APIs REST y Django Rest Framework

Ezequiel Remus: < *ezequielremus@gmail.com* >

Índice

1. Introducción

1.1. ¿Qué es REST ?

REST es cualquier interfaz entre sistemas que use HTTP [1] para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como **XML** y **JSON**. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como **SOAP** (*Simple Object Access Protocol*), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST .

1.2. Características de REST

- **Protocolo cliente/servidor sin estado:**
  Cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como protocolo cliente-caché-servidor sin estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.

- **Operaciones:**
  Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro:

  1. **POST** (crear)
  2. **GET** (leer y consultar)
  3. **PUT** (editar)
  4. **DELETE** (eliminar)

- **Manipulación:**

  Los objetos en REST siempre se manipulan a partir de la URL . Es la URL y ningún otro elemento el identificador único de cada recurso de ese sistema REST . La URL nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

- **Interfaz uniforme:**

  Para la transferencia de datos en un sistema REST , este aplica acciones concretas (**POST**, **GET**, **PUT** y **DELETE**) sobre los recursos, siempre y cuando estén identificados con una URL . Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.

1.3. Ventajas

- **Separación entre el cliente y el servidor:**

  El protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.

- **Visibilidad, fiabilidad y escalabilidad:**

  La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.

- **La API REST siempre es independiente del tipo de plataformas o lenguajes:**

  La API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores **PHP**, **Java**, Python o **Node.js**. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente **XML** o **JSON**.

[1] La petición o HTTP request es el mensaje que se envia desde el cliente al servidor para solicitar un resource (Refe)

Referencias

[1] Documentación Oficial django REST
bbvaopen4u

[2] Adictos al trabajo

[3] Rip Tutorial

[4] Coreapi Documentación