Proyecto Final

Webservices con express

Proyecto final:

1. Crear un proyecto de Express con SEQUELIZE que tenga 3 tablas:
   - users:
     - id: autonumérico
     - email: de tipo string y único (unique) y no nulo
     - password: de tipo string y no nulo
     - type: de tipo string
     - active: de tipo boolean con valor por defecto true
   - teachers:
     - id: autonumérico
     - dni: de tipo string
     - name: de tipo string
     - last_name: de tipo string
     - date_of_birth: de tipo date
     - user_id: id de un usuario. Relación 1:1
   - students:
     - id: autonumérico
     - dni: de tipo string
     - name: de tipo string
     - last_name: de tipo string
     - date_of_birth: de tipo date
     - teacher_id: id de un profesor. Un profesor puede tener varios estudiantes, un estudiante sólo puede pertenecer a un profesor.

Para las asociaciones revisar estos artículos: Artículo 1, Artículo 2

Proyecto final:

2. Generar seeds para poblar cada tabla, con usuarios activos y no activos, de tipo (type) “admin” y “user”. Guardar la contraseña encriptada (bcrypt con promesas)
3. Crear endpoints que ejecuten métodos CRUD (**C**reate - POST, **R**ead - GET, **U**pdate - PUT and **D**elete - DELETE) para cada recurso (users, teachers y students). Es decir, los endpoints para user son:
    - GET /api/users: para obtener un json con todos los usuarios
    - GET /api/users/:id: para obtener un json con el usuario de ese id
    - POST /api/users: para dar de alta un usuario
    - PUT /api/users/:id: para actualizar un usuario
    - DEL /api/users/:id: para eliminar un usuario
4. Restricciones en el CRUD:
    - No se puede borrar un usuario que tenga un profesor asociado
    - No se puede borrar un profesor que tenga alumnos asociados
5. Crear el endpoint GET /api/teachers/:teacher_id/students que:
    - Compruebe que el profesor con teacher_id exista y su usuario asociado esté activo, en caso contrario devolver un mensaje 401 informando de ello.
    - Devuelva un json con todos los alumnos de ese profesor ordenados por fecha de nacimiento
6. Crear un endpoint POST y GET /api/users/:id/active que:
    - POST: Compruebe que el usuario exista y actualice el campo **active** a true, debe devolver todos los datos del usuario actualizado
    - GET: Compruebe que el usuario exista y devuelva únicamente el campo **active**

Proyecto final:

7. Crear un endpoint **GET /login** que renderice la vista login.html que puedes encontrar en los materiales de la clase 6.
8. Crear un endpoint **POST /login** que reciba dos campos en el body, un username y un password y busque si en la tabla users existe un usuario con el campo email igual al username enviado en el body. Si existe el usuario y la contraseña de la base de datos es igual a la encriptación (bcrypt con promesas) del password enviado, setear una variable de sesión que indique que el usuario está logueado y otra con los datos del usuario sin la contraseña, si todo es correcto debe hacer un redirect a la página home. Si el usuario no existe debe renderizar una página error-login.html informando de ello.
9. Crear un endpoint **GET /users** que renderice una vista users.html con el listado de todos los usuarios del sistema y un botón para hacer logout. Este endpoint debe estar protegido para que sólo un usuario de tipo “admin” pueda acceder, en otro caso devolver una respuesta 401.
10. Crear un endpoint **GET /home** que compruebe si el usuario es de tipo “admin”, en ese caso, redirija al endpoint /users. En otro caso, renderice una vista home.html con los datos del profesor asociado al usuario logueado y un botón para hacer logout.
11. Crear un endpoint **POST /logout** que elimine las variables de sesión de login y haga un redirect a /login
12. Crear un endpoint **POST /api/token** que reciba dos campos en el body, un username y un password y busque si en la tabla existe un usuario con el email igual al username enviado. Si existe el usuario y la contraseña de la base de datos es igual a la encriptación del password enviado, devuelva un json con un token JWT que contenga el username. El tiempo de validez del JWT tiene que ser de 15 minutos.

ALBAÑILESDIGITALES

Developing your new career

servicio navarro de empleo

das·nano

VeriDas