

SEQUELIZE MIGRATIONS

<https://sequelize.org/v6/manual/migrations.html>



Crear una base de datos en MySQL

```
prompt> mysql -u root -p
mysql> create database escolar;
mysql> create user 'backenduser'@'localhost' identified by
'superpassword';
mysql> grant all privileges on escolar.* to
'backenduser'@'localhost';
mysql> flush privileges;
mysql> quit
```

■ Probar:

```
prompt> mysql -u backenduser -p escolar
Enter password: *****
mysql> quit
```

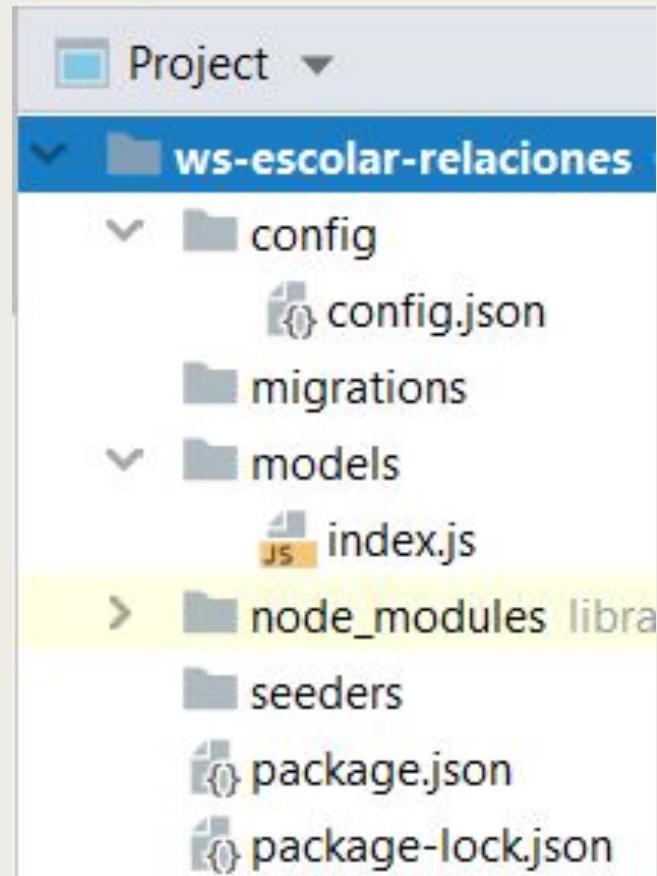
Migrations

- Interfaz de comandos para gestionar el estado de la base de datos
 - *sequelize-cli*
- Sirve para crear un nuevo estado de la base de datos o revertir a un estado previo

Creación de un nuevo proyecto

- `mkdir <nombre de directorio>`
- `cd <nombre de directorio>`
- `npm init`
- `npm install sequelize`
- `npm install mysql2`
 - *o el driver del DBMS utilizado: mariadb / sqlite3 / tedious*
- `npx sequelize-cli init`
 - *Esto creará la estructura de directorios para mantener la migración*

Estructura de directorios creada por el sequelize-cli init



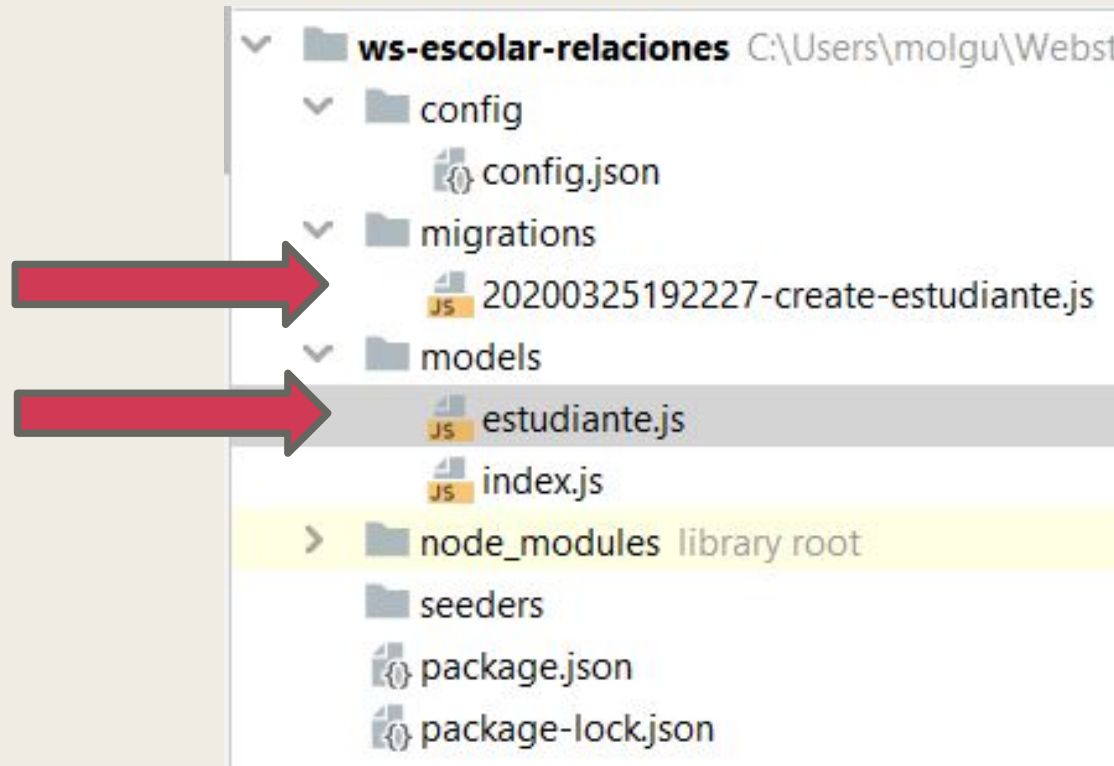
Editar el archivo config/config.json para poner los datos de conexión al DBMS

```
{
  "development": {
    "username": "backenduser",
    "password": "superpassword",
    "database": "escolar2",
    "host": "localhost",
    "dialect": "mysql",
  },
  "test": {
    "username": "backenduser",
    "password": "superpasswordtest",
    "database": "escolar2test",
    "host": "localhost",
    "dialect": "mysql"
  },
  "production": {
    "username": "backenduser",
    "password": "superpasswordproduction",
    "database": "escolar2production",
    "host": "localhost",
    "dialect": "mysql"
  }
}
```

Crear un modelo

- **`npx sequelize-cli model:generate --name Estudiante --attributes \ nombre:string,matricula:integer,semestreIngreso:string,creditosCursados:integer`**

Nuevos
archivos



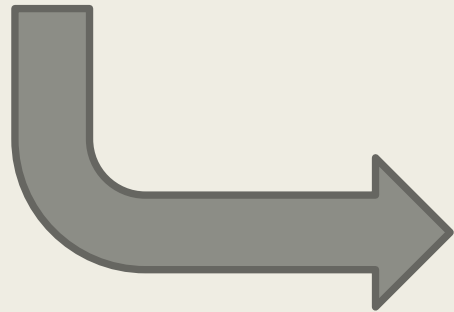
Archivo de definición del modelo

```
const Estudiante =  
sequelize.define('Estudiante', {  
  nombre: DataTypes.STRING,  
  matricula: DataTypes.INTEGER,  
  semestreIngreso: DataTypes.STRING,  
  creditsCursados: DataTypes.INTEGER  
}, {});
```

Hay que modificarlo si se requiere que la definición de atributos tenga más características

Modificar el archivo generado models/estudiante.js

```
const Estudiante =  
sequelize.define('Estudiante', {  
  nombre: DataTypes.STRING,  
  matricula: DataTypes.INTEGER,  
  semestreIngreso: DataTypes.STRING,  
  creditsCursados: DataTypes.INTEGER  
}, {});
```



```
const Estudiante =  
sequelize.define('Estudiante', {  
  nombre: {  
    type: DataTypes.STRING,  
    allowNull: false  
  },  
  matricula: {  
    type: DataTypes.INTEGER,  
    allowNull: false,  
    unique: true  
  },  
  semestreIngreso: {  
    type: DataTypes.STRING,  
    allowNull: true  
  },  
  creditsCursados: {  
    type: DataTypes.INTEGER,  
    allowNull: true,  
    defaultValue: 0  
  }  
}, {});
```

Archivo de migración

Instrucciones para migrar

Instrucciones para deshacer la migración

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('Estudiantes', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nombre: {
        type: Sequelize.STRING
      },
      matricula: {
        type: Sequelize.INTEGER
      },
      semestreIngreso: {
        type: Sequelize.STRING
      },
      creditosCursados: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('Estudiantes');
  }
};
```

Modificar xxxxx-create-estudiante.js

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('Estudiantes', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nombre: {
        type: Sequelize.STRING
      },
      matricula: {
        type: Sequelize.INTEGER
      },
      semestreIngreso: {
        type: Sequelize.STRING
      },
      creditosCursados: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('Estudiantes');
  }
};
```

```
nombre: {
  type: Sequelize.STRING,
  allowNull: false
},
matricula: {
  type: Sequelize.INTEGER,
  allowNull: false,
  unique: true
},
semestreIngreso: {
  type: Sequelize.STRING,
  allowNull: true
},
creditosCursados: {
  type: Sequelize.INTEGER,
  allowNull: true,
  defaultValue: 0
},
```

Crear la tabla correspondiente en el DBMS

- `npx sequelize-cli db:migrate`

```
mysql> show tables;
```

```
+-----+  
| Tables_in_demo_migraciones |  
+-----+  
| estudiantes                |  
| sequelizemeta              |  
+-----+  
2 rows in set (0.00 sec)
```

```
mysql> describe estudiantes;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(255)	NO		NULL	
matricula	int(11)	NO	UNI	NULL	
semestreIngreso	varchar(255)	YES		NULL	
creditosCursados	int(11)	YES		0	
createdAt	datetime	NO		NULL	
updatedAt	datetime	NO		NULL	

```
7 rows in set (0.00 sec)
```

La tabla sequelizemeta lleva el control de los archivos de migration

- El comando **db:migrate** usa esta tabla

```
mysql> describe sequelizemeta;
```

Field	Type	Null	Key	Default	Extra
name	varchar(255)	NO	PRI	NULL	

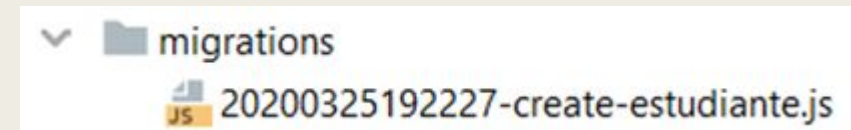
1 row in set (0.00 sec)

```
mysql> select * from sequelizemeta;
```

name
20200325192227-create-estudiante.js


1 row in set (0.00 sec)

Cada vez que se utiliza **migrate** se agregan o borran registros en esta la tabla para llevar el control de los archivos de migración y poder crear un nuevo estado o regresar a un estado anterior **undo**



Regresar la BD a un estado previo

- **npx sequelize-cli db:migrate:undo**
 - *Obtiene el registro más reciente de sequelizemeta y ejecuta el **down***



```
});  
},  
down: (queryInterface, Sequelize) => {  
  return  
  queryInterface.dropTable('Estudiantes');  
}  
};
```

```
mysql> show tables;  
+-----+  
| Tables_in_demo_migraciones |  
+-----+  
| sequelizemeta              |  
+-----+  
1 row in set (0.00 sec)
```

Hizo un drop a la tabla
estudiantes

Migrar de nuevo

- npx sequelize-cli db:migrate
 - *Crear  de nuevo la tabla estudiantes porque ejecuta el **up***

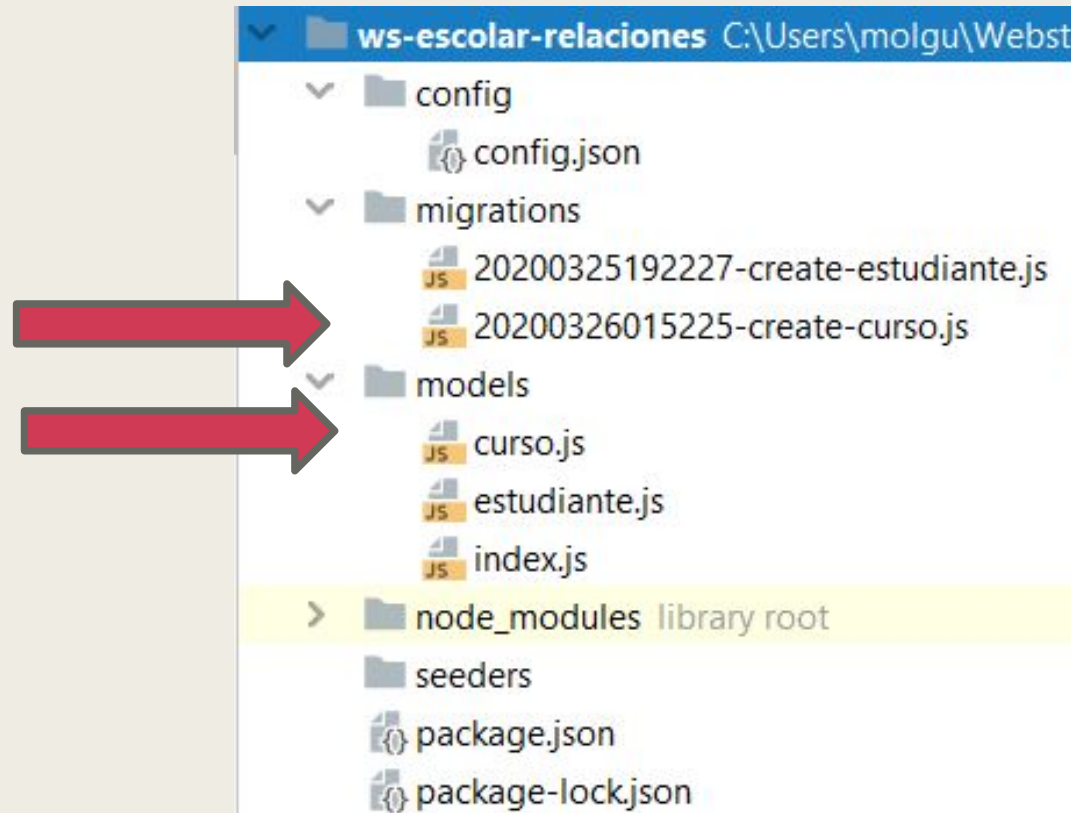


```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('Estudiantes', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nombre: {
        type: Sequelize.STRING
      },
      matricula: {
        type: Sequelize.INTEGER
      },
      semestreIngreso: {
        type: Sequelize.STRING
      },
      creditosCursados: {
        type: Sequelize.INTEGER
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
};
```

Crear un nuevo modelo: Curso

- `npx sequelize-cli model:generate --name Curso --attributes \ nombre:string,clave:integer,creditos:integer`

Nuevos
archivos



Modificar archivos

curso.j

```
'use strict';
module.exports = (sequelize, DataTypes) => {
  const Curso =
    sequelize.define('Curso', {
      nombre: {
        type: DataTypes.STRING,
        allowNull: false
      },
      clave: {
        type: DataTypes.INTEGER,
        allowNull: false,
        unique: true
      },
      creditos: {
        type: DataTypes.INTEGER,
        allowNull: true,
        defaultValue: 0
      }
    }, {}));
  Curso.associate = function(models) {
    // associations can be defined here
  };
  return Curso;
};
```

xxxxxx-create-curs
o.js

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('Cursos', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      nombre: {
        type: Sequelize.STRING,
        allowNull: false
      },
      clave: {
        type: Sequelize.INTEGER,
        allowNull: false,
        unique: true
      },
      creditos: {
        type: Sequelize.INTEGER,
        allowNull: true,
        defaultValue: 0
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('Cursos');
  }
};
```

Migrar para crear la nueva tabla

■ `npx sequelize-cli db:migrate`

- Ejecuta el **up** del archivo que aun no tiene registrado en la tabla `sequelizemeta`:
`xxxxxx-create-curso.js`

```
mysql> show tables;
+-----+
| Tables_in_demo_migraciones |
+-----+
| cursos                      |
| estudiantes                 |
| sequelizemeta               |
+-----+
3 rows in set (0.00 sec)
```

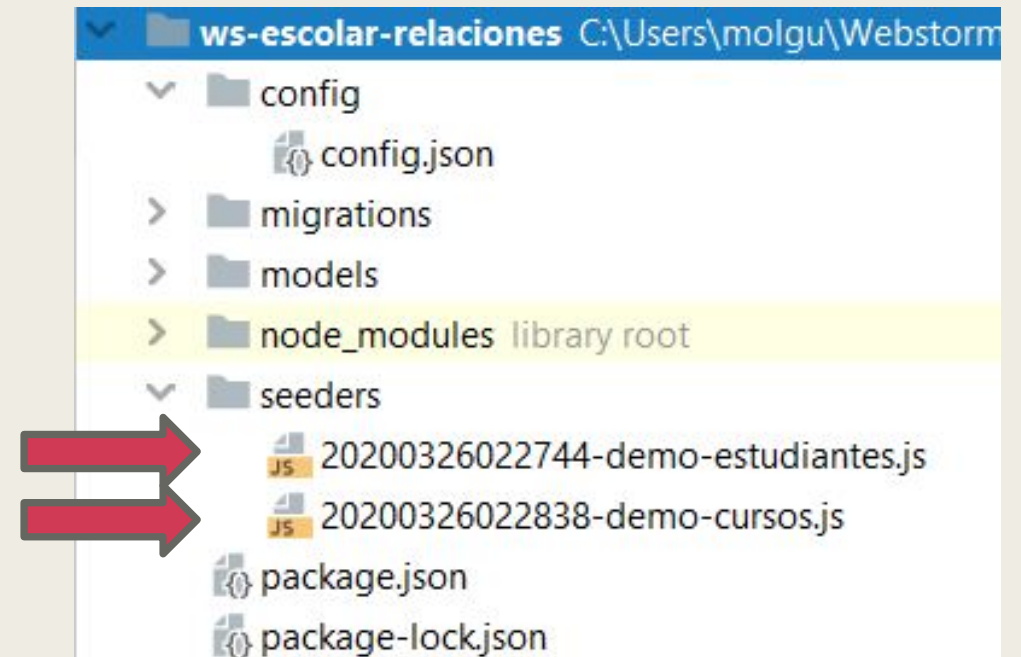
```
mysql> describe cursos;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | int(11)       | NO   | PRI | NULL    | auto_increment |
| nombre | varchar(255)  | NO   |     | NULL    |                |
| clave  | int(11)       | NO   | UNI | NULL    |                |
| credits | int(11)       | YES  |     | 0       |                |
| createdAt | datetime     | NO   |     | NULL    |                |
| updatedAt | datetime     | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Undo

- `npx sequelize-cli db:undo`
 - *Hace drop a la tabla recién creada ejecutando el down del archivo js más reciente*
- `npx sequelize-cli db:undo:all`
 - *Ejecuta todos los down de todos los archivos en orden cronológico. En nuestro caso borra ambas tablas*
- `npx sequelize-cli db:undo:all --to XXXXXX-create-estudiante.js`
 - *Ejecuta los down de todos los archivos en orden cronológico hasta el especificado*

Automatizar carga inicial de datos

- Sequelize-cli provee comandos para automatizar carga y borrado de datos iniciales en las tablas
 - `npx sequelize-cli seed:generate --name demo-estudiantes`
 - `npx sequelize-cli seed:generate --name demo-cursos`



Modificar archivos

xxxx-demo-curso

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.bulkInsert('Cursos', [
      {
        nombre: 'Programacion Perl',
        clave: 100,
        credits: 6,
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        nombre: 'P00',
        clave: 99,
        credits: 6,
        createdAt: new Date(),
        updatedAt: new Date()
      }
    ], {});
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.bulkDelete('Cursos', {}, {});
  }
};
```

xxxx-demo-estudiante
sis

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.bulkInsert('Estudiantes', [
      {
        nombre: 'John Doe',
        matricula: 1000,
        semestreIngreso: '2016-2',
        creditsCursados: 300,
        createdAt: new Date(),
        updatedAt: new Date()
      },
      {
        nombre: 'Jane Doe',
        matricula: 2000,
        semestreIngreso: '2018-2',
        creditsCursados: 200,
        createdAt: new Date(),
        updatedAt: new Date()
      }
    ], {});
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.bulkDelete('Estudiantes', {}, {});
  }
};
```

Ejecutar los seeders

- `npx sequelize-cli db:seed:all`

```
mysql> select * from estudiantes;
```

id	nombre	matricula	semestreIngreso	creditosCursados	createdAt	updatedAt
1	John Doe	1000	2016-2	300	2020-03-26 03:10:14	2020-03-26 03:10:14
2	Jane Doe	2000	2018-2	200	2020-03-26 03:10:14	2020-03-26 03:10:14

```
2 rows in set (0.00 sec)
```

```
mysql> select * from cursos;
```

id	nombre	clave	creditos	createdAt	updatedAt
1	Programacion Perl	100	6	2020-03-26 03:10:14	2020-03-26 03:10:14
2	P00	99	6	2020-03-26 03:10:14	2020-03-26 03:10:14

```
2 rows in set (0.00 sec)
```

Se pueden revertir de la misma forma que la creación/alteración de tablas

■ **sequelize-cli db:seed:undo --seed XXXXXX-demo-estudiantes**

- *Ejecuta el down de ese seed, que en este caso borra todos los registros de la tabla estudiantes*

```
down: (queryInterface, Sequelize) => {  
  return queryInterface.bulkDelete('Estudiantes', {}, {});  
}
```

```
mysql> select * from estudiantes;  
Empty set (0.00 sec)
```



Uso de los modelos



Como utilizar los modelos en la aplicación

```
const models = require('./models');

//models.sequelize.sync().then(() => {
  models.Estudiante.findAll()
    .then(r => {
      r.forEach(estudiante =>{
        console.log(estudiante.dataValues);
      });
      models.sequelize.close();
    });
//});
```

El código comentado sirve para crear las tablas si no existen.

Usar `sync({ force: true })` hace que se borren las tablas y se reconstruyan vacías

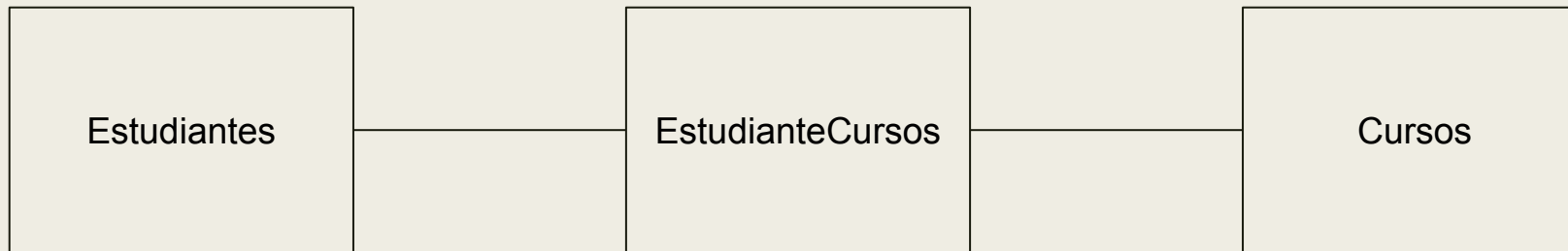


Relaciones muchos a muchos



Modelo intermedio para la asociación de muchos a muchos

Many-to-many relation



cursoId: integer
estudianteId: integer
semestre: string
createdAt: date
updatedAt: date

Generación del modelo y archivo de migración

- `npx sequelize-cli model:generate --name EstudianteCursos --attributes estudianteId:integer,cursoid:integer,se
mestre:string`

estudiantecursos.js

```
'use strict';
module.exports = (sequelize, DataTypes) => {
  const EstudianteCursos = sequelize.define('EstudianteCursos', {
    estudianteId: DataTypes.INTEGER,
    cursoid: DataTypes.INTEGER,
    semestre: DataTypes.STRING
  }, {});
  EstudianteCursos.associate = function(models) {
    // associations can be defined here
  };
  return EstudianteCursos;
};
```

xxxxxxx-create-estudiante-cursos.js

```
'use strict';
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('EstudianteCursos', {
      id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.INTEGER
      },
      estudianteId: {
        type: Sequelize.INTEGER
      },
      cursoid: {
        type: Sequelize.INTEGER
      },
      semestre: {
        type: Sequelize.STRING
      },
      createdAt: {
        allowNull: false,
        type: Sequelize.DATE
      },
      updatedAt: {
        allowNull: false,
        type: Sequelize.DATE
      }
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('EstudianteCursos');
  }
};
```

Crear la tabla

- `npx sequelize-cli db:migrate`

```
mysql> describe estudiantecursos;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
estudianteId	int(11)	YES		NULL	
cursoId	int(11)	YES		NULL	
semestre	varchar(255)	YES		NULL	
createdAt	datetime	NO		NULL	
updatedAt	datetime	NO		NULL	

```
6 rows in set (0.00 sec)
```

Modificar los archivos de los modelos para definir la relación

estudiante.js

```
'use strict';
module.exports = (sequelize, DataTypes) => {
  const Estudiante = sequelize.define('Estudiante',
  {
    nombre: {
      type: DataTypes.STRING,
      allowNull: false
    },
    matricula: {
      type: DataTypes.INTEGER,
      allowNull: false,
      unique: true
    },
    semestreIngreso: {
      type: DataTypes.STRING,
      allowNull: true
    },
    creditosCursados: {
      type: DataTypes.INTEGER,
      allowNull: true,
      defaultValue: 0
    }
  }, {});
  Estudiante.associate = function(models) {
    Estudiante.belongsToMany(models.Curso, {
      through: models.EstudianteCursos
    });
  };
  return Estudiante;
};
```

curso.js

```
'use strict';
module.exports = (sequelize, DataTypes) => {
  const Curso = sequelize.define('Curso', {
    nombre: {
      type: DataTypes.STRING,
      allowNull: false
    },
    clave: {
      type: DataTypes.INTEGER,
      allowNull: false,
      unique: true
    },
    creditos: {
      type: DataTypes.INTEGER,
      allowNull: true,
      defaultValue: 0
    }
  }, {});
  Curso.associate = function(models) {
    Curso.belongsToMany(models.Estudiante, {
      through: models.EstudianteCursos
    });
  };
  return Curso;
};
```

Código de prueba de la relación muchos a muchos

demo.js

```
const models = require('./models');

async function demoAsociacionMuchosAMuchos() {
  let estudiante = await models.Estudiante.findOne({
    where: {
      matricula: 2000
    }
  });
  let cursos = await models.Curso.findAll();
  console.log(
    "Datos del estudiante: ",
    estudiante.id,
    estudiante.nombre,
    estudiante.matricula,
    estudiante.semestreIngreso);
  // Asociar todos los cursos al estudiante
  await estudiante.addCursos(cursos, {through: {semestre: "2020-1"}});
  // Desplegar los datos de los cursos asociados al estudiante
  let cursosAsociados = await estudiante.getCursos();
  console.log("Cursos del estudiante con matricula:", estudiante.matricula);
  cursosAsociados.forEach( curso => {
    console.log(curso.nombre, curso.clave, curso.creditos);
  });
  // AL hacer la asociacion de estudiante con curso se puede acceder
  // el dato del alumno através del curso
  let cursoP00 = await models.Curso.findOne({where: {nombre: "P00"}});
  let estudiantesP00 = await cursoP00.getEstudiantes();
  console.log("Estudiantes de P00:")
  estudiantesP00.forEach( e => {console.log(e.nombre, e.matricula)})
  models.sequelize.close();
}

demoAsociacionMuchosAMuchos();
```

Contenido de la tabla de asociación

- El código anterior deja la tabla de asociación en el siguiente estado:

```
mysql> select * from estudiantecursos;
```

id	estudianteId	cursoId	semestre	createdAt	updatedAt
5	2	1	2020-1	2020-03-27 00:28:09	2020-03-27 00:28:09
6	2	2	2020-1	2020-03-27 00:28:09	2020-03-27 00:28:09

```
2 rows in set (0.00 sec)
```


Métodos de acceso a la asociación

- Definir las relación muchos a muchos en los modelos genera automáticamente los métodos para consultar y agregar asociaciones entre los modelos.
- `belongsToMany` crea:
 - `Estudiante.getCursos()` y `Estudiante.addCursos()`
 - `Cursos.getEstudiantes()` y `Cursos.addEstudiantes()`
- Revisar la documentación para los demás tipos de relaciones:
 - `hasMany`
 - `belongsToMany`
 - `hasOne`

<https://sequelize.org/v5/manual/associations.html>