



## **Relatório**

**Projeto de Desenvolvimento de Software**

**Alunos:** Diogo Oliveira - 21111 | Joshua Jones - 21116 | Rui Peixoto - 21137

**Professor:**

Nuno Rodrigues

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, junho, 2022

## Índice

1. Introdução .....	4
1.1. Contextualização .....	4
1.2. Motivação e Objetivos.....	4
1.3. Estrutura do Documento.....	5
2. Iniciação.....	5
2.1. Diagramas.....	5
2.1.1. Casos de Uso .....	5
2.1.2. Diagrama de Componentes.....	6
2.1.3. Diagrama de Classes.....	7
2.1.4. Diagrama de Atividades.....	8
2.1.5. Diagrama de Sequência.....	9
2.1.6. Diagrama Entidade-Relação .....	10
2.2. Tecnologias.....	11
2.3. Requisitos não funcionais.....	11
2.4. Mockups .....	12
2.4.1. Página Inicial.....	12
2.4.2. Conta .....	12
2.4.3. Perfil .....	13
2.4.4. Produtos .....	13
2.4.5. Carrinho.....	14
3. Implementação .....	14
4. Output e Resultados.....	15
4.1. Back End .....	15
4.1.1. app.js – Ligação à Base de Dados.....	15
4.1.2. server.js – Atribuição da Porta .....	15
4.1.3. Services.....	16
4.1.4. Config .....	26
4.1.5. routes .....	27
4.1.7. test.....	37
4.2 Front-End.....	38
4.2.1 Pasta angular do checkout .....	38
4.2.2. Componente do checkout.....	38

4.2.3.	Serviços do checkout.....	39
4.2.4	App routing.....	41
4.2.5.	Ligação aos componentes .....	42
4.3.	Base de Dados .....	43
5.	Conclusão .....	44
5.1.	Lições Aprendidas .....	44

## 1. Introdução

Este relatório é do âmbito da unidade curricular de Projeto de Desenvolvimento de Software, e tem como finalidade apresentar o objetivo do trabalho, cujo tema é relativo à criação de uma empresa “FetchCode” que desenvolve um determinado projeto.

### 1.1. Contextualização

Este trabalho prático está inserido na unidade curricular de Projeto de Desenvolvimento de Software, do curso de Licenciatura em Engenharia de Sistemas Informáticos.

A realização deste trabalho prático consistiu na criação de uma empresa FetchCode, que visa ser uma empresa de venda de roupa. Através da criação de um website bem como toda uma base de dados e api capaz de assegurar todas as vendas e registos feitos no site, a FetchCode é a empresa ideal para se vestir bem da forma mais segura.

### 1.2. Motivação e Objetivos

Este projeto foi um desafio uma vez que nenhum membro do grupo tinha alguma vez desenvolvido um software tão complexo, por isso muito do conhecimento adquirido veio de diversas pesquisas, quer seja em alguns sites ou até mesmo em repositórios online.

O nosso foco foi implementar um projeto que ao mesmo tempo fosse complexo, mas fácil de ser usado pelo cliente menos experiente com o mundo tecnológico.

### 1.3. Estrutura do Documento

O relatório encontra-se dividido em 4 capítulos:

**1. Introdução** – Neste capítulo encontra-se um breve resumo sobre o que consiste este projeto e quais são os objetivos da realização do mesmo;

**2. Iniciação** – Planeamento inicial do projeto FetchCode;

**3. Implementação** - Na implementação encontra-se uma descrição completa de todos os pormenores do trabalho, explicando o funcionamento do mesmo;

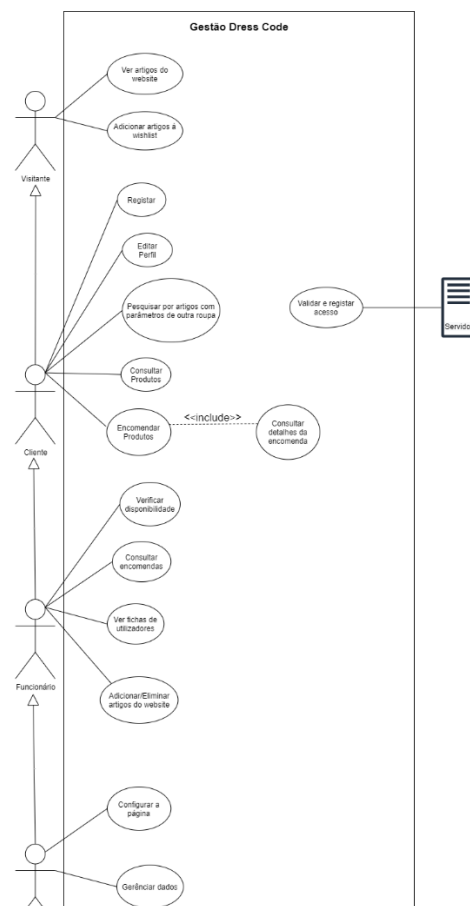
**4. Output e Resultado** – Nesta fase estão apresentadas imagens sobre o programa final bem como uma breve descrição das suas funcionalidades;

**5. Conclusão** – E por fim na conclusão fala sobre o que achamos deste trabalho prático, quer a nível de dificuldades encontradas a meio do projeto e apreciação final sobre o que este trabalho melhorou em nós.

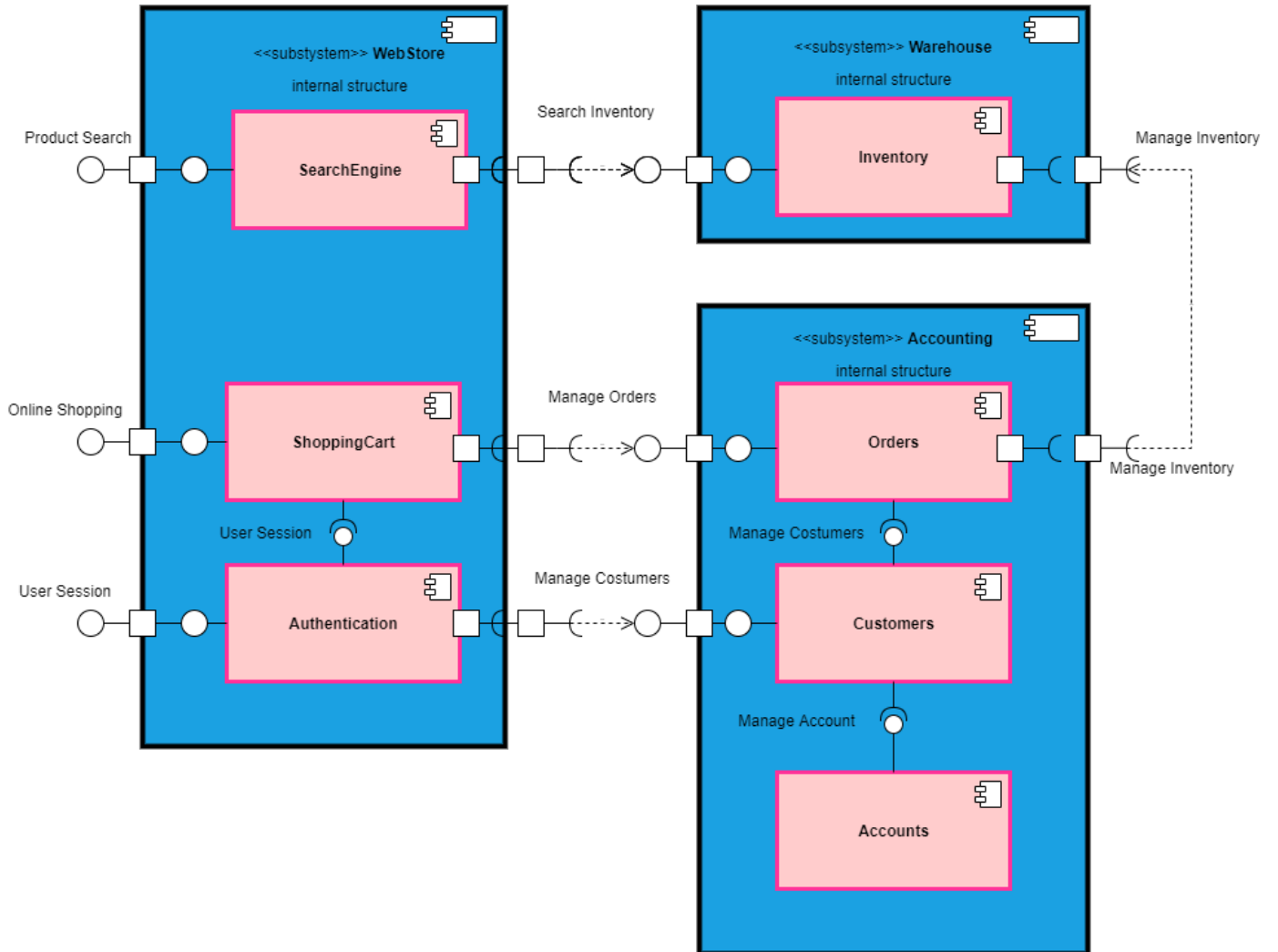
## 2. Iniciação

### 2.1. Diagramas

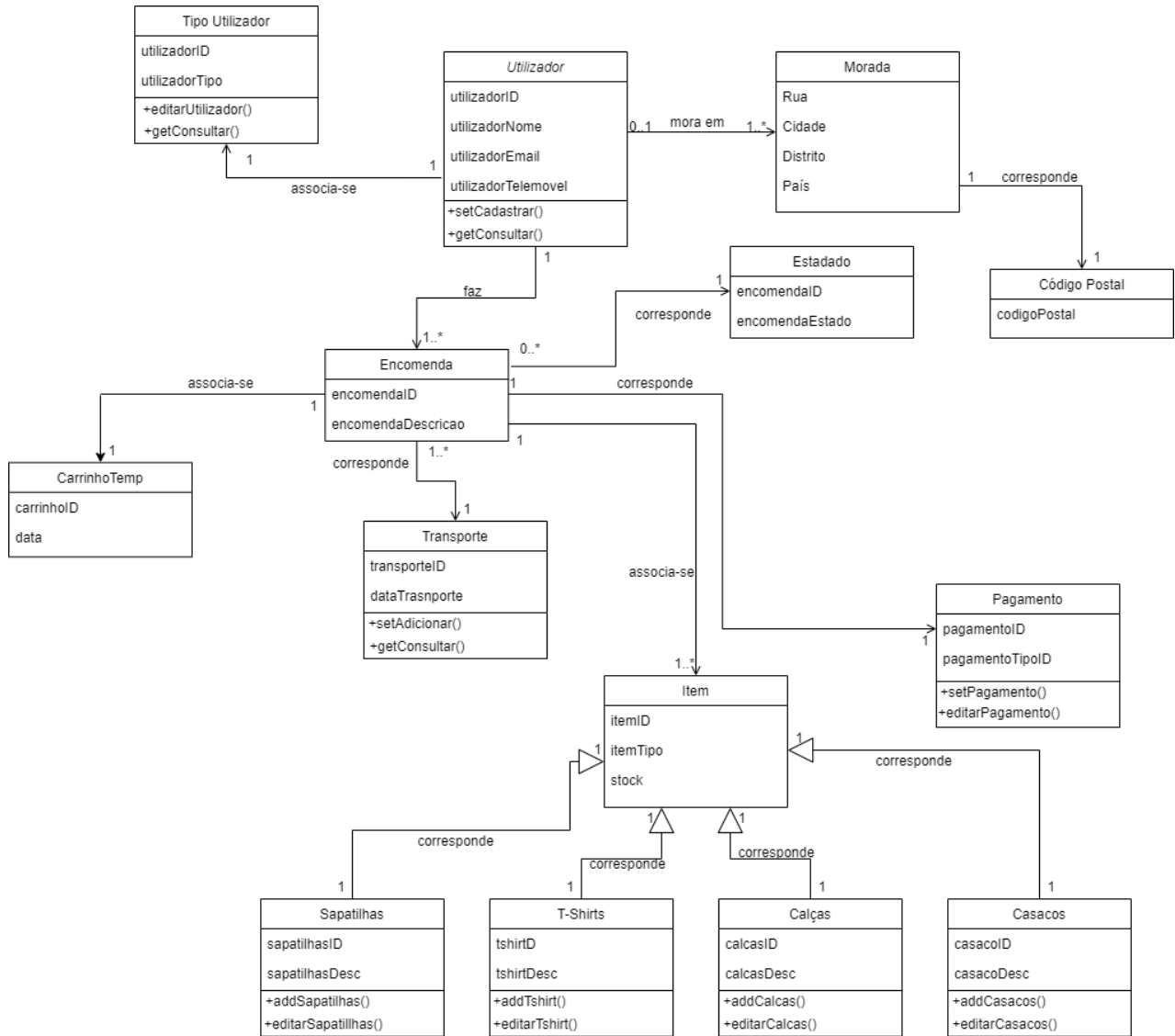
#### 2.1.1. Casos de Uso



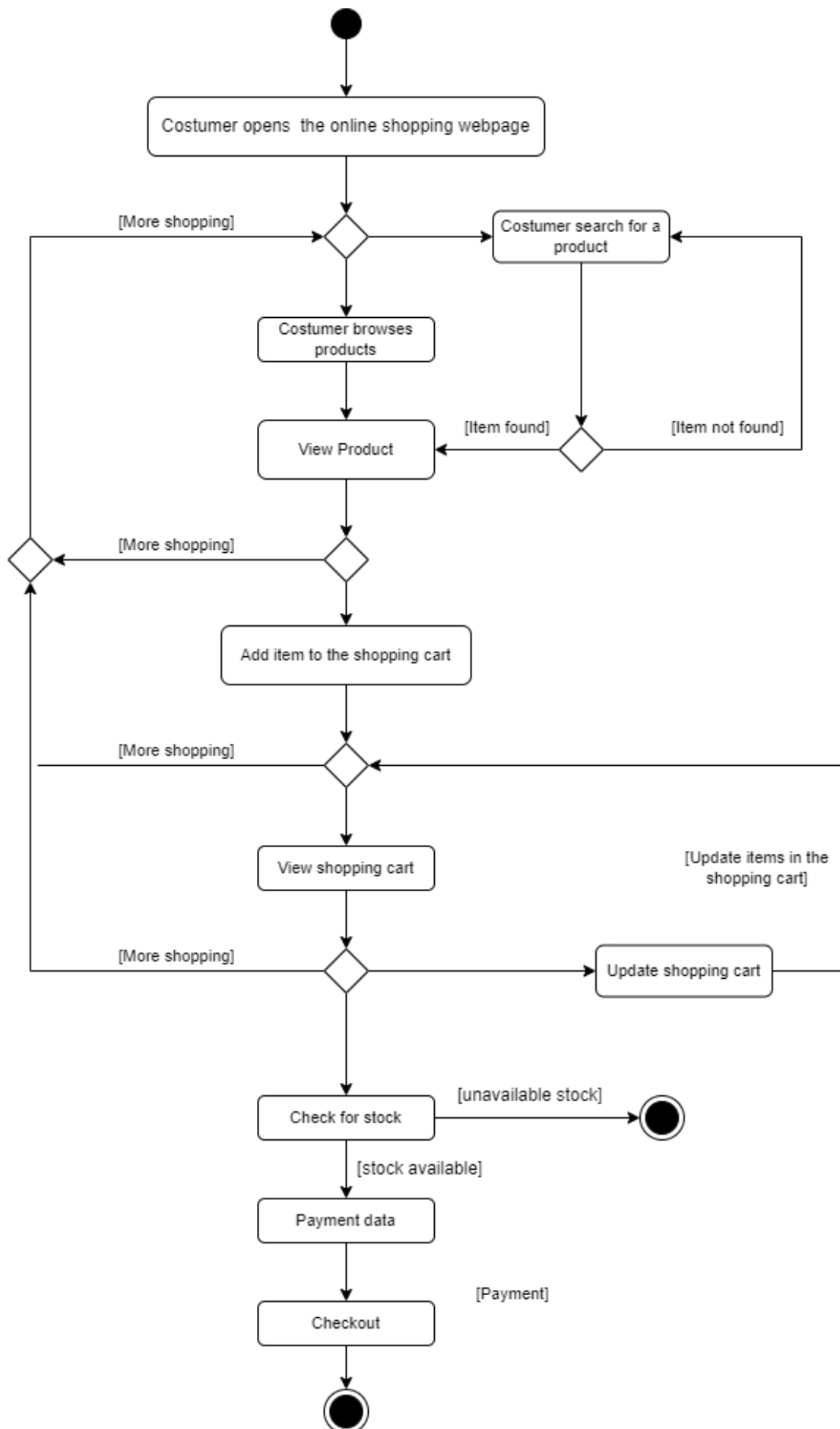
### 2.1.2. Diagrama de Componentes



### 2.1.3. Diagrama de Classes

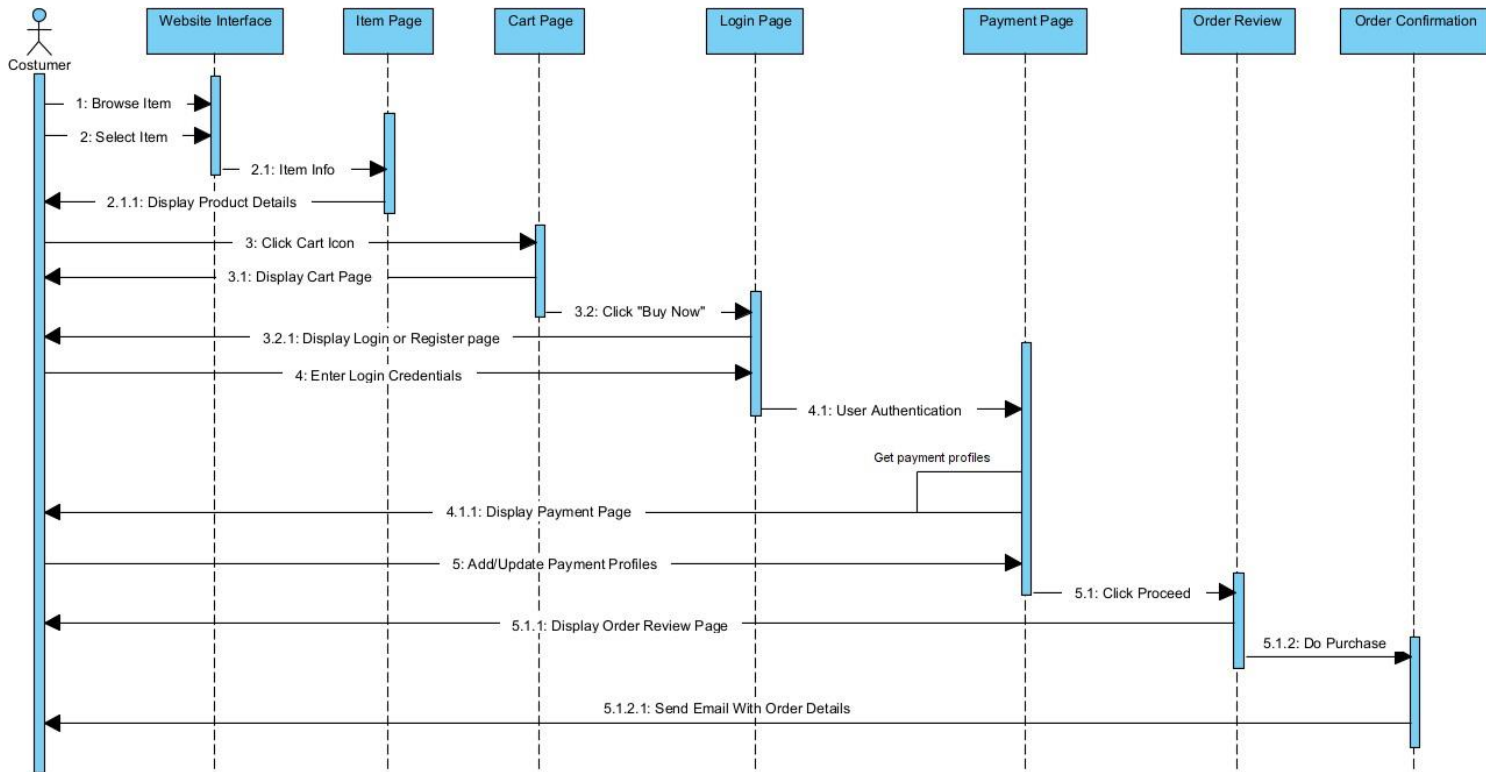


#### 2.1.4. Diagrama de Atividades

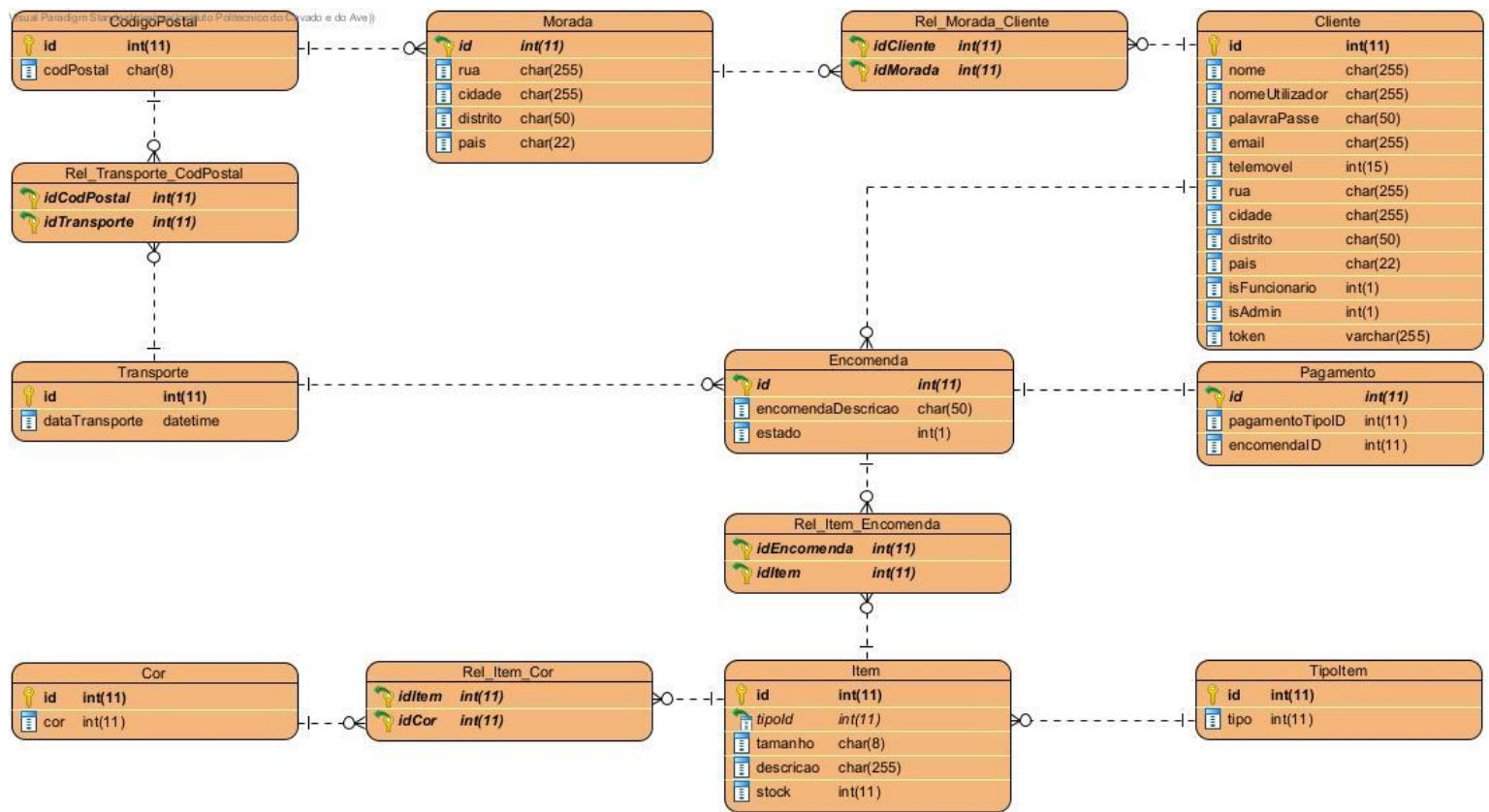




## 2.1.5. Diagrama de Sequência



## 2.1.6. Diagrama Entidade-Relação



## 2.2. Tecnologias

**Angular** - Frontend

**Node.js e MySQL** - Backend

**Azure DevOps** - Software de ALM

**Git** - Sistema de Controlo de versões

## 2.3. Requisitos não funcionais

**DESEMPENHO:** Deverá suportar n compras ao mesmo tempo;

**PORTABILIDADE:** Deverá funcionar em qualquer browser;

**COMPATIBILIDADE:** Deverá ser capaz de reconhecer se o acesso está a ser feito por um telemóvel e converter o site para o mesmo;

**DISPONIBILIDADE:** O sistema estará disponível 24h por dia;

**MANUTENÇÃO:** Modificações a qualquer erro encontrado deverão ser implementadas em menos de 24h;

**SEGURANÇA:** Passwords e outros dados sensíveis serão mascarados;

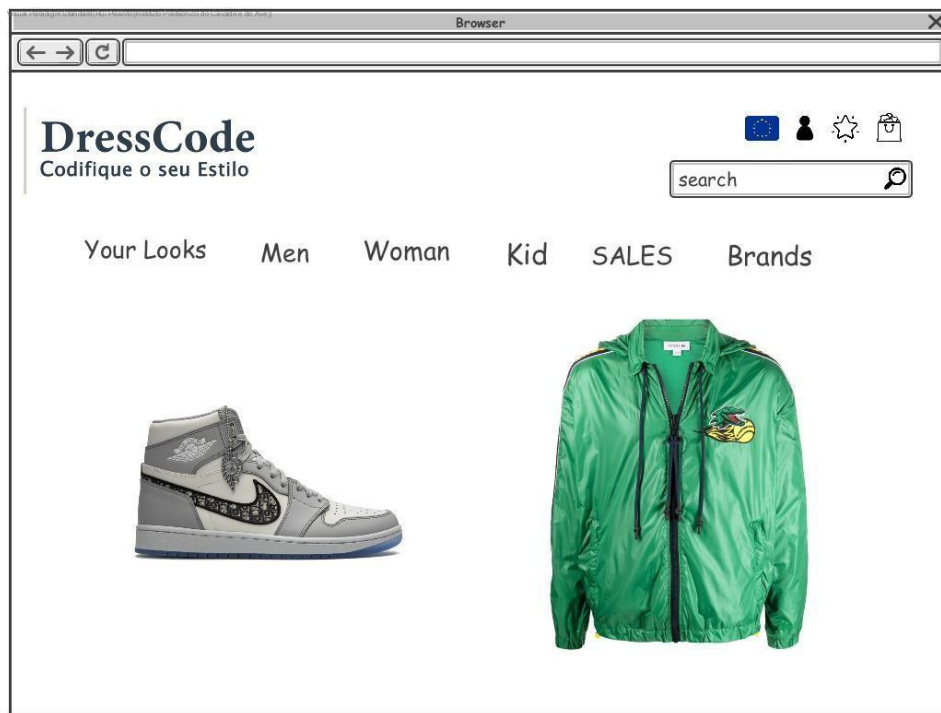
**CULTURA:** Adaptação da data e dos dados do pagamento consoante a região do utilizador;

**USABILIDADE:** Facilidade no uso;

**ESCALABILIDADE:** A possibilidade de crescer em número de clientes e faturação, sem precisar de aumentar os custos mensais;

## 2.4. Mockups

### 2.4.1. Página Inicial



### 2.4.2. Conta

The image shows a browser window displaying the 'CREATE NEW ACCOUNT' form on the DressCode website. The browser's address bar is empty. The website's header features the 'DressCode' logo with the tagline 'Codifique o seu Estilo' below it. To the right of the logo are four icons: a European Union flag, a user profile, a star, and a shopping bag. Below these icons is a search bar with the placeholder text 'search' and a magnifying glass icon. The main content area is titled 'CREATE NEW ACCOUNT' and contains the following form fields:

E-MAIL: <input type="text"/>	ADDRESS: <input type="text"/>
FIRST NAME: <input type="text"/>	COUNTRY: <input type="text"/>
LAST NAME: <input type="text"/>	CITY: <input type="text"/>
PASSWORD: <input type="text"/>	POSTAL CODE: <input type="text"/>
PHONE NUMBER: <input type="text"/>	
GENDER: <input type="text"/>	
DATE OF BIRTH: <input type="text"/>	

### 2.4.3. Perfil

Browser

**DressCode**  
Codifique o seu Estilo

search

CODE YOUR STYLE

SELECT YOUR FAVORITE

☐ CLASSIC ☐ STREETWEAR ☐ SPORT ☐ CREATIVE ☐ ELEGANT ☐ MODERN ☐ VINTAGE

SELECT YOUR FAVORITE COLOR

COLORS

YOUR AGE

YOUR GENDER

YOUR FAVORITE SEASON OF THE YEAR

YOUR FAVORITE PIECE OF CLOTHING


YOUR FAVORITE BRAND

### 2.4.4. Produtos

Browser

**DressCode**  
Codifique o seu Estilo

search



**The North Face**  
1996 Retro Nuptse padded jacket

609 €

SELECT SIZE

ADD TO BAG

THE

DELIVERY & FREE RETURNS

SIZE & FIT

#### 2.4.5. Carrinho



### 3. Implementação

Este trabalho tem como principal objetivo a criação de um projeto baseado numa empresa e desenvolver todo um sistema capaz de suprir as suas necessidades, front end e back end.

De seguida haverá uma explicação, com fotos, detalhada de todo o projeto, programação e site.

## 4. Output e Resultados

### 4.1. Back End

#### 4.1.1. app.js – Ligação à Base de Dados

```
1 const app = require('express')();
2 const consign = require('consign');
3 const cors = require('cors');
4 const mysql = require('mysql');
5
6 const knex = require('knex')({
7   client: 'mysql',
8   connection: {
9     host: 'fetchcodeserver.mysql.database.azure.com',
10    port: 3306,
11    database: 'fetchcode',
12    user: 'fetchcode',
13    password: 'sldyfgygIJS956_kjhvfddv87'
14  }
15 });
16
17 app.db = knex;
18 app.use(cors());
19
20 consign({ cwd: 'source', verbose: false })
21   .include('./config/passport.js')
22   .then('./config/middlewares.js')
23   .then('./services')
24   .then('./routes')
25   .then('./config/router.js')
26   .into(app);
27
28 app.get('/', (req, res) => {
29   res.status(200).send('FetchCode');
30 });
31
32 app.use((err, req, res, next) => {
33   const { name, message, stack } = err;
34   if (name === 'ValidationError') res.status(400).json({ error: message });
35   else if (name === 'ForbiddenError') res.status(403).json({ error: message });
36   else {
37     console.log(message);
38     res.status(500).json({ name, message, stack });
39   }
40   next(err);
41 });
42
43 module.exports = app;
```

É através deste trecho de código que fazemos a ligação a nossa API à Base de Dados.

#### 4.1.2. server.js – Atribuição da Porta

```
1 const app = require('./app');
2
3 const port = process.env.PORT || 3000;
4 const host = '0.0.0.0';
5
6 app.listen(port, host, () => console.log(`Server is Running! ${port}, Hosted in ${host}`));
7
```

Atribuição da Porta 3000 ao Servidor.

### 4.1.3. Services

#### 4.1.3.1. códigopostal.js

```
/**Selecionar todos os códigos postais */  
const getAll = async () => {  
  return app.db('codigopostal').select(['*']);  
};
```

É neste trecho de código que são selecionados todos os códigos postais.

```
/**Filtragem dos codigos postais por ID */  
const getAllCodPost = async(filter) => {  
  return app.db('codigopostal').where(filter).select(['*']);  
};
```

Seleciona os códigos postais por ID

```
/**Criação do registo de um novo código postal*/  
const create = async (req, res) => {  
  if(!req.codPostal) throw new ValidationError('O Codigo Postal é um campo obrigatorio');  
  
  const newCodigopostal = {...req};  
  return app.db('codigopostal').insert(newCodigopostal, ['codPostal']);  
};
```

É neste trecho de código que são criados os registos dos novos códigos postais

```
/**Atualizar para um novo código postal */  
const update = async (req, res) => {  
  console.log(req)  
  return app.db('codigopostal').insert([newCodigopostal, 'codPostal']);  
};
```

Atualização para um novo código postal

```
/**Remover um código postal */  
const remove = async (id) => {  
  return app.db('codigopostal').where({ id }).del();  
};
```

Remover um código postal



4.1.3.2. cor.js

```
/**Selecionar todas as cores */  
const getAll = async () => {  
  return app.db('cor').select(['*']);  
};
```

Seleciona todas as cores

```
/**Filtragem apenas as cores */  
const getAllIdCor = async (filter) => {  
  return app.db('cor').where(filter).select(['*']);  
};
```

Filtragem das cores

```
/**Criação do registo de uma nova cor */  
const create = async (req, res) => {  
  if(!req.cor) throw new ValidationError('A cor é um campo obrigatorio');  
  
  const newCor = {...req};  
  return app.db('cor').insert(newCor, ['cor']);  
};
```

Criação do registo de uma nova cor

```
/**Atualizar a cor selecionada */  
const update = async (req, res) => {  
  console.log(req)  
  
  return app.db('cor').insert([newCor, 'cor']);  
};
```

Atualização da cor selecionada

```
/**Remover uma cor */  
const remove = async (id) => {  
  return app.db('cor').where({ id }).del();  
};
```

Remoção de uma cor

### 4.1.3.3. encomenda.js

```
module.exports = (app) => {  
  /**Encontrar um certa encomenda */  
  const findOne = (filter= {}) => {  
    return app.db('encomenda').where(filter).select(['idEncomenda', 'encomendaDescricao', 'estado', 'Transporte_id', 'Pagamento_id', 'Utilizador_idUtilizador']);  
  };  
};
```

Encontra uma determinada encomenda

```
/**Selecionar todas as encomendas */  
const getAll = async () => {  
  return app.db('encomenda').select(['*']);  
};
```

Seleciona todas as encomendas

```
/**Criação do registo de uma nova encomenda */  
const create = async (req, res) => {  
  if(!req.encomendaDescricao) throw new ValidationError('A descrição da encomenda é um campo obrigatorio');  
  if(!req.estado) throw new ValidationError('O estado da encomenda é um campo obrigatorio');  
  
  const newEncomenda = {...req};  
  return app.db('encomenda').insert(newEncomenda, ['encomendaDescricao', 'estado', 'Transporte_id', 'Pagamento_id', 'Utilizador_idUtilizador']);  
};
```

Criação do registo de uma nova encomenda

```
/**Atualizar a encomenda selecionada */  
const update = async (req, res) => {  
  console.log(req)  
  
  return app.db('encomenda').insert([newEncomenda, 'encomendaDescricao', 'estado', 'Transporte_id', 'Pagamento_id', 'Utilizador_idUtilizador']);  
};
```

Atualização da encomenda selecionada

```
/**Remover uma encomenda */  
const remove = async (id) => {  
  return app.db('encomenda').where({ id }).del();  
};
```

Remoção de uma encomenda

### 4.1.3.4. Item.js

```
/**Selecionar todos os itens */  
const getAll = async () => {  
  return app.db('item').select(['*']);  
};
```

Seleciona todos os itens

```
/**Filtragem apenas dos itens por ID */  
const getAllID = async(filter) => {  
  return app.db('item').where(filter).select(['*']);  
};
```

Seleciona os itens por ID

```
/**Criação do registo de um novo item */  
const create = async (req, res) => {  
  if(!req.tipoId) throw new ValidationError('O tipo de ID é um campo obrigatorio');  
  if(!req.tamanho) throw new ValidationError('O tipo de tamanho é um campo obrigatorio');  
  if(!req.descricao) throw new ValidationError('A descrição é um campo obrigatorio');  
  if(!req.stock) throw new ValidationError('O stock é um campo obrigatorio');  
  if(!req.imagem) throw new ValidationError('A imagem é um campo obrigatorio');  
  
  const newItem = {...req};  
  return app.db('item').insert(newItem, ['tipoId', 'tamanho', 'descricao', 'stock', 'imagem', 'tipoItem_id']);  
};
```

Criação do registo de um novo item

```
/**Atualizar o item selecionado */  
const update = (id, item) => {  
  return app.db('item').where({ id }).update(item, ['tipoId', 'tamanho', 'descricao', 'stock', 'imagem', 'tipoItem_id']);  
};
```

Atualização do item selecionado

```
/**Remover um item */  
const remove = async(id) => {  
  return app.db('item').where(id).del();  
};
```

Remoção de uma encomenda

### 4.1.3.5. Morada.js

```
/**Encontrar uma certa morada */  
const findOne = (filter = {}) => {  
  return app.db('morada').where(filter).select(['id', 'rua', 'cidade', 'distrito', 'pais', 'CodigoPostal_id']);  
}
```

Encontra uma determinada morada

```
/**Selecionar todas as moradas*/  
const getAll = async () => {  
  return app.db('morada').select(['*']);  
};
```

Seleciona todas as moradas

```
/**Criação do registo de uma nova morada */  
const create = async (req, res) => {  
  if(!req.rua) throw new ValidationError('A rua é um campo obrigatorio');  
  if(!req.cidade) throw new ValidationError('A cidade é um campo obrigatorio');  
  if(!req.distrito) throw new ValidationError('O distrito é um campo obrigatorio');  
  if(!req.pais) throw new ValidationError('O pais é um campo obrigatorio');  
  
  const newMorada = {...res};  
  //console.log(['rua', 'cidade', 'distrito', 'pais', 'CodigoPostal_id']);  
  return app.db('morada').insert(newMorada, ['rua', 'cidade', 'distrito', 'pais', 'CodigoPostal_id']);  
};
```

Criação do registo de uma nova morada

```
/**Atualizar a morada selecionada */  
const update = async (id) => {  
  //return app.db('morada').insert([newMorada, 'rua', 'cidade', 'distrito', 'pais', 'CodigoPostal_id']);  
  const newMorada = {...req};  
  return app.db('morada').where( {id} ).insert(newMorada, ['rua', 'cidade', 'distrito', 'pais', 'CodigoPostal_id']);  
};
```

Atualização da morada selecionada

```
/**Remover uma morada */  
const remove = async (id) => {  
  return app.db('morada').where({ id }).del();  
};
```

Remoção de uma morada

### 4.1.3.6. Pagamento.js

```
/**Encontrar um certo pagamento*/  
const findOne = (filter = {}) => {  
  return app.db('pagamento').where(filter).select(['id', 'pagamentoTipoID', 'encomendaID']);  
}
```

Encontra um determinado pagamento

```
/**Encontrar todos os pagamentos */  
const findAll = async (filter = {}) => {  
  return app.db('pagamento').where(filter).select(['id', 'pagamentoTipoID', 'encomendaID']);  
};
```

Encontra todos os pagamentos

```
/**Criação do registo de um novo pagamento */  
const create = async (req, res) => {  
  if(!req.pagamentoTipoID) throw new ValidationError('O tipo de pagamento é um campo obrigatorio');  
  if(!req.encomendaID) throw new ValidationError('A encomenda é um campo obrigatorio');  
  
  const newPagamento = {...req};  
  return app.db('pagamento').insert(newPagamento, ['pagamentoTipoID', 'encomendaID']);  
};
```

Criação do registo de um novo pagamento

```
/**Atualizar o pagamento selecionado */  
const update = async (req, res) => {  
  const newPagamento = {...req};  
  return app.db('pagamento').insert(newPagamento, ['pagamentoTipoID', 'encomendaID']);  
};
```

Atualização do pagamento selecionado

```
/**Remover um pagamento */  
const remove = async (id) => {  
  return app.db('pagamento').where({ id }).del();  
};
```

Remoção de um pagamento

### 4.1.3.7. Tipoltem.js

```
/**Encontrar um certo tipo de item*/  
const findOne = (filter = {}) => {  
  return app.db('tipoitem').where(filter).select(['id', 'tipo']);  
}
```

Encontra um determinado tipo de item

```
/**Encontrar todos os tipos de item */  
const findAll = async (filter = {}) => {  
  return app.db('tipoitem').where(filter).select(['id', 'tipo']);  
};
```

Encontra todos os tipos de item

```
/**Criação de um novo tipo de item */  
const create = async (req, res) => {  
  if(!req.tipo) throw new ValidationError('O tipo é um campo obrigatorio');  
  
  const newTipoItem = {...req};  
  return app.db('tipoitem').insert(newTipoItem, ['tipo']);  
};
```

Criação do registo de um novo tipo de item

```
/**Atualizar o tipo de item selecionado */  
const update = async (req, res) => {  
  const newTipoItem = {...req};  
  return app.db('tipoitem').insert(newTipoItem, ['tipo']);  
};
```

Atualização do tipo de item selecionado

```
/**Remover um tipo de item */  
const remove = async (id) => {  
  return app.db('tipoitem').where({ id }).del();  
};
```

Remoção de um tipo de item

4.1.3.8. Transporte.js

```
/**Encontrar um certo tipo de transporte*/  
const findOne = (filter = {}) => {  
  return app.db('transporte').where(filter).select(['id', 'dataTransporte']);  
}
```

Encontra um determinado tipo de transporte

```
/**Selecionar todos os transportes*/  
const getAll = async () => {  
  return app.db('transporte').select(['*']);  
};
```

Seleciona todos os transportes

```
/**Criação do registo de um novo transporte */  
const create = async (req, res) => {  
  if(!req.dataTransporte) throw new ValidationError('A data do transporte é um campo obrigatorio');  
  
  const newTransporte = {...req};  
  return app.db('transporte').insert(newTransporte, ['dataTransporte']);  
};
```

Criação do registo de um novo transporte

```
/**Atualizar o transporte selecionado */  
const update = async (req, res) => {  
  const newTransporte = {...req};  
  return app.db('transporte').insert(newTransporte, ['dataTransporte']);  
};
```

Atualização do transporte selecionado

```
/**Remover um transporte */  
const remove = async (id) => {  
  return app.db('transporte').where({ id }).del();  
};
```

Remoção de um transporte

### 4.1.3.9. Utilizador.js

```
/**Selecionar todos os utilizadores*/  
const getAll = async () => {  
  return app.db('utilizador').select(['*']);  
};
```

Seleciona todos os utilizadores

```
/**Selecionar todos por nome */  
const getAllName = async () => {  
  return app.db('utilizador').select(['idUtilizador', 'nome']);  
};
```

Seleciona todos os utilizadores por nome

```
/**Encontrar um certo utilizador*/  
const findOne = (filter = {}) => {  
  return app.db('utilizador').where(filter).select(['idUtilizador', 'nome', 'nomeUtilizador', 'palavraPasse', 'email', 'telemovel', 'rua', 'cidade', 'distrito', 'pais', 'isFuncionario', 'isAdmin', 'token']);  
};
```

Encontra um determinado utilizador

```
/**Encriptação de password */  
const getPasswordHash = (password) => {  
  const salt = bcrypt.genSaltSync(10);  
  return bcrypt.hashSync(password, salt);  
};
```

Encriptação da password



## Relatório de PDS

```
/**Criação do registo de um novo utilizador */
const create = async(req, res) => {
  if (!req.nome) throw new ValidationError('O nome é um campo obrigatorio');
  if (!req.nomeUtilizador) throw new ValidationError('O nome de utilizador é um campo obrigatorio');
  if (!req.palavraPasse) throw new ValidationError('A palavra-passe é um campo obrigatorio');
  if (!req.email) throw new ValidationError('O email é um campo obrigatorio');
  if (!req.telemovel) throw new ValidationError('O número de telemóvel é um campo obrigatorio');
  if (!req.rua) throw new ValidationError('A rua é um campo obrigatorio');
  if (!req.cidade) throw new ValidationError('A cidade é um campo obrigatorio');
  if (!req.districto) throw new ValidationError('O distrito é um campo obrigatorio');
  if (!req.pais) throw new ValidationError('O país é um campo obrigatorio');
  if (!emailRegex.test(req.email)) throw new ValidationError('O email não segue os padrões convencionais!');
  //if (!passwordRegex.test(req.password)) throw new ValidationError('A password não segue os padrões convencionais!');

  // const userDBEmail = await findOne({ email: req.email });
  // if (userDBEmail) throw new ValidationError('Email duplicado');

  // const userDBNTelemovel = await findOne({ telemovel: req.telemovel });
  // if (userDBNTelemovel) throw new ValidationError('Número telemóvel duplicado');

  const newUser = {...req };
  newUser.palavraPasse = getPasswordHash(req.palavraPasse);

  return app.db('utilizador').insert(newUser, ['nome', 'nomeUtilizador', 'palavraPasse', 'email', 'telemovel', 'rua', 'cidade', 'distrito', 'pais']);
};
```

Criação do registo de um novo utilizador

```
/**Atualizar o utilizador selecionado */
const update = async(req, res) => {
  const newUser = {...req };

  return app.db('utilizador').where(req).update(newUser, ['nome']);
};
```

Atualização do utilizador selecionado

```
/**Remover um utilizador */
const remove = async(id) => {
  return app.db('utilizador').where({ idUtilizador: id }).del();
};
```

Remoção de um utilizador

```
/**Recuperação de password */
const forgotPassword = async(req, res) => {
  if (!req.email) throw new ValidationError('O email é um campo obrigatório');
  if (!req.telemovel) throw new ValidationError('O número telemóvel é um campo obrigatório');
  if (!req.palavraPasse) throw new ValidationError('A password é um campo obrigatório');
  if (!req.confirmpassword) throw new ValidationError('A password é um campo obrigatório');
  if (req.palavraPasse !== req.confirmpassword) throw new ValidationError('As password têm que coincidir');
  if (!emailRegex.test(req.email)) throw new ValidationError('O email não segue os padrões convencionais!');
  if (!passwordRegex.test(req.palavraPasse)) throw new ValidationError('A password não segue os padrões convencionais!');
  req.palavraPasse = getPasswordHash(req.palavraPasse);

  const result = await app.db('utilizador').where('email', req.email).first();

  if (result.telemovel === req.telemovel) {
    return app.db('utilizador').where('telemovel', req.telemovel).update('palavraPasse', req.palavraPasse);
  }
  throw new ValidationError('Verifique os seus detalhes!');
};
```

Recuperação de password

### 4.1.4. Config

#### 4.1.4.1. router.js

```
module.exports = (app) => {  
  app.use('/auth', app.routes.utilizadores);  
  
  const secureRouter = express.Router();  
  
  secureRouter.use('/utilizador', app.routes.utilizadores);  
  secureRouter.use('/codigoPostal', app.routes.codigosPostais);  
  secureRouter.use('/cor', app.routes.cores);  
  secureRouter.use('/encomenda', app.routes.encomendas);  
  secureRouter.use('/item', app.routes.itens);  
  secureRouter.use('/morada', app.routes.moradas);  
  secureRouter.use('/pagamento', app.routes.pagamentos);  
  secureRouter.use('/tipoItem', app.routes.tiposItens);  
  secureRouter.use('/transporte', app.routes.transportes);  
  secureRouter.use('/compra', app.routes.checkouts);  
  
  app.use('/v1', secureRouter);  
};
```

#### 4.1.4.2. routes.js

```
module.exports = (app) => {  
  // AUTH  
  app.route('/auth/signin')  
    .get(app.routes.utilizadores.getAll);  
  
  app.route('/auth/signup')  
    .post(app.routes.utilizadores.signup);  
  
  app.route('/auth/forget-password')  
    .put(app.routes.users.forgotPassword());  
  
  // UTILIZADORES  
  app.route('/utilizador')  
    .all(app.config.passport.authenticate())  
    .get(app.routes.utilizadores.getAll)  
    .get(app.routes.utilizadores.getAllName)  
    .post(app.routes.utilizadores.create);  
  
  app.route('/utilizador/:id')  
    .all(app.config.passport.authenticate())  
    .get(app.routes.utilizadores.findOne)  
    .put(app.routes.utilizadores.update)  
    .delete(app.routes.utilizadores.remove);  
  
  // CODIGO POSTAL  
  app.route('/codigoPostal')  
    .all(app.config.passport.authenticate())  
    .get(app.routes.codigosPostais.getAll)  
    .post(app.routes.codigosPostais.create)  
    .delete(app.routes.codigosPostais.remove);  
  
  app.route('/codigoPostal/:id')  
    .all(app.config.passport.authenticate())  
    .get(app.routes.codigosPostais.getAllCodPost)  
    .post(app.routes.codigosPostais.create)  
    .put(app.routes.codigosPostais.update)  
    .delete(app.routes.codigosPostais.remove);  
}
```

#### 4.1.5. routes

##### 4.1.6.1. checkout.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.checkout.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/', async (req, res, next) => {
  try {
    const result = await app.services.checkout.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.get('/:id', (req, res, next) => {
  app.services.checkout.getAllID({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.checkout.remove({ id: req.params.id })
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.put('/:id', (req, res, next) => {
  app.services.checkout.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

4.1.6.2. codigosPostais.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.codigoPostal.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/', async (req, res, next) => {
  try {
    const result = await app.services.codigoPostal.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.get('/:id', (req, res, next) => {
  app.services.codigoPostal.getAllCodPost({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.codigoPostal.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

4.1.6.3. cores.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.cor.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/create', async (req, res, next) => {
  try {
    const result = await app.services.cores.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.get('/:id', (req, res, next) => {
  app.services.cor.getAllIdCor({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.cor.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.6.4. encomendas.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.encomenda.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.encomenda.findOne({ idEncomenda: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/create', async (req, res, next) => {
  try {
    const result = await app.services.encomenda.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.put('/:id', (req, res, next) => {
  app.services.encomenda.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.encomenda.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.6.5. itens.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.item.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.item.getAllID({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.item.remove({ id: req.params.id })
    .then((result) => res.status(204).json(result))
    .catch((err) => next(err));
});

router.post('/', async (req, res, next) => {
  try {
    const result = await app.services.item.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.put('/:id', (req, res, next) => {
  app.services.item.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.6.6. moradas.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.morada.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.morada.findOne({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/create', async (req, res, next) => {
  try {
    const result = await app.services.morada.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.put('/:id', (req, res, next) => {
  app.services.morada.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.morada.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```



4.1.6.7. pagamentos.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.transaction.findAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.transaction.findOne({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/', async (req, res, next) => {
  try {
    const result = await app.services.transaction.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.put('/:id', (req, res, next) => {
  app.services.transaction.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.pagamento.remove({ id: req.params.id })
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.6.8. tiposItens.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.tipoItem.findAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.tipoItem.findOne({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/', async (req, res, next) => {
  try {
    const result = await app.services.tipoItem.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.put('/:id', (req, res, next) => {
  app.services.tipoItem.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.tipoItem.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.6.9. transportes.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.transporte.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.transporte.findOne({ id: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/', async (req, res, next) => {
  try {
    const result = await app.services.transporte.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.put('/:id', (req, res, next) => {
  app.services.transporte.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.delete('/:id', (req, res, next) => {
  app.services.transporte.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.6.10. utilizadores.js

```
const router = express.Router();

router.get('/', (req, res, next) => {
  app.services.utilizador.getAll()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/', (req, res, next) => {
  app.services.utilizador.getAllName()
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.get('/:id', (req, res, next) => {
  app.services.utilizador.findOne({ idUtilizador: req.params.id })
    .then((result) => res.status(200).json(result))
    .catch((err) => next(err));
});

router.post('/signin', (req, res, next) => {
  app.services.utilizador.findOne({ email: req.body.email })
    .then((user) => {
      if (!user) throw new ValidationError('Autenticação inválida! #2');

      const {
        idUtilizador, email, telemovel,
      } = user;

      if (bcrypt.compareSync(req.body.password, user.password)) {
        const payload = {
          idUtilizador,
          email,
          telemovel,
        };

        const token = jwt.encode(payload, secret);
        res.status(200).json({ token });
      } else throw new ValidationError('Autenticação inválida!');
    })
    .catch((err) => next(err));
});
```

```
router.put('/:id', (req, res, next) => {
  app.services.utilizador.update(req.params.id, req.body)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

router.post('/signup', async (req, res, next) => {
  try {
    const result = await app.services.utilizador.create(req.body);
    return res.status(201).json(result[0]);
  } catch (err) {
    return next(err);
  }
});

router.delete('/:id', (req, res, next) => {
  app.services.utilizador.remove(req.params.id)
    .then((result) => res.status(204).json(result[0]))
    .catch((err) => next(err));
});

return router;
```

## 4.1.7. test

4.1.7.1. *checkout.js*

```
test('Test #1 - Listar os checkouts', () => {
  return request(app).get(ROUTE)
    .set('authorization', `bearer ${user.token}`)
    .then((res) => {
      expect(res.status).toBe(200);
      expect(res.body.length).toBeGreaterThan(0);
    });
});

test('Test #1.1 - Listar os checkouts por ID', () => {
  return request(app).get(`${ROUTE}/${checkoutA.id}`)
    .set('authorization', `bearer ${user.token}`)
    .then((res) => {
      expect(res.status).toBe(200);
    });
});

test('Test #3 - Apagar checkout', () => {
  return app.db('compra').insert({
    nome: 'alo',
    email: 'olo@gmail.com',
    rua: 'Vila Frescainha',
    cidade: 'Barcelos',
    distrito: 'Braga',
    codPostal: '4700-289',
    nomeCartao: 'Andre',
    cartaoCredito: '1122-2233-3344-4455',
    mesValidade: 'Agosto',
    anoValidade: '1999',
    cvv: '123',
  }, ['id']).then((result) => request(app).delete(`${ROUTE}/${result[0].id}`)
    .set('authorization', `bearer ${user.token}`)
    .then((res) => {
      expect(res.status).toBe(204);
    }));
});
```

**PASS** test/routes/checkout.test.js

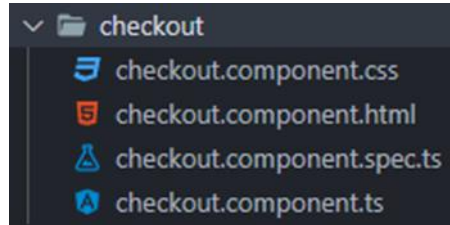
✓ Test #1 - Listar os checkouts (68 ms)

✓ Test #1.1 - Listar os checkouts por ID (55 ms)

✓ Test #3 - Apagar checkout (123 ms)

### 4.2 Front-End

#### 4.2.1 Pasta angular do checkout



#### 4.2.2. Componente do checkout

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { CheckoutService } from '../services/checkout.service';

You, 2 minutes ago | 1 author (You)
@Component({
  selector: 'app-checkout',
  templateUrl: './checkout.component.html',
  styleUrls: ['./checkout.component.css']
})
export class CheckoutComponent implements OnInit {

  constructor(private CheckoutService: CheckoutService) { }
  Checkouts;
  ngOnInit(): void {
    this.GetAllCheckouts()
  }

  GetAllCheckouts() {
    this.CheckoutService.GetAllCheckouts().subscribe(checkouts => this.Checkouts = checkouts);
  }

  modeUpdate: boolean = false;
  closeModal: boolean = true;

  onSubmit(checkoutDetail: NgForm) {
    this.CheckoutService.PostCheckout(checkoutDetail.form.value).subscribe(
      data => console.log(data),
      error => console.log(error)
    );
  }

  updateLast(checkoutDetail : NgForm){
    this.CheckoutService.PutCheckoutLast(this.Checkouts[this.Checkouts.length - 1].id, checkoutDetail.form.value).subscribe(
      data => console.log(data),
      error => console.log(error)
    );
    this.modeUpdate = this.CheckoutService.modeUpdate;
  }

  cancelUpdate(checkoutDetail : NgForm){
    this.CheckoutService.cancelUpdate(checkoutDetail);
    this.modeUpdate = this.CheckoutService.modeUpdate;
  }

  deleteLastCheckout(){
    this.CheckoutService.DeleteCheckoutID(this.Checkouts[this.Checkouts.length - 1].id).subscribe(
      data => console.log(data),
      error => console.log(error)
    );
  }
}
```

## 4.2.3. Serviços do checkout

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Checkout } from '../checkout';
import { NgForm } from '@angular/forms';
import { IdentifierName } from '@angular/compiler';

You, 3 minutes ago | 1 author (You)
@Injectable({
  providedIn: 'root'
})
export class CheckoutService {

  constructor(private http: HttpClient) { }

  ..checkouts;
  modeUpdate: boolean = false;

  GetAllCheckouts() : Observable<unknown>{
    return this.http.get<unknown>("http://localhost:3000/v1/compra")
  }

  PostCheckout(body) {
    return this.http.post("http://localhost:3000/v1/compra", body)
  }

  DeleteCheckoutID(id){
    console.log(id)
    return this.http.delete("http://localhost:3000/v1/compra/" + id)
  }

  PutCheckoutLast(id, body) {
    console.log(id)
    return this.http.put("http://localhost:3000/v1/compra/" + id, body)
  }

  createCheckout(checkoutDetail: Checkout) {
    this.checkouts.push(checkoutDetail);
    console.log(this.checkouts);
  }
}
```

```
startUpdateLast(checkoutDetail : NgForm){
  this.modeUpdate = true;
  const lastCheckout = this.checkouts[this.checkouts.length - 1];

  checkoutDetail.form.patchValue({
    fullName: lastCheckout.fullName,
    email: lastCheckout.email,
    address: lastCheckout.address,
    city: lastCheckout.city,
    state: lastCheckout.state,
    postalCode: lastCheckout.postalCode,
    cardName: lastCheckout.cardName,
    creditCardNumber: lastCheckout.creditCardNumber,
    expirationMonth: lastCheckout.expirationMonth,
    expirationYear: lastCheckout.expirationYear,
    cvv: lastCheckout.cvv,
  })
}

updateLast(checkoutDetail : NgForm){
  this.checkouts[this.checkouts.length - 1] = checkoutDetail.form.value;
  this.modeUpdate = false;
  checkoutDetail.reset();
}

cancelUpdate(checkoutDetail : NgForm){
  this.modeUpdate = false;
  checkoutDetail.reset();
}

deleteLastCheckout(){
  this.checkouts.pop();
}
```



#### 4.2.4 App routing

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { EncomendasComponent } from './encomendas/encomendas.component';
import { PerfilComponent } from './perfil/perfil.component';
import { CarrinhoComponent } from './carrinho/carrinho.component';
import { CheckoutComponent } from './checkout/checkout.component';
import { SlideshowComponent } from './slideshow/slideshow.component';
import { CriarContaComponent } from './criar-conta/criar-conta.component';
import { PaginaInicialComponent } from './pagina-inicial/pagina-inicial.component';
import { FavoritosComponent } from './favoritos/favoritos.component';

const routes: Routes = [
  { path: '', component: PaginaInicialComponent },
  { path: 'encomendas', component: EncomendasComponent },
  { path: 'perfil', component: PerfilComponent },
  { path: 'carrinho', component: CarrinhoComponent },
  { path: 'checkout', component: CheckoutComponent },
  { path: 'slideshow', component: SlideshowComponent },
  { path: 'criarconta', component: CriarContaComponent },
  { path: 'paginainicial', component: PaginaInicialComponent },
  { path: 'favoritos', component: FavoritosComponent },
];

You, 6 days ago | 1 author (You)
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

## 4.2.5. Ligação aos componentes

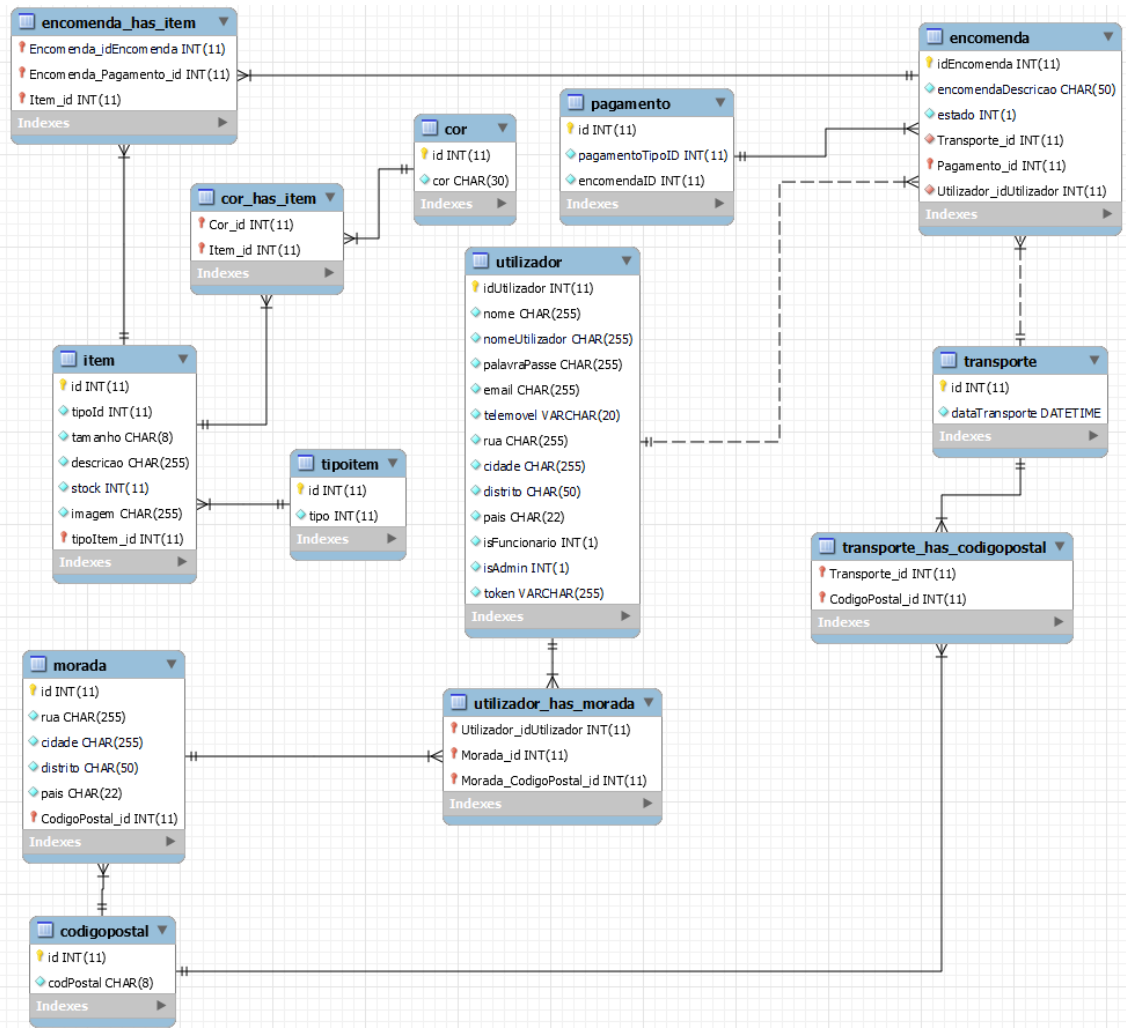
```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { FooterComponent } from './footer/footer.component';
import { EncomendasComponent } from './encomendas/encomendas.component';
import { PerfilComponent } from './perfil/perfil.component';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { CarrinhoComponent } from './carrinho/carrinho.component';
import { CheckoutComponent } from './checkout/checkout.component';
import { SlideshowComponent } from './slideshow/slideshow.component';
import { CriarContaComponent } from './criar-conta/criar-conta.component';
import { FavoritosComponent } from './favoritos/favoritos.component';
import { PaginaInicialComponent } from './pagina-inicial/pagina-inicial.component';
```

You, 5 days ago | 1 author (You)

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    FooterComponent,
    EncomendasComponent,
    PerfilComponent,
    CarrinhoComponent,
    CheckoutComponent,
    SlideshowComponent,
    CriarContaComponent,
    FavoritosComponent,
    PaginaInicialComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

### 4.3. Base de Dados



## 5. Conclusão

### 5.1. Lições Aprendidas

Deparamo-nos com um trabalho bastante interessante, desafiador e complicado, uma vez mais a pesquisa por respostas e a entreajuda foram chaves fundamentais para a implementação deste código, achamos sem dúvida que foi um trabalho gratificante e que poderá impulsionar-nos em várias vertentes no mercado de trabalho.

Para a produção deste projeto, tivemos de explorar e aprofundar conhecimentos apreendidos durante o semestre, coisa que na visão do grupo é bastante benéfico e enriquecedor pois ao mesmo tempo que tivemos em constante contacto uns com os outros foi também necessário trazer ao de cima o nosso espírito investigador, uma vez que foi crucial a busca de informação em diversos tipos de plataformas.