Politécnico de Coimbra

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Programação para a Internet II 2023/24

Trabalho prático nº 2

Guilherme Afonso Ferreira Gonçalves

a2022156457

Elaborado em: 15 de dezembro 2023

1. Introdução:

O trabalho prático 2 proposto a ser desenvolvido na Unidade Curricular de Programação para a Internet 2 visa dar continuidade ao trabalho anteriormente desenvolvido (Trabalho Prático 1), neste caso realizando uma API RESTful utilizando a framework slim com os conhecimentos abordados durante as aulas.

Para realizar o trabalho foi necessário, para além dos conhecimentos adquiridos da linguagem PHP, utilizar a extensão Rest Client do VSCode de forma a ser possível a testagem da API. Esta extensão permite fazer pedidos à API ao pressionar-se em “Send Request” no qual irá ser apresentada de seguida a resposta num novo separador.

Uma API, do inglês Application Programming Interface, de uma forma geral é um conjunto de serviços/funções que são disponibilizados para que outros programas possam utilizar de uma forma simples. A API Rest é composta por vários endpoints, no qual cada um corresponde a uma funcionalidade disponibilizada pelo servidor.

Os requisitos propostos para que a realização do trabalho fosse conseguido foram os seguintes: inserir um novo cliente; autenticar o cliente; terminar a sessão do cliente; obter a lista das entidades ativas; obter a lista das entidades ativas e com ofertas para aquele dia; obter detalhes de uma entidade; obter a lista de ofertas ainda disponíveis de uma determinada entidade para aquele dia; obter a lista de todas as ofertas ainda disponíveis para aquele dia; registar a compra de uma oferta; obter a lista compras efetuadas pelo utilizador e por fim cancelar uma compra.

2. Endpoints:

Nesta secção do relatório estarão representados e descritos todos os endpoints propostos para o trabalho, no qual serão justificadas algumas opções tomadas tendo em conta o que se realizou no trabalho e do que foi pedido no enunciado deste.

2.1. [R1] - Inserir novo cliente

Este primeiro requisito proposto permite criar um novo utilizador enviando em formato JSON, o nome, o email, o telefone e a password, no qual depois são verificados e guardados na base de dados. E depois é devolvido uma resposta verificando se o registo foi feito com sucesso, ou se ocorreu algum erro, por exemplo, existir um mesmo mail já em uso. De seguida estarão representados os endpoints com os dados em JSON e a resposta devolvida nos dois casos. Para este caso não foi necessário o token, este que será visto mais à frente.

- Registo correto (email ainda não se encontra em uso):

```
### REQUISITO 1 - Registar Novo Cliente
Send Request
POST {{apiurl}}/api/users/ HTTP/1.1
Content-Type: application/json

{
    "nome" : "Pedro Saramago",
    "email" : "psaramago@gmail.com",
    "telefone" : "960000002",
    "password" : "exemplo2"
}
```

```
HTTP/1.1 200 OK
Host: 127.0.0.1:8080
Date: Fri, 15 Dec 2023 16:03:36 GMT
Connection: close
X-Powered-By: PHP/8.2.10
Content-Type: application/json
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0
Pragma: no-cache

{
  "res": "OK",
  "message": "Cliente inserido com sucesso"
}
```

- Registo incorreto (email já se encontra em uso):

```
### REQUISITO 1 - Registar Novo Cliente
Send Request
POST {{apiurl}}/api/users/ HTTP/1.1
Content-Type: application/json

{
    "nome" : "Paulo Saramago",
    "email" : "psaramago@gmail.com",
    "telefone" : "960000003",
    "password" : "exemplo3"
}
```

```
HTTP/1.1 401 Unauthorized
Host: 127.0.0.1:8080
Date: Fri, 15 Dec 2023 16:11:01 GMT
Connection: close
X-Powered-By: PHP/8.2.10
Content-Type: application/json
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0
Pragma: no-cache

{
  "res": "Erro",
  "message": "Email ja se encontra em uso. Tente outro."
}
```

2.2. [R2] - Autenticação de um cliente

O segundo requisito serve para o cliente se autenticar, e para isso são enviados o email e a password. Primeiro é verificado se o utilizador existe e se está ativo, se estiver então é verificada a password enviada com a password guardada na base de dados. Se estas passwords coincidirem então é feita a autenticação e é devolvido o token que será necessário para se aceder aos outros requisitos.

- Autenticação correta (email, cliente ativo e password estão corretos)

```
### REQUISITO 2 - Autenticação dos clientes
Send Request
PATCH {{apiurl}}/api/users/ HTTP/1.1
Content-Type: application/json

{
    "email" : "psaramago@gmail.com",
    "password" : "exemplo2"
}
```

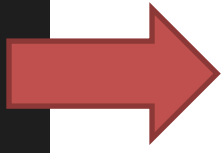
```
HTTP/1.1 200 OK
Host: 127.0.0.1:8080
Date: Fri, 15 Dec 2023 16:23:04 GMT
Connection: close
X-Powered-By: PHP/8.2.10
Content-Type: application/json
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0
Pragma: no-cache

{
  "res": "OK",
  "message": "Cliente autenticado com sucesso",
  "token": "cdbec0a53358be471a4c5f3b8902a0fb"
}
```

- Autenticação incorreta (email ou password incorretos ou cliente inativo) – existem diferentes mensagens, para caso o email não existe ou o cliente estar inativo (será o exemplo seguinte), ou caso o cliente exista mas a password esteja errada será apresentada uma mensagem a dizer que os dados de autenticação estão incorretos.

```
### REQUISITO 2 - Autenticação dos clientes
Send Request
PATCH {{apiurl}}/api/users/ HTTP/1.1
Content-Type: application/json

{
    "email" : "pedrosaramago@gmail.com",
    "password" : "exemplo2"
}
```



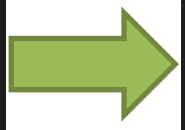
```
HTTP/1.1 401 Unauthorized
Host: 127.0.0.1:8080
Date: Fri, 15 Dec 2023 16:31:33 GMT
Connection: close
X-Powered-By: PHP/8.2.10
Content-Type: application/json
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0
Pragma: no-cache

{
  "res": "Error",
  "message": "Dados de acesso invalidos."
}
```

2.3. [R3] – Terminar sessão de um cliente

Este terceiro requisito permite ao cliente terminar sessão, e neste caso é necessário já ter o token devolvido após autenticar-se. Isto porque, foi utilizado o AuthMiddleware que verifica o estado do token e devolve erro se o token estiver errado. Caso o token esteja correto, o cliente poderá terminar a sua sessão e isto faz com que o token passe a "NULL" na base de dados, sendo depois necessário autenticar-se novamente para ter acesso ao token. A partir deste requisito, todos os outros serão necessários terem o token associado.

```
### REQUISITO 3 - Terminar Sessão de um cliente
Send Request
PATCH {{apiurl}}/api/users/logout/ HTTP/1.1
Authorization : {{auth-token}}
```

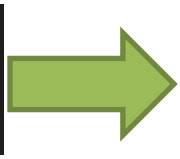



```
{
  "res": "OK",
  "message": "Sessao terminada com sucesso"
}
```


2.4. [R4] – Obter lista das entidades ativas

Neste requisito são apresentadas todas as entidades que se encontram ativas. Caso existam entidades que não estejam ativas, não irão aparecer.

```
### REQUISITO 4 - Obter lista das entidades ativas
Send Request
GET {{apiurl}}/api/entidades/ HTTP/1.1
Authorization : {{auth-token}}
```

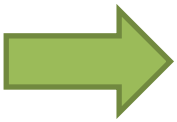



```
"res": "OK",
"data": [
  {
    "id": 10,
    "nome": "Padaria Fresquinha 2",
    "logotipo": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "descricao": "P\u00e3o",
    "morada": "Rua do P\u00e3o",
    "telefone": "999999999",
    "email": "padaria123@gmail.com"
  },
  {
    "id": 11,
    "nome": "Pastelaria",
    "logotipo": "52369d3a8ee83b5fd9d176c317378b64.jpg",
    "descricao": "Bolos",
    "morada": "Rua dos Bolos",
    "telefone": "235000000",
    "email": "bolos@gmail.com"
  }
]
```

2.5. [R5] – Obter lista das entidades ativas e com ofertas para o dia

Este quinto requisito apresenta a lista das entidades ativas como se fez anteriormente só que apenas as que têm ofertas para o dia. Ou seja, caso estejam ativas, mas não tenham ofertas para o dia, não serão apresentadas entidades. Para este requisito foi necessário ter conhecimento da Unidade Curricular de Base de Dados, visto que optei por fazer junções entre as tabelas entidade e oferta para poder ter acesso às entidades com as ofertas apenas para aquele dia.

```
### REQUISITO 5 - Obter lista das entidades ativas com ofertas para o dia
Send Request
GET {{apiurl}}/api/entidades/comOfertas/ HTTP/1.1
Authorization : {{auth-token}}
```

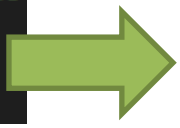



```
"res": "OK",
"data": [
  {
    "id": 11,
    "nome": "Pastelaria",
    "logotipo": "52369d3a8ee83b5fd9d176c317378b64.jpg",
    "descricao": "Bolos",
    "morada": "Rua dos Bolos",
    "telefone": "235000000",
    "email": "bolos@gmail.com"
  }
]
```


2.6. [R6] – Obter detalhes de uma entidade

Neste requisito será necessário passar o id da entidade que se pretende obter os detalhes. Depois disso serão apresentados os dados da respetiva entidade. Não é verificado se a entidade está ativa pois não era pedido no enunciado, no entanto, estes pontos serão justificados mais à frente no relatório. O seguinte exemplo é feito com a entidade com o id 10.

```
### REQUISITO 6 - Obter detalhes de uma entidade
Send Request
GET {{apiurl}}/api/entidades/10/ HTTP/1.1
Authorization : {{auth-token}}
```




```
"res": "OK",
"data": [
  {
    "id": 10,
    "nome": "Padaria Fresquinha 2",
    "logotipo": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "descricao": "P\u00e3o",
    "morada": "Rua do P\u00e3o",
    "telefone": "999999999",
    "email": "padaria123@gmail.com"
  }
]
```

2.7. [R7] – Obter lista de ofertas disponíveis de uma entidade para aquele dia

O requisito 7 vai apresentar todas as ofertas disponíveis para a entidade que foi passada através do id. O seguinte exemplo é feito com a entidade com o id 10. Caso as ofertas pertençam a outro dia, não serão apresentadas quaisquer ofertas.

```
### REQUISITO 7 - Obter lista de ofertas disponiveis de uma entidade para aquele dia
Send Request
GET {{apiurl}}/api/entidades/ofertas/10/ HTTP/1.1
Authorization : {{auth-token}}
```


```
"res": "OK",
"data": [
  {
    "id": 4,
    "entidade_id": 10,
    "nome": "P\u00e3o",
    "descricao": "P\u00e3o",
    "foto": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "pre\u00e7o": 2,
    "data": "2023-12-15",
    "disponivel": 1
  }
]
```


2.8. [R8] – Obter lista de todas ofertas disponíveis para aquele dia

Este requisito apresentará todas as ofertas disponíveis de quaisquer entidades para aquele determinado dia. Neste caso serão apresentadas tanto as informações das ofertas, assim como as informações da respetiva entidade.

```
### REQUISITO 8 - Obter lista de todas as ofertas ainda disponíveis para aquele dia
Send Request
GET {{apiurl}}/api/entidades/ofertas/ HTTP/1.1
Authorization : {{auth-token}}
```


```
"res": "OK",
"data": [
  {
    "id": 4,
    "entidade_id": 10,
    "Nome_Oferta": "P\u00e3o",
    "Descricao_Oferta": "P\u00e3o",
    "foto": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "pre\u00e7o": 2,
    "Nome_Entidade": "Padaria Fresquinha 2",
    "Descricao_Entidade": "P\u00e3o",
    "logotipo": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "morada": "Rua do P\u00e3o",
    "telefone": "999999999",
    "email": "padaria123@gmail.com",
    "data": "2023-12-15"
  }
]
```

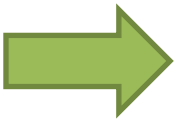
2.9. [R9] – Registar a compra de uma oferta

Neste requisito o cliente poderá registar a compra de uma determinada oferta, enviando o id da oferta que pretende comprar. Primeiramente é necessário verificar se a oferta ainda está disponível e se o cliente também está disponível. Se tudo estiver correto a oferta será inserida na tabela compra da base de dados e a disponibilidade da oferta passará a 0, visto que esta foi comprada, tornando-se indisponível.

- Compra de uma oferta disponível no dia

```
### REQUISITO 9 - Registar a compra de uma oferta
Send Request
POST {{apiurl}}/api/entidades/ofertas/comprar HTTP/1.1
Authorization : {{auth-token}}
Content-Type: application/json

{
    "id_oferta" : "4"
}
```

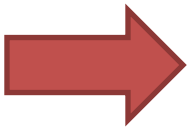



```
{
  "res": "OK",
  "message": "Compra realizada com sucesso!"
}
```


- Compra de uma oferta indisponível no dia

```
### REQUISITO 9 - Registar a compra de uma oferta
Send Request
POST {{apiurl}}/api/entidades/ofertas/comprar HTTP/1.1
Authorization : {{auth-token}}
Content-Type: application/json

{
    "id_oferta" : "2"
}
```

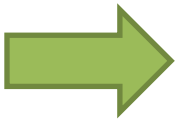



```
{
  "res": "Error",
  "message": "Oferta nao encontrada."
}
```


2.10. [R10] – Lista de compras efetuadas pelo utilizador

Aqui serão apresentadas todas as compras efetuadas pelo cliente.

```
### REQUISITO 10 - Obter lista compras efetuadas pelo utilizador
Send Request
GET {{apiurl}}/api/users/compras/ HTTP/1.1
Authorization : {{auth-token}}
```



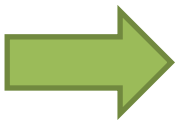

```
"res": "OK",
"data": [
  {
    "id_compra": 6,
    "id_oferta": 4,
    "entidade_id": 10,
    "Nome_Oferta": "P\u00e3o",
    "Descricao_Oferta": "P\u00e3o",
    "foto": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "pre\u00e7o": 2,
    "Nome_Entidade": "Padaria Fresquinha 2",
    "Descricao_Entidade": "P\u00e3o",
    "logotipo": "20d0c021fee7ceafb83e6d44dd04bdde.jpg",
    "morada": "Rua do P\u00e3o",
    "telefone": "999999999",
    "email": "padaria123@gmail.com",
    "data": "2023-12-15"
  }
]
```

2.11. [R11] – Cancelar uma compra

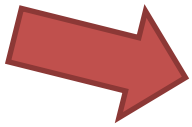
Para este último requisito, o cliente pode cancelar uma compra previamente feita no mesmo dia, necessitando de enviar o id da compra. Após fazer isto, a compra será cancelada e essa determinada oferta passará outra vez a estar disponível naquele determinado dia. Caso a compra seja de outro dia, será apresentado um erro.

```
### REQUISITO 11 - Cancelar uma compra
Send Request
PATCH {{apiurl}}/api/users/compras/ HTTP/1.1
Authorization : {{auth-token}}
Content-Type: application/json

{
    "id_compra" : "6"
}
```



```
"res": "OK",
"message": "Compra cancelada com sucesso!"
```



```
{
  "res": "Error",
  "message": "Compra nao encontrada."
}
```

(Caso a compra já fosse de outro dia)

3. Justificações e Conclusão:

Tal como foi dito anteriormente ao longo do relatório, existiram alguma coisas que não foram verificadas nas rotas em todos os endpoints. Nomeadamente a verificação do cliente estar ativo ou não e da oferta. Apenas se fez para os requisitos que o docente colocou como indicação no enunciado, no entanto, se estes fossem explicitamente colocados em todos os requisitos seria seguir a mesma lógica do código implementado na verificação destes pontos, nomeadamente feito nos atributos: 2 – para verificar se o cliente está ativo ou não, e nos atributos 9 e 10 – que pedem para verificar se a oferta e o cliente estão disponíveis.

Para o último requisito foi necessário acrescentar um atributo “id_compra” à tabela compra da base de dados, dado que no enunciado era pedido para cancelar a compra através do id da compra, e visto que inicialmente este atributo não existia foi necessário adicioná-lo.

Neste mesmo requisito é pedido para utilizarmos a operação PATCH, o que foi feito no trabalho, mas talvez não fosse a melhor opção visto que ao realizar a compra teve de se fazer mais do que um Update, o que não é o ideal para o PATCH. Outra verificação a apontar, é que como não foi esclarecido no enunciado o que se fazia aos atributos “pago”, “levantado” e “ativo”, ao realizar a compra entendeu-se que todos os atributos ficariam como “1”, indo de encontro ao pensamento de

que a compra foi paga, levantada e ativa no sentido de que foi realizada. Ao cancelar a compra estes atributos passam todos a “0” e a disponibilidade da oferta, tal como dito nos endpoints volta a 1.

Concluindo, este trabalho foi realizado com sucesso, tendo em conta todos os requisitos pedidos para a realização do mesmo. Tal como foi descrito anteriormente todos os endpoints funcionaram corretamente, tanto na parte da verificação dos mesmo estarem corretos e apresentar a devida mensagem, assim como para os casos da verificação estar errada.

Para este trabalho foi essencial ir às aulas em foram abordados estes temas e a realização de alguns exercícios propostos pelo professor também durante as aulas, no qual possibilitaram perceber e aproveitar parte desse código para a realização deste projeto. Também foi necessário, tal como já foi dito anteriormente, ter conhecimentos da Unidade Curricular de Base de Dados, por exemplo, na utilização de junções entre tabelas.

Junto com este relatório será enviada uma pasta com os ficheiros utilizados para o trabalho. O ficheiro “routes.php” que se encontra dentro da pasta “app” é o ficheiro que contém o código onde estão definidos os endpoints. O ficheiro “testes.http” é onde se pode testar os pedidos.