Definición OpenAPI

Microservicio “Monitorización de tráfico y predicciones”

Version: 0.2

Siguiente version en swagger, con los JSON de entrada y de salida.

Investigar documentación fields JSON en swagger, que cada campo pueda tener su explicación.

Realizado por: Enrique Fernández Sánchez

Indice

CRUD Enpoints

- ‘’*/networks*’’ → Acceso a la información de una red.
- ‘’*/networks/<id_net>/interfaces*’’ → Acceso a la información de interfaz de una red.

Networks

Resumen

CRUD asociada a la creación de una red en la que aplicaremos la monitorización. Dentro de la red, tendremos diferentes interfaces que serán dadas de alta en la aplicación para ser monitorizadas. Una única red puede tener muchas interfaces en su interior.

Endpoint base → “/networks”

Definición

Recurso	GET	POST	PUT	DELETE
Colección de redes: /networks	Lista de redes dadas de alta, devolvemos metadata	Añade una nueva red. El usuario da la ID	*	*
Red en particular: /networks/<id_net>	Información de una red en particular	*	Modificamos la información de una red	Eliminamos una red

* Denegado el acceso

Interfaces

Resumen

CRUD asociada a la creación de interfaces dentro de una red a monitorizar. Sirve para dar de alta las interfaces que queremos que sean monitorizadas. Una interfaz puede tener muchas muestras guardadas, dichas muestras están almacenadas en la base de datos para time series InfluxDB (https://www.influxdata.com/)

Endpoint base → "/networks/<id_net>/interfaces"

Definición

Recurso	GET	POST	PUT	DELETE
Colección de interfaces: /networks/<id_net>/interfaces	Listado de interfaces dentro de una red. Devuelve metadatos de la red	Añade una nueva interfaz a una red. La ID la da el usuario.	*	*
Interfaz en particular: /networks/<id_net>/interfaces/<id_if>	Información de una interfaz en particular	*	Modificamos la información de una interfaz	Eliminamos una interfaz

* Denegado el acceso

Non-CRUD Endpoints

- ‘‘/query/<id_net>/values’’ → Realizar peticiones de información sobre la información de monitorización almacenada.
- “/forecast” → Ejecutar predicción dado un JSON de entrada. Es una llamada de tipo “server-side event” (SSE).

Query enpoint

Resumen

Endpoint que permite hacer querys para obtener datos acerca de la monitorización que esta almacenada en la aplicación.

Propuesta de endpoint: ‘‘/query/<id_net>/values’’ o ‘‘/query/<id_net>/values/timeseries’’

Definición

Para ejecutar una petición, se realiza un GET a ‘‘/query/<id_net>/values’’, para filtrar los datos podemos utilizar *Query Strings.*

Query_strings disponibles:

- ‘‘keys’’ → Podemos seleccionar entre “tx”, “rx”
- ‘‘startTs’’ → Epoch de inicio para devolver los datos (ms)
- ‘‘endTs’’ → Epoch de fin para devolver los datos (ms)
- ‘‘limit’’ → Limitar el número de puntos de salida
- ‘‘interval’’ → Intervalor entre muestras (ejemplo: 600000 ms → 10 min)

Filtering: un JSON que nos permite filtrar la salida de datos (percentil, tipo, valor diario…) PLACE_HOLDER, permite hacer esquemas de filtrado. Las query strings dentro del JSON también, asi tenemos todo dentro. Cuando hagamos predicción, podemos reutilizar este mismo filtrado.

/query devuelve las muestras

/forecast un fit con solo los datos filtrados (JSON de configuracion), y los parametros de prophet también dentros de un JSON.

El servidor devuelve:

{“<keys>”: [{“ts”: <epoch>, “value”: “<valor>”}, {“ts”: <epoch>, “value”: “<valor>”}, ...]}

Forecast endpoint

Resumen

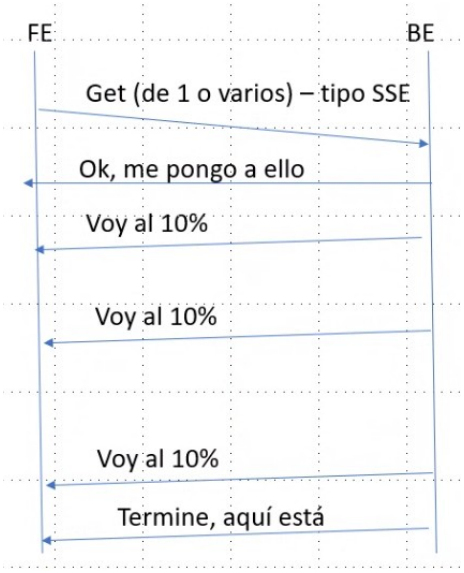
Endpoint asociado a la creación de predicciones dada una red a monitorizar. Permite lanzar las predicciones y obtener los resultados asociados a ellas. Se realiza una predicción por cada interfaz de la network. Para cada predicción, tenemos que elegir si lo realizamos sobre el “link count” de TX o de RX.

Si queremos realizar predicciones en batch, eso se ve más adelante.

Endpoint → “/forecast/<id_fore>” [POST con JSON de configuración]

Se opta por que la predicción no se guarde (ejemplo, borramos del influxdb la información pasado X días{en el caso de guardar algo}). La ejecución se realiza como un SSE (server-side event), el servidor ejecuta y va mandando actualizaciones de la ejecución. Una vez terminada la ejecución, enviamos un JSON con toda la info.

Entendemos el forecast como un proceso dividido en: FIT y Make prediction, según la fase en la que estemos, vamos devolviendo el % en el SSE para que el frontend pueda actualizar el porcentaje.



Definición

Recurso	GET	POST	PUT	DELETE
Colección de predicciones: /forecasts	Listado de predicciones	Creamos una predicción [*Usamos JSON file para delimitar la predicción*]	*	*
Predicción en particular: /forecasts/<id_fore>	Información de un forecast	*	*	Eliminamos una predicción
Ejecución de una predicción: /forecasts/<id_fore>/exec	*	Ejecutamos la predicción	*	*

* Denegado el acceso

InfluxDB

Base de datos para datos tipo timeseries. Aquí almacenamos todas las muestras de la monitorización, con el objetivo de agilizar la gestión de los datos de tipo timeseries. Para ello, configuramos la base de datos tal que:

- bucket → Corresponde con la aplicación a gestionar, en este caso sería la presente aplicación de monitorización de tráfico.
- _measurement → Corresponde con la red en particular a la que añadiremos muestras. Por lo tanto, recibirá el nombre asociado a una rede a monitorizar.
- _tag → Corresponde con las muestras asociadas a una interfaz, tendremos dos interfaces para cada una de nuestro sistema, una interfaz para el “link count” de RX y otra para TX.
- _field → Corresponde con el valor de “link count” de 5 minutos asociados a una interfaz.

Prophet

Paquete de Python desarrollado por Facebook (Open Source: https://github.com/facebook/prophet), que permite realizar predicciones sobre una serie de datos, siguiendo un modelo no lineal basado en tendencias, que puede seguir diferentes tendencias a la vez (anual, semanal, diario o efectos de vacaciones).

Página principal: https://facebook.github.io/prophet/

Documentación: https://facebook.github.io/prophet/docs/quick_start.html

Parametros configurables dentro de Prophet:

- ‘‘**holidays**’’ → dataframe con columnas ‘‘ds’’ (fecha: *%Y-%M-%d*) y ‘‘holiday’’ (nombre). Permite configurar las fechas de vacaciones. Además, está la posibilidad de usar al función: ‘‘*add_country_holidays(country_name=’ES’)*’’ para utilizar las vacaciones precargadas. [ref]
- ‘‘**seasonality**’’ → Valor default 10. Si queremos detectar mayor frecuencia de cambios, y que en general sean menos suaves las variaciones, podemos incrementar el valor. [ref]
- ‘‘**seasonality_mode**’’ → by default, prohphet no utiliza más de un factor de seasonality, por lo que para tener en cuenta todos los efectos, y obtener buenos resultados en datasets que tienden a crecer con el tiempo siguiendo diferentes seasonality (como es el crecimiento del tráfico), hay que usar: ‘‘*seasonality_mode=’multiplicative’* ’’ [ref]
- ‘‘**interval_witdh**’’ → TODO
- TODO

En la documentación final no queremos mostrar nada de Prophet, queremos que sea algo oculto.

Prophet solo saca muestras unidemensionales, por lo que tendremos que realizar una predicción por una interfaz.

Propuesta → Modificar funcionamiento de Prophet según un archivo de configuración en JSON que se envia en el POST del forecast.

Ejemplos prácticos en Python:

- https://www.oncrawl.com/technical-seo/forecasting-seo-traffic-prophet-python/
- (Interesante) https://vijayv500.medium.com/time-series-forecasting-using-facebooks-prophet-in-python-1e13ea20a52b
- https://www.kaggle.com/code/sudosudoohio/forecasting-web-traffic-with-prophet-in-python/notebook
- (Interesante) https://pbpython.com/prophet-overview.html

Modelos de datos

Networks

Networks
• ‘‘id_net’’: Int, PubK, Unique (lo pasa user) • ‘‘name’’: String() • “descripcion”: String() • ‘‘ip_red’’: String() • ‘‘influx_net’’: String()

Interfaces

Interfaces
• ‘‘id_if’’: Int, PubK, Unique (lo pasa user) • “id_net”: Int, ForK • ‘‘name’’: String() • “description”: String() • ‘‘influx_if_rx’’: String() • ‘‘influx_if_tx’’: String()

Importar datos de otros providers

Subimos un conjunto de muestras, siempre

Por cada interfaz, post de muestras (con id_net y con id_if, y con rx o tx) y get de muestras (con parametros de filtrado, con paginacion ejemplo fecha inicio fecha de fin)

Reference links

- https://blog.stoplight.io/crud-api-design