Tema 1: Decorators en Python

Definición

Un decorator en Python es una función que recibe otra función como argumento, la envuelve y devuelve una n

Cómo funcionan

Cuando se ejecuta el módulo, Python aplica decorators evaluando la función original y pasando el objeto funció

Casos de uso comunes

• Autenticación y autorización en aplicaciones web.
• Registro de entradas y salidas (logging).
• Medición de tiempo de ejecución de funciones.
• Validación de parámetros antes de la ejecución.

Ejemplo detallado

```python
def temporizador(func):
    import time
    def wrapper(*args, **kwargs):
        inicio = time.time()
        resultado = func(*args, **kwargs)
        fin = time.time()
        print(f"{func.__name__} tardó {fin - inicio:.4f} segundos.")
        return resultado
    return wrapper

@temporizador
def calcular_suma(n):
    return sum(range(n))

calcular_suma(1000000)
```

Patrones avanzados

• Decorators con parámetros: funciones que devuelven decorators.
• Uso de functools.wraps para preservar metadatos de la función original.
• Composición de múltiples decorators.

Tema 2: Concurrencia en JavaScript

Modelo de un solo hilo y event loop

JavaScript en entornos como navegadores o Node.js corre en un solo hilo. El bucle de eventos (event loop) ge

Callbacks

Las funciones de callback se pasan como argumentos a operaciones asíncronas: lectura de ficheros, peticione

Promises

Una Promise representa la eventual finalización o fallo de una operación asíncrona.

```javascript
function retraso(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
retraso(1000)
  .then(() => console.log('1 segundo después'))
  .catch(err => console.error(err));
```

async/await

Azúcar sintáctico sobre Promises que facilita la lectura y manejo de errores.

```javascript
async function tarea() {
  try {
    console.log('Inicio');
    await retraso(1000);
    console.log('Fin');
  } catch (err) {
    console.error(err);
  }
}
tarea();
```

Microtasks vs Macrotasks

• Microtasks: Promises (cola de microtareas) permiten ejecutar callbacks antes de renderizar.
• Macrotasks: setTimeout, I/O, eventos del DOM (cola de tareas).

Buenas prácticas

• Evitar callbacks anidados; usar async/await.
• Manejar siempre errores con try/catch o .catch().
• No bloquear el hilo principal con operaciones pesadas.
• Para cálculos intensivos, usar Web Workers en el navegador.

Tema 3: Introducción a GraphQL

¿Qué es GraphQL?

GraphQL es un lenguaje de consultas para APIs desarrollado por Facebook en 2015. Permite al cliente especif

Esquema y tipos

El servidor define un esquema fuertemente tipado que especifica consultas (Query), mutaciones (Mutation) y ti

```graphql
type Query {
  libro(id: ID!): Libro
}
type Libro {
  id: ID!
  titulo: String
  autor: Autor
}
type Autor {
  id: ID!
  nombre: String
}
```

Consultas y Mutaciones

Consulta de ejemplo:

```graphql
{
  libro(id: "1") {
    titulo
    autor {
      nombre
    }
  }
}
```

Mutación de ejemplo:

```graphql
mutation {
  crearAutor(nombre: "Ana") {
    id
    nombre
  }
}
```

Ventajas y casos de uso

- Reduce overfetching y underfetching de datos.
- Ideal para apps móviles y SPAs donde la eficiencia de red es crítica.
- Facilita la evolución de APIs sin múltiples versiones.

Herramientas populares

- Apollo Server y Client.
- GraphQL Yoga.
- Relay (Facebook).