Parte 2 – Parte prática (14 valores)

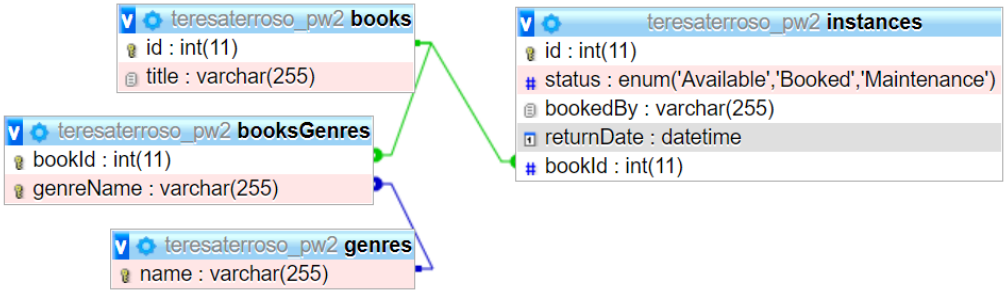
Durante a realização da Parte 2, pode APENAS consultar a página do Moodle da disciplina, exercícios que tenha desenvolvido nesta disciplina e as páginas de documentação das ferramentas que usar no desenvolvimento das respostas às questões colocadas.

ATENÇÃO:

- 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
- Para cada questão, é obrigatório submeter também capturas de ecrã (completo) que **comprovem o consumo com SUCESSO das rotas** (usando o Postman, por exemplo). A não submissão implica uma **penalização de 20% da cotação total.**

1. Node + Express + MySQL + Sequelize (7 valores)

Considere que pretende implementar uma API REST para modernizar a gestão de reservas de livros de uma biblioteca. Para tal, use a seguinte base de dados relacional:



A entidade **books** lista os livros presentes na biblioteca. Os livros podem ser pesquisáveis por género, tendo sido assim criada a entidade **genres** que se relaciona com **books** numa relação N:M. Cada livro pode ter zero ou mais exemplares (instâncias) na biblioteca (entidade **instances**), podendo estas ser reservadas. Os campos **bookedBy** e **returnDate** só contêm valores quando um exemplar for reservado.

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, desenvolva uma API REST, implementando somente as rotas para as seguintes funcionalidades:

- Obter os dados de um determinado livro, incluindo a que género(s) pertence e os exemplares (instâncias) que existem na biblioteca. Usando HATEOAS, a resposta de sucesso deve informar o acesso às seguintes rotas: adicionar um exemplar desse livro, listar todos os exemplares e listar todos os livros da biblioteca do(s) mesmo(s) género(s).
- Adicionar um exemplar (instância) a um livro, devendo apenas ser fornecido o valor do campo **status** (os valores dos campos **bookedBy** e **returnDate** devem ficar a nulo). Em caso de sucesso, a API devolve a rota de acesso à instância criada.
- Reservar um livro: o pedido deve apenas incluir o título do livro a ser reservado. A API deve procurar por um exemplar disponível desse livro e alterar o seu estado para **Booked**; o campo **returnDate** dessa instância deve registar a data atual acrescida de 5 dias:

```
const currentDate = new Date();
const after5days = currentDate.setDate(currentDate.getDate() + 5);
```

2. Node + Express + MongoDB + Mongoose (7 valores)

Passado algum tempo, foi decidido alterar a base de dados relacional para uma não relacional (MongoDB). Verificou-se que não era relevante os dados sobre géneros, mas que seria necessário incluir dados dos utilizadores da API. Assim a base de dados contém os seguintes documentos: **`books`**, **`instances`** e **`users`**, onde as relações entre eles foram implementadas usando documentos referenciados:

`books`	`instances`	`users`
_id: **ObjectId** title: **String** (unique) instances: [**ObjectId REF Instances**]	_id: **ObjectId** book: **ObjectId REF Books** status: **Enum**("Available" | "Booked" | "Maintenance") bookedBy: **ObjectId REF USERS** returnDate: **Date**	_id: **ObjectId** username: **String** (unique) password: **String** isAdmin: **Boolean** rentals: [{ instance: **ObjectId REF Instances**, requestedAt: **Date**, deliveredAt:**Date** }]

Também acharam prudente incluir a técnica de autenticação por JSON Web Tokens. No entanto, não se achou necessário que as palavras-passe fossem encriptadas antes de serem armazenadas na base de dados.

Usando o ODM Mongoose como cliente da base de dados Mongo, somente as rotas para as seguintes funcionalidades:

- (rota pública) Fornecido o nome de registo e palavra-passe, autentica um utilizador, devolvendo na resposta o *token* (cujo *payload* contém apenas o identificador)
- Devolver um exemplar (rota apenas disponível para utilizadores regulares). Não esquecer que um exemplar só pode ser devolvido à biblioteca pela pessoa que o reservou e que implica alterar campos nas coleções **`instances`** e **`users`**.
- Eliminar um livro da biblioteca (rota apenas disponível para administradores): um livro pode ser eliminado se não tiver nenhum exemplar na base de dados ou se nenhum dos exemplares estiver reservado;
  - Se existirem, todos os exemplares e reservas desse livro devem ser eliminados da base de dados.

BOA SORTE!