



**Instituto Tecnológico de
Sonora**

Ingeniería en software

Tópico:

**Temas Emergentes de Aplicaciones
Web**

Mtro. Martin Guadalupe Bernal Lugo

Santana Celaya Alec Demian

ID:216750

Jueves 18 de Enero del 2024

En este documento, se indagará en dos paquetes de Node.js: `sequelize` y `jsonwebtoken`. Describiremos sus propósitos, explicaremos el proceso de instalación y se desarrollaran ejemplos de uso prácticos para cada uno de los paquetes.

1. Introducción a los paquetes

Al momento de crear un proyecto tiene que llenar el formulario que se obtiene al ejecutar en la consola y sobre el directorio del proyecto el comando:

npm init

A partir de ahí podemos proseguir con la instalación

1.1 Sequelize

Sequelize es un ORM moderno de TypeScript y Node.js para Oracle, Postgres, MySQL, MariaDB, SQLite y SQL Server, y distintas bases de datos. Con soporte sólido para transacciones, relaciones, carga entusiasta y diferida, replicación de lectura y más.

1.2 Jsonwebtoken

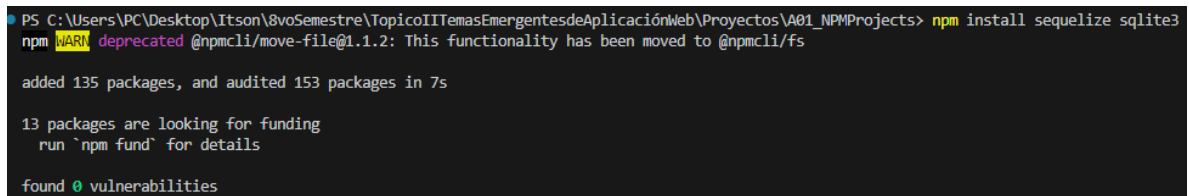
`jsonwebtoken` es un paquete que le permite trabajar con JSON Web Tokens (JWT) en sus aplicaciones Node.js. Los JWT se utilizan para la transmisión segura de información entre partes. Este paquete facilita la creación, verificación y decodificación de tokens, lo cual es crucial para los mecanismos de autenticación y autorización.

2. Instalación

2.1 Instalando Sequelize

Para instalar el paquete `sequelize`, en la terminal y en la dirección de su proyecto. Ejecute el siguiente comando:

npm install sequelize sqlite3



```
PS C:\Users\PC\Desktop\Itson\8voSemestre\TopicoII\TemasEmergentesdeAplicaciónWeb\Proyectos\A01_NPMProjects> npm install sequelize sqlite3
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
added 135 packages, and audited 153 packages in 7s

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

found 0 vulnerabilities
```

Adicionalmente También puede instalar manualmente el controlador para la base de datos de su elección;

\$ npm install --save mysql2

\$ npm install --save sqlite3

\$ npm install --save tedious # Microsoft SQL Server

2.2 Instalando Jsonwebtoken

Siguiendo el mismo proceso en la terminal se ejecuta el siguiente comando:

npm install jsonwebtoken

```
PS C:\Users\PC\Desktop\Itson\8voSemestre\TopicoIItemasEmergentesdeAplicaciónWeb\Proyectos\A01_NPMProjects> npm install jsonwebtoken
added 17 packages, and audited 18 packages in 2s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

3. Ejemplos de usos prácticos

3.1 Sequelize

Sequelize nos es útil para la conexión a una base de datos, así como su modelado, definición y mapeo. El siguiente es un ejemplo de una base de datos de memoria volátil, con una tabla simple de usuarios.

```
async function initDatabase() {
  const sequelize = new Sequelize("sqlite::memory:");

  User = sequelize.define("User", {
    username: DataTypes.STRING,
    password: DataTypes.STRING,
    type: DataTypes.INTEGER,
  });

  await sequelize.sync();
}
```

En el siguiente ejemplo creamos un registro de usuario

```
async function registerUser(username, password, type) {
  await User.create({
    username: username,
    password: password,
    type: type,
  });
}
```

Y de igual forma se puede hacer un login básico

```

async function login(username, password) {
  let user = await User.findOne({
    where: {
      username: username,
      password: password,
    },
  });

  if (!user) {
    throw new Error("Invalid credentials");
  }

  let token = await createTokenJWT(username);

  return token;
}

```

3.2 Jsonwebtoken

`jsonwebtoken` es de utilidad para manejar tokens de autenticación. A continuación, se muestra un ejemplo de cómo generar y validar los JWTs:

Generación del JWT. En cuanto un usuario inicia sesión se le asigna un token para poder acceder a las funciones del sistema:

```

async function createTokenJWT(username) {
  var token = await jwt.sign({ username: username }, "DatesSecret");
  return token;
}

```

Validación JWT. En base a la validación del JWT le permite ejecutar diferentes funciones:

```

async function showPermissions(token) {
  let decoded = await jwt.verify(token, "DatesSecret");
  let user = await User.findOne({
    where: {
      username: decoded.username,
    },
  });

  if (!user) {
    throw new Error("Invalid credentials");
  }

  switch (user.type) {
    case 1:
      getDay().then((day) => console.log(day));
      break;
    case 2:
      sum(Number(readline.question("num1: ")), Number(readline.question("num2: "))).then(
        (result) => console.log(result)
      );
      break;
    default:
      console.log("Invalid type");
  }
}

```