Repaso & Relaciones muchos a muchos

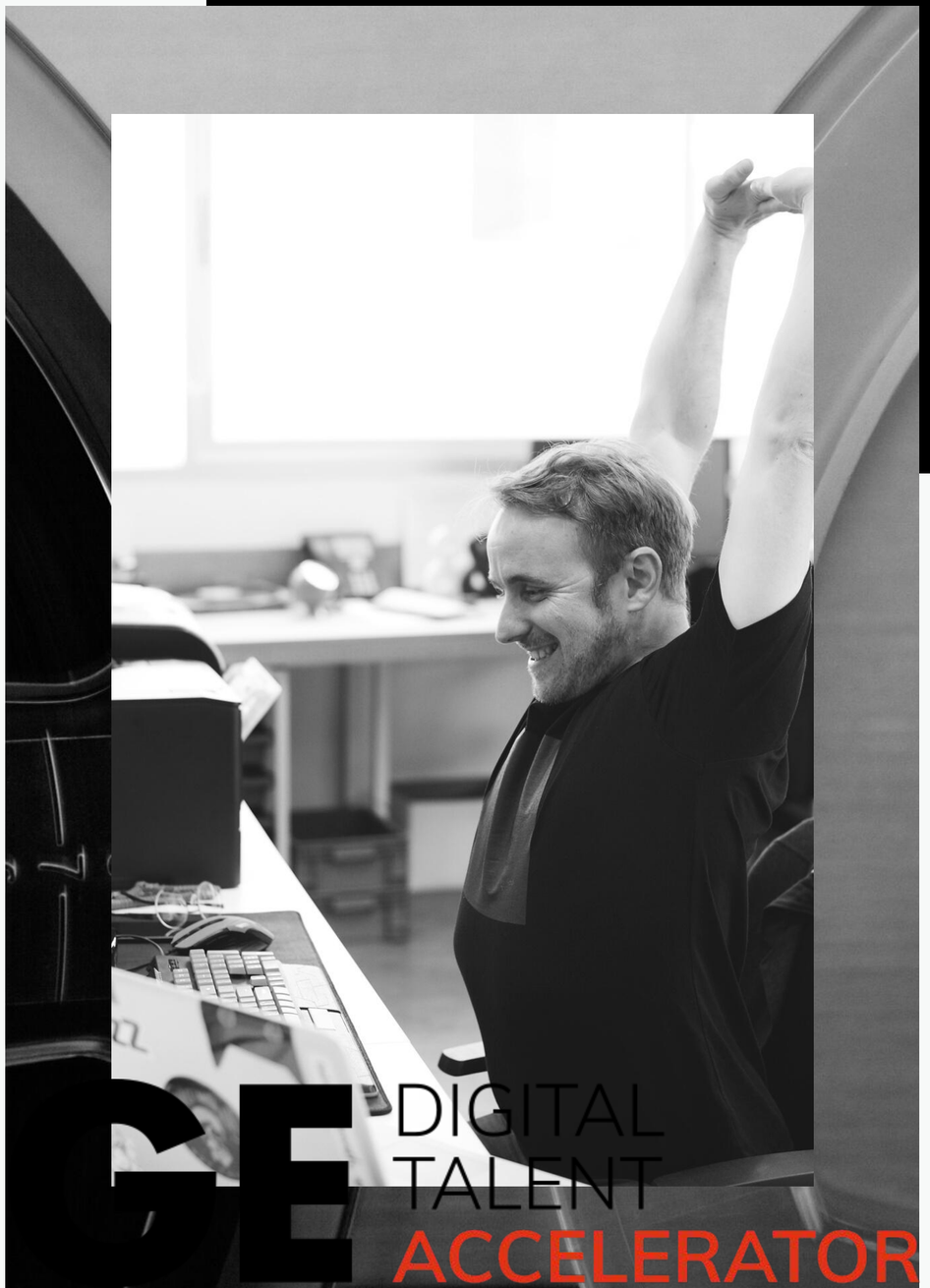
THE BRIDGE
DIGITAL TALENT ACCELERATOR

Índice

Empezando proyecto

Modelos y migraciones

Relaciones muchos a muchos



THE BRIDGE DIGITAL TALENT ACCELERATOR

Creando un proyecto en sequelize

Generando proyecto de Sequelize

Nos situaremos sobre el directorio en el que queremos crear el nuevo proyecto y escribiremos los comandos siguientes:

```
$ npm init -y
$ npm install express sequelize mysql2
$ sequelize init
```

Inicializamos nuestro proyecto de sequelize

Creando un Modelo (y migración) Book

Para crear un modelo (y migración) utilizaremos también el CLI. Definiremos el **nombre**, los **atributos** y sus **tipos de datos**.

Nos creará el modelo **Book** y su migración (que al ejecutar creará la tabla con estos campos).

```
$ sequelize model:generate --name Book --attributes name:string,price:float
```

Creando un Modelo (y migración) Genre

Para crear un modelo (y migración) utilizaremos también el CLI. Definiremos el **nombre**, los **atributos** y sus **tipos de datos**.

Al darle enter nos creará el modelo **Genre** y su migración (que al ejecutarla creará la tabla con estos campos).

```
$ sequelize model:generate --name Genre --attributes name:string
```

Creando un Modelo (y migración) GenreBook

Para crear un modelo (y migración) utilizaremos también el CLI. Definiremos el **nombre**, los **atributos** y sus **tipos de datos**.

Al darle enter nos creará el modelo **GenreBook** y su migración (que al ejecutarla creará la tabla con estos campos).

```
$ sequelize model:generate --name GenreBook --attributes
GenreId:integer,BookId:integer
```

Levantando la base de datos

Para crear una base de datos, escribiremos el **nombre** en el **config.json** en la propiedad **database**. Una vez hecho, ejecutamos el siguiente comando:

```
{
  "development": {
    "username": "root",
    "password": null,
    "database": "database_development",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },

...
$ sequelize db:create
```

Levantando las tablas

Para crear las tablas en la base de datos, debemos ejecutar el siguiente comando:

```
$ sequelize db:migrate
```

Modelo Book

En el archivo generado en **models** tendremos nuestro **modelo Book**.

Por convención el **modelo** debe empezar por **letra mayúscula** y **singular**.

En el método estático **associate** definimos la relación:

```
'use strict';
const { Model } = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Book extends Model {
    static associate(models) {
      Book.belongsToMany(models.Genre,{
        through:models.GenreBook
      })
    }
  }
  Book.init({
    name: DataTypes.STRING,
    price: DataTypes.FLOAT
  }, {
    sequelize,
    modelName: 'Book',
  });
  return Book;
};
```

relación muchos a muchos

Modelo Genre

En el archivo generado en **models** tendremos nuestro **modelo Genre**.

Por convención el modelo **debe** empezar por **letra mayúscula** y **singular**.

En el método estático **associate** definimos la relación:

```
'use strict';
const { Model } = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class Genre extends Model {
    static associate(models) {
      Genre.belongsToMany(models.Book,{
        through:models.GenreBook,
      })
    }
  }
  Genre.init({
    name: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'Genre',
  });
  return Genre;
};
```

relación muchos a muchos

Controlador Book

Creamos **BookController.js** (en la carpeta controllers) en el cual nos importamos el modelo Book. Creamos un objeto **BookController** que contendrá los diferentes métodos que vamos a implementar, en este caso la creación de un libro:

```
const { Book } = require('../models/index');

const BookController = {
    insert(req,res){
        Book.create(req.body)
        .then(book=>{
          book.addGenre(req.body.GenreId)
          res.send(book)
        })
        .catch(err => console.error(err))
    },
}

module.exports = BookController;
```

importamos el modelo book

sequelize nos da métodos especiales formados por un prefijo. En este caso **add** concatenado con el **nombre del modelo** (con la primera letra en mayúscula)

Ruta books

Creamos el archivo **books.js** en la carpeta **routes** con el siguiente código. Nos importamos el archivo **BookController.js** que estará en la carpeta controllers:

```
const express = require('express');
const router = express.Router();
const BookController = require('../controllers/BookController')

router.post('/',BookController.insert)

module.exports = router;
```

importamos el BookController

Creación de index.js

Creamos nuestro archivo **index.js** con el siguiente código. Nos importamos la carpeta routes con el archivo **books.js** que contiene nuestras rutas:

```
const express = require('express');
const app = express();
const PORT = 3000

app.use(express.json())

app.use('/books', require('./routes/books'));

app.listen(PORT, () => console.log('servidor levantado en el puerto' + PORT))
```

importamos nuestras rutas de books

Controlador Genre

Creamos **GenreController.js** (en la carpeta controllers) en el cual nos importamos el modelo Genre. Creamos un objeto **GenreController** que contendrá los diferentes métodos que vamos a implementar, en este caso la creación de un género:

```
const { Genre } = require('../models/index');

const GenreController = {
    insert(req,res){
        Genre.create(req.body)
        .then(genre=>{
            res.send(genre)
        })
        .catch(err => console.error(err))
    },
}

module.exports = GenreController
```

importamos el modelo genre

Ruta genres

Creamos el archivo **genres.js** en la carpeta **routes** con el siguiente código. Nos importamos el archivo **GenreController.js** que estará en la carpeta controllers:

```
const express = require('express');
const router = express.Router();
const GenreController = require('../controllers/GenreController')

router.post('/',GenreController.insert)

module.exports = router;
```

importamos el GenreController

Actualizamos index.js

Actualizamos **index.js** con las rutas de **genres.js**

```
const express = require('express');
const app = express();
const PORT = 3000

app.use(express.json())

app.use('/books', require('./routes/books'));
app.use('/genres, require('./routes/genres));

app.listen(PORT, () => console.log('servidor levantado en el puerto' + PORT))
```

importamos nuestras rutas de genres

Get

Nos traemos los libros junto a sus géneros de la siguiente forma:

```
const { Book ,Genre } = require('../models/index');

const BookController = {
. . .
  async getAll(req, res) {
    try {
      const books = await Book.findAll({
        include: [{ model: Genre, through: { attributes: [] } }],
      });
      res.send(books);
    } catch (error) {
      console.error(error);
    }
  },

}
module.exports = BookController;
```

Ruta books

Añadimos la nueva ruta en nuestro archivo **books.js**:

```
const express = require('express');
const router = express.Router();
const BookController = require('../controllers/BookController')

router.post('/',BookController.insert)
router.get('/',BookController.getAll)

module.exports = router;
```

Delete

Para eliminar un Libro lo haremos de su tabla y de la tabla intermedia de la siguiente forma:

```
const { Book, Genre, GenreBook } = require('../models/index');

const BookController = {
. . .
    async delete(req, res) {
        try {
            await Book.destroy({
                where: {
                    id: req.params.id
                }
            })
            await GenreBook.destroy({
                where: {
                    BookId: req.params.id
                }
            })
            res.send({ message: 'The book has been removed'})
        }
         catch (error) {
            console.log(error)
        }
    }
}

module.exports = BookController;
```

Ruta books

Añadimos la nueva ruta en nuestro archivo books.js:

```
const express = require('express');
const router = express.Router();
const BookController = require('../controllers/BookController')

router.post('/',BookController.insert)
router.get('/',BookController.getAll)
router.delete('/:id',BookController.delete)


module.exports = router;
```

Update

Para actualizar un Libro lo hacemos de su tabla y de la tabla intermedia de la siguiente forma:

```
  async update(req, res) {
      try {
        await Book.update(req.body,
          {
            where: {
              id: req.params.id,
            },
          }
        );
        const book = await Book.findByPk(req.params.id)
        book.setGenres(req.body.GenreId);
        res.send("Libro actualizado con éxito");
      } catch (error) {
        console.error(error);
        res
          .status(500)
          .send({ message: 'no ha sido posible actualizar el
libro' });
      }
    },
```

Rutas Books

Añadimos nuestra nueva ruta en nuestro archivo books.js:

```
const express = require('express');
const router = express.Router();
const BookController = require('../controllers/BookController')

router.post('/',BookController.insert)
router.get('/',BookController.getAll)
router.delete('/:id',BookController.delete)
router.put('/:id',BookController.update)

module.exports = router;
```