

# PLATAFORMA DE APOIO A CENTRO DE TESTES COVID-19

Programação em Ambiente Web

Grupo 39

Diogo Costa 8170455 | Luis Marques 8170485 | Luis Teixeira 8110156

## Índice

1	Introdução .....	4
1.1	Contextualização do Documento.....	4
1.2	Apresentação do caso de estudo.....	4
2	Objetivo.....	5
2.1	Ferramentas e tecnologias utilizadas .....	5
3	Definições, Acrónimos e Abreviaturas.....	6
4	Visão geral do projeto .....	7
4.1	Objetivo do Software.....	7
5	Implementação da API REST.....	8
5.1	Definição .....	8
5.2	Definição de tabelas da base de dados.....	8
5.3	Operações CRUD .....	9
5.4	Users.....	10
5.4.1	Registar Utilizador (POST).....	10
5.4.2	Listar Utilizadores (GET) .....	11
5.4.3	Listar Utilizadores pelo ID (GET) .....	12
5.4.4	Atualizar Utilizador através do ID (PUT) .....	13
5.4.5	Remover Utilizador através do ID (DELETE).....	0
5.5	Testes .....	1
5.5.1	Inserir Teste (POST) .....	2
5.5.2	Listar Teste (GET).....	3
5.5.3	Listar Teste pelo ID (GET).....	4
5.5.4	Atualizar Teste pelo ID (PUT) .....	5
5.5.5	Remover Teste através de ID (DELETE).....	6
5.6	Authenticate (POST) .....	6
5.7	Documentação .....	7
6	FrontEnd (Angular) .....	7
6.1	Introdução .....	7
6.2	Estrutura.....	8
6.2.1	Services.....	8
6.2.2	Models.....	9
6.2.3	Components .....	9
7	Conclusão .....	14
8	GitLab .....	15
9	Bibliografia .....	16

## Histórico Alterações Documento

Date	Description	Version
02/05/2020	<ul style="list-style-type: none"><li>Contextualização do Documento</li><li>Apresentação do Caso de Estudo</li><li>Definições, Acrónimos e Abreviaturas</li></ul>	1.0
02/06/2020	<ul style="list-style-type: none"><li>Descrição da implementação da REST API e Front End</li></ul>	1.1
12/06/2020	<ul style="list-style-type: none"><li>Conclusões finais</li></ul>	2.0

# 1 Introdução

## 1.1 Contextualização do Documento

Este documento descreve o processo de implementação do projeto desenvolvido no âmbito da unidade curricular ‘Programação em Ambiente Web’, e fornece uma visão geral sobre o produto final.

## 1.2 Apresentação do caso de estudo

Devido ao panorama atual e à crescente necessidade de dar resposta a um elevado número de pedidos de testes de despiste à COVID.19, é necessário montar um novo centro de análises regional para a realização dos testes. Este documento descreve todo trabalho realizado, durante o desenvolvimento de uma plataforma web, que dará suporte ao novo centro de análises.

## 2 Objetivo

O trabalho descrito neste documento, tem como objetivo o desenvolvimento de uma plataforma web, para processamento de pedidos de teste de diagnóstico, agendamento de testes e registo de resultados de um centro de testes despiste e imunização à Covid-19. Para isso, será necessário ir de encontro a uma série de requisitos, que serão descritos numa secção posterior deste documento.

### 2.1 Ferramentas e tecnologias utilizadas

No processo de desenvolvimento do projeto, recorreremos às seguintes ferramentas:

- NodeJS e a framework ExpressJS
- Postman
- MongoDB
- Angular
- VSCode
- Git e Github/Gitlab

### 3 Definições, Acrónimos e Abreviaturas

*REST* - Representational State Transfer

*API* - Application Programming Interface

## 4 Visão geral do projeto

### 4.1 Objetivo do Software

A aplicação web desenvolvida neste trabalho, foi idealizada para agilizar o processo de pedidos de teste de diagnóstico, agendamento de testes e registo do histórico de cada paciente testado no centro de análises. No pedido de teste de diagnóstico, o utilizador deverá indicar se foi encaminhado pela linha Saúde24, se pertence a um grupo de risco ou se trabalha em locais de risco. Após a realização do teste, a ficha do utilizador será alterada para 'teste realizado'. Quando obtidos os resultados, será possível registar o resultado clínico na ficha do pedido do utilizador anexando um ficheiro (pdf) com os resultados clínicos e adicionando o resultado ao pedido. Por defeito, todos os pacientes estarão classificados como 'suspeito', e consoante os resultados de testes forem inseridos no histórico do paciente, a classificação será atualizada.

## 5 Implementação da API REST

### 5.1 Definição

API (Application Program Interface): é um conjunto de funções e procedimentos que permitem a criação de aplicações que acedem a recursos ou dados de um sistema, aplicação ou outro serviço REST API: definição de apis baseadas no protocolo HTTP universal.

Os métodos HTTP mais usados em serviços REST, são GET, POST, PUT, DELETE e que predefinem o CRUD em HTTP.

Partindo do que foi descrito anteriormente elaboramos então o CRUD da nossa API para responder às nossas necessidades.

### 5.2 Definição de tabelas da base de dados

A primeira abordagem foi delinear as tabelas que necessitávamos para responder ao nosso problema.

Para tal, utilizamos a ferramenta de base de dados abordada nas aulas, nomeadamente o MongoDB.

Possuímos duas tabelas na nossa base de dados. A tabela “Test” constituída pelos campos:



- saude24 (boolean)
- risk\_group (boolean)
- risk\_local (boolean)
- information (String)
- user\_state (String)
- test\_state (String)
- test\_result (String)
- date (date)
- pdf
- user\_id (mongoose.Schema.Types.ObjectId)



E a tabela “User” é constituída pelos seguintes campos:

- name (String)
- email (String)
- password (String)
- Role (String)

Estas tabelas foram implementadas na pasta “models”, sendo essa pasta por padrão onde ficam implementadas as tabelas.

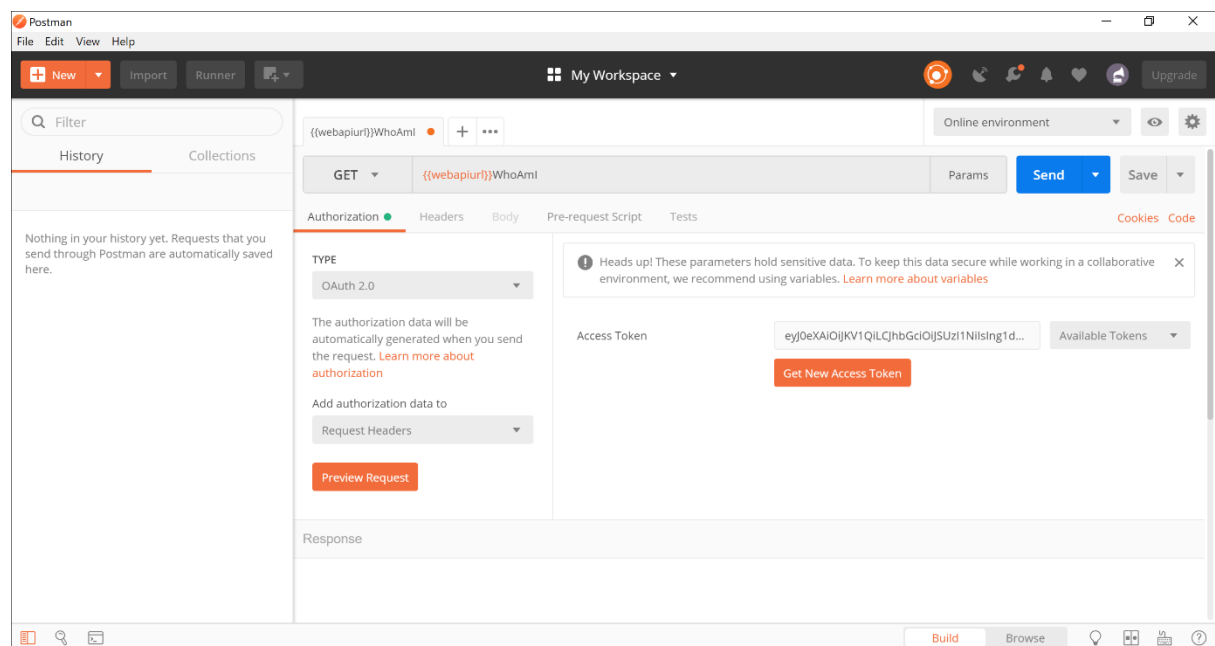
### 5.3 Operações CRUD

Depois de já definidos os “models” elaboramos as operações CRUD ( Create, Remove ,Update and Delete) dos “Users” e “Tests”.

Todas estas operações foram inseridas na pasta “controllers” onde se encontram todas as ações do controlador.

Posteriormente, para testar os nossos pedidos REST utilizamos o Postman sendo uma ferramenta bastante vantajosa, que permite:

- Definir parâmetros na área Header e Body dos pedidos HTTP;
- Verificar as respostas do servidor ;
- Criar projetos para teste e validação de APIs.



## 5.4 Users

### 5.4.1 Registar Utilizador (POST)

Neste pedido é feito o registo do utilizador, ou seja, após este pedido o utilizador é inserido na base de dados.

The screenshot shows a REST client interface with a tab titled "POST User Register". The request is a POST to the URL "localhost:3000/api/v1/users/register". The request body is set to "x-www-form-urlencoded" and contains the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	joao	
<input checked="" type="checkbox"/> email	joao@hotmail.com	
<input checked="" type="checkbox"/> password	joaozinho123	
Key	Value	Description

The response status is "200 OK" with a time of "205 ms" and a size of "283 B". The response body is displayed in JSON format:

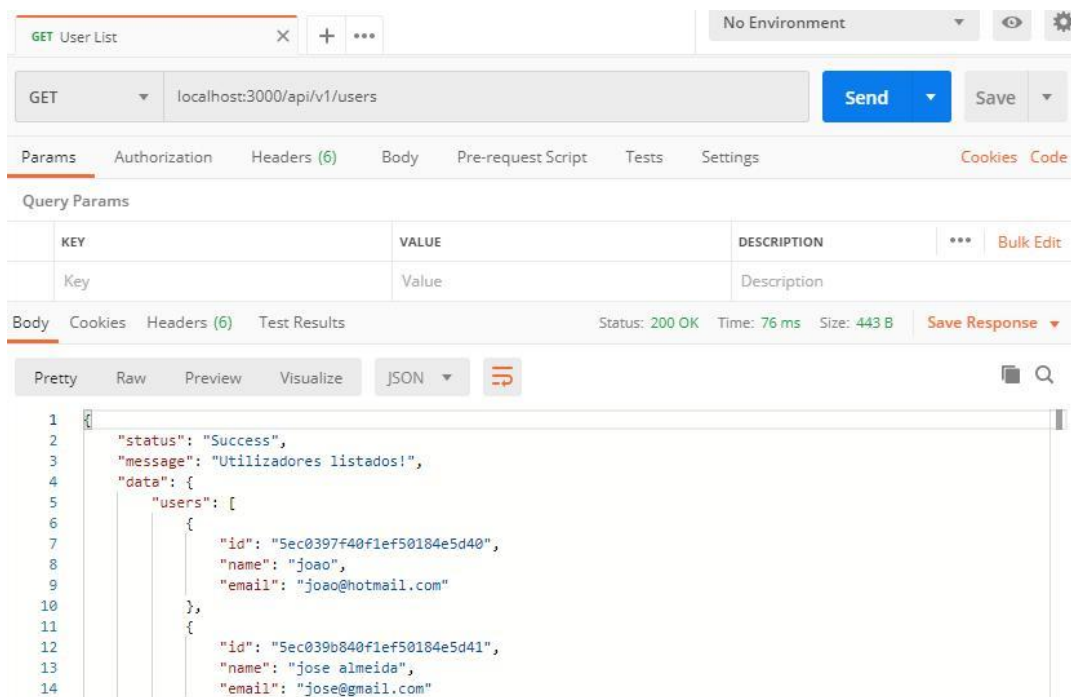
```

1 {
2   "status": "Success",
3   "message": "User added Successfully!!!",
4   "data": null
5 }

```

### 5.4.2 Listar Utilizadores (GET)

Após o pedido POST é possível listar os utilizadores já existentes na base de dados. Se não tiver qualquer utilizador adicionado, este responde com uma lista vazia.



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** localhost:3000/api/v1/users
- Status:** 200 OK
- Time:** 76 ms
- Size:** 443 B

The response body is displayed in JSON format:

```
1 {
2   "status": "Success",
3   "message": "Utilizadores listados!",
4   "data": {
5     "users": [
6       {
7         "id": "5ec0397f40f1ef50184e5d40",
8         "name": "joao",
9         "email": "joao@hotmail.com"
10      },
11      {
12        "id": "5ec039b840f1ef50184e5d41",
13        "name": "jose almeida",
14        "email": "jose@gmail.com"
15      }
16    ]
17  }
18 }
```

### 5.4.3 Listar Utilizadores pelo ID (GET)

É possível listar o utilizador com o respetivo ID com o Model.findById()

The screenshot shows a REST client interface with the following details:

- Request Method:** GET
- URL:** localhost:3000/api/v1/users/5ec0397f40f1ef50184e5d40
- Response Status:** 200 OK
- Response Time:** 46 ms
- Response Size:** 356 B
- Response Body (JSON):**

```

1 {
2   "status": "Success",
3   "message": "Utilizador listado!",
4   "data": {
5     "user": {
6       "id": "5ec0397f40f1ef50184e5d40",
7       "name": "joao",
8       "email": "joao@hotmail.com"
9     }
10  }
11 }

```

#### 5.4.4 Atualizar Utilizador através do ID (PUT)

Pedido REST que através do `Model.findByIdAndUpdate()` atualiza os campos fornecendo o ID que é gerado pelo Mongo ao ser inserido na base de dados.

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** localhost:3000/api/v1/users/5ec0397f40f1ef50184e5d40
- Body Type:** x-www-form-urlencoded
- Body Data:**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	joao ferreira	
<input checked="" type="checkbox"/> email	joao.ferreira@hotmail.com	
<input type="checkbox"/> password	jose321	
- Status:** 200 OK
- Time:** 63 ms
- Size:** 293 B
- Response Body (JSON):**

```
{  "status": "Success",  "message": "Utilizador atualizado com sucesso!!!",  "data": null}
```

### 5.4.5 Remover Utilizador através do ID (DELETE)

Neste pedido é usado o `Model.findByIdAndDelete()` para remover o utilizador da nossa base de dados, sendo o ID fornecido.

The screenshot shows a REST client interface with a tab titled "DEL User Delete". The request method is "DELETE" and the URL is "localhost:3000/api/v1/users/5ec031c9a83e994d5ebef0cd". The response status is "200 OK", time is "43 ms", and size is "290 B". The response body is displayed in JSON format:

```

1 {
2   "status": "Success",
3   "message": "Utilizador apagado com sucesso!!!",
4   "data": null
5 }

```

## 5.5 Testes

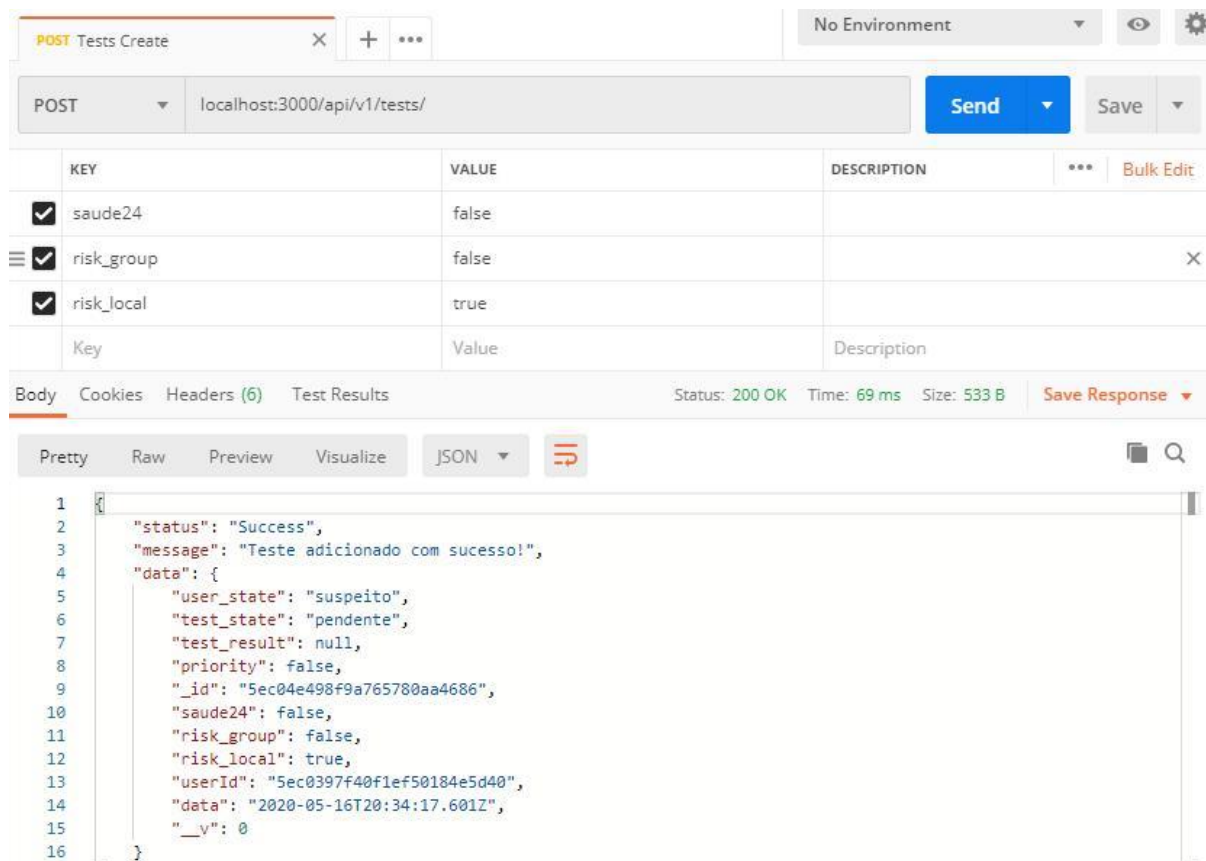
Como já foi dito anteriormente, esta é uma “private route”, ou seja, só fornecendo o JWT é que é possível realizar as operações CRUD apresentadas posteriormente. Todas estas operações terão de passar no ficheiro `validateuser.js` da pasta `middleware`.

```
You, a few seconds ago | 2 authors (Diogo Costa and others)
1 // private route
2 app.use('/testes', validateUser, testes);
3
4 function validateUser(req, res, next) {
5   jwt.verify(
6     req.headers['x-access-token'],
7     req.app.get('secretKey'),
8     function (err, decoded) {
9       if (err) {
10         res.json({ status: 'error', message: err.message, data: null });
11       } else {
12         // add user id to request
13         req.body.userId = decoded.id;
14         next();
15       }
16     }
17   );
18 }
```

You, a few seconds ago • Uncommitted changes

### 5.5.1 Inserir Teste (POST)

Neste pedido é feito o registo do teste, ou seja, após este pedido o teste é inserido na base de dados.



POST Tests Create

POST localhost:3000/api/v1/tests/ Send Save

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	saude24	false	
<input checked="" type="checkbox"/>	risk_group	false	
<input checked="" type="checkbox"/>	risk_local	true	
	Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 69 ms Size: 533 B Save Response

Pretty Raw Preview Visualize JSON

```

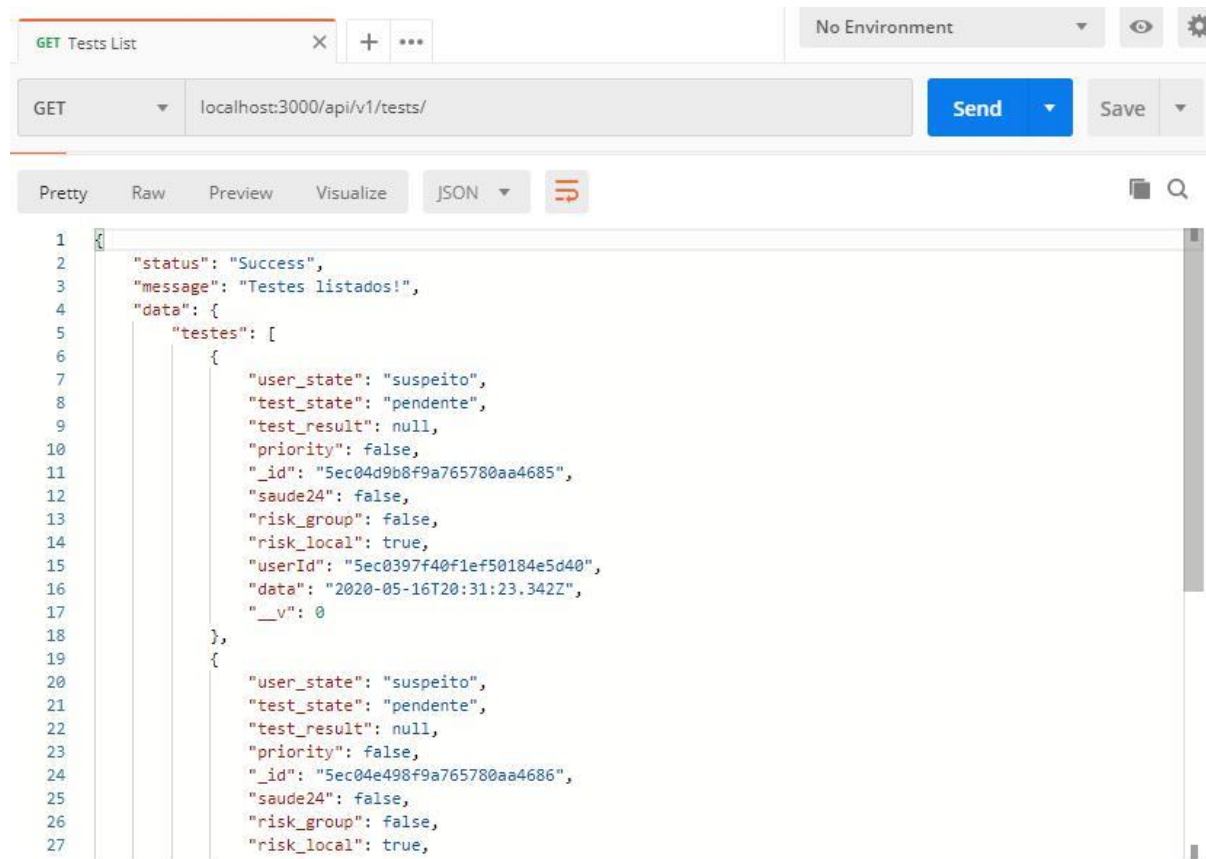
1 {
2   "status": "Success",
3   "message": "Teste adicionado com sucesso!",
4   "data": {
5     "user_state": "suspeito",
6     "test_state": "pendente",
7     "test_result": null,
8     "priority": false,
9     "_id": "5ec04e498f9a765780aa4686",
10    "saude24": false,
11    "risk_group": false,
12    "risk_local": true,
13    "userId": "5ec0397f40f1ef50184e5d40",
14    "data": "2020-05-16T20:34:17.601Z",
15    "__v": 0
16  }

```



### 5.5.2 Listar Teste (GET)

Após o pedido POST é possível listar os testes feitos por um determinado utilizador já existentes na base de dados. Se não tiver qualquer utilizador adicionado, este responde com uma lista vazia.



The screenshot shows a REST client interface with a tab titled "GET Tests List". The request method is "GET" and the URL is "localhost:3000/api/v1/tests/". The response is displayed in JSON format, showing a successful status and a list of two test records.

```
1 {
2   "status": "Success",
3   "message": "Testes listados!",
4   "data": {
5     "testes": [
6       {
7         "user_state": "suspeito",
8         "test_state": "pendente",
9         "test_result": null,
10        "priority": false,
11        "_id": "5ec04d9b6f9a765780aa4685",
12        "saude24": false,
13        "risk_group": false,
14        "risk_local": true,
15        "userId": "5ec0397f40f1ef50184e5d40",
16        "data": "2020-05-16T20:31:23.342Z",
17        "__v": 0
18      },
19      {
20        "user_state": "suspeito",
21        "test_state": "pendente",
22        "test_result": null,
23        "priority": false,
24        "_id": "5ec04e498f9a765780aa4686",
25        "saude24": false,
26        "risk_group": false,
27        "risk_local": true,
```

### 5.5.3 Listar Teste pelo ID (GET)

É possível listar o teste com o respetivo ID com o Model.findById()

GET Tests ListById

GET localhost:3000/api/v1/tests/5ec04f2d8f9a765780aa4687

Send Save

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJp...	
Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 135 ms Size: 531 B Save Response

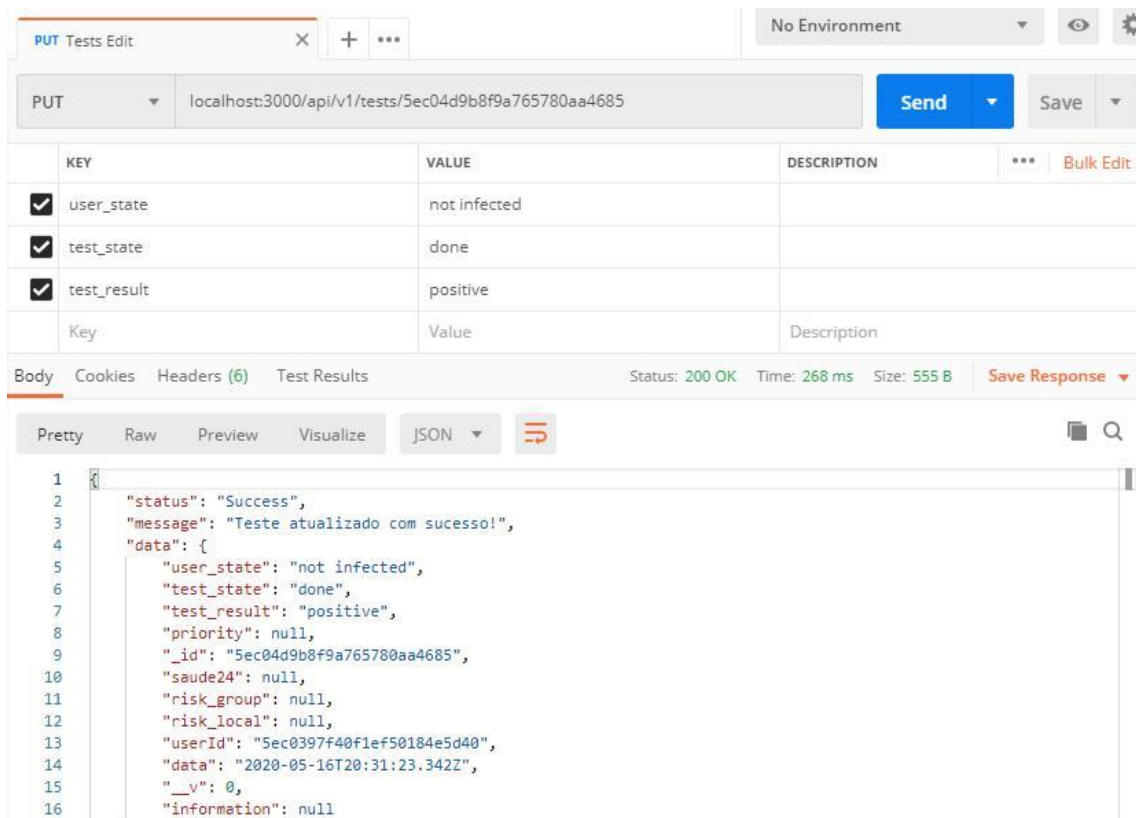
Pretty Raw Preview Visualize JSON

```

1 {
2   "status": "Success",
3   "message": "Testes encontrado!",
4   "data": {
5     "testes": {
6       "user_state": "suspeito",
7       "test_state": "pendente",
8       "test_result": null,
9       "priority": false,
10      "_id": "5ec04f2d8f9a765780aa4687",
11      "saude24": true,
12      "risk_group": true,
13      "risk_local": true,
14      "userId": "5ec0397f40f1ef50184e5d40",
15      "data": "2020-05-16T20:38:05.743Z",
16      "__v": 0
17    }
18  }
19 }
```

#### 5.5.4 Atualizar Teste pelo ID (PUT)

Pedido REST que através do `Model.findByIdAndUpdate()` atualiza os campos fornecendo o ID que é gerado pelo Mongo ao ser inserido na base de dados.



The screenshot shows a REST client interface with a PUT request to `localhost:3000/api/v1/tests/5ec04d9b8f9a765780aa4685`. The request body is a JSON object with the following fields:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> user_state	not infected	
<input checked="" type="checkbox"/> test_state	done	
<input checked="" type="checkbox"/> test_result	positive	
Key	Value	Description

The response status is 200 OK, with a time of 268 ms and a size of 555 B. The response body is a JSON object:

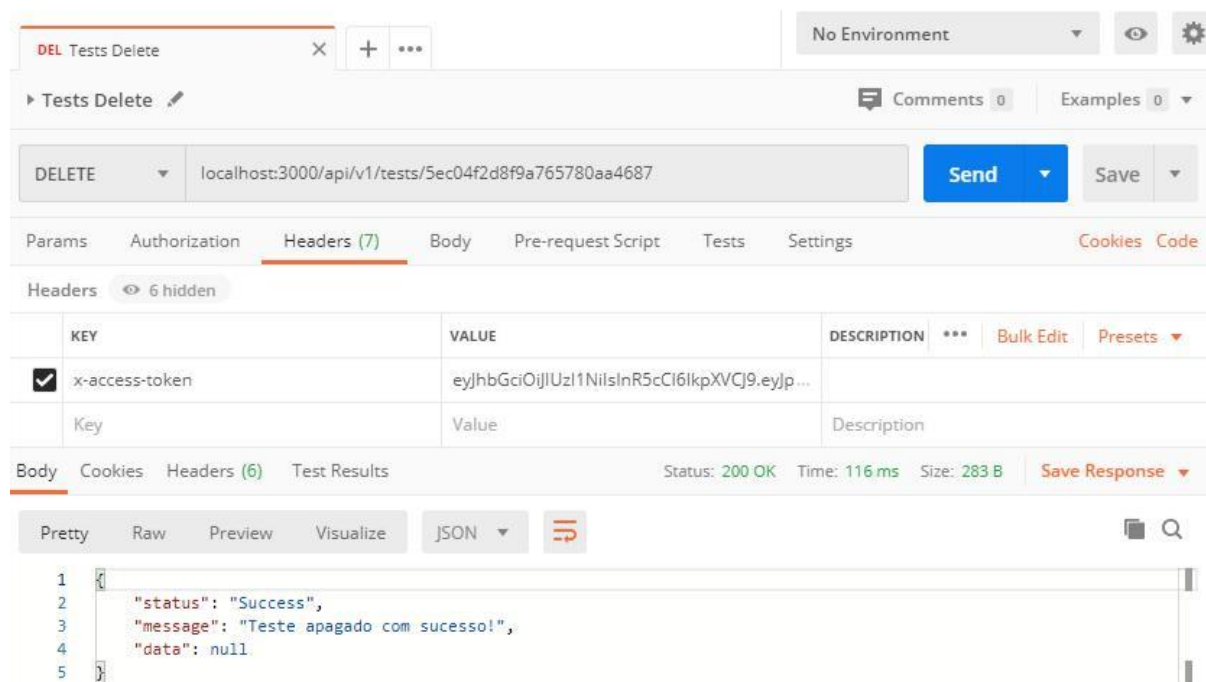
```

1 {
2   "status": "Success",
3   "message": "Teste atualizado com sucesso!",
4   "data": {
5     "user_state": "not infected",
6     "test_state": "done",
7     "test_result": "positive",
8     "priority": null,
9     "_id": "5ec04d9b8f9a765780aa4685",
10    "saude24": null,
11    "risk_group": null,
12    "risk_local": null,
13    "userId": "5ec0397f40f1ef50184e5d40",
14    "data": "2020-05-16T20:31:23.342Z",
15    "__v": 0,
16    "information": null
  }
}

```

### 5.5.5 Remover Teste através de ID (DELETE)

Neste pedido é usado o `Model.findByIdAndDelete()` para remover o teste da nossa base de dados, sendo o ID fornecido.



The screenshot shows a REST client interface with a DELETE request to `localhost:3000/api/v1/tests/5ec04f2d8f9a765780aa4687`. The request has an `x-access-token` header. The response is a JSON object: `{\"status\": \"Success\", \"message\": \"Teste apagado com sucesso!\", \"data\": null}`.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJp...	
Key	Value	Description

```

1 {
2   "status": "Success",
3   "message": "Teste apagado com sucesso!",
4   "data": null
5 }

```

## 5.6 Authenticate (POST)

Para este pedido utilizamos uma ferramenta lecionada nas sessões desta unidade curricular nomeadamente o JWT.

JSON Web Token (JWT) é um open standard (RFC 7519) que define um método compacto e autocontido para transmitir com segurança informações entre as partes num objeto JSON.



Quando os tokens são assinados usando pares de chaves pública/privada, a assinatura também certifica que a parte que é proprietária da chave privada é aquela que a assinou.

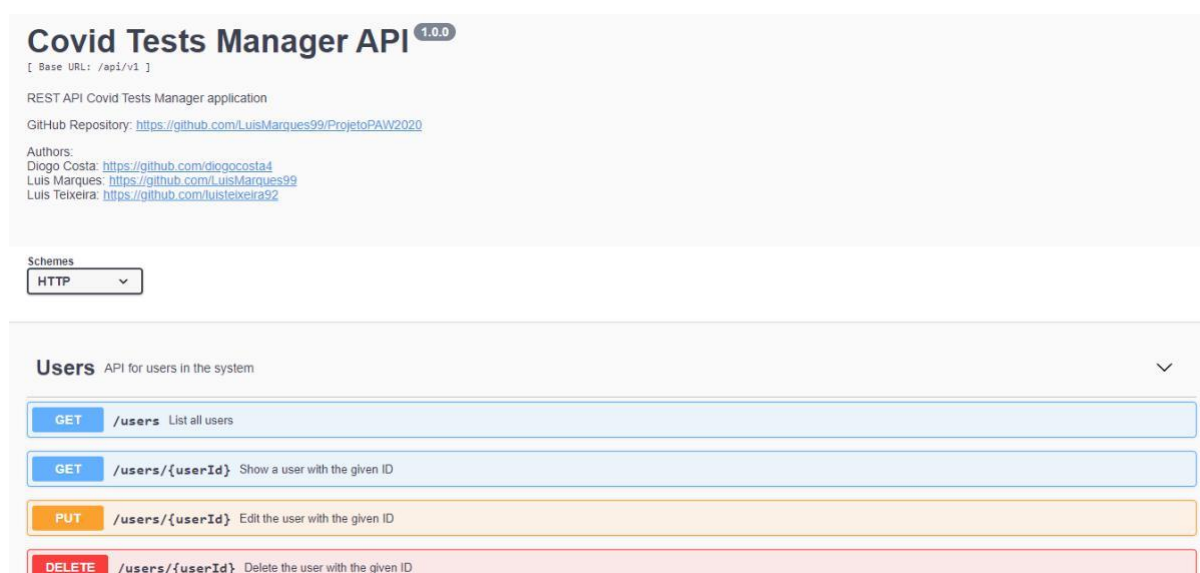
## 5.7 Documentação

Para a documentação dos nossos pedidos REST utilizamos o SWAGGER, que foi a ferramenta de documentação abordada nas aulas, da qual tiramos partido para facilitar a execução dos testes aos nossos pedidos.



Na imagem em baixo podemos ver todos os pedidos REST dos “Users” sendo que em cada um é possível enviar/receber os respetivos parâmetros.

Em todos os pedidos PUT e POST que é necessário inserir dados no body neste momento ainda não está funcional pelo que só poderá se testado na aplicação POSTMAN.



## 6 FrontEnd (Angular)

### 6.1 Introdução

Para a parte do cliente utilizamos a Framework Angular. Angular é uma plataforma e framework para construção da interface de aplicações usando HTML, CSS e TypeScript, criada pelos desenvolvedores da Google.

Dentre os principais, podemos destacar os componentes, templates, pastas, roteamento, módulos, serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas, como a execução de testes unitários de uma aplicação.

## 6.2 Estrutura

### 6.2.1 Services

Para dar resposta aos nossos pedidos HTTP feitos ao servidor criamos serviços em Angular nomeadamente:

- **Serviço para os Users (users.service.ts)**

Serviço para responder aos pedidos READ, UPDATE e DELETE dos utilizadores

- **Serviço para os Testes (tests.service.ts)**

Serviço para responder aos pedidos CREATE, READ, UPDATE e DELETE dos testes

- **Serviço para Autenticação (auth.service.ts)**

Serviço para responder aos pedidos CREATE, READ dos utilizadores nomeadamente o registar e fazer login do utilizador.

- **Serviço para o Token (token-interceptor.service.ts)**

Serviço para o token pois precisamos de colocar o esquema de autenticação HTTP: Bearer Authentication.

## 6.2.2 Models

- **Model para o User**

Classe com todos os campos do utilizador e seus respetivos tipos.

- **Model para o Teste**

Classe com todos os campos do teste e seus respetivos tipos.

## 6.2.3 Components

Componentes que contêm um ficheiro TypeScript, responsável por uma parte da lógica do projeto.

### 6.2.3.1 Sign in (Login)

Covid Tests Manager   Visualizar Testes   Inserir Teste   Ver os meus testes   ADMIN   Login   Registrar

Login

E-mail e/ou password incorretos

Login

### 6.2.3.2 Sign-up (Registo de utilizador)

Covid Tests Manager Visualizar Testes Inserir Teste Ver os meus testes ADMIN Log'in Registrar

Register

Campo Obrigatório

Campo Obrigatório

Campo obrigatório

Registrar

### 6.2.3.3 User Management

Componente de gestão de utilizadores, não conseguimos adaptar todos os roles que idealizamos por isso todos os utilizadores do sistema são considerados ADMIN, mesmo não estando registados.

Covid Tests Manager Visualizar Testes Inserir Teste Ver os meus testes ADMIN Sair

Id utilizador: 5ec0397f40f1ef50184e5d40

Nome: joao ferreira  
E-mail: joao.ferreira@hotmail.com  
Role:

Apagar utilizador

Id utilizador: 5ee1299719a9f5193f143ce5

Nome: Luis Marques  
E-mail: luisserafim99@gmail.com  
Role:

Apagar utilizador



### 6.2.3.4 Listar Testes

Covid Tests Manager Visualizar Testes Inserir Teste Ver os meus testes ADMIN Sair

Visualizar Testes Pendentes

Visualizar Testes Pendentes Prioritários

Id utilizador: 5ec0397f40f1ef50184e5d40

Teste nº: 1

O teste está no estado: pendente

Ver Detalhes

Id utilizador: 5ee2ea9a4f9cc11e50b90550

Teste nº: 2

O teste está no estado: pendente

### 6.2.3.5 Criar Teste

Teste Diagnóstico

Alguma vez contactou a linha Saúde 24?

☐ Sim
 ☐ Não

Faz parte de um grupo de risco?

☐ Sim
 ☐ Não

[Veja aqui quais são os grupos de risco](#)

Esteve num local de risco nas 2 últimas semanas?

☐ Sim
 ☐ Não

Observações que gostaria de mencionar

### 6.2.3.6 Listar Testes de Utilizador

Covid Tests Manager Visualizar Testes Inserir Teste Ver os meus testes ADMIN Sair

Seu Id: 5ee1299719a9f5193f143ce5

O seu teste está no estado: pendente  
O resultado do seu teste é: suspeito  
A date do exame foi:

Se o seu teste estiver no estado pendente ignore a data do exame

Seu Id: 5ee1299719a9f5193f143ce5

O seu teste está no estado: pendente  
O resultado do seu teste é: suspeito  
A date do exame foi:

Se o seu teste estiver no estado pendente ignore a data do exame

### 6.2.3.7 Apagar Utilizador

×

Eliminar Utilizador

Tem a certeza que deseja eliminar o utilizador?

Cancelar

Eliminar

### 6.2.3.8 Detalhes do Teste

Covid Tests Manager Visualizar Testes Inserir Teste Ver os meus testes ADMIN

Sair

#### Detalhes do Teste

Id do Utilizador: 5ee106f304e3d72828315307

O utilizador está no estado: infetado

O resultado do teste é: negativo

#### Outras Informações

Observações: Nao tenho mais nada a acrescentar.

Contactou a linha Saúde24? true

É um doente de risco? true

Esteve num local de risco? true

Editar

### 6.2.3.9 Editar Teste

Covid Tests Manager Visualizar Testes Inserir Teste Ver os meus testes ADMIN

Sair

#### Análise do Teste

Estado do utilizador

infetado

Resultado do teste

negativo

Data do exame

2020-06-12

Atualizar

## 7 Conclusão

Ao realizar este trabalho foi nos permitido consolidar toda a matéria lecionada ao longo do semestre na unidade curricular.

Encontramos algumas dificuldades no processo de desenvolvimento, mas no final conseguimos implementar a grande maioria dos requisitos propostos para o trabalho.

## 8 Git

<https://github.com/LuisMarques99/Covid-Tests-Manager>

## 9 Bibliografia

- <https://angular.io/cli/generate>
- <https://www.mongodb.com/>
- <https://getbootstrap.com/>
- <https://nodejs.org/en/>