Tu primera REST API con Python y FastAPI

Meetup 01: Python para todos

Jose Laruta

Python La Paz - Septiembre, 2021

Agenda

1. El protocolo HTTP en la web
2. REST
3. APIs REST
4. FastAPI
5. Demo

HTTP - Hypertext Transfer Protocol

Protocolo sobre la capa de **aplicación** base de la comunicaciòn en la web.

Los documentos de **hipertexto** incluyen **hipervínculos** a otros recursos y son simples de acceder.

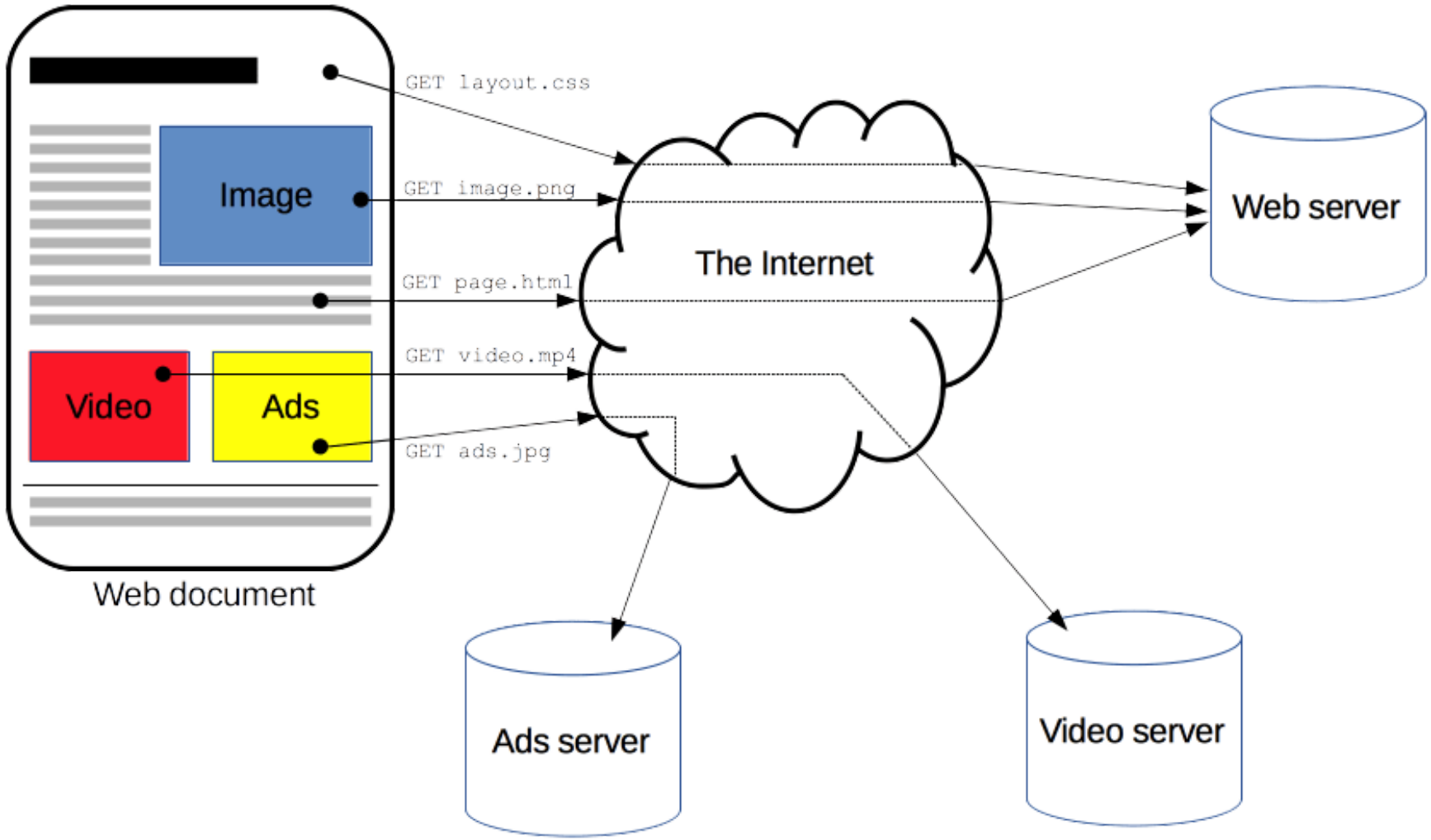
HTTP es un protocolo sin estado o *stateless*, sin embargo se permite el uso de sesiones a través de *cookies*.

HTTP - Hypertext Transfer Protocol

HTTP está basado en el flujo **petición-respuesta** o *request-response* sobre un modelo cliente-servidor.

- El cliente envía una petición o *request* al servidor 🚀.
- El servidor responde con un recurso (ej: archivo HTML) 🚚

La respuesta o *response* posee información del resultado de la petición y otro contenido en el *body*


GET layout.css
Image
GET image.png
GET page.html
Video
Ads
GET video.mp4
GET ads.jpg
Web document
The Internet
Web server
Ads server
Video server

HTTP request

El cliente envía una solicitud al servidor que consiste de:

- Un método y la versión del protocolo.
- Cero o más cabeceras o *headers*.
- Un cuerpo de mensaje o *body* opcional.

HTTP request methods

Existen diversos métodos o verbos para indicar la acción requerida por un cliente.

- **GET**: Recuperar el estado y `leer` datos, no debería tener efectos secundarios.
- **HEAD**: Recuperar el estado y `metadatos`.
- **POST**: Procesar, enviar datos o `crear` un recurso, usualmente modifica el estado del servidor o recurso requerido.
- **PUT**: `Actualizar` datos o recursos existentes.
- **DELETE**: Solicitar `remover` un recurso.

HTTP Response messages

A cada solicitud, el servidor responde con un mensaje al cliente que consiste en:

- Una línea de status con la versión del protocolo y el *response status code*.
- Cero o más cabeceras o *headers*.
- Un cuerpo de mensaje o *body* opcional.

HTTP response status codes

El status code es un número entero de tres dígitos que representa el resultado de la solicitud enviada desde el servidor.

- `1XX`: Información
- `2XX`: Éxito
- `3XX`: Redirección
- `4XX`: Error del cliente
- `5XX`: Error del servidor

HTTP response status codes

El status code es un número entero de tres dígitos que representa el resultado de la solicitud enviada desde el servidor.

- `100 Continue`
- `200 OK`, `201 Created`, `206 Partial Content`
- `301 Moved Permanently`, `302 Found`
- `404 Not Found`, `400 Bad Request`, `422 Unprocessable Entity`
- `500 Internal Server Error`, `502 Bad Gateway`, `503 Service Unavailable`

REST - Representational state transfer

REST define un estilo de arquitectura de software con un conjunto de condiciones para el desarrollo de un sistema distribuido.

REST hacer énfasis en la escalabilidad, interación, interfaces uniformes y una arquitectura en capas.

Es ampliamente utilizado en la industria para el desarrollo de APIs web.

REST Architectural constraints

- Arquitectura Cliente-Servidor
- Stateless
- Cacheability
- Layered
- Uniform Interface

API - Application Programming Interface

Una API representa un intermediario entre dos aplicaciones. De esta forma, la funcionalidad de muchos sistemas se puede reutilizar o encapsular exponiendo solamente una interfaz (API).

Existen estándares para la implementación de APIs de diversos tipos. Se suelen tratar más como productos independientes que como código.

Como cualquier pieza de software moderno, las APIs poseen su propio ciclo de desarrollo, versionamiento y documentación.

RESTful API

Una API de un servicio web que sigue las restricciones de arquitectura REST se denomina RESTful API, usualmente posee:

- Una URL base: `http://api.mi-dominion.com/`.
- Métodos HTTP estándar: `GET, POST, PUT, DELETE`.
- Un tipo de representación de transición de estado, `JSON, XML`.

FastAPI

FastAPI es un framework web para Python que permite crear APIs RESTful:

- Validación de datos con modelos de Pydantic.
- Generación automática de documentación.
- Alto rendimiento con ejecución asíncrona.

FastAPI

En FastAPI, es sencillo crear una API en pocas líneas de código:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def root():
  return {"message": "hola mundo"}
```

Modelos de pydantic

Se puede definir modelos de datos para validación automática y serialización:

```
from pydantic import BaseModel
class Stats(BaseModel):
    hp: int
    attack: int
    defense: int
    speed: int
class Pokemon(BaseModel):
    id: int
    name: str
    type: List[str]
    stats: Stats
```

Generación de documentación

Usando los modelos de pydantic, FastAPI puede usar esa información para generar documentación automáticamente:

```
@app.get("/pokemons/{id}", response_model=Pokemon)
def get_pokemon(id: int):
  pokemon = db.get(id)
  return pokemon
```

Demo Time!

repo en github

¿Preguntas?

Muchas gracias!