



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

```

mini_pokeapi > src > api > JS pokeapi.js > getPokemons
1  const getPokemons = async () => {
2      //const pokemon = [];
3      //let pokemons = [];
4      // Link de imagen de pokemon en .svg
5
6
7      const response = await fetch(`https://pokeapi.co/api/v2/pokemon?limit=151`);
8      const { results } = await response.json();
9
10     const pokemons = results.map((pokemon, index) => {
11         return { ...pokemon, image: `https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/other/dream-world/${index + 1}.svg` }
12     });
13
14     // for (let i = 1; i <= 151; i++) {
15     //     // const response = await fetch(`https://pokeapi.co/api/v2/pokemon/${i}`);
16     //     // const data = await response.json();
17
18     //     const pokemon = {
19     //         name: data.name,
20     //         imagen: data.sprites.other.dream_world.front_default,
21     //     }
22     //     pokemons.push(pokemon);
23     // }
24
25
26     return pokemons;
27 }
28
29
30 module.exports = {
31     getPokemons
32 };

```

ENCAPSULA LA COMUNICACIÓN CON UN API EXTERNA DE POKEMON

```
mini_pokeapi > src > config > JS dbconnection.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

```
mini_pokeapi > src > controllers > JS game.js > ...
1  const { request, response } = require('express');
2  const pool = require('../config/dbconnection');
3  const { gameQuery } = require('../models/game');
4  const { users } = require('../models/user');
5
6  const win = async (req = request, res = response) => {
7    const { id } = req.params;
8
9    if (isNaN(Number(id))) {
10     res.status(400).send("Esto no es un numero");
11     return;
12   }
13
14
15
16   let conn;
17   try {
18     conn = await pool.getConnection();
19     const [user] = await conn.query(users.view, [id]);
20
21     if (!user) {
22       res.status(500).send("No se pudo registrar la victoria");
23       return;
24     }
25     const [game] = await conn.query(gameQuery.getGame, [id]);
26
27     if (!game) {
28       const newGame = await conn.query(gameQuery.addGame, [id, 1, 0]);
29       if (newGame.affectedRows === 0) {
30         res.status(500).send("No se pudo registrar la victoria");
31         return;
32       }
33       res.send({ msg: "registro del juego creado" });
34       return;
35     }
36
37     const gameUpdated = await conn.query(gameQuery.updateGame, [game.win + 1, game.lose, id]);
```

Pokemon.js el controlador de pokemon

```
mini_pokeapi > src > controllers > JS pokemonjs > ...
1  const { request, response } = require('express');
2  const pokeapi = require('../api/pokeapi');
3  const pool = require('../config/dbconnection');
4  const { pokemonQuery } = require('../models/pokemon');
5
6  const pokemonSeeder = async (req = request, res = response) => {
7    const pokemons = await pokeapi.getPokemons();
8
9    let conn;
10
11    try {
12      conn = await pool.getConnection();
13      await conn.query('SET FOREIGN_KEY_CHECKS = 0');
14      await conn.query('TRUNCATE TABLE pokemons');
15      await conn.query('SET FOREIGN_KEY_CHECKS = 1');
16
17      pokemons.forEach(async (pokemon) => {
18        await conn.query(pokemonQuery.add, [pokemon.name, pokemon.image]);
19      });
20      res.send("Pokemones agregados en la Base de Datos");
21    } catch (err) {
22      return res.status(500).send(err);
23    } finally {
24      if (conn) {
25        conn.end();
26      }
27    }
28  }
29
30
31  const randomPokemon = async (req = request, res = response) => {
32    let conn;
33    try {
34      conn = await pool.getConnection();
35      const pokemons = await conn.query(pokemonQuery.random);
36      if (pokemons.length === 0) {
37        return res.status(500).send("No hay pokemones en la base de datos");
```

```
38    }
39    res.send(pokemons);
40
41  } catch (err) {
42    return res.status(500).send(err);
43  } finally {
44    if (conn) {
45      conn.end();
46    }
47  }
48  }
49
50  const idPokemon = async (req = request, res = response) => {
51    let conn;
52    try {
53      const { id } = req.params
54      if (isNaN(Number(id))) {
55        res.status(400).send("Esto no es un numero");
56        return;
57      }
58
59      conn = await pool.getConnection();
60      const pokeid = await conn.query(pokemonQuery.view, [id]);
61      if (!pokeid) {
62        res.status(404).send("Pokemon no encontrado");
63        return;
64      }
65      res.send(pokeid);
66    } catch (err) {
67      return res.status(500).send(err);
68    } finally {
69      if (conn) {
70        conn.end();
71      }
72    }
73  }
```

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/dbconnection');
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;
37     }
38   }
39 }
```

```
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]);
75   }
76 }
```

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 interacciones 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 > ...
1  // Estructura basica de la aplicación
2  const express = require('express');
3  const cors = require('cors');
4
5  const env = require('./config/environment');
6
7  class App {
8    constructor() {
9      this.app = express();
10     this.port = env.port;
11     this.middlewares();
12     this.routes();
13   }
14
15   middlewares() {
16     this.app.use(cors());
17     //ejecutar npm install cors
18     this.app.use(express.json());
19   }
20
21   start() {
22     this.app.listen(this.port, () => {
23       console.log(`Servidor esta ejecutando en el puerto ${this.port}`);
24     });
25   }
26
27   routes() {
28     //Para que responda localhost:3000
29     this.app.get('/', (req, res) => {
30       res.send('Hola Mundo!');
31     });
32     //Para que responda localhost:3000/user/
33     this.app.use('/user', require('./routes/user'));
34     //Para que responda localhost:3000/pokemon/
35     this.app.use('/pokemon', require('./routes/pokemon'));
36     //Para que responda localhost:3000/game/
37
```

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

```
mini_pokeapi > {} package-lock.json > {} packages > {} node_modules/etag
```

```

1 {
2   "name": "mini_pokeapi",
3   "version": "1.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "mini_pokeapi",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "bcrypt": "^6.0.0",
13        "cors": "^2.8.5",
14        "dotenv": "^17.2.3",
15        "express": "^5.1.0",
16        "mariadb": "^3.4.5"
17      }
18    },
19    "node_modules/@types/geojson": {
20      "version": "7946.0.16",
21      "resolved": "https://registry.npmjs.org/@types/geojson/-/geojson-7946.0.16.tgz",
22      "integrity": "sha512-6C8nqWur3j98U6+LXDfTUWIfgvZU+EumvPHKcYjujKH7woYyLj2sUmff0tRhrqM7BohUw7Pz3ZB1jj2gW9Fvmg=",
23      "license": "MIT"
24    },
25    "node_modules/@types/node": {
26      "version": "24.10.1",
27      "resolved": "https://registry.npmjs.org/@types/node/-/node-24.10.1.tgz",
28      "integrity": "sha512-GNwCUTRBgIRJD5zj+Tq0fK0J5XZaJiBro0F0yv72bSU1WwNdYS/dn9UxwsujGW4JX06dnHyjv2y9rRaybH0iQ==",
29      "license": "MIT",
30      "dependencies": {
31        "undici-types": "~7.16.0"
32      }
33    },
34    "node_modules/accepts": {
35      "version": "2.0.0",
36      "resolved": "https://registry.npmjs.org/accepts/-/accepts-2.0.0.tgz",
37      "integrity": "sha512-urbo18ZJcDkN08hP/367C/NWkOYV7dtd/NIW3ghOoRT/386340UaKQoiw/9XLLv2Y8F4B28U4QJ48GPA=="
38    }
39  }
40 }

```

```

39     "dependencies": {
40       "mime-types": "^3.0.0",
41       "negotiator": "^1.0.0"
42     },
43     "engines": {
44       "node": ">= 0.6"
45     }
46   },
47   "node_modules/bcrypt": {
48     "version": "6.0.0",
49     "resolved": "https://registry.npmjs.org/bcrypt/-/bcrypt-6.0.0.tgz",
50     "integrity": "sha512-cU8v/EGSrnH+HnxV2z0J7/blxH8gq7Xh2JFT6Aroax7UohdmiJJlxApmXtKfuI7z68NvvVcmR78k2LbT6efhRg==",
51     "hasInstallScript": true,
52     "license": "MIT",
53     "dependencies": {
54       "node-addon-api": "^8.3.0",
55       "node-gyp-build": "^4.8.4"
56     },
57     "engines": {
58       "node": ">= 18"
59     }
60   },
61   "node_modules/body-parser": {
62     "version": "2.2.1",
63     "resolved": "https://registry.npmjs.org/body-parser/-/body-parser-2.2.1.tgz",
64     "integrity": "sha512-nfDwkuwizZYQIGwxdy0RUMowMhKcFVcYXUU7m4QlKYim1rUt83xm2yJz40QjDuc291AJjjeSc9b++AWHSgSHw==",
65     "license": "MIT",
66     "dependencies": {
67       "bytes": "^3.1.2",
68       "content-type": "^1.0.5",
69       "debug": "^4.4.3",
70       "http-errors": "^2.0.0",
71       "iconv-lite": "^0.7.0",
72       "on-finished": "^2.4.1",
73       "qs": "^6.14.0"

```

Package.json

mini_pokeapi > {} package.json > ...

```
1  {
2    "name": "mini_pokeapi",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "type": "commonjs",
13   "dependencies": {
14     "bcrypt": "^6.0.0",
15     "cors": "^2.8.5",
16     "dotenv": "^17.2.3",
17     "express": "^5.1.0",
18     "mariadb": "^3.4.5"
19   }
20 }
21
```

Pokemondb.sql es el script de la base de datos

```
mini_pokeapi > ≡ pokemondb.sql
1 CREATE DATABASE IF NOT EXISTS `pokemondb` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
2 USE `pokemondb`;
3 -- MySQL dump 10.13 Distrib 8.0.44, for Win64 (x86_64)
4 --
5 -- Host: 127.0.0.1 Database: pokemondb
6 --
7 -- Server version 8.0.44
8
9 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
10 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
11 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
12 /*!50503 SET NAMES utf8 */;
13 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
14 /*!40103 SET TIME_ZONE='+00:00' */;
15 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
16 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
17 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
18 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
19
20 --
21 -- Table structure for table `games`
22 --
23
24 DROP TABLE IF EXISTS `games`;
25 /*!40101 SET @saved_cs_client = @@character_set_client */;
26 /*!50503 SET character_set_client = utf8mb4 */;
27 CREATE TABLE `games` (
28   `id` int NOT NULL AUTO_INCREMENT,
29   `user_id` int DEFAULT NULL,
30   `win` int DEFAULT NULL,
31   `lose` int DEFAULT NULL,
32   `date` timestamp NULL DEFAULT NULL,
33   PRIMARY KEY (`id`)
34 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
35 /*!40101 SET character_set_client = @saved_cs_client */;
36
```


mini_pokeapi > ≡ pokemondb.sql

```
38  -- Dumping data for table `games`
39  --
40
41  LOCK TABLES `games` WRITE;
42  /*!40000 ALTER TABLE `games` DISABLE KEYS */;
43  /*!40000 ALTER TABLE `games` ENABLE KEYS */;
44  UNLOCK TABLES;
45
46  --
47  -- Table structure for table `pokemons`
48  --
49
50  DROP TABLE IF EXISTS `pokemons`;
51  /*!40101 SET @saved_cs_client      = @@character_set_client */;
52  /*!50503 SET character_set_client = utf8mb4 */;
53  CREATE TABLE `pokemons` (
54    `id` int NOT NULL AUTO_INCREMENT,
55    `name` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
56    `image` varchar(255) COLLATE utf8mb4_general_ci DEFAULT NULL,
57    PRIMARY KEY (`id`)
58  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
59  /*!40101 SET character_set_client = @saved_cs_client */;
60
61  --
62  -- Dumping data for table `pokemons`
63  --
64
65  LOCK TABLES `pokemons` WRITE;
66  /*!40000 ALTER TABLE `pokemons` DISABLE KEYS */;
67  /*!40000 ALTER TABLE `pokemons` ENABLE KEYS */;
68  UNLOCK TABLES;
69
70  --
71  -- Table structure for table `users`
72  --
73
```

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.