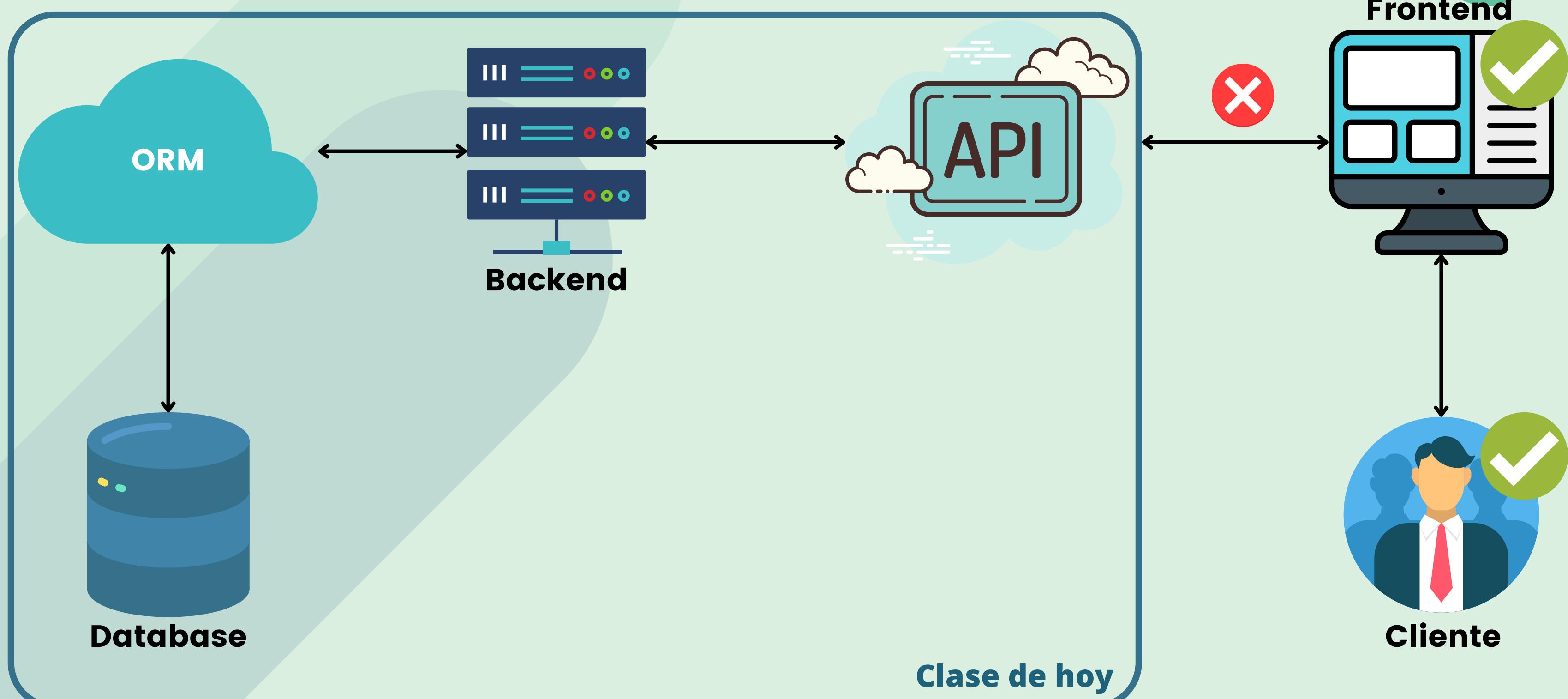


Arquitectura de servidores Backend

Aplicaciones Interactivas - 2023

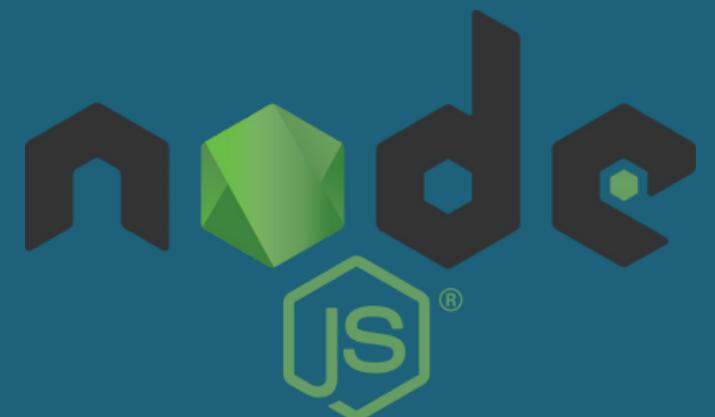
Ing. Francisco Fares

ARQUITECTURA BÁSICA DE UNA WEB

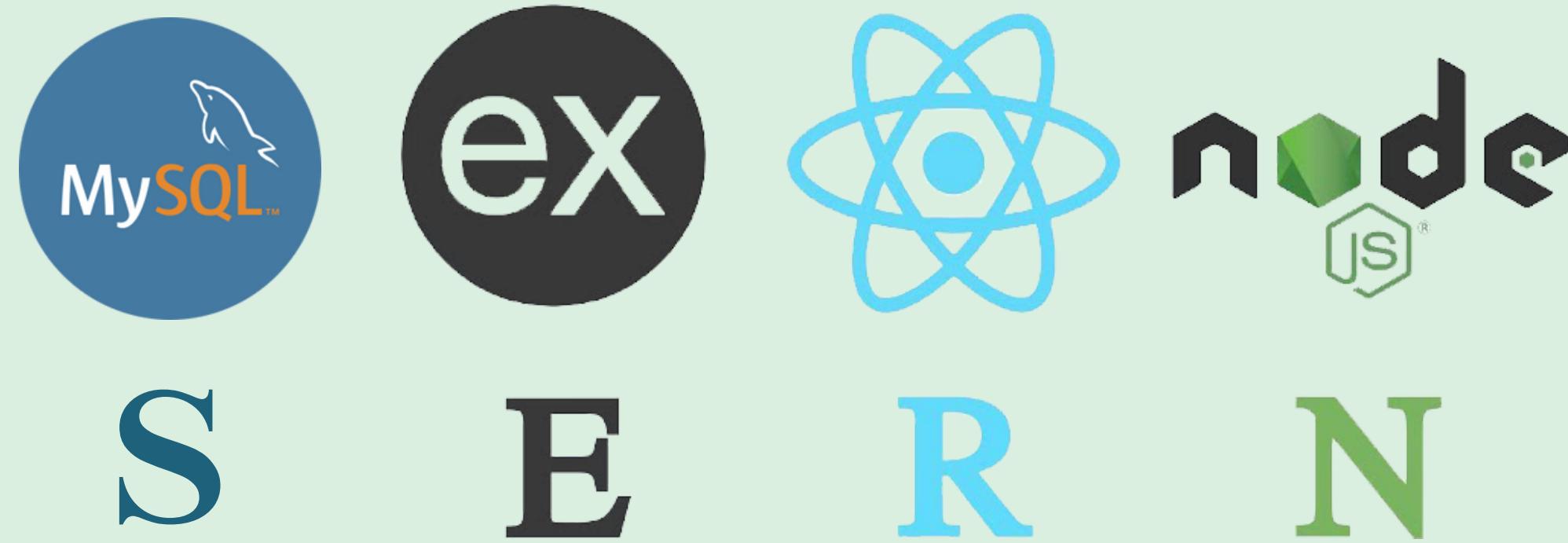
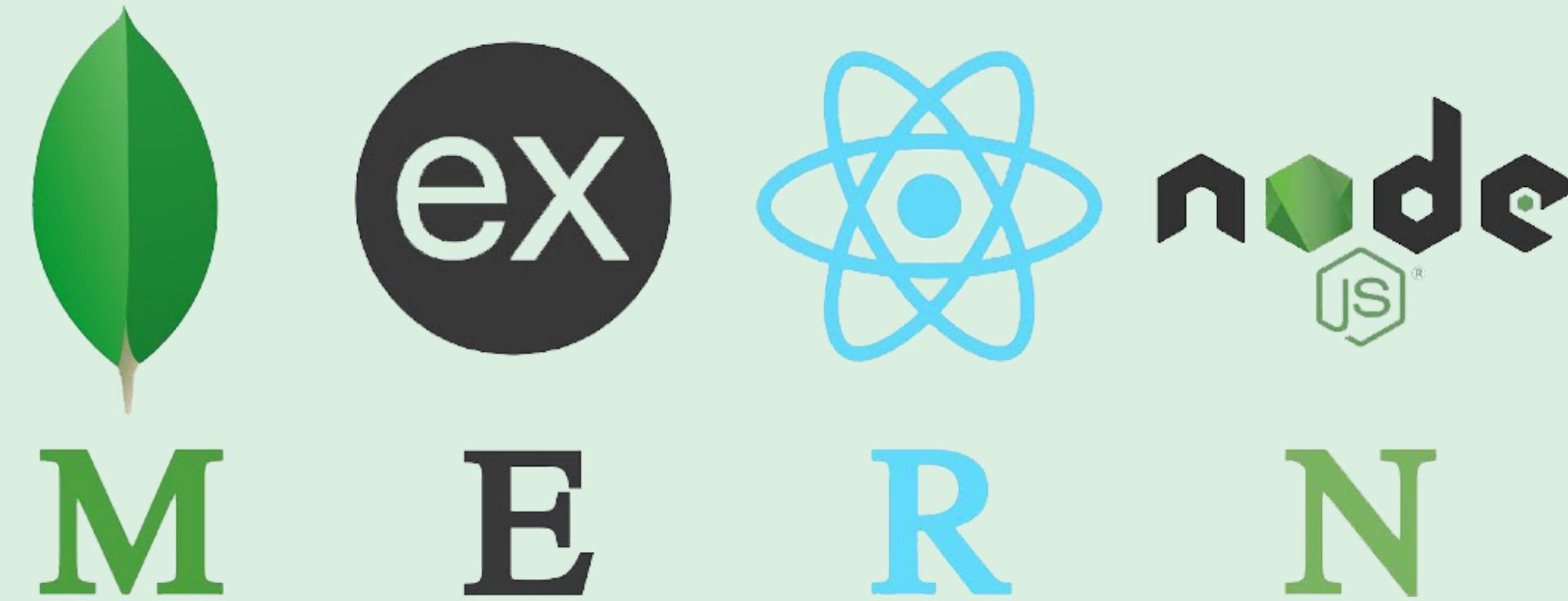


Backend

- Área lógica de la aplicación.
- Es lo que el cliente no ve.
- Se encarga de la funcionalidad de la aplicación y de la comunicación con la base de datos.
- Lenguajes más utilizados:
 - Java
 - Node.js
 - Python



STACKS



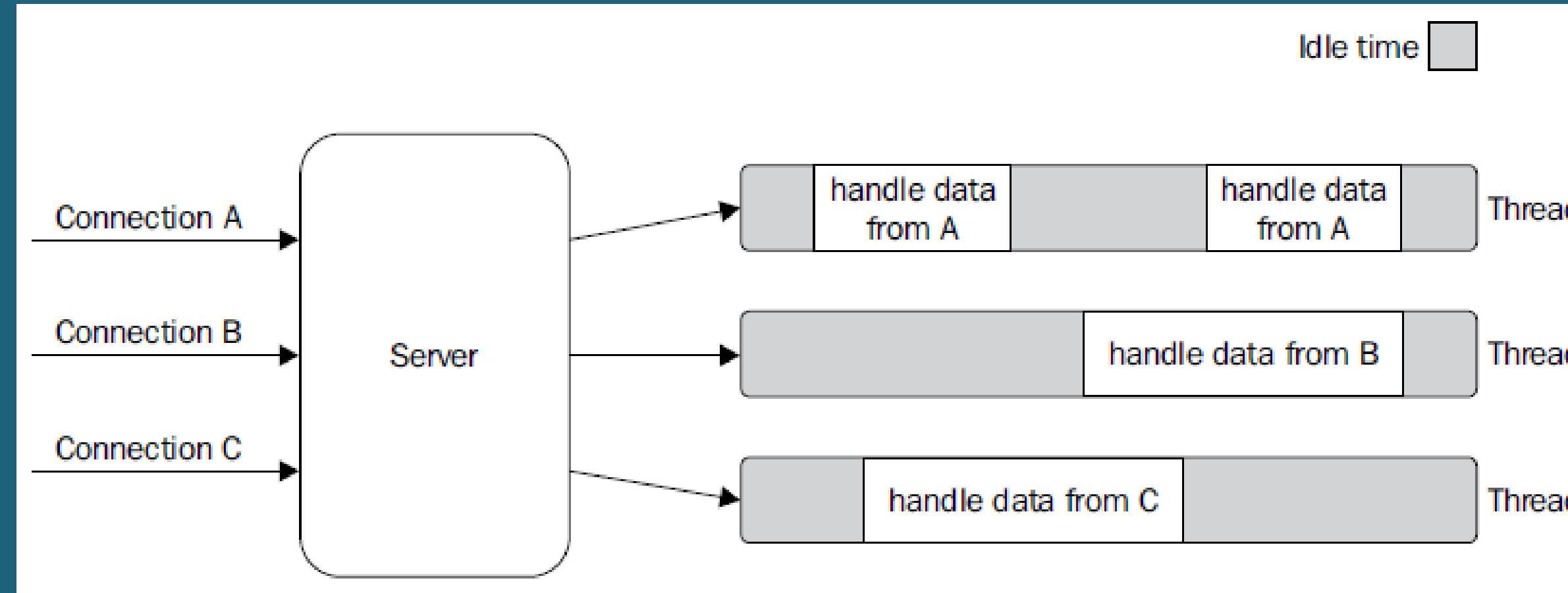
NodeJS

- Node es un entorno de ejecución de Javascript minimalista.
Permite correr código JS fuera del navegador.
- Utiliza el motor de V8 de Chrome que reduce el tiempo de compilación.
- Trabaja con un único hilo de ejecución (single-threaded).

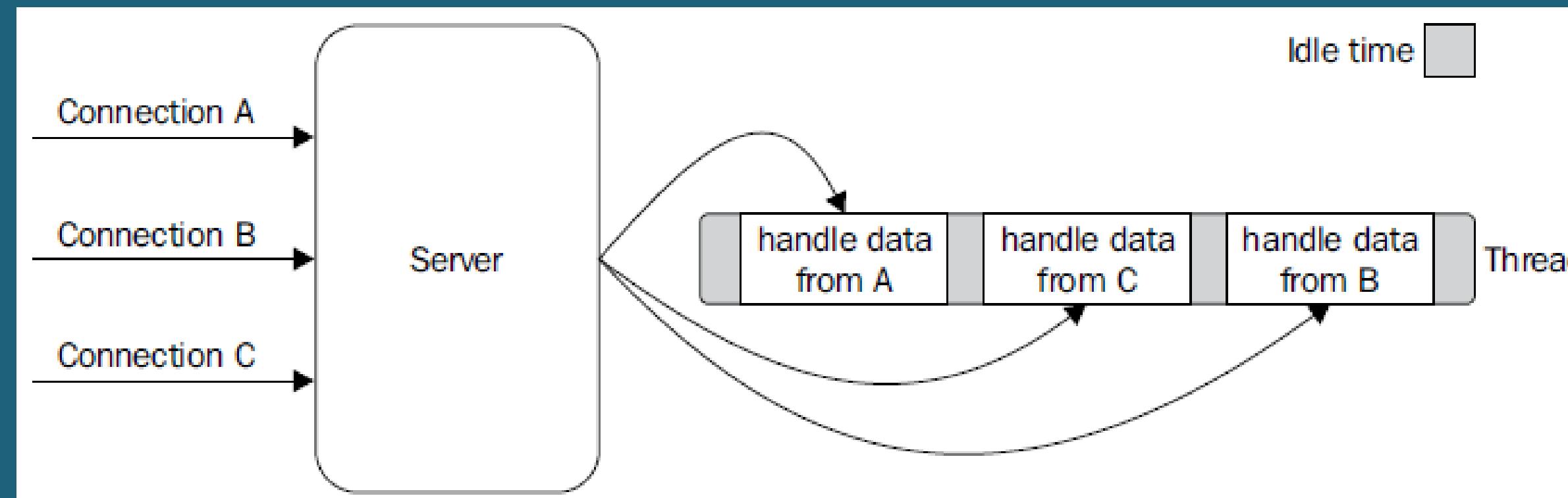


Single Threaded!

Java,
C#, etc



Node.js



Tareas bloqueantes vs no bloqueantes

Bloqueantes:
como su nombre lo indica,
bloquean la ejecución de nuevas
tareas hasta que termine la
tarea actual (Síncrono).

No Bloqueantes:
permite la ejecución de tareas
posteriores sin siquiera haber
terminado la tarea actual
(Asíncrono).

```
10 |         throw new Error(`Error in getUsers Service`);  
11 |     }  
12 | }
```

async/await

```
14 async getUserId(id) {  
15     try {  
16         let user = await UsuariosModel.findOne({ _id: id });  
17         return user;  
18     } catch (err) {  
19         throw new Error("Error in getUserId Service");  
20     }  
21 }  
  
23 async getUserByEmail(email) {  
24     try {  
25         let user = await UsuariosModel.findOne({ email });  
26         return user;
```

Express.js

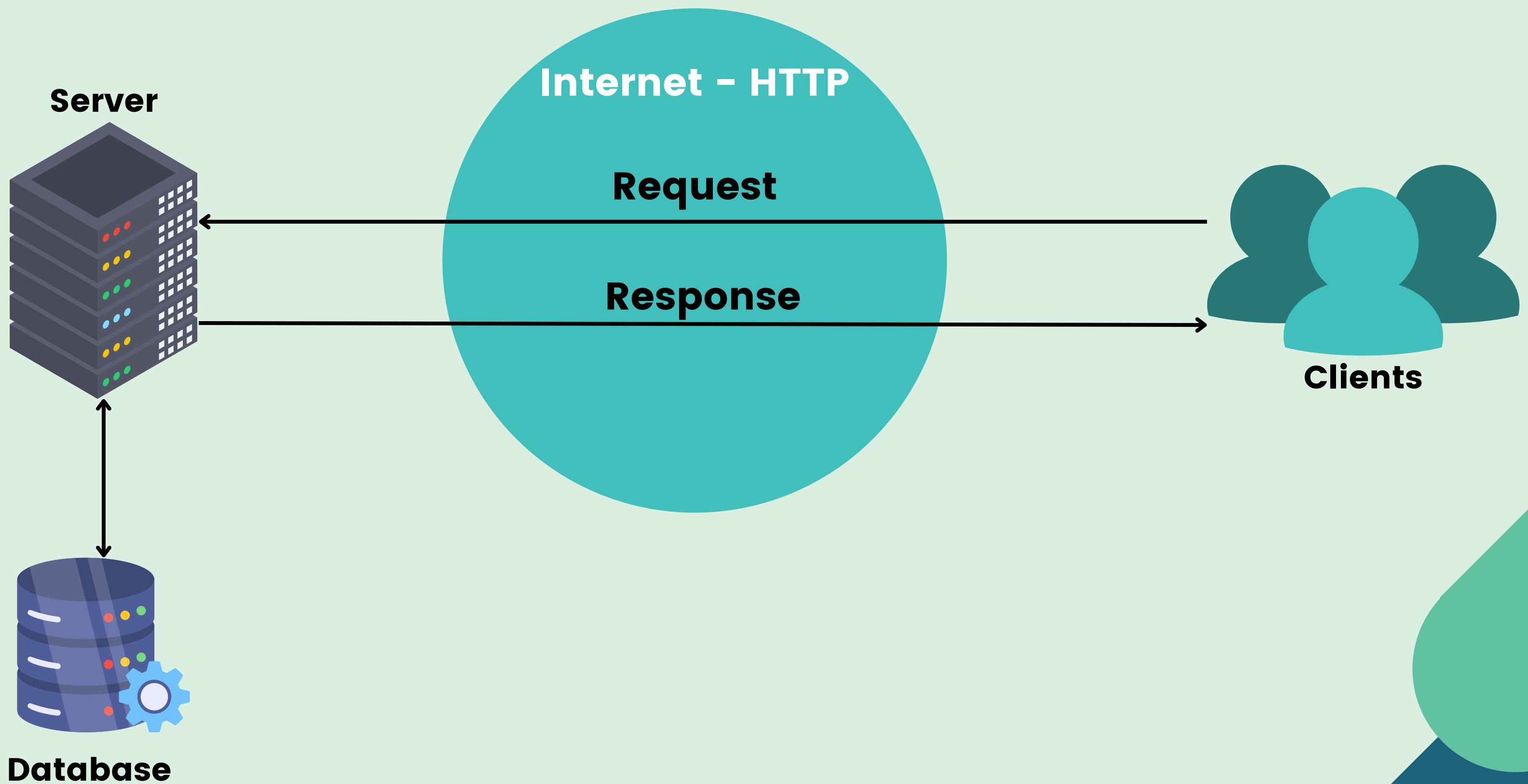
- Express es un framework de backend para Node.
- Facilita la creación de servidores backend y API' s.
- 3 líneas de código y tenemos un servidor backend corriendo!



Práctica: servidor básico con node + express + nodemon

```
J5 server.js > ...
1  const express = require('express');
2
3  const app = express();
4  const PORT = 8080;
5
6  app.get('/', (req,res) => {
7      res.send('Welcome to the server!');
8  });
9
10 app.listen(PORT, () => {
11     console.log('Server running on port: ',PORT);
12 });
13
```

Modelo Cliente-Servidor



TIPOS DE MÉTODOS EN UNA REQUEST

C



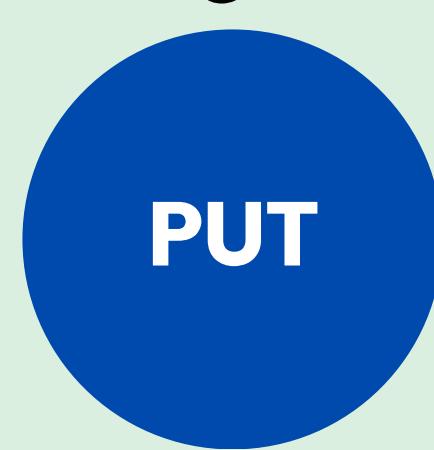
Generalmente utilizado para crear registros en la base de datos, autenticación y conexión con servicios externos.

R



Solicita la obtención de uno o muchos registros de la base de datos o de servicios externos.

U



Solicitado para actualizar valores de registros en la base de datos.

D



Utilizado para eliminar registros de la base de datos.

ESTRUCTURA DE UNA REQUEST

Método	Parámetros
PUT /API/CLIENTES/:1 HTTP/1.1	URL
HOST: LOCALHOST:8080	
X-ACCESS-TOKEN: {{AUTH}}	Headers
CONTENT-TYPE: APPLICATION/JSON	
CONTENT-LENGTH: 209	
{	
"NOMBRE": "FRANCISCO",	
"APELLIDO": "FARES",	
"USUARIO": "FFARES"	
}	

PETICIÓN A:
HTTP://LOCALHOST:8080/API/CLIENTES/1

ESTRUCTURA DE UNA RESPONSE

Status code
HTTP/1.1 200 OK

DATE: SAT, 09 OCT 2010 14:28:02 GMT

SERVER: APACHE

LAST-MODIFIED: TUE, 01 DEC 2009
20:18:22 GMT

ACCEPT-RANGES: BYTES

CONTENT-LENGTH: 29769

CONTENT-TYPE: APPLICATION/JSON

{

"NOMBRE": "FRANCISCO",
"APELLIDO": "FARES",
"USUARIO": "FFARES",
"MAIL": "FFARES@UADE.EDU.AR"

}

Headers

Body

CÓDIGOS DE ESTADO MÁS COMUNES

2xx Success	
200	OK
201	Created
202	Accepted
204	No Content

4xx Client Error	
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found

5xx Server Error	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable





API (APPLICATION PROGRAMMING INTERFACE)

A large, rounded rectangular button with a blue gradient background. The letters "API" are written in white, bold, sans-serif font, centered on the button.

API

- Canal de comunicación entre dos o más aplicaciones.
- Publica URLs del lado del Backend (llamados endpoints), mediante las cuales el Frontend se comunica.
- En las aplicaciones web, las API suelen seguir la arquitectura REST.

API REST

- No dispone de una interfaz gráfica.
- Utiliza HTTP como protocolo de comunicación entre sistemas.
- Posee una arquitectura cliente-servidor sin estado.
- Ofrece una interfaz uniforme a través de URIs.

Ejemplo de URI: <http://localhost:8080/api/usuarios>



Práctica: router con express

```
const {Router} = require('express');
const usuariosController = require('../controllers/usuarios.controller');
const router = Router();

router.get('/',usuariosController.getUsuarios); //GET USUARIOS
router.post('/',usuariosController.createUsuario); //POST USUARIOS
router.get('/:id',usuariosController.getUsuarioById); //GET USUARIOS BY ID

module.exports = router;
```



mongoose

ORM (OBJECT RELATIONAL MAPPING)

- Permite generar modelos que mapean estructuras de una base de datos sobre una estructura lógica de entidades.
- Busca acelerar y facilitar el desarrollo de aplicaciones
- Existen ORMs para bases de datos relacionales y no relacionales.

Práctica: conexión a base de datos mongo

```
src > db > JS config.js > ...
1  const mongoose = require('mongoose');
2  require('dotenv').config();
3
4  const dbConnection = async () =>{
5      try{
6          await mongoose.connect(process.env.CONNECTION_STRING)
7          console.log('DB online!');
8      }
9      catch(err){
10         console.error(err);
11         throw new Error('Error en la conexión de la BD');
12     }
13 }
14
15 module.exports = {dbConnection};
```

```
JS server.js > ...
1  require('dotenv').config();
2  const express = require('express');
3  const {dbConnection} = require('../src/db/config');
4
5
6  const app = express();
7  dbConnection();
```

Mongo Atlas

 Reporting

VERSION 6.0.6 REGION AWS Sao Paulo (sa-east-1)

Overview Real Time Metrics Collections Search Profiler Performance Advisor Online Archive Cmd Line Tools

DATABASES: 1 COLLECTIONS: 5 REFRESH

+ Create Database SEARCH NAMESPACES

test.historicoEmociones

STORAGE SIZE: 148KB LOGICAL DATA SIZE: 873.59KB TOTAL DOCUMENTS: 1802 INDEXES TOTAL SIZE: 76KB

Find Indexes Schema Anti-Patterns 1 Aggregation Search Indexes Charts •

INSERT DOCUMENT

Filter • Type a query: { field: 'value' } Reset Apply More Options ▾

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('63499921ba2577bfa028c534')
▶ prenda: Object
▶ centroComercial: Object
emocion: "happy"
fecha: 2022-10-14T17:15:13.532+00:00
__v: 0
genero: "mujer"
```

ESCTRUCTURA DEL SERVIDOR EN CAPAS

01 - API Route

02 - Controller

03 - Service

04 - ORM

05- Database

A practicar! Mostrar el
proyecto de ejemplo y.
probarlo con Postman.



Fin de la clase!