Escuela Técnica Superior de Ingeniería de Telecomunicación

Diseño y desarrollo de un microservicio para la gestión de información de monitorización y predicciones de tráfico en red

30 de noviembre de 2022

TRABAJO FIN DE MÁSTER

Máster Universitario en Ingeniería de Telecomunicación

Autor: Enrique Fernández Sánchez

Tutor: Pablo Pavón Mariño

Universidad Politécnica de Cartagena

Índice general

Índice de figuras

Listado de ejemplos

Capítulo 1

Introducción

1.1 Contexto del trabajo

1.2 Motivación

1.3 Descripción Global

1.4 Objetivos

1.5 Resumen capítulos de la memoria

Capítulo 2

Tecnologías empleadas

En este capítulo, se van a presentar las diferentes tecnologías utilizadas para la implementación de la aplicación.

2.1 Arquitectura y microservicios

En primer lugar, se va a comentar acerca de la arquitectura escogida. En este caso, se decide realizar una implementación basada en microservicios utilizando una REST API.

Arquitectura basada en microservicios

Lo primero, es entender en que consiste un microservicio. Para ello, podemos definirlo como los sistemas que cumplen las siguientes premisas: [2]

- Los microservicios son sistemas pequeños, independientes y poco "acoplados".
- Cada servicio tiene su propio código fuente, que esta separado del resto de códigos de los servicios.
- Cada servicio se puede desplegar de manera independiente.
- Cada servicio es responsable de la persistencia de sus datos.
- Los servicios se comunican entre sí utilizando APIs
- Además, como ventaja, los servicios no tienen por qué estar implementados todos en el mismo lenguaje de programación.

Por lo tanto, dado los objetivos presentados en este trabajo, se llegó a la conclusión de que tratar el sistema propuesto como un microservicio podría aportar numerosas ventajas, ya que permitiría ser utilizado por otros servicios, extendiendo la funcionalidad de estos y añadiendo un valor extra. Para ello, será necesario definir la API que utilizaremos para comunicarnos con el sistema.

Comunicación basada en API

Una API permite a dos componentes comunicarse entre sí mediante una serie de reglas. Además, supone un "contrato" en el que se establecen las solicitudes y respuestas esperadas en la comunicación. [1]

Dependiendo de la implementación de la API que se realice, distinguimos cuatro tipos de API:

- API de SOAP. Utilizan un protocolo de acceso a objetos. Los interlocutores intercambian mensajes XML. En general, es una solución poco flexible.
- API de RPC. Basado en llamadas de procedimientos remotos. El cliente ejecuta una función en el servidor, y este responde con la salida de la función.
- API de WebSocket. Solución moderna de desarrollo de API, que utiliza objetos JSON y un canal bidireccional para realizar la comunicación entre el cliente y el servidor.
- API de REST. Solución más popular. El cliente envía solicitudes al servidor como datos, utilizando métodos HTTP. Es una opción muy flexible.

En en caso de nuestra aplicación, se decidió utilizar el tipo REST API, ya que permite una sencilla implementación de cara al cliente que quiera utilizar dicha interfaz.

2.2 Bases de datos

Para asegurar la persistencia de los datos en nuestra aplicación, es necesario utilizar una base de datos. En dicha base de datos, guardaremos información relevante para el correcto funcionamiento del sistema, en nuestro caso, redes y/o interfaces a monitorizar, o los datos monitorizados.

En la aplicación de monitorización, distinguimos entre datos de dos tipos:

- Datos clásicos. Como por ejemplo, la información asociada a una red a monitorizar.
- Datos de tipo "time series". Como por ejemplo, las muestras de monitorización de una red.

En primer lugar, se diseña una base de datos tipo SQL para almacenar los "datos clásicos". Y por otro lado, se diseña una base de datos diferente, especializada para el almacenamiento de datos tipo "time series", en este caso, se elige una base de datos llamada InfluxDB.

2.2.1 Base de datos tipo relacional/SQL

SQL es una base de datos de tipo relacional. Dichas bases de datos, suponen una colección de información que organizan los datos en una serie de "relaciones" cuando la información es almacenada en una o varias "tablas". Por lo tanto, las relaciones suponen conexiones entre diferentes tablas, permitiendo así una asociación entre información diferente. [3]

Por ejemplo, si vemos la figura 2.1, podemos comprobar como se realizan las relaciones entre las diferentes tablas (Ratings, Users, Movies o Tags), se realiza mediante uno de los campos definidos en la propia tabla. Por ejemplo, el campo "user_id" de la tabla Ratings, permite una relación con la tabla Users, con el campo "id".

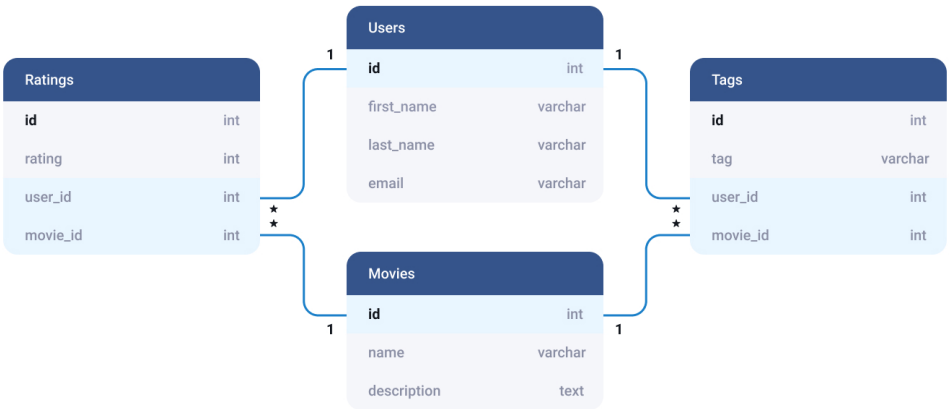

Figura 2.1: Ejemplo de relaciones dentro de una base de datos SQL. [1]

Para el caso de nuestra aplicación de monitorización de tráfico en red, modelamos la base de datos según la información que vamos a almacenar. Por lo que tenemos que definir la estructura de tablas, los campos que van a tener cada una de las tablas, y los campos por los que se van a relacionar entre sí. A esta información la llamamos modelo de datos.

Modelos de datos

El primer modelo de datos de la aplicación, es el referido a la información de una red a monitorizar. La descripción del modelo la podemos ver en la figura 2.2.

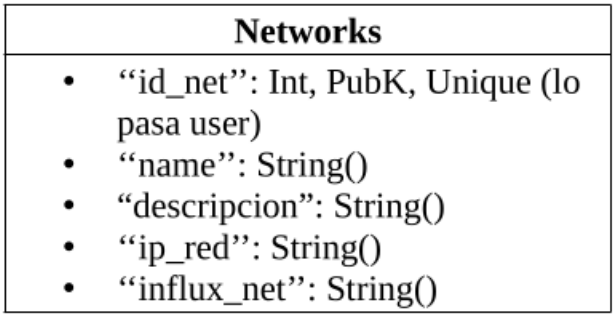

Figura 2.2: Modelo de datos para las redes a monitorizar. Equivale con la tabla ”networks”.

El segundo modelo de datos de la aplicación, es el referido a la información de una interfaz a monitorizar, que esta dentro de una red monitorizada. La descripción del modelo la podemos ver en la figura 2.3.

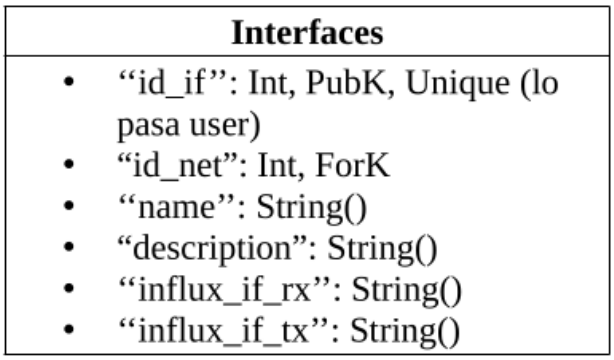

Figura 2.3: Modelos de datos para las interfaces a monitorizar. Equivale con la tabla ”interfaces”

Se establece una relación del modo que una interfaz solo puede pertenecer a una red monitorizada, y que además, una red monitorizada puede tener muchas interfaces. Dicha relación se realiza mediante el campo ”id_net” de la tabla Interfaces.

Por último, la base de datos SQL utilizada para esta aplicación es PostgreSQL [4], ya que además de ser Open Source, permite una gran escalabilidad, amoldándose a los recursos de la máquina en la que esté funcionando.

2.2.2 Base de datos tipo "time series"/InfluxDB

2.3 Modelo de predicción

2.4 Lenguajes de programación y frameworks

2.5 Tecnologías utilizadas en un despliegue en producción

Capítulo 3

Diseño e implementación del sistema

3.1 Descripción REST API

3.2 Estructura de la aplicación

3.3 Modelos de datos

3.4 Endpoints

3.5 Implementación del sistema

Capítulo 4

Pruebas y validación del sistema

Capítulo 5

Conclusiones

5.1 Propuestas futuras

Capítulo 6

Bibliografía

Enlaces y referencias

1. ¿Qué es una API?
2. Microservice architecture style
3. What is a relational database?
4. PostgreSQL

Figuras

1. Relational Databases

Anexos

Anexo I. Generación dataset sintético