Summer School 2025

Backend

Sergio Morales & Andrei

0xDecode

Deusto Electronic Club Of
Developers & Engineers

Vasile Andrei Popan

Soy estudiante de 2º del grado en carrera en Ingeniería Informática en la Universidad de Deusto.

He participado en el Hackathon de la UPC, así como en varios CTFs.

Puedes encontrarme o seguir mi trabajo aquí:
Linkedin: https://es.linkedin.com/in/vasile-andrei-popan
Github: https://github.com/vasileandreipopan

Sergio Morales

Soy técnico superior en desarrollo de aplicaciones web con más de un año de experiencia en LKS Next.

Actualmente estoy cursando 2º del Grado en Ingeniería informática de la Universidad de Deusto.

Me gusta programar y los macarrones.

Y... prefiero windows a Linux (es una secta)

Puedes encontrarme en:
Web: https://sergiomorales.dev
Linkedin: https://es.linkedin.com/in/sergiomoralescobo
Github: https://github.com/sergitxin22

Sergio Morales & Vasile Andrei Popan

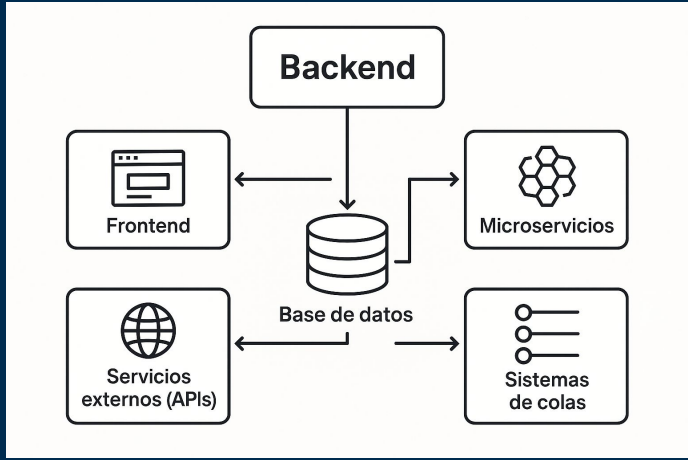
¿Qué es el backend?

- Es la parte del software que no se ve, pero que hace funcionar todo por detrás.
  - Procesar lógica (ej. calcular total de un carrito)
  - Gestionar datos (leer/escribir en la bd)
  - Control de acceso y usuarios (autenticación...)

- *Ejemplo: Cuando compras algo en Amazon, el backend se encarga de decidir si el producto está disponible, calcula los impuestos, guarda el pedido en la bd y te responde.*

Relación con otros componentes

- El backend **no trabaja solo**.
- **BD:** Aquí se guarda la información.
  - Se conecta a través de drivers (MongoDB...) o consultas SQL.
- **Frontend:** La parte visual de la web, app...
  - Se comunica con el Backend usando APIs (normalmente por HTTP)
- **Microservicios:** Pequeños servicios personalizados.
  - Usan HTTP, colas o mensajes para coordinarse.
- **Servicios externos (APIs):** APIs de terceros [pagos (Stripe)...]
  - Se llama desde el backend como si fuese otro cliente más.
- **Sistemas de colas:** Para tareas en segundo plano o muy pesadas (reportes...).
  - El backend "envía trabajos" a una cola (ej. Redis + BullMQ) y otro proceso los ejecuta



¿Qué es una API (Application Programming Interface)?

- Puerta de entrada al backend para que otras apps se comuniquen.
    - *Es como el menú de un restaurante: tú (cliente) ves lo que puedes pedir (endpoints), haces una solicitud, y la cocina (servidor/backend) te devuelve un plato (respuesta).*
- **Sirve para:**
    - Conectar el frontend con el backend.
    - Permitir que servicios externos accedan a tu sistema.
    - Automatizar tareas o integrarse con otros sistemas (enviar un email desde tu backend usando una API de Mailgun).

Funcionamiento de una API REST

- **REST (Representational State Transfer):** Es un estIlo de arquitectura que usa el protocolo HTTP para acceder a recursos a través de URLs llamadas *endpoints*.
- **Tipo de datos que viajan en una API REST:**

Tipo	Dónde va	Ejemplo	¿Para qué sirve?
Query	En la URL, después del ?	/productos?categoria=ropa	Filtrar, buscar, paginar
Path	En la ruta directamente	/productos/5	Identificar un recurso concreto
Body	En el cuerpo del mensaje HTTP	{ "nombre": "Zapatilla" }	Enviar información (crear, actualizar)
Headers	En los metadatos de la petición	Authorization: Bearer TOKEN123	Autenticación, formatos, control

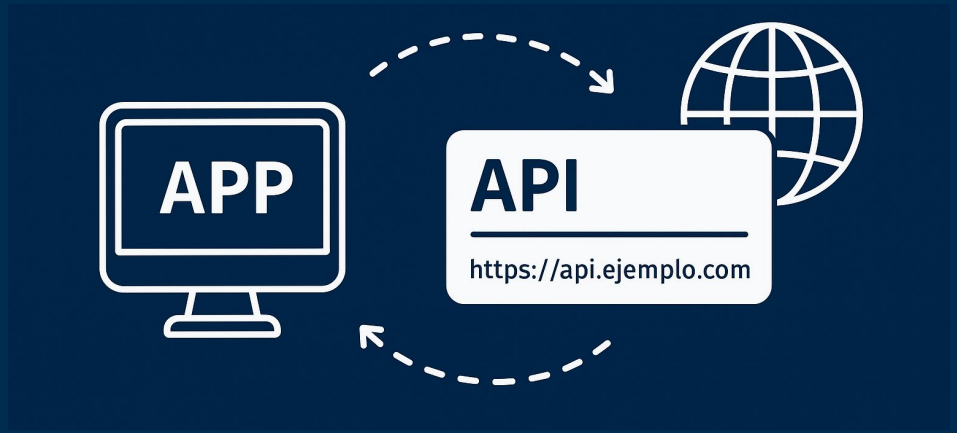
Funcionamiento de una API REST

- **Imagina que tienes una app de tienda online. La API tendría endpoints como estos:**

Método	Endpoint	¿Qué hace?
GET	/productos	*Devuelve todos los productos*
GET	/productos/5	*Devuelve el producto con ID 5*
POST	/productos	*Crea un producto nuevo*
PUT	/productos/5	*Actualiza completamente el producto 5*
DELETE	/productos/5	*Elimina el producto 5*

Exponer una API

- Exponer una API significa “hacer que otros puedan usarla” a través de una URL pública.
- Se hace usando frameworks (FastAPI, Express...)
- Cada endpoint se define con una ruta y un método (GET /productos)



Ej. weatherAPI


```
{
  "current_condition": [
    {
      "temp_C": "17",
      "weatherDesc": [
        { "value": "Partly cloudy" }
      ],
      "humidity": "72",
      "wind_kph": "13"
    }
  ],
  "nearest_area": [
    {
      "areaName": [{ "value": "Bilbao" }],
      "region": [{ "value": "Basque Country" }],
      "country": [{ "value": "Spain" }]
    }
  ]
}
```


```python
import requests

def get_weather(city):

    #Sends request to the API server
    url = f"https://wttr.in/{city}?format=j1"
    response = requests.get(url)

    if response.status_code == 200:

        data = response.json()

        temp = data["current_condition"][0]["temp_C"]

        weather_desc = data["current_condition"][0]["weatherDesc"][0]["value"]

        print(f"\nWeather in {city}:")
        print(f"Temperature: {temp}°C")
        print(f"Condition: {weather_desc}\n")

    else:
        print("Error: Could not retrieve weather data.")

# Main loop to check multiple cities
condition = True

while condition:

    city = input("Enter a city name (or type 'exit' to quit): ")

    if city.lower() == "exit":

        print("Goodbye!")

        condition = False

    get_weather(city)
```

Ej. cryptoTracking

```python
import requests

def get_crypto_price(crypto, currency):

    url = f"https://api.coingecko.com/api/v3/simple/price?ids={crypto}&vs_currencies={currency}"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        if crypto in data:
            price = data[crypto][currency]
            print(f"\n{crypto.capitalize()} Price: {price} {currency.upper()}\n")
        else:
            print("Error: Cryptocurrency not found.")
    else:
        print("Error: Could not retrieve crypto data.")

# Main loop for multiple searches
while True:

    currency = input("Enter a cryptocurrency (e.g., eur, usd) or type 'exit' to quit: ").lower()
    crypto = input("Enter a cryptocurrency (e.g., bitcoin, ethereum) or type 'exit' to quit: ").lower()
    if crypto == "exit":
        print("Goodbye!")
        break
    get_crypto_price(crypto, currency)
```

Ej. getLandPhoto

```python
import requests
from datetime import datetime
import os

# NASA API Endpoint & Key
NASA_API_URL = "https://api.nasa.gov/planetary/earth/imagery"
NASA_ASSETS_URL = "https://api.nasa.gov/planetary/earth/assets"
API_KEY = "                                        "  # Personal key

def checkImageAvailable(lat, lon):
    """Check if NASA has an image for the given location."""
    params = {"lat": lat, "lon": lon, "api_key": API_KEY}
    response = requests.get(NASA_ASSETS_URL, params=params)

    if response.status_code == 200:
        data = response.json()
        if "date" in data:
            return data["date"][:10]  # Extract only "YYYY-MM-DD"
    return None
```

Ej. getLandPhoto

```python
def getLand(date, lat, lon, dim):
    """Fetches the satellite image if available."""

    params = {"api_key": API_KEY, "lat": lat, "lon": lon, "dim": dim, "date": date}
    response = requests.get(NASA_API_URL, params=params, stream=True)

    print(f"\nRequesting Image for {lat}, {lon} on {date}")
    print(f"Full API URL: {response.url}")  # Print the full request URL
    print(f"Response Status Code

    if response.status_code == 2
        content_type = response.headers.get("Content-Type", "")

        if "image" in content_type:  # Check if response is an image

            filename = f"land_{lat}_{lon}.jpg"

            with open(filename, "wb") as file:

                for chunk in response.iter_content(1024):
                    file.write(chunk)

            print(f"Image saved: {filename}")

            # Open the image automatically
            os.system(f"xdg-open {filename}")
        else:
            print("API did not return an image.")
            print(response.text)  # Print actual text response
    else:
        print(f"Error: NASA API request failed with status code {response.status_code}")
```

```
(variable) headers: CaseInsensitiveDict[str]
See Real World Examples From GitHub
```

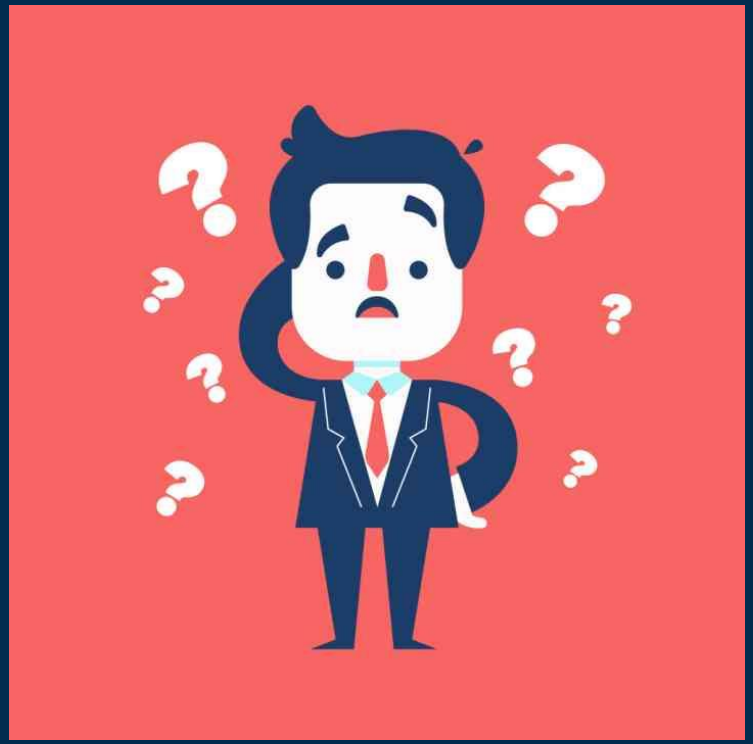
Ej. getLandPhoto

```python
# Get user input
date = input("Enter a date (YYYY-MM-DD) or press 'ENTER' for today: ")
lat = float(input("Enter the latitude (Y): "))
lon = float(input("Enter the longitude (X): "))

# Check if NASA has an image for this location
available_date = checkImageAvailable(lat, lon)

if available_date:
    print(f"Using available image from: {available_date}")
    getLand(available_date, lat, lon, dim=0.1)
else:
    print("No recent satellite images found for this location.")
```

Preguntas/aportaciones



Summer School 2025

Muchas gracias

Descubre comunidad, descubre 0xDecode

0xDecode

Deusto Electronic Club Of
Developers & Engineers