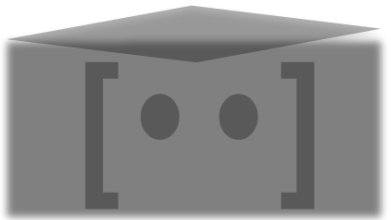
ANÁLISIS DE DATOS

Profesor: Nahuel Meza

Cod.Ar[• •]

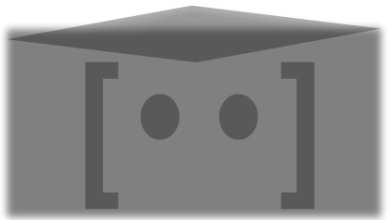
REPASO

¿Que vimos la clase anterior?



Resolución de ejercicios

Resolvemos los ejercicios de la clase anterior.



Módulo 3

Empezaremos con el dictado del módulo 3

UNIDAD 3: Introducción a los pedidos HTTPS

- Introducción a datos JSON.
- Conocer tipos de pedidos GET, POST, PUT, DELETE.
- Interactuar con distintas APIs.

¿Que vamos a ver ahora?

- Qué es HTTPS
- Que es una API
- Cómo pedir información a una API

¿Qué es HTTPS?

HTTPS = Hypertext Transfer Protocol Secure

Es la versión **segura** del protocolo HTTP, que se usa para transferir datos entre tu navegador y un sitio web.

¿Qué es HTTPS?

¿HTTP?¿No es lo mismo?
no jiji

HTTP es el **lenguaje de comunicación** que usan tu navegador y los servidores web.

¿Qué es HTTP?

Sirve para pedir y recibir, así como enviar informacion.

Utiliza distintos métodos para completar estos procesos.

Ejemplo cotidiano:
http://www.source.com

¿Qué es HTTPS?

Entonces, ¿Qué es HTTPS?

Es HTTP, pero con una capa extra. La de SECURITY

Esto permite que antes de enviar o recibir información, se encripte o desencripte según lo necesitado.

Métodos HTTP

HTTP tiene diversos **métodos**, en consecuencia HTTPS también.

Estos métodos son los que permiten saber que tipo de información le estás pidiendo al protocolo.
HTTP tiene **4 métodos principales** y cada uno tiene sus **Headers** (Encabezados)

Método GET

Sirve para PEDIR información al servidor

```
GET /api/usuarios HTTP/1.1
Host: ejemplo.com
User-Agent: Mozilla/5.0
Accept: application/json
Authorization: Bearer TOKEN123
```

Método POST

Sirve para ENVIAR (y crear) información al servidor

```
POST /api/usuarios HTTP/1.1
Host: ejemplo.com
Content-Type: application/json
Authorization: Bearer TOKEN123
Content-Length: 51

{
  "nombre": "Ana",
  "edad": 25
}
```

Método PUT

Sirve para ACTUALIZAR información del servidor

```
PUT /api/usuarios/5 HTTP/1.1
Host: ejemplo.com
Content-Type: application/json
Authorization: Bearer TOKEN123

{
  "nombre": "Ana María",
  "edad": 26
}
```

Método DELETE

Sirve para ELIMINAR información del servidor

```
DELETE /api/usuarios/5 HTTP/1.1
Host: ejemplo.com
Authorization: Bearer TOKEN123
```

Respuesta HTTP

Eso fueron nuestras peticiones, ¿Con que responde HTTP? Hay distintos **códigos de estado (status code)** que nos dice sobre el resultado de nuestra petición.

Status Code: 200

Respuesta exitosa. todo OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 47

{
  "id": 1,
  "nombre": "Ana",
  "edad": 25
}
```

Status Code: 201

Respuesta exitosa de creación.

```
HTTP/1.1 201 Created
Location: https://api.ejemplo.com/usuarios/5
Content-Type: application/json

{
  "id": 5,
  "nombre": "Luis",
  "edad": 30
}
```

Status Code: 400

Respuesta de error, de parte del cliente

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "error": "Faltan campos requeridos"
}
```

Status Code: 401

Respuesta de error, no hay autorización

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="Acceso API"
Content-Type: application/json

{
  "error": "Token inválido o faltante"
}
```

Status Code: 404

Respuesta de error, no encontrado

```
HTTP/1.1 404 Not Found
Content-Type: text/plain

Recurso no encontrado
```

Finalizamos HTTP

Con eso sabemos lo básico y de cultura general de HTTPS, podemos pasar a lo que sigue.

¿Que es una API?

Una **API (Application Program Interface)** es un conjunto de **reglas y protocolos** que permiten que diferentes aplicaciones se **comuniquen** entre sí.

Metáfora API

Imaginá que una **API** es como el **mozo** de un restaurante. Vos, como cliente, no vas directamente a la cocina, ni preparás la comida. En cambio:

- Vos **hacés un pedido** al mozo (API).
- El mozo lleva tu pedido a la cocina (el sistema o servidor).
- La cocina prepara lo que pediste.
- El mozo te trae la comida (la respuesta de la API).

Ejemplo API

Un ejemplo cotidiano sobre las APIs pueden ser el propio **sistema de clima** de tu celular.
Cuando vos ingresas a ver el clima de tu celular, la aplicación hace un llamado a la **API del clima,** para obtener la información

API REST

Vamos a utilizar principalmente **APIs REST**, que son APIs que se basan en el **principio REST**.
El cual define las **reglas** de cómo debe ser la API. No vamos a indagar más a fondo en esas reglas, pero es una buena tarea de investigación.

La respuesta típica de llamar a una API es **JSON**

API REST

Vamos a utilizar principalmente **APIs REST**, que son APIs que se basan en el **principio REST**.
El cual define las **reglas** de cómo debe ser la API. No vamos a indagar más a fondo en esas reglas, pero es una buena tarea de investigación.

La respuesta típica de llamar a una API es **JSON**

Exploremos un poco

Vemos cómo funcionaría una API y sus métodos.

https://jsonplaceholder.typicode.com/

https://pokeapi.co/api/v2/pokemon/ditto

Como pedir la información

como podemos acceder a esta información desde nuestra aplicación?

Hacemos **FETCH.** Que es hacerle un pedido a la API y traernos su respuesta.

FETCH en Python

Se adivina?...

….

Si, librerías.

requests