P.PORTO

ESCOLA SUPERIOR DE MEDIA ARTES E DESIGN
POLITÉCNICO DO PORTO

ALP

PROVA DE AVALIAÇÃO
ATIVIDADE LETIVA

Licenciatura em Tecnologias e Sistemas de Informação para a Web			
CURSO			
2023/2024	14-06-2024	14:00	2h00
ANO LETIVO	DATA	HORA	DURAÇÃO
Programação Web II			2º
UNIDADE CURRICULAR			ANO
Teresa Terroso			Frequência
DOCENTE			ÉPOCA

Durante a realização da prova, pode APENAS consultar a página do Moodle da disciplina, exercícios que tenha desenvolvido nesta disciplina e as páginas de documentação das ferramentas que usar no desenvolvimento das respostas às questões colocadas.

Para cada uma das questões, deverá desenvolver um projeto separado. Deve usar apenas um ficheiro para a definição das rotas e apenas um ficheiro para todos os métodos do controlador.

No final da prova, deve submeter via Moodle um único ficheiro `.zip` contendo os ficheiros dos projetos que criou durante a prova, EXCEPTO as pastas `NODE_MODULES`.

ATENÇÃO:

- Submeta **apenas o código necessário** (e sem erros de compilação ou de execução) para responder às questões colocadas! O não cumprimento implica uma **penalização de 10% da cotação total**.
- Para cada questão, é obrigatório submeter também capturas de ecrã (completo) que **comprovem o consumo das rotas** (usando o Postman, por exemplo). A não submissão implica uma **penalização de 20% da cotação total**.

1. Node + Express + Sequelize (10 valores)

Num projeto escolar do P.Porto um estudante pretende criar uma aplicação web para gestão do próximo campeonato de futsal inter-escolas. Colabore no desenvolvimento da aplicação, criando uma API REST, acessível pelo URL `localhost:3000/futsal_api/`. Para tal, considere a seguinte base de dados relacional:

- A tabela `players` contêm os dados dos jogadores:
  - `nome`: nome do jogador (campo obrigatório)
  - `team`: nome da equipa onde joga (chave estrangeira - campo obrigatório)
- A tabela `teams` guarda os dados das equipas de futebol:
  - `name`: nome da equipa (chave primária)

  - `school`: escola do IPP à qual a equipa pertence (campo obrigatório, cujos valores devem pertencer à seguinte lista – [ESE, ESHT, ESMAD, ESMAE, ESS, ESTG, ISEP, ISCAP])
- A tabela `matches` guarda os jogos de futebol que vão acontecer:
  - `team1`: nome da primeira equipa (chave estrangeira - campo obrigatório)
  - `team2`: nome da segunda equipa (chave estrangeira - campo obrigatório)
  - `date`: data do jogo (os jogos decorrem sempre à mesma hora - campo obrigatório)
  - `local`: local (escola do IPP) onde o jogo será disputado (campo obrigatório, cujos valores devem pertencer à seguinte lista – [ESE, ESHT, ESMAD, ESMAE, ESS, ESTG, ISEP, ISCAP])

PLAYERS	TEAMS	MATCHES
id: número (PK: chave primária) **name**: string **school**: string (FK: chave estrangeira)	**name**: string (PK) **school**: string	**id**: número (PK) **team1**: string (FK) **team2**: string (FK) **date**: data **local**: string

Usando como ferramentas a combinação de Node (servidor) + Express (roteamento) + Sequelize (cliente da base dados), e seguindo as boas práticas que aprendeu sobre desenvolvimento de APIs REST, implemente somente as rotas para as seguintes funcionalidades:

- **Listar as equipas**, podendo a lista ser filtrada pelo nome da escola à qual as equipas pertencem. A lista deve sempre incluir os nomes dos jogadores de cada equipa (remova informação duplicada ou desnecessária sobre os jogadores).
- **Adicionar um novo jogador a uma equipa.** O pedido deve fornecer à API o nome do novo jogador e o nome da sua equipa. Atenção, que uma equipa não pode ter jogadores com o mesmo nome! Inclua na resposta de sucesso o acesso ao jogo criado.
- **Registar um novo jogo.** Esta rota permite criar um jogo, que deverá acontecer sempre pelo menos após 5 dias da data do registo do jogo. O pedido deve conter os seguintes dados: nome de ambas as equipas, data (basta indicar o dia, pois todos os jogos são à mesma hora) e local (escola). A API deve validar os parâmetros recebidos e prevenir todos os erros possíveis, incluindo erros do pedido, como por exemplo, verificar se as equipas são diferentes (uma equipa não joga contra si própria!) ou se a data inserida é superior a 5 dias após a data atual. Inclua na resposta de sucesso o acesso ao jogo criado.

  DICA de como obter a data daqui a 5 dias:

```
const now = new Date();
const inFiveDays = new Date(new Date(now).setDate(now.getDate() + 5))
```

2. Node + Express + Mongoose (10 valores)

Tendo em consideração o número de jogos e jogadores registados na base de dados, o estudante decidiu usar uma base de dados não relacional (MongoDB.) Também achou prudente incluir a técnica de autenticação por JWT - JSON Web Tokens. No entanto, não achou necessário que as palavras-passe fossem encriptadas antes de serem guardadas na base de dados. A tabela seguinte esquematiza as 4 coleções existentes na base de dados, onde todos os campos são obrigatórios:

PLAYERS	TEAMS	MATCHES

_id: ObjectId (PK)
name: string
team: string (REF TEAMS)
	_id: ObjectId (PK)
	name: string (unique)
	school: string
	players: [ObjectId REF PLAYERS]
	coach: {
	email: string (unique),
	password: string
	}
		_id: ObjectId (PK)
		team1: ObjectId REF TEAMS,
		team1_titulars: [ObjectId REF PLAYERS],
		team2: ObjectId REF TEAMS,
		team2_titulars: [ObjectId REF PLAYERS],
		date: data
		local: string

Usando o ODM Mongoose como cliente da base de dados Mongo, e mantendo as boas práticas que aprendeu sobre desenvolvimento de APIs REST, implemente somente as rotas para as seguintes funcionalidades:

- **Autenticação** (rota pública): apenas os treinadores têm permissões de acesso a rotas protegidas; assim, fornecidos o email e a palavra-passe, esta rota autentica-o, devolvendo na resposta o *token* (cujo *payload* contém apenas o identificador da sua equipa).
- **Obter os jogos da sua equipa** (privado, apenas acessível para o treinador dessa equipa): rota que informa todos os detalhes dos jogos da sua equipa, incluindo os nomes das equipas e dos jogadores titulares. Se indicado numa *query string*, a rota poderá devolver apenas os jogos num determinado local específico ou os que são jogados "em casa".

  DICA: se precisar de utilizar o operador OR, aqui fica um exemplo de como pesquisar na coleção `item` de documentos onde o campo `quantity` é igual a 20 **ou** o campo `price` é igual a 10

  ```
  item.find( { $or: [ { quantity: 20 }, { price: 10 } ] } )
  ```

  (https://www.mongodb.com/docs/v6.0/reference/operator/query/or/)
- **Definir um jogador como titular de um jogo** (rota privada, apenas acessível para o treinador da equipa): um treinador pode usar esta rota para adicionar um jogador da sua equipa como titular de um jogo. Atenção que um jogo de futsal apenas pode ter cinco jogadores titulares. Em caso de sucesso, a resposta deve fornecer a lista atualizada dos nomes dos jogadores titulares da sua equipa para esse jogo.

BOA SORTE!