CODIFICACIÓN DE MÓDULOS DEL SOFTWARE SEGÚN REQUERIMIENTOS DEL PROYECTO

GA7-220501096-AA2-EV01

WILLIAM SALCEDO LACOUTURE

INSTRUCTOR

EDGAR CARVAJAL CACERES

CENTRO INDUSTRIAL DE MANTENIMIENTO INTEGRAL GIRON(SANTANDER)

TECNOLOGO EN ANALISIS Y DESARROLLO DE SOFTWARE

FICHA: 2879661

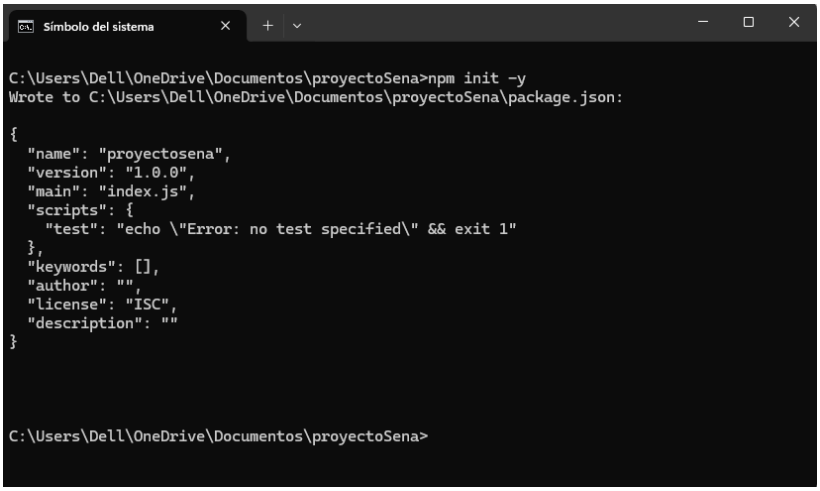
INTRODUCCIÓN

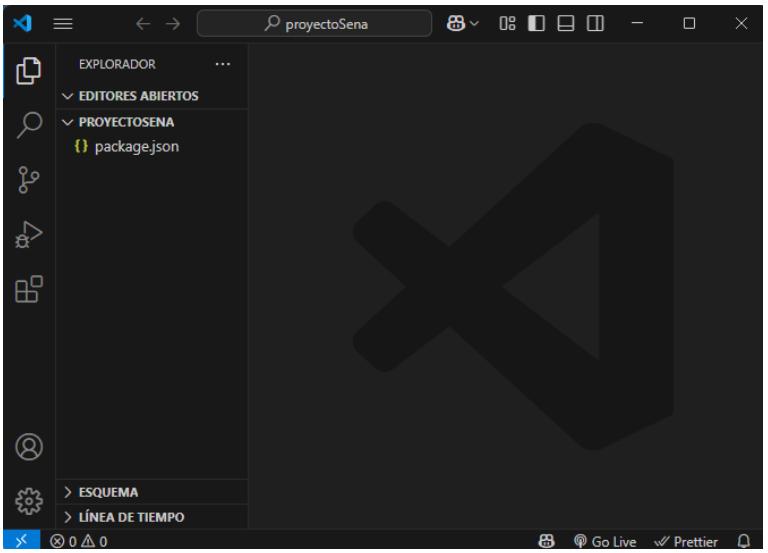
En el desarrollo de software, la gestión de usuarios es un componente fundamental para cualquier aplicación web. Este trabajo presenta la implementación de un módulo de usuarios utilizando Node.js y MySQL, siguiendo las mejores prácticas de codificación y respetando los artefactos del ciclo de vida del software.

El módulo desarrollado permite realizar operaciones básicas CRUD (Crear, Leer, Actualizar y Eliminar) sobre una base de datos de usuarios, garantizando un acceso eficiente y estructurado a la información. Para ello, se ha empleado el framework Express.js para la gestión de rutas y el paquete MySQL para la conexión con la base de datos.

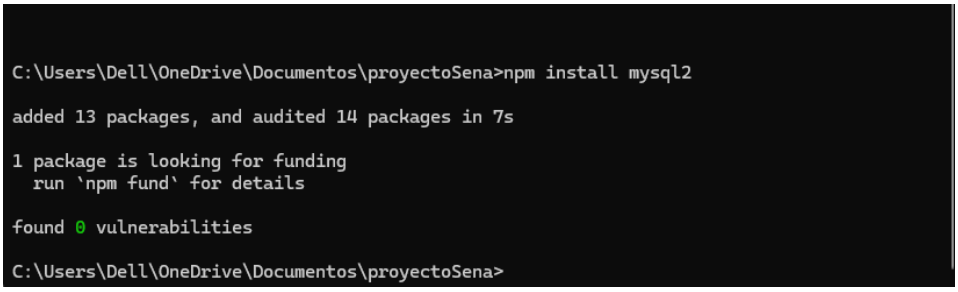
1. Conexión a base de datos

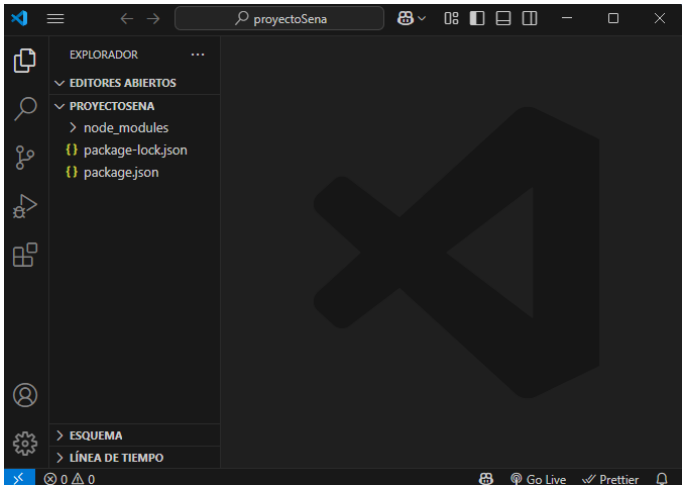
Comenzamos creando un proyecto en node.js





Luego instalamos la librería de MySQL y express





```
C:\Users\Dell\OneDrive\Documentos\proyectoSena>npm install express

added 69 packages, and audited 83 packages in 10s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Por último, vamos a instalar cors para no tener problemas con la conectividad a la base de datos.

```
C:\Users\Dell\OneDrive\Documentos\proyectoSena>npm install cors

added 2 packages, and audited 85 packages in 4s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

2. Funcionalidades CRUD

- Crear usuario (POST /usuarios)

  Permite agregar un nuevo usuario a la base de datos.
  Recibe nombre, email y password en el cuerpo de la solicitud.
  Inserta el usuario en la tabla usuarios.

```
app.post('/usuarios', (req, res) => {
    const { nombre, email, password } = req.body;
    const sql = 'INSERT INTO usuarios (nombre, email, password) VALUES (?, ?, ?)';
    connection.query(sql, [nombre, email, password], (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json({ message: 'Usuario creado', id: result.insertId });
    });
});
```

- Consultar usuarios (GET /usuarios)

  Obtiene la lista de todos los usuarios registrados

```
app.get('/usuarios/:id', (req, res) => {
    const { id } = req.params;
    connection.query('SELECT * FROM usuarios WHERE id = ?', [id], (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        if (result.length === 0) return res.status(404).json({ message: 'Usuario no encontrado' });
        res.json(result[0]);
    });
});
```

  .

- Consultar usuario por ID (GET /usuarios/:id)

  Busca un usuario específico según su id.
  Retorna los datos del usuario si existe.

```
app.get('/usuarios/:id', (req, res) => {
    const { id } = req.params;
    connection.query('SELECT * FROM usuarios WHERE id = ?', [id], (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        if (result.length === 0) return res.status(404).json({ message: 'Usuario no encontrado' });
        res.json(result[0]);
    });
});
```

- Actualizar usuario (PUT /usuarios/:id)

  Modifica los datos de un usuario existente.

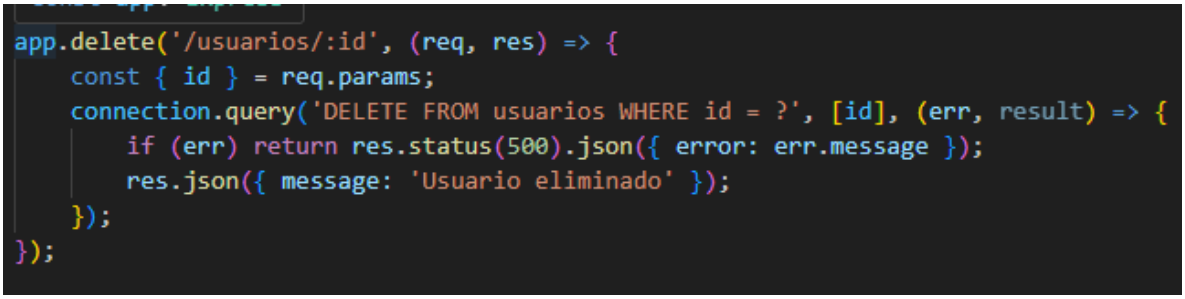
  Recibe nombre, email y password en el cuerpo de la solicitud.

  Actualiza los datos del usuario en la base de datos.

```
app.put('/usuarios/:id', (req, res) => {
    const { id } = req.params;
    const { nombre, email, password } = req.body;
    const sql = 'UPDATE usuarios SET nombre = ?, email = ?, password = ? WHERE id = ?';
    connection.query(sql, [nombre, email, password, id], (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json({ message: 'Usuario actualizado' });
    });
});
```

- Eliminar usuario (DELETE /usuarios/:id)

  Elimina un usuario de la base de datos según su id.



```
app.delete('/usuarios/:id', (req, res) => {
    const { id } = req.params;
    connection.query('DELETE FROM usuarios WHERE id = ?', [id], (err, result) => {
        if (err) return res.status(500).json({ error: err.message });
        res.json({ message: 'Usuario eliminado' });
    });
});
```

CONCLUSIÓN

La implementación del módulo CRUD de usuarios en Node.js con MySQL permite gestionar de manera eficiente la base de datos Ghost Play, asegurando operaciones de creación, lectura, actualización y eliminación de registros. A través del uso de Express.js y mysql2, logramos establecer una conexión estable con la base de datos y definir endpoints que permiten interactuar con los datos de forma sencilla y estructurada.