IIC2413

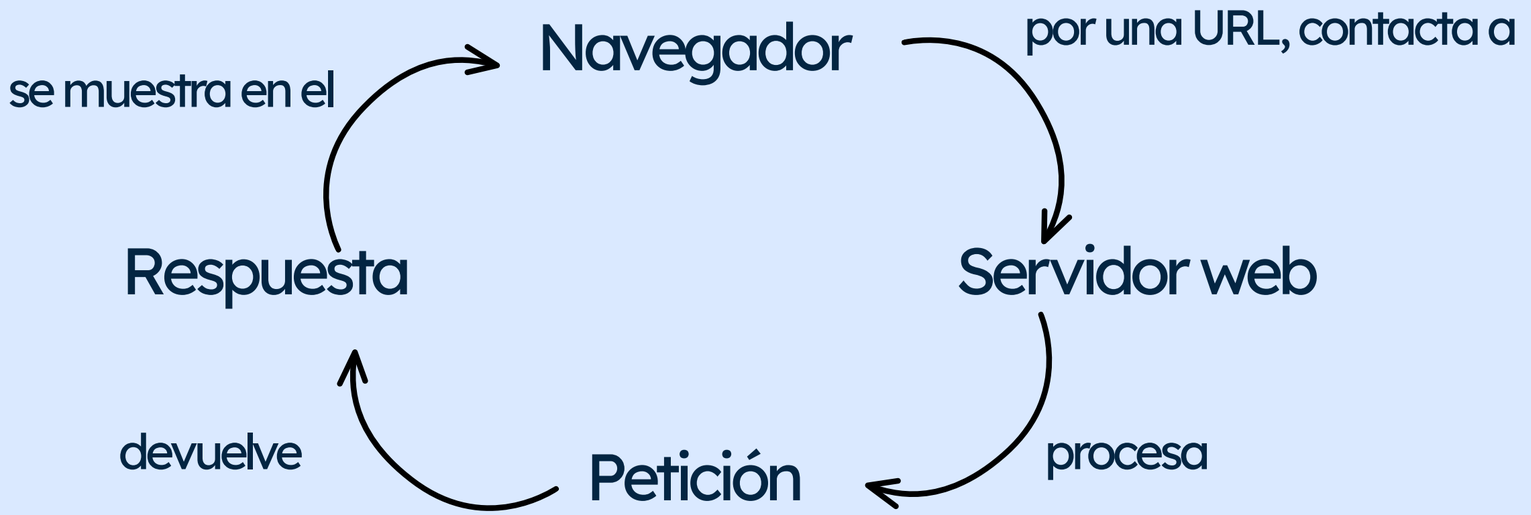
AYUDANTÍA 10

Web y NoSQL

Cliente - Servidor

¿Qué es un servidor web?

Es un programa que recibe peticiones HTTP desde un navegador y devuelve una respuesta (puede ser HTML, una imagen, un JSON, el resultado de un programa).



Métodos HTTP

GET

Los datos van en la URL (son visibles)
Es para leer información, no modificarla

Un ejemplo de URL sería:
https://ejemplo.com/ejemplo.php?persona=Pepito&edad=27

En PHP, se pueden acceder:
$_GET[‘persona’] → Pepito
$_GET[edad] → 27

POST

- Los datos van ocultos en el cuerpo de la petición (no se ven en la URL)
- Se usa para enviar o modificar datos (por ejemplo, al querer iniciar sesión o querer registrar algo nuevo)

En PHP, se pueden acceder:
$_POST[‘usuario’]
$_POST[‘contraseña’]

HTML y CSS

¿Qué son?

- HTML define la estructura de la página: títulos, párrafos, imágenes, formularios, etc.
- Las etiquetas se escriben como: <tag>contenido</tag>
- Por otro lado, CSS le da estilo (colores, tamaños, etc).

En el código de ejemplo se muestra una estructura básica de un archivo HTML.

```html
<!--Archivo index.html-->
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Página random - Iniciar sesión</title>
    <link rel="stylesheet" href="stylesheets.css">
</head>
<body>
    <div class="container">
        <h1>Bienvenido a mi página</h1>

        <form action="validar_login.php" method="POST" class="formulario">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required>

            <label for="clave">Contraseña:</label>
            <input type="password" id="clave" name="clave" required>

            <button type="submit">Iniciar sesión</button>
        </form>

        <p>¿No tienes cuenta? <a href="registro.php">Regístrate aquí</a></p>
    </div>
</body>
</html>
```

Conexión a BDD en PHP

¿Cómo se conecta?

- PDO (PHP Data Objects) es una librería que permite a PHP hablar con la base de datos. Le mandas un query, lo ejecuta en PostgreSQL, y te devuelve los resultados como un array de PHP.
- Se crea un objeto PDO pasándole host, dbname, usuario y clave.
- Se usa try/catch para manejar errores de conexión.
- Esta función se reutiliza en todos los archivos PHP del proyecto.

```php
<?php
0 references
function conectar_db() {
    $host = 'localhost'; // Cambiar si se quiere usar el servidor remoto
    $dbname = nombre_bdd; // Nombre de base de datos
    $user = usuario; // Nombre de usuario
    $password = contraseña; // Número de alumno

    try {
        $db = new PDO("pgsql:host=$host;dbname=$dbname", $user, $password);
        $db -> setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $db;
    } catch (PDOException $e) {
        echo "Error de conexión: " . $e->getMessage();
        exit();
    }
}
?>
```

Queries seguras

- Es importante que las queries y transacciones que hagan sean seguras (es decir, que eviten “SQL injection”).
- Para eso utilizamos prepare (como dice el nombre, “´prepara” la query) y bindParam (une los parámetros pedidos con los del input)

```php
<?php
include 'conectar_bdd.php';
$db = conectar_db();

$email = $_POST['email'];
$clave = $_POST['clave'];

// 1. Preparar el query con placeholders
$query = "SELECT id_persona, nombre_completo FROM usuario
          WHERE email = :email AND clave = :clave";
$stmt = $db->prepare($query);

// 2. Vincular los valores reales (PDO los sanitiza)
$stmt->bindParam(':email', $email);
$stmt->bindParam(':clave', $clave);

// 3. Ejecutar
$stmt->execute();

// 4. Obtener resultados
$usuario = $stmt->fetch(PDO::FETCH_ASSOC);

if ($usuario) {
    echo "Bienvenido " . $usuario['nombre_completo'];
} else {
    echo "Credenciales inválidas";
}
?>
```

Queries seguras

- Para las transacciones hacemos lo mismo en su respectivo flujo:
- Se usa beginTransaction() y endTransaction()
- Usamos prepare y bindParam
- Se usan bloques try y catch para el manejo de errores
- De acuerdo a estos bloques, la transacción se confirma (commit()) o se deshace (rollBack()).

```php
<?php
// Ejemplo: Insertar un usuario
include 'conectar_bdd.php';
$db = conectar_db();

try {
    // 1. Inicia transacción
    $db->beginTransaction();

    // 2. Insertar al la tabla Persona
    $query = "INSERT INTO Persona (run, nombre, email)
    VALUES (:run, :nombre, :email)";
    $stmt = $db->prepare($query);
    $stmt->bindParam(':run', $run);
    $stmt->bindParam(':nombre', $nombre);
    $stmt->bindParam(':email', $email);
    $stmt->execute();

    // 3. Si todo está bien, se confirma
    $db->commit();
    echo "Persona registrada correctamente";

} catch (PDOException $e) {
    // Si falla, deshacer
    $db->rollBack();
    echo "Error: " . $e->getMessage();
}
?>
```

NoSQL

NoSQL vs SQL

SQL

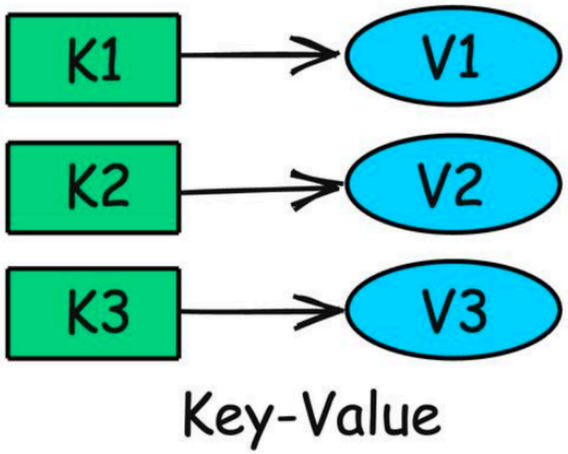
- Datos muy estructurados
- Muchos cruces (joins) frecuentes
- Necesito ACID garantizado
- Datos cambian constantemente

NoSQL

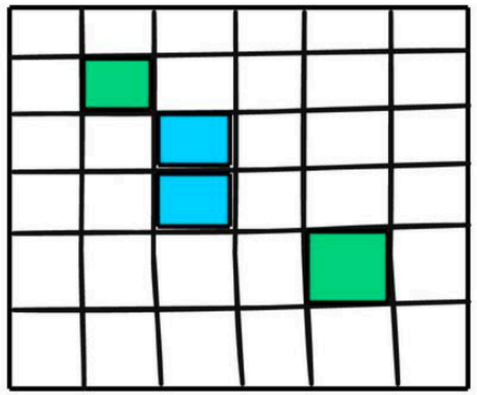
- Datos semi/no estructurados
- Alta escala y distribución
- Despliegue de información estática
- Consistencia eventual es aceptable

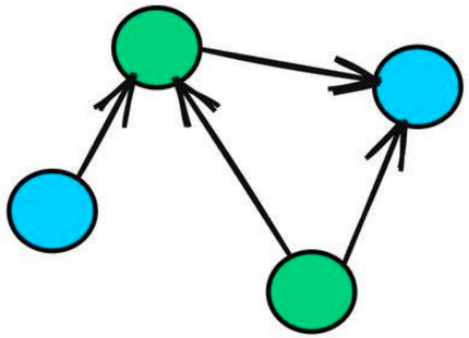
Tipos

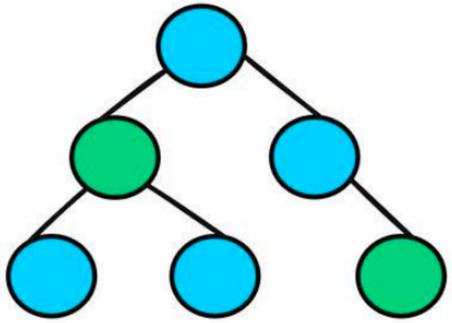
- Key-Value: operaciones put/get/delete, gran volumen, simple. Ejemplos: Redis, DynamoDB
- Columnar: alto volumen, complejidad media. Ejemplo: Cassandra
- Documentos: JSON, colecciones, flexibilidad. Ejemplos: MongoDB, CouchDB
- Grafos: relaciones complejas, menor volumen. Ejemplos: Neo4j, Virtuoso



Key-Value


Column Store


Graph


Document

MongoDB - Conceptos

- Documento: unidad básica, es un JSON con un _id ObjectId generado automáticamente
- Colección: agrupación de documentos similares (≈ tabla)
- Base de datos: contiene colecciones relacionadas
- Un servidor puede tener varias bases de datos
- Esquema flexible: dos documentos en la misma colección pueden tener campos distintos

Colección

Usuarios	
Key	JSON
60bfd90e002ce228636e506b	
60bfd90e002ce228636e5215	...

Base de datos

Usuarios	
Key	JSON
...	...

Mensajes	
Key	JSON
...	...

Likes	
Key	JSON
...	...

MongoDB - Sintáxis

- find() permite consultar documentos de una colección
- Filtros y proyecciones son análogos a WHERE y SELECT
- Operadores de comparación: $gte, $gt, $lt, $lte, $ne
- createIndex() crea índices para optimizar consultas
- Sintaxis se basa en documentos JSON

```
// Ver todo
db.usuarios.find()

// WHERE
db.estudiantes.find({"carrera":"Ingeniería"})

// WHERE edad >= 22
db.estudiantes.find({"edad": {$gte:22}})

// SELECT nombre, edad
db.estudiantes.find({}, {"nombre":1, "edad":1, "_id":0})

// Crear índice
db.estudiantes.createIndex({"carrera":1})

// Búsqueda de texto
db.ramos.find({$text: {$search:"Bases de datos"}})
```

Ejercicios