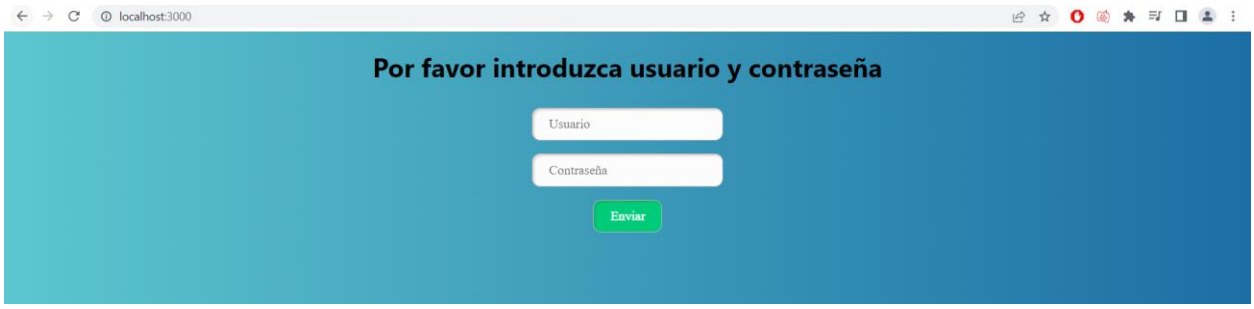

Ilustración 39: captura de pantalla de la web DIR_IP:Puerto/

- Opciones.js

Web de la ilustración 40 que aparece tras autentificarse, en ella se puede elegir sobre qué tipo de información actuar, rutas, sedes…

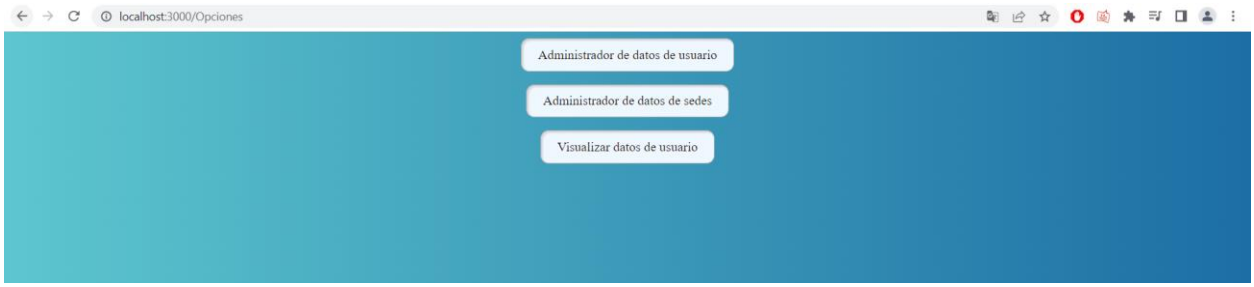

Ilustración 50: captura de pantalla de la web DIR_IP:Puerto/Opciones

- Ruta.js

Web donde se muestra la ruta completa del conductor, que se selecciona en el portal de Datos, desde el primer registro de este, aparece en la ilustración 41.

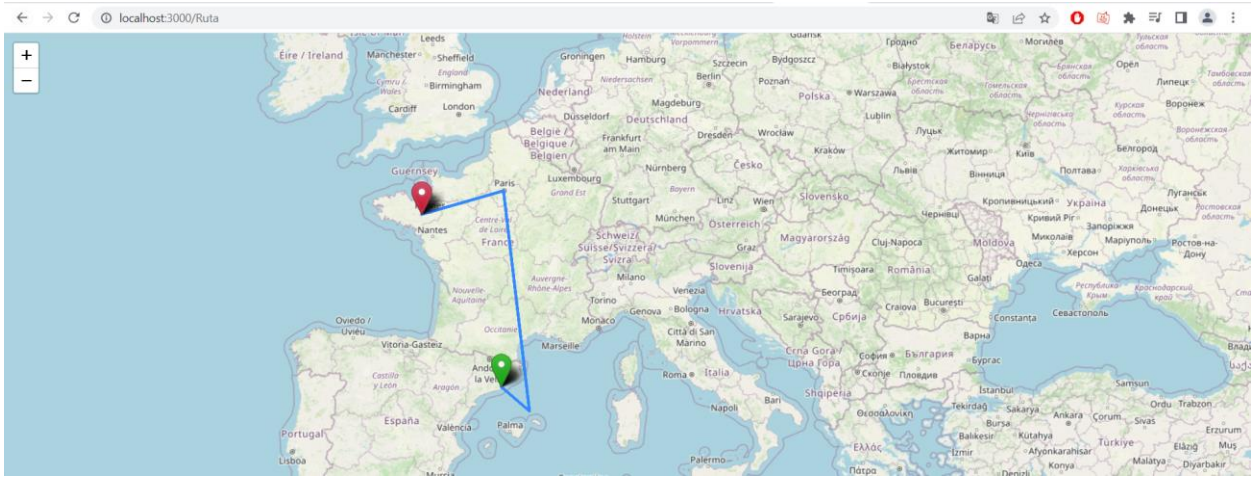

Ilustración 41: captura de pantalla de la web DIR_IP:Puerto/Ruta

5.2.5 Styles

Directorio en el que se encuentra una hoja de estilos CSS para el diseño de todos los botones, cuadros de texto… del cliente web.

6 APLICACIÓN MÓVIL

There's no more important consumer product today than a cell phone.

- Mary Dillon -

En el siguiente apartado se describirán las funcionalidades de la aplicación móvil desarrollada en el lenguaje híbrido React Native.

Es una sencilla aplicación móvil que se encarga de enviar la información de contexto al servicio REST y dispone de algunas funciones extras para los conductores. La aplicación consta de un único fichero App.js. Para poder incluir distintas pantallas dentro de la misma aplicación es necesario declarar un elemento NavigationContainer, este tiene un Stack.Navigator, al que se le anidan Stack.Screens, estas Stack.Screens quedan declaradas en el mismo fichero y son las propias pantallas de la aplicación.

Para el envío de los datos se usa BackgroundTimer, para utilizarlo es necesario la instalación del paquete vía npm, aunque se profundizará más sobre la instalación de los paquetes en anexos posteriores. Este temporizador enviará cada 30 segundos (configurables) el elemento location, que se recibe de la función GetLocation.getCurrentPosition(), la location contiene elementos como la altitud, la velocidad o el bearing. Una vez se reciba en el servidor web, se guardarán los datos que precedan.

Las pantallas presentes en la aplicación serán las siguientes:

- HomeScreen

Ilustración 42: pantalla de inicio de sesión

Como se observa en la ilustración 42 es la pantalla inicial donde se encuentran dos campos para introducir usuario y contraseña, en el caso de que el login sea incorrecto se mostrará una alerta de error, como también se muestra en la imagen 43. Como para el funcionamiento del temporizador es necesario que se envíe al servicio web el nombre del usuario introducido se usa AsyncStorage, un sistema de almacenamiento de datos clave-valor global a todo el sistema, así cuando el usuario se autentique no necesitará usar una variable global para guardarlo. AsyncStorage tiene 2 métodos, uno que permite obtener la información y otro que permite alterarlo. Es un método asíncrono, por lo que es necesario usarlo junto el método await para esperar a que se resuelva la asignación y obtención de la información.

Como es necesario que el temporizador no envíe ninguna información si el logeado no ha sido correcto, es necesario implementar otro AsyncStorage, cuyo valor se pone a 0 en el inicio de la aplicación, y que solo si el logeo es correcto, en cuyo caso el valor se cambiará a 1, permitirá al temporizador comenzar a ejecutarse.

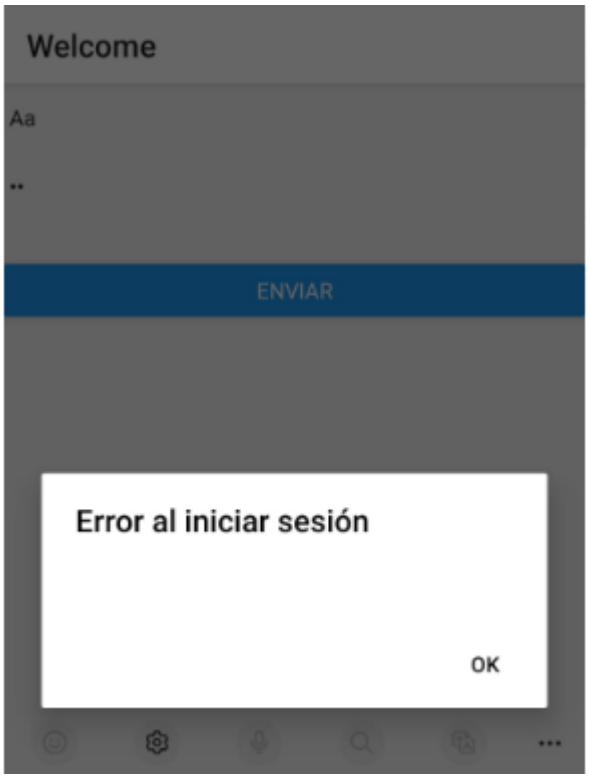

Ilustración 43: Pantalla de error de inicio de sesión

- EleccionScreen

Pantalla que aparece en la imagen 44, a la que se llega después de iniciar sesión correctamente, a partir de esta pantalla se comienzan a enviar los datos de ubicación al OCB. Cuenta con 3 botones que dirigen hacia las 3 pantallas restantes de la aplicación.

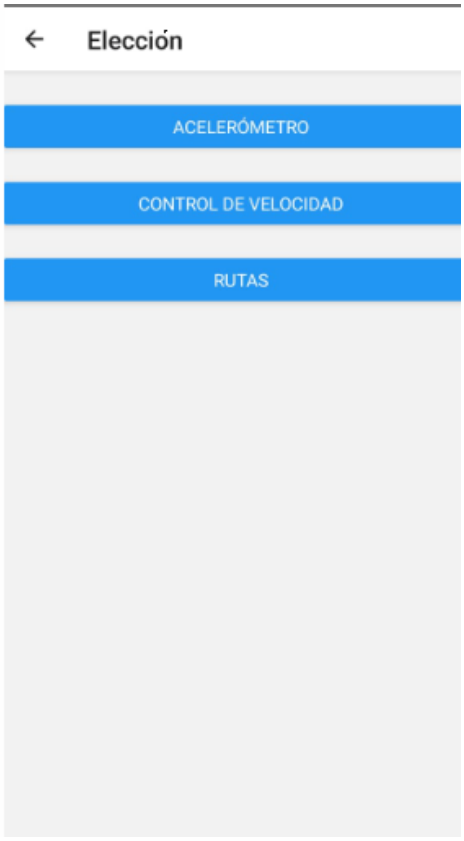

Ilustración 44: pantalla de elección

- ProfileScreen

Pantalla que importa e implementa unos acelerómetros declarados en otro fichero.

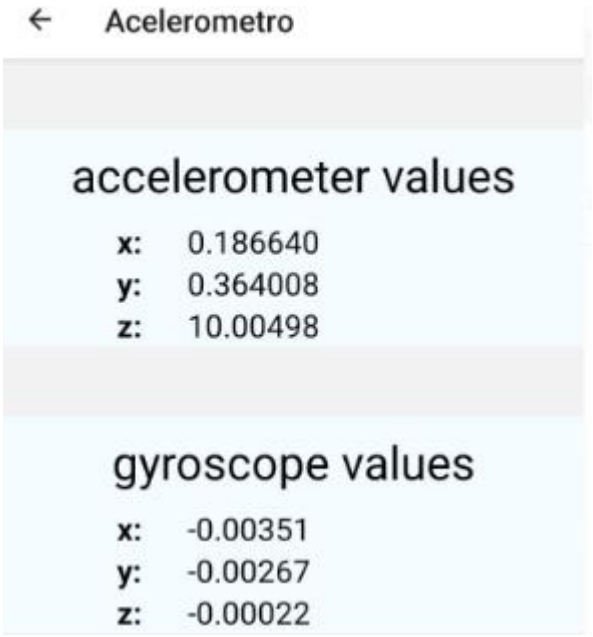

Ilustración 45: pantalla de velocidad del vehículo

- GPSScreen

Pantalla en la que se compara la velocidad del vehículo con la máxima de la vía. Dependiendo de si la velocidad es mayor o menor el color de la velocidad cambiará entre rojo o verde. Para obtener la velocidad del vehículo se usa la función mencionada anteriormente, que devuelve un objeto de tipo location que contiene la velocidad del vehículo. Para obtener la velocidad de la vía es necesario realizar 2 consultas a OSM. En una de estas se obtiene el nodo de la posición en la que se encuentra el conductor, después se realiza otra petición incluyendo la id OSM que se ha obtenido anteriormente, la que devuelve, si existiese, la velocidad máxima de la vía, que se guarda en un hook de react y se muestra en pantalla.

Ilustración 46: pantalla de velocidad del vehículo

En la imagen 46 se muestra la velocidad actual y como cambia a color rojo si supera la máxima de la vía, en este caso la velocidad máxima es roja ya que esa vía no cuenta con velocidad máxima en la BBDD de OSM.

- MapScreen

Como se muestra en la imagen 47, enseña la ruta que ha recorrido el conductor desde que se abrió esta pantalla, utiliza los mapas de Google en vez de los de OSM. Cada 20 ó 30 segundos, configurables, añade una nueva ubicación a la ruta dibujada.

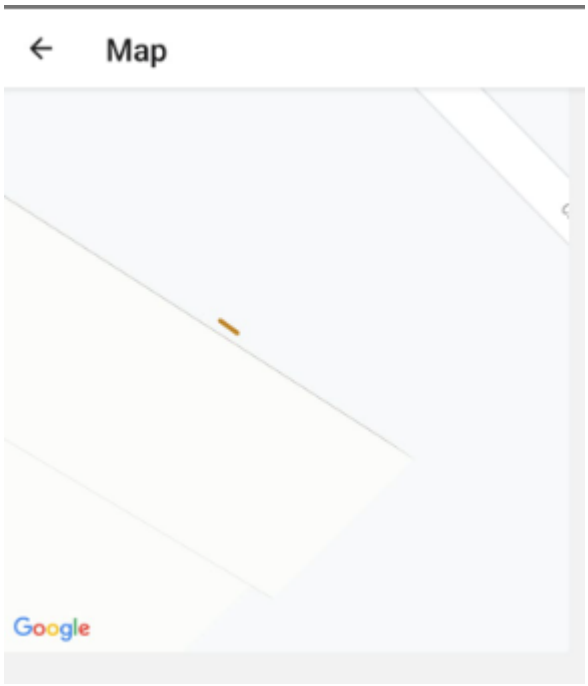

Ilustración 47: Pantalla de rutas

Por último, en la ilustración 48 se va a añadir una captura de Wireshark donde se observa el intercambio de paquetes HTTP entre el dispositivo móvil y el servicio web. Se observan unos mensajes de login incorrecto, otros de loging correcto, y una de las actualizaciones periódicas de los parámetros del vehículo, en la que se muestran algunos de los datos enviados como la altitud o ubicación.

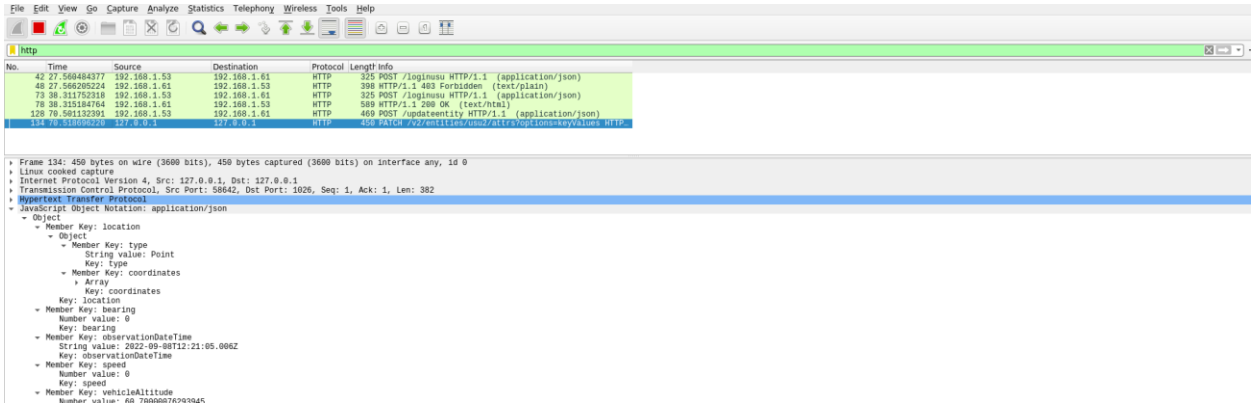

Ilustración 48: captura de Wireshark de intercambio de mensajes

7 CONCLUSIONES Y LÍNEAS FUTURAS

Reflexive thinking turns experience into insight

- John C. Maxwell -

En la realización de este proyecto se ha logrado conseguir una aplicación que permita la gestión además de la recogida de datos de una empresa de camiones utilizando todas las herramientas descritas anteriormente y usando conceptos tan novedosos como el IoT o las aplicaciones móviles.

Una de las intenciones del proyecto fue hacerlo lo más simple y sencillo para cualquier tipo de usuario, sin necesidad de tener conocimiento en programación, aunque se han logrado conseguir unas aplicaciones básicas y sencillas, considero que sí que se podrían haber simplificado algunas tareas y funciones para hacerlo más instintivo y reducir la necesidad de clics, sobre todo en la aplicación web.

La aplicación móvil no dispone de una gran cantidad de funciones para los conductores, y se acaba reduciendo a una aplicación que marca la velocidad máxima y si se ha superado o no, de existir la velocidad en OSM, además de enviar los datos a FIWARE. Quizás se podrían haber añadido más detalles como mostrar el tráfico de las carreteras e incluso con más tiempo y conocimiento, desarrollar un algoritmo que fuese capaz de predecir la ruta más rápida y segura hacia un destino.

Hoy en día los conductores de mercancías tienen que cumplir un complicado reglamento laboral que les puede acarrear multas graves de hasta 4500€ [18]. Por lo que su tiempo de conducción queda registrado en el tacómetro del camión. Si el modelo de datos de FIWARE fuese más flexible y no se dependiese de la aplicación móvil, podría haber usado el registro de información para comprobar que los tiempos de conducción del transportista estuviesen de acuerdo con el reglamento, mostrando avisos por pantalla a los administradores y conductores. Con la implementación del coche inteligente en el futuro, puede llegar a ser muy interesante usar el software del vehículo para comprobar que la conducción se está realizando de forma correcta y legal, ya que no se dependerá de la buena praxis del conductor a la hora de usar la app.

El desarrollo además se ha realizado, como ya se comentó antes, en una máquina virtual VMWARE, la que se ha configurado para que disponga de una IP propia dentro de una red privada. Esta configuración es inviable si el producto busca su comercialización o implementación real, para llegar a ser completamente funcional habría dos opciones, la primera usar un servidor privado o alquilado que disponga de una IP pública en la que se ejecute la máquina virtual junto con el servicio REST, o la segunda opción y la más novedosa: usar servicios de computación en la nube como Google Cloud o Microsoft Azure. Los servicios de computación en la nube han superado a los de computación tradicional la última década mayormente debido a que acaban resultando más baratos de mantener y desplegar para las empresas y la seguridad proporcionada por grandes compañías del sector. No obstante, no se ha usado ninguna de estas plataformas para el desarrollo del proyecto debido al coste de estas.

En conclusión, con la implementación y difusión de las nuevas tecnologías como el IoT y la computación en la nube, lo que hoy en día es solo una idea: administrar todos los conductores de una empresa de transportes,

podría llegar a convertirse en una aplicación ampliamente comercializada y difundida que ayudaría a mejorar la entrega de mercancías.

ANEXO A: INSTALACIÓN Y CONFIGURACIÓN DE REACT NATIVE

En este primer anexo se mostrará como instalar y configurar React Native, además de todas sus dependencias.

React Native necesitará un interfaz de línea de comandos, JDK, Node y Android Studio. Se ha seguido el tutorial de: *https://reactnative.dev/docs/environment-setup* y algunos de los enlaces a los que redirige.

A.1 Instalación de JDK y Node

Para instalar JDK y Node es recomendable instalar Chocolatey, herramienta que cuenta con el mayor registro de paquetes para Windows [19]. Antes de instalarlo deberá ejecutarse el siguiente comando en el PowerShell de Windows.

```
Get-ExecutionPolicy
```

Si devolviese Restricted, habría que utilizar la siguiente orden para poder instalar Chocolatey:

```
Set-ExecutionPolicy Allsigned
```

Después, solo quedaría ejecutar el siguiente comando para finalizar la instalación.

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

Por último, solo quedaría abrir el cmd de Windows como administrador y utilizar Chocolatey para instalar Node y JDK. Npm, que se usará para instalar paquetes al proyecto se instala junto con Nodejs.

```
choco install -y nodejs-lts openjdk11
```

Cabe destacar que se recomienda usar JDK11, así como es necesaria la versión 14 de Node o superior.

Node se ha instalado y usado junto con React Native, React y Express en todo el proyecto, además se ha utilizado para instalar paquetes no nativos a estas tecnologías, tales como el datepicker de React o el temporizador de React Native se ha usado un código del siguiente tipo:

```
npm install nombre_del_paquete
```

Siendo nombre_del_paquete el paquete a instalar. Es importante importarlo después si no, no se podría utilizar en la aplicación.

A.2 Instalación de Android Studio

La instalación de Android Studio será la instalación por defecto que se realiza a la hora de descargar el instalador de su web, se deben marcar las casillas Android SDK, Android SDK Platform y Android Virtual Device.

Es importante que durante la instalación o bien después en la pantalla SDK Manager de la pestaña Tools se marquen las opciones que aparecen en la ilustración 49, o React Native no podrá ejecutarse.

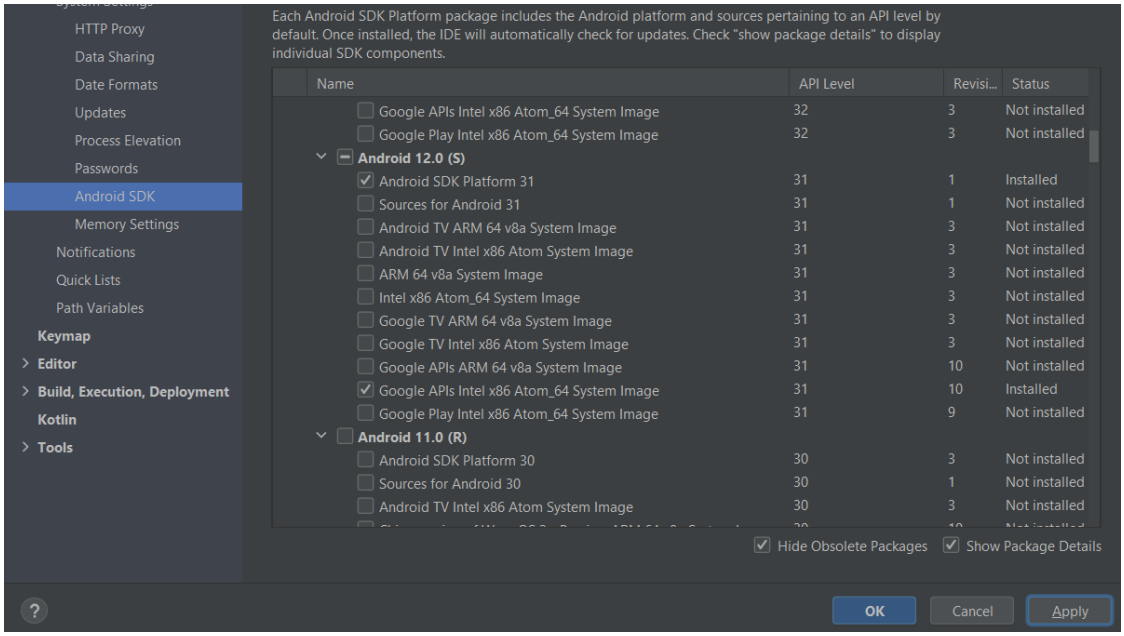

Ilustración 49: opciones de instalación Android SDK.

Como nota, podría haberse instalado Intelx86 Atom_64 System Image en vez de Google APIs Intel x86 Atom_64 System Image.

Posteriormente es necesario configurar la variable de entorno ANDROID_HOME. Lo cual se hace desde el panel de control de Windows, haciendo clic en cuentas de usuario y después en cambiar variables de entorno. Por defecto, la ruta que hay que colocar es la siguiente:

```
%LOCALAPPDATA%\Android\Sdk
```

También hay que añadir la variable platform-tools cuya localización es:

```
%LOCALAPPDATA%\Android\Sdk\platform-tools
```

A.3 Instalación de React Native y creación del proyecto

Para instalar React Native se recomienda en primer lugar instalar la última versión de la línea de comandos de React Native usando el comando npx, que se instala junto con Node, mediante la siguiente orden en un cmd con permisos de administrador:

```
npx react-native <command>
```

Tras este paso, se podrá crear un proyecto de React Native llamado AwesomeProject escribiendo:

```
npx react-native init AwesomeProject
```

A.4 Ejecución de todo el sistema

Por último, para ejecutar el sistema se necesitará tener abiertos 2 cmds con permisos de administrador, aunque como se explicará a continuación solo 1 es necesario, además de Android Studio.

- **Primer paso(opcional):** en primer lugar, se ejecutará Metro, el bundler de React Native, como ya se mencionó antes. Para ejecutarlo se usará en el directorio donde se encuentre el proyecto:

```
npx react-native start
```

El resultado de la ejecución se mostrará en imagen 50:

Ilustración 50: cmd tras ejecutar npx react-native start

- **Segundo paso:** con Android Studio abierto podemos elegir en qué dispositivo ejecutar la aplicación móvil: un dispositivo físico o un dispositivo virtual. Eligiéndolo en el desplegable de dispositivos disponibles.

  Para ejecutarlo en un dispositivo virtual es necesario haberlo creado antes en la pantalla de AVD Manager que se encuentra en Tools, que se muestra en la ilustración 51.

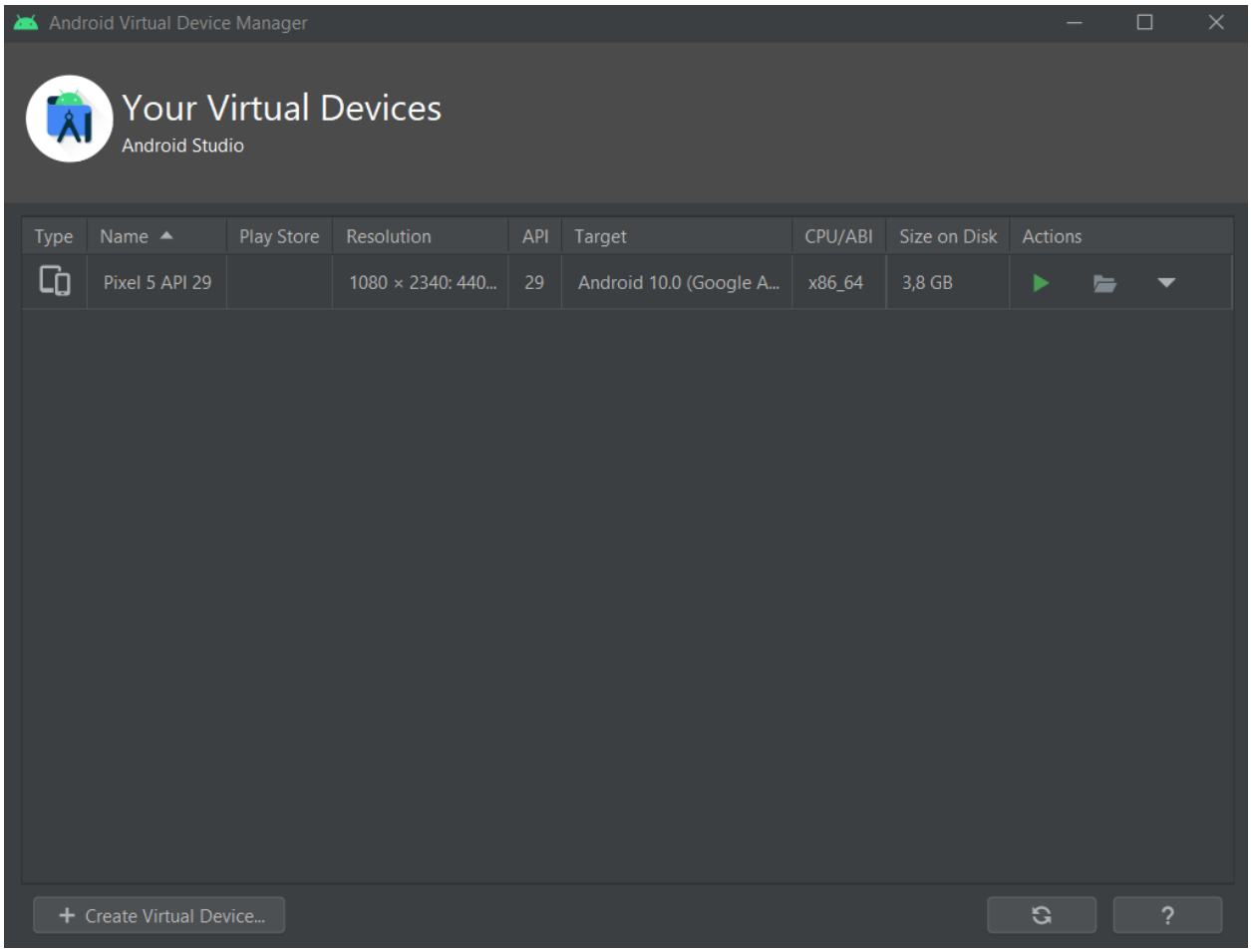

Ilustración 51: pantalla de dispositivos virtuales de Android Studio

Para ejecutarla en un dispositivo físico, con este conectado por USB al ordenador, es necesario haber activado el modo desarrollador y permitir la opción de depuración USB, en la imagen 52 se muestra el proceso para realizar la depuración.

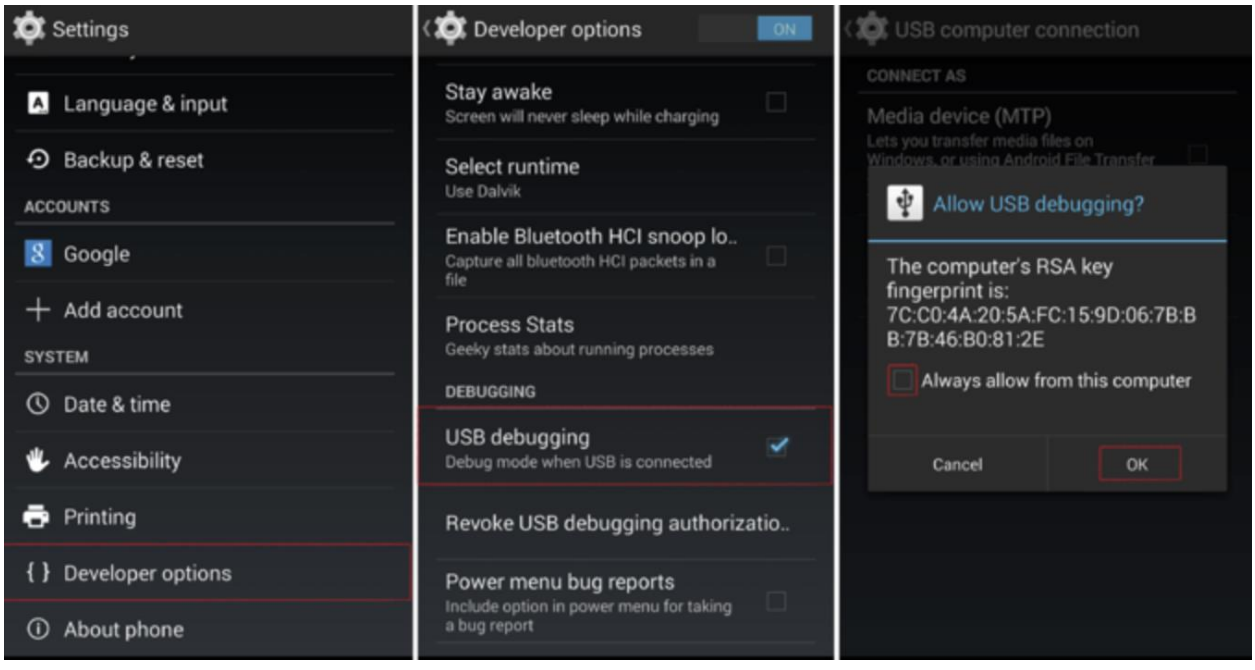

Ilustración 52: pasos para permitir depuración USB.

- **Tercer paso:** en este paso habrá que ejecutar la siguiente orden.

```
Npx react-native init
```

Esta orden ejecutará la aplicación en el dispositivo que se haya seleccionado, y de no haberse realizado el primer paso, o de no tener abierto el simulador, abrirá un dispositivo virtual y ejecutará Metro por sí mismo. Todos los errores de compilación y de ejecución se mostrarán tanto en el dispositivo móvil como en el terminal en el cual se ejecuta Metro, que es la consola por defecto para React-Native.

ANEXO B: INSTALACIÓN Y CONFIGURACIÓN DE VISUAL STUDIO CODE

B.1 Instalación

Para la instalación solo es necesario descargar el ejecutable presente en su página web y ejecutarlo.

B.2 Configuración

Para que Visual actúe como IDE es necesario instalar las extensiones necesarias para los lenguajes en los que se desee programar, ya que de lo contrario no será más que un simple editor de texto. Para instalarlas es necesario clicar en el recuadro de extensiones de la barra de herramientas y buscar e instalar las extensiones necesarias. En la imagen 53 se observa el recuadro mencionado antes y el cuadro de búsqueda, además de los módulos instalados para realizar el proyecto.

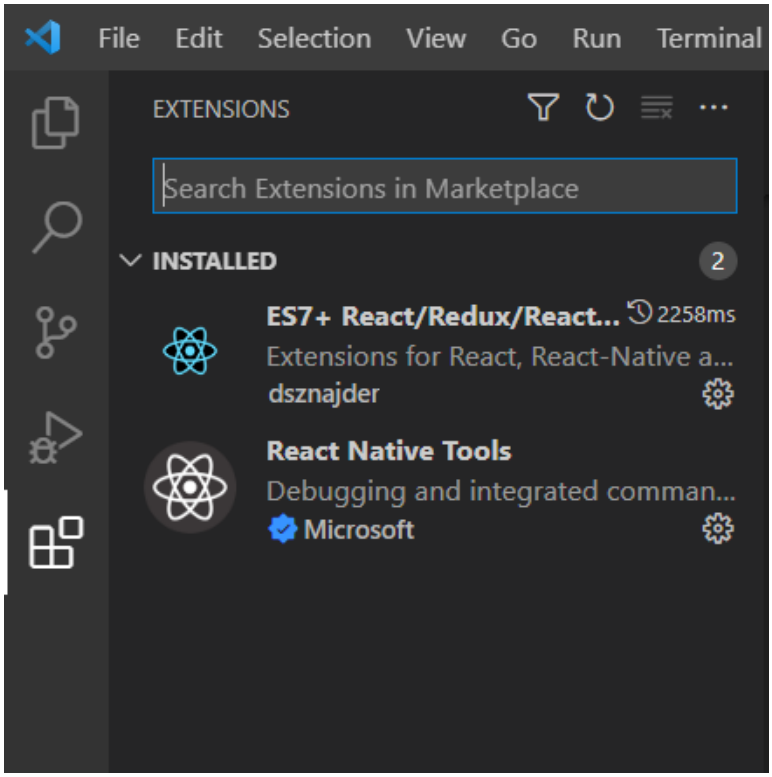

Ilustración 53: extensiones Visual Studio Code

ANEXO C: INSTALACIÓN Y CONFIGURACIÓN DE REACT

En este anexo se explicará cómo instalar y configurar React en una máquina Windows, sin embargo, si se han seguido los pasos del Anexo A relacionados con la instalación de JDK y Node, no será necesario ninguna instalación, puesto se puede crear un proyecto solo usando Nodejs y npm.

C.1 Creación de proyecto y ejecución

Para crear un proyecto y ejecutarlo será necesario usar los siguientes comandos en un cmd con permisos de administrador

```
npx create-react-app my-app
cd my-app
npm start
```

Esto creará el proyecto, se cambiará al directorio del mismo y lo ejecutará, abriendo en el navegador predeterminado de Windows el elemento Home del index.js que se explicó en la sección 5.2.1.

C.2 Depuración de errores

Tras la ejecución el cmd seguirá ejecutando el cliente web y se mostrará lo que aparece en la ilustración 54:

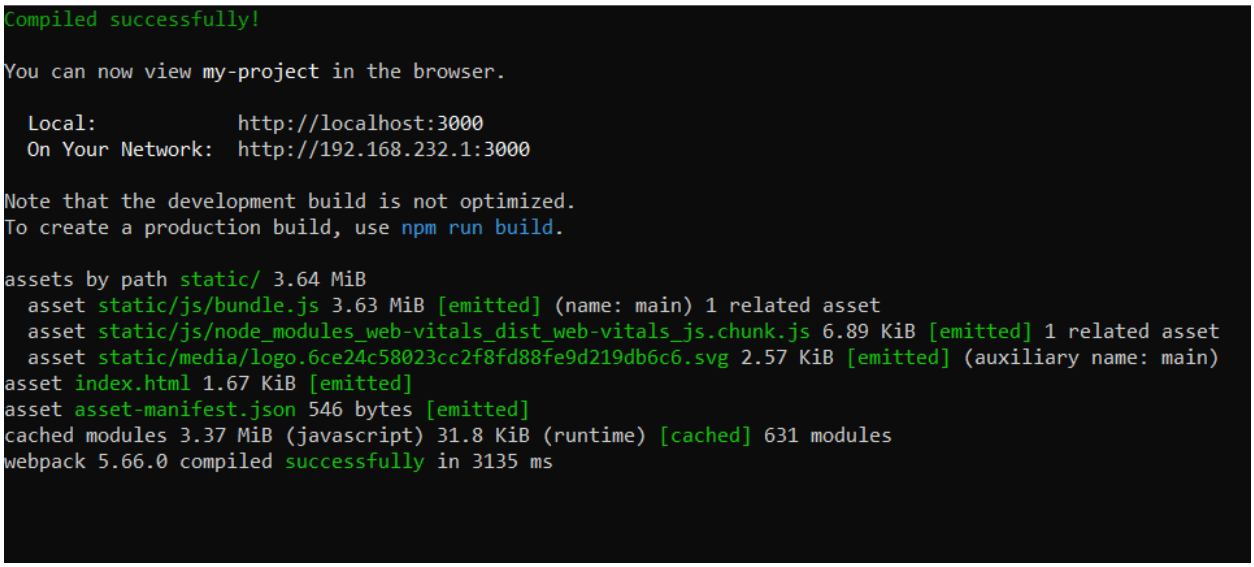

Ilustración 54: cmd tras ejecutar npm start

Aquí únicamente se muestran los errores de compilación del proyecto, sin embargo, resulta muy útil utilizar las herramientas de desarrollo del navegador para observar errores más concretos y que no afectan a la ejecución del cliente web, por ejemplo: un error en la conexión al servicio REST. En Google Chrome el atajo de teclado para mostrar estas herramientas es: Ctrl + shift + j. Tras presionarlo aparece en pantalla lo mostrado en la imagen 55.

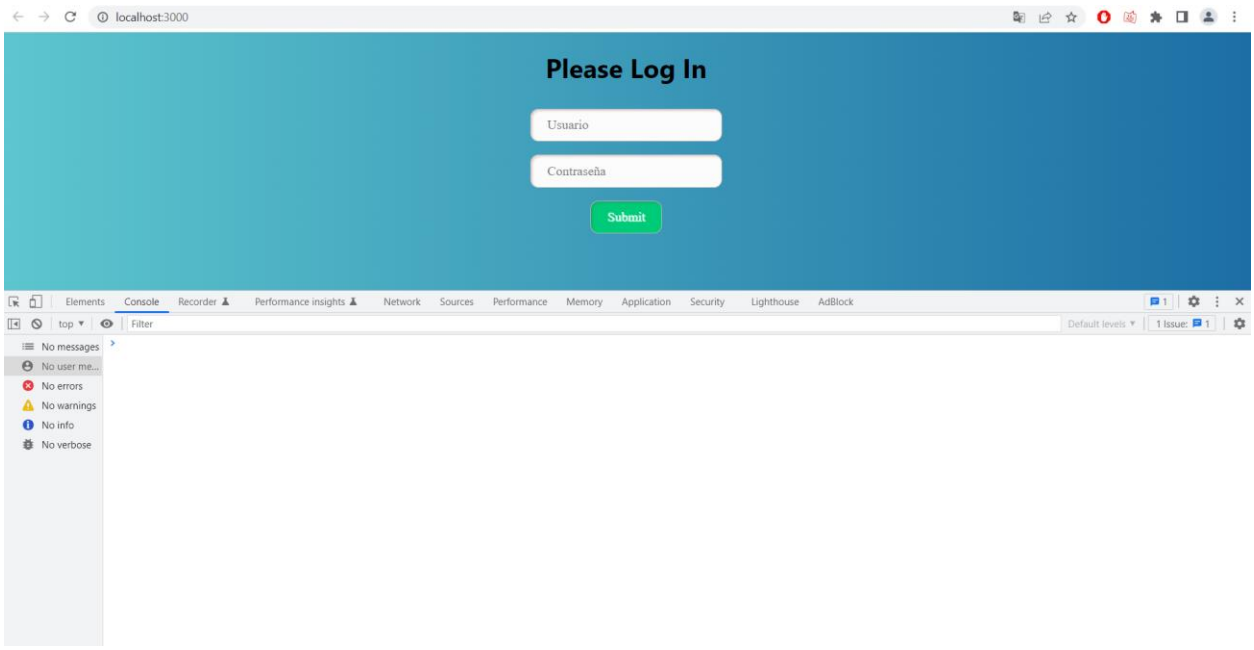

Ilustración 55: herramientas de desarrollo de Google Chrome

Como se observa en la imagen incluye múltiples herramientas, como una consola, que es la salida estándar para todo lo que se desarrolle por react, o permite comprobar el tráfico de paquetes de la aplicación.

ANEXO D: CONFIGURACIÓN DE VMWARE

Considerando VMWARE ya instalado y con la máquina virtual (VM), del departamento de Ingeniería Telemática ya abierta y creada, para permitir la comunicación de un dispositivo móvil con la VM hay que configurar esta para que se le asigne una dirección IP dentro de la red en la que se encuentre el ordenador, en nuestro caso será una IP privada, pero podría ser una pública.

Para realizarlo será necesario acceder a la configuración de la VM y dentro de Network Adapter cambiar la Network Connection a bridged, lo que le dará una IP dentro de la red que use el dispositivo físico, como se ve en la imagen 56.

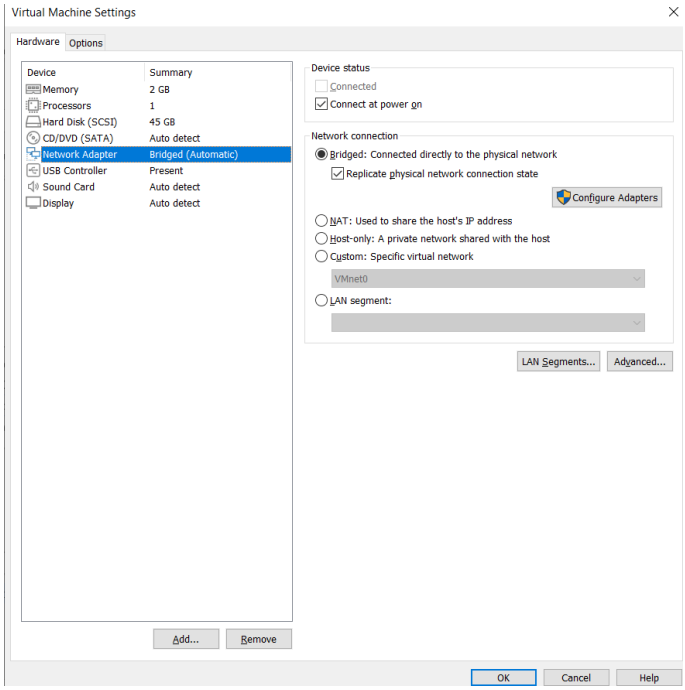

Ilustración 56: configuración VMware

Esta opción convierte la VM en una entidad separada del dispositivo físico en su propia red. [20]

Como nota, si se quisiese aumentar la memoria de la máquina virtual, necesario para instalar alguna de las herramientas usadas en el proyecto, además de aumentar la cantidad de memoria disponible para la VM es necesario expandir la memoria de esta. Para realizarlo es necesario instalar y utilizar herramientas externas como GParted, de lo contrario la VM seguirá usando el mismo espacio de disco que se le asignó la primera vez que creó.

ANEXO E: INSTALACIÓN, CREACIÓN Y EJECUCIÓN DEL SERVICIO REST EN UBUNTU

En esta sección se va a explicar cómo instalar node en Ubuntu junto con Express, para poder desarrollar el servicio web.

E.1 Creación del servicio web

Para la creación del servicio es necesaria la instalación de Node junto con su gestor de paquetes npm lo que será sencillo, pues se puede usar apt, el gestor de paquetes propios de Ubuntu de la siguiente forma.

```
sudo apt update
sudo apt install nodejs
sudo apt install npm
```

Después de realizar esta operación será necesario instalar Express un framework de aplicación para Node, como ya se comentó antes mediante el uso de npm.

```
npm install express
```

Esto debe realizarse en el mismo directorio donde se vaya a encontrar el servicio web, en nuestro caso un simple fichero JavaScript llamado index.js. Después de crear este simple fichero, debe declararse que se va a usar express e importarse.

```
const express = require('express')
const app = express()
```

E.2 Ejecución del servicio web

La ejecución se realiza con un simple comando, en el directorio donde se encuentre el fichero, tras lo cual el servicio queda ejecutándose en la dirección y puertos configurados en el fichero.

```
node nombre_del_servicio
```

Siendo nombre_del_servicio el nombre dado al fichero JavaScript creado anteriormente.

ANEXO F: INSTALACIÓN, CONFIGURACIÓN Y EJECUCIÓN DE MYSQL EN UBUNTU

A continuación, se procederán a describir todas las operaciones realizadas para poder crear y usar una Base de Datos MySQL en Ubuntu.

F.1 Instalación de MySQL

La instalación de MySQL desde un usuario no root es sencilla, puesto que solo hace falta usar apt para realizarlo:

```
sudo apt update
sudo apt install mysql-server
```

Tras la instalación, es necesario determinar las políticas de seguridad de MySQL, hay que ejecutar el siguiente comando.

```
sudo mysql_secure_installation
```

Las políticas de seguridad determinarán la complejidad de la contraseña necesaria, si está permitido o no el acceso a usuarios remotos a la BBDD etc.

Tras este último paso, el servicio MySQL típicamente quedará en estado activo, se puede comprobar su estado, parar e iniciar mediante el uso de systemctl, además se habrá creado un usuario root con una contraseña que debe estar de acuerdo con la política elegida

F.2 Creación de la Base de Datos.

Lo siguiente será crear un usuario no root, con una determinada contraseña y una BBDD dentro del servicio MySQL. Los pasos serán los siguientes:

```
sudo mysql -u root -p
mysql > CREATE USER ‘username’@’localhost’ IDENTIFIED BY ‘password’;
mysql> CREATE DATABASE database;
mysql> GRANT ALL PRIVILEGE ON database TO ‘username’@’localhost’;
mysql> FLUSH PRIVILEGES;
```

En el primer comando se abrirá mysql usando el usuario root e introduciendo la contraseña, posteriormente se creará otro usuario de nombre ‘username’ al que solo se puede acceder localmente (‘localhost’) y la contraseña especificada, que debe seguir las políticas de seguridad. Por último, se creará una BBDD dentro de MySQL llamada ‘database’ y se le darán todos los privilegios al ‘username’ sobre la última. Usar FLUSH es una buena práctica porque se recargarán los privilegios para todos los usuarios.

Y, por último, se accede a MySQL con el usuario recientemente creado, y se usa la BBDD anteriormente creado. Sobre esta BBDD, se pueden hacer todo tipo de operaciones, crear tablas, alterarlas…

```
sudo mysql -u username -p
mysql > USE database;
```

MySQL permite multitenancy, es decir, dentro del servicio MySQL se pueden crear varias BBDD accedidas por todos los usuarios que tengan permisos sobre estas.

F.3 Acceso desde el servicio REST

Como se comentó anteriormente para poder acceder al servidor MySQL desde node, se debe instalar el paquete necesario en el directorio del servicio REST.

```
Npm install mysql
```

Para acceder desde el servicio es necesario incluir la información de autentificación en el fichero del servicio desde el que se realiza la conexión. Por no incluir los datos en el fichero y para permitir cambiarlos sin necesitar de alterarlo, se han incluido en un documento JSON externo que se importa en el servicio.

ANEXO G: INSTALACIÓN DE DOCKER Y EJECUCIÓN PARA INSTALAR ORION CONTEXT BROKER, CYGNUS Y POSTGRESQL

En este anexo se instalarán Docker y Docker-compose de acuerdo con el tutorial de (*https://docs.docker.com/engine/install/ubuntu/*).

G.1 Instalación de Docker

Como ya se comentó en secciones anteriores, Docker es una herramienta que permite instalar y ejecutar servicios en entornos controlados de forma fácil. Para instalarlos sería necesario:

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
sudo apt install docker-ce
```

En los pasos anteriores, se actualiza apt, se le permite usar paquetes sobre HTTPS, se le añade la clave CPG del repositorio Docker y se instala el mismo.

Posteriormente se debe instalar Docker-compose, Docker compose es una herramienta de Docker que permite la instalación de varios paquetes desde un fichero de configuración yml, en vez de solo un paquete desde la línea de comandos. Por lo tanto, Docker-compose es preferible para una instalación tan compleja como esta.

Para instalarlo es necesario descargarlo desde el repositorio de github y darle permisos de ejecución.

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-composesudo apt install apt-transport-https ca-certificates curl software-properties-common
sudo chmod +x /usr/local/bin/docker-composesudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

Por último, es necesario crear el fichero docker-compose.yml que servirá para la instalación. El fichero es demasiado largo para escribirlo en la memoria y se proporcionará junto con el resto del proyecto.

G.2 Ejecución

Para la ejecución de todos los contenedores del fichero docker-compose es necesario primero observar si alguno de los contenedores que se quieren ejecutar está activo, ya que si lo está puede llegar a fallar en su ejecución, y si lo estuviese, tomar su ID y eliminarla manualmente para después ejecutar el YML.

```
docker ps -a
docker rm -f CONTAINER_ID
docker-compose up
```

Siendo CONTAINER_ID el ID obtenido con el primer comando.

REFERENCIAS

[1] Comscore, «Smartphone Penetration Across the EU5 Countries,» [En línea]. Available: https://www.comscore.com/lat/Prensa-y-Eventos/Infographics/Smartphones-Reach-Majority-in-all-EU5-Countries.

[2] L. Á. Gallego, Control y Monitorización de hábitats para animales mediante Raspberry-Pi, Fiware, Spring y Android, vol. 2, Sevilla: Universidad de Sevilla, 2019.

[3] S. M. Contioso, Aplicación Android y Servicio Web Spring para la monitorización de datos obtenidos en un vehículo haciendo uso de la plataforma FIWARE, Sevilla: Universidad de Sevilla, 2020, p. 12.

[4] Google, «Chrome Developers Docs,» [En línea]. Available: https://developer.chrome.com/docs/devtools/.

[5] S. Overflow, «Stack Overflow Developer Survey,» [En línea]. Available: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-integrated-development-environment.

[6] Node.js, «Acerca de Node.js,» [En línea]. Available: https://nodejs.org/es/about/.

[7] Express, «Expressjs Home,» [En línea]. Available: https://expressjs.com/.

[8] npmjs, «About npm,» [En línea]. Available: https://docs.npmjs.com/about-npm.

[9] Fiware.org, «About Fiware,» [En línea]. Available: https://www.fiware.org/about-us/.

[10] FIWARE, «Ecosistema FIWARE,» [En línea]. Available: https://fiware-training.readthedocs.io/es_MX/latest/ecosistemaFIWARE/ocb/.

[11] FIWARE, «FIWARE-CYGNUS home,» [En línea]. Available: https://fiware-cygnus.readthedocs.io/en/latest/index.html.

[12] MySQL, «MySQL 8.0 Reference Manual,» [En línea]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[13] MongoDB, «¿Qué es MongoDB?,» [En línea]. Available: https://www.mongodb.com/es/what-is-mongodb.

[14] B. Universidad de California, «El diseño de Postgres,» [En línea]. Available: https://dsf.berkeley.edu/papers/ERL-M85-95.pdf.

[15] Docker, «Docker get started,» [En línea]. Available: https://docs.docker.com/get-started/overview/.

[16] FIWARE, «Smart data models,» [En línea]. Available: https://github.com/smart-data-models/dataModel.Transportation/blob/master/Vehicle/doc/spec_ES.md.

[17] React, «reactjs,» [En línea]. Available: https://reactjs.org/docs/hooks-intro.html.

[18] m. y. a. u. Ministerio de transportes, «https://www.mitma.gob.es/transporte-terrestre/inspeccion-y-seguridad-en-el-transporte/tiempos-de-conduccion-y-descanso/conduccion/tiempos-de-conduccion#:~:text=El%20tiempo%20de%20conducci%C3%B3n%20en,el%20m%C3%A1ximo%20de%2090%20horas.,» [En línea].

[19] chocolatey, «https://chocolatey.org/,» [En línea].

[20] VMware, «VMware docs,» [En línea]. Available: https://docs.vmware.com/en/VMware-Workstation-Pro/16.0/com.vmware.ws.using.doc/GUID-BAFA66C3-81F0-4FCA-84C4-D9F7D258A60A.html.