Universidade do Porto

Faculdade de Engenharia

FEUP

The Acme Café Order And Payment System

Relatório

Mobile Computing
2016/2017

5º ano
1º semestre
Mestrado Integrado em Engenharia Informática e Computação

Grupo:

Hugo Drumond - nº 201102900 - hugo.drumond@fe.up.pt
Pedro Moura - nº 201306843 - up201306843@fe.up.pt

Architecture Description

Technological Architecture

Backend

Usou-se o **go** uma vez que é uma ótima linguagem para trabalhar em equipa. A biblioteca standard é robusta e particularmente boa para criar uma API Rest. É simples, garbage collected, tem suporte nativo para concorrência e existem ótimas bibliotecas externas. Para além disto, é bastante escalável e rápida comparativamente ao ruby, node, php, entre outras.

Go libs, biblioteca : uso

- golang.org/x/crypto/scrypt : hashing e salting seguro
- github.com/elithrar/simple-scrypt : hashing e salting fácil de passwords
- github.com/pressly/chi : configurador de rotas bastante eficiente
- github.com/dgrijalva/jwt-go: sessão guardada cliente-side, utilizámos a vertente simétrica, mais em https://jwt.io/. Em suma, servidor cria uma token com 3 partes. A terceira parte é o resultado da encriptação das duas primeiras com uma chave que só o servidor sabe. Quando o servidor receber uma token faz a verificação, assim o servidor não precisa de guardar a token na base de dados.
- github.com/mattn/go-sqlite3 : conexão com o sqlite

Code structure

```
server/
        bin/
                server                  - ficheiro executável do servidor
        pkg/                            - ficheiros compilados
        secrets/
                keypair.pem             - private e public key
                publicKey.pem           - public key
        src/
        vendor/
                src/            - código fonte das bibliotecas externas
                manifest        - indicação das bibliotecas externas
sqlite/
        app.sqlite3             - base de dados (tem de ter este nome, hard coded no go)
        create.sql              - ficheiro de criação da base de dados
        insert.sql              - ficheiro com items, tipos de items, tipos de vouchers, etc
        *.sql                   - ficheiros sql restantes utilizados nos testes
        image/item/:id          - imagens dos items
setup.sh                        - inicializa a base de dados e começa o server
```

Instructions how to compile backend if no binary

1. Instalar go >= 1.7 e sqlite3
2. Criar workspace do go, export GOPATH=”$HOME/golangWorkspace”, mais informação em https://golang.org/doc/code.html .
3. go get github.com/constabulary/gb/... # três pontos são importantes
4. cd backend/server
5. gb vendor restore && gb build server
6. ./setup.sh

O go por default usa o conceito de workspace. Para projetos isso não é o ideal. Por essa razão usámos o gb, “A project based build tool for the Go programming language.”

API com exemplos de uso

GET /cardtype

[{"idCreditCardType":1,"description":"Visa"},{"idCreditCardType":2,"description":"Mastercard"}]

GET /customer/voucher

[{"idVoucher":1,"voucherTemplate":{"idVoucherTemplate":2,"voucherTemplateType":{"idVoucherTemplateType":2,"description":"Free Item"},"items":[41],"description":"Free popcorn","value":1},"code":"xDXydsHFGYJQyygmYDVp.R3LtxHPr2qdZImIPtSaZjE2FGCVE4poQfTBrrANajb5qX8SvTWNVSQrfZ+zsJX1R9y9uszrv9LMafxQVNOIdzMTVtLTHyBFY9X4tRnLK8SqUPuK6t+E6RIyww+SXfREBCeAsclSD9Gqa9luXc4r3tBZx0UY9CfwBwXtPZ60r+03jGaHSbNNKRdQtOX2yy7HjgTsrRvEKvIGteH41YqKDvil56uqvNp5hxNn4Rb/gCmddEStam4wWafUNxpDw6yxiAe9SFurn902cqHfkYbAhAkG1muMTrb3dSe+8E8Lxi+4o/cpznxgpatYs4q6W1TIt7ku8LVN6EI4w8UEcPs07fw==","gotVoucher":"2016-11-13T16:30:47.271390464Z"}]

GET /customer/order

[{"idSale":2,"myDateTime":"2016-11-13T16:30:47.271390464Z","total":34.55,"items":[{"idItem":1,"quantity":1},{"idItem":3,"quantity":10},{"idItem":41,"quantity":40}],"vouchers":[]},{"idSale":2,"myDateTime":"2016-11-13T16:35:07.297819814Z","total":15.05,"items":[{"idItem":1,"quantity":1},{"idItem":3,"quantity":10},{"idItem":41,"quantity":2}],"vouchers":[{"idVoucher":1,"voucherTemplate":{"idVoucherTemplate":2,"voucherTemplateType":{"idVoucherTemplateType":2,"description":"Free Item"},"items":[41],"description":"Free popcorn","value":1},"code":"xDXydsHFGYJQyygmYDVp.R3LtxHPr2qdZImIPtSaZjE2FGCVE4poQfTBrrANajb5qX8SvTWNVSQrfZ+zsJX1R9y9uszrv9LMafxQVNOIdzMTVtLTHyBFY9X4tRnLK8SqUPuK6t+E6RIyww+SXfREBCeAsclSD9Gqa9luXc4r3tBZx0UY9CfwBwXtPZ60r+03jGaHSbNNKRdQtOX2yy7HjgTsrRvEKvIGteH41YqKDvil56uqvNp5hxNn4Rb/gCmddEStam4wWafUNxpDw6yxiAe9SFurn902cqHfkYbAhAkG1muMTrb3dSe+8E8Lxi+4o/cpznxgpatYs4q6W1TIt7ku8LVN6EI4w8UEcPs07fw==","gotVoucher":"2016-11-13T16:30:47.271390464Z"}]}]

POST /customer/register

curl -v -d '{"name": "Hugo", "username": "Hugo", "password": "123456", "creditCard": {"month":12, "year": 2020, "code": "5555555555554444", "idCreditCardType": 2}}' -H "Content-type: application/json" http://127.0.0.1:8080/customer/register

-

{"token":"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJJZEN1c3RvbWVyIjoxLCJOYW1lIjoiSHVnbyIsIlVzZXJuYW1lIjoiSHVnbyIsImV4cCI6MTQ4NDE2MjQ0Nn0.KNGbyLr3NAE5TSK7wxzx92IpsxPStXXS5tXTOrmIutYyZzUNydwwczQ389_IF5ZTGqZFntjtUkOuxP8H1R_3vg","PIN":7479}

POST /customer/login

curl -v -d '{"username": "Hugo", "password": "123456"}' -H "Content-type: application/json" http://127.0.0.1:8080/customer/login

-

{"token":"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJJZEN1c3RvbWVyIjoxLCJOYW1lIjoiSHVnbyIsIlVzZXJuYW1lIjoiSHVnbyIsImV4cCI6MTQ4NDE2MjUyNX0.YdFKYplOrtRqKfR8l35Lkg3sDrJfmIBommc6zrZW_wtzl3UT3L2W5YkyXtQTxkte2NkVqvdbDLkbiqiRbQaVdA","PIN":7479}

GET /item # Parte do output

{"hash":"ZTYWNNDCWTGQYLZ7ACHKRSFGKFAJSTIWMCBPQBJDTHXJBUISQLZA====","items":[{"idItem":1,"itemType":{"idItemType":1,"description":"Beverage"},"price":1.35,"name":"Pepsi","description":"Pepsi","image":"..."},{"idItem":2,"itemType":{"idItemType":1,"description":"Beverage"},"price":1.3,"name":"Coca cola","description":"Coca cola","image":"..."}]}

GET /item/:id (neste caso :id é 1)

{"hash":"F5GW7ESBX6OKO6PZWM4A7G4PUHUIJAT7N2MO5KZC3OOHEGKN6UIQ====","item":{"idItem":1,"itemType":{"idItemType":1,"description":"Beverage"},"price":1.35,"name":"Pepsi","description":"Pepsi","image":"..."}}

POST /terminal/order (retorna blacklist [1,2,3,4,...] se o user ficar blacklist)

curl -v -H 'Content-type: application/json' -H 'Authentication: Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJJZEN1c3RvbWVyIjoxLCJOYW1lIjoiSHVnbyIsIlVzZXJuYW1lIjoiSHVnbyIsImV4cCI6MTQ4MzgwNDgxM30.EM0Qh6O_6gpEMcl9wK20da2aemgq7vTYoUnV16nO_uYXS_ZmR8dboLjdCx8njrJO3_UiT1qDVo7p-1cLOqO5Kg' -d '{"vouchers":[{"idVoucher":13,"code":"KAkwvfEHPoaxpATZtwUR.B2jBLss7Ns3kNFLZYTQk2dF4wir6zOPRzEiNy5vWuVZJ8cZbGOawbQVKz9TiNhaQZZ3NcAd1JhEZFAKgBRUzQDbGb6tODvS6KfeskijxiXrsn+eMj2uNtFZJFN9THq4++WpTb+gO9eml1GDsFE8iHtet53jl4INJobM9IEsUQDCYufpbL4MajM0H1bAyHNoAIbnNCu8h6B3gagKyOIXNqpSTrnVIYc1zcdAv9qmh3EteBMKSX10i/m0CFfRQnL51RCeg4IF27pFhqiA3BkxO03dw2YTqhELUedg7dMxBA0u+Z3ZgjUgGhnB0gloS2MnbntfVnKXTDj9uhORzDaXeMA=="},{"idVoucher":12,"code":"dTGBgJJjlXKFgLzSQWGb.MeuXk6VSx9jPXTM2u8BkdsRs9a+qvvm7ylcNrlWmT0gqXjTVF2YJCyAznqycu9eO0q+SfXC9KB/Mf/Cl+zvRnxURcC96YHI6oIj1EVP6Gdbi4m5SuljPfbmIpHP3yxSaNjMPlhIvkPR0QsxyojA3f3EeQHznopYPhiUgU+7M8/8SB1jBOkDOlPWBrrBBcKS/bYtkbxYifLIhu6oepOO7igfMOziFoZxyzEwC4q5u4+cmyaHLg6Chz20XH5OBK3k0EgczPU/dWgSMbFns4xzDdHAif0X2mld1ZKiuiDKsJU6JVjN0uhQSJmRlh1/3Ipe+8yA5QZ2ahCjID6utoTfReg=="}],"items":[{"idItem":41,"quantity":3},{"idItem":44,"quantity":2},{"iditem":41,"quantity":3}],"idcustomer":1}' http://127.0.0.1:8080/terminal/order

-

{"idSale":11,"total":3.87,"discount":0.73}

GET /terminal/blacklist

[1,2]

Physical Architecture

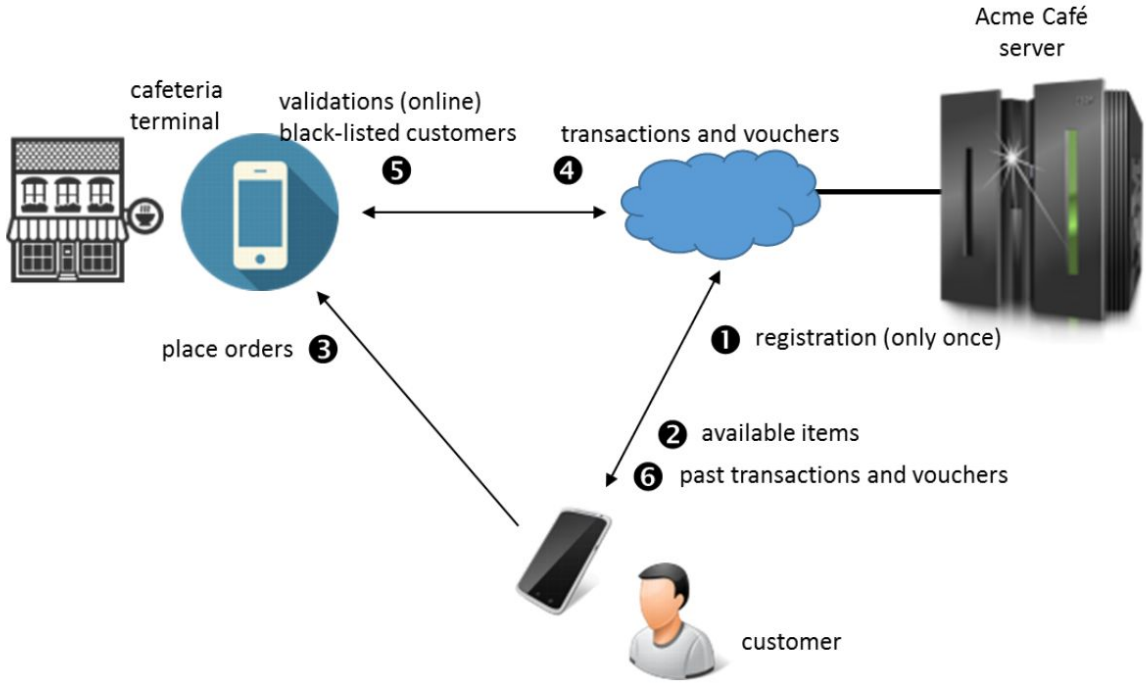


Fig1. Arquitectura sugerida pelo professor

Differences

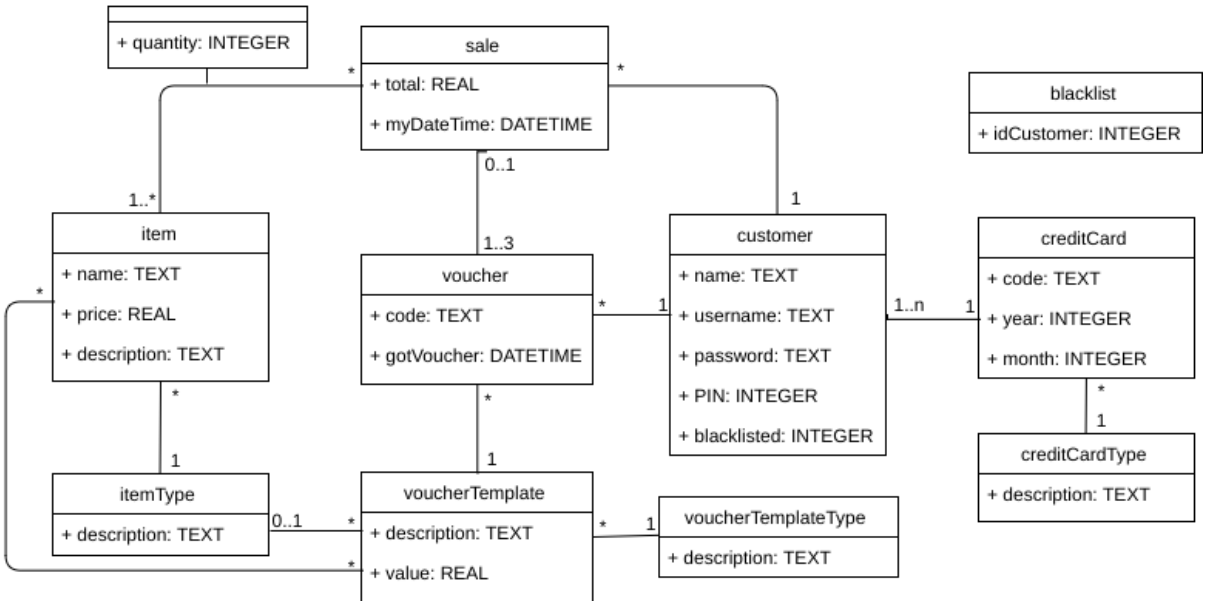
<sessionid> : no nosso caso chama-se token. É composta por <header>.<payload>.<signature> . A token é enviada no header, Authentication: Bearer <token> . O servidor verifica com a sua chave privada verify(header+payload, simetricKey, <signature>) . Caso retorne true a token é válida e o user encontra-se autenticado. O payload contém alguns dados do customer e data de expiração. E o header, informação sobre o algoritmo usado para criar a assinatura.

<voucher code> : é composto por: <code>.<signature> . <code> são [a-zA-Z]{20} aleatórios. <signature> é a assinatura rsa 2048 pss sha1.

Os *posts* recebem sempre json. E respondem com json se não houver algo inesperado, como uma falha do servidor ou informação mal formada enviada pelo utilizador. Nestes casos simplesmente mandamos o status code apropriado juntamente com uma string de erro sem json. Excepto em /terminal/order. Em que, caso o user passe a ficar blacklisted, enviamos o status code 403 com um array de idCustomer(int64).

Data Schema

Backend



Na pasta sqlite do backend encontram-se os ficheiros necessários para a criação e população da base de dados. Respectivamente, ***create.sql*** e ***insert.sql***. Os restantes ficheiros *sql* foram utilizados para testes durante o desenvolvimento seguindo o flow **.sql+curl+resultados*. A coluna blacklisted do customer pode ter os valores 0 se não estiver banido ou 1.

Optou-se por introduzir uma tabela chamada *voucherTemplate* cujo objetivo é reutilizar a metadata dos *vouchers*. Isto é, na nossa implementação a tabela *voucher* é a concretização de um *voucherTemplate* com um *customer* que poderá ou não estar associado a um *sale*. Existem 3 tipos de *voucherTemplateTypes*: “Global Discount”, “Free Item” e “Free Item Type”. A tabela *voucherTemplate* baseia-se num desses tipos e indica respectivamente:

- o valor de desconto (*value*), ou
- quais os items gratuitos(*voucherTemplateItem*) e quantos(*value*), ou
- A categoria dos items gratuitos(voucherTemplate.idItemType) e quantos (value)

As imagens dos *items* não foram colocadas na base de dados. Estas encontram-se em ***sqlite/image/item/:id*** . Temos 44 items na base de dados cada um com uma imagem e 8 tipos de items.

Included Features

Todas as features foram incluídas excepto o modo offline do terminal, caso o servidor esteja incontactável. A blacklist está implementada e os vouchers são assinados com pss sha1 rsa 2048, “code”.”base64(sig)”, só que não é feita a verificação em modo offline. Implementámos tanto o NFC como os QR codes, imagens para os *items*, e um suporte robusto para vouchers de modo a preencher esta lacuna.

Performance Tests

Durante o desenvolvimento da backend utilizámos o curl e alguns ficheiros sql de teste de modo a testar a API REST. A backend foi desenvolvida de forma robusta. Isto é, tentámos ser o mais exaustivos possível no tratamento de erros e na criação de mensagens de erro esclarecedoras. Tomámos particular cuidado ao lidar com os vouchers, uma vez que existem certos casos especiais: haver repetidos; serem inválidos; não servirem para a compra; múltiplos items afetados pelo voucher tipo café, entre outros. Os que não servem para a compra são ignorados, quando existem múltiplos items de um tipo cobertos por um desconto são descontados os mais caros primeiro.

Testámos a vertente multi-utilizador fazendo o registo e troca de utilizadores na aplicação para ver se havia heranças de dados entre eles o que não acontece.
Foram feitas várias compras com variadas seleções para tentar encravar a aplicação, esta não permite avançar na compra se não forem selecionados items para comprar. Na seleção dos vouchers também são feitas algumas validações, por exemplo: não é permitido selecionar mais de 3 vouchers; só se pode selecionar 1 voucher de desconto por compra, etc. É mostrado um aviso sempre que o utilizador tentar efetuar uma ação não permitida. Antes de enviar é também verificado o pin localmente aparecendo uma mensagem quando este estiver errado.
Caso haja algum problema com a ligação ao servidor é sempre apresentado uma mensagem de erro.

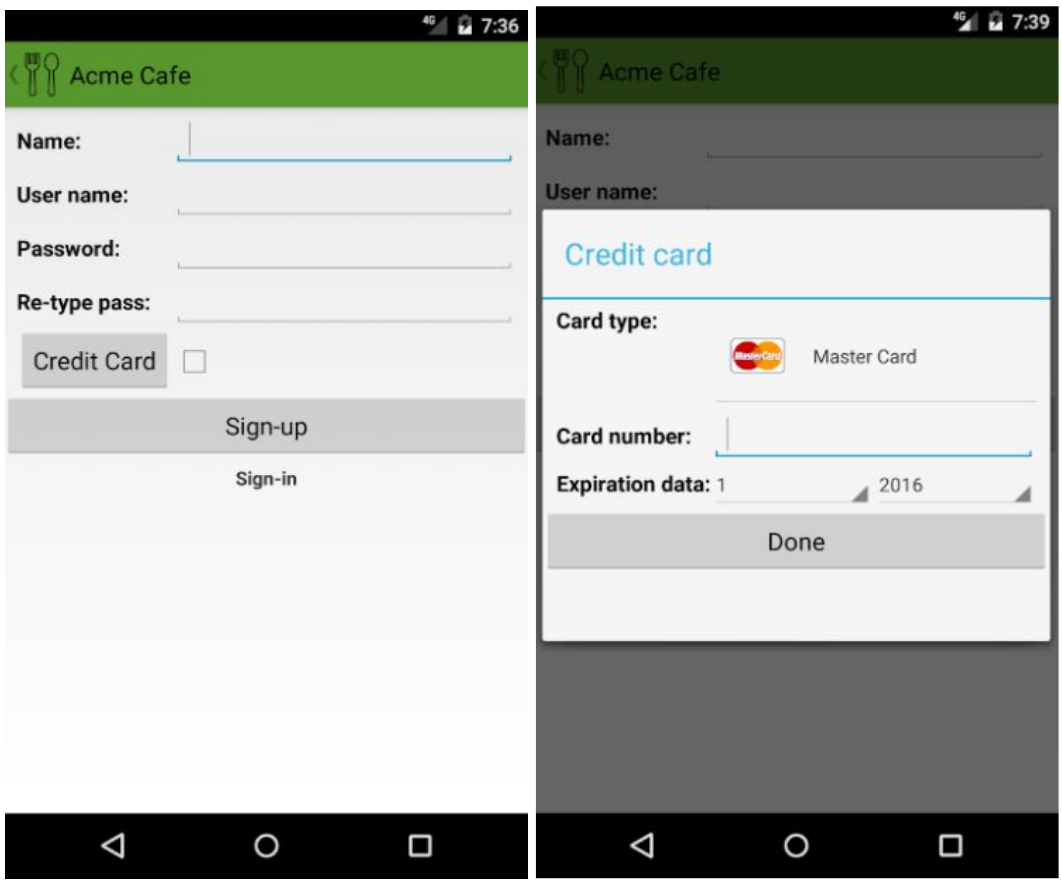
Na 1ª vez que se navega para os menus ou order há um delay que é resultado do download das imagens dos items. Isto é um pouco demorado porque: as imagens são muitas e grandes, e vêm todas na mesma resposta. Para não haver esta demora foi implementado um sistema de hashing, que permite poupar esse tempo.

How to use (Android Instructions)

- **Backend side**

  Tanto na aplicação *Client* como a Terminal é preciso ir a uma classe chamada serverConnectionAPI e mudar a constante `address` para o ip da máquina onde está a correr o servidor da nossa aplicação.

Register/Login



No registo (imagem da esquerda) somos obrigados a escolher o nosso nome, user name (com/ou mais de 3 letras), password(maior ou igual a 6 caracteres), fazer uma confirmação desta escrevendo-a de novo e introduzir um cartão de crédito.

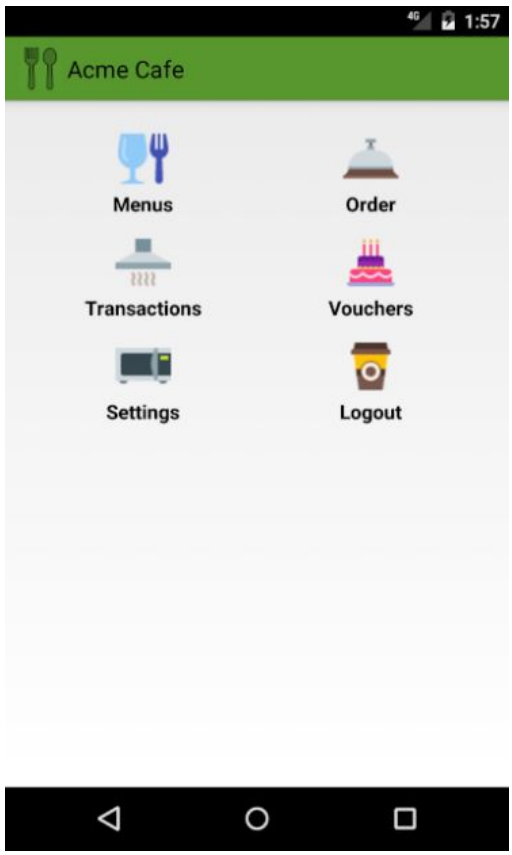
Para introduzir o cartão de crédito o utilizador terá de carregar no botão que diz Credit Card que por sua vez irá abrir uma janela para introduzir os dados deste. Nesta nova janela teremos de escolher o tipo de cartão de crédito, o número deste e a data de validade. Se estes dados forem válidos a checkbox fica ativa.

Se todos os dados introduzidos forem válidos podemos então carregar no botão sign-up para efectuar o registo:

- Em caso de sucesso é nos mostrado um número de 4 dígitos (pin) que teremos de lembrar. Que é usado para confirmar uma compra.
- Em caso de erro irá ser mostrada uma mensagem de ajuda.

Temos ainda um botão de sign-in que permite fazer o login com apenas o username e a password com aspecto similar. Após o primeiro registo o login será feito automaticamente até que um logout seja feito.

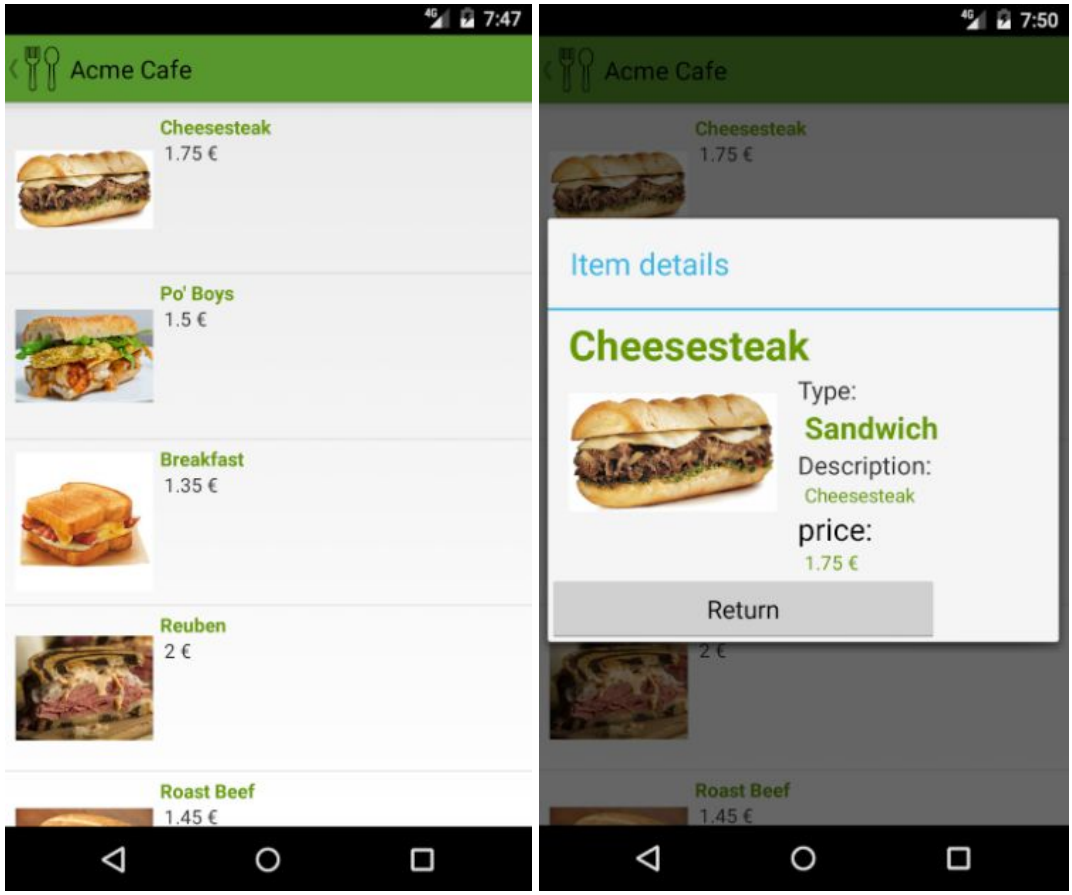
Main menu



Após o registo ou login o menu principal será apresentado. Este menu permite ver e organizar tudo o que a aplicação do acme café permite ver e fazer.

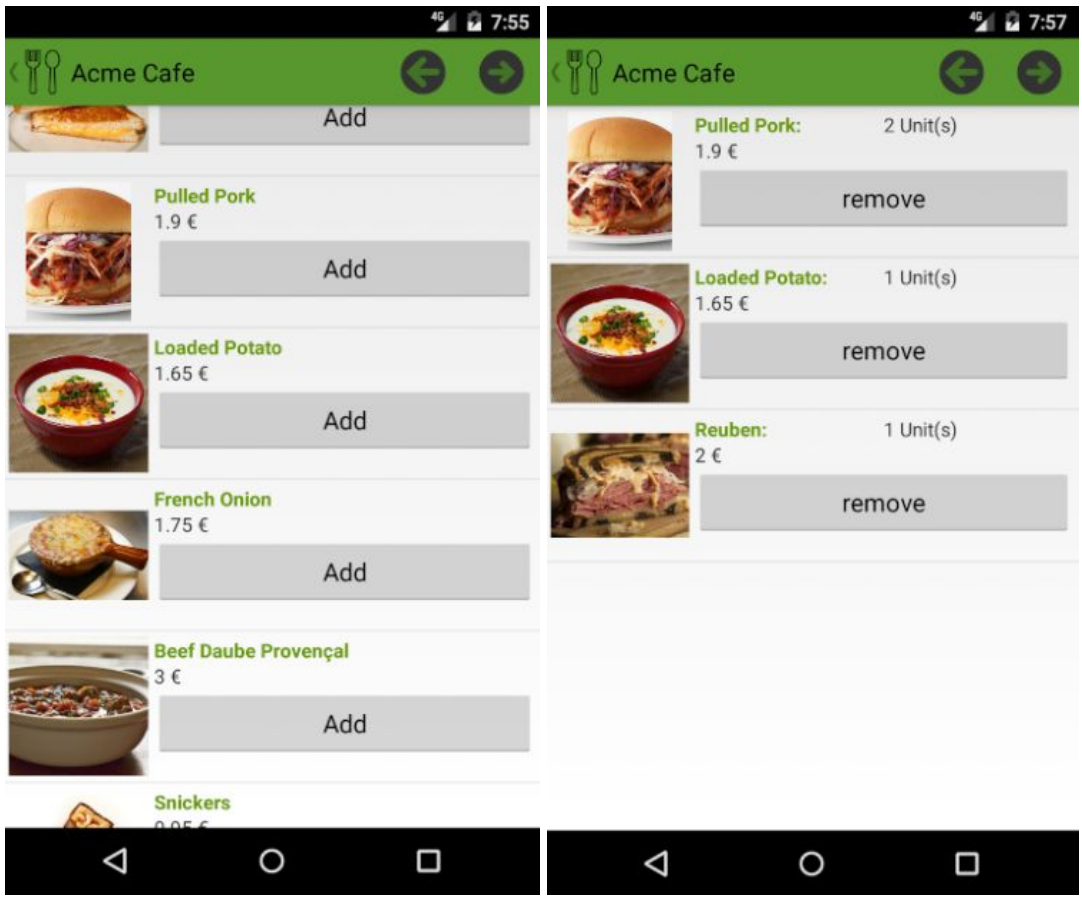
- Opção Menus permite ver uma lista com todos os menus e items do café.
- Opção Order permite fazer uma compra na aplicação.
- Opção transactions para ver todas as transações passadas do utilizador.
- Opção vouchers para ver todos os vouchers do utilizador que ainda não foram usados.
- Opção logout que permite sair da conta do utilizador.
- Opção settings não foi implementada.

Menus



Carregando na opção menus da nossa aplicação podemos ver todos os produtos que o acme café tem para venda (imagem do lado esquerdo). Ao carregar num produto podemos vê-lo mais ao pormenor (imagem do lado direito).

Order

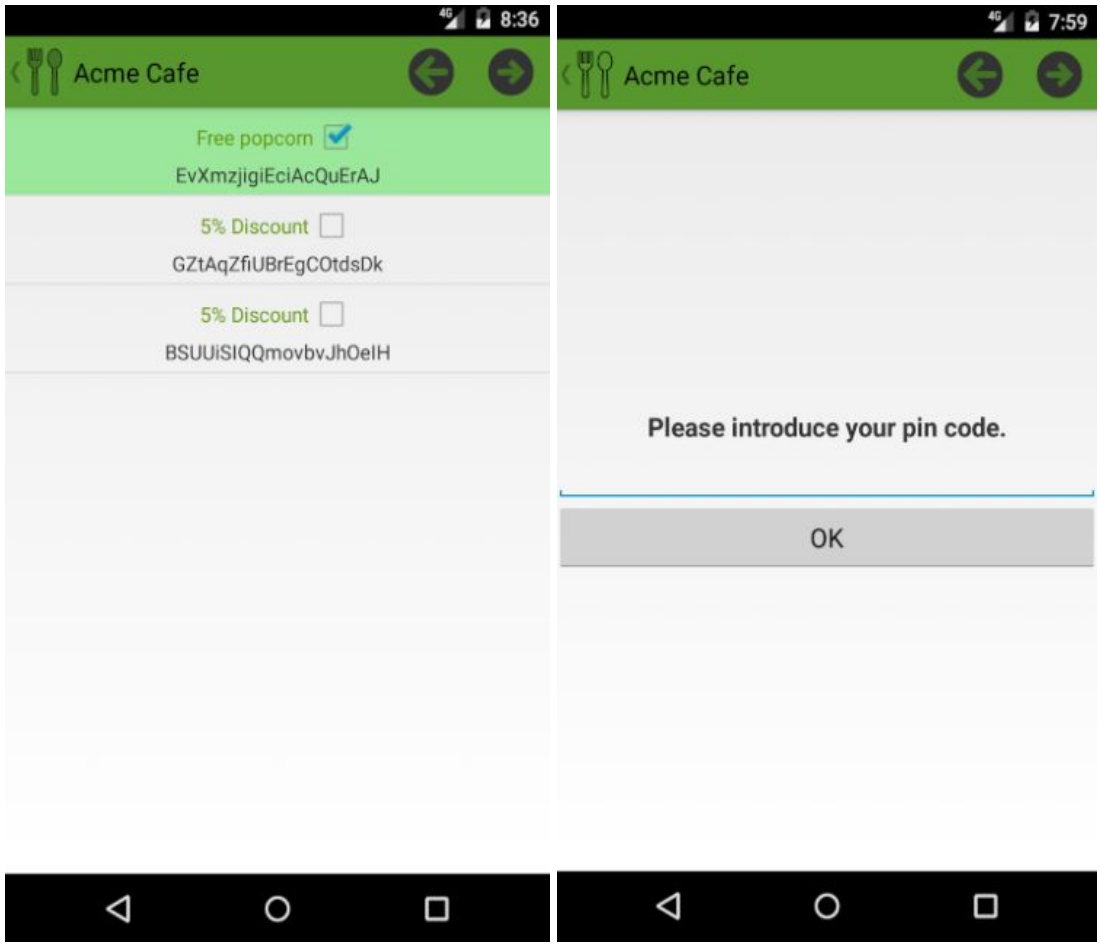


Na opção order é feita uma listagem de itens (imagem da esquerda) muito parecida à dos menus. É acrescentado um botão add que permite adicionar um item ao carrinho de compras da aplicação, este botão poderá ser selecionado várias vezes para adicionar vários itens ao carrinho.

Também são acrescentados dois botões em forma de setas no topo do menu que permitem avançar ou recuar na compra de forma a facilitar a mesma. Selecionando a seta da direita avançamos para o carrinho(imagem da direita) onde podemos ver todos os itens selecionados e se for desejado podemos removê -los carregando no botão remove.

Tal como na opção menus da nossa aplicação podemos ver os detalhes de cada item. Tanto na lista de escolha como no carrinho de compras.

Carregando novamente na setinha para avançar podemos escolher os vouchers que pretendemos utilizar, o que apenas será permitido se existir pelo menos 1 item no carrinho.

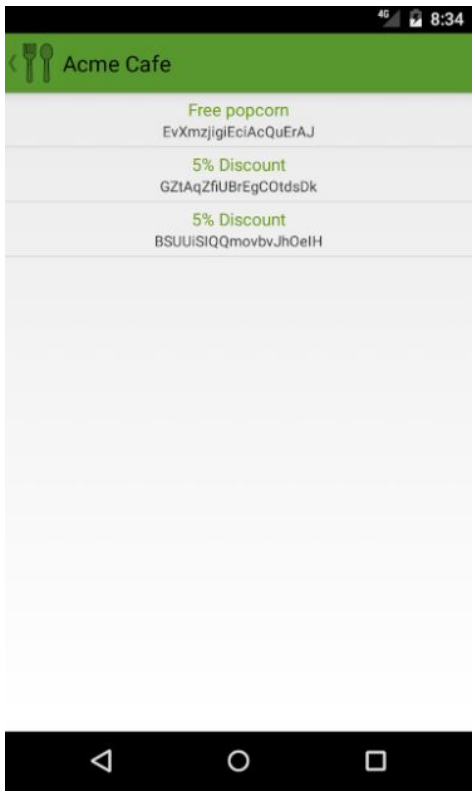


Para selecionar um voucher(figura da esquerda) basta carregar neste, mas só é possível selecionar até 3 vouchers e não será permitido usar mais que 1 voucher de desconto por compra.

Após a seleção dos vouchers pretendidos teremos de introduzir o nosso pin (figura da direita) que nos foi dado após o registo.

Após a introdução do pin uma janelinha com apenas 2 botões irá aparecer para se poder escolher um método de envio nfc ou qr code para que os dados da compra possam ser enviados para o terminal do café pelo método pretendido.

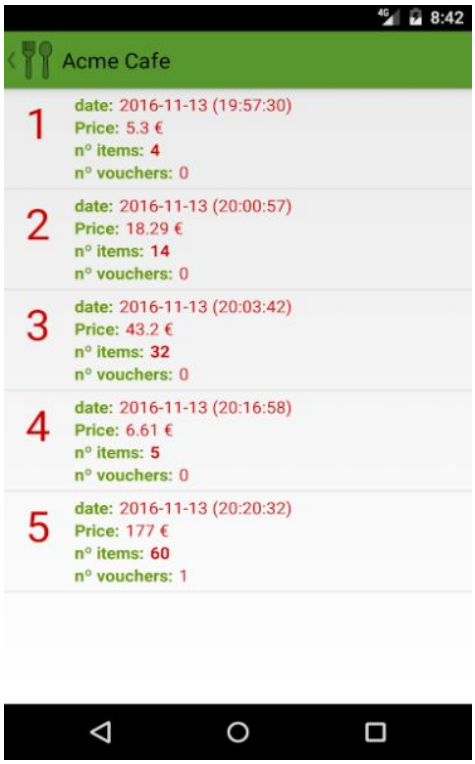
Vouchers



Selecionado a opção voucher do menu iremos ter a uma lista com todos os vouchers que ainda não foram utilizados e que poderão ser usados nas próximas compras.

São mostrados ao utilizador o tipo de desconto a verde como título, e o código sem assinatura.

Transactions



Selecionando a opção Transactions o utilizador pode ver todas as suas transações. Tendo em realce do lado esquerdo, em cada linha, o id da compra. E do lado direito: a data, preço final (já com o desconto), nº de items e nº de vouchers usados.

Na 1º vez que o utilizador selecionar esta opção na seção em que se encontra terá de introduzir o seu username e palavra pass. Após isto ter sido feito não será mais preciso fazê-lo até que aplicação seja encerrada.

- **Terminal side**

Para ler o pedido do cliente

Se o nfc for escolhido ao fazer uma order, teremos de aproximar o nosso dispositivo ao do terminal e premir o ecrã no lado do cliente quando este der sinal, para que os dados sejam enviados.

Para se usar qr codes teremos de premir o botão com a imagem deste tipo para abrir uma câmara que permite a leitura destes.
Após a leitura da compra será mostrado os dados da compra ao utilizador e em caso de erro será apenas mostrada uma mensagem com a descrição deste.