



# **Computação Móvel**

## **Projeto 1**

Android applications and services for a theater ticketing and payment

Nuno Filipe Sousa e Silva - up201404380  
Gustavo Fernando Marques Duarte de Faria - up201304501

19/11/2018

<b>1. Arquitetura</b>	<b>3</b>
1.1. Esquema Base de Dados do Servidor	4
1.2. Esquema Base de Dados Local	5
<b>2. Funcionalidades</b>	<b>6</b>
<b>3. Testes</b>	<b>6</b>
<b>4. Como correr</b>	<b>7</b>
1º Fazer o setup inicial e correr o servidor	7
Setup pela primeira vez	7
Correr o servidor	8
Como usar o “ngrok”	9
2º Mudar o hostname nas aplicações android	9
3º Correr as aplicações android	10
<b>5. Instruções de uso</b>	<b>11</b>
5.1. Company Server	11
5.2. Customer App	17
5.3. Ticket Validation Terminal	21
5.4. Cafeteria Order Terminal	23
<b>6. Conclusão</b>	<b>25</b>

# 1. Arquitetura

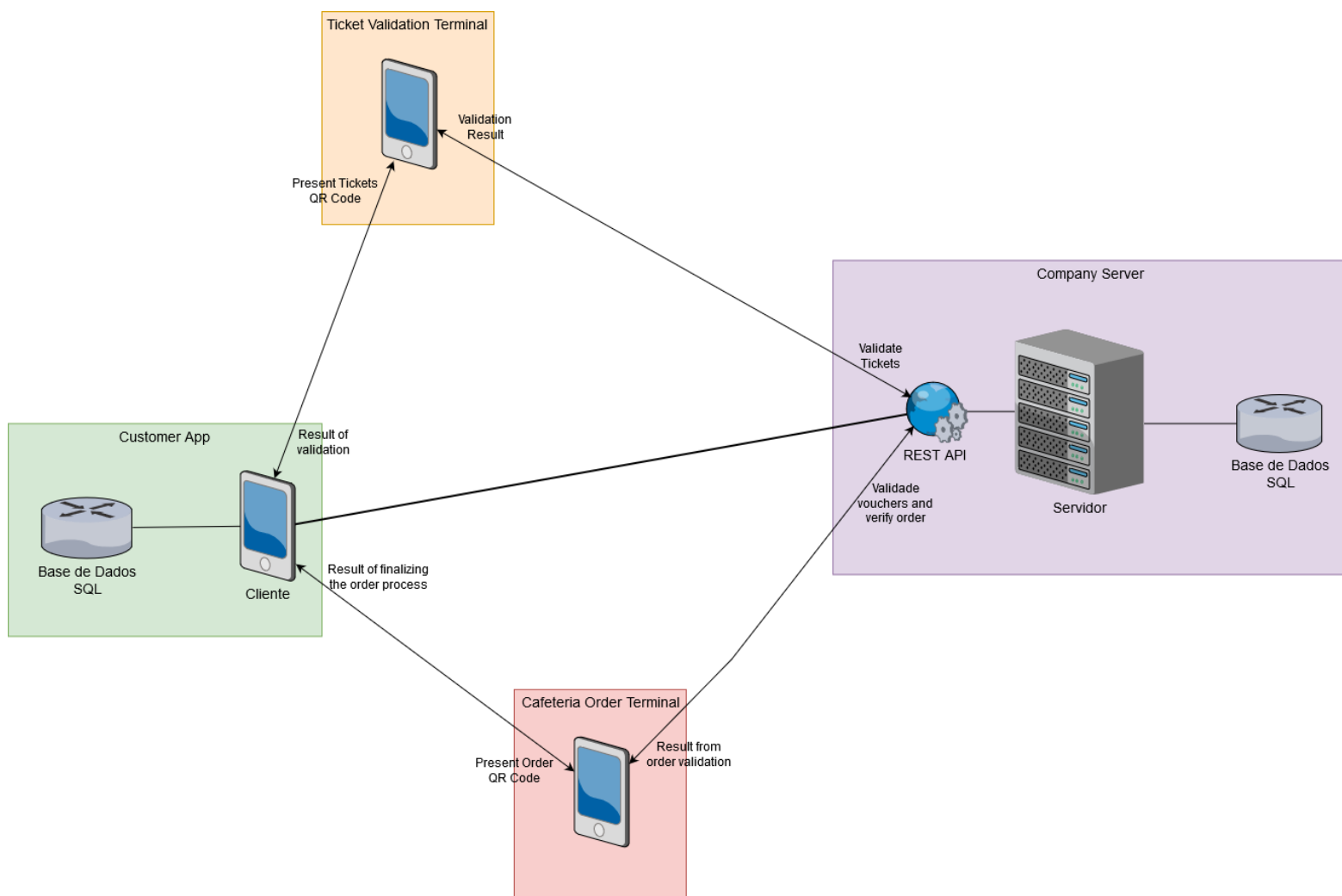
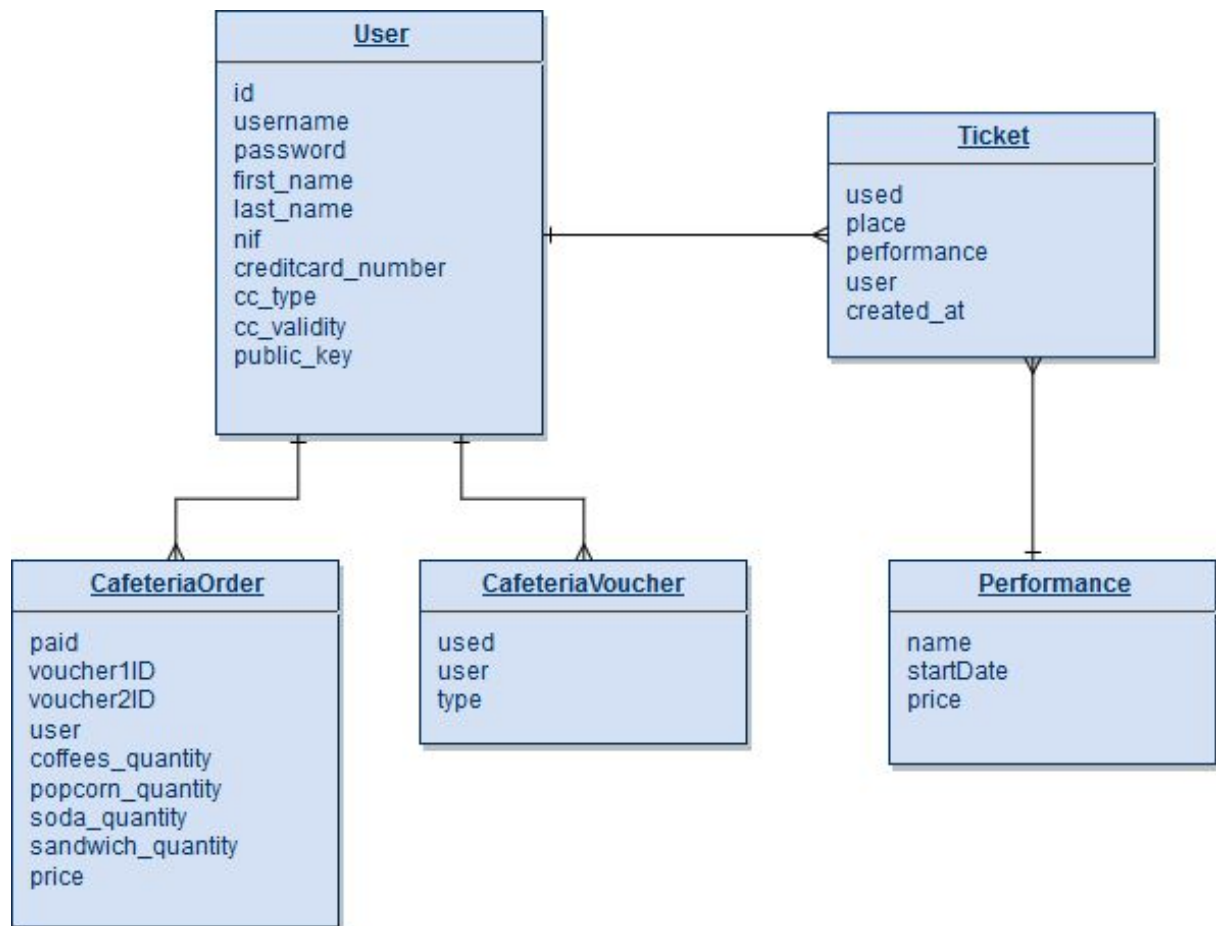


Fig. 1. Arquitetura física

- O servidor foi feito usando Django que é uma framework para a linguagem Python, com uma base de dados SQL. A informação é transmitida em formato JSON usando a REST API (através do browser, o uso de HTML forms também é uma opção). Para lidar com as sessões dos utilizadores é usada autenticação baseada em sessão e em tokens.
- Como não tínhamos acesso a dispositivos com NFC usamos QR Codes para as operações que envolvem os scans do “Cafeteria Order Terminal” e do “Ticket Validation Terminal”.
- O conteúdo das operações de compra de tickets, e do pagamento da ordem da cafeteria, é assinado com um algoritmo de assinatura sha256 com RSA que posteriormente é verificado pelo servidor com a chave pública do utilizador, aquando as respetivas validações. A informação dos QR Codes das ordens de cafeteria geradas também são assinadas da mesma forma, visto que para processar o pagamento, o utilizador tem de criar a ordem e depois quando quiser levantar a sua

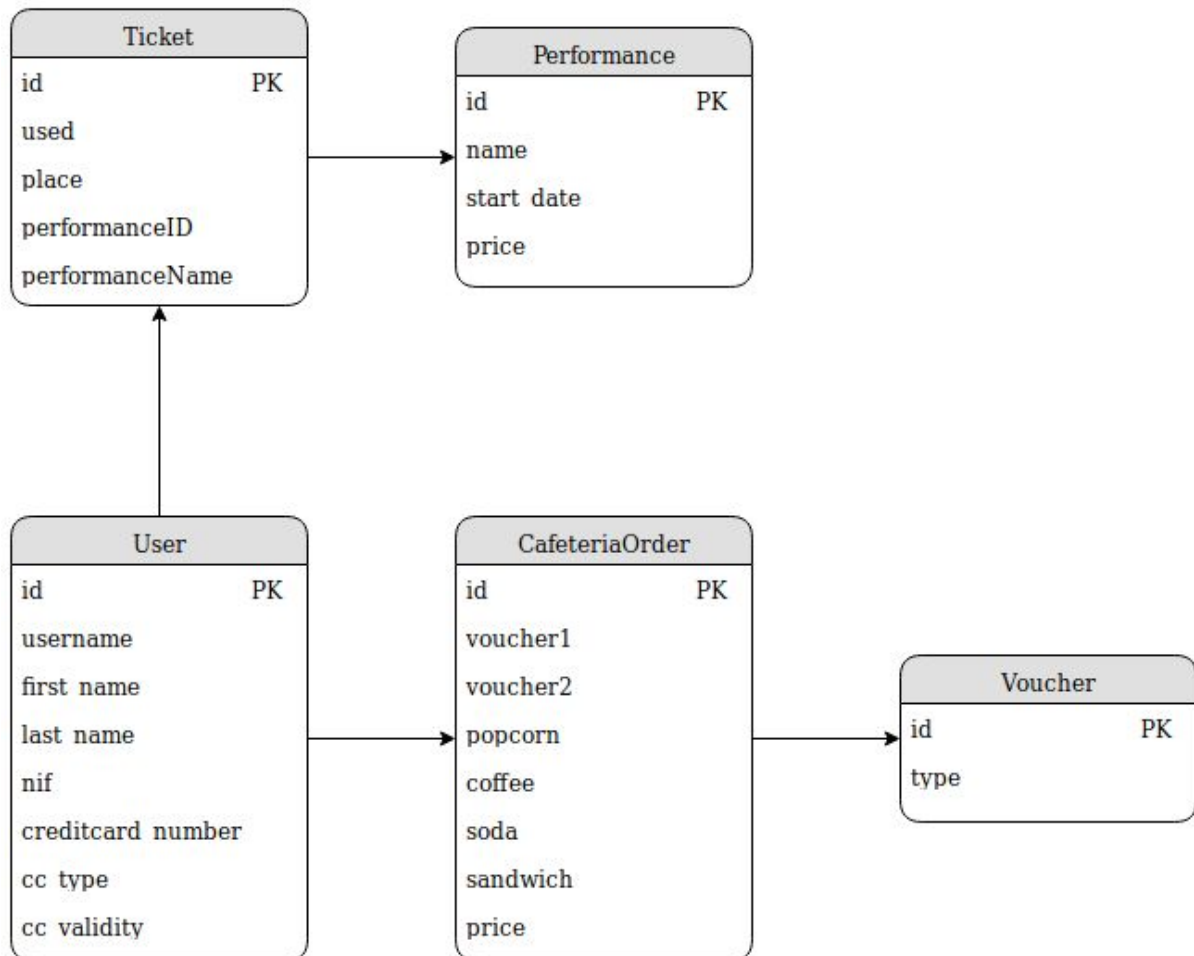
comida, vai à sua lista de ordens da cafeteria, selecionar a ordem e gerar o respetivo QR Code, ir ao “Cafeteria Order Terminal”, fazer o scan do mesmo e só aí é que vai ser feita a validação da ordem de cafeteria e o respetivo processamento do pagamento, logo a informação tem de estar assinada durante todo este processo.

## 1.1. Esquema Base de Dados do Servidor



**Fig. 2.** Esquema da base de dados do “Company Server”

## 1.2. Esquema Base de Dados Local



**Fig. 3.** Esquema da base de dados local da aplicação "Customer App"

Ligações *User - CafeteriaOrder* e *User - Ticket* presentes apenas para ilustrar que objectos estão de facto associados, não havendo, no entanto nenhuma ligação explícita isto porque localmente só é gravada informação de um utilizador de cada vez, o que tornaria essas operações de *JOIN* desnecessárias (i.e. todos os *Tickets* pertencem ao mesmo *User* tal como com todas as *CafeteriaOrders* e *Vouchers*).

## 2. Funcionalidades

- Registo, *login* e *logout*;
- Lista de *performances*;
- *Refresh* de *performances* para manter conteúdo actualizado com servidor;
- Escolher *performance* para comprar bilhete;
- Escolher número de bilhetes a comprar para certa *performance*;
- Lista de bilhetes comprados;
- *Refresh* de bilhetes comprados de forma a eliminar bilhetes usados;
- Selecção de até 4 bilhetes da mesma *performance* para gerar código QR;
- Leitura de bilhete(s) em formato código QR através de terminal validador;
- Validação de bilhetes entre terminal e servidor;
- Criação de encomenda/pedido de cafeteria:
  - Escolha entre pipocas, refrigerante, café e/ou sanduíche;
  - Associação de até 2 *vouchers* de desconto, onde as hipóteses são:
    - Pipocas grátis;
    - Refrigerante grátis;
    - Café grátis;
    - Sanduíche grátis;
    - 5% de desconto (não podem ser escolhidos dois *vouchers* de 5% ao mesmo tempo para o mesmo pedido de cafeteria)
- Lista de encomendas/pedidos de cafeteria activos (por levantar);
- *Refresh* de encomendas/pedidos de cafeteria de forma a eliminar encomendas já satisfeitas;
- Selecção de encomenda/pedido de cafeteria para gerar código QR;
- Leitura de encomenda/pedido em formato código QR através de terminal validador;
- Validação de encomenda entre terminal e servidor;

## 3. Testes

Para os testes das aplicações, foram realizados testes de usabilidade, correspondendo cada teste a cada possível cenário possibilitado por cada uma das funcionalidades descritas acima, sendo eles:

- Registo de utilizadores, procedendo ao *logout* e posteriormente ao *login* do mesmo utilizador;
- *Logout* do utilizador A e posterior *login* do utilizador B de forma a garantir que a informação do primeiro utilizador não está acessível ao segundo;
- Compra de 1 ou mais bilhetes para diferentes *performances*, verificando que passam a constar na lista de bilhetes;
- Selecção e validação de bilhetes, passando o atributo *used* a *true* na base de dados remota e verificando que ao actualizar a lista de bilhetes eles são removidos da base de dados local e verificando também a geração de *vouchers* associados a cada compra;

- Realização de encomenda/pedido de cafeteria com e sem *voucher* associado confirmando que passa a constar na lista de encomendas/pedidos de cafeteria, que a combinação de *vouchers* é permitida e que o preço é o suposto;
- Seleção e validação de encomendas/pedidos de cafeteria de forma semelhante aos bilhetes;
- Testes a mensagens de erro e ações não permitidas como manipulação de dados e adulteração da assinatura do conteúdo de modo a comprovar que as ações são tratadas e rejeitadas da devida maneira;

Para os testes do servidor e da sua base de dados, foram feitos testes manuais de a todas as operações de forma semelhante às operações testadas nas aplicações, assim como testes unitários e automáticos.

Os testes estão divididos em 8 categorias, 5 são conjuntos de testes de operações CRUD para cada uma das tabelas da base de dados SQL, as outras 3 testam a REST API, funções de lógica do servidor e testes que simulam operações com as aplicações android.

Estes testes podem ser corridos executando o comando:

> python manage.py test

```

-----
Ran 8 tests in 1.974s

OK
Destroying test database for alias 'default'...

(venv) C:\Users\Nuno Silva\Desktop\CMOV_apresentacao\Server\CompanyServer\WebServer>

```

**Fig. 4.** Testes do servidor passados com sucesso

## 4. Como correr

### 1º Fazer o setup inicial e correr o servidor

Para correr no seu computador, precisa de ter instalado Python versão 3.7

(<https://www.python.org/downloads/>)

#### - Setup pela primeira vez

**1º** Com a linha de comandos aceder à pasta que contém os ficheiros do servidor:

> cd CompanyServer

**2º** - Criar o virtual environment

> python -m venv venv

**3º** - Ativar o virtual environment

> venv/Scripts/activate

4º - Instalar as dependências do projeto no virtual environment

> pip install -r requirements.txt

5º - Aceder dentro da pasta "WebServer", a primeira, que contém o ficheiro "manage.py" dentro

> cd WebServer

6º - Setup da base de dados

> python manage.py makemigrations WebService

> python manage.py migrate

7º - Criar o super user

> python manage.py createsuperuser

Apenas preencha a informação pedida e o Super User será criado. Este utilizador é o utilizador usado no backoffice do admin para CRUD a base de dados e gerir o servidor.

## - Correr o servidor

1º Com a linha de comandos aceder à pasta que contém os ficheiros do servidor:

> cd CompanyServer

2º - Ativar o virtual environment

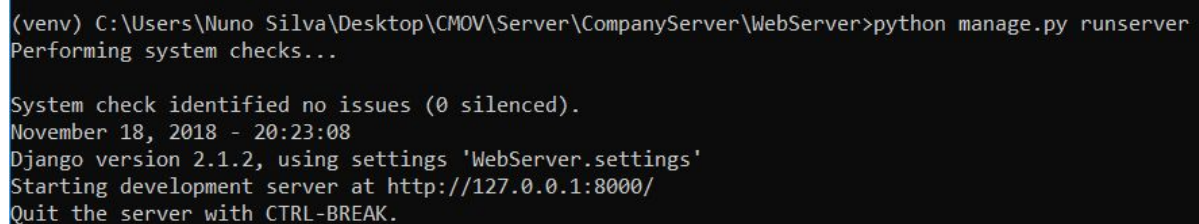
> venv/Scripts/activate

3º -Aceder dentro da pasta "WebServer", a primeira, que contém o ficheiro "manage.py" dentro

> cd WebServer

4º - Correr o servidor

> python manage.py runserver



```
(venv) C:\Users\Nuno Silva\Desktop\CMOV\Server\CompanyServer\WebServer>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
November 18, 2018 - 20:23:08
Django version 2.1.2, using settings 'WebServer.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

**Fig. 5.** Mensagem do servidor a correr corretamente

O servidor irá correr no localhost na porta 8000. Para as aplicações conseguirem aceder pelo exterior, é necessário configurar o servidor para tal. Uma forma simples de o fazer é usar o “ngrok.exe” fornecido.

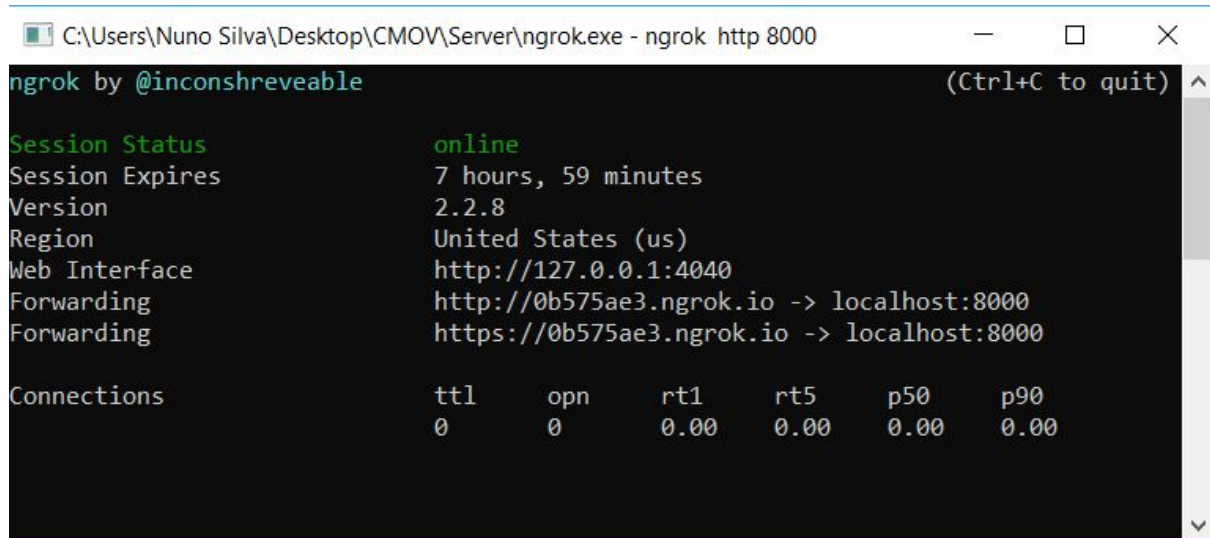


## - Como usar o “ngrok”

1º Abrir o ngrok.exe, que iniciará uma aplicação de linha de comandos

2º Executar nessa linha de comandos

> ngrok http 8000



```
C:\Users\Nuno Silva\Desktop\CMOV\Server\ngrok.exe - ngrok http 8000
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Session Expires     7 hours, 59 minutes
Version             2.2.8
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://0b575ae3.ngrok.io -> localhost:8000
Forwarding           https://0b575ae3.ngrok.io -> localhost:8000

Connections         ttl    opn    rt1    rt5    p50    p90
                   0      0      0.00   0.00   0.00   0.00
```

Fig. 6. Ngrok a fazer a fazer forwarding para o endereço “0b575ae3.ngrok.io”

Irá gerar um endereço que poderá ser usado por aplicações exteriores. Neste caso o endereço é “0b575ae3.ngrok.io”.

## 2º Mudar o hostname nas aplicações android

Como não temos um servidor permanente, temos de mudar o “hostname” nas diferentes aplicações.



```
Cafeteria Order Terminal
cafeteria-order-terminal > app > src > main > java > com > example > nunosilva > cafeteriaorderterminal > Handlers > HttpHandler
HttpHandler.java
1 package com.example.nunosilva.cafeteriaorderterminal.Handlers;
2
3 import ...
13
14 public class HttpHandler
15 {
16     private static final String DOMAIN = "0b575ae3.ngrok.io";
17 }
```

Fig. 7. “cafeteria-order-terminal\app\src\main\java\com\example\nunosilva\cafeteriaorderterminal\Handlers\HttpHandler.java”

### Ticket Validation Terminal



Fig. 8.

"ticket-validation-terminal\app\src\main\java\com\nfss10\cmov\ticketvalidationapp\Handlers\HttpHandler.java"

### Customer App



Fig. 9. "customer-app\app\src\main\java\com\cmov\gustavoenuno\customerapp\Handlers\HttpHandler.java"

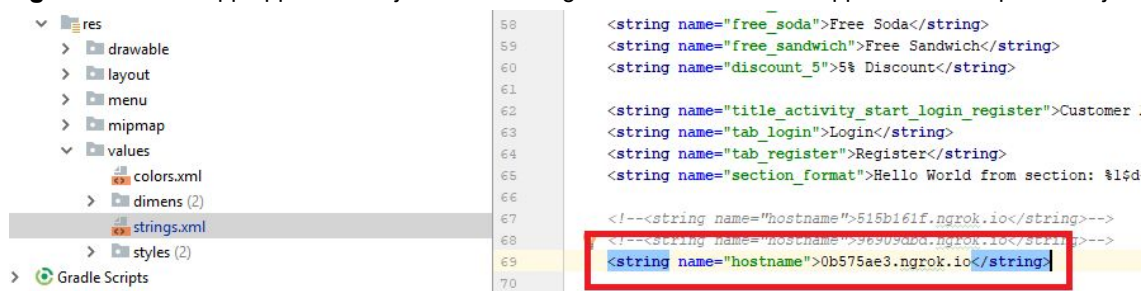


Fig. 10. "customer-app\app\src\main\res\values\strings.xml"

## 3º Correr as aplicações android

Após fazer todos os passos anteriores, através do visual studio compilar e instalar as três aplicações. Para iniciar as aplicações, no seu dispositivo android, só necessita de "tocar" no seu respetivo icon.

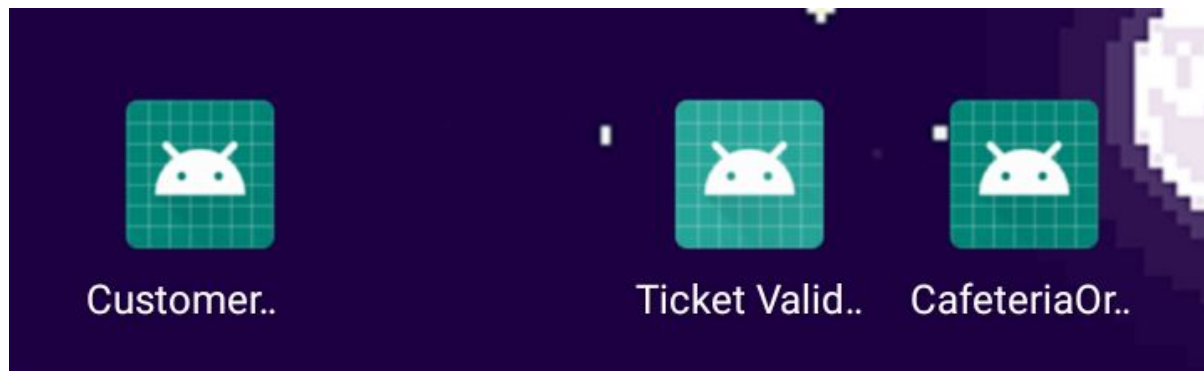


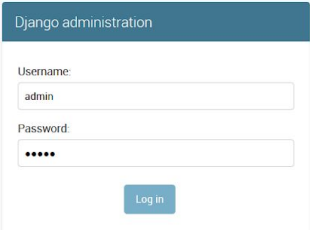
Fig. 11. Icons das aplicações

Ícons das aplicações (da esquerda para a direita): “Customer app”, “Ticket Validation Terminal”, “Cafeteria Order Terminal”

## 5. Instruções de uso

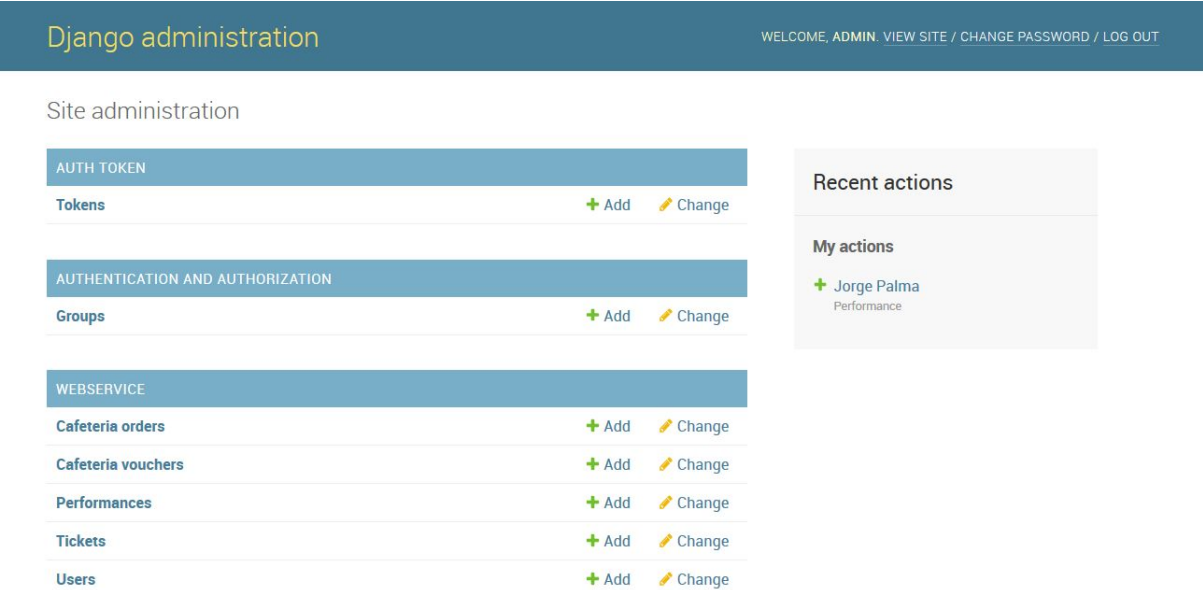
### 5.1. Company Server

Após ter o servidor a correr (ver ponto 4), o administrador pode, através do browser aceder ao backoffice do servidor da companhia acedendo ao endereço “127.0.0.1:8000/admin/” onde será apresentada a página de login do backoffice.



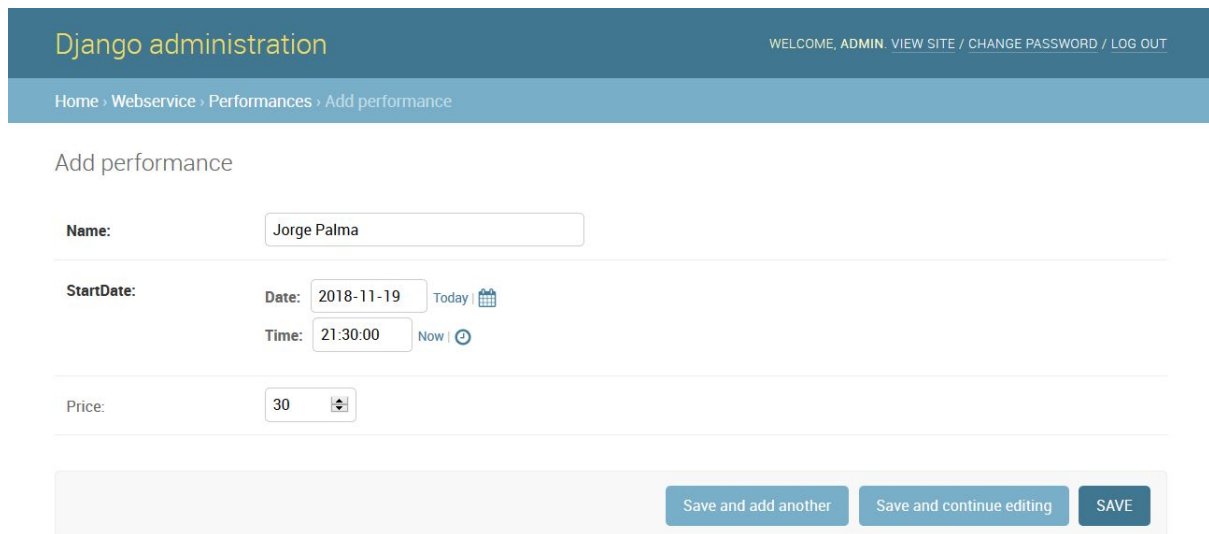
**Fig. 12.** Página Login do backoffice administrativo

Após logar-se com as credenciais do Super User (ver ponto 4), o administrador será levado à página administrativa onde pode facilmente fazer operações CRUD e outras mais, como ver a atividade recente do servidor e verificar outras mais informações.



**Fig. 13.** Página principal do backoffice

A única forma de criar “Performances” é através deste backoffice. O administrador pode criá-las carregando em “+ Add” à frente de “Performances” (fig. 13).



Django administration WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Webservice › Performances › Add performance

Add performance

Name: Jorge Palma

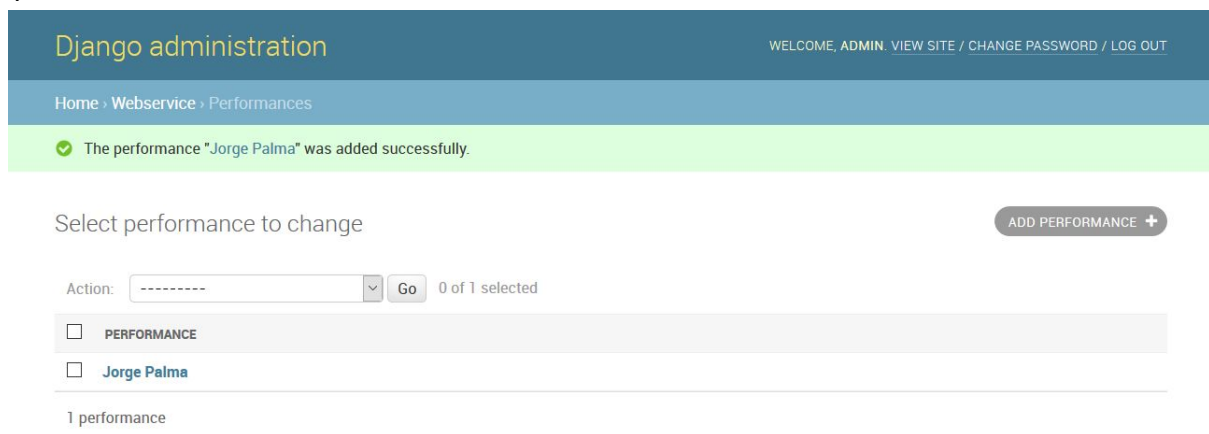
StartDate: Date: 2018-11-19 Today Time: 21:30:00 Now

Price: 30

Save and add another Save and continue editing SAVE

**Fig. 14.** Página de criação de “Performance” do backoffice

Irá ser apresentada página (fig. 14) onde o administrador introduz a informação da “performance”.



Django administration WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Webservice › Performances

✓ The performance "Jorge Palma" was added successfully.

Select performance to change ADD PERFORMANCE +

Action: ----- Go 0 of 1 selected

<input type="checkbox"/>	PERFORMANCE
<input type="checkbox"/>	Jorge Palma

1 performance

**Fig. 15.** Página de lista de “Performance”

Após carregar no botão “SAVE” (canto inferior direito da fig. 14), irá ser levado para a página do backoffice da lista das “Performances” onde será mostrada uma mensagem de sucesso da operação (fig. 15).

## REST API

O servidor também dispões de vários endpoints na qual são usados pelas aplicações móveis. Alguns endpoints não precisam que o utilizador esteja logado para serem usados,

como por exemplo “/api/login/”, outras como “/api/performances/” precisam e mostram o seguinte erro, caso o utilizador não tenha sessão iniciada:



**Fig. 16.** Endpoint /api/performances/” sem ter sessão iniciada

O utilizador pode fazer o login pelo botão “Log in” no canto superior do browser.

Para além do backoffice, o administrador pode usar a REST API para gerir o servidor enviando informação em formato json, ou usando o formulário.

Exemplo de um endpoint útil para o administrador:

Django REST framework

admin with id 8bb05004-02e1-4577-bff6-9c19e5b64209

Api Root / Users List

Users List

OPTIONSGET

GET /api/users/

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": "8bb05004-02e1-4577-bff6-9c19e5b64209",
      "username": "admin",
      "password": "pbkdf2_sha256$120000$h1qnTBbUwZwg$CeXCp0ckQTVz08ijhCyZLzW42/TBQQ1FzBE6Q3LM2y8=",
      "first_name": "",
      "last_name": "",
      "nif": 0,
      "creditcard_number": "0000000000000000",
      "cc_type": 2,
      "cc_validity": "00/00",
      "public_key": ""
    }
  ]
}
```

Raw dataHTML form

Username

Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password

First name

Last name

Nif

Creditcard number

Cc type

Mastercard

Cc validity

Public key

POST

Fig. 17. Endpoint /api/users/

Ações como comprar tickets ou ver os tickets feitas enquanto logado, através da API, vão ficar vinculadas a essa mesma conta.

Nos pedidos GET, pode-se especificar o id, para ter o objecto específico, ou não, caso queiramos todos os objetos.

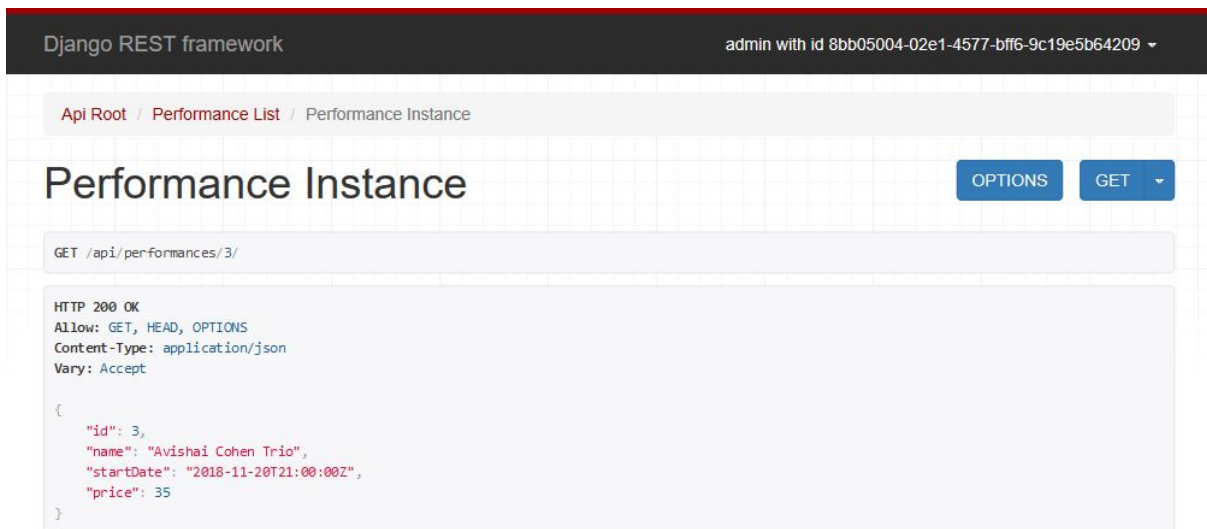
Exemplo para listagem de performances:

The screenshot displays the Django REST framework API interface. At the top, a dark header bar shows 'Django REST framework' on the left and 'admin with id 8bb05004-02e1-4577-bff6-9c19e5b64209' on the right. Below this, a breadcrumb trail reads 'Api Root / Performance List'. The main heading is 'Performance List', with 'OPTIONS' and 'GET' buttons to its right. A sub-header indicates the request: 'GET /api/performances/'. The response area shows the following details:

HTTP 200 OK  
Allow: GET, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "name": "Jorge Palma",
      "startDate": "2018-11-19T21:30:00Z",
      "price": 30
    },
    {
      "id": 2,
      "name": "José Soares Quarteto",
      "startDate": "2018-11-21T19:30:00Z",
      "price": 10
    },
    {
      "id": 3,
      "name": "Avishai Cohen Trio",
      "startDate": "2018-11-20T21:00:00Z",
      "price": 35
    },
    {
      "id": 4,
      "name": "Danças Ocultas",
      "startDate": "2018-11-21T21:00:00Z",
      "price": 22
    }
  ]
}
```

**Fig. 18.** Endpoint `/api/performances/`, request GET



**Fig. 19.** Endpoint `/api/performances/3/`, request GET

Para ver quais os tipos de request aceitáveis e como usar os endpoints, aceder aos mesmos enquanto logado como “Super User”.

## Listagem de todos os endpoints:

- `/api/`
- `/api/users/`
- `/api/performances/`
- `/api/tickets/`
- `/api/unusedTickets/`
- `/api/cafeteriaVouchers/`
- `/api/unusedCafeteriaVouchers/`
- `/api/cafeteriaOrders/`
- `/api/unpaidCafeteriaOrders`
- `/api/login/`
- `/api/buyTickets`
- `/api/validateTickets`
- `/api/payCafeteriaOrder`



## 5.2. Customer App

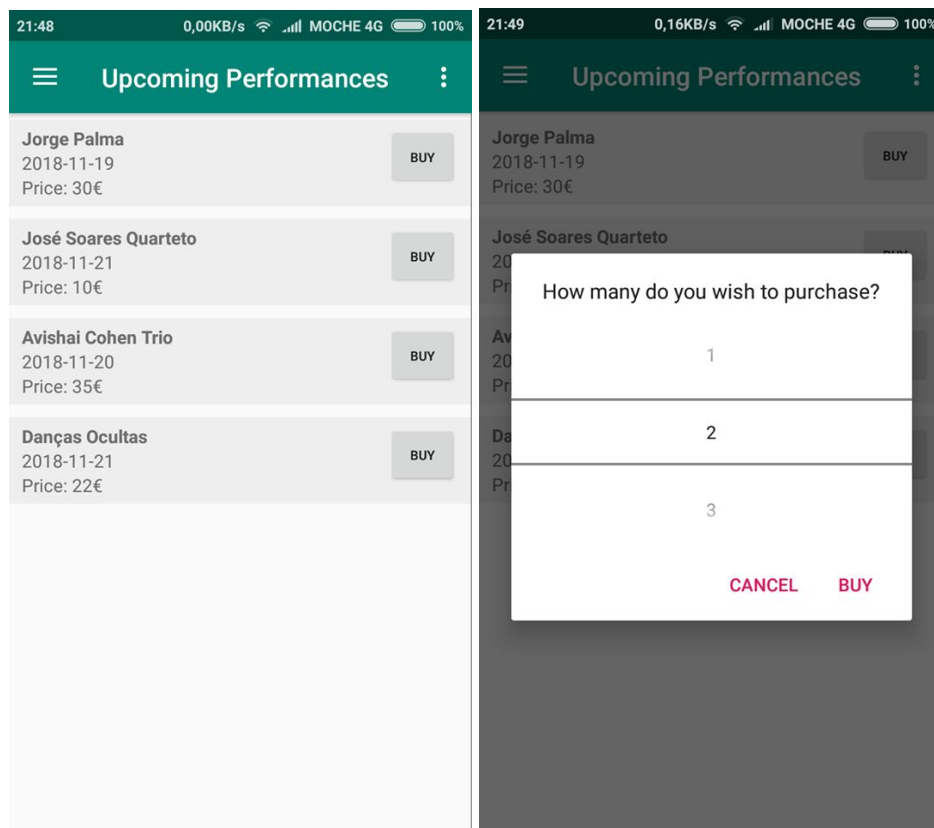
Ao iniciar a aplicação pela primeira vez o utilizador será apresentado com uma view com duas tabs: uma de para se logar e outra para se registar.

(As sessões ficam guardadas até o utilizador fazer o logout, mesmo que o utilizador feche a aplicação, logo caso já tenha feito login ele avançará esta etapa).

The figure consists of two side-by-side screenshots of a mobile application interface. The left screenshot, labeled Fig. 20, shows the 'Customer App' login screen. It has a teal header with 'Customer App' and two tabs: 'LOGIN' and 'REGISTER'. Below the tabs, there are two input fields: 'Username' and 'Password'. A 'SIGN IN' button is at the bottom. The right screenshot, labeled Fig. 21, shows the 'REGISTER' screen. It has a teal header with 'LOGIN' and 'REGISTER' tabs. Below the tabs, there are several input fields: 'First Name' (Nuno), 'Last Name' (Silva), 'NIF' (263412469), 'Credit Card Type' (Master Card), 'Credit Card Number' (5176018009662718), 'Credit Card Expiration Date' (10/22), 'Username' (NFSS10), and 'Password' (masked with dots). A 'SIGN UP' button is at the bottom.

**Fig. 20 e 21.** View de Login e view de Registo respetivamente

Depois de fazer o login (fig. 20), ou de se registar (fig. 21), o utilizador será levado à view das “Upcoming Performances”.

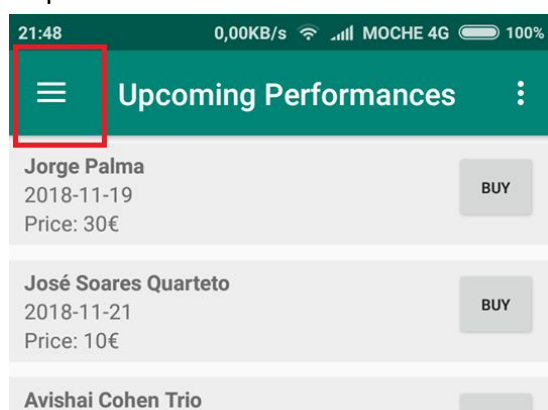


**Fig. 22 e 23.** Lista das próximas atuações e pop-up de compra de bilhetes respetivamente

Aqui (fig. 22) o utilizador pode visualizar os próximos espetáculos e comprar bilhetes para os mesmos.

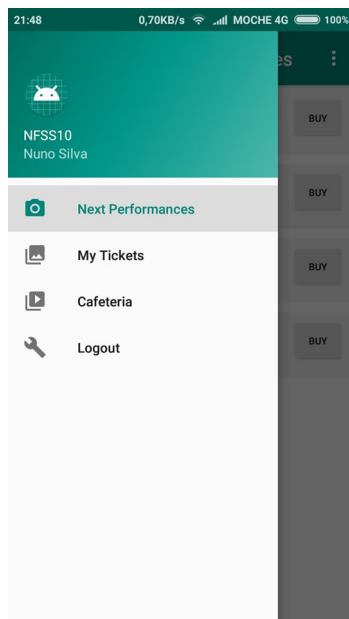
Para isso o utilizador carrega no botão “BUY” que apresentará a janela pop-up (fig. 23) que o deixa escolher a quantidade de bilhetes a comprar.

Para o utilizador aceder às diferentes views, ele tem de carregar no botão do canto superior esquerdo



**Fig. 24.** Botão do menu principal

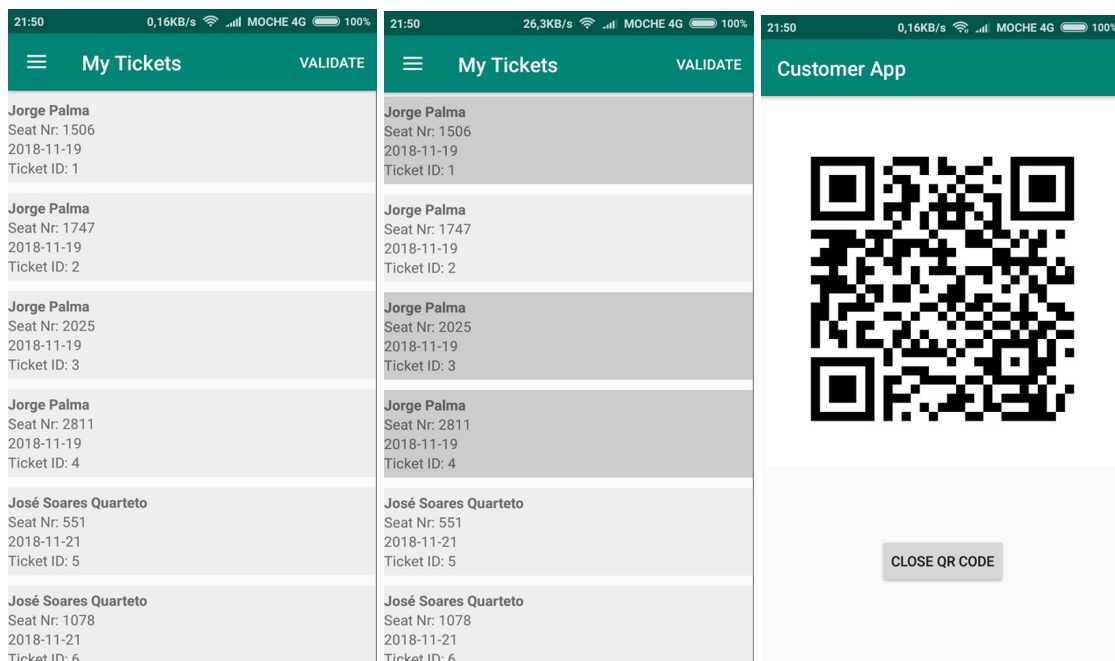
O que apresentará o menu principal (fig. 25).



**Fig. 25.** Menu principal

Neste menu podemos aceder à view “Upcoming Performances” já descrita, “My Tickets”, “Cafeteria” e fazer o “Logout”, assim como visualizar alguma informação sobre o utilizador, como o username e o seu nome.

## “My tickets”

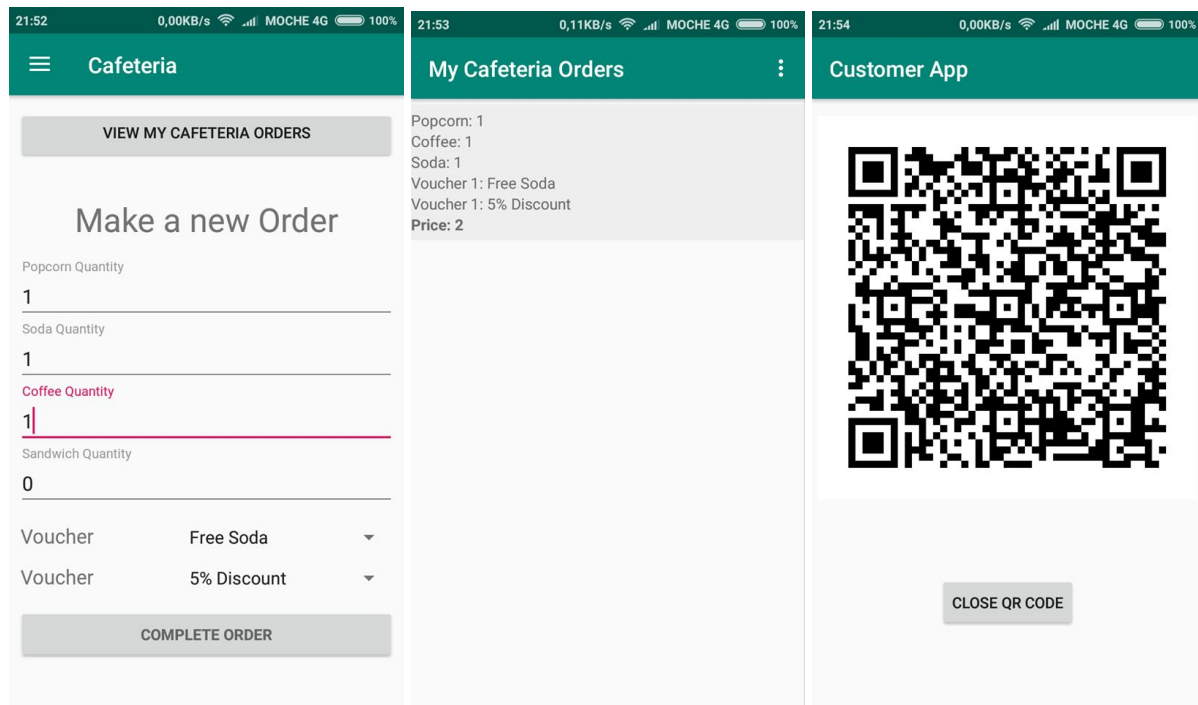


**Fig. 26, 27 e 28.** Lista dos meus tickets, tickets seleccionados, QR Code gerado

Nesta view, o utilizador pode ver os teus tickets que ainda não foram usados(fig. 26).

Também pode seleccionar os tickets que quer validar (carregando em cima e no máximo 4 de cada vez) (fig. 27) e gerar o QR para apresentar no terminal de validação (fig. 28) carregando no canto superior direito em “VALIDATE”.

## “Cafeteria”



**Fig. 29, 30 e 31.** View da “Cafeteria”, Lista de pedidos de cafeteria e QR Code gerado

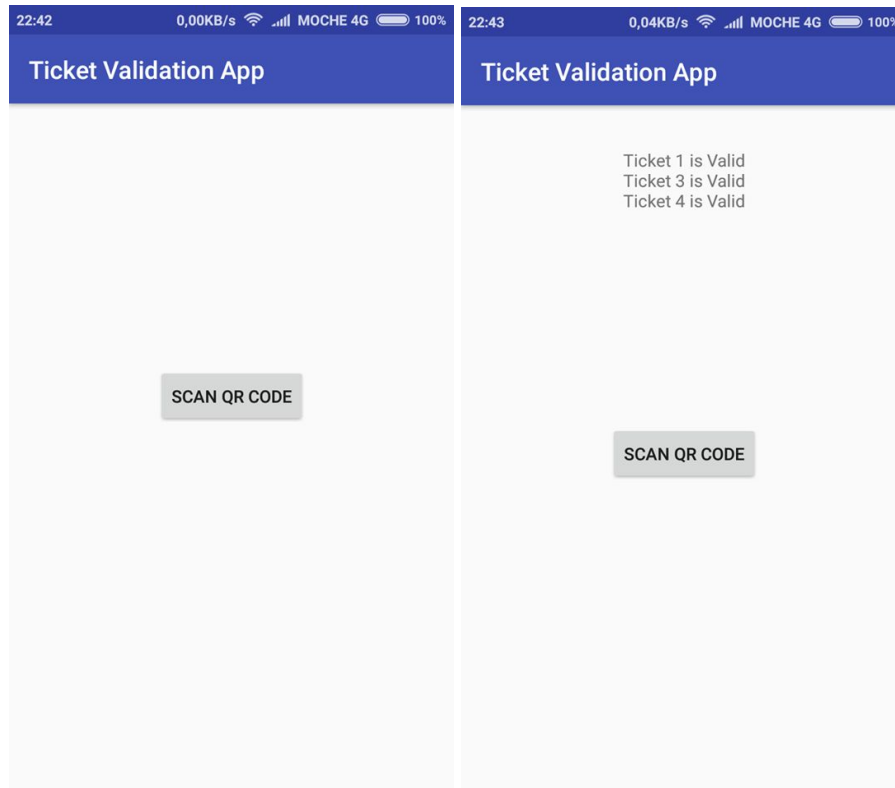
Ao abrir esta view, o utilizador pode fazer um pedido para a cafeteria, especificando a quantidade dos produtos, e os vouchers que quer ou não usar e carregar no botão “COMPLETE ORDER”. Além disso pode ver os seus pedidos já criados carregando no botão “VIEW MY CAFETERIA ORDERS”. Para gerar o QR code (fig. 31) que deverá apresentar no terminal da cafeteria, o utilizador só precisa de carregar no pedido da lista de pedidos.

## “Logout”

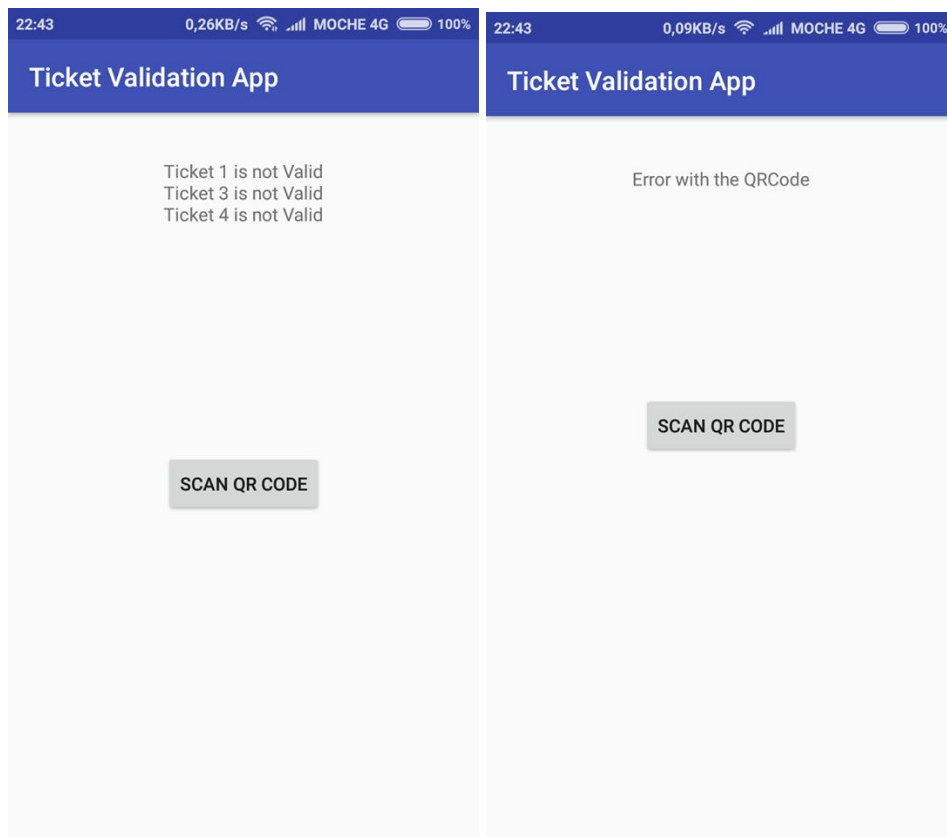
Ao carregar em logout, o utilizador terminará a sessão na aplicação e será redireccionado para a view de login/registo.

### 5.3. Ticket Validation Terminal

A aplicação contém apenas um botão “SCAN QR CODE” (fig. 32) que quando pressionado irá iniciar a aplicação de que trata, através da câmera do dispositivo, de ler o QR code dos bilhetes.



**Fig. 32 e 33.** View inicial do “Ticket Validation Terminal” e Mensagem de resposta da validação do QR Code

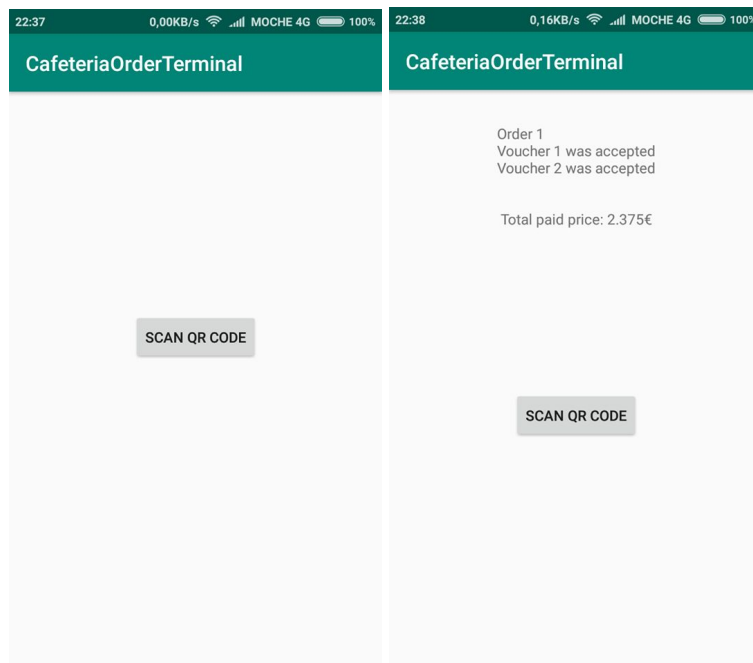


**Fig. 34 e 35.** Mensagem de resposta da validação do QR Code e mensagem de erro

Após fazer o scan do QR code, irá apresentar quais os tickets que foram (ou não) válidos. Na (fig. 33) os bilhetes eram todos válidos, mas se voltarmos a fazer o scan, já vão ser inválidos (fig. 34). Além disso mostra mensagens de erro, como por exemplo (fig. 35) onde se tentou fazer o scan de um QR Code da cafeteria.

## 5.4. Cafeteria Order Terminal

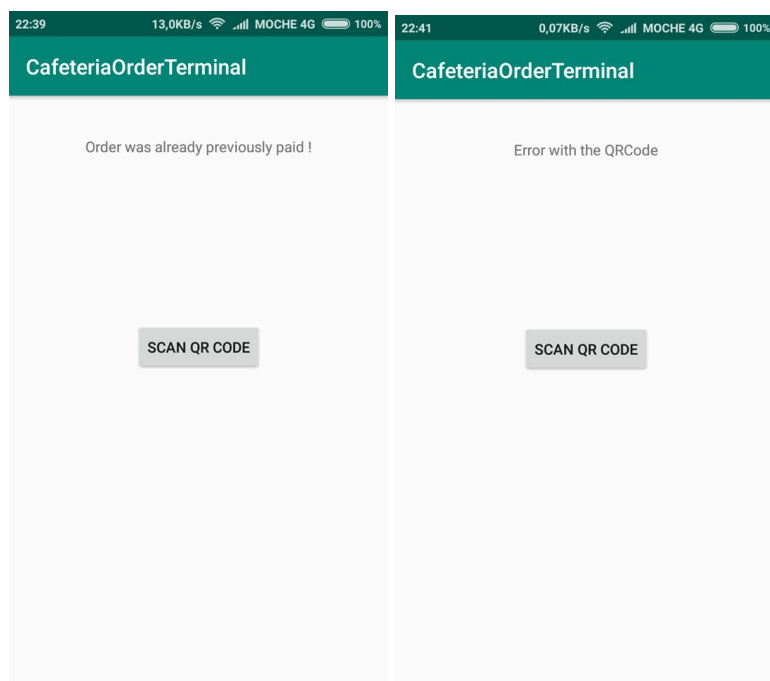
De forma similar à aplicação “Ticket Validation Terminal”, esta aplicação também contém apenas um botão “SCAN QR CODE” (fig. 36) que quando pressionado irá iniciar a aplicação de que trata, através da câmara do dispositivo, de ler o QR code do pedido para a cafeteria.



**Fig. 36 e 37.** View inicial do “Cafeteria Order Terminal” e Mensagem de resposta do processamento do QR Code

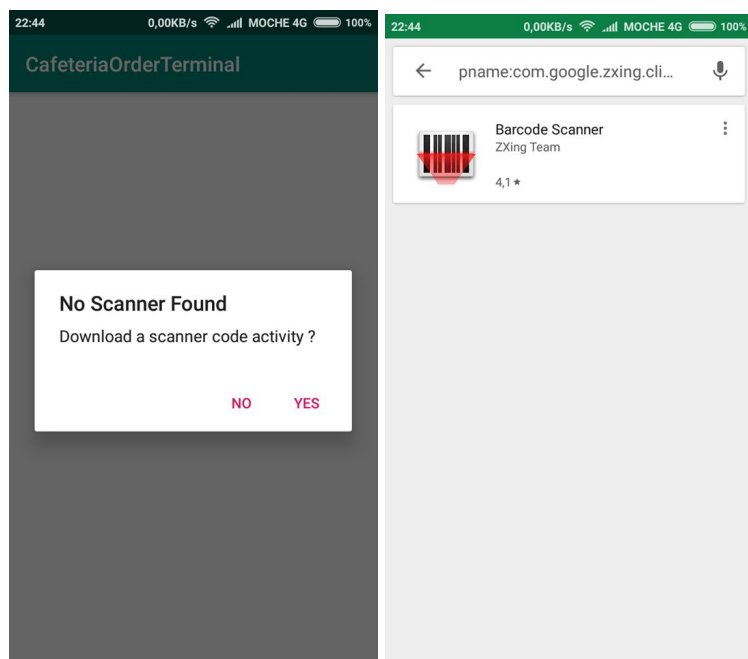
Após fazer o scan do QR code, irá apresentar qual o número do pedido e os vouchers que foram (ou não) aceites. Na (fig. 37) os dois vouchers foram aceites, sendo que com o desconto de 5% e do café grátis através dos vouchers, o utilizador pagou apenas 2 euros e 375.

Se voltarmos a fazer scan do mesmo QR Code, vai aparecer que o pedido já foi pago (fig. 38). Além disso mostra mensagens de erro, como por exemplo (fig. 39) onde se tentou fazer o scan de um QR Code dos bilhetes.



**Fig. 38 e 39.** Resposta de scan de um pedido que já foi pago e mensagem de erro de leitura do QR Code

Caso o utilizador não tenha a aplicação de scan de QR Codes no seu dispositivo, ao carregar no botão “SCAN QR CODE” (tanto no “Ticket Validation Terminal”, como no “Cafeteria Order Terminal”) será apresentado um pop-up a perguntar se quer fazer o download da aplicação de scan (fig. 40). Caso carregue “YES”, o utilizador será levado para a playstore para fazer o download da mesma (fig. 41)



**Fig. 40 e 41.** Pop-up de pedido de download da aplicação de scan e view da aplicação na playstore



## 6. Conclusão

Sentimos que este projecto foi uma óptima oportunidade de aprender/consolidar conhecimentos em Android. Foi necessária a implementação de diferentes elementos de interacção com o utilizador, desenhando vários *layouts*, formulários, listas, menus e acções. Adicionando a isto o aspecto de comunicação com um servidor remoto, a sua implementação de lógica interna e serviço REST associado com tantos tipos de pedidos discretos levam a complexidade deste projecto além do que seria ideal, ou mesmo necessário, para adquirir os conhecimentos seriam obtidos implementando somente as funcionalidades de compra e validação de bilhetes, esquecendo a cafeteria e os *vouchers*. A constrição trazida pelos prazos estipulados levou-nos a cortar a direito no desenho das interfaces, sendo que na computação móvel, onde quem interage maioritariamente são utilizadores casuais, o desenho da interface, tanto a nível estético como de intuição e facilidade de uso, tem uma importância muito maior do que, diga-se, em software de gestão de empresas onde os utilizadores recebem treino para interagir com essas aplicações.