**[API Developer] Examen final**

1. **Para qué se puede usar Python en lo que respecta a datos. Dar 5 casos y explicar brevemente**

Python tiene muchas aplicaciones con datos, entre ellas podemos citar:

* Visualizar conjuntos de datos
* Modelar comportamientos de datos para predicción y clasificación
* Manejar motores de bases de datos
* Implementar APIs para intercambio de datos entre aplicaciones
* Publicar servicios web basados en datos no estructurados

1. **¿Cómo se diferencian Flask de Django? Argumentar.**

Flask es un micro-framework con el que podemos implementar aplicaciones pequeñas de manera rápida y modular. Por ejemplo, en clase creamos una API y un servicio de web sockets usando Flask. Por otro lado, Django es un framework para aplicaciones web escalables que tiene incorporado funcionalidades importantes como son ORM y una capa de gestión. En conclusión, para un proyecto pequeño con funcionalidades simples sería más adecuado Flask; mientras que para un proyecto complejo lo sería Django.

1. **¿Qué es un API? Explicar en sus propias palabras.**

Una API es una herramienta con la que podemos intercambiar datos entre distintas aplicaciones sin importar cómo estén construidas o en qué lenguaje. Un ejemplo muy común de uso de una API son las aplicaciones de Smartphones que proporcionan información sobre el clima, las cuales realizan consultas a servidores que les devuelven las predicciones en formato JSON. Existen distintos tipos de APIs que se han venido usando, siendo REST el estándar más popular en la actualidad.

1. **¿Cuál es la principal diferencia entre REST y WebSockets?**

La principal diferencia es que el REST trabaja de manera *stateless,* es decir cada interacción es independiente de las interacciones previas y debe contener toda la información necesaria como son la acción, el recurso y los datos asociados; mientras que Web Sockets trabaja de manera persistente y bidireccional, por lo que los datos pueden ir de un lado a otro y se puede instanciar datos pasados. Por ejemplo, REST se puede usar para consultar el pronóstico del clima para los siguientes días; y pese a que éste pueda cambiar no hay manera de enterarse si no se vuelve a consular; mientras que Web Sockets puede estar implementado en un juego en línea en donde el servidor está informando a los usuarios del estado general del juego.

1. **Describir un ejemplo de API comercial y como funciona – usar otros ejemplos no vistos en el curso.**

Una API comercial bastante usada es la de Flicker mediante la cual podemos obtener imágenes subidas a la plataforma por otros usuarios, como también podemos publicar en ella desde nuestra propia aplicación. En el caso de Flicker debemos crear una cuenta para poder hacer las consultas mediante una API Key que es gestionada mediante OAuth. El servicio cuenta con un límite de consultas de 3600 consultas por hora.

**Parte práctica:**

Sobre este laboratorio (<https://github.com/aladelca/my_first_API/tree/main>) visto en clase, convertir las apis a REST API con soporte de motor de base de datos (ver referencia aquí <https://github.com/aladelca/app_sql_flask>). Es decir, las apis ya no tienen que estar en el archivo swagger, sino en el archivo .py principal. Deberás implementar los siguientes métodos:

• Método GET – lista de estudiantes (2 puntos)

• Método GET <id> - estudiante por id (2 puntos)

• Método POST – crear estudiante nuevo (2 puntos)

• Método PUT <id> - actualizar estudiante (2 puntos)

• Método DELETE <>id – eliminar estudiante (2 puntos)

Debes subir pantallazos de Postman mostrando los métodos y su correcto funcionamiento.

Responder todo en un archivo pdf y/o word, además de los archivos .py/.html/etc (si crean un repo sería mucho más fácil pero no es obligatorio).

**Repositorio**

https://github.com/jorgevalverdev/api\_developer\_exam

**# Método GET – lista de estudiantes**

http://127.0.0.1:5000/estudiantes

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**# Método GET <id> - estudiante por id**

http://127.0.0.1:5000/estudiantes/3

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**# Método POST – crear estudiante nuevo**

http://127.0.0.1:5000/estudiantes

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**# Método PUT <id> - actualizar estudiante**

http://127.0.0.1:5000/estudiantes/2

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

**# Método DELETE <>id – eliminar estudiante**

http://127.0.0.1:5000/estudiantes/4

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.