S.E.P. TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO de Tuxtepec

Programación Cliente-Servidor

Reporte de Practica

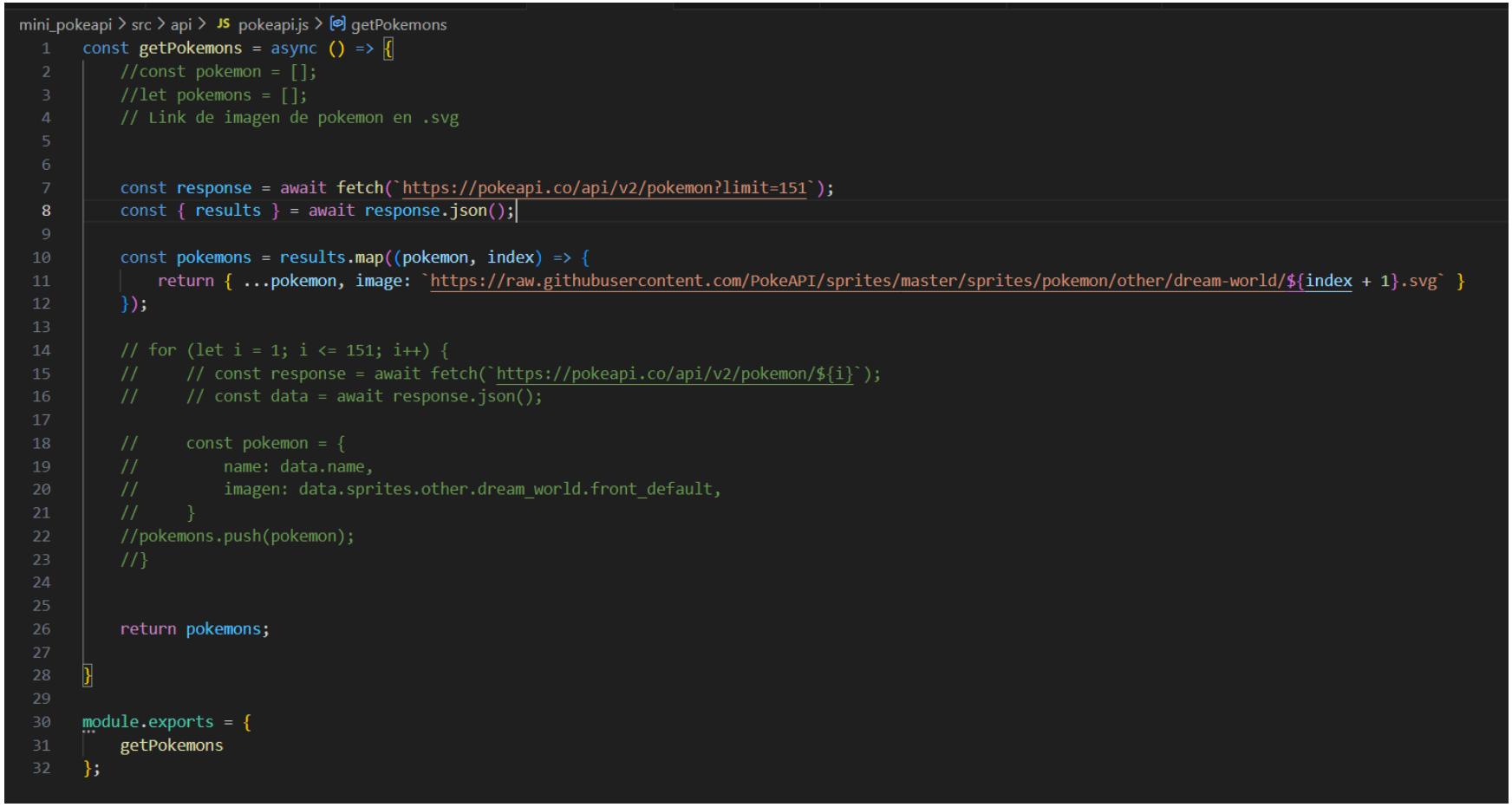
PRESENTAN:
Juan Antonio Cobos Hilario

DOCENTE:

Julio Aguilar Carmona

CARRERA:
INGENIERIA INFORMÁTICA

16-diciembre-2025



ENCAPSULA LA COMUNICACIÓN CON UN api EXTERNA DE POKEMON

mini_pokeapi > src > config > JS dbconection.js > ...

```
1   const mariadb = require('mariadb');
2   const env = require(`./environment`);
3
4   const config = {
5       host: '127.0.0.1',
6       user: env.dbUser,
7       password: env.dbPassword,
8       database: env.dbName,
9       port: env.dbPort,
10      connectionLimit: 10,
11  }
12
13  const pool = mariadb.createPool(config);
14
15  module.exports = pool;
```

RESPONSABLE DE LA CONEXIÓN A LA BASE DE DATOS MySQL

Después nos dirigimos a la carpeta controllers y se despliega el game.js, pokemon.js y el user.js

Game.js el controlador de partida

Pokemon.js el controlador de pokemon

mini_pokeapi > src > controllers > JS pokemon.js > ...

```
const { request, response } = require('express');
const pokeapi = require('../api/pokeapi');
const pool = require('../config/dbconection');
const { pokemonQuery } = require('../models/pokemon');

const pokemonSeeder = async (req = request, res = response) => {
    const pokemons = await pokeapi.getPokemons();

    let conn;

    try {
        conn = await pool.getConnection();
        await conn.query('SET FOREIGN_KEY_CHECKS = 0');
        await conn.query('TRUNCATE TABLE pokemons');
        await conn.query('SET FOREIGN_KEY_CHECKS = 1');

        pokemons.forEach(async (pokemon) => {
            await conn.query(pokemonQuery.add, [pokemon.name, pokemon.image]);
        });
        res.send("Pokemones agregados en la Base de Datos");
    } catch (err) {
        return res.status(500).send(err);
    } finally {
        if (conn) {
            conn.end();
        }
    }

}

const randomPokemon = async (req = request, res = response) => {
    let conn;
    try {
        conn = await pool.getConnection();
        const pokemons = await conn.query(pokemonQuery.random);
        if (pokemons.length === 0) {
            return res.status(500).send("No hay pokemones en la base de datos");
        }
        res.send(pokemons);

    } catch (err) {
        return res.status(500).send(err);
    } finally {
        if (conn) {
            conn.end();
        }
    }
}

const idPokemon = async (req = request, res = response) => {
    let conn;
    try {
        const { id } = req.params
        if (isNaN(Number(id))) {
            res.status(400).send("Esto no es un numero");
            return;
        }

        conn = await pool.getConnection();
        const pokeid = await conn.query(pokemonQuery.view, [id]);
        if (!pokeid) {
            res.status(404).send("Pokemon no encontrado");
            return;
        }
        res.send(pokeid);
    } catch (err) {
        return res.status(500).send(err);
    } finally {
        if (conn) {
            conn.end();
        }
    }
}
```

User.js controlador de usuarios

mini_pokeapi > src > controllers > JS user.js > ...

```
1   // Acciones que hacen los endpoints
2
3   const {request , response } = require('express');
4   const userQueries = require('../models/user');
5   const pool = require('../config/dbconection');
6   const bcrypt = require('bcryptjs');
7
8   const saltRounds = 10;
9
10  const showUsers =  async (req= request, res =response) => {
11      let conn;
12      try {
13          //se pone lo que se quiere hacer
14          conn = await pool.getConnection();
15          const users = await conn.query(userQueries.users.show);
16          res.send(users);
17      } catch (err) {
18          res.status(500).send("error");
19      } finally {
20          if (conn) conn.end();
21      }
22  }
23
24  const viewUser = async (req= request, res =response) => {
25      let conn;
26      try {
27          const {id} = req.params
28          if (isNaN(Number(id))) {
29              res.status(400).send("Esto no es un numero");
30              return;
31          }
32          conn = await pool.getConnection();
33          const user = await conn.query(userQueries.users.view, [id]);
34          if (!user) {
35              res.status(404).send("Usuario no encontrado");
36              return;
```

mini_pokeapi > src > controllers > JS user.js > ...

```
24  const viewUser = async (req= request, res =response) => {
39          res.send(user);
40      } catch (err) {
41          res.status(500).send("error");
42      } finally {
43          if (conn) conn.end(viewUser);
44      }
45
46  }
47
48  const createUser = async (req = request, res = response) => {
49      const {name, lastname, email, password} = req.body;
50      if (!name || !lastname || !email || !password){
51          res.status(400).send("Faltan datos");
52          return;
53      }
54      //if(require('../models/user').users.find((u)=> u.email === email)){
55      ///    res.status(400).send(`Este usuario con correo ${email} ya existe`);
56      //    return;
57      //}
58      if (password.length < 6){
59          res.status(400).send("La contraseña debe tener al menos 6 caracteres");
60          return;
61      }
62      //require('../models/user').users.push({
63      //    id: require('../models/user').users.length + 1,
64      //    name,
65      //    lastname,
66      //    email,
67      //    password
68      //});
69      //res.send({msg: "Usuario creado ", user : {name, lastname, email, password}});
70      //res.send(req.body)
71      let conn;
72      try {
73          conn = await pool.getConnection();
74          const userExist = await conn.query( userQueries.users.verifyEmail, [email]);
```

⊗ 0 ⚠ 0 Connect

Carpeta de models encontramos lo que representa el juego

mini_pokeapi > src > models > JS game.js > ...

```
1   const gameQuery = {
2       getGame: 'SELECT * FROM games WHERE user_id = ?',
3       addGame: 'INSERT INTO games (user_id, win, lose) VALUES (?, ?, ?)',
4       updateGame: 'UPDATE games SET win = ?, lose = ? WHERE user_id = ?'
5   }
6
7   module.exports = {
8       gameQuery
9   };
10
```

mini_pokeapi > src > models > JS pokemon.js > ...

```
1   const pokemonQuery = {
2       add: 'INSERT INTO pokemons (name, image) VALUES (?, ?)',
3       random: 'SELECT * FROM pokemons ORDER BY RAND() LIMIT 4',
4       view: 'SELECT * FROM pokemons WHERE id = ?'
5   }
6
7
8   module.exports = {
9       pokemonQuery
10  };
11
```

mini_pokeapi > src > models > JS user.js > ...

```
1   // Se escriben las interacciónes con la BD
2
3   const users = {
4       show: "SELECT * FROM users",
5       view: "SELECT * FROM users WHERE id = ?",
6       verifyEmail: "SELECT * FROM users WHERE email = ?",
7       create : "INSERT INTO users (name, lastname, email, password) VALUES (?, ?, ?, ?)",
8       delete: "DELETE FROM users WHERE id = ?",
9       update: "UPDATE users SET name = ?, lastname = ?, email = ?, password = ? WHERE id = ?",
10      viewByEmail: "SELECT * FROM users WHERE email = ?"
11  }
12
13
14  module.exports = {
15      users
16  };
17
```

ROUTES define las rutas http del api

mini_pokeapi > src > routes > JS game.js > ...

```
1   const { Router } = require('express');
2   const router = Router();
3   // Controladores
4
5   router.get('/win/:id', require('../controllers/game').win);
6   router.get('/lose/:id', require('../controllers/game').lose);
7   // router.get('/:id', require('../controllers/game').view);
8
9
10  module.exports = router;
```

mini_pokeapi > src > routes > JS pokemon.js > ...

```
1   const { Router } = require('express');
2
3   const router = Router();
4
5   // Controladores
6
7   router.get('/seed', require('../controllers/pokemon').pokemonSeeder);
8   router.get('/random', require('../controllers/pokemon').randomPokemon);
9
10
11  module.exports = router;
```

mini_pokeapi > src > routes > JS user.js > ...

```
1   //Rutas de los Endpoints
2   const { Router } = require('express');
3   const router = Router();
4   router.get('/', require('../controllers/user').showUsers);
5   router.get('/:id',require('../controllers/user').viewUser);
6   router.post('/',require('../controllers/user').createUser);
7   router.delete('/:id',require('../controllers/user').removeUser);
8   router.put('/:id',require('../controllers/user').updateUser);
9   router.post('/login',require('../controllers/user').loginUser);
10  module.exports = router;
11
```

TEST archivos. http para las pruebas

mini_pokeapi > src > JS app.js > ...

```
// Estructura basica de la aplicación
const express = require('express');
const cors = require('cors');

const env = require('./config/environment');

class App {
    constructor() {
        this.app = express();
        this.port = env.port;
        this.middlewares();
        this.routes();
    }

    middlewares() {
        this.app.use(cors());
        //ejecutar npm install cors
        this.app.use(express.json());

    }

    start() {
        this.app.listen(this.port, () => {
            console.log(`Servidor esta ejecutando en el puerto ${this.port}`);
        });
    }

    routes() {
        //Para que resonponda localhoost:3000
        this.app.get('/', (req, res) => {
            res.send('Hola Mundo!');
        });
        //Para que responda localhoost:3000/user/
        this.app.use('/user', require('./routes/user'));
        //Para que responda localhoost:3000/pokemon/
        this.app.use('/pokemon', require('./routes/pokemon'));
        //Para que responda localhoost:3000/game/
```

Index.js punto de entrada del sistema

mini_pokeapi > JS index.js > ...

```
1   //punto de entrada de la API equivalente al ejecutable
2   const App = require('./src/app');
3   const app = new App();
4
5   app.start();
```

.env archivo de variables de entorno reales

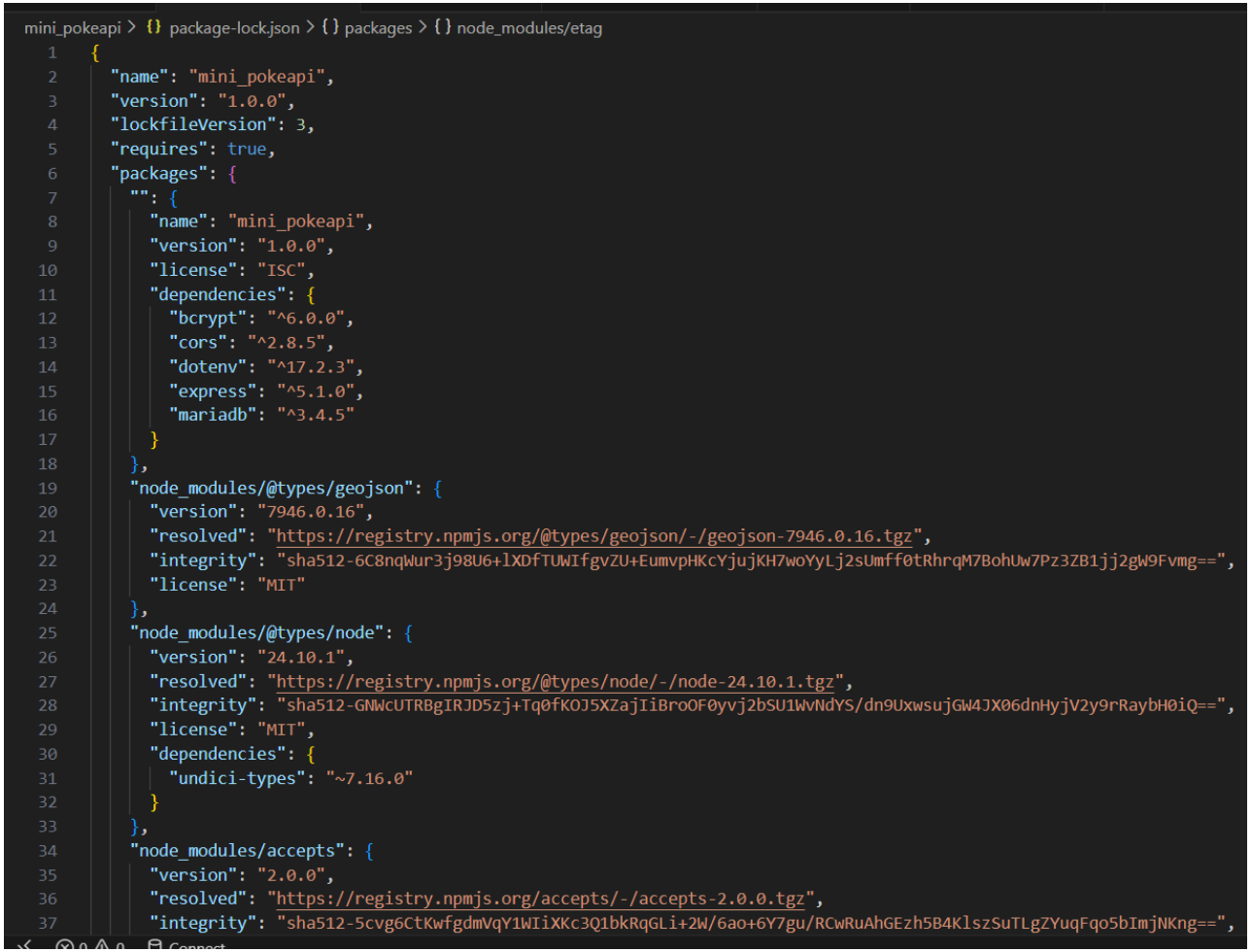
mini_pokeapi > .env

```
1   #guarda variables de entorno e información sensible
2
3   DB_PORT = 3306
4   DB_NAME = pokemondb
5   DB_USERNAME = root
6   DB_PASSWORD = 5507
7   PORT = 3000
```

mini_pokeapi > .env.template

```
1   #guarda variables de entorno e información sensible
2   DB_PORT =
3   DB_NAME =
4   DB_USERNAME =
5   DB_PASSWORD =
6   PORT =
```

Package-lock.json




"dependencies": {
"mime-types": "^3.0.0",
"negotiator": "^1.0.0"
},
"engines": {
"node": ">= 0.6"
}
},
"node_modules/bcrypt": {
"version": "6.0.0",
"resolved": "https://registry.npmjs.org/bcrypt/-/bcrypt-6.0.0.tgz",
"integrity": "sha512-cU8v/EGSrnH+HnxV2z0J7/blxH8gq7Xh2JFT6Aroax7UohdmiJJlxApMxtKfuI7z68NvvVcmR78k2LbT6efhRg==",
"hasInstallScript": true,
"license": "MIT",
"dependencies": {
"node-addon-api": "^8.3.0",
"node-gyp-build": "^4.8.4"
},
"engines": {
"node": ">= 18"
}
},
"node_modules/body-parser": {
"version": "2.2.1",
"resolved": "https://registry.npmjs.org/body-parser/-/body-parser-2.2.1.tgz",
"integrity": "sha512-nfDwkulwiZYQIGwxdy0RUmowMhKcFVcYXUU7m4QlKYim1rUtg83xm2yjZ40QjDuc291AJjjeSc9b++AWHSgSHw==",
"license": "MIT",
"dependencies": {
"bytes": "^3.1.2",
"content-type": "^1.0.5",
"debug": "^4.4.3",
"http-errors": "^2.0.0",
"iconv-lite": "^0.7.0",
"on-finished": "^2.4.1",

Package.json

Pokemondb.sql es el script de la base de datos

mini_pokeapi > pokemondb.sql

```sql
CREATE DATABASE  IF NOT EXISTS `pokemondb` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
USE `pokemondb`;
-- MySQL dump 10.13  Distrib 8.0.44, for Win64 (x86_64)
--
-- Host: 127.0.0.1    Database: pokemondb
-- ------------------------------------------------------
-- Server version 8.0.44

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `games`
--

DROP TABLE IF EXISTS `games`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `games` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int DEFAULT NULL,
  `win` int DEFAULT NULL,
  `lose` int DEFAULT NULL,
  `date` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

```

mini_pokeapi > pokemondb.sql

```sql
-- Dumping data for table `games`
--

LOCK TABLES `games` WRITE;
/*!40000 ALTER TABLE `games` DISABLE KEYS */;
/*!40000 ALTER TABLE `games` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `pokemons`
--

DROP TABLE IF EXISTS `pokemons`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `pokemons` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
  `image` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `pokemons`
--

LOCK TABLES `pokemons` WRITE;
/*!40000 ALTER TABLE `pokemons` DISABLE KEYS */;
/*!40000 ALTER TABLE `pokemons` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `users`
--
```

En esta práctica se implementó un sistema cliente-servidor para un juego de Pokémon, aplicando una arquitectura modular y buenas prácticas de desarrollo backend. El proyecto integra Node.js como servidor, MySQL como gestor de base de datos y una estructura clara basada en rutas, controladores y modelos. El proyecto permitió comprender de forma práctica la separación de responsabilidades entre capas (rutas, controladores, modelos y configuración), así como el uso de variables de entorno para proteger información sensible como credenciales de base de datos. Asimismo, se reforzaron conceptos clave como la conexión a bases de datos relacionales, el manejo de dependencias con npm y la organización modular de un proyecto backend.

NOTA: La aplicación no logró conectar con la base de datos local, pero el análisis del código muestra que el sistema está bien diseñado y puede funcionar correctamente.