Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Ingeniería de Software

Aplicación para provisión de datos de transporte público en tiempo real

Alumno:
Eduardo Velasco Martínez

Tutores:
Rodrigo Sanz Muñoz
César Vaca Rodríguez

Resumen

En este proyecto se ha desarrollado una aplicación para el aprovisionamiento de datos de transporte en tiempo real en Google a través de ficheros en formato GTFS (General Transit Feed Specification). La aplicación consiste en un microservicio integrado en un entorno con otros microservicios relacionados con el transporte.

El objetivo del microservicio desarrollado en este proyecto es realizar solicitudes a otros microservicios del entorno para obtener información acerca del estado de los trayectos, las expediciones y los vehículos. Y posteriormente transformar dicha información en ficheros GTFS para finalmente publicarlo en Google de forma automática. Estos ficheros siguen el estándar establecido por Google, lo que garantiza la compatibilidad y facilidad de uso de los datos.

El microservicio es capaz de suscribirse y realizar solicitudes a los microservicios necesarios. También es capaz de realizar estas solicitudes de manera inteligente para minimizar al máximo el consumo de recursos de los otros microservicios.

Además, puede funcionar de manera automática y autónoma. Para ello genera y publica periódicamente los ficheros con las actualizaciones de la información en tiempo real. Esta labor lo realiza con la frecuencia y rapidez suficiente para que dicha información sea precisa y actual.

Con este desarrollo se espera generar una mejora en la planificación de viajes y cálculo de rutas de cara a los usuarios que utilicen Google para ello. Google podrá utilizar información más precisa y actualizada para generar mejores resultados a sus usuarios gracias al aprovisionamiento de estos ficheros por parte del microservicio desarrollado.

En conclusión, la aplicación ofrece una solución eficiente y actualizada para acceder a información precisa sobre el transporte público, mejorando así la experiencia de los usuarios y fomentando el uso de este medio de transporte.

Abstract

In this project an application has been developed for the provisioning of real-time transport data in Google through files in GTFS (General Transit Feed Specification) format. The application consists of a microservice integrated in an environment with other transport-related microservices.

The objective of the microservice developed in this project is to make requests to other microservices in the environment to obtain information about the status of journeys, shipments and vehicles. And then transform this information into GTFS files to finally publish it automatically in Google. These files follow the standard established by Google, which guarantees the compatibility and ease of use of the data.

The microservice is able to subscribe and make requests to the necessary microservices. It is also able to make these requests intelligently to minimize the resource consumption of the other microservices as much as possible.

In addition, it can operate automatically and autonomously. For this purpose, it generates and periodically publishes the files with the information updates in real time. This task is performed with sufficient frequency and speed to ensure that the information is accurate and up to date.

With this development it is expected to generate an improvement in travel planning and route calculation for users who use Google for this purpose. Google will be able to use more accurate and updated information to generate better results for its users thanks to the provisioning of these files by the developed microservice.

In conclusion, the application offers an efficient and up-to-date solution for accessing accurate information on public transport, thus improving the user experience and encouraging the use of this means of transport.

Tabla de contenidos

Este documento se estructura de la siguiente forma:

- **Capítulo 1 Introducción:** Descripción general del proyecto. Se explicará el contexto en el que se desarrolla, se expondrá la motivación detrás del mismo y se presentarán sus objetivos.
- **Capítulo 2 Marco teórico:** Exposición detallada de los conceptos teóricos necesarios para una mejor comprensión del documento. Servirá de guía de referencia para futuras menciones de los conceptos específicos del tema a tratar.
- **Capítulo 3 Tecnologías utilizadas:** Exposición de las tecnologías utilizadas para el desarrollo. Se conocerán todas las funcionalidades de cada herramienta y se expondrán las posibilidades que otorga cada una.
- **Capítulo 4 Análisis:** Estudio del contexto y las condiciones en las que se realizará el proyecto. Se determinarán los requisitos en base a las expectativas de los stakeholders y se realizará una estimación de costes y una planificación para finalmente estudiar la viabilidad técnica y económica del proyecto.
- **Capítulo 5 Diseño:** Presentación del modelado del proyecto. Se describirá la arquitectura del sistema, los diferentes componentes y su interacción, así como los diagramas de flujo, diagramas de clase y otros artefactos de diseño utilizados.
- **Capítulo 6 Implementación:** Pasos seguidos para llevar a cabo el diseño propuesto para la creación de la aplicación y profundización en detalles y aspectos importantes.
- **Capítulo 7 Pruebas:** Explicación de pruebas realizadas. Se trata con detalle la validación de feeds GTFS y las herramientas disponibles para ello.
- **Capítulo 8 Seguimiento del proyecto:** Desarrollo de la planificación durante el transcurso del proyecto. Se observará la similitud entre la planificación del proyecto y la evolución real de las tareas del desarrollo.
- **Capítulo 9 Conclusiones:** Análisis del proyecto y extracción de ideas y conceptos extraídos de una visión general obtenida al finalizar el desarrollo del proyecto.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

En este capítulo introductorio se dará una visión general de los temas y objetivos que se tratan en este proyecto. En concreto se expondrá el contexto en el que se desarrolla este proyecto, la motivación que ha llevado a este desarrollo y los objetivos que se esperan conseguir después de la realización del mismo.

Finalmente se mostrará la estructura de esta memoria junto con una breve explicación del contenido de cada capítulo para ofrecer una visión general de la memoria.

1.1. Contexto

En el ámbito de la movilidad urbana, el acceso en tiempo real a la información de transporte se ha convertido en una herramienta fundamental para que los usuarios planifiquen y tomen decisiones en el área de la movilidad urbana. La tecnología conocida como GTFS Realtime (General Transit Feed Specification Realtime) se ha convertido en el estándar de la industria para la transmisión de datos de transporte en tiempo real. Con la ayuda de esta especificación, las agencias de transporte público pueden intercambiar información actual sobre rutas, horarios y ubicaciones de vehículos, brindando a los usuarios datos precisos.

En este contexto, el presente trabajo de final de grado se centra en la creación de una aplicación para el aprovisionamiento de datos de transporte en tiempo real utilizando GTFS Realtime. La aplicación se basará en la especificación GTFS Realtime para recibir los datos de transporte proporcionados por las agencias de transporte público.

Para su desarrollo se utilizarán herramientas actuales y se buscará garantizar la eficacia, robustez, seguridad, escalabilidad y eficiencia para ofrecer un servicio de calidad. El desarrollo de esta aplicación en una empresa líder en diseño, desarrollo, implementación y despliegue de Sistemas Inteligentes de Transporte permite cumplir de un modo adecuado estas propiedades.

Este trabajo de final de grado busca contribuir al avance de la tecnología en el ámbito

del transporte público y mejorar la experiencia de los usuarios al utilizar este medio de transporte. Asimismo, pretende sentar las bases para futuras investigaciones y desarrollos en el campo de la movilidad urbana, explorando nuevas formas de aprovechar los datos en tiempo real para optimizar la planificación y operación del transporte público.

1.2. Motivación

La necesidad de mejorar la experiencia del usuario al usar el transporte público ha llevado al desarrollo de una aplicación para la entrega en tiempo real de datos de transporte utilizando GTFS Realtime. Para planificar eficientemente sus viajes, los usuarios, hoy en día, dependen en gran medida de información precisa y actualizada sobre rutas, horarios y ubicaciones de vehículos.

Actualmente no disponen de ningún tipo de información o de información estática, la cual suele distar de la realidad debido al entorno tan variable en el que se encuentran los medios de transporte público. Esto puede servir para ofrecer a los usuarios una información aproximada pero no suple de forma adecuada las necesidades de los usuarios.

La motivación detrás de este proyecto es la importancia de proporcionar a los usuarios acceso a la información en tiempo real para que puedan planificar sus viajes de manera inteligente. Los usuarios podrán determinar con precisión los tiempos de espera, planificar sus rutas y evitar retrasos innecesarios.

Por otro lado, se busca aumentar la fiabilidad y eficacia del transporte público. Las agencias podrán optimizar los servicios de transporte y modificarlos en respuesta a las necesidades de los usuarios al tener acceso a datos precisos sobre la demanda y ubicación de los vehículos.

Esta tecnología abre la puerta a una gran innovación en el transporte público y acerca al presente la realidad de un transporte público sostenible e inteligente con el cual ofrecer un mejor servicio a sus usuarios a un menor coste.

1.3. Objetivos

El objetivo principal de este proyecto es crear una aplicación que utilice GTFS Realtime para aprovisionar datos de transporte en tiempo real. A continuación, se muestra una lista con los objetivos específicos:

1. Crear una aplicación que se integre con un grupo de microservicios destinados a la gestión y desarrollo del transporte público. Deberá ser capaz de comunicarse y obtener información de los microservicios necesarios para su posterior utilización.
2. Deberá ser un microservicio con endpoints a través de los cuales se puedan hacer solicitudes y obtener la información en tiempo real solicitada.

3. Utilizará feeds de GTFS Realtime para proporcionar la información en tiempo real. Inicialmente esta información debe ser recibida y reconocida por Google para que pueda actualizar la información en sus servicios y ofrecerlo a sus usuarios en aplicaciones de la compañía, como Google Maps. También se espera que Google utilice dicha información para un mejor cálculo de rutas de forma que se aumente la eficiencia y se reduzcan los tiempos de desplazamiento de los usuarios.

4. Publicará la información de manera actualizada en los servicios de Google en forma de feeds. Esta labor deberá realizarla de forma automática sin necesidad de ningún tipo de acción por parte de ningún usuario.

5. La información debe ser precisa y el microservicio debe funcionar de manera eficiente para que utilice la menor cantidad de recursos posibles manteniendo un correcto funcionamiento del mismo.

Capítulo 2

Marco teórico

Este capítulo tiene como propósito brindar una sólida base de conocimientos previos en esta área de estudio. Permitiendo así una mejor comprensión del contenido de este trabajo por parte del lector. Para ello, se proporcionará una visión global de los conocimientos existentes en el área de estudio, los cuales sentarán las bases de este trabajo.

En algunos casos se brindará una visión más detallada y profunda de algunos conceptos debido a la importancia y relevancia de las especificaciones en el trabajo.

A través de este recorrido teórico, se sentarán las bases para una comprensión más completa y crítica de la investigación presentada en este TFG, contribuyendo así al avance del conocimiento en el sistema GTFS de Google enfocado a la parte de tiempo real.

2.1. GTFS

El General Transit Feed Specification (GTFS) es un estándar de datos ampliamente utilizado en el ámbito del transporte público. Se trata de una especificación de datos que permite publicar sus datos de tránsito: agencias de transporte, las paradas, las rutas, los horarios y las reglas de viaje.[1]

Esta especificación facilita el intercambio de datos y la interoperabilidad entre diferentes sistemas y aplicaciones relacionadas con el transporte público. Para ello, se utilizan archivos de texto con un formato de texto específico denominados Feeds. Este formato permite una forma eficiente de transferencia de los datos, lo que conlleva a una reducción de costes considerable.[2]

A pesar de que el estándar está bien definido, la calidad y consistencia puede variar significativamente según la agencia de transporte, generando complicaciones a la hora del manejo de estos ficheros a través de diversas fuentes dado que puede variar según la agencia

de transporte. También puede llegar a dificultar la interoperabilidad y la integración de los datos en diferentes sistemas y aplicaciones.

Respecto al usuario final, permite una mejor planificación de viajes, una reducción en los tiempos de espera y una mayor precisión respecto a la hora de llegada y salida de los transportes.[3]

2.2. GTFS Estático

El GTFS Estático conforma una de las dos partes que ofrece GTFS. Esta parte se encarga de proporcionar la información estática de las agencias de transporte, es decir, aquella que no cambia con frecuencia.[4]

Para ello la información se organiza en una serie de ficheros en formato *txt* los cuales se describen con más detalle a continuación, (no se profundizará en exceso dado que el objeto de este trabajo es la parte de tiempo real):

- **Obligatorio:**
  - **agency.txt:** Define las empresas de transporte público que tienen un servicio representado en el conjunto de datos.
  - **stops.txt:** Define las paradas en las que los vehículos recogen o dejan pasajeros. También indica las estaciones y las entradas de las estaciones.
  - **routes.txt:** Define las rutas de transporte público. Una ruta es un grupo de viajes que se muestra a los pasajeros como un solo servicio.
  - **trips.txt:** Define los viajes para cada ruta. Un viaje es una secuencia de dos o más paradas que ocurre durante un período específico.
  - **stop times.txt:** Proporciona las horas en las que un vehículo llega a una parada y sale de ella en cada viaje.
- **Condicionalmente obligatorio:**
  - **calendar.txt:** Contiene información sobre los días en que se proporciona el servicio de transporte. También incluye información sobre excepciones de servicio, como días festivos.
  - **calendar_dates.txt:** Contiene todas las fechas de servicio, y si el servicio va a estar activo o no ese día. Si se incluyera el archivo *calendar.txt*, este archivo no sería obligatorio.
  - **feed_info.txt:** Incluye metadatos sobre el conjunto de datos, incluida la información sobre el publicador, la versión y el vencimiento.
- **Opcional:**
  - **fare_attributes.txt:** Define la información sobre las tarifas correspondientes a las rutas de una empresa de transporte público.

- **fare_rules.txt:** Define las reglas para aplicar la información sobre tarifas de los itinerarios.
- **shapes.txt:** Define las reglas para las rutas de viaje de los vehículos, también conocido como alineamientos de rutas.
- **frequencies.txt:** Indica el tiempo entre viajes para los servicios basados en intervalos o una representación comprimida de un servicio con horarios fijos.
- **transfers.txt:** Reglas para establecer conexiones en los puntos de transbordo entre rutas.
- **pathways.txt:** Define los recorridos que conectan ubicaciones dentro de las estaciones.
- **levels.txt:** Describe los niveles dentro de las estaciones.
- **translations.txt:** Contiene información traducida sobre una empresa de transporte público.
- **attributions.txt:** Especifica las atribuciones que se aplican al conjunto de datos.

Como se puede observar para la parte estática se utilizan ficheros txt siguiendo el estándar que especifica GTFS para los datos estáticos. Una vez generados los ficheros necesarios, se comprimen en un archivo ZIP, el cual comprende una versión del feed. El feed es el producto final que debe ser publicado utilizando cualquiera de las opciones disponibles para que otros servicios, como en nuestro caso Google, puedan consumirlo y utilizarlo en sus aplicaciones.[5][6]

2.3. GTFS Realtime

El GTFS en Tiempo Real (GTFS RT) es la segunda parte que ofrece GTFS. Esta parte permite la transmisión de información actualizada en tiempo real sobre el transporte público. En concreto, brinda información dinámica y en tiempo real sobre actualizaciones de horarios, demoras, información geográfica asociada, modificaciones de rutas y otros eventos pertinentes, en contraste con el GTFS estático, que se concentra en los datos estáticos del sistema.[7]

Esto hace posible que los consumidores obtengan actualizaciones en tiempo real sobre la disponibilidad y el estado de los servicios de transporte público. Permitiendo así, una mejor planificación y tiempo de reacción por parte de los usuarios ante posibles imprevistos o cambios de última hora en el transporte público.

Para esta labor se utilizan Feeds, los cuáles siguen la siguiente estructura:[8]

header:

gtfs_realtime_version: Versión de GTFS Realtime utilizada.

incrementality: Indica si el feed es completo o incremental.

timestamp: Marca de tiempo del momento en que se generó el feed.

entity (repetible):

id: Identificador único de la entidad.

is_deleted: Indica si la entidad ha sido eliminada.

trip_update (opcional):

trip (opcional):

trip_id: ID del viaje.

start_time: Hora de inicio del viaje.

start_date: Fecha de inicio del viaje.

schedule_relationship: Relación del horario del viaje.

stop_time_update (repetible):

stop_sequence: Secuencia de paradas.

stop_id: ID de la parada.

arrival (opcional):

time: Hora estimada de llegada.

delay: Retraso en minutos.

uncertainty: Incertidumbre en minutos.

departure (opcional):

time: Hora estimada de partida.

delay: Retraso en minutos.

uncertainty: Incertidumbre en minutos.

schedule_relationship: Relación del horario de la parada.

trip (opcional):

trip_id: ID del viaje.

start_time: Hora de inicio del viaje.

start_date: Fecha de inicio del viaje.

schedule_relationship: Relación del horario del viaje.

vehicle (opcional):

id: ID del vehículo.

label: Etiqueta del vehículo.

license_plate: Matrícula del vehículo.

position (opcional):

latitude: Latitud de la posición.

longitude: Longitud de la posición.

bearing: Rumbo del vehículo.

odometer: Odómetro del vehículo.

speed: Velocidad del vehículo.

current_stop_sequence: Secuencia de paradas actual.

stop_id: ID de la parada actual.

current_status: Estado actual del vehículo.

timestamp: Marca de tiempo del estado actual.

En este caso se utiliza un conjunto de mensajes y actualizaciones específicos que se transmiten mediante protocolos de transferencia de datos. GTFS Realtime de Google utiliza protobuf, el cual es un mecanismo de serialización de datos desarrollado por Google que permite suplir la necesidad de velocidad y cantidad de transferencia de datos en tiempo real.

2.4. Relación entre GTFS Estático y GTFS Realtime

GTFS Realtime y GTFS Estático están estrechamente relacionados conformando un sistema completo suministro de información de medios de transporte. GTFS Realtime funciona sobre GTFS estático y pueden llegar a producirse fallos o un mal funcionamiento si la información en tiempo real no se vincula de forma correcta con la información estática.

El GTFS Estático establece la base inicial de los datos sobre los cuales GTFS Realtime completa con información actualizada en tiempo real. Por lo tanto, si no se enlaza de forma correcta la información en tiempo real con la información estática a la que corresponde se producirá un error, debido a que, en este caso Google, no puede relacionar dicha información con ninguna información estática. En consecuencia, se puede llegar a una situación de inconsistencia lo que produciría una incoherencia de los datos. También existe la posibilidad de referenciar datos estáticos no correspondientes a la información de tiempo real debido a una referencia a una información estática diferente. En este caso puede no obtenerse un error, pero los datos serán incoherentes y no tendrán ningún tipo de sentido, utilidad y uso.

Finalmente, al igual que en otros capítulos, es importante la eficiencia y la rapidez para una buena sincronización y actualización de los datos para ofrecer la información más precisa posible.

2.5. Ventajas y desventajas de GTFS Realtime

A pesar de su potencial se debe tener claro cuáles son las ventajas y desventajas de este sistema para completar el conocimiento acerca del manejo de esta herramienta y asegurar si esta herramienta es la adecuada para realizar la labor.[7][2]

Ventajas de GTFS Realtime:

- Permite a las agencias de transporte público proporcionar actualizaciones en tiempo real sobre su flota a los desarrolladores de aplicaciones.
- Es una extensión de GTFS, un formato de datos abierto para horarios de transporte público y datos geográficos asociados.
- Fue diseñado para ser fácil de implementar, interoperable con GTFS y enfocado en la información del pasajero.
- Permite presentar el estado real de la flota.

- Facilita la implementación de aplicaciones, productos y servicios.

Desventajas de GTFS Realtime:

- La información dentro de un feed de GTFS Realtime no debe ser más antigua de 90 segundos para actualizaciones de viajes y posiciones de vehículos y no más antigua de 10 minutos para alertas de servicio.
- Si hay múltiples instancias de un feed de GTFS Realtime detrás de un balanceador de carga, cada instancia puede estar publicando información ligeramente fuera de sincronización.
- El feed debe actualizarse regularmente, preferiblemente cada vez que lleguen nuevos datos del sistema de ubicación automática de vehículos.

2.6. Feed de GTFS RT

El Feed de GTFS RT es el conjunto de datos que conforman el paquete que se envía con la información en tiempo real. Estos datos se estructuran como se ha presentado en la sección 2.3.

Se puede observar que siguen una estructura jerárquica que facilita la encapsulación de los datos y permite una mejor visión de la relación entre los mismos. También cabe destacar que no todos los campos son obligatorios, aunque cuantos más datos se envíen, la información proporcionada será más completa y precisa.

Algunos de los campos tienen cardinalidad superior a uno, como por ejemplo las entidades (entity), dado que se puede enviar información acerca de varios elementos en el mismo Feed. Esto tiene sentido debido a que gracias a enviar toda la información de forma simultánea se obtiene un gran aumento en la eficiencia y una reducción de carga de los equipos involucrados en estas transferencias.

Para el transporte de estos datos se utilizan Búferes de Protocolo o comúnmente abreviado como Protobuf.

2.6.1. Protobuf

Protobuf o Buffer de Protocolo es una forma creada y desarrollada por Google para serializar datos estructurados. Este mecanismo otorga un gran abanico de posibilidades a los desarrolladores y permite que sea utilizado de una forma sencilla y eficiente por los desarrolladores.

Otorga un formato adecuado tanto para el tráfico de red efímero como para el almacenamiento de datos más persistentes. Su formato también puede ser ampliado con nueva

información, si fuera necesario, sin invalidar los datos existentes y sin necesidad de modificar el código.

Este mecanismo es muy utilizado tanto en la comunicación de sistemas distribuidos como en el almacenamiento de datos persistentes. Los mensajes y servicios se describen mediante archivos *.proto*.

El protocompilador se ejecuta en tiempo de compilación para generar el código en varios lenguajes de programación. Permitiendo así, la manipulación del buffer de protocolo correspondiente.

Los principales lenguajes de programación que soporta son los siguientes:

- C++
- C#
- Java
- Kotlin
- Objective-C
- PHP
- Python
- Ruby

Además de soportar gran variedad de lenguajes, también permite el soporte continuo de cambios. Esto es muy importante porque los búferes de protocolo se utilizan ampliamente en todo tipo de servicios de Google y por lo tanto sus datos pueden persistir a lo largo del tiempo. Gracias al soporte continuo se mantiene la compatibilidad con versiones anteriores, incluida la adición de nuevos campos y la eliminación de campos existentes.[9]

A continuación, se muestra un diagrama de flujo de trabajo de los búferes de protocolo para ofrecer un esquema general de su funcionamiento.

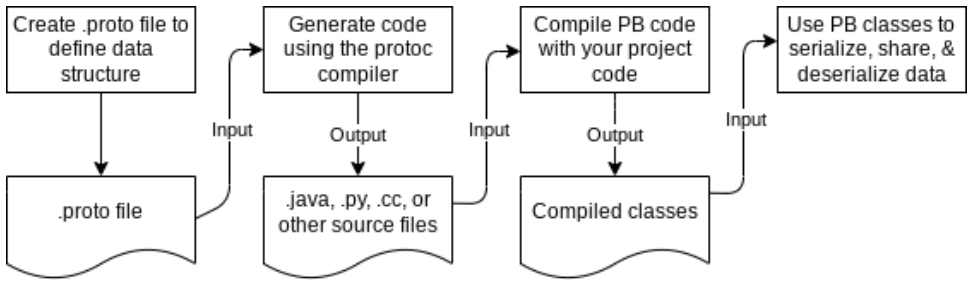

Figura 2.1: Diagrama de flujo de trabajo de Protobuf

2.7. Scrum

Scrum es un marco de trabajo ágil utilizado en la gestión de proyectos, principalmente proyectos de desarrollo de software. Se basa en una metodología de trabajo constante, iterativa e incremental.

Su objetivo principal es mejorar la productividad y la calidad del producto a través de una estructura organizada y un enfoque colaborativo, principalmente en proyectos complejos donde los requisitos son muy variables y no se encuentran muy definidos, lo cual generalmente conlleva modificaciones en los requisitos y la planificación del proyecto durante el desarrollo del mismo. Ofrece una gran adaptabilidad y flexibilidad para poder hacer frente a estos fallos de una forma sólida y eficiente sin hacer peligrar el futuro del proyecto.

El trabajo a realizar se distribuye en períodos fijos de tiempo llamados sprints. Esta duración se puede elegir, pero para que este mecanismo funcione de manera adecuada suele establecerse una duración de sprint de entre unas 2 y 4 semanas. Al final de cada período se obtiene un resultado completo.[10][11]

Los elementos principales de Scrum son los siguientes:

- **Roles:** Define tres roles principales en los que cada uno tiene responsabilidades específicas en el proyecto.
  - **Product Owner:** Es el responsable de maximizar el valor del producto, representar los intereses de los stakeholders y asegurar el éxito general del producto que se está desarrollando.
  - **Scrum Master:** Es el líder del equipo Scrum. Su labor es liderar y asegurar el funcionamiento de los miembros del equipo dentro del marco de trabajo. Para ello ayuda al equipo si es necesario y se encarga de la organización del proyecto.
  - **Equipo de desarrollo:** Grupo de profesionales que trabajan juntos para llevar a cabo el desarrollo del proyecto. Son los encargados de completar las tareas del Sprint Backlog.
- **Artefactos:** Son aquellos elementos utilizados para organizar y gestionar el trabajo. Consta de los siguientes elementos:
  - **Product Backlog:** Lista ordenada de las tareas necesarias para llevar a cabo el desarrollo del producto. Esta lista se ordena por prioridades y se basa en las expectativas y requisitos del cliente.
  - **Sprint Backlog:** Selección de tareas pendientes que el equipo planea finalizar durante un sprint. Debe incluir componentes importantes como las historias de usuarios y las descripciones de las tareas. Es flexible y puede evolucionar a lo largo del sprint.
  - **Incremento:** Suma de todos los elementos del Product Backlog completados durante el Sprint presente y valor de los incrementos de todos los Sprints anteriores. El Incremento es presentado en la Sprint Review, que es una sesión colaborativa entre Equipo Scrum y stakeholders, en la que se obtiene feedback sobre el producto y se adapta el Product Backlog.

- **Eventos:** Bloques de tiempo de duración fija periódicos que tienen como finalidad crear regularidad y consistencia en el proceso, evitando la necesidad de reuniones innecesarias que solo entorpecen los procesos. Los principales eventos son:
  - **Reunión Diaria:** De corta duración, generalmente unos 15 minutos. Su objetivo es promover la comunicación y la transparencia dentro del equipo. Cada miembro expone las tareas realizadas durante el día anterior, las tareas que planea hacer hoy y si hay algún impedimento que le impida avanzar en el trabajo. De esta forma el equipo puede estar mejor coordinado y colaborar entre ellos si algún miembro necesitara algo o tuviera algún problema.
  - **Revisión del Sprint:** Se realiza al final del sprint. En ella el equipo presenta el trabajo realizado durante el Sprint y se obtiene feedback de los stakeholders para inspeccionar y evaluar el producto a fin de ajustar el Product Backlog.
  - **Retrospectiva del Sprint:** También se realiza al final de cada sprint. En ella se reflexiona sobre el desarrollo del Sprint y se decide en conjunto cómo mejorar los procesos. El objetivo es mejorar continuamente el proceso de sprints y garantizar que se incorporen los aprendizajes clave para el proceso en los sprints posteriores.
- **Principios:** Scrum se rige por una serie de principios:
  - Transparencia
  - Inspección
  - Adaptación continua
  - Colaboración
  - Enfoque en la entrega de valor

2.8. Transmodel

Transmodel es una referencia de modelos de datos para el transporte público en Europa que desempeña un papel importante en la mejora de los sistemas de información y gestión de servicios relacionados con el transporte público. Su objetivo principal es mejorar los sistemas de información de transporte público y gestión de servicios.

Para ello proporciona una representación de conceptos y estructuras de datos comunes relacionadas con el transporte público. Facilita la interoperabilidad entre los sistemas de procesamiento de información de operadores y agencias de transporte mediante el uso de estructuras, definiciones y significados coincidentes para los datos transferidos. Muy útil en casos en los que diferentes aplicaciones se conectan a una misma organización.

Establece terminología consistente para describir la mayoría de conceptos del transporte público. De esta forma se ofrece una sencilla conversión y adaptación a los diferentes idiomas.

Esto permite unificar y normalizar la construcción de sistemas de información de transporte público en Europa. Constituye la base sobre la que se estructuran sistemas como el de GTFS creado por Google.

Transmodel ha sido fundamental para el desarrollo de varios modelos de datos nacionales concretos y estándares europeos, incluidos los estándares europeos de intercambio de datos de transporte público (NeTEx).[12]

Capítulo 3

Tecnologías utilizadas

3.1. Jira

Jira es una plataforma de gestión de proyectos ágiles desarrollada por Atlassian. Esta plataforma es ampliamente utilizada por los equipos para planificar, supervisar, rastrear y entregar proyectos de manera eficiente y colaborativa. Para ello ofrece una amplia gama de características y funcionalidades para adaptarse a las necesidades de cada equipo.

Posee una gran capacidad de adaptación a diferentes metodologías de trabajo, como Scrum o Kanban. También permite establecer prioridades, la asignación de tareas, realizar un seguimiento de las mismas y trabajar de forma colaborativa en tiempo real.

Su principal ventaja es su versatilidad gracias a la cual se puede integrar con diversas herramientas de Atlassian. Para el desarrollo software destaca la posibilidad de sincronización con Bitbucket, Confluence y Open DevOps. Esta versatilidad e integración permite tener un seguimiento más preciso y detallado de los proyectos. Por otro lado, otorga mayores facilidades a la hora de gestionar los cambios de software en relación con las tareas asignadas.

En resumen, Jira es una plataforma de gestión de proyectos y seguimiento de problemas que proporciona a los equipos las herramientas necesarias para planificar, rastrear y entregar proyectos de manera efectiva. Su flexibilidad, adaptabilidad y capacidad de integración lo convierten en una opción popular para la gestión de proyectos en diversos entornos.[13]

3.2. Scrum Poker

Scrum Poker es una página web gratuita en la que poder planificar tareas de forma online y sin necesidad de instalar ningún software. Permite una mejor planificación en equipos formados por miembros en diferentes ubicaciones geográficas de una forma sencilla. Se basa en la técnica de estimación Planning Poker para equipos Scrum.

Para ello permite la creación de salas virtuales con un ID único a las cuales se pueden unir todos los integrantes del equipo. Una vez dentro de la sala, ofrece una interfaz muy sencilla compuesta por tarjetas con las posibles estimaciones. Dentro de ellas se encuentra una tarjeta con un interrogante que actúa como "comodín" en caso de no saber que estimación elegir.

La idea de esta herramienta es conversar y discutir acerca del contenido de la tarea. Una vez resueltas todas las dudas y preguntas, cada miembro realiza su votación a través de esta herramienta, las votaciones permanecerán ocultas hasta que el líder considere que todos los miembros han votado. En ese momento se revelarán todas las tarjetas y se elegirá la opción más votada, en caso de empate se debatirá y llegará a un consenso entre las opciones.

Finalmente se pueden almacenar las estimaciones realizadas en un registro, lo que permite llevar un registro y utilizarlo para el seguimiento o mejora en futuros sprints o proyectos.[14]

3.3. EasyRetro

EasyRetro es una herramienta online que permite a los equipos hacer un análisis sobre el trabajo en equipo y tener una visión general del progreso de todos los miembros.

Su principal enfoque de uso es en metodologías de trabajo con sprints como puede ser la metodología Scrum 2.7. Gracias a sus diferentes plantillas y estructura, los equipos pueden escribir las cosas que funcionaron bien y los problemas que hubo durante el sprint.

Para ello cada usuario puede crear tarjetas, las cuales se alojan en diferentes columnas según el contenido de la tarjeta. Estas tarjetas son personalizables, aunque generalmente se utilizan tres columnas:

- **Funcionó bien:** Aquellas cosas que fueron bien durante el sprint y se consideran correctas.
- **Funcionó mal:** Problemas que hubo durante el sprint que, generalmente, supusieron una dificultad para sacar el trabajo adelante.
- **A mejorar:** Elementos que no generan, o al menos no en gran medida, un problema para el desarrollo pero que estaría bien tener en cuenta y mejorar a ser posible. No tienen que ser problemas en sí, sino que también pueden ser ideas de mejora respecto a algún ámbito del trabajo.

Estas tarjetas son ocultas y se suelen mostrar cuando todos los miembros del equipo han escrito sus tarjetas. Esto tiene sentido dado que de esta forma los miembros del equipo no quedan condicionados por las respuestas que puedan ver escritas por otros miembros.

Finalmente, después de mostrar las tarjetas se puede elegir que tarjetas poder votar y cuantos votos se otorgan a los miembros. Cada miembro podrá distribuir sus votos de la forma que desee entre las tarjetas que hayan sido elegidas para someterse a votación.[15]

3.4. Confluence

Confluence es una herramienta de colaboración y gestión de conocimiento desarrollada por Atlassian. Está enfocada al uso empresarial y es ampliamente utilizada por las empresas para almacenar todo el conocimiento de la compañía de modo que pueda transmitirse de forma rápida y eficiente entre sus trabajadores. Gracias a su flexibilidad y personalización, se adapta considerablemente bien a las necesidades de cada equipo.

Consiste en un espacio de trabajo estructurado por páginas que actúan como espacios de trabajo virtuales en las que los participantes pueden crear, estructurar, organizar, comentar y compartir cualquier idea o proyecto. Cada página puede contener todo tipo de elementos como imágenes, enlaces, tablas, documentos adjuntos y, por supuesto, texto. Esto permite plasmar mejor el conocimiento apoyándose en bibliografía o elementos gráficos.

Cuenta con un buscador en el cual poder buscar páginas o contenido en las mismas de forma que la información a buscar sea más accesible y los miembros puedan encontrar la información de una forma mucho más rápida. Esto se puede complementar con las relaciones entre páginas que representan una vinculación de la información contenida en las mismas.

Por último, al ser una herramienta de Atlassian, se pueden vincular elementos de otras herramientas de la compañía como pueden ser tareas de Jira. También tiene la opción de añadir gran cantidad de complementos y extensiones que permiten aumentar aún más su funcionalidad e integración con las necesidades específicas de cada equipo.[16]

3.5. Visual Studio 2022

Visual Studio es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft. Principalmente enfocado a desarrolladores de .NET y C#, esta herramienta permite el desarrollo en gran cantidad de lenguajes de programación como C++ y JavaScript.

Su principal característica es la facilidad de escritura de código. Esta herramienta cuenta con IntelliSense, que es un complemento que cuenta con una serie de características que se encargan de proporcionar asistencia y completar el código de forma automática. Esto permite al desarrollador mejorar su rendimiento frente a fallos y agilizar la creación de clases y todo tipo de programación sistemática y general.[17]

También ofrece una gran cantidad de extensiones y complementos desarrollados por la comunidad, que personalizan y aumentan la funcionalidad de la herramienta.

Finalmente, al ser una herramienta de Microsoft, está integrada con los servicios en la nube de Microsoft.[18]

3.6. Bitbucket

Bitbucket es una plataforma de alojamiento de repositorios en la nube y control de versiones basada en Git desarrollada por Atlassian. Esto permite a equipos de desarrolladores almacenar y colaborar en proyectos de programación.

Posee todo tipo de características propias de Git como la posibilidad de crear, manejar, borrar y fusionar ramas, control de versiones. En adición, ofrece otras características, que no son propias de Git, pero son muy útiles para los equipos de desarrollo en ámbitos tan importantes como la seguridad.[19]

Para ello cuenta con la posibilidad de crear repositorios públicos y privados. También tiene un sistema de permisos con el que poder decidir qué tipo de acciones puede realizar cada miembro del equipo sobre el proyecto. Estos permisos se pueden gestionar por grupos lo que hace más simple y fácil el proceso al agrupar usuarios con roles comunes dentro del proyecto.

Por otro lado, cuenta con una amplia gama de integraciones y complementos. Cuenta con una herramienta de integración continua (CI), llamada Pipelines; servicios de gestión de problemas; y servicios de implementación automatizada.

Cuenta con integración dentro del entorno Atlassian, lo que permite la posibilidad de realizar diversas acciones en sincronía con el entorno. Por ejemplo, la posibilidad de crear y relacionar ramas desde una tarea de Jira.[20]

3.7. Sourcetree

Sourcetree es un programa que permite el manejo de repositorios Git o Mercurial a través de una interfaz de usuario (GUI).

También desarrollado por Atlassian, está integrado en el ecosistema de la compañía, beneficiándose de todas sus ventajas. El aspecto más destacable es la posibilidad de clonar un repositorio de Bitbucket con un solo clic.[21]

3.8. Postman

Postman es un programa gratuito que permite a los desarrolladores probar los endpoints de aplicaciones backend basadas en APIs o API REST.

Para ello consta de una interfaz enfocada a la ejecución de peticiones HTTP en cualquiera de los métodos posibles (POST, GET, PUT, PATCH, DELETE). Además, permite modificar los encabezados y elegir el cuerpo de la solicitud. Estas solicitudes pueden ser agrupadas, organizadas y guardadas en colecciones.

Se pueden crear y ejecutar pruebas automatizadas para realizar validaciones sobre la API objetivo. El tipo puede ser cualquiera de los posibles a realizar sobre una API, como por ejemplo la comprobación del tiempo de respuesta.

Los equipos de desarrollo pueden trabajar de forma colaborativa y Postman permite generar de forma automática documentación interactiva.

Ofrece la posibilidad de integrase con herramientas como Git y Jenkins entre otras.[22]

3.9. Jenkins

Jenkins es una herramienta de integración y entrega continuas (CI/CD) de código abierto desarrollada en Java.

Proporciona un entorno automatizado de compilación, despliegue y pruebas de software con el que poder agilizar y optimizar los procesos de desarrollo.

Cuenta con la capacidad de compilar y realizar pruebas sobre un repositorio de forma automática cada vez que se realice un cambio en el mismo. Puede realizar despliegues en diferentes entornos evitando errores y asegurando que los despliegues sean consistentes.

Como complemento, Jenkins puede enviar notificaciones a sus usuarios acerca del resultado de las compilaciones, despliegues y pruebas. Así los miembros del equipo de desarrollo se mantendrán informados sobre el estado de sus procesos y revisar o arreglar errores que pudieran surgir.[23]

3.10. C#

C# (C Sharp) es un lenguaje de programación orientado a objetos de propósito general. Creado y desarrollado por Microsoft, es uno de los lenguajes al alza en la actualidad. Este lenguaje surge dentro de la iniciativa de .NET de Microsoft, es por esto por lo que C# principalmente está diseñado para trabajar con el framework .NET de Microsoft.

Es uno de los lenguajes para la Infraestructura de Lenguaje Común (CLI), es decir, puede ser ejecutado en diferentes plataformas y sistemas operativos. Su sintaxis y nomenclatura es muy similar a Java y C++, lo que lo reduce considerablemente la curva de aprendizaje para los usuarios que tienen conocimientos de C, C++ o Java.

Permite crear aplicaciones robustas y escalables, brindando soporte para programación concurrente, manejo de excepciones, acceso a bases de datos, creación de interfaces gráficas de usuario, entre otros aspectos. Se utiliza principalmente en el desarrollo de aplicaciones web, aunque puede utilizarse para muchos más propósitos. Su principal punta de lanza es su estrecha integración con .NET, otorgando acceso a una extensa biblioteca de clases y funcionalidades predefinidas que facilitan y simplifican el desarrollo.[24]

Un resumen de las características de C# es:

- Es un lenguaje de programación orientado a objetos.
- Es un lenguaje de programación de alto nivel.
- Es un lenguaje de programación multiplataforma.
- Es un lenguaje de programación fácil de aprender para los usuarios que tienen conocimientos de C, C++ o Java.

3.11. .NET

.NET es una plataforma de desarrollo de software creada por Microsoft que permite la creación y ejecución de aplicaciones. Además, es una plataforma gratuita de código abierto por lo que es transparente para todos los usuarios.

Incluye un framework, el cual contiene herramientas, bibliotecas y lenguajes destinados a un desarrollo eficiente, escalable, robusto y de alto rendimiento. Las herramientas incluyen un potente entorno de desarrollo integrado (IDE), depuradores y entornos de pruebas, entre otros. Todo esto ayuda a agilizar el proceso de desarrollo y mejora la productividad de los desarrolladores. Las bibliotecas cubren una amplia cantidad de áreas, como el acceso a bases de datos, la creación de interfaces de usuario, la seguridad y el manejo de errores. Gracias a esto, los desarrolladores pueden ahorrar más tiempo y esfuerzo al no tener que desarrollar estas funcionalidades desde cero.

También es compatible con varios lenguajes de programación entre los que destacan C#, Visual Basic y F#. Esto permite un avance rápido en desarrolladores y usuarios con experiencia previa en alguno de los lenguajes soportados por esta plataforma.

En adición cuenta con soporte multiplataforma, permitiendo la ejecución de aplicaciones en varios sistemas operativos como Windows, Linux y macOS. Esto incrementa el número de dispositivos compatibles y, por lo tanto, otorga un alcance y flexibilidad mayor en el despliegue de aplicaciones por parte de los desarrolladores.[25]

3.12. Docker

Docker es una plataforma de código abierto con la que poder crear, implementar y ejecutar aplicaciones en contenedores. Un contenedor es un formato de empaquetamiento en el que poder encapsular una aplicación de forma que esta sea una unidad ligera y portátil. Dentro de este contenedor se incluyen, además del código, la bibliotecas y dependencias necesarias para el funcionamiento de la aplicación además de la configuración de la misma.

Para llevar a cabo este proceso utiliza una tecnología basada en la virtualización a nivel del sistema operativo, lo que significa que los contenedores comparten el mismo sistema

operativo subyacente, pero siguen estando aislados y tienen su propio entorno de ejecución. Por lo que se obtiene un aumento del rendimiento y la eficiencia respecto a otras formas de virtualización.

Todo esto lleva a la creación de entornos de desarrollo consistentes y reproducibles debido a que los contenedores son altamente portables y pueden ser ejecutados sobre cualquier sistema operativo que tenga Docker instalado. Gracias a esto se simplifica el proceso de implementación y se reducen los problemas de compatibilidad entre diferentes entornos.

Por último, permite crear y gestionar de forma sencilla múltiples contenedores, de lo cual se obtiene una gran flexibilidad y escalabilidad en el desarrollo de aplicaciones.[26][27]

3.13. Kubernetes

Kubernetes es una plataforma de orquestación de contenedores de código abierto y extensible creada originalmente por Google y actualmente mantenida por la Cloud Native Computing Foundation (CNCF).

Su función principal es implementar, automatizar, escalar y administrar aplicaciones en contenedores en cualquier lugar. Para ayudar a esta labor, permite gestionar de forma eficiente la asignación de recursos, el balanceo de carga, la escalabilidad horizontal y la autorrecuperación de las aplicaciones en contenedores. También cuenta con características avanzadas como la configuración declarativa, la gestión de almacenamiento, la distribución de tráfico y la administración de secretos.

Kubernetes ofrece una forma de despliegue de aplicaciones con independencia de infraestructura. Gracias a esto se pueden realizar portabilidades sencillas de las aplicaciones entre diferentes proveedores de infraestructura. Además, se pueden desplegar aplicaciones en entornos locales, en la nube o híbridos sin necesidad de realizar modificaciones según la infraestructura.

Algunas características a mayores de Kubernetes son:

- Administración automática de la detección de servicios, incorpora equilibrio de carga, realiza un seguimiento de la asignación de recursos y los escala en función del uso de la capacidad de proceso.
- Comprobación constante de recursos para poder recuperar de forma automática aplicaciones mediante un reinicio o replicando contenedores.
- Control total sobre las operaciones y la seguridad. Dando lugar a actualizaciones más rápidas y seguras ahorrando tiempo en la administración de la infraestructura.

Kubernetes puede utilizarse con Docker, aunque no es obligatorio. Utiliza una unidad básica de planificación llamada *pod* (cápsula en español), la cual aumenta el nivel de abstracción pudiendo contener varios contenedores dentro de sí misma. Proporciona un aislamiento

respecto a todo lo que se encuentra fuera del *pod* y a nivel interno garantiza la ubicación común de todos sus elementos dentro del mismo dispositivo y la posibilidad de compartir recursos.

Cada *pod* en Kubernetes es asignado a una única dirección IP (dentro del clúster) que permite a las aplicaciones utilizar puertos sin riesgos de conflictos.[28]

Capítulo 4

Análisis

En este capítulo se presentará la fase de análisis realizada para el desarrollo de este proyecto. Esta fase es fundamental en el desarrollo de cualquier proyecto de software, ya que es la base y la hoja de ruta sobre la que se desarrollarán las fases posteriores.

La fase de análisis permite definir con precisión los intereses de los stakeholders y transformarlos en requisitos concretos y tangibles, que puedan ser valorados por todas las partes y así evitar cualquier tipo de confusión o ambigüedad acerca de las expectativas del proyecto.

Por último, con todo el análisis realizado, podemos valorar la viabilidad del proyecto y ahorrar costes en caso de que el proyecto no sea viable, evitando así el inicio de su desarrollo gracias a un buen análisis.

4.1. Requisitos

En esta sección se describirán los requisitos extraídos después de un estudio y análisis exhaustivo, los cuales establecen las reglas y directrices que debe seguir y cumplir la aplicación, después de su desarrollo, como resultado final.

Estos requisitos son fruto de las entrevistas realizadas con los stakeholders, en las cuales se enfatizó la importancia de obtener una comprensión completa de las necesidades y objetivos del proyecto. Esto se debe a la decisión de no realizar modificaciones en los requisitos durante el desarrollo de la aplicación dadas las funcionalidades y las restricciones presentes. Por lo que deben proporcionar una base sólida y detallada de lo que será el resultado final para satisfacer las expectativas de los stakeholders.

Además, se ha llevado a cabo un análisis exhaustivo de los riesgos y obstáculos que podrían surgir durante el desarrollo, para poder anticiparlos y prevenirlos de manera efectiva. Pero esto lo veremos en el siguiente apartado (4.2).

Requisitos Funcionales

- **RF-1:** La aplicación debe ser capaz de recibir información de los microservicios del ecosistema en el que va a instalarse.
- **RF-2:** La aplicación debe ser capaz de transformar la información recibida acerca de los vehículos y viajes a un fichero en formato de GTFS Realtime.
- **RF-3:** La aplicación debe ser capaz de publicar la información en Google.
- **RF-4:** La aplicación debe realizar actualizaciones periódicas para reflejar los cambios en la información en tiempo real.
- **RF-5:** La aplicación debe permitir a los usuarios consultar y visualizar los datos GTFS Realtime cargados.

Requisitos No Funcionales

- **RNF-1:** La aplicación debe ser capaz de realizar solicitudes de forma eficiente para no saturar los microservicios del ecosistema.
- **RNF-2:** La aplicación debe ser capaz de procesar grandes cantidades de datos de forma eficiente.
- **RNF-3:** La aplicación debe ser escalable para poder gestionar el crecimiento de datos y usuarios sin degradar su rendimiento.
- **RNF-4:** La aplicación debe ser segura de forma que nadie sea capaz de modificar su comportamiento u obtener datos internos privados.
- **RNF-5:** la aplicación debe tener una disponibilidad del 99 % durante el horario de trabajo establecido.
- **RNF-6:** Los informes generados por la aplicación deben ser precisos y estar actualizados en tiempo real.
- **RNF-7:** El tiempo de respuesta de la aplicación para las consultas de datos no debe exceder los 2 segundos.
- **RNF-8:** La tasa de publicación de los feeds GTFS en Google no debe exceder los 5 segundos.

Reglas de Negocio

- **RN-1:** Las actualizaciones y modificaciones de los datos deben reflejarse correctamente en tiempo real y de manera coherente en Google.
- **RN-2:** Los datos deben actualizarse de forma regular y oportuna para mantener la información de transporte público actualizada.
- **RN-3:** La carga de datos GTFS Realtime debe seguir un formato válido y consistente para asegurar la integridad de los datos almacenados.

4.2. Riesgos

Como se ha mencionado en el apartado de requisitos, se ha enfatizado en las reuniones con los stakeholders para reducir al mínimo los posibles contratiempos y tener un plan de actuación para todos los riesgos posibles.

Riesgo	Probabilidad	Impacto	Tipo de acción	Acción a tomar
Pérdida de información	Baja	Alto	Prevención	Toda la información se irá almacenando en un servicio de almacenamiento en la nube y el código se irá repositando.
Fallo de funcionamiento del ordenador portátil	Media	Alto	Correctiva	La empresa cuenta con gran cantidad de ordenadores y se realizaría un reemplazo de forma rápida y efectiva.
Falta de formación	Alta	Medio	Mitigación	Realizar horas extra de formación y consultar dudas o problemas con los miembros del equipo
Cambios en el estándar GTFS	Baja	Medio	Aceptación	Las modificaciones realizadas por Google suelen ser mínimas y suele haber un período de transición en el que versiones antiguas siguen siendo compatibles.
Ausencia del desarrollador	Media	Medio	Mitigación	El desarrollo del proyecto se llevará cabo con la suficiente antelación como para tener muchos días de margen.
Retrasos en las tareas	Media	Bajo	Mitigación	Los retrasos no serán muy grandes y el desarrollo se ha iniciado con mucha antelación por lo que hay mucho tiempo de margen.

Tabla 4.1: Clasificación de Riesgos y Acciones

Como se puede observar, hay una buena protección frente a los posibles riesgos y es muy difícil que el proyecto no pueda ser realizado. Por otro lado, dada al pronto inicio del desarrollo del proyecto, el tiempo disponible es mucho mayor que el esperado para el desarrollo del proyecto lo que evita posibles problemas debido a retrasos o imprevistos.

4.3. Planificación

4.3.1. Metodología de trabajo

La metodología de trabajo elegida para este desarrollo es la metodología ágil Scrum. El funcionamiento de esta metodología se describe con más detalle en el apartado 2.7.

Scrum es una metodología adecuada para el desarrollo de un microservicio porque se enfoca en la entrega temprana y continua de valor, lo que es esencial en un proyecto de desarrollo de software. Además, Scrum es una metodología flexible que se adapta fácilmente a los cambios, lo que es importante en un proyecto de desarrollo de software en el que los requisitos pueden cambiar con cierta frecuencia.

Por otro lado, el desarrollo de este microservicio ha sido realizado en un ámbito empresarial en el cual se utiliza esta metodología y se realiza un trabajo colaborativo. En un proyecto de desarrollo de software, el trabajo en equipo es esencial para lograr los objetivos del proyecto y mantener la motivación del equipo. Además, Scrum ofrece una estructura clara de roles y responsabilidades, lo que ayuda a evitar conflictos y malentendidos en el equipo.

Como se explica en el apartado 2.7, esta metodología otorga cierta flexibilidad respecto a la organización del tiempo. En este caso, se ha decidido utilizar sprints de una duración de dos semanas. A lo largo de los mismos se realizará una reunión diaria y al finalizar el sprint se realizarán tres reuniones.

La primera reunión será una revisión del sprint, en la que se realizarán demostraciones de lo realizado durante el sprint. Solo se presentarán aquellas tareas que estén completadas, dejando fuera aquellas que, aunque parezcan completadas, todavía estén pendientes de aprobación.

La segunda reunión será una retrospectiva del sprint. En ella, todos los miembros del equipo expondrán su perspectiva sobre el desarrollo del sprint a través de la herramienta EasyRetro 3.3. Después, cada uno expondrá sus puntos ante el grupo y se hablará sobre ellos, aportando consejos y soluciones si fuera necesario. Para finalizar, se elegirán aquellas tarjetas de acción que se consideren importantes y se votarán. Se consideran tarjetas de acción aquellas que expongan un problema o un aspecto a mejorar y de las cuales se pueda extraer una tarea para mejorarlo. No se crearán tareas en el sprint para aquellos problemas que se salgan fuera del alcance del equipo ni para tarjetas de elementos positivos, en los que no se espera ningún cambio debido a su correcto funcionamiento. Cada participante tendrá varios votos, los cuales distribuirá a su decisión personal entre las opciones disponibles. Una vez obtenidos los resultados, se creará una tarea a partir del elemento de acción más votado para incluir en sprints futuros en los que se solucione.

La tercera reunión será una reunión de inicio del sprint. En ella se explicarán con más detalle las nuevas tareas que debe realizar cada miembro del equipo para que los miembros tengan un mejor contexto. Después de cada explicación, se debatirá acerca de lo que puede conllevar la realización de la misma y después se realizará una votación para estimar los puntos de historia a través de la herramienta Scrum Poker 3.2. Cada punto de historia se supone como una jornada de trabajo de ocho horas, por lo que al ser sprints de dos semanas se asignan como mucho diez puntos de historia en condiciones normales. Las tareas se agrupan por miembro para que cada vez que se termine con las tareas de un miembro, el Scrum Master decida qué tareas asignar para dicho sprint y cuáles enviar a la parte superior del Backlog.

4.3.2. Planificación de tareas y sprints

Siguiendo la metodología de trabajo Scrum se ha dividido el proyecto en subtareas, han sido estimadas y se han distribuido a lo largo de los sprints. De esta forma se obtiene una idea general del tiempo que supondrá llevar a cabo este proyecto y ver si su evolución a lo largo del desarrollo es correcta.

El desarrollo será llevado a cabo por una persona contratada como ingeniero de software. Aunque el trabajo sea realizado dentro de un equipo de desarrollo, el resto de partes realizarán tareas complementarias en microservicios adyacentes que trabajarán en sincronía con este microservicio.

En primer lugar, se ha creado una planificación general sin dividir las tareas en sprints

para obtener una estimación de la duración total del proyecto. Como se puede observar a continuación en la tabla, si se suman todas las duraciones, obtenemos un total de 47,5 días, que se traduce en 380 horas de trabajo dado que las jornadas utilizadas han sido de 8 horas.

Task Name	Duration	Start	Finish	Predecessors
Análisis del microservicio GTFS RT	**4 days**	**Mon 01/05/23**	**Thu 04/05/23**	
Leer documentación GTFS	2 days	Mon 01/05/23	Tue 02/05/23	
Determinar requisitos del microsevicio	1 day	Wed 03/05/23	Wed 03/05/23	2
Diseñar estructura del microservicio	1 day	Thu 04/05/23	Thu 04/05/23	3
Desarrollo del microservicio	**21,5 days**	**Fri 05/05/23**	**Mon 05/06/23**	**1**
Crear microservicio GTFS RT	2 days	Fri 05/05/23	Mon 08/05/23	
Eliminar base de datos del microservicio	1 day	Mon 15/05/23	Mon 15/05/23	6
Crear endpoints	2 days	Tue 09/05/23	Wed 10/05/23	6
Configurar contenedor Docker	1 day	Thu 11/05/23	Thu 11/05/23	8
Configurar Kubernetes	1 day	Fri 12/05/23	Fri 12/05/23	9
Configurar Jenkins	1 day	Tue 16/05/23	Tue 16/05/23	10
Crear interfaz	0,5 days	Wed 17/05/23	Wed 17/05/23	6;7;8;9;10;11
Implementar interfaz	**13 days**	**Wed 17/05/23**	**Mon 05/06/23**	**12**
Añadir y configurar clientes	1 day	Wed 31/05/23	Thu 01/06/23	
Crear cachés	2 days	Wed 17/05/23	Fri 19/05/23	
Hacer Publish Subscribe a Topology	2 days	Fri 19/05/23	Fri 26/05/23	
Hacer Vehicle Position	2 days	Thu 01/06/23	Mon 05/06/23	15;16
Hacer Trip Update	3 days	Fri 26/05/23	Wed 31/05/23	15;16
Pruebas	**8 days**	**Mon 05/06/23**	**Thu 15/06/23**	
Pruebas de carga, rendimiento y robustez	3 days	Mon 05/06/23	Thu 08/06/23	13
Pruebas de integración con google en el primer cliente	3 days	Fri 09/06/23	Tue 13/06/23	20
Añadir tests de integración en el repositorio para funcionamiento con la suite	2 days	Wed 14/06/23	Thu 15/06/23	21
Documentación GTFS RealTime	1 day	Fri 16/06/23	Fri 16/06/23	19

Tabla 4.2: Planificación general del proyecto

Con esta información se procede a dividir las tareas por sprints acorde con la metodología Scrum. En este caso, al ser sprints de una duración de 2 semanas, tendremos un total de 4 sprints. Aunque el último sprint, si todo se desarrolla según lo estimado, no se completará con todos los puntos de historia posibles.

A continuación, se presenta una tabla con las fechas de los sprints.

Sprint	Fecha de inicio	Fecha de fin
1	1/05/2023	15/05/2023
2	15/05/2023	29/05/2023
3	29/05/2023	12/06/2023
4	12/06/2023	26/06/2023

Tabla 4.3: Planificación de los sprints

Para finalizar la planificación elegiremos las tareas que se realizarán en cada sprint teniendo en cuenta los puntos de historia máximos de cada sprint.

Sprint 1

Task Name	Story Points
Leer documentación GTFS	2
Determinar requisitos del microsevicio	1
Diseñar estructura del microservicio	1
Crear microservicio GTFS RT	2
Eliminar base de datos del microservicio	1
Crear endpoints	2
Configurar contenedor Docker	1

Tabla 4.4: Planificación del sprint 1

Sprint 2

Task Name	Story Points
Configurar Kubernetes	1
Configurar Jenkins	1
Crear interfaz	0,5
Añadir y configurar clientes	1
Crear cachés	2
Hacer Publish Subscribe a Topology	2
Hacer VehiclePosition	2

Tabla 4.5: Planificación del sprint 2

Sprint 3

Task Name	Story Points
Hacer TripUpdate	3
Pruebas de carga, rendimiento y robustez	3
Pruebas de integración con google en el primer cliente	3

Tabla 4.6: Planificación del sprint 3

Sprint 4

Task Name	Story Points
Añadir tests de integración en el repositorio para funcionamiento con la suite	2
Documentación GTFS RealTime	1

Tabla 4.7: Planificación del sprint 4

4.4. Presupuesto

4.4.1. Material

El trabajo se realizará con un ordenador portátil Lenovo Thinkpad p14s acompañado por una tarjeta SIM con acceso a internet para poder trabajar desde cualquier lugar. A mayores se tendrá en cuenta el precio de las licencias y software utilizado durante el desarrollo del proyecto.

A continuación, se muestra una lista con todo el material y el coste de cada elemento.

Concepto	**Coste (€)**
Ordenador portátil	1.499,99
Tarjeta SIM con internet	25,95
Licencia Confluence	9,00
Licencia Bitbucket	18,00
Licencia Visual Studio 2022	100,00
Licencia Office 365	14,00
Licencia Jenkins	0,00
Licencia SonarQube	25,00
EasyRetro	0,00
ScrumPoker	0,00
Postman	0,00
Coste Total	**1.691,94**

Tabla 4.8: Desglose del coste de materiales

4.4.2. Mano de obra

Como se ha descrito en la parte de planificación, en este desarrollo trabajará un ingeniero de software a tiempo completo en una jornada de cuarenta horas semanales de lunes a viernes en un horario estándar sin ninguna plusvalía.

Para el cálculo de costes se ha utilizado la página web Glassdoor en la cual se puede buscar el salario medio de un puesto de trabajo con la posibilidad de filtrar por zona geográfica. En este caso el salario bruto medio anual de un ingeniero de software es de 28.809 €. Esto se traduce en un salario bruto de 16,60 €/hora aplicando el convenio colectivo de 2022 que establece 1736 horas/año. Dato que se utilizará para realizar la estimación de costes humanos teniendo en cuenta las horas de trabajo necesarias para completar el proyecto.[29]

La fórmula para calcular el coste es:

$$\text{Coste total} = \text{Horas necesarias} \times \text{Sueldo por hora}$$

Sustituyendo los valores:

$$\text{Coste} = 252\,\text{h} \times 16{,}60\,€ = 4183{,}20\,€$$

Por lo tanto el coste humano global de este proyecto será de 4.183,20 €, suponiendo una estimación correcta. En caso de un aumento o reducción de las horas de trabajo totales, el coste aumentará o se reducirá respectivamente.

Concepto	Coste (€)
Horas necesarias	252
Sueldo por hora	16,60
Coste Total	**4.183,20**

Tabla 4.9: Desglose del coste humano del proyecto

4.4.3. Resumen global

En este apartado realizaremos un cálculo exhaustivo de todos los costes presentados en los apartados anteriores. Esto tiene como objeto proporcionar una visión general de los costes totales del proyecto, permitiendo evaluar de manera precisa y completa la viabilidad económica del mismo.

Los stakeholders tendrán gran interés en este apartado dado que supondrá en gran medida la inversión a realizar y deberá realizarse una valoración acerca del posible rendimiento del proyecto y si el valor de retorno compensa la inversión inicial.

Para el cálculo total sumaremos el coste humano y el coste material y obtendremos el coste total del proyecto.

Concepto	Coste (€)
Materiales	1.691,94
Mano de obra	4.183,20
Coste Total	**5.875,14**

Tabla 4.10: Desglose por secciones del coste total del proyecto

Finalmente obtenemos un coste total de 5.875,14 €. También podemos observar donde se encuentra el mayor peso del coste del proyecto. En este caso la mano de obra es considerablemente mucho más costosa que los materiales. En el desarrollo de este tipo de proyectos suele ser normal que esto sea así.

A pesar de ello, en este caso, al realizarse en un entorno empresarial, la diferencia es considerablemente mayor. Una empresa dispone de mayores medios materiales y a un precio menor comparado con un particular, debido a su tamaño. Esto permite a una empresa realizar una compra de productos y servicios al por mayor y por lo tanto obtener precios por unidad menores.

Capítulo 5

Diseño

5.1. Estructura de paquetes

En este capítulo, se abordará el diseño detallado de la solución propuesta para la creación de un microservicio que provee feeds GTFS (General Transit Feed Specification) en tiempo real a Google. El objetivo principal de este capítulo es presentar las decisiones de diseño clave que se han tomado para construir un sistema eficiente, escalable y confiable.

Para lograr este objetivo, se realizará un análisis exhaustivo de los requisitos funcionales y no funcionales identificados previamente en el capítulo de especificación de requisitos. A partir de esta información, se definirán las diferentes capas y componentes del sistema, así como las interacciones entre ellos.

Además, se considerarán aspectos importantes como la arquitectura del sistema, los protocolos de comunicación y las estrategias de despliegue y escalabilidad. Se presentarán diagramas, esquemas y descripciones detalladas que permitirán comprender el diseño del microservicio en su totalidad.

El diseño propuesto se basa en las mejores prácticas y estándares de la industria, con el objetivo de cumplir con los requisitos establecidos y proporcionar un servicio confiable y eficiente para la generación y entrega de feeds GTFS en tiempo real a Google.

5.1.1. General

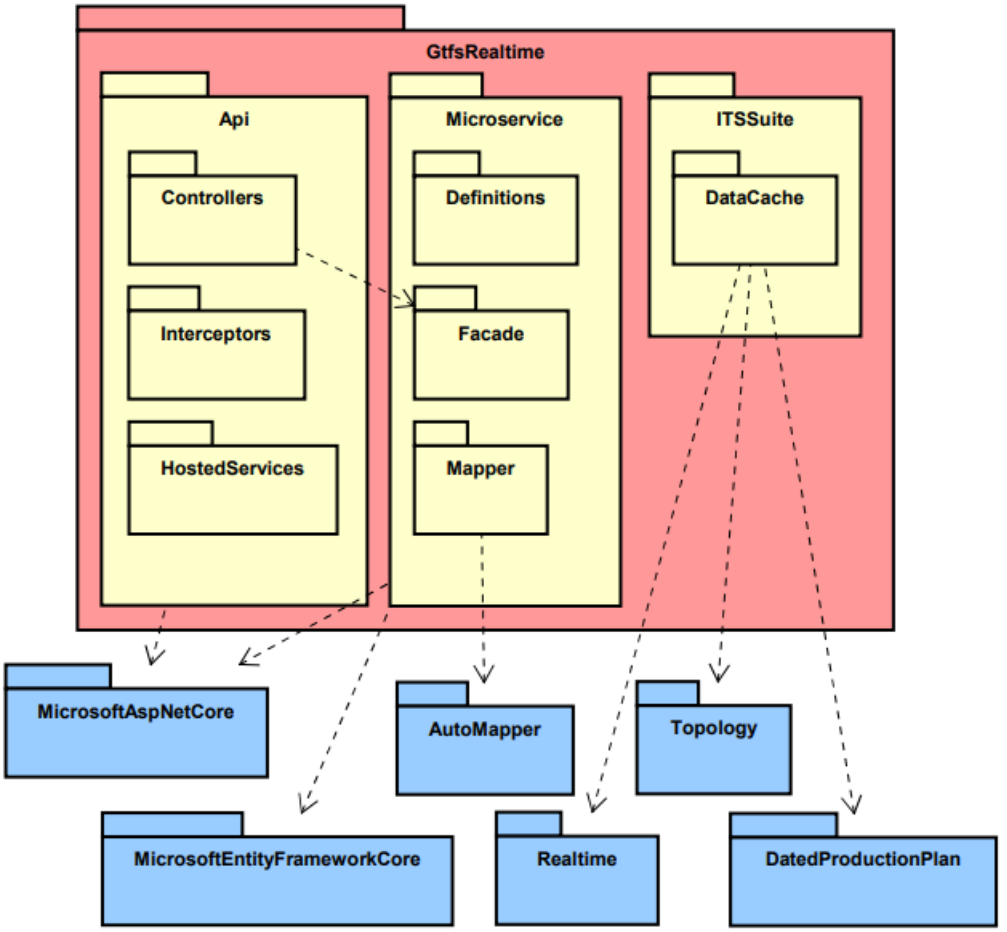

Figura 5.1: Estructura de paquetes general

La estructura general para este microservicio se divide en cuatro partes. Una de estas partes son los test, los cuales no se incluirán en esta parte dada su irrelevancia en este capítulo. Por lo que se han representados los tres paquetes restantes que son: API, Microservice, ITSSuite.

Se pueden observar las dependencias de la parte de caché y manejo de datos con los microservicios de los que se obtiene la información. En este caso la comunicación y obtención de datos se realiza a través de clientes para evitar el retrabajo volviendo a programar las solicitudes.

A nivel general se puede observar la utilización del patrón fachada. De esta forma se obtiene un sistema de fácil utilización a través de llamadas sencillas a las interfaces proporcionadas

por los servicios objetivo.

5.1.2. API

Debido a que esta aplicación no cuenta con interfaz de usuario, el paquete API principalmente se encarga del arranque de la aplicación. Esto se puede observar en el paquete HostedServices del cual depende el arranque público.

Por otro lado, también se encuentran los controladores encargados de la gestión de la comunicación con otros microservicios. En este caso se puede observar que no tienen dependencias internas.

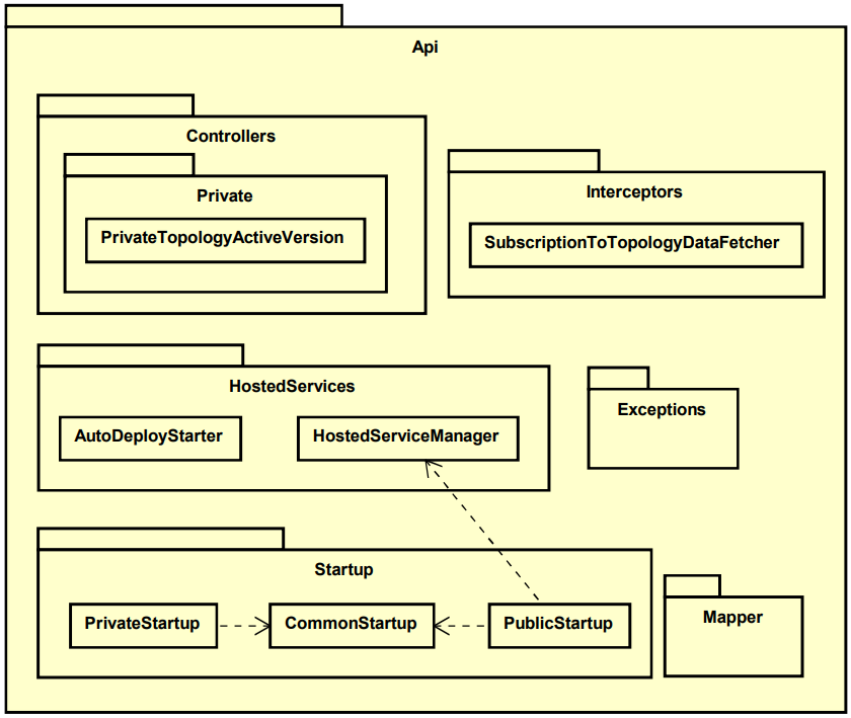

Figura 5.2: Estructura de paquetes de API

5.1.3. Microservice

Este paquete constituye el núcleo de la aplicación. En él se encuentra el núcleo y, por lo tanto, el funcionamiento base de la aplicación. En el paquete de definiciones se establecen los valores de configuración de los que va a constar la aplicación durante su funcionamiento. Gracias a esto se puede adaptar la aplicación según las necesidades simplemente modificando los valores de configuración.

Además, el microservicio también incluye un paquete de fachada que ofrece una capa de exposición para las interfaces necesarias. Este paquete fachada actúa como un punto de entrada único, simplificando la interacción con las diversas funcionalidades y componentes internos. Por otro lado, se consigue un mayor control sobre los puntos de acceso al unificarse de esta forma.

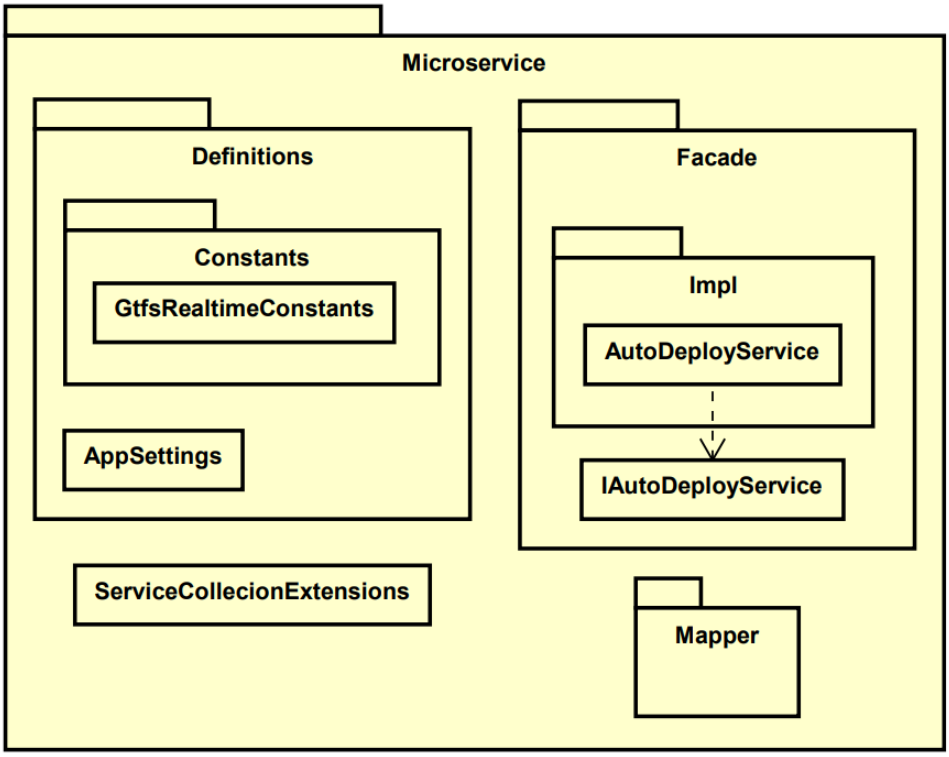

Figura 5.3: Estructura de paquetes de Microservice

5.1.4. ITSSuite

Este paquete es el encargado de la implementación de la funcionalidad de la interfaz encargada de la creación de los feeds. Por lo tanto, en este paquete se alojará todo lo necesario para el funcionamiento de esta implementación.

En este caso, existe un paquete llamado DataCache para implementar toda la funcionalidad y manejo de las memorias caché. Esto permite una sencilla utilización por parte de la implementación de IGtfsRealtime a través de las interfaces creadas para ello.

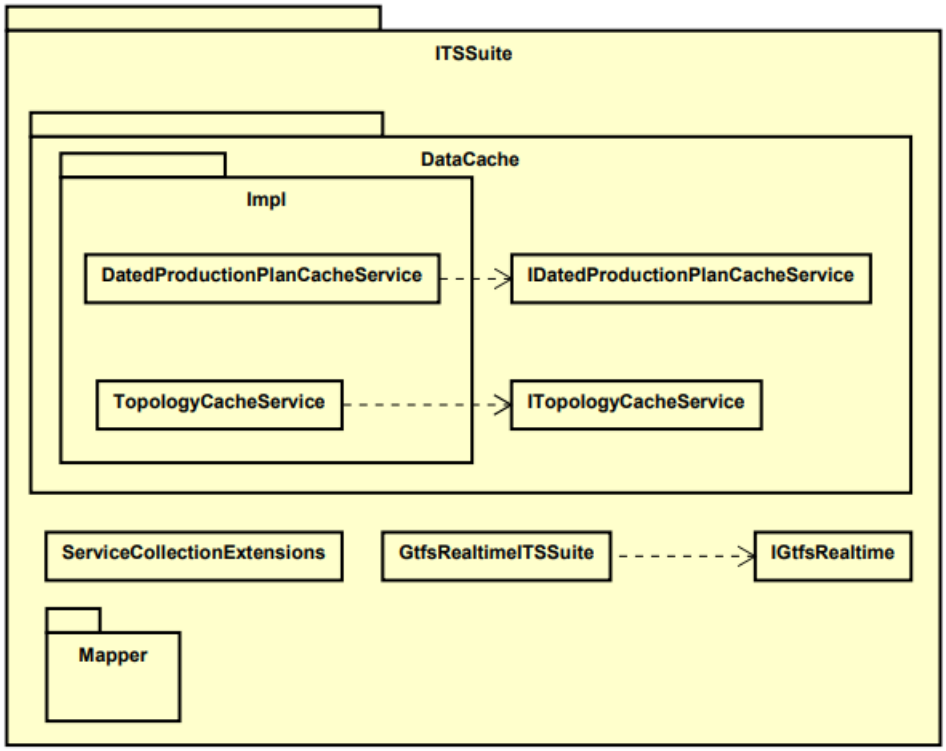

Figura 5.4: Estructura de paquetes ITSSuite

5.2. Diagrama de despliegue

Para el despliegue de esta aplicación se utilizará una única máquina. Esta máquina local representada en el diagrama mostrado a continuación, será un servidor en el que se ejecute un Pod de Kubernetes y se alojen dentro todos los microservicios contenerizados de manera individual.

El microservicio depende de Topology, DatedProductionPlan y Realtime debido a que de estos microservicios es de donde obtiene toda la información necesaria para la posterior conversión de la misma a feeds GTFS y publicarlos en Google.

Para ello se publica la información en un espacio destinado por Google cuya estructura y construcción no está disponible para sus usuarios. Por ello solo se muestra la entidad en sí como punto de entrada de datos.

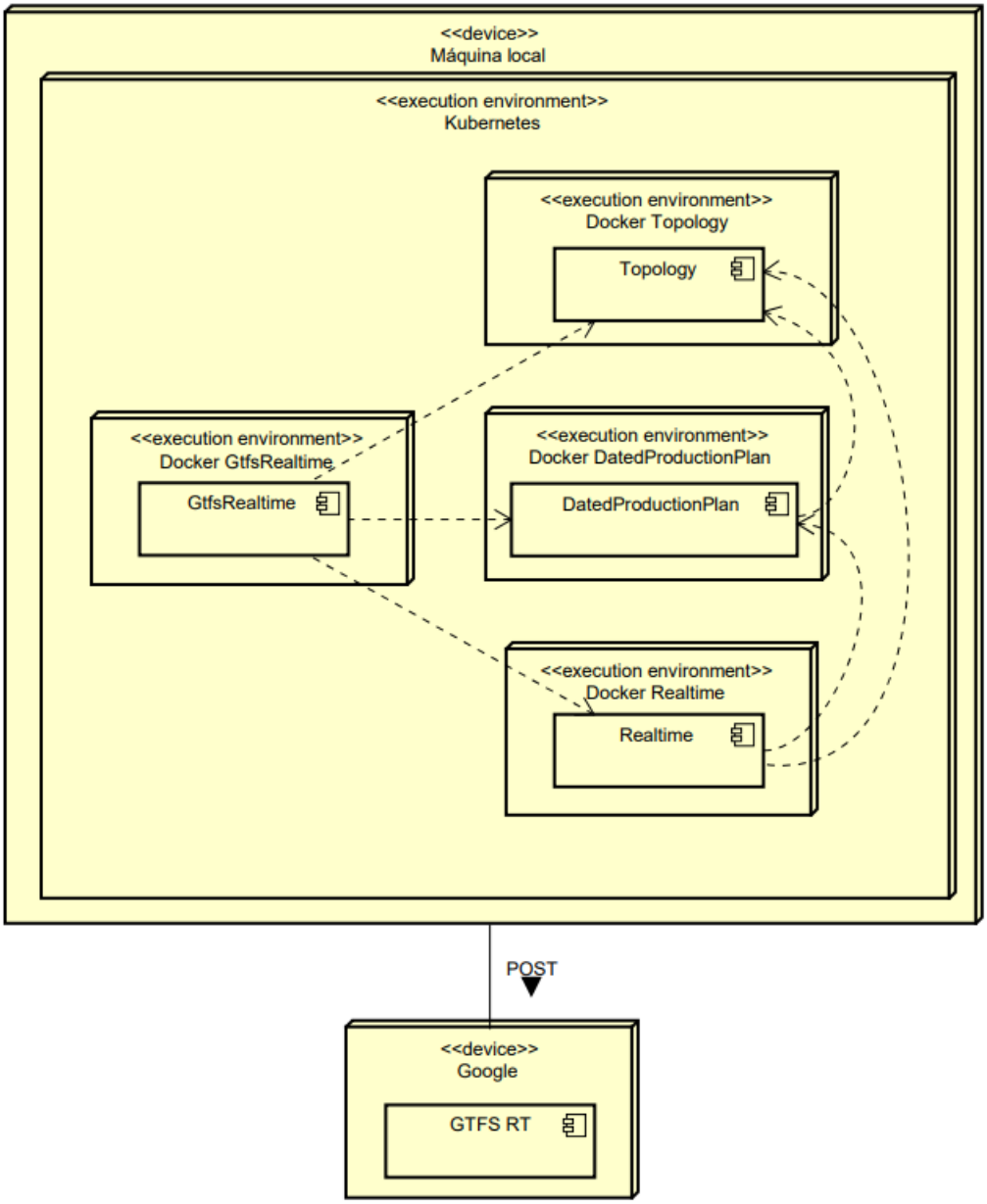

Figura 5.5: Diagrama de despliegue

Capítulo 6

Implementación

En este capítulo, se abordarán las partes específicas del diseño de la aplicación que presentan ciertas particularidades que requieren una explicación detallada de su implementación. Estos detalles son esenciales para comprender completamente la aplicación en su totalidad y cómo todas las piezas se unen en un sistema coherente y funcional.

El objetivo principal de este capítulo es proporcionar una visión completa del proceso de implementación de la aplicación, destacando los aspectos clave que influyen en su funcionamiento y rendimiento. Principalmente, aquellos aspectos que contienen ciertas particularidades a tener en cuenta en la construcción de la aplicación.

6.1. Caché

Una de las partes fundamentales para la velocidad y la eficiencia del microservicio es la utilización de la memoria caché. La memoria caché permite almacenar información para su posterior utilización sin necesidad de realizar una nueva solicitud al servicio de donde se obtuvo.

Para nuestro caso particular, tenemos un microservicio que necesita obtener información de varios microservicios para construir los feeds de GTFS. Esta información se obtiene mediante solicitudes HTTP a través de la utilización de los clientes necesarios. Se profundizará en el funcionamiento de los clientes en el apartado 6.4.

En este caso, se dispone de información en tiempo real, la cual va vinculada a información estática. Por lo tanto, diferenciaremos entre estos dos tipos de información: información estática e información en tiempo real. El uso de memoria caché se aplicará a la información estática, ya que esta variará con poca frecuencia y no será necesario solicitarla cada vez que se desee generar un feed.

Esta información permanecerá en la memoria del microservicio y esto evitará solicitudes

innecesarias al resto de microservicios y esto también aumentará la velocidad de nuestro servicio. El rendimiento y eficiencia es lo más importante en una aplicación de información tiempo real y por lo tanto estos dos aspectos siempre se encontrarán dentro de las prioridades.

Además del almacenamiento de la información en memoria caché, se ha implementado un sistema de limpieza de la misma para ahorrar espacio en memoria y un uso excesivo de recursos. Esta política de limpieza automática consiste en la eliminación periódica de aquellos elementos vinculados a momentos temporales pasados. Por ejemplo, no es necesario almacenar la información de una expedición pasada dado que ya no vamos a enviar ninguna información en tiempo real al respecto. A pesar de ello, se dejará un margen temporal desde su finalización por si hubiera algún contratiempo en la planificación y por lo tanto el elemento tuviera actualizaciones en tiempo real fuera del tiempo planificado.

Gracias a esto se evita la acumulación de basura en la memoria caché, evitando así, una saturación de memoria que provocaría una interrupción inintencionada del microservicio.

6.2. Publish Subscribe

Una vez creada la caché, se debe decidir cuando esta debe ser actualizada. Hay que recordar que todo esto se ha realizado para evitar realizar todas las solicitudes al resto de microservicios cada vez que se vaya a generar un nuevo feed. Principalmente porque no es una solución sostenible y es imposible que funcione sin una cantidad inmensa de recursos asignada a cada microservicio.

Esto nos lleva a este punto. Los microservicios de los que obtenemos la información estática cuentan con puntos de suscripción que puede ser usados a través de interceptores. De esta forma solo se actualiza nuestra memoria caché cuando haya habido alguna modificación del grupo de datos en cuestión.

Uniendo la suscripción a los microservicios con el uso de memorias caché se obtiene un manejo de memoria correcto y eficiente. Para este microservicio se ha creado una memoria caché por microservicio asociado. En cada clase encargada del manejo de la caché, se maneja la política de borrado y los interceptores utilizan estas clases a través de las interfaces proporcionadas para refrescar la caché cuando se reciba una notificación de un cambio en la información.

6.3. Interfaz GTFS Realtime

Para realizar la labor de creación de los feeds se ha unificado todo el proceso en una interfaz en la cual se define un método por cada tipo de feed. Cada uno de estos métodos será el encargado de obtener la información necesaria y encapsularla para la generación del feed correspondiente. Los tipos de feed de tiempo real son los siguientes: actualizaciones de viaje, alertas de servicio y posiciones de los vehículos.

A continuación, cuando se hable de generación de objetos, se hará por simplicidad en la explicación del funcionamiento. Hay que tener en cuenta que muchas entidades se actualizan al mismo tiempo y estos métodos manejan listas que contienen varias entidades y por lo tanto solo se realiza una única llamada a estos métodos con todos los datos de un instante de tiempo.

6.3.1. Actualizaciones de viaje

Recopila toda la información acerca de actualizaciones en los horarios en tiempo real. Para ello se obtiene información del microservicio de horarios y se complementa dicha información con la proporcionada por el microservicio que gestiona los cambios en tiempo real.

Finalmente se construye un objeto Viaje específico para esta causa. Posteriormente se transforma toda esa información en un feed de actualizaciones de viaje. En GTFS, un viaje es una secuencia de dos o más paradas que tienen lugar a una hora específica.

6.3.2. Alertas de servicio

En caso de una interrupción en la red, se recibirá toda la información acerca de la cancelación y este método generará un objeto Alerta que contendrá toda esta información. Este objeto será el utilizado para la posterior generación del feed de alertas de servicio.

Las demoras y cancelaciones generalmente no se deben realizar a través de este método. Para ello existen los métodos de actualizaciones de viaje y de posiciones de los vehículos.

6.3.3. Posiciones de los vehículos

Este método utiliza principalmente la información en tiempo real proporcionada por el microservicio de tiempo real para obtener toda la información relativa al posicionamiento del vehículo. Esta información se asocia al identificador del vehículo, cuya información es estática dado que este identificador no cambia con el tiempo.

Una vez realizada la recopilación de datos, se genera un objeto Vehículo que contiene toda esta información y, al igual que en los otros métodos, se utiliza después para la generación de su respectivo feed, en este caso el feed de posiciones de los vehículos.

6.4. Clientes

La comunicación entre microservicios es esencial para el funcionamiento eficiente de un sistema distribuido. El uso de clientes particulares, que sirven como interfaces para interactuar entre varios microservicios, es un método típico para establecer dicha comunicación.

Es una forma muy conveniente para acceder a los datos y funcionalidades ofrecidos por un microservicio sin la necesidad de programar llamadas HTTP manualmente.

Los clientes proporcionan una capa de abstracción que simplifica la interacción entre microservicios. Son creados específicamente para cada microservicio y brindan técnicas y características que permiten realizar operaciones relacionadas con ese servicio específico como, en nuestro caso particular, solicitar datos. No es necesario programar ninguna llamada HTTP. Simplemente, hay que inyectar los clientes como dependencias para comenzar a utilizarlos. De esta forma se obtiene un desarrollo más eficaz y flexible en diferentes entornos.

La principal ventaja de usar clientes es que se abstrae y simplifica la complejidad interna que conlleva la comunicación con estos microservicios. Con los clientes no es necesario escribir código para establecer conexiones HTTP, gestionar peticiones, etc. De esta forma se encapsulan todas estas operaciones en métodos más simples y orientados a la lógica de negocio.

Respecto a la inyección de dependencias, es necesario configurar las dependencias en el entorno de desarrollo e incluir las bibliotecas relevantes. Después de completar esto, se puede acceder de manera más fácil y legible a los métodos proporcionados por el cliente y realizar llamadas a las funcionalidades del microservicio de forma rápida y simple.

Gracias a esto se consigue aumentar la seguridad y la tolerancia a fallos en la comunicación entre microservicios. Además, los clientes también pueden proporcionar mecanismos para el manejo y la autenticación de errores, otorgando un mayor control sobre posibles excepciones durante la ejecución.

Capítulo 7

Pruebas

En el presente capítulo de pruebas, se tratará la corrección, robustez y validez del código. Para ello se detallarán los diferentes tipos de pruebas realizadas para probar el funcionamiento del código en diferentes ámbitos.

Por un lado, se expondrán las pruebas realizadas sobre el propio código las cuales verifican la robustez, calidad y funcionamiento sin errores del código. Se cuentan como errores aquellos fallos los cuales no son manejados por el código y por lo tanto su resultado es inesperado pudiendo dar lugar a errores o fallos de seguridad.

Por otro lado, se estudiarán los validadores de datos disponibles en internet para realizar la elección de uno de ellos para llevar a cabo la validación de formato de los ficheros GTFS generados por la aplicación. Una vez elegido simplemente se llevarán a cabo las pruebas con la herramienta elegida.

7.1. Test unitarios

Los test unitarios son pruebas destinadas a probar de forma individual los elementos más pequeños y aislados del código, conocidos como unidades funcionales. Estas unidades pueden ser métodos, funciones, clases o incluso componentes más pequeños.

El propósito de los test unitarios es validar que cada unidad funciona correctamente de forma aislada sin depender del funcionamiento de otras unidades del sistema. Esto pretende identificar y localizar de manera precisa cualquier posible error de código, lo que ayuda a corregir de forma rápida y temprana estos errores evitando mayores problemas futuros.

Para esta aplicación se ha utilizado xUnit para llevar a cabo los test unitarios. Todos los marcos de pruebas unitarias xUnit comparten la misma arquitectura básica de componentes, con algunos detalles de implementación variados:

- **Test runner:** Es un programa ejecutable que ejecuta pruebas implementadas utilizando un marco de pruebas xUnit y reporta los resultados de la prueba.
- **Test case:** Es la clase más elemental. Todas las pruebas unitarias se heredan de aquí.
- **Test fixtures:** Son clases que contienen un conjunto de pruebas unitarias que comparten el mismo contexto.

Debido a la integración con Jenkins, este realizará todos los test del proyecto en su propio entorno y en caso de no superar todos los test devolverá un error. Esto hará que no pueda fusionarse el código con la rama principal, evitando así, errores en el código que pertenecerá al producto final.

7.2. Validadores GTFS

Existen varios validadores GTFS disponibles para comprobar la correcta generación de los feeds. En este caso se ha realizado un análisis de varios validadores y después se ha realizado la elección de uno de ellos para realizar las pruebas. Por un lado, se seleccionará un validador para los datos estáticos para asegurar que sean los correctos dado que la parte de tiempo real depende de la parte estática. Después se utilizará el validador existente que ofrece Google para validar los datos de tiempo real.

7.2.1. Validadores de información estática

Gtfsvtor

Es un validador de código abierto que se encuentra en GitHub. Es un proyecto disponible bajo la licencia GPLv3 y tiene como objetivo proporcionar una validación rápida y exhaustiva de los archivos GTFS.

Sus principales objetivos y ventajas son:

- **Velocidad:** Se enfoca en ser rápido y eficiente en el procesamiento de archivos GTFS, incluso para conjuntos de datos muy grandes.
- **Eficiencia en la utilización de recursos de memoria:** Está diseñado para utilizar la memoria de manera eficiente, lo que permite procesar grandes conjuntos de datos GTFS sin agotar los recursos del sistema.
- **Extensible y fácil de mantener:** El código de GTFSVTOR está diseñado para ser extensible y fácil de leer y mantener. Esto facilita la adición de nuevas reglas de validación y la mejora continua del proyecto.

- **Cobertura extensa de validación:** Proporciona una cobertura de validación exhaustiva, asegurando que se verifiquen múltiples aspectos de los archivos GTFS, incluyendo rutas, paradas, viajes, horarios y puntos de forma (shape points). Aun así, no es la mejor cobertura de los validadores que veremos.
- **Retrocompatibilidad:** Diseñado para ser compatible con el validador anterior de Python. Esto significa que muchas de las reglas de validación implementadas en el validador anterior también están presentes en esta nueva versión del validador.
- **Múltiples formatos de salida:** Ofrece la posibilidad de generar informes de validación en diferentes formatos, como JSON y HTML. Estos informes proporcionan detalles sobre los problemas encontrados en los archivos GTFS durante el proceso de validación.
- **Facilidad de uso:** Es una herramienta muy fácil de utilizar pudiendo entender su funcionamiento y empezar a validar de forma correcta de forma rápida
- **Posibilidad de inclusión en Docker:** Ofrece la posibilidad de utilización con la herramienta Docker.

Antes de ver los pasos de instalación exploraremos las dependencias necesarias para el funcionamiento de esta aplicación. En este caso los requisitos necesarios son disponer de Java 8 o superior, al ser código abierto desarrollado por la comunidad no se recomienda utilizar versiones muy superiores dado que puede dar lugar a posibles errores. Durante el desarrollo de este proyecto se han experimentado errores con versiones modernas de Java. También es necesario tener instalado Gradle y finalmente Git para poder clonar el repositorio.

Para utilizar GTFSVTOR, se pueden seguir los siguientes pasos:

1. Descargar la última versión disponible desde el repositorio de GitHub del proyecto.
2. Descomprimir el archivo en una ubicación deseada en tu sistema.
3. Ejecutar GTFSVTOR a través de la línea de comandos, utilizando el archivo binario correspondiente a tu sistema operativo. Por ejemplo:
   - En sistemas Unix/Linux: './gtfsvtor/bin/gtfsvtor [opciones] <archivo GTFS>'
   - En Windows: Utilizar el archivo 'gtfsvtor.bat' proporcionado.
4. Es importante tener en cuenta que se requiere tener instalado Java JRE en el sistema para ejecutar GTFSVTOR.
5. También se puede configurar la validación ajustando el archivo 'config.properties' que se encuentra en la raíz del proyecto.
6. Para conjuntos de datos GTFS grandes, se puede aumentar el tamaño predeterminado del montón (heap) JVM ajustando las opciones adecuadas en la variable 'GTFSVTOR_OPTS'.
7. Adicionalmente, descomprimir los datos en el disco puede reducir el uso de memoria para conjuntos de datos GTFS grandes.

Por último, cabe destacar que este validador tiene algunas desventajas. No valida ciertas cosas como por ejemplo el que no se repita la misma clave de shape con distintas secuencias. También tiene una escasa documentación por lo que no se dispone de tanta ayuda complementaria ante posibles problemas aparte de la ayuda que pueda ofrecer la comunidad.[30]

MobilityData

Este validador también es de código abierto y se encuentra subido a GitHub. Ofrece diferentes opciones de uso entre las que se encuentran la opción de uso web, programa de escritorio y línea de comandos.

A la versión web puede accederse desde `https://gtfs-validator.mobilitydata.org/`. Permite cargar conjuntos de datos GTFS en formato zip desde un fichero local o desde una URL. Los informes de validación generados tienen una URL única que puede compartirse y están disponibles durante 30 días.

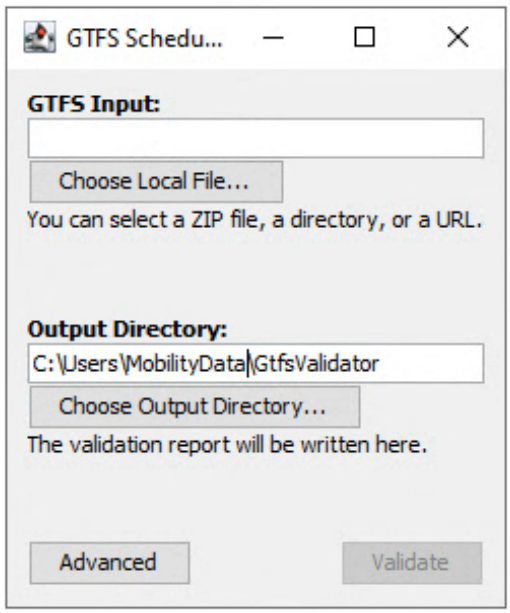

Figura 7.1: Interfaz de usuario de MobilityData en su versión de escritorio

Para utilizar la aplicación de escritorio, se debe descargar el instalador más reciente del validador GTFS desde la página de lanzamientos (Releases) en GitHub, según el sistema operativo. Después simplemente se debe ejecutar el instalador y este realizará todo el proceso de instalación. Luego, al ejecutar la aplicación, se presentarán dos opciones principales: “GTFS Input” y “Output Directory”. En “GTFS Input”, se debe especificar la ubicación del feed GTFS a validar, a través de URL, un archivo ZIP o un directorio con archivos .txt individuales. En ”Output Director”, se debe seleccionar el directorio donde se escribirán los informes

de validación. Después de configurar estas opciones simplemente se presiona el botón "Validate" para iniciar la validación. Una vez completada, la aplicación abrirá automáticamente el informe de validación en formato HTML en el navegador local.

En resumen, es una herramienta muy sencilla de utilizar y con varios medios de uso posibles. Además, cuenta con una abundante documentación, lo que facilita, aún más, el aprendizaje y uso de la herramienta.[31]

Para este desarrollo esta ha sido la herramienta seleccionada para la validación de datos estáticos.

FeedValidator

Es un validador de código abierto también repositado en GitHub. Su función es validar los datos estáticos de GFTS.

Para su utilización, se debe cargar un archivo ZIP desde una URL o subiendo un archivo local. En primer lugar, realiza una verificación de la integridad del archivo, el análisis de tipo numérico y los rangos. Una vez realizada esta verificación, el validador comienza con la validación completa respecto a la especificación del estándar GTFS.

Este validador es mantenido por MobilityData, pero actualmente no dispone de actualizaciones por lo que no es recomendable su utilización. No tiene opción de uso online y funciona a través de Python (Python 2.7). No permite la posibilidad de inclusión de Docker lo que supone otro aspecto negativo a tener en cuenta. Respecto a la documentación es bastante escasa por lo que no hay mucho soporte para el desarrollador.

En conclusión, no se recomienda la utilización de este validador tomando como principal razón de peso la rescisión del mantenimiento de la herramienta y, por tanto, perdiendo cualquier tipo de soporte en versiones modernas del estándar. Existen otras razones influyentes en la decisión de la no utilización de esta herramienta, comentadas anteriormente, pero ni se plantean debido al riesgo de mal funcionamiento que presenta la herramienta en sí.[32]

7.2.2. Validadores de información en tiempo real

En esta sección solo trataremos un validador dado que el desarrollo de los validadores de información en tiempo real y, por consecuente, los validadores de información de tiempo estático, está evolucionando en base a una herramienta principal, aquella desarrollada por MobilityData. En la documentación de Google podemos encontrar esta herramienta como la sugerida para realizar validaciones.

MobilityData

El validador de datos en tiempo real desarrollado por MobilityData es una herramienta de código abierto basada en Java publicada en GitHub. Su creación original y temprano

desarrollo fue llevada a cabo por CUTR en USF (Center for Urban Transportation Research at the University of South Florida) y en abril de 2022 se unió a MobilityData para mejorar y continuar su desarrollo como un recurso comunitario.[33]

MobilityData también mantiene un validador de GTFS para archivos GTFS que contienen información estática. Pero en esta sección no profundizaremos en ella dado que se explica con más detalle en la sección 7.2.1.

En primer lugar, para ejecutar y compilar la aplicación es necesario disponer de Java JDK 11 o superior y de Apache Maven. Después es necesario descargar los dos componentes del proyecto alojado en GitHub:

- **gtfs-realtime-validator-lib:** Es la biblioteca central que implementa la validación GTFS Realtime, así como el sistema de procesamiento por lotes.
- **gtfs-realtime-validator-webapp:** Un servidor y una página web desde la que poder validar los ficheros GTFS de forma rápida y sencilla.

Una vez descargados e instalados todos los elementos necesarios, hay que seguir los siguientes pasos para construir, ejecutar y acceder al validador:

1. **Construir el proyecto:** Desde la línea de comandos se debe ejecutar el comando "mvn package". Esto generará un ejecutable en el directorio "gtfs-realtime-validator-webapp/target/" con todo lo necesario para ejecutar la aplicación.
2. **Ejecutar la aplicación:** Para ejecutar el servidor simplemente hay que ejecutar en la línea de comandos el comando "java -jar Nombre del fichero JAR" donde el nombre del fichero debe ser la ruta al fichero jar de la webapp.
3. **Ver la aplicación:** Para acceder a la aplicación simplemente hay que acceder a la URL "`http://localhost:8080`" en cualquier navegador.

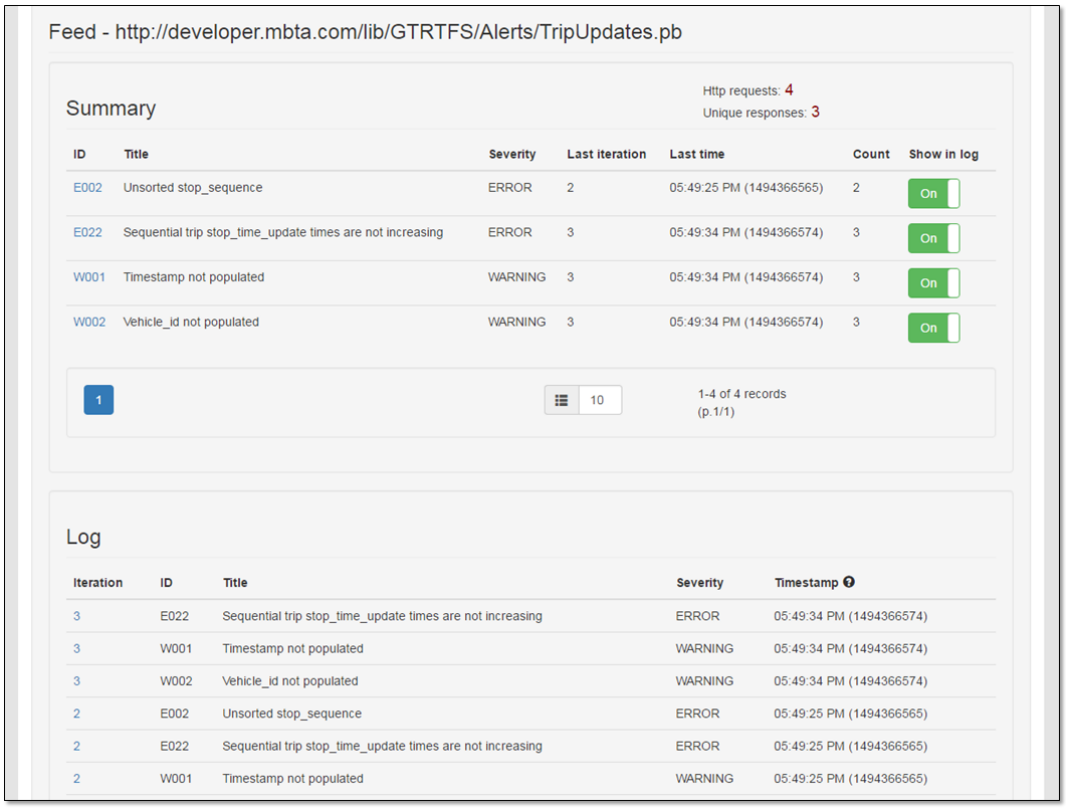

Figura 7.2: Interfaz de uso de la web del validador de datos GTFS Realtime de MobilityData

Después de introducir todos los datos y realizar la validación se puede observar el registro del resultado de la validación. Como se puede ver en la imagen 7.3, se muestran los errores del fichero si hubiera algún error en el mismo.

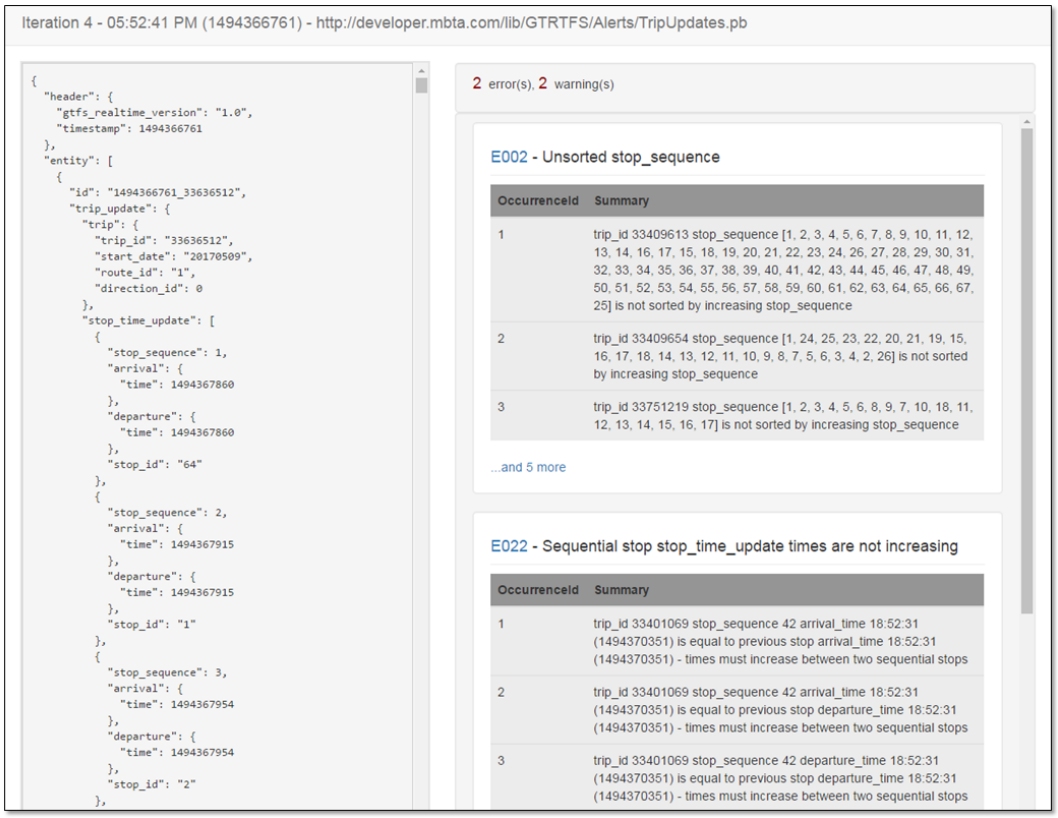

Figura 7.3: Interfaz de resultados de la web del validador de datos GTFS Realtime de MobilityData

Otras ventajas que ofrece es la posibilidad de inclusión en Docker para poder utilizar en contenedores. También cuenta con una buena documentación y cuenta con un mantenimiento actualizado y es el validador recomendado por la propia página de Google dedicada a la especificación de este estándar.[7]

Dada su oferta de servicio de validación de información estática y en tiempo real y que es la mejor y única opción actualmente para la validación de la información en tiempo real, se ha optado por este validador y su análogo de validación de información estática. De esta forma tendremos escasas diferencias de funcionamiento entra uno y otro y la plataforma de desarrollo y dependencias son mayoritariamente comunes entre las dos herramientas. Además, obviamente, de su correcto funcionamiento y cobertura frente a posibles casos especiales que otros validadores dan por válido cuando se trata de un error.[34]

Capítulo 8

Seguimiento del proyecto

En este capítulo, se tratará el seguimiento del proyecto, una fase fundamental para tener un control sobre el desarrollo de la planificación del proyecto y la adaptación ante posibles contratiempos. Para ello en esta ocasión se ha seguido la metodología ágil SCRUM, la cual permite una entrega y desarrollo eficientes del proyecto.

La estructura elegida a seguir han sido sprints, debido a la metodología utilizada. Estos sprints tienen una duración de dos semanas y por lo tanto eso conlleva una asignación de aproximadamente 80 "Story Points" por miembro del equipo. Cada "Story Point" equivale a una hora de trabajo y se utiliza para medir la posible duración de cada tarea. Las tareas son asignadas por los jefes de proyecto y estimadas por todos los miembros del equipo. Para realizar las estimaciones se pueden utilizar diversas herramientas de votación, pero en este caso se ha utilizado Scrum Poker. Los números a elegir como puntos de historia sobre una tarea son los siguientes: 0.5, 1, 2, 3, 5. Cualquier puntuación superior indicaría que se trata de una tarea épica y por lo tanto según la metodología SCRUM esta tarea debería ser dividida en subtareas.

La estimación de tareas, como su propio nombre implica, son una estimación, por lo que la duración estimada no tiene por qué corresponderse con la duración real, aunque ese sea su propósito. Por eso al final de cada Sprint se evaluará el porcentaje de tareas completado para dicho sprint y se asignará un estado a cada una, (esto se tendrá en cuenta a la hora de asignar nuevas tareas en el siguiente sprint).

Finalmente, para completar el sprint, se realizará una retrospectiva en la cual se valorará y analizará el transcurso del sprint, los avances y los resultados obtenidos.

8.1. Sprint 1

El objetivo de este sprint es crear la base del microservicio para después implementar toda la lógica de negocio. Se espera tener un microservicio capaz de recibir y realizar solicitudes

y de funcionar de forma automática. En primer lugar, se obtendrá la mayor información posible sobre GTFS y después será utilizada para diseñar el modelo más óptimo para este microservicio.

Después, para la creación del mismo, se partirá de una plantilla base de un microservicio y a partir de ella se irá personalizando según las necesidades. Por último, se revisará todo el microservicio para eliminar aquellos elementos innecesarios para obtener un código limpio y estructurado.

8.1.1. Retrospectiva de sprint 1

Se ha dedicado más tiempo del estimado a la lectura de la documentación y al modelado y especificación del microservicio. Esto ha llevado a no completar todas las tareas del sprint y por tanto el desarrollo sufre actualmente de un ligero retraso.

A pesar de ello, se ha realizado buena parte de la construcción de la estructura base del microservicio y el diseño elegido para el microservicio está siendo el correcto. Por tanto, se considera aceptable el tiempo de retraso dado que ha sido por asegurar las bases, reducir las posibilidades de grandes retrasos en el desarrollo y evadir el retrabajo lo máximo posible.

Sprint 1

Task Name	Story Points	Status
Leer documentación GTFS	2	Closed
Determinar requisitos del microsevicio	1	Closed
Diseñar estructura del microservicio	1	Closed
Crear microservicio GTFS RT	2	Closed
Eliminar base de datos del microservicio	1	Closed
Crear endpoints	2	In progress
Configurar contenedor Docker	1	Not started

Tabla 8.1: Resultado final del sprint 1

8.2. Sprint 2

Para este sprint se plantea finalizar las tareas pendientes del sprint anterior y completar las tareas del sprint actual. Dado que esto no es muy realista, debido a los puntos de historia acumulados del sprint anterior, se ha decidido mover la última tarea del segundo sprint al inicio del sprint 3. De esta forma se espera cumplir con todas las tareas del sprint.

El objetivo de este sprint es finalizar la construcción del microservicio base. Una vez realizada esta labor, se realizará la configuración necesaria para la integración con el resto de microservicios que forman el entorno. Para ello se configurarán Docker, Kubernetes y Jenkins para que el proyecto pueda ser testeado de forma automática y pueda ser alojado y puesto en funcionamiento junto con el resto de microservicios.

8.2.1. Retrospectiva de sprint 2

La falta de experiencia con el lenguaje de programación ha supuesto un ligero retraso en el desarrollo y en parte es lo que no ha permitido que se completen con éxito todas las tareas del sprint. También ha producido cierto retraso el desconocimiento del funcionamiento de las suscripciones y los interceptores con otros microservicios del entorno.

Finalmente la tarea de creación de las cachés ha sido completada pero, al no haber sido aprobada la "Pull Pequest" por todos los miembros necesarios, no ha podido fusionarse el código con la rama principal y por lo tanto no se cuenta la tarea como completada.

Respecto a la tarea de hacer la suscripción al microservicio de topología, se calcula que queda un punto de historia restante. Esta tarea ha llevado más tiempo de esperado debido al desconocimiento previamente mencionado.

Sprint 2

Task Name	Story Points	Status
Crear endpoints	2	Closed
Configurar contenedor Docker	1	Closed
Configurar Kubernetes	1	Closed
Configurar Jenkins	1	Closed
Crear interfaz	0,5	Closed
Añadir y configurar clientes	1	Closed
Crear cachés	2	In progress
Hacer Publish Subscribe a Topology	2	In progress

Tabla 8.2: Resultado final del sprint 2

8.3. Sprint 3

En este sprint se ha decidido mantener las tareas planeadas para este sprint a pesar de tener aproximadamente un punto de historia del sprint anterior. La razón de esta decisión se debe a la tarea de pruebas de carga, rendimiento y robustez. Se considera que esta tarea,

además de su desarrollo, va a conllevar un tiempo durante el que se pueden realizar otras tareas dado que, por ejemplo, en el caso de las pruebas de carga, el microservicio debe estar funcionando de forma autónoma sin fallos durante un tiempo determinado.

Las expectativas para este sprint son, finalizar las suscripciones a otros microservicios para completar la parte de obtención de datos y desarrollar los elementos que intervienen en la conversión de dicha información a un fichero de formato GTFS. Por lo tanto, se espera realizar este desarrollo y, mientras las tareas se validan y aprueban, realizar las pruebas.

8.3.1. Retrospectiva de sprint 3

No ha habido mayores problemas en este sprint por lo que todo ha evolucionado como se había planeado. Las pruebas han llevado algo más de tiempo del esperado debido a una saturación de recursos del servidor.

El servidor tenía bastantes procesos abiertos y estaba realizando las pruebas de otro microservicio. Si se hubieran lanzado las pruebas existían altas probabilidades de un agotamiento de recursos y esto hubiera estropeado las pruebas en proceso y ralentizado otros desarrollos. Por ello se decidió posponer el lanzamiento para el día siguiente. En paralelo se avisó a los equipos implicados para que lo tuvieran en cuenta y no consumieran muchos recursos durante el período de tiempo que llevaran las pruebas.

Sprint 3

Task Name	Story Points	Status
Hacer Publish Subscribe a Topology	2	Closed
Hacer VehiclePosition	2	Closed
Hacer TripUpdate	3	Closed
Pruebas de carga, rendimiento y robustez	3	In progress

Tabla 8.3: Resultado final del sprint 3

8.4. Sprint 4

Este es el último sprint y, según la planificación no completa todos los puntos de historia disponibles para el sprint. La idea es finalizar las pruebas de rendimiento que se están ejecutando actualmente y finalizar las tareas restantes.

Hay algo de margen a nivel temporal por lo que se espera completar todas las tareas del sprint si no ocurre ningún contratiempo de fuerza mayor. Después el proyecto quedaría finalizado. Aunque, el microservicio queda abierto a posibles mejoras en un futuro como por ejemplo un incremento de funcionalidad.

8.4.1. Retrospectiva de sprint 4

Este sprint ha sido bastante tranquilo y se ha cumplido con la planificación. Todas las tareas se han completado y el microservicio funciona de manera correcta.

Por otro lado, revisando los requisitos extraídos al inicio del proyecto, se ha comprobado que la aplicación cumple con todos los requisitos establecidos satisfaciendo así, las expectativas de los stakeholders.

Sprint 4

Task Name	Story Points	Status
Pruebas de carga, rendimiento y robustez	3	In progress
Pruebas de integración con google en el primer cliente	3	Closed
Añadir tests de integración en el repositorio para funcionamiento con la suite	2	Closed
Documentación GTFS RealTime	1	Closed

Tabla 8.4: Resultado final del sprint 4

Capítulo 9

Conclusiones

En este capítulo se presentarán las conclusiones extraídas de la creación de la aplicación de aprovisionamiento de datos en tiempo real utilizando GTFS Realtime de Google. Se llevará a cabo un examen exhaustivo de todo el proceso de desarrollo e implementación de la aplicación, y los resultados se utilizarán para resaltar los logros, los obstáculos superados y las lecciones aprendidas.

9.1. Alcance

La investigación en el aprovisionamiento de datos en tiempo real en el transporte público aplicado a Google supone una gran mejoría en los servicios de transporte público. Muchas organizaciones de transporte público todavía no cuentan con la integración de esta funcionalidad en sus sistemas.

En este caso particular el alcance es máximo. Google es la plataforma más utilizada del mundo por parte de los usuarios y una implementación como la creada en este proyecto puede llegar a ser utilizada por una gran cantidad de usuarios.

9.2. Logros

Concluido el desarrollo de este proyecto se han alcanzado los siguientes logros:

- **Utilización de feeds y protobuf:** Se ha logrado entender y utilizar el formato de ficheros y de transferencia exigido por el estándar general y de Google para la utilización del servicio de GTFS de tiempo real.

- **Creación e implementación del microservicio:** Se ha logrado crear e implementar un microservicio de aprovisionamiento de datos en tiempo real a través de GTFS con una arquitectura consistente y robusta. También se ha conseguido integrar en una suite de microservicios dedicados al transporte público.
- **Integración con Google:** Se ha logrado establecer una comunicación efectiva con los servicios de Google para la transferencia de feeds de tiempo real.
- **Rendimiento y eficiencia:** Se ha logrado crear un microservicio con un gran rendimiento y un bajo tiempo de respuesta. Proporcionando así, datos de forma rápida y precisa. Además, se ha conseguido un microservicio con una utilización eficiente de recursos dadas sus funcionalidades.
- **Validación y calidad de la información:** Se han realizado pruebas de validación para verificar la calidad de los datos y de esta forma proporcionar un servicio probado y validado.
- **Satisfacción:** La empresa donde se ha realizado este proyecto está satisfecha con el trabajo realizado y volvería a solicitar los servicios si fuera necesario.

9.3. Líneas de trabajo futuras

En primer lugar, existe una línea de trabajo clara a realizar de aquí en adelante. Esta línea de trabajo es el mantenimiento y actualización de la aplicación ante las actualizaciones y cambios en el estándar. En algunos casos puede ser opcional el cambio, porque sean complementos, y en otros casos será obligatorio si los cambios afectan al funcionamiento actual de la aplicación.

Respecto a mejoras significativas y más relacionadas con la expansión de la aplicación, se pueden extraer las siguientes:

- **Incorporación de nuevas funcionalidades:** Existe la posibilidad de desarrollar la API para crear un acceso a usuarios específicos que quisieran obtener y utilizar los ficheros generados por este microservicio. Actualmente se realiza una comunicación exclusiva con Google, pero esto puede ser modificado y ampliado.
- **Integración con otras plataformas:** En este caso se ha utilizado el estándar GTFS aplicado a Google. Se puede ampliar la aceptación de datos a varias plataformas que utilicen este estándar o incluso implementar otros estándares para maximizar la compatibilidad y proveer de información en tiempo real a la mayor cantidad de servicios de información.
- **Optimización del rendimiento y escalabilidad:** Será necesario mantener estos requisitos para que funcione con grandes cantidades de datos en un futuro sin ningún problema. En ese caso será necesario dotar de más recursos a la máquina en la que se vaya a ejecutar.

- **Investigación:** Es importante conocer la actualidad y estudiar la utilización de nuevas tecnologías las cuales podrían ser útiles para la mejora de esta aplicación.

Apéndice A

Resumen de enlaces adicionales

A pesar de la importancia de todos los enlaces bibliográficos, se muestran a continuación aquellos de mayor importancia para el proyecto.

- **Acerca del estándar GTFS [2]:**
  `https://gtfs.org/`
- **Documentación del estándar para uso en Google [1]:**
  `https://developers.google.com/transit?hl=es-419`
- **GTFS Static de Google [5]:**
  `https://developers.google.com/transit/gtfs?hl=es-419`
- **GTFS Realtime de Google [7]:**
  `https://developers.google.com/transit/gtfs-realtime?hl=es-419`
- **Documentación acerca de Protobuf [9]:**
  `https://protobuf.dev/overview/`
- **Documentación de validador de GTFS estático [31]:**
  `https://github.com/MobilityData/gtfs-validator`
- **Documentación de validador de GTFS de tiempo real [34]:**
  `https://github.com/MobilityData/gtfs-realtime-validator`

Bibliografía

[1] Google. Google transit. https://developers.google.com/transit?hl=es-419, 2023.

[2] Google. Google transit feed specification. https://gtfs.org/, 2023.

[3] Google. Conceptos básicos sobre google transit. https://support.google.com/transitpartners/answer/1111471, 2023.

[4] Google. Gtfs static. https://developers.google.com/transit/gtfs?hl=es-419, 2022.

[5] Google. Gtfs static reference. https://developers.google.com/transit/gtfs/reference?hl=es-419, 2022. Accessed: May 16, 2023.

[6] GTFS. Gtfs schedule. https://gtfs.org/schedule/reference/, 2022.

[7] Google. Gtfs realtime. https://developers.google.com/transit/gtfs-realtime?hl=es-419, 2022.

[8] Google. Gtfs realtime reference. https://developers.google.com/transit/gtfs-realtime/reference?hl=es-419, 2023.

[9] Google. Protobuf dev. https://protobuf.dev/overview/, 2023.

[10] Claire Drumond. Scrum. https://www.atlassian.com/es/agile/scrum, 2023.

[11] Sakshi Sachdeva. Scrum methodology. *Int. J. Eng. Comput. Sci*, 5(16792):16792–16800, 2016.

[12] Transmodel - cen reference data model for public transport. https://transmodel-cen.eu/, 2019.

[13] Atlasian. Jira software. https://www.atlassian.com/es/software/jira/guides/getting-started/introduction, 2023.

[14] Impresum. Scrum poker. https://www.scrumpoker-online.org, 2023.

[15] EasyRetro. Easyretro. https://easyretro.io/about, 2023.

[16] Atlasian. Confluence. https://www.atlassian.com/es/software/confluence/resources/guides/get-started/overview#about-confluence, 2023.

[17] Microsoft. Intellisense. https://learn.microsoft.com/es-es/visualstudio/ide/using-intellisense?view=vs-2022, 2023.

[18] Microsoft. Visual studio. https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022, 2023.

[19] Software Freedom Conservancy. Git documentation. https://git-scm.com/docs, 2023.

[20] Atlassian. Bitbucket. https://www.atlassian.com/es/software/bitbucket/guides/basics/bitbucket-interface, 2023.

[21] Atlassian. Sourcetree. https://www.sourcetreeapp.com/, 2023.

[22] Postman. Postman. https://www.postman.com/api-platform/, 2023.

[23] Jenkins. Jenkins documentation. https://www.jenkins.io/doc/, 2023.

[24] C# — modern, open-source programming language for .net. https://dotnet.microsoft.com/en-us/languages/csharp, 2023.

[25] Microsoft. ¿qué es .net? https://dotnet.microsoft.com/es-es/learn/dotnet/what-is-dotnet, 2023.

[26] Docker. Docker documentation. https://docs.docker.com/, 2023.

[27] Hewlett Packard. ¿qué es docker? https://www.hpe.com/lamerica/es/what-is/docker.html, 2023.

[28] Kubernetes. ¿qué es kubernetes? https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/, Jul 2022.

[29] Glassdoor. Sueldos para el puesto de software engineer en valladolid. https://www.glassdoor.es/Sueldos/valladolid-software-engineer-sueldo-SRCH_IL.0,10_IC2649917_KO11,28.htm?clickSource=searchBtn, 2023.

[30] mecatran. Gtfsvtor - fast gtfs validator. https://github.com/mecatran/gtfsvtor, Sep 2022.

[31] MobilityData. Mobilitydata gtfs validator project for schedule (static) files. https://github.com/MobilityData/gtfs-validator, Jun 2023.

[32] Google. Feedvalidator. https://github.com/google/transitfeed/wiki/FeedValidator, Jun 2022.

[33] CUTR at USF. Github - cutr-at-usf/gtfs-realtime-validator: Java-based tool that validates general transit feed specification (gtfs)-realtime feeds. https://github.com/CUTR-at-USF/gtfs-realtime-validator, 2022.

[34] MobilityData. Mobilitydata/gtfs-realtime-validator: Java-based tool that validates general transit feed specification (gtfs)-realtime feeds. https://github.com/MobilityData/gtfs-realtime-validator, Jan 2023.