Crear nuevo proyecto de node.js

Para crear un nuevo proyecto de node.js

1. Crear una nueva carpeta en algún directorio local, donde no exista git
2. Ingresar a la carpeta creada, y agregar dos carpetas mas:
    1. **src**
    2. **test**
3. Instalar globalmente *Express Generator* con el comando

```
npm i -g express-generator
```

4. Inicializar un proyecto base con el comando

```
express --view=ejs
```

5. Inicializar git dentro del proyecto con el comando

```
git init
```

6. Instalar las dependencias con el comando

```
npm i
```

7. La carpeta ***node_modules*** no se debe subir al repositorio, y para eso creamos el archivo `.gitignore` en la carpeta raíz del proyecto y agregarlos del siguiente modo:

```
/node_modules
/config/database.json
```

9. Ingresar a GitHub, crear un repositorio privado sólo con el nombre del proyecto.
10. Los demás miembros del equipo, deberán clonar el repositorio recientemente creado mediante el comando

```
git clone <URL del repositorio>
```

10. Para utilizar el ORM **Sequelize**, se deben instalar las siguientes librerías:
    1. Para instalar **Sequelize** `npm i sequelize`
    2. Para instalar la dependencia de **MySQL** `npm i mysql2`
    3. Para instalar la dependencia de **Sequelize-cli** `npm i sequelize-cli`
11. Inicializar el ORM de **Sequelize** con el siguiente comando, el cual creará las siguientes carpetas:

```
npx sequelize init
```

- `config` la cual contiene el archivo de configuración, que le dirá a **Sequelize CLI** cómo conectar con la Base de Datos.
- `models` el cual contiene todos los modelos para el proyecto.
- `migrations` el cual contiene todos los archivos de las migraciones.
- `seeders` el cual contiene todos los archivos de los seeders.

13. Se deberá crear la carpeta **db** en la raíz del proyecto, a la cual se deberán mover posteriormente las carpetas ***migrations***, ***models*** y ***seeders***.
14. En el archivo *index.js* ubicado en `db > models > index.js` se deberá modificar el directorio de la importación del ***database.json***:

```
'use strict';

const fs = require('fs');
const path = require('path');
const Sequelize = require('sequelize');
const process = require('process');
const basename = path.basename(__filename);
const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../../config/database.json')[env];
const db = {};
```

14. Crear en la raíz del proyecto, un nuevo archivo con el nombre **.sequelizerc**, que contendrá la siguiente información:

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

module.exports = {
        'config': path.resolve('config', 'database.json'),
        'models-path': path.resolve('db', 'models'),
        'seeders-path': path.resolve('db', 'seeders'),
        'migrations-path': path.resolve('db', 'migrations')
};
```

15. Crear los modelos que vaya a requerir el proyecto ejecutando el siguiente comando, por cada modelo:

```
npx sequelize-cli model:generate --name User --attributes
columnName:string,columnName:string,columnName:string
```

- `name` es el nombre del modelo
- `attributes` es la lista de propiedades/atributos que tendrá el modelo
- El modelo se creará dentro de la carpeta identificada con el nombre `models` en la ruta `db > models`.
- Como referencia para el tipo de dato a utilizar para cada atributo/propiedad del modelo, se puede usar las siguientes referencias Data Types de *Sequelize*.

16. Luego de completar la creación de todos los modelos necesarios, se deberán ejecutar todas las Migraciones con el siguiente comando, lo cual permitirá generar las tablas con sus respectivos campos en la Base de Datos que fue configurada previamente:

```
npx sequelize-cli db:migrate
```

17. Crear cada uno de los **seeders** que haga falta en el proyecto, mediante el siguiente comando, donde `seeder-name` es el nombre que le daremos al seeder para su identificación:

```
npx sequelize-cli seed:generate --name seeder-name
```

18. Al ejecutar el comando para crear el seeder, se creará un archivo dentro de la carpeta *seeders*, y al ingresar se habrá generado el código para **insertar** y **eliminar** el seeder desde la base de datos, como se muestra a continuación:

```
'use strict';

/** @type {import('sequelize-cli').Migration} */

module.exports = {
  async up (queryInterface, Sequelize) {
    /**
     * Add seed commands here.
     *
     * Example:
     * await queryInterface.bulkInsert('People', [{
     *   name: 'John Doe',
     *   isBetaMember: false
     * }], {});
    */
```

```
  },

  async down (queryInterface, Sequelize) {
    /**
     * Add commands to revert seed here.
     *
     * Example:
     * await queryInterface.bulkDelete('People', null, {});
     */
  }
};
```

19. Se deberá descomentar el código necesario para habilitar los métodos `bulkInsert` y `bulkDelete`, que se conectarán con la DB en el momento de ser ejecutados, quedando por ejemplo del siguiente modo:

```
"use strict";

module.exports = {
  async up(queryInterface, Sequelize) {
    /* *
     * Add seed commands here.
     *
     * Example: */
    await queryInterface.bulkInsert("Authors",
      [
        {
          name: "Fiódor Dostoievski",
          createdAt: "2022-01-01 22:58:01",
          updatedAt: "2022-01-01 22:58:01"
        },
        {
          name: "Hans Christian Andersen",
          createdAt: "2022-01-01 22:58:01",
          updatedAt: "2022-01-01 22:58:01"
        },
      ], {});


  },

  async down(queryInterface, Sequelize) {
    /* *
     * Add commands to revert seed here.
     *
     * Example: */
     await queryInterface.bulkDelete('Authors', null, {});
  },
};
```

20. Cuando tengamos todos los *seeders* definidos, deberemos ejecutar el siguiente comando, el cual permitrá ejecutar cada uno de los archivos seeders e insertará en las respectivas

tablas de la DB, la información definida (verificar en la DB, que la información fue persistida correctamente):

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