UNIVERSIDADE da MADEIRA

ESCOLA SUPERIOR DE TECNOLOGIAS E GESTÃO

Ano Letivo 2020/2021

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

1 º **Ano**/ 2 º **Semestre**

Unidade Curricular: Desenvolvimento Web - Back-End **Docente:** David Jardim

---

FICHA DE TRABALHO 9

Exercícios:

1. Crie uma pasta para a aula 9
    a. Crie um ficheiro *app.js*
    b. Usando o terminal e o *node packet manager* (npm) instale o Express.js
    c. Usando o terminal e o *node packet manager* (npm) instale o pacote **mysql2**
    d. Usando o terminal e o *node packet manager* (npm) instale o pacote **sequelize**
    e. Usando o terminal e o *node packet manager* (npm) instale o pacote **swagger-ui-express**
2. Crie um servidor em express tal como foi explicado nas aulas anteriores
3. Crie uma base de dados denominada por ficha9 utilizando o MySQL Workbench
    a. Tenha em atenção o username e a password de acesso à base de dados que será necessário nas alíneas seguintes
4. De acordo com as indicações dadas na teoria importe e crie uma ligação à base de dados utilizando o pacote ***sequelize***
    a. Efetue a ligação à base de dados através do sequelize, utilizando os dados de autenticação utilizados na criação da mesma
    b. Utilizando a função ***define*** do sequelize crie um modelo Person para representar pessoas com as seguintes colunas:
        i. *Firstname* (var char)
        ii. *Lastname* (var char)
        iii. *Profession* (var char)
        iv. Age (int)
    c. Utilize a função ***sync*** do sequelize para sincronizar o modelo com a base de dados
    d. Adicione várias entradas na tabela utilizando a função ***bulkCreate*** do sequelize
5. Tendo em conta a tabela 1, implemente os seguintes *endpoints* no app.js:
    a. Listar todas as pessoas existentes na tabela *Persons* e devolver a resposta no *body*
    b. Adicionar uma nova pessoa à tabela *Persons*, o ID deve ser gerado automaticamente pelo MySQL tendo em conta o número de pessoas existentes. O ID da pessoa adicionada deve ser devolvido na resposta.

c. Apagar uma pessoa da tabela *Persons* pelo seu ID recebido no *body*. O número de linhas afetadas deve ser devolvido na resposta. Caso a pessoa a apagar não exista o erro deverá ser tratado de forma adequada.
d. Apagar uma pessoa da tabela *Persons* pelo seu ID recebido como parâmetro. O número de linhas afetadas deve ser devolvido na resposta. Caso a pessoa a apagar não exista o erro deverá ser tratado de forma adequada.
e. Selecionar apenas uma pessoa pelo seu ID (como query) e devolver essa mesma pessoa na resposta. Caso a pessoa a selecionar não exista, o erro deverá ser tratado de forma adequada.
f. Selecionar as pessoas pelo sua idade e profissão. Devolver todas as pessoas que reúnam essas condições. Caso não exista, o erro deverá ser tratado de forma adequada.
g. Alterar os detalhes de uma pessoa selecionada pelo seu ID. Os novos detalhes deverão ser devolvidos na resposta.

6. Documente todos os endpoints utilizando o módulo **swagger-ui-express**
   a. Crie um ficheiro **swagger.json** contendo as configurações necessárias para o **swagger-ui-express**
   b. Adicione ao ficheiro **swagger.json** as definições necessárias para descrever o modelo Product
   c. Adicione ao ficheiro **swagger.json** os *paths* necessários para documentar todos os endpoints que foram implementados
   d. Verifique a documentação gerada, deve ser possível efetuar todos os testes a partir da interface gerada pelo *Swagger*

URI	Método HTTP	Body	Resultado
/persons	GET	empty	Show list of all the persons.
/persons	POST	JSON String	Add details of new person.
/persons	DELETE	JSON String	Delete an existing person.
/persons/:id	DELETE	empty	Delete an existing person.
/persons/:id	GET	empty	Show details of a person.
/persons/:age/:profession	GET	empty	Show details of multiple persons.
/persons/:id	PUT	JSON String	Update details of a person

Tabela 1 - Endpoints a implementar