Parte 2 – Parte prática

(12 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.

Para cada uma das questões da prova, deverá desenvolver um projeto separado.

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: Para cada resposta, coloque **apenas o código necessário** (e sem erros de compilação) para responder às questões colocadas! O não cumprimento implica uma **penalização até 10% da cotação total da questão**.

No 2º exercício, é obrigatório submeter, para além do código, 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 da questão**.

1. Node (4 valores)

Implemente um módulo em Node.js que exporte a função `getAge()`. Esta função, fornecida uma *string* com a sua data de nascimento no formato YYYY-MM-DD, devolve a idade de uma pessoa.

DICA: use o seguinte código para obter o número de anos entre a data atual e a data de aniversário

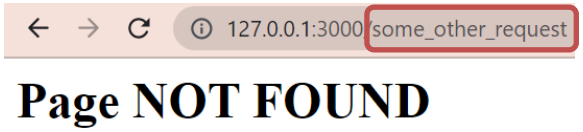
```
let today = new Date();
let birthDate = new Date(dateString); // dateString in format YYYY-MM-DD
let age = today.getFullYear() - birthDate.getFullYear();
```

A idade obtida deve ser decrementada em 1 valor, se o mês atual for inferior ao mês do aniversário ou, se o mês for o mesmo, o dia atual for inferior ao dia atual. Use as funções `getMonth` e `getDate` para obter o mês e o dia de uma data, respetivamente.

Implemente um servidor Web em Node.js que:

- se receber pedidos do tipo `GET /age?dob=YYYY-MM-DD` seja apresentada uma página HTML, com a idade calculada dentro de um cabeçalho `<h1>`. A idade deve ser calculada usando o módulo criado. Assuma sempre que a data inserida está no formato correto e que é uma data válida, anterior à data atual:
- se o URL corresponder a um qualquer outro pedido para o recurso `age`, o servidor deve retornar uma página `INVALID_AGE_REQUEST.html` (ficheiro existente no servidor), com o código de erro 400;

- se o URL corresponder a um outro pedido, o servidor deve retornar uma página `NOT_FOUND.html` (ficheiro existente no servidor), com o código de erro 404.



2. Node + Express (8 valores)

Considere que pretende implementar uma API REST para gerir o *stock* e vendas de uma máquina de *vending*, seguindo os princípios de uma arquitetura MVC e comunicando no formato JSON.

Sendo ainda um protótipo, assuma que os dados irão no futuro ficar alojados numa base de dados, mas para já os dados estarão no código, na forma de *arrays* de objetos:

```
let products = [
    {"id": 1, "name": "Coca-Cola", "price": 2.5, "stock": 5, "type": "drink"},
    {"id": 2, "name": "Pepsi", "price": 2, "stock": 3, "type": "drink"},
    {"id": 3, "name": "Kit Kat", "price": 3, "stock": 2, "type": "snack"}
]

let purchases = [
    {"id": 1, "id_product": 2, "inserted_money": 2, "received_money": 0},
    {"id": 2, "id_product": 1, "inserted_money": 3, "received_money": 0.5},
    {"id": 3, "id_product": 1, "inserted_money": 5, "received_money": 2.5}
]
```

Usando como ferramentas a combinação de Node (servidor) + Express (roteamento), e seguindo uma arquitetura MVC, implemente as rotas para as seguintes funcionalidades:

- Criar um produto, depois de validados os dados fornecidos no corpo do pedido, e em caso de sucesso devolve o URL de acesso ao produto. Algumas possíveis respostas (poderão haver outras):

400 BAD REQUEST – { "message": "Must provide a valid type (drink or snack)" }

400 BAD REQUEST – { "message": "Product name already exists!" }

DICA: o identificador do novo produto deve ser automaticamente calculado pelo tamanho do *array* de produtos.

- Registar a compra de um produto. O corpo do pedido deve incluir o dinheiro inserido na máquina. A rota deve validar se o produto existe, se tem pelo menos um item em stock, e se o montante inserido é suficiente para realizar a compra. Se tudo estiver bem, a compra é registada, a quantidade em stock do produto é atualizada e a resposta ao cliente deve informar o valor do troco.

- Listar os produtos, com a opção de incluir o número total de compras por produto. Exemplo de resposta:

```
[{"id":1, "name":"Coca-Cola", "price":2.5, "stock":5, "type":"drink", "totalPurchases":2},
 {"id":2, "name":"Pepsi", "price":2, "stock":3, "type":"drink", "totalPurchases":1},
 {"id":3, "name":"Kit Kat", "price":3, "stock":2, "type":"snack", "totalPurchases":0}
]
```

- Listar as compras efetuadas para um determinado produto, cujo identificador deve aparecer no URI do pedido. A resposta deve incluir os detalhes do produto e o as compras efetuadas. Exemplo de resposta:

```
{"id":1, "name":"Coca-Cola", "price":2.5, "stock":5, "type":"drink", "purchases": [
    {"id": 2, "inserted_money": 3, "received_money": 0.5},
    {"id": 3, "inserted_money": 5, "received_money": 2.5}]
}
```

BOA SORTE!