

Computação Móvel - Projeto 1

Aplicação Android de um Supermercado Eletrónico

Relatório

Realizado por:

Ana Margarida Silva - up201505505

Julieta Frade - up201506530

14 de Novembro de 2019

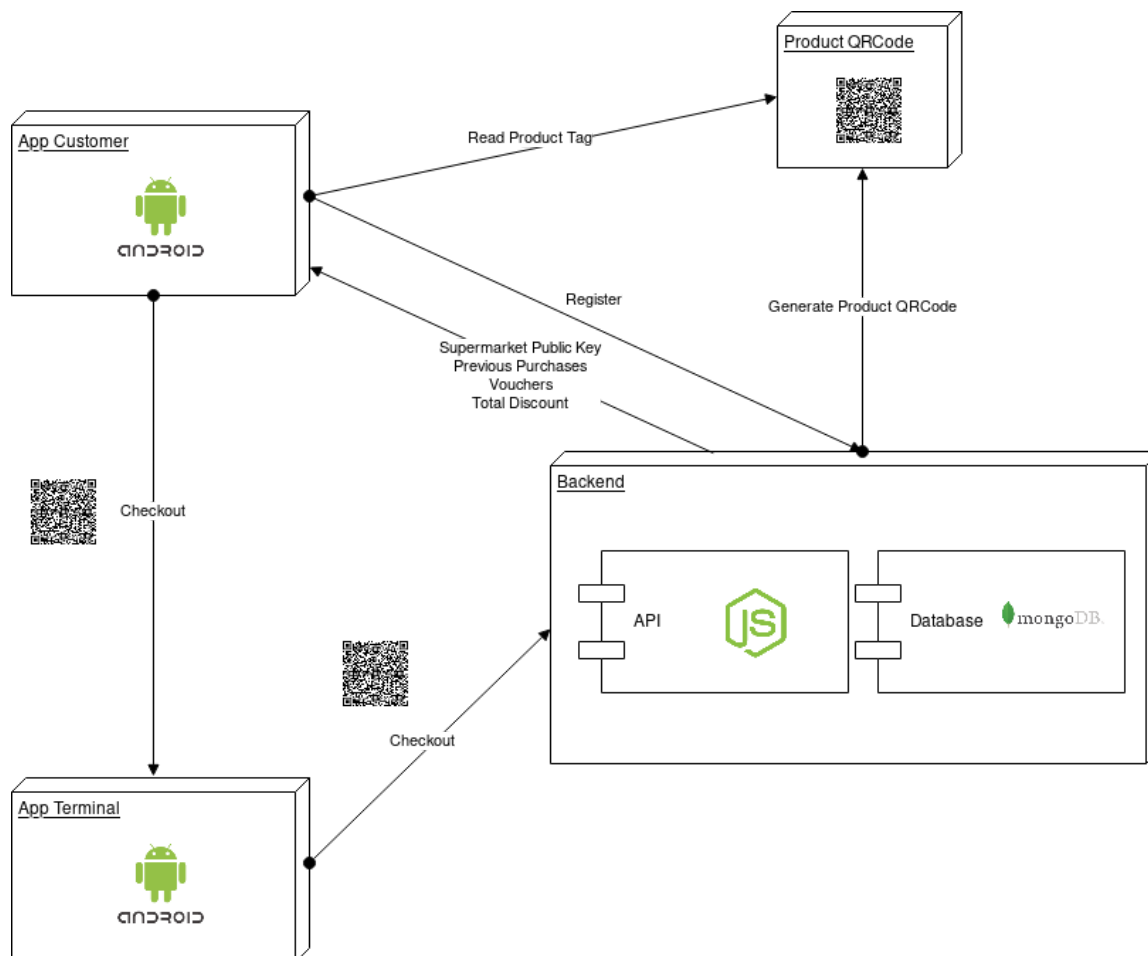
Índice

1. Introdução	2
2. Arquitetura	3
2.1 Servidor	3
2.2 Aplicação do cliente	4
2.3 Aplicação do terminal	4
3. Data Schema	4
3.1 Endpoints da API	4
3.2 Base de Dados	5
3.3 Representação de Informação	6
3.3.1 Informação de Produto	6
3.3.2 Informação de Checkout e Assinatura	6
4. Interface	8
4.1 Sequência de Uso	8
4.2 Manual de Interação e Funcionalidade	9
4.4.1 Aplicação do Utilizador	9
4.4.1.1 Registo	9
4.4.1.2 Login	10
4.4.1.3 Menu Principal	11
4.4.1.4 Vouchers and History	12
4.4.1.5 Basket	13
4.4.1.6 Checkout	14
4.4.2 Aplicação do Terminal	15
4.4.2.1 Menu Principal	15
4.4.2.2 Resultado do Checkout	16
4.3 Funcionalidades Extra	16
5. Testes Realizados	18
6. Correr e Instalar	19
7. Trabalho Futuro	20
8. Bibliografia e Referências	21

1. Introdução

Um supermercado eletrónico chamado Acme Electronic Supermarket decidiu disponibilizar aos seus clientes uma aplicação Android de modo a implementar um sistema mais eficiente para ser utilizado na compra e pagamento dos produtos da loja. À medida que os clientes colocam os itens no seu cesto físico, registam-no na aplicação de forma a aparecer no cesto da aplicação. Quando estão prontos para sair da loja e se a informação do utilizador estiver correta, a aplicação gera um QR Code que será usado pela aplicação do terminal para abrir as portas. De modo a usar a aplicação, os clientes têm primeiro que se registar e terão a possibilidade de ver as compras anteriores, o desconto acumulado e vouchers disponíveis. O cliente ganha vouchers quando faz uma compra num valor superior a 100€.

2. Arquitetura



O nosso sistema é composto de 3 blocos principais: aplicação do cliente, aplicação do terminal e o servidor. Consideramos como um bloco o código QR do produto pois apesar de ser criado pelo servidor, este existe isolado deste num possível ambiente de loja.

2.1 Servidor

É composto por 2 módulos: uma API NodeJS + Express e uma base de dados MongoDB. A API está responsável por receber os pedidos das aplicações de cliente e terminal assim como gerar os códigos QR dos produtos. A base de dados está responsável pela persistência do sistema, guardando informação sobre utilizadores, cartões de crédito, produtos, *vouchers* e compras.

O servidor, quando fica *online*, precisa de gerar ou verificar se se encontra gerada um par de chaves privada/pública para o supermercado. Essa ação é feita no *startup* e as chaves são guardadas num ficheiro *.env*.

Para gerar produtos, o servidor tem *endpoints* que recebem informações sobre o produto - nome, preço - e geram a imagem com o código QR deste assim como adicionam o produto à base de dados. A geração deste código QR é explicada no capítulo seguinte.

2.2 Aplicação do cliente

A aplicação do cliente interage com o servidor fazendo o registo e pedindo dados sobre compras prévias, *vouchers* e o desconto total acumulado.

Quando o utilizador usa aplicação pela primeira vez, é gerado um par de chaves privada/pública para o utilizador. Ao fazer o registo, a chave pública do utilizador é mandada para o servidor, onde será guardada, e a chave privada é guardada na *KeyStore* da aplicação. Como resposta ao registo, o servidor envia a chave pública do supermercado, que é guardada na *SharedPreferences* da aplicação.

Na loja, o utilizador pode adicionar produtos ao cesto de compra interagindo com o código QR do produto, lendo-o com a câmara do seu dispositivo móvel. Informação sobre o produto é extraída adicionada ao cesto de compras depois de ser descriptada com a chave pública do supermercado e decodificada de hexadecimal para *byte array*.

Quando o utilizador está pronto para efetuar a compra, a aplicação gera um código QR com a informação de todos os produtos no cesto do utilizador, assim como o UUID do utilizador, se este pretende usar o desconto acumulado e *voucher* (se pretender usar *voucher*, o UUID deste é enviado).

2.3 Aplicação do terminal

A aplicação do terminal lê o código QR com os produtos do cesto do utilizador gerado pela aplicação deste. Depois de ler, a aplicação envia o seu conteúdo para o servidor e mostra se a compra foi bem sucedida ou não.

3. Data Schema

3.1 Endpoints da API

- POST /auth/signup - registo do utilizador na plataforma ACME.
- POST /supermarket/checkout - pedido ao servidor com os dados do código QR gerado pela aplicação do cliente, é atualizada a informação do cliente (total acumulado, desconto acumulado, *vouchers*) e criada nova entrada de compras efetuadas.
- GET /supermarket/list - retorna todas as compras efetuadas de um utilizador.
- GET /supermarket/vouchers - retorna todos os *vouchers* de um utilizador.
- GET /supermarket/discount - retorna o desconto acumulado de um utilizador.
- GET /qrcode/peach - gera um produto chamado “peach” e o seu código QR. Existem outros endpoints similares para produtos diferentes.
- GET /qrcode/custom?name=<nome>&euros=<preço euros>¢s=<preço cêntimos> - gera um produto com nome e preço dados na *query*.

Todos os pedidos GET efetuados relacionados com o supermercado - /supermarket - tinham como parâmetro o UUID do utilizador assinado com a chave privada deste.

3.2 Base de Dados

A base de dados foi implementada em MongoDB e é composta por 5 tabelas:

- Utilizadores: composto por *_id*, nome, *username*, email, desconto acumulado, total acumulado, chave pública, cartão de pagamento, compras efetuadas, *vouchers*.
- Cartões de pagamento: *_id*, nome do cartão, número do cartão, mês da data de validade, ano da data de validade, cvc, utilizador.
- Vouchers: *_id*, utilizador.
- Produtos: *_id*, nome, preço, caminho para a imagem do código QR.
- Compras Efetuadas: *_id*, produtos, utilizador, preço total, preço pago, *timestamp* de criação.

3.3 Representação de Informação

Existem 3 situações que necessitam explicação no que toca à representação de informação: código QR dos produtos, código QR de um checkout e a representação de uma chave.

3.3.1 Informação de Produto

O código QR de um produto contém o seguinte:

Conteúdo	Descrição
Tag	Código hexadecimal da palavra “acme” - 4 bytes
UUID - bytes mais significativos	8 bytes mais significativos do UUID do produto
UUID - bytes menos significativos	8 bytes menos significativos do UUID do produto
Preço - euros	Número inteiro que representa os euros do preço do produto - 4 bytes
Preço - cêntimos	Número inteiro que representa os cêntimos do preço do produto - 4 bytes
Comprimento do nome	Comprimento do nome do produto - 1 byte
Nome do produto	Nome do produto codificado em ISO-8859-1 - <comprimento do nome> bytes

Esta informação é encriptada com a chave privada do supermercado e codificada em hexadecimal.

3.3.2 Informação de Checkout e Assinatura

O código QR de checkout contém o seguinte:

Conteúdo	Descrição
Número de produtos	Número de produtos no cesto - 1 byte
Lista de produtos no cesto	Para cada produto, é guardado 16 bytes , repartido em 2 conjuntos de 8 bytes - bytes mais significativos e menos significativos do UUID do produto.
UUID do utilizador - bytes mais significativos	8 bytes mais significativos do UUID do utilizador

UUID do utilizador - bytes menos significativos	8 bytes menos significativos do UUID do utilizador
Desconto	0 caso o utilizador não pretenda usar/não tenha desconto, 1 caso pretenda usar - 1 byte
UUID do voucher - bytes mais significativos	8 bytes mais significativos do UUID do voucher a utilizar - 0 se não pretender usar voucher/não tenha voucher
UUID do voucher - bytes menos significativos	8 bytes menos significativos do UUID do voucher a utilizar - 0 se não pretender usar voucher/não tenha voucher

O conteúdo acima é assinado com a chave privado do utilizador e com o algoritmo SHA256WithRSA, é codificado em ISO-8859-1 e transformado em código QR.

4. Interface

4.1 Sequência de Uso



4.2 Manual de Interação e Funcionalidade

4.4.1 Aplicação do Utilizador

4.4.1.1 Registo

The image displays two side-by-side screenshots of a mobile application's registration screen. The left screenshot shows the form with the 'acme' logo at the top. The right screenshot shows the same form with a 'REGISTER' button at the bottom. Both forms include input fields for name, username, e-mail, password, credit card name, credit card number, credit card valid date (with month and year dropdowns), and credit card CVC.

16:32 P 57%

acme

name

username

e-mail

password

credit card name

credit card number

credit card valid date

January 2019

credit card CVC

16:33 P 57%

name

username

e-mail

password

credit card name

credit card number

credit card valid date

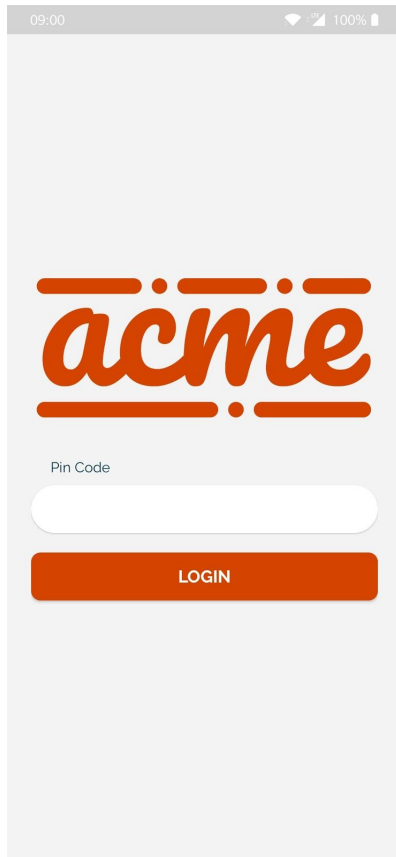
January 2019

credit card CVC

REGISTER

1. **Input Name**
2. **Input Username**
3. **Input E-mail**
4. **Input Password**
5. **Input Credit Card Name**
6. **Input Credit Card Number**
7. **Input Credit Card Valid Date**
8. **Input Credit Card CVC**
9. **Botão Register** - Permite que o utilizador se registe na aplicação

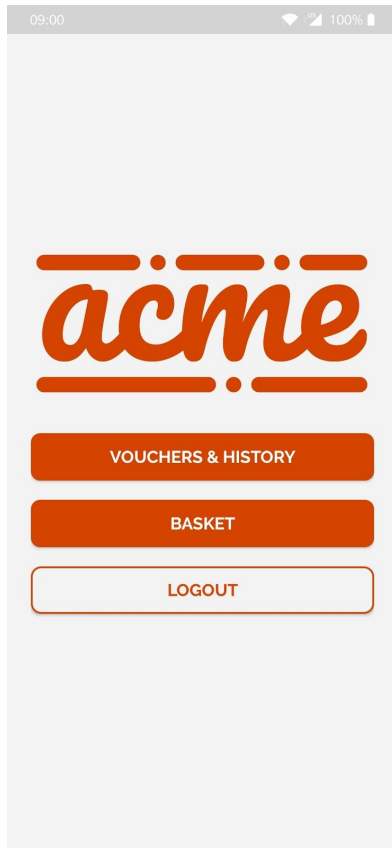
4.4.1.2 Login



The image shows a mobile application interface for a login screen. At the top, there is a status bar with the time '09:00', signal strength, and battery level at '100%'. Below the status bar is a large orange logo for 'acme' in a stylized, lowercase font. Underneath the logo is a label 'Pin Code' followed by a white, rounded rectangular input field. Below the input field is an orange button with the text 'LOGIN' in white, uppercase letters. The background of the screen is a light gray.

1. **Input Pin Code** - Pin do utilizador registado no telemóvel
2. **Botão Login** - Permite fazer o login localmente

4.4.1.3 Menu Principal



1. **Botão Vouchers and History** - Permite aceder à atividade que mostra o número de vouchers, desconto acumulado e histórico de compras
2. **Botão Basket** - Permite aceder ao cesto de produtos do utilizador
3. **Botão Logout** - Permite realizar o logout

4.4.1.4 Vouchers and History

09:00

100%

vouchers & history

Number of Vouchers	Accumulated Discount
4	45.79€

ID: 81a955

Date: 11/12/2019 14:45:54

Products:

Potato - 0.55€
Bonduelle Corn - 2.24€
Expensive Melon - 99.00€

Total Price: 101.79€
Paid Price: 56.00€

ID: gc8db2

Date: 11/12/2019 14:49:31

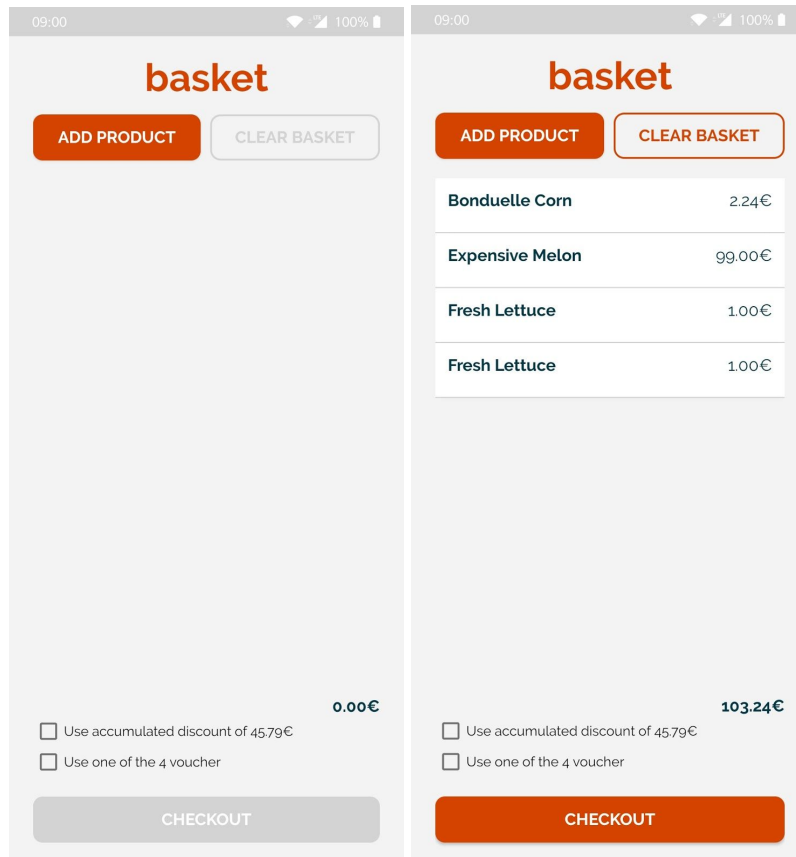
Products:

Manga - 3.00€
Expensive Melon - 99.00€
Expensive Melon - 99.00€
Bonduelle Corn - 2.24€
Manga - 3.00€
Expensive Melon - 99.00€

Total Price: 305.24€
Paid Price: 305.24€

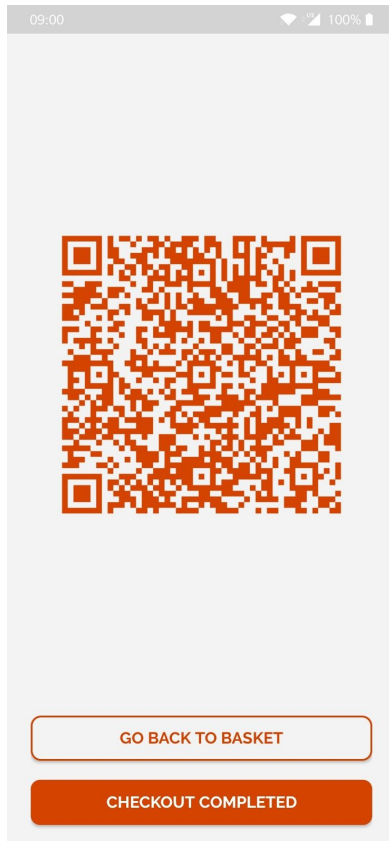
1. **Number of Vouchers** - Número de vouchers do utilizador
2. **Accumulated Discount** - Desconto acumulado do utilizador
3. **Lista de compras** - Lista de compras antigas e respetiva informação realizadas pelo utilizador. Se uma das compras for clicada, os produtos dessa compra são adicionados ao cesto.

4.4.1.5 Basket



1. **Botão Add Product** - Permite aceder à funcionalidade de scan de um QR Code de um produto para o adicionar ao cesto
2. **Botão Clear Product** - Permite apagar todos os produtos do cesto
3. **Lista de Produtos** - Lista de produtos adicionados ao cesto e respetiva informação
4. **Total** - Soma do preço de todos os produtos adicionados ao cesto
5. **Checkbox Accumulated Discount** - Permite utilizar o desconto acumulado
6. **Checkbox Vouchers** - Permite utilizar um voucher
7. **Botão Checkout** - Permite criar um QR Code com toda a informação relevante para a aplicação do terminal realizar o checkout

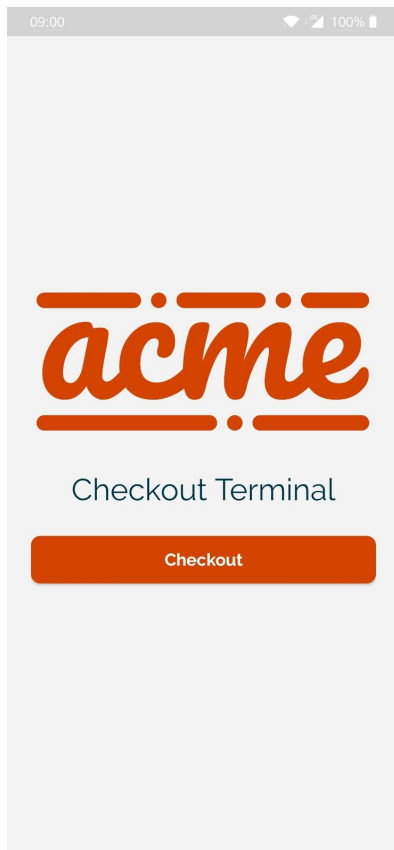
4.4.1.6 Checkout



1. **QR Code** - Tem toda a informação relevante para a aplicação do terminal realizar o checkout
2. **Botão Go Back to Basket** - Permite voltar ao basket sem apagar os produtos que foram comprados
3. **Botão Checkout Completed** - Permite voltar ao Menu Principal, apagando do cesto os produtos comprados

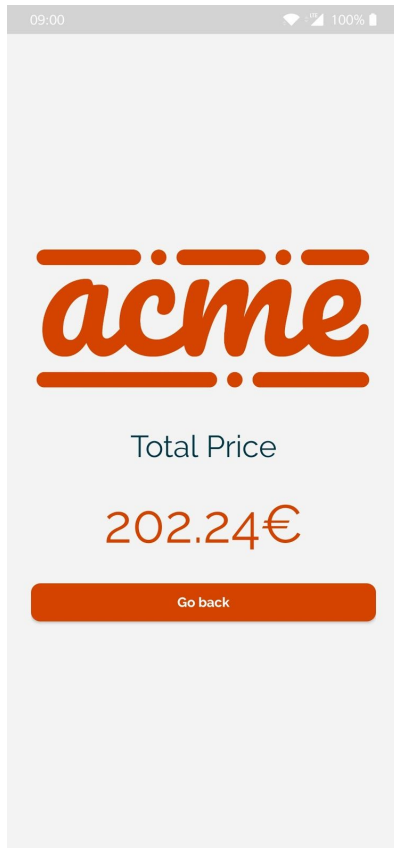
4.4.2 Aplicação do Terminal

4.4.2.1 Menu Principal



1. **Botão Checkout** - Permite aceder à funcionalidade de scan de um QR Code com a informação relevante para dar checkout

4.4.2.2 Resultado do Checkout



1. **Total Price** - Preço total do checkout que se realizou
2. **Botão Go Back** - Permite voltar ao Menu Principal

4.3 Funcionalidades Extra

Na maioria das vezes, quando os clientes regulares se deslocam ao supermercado têm por hábito comprar aproximadamente os mesmos produtos que compraram anteriormente. De modo a melhorar a experiência desses utilizadores na aplicação, foi implementada a **possibilidade de ao clicar numa compra do histórico, a lista de produtos dessa compra ser automaticamente adicionada ao basket.**

Por forma a facilitar os testes à aplicação, foi desenvolvida uma **página web que mostra a informação de todos os produtos disponíveis na loja.** Esta informação consiste no nome, QR Code, ID e preço do produto.

Welcome To ACME Supermarket!

Here are all our products, available to be bought by you!

Please buy a lot!



Tomato

62929597-6862-4136-80aa-1609826e020f

Price: 6.29€



Bonduelle Corn

c55a4165-3395-4a40-9a05-093962760308

Price: 2.29€



Peach

39512682-e591-4646-976c-c6598071e616

Price: 6.29€



Expensive Melon

#119a26-7462-4910-a04c-6321627ef1ea

Price: 99€



Manga

afc10712-1366-466f-b662-e64670660954

Price: 3€

5. Testes Realizados

Foram introduzidos vários erros nas aplicações de modo a perceber se o sistema se portava como era previsto:

1. Não preenchimento de alguns inputs do registo
2. Introdução de uma data do Cartão de Crédito prévia à data corrente
3. Introdução no login de um Pin Code errado
4. Introdução de mais de 10 produtos no cesto
5. Seleção de várias compras antigas de modo a serem adicionados ao cesto mais de 10 produtos
6. Scan de um QRCode com um formato errado
7. Introdução de erros na parte do servidor

A aplicação não fechou inesperadamente em nenhum dos casos e deu um feedback ao utilizador com a origem do erro.

6. Correr e Instalar

1. MongoDB

- a. Instalar
 - i. Windows:
<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>
 - ii. Linux: <https://docs.mongodb.com/manual/administration/install-on-linux/>
 - iii. Arch: Instalar **mongodb-bin** from **_AUR_**
 - iv. Mac: <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>
- b. Correr o comando **mongod** no terminal

2. API

- a. Na pasta do projeto correr o comando **cd api** no terminal
- b. Correr o comando **npm install** no terminal
- c. Correr o comando **npm run dev** ou **npm run start** no terminal

3. NGrok

- a. Instalar: Correr o comando **npm install ngrok -g** no terminal
- b. Correr o comando **ngrok http 8080** no terminal

4. Android Studio

- a. Customer App
 - i. Copiar o endereço do ngrok (ex: <http://319c5d33.ngrok.io>) para a variável **URL** que se encontra no ficheiro:
`app/java/feup.cmov.mobile/common/Utils.java`
 - ii. Compilar e correr.
- b. Terminal App
 - i. Copiar o endereço do ngrok (ex: <http://319c5d33.ngrok.io>) para a parte inicial da variável **URL** que se encontra na linha 78 do ficheiro:
`app/java/feup.cmov.terminal/QRCodeActivity.java`
 - ii. Compilar e correr.

5. Adicionar produtos

- a. Aceder ao endereço do produto desejado:
 - i. <http://localhost:8080/qrcode/custom?name=Salmon&euros=1¢s=80>
 - ii. <http://localhost:8080/qrcode/peach>
 - iii. <http://localhost:8080/qrcode/manga>
 - iv. <http://localhost:8080/qrcode/melon>
 - v. <http://localhost:8080/qrcode/lettuce>
 - vi. <http://localhost:8080/qrcode/potato>
 - vii. <http://localhost:8080/qrcode/tomato>
 - viii. <http://localhost:8080/qrcode/corn>
 - ix. <http://localhost:8080/qrcode/chicken>

Nota: Se houver algum erro relacionado com o mongoose-uuid2, mudar no ficheiro `api/node_modules/mongoose-uuid2/node_modules/bson/lib/bson/objectid.js` a linha **const hostname = require('os').hostname;** para **const hostname = require('os').hostname();**

7. Trabalho Futuro

Foram consideradas algumas funcionalidades mas por falta de tempo não puderam ser implementadas. No futuro podem ser consideradas essas mesmas funcionalidades para melhor o projeto:

1. Possibilidade de apagar um produto específico do Basket, em vez de só se ter a possibilidade de apagar todos os produtos.
2. Possibilidade de escolher a percentagem que se pretende utilizar do desconto, em vez de só se poder utilizar na sua totalidade ou não utilizar.
3. Existência de um pop-up quando o utilizador clica numa compra do histórico para verificar se deseja mesmo adicionar os produtos ao basket porque pode ter clicado por engano.
4. Possibilidade de ordem ascendente ou descendente do histórico por data ou preço.

8. Bibliografia e Referências

1. Miguel Pimenta Monteiro. Mobile Computing 2019/20 Demos. Acedido a Outubro de 2019. <https://paginas.fe.up.pt/~apm/CM/>
2. Google. *Android Developers*. Acedido a Outubro de 2019. <https://developer.android.com/>
3. NodeJS. *Buffer*. Acedido a Outubro de 2019. <https://nodejs.org/api/buffer.html>
4. Dave L. *Convert a string representation of a hex dump to a byte array using Java*. Acedido a Outubro de 2019.