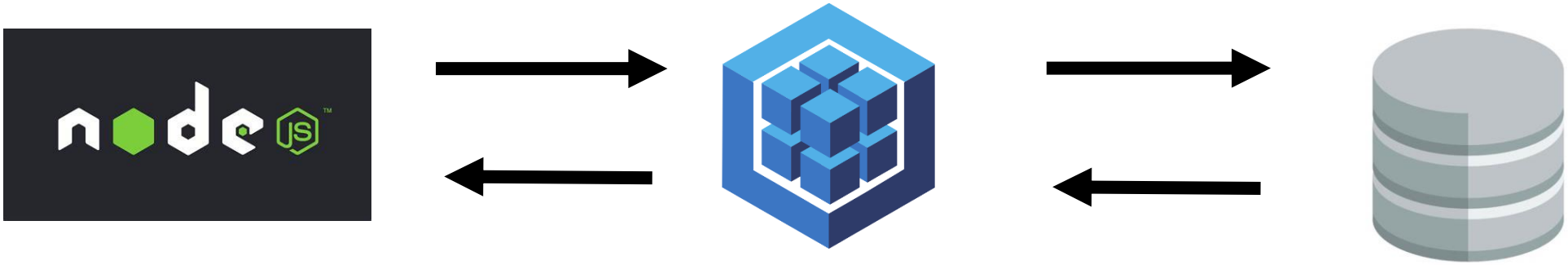
SEQUELIZE

Sequelize es un ORM

(object relational mapper)



Sequelize nos permite trabajar con:

Postgres, MariaDB, MySQL, SQLite y Microsoft SQL server

Promise-based

Muchos de los métodos con los que trabaja sequelize devuelven una promesa.

Instalación

```
npm install sequelize

npm install sequelize-cli

npm install mysql2
```

Configuración de carpetas

```
const path = require('path')

module.exports = {
config: path.resolve('./database/config', 'config.js'),
'models-path': path.resolve('./database/models'),
'seeders-path': path.resolve('./database/seeders'),
'migrations-path': path.resolve('./database/migrations'),
}

Ejecutar: Sequelize init
```

Primera conexión

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('mysql://user:pass@example.com:5432/dbname')
```

Comprobar conexión

```
sequelize.authenticate();
```

DataTypes

```
DataTypes.STRING              // VARCHAR(255)
DataTypes.STRING(1234)        // VARCHAR(1234)
DataTypes.STRING.BINARY       // VARCHAR BINARY

DataTypes.BOOLEAN             // TINYINT(1)

DataTypes.INTEGER             // INTEGER
DataTypes.BIGINT              // BIGINT
DataTypes.FLOAT               // FLOAT

DataTypes.DATE                // DATE WITH TIMESTAMPS
```

Documentación: Data Types

Modelos

```
    const { Sequelize, DataTypes } = require('sequelize');


     const Personas = sequelize.define('User', {

     Nombre: {
     type: DataTypes.STRING,
     },
     Apellido: {
     type: DataTypes.STRING
     }
});
```

Configuración de la columna

```
     allowNull:false,
     defaultValue: “string” / booleano / integer
     primarykey: true/false
```

Sincronizar modelos

```
// CREA LAS TABLAS SI NO EXISTEN, SI EXISTEN, NO HACE NADA

sequelize.sync()


// BORRA LAS TABLAS EXISTENTES Y LAS CREA DE NUEVO

sequelize.sync({force:true})


// ACTUALIZA EL MODELO DE LA TABLA SI ES QUE EXISTEN CAMBIOS

sequelize.sync({alter:true})
```

timeStamps

```
Nombre: {
     type: DataTypes.STRING,
     allowNull:false,
     },
Apellido: {
     Type: DataTypes.STRING,
     defaultValue: "Sin apellido",
     },
      },{
timeStamps: false
},);

Nombre: {
     type: DataTypes.STRING,
     allowNull:false,
     },
Apellido: {
     Type: DataTypes.STRING,
     defaultValue: "Sin apellido",
     },
      },{
timestamps: true,
createdAt: false
},);
```

Instancias

```
INSERT

await Personas.create({
  Apellido: "Lauga"
});

UPDATE

await Persona.update({ Nombre: "Pedrito"},{
  where: {
    Apellido: "Perez"
  }
  })

DELETE

await Personas.destroy({
     where: {
          Nombre: "Pepe"
     }
})
```

Queries

```
// BUSCAR TODOS LOS DATOS DE LA COLUMNA

const users = await Personas.findAll();


// BUSCAR TODOS LOS DATOS SEGÚN PARAMETRO

const usersfind = await Personas.findAll({
attributes: ['foo', 'bar']
});

// ENCONTRAR UNO SEGÚN PARAMETRO

const findone = await Personas.findOne()

// ENCONTRAR UNO SEGÚN ID

const byid = await Personas.findByPk()
```

Operadores

```
const findbyage = await Personas.findAll({
    where: {
    [Op.and]: [{ a: 5 }, { b: 6 }],            // (a = 5) AND (b = 6)
    [Op.or]: [{ a: 5 }, { b: 6 }],             // (a = 5) OR (b = 6)
    someAttribute: {
 // Basics
        [Op.eq]: 3,                              // = 3
        [Op.ne]: 20,                             // != 20
        [Op.is]: null,                           // IS NULL
        [Op.not]: true,                          // IS NOT TRUE
        [Op.or]: [5, 6],
        }
    }
})
```

Documentación: Operadores

Order, Limit y Offset

```
const findall = await db.Persona.findAll({

        order:[ ["edad","ASC"]],
        limit: 3,
        offset: 3

})
```