



ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

# Projeto de Programação em Ambiente Web

Licenciatura em Engenharia Informática

2023/2024

Grupo 20

Bruno Miguel Magalhães Soares Pontvianne Ferreira - 8220191

Miguel Correia Ribeiro Rangel Tavares - 8220229

Francisco Jose Pinto Costa - 8220805

## Índice

1. Introdução .....	3
2. Credenciais de acesso .....	3
3. Modelos.....	4
2.1 Doador.....	4
2.2 Entidade .....	6
2.3 Funcionário.....	8
2.4 Doação.....	10
2.5 Pontos.....	13
2.6 Vales .....	15
4. Funcionalidades.....	16
3.1 Administradores/Funcionários da plataforma .....	16
3.2 Entidades beneficiadoras .....	19
3.3 Doadores .....	21
5. Funcionalidades propostas.....	23
4.1 Sistema de envio de emails (integração com MailGun) .....	23
4.2 Implementação de sistema de angariações .....	23
4.3 Integração com API de pagamento para doações diretas em Euros (PayPal) .....	24
6. Estrutura analítica do projeto .....	24
5.1 Arquitetura .....	24
5.2 Desenvolvimento do projeto.....	25
5.3 Flows e atividades .....	27
7. Conclusão .....	30

## 1. Introdução

A Recicla Têxtil tem como objetivo desenvolver uma plataforma de doações de produtos têxtil a diversas entidades.

Este sistema consiste em dois componentes principais: BackOffice (portal de administração do sistema) e FrontOffice (aplicação web pública que permite o acesso ao sistema por parte de doadores ou entidades).

Qualquer doador tem acesso às diversas entidades registadas, sendo então possível realizar uma doação, têxtil ou monetária, para a entidade desejada. As doações têxteis realizadas recompensam o doador que a realizou, e/ou um possível doador que o convidou para utilizar a plataforma, com a atribuição de pontos que mais tarde podem ser usados para resgatar vales afetos a diversas empresas conhecidas a nível nacional.

Todo e qualquer tipo de validação pode ser realizada a partir do BackOffice, onde os funcionários da nossa empresa são responsáveis pela correta verificação das doações realizadas, entidades que submetem o seu registo para validação e registo de vales de empresas com parceria.

## 2. Credenciais de acesso

### Acesso ao BackOffice:

- Administrador:
  - Email: projetopaw2324082203@gmail.com
  - Password: admin082203
- Funcionário:
  - Email: testerFuncionario@gmail.com
  - Password: tester

### Acesso ao FrontOffice:

- Utilizador sem convite:
  - Email: 8220191@estg.ipp.pt
  - Password: 123
- Utilizador convidado:
  - Email: jrnr@estg.ipp.pt
  - Password: 123
- Entidade:
  - Email: defesa@gmail.com
  - Password: 123

## Acesso à base de dados:

- Connect:
  - mongodb+srv://BrunoFerreira:wPc1l5M7rtJUBAux@reciclatextil.lufifqt.mongod  
db.net/

### 3. Modelos

#### 2.1 Doador

```
_id: ObjectId('6665d15ab054fe34e30b9a9f')
nome: "Bruno"
apelido: "Ferreira"
email: "bruno.pontvianne2004@gmail.com"
password: "$2a$10$BIfTKFRnW5Dl37lqseJXl08xxnH1osdn1lPcdTuG1q6dzeffoWhPq"
nif: "999229999"
morada: "R. Benjamim Pinto dos Santos"
contacto: "916661739"
numeroPontos: 0
numeroDoacoes: 0
doacoes: Array (empty)
fotoPerfil: "uploads\356b554d766fee00d0d5eb9d8f20abcb"
role: "doador"
codigoAngariador: "bruno1"
codigoAngariadorConvidado: "Sem código"
doacoesAngariadas: Array (empty)
data: 2024-06-09T15:59:22.334+00:00
__v: 0
```

```

1  const mongoose = require('mongoose');
2  const doadorSchema = new mongoose.Schema({
3    nome: {
4      type: String,
5      required: true,
6    },
7    apelido: {
8      type: String,
9      required: true,
10   },
11   email: {
12     type: String,
13     required: true,
14     unique: true,
15     match: [
16       /^([a-zA-Z0-9]+)?@([a-zA-Z0-9]+)?(\.([a-zA-Z0-9]+){2,3})+$/,
17       "Por favor, insira um email válido",
18     ],
19   },
20   password: {
21     type: String,
22     required: true,
23   },
24   nif: {
25     type: String,
26     unique: true,
27     required: true,
28     match: /^[0-9]{9}$/, "O NIF deve conter exatamente 9 dígitos."],
29   },
30   morada: {
31     type: String,
32     required: true,
33   },
34   contacto: {
35     type: String,
36     match: /^[0-9]{8}$/, "Por favor, insira um número de telefone válido"],
37   },
38   numeroPontos: {
39     type: Number,
40     default: 0,
41   },
42   numeroDoacoes: {
43     type: Number,
44     default: 0,
45   },
46   doacoes: [
47     {
48       type: String,
49     },
50   ],
51   data: {
52     type: Date,
53     default: Date.now,
54   },
55   fotoPerfil: {
56     type: String,
57     maxItems: 1,
58   },
59   role: {
60     type: String,
61     default: "doador",
62   },
63   codigoAngariador: {
64     type: String,
65     unique: true,
66     required: true,
67   },
68   codigoAngariadorConvidado: {
69     type: String,
70     required: true,
71     default: "Sem código",
72   },
73   doacoesAngariadas: [
74     {
75       type: String,
76     },
77   ],
78 });
79
80 const Doador = mongoose.model('Doadores', doadorSchema);
81
82 module.exports = Doador;

```

## 2.2 Entidade

```
_id: ObjectId('6665d247b054fe34e30b9ab0')
nome : "EntidadeABC"
email : "entidadeabc@gmail.com"
contacto : "911123123"
numeroDoacoes : 0
morada : "R. Benjamim Pinto dos Santos"
descricao : "Entidade de teste e utilizada no relatório."
password : "$2a$10$qgrXdICYt8FWNd6atlieQeZpxBKf74cuokEoIkEWqZwu1QUeeNYMe"
▸ doacoesPorValidar : Array (empty)
▸ doacoesValidadas : Array (empty)
  fotoPerfil : "uploads\a5a1e8b52dca572cde082cbda79f8a6c"
▸ fotos : Array (empty)
  status : "em_espera"
  role : "entidade"
  __v : 0
```

```

1  const mongoose = require("mongoose");
2
3  const entidadeBeneficiadoraSchema = new mongoose.Schema({
4    nome: {
5      type: String,
6      required: true,
7      unique: true,
8    },
9    email: {
10     type: String,
11     required: true,
12     unique: true,
13     // Validação do formato de email utilizando expressão regular
14     match: [
15       /^[\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/,
16       "Por favor, insira um email válido",
17     ],
18   },
19   contacto: {
20     type: String,
21     required: true,
22     // Validação do formato de número de telefone utilizando expressão regular
23     match: [
24       /^[9][8-9]{8}$/,
25       "Por favor, insira um número de telefone válido",
26     ],
27   },
28   numeroDoacoes: {
29     type: Number,
30     default: 0,
31   },
32   morada: {
33     type: String,
34     required: true,
35   },
36   descricao: {
37     type: String,
38     required: true,
39   },
40   password: {
41     type: String,
42     required: true,
43   },
44   doacoesPorValidar: [{
45     type: String,
46   }],
47   doacoesValidadas: [{
48     type: String,
49   }],
50   fotoPerfil: {
51     type: String,
52     required: true,
53     maxItems: 1,
54   },
55   fotos: [{
56     type: String,
57     required: true,
58   }],
59   status: {
60     type: String,
61     enum: ['aceite', 'rejeitada', 'em_espera'],
62     default: 'em_espera'
63   },
64   role: {
65     type: String,
66     default: "entidade",
67   },
68 });
69
70 const EntidadeBeneficiadora = mongoose.model("EntidadeBeneficiadora", entidadeBeneficiadoraSchema);
71 module.exports = EntidadeBeneficiadora;

```

## 2.3 Funcionário

```
_id: ObjectId('6634a7ccbbce1e48a4315d38')
nome : "Miguel"
apelido : "Tavares"
email : "projetopaw2324082202@gmail.com"
numero_empresa : "4930"
password : "$2a$10$kA0Dav3DPkHl3Yqsy8WJQeJWgJnKo6mSgMnvVn/M06/WeF0WXXVc7S"
admin : true
contacto : "999999999"
morada : "Maia"
role : "admin"
getEmails : true
__v : 0
```



```

1  const mongoose = require("mongoose");
2
3  const funcionarioSchema = new mongoose.Schema({
4      nome: {
5          type: String,
6          required: true,
7      },
8      apelido: {
9          type: String,
10         required: true,
11     },
12     email: {
13         type: String,
14         required: true,
15         unique: true,
16         // Validação de formato de email utilizando expressão regular
17         match: [
18             /^w+([\.-]?\w+)*@w+([\.-]?\w+)*(\.w{2,3})+$/,
19             "Por favor, insira um email válido",
20         ],
21     },
22     numero_empresa: {
23         type: String,
24         required: true,
25         unique: true,
26     },
27     password: {
28         type: String,
29         required: true,
30     },
31     admin: {
32         type: Boolean,
33         default: false,
34     },
35     contacto: {
36         type: String,
37         required: true,
38         match: [
39             /^[9][0-9]{8}$/,
40             "Por favor, insira um número de telefone válido",
41         ],
42     },
43     morada: {
44         type: String,
45         required: true,
46     },
47     role: {
48         type: String,
49         enum: ['funcionario', 'admin'],
50         default: "funcionario",
51     }, getsEmails: {
52         type: Boolean,
53         default: false,
54     }
55 });
56
57 const Funcionario = mongoose.model("Funcionario", funcionarioSchema);
58 module.exports = Funcionario;

```

## 2.4 Doação

```
_id: ObjectId('6665d6f9adce540bfe0a48c7')
email: "bruno.pontvianne2004@gmail.com"
pecasCrianca: 12
pecasAdulto: 10
pecasAdolescente: 11
pecasTipoInterior: 12
pecasTipoTronco: 11
pecasTipoInferior: 9
pecasTipoCalcado: 1
entidade: "EntidadeABC"
kilos: 1.7
pontos: 794
data: 2024-06-09T16:22:50.790+00:00
estado: "Agendado"
▸ fotos: Array (1)
  estadoRoupa: "Semi-Novo"
  codigoAngariador: "Sem código"
  numeroPecas: 33
__v: 0
```

```

1  const mongoose = require("mongoose");
2
3  const requestSchema = new mongoose.Schema({
4    email: {
5      type: String,
6      required: true,
7      match: [
8        /^[\w+]{1,25}@\w+([\.-]?\w+)*(\.[\w{2,3}])+$/,
9        "Por favor, insira um email válido",
10     ],
11   },
12   pecasCrianca: {
13     type: Number,
14     required: true,
15     min: 0,
16   },
17   pecasAdulto: {
18     type: Number,
19     required: true,
20     min: 0,
21   },
22   pecasAdolescente: {
23     type: Number,
24     required: true,
25     min: 0,
26   },
27   numeroPecas: {
28     type: Number,
29     default: function () {
30       return this.pecasCrianca + this.pecasAdulto + this.pecasAdolescente;
31     },
32   },
33   pecasTipoInterior: {
34     type: Number,
35     required: true,
36     min: 0,
37     validate: {
38       validator: function (v) {
39         return (
40           v +
41             this.pecasTipoInterior +
42             this.pecasTipoCalcado +
43             this.pecasTipoTronco ==
44             this.numeroPecas
45         );
46       },
47       message: (props) =>
48         'A soma das peças de diferentes tipos tem que ser igual ao número total de peças (${this.numeroPecas})',
49     },
50   },
51   pecasTipoTronco: {
52     type: Number,
53     required: true,
54     min: 0,
55     validate: {
56       validator: function (v) {
57         return (
58           v +
59             this.pecasTipoInterior +
60             this.pecasTipoCalcado +
61             this.pecasTipoInterior ==
62             this.numeroPecas
63         );
64       },
65       message: (props) =>
66         'A soma das peças de diferentes tipos tem que ser igual ao número total de peças (${this.numeroPecas})',
67     },

```

```

68   },
69   pecasTipoInferior: {
70     type: Number,
71     required: true,
72     min: 0,
73     validate: {
74       validator: function (v) {
75         return (
76           v +
77             this.pecasTipoInferior +
78             this.pecasTipoCalcado +
79             this.pecasTipoTronco ==
80             this.numeroPecas
81         );
82       },
83       message: (props) =>
84         `A soma das peças de diferentes tipos tem que ser igual ao número total de peças (${this.numeroPecas})`,
85     },
86   },
87   pecasTipoCalcado: {
88     type: Number,
89     required: true,
90     min: 0,
91     validate: {
92       validator: function (v) {
93         return (
94           v +
95             this.pecasTipoInferior +
96             this.pecasTipoInferior +
97             this.pecasTipoTronco ==
98             this.numeroPecas
99         );
100       },
101       message: (props) =>
102         `A soma das peças de diferentes tipos tem que ser igual ao número total de peças (${this.numeroPecas})`,
103     },
104   },
105   entidade: {
106     type: String,
107     required: true,
108   },
109   kilos: {
110     type: Number,
111     required: true,
112     min: 0.1,
113   },
114   pontos: {
115     type: Number,
116     default: 0,
117   },
118   data: {
119     type: Date,
120     required: true,
121     validate: {
122       validator: function (value) {
123         const today = new Date();
124         return value.getDay() >= today.getDay();
125       },
126       message: "A data deve ser estabelecida após hoje",
127     },
128   },
129   estado: {
130     type: String,
131     enum: [
132       "Agendado",
133       "Por iniciar",
134       "Aprovado",
135       "A realizar",
136       "Validado",
137       "Cancelado",
138     ],
139     default: "Agendado",
140   },

```

```

141   fotos: [
142     {
143       type: String,
144       required: true,
145     },
146   ],
147   estadoRoupa: {
148     type: String,
149     enum: ["Novo", "Semi-Novo", "Usado"],
150     default: "Usado",
151   },
152   codigoAngariador: {
153     type: String,
154   },
155 ];
156
157 const Requests = mongoose.model("Requests", requestSchema);
158 module.exports = Requests;
159

```

## 2.5 Pontos

```
_id: ObjectId('661d4d0a93b3907303e6c970')
pontosPorKilo : 20
pontosPorPecaAdulto : 20
pontosPorPecaCrianca : 20
pontosPorPecaAdolescente : 20
pontosPorTipoInterior : 20
pontosPorTipoInferior : 20
pontosPorTipoTronco : 20
pontosPorTipoCalcado : 20
data : 2024-06-08T20:00:18.861+00:00
__v : 0
pontosPorEstadoRoupaNovo : 20
pontosPorEstadoRoupaUsado : 20
pontosPorEstadoRoupaSemiNovo : 20
pontosCodigo : 100
```

```

1  const mongoose = require("mongoose");
2
3  const rulesSchema = new mongoose.Schema({
4    pontosPorKilo: {
5      type: Number,
6      required: true,
7      min: 0,
8    },
9    pontosPorPecaAdulto: {
10     type: Number,
11     required: true,
12     min: 0,
13   },
14   pontosPorPecaCrianca: {
15     type: Number,
16     required: true,
17     min: 0,
18   },
19   pontosPorPecaAdolescente: {
20     type: Number,
21     required: true,
22     min: 0,
23   },
24   pontosPorTipoInterior: {
25     type: Number,
26     required: true,
27     min: 0,
28   },
29   pontosPorTipoInferior: {
30     type: Number,
31     required: true,
32     min: 0,
33   },
34   pontosPorTipoTronco: {
35     type: Number,
36     required: true,
37     min: 0,
38   },
39   pontosPorTipoCalcado: {
40     type: Number,
41     required: true,
42     min: 0,
43   },
44   pontosPorEstadoRoupaNovo: {
45     type: Number,
46     required: true,
47     min: 0,
48   },
49   pontosPorEstadoRoupaSeniNovo: {
50     type: Number,
51     required: true,
52     min: 0,
53   },
54   pontosPorEstadoRoupaUsado: {
55     type: Number,
56     required: true,
57     min: 0,
58   },
59   data: {
60     type: Date,
61     default: Date.now,
62   },
63   pontosCodigo: [
64     type: Number,
65     required: true,
66     min: 0,
67   ],
68 });
69
70 const Rules = mongoose.model("Rules", rulesSchema);
71 module.exports = Rules;

```

## 2.6 Vales

```
_id: ObjectId('6665d82cadce540bfe0a48d6')
empresa: "ESTG"
custoPontos: 4000
numeroResgates: 0
descricao: "Vale de 20 euros em merch da AE"
fotos: Array (1)
__v: 0
```

```
1  const mongoose = require("mongoose");
2
3  const valeSchema = new mongoose.Schema({
4    empresa: {
5      type: String,
6      required: true,
7    },
8    custoPontos: {
9      type: Number,
10     required: true,
11     min: 0,
12   },
13   numeroResgates: {
14     type: Number,
15     default: 0,
16   },
17   descricao: {
18     type: String,
19     required: true,
20   },
21   fotos: [
22     {
23       type: String,
24       required: true,
25     },
26   ],
27 });
28
29 const vales = mongoose.model("vale", valeSchema);
30 module.exports = vales;
```

## 4. Funcionalidades

Todas as funcionalidades requisitadas foram desenvolvidas pelo grupo.

### 3.1 Administradores/Funcionários da plataforma

Os funcionários da empresa têm acesso, a partir do BackOffice, a:

- Editar a sua própria conta a partir do menu “Minha conta”:

RECICLA TÊXTIL

Minha Conta | Gestão de Utilizadores | Pedidos de Recicla | Dashboard | Log Out

### Meus dados

Nome:

Apelido:

Email:

Contacto:

Morada:

Senha Atual:

☐ Show Password

Nova Senha:

☐ Show Password

Guardar Alterações

Recicla Têxtil © 2024

- Gestão de todo o tipo de utilizadores, caso o funcionário seja um administrador da empresa este tem acesso a um maior número de funcionalidades, através do menu “Gestão de Utilizadores”. Através da API do MailGun, que por limitação apenas nos deixa registar 5 emails, os administradores registados que estão disponíveis para receber emails são notificados. Exemplo:

Caso o funcionário seja um administrador:





Caso o funcionário não seja um administrador:



- Gerir as regras de atribuição de pontos para as doações (APENAS ADMINISTRADORES), através do menu “Alteráveis”:

RECICLA TÊXIL

[Minha Conta](#) [Gestão de Utilizadores](#) [Alteráveis](#) [Vales](#) [Pedidos de Recolha](#) [Dashboard](#) [Log Out](#)

Definição de regras

Regras de pontos:

Pontos por Kg: 20

Pontos por peça criança: 20

Pontos por peça adulto: 20

Pontos por peça adolescente: 20

Pontos por tipo interior: 20

Pontos por tipo inferior: 20

Pontos por tipo tronco: 20

Pontos por tipo calçado: 20

Pontos por peça de roupa nova: 20

Pontos por peça de roupa semi nova: 20

Pontos por peça de roupa usada: 20

Pontos por código de angariador: 100

Guardar

Recicla Têxtil © 2024

- Gestão dos pedidos de recolha dos doadores, através do menu “Pedidos de Recolha”:

RECICLA TÊXIL

[Minha Conta](#) [Gestão de Utilizadores](#) [Alteráveis](#) [Vales](#) [Pedidos de Recolha](#) [Dashboard](#) [Log Out](#)

Criar pedido

Lista de pedidos

Por Iniciar

Id	Email do utilizador	Entidade	Pontos	Quilos	Número de peças total	Peças de roupa de criança	Peças de roupa de adulto	Peças de roupa de adolescente	Peças de roupa tipo interior	Peças de roupa tipo inferior	Peças de roupa tipo tronco	Calçado	Data	Ações
----	---------------------	----------	--------	--------	-----------------------	---------------------------	--------------------------	-------------------------------	------------------------------	------------------------------	----------------------------	---------	------	-------

A Realizar

Id do pedido	Email do utilizador	Entidade	Pontos	Quilos	Número de peças total	Peças de roupa de criança	Peças de roupa de adulto	Peças de roupa de adolescente	Peças de roupa tipo interior	Peças de roupa tipo inferior	Peças de roupa tipo tronco	Calçado	Data	Ações
--------------	---------------------	----------	--------	--------	-----------------------	---------------------------	--------------------------	-------------------------------	------------------------------	------------------------------	----------------------------	---------	------	-------

Validado

Id do pedido	Email do utilizador	Entidade	Pontos	Quilos	Número de peças total	Peças de roupa de criança	Peças de roupa de adulto	Peças de roupa de adolescente	Peças de roupa tipo interior	Peças de roupa tipo inferior	Peças de roupa tipo tronco	Calçado	Data	Ações
--------------	---------------------	----------	--------	--------	-----------------------	---------------------------	--------------------------	-------------------------------	------------------------------	------------------------------	----------------------------	---------	------	-------

Cancelado

Id do pedido	Email do utilizador	Entidade	Pontos	Quilos	Número de peças total	Peças de roupa de criança	Peças de roupa de adulto	Peças de roupa de adolescente	Peças de roupa tipo interior	Peças de roupa tipo inferior	Peças de roupa tipo tronco	Calçado	Data	Ações
--------------	---------------------	----------	--------	--------	-----------------------	---------------------------	--------------------------	-------------------------------	------------------------------	------------------------------	----------------------------	---------	------	-------

Recicla Têxtil © 2024

- Dashboard, através do menu “Dashboard”:

RECICLA TÊXIL

[Minha Conta](#) [Gestão de Utilizadores](#) [Alteráveis](#) [Vales](#) [Pedidos de Recolha](#) [Dashboard](#) [Log Out](#)

Dashboard de Pedidos(Ano)

Selecione o ano:

Procurar

Voltar ao Menu Principal

Recicla Têxtil © 2024

- Os administradores da plataforma podem também registar e remover os vales das empresas com parceria através do menu “Vales”:

The screenshot shows the 'Criar vale' (Create voucher) form in the Recicla Têxtil platform. The form is titled 'Criar vale' and is located in the center of the page. It contains the following fields and buttons:

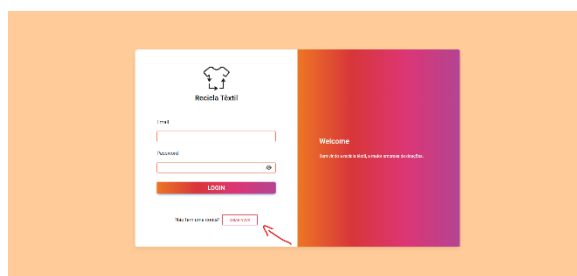
- Nome da empresa:** A text input field.
- Descrição do vale:** A text input field.
- Custo em pontos:** A text input field.
- Fotos:** A section with a button 'Escolher Ficheiros' and the text 'Nenhum ficheiro selecionado'.
- Buttons:** 'Criar vale' (orange), 'Apagar vales' (red), and 'Voltar' (grey).

The page header includes the logo 'RECICLA TÊXTEL' and a navigation menu with links: 'Minha Conta', 'Gestão de Utilizadores', 'Alterações', 'Vales', 'Pedidos de Recolha', 'Dashboard', and 'Log Out'. The footer shows 'Recicla Têxtil © 2024'.

### 3.2 Entidades beneficiadoras

As entidades interessadas têm acesso, a partir do FrontOffice, a:

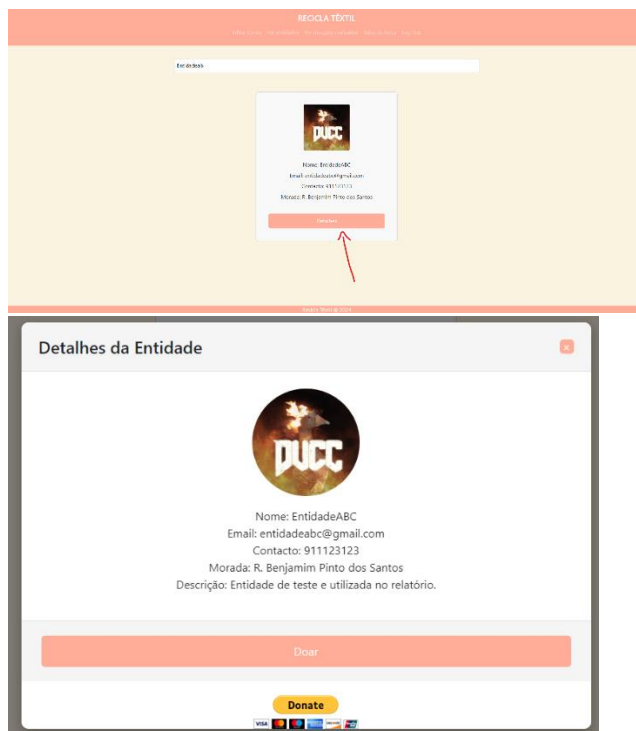
- Registarem-se na plataforma e ficarem eleitas para o processo de validação por parte da Recicla Têxtil, através dos seguintes passos:



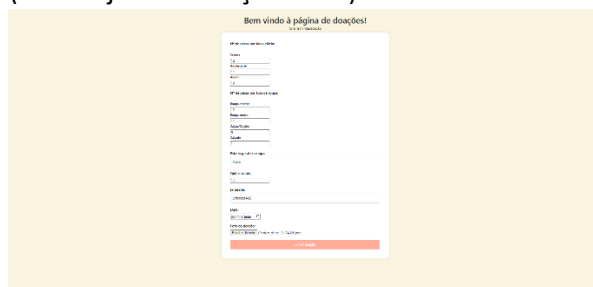




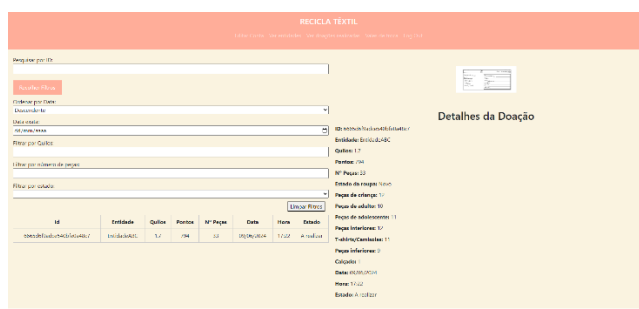
- Efetuar doações a entidades registadas no sistema através do menu “Ver entidades”, onde é possível pesquisar pela entidade pretendida e doar financeiramente, através da integração da API do PayPal, ou a nível têxtil:



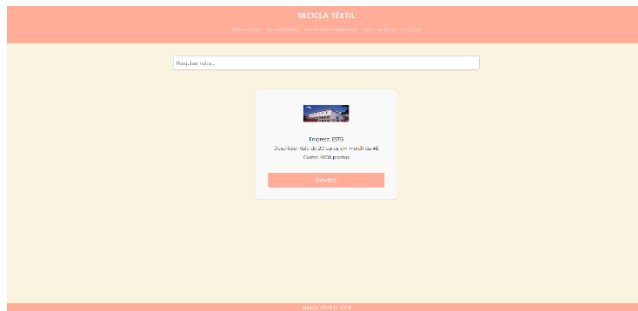
(Caso seja uma doação têxtil):



- Verificar o estado e informações de todas as doações que realizou previamente, através do menu "Ver doações realizadas":



- Resgatar vales de promoção com base nos pontos provenientes das doações, através do menu “Vales de troca”:



## 5. 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.

Nome:

Apelido:

Email:

Senha:

Confirmar Senha:

NIF:

Morada:

Cidade:

Código postal:

Código de entidade que o beneficiário:

Comentário (opcional):

Foto de Perfil (opcional):

### 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.

**Detalhes da Entidade**



Nome: EntidadeABC  
 Email: entidadeabc@gmail.com  
 Contacto: 911123123  
 Morada: R. Benjamim Pinto dos Santos  
 Descrição: Entidade de teste e utilizada no relatório.



## 6. 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:

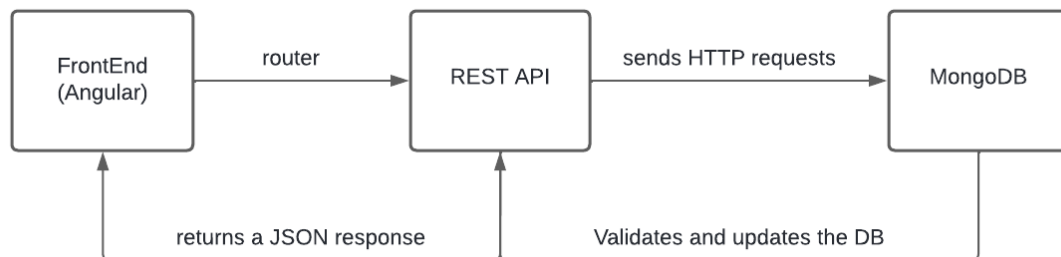
- Model
  - Implementados através do mongoose para definir os esquemas da base de dados.
- View
  - Desenvolvidas através de ficheiros .ejs.
  - Estas representam o frontend desta parte do projeto.
  - São utilizadas para receber e enviar os dados através do router para as respetivas rotas. Rotas essas 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.

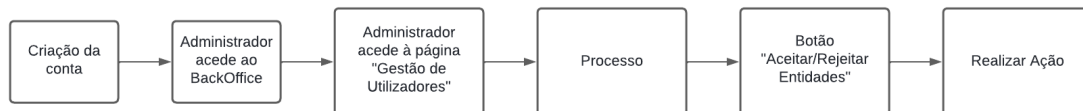
## 5.3 Flows e atividades

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

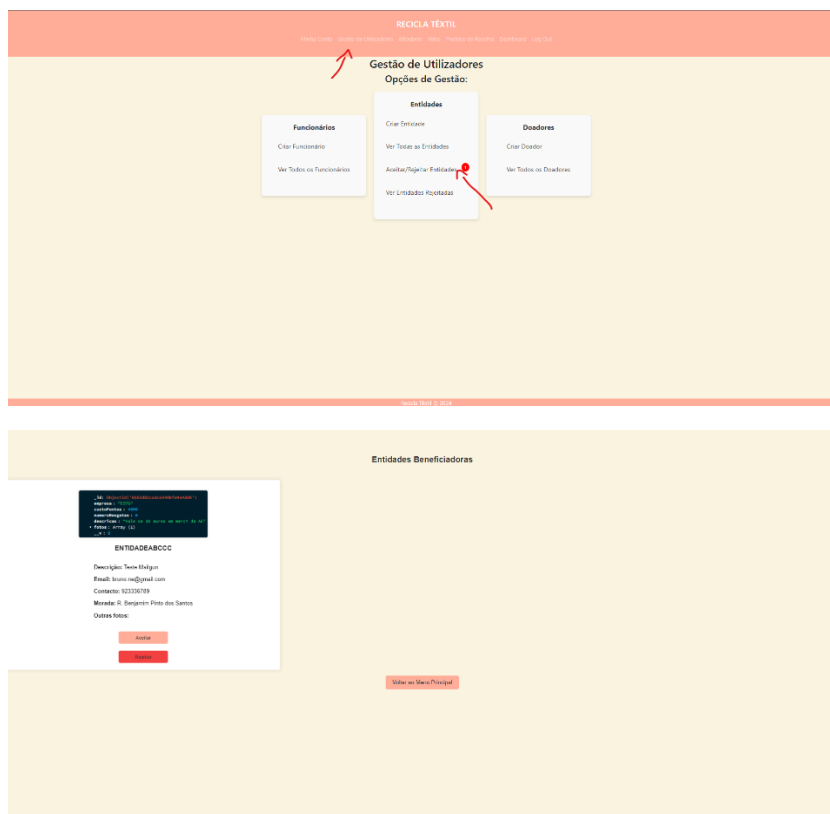
### 5.3.1 Entidades

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:

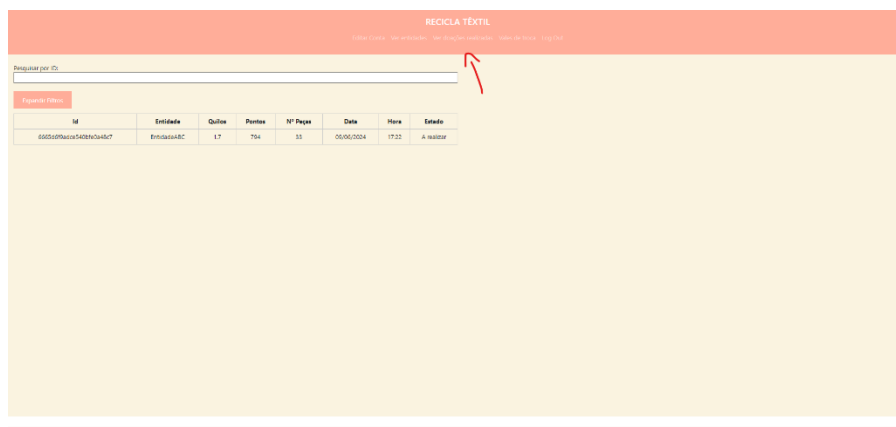
- Agendado: O doador submeteu uma doação e quando a data que ele definiu for verificada a doação passa a “Por iniciar”, durante este tempo o doador pode cancelar a doação. (Nota: Todas as doações registadas através do BackOffice são agendadas para o próprio dia e são iniciadas como “Por iniciar” automaticamente).
- 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.

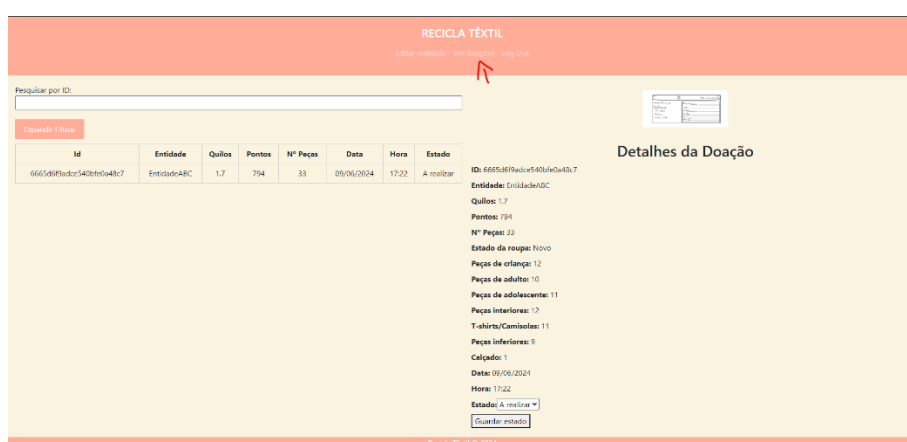
As datas atualizam quando ocorre uma alteração de estado na doação.

Pontos de acesso às doações realizadas:

- Pelos doadores:



- Pelas entidades:



- Pelos funcionários:



## 7. 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.