

Instituto Politécnico do Porto

Licenciatura em Engenharia Informática

Ano letivo 2024/2025

Unidade Curricular de Programação em Ambiente Web

Trabalho Prático – PAW Food Hub

Grupo:

Diogo Silva – 8220238

Guilherme Barreiro – 8220849

Miguel Costa – 8210382

Docentes:

Nelson Rodrigues

João Ramos

Fábio Silva

Índice

1. Introdução	4
2. Credenciais de acesso	5
3. Definição dos modelos de base de dados em MongoDB.....	6
3.1 Categories.....	6
3.2 Employee.....	7
3.3 Orders.....	8
3.4 Restaurant	10
3.5 User	12
4. Funcionalidades do BackOffice	14
4.1 Registo/Login de utilizadores	14
4.2 Página principal	15
4.3 Administração de Restaurantes	15
4.3.2 Gerir Restaurantes	16
4.3.3 Editar Restaurantes	18
4.3.3 Remover Restaurantes	18
4.4 Perfil do Utilizador.....	19
4.4.1 Editar Perfil do Utilizador	20
4.4.2 Criar Restaurante	21
4.4.3 Gerir Funcionários	21
4.5 Funcionalidades de Admin	24
4.5.1 Ver Restaurante	24
4.5.1 Perfil de Administrador	24
4.5.2 Dashboard de Admin.....	25
4.5.3 Validar Restaurantes	26
4.5.4 Remover utilizadores	26
4.5.5 Gerir Categorias.....	27
3.2 Entidades beneficiadoras	Error! Bookmark not defined.
3.3 Doadores	Error! Bookmark not defined.
1. Funcionalidades propostas	33
4.1 Sistema de envio de emails (integração com MailGun)	33
4.2 Implementação de sistema de angariações	34
4.3 Integração com API de pagamento para doações diretas em Euros (PayPal)	34
2. Estrutura analítica do projeto	34
5.1 Arquitetura	34
5.2 Desenvolvimento do projeto.....	35

5.3 Flows e atividades	37
3. Conclusão	40

1. Introdução

Este relatório descreve o trabalho desenvolvido no âmbito do projeto da unidade curricular de Programação em Ambiente Web, tendo como objetivo a criação de uma plataforma web para apoio à gestão de pedidos, menus e avaliações por parte de restaurantes.

A aplicação contempla dois módulos principais:

- Um BackOffice, desenvolvido com ExpressJS e EJS, destinado à administração da plataforma, onde os gestores validam restaurantes, gerem menus e monitorizam encomendas;
- Um FrontOffice, desenvolvido com Angular, voltado para a interação do cliente com o sistema, permitindo explorar menus, realizar encomendas.

Durante a Milestone 1, foram implementadas as fundações do sistema: registo de restaurantes, autenticação básica e a gestão de menus. Estas funcionalidades foram desenvolvidas recorrendo à framework ExpressJS com EJS como motor de templates, e com MongoDB como base de dados para persistência da informação.

Na Milestone 2, procedeu-se à expansão significativa da aplicação, com a implementação de uma SPA em Angular para os clientes, sustentada por uma API RESTful totalmente documentada via Swagger. Entre as novas funcionalidades destacam-se: carrinho de compras com contador decrescente, sistema de cancelamento com regras, visualização de histórico de encomendas, submissão de avaliações com imagem, e sistema de notificações em tempo real via interface visual.

A plataforma foi desenvolvida respeitando os princípios de arquitetura cliente-servidor, o padrão MVC e os requisitos técnicos definidos no enunciado, assegurando uma separação clara entre lógica de negócio, apresentação e persistência de dados.

2. Credenciais de acesso

Acesso ao BackOffice:

- Administrador:
 - username: admin
 - password: admin
- Dono de restaurante 1 (possui 1 restaurantes)
 - username: miguel
 - password: miguel
- Dono de restaurante 2 (possui 1 restaurantes)
 - username: JesusCristo
 - password: a
- Dono de restaurante 3 (possui 3 restaurantes)
 - username: JoseFontes
 - password: a
- Dono de restaurante 4 (possui 4 restaurantes)
 - username: CristianoAveiro
 - password: a

Acesso ao FrontOffice:

- Administrador:
 - username: admin
 - password: admin
- User 1:
 - username: candidato
 - password: candidato
- User 2:
 - username: teste
 - password: teste
- User 3:
 - username: miguel
 - password: miguel

Acesso à base de dados:

- Connect:
 - mongodb+srv://admin:admin@p-paw.6s7oww0.mongodb.net/Modelos

3. Definição dos modelos de base de dados em MongoDB

3.1 Categories

```
_id: ObjectId('680a421d731ffa7ba5302656')  
name: "Carne"  
__v: 0
```

```
_id: ObjectId('680a4224731ffa7ba530265b')  
name: "Peixe"  
__v: 0
```

```
_id: ObjectId('680a422c731ffa7ba5302660')  
name: "Vegetariano"  
__v: 0
```

//models/Category.js

```
1  const mongoose = require('mongoose');  
2  
3  const categorySchema = new mongoose.Schema({  
4    name: { type: String, required: true, unique: true },  
5  });  
6  
7  module.exports = mongoose.model('Category', categorySchema);
```

3.2 Employee

```

_id: ObjectId('68250c5f8556e8fac63296b5')
username: "dids"
password: "$2b$10$Ltc08JK7ceKosXKMtMPEj03HuyX01Zm6Q0.IkJlPyJSIdkPEkJVx."
nomeCompleto: "Diogo"
email: "dids@gmail.com"
telefone: "123456789"
owner: ObjectId('680d2ee653401479fc6c2851')
▼ restaurants: Array (1)
  0: ObjectId('680d4947e7d80530e00916cd')
__v: 0

```

```
//models/Employee.js
```

```

1  const mongoose = require('mongoose');
2
3  const employeeSchema = new mongoose.Schema({
4    username: {
5      type: String,
6      required: true,
7      unique: true,
8      match: [/^[A-Za-z]+$/, 'O nome de utilizador só pode conter letras.'],
9    },
10   password: {
11     type: String,
12     required: true,
13   },
14   nomeCompleto: {
15     type: String,
16     required: true,
17     match: [/^[A-Za-zÀ-ÿ\s]+$/, 'O nome não pode conter números ou símbolos.'],
18   },
19   email: {
20     type: String,
21     required: true,
22     unique: true,
23     match: [/.+@.+\.+$/, 'Email inválido.'],
24   },
25   telefone: {
26     type: String,
27     required: true,
28     match: [/^\d{9}$/, 'O telefone deve ter 9 dígitos.'],
29   },
30   owner: {
31     type: mongoose.Schema.Types.ObjectId,
32     ref: 'User',
33     required: true
34   },
35   restaurants: [{
36     type: mongoose.Schema.Types.ObjectId,
37     ref: 'Restaurant'
38   }]
39 });
40
41 module.exports = mongoose.model('Employee', employeeSchema);
42

```

3.3 Orders

```
_id: ObjectId('683337ee0bf9099ee39902ed')
restaurant: ObjectId('68328c161ed51c5c898dc6e4')
client: ObjectId('683269dbf4a25bd3e039915d')
▼ items: Array (1)
  ▼ 0: Object
    dish: ObjectId('68332d9c6e231d82c180041b')
    name: "Picanha"
    price: 16
    quantity: 1
    subtotal: 16
    _id: ObjectId('683337ee0bf9099ee39902ee')
total: 16
status: "pending"
tempoTotal: 35
cancelado: false
createdAt: 2025-05-25T15:31:58.497+00:00
__v: 0
```


//models/Order.js

```
1  const OrderSchema = new mongoose.Schema({
2    restaurant: {
3      type: mongoose.Schema.Types.ObjectId,
4      ref: 'Restaurant',
5      required: true
6    },
7    restaurantName: String,
8    client: {
9      type: mongoose.Schema.Types.ObjectId,
10     ref: 'User'
11   },
12   clientName: String,
13
14   employee: {
15     type: mongoose.Schema.Types.ObjectId,
16     ref: 'User'
17   },
18
19   items: [{
20     dish: mongoose.Schema.Types.ObjectId,
21     name: String,
22     price: Number,
23     quantity: Number,
24     subtotal: Number,
25     tipo: String
26   }],
27   total: Number,
28   tempoTotal: Number,
29   status: {
30     type: String,
31     enum: ['pending', 'preparing', 'shipped', 'delivered', 'cancelled'],
32     default: 'pending'
33   },
34   cancelado: {
35     type: Boolean,
36     default: false
37   }
38 }, {
39   timestamps: true
40 });
```

3.4 Restaurant

```
_id: ObjectId('680ceb26b8d3bae0bb7a0db7')
name: "Rolhas e Raízes"
location: "Sobrosa"
image: "semmaneiras (1).jpg"
status: "validado"
createdBy: ObjectId('680ce7454e2a15c919bf95f2')
▼ menu: Array (2)
  ▼ 0: Object
    name: "Spaghetti Bolognese"
    category: "Carne"
    description: "Da boa"
    image: "picanha (1).jpg"
    nutrition: "Calorias: 614 calories, Proteínas: 32 g, Gordura: 22 g"
    ▼ price: Object
      meia: 5
      inteira: 12
    _id: ObjectId('680cecd9b8d3bae0bb7a0dd6')
  ▼ 1: Object
    name: "Lasagna"
    category: "Carne"
    description: "Da boa também"
    image: "lasagna.png"
    nutrition: "Calorias: 416 calories, Proteínas: 26 g, Gordura: 22 g"
    ▼ price: Object
      meia: 4
      inteira: 9
    _id: ObjectId('680ced56b8d3bae0bb7a0e08')
__v: 2
raioEntrega: 5
tempoEntrega: 25
```

//models/Restaurant.js

```
1  const dishSchema = new mongoose.Schema({
2    name: { type: String, required: true },
3    category: { type: String, required: true },
4    description: { type: String, default: '' },
5    image: { type: String, default: 'default-prato.png' },
6    nutrition: { type: String, default: '' },
7    tempoPreparacao: { type: Number, default: 15, min: 1 },
8    price: {
9      meia: { type: Number, min: 0, required: true },
10     inteira: { type: Number, min: 0, required: true }
11   }
12 });
13
14 const restaurantSchema = new mongoose.Schema({
15   name: { type: String, required: true, unique: true, trim: true },
16   location: { type: String, required: true },
17   image: { type: String, default: 'default-restaurant.png' },
18   status: {
19     type: String,
20     enum: ['pendente', 'validado', 'rejeitado'],
21     default: 'pendente'
22   },
23   createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
24   raioEntrega: { type: Number, default: 5, min: 1 },
25   tempoEntrega: { type: Number, default: 20, min: 5 },
26   menu: {
27     type: [dishSchema],
28     validate: [arrayLimit, 'O menu só pode ter até 10 pratos.']
29   }
30 });
31
32 function arrayLimit(val) {
33   return val.length <= 10;
34 }
```

3.5 User

```
_id: ObjectId('683269dbf4a25bd3e039915d')
username: "teste"
password: "$2b$10$6f8Sk7C9iLFzPPSxIlypl.3zVjQsvMRxI5/4cnFyWZIGwtdbLkrdS"
role: "cliente"
email: "teste@gmail.com"
nomeCompleto: "Gervasio"
morada: "Como quem vira"
telefone: "910910000"
dataNascimento: 2000-02-20T00:00:00.000+00:00
nif: "200100321"
▼ historicoEncomendas: Array (empty)
  __v: 4
▼ cancelamentos: Array (4)
  0: 2025-05-28T15:12:49.882+00:00
  1: 2025-05-28T15:49:56.465+00:00
  2: 2025-05-29T16:52:25.125+00:00
  3: 2025-05-29T16:52:30.020+00:00
```

//models/User.js

```
1  const userSchema = new mongoose.Schema({
2    username: {
3      type: String,
4      required: true,
5      unique: true,
6      match: [/^[A-Za-z]+$/, 'O nome de utilizador só pode conter letras.'],
7    },
8    password: {
9      type: String,
10     required: true,
11     minlength: [6, 'A palavra-passe deve ter pelo menos 6 caracteres.'],
12   },
13   role: {
14     type: String,
15     enum: ['cliente', 'admin', 'employee'],
16     required: true
17   },
18   email: {
19     type: String,
20     required: true,
21     unique: true,
22     match: [/.+@.+\.+$/, 'Email inválido.'],
23   },
24   nomeCompleto: {
25     type: String,
26     required: true,
27     match: [/^[A-Za-zÄ-ÿ\s]+$/, 'O nome completo não pode conter números ou símbolos.'],
28   },
29   morada: {
30     type: String,
31     required: true,
32     match: [/^[A-Za-z]/, 'A morada deve conter letras.'],
33   },
34   telefone: {
35     type: String,
36     required: true,
37     unique: true,
38     match: [/^\d{9}$/, 'O telefone deve ter exatamente 9 dígitos.'],
39   },
40   dataNascimento: {
41     type: Date,
42     required: true,
43     validate: {
44       validator: function (value) {
45         return value <= new Date();
46       },
47       message: 'A data de nascimento não pode ser no futuro.'
48     }
49   },
50   nif: {
51     type: String,
52     required: true,
53     unique: true,
54     match: [/^\d{9}$/, 'O NIF deve ter exatamente 9 dígitos.'],
55   },
56   historicoEncomendas: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Order' }],
57   restaurant: { type: mongoose.Schema.Types.ObjectId, ref: 'Restaurant' },
58   cancelamentos: [{
59     type: Date,
60     default: []
61   }],
62   bloqueadoAte: {
63     type: Date,
64     default: null
65   }
66 });
```

4. Funcionalidades do BackOffice

Todas as funcionalidades requisitadas foram desenvolvidas pelo grupo.

4.1 Registo/Login de utilizadores

Qualquer pessoa pode criar uma conta. Após criar uma conta, qualquer utilizador pode criar um restaurante, mas este apenas ficará “ativo” após permissão do admin.

Criar Conta

Nome de Utilizador

Senha

Nome Completo

Email

Morada

Telefone (9 dígitos)

NIF (9 dígitos)

Data de Nascimento
dd/mm/aaaa

Criar Conta

Já tens conta? [Faz login aqui](#)

Login é realizado através de um username + password.

Login

Utilizador

Senha

Login

Ainda não tens conta? [Regista-te aqui](#)

[Esqueci-me da palavra-passe](#)

4.2 Página principal


A página principal pode ser acedida por qualquer utilizador e não precisa de ter sessão iniciada.

Bem-vindo ao PAW Food Hub

Descobre os melhores restaurantes, explora menus e faz parte da nossa comunidade!


[Ir para Restaurantes](#)

Restaurantes em destaque




Rolhas e Raízes
Pratos: 2
Localização: Sobrosa

[Ver Restaurante](#)





Maré Alta
Pratos: 2
Localização: Rua do Farol, 88, Cascais

[Ver Restaurante](#)



Taberna do Solar
Pratos: 2
Localização: Largo do Carmo, 17, Lisboa

[Ver Restaurante](#)



4.3 Administração de Restaurantes

4.3.1 Ver Restaurantes

É possível ver, filtrar e administrar os restaurantes o qual o utilizador é dono.

Restaurantes Registados					
<input type="text" value="Nome"/>		<input type="text" value="Categoria"/>	<input type="text" value="Localização"/>	<input type="text" value="Preço mín"/>	<input type="text" value="Preço máx"/>
		<input type="button" value="Aplicar"/>		<input type="button" value="Limpar"/>	
#	Nome	Localização	Nº de pratos	Ações	
1	Taberna do Solar	Largo do Carmo, 17, Lisboa	2	<input type="button" value="Gerir"/>	<input type="button" value="Editar"/>
2	Brisa do Douro	Avenida Marginal, 210, Vila Nova de Gaia	2	<input type="button" value="Gerir"/>	<input type="button" value="Editar"/>

4.3.2 Gerir Restaurantes

Clicando no botão gerir, na linha de um restaurante, pode ver, criar, editar e remover pratos.

Gestão do Restaurante

Pratos do Menu

#	Nome	Categoria	Meia Dose (€)	1 Dose (€)	Imagem	Ações
1	Bacalhau com natas	Peixe	€17.00	€33.00		Editar Remover Ver
2	Alheira	Carne	€12.00	€24.00		Editar Remover Ver

[Adicionar Prato](#)



Taberna do Solar

4.3.2.1 Editar Pratos

Selecionando a opção editar, pode editar todas as informações de um prato.

Editar Prato

Nome do prato

Bacalhau com natas


Categoria

Peixe

Descrição

Bacalhau desfiado no forno com natas cremosas e toque dourado de gratinado.

Imagem atual:



Alterar imagem

Escolher Ficheiro

Não foi escolhido nenhum ficheiro

Informação Nutricional

Calorias: undefined undefined, Proteínas: undefined undefined, Gordura: undefined undefined

Preços (€)

Meia Dose

17

1 Dose

33



Guardar Alterações

[← Voltar](#)

4.3.2.2 Remover Pratos

A opção remover, permite remover um prato, fornecendo um aviso para garantir que não apaga acidentalmente.

localhost:3000 diz
 Tem a certeza que deseja remover este prato?
OK
Cancelar


se (€)	1 Dose (€)	Imagem	Ações
	€25.00		Editar Remover Ver
	€160.00		Editar Remover Ver

Adicionar Prato
0

4.3.2.3 Ver Pratos

A opção ver permite visualizar as informações de um prato, como a informação nutricional, que não era anteriormente visível.

Rojões



Categoria: Carne

Descrição: Uma boa carne de porco acompanhada com arroz e batata frita

Preço Meia Dose: €12.00

Preço Dose Inteira: €25.00

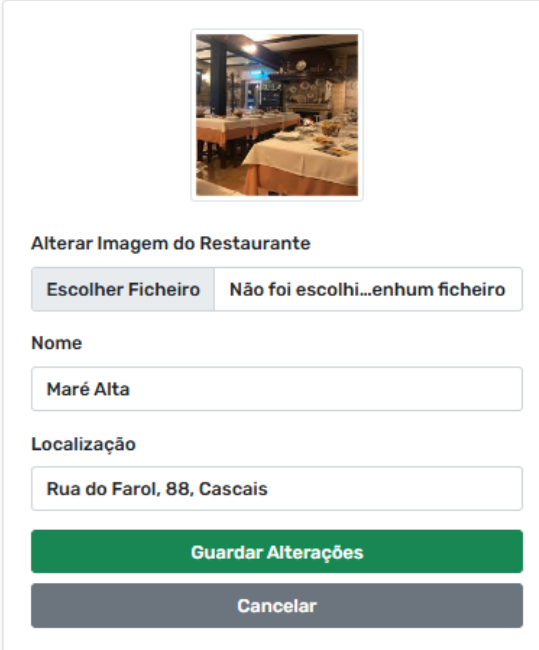
Informação Nutricional: Calorias: 368 calories, Proteínas: 22 g, Gordura: 17 g

[← Voltar ao menu](#)


4.3.3 Editar Restaurantes

No botão editar restaurantes é possível alterar a foto, nome e localização do restaurante.

Editar Restaurante



Editar Restaurante



Alterar Imagem do Restaurante

Escolher Ficheiro Não foi escolhi...enhum ficheiro

Nome

Maré Alta

Localização

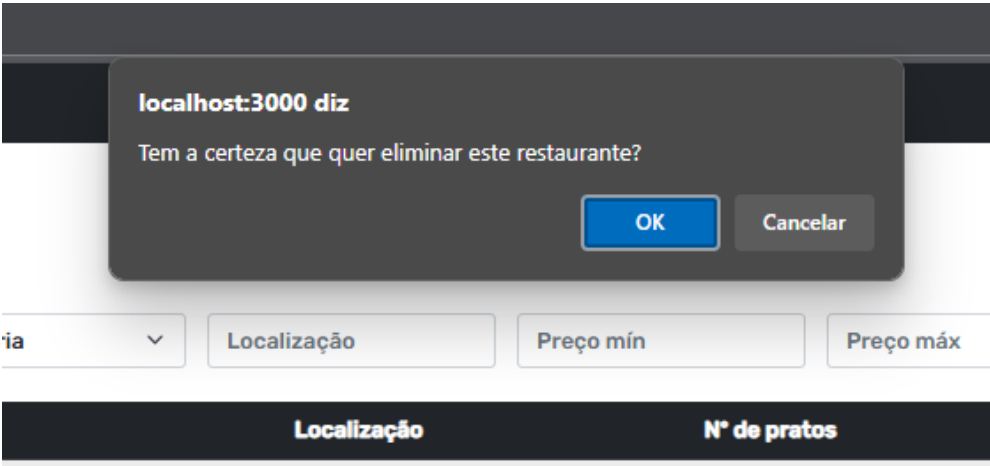
Rua do Farol, 88, Cascais

Guardar Alterações

Cancelar

4.3.3 Remover Restaurantes

Em remover restaurante, o utilizador pode remover um restaurante. Há um alerta antes de ser realizada essa remoção.



localhost:3000 diz

Tem a certeza que quer eliminar este restaurante?

OK Cancelar

Localização Preço mín Preço máx

Localização N° de pratos

4.4 Perfil do Utilizador

São mostradas as informações do utilizador, os restaurantes criados e o histórico de encomendas.

Perfil do Utilizador

Nome de Utilizador: JoseFontes

Nome Completo: Jose Fontes

Email: JoseFontes@gmail.com

Morada: Estrada ali ao lado

Telefone: 100200300

NIF: 103120301

Data de Nascimento: 12/12/2001


Tipo de Conta: CLIENTE

Restaurantes Criados

Maré Alta

Localização: Rua do Farol, 88, Cascais


Validado




O Cantinho da Serra

Localização: Avenida da Montanha, 45, Braga

Validado



Histórico de Encomendas



Ainda não realizaste nenhuma encomenda.

[✍ Editar Perfil](#)
[⊕ Criar Restaurante](#)
[👤 Gerir Funcionários](#)

[← Voltar ao início](#)

4.4.1 Editar Perfil do Utilizador

Selecionando a opção editar, pode editar todas as informações do utilizador.

Editar Perfil

Nome de Utilizador

JoseFontes

Nome Completo

Jose Fontes

Email

JoseFontes@gmail.com

Morada

Estrada ali ao lado

Telefone

100200300

NIF

103120301

Data de Nascimento

12/12/2001

Guardar Alterações

← Voltar

4.4.2 Criar Restaurante

O utilizador pode criar um restaurante. Este ficará pendente, até que o admin o valide.

Registrar Restaurante

Imagem do Restaurante

Escolher Ficheiro

Não foi escolhi...enhum ficheiro

Nome

Localização

Registrar

[← Voltar à lista](#)

4.4.3 Gerir Funcionários

O utilizador criar funcionários ou editar/remover funcionários já existentes.

Funcionários

[Novo Funcionário](#)

Username	Email	Restaurantes	Ações	
novouser	novo@user.novouser	Sabores da Ribeiraa	Editar	Remover
leo	leo@s.s	Sabores da Ribeiraa, O Cantinho da Serra	Editar	Remover
argreg	argreg@argreg.argreg	Sabores da Ribeiraa	Editar	Remover

4.4.3.1 Criar Funcionário

O dono do restaurante pode criar um funcionário e escolher os restaurantes a que este novo funcionário pode ter acesso e fazer pedidos.

Novo Funcionário

Username

Palavra-passe

Nome Completo

Email

Telefone

Restaurantes atribuídos
☐ Sabores da Ribeiraa – Rua das Flores Porto
☐ Maré Alta – Rua do Farol, 88, Cascais
☐ O Cantinho da Serra – Avenida da Montanha, 45, Braga

4.4.3.2 Editar Funcionário

A página de editar funcionário permite editar as informações do funcionário.

Editar Funcionário

Username

novouser

Nome Completo

novo user

Email

novo@user.novouser

Telefone

123456789

Restaurantes atribuídos

☒ Sabores da Ribeiraa — Rua das Flores Porto
 ☐ Maré Alta — Rua do Farol, 88, Cascais
 ☐ O Cantinho da Serra — Avenida da Montanha, 45, Braga

Guardar Alterações

Cancelar

4.4.3.3 Remover Funcionário

Em remover funcionário, o dono do restaurante pode remover um funcionário. Há um alerta antes de ser realizada essa remoção.

localhost:3000 diz

Tem a certeza?

OK

Cancelar

Restaurantes	
user	Sabores da Ribeiraa

4.5 Funcionalidades de Admin

4.5.1 Ver Restaurante

O admin pode ver os restaurantes dos utilizadores, mas não pode alterar nada.

Restaurantes Registados				
Nome	Categoria	Localização	Preço mín	Preço máx
			Aplicar	Limpar
#	Nome	Localização	Nº de pratos	Ações
1	Rolhas e Raízes	Sobrosa	2	Ver Pratos
2	Maré Alta	Rua do Farol, 88, Cascais	2	Ver Pratos
3	Taberna do Solar	Largo do Carmo, 17, Lisboa	2	Ver Pratos
4	Encantos do Pátio	Travessa do Castelo, 3, Évora	2	Ver Pratos
5	Brisa do Douro	Avenida Marginal, 210, Vila Nova de Gaia	2	Ver Pratos
6	O Cantinho da Serra	Avenida da Montanha, 45, Braga	2	Ver Pratos
7	Três Pinheiros	Sobrosa	1	Ver Pratos
8	Calabresa	Milano	1	Ver Pratos
9	Os Imparciais	Trofa	0	Ver Pratos

4.5.1 Perfil de Administrador

No perfil do administrador, ele pode ver a dashboard com alguns gráficos com informação interessante, aprovar ou rejeitar a criação de restaurantes, gerir utilizadores e gerir as categorias dos pratos.

Perfil do Big Boss

Nome de Utilizador: admin
Nome Completo: admin
Email: admin@gmail.com
Morada: admin
Telefone: 930120230
NIF: 260417129
Data de Nascimento: 11/11/1920
Tipo de Conta: ADMIN

[✍ Editar Perfil](#)

Visão de Administrador

Acesso rápido às funções administrativas do sistema:

[🏠 Dashboard Principal](#)
[✅ Validar Restaurantes](#)
[👤 Gerir Utilizadores](#)
[📝 Gerir Categorias](#)

Como administrador, tens permissões para validar novos restaurantes, gerir perfis de utilizadores e categorias, e monitorizar a atividade da plataforma.

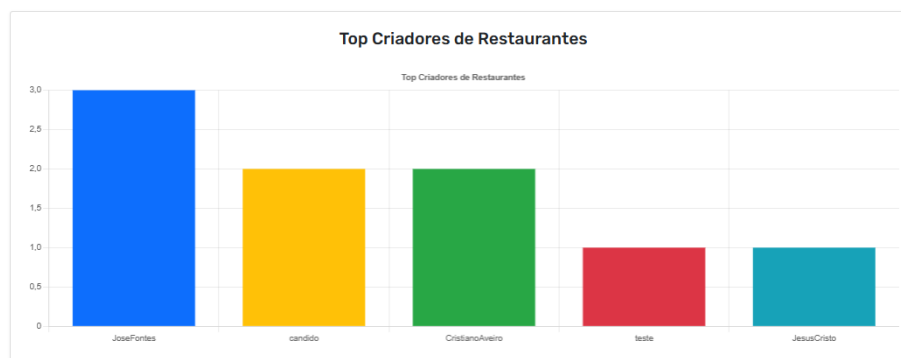
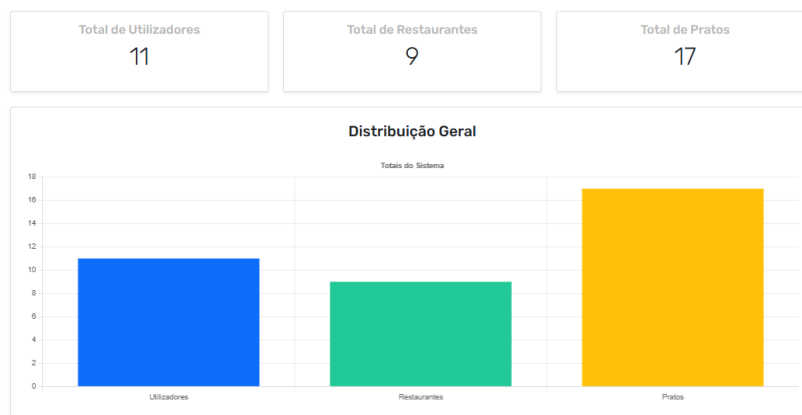
[← Voltar ao início](#)

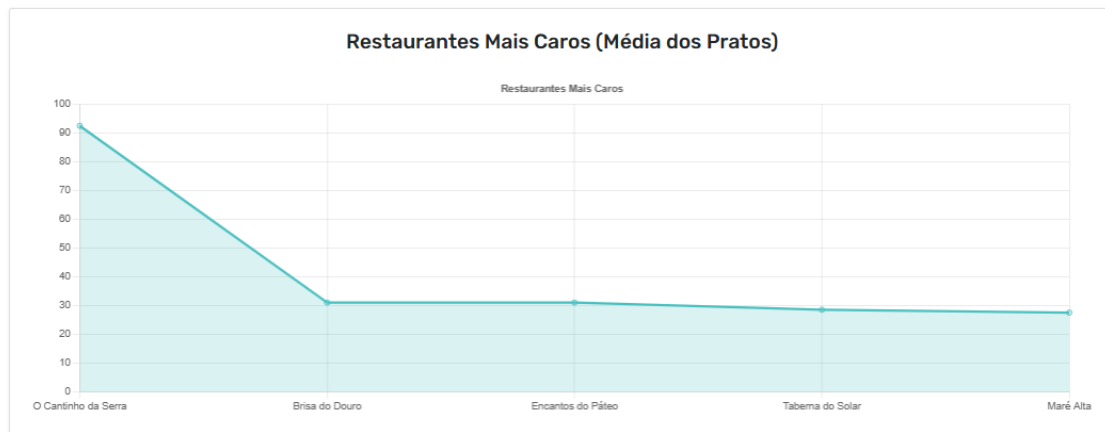
4.5.2 Dashboard de Admin

São mostradas informações sobre:

- total de users, restaurante e pratos;
- nº de restaurantes por user;
- nº de restaurantes com mais pratos;
- Restaurante com média de preço dos pratos

Painel de Administração





4.5.3 Validar Restaurantes

O admin pode escolher validar ou rejeitar a criação de restaurantes.

Validar Restaurantes

Nome	Localização	Criado por	Ação
Boa comida	Viana do Castelo, Conde	JoseFontes	<button>Validar</button> <button>Recusar</button>
Umi no Hana	Avenida Marginal de Cascais	JoseFontes	<button>Validar</button> <button>Recusar</button>
Verde no Prato	Bairro Alto, Lisboa	JoseFontes	<button>Validar</button> <button>Recusar</button>

4.5.4 Remover utilizadores

Gestão de Utilizadores

#	Username	Role	Ações
1	admin	admin	
2	miguel	cliente	<button>Remover</button>
3	asdrubal	cliente	<button>Remover</button>
4	barreiro	cliente	<button>Remover</button>
5	JoseFontes	cliente	<button>Remover</button>
6	CristianoAveiro	cliente	<button>Remover</button>
7	JesusCristo	cliente	<button>Remover</button>
8	dids	cliente	<button>Remover</button>
9	teste	cliente	<button>Remover</button>
10	candido	cliente	<button>Remover</button>
11	nox	cliente	<button>Remover</button>

4.5.5 Gerir Categorias

O administrador pode criar, editar e remover categorias de pratos.

Categorias

[Nova Categoria](#)

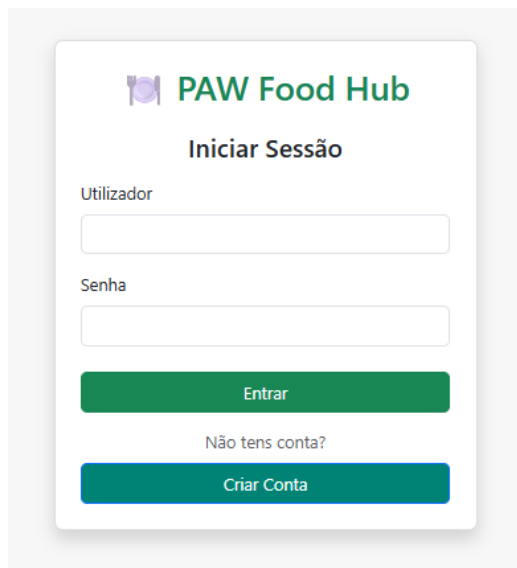
Nome	Ações
Carne	Editar Remover
Entradas	Editar Remover
Mexicana	Editar Remover
Peixe	Editar Remover
Sobremesa	Editar Remover
Sopa	Editar Remover
Vegetariano	Editar Remover

5. Funcionalidades do Front Office

5.1 Registo/Login de utilizadores

Qualquer pessoa pode criar uma conta.

O login é realizado através de utilizador + password.



PAW Food Hub

Iniciar Sessão

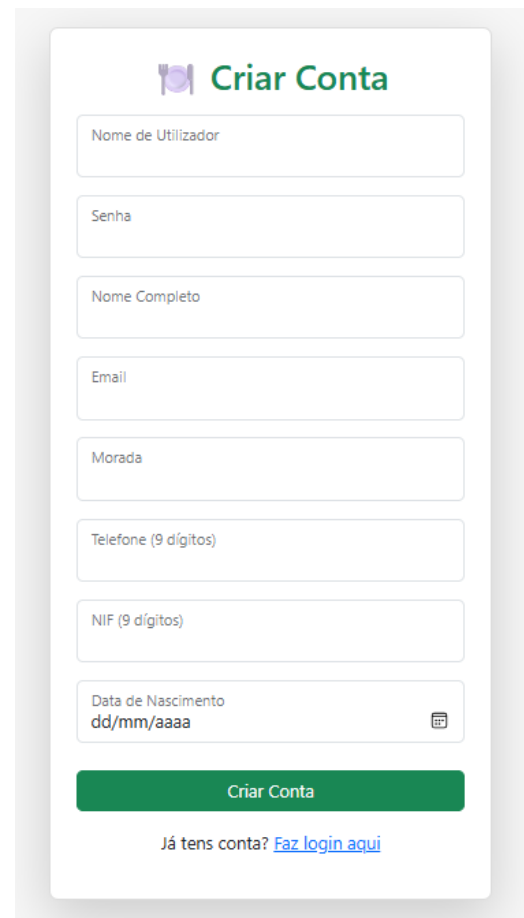
Utilizador

Senha

Entrar

Não tens conta?

Criar Conta



Criar Conta

Nome de Utilizador

Senha

Nome Completo

Email

Morada

Telefone (9 dígitos)

NIF (9 dígitos)


Data de Nascimento
dd/mm/aaaa

Criar Conta

Já tens conta? [Faz login aqui](#)


5.2 Página Principal

A página principal pode ser acedida por qualquer utilizador e não precisa de ter sessão iniciada.




Restaurantes Disponíveis


Explore os restaurantes e os seus pratos disponíveis.




Rolhas e Raízes


 Sobrosa

Sem descrição disponível.

[Ver Pratos](#) 




Maré Alta

 Rua do Farol, 88, Cascais

Sem descrição disponível.


[Esconder Pratos](#)

Pratos:




Arroz de Cabidela

Inteira: 29 €
Meia: 15 €




Arroz de Pato


Inteira: 26 €
Meia: 12 €



Taberna do Solar

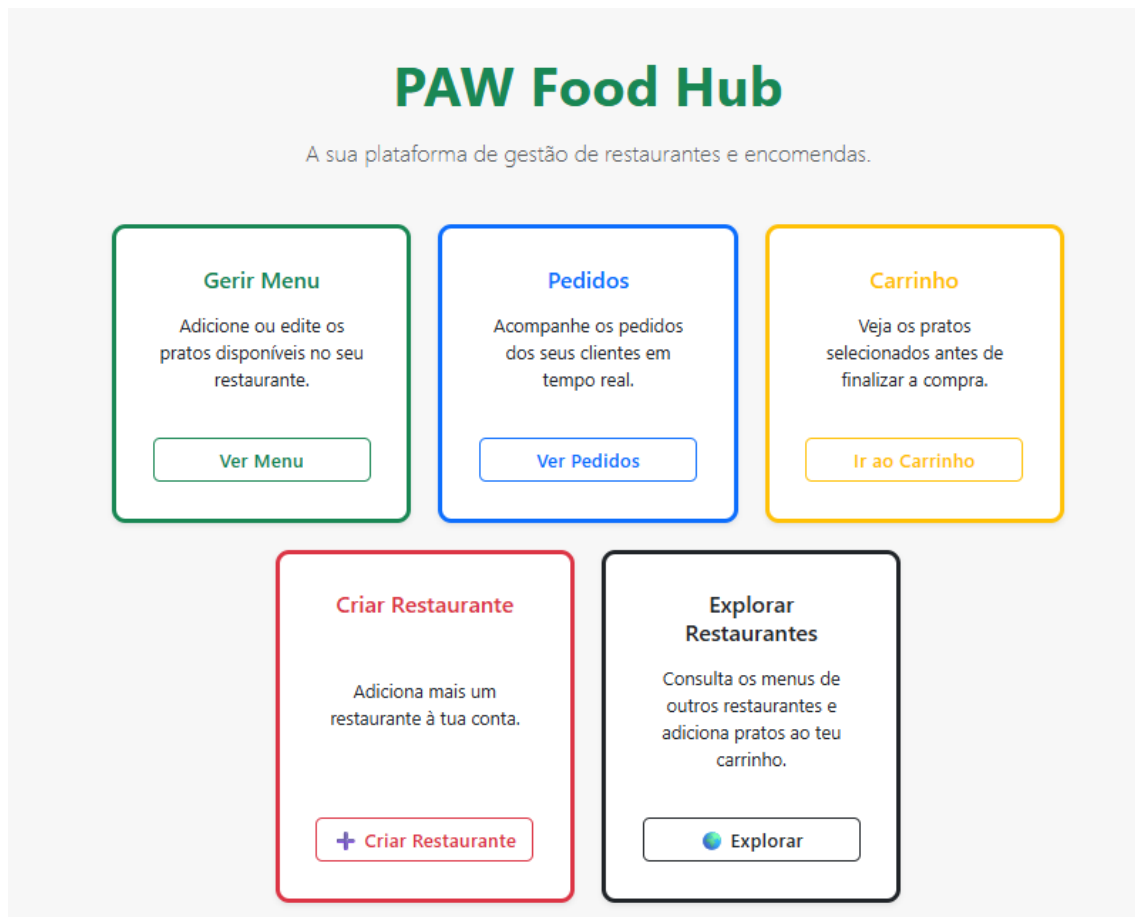
 Largo do Carmo, 17, Lisboa

Sem descrição disponível.

[Ver Pratos](#) 

5.3 Home Page (Após login)

Página inicial do FrontOffice com acesso rápido às principais funcionalidades da plataforma. Permite gerir menus, acompanhar pedidos, explorar restaurantes e utilizar o carrinho. Design intuitivo com blocos funcionais coloridos.



1. Funcionalidades propostas

4.1 Sistema de envio de emails (integração com MailGun)

A plataforma dispõe de um sistema de notificações baseado na API do MailGun que notifica um limitado número de funcionários sempre que é efetuado um registo de um doador e/ou entidade. Consequentemente o sistema também notifica que determinada entidade está a aguardar a sua validação, e (caso seja esse o caso) quando essa entidade for validada.

O MailGun está “configurado” no ficheiro “emailController.js”.

```
1  const Funcionario = require("../models/funcionarioModel");
2  const formData = require("form-data");
3  const Mailgun = require("mailgun.js");
4  const mailgun = new Mailgun(formData);
5  const mg = mailgun.client({username: "reciclaTextil", key: process.env.MAILGUN_API_KEY});
6
7  const sendEmail = async (emailInfo) => {
8    try {
9      const Funcionarios = await Funcionario.find();
10     Funcionarios.forEach(async (funcionario) => {
11       if (funcionario.role === "admin" && funcionario.getEmails) {
12         const emailOptions = {
13           from: "Recicla Têxtil <reciclaTextil@gmail.com>",
14           to: funcionario.email,
15           subject: emailInfo.subject,
16           html: emailInfo.text
17         };
18         await mg.messages.create(process.env.MAILGUN_DOMAIN, emailOptions);
19       }
20     });
21   } catch (err) {
22     console.log(err);
23     res.status(500).render("error", {
24       message: "Ocorreu um erro ao enviar email",
25       error: err,
26     });
27   }
28 }
29
30 module.exports = {
31   sendEmail
32 };
```

4.2 Implementação de sistema de angariações

Através do código de angariador associado a um doador previamente registado, um novo doador pode referenciar o tal doador ao preencher o campo “Código de angariador que o convidou”. Se for esse o caso, o doador referenciado passa a receber um número estabelecido de pontos, definido pelos administradores da plataforma na página “Alteráveis” no BackOffice.

4.3 Integração com API de pagamento para doações diretas em Euros (PayPal)

Como foi previamente referenciado, também é possível doar monetariamente para uma entidade registada. Para tal acontecer basta o doador escolher o botão PayPal presente nos detalhes da entidade escolhida.

2. Estrutura analítica do projeto

5.1 Arquitetura

5.1.1 Explicação

- No nosso projeto, a arquitetura cliente-servidor foi implementada através da framework Angular para o frontend, no FrontOffice, e utilizando ficheiros .ejs para o BackOffice (cliente) e Node.js como runtime com a framework ExpressJS para o backend (servidor).
- O frontend, Angular e ficheiros.ejs, é responsável por interagir com o utilizador final, apresentando a interface gráfica e enviando HTTP requests para o servidor.

- O servidor Express processa esses requests, e comunica com a nossa base de dados em MongoDB para obter ou manipular dados, de seguida é enviada uma resposta com os resultados do respetivo request.

5.1.2 BackOffice

No BackOffice obedecemos ao padrão MVC (Model-View-Controller). Este separa a aplicação em três componentes que se conectam, ou seja:

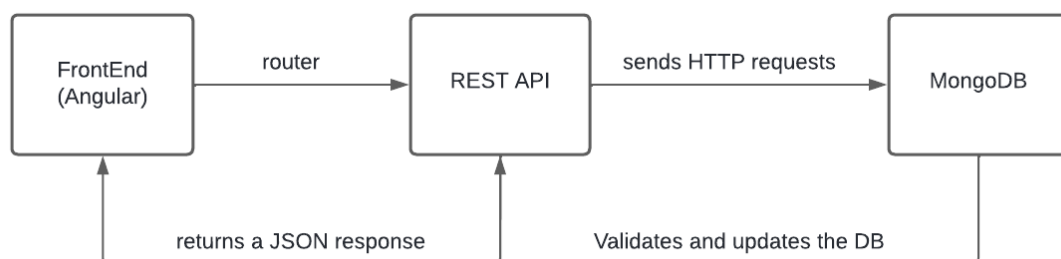
- Model
 - Implementados através do mongoose para definir os esquemas da base de dados.
- View
 - Desenvolvidas através de ficheiros .ejs.
 - Estas representam o frontend desta parte do projeto.
 - São utilizadas para receber e enviar os dados através do router para as respetivas rotas. Rotas essas que comunicam com os DIVERSOS controllers.
- Controller
 - Responsável por comunicar com a base de dados através de HTTP requests.
 - Estes processam os dados recebidos no request, validam-nos e manipulam a base de dados em prol dessa informação.

5.1.3 FrontOffice

Através da framework Angular, o processo de desenvolvimento de um frontend integrado com a API previamente desenvolvido tornou-se mais fácil e direto.

Devido à modelação através de componentes e ao facto de ter sido desenvolvido uma SPA, características próprias desta framework, tornou-se mais acessível a reutilização e a otimização de código, bem como a vasta melhoria na sua leitura e implementação.

O frontend desenvolvido para o FrontOffice, comunica com o backend implementado em EJS, tirando partido dos seus endpoints e separando, de uma forma mais perceptível, o desenvolvimento para o cliente (client-side) e para o servidor (server-side).



5.2 Desenvolvimento do projeto

O projeto foi organizado de forma a facilitar a sua manutenção e clareza. Vamos detalhar a sua estrutura principal:

5.2.1 Estrutura do BackOffice

- controllers: pasta onde se encontram todos os ficheiros respetivos aos métodos que garantem a integração da base de dados bem como a integração do MailGun.
- models: pasta onde se encontram todos os schemas da nossa base de dados,
- public: pasta onde se encontram todos os ficheiros javascript e css que estão associados às views.
- routes: pasta com todas as rotas utilizadas, quer pelo BackOffice quer pelo FrontOffice.
- views: pasta com todos os ficheiros .ejs, ou seja, frontend do BackOffice.
- app.js: Ficheiro executado para iniciar o servidor e os middlewares definidos.
- swagger.json: Ficheiro que configura a documentação de todas as rotas que retornam .json.

(Toda a documentação relacionada com as rotas encontra-se em: <http://localhost:3000/api-docs/>)

5.2.2 Estrutura do FrontOffice (Angular)

- componentes: pasta onde se encontram todos os componentes utilizados no desenvolvimento do Frontend.
 - about-us: componente que contém um curto resumo da empresa.
 - check-donations: componente que contém todas as informações sobre as doações do doador/entidade autenticada. Este componente permite a aplicação de filtros de pesquisa. É reutilizado pelos dois tipos de utilizadores. Quando os doadores acedem a este componente é lhes permitido que cancelem a sua doação caso ela ainda esteja agendada. No caso das entidades, estas podem validar uma doação caso esteja tenha sido aprovada pelos funcionários da Recicla Têxtil, ou seja, se o estado da doação foi “A realizar”.
 - check-entitys: componente acessível pelos utilizadores que carrega a informação de todas as entidades validadas no sistema. Este componente permite a pesquisa dinâmica das diversas entidades. Cada entidade tem um botão de doação associado, a partir deste componente é possível ser redirecionado para o componente “criar-doacao”.
 - create-entity e create-user: componentes responsáveis pela criação de novas contas.
 - criar-doacao: componente que permite o registo de uma doação para a entidade previamente selecionada.
 - editar-entidade e editar-user: componentes responsáveis pela edição das respetivas contas.
 - entidade-initial-page e user-initial-page: componentes iniciais. Este componente “carrega” o componente previamente mencionado “about-us” e age como página inicial após ser dado o login.
 - footer: footer da aplicação.
 - header-entity e header-user: menus acessíveis pelos respetivos tipos de conta.
 - login: componente responsável pelo login na plataforma.
 - message-dialog: responsável pelos popups de erros presentes nas validações. Este componente é chamado pelos vários componentes do FrontOffice.
 - page-not-found: Em caso de a rota não conseguir ser concluída este componente é carregado.

- vales: componente que permite ao doador resgatar os vales registados em troca de pontos.

- guards: responsável por verificar as permissões do utilizador.
- interceptor: responsável por editar o pedido, adicionando o token para futura validação.
- services: pasta com todos os services que acedem à API.

5.2.3 Extras

Para agilizar e otimizar o ritmo de trabalho apresentado pelo grupo foram criados dois ficheiros (runner.bat e killer.bat) para iniciar e encerrar automaticamente quer o FrontOffice quer o BackOffice.

- runner.bat: executa o comando “npm install” e “npm start” em ambas as pastas: BackOffice e FrontOffice.
- killer.bat: termina todos os processos node a rodar na máquina.

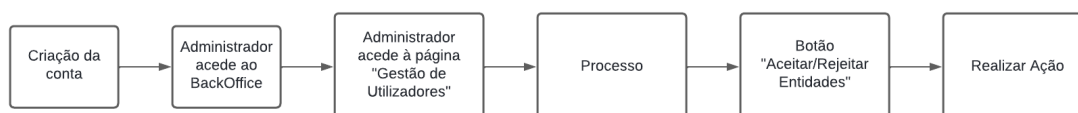
5.3 Flows e atividades

Em caso de dúvida esta secção destina-se a explicar a ordem de acontecimentos de eventos mais complexos e outras possíveis dúvidas.

5.3.1 Entidades

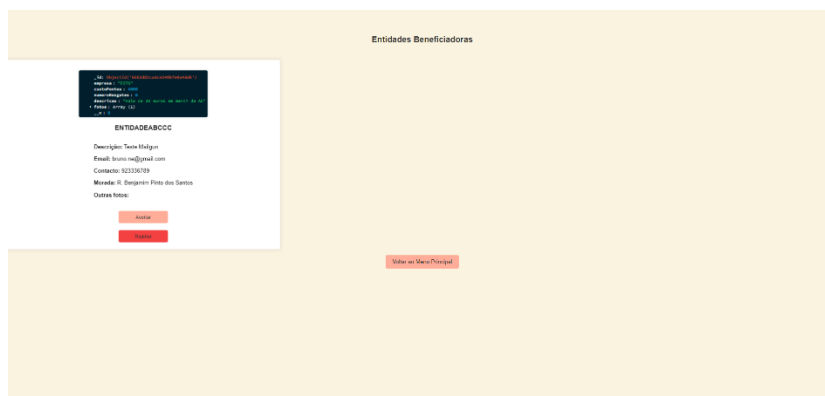
Como foi mencionado ao longo deste relatório as entidades necessitam validação por parte dos administradores da Recicla Têxtil para estarem elegíveis a receber doações.

Tendo em conta que a criação da conta pode ser realizada quer pelo BackOffice (funcionário) quer pelo FrontOffice (próprio utilizador), estas são as etapas para a validação da entidade:



Passo a Passo após a criação da respetiva conta e login no BackOffice como administrador:





Após este processo, caso a entidade fique validada, esta passa a ter acesso a toda a plataforma, caso contrário esta fica guardada na base de dados, na eventualidade de mais tarde vir a ser aceite, mas não tem acesso a nenhuma parte do sistema.

5.3.2 Doações

No nosso sistema as doações podem ter apenas um estado de cada vez, esse estado está limitado em: “Agendado”, “Por iniciar”, “Aprovado”, “A realizar”, “Validado” e “Cancelado”, ou seja:

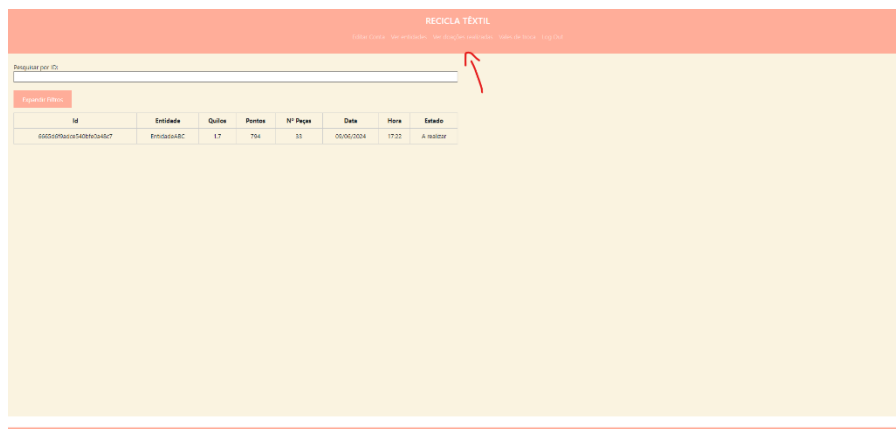
- **Agendado:** O doador submeteu uma doação e quando a data que ele definiu for verificada a doação passa a “Por iniciar”, durante este tempo o doador pode cancelar a doação. (Nota: Todas as doações registadas através do BackOffice são agendadas para o próprio dia e são iniciadas como “Por iniciar” automaticamente).
- **Por iniciar:** A data de agendamento aconteceu e os administradores passam a ter acesso ao pedido, através do menu “Pedidos de Recolha” no BackOffice, sendo possível a aprovação do mesmo ou o seu cancelamento, bem como a correção de qualquer detalhe que esteja errado com a doação.
- **Aprovado:** Os funcionários da Recicla Têxtil aprovaram o pedido e este passa ao estado “A realizar”.
- **A realizar:** A entidade para onde foi efetuada a doação passa a conseguir validar o pedido recebido.
- **Validado:** A entidade validou a doação e o doador, e respetivo angariador se for o caso, recebem os pontos na sua conta.
- **Cancelado:** Em qualquer ponto do processo a doação foi rejeitada ou cancelada pelos intervenientes.

Nota: Para realizar a transição de “Agendado” para “Por iniciar” e de “Aprovado” para “A realizar” está configurado no ficheiro “app.js” um método que a cada minuto verifica o estado e a data de todos os pedidos presentes na base de dados possibilitando assim a sua alteração automática.

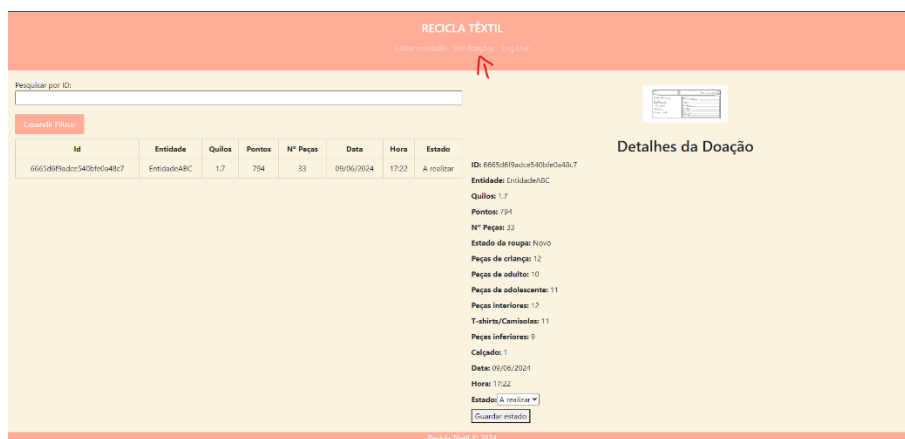
As datas atualizam quando ocorre uma alteração de estado na doação.

Pontos de acesso às doações realizadas:

- Pelos doadores:



- Pelas entidades:



- Pelos funcionários:



3. Conclusão

Por fim, o grupo reconhece que conseguiu um bom trabalho. Durante todo este processo trabalhamos em equipa e colaboramos de forma otimizada e organizada em prol de um projeto bem conseguido.

Possíveis melhorias futuras envolveriam: interface mais detalhada e “embelezada”, maior organização a nível de nomenclaturas e organização de ficheiros, implementação de dashboards mais completas e mais informativas.