

Aplicação WEB de registo, processamento e agendamento de testes de diagnóstico do covid19.

Relatório de reflexão crítica do trabalho
desenvolvido na disciplina de PAW

Realizado por Rui Soares nº 8150289

Índice

Introdução	3
Proposta	3
Scope do Produto	3
Projeto - Teste de Covid19	4
Descrição Geral	5
Requisitos Funcionais	6
Gerir utilizadores	6
Gerir Pedidos	8
Agendar Pedidos	10
Implementação	12
Front-end	18
Conclusão e análise	21

Revisão

	Nome	Data	Razão da mudança	Versão
Entrega 1	Milestone 1	16-05-2020	Criou-se uma API com as quatro operações base para os pedidos e para os utilizadores, e uma autenticação com JWTtoken, mockups feitos em angular.	1.0
Entrega 2	Milestone 2	31-05-2020	Acrescentou-se autorização por Roles, operação de ver perfil, ver estatísticas, agendar pedidos na API e fez-se o front-end em angular.	1.1
Entrega 2	Milestone 2	09-06-2020	Deploy no heroku, criação de uma pasta só para routes no backend	1.2

Introdução

Proposta

‘Teste de Covid19’ é uma aplicação feita para automatizar e ajudar na gestão de pedidos.

Esta aplicação guarda todos os estágios dos pedidos e permite mostrar essa informação aos utilizadores forma rápido e fácil de interpretar com isto pretendemos ajudar o centro de análises a agir mais rápido e ter uma gestão mais eficiente com o objetivo de obter uma melhoria do serviço ao cliente.

Scope do Produto

O objetivo do ‘Teste de Covid19’ é guardar toda a informação dos pedidos e utilizadores e agendar consultas.

Projeto - Teste de Covid19

O projeto desenvolvido insere-se num centro de análises regional onde foi decidido que o centro de análises deve suportar uma aplicação web para registar, diagnosticar e agendar pedidos de testes.



Testes Covid19

Esta aplicação web visa a adoção das melhores práticas no centro de análises, organizando os processos de forma a permitir obter melhorias de desempenho.

Objetivos Testes Covid19:

Melhorar a eficiência resultado da adoção generalizada de melhores práticas;

Obter maior controlo de gestão de tempo;

Facilitar a organização de testes e pacientes;

Organizar as diferentes funções no centro de análises.

Para atingir o objetivo do projeto, dividiu-se o mesmo em requisitos que foram implementados em ambiente web.

Descrição Geral

Perspetiva do Produto

Com este Produto pretende-se ajudar o centro de análises a fazer um melhor processo de gestão de testes com uma ferramenta que faz a gestão, pedidos, utilizadores e agendamento com o objetivo de obter uma melhoria na eficácia e eficiência.

Funções do Produto

Gestão de pedidos - mostra os estágios de cada pedido, o resultado e permite criar e modificar pedidos.

Gestão de utilizadores - permite criar e modificar utilizadores na aplicação.

Agendamento - permite associar Doutores com pedidos e marcar consultas

Requisitos Funcionais

Gerir utilizadores

Descrição e prioridade

Nesta categoria estarão englobados os requisitos que permitirão ao administrador registar, modificar, e apagar utilizadores e também visualizar as estáticas diárias e mensais de pedidos feitos e com resultado positivo.

Nome: Registar Utilizadores.

Categoria: Gerir utilizadores

Descrição:

Esta funcionalidade permitirá que se registre e guarde informação sobre os funcionários, aos mesmo serão atribuídas roles que ditaram as partes da aplicação que cada um poderá aceder

Prioridade: 5/5

Estado: fixo

Restrição: Somente os Administradores podem registar utilizadores

Verificação: O utilizador só é registado quando se preencher os campos obrigatórios.

Nome: Modificar Utilizadores

Categoria: Gerir Utilizadores

Descrição:

Esta funcionalidade permitirá que se altere as informações sobre os utilizadores do centro de análises, atribuí-los a uma nova role, e excluir o utilizador do Sistema.

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito só será acessível somente para Administradores.

Verificação: Ao modificar o utilizador o formulário tem que conter todos os campos obrigatórios.

Nome: Visualizar Utilizadores registados

Categoria: Gerir Utilizadores

Descrição:

Esta funcionalidade permitirá que um administrador possa visualizar todos os utilizadores registados posteriormente na aplicação

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito só será acessível somente para Administradores.

Nome: Visualizar estatísticas

Categoria: Gerir Utilizadores

Descrição:

Esta funcionalidade permitirá que um Administrador visualizar as estatísticas diárias e mensais de pedidos feitos e com resultado positivo.

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito não pode ser acedido por um Administrador

Gerir Pedidos

Nesta categoria estarão englobados os requisitos que permitirão ao administrador registar, modificar, apagar, e visualizar as estáticas diárias e mensais de pedidos feitos e com resultado positivo.

Nome: Requisitar pedidos de teste

Categoria: Gerir Pedidos

Descrição:

Esta funcionalidade permitirá que um utilizador autorizado registe pedidos de teste, este pedido deve conter as informações do paciente

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito não pode ser acedido por um doutor

Nome: Modificar Pedidos de teste

Categoria: Gerir Pedidos

Descrição:

Esta funcionalidade permitirá que se altere as informações sobre o pedido de teste e excluir o mesmo.

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito não pode ser acedido por um doutor

Verificação: Os campos obrigatórios têm que estar preenchidos para ser possível modificar o pedido.

Nome: Visualizar pedidos registados

Categoria: Gerir Pedidos

Descrição:

Esta funcionalidade permitirá que um utilizador possa visualizar todos os pedidos registados posteriormente na aplicação que não estão completos ou agendados

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito não pode ser acedido por um doutor.

Agendar Pedidos

Nesta categoria estarão englobados os requisitos que permitirão a um técnico selecionar um pedido e um doutor e agendar um pedido e que um doutor visualize a sua agenda

Nome: Agendar pedidos

Categoria: Agendar pedidos

Descrição:

Esta funcionalidade permitirá que um Técnico selecionar um pedido e um doutor e agendar um pedido

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito só poderá ser acedido por um Administrador e um Técnico.

Verificação: O agendamento só pode ser feito quando selecionar um pedido, um utilizador e a hora e data da consulta.

Nome: Visualizar Agenda

Categoria: Agendar pedidos

Descrição:

Esta funcionalidade permitirá a um Doutor ver todos os pedidos atribuídos ao mesmo, assim como a data e a hora de início e fim da consulta.

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito só poderá ser acedido por um Doutor

Nome: Completar pedidos

Categoria: Agendar pedidos

Descrição:

Esta funcionalidade permitirá a um Doutor informar se um pedido está completo e atribuir um resultado ao mesmo.

Prioridade: 5/5

Estado: fixo

Restrição: Este requisito só poderá ser acedido por um Doutor.

Verificação: Um pedido só pode ser completo após selecionar uma consulta e atribuir um resultado.

Implementação

Nesta fase, de forma a facilitar a parte do front-end foi feita uma API com as 4 operações básicas (CRUD) para gestão de pedidos e de utilizadores.

Após a base da API foi feito um sistema de autenticação com JWTToken, que cria um token com as informação nome, email, role id do utilizador, esse token é intercetado no front end por um token-intercept.service que vai fazer um clone do request e inserir no header 'Authorization' um Bearer token, com o bearer token foi mais fácil testar a API no postman porque tem a opção de guardar na autenticação um bearer token.

Descodificando o JWTToken foi acrescentado á API um get que devolve o utilizador autenticado e uma autenticação por Role.

Autenticação

No userService foram adicionados os métodos de autenticação

```
setToken(token:string){
  localStorage.setItem('token', token);
}

getToken(){
  return localStorage.getItem('token');
}

loggedIn(){
  return !!localStorage.getItem('token');
}

logout(){
  localStorage.removeItem('token');
  this.router.navigate(['user/login']);
}
```

No submit do formulário de login do componente de login é usado o “setToken” quando o utilizador é autenticado com sucesso.

```
Submit(form: NgForm){
  return this.userService.postUser('login', form.value).subscribe(
    res =>{
      this.userService.setToken(res['token']);
      location.reload();
      this.resetForm();
    }, err =>{
      this.resetForm();
      this.error = "username ou password incorrrreta";
    }
  );
});
```

Este token é guardado no local storage que vai ser intercetado pelo serviço “token-intercept.service”, este serviço faz um clone do request e insere no header o token com a palavra bearer na Authorization.

```
export class TokenInterceptService implements HttpInterceptor {

  constructor(private injector: Injector) { }

  intercept(req, next){
    let userService = this.injector.get(UserService);
    let tokenizedReq = req.clone({
      setHeaders: {
        Authorization: `Bearer ${userService.getToken()}`
      }
    });
    return next.handle(tokenizedReq);
  }
}
```

É usado no app.module nos providers como um objeto com três propriedades uma é o provide que vai ter HTTP_INTERCEPTORS, a segunda é useClass que vai usar o interceptService criado e a terceira “multi” que vai ser true para se mais tarde for preciso usar mais interceptors.

```
providers: [
  {
    provide: HTTP_INTERCEPTORS,
    useClass: TokenInterceptService,
    multi:true
  },
  AuthGuard
],
```

Esse header é depois verificado pelo backend num middleware que vai dividir a “authorization” e verificar com o “jwt.verify”.

```
module.exports = (req,res,next) => {
  const header = req.headers.authorization;

  if(typeof header !== 'undefined'){
    const bearer = header.split(' ');
    const bearerToken = bearer[1];
    jwt.verify(bearerToken, process.env.ACESS_TOKEN_SECRET,(err,authD) => {
      if(err){
        res.sendStatus(401);
      }else authD;
    });
    next();
  }else{
    res.sendStatus(403);
  };
}
```

Cada route no backend vai ter uma verificação com o middleware que vai decidir se o utilizador pode usar a API ou não. O nome do middleware acima é authToken.

```
const express = require('express')
const pedidoController = require('../controllers/pedidoController')
const pedidoRouter = express.Router()

const authToken = require('../middleware/auth');
const authRole = require('../middleware/authRole');

pedidoRouter.get("/",authToken,authRole("Admin Normal Tecnico"),pedidoController.pedidoList)
pedidoRouter.get("/find/:id",authToken,authRole("Admin Normal Tecnico"),pedidoController.pedidoFindById)
pedidoRouter.post("/",authToken,authRole("Admin Normal Tecnico"),pedidoController.pedidoCreat)
pedidoRouter.put("/find/:id",authToken,authRole("Admin Normal Tecnico Doutor"),pedidoController.pedidoUpdate)
pedidoRouter.delete("/find/:id",authToken,authRole("Admin Normal Tecnico"),pedidoController.getPedido, pedidoController.pedidoDelete)
pedidoRouter.get("/estatisticas",authToken,authRole("Admin"),pedidoController.estatisticas)

module.exports = pedidoRouter
```

Também foi feito no frontend um authGuard que implementa a interface canActivate este authGuard tem um método canActivate() que vai verificar se o user está autenticado com o método "loggedIn()" do userService se este método for true o authGuard retorna true se for false retorna para a pagina de login.

```
export class AuthGuard implements CanActivate {

  constructor(private authService : UserService,
    private router: Router){

  }

  canActivate(): boolean{
    if(this.authService.loggedIn()) {
      return true;
    }else{
      this.router.navigate(['user/login'])
      return false;
    }
  }
}
```

O authGuard é depois usado nas rotas para verificar se esta autenticado ou não.

```
const routes: Routes = [
  {path: '', component: HomeComponent, canActivate: [AuthGuard]},
  {path: 'pedido', component: PedidoComponent, canActivate: [AuthGuard]},
  {path: 'user/registo', component: ResgistroComponent, canActivate: [AuthGuard]},
  {path: 'user/login', component: LoginComponent,},
  {path: 'user', component: UmanageComponent,canActivate: [AuthGuard] },
  {path: 'user/perfil', component: UserinfoComponent,canActivate: [AuthGuard] },
  {path: 'agendamento', component: AgendamentoComponent,canActivate: [AuthGuard] },
  {path: 'refresh', component: HomeComponent },
];
```

Heroku

No final do projeto foi feito um deploy com o heroku foi usado no angular o comando “ng build –prod” que vai fazer com que a aplicação seja de produção e gerar a pasta dist que vai ser usada para servir o angular no backend.

O heroku usa a porta 5000 como default então o server foi mudado para a porta 5000.

Definiu-se como estático os ficheiros da pasta dist/Frontend e em qualquer path são usados esses ficheiros.

```
app.listen(process.env.PORT || 5000, () => console.log('Server começou na porta : 5000'));

app.use('/agendamento', agendamentoRouter);
app.use('/pedido', pedidoRouter);
app.use('/user', userRouter);
app.use('/api-docs', swaggerUI.serve, swaggerUI.setup(swaggerDoc));

app.use(express.static(__dirname + '/Frontend/dist/Frontend'));
app.use('/*', function (req,res){
  console.log("sent");
  try{
    res.sendFile(path.join(__dirname + '/Frontend/dist/Frontend'));
  }catch(err){
    console.log(err);
  }
})
```

A pasta do frontend foi posta dentro da pasta do servidor, assim o heroku reconhece a linguagem e é possível aceder á pasta dist.



O deploy do projeto foi feito com sucesso, mas é muito lento, ao fazer login a aplicação faz uma atualização na página e o heroku demora mais de 2 segundos a fazer a atualização então dá erro.

<https://covid19teste.herokuapp.com/user/login>

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

 master 

Deploy Branch

Receive code from GitHub



Build master c9262235



Release phase



Deploy to Heroku



Your app was successfully deployed.

 View



Jump to Favorites, Apps, Pipelines, Spaces...



Personal > covid19teste



Open app

More

GitHub rui099/PAW_FINAL_BACKEND master

Overview Resources Deploy Metrics Activity Access Settings

Application Logs

ALL PROCESSES

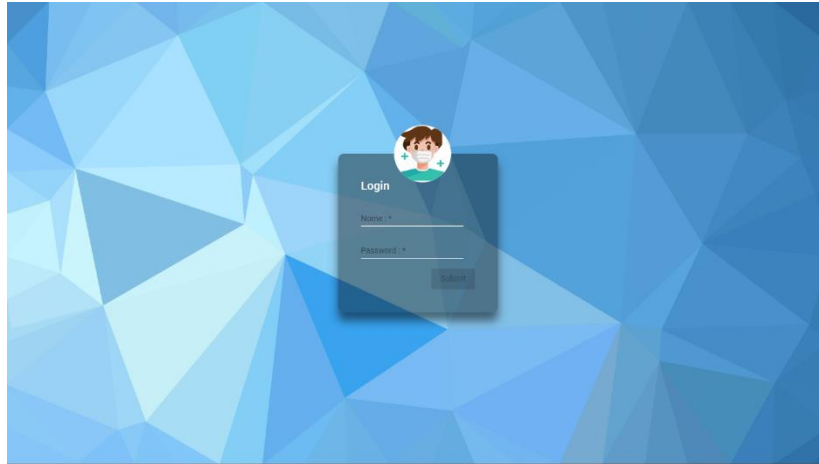
```
2020-06-08T22:14:02.909116+00:00 app[web.1]:
2020-06-08T22:14:03.597227+00:00 app[web.1]: Server começou na porta : 5000
2020-06-08T22:14:03.644530+00:00 heroku[web.1]: State changed from starting to up
2020-06-08T22:14:04.280264+00:00 app[web.1]: Conectado ao Mongo ...
2020-06-08T22:14:04.841731+00:00 app[web.1]: Admin:
2020-06-08T22:14:04.845447+00:00 app[web.1]:
2020-06-08T22:14:04.845448+00:00 app[web.1]: | (index) | pedidos | _id | nome | email | password |
2020-06-08T22:14:04.845448+00:00 app[web.1]: | role | _v |
2020-06-08T22:14:04.845449+00:00 app[web.1]: | 0 | [] | 5ed9582cf321a3238c33bb76 | 'Admin' | 'XXXX@estg.ipp.pt' |
2020-06-08T22:14:04.845449+00:00 app[web.1]: | '$2b$10$LNixBCLoHFszesfCbKLiUOEVRwpXBHiFgnleisDkARmpdWuVokQLC' | 'Admin' | 0 |
2020-06-08T22:14:04.845449+00:00 app[web.1]:
```

☒ Autoscroll with output

[Save](#)

Front-end

No front-end se um utilizador tentar aceder a páginas sem estar autenticado será redirecionado para a página de login.



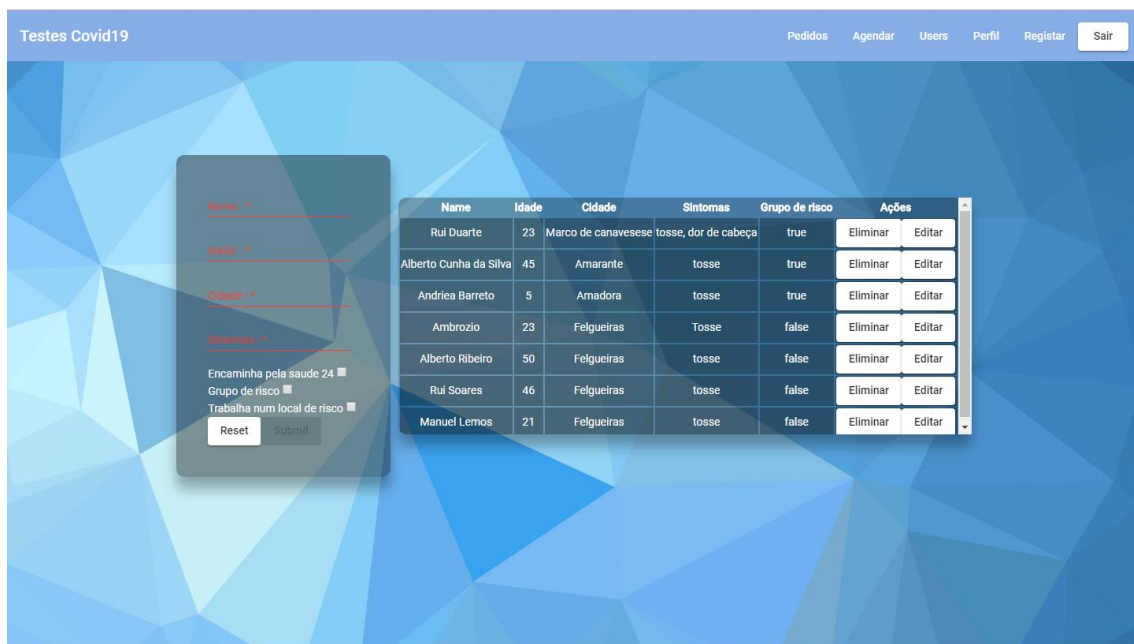
Após a autenticação o utilizador será levado á home page, a home page está muito simples.

Um utilizador com a role 'Doutor' só tem uma opção na barra de navegação que é a 'Agenda' onde tem as sua informações e uma lista com as datas e horas da consultas.

[illegible]

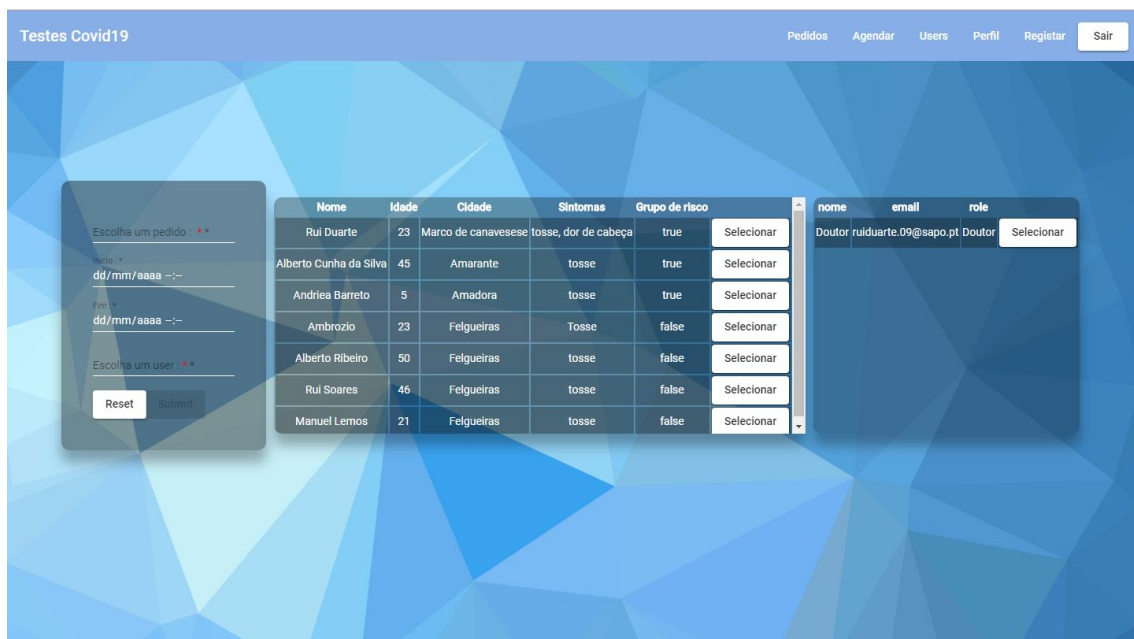
E um formulário que permite selecionar uma consulta e completá-la com o resultado do teste.

Um utilizador normal só tem a opção pedidos onde tem um formulário para a criação e edição de pedidos e uma lista com os botões de editar e editar.



Tem também a opção de ver perfil onde estão as informações do utilizador.

O Técnico tem as opções de um utilizador normal e uma opção de 'Agendar', onde são listados os pedidos não agendados e não completo assim como uma lista de Utilizadores com a role Doutor.



Por fim um Administrador tem todas as opções de um Técnico e a opção de registar e gerir utilizadores.

Testes Covid19

Pedidos Agendar Users Perfil Registrar Sair

Registo

Nome : *

Email : **

Password : **

Escolha uma role *

Submit

localhost:4200/user/registro

Na opção Users temos uma lista com todos os utilizadores e um formulário que vai permitir editar e apagar utilizadores.

Testes Covid19

Pedidos Agendar Users Perfil Registrar Sair

Nome : *

Email : *

Password : *

Role :

Reset Submit

Nome	email	role		
Doutor	ruiduarte.09@sapo.pt	Doutor	Eliminar	Editar
Normal	ruiduarte.09@sapo.pt	Normal	Eliminar	Editar
Tecnico	ruiduarte.09@sapo.pt	Tecnico	Eliminar	Editar
Admin	XXXX@estg.lpp.pt	Admin	Eliminar	Editar

Conclusão e análise

Este projeto foi interessante, consegui usar a matéria dada nas aulas praticas, a interpretação do enunciado é fácil. No início a parte do angular parecia difícil, mas com as aulas tornou-se mais claro.

Tive dificuldades a mudar a barra de navegação com forme a role do utilizador, quando consegui que funcionasse, começou a dar erros porque está a fazer um `location.reload()` após o login.

Na parte da API nas estatísticas está a filtrar os pedidos completos por utilizador, mas não consegui passar isso para o angular.

Em geral o projeto está extremamente simples, mas deu para aprender bastante e acho que esse é o objetivo.