Ticket and Payment System

Faculdade de Engenharia da Universidade do Porto
Mestrado Integrado em Engenharia Informática e Computação
Computação Móvel - EIC0050

Diogo Serra Duque - 201406274 (up201406274@fe.up.pt)
Sara Filipa Couto Fernandes - 201405955 (up201405955@fe.up.pt)

19 de Novembro de 2018

Conteúdo

1 Introdução **4**

2 Arquitetura **5**
2.1 Arquitetura Física 5
2.1.1 Tecnologias utilizadas 5
2.1.2 Servidor 6
2.2 Arquitetura Lógica 7
2.2.1 *Customer* 7
2.2.2 *Cafeteria* 8
2.2.3 Terminal 8

3 Interações Principais do Sistema **9**

4 Modelo de Domínio **10**

5 Funcionalidades **12**
5.1 User Stories 12
5.1.1 *Customer* 12
5.1.2 *Cafeteria* 13
5.1.3 Terminal 13
5.2 Mockups 13
5.2.1 *Customer* 13
5.2.2 *Cafeteria* 16
5.2.3 Terminal 17

6 Testes **18**
6.1 Registo 18
6.2 *Login* 18
6.3 Consultar novos eventos 18
6.4 Comprar bilhetes 18
6.5 Apresentar bilhetes 18
6.6 Validar bilhetes 19
6.7 Emitir uma ordem de *cafeteria* 19
6.8 Validar ordens de cafeteria 19
6.9 Pagar uma ordem de *cafeteria* 19
6.10 Consultar transações 19
6.11 Consultar *vouchers* não validados 20
6.12 Visualizar perfil 20
6.13 Editar perfil 20
6.14 *Logout* 20

7 Modo de Funcionamento **21**
7.1 Instruções de instalação 21
7.2 Instruções de utilização 21
7.2.1 *Customer* 21

7.2.2 *Cafeteria* 27
7.2.3 Terminal 29

8 Conclusão **31**

9 Recursos **32**
9.1 Bibliografia 32
9.2 *Software* Utilizado 32

1 Introdução

O projeto "***Ticket and Payment System***"foi proposto no âmbito da unidade curricular de **Computação Móvel**, consistindo no desenvolvimento de um sistema distribuído, em que é necessário implementar três aplicações móveis distintas, todas com o sistema operativo ***Android*** e um servidor, que neste caso foi implementado recorrendo à tecnologia ***NodeJS***.

Primeiramente, tem de se ter a aplicação ***Customer***, em que um utilizador poderá registar-se e, posteriormente, conseguirá visualizar os espetáculos futuros, os espetáculos para os quais comprou bilhetes e os *vouchers* que lhe foram atribuídos. Para além disso, este utilizador poderá comprar bilhetes para os respetivos espetáculos e, posteriormente, conseguirá fazer pedidos de *cafeteria*, onde poderá utilizar *vouchers* a descontar no valor total a pagar pelo seu pedido.

Na aplicação referente à ***Cafeteria***, um utilizador poderá verificar os pedidos efetuados e conseguirá, consequentemente, verificar os *vouchers* associados aos pedidos. Para tal recorreu-se à transmissão de informação via ***QR codes***.

Por último, na aplicação do **Terminal** serão validados os bilhetes referentes aos espetáculos, recorrendo a ***QR codes***, tal como acontece na aplicação ***Cafeteria***.

2 Arquitetura

De seguida, estão explicadas e exemplicadas as **arquiteturas física** e **lógica** do sistema implementado.

2.1 Arquitetura Física

O sistema baseia-se num modelo **cliente-servidor**, sendo que se tem três aplicações **cliente**, desenvolvidas no ***Android Studio***, para o sistema operativo ***Android*** e uma aplicação **servidor**, desenvolvida em ***NodeJS***.

Toda a informação do sistema é guardado numa base de dados em ***PostgreSQL***, existente no lado do **servidor**.

Esta informação relativa à arquitetura física do sistema pode ser verificada na **figura 1**, apresentada de seguida.

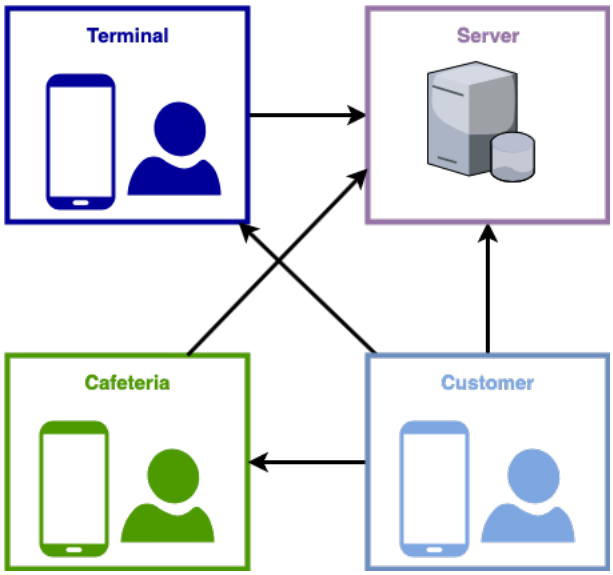

Figura 1: Arquitetura física do sistema

2.1.1 Tecnologias utilizadas

Como referido, anteriormente, na **Secção 2.1**, para a criação das **aplicações móveis** com sistema operativo ***Android***, utilizou-se o ***Android Studio***, sendo que a linguagem de programação usada foi **Java**. Na aplicação ***Customer*** são ainda implementados ***QR codes*** e nas aplicações ***Cafeteria*** e **Terminal** fazem-se a leitura dos mesmos, para a validação das ordens de *cafeteria* e para a validação dos bilhetes dos espetáculos, respetivamente. Para tal usou-se a biblioteca ***zxing*** do ***Google***. Ainda na aplicação ***Customer*** utilizou-se ***Room Persistence Library***, para guardar, localmente, os bilhetes comprados e os *vouchers*.

Para além disso, utilizou-se ***NodeJS*** para a crição do **servidor**, sendo que os pedidos à API são efetuados através de serviços ***REST***. A base de dados, utilizada nos pedidos efetuados, foi implementada em ***PostgreSQL***.

2.1.2 Servidor

Como já referido na **Secção 2.1.1**, o servidor foi desenvolvido em ***NodeJS***. Neste servidor foi desenvolvida uma **API**, com diferentes routes, de forma a fazer face às necessidades das diferentes aplicações.

De seguida, são explicadas as diferentes chamadas à **API**.Na Tabela 1 são apresentadas as chamadas feitas pela aplicação ***Customer***, na Tabela 2 são apresentadas as chamadas feitas pela aplicação ***Cafeteria*** e na Tabela 3 são apresentadas as chamadas executadas pela aplicação **Terminal**.

Tipo	*Rota*	*Parâmetros*	*Descrição*
GET	/customer/auth		Verifica se o utilizador está *logged in* no sistema
POST	/customer/auth/register	name, username, nif, password, cardNumber, cardCode, cardValidity, cardType, publicKey	Regista o utilizador no sistema
POST	/customer/auth/login	username, password	Faz o *login* do utilizador
GET	/customer/ auth/logout		Faz o *logout* do utilizador
GET	/customer/show/ next_shows		Retorna os próximos eventos
GET	/customer/profile		Retorna os dados pessoais do utilizador
PUT	/customer/profile	name, username, nif, password	Edita a informação pessoal do utilizador
PUT	/customer/ profile/ credit_card	card_type, card_number, card_validity	Edita a informação do cartão bancário do utilizador
GET	/customer/tickets/ not_used_tickets		Retorna os bilhetes não validados
GET	/customer/tickets/ all_tickets		Retorna os todos os bilhetes
PUT	/customer/tickets	showId, quantity	Efetua a compra dos bilhetes
GET	/customer/vouchers/ my_vouchers		Retorna os *vouchers* do utilizador
GET	/customer/vouchers/ my_vouchers_by_status	status	Retorna os *vouchers* do utilizador por *status*

Tipo	*Rota*	*Parâmetros*	*Descrição*
GET	/customer/tickets	showId, quantity	Efetua a compra dos bilhetes
GET	/customer/cafeteria/ orders_customer_validated		Retorna as ordens de *cafeteria* do utilizador que já foram validadas
GET	/customer/cafeteria/ order_products	id	Retorna os produtos de uma determinada ordem
GET	/customer/cafeteria/ product_price	product	Retorna o preço de um produto
POST	/customer/cafeteria/ make_order	date	Efetua uma ordem de *cafeteria*

Tabela 1: Tabela com as chamadas à **API** feitas pela aplicação ***Customer***

Tipo	*Rota*	*Parâmetros*	*Descrição*
PUT	/cafeteria/verify	JSONObject	Verifica a uma ordem de *cafeteria*

Tabela 2: Tabela com as chamadas à **API** feitas pela aplicação ***Cafeteria***

Tipo	*Rota*	*Parâmetros*	*Descrição*
PUT	/cafeteria/verify	JSONObject	Verifica a uma ordem de *cafeteria*

Tabela 3: Tabela com as chamadas à **API** feitas pela aplicação **Terminal**

2.2 Arquitetura Lógica

2.2.1 *Customer*

O módulo ***Customer*** inclui a aplicação móvel ***Android*** (*layouts* e atividades), em que o utilizador pode comprar os bilhetes para um espetáculo e, posteriormente, pode fazer pedidos para a ***cafeteria***. Para além disso, pode verificar a listagem de futuros espetáculos, espetáculos para os quais tem bilhete comprado e os *vouchers* que lhe foram atribuídos. Para além disso, tem ainda acesso às suas transações e ao seu perfil, podendo editá-lo.

De seguida, é apresentado um diagrama de fluxo de atividades desta aplicação (**figura 2**).

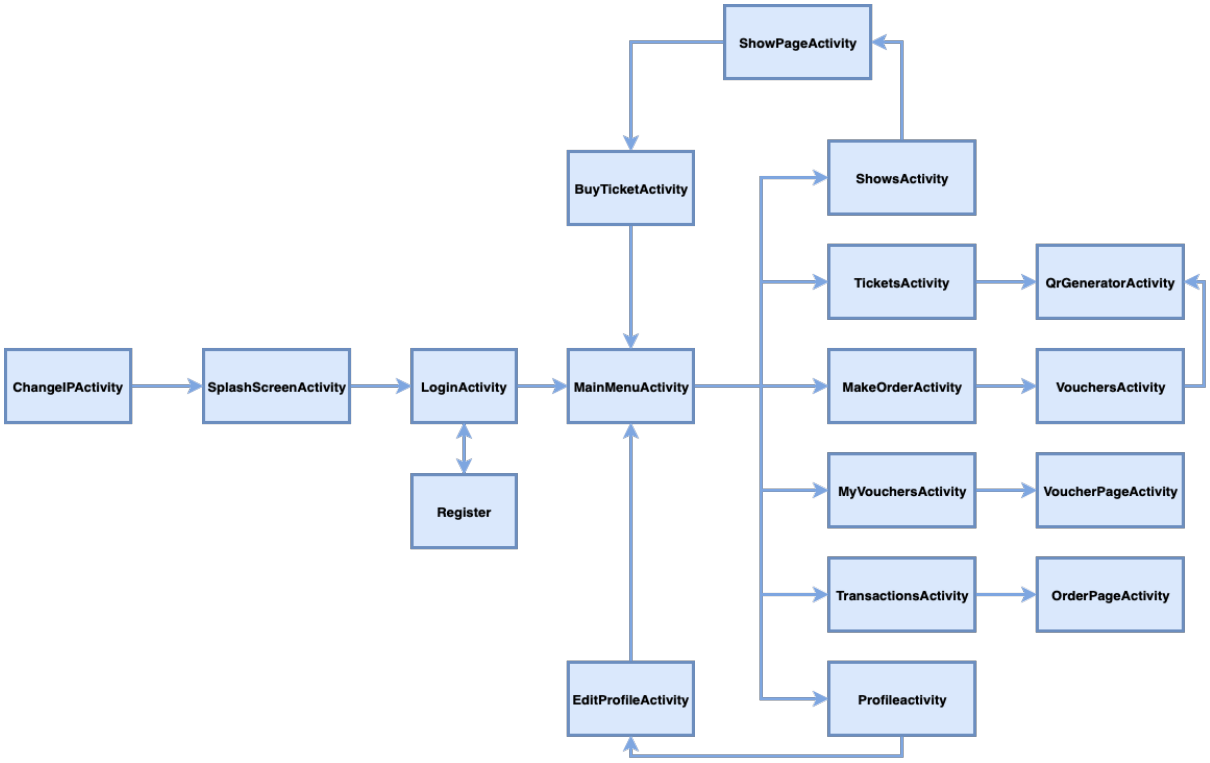

Figura 2: Fluxo de atividades na aplicação ***Customer***

2.2.2 *Cafeteria*

O módulo ***Cafeteria*** inclui a aplicação móvel ***Android*** (*layouts* e atividades), em que se validam as ordens de *cafeteria* utilizando ***Qr codes***, como referido na **Secção 2.1.1**.

De seguida, é apresentado um diagrama de fluxo de atividades desta aplicação (**figura 3**).

Figura 3: Fluxo de atividades na aplicação ***Cafeteria***

2.2.3 Terminal

O módulo **Terminal** inclui a aplicação móvel ***Android*** (*layouts* e atividades), em que o utilizador pode validades os bilhetes comprados para espetáculos, recorrendo à leitura de ***QR codes***, como referido na **Secção 2.1.1**.

De seguida, é apresentado um diagrama de fluxo de atividades desta aplicação (**figura 4**).

Figura 4: Fluxo de atividades na aplicação **Terminal**

3 Interações Principais do Sistema

Nesta secção são apresentadas as interações principais presentes no sistema desenvolvido, através de um diagrama de sequências.

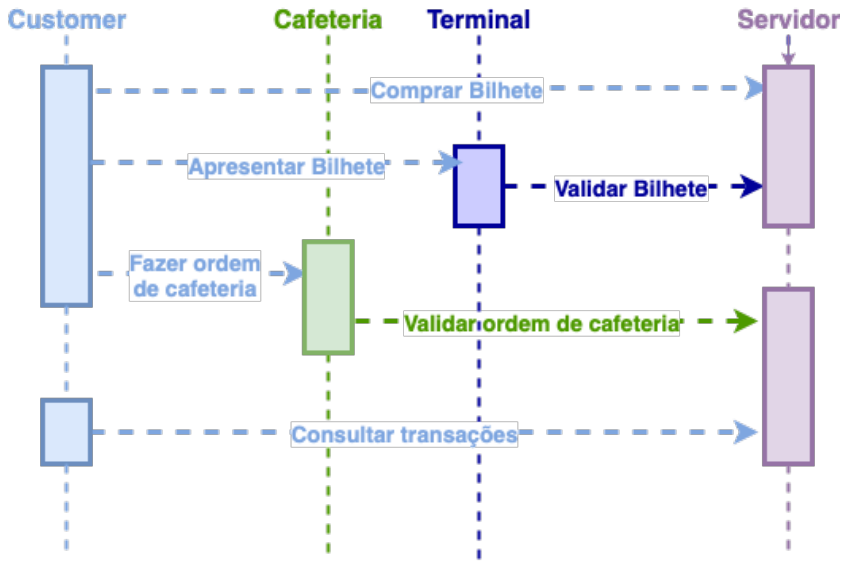

Figura 5: Diagrama de sequências, com as principais interações presentes no sistema

O sistema implementado permite que um *customer* possa comprar bilhetes para um evento, sendo que essa informação é passada ao servidor. Posteriormente, este utilizador pode apresentar os seus bilhetes ao **terminal de validação**, que os irá verificar via ***QR code***. Para que a validação seja feita, o **terminal** necessita de verificar a informação, passada por ***QR code***, no servidor.

Um *customer* pode, ainda, efetuar ordens de cafeteria, em que após efetuar a sua ordem apresenta um ***QR code*** ao **terminal de *cafeteria***, sendo que este terminal comunica com o servidor de modo a verificar se a ordem passada por ***QR code*** é válida, nomeadamente, verifica se os seus *vouchers* utilizados, são verdadeiros e podem ser descontados.

Por fim, o utilizador do tipo *customer* pode ainda verificar as suas transações passadas, nomeadamente **bilhetes** que comprou (validados ou não) e **ordens de *cafeteria***, já validadas e pagas, respetivamente. Ao verificar as suas transações, são atualizados os dados locais onde são armazenadas as informações referentes aos **bilhetes** comprados e aos *vouchers* já utilizados.

Para além das interações indicadas, efetuaram-se ainda outras interações de menor importância. Uma delas é referente à visualização e edição do perfil, incluindo a informação do cartão bancário, de um *customer*. Depois, incluiu-se ainda a possibilidade do utilizador conseguir visualizar todos os seus *vouchers* que ainda não foram validados, podendo verificar o seu conteúdo.

4 Modelo de Domínio

De seguida, através da **figura 6**, é apresentado o modelo que serviu de base para a construção da base de dados do sistema.

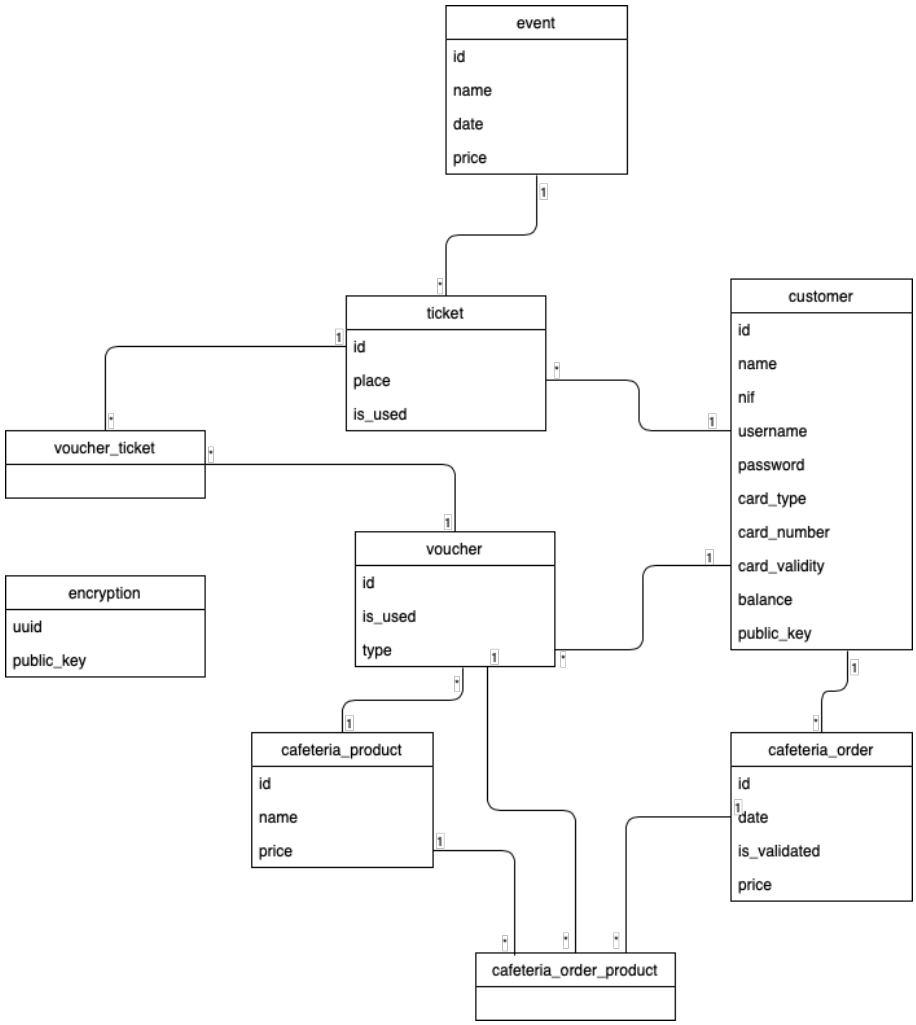

Figura 6: Modelo de Domínio

De seguida é representado o modelo relacional referente à base de dados visualizada, anteriormente, na **figura 6**.

- **customer**(id, name, nif, username, password, card_type, card_number, card_-

validity, balance, public_key)

- **event**(id, name, nif, date, price)
- **ticket**(id, place, is_used, customer_id -> **customer**, event_id -> **event**)
- **cafeteria_order**(id, date, customer_id -> **customer**, is_validated, price)
- cafeteria_product(id, name, price)
- **voucher**(id, is_used, type, customer_id -> **customer**, product_id -> **cafeteria_product**)
- **voucher_ticket**(voucher_id -> **voucher**, ticket_id -> **ticket**)
- **cafeteria_order_product**(order_id -> **cafeteria_order**, product_id -> **cafeteria_product**, voucher_id -> **voucher**)
- **encryption**(uuid, public_key)

5 Funcionalidades

5.1 User Stories

5.1.1 *Customer*

ID	*Nome*	*Descrição*
US01	Registo	Como ***customer*** quero ser capaz de me registar na aplicação, de forma a conseguir aceder às funcionalidades da mesma
US02	Cartão bancário	Como ***customer*** quero ser capaz de associar o meu cartão quando me registo, de forma a conseguir efetuar pagamentos
US03	*Login*	Como ***customer*** quero ser capaz de fazer *login* usando *username* e *password*, de forma a conseguir aceder às funcionalidades da mesma
US04	Ver eventos	Como ***customer*** quero ser capaz de ver todos os eventos disponíveis, de forma a conseguir saber quais os próximos eventos que irão ocorrer
US05	Ver meu eventos	Como ***customer*** quero ser capaz de ver todos os eventos para o qual comprei bilhete, de forma a saber quais os eventos a que tenho que ir
US06	Ver *vouchers*	Como ***customer*** quero ser capaz de ver os meus *vouchers*, de forma a saber quais os *vouchers* que ainda posso utilizar
US07	Comprar bilhete	Como ***customer*** quero ser capaz de comprar bilhetes para um evento, de forma a conseguir ir ao evento respetivo
US08	Fazer ordem cafeteria	Como ***customer*** quero ser capaz de fazer uma compra de cafeteria, de forma a conseguir obter os produtos desejados durante o evento
US09	Receber *voucher*	Como ***customer*** quero ser capaz de receber um *voucher*, de forma a conseguir descontar o total a pagar numa ordem de *cafeteria*
US10	Usar *voucher*	Como ***customer*** quero ser capaz de usar os meus *vouchers*, de forma a conseguir descontar o total a pagar numa ordem de *cafeteria*
US11	Apresentar bilhetes	Como ***customer*** quero ser capaz de apresentar os meus bilhetes de forma a que estes sejam validados
US12	Apresentar ordens de *cafeteria*	Como ***customer*** quero ser capaz de apresentar as minhas ordens de cafeteria de forma a que estas sejam validadas
US13	Consultar transações	Como ***customer*** quero ser capaz de consultar as minhas transações, de forma a ver os bilhetes que comprei e as ordens de *cafeteria* que paguei

ID	*Nome*	*Descrição*
US14	Visualizar perfil	Como ***customer*** quero ser capaz de visualizar o meu perfil, de forma a conseguir verificar as minhas informações pessoais
US15	Editar perfil	Como ***customer*** quero ser capaz de editar o meu perfil, de forma forma a conseguir alterar as minhas informações pessoais
US16	*Logout*	Como ***customer*** quero ser capaz de fazer *logout* da aplicação, de forma a conseguir terminar a minha sessão na mesma

5.1.2 *Cafeteria*

ID	*Nome*	*Descrição*
US17	Validar ordens de *cafeteria*	Como ***cafeteria*** quero ser capaz de validar as ordens efetuadas, de forma a verificar se as mesmas são válidas, nomeadamente, verificar se os *vouchers* usados são verdadeiros

5.1.3 Terminal

ID	*Nome*	*Descrição*
US18	Validar bilhetes	Como **terminal** quero ser capaz de validar os bilhetes, de forma a verificar se os mesmos são válidos

5.2 Mockups

Nesta secção, são exemplificadas as ***User Stories*** referidas na **Secção 5.1**, sob a forma de *mockups*.

5.2.1 *Customer*

De seguida são apresentados os *mockups* referentes à aplicação móvel do ***customer*** (**figuras 7, 8, 9, 10, 11 e 12**).

Figura 7: Registo e *login* na aplicação

Na **figura 7** são apresentadas as interfaces iniciais, em que se consegue fazer o registo ou *login* na aplicação, de modo a que, posteriormente, se consiga aceder às funcionalidades da mesma.

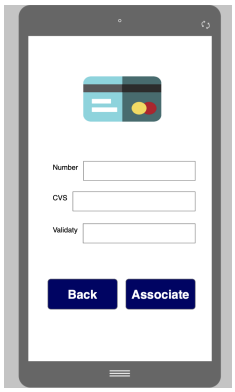

Figura 8: Registo do cartão de multibanco

Na **figura 8** é apresentado um *mockup* referente à interface de registo do cartão de multibanco do utilizador, de forma a posteriormente se conseguir efetuar pagamentos na aplicação.

Figura 9: Listagem de próximos eventos, eventos para os quais o ***customer*** tem bilhetes e *vouchers* do mesmo

A **figura 9** apresenta as diferentes listagens que a aplicação tem. A partir destas listagens o utilizador pode efetuar a compra de bilhetes, fazer ordens de cafeteria ou utilizar *vouchers*, por exemplo.

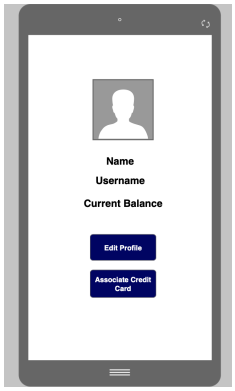

Figura 10: Perfil do utilizador

A **figura 10** apresenta a interface em que é possível visualizar o perfil do utilizador e, consequentemente, editar quer os seus dados pessoais quer os dados do cartão de multibanco que lhe está associado.

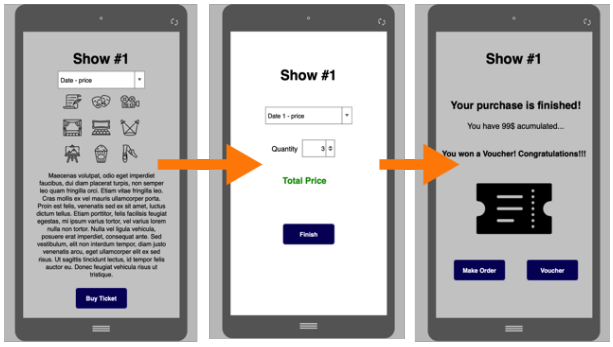

Figura 11: Compra de bilhetes para um evento

Na **figura 11** está representada a sequência respetiva, para que seja efetuada uma compra de bilhetes para um evento.

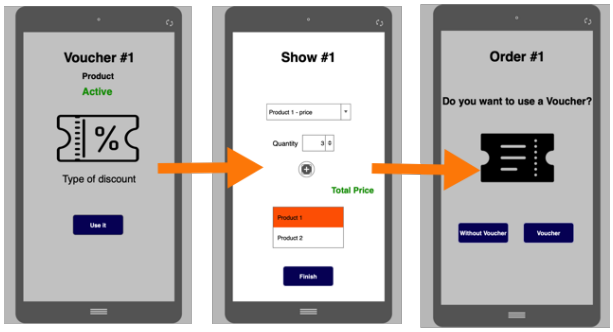

Figura 12: Pedido de *cafeteria*

Na **figura 12** está representada a sequência respetiva, para que seja efetuada ordem de *cafeteria*.

5.2.2 *Cafeteria*

De seguida são apresentados os *mockups* referentes à aplicação móvel da ***cafeteria*** (**figura 13**).

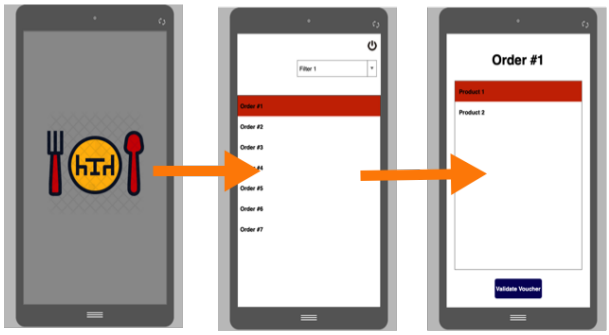

Figura 13: Visualização de pedidos de *cafeteria*, podendo validar os *vouchers*

Na **figura 13** está representada a sequência necessária para que na aplicação da ***cafeteria*** sejam visualizadas as ordens efetuadas por um ***customer*** e, posteriormente, sejam feitas as respetivas validações das mesmas.

5.2.3 Terminal

De seguida são apresentados os *mockups* referentes à aplicação móvel do **Terminal** (**figura 14**).

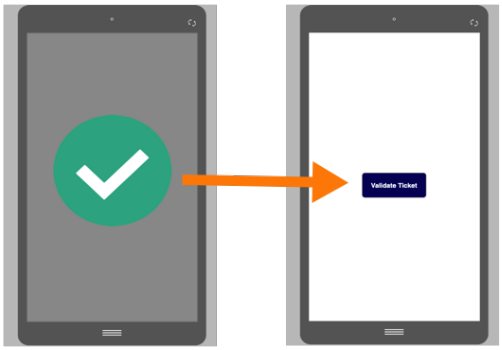

Figura 14: Validação de bilhetes para eventos

Na **figura 14** está representada a sequência necessária para que na aplicação do **terminal** se possa validar um ou mais bilhetes comprados para eventos, através da leitura de ***QR codes***.

6 Testes

De seguida são enumerados os testes executados para verificar o funcionamento das principais funcionalidades implementadas.

6.1 Registo

De forma a testar a funcionalidade de registo de um ***customer***, inseriram-se alguns valores quer na parte de registo da informação pessoal do utilizador quer na parte de registo do cartão bancário. Caso esses valores estejam incorretos ou já existam, é apresentada uma mensagem de erro. Caso o registo seja efetuado com sucesso, o utilizador é redirecionado para a página de *login*.

6.2 *Login*

De forma a testar a funcionalidade de *login*, inseriram-se alguns valores quer no campo referente ao *username* como à *password*. Caso esses valores estejam incorretos, é apresentada uma mensagem de erro. Caso o *login* seja efetuado com sucesso, o utilizador é redirecionado para a página principal da aplicação.

6.3 Consultar novos eventos

De modo a que se conseguisse testar a consulta de eventos, criaram-se diversos eventos na base de dados e verificou-se, posteriormente, que na página referente à visualização de novos eventos se estes estavam lá presentes e se a informação visualizada correspondia à inserida na base de dados.

6.4 Comprar bilhetes

De modo a que se conseguisse testar a compra de bilhetes para os eventos, inseriram-se os dados necessários para efetuar a respetiva compra, nomeadamente, o número de bilhetes a serem comprados, na interface criada para tal. Caso a compra seja efetuada com sucesso, o utilizador recebe uma mensagem na aplicação a confirmar a sua compra e a informá-lo dos *vouchers* que este ganhou. Caso a compra não seja efetuada com sucesso, uma mensagem de erro é apresentada ao utilizador.

6.5 Apresentar bilhetes

De modo a testar a apresentação de bilhetes por parte do utilizador, criou-se uma interface onde este pode selecionar os bilhetes que pretende apresentar ao **terminal**. Caso este selecione mais do que 4 bilhetes ou os bilhetes selecionados não sejam relativos ao mesmo evento e à mesma data, uma mensagem de erro é apresentada. Caso não ocorram os aspetos, anteriormente, descritos o utilizador é redirecionado para uma nova página onde é criado o ***QR code*** necessário para a validação dos bilhetes.

6.6 Validar bilhetes

De modo a validação dos bilhetes, na aplicação do **terminal**, criou-se uma interface onde é efetuada a leitura do ***QR code*** apresentado pelo ***customer***. Ao ler o ***QR code***, o **terminal** envia a informação transmitida para servidor, verificando a veracidade dos bilhetes. Caso a informação transmitida não valide os bilhetes, a aplicação do **terminal** apresenta uma interface referindo que os bilhetes não foram validados. Caso a informação sejam validada pelo servidor, é apresentada uma interface a referir que todos os bilhetes foram validados.

6.7 Emitir uma ordem de *cafeteria*

De forma a testar a funcionalidade de emição de uma ordem de *cafeteria*, criou-se uma interface para esse propósito. Aí o utilizador adiciona os produtos que pretende comprar e consequentemente, numa interface seguinte, pode ou não adicionar os *vouchers* que quer utilizar para descontar no total a pagar pela ordem. Caso a ordem seja bem criada, em que o utilizador só pode utilizar um *voucher* de desconto de 5% e um *voucher* a descontar um produto, é gerado um ***QR code***. Caso a ordem tenha erros, nomeadamente, a nível dos *vouchers* selecionados, uma mensagem de erro é apresentada.

6.8 Validar ordens de cafeteria

De forma a testar a validação de ordens de *cafeteria*, nomeadamente verificar os *vouchers* nelas utilizados, criou-se uma interface na aplicação da ***cafeteria***, onde é efetuada a leitura do ***QR code*** referente à ordem. Caso a informação passada pelo ***QR code*** seja validade pelo servidor, a ordem é verificada e a aplicação da ***cafeteria*** redireciona para uma nova interface que faz referência ao facto da ordem ter sido validada. Caso tenho ocorrido erro na validação da ordem, aplicação da ***Cafeteria*** redireciona para uma nova interface que faz referência ao facto da ordem não ter sido validada pelo servidor.

6.9 Pagar uma ordem de *cafeteria*

De forma a testar o pagamento de uma ordem de *cafeteria*, após a validação da respetiva ordem por parte da aplicação da ***Cafeteria***, o ***customer*** recebe na sua aplicação a informação do custo final da ordem que este efetuou, sabendo quando será descontado no seu cartão bancário. Esta informação só é verificada caso a ordem seja, previamente, validada pela outra aplicação.

6.10 Consultar transações

De modo a testar a consulta de transações efetuadas pelo ***customer***, criou-se uma interface onde este pode verificar quer os bilhetes que comprou (validados ou não), assim como todas ordens de *cafeteria* anteriormente executadas. Neste caso, basta verificar que caso se compre um bilhete ou se valide uma ordem de *cafeteria*, respetivamente, estes apareçam nas respetivas listagens.

6.11 Consultar *vouchers* não validados

De modo a testar a consulta de *vouchers* não validados, criou-se uma interface onde este pode verificar os respetivos *vouchers* que ainda não foram utilizados. Neste caso, basta verificar que caso se valide uma ordem de *cafeteria*, contendo um determinado *voucher*, este desaparece da listagem *vouchers* não validados.

6.12 Visualizar perfil

De modo a testar a visualização do perfil do utilizador, criou-se uma interface para esse propósito, em que caso o pedido ao servidor pela informação pessoal do mesmo seja bem processado, deve ser apresentado o nome, *username* e total gasto até agora na aplicação quer na compra de bilhetes quer na emissão de ordens de *cafeteria*.

6.13 Editar perfil

De modo a testar a edição do perfil do utilizador, criou-se uma interface para esse propósito. Nesta interface o utilizador pode alterar qualquer informação referente a si próprio quer relativa ao seu cartão bancário registado. Para efetuar a edição este tem de obrigatoriamente de inserir a sua *password*, de forma a confirmar a edição. Caso a edição seja efetuada com sucesso, este deve conseguir visualizar a sua nova informação no seu perfil. Caso não consiga efetuar a edição de perfil é visualizada uma mensagem de erro.

6.14 *Logout*

De forma a testar a funcionalidade de *logout*, o utilizador tem de clicar na opção relativa a tal no menu principal da aplicação. Caso esta operação seja efetuada com sucesso, o utilizador é redirecionado para a página de *login*. Caso ocorra algum erro, o utilizador visualiza uma mensagem de erro.

7 Modo de Funcionamento

7.1 Instruções de instalação

Para que se possa executar as diferentes aplicações em pleno é necessário executar diferentes instalações, quer em relação ao servidor quer em relação às aplicações.

Para instalar e executar o servidor é necessário:

- Instalar **Docker** (`https://www.docker.com`) e **Docket-Compose** (`https://docs.docker.com/compose/install/#prerequisites`), uma vez que se usa **Docker** para correr o servidor
- Executar um terminal no diretório **TP1** deste projeto
- Executar o comando **docker-compose build**
- Executar o comando **docker-compose up**
- Se não ocorreram erros ao executar estes passos, o servidor está operacional

Para instalar e correr as aplicações basta-se aceder ao **Android Studio** no diretório "*project/frontend*"e, individualmente, já que as aplicações fazem parte do mesmo projeto, compilá-las e executá-las num telemóvel com sistema operativo **Android** mínimo de **4.4 - KitKat**.

7.2 Instruções de utilização

De seguida são apresentadas as instruções de utilização das diferentes aplicações móveis desenvolvidas[1].

7.2.1 *Customer*

A aplicação do ***Customer*** tem como objetivo permitir ao utilizador comprar bilhetes para eventos/espetáculos e fazer ordens de *cafeteria*. Para tal, esta inicia com uma interface onde o utilizador pode configurar o **IP** do servidor, como se pode ver na **figura 15**.

[1] Todas os *layouts* apresentados são os *layouts* visualizados no menu de edição do ***Android Studio***, sendo que algumas informações, nomeadamente, no que diz respeito às diversas listagens poderão ser apresentadas de forma diferente no dispositivo móvel

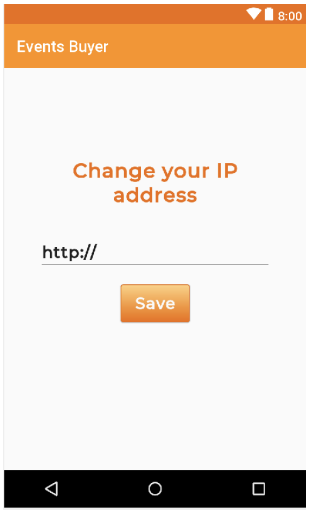

Figura 15: Alteração do **IP** da aplicação

Após a inserção do **IP**, o utilizador pode-se registar na aplicação, inserindo a sua informação pessoal (nome, *username*, NIF e *password*) e a informação relativa ao seu cartão bancário (número, CSC, validade e tipo de cartão) ou pode fazer *login* na mesma, inserindo apenas o seu *username* e *password* (**figura 16**).

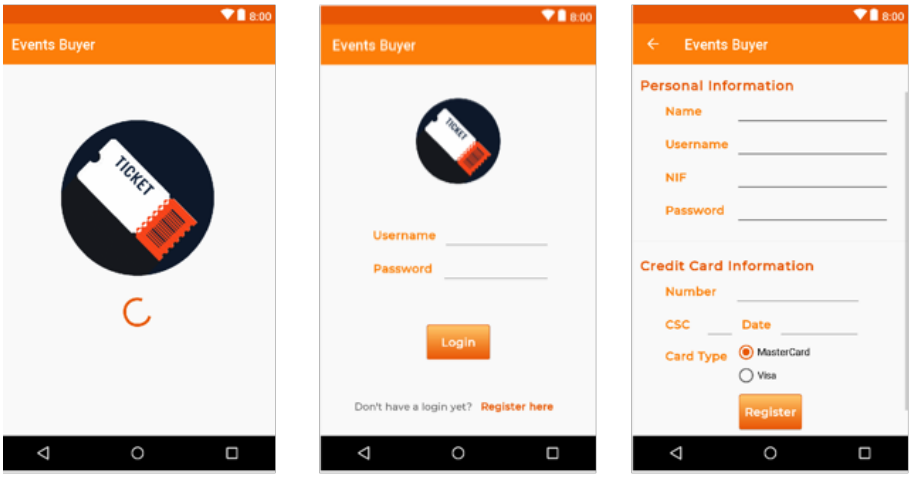

Figura 16: Ações para efetuar o *login* ou registo na aplicação

Após o *login* ou registo na aplicação, o utilizador é redirecionado para o menu principal, onde este pode escolher visualizar os próximos eventos, validar os seus bilhetes comprados, ver quais são os *vouchers* que ainda não utilizou, fazer uma

ordem de *cafeteria*, visualizar todas as suas transações, ver o seu perfil e fazer *logout* (**figura 17**).

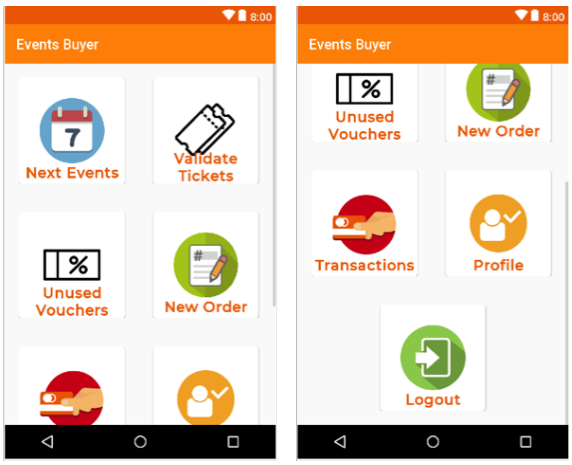

Figura 17: Menu principal da aplicação

Caso se faça *logout*, o utilizador é redirecionado para a página de *login*.

Caso se opte por ver quais são os próximos eventos/espetáculos, é-se redirecionado para a interface de listagem dos eventos, onde caso o utilizador clique num determinado evento este é redirecionado para a página individual do evento, podendo comprar bilhetes para esse espetáculo. Este fluxo pode ser verificado na **figura 18**. Terminando a compra este é redirecionado para o menu inicial.

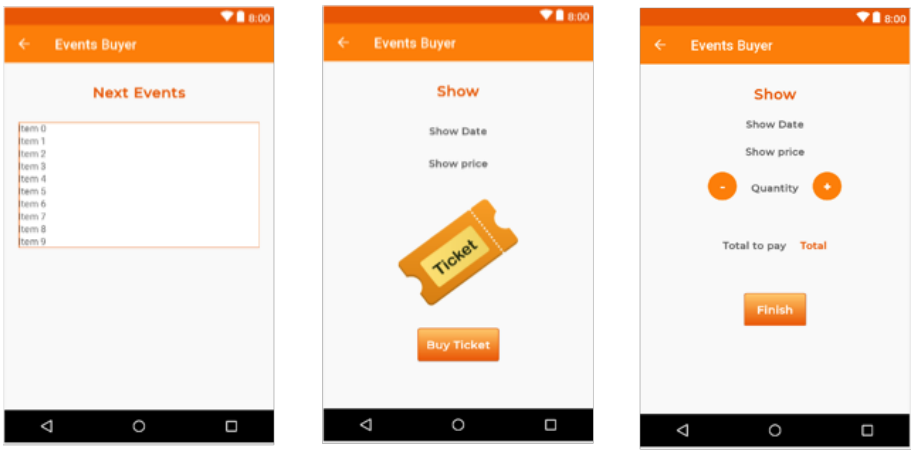

Figura 18: Sequência efetuada para comprar um bilhete, passando pela listagem de eventos/espetáculos, pela página individual do evento/espetáculo e terminando de compra de bilhetes

Caso se selecione a opção de validação de bilhetes, o utilizador é redirecionado para uma listagem de bilhetes por validar, onde este pode selecionar apenas 4 bilhetes no máximo, sendo que estes têm de ser referentes ao mesmo evento/espetáculo e têm de ser relativos à mesma data. Após concluir a seleção, seguindo as regras enumeradas, o utilizador é redirecionado para uma interface contendo o ***QR code***, que permite a validação dos bilhetes. Caso não se selecione corretamente os bilhetes a serem selecionados, uma mensagem de erro é apresentada. Esta sequência descrita pode ser visualizada na figura **19**.

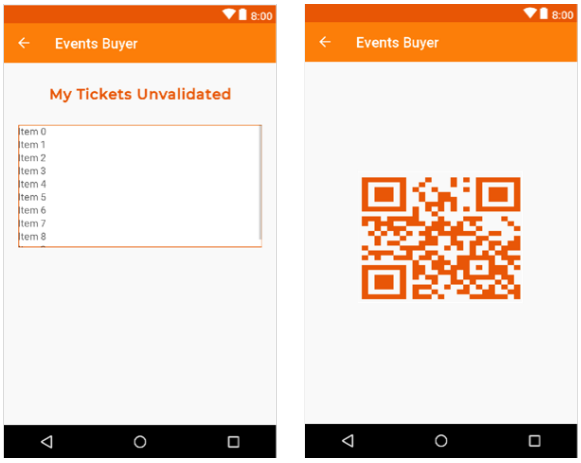

Figura 19: Sequência de validação de um ou mais bilhetes

Caso se queira visualizar os *vouchers* que ainda não foram utilizados, o utilizador é redirecionado para uma listagem dos *vouchers* respetivos. Caso na listagem o usuário clique num dos *vouchers* listados é-lhe apresentada uma interface onde este pode verificar os detalhes do respetivo *voucher*. Este fluxo entre interfaces pode ser verificado na **figura 20**.

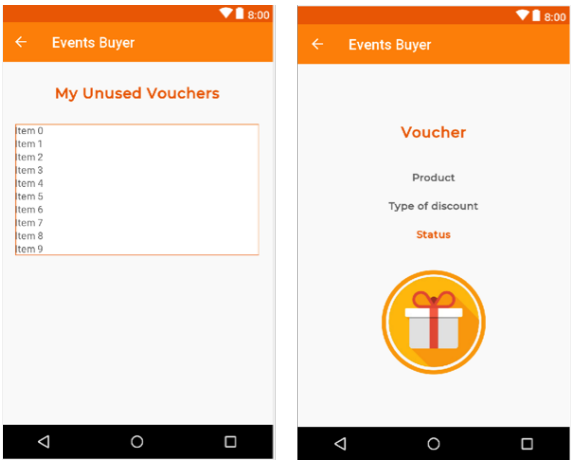

Figura 20: Sequência que permite visualizar os *vouchers* não validados, podendo ver individualmente os seus detalhes

Caso se pretenda efetuar uma ordem de *cafeteria*, o utilizador é redirecionado para uma interface onde este pode selecionar os produtos a serem comprados, indicando as respetivas quantidades pretendidas. Após essa primeira seleção, este avança para uma segunda interface onde pode selecionar os *vouchers* a usar na sua ordem de *cafeteria*, podendo somente usar um *voucher* de desconto de 5% e outro *voucher* referente a um produto, por ordem. Caso não siga as regras referidas, uma mensagem de erro é apresentada. Caso a ordem seja efetuada com sucesso, um ***QR code*** é apresentado de forma a que o utilizador possa validar a sua ordem. Toda a

sequência descrita pode ser verificada na **figura 21**.

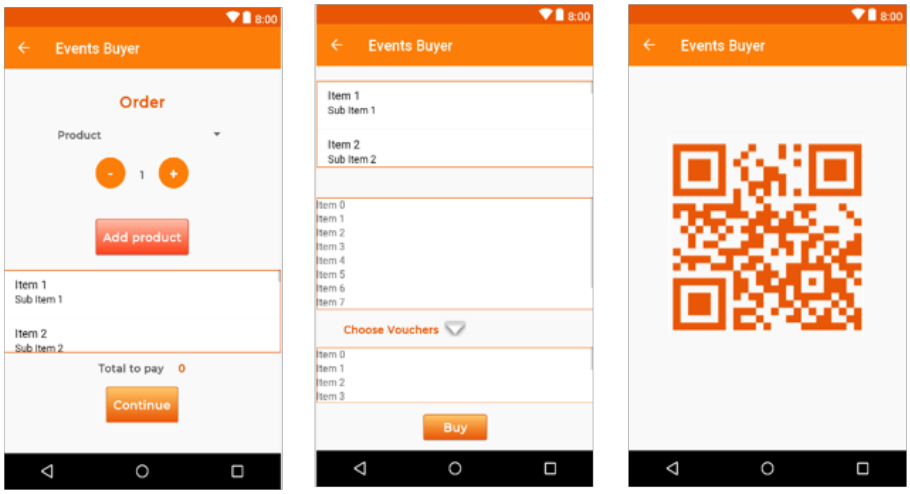

Figura 21: Sequência que permite efetuar uma ordem de *cafeteria*

Caso no menu inicial o utilizador tenha pretendido ver as suas transações, este é redirecionado para uma interface onde são listados todos os bilhetes que o utilizador comprou até ao momento, sendo detalhada nessa listagem todos os pormenores individuais de cada bilhete e, para além disso, são listadas todas as ordens de *cafeteria* já validadas. Ao aceder a esta interface, os dados guardados localmente na aplicação são atualizados, nomeadamente, os respetivos bilhetes e *vouchers* (*vouchers* estes usados nas ordens de *cafeteria*). Ao clicar numa ordem de cafeteria listada, o utilizador tem acesso a uma interface que detalha a ordem em questão, ao pormenor. A sequência descrita encontra-se representada, de seguida, na **figura 22**.

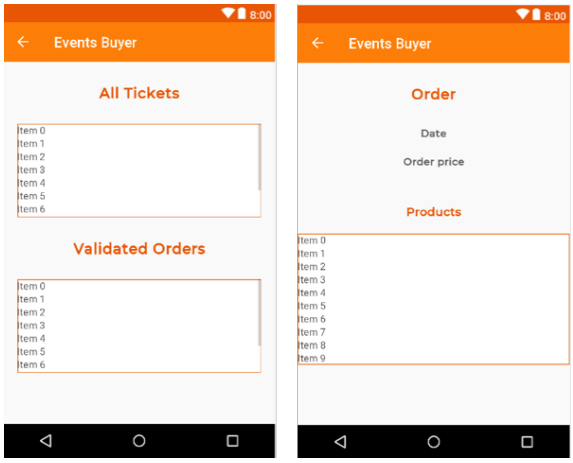

Figura 22: Sequência que permite visualizar as transações efetuadas por um utilizador, mostrando a interface que detalha uma ordem de *cafeteria*

Por fim, caso o utilizador queira ter acesso ao seu perfil, este é redirecionado para a respetiva interface, onde pode visualizar o seu nome, *username* e o valor total gasto na aplicação a comprar bilhetes ou a fazer ordens de *cafeteria*. Nesta interface, o utilizador pode ainda selecionar o botão de edição de perfil, sendo que após tal seleção é redirecionado para a interface relativa a tal propósito, podendo atualizar todos os campos presentes no formulário, sendo sempre obrigatória a inserção da sua *password*, de forma a confirmar a edição (**figura 23**). Caso a edição seja efetuada com sucesso, este é redirecionado para o menu principal, caso ocorra um erro na edição, uma mensagem de erro é apresentada.

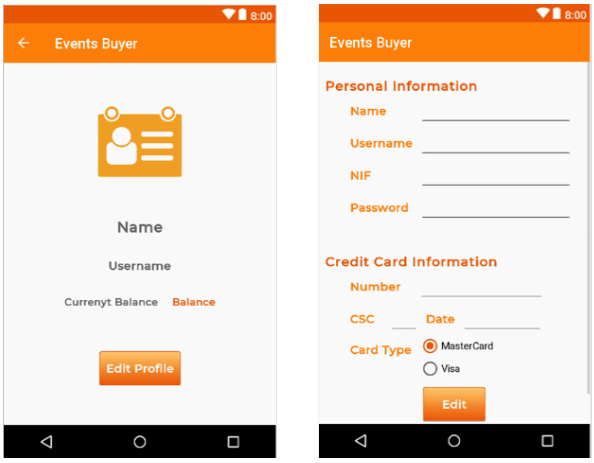

Figura 23: Sequência que permite visualizar o perfil do utilizador e editar as suas informações pessoais e as informações do seu cartão bancário

7.2.2 *Cafeteria*

A aplicação da ***Cafeteria*** tem como objetivo validar as ordens de *cafeteria*, respetivamente. Para tal, esta inicia com uma interface onde o utilizador pode

configurar o **IP** do servidor (**figura 24**).

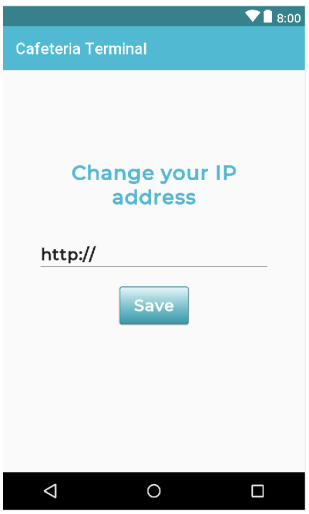

Figura 24: Alteração do **IP** da aplicação

Após a configuração do **IP**, este é redirecionado para a interface onde valida as respetivas ordens de *cafeteria*, através da leitura de ***QR codes*** (**figura 25**).

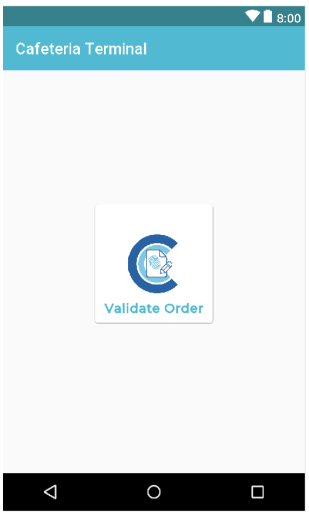

Figura 25: Validação de uma ordem de *cafeteria*

Após ser efetuada a leitura da respetiva ordem de *cafeteria*, o utilizador é redirecionado para uma interface onde este consegue obter o resultado da validação, sabendo se esta foi ou não efetuada com sucesso. Para além disso, caso a ordem tenha sido validada com sucesso, no terminal da *cafeteria* também é verificado o valor final da respetiva ordem, após a verificação dos *vouchers* utilizados (**figura 25**).

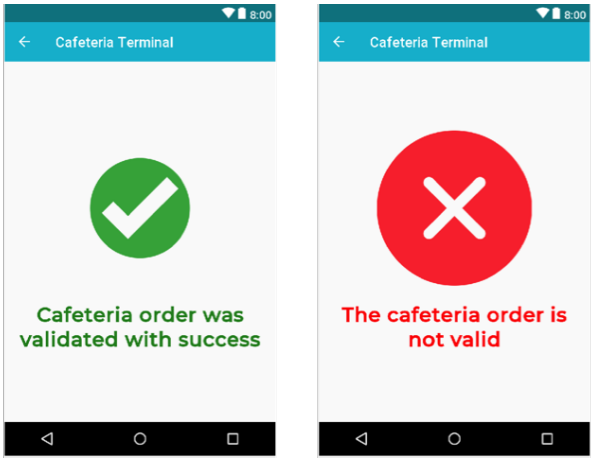

Figura 26: Resultado da validação de uma ordem de *cafeteria*

7.2.3 Terminal

A aplicação da **Terminal** tem como objetivo validar os bilhetes para um espetáculo, respetivamente. Para tal, esta inicia com uma interface onde o utilizador pode configurar o **IP** do servidor (**figura 27**).

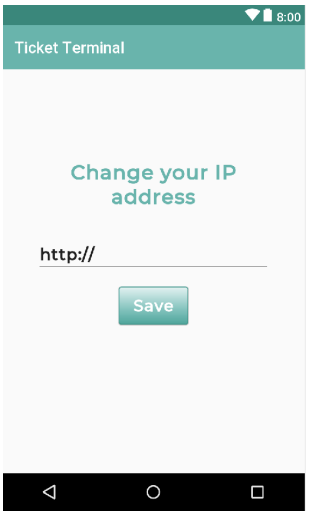

Figura 27: Alteração do **IP** da aplicação

Após a configuração do **IP**, este é redirecionado para a interface onde valida os respetivos bilhetes, através da leitura de ***QR codes*** (**figura 28**).

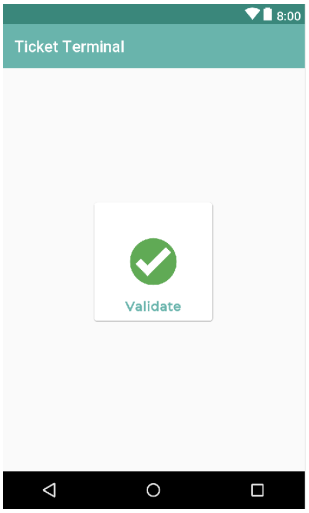

Figura 28: Validação de um bilhete

Após ser efetuada a leitura do respetivo bilhete, o utilizador é redirecionado para uma interface onde este consegue obter o resultado da validação, sabendo se esta foi ou não efetuada com sucesso (**figura 29**).

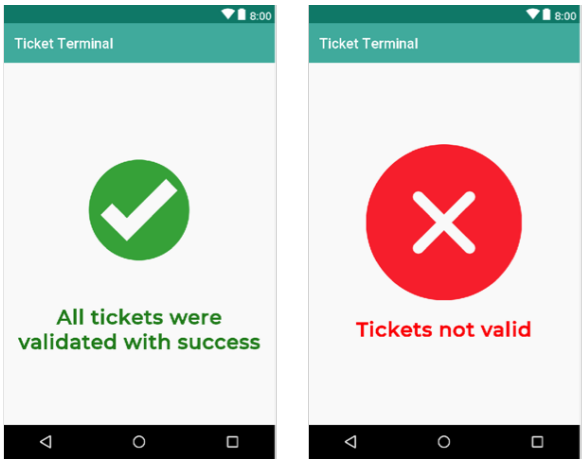

Figura 29: Resultado da validação de bilhetes

8 Conclusão

Com o desenvolvimento deste projeto, o grupo acredita ter conseguido adquirir novos conhecimentos e competências relativamente ao desenvolvimento de aplicações móveis, com sistema operativo **Android**, através da utilização do ***Android Studio***, nomeadamente em relação ao uso de ***QR codes*** e de ***Room Persistence Library*** (para guardar dados em memória local).

O sistema (aplicações móveis e servidor) está terminado, tendo sido desenvolvidos todos os parâmetros pedidos. Assim, com este sistema, o utilizador consegue registar-se e ligar-se a um servidor, conseguindo de seguida verificar os eventos futuros e os seus eventos, podendo comprar bilhetes para um evento pretendido. Ao comprar os bilhetes pode ganhar *vouchers* para serem descontados em pedidos de *cafeteria* que, igualmente, podem ser efetuados pelo utilizador, nesta aplicação. Por conseguinte, na aplicação da ***Cafeteria***, consegue-se verificar os pedidos efetuados, podendo-se validar os respetivos pedidos e os vouchers a estes associados. Por fim, na aplicação do **Terminal**, consegue-se validar os bilhetes comprados para os eventos.

Caso tivesse havido mais tempo, o grupo gostaria de ter melhorado o aspeto dos *layouts*, tornando-as ainda mais apelativos, responsivos e com maior usabilidade para o utilizador, pois neste momento alguns, ainda que poucos, dos *layouts* criados apenas dão para o modo *portrait*.

9 Recursos

9.1 Bibliografia

- Página Web da UC de Computação Móvel, mantida pelo professor António Miguel Pimenta Monteiro (acesso efetuado em outubro e novembro de 2018). **URL:** `https://web.fe.up.pt/~apm/CM/`

9.2 *Software* Utilizado

- Android Studio
- Intellij IDEA
- Room Persistence Library
- Docker - Enterprise Container Platform **URL:** `https://www.docker.com`
- Docker Compose **URL:** `https://docs.docker.com/compose/install/#prerequisites`
- Recycler View retirado de uma página Web **URL:** `https://www.androidhive.info/2016/01/android-working-with-recycler-view/`