Trabalho Prático #1

Android system for train tickets

Computação Móvel

Francisco Miguel Amaro Maciel - 201100692

Hugo Miguel Ribeiro de Sousa – 201100690

9 de Novembro de 2015

Índice

Introdução

Este projeto foi desenvolvido no âmbito da unidade curricular Computação Móvel do 5º ano do Mestrado Integrado de Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto. O principal objetivo deste trabalho é odesenvolvimento de duas aplicações para dispositivos Android para gerir um sistema de bilhética ferroviário, apoiado por um servidor externo. As principais funcionalidades de cada uma das aplicações são as seguintes:

Aplicação do utilizador:

- Registar conta, introduzindo dados pessoais incluindo dados do cartão de crédito
- Iniciar sessão (Facilitado ao utilizador, necessitando apenas da autenticação uma vez)
- Consulta de horários e comboios, através de local de partida, chegada e data
- Visualização em detalhe da viagem selecionada, incluindo viagem constituída por vários bilhetes e tempo de espera intermédio.
- Compra de bilhete digital
- Guardar bilhete digital no dispositivo, permitindo mostrar o bilhete digital (*QR code* ou *NFC*) ao inspetor sem necessitar de conexão

Aplicação do inspetor:

- Iniciar sessão
- Descarregar bilhetes para o percurso de um comboio
- Ler e validar bilhetes digitais dos utilizadores (*QR code* ou *NFC*), ao longo de uma viagem
- Fazer *upload* dos bilhetes validados para o servidor
- Ver estatísticas relacionadas com as validações de bilhetes efetuadas, incluindo tentativas de fraude

Optou-se por escolher a API 16 (*Android* 4.0.0) como sdk mínimo de suporte, visto suportar cerca de 96% dos dispositivos atuais e incluir a maioria das funcionalidades expectáveis das aplicações *Android* atuais (segundo dados atualizados em Novembro de 2015).

Para o servidor, optou-se por criar um conjunto de serviços REST, desenvolvidos em *Node.js*, suportando toda a estrutura de base de dados e operações relacionadas com contas, consulta, compra e validação de bilhetes.

Arquitetura do sistema

Ambas as aplicações *Android* comunicam com os serviços REST do servidor. Este servidor, por sua vez, para um dos pedidos (compra de bilhetes) comunica também com um outro servidor que simula um serviço externo com o intuito de validar a compra.

Estas dependem também fortemente do registo de dados localmente, visto grande parte das suas funcionalidades serem necessariamente executadas em modo offline. Para este registo local, recorreu-se ao armazenamento nas *SharedPreferences*, funcionalidade do *Android* que permite guardar informação numa estrutura de dados do tipo chave-valor. Por vezes, sendo necessário guardar tipos mais estruturados de informação, opta-se por guardar nestes valores objetos JSON que podem ser convertidos para *String*, e posteriormente, novamente para objeto JSON, facilitando a sua manipulação.

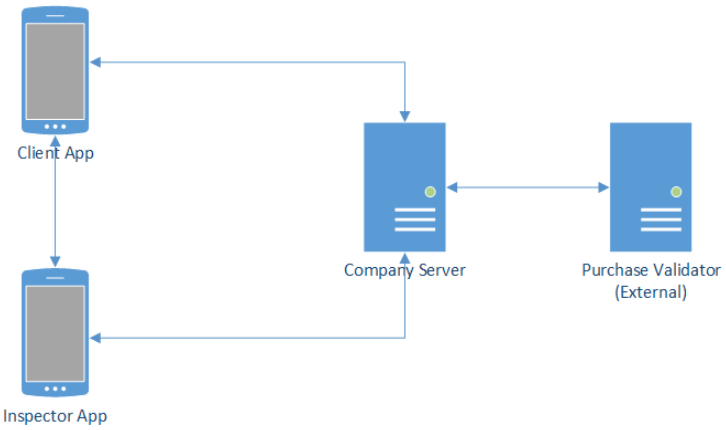


Fig. 1 - Arquitetura do projeto

Data Schema

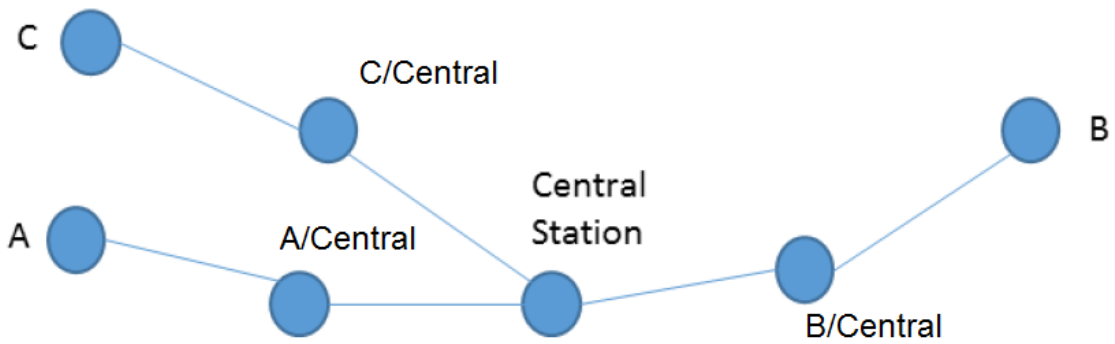


Fig. 2 - Rede ferroviária simulada

A identificação de uma dada *route* é feita pela sua estação de partida, estação de chegada e data/hora de partida. Na tabela *routes* encontram-se todas as combinações de partida-chegada possíveis na rede ferroviária. Para cada *route*, está associada um conjunto de *station_stop*, que indica a hora e o comboio que passa em cada estação do trajeto. Caso uma *route* necessite de trocar de comboio na estação central, é identificada como 2 routes alternativas (da partida à central e da central à chegada) e um *boolean* que indica a necessidade desta troca. No ato de compra, esta distinção é feita pela aquisição de 2 bilhetes, de forma facilitada ao utilizador.

A base de dados foi implementada em *MySQL 5.6*.

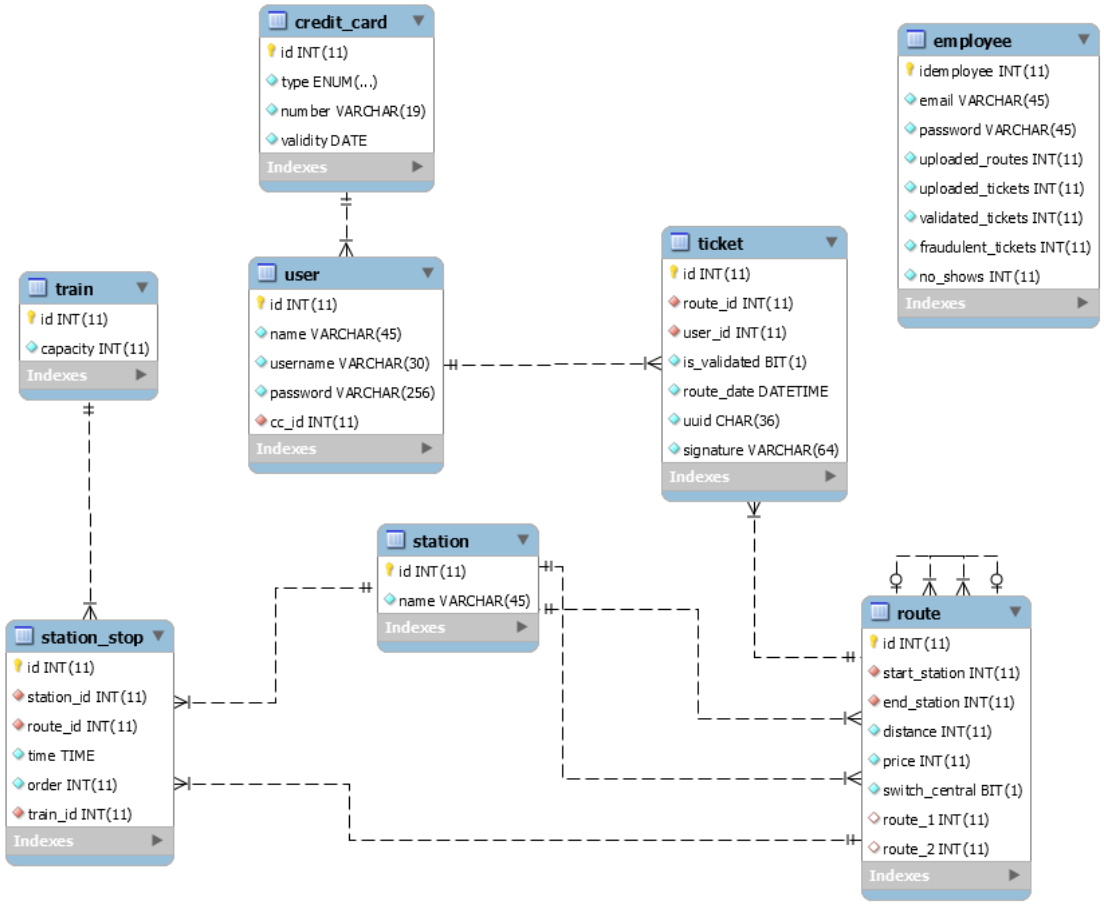


Fig. 3 - Modelo de dados

Tendo em conta que, de A para B e de C para Central partem 3 comboios por dia, foi definida a seguinte *timetable*, com horários e comboios válidos num cenário real:

Partida: C / Chegada: Central

C	C/Central	Central	Comboio
09:00	09:30	10:00	1
13:00	13:30	14:00	1
17:00	17:00	18:00	1

Partida: Central / Chegada: C

Central	C/Central	C	Comboio
11:00	11:30	12:00	1
15:00	15:30	16:00	1
19:00	19:30	20:00	1

Partida: A Chegada: B

A	A/Central	Central	B/Central	B	Comboio
09:00	10:00	10:45	11:45	12:30	2
14:00	15:00	15:45	16:45	17:30	3
18:30	19:30	20:15	21:15	22:00	2

Partida: B / Chegada: A

B	B/Central	Central	A/Central	A	Comboio
09:30	10:15	11:15	12:00	13:00	3
14:45	15:30	16:30	17:15	18:15	2
19:00	19:45	20:45	21:30	22:30	3

Servidor

O servidor consiste num conjunto de serviços REST. O corpo da mensagem dos serviços que assim o requerem e a resposta são formatados em JSON. Alguns destes serviços requerem autenticação, sendo necessário fazer login previamente, e enviar o *token* recebido no ato de login no *header* dos pedidos HTTP consequentes. Por exemplo, a compra de um bilhete requer que o utilizador esteja autenticado. Pretendia-se que os pedidos fossem o mais genéricos possível, de forma a serem facilmente adaptados caso a rede ferroviária fosse alterada. Alguns destes serviços podem ser considerados desnecessários, assumindo uma rede ferroviária fixa, permitindo às aplicações móveis a manutenção da sua própria informação sobre as linhas. No entanto, seguindo as boas práticas de escalabilidade e coerência, os dados estão unificados no servidor e são enviados para o cliente, sendo transparente para a aplicação a estrutura complexa da linha. Os *endpoints* disponibilizados são os seguintes:

Tipo	Route	Parâmetros	Descrição
AUTH			
POST	/register	name, username, password, creditcard_type, creditcard_number, creditcard_validity	Regista um novo utilizador. Retorna *token JWT* (one-step register/login)
POST	/login	username, password	Valida *login* de um utilizador. Retorna *token JWT*.
POST	/loginemployee	email, password	Valida *login* de um inspetor, Retorna *token* para funcionário *JWT*.
ROUTING			
GET	/stations		Retorna todas as estações.
GET	/routes	from, to, date	Retorna as viagens identificadas pelos parâmetros. A data serve para identificar se essa viagem já se encontra esgotada (pode ser passado a *null*).
GET	/route	from, to, date, time	Retorna a viagem identificada pelos parâmetros.
GET	/simpletrains		Retorna as 4 viagens principais efetuadas, horários e especificações, para as quais o inspetor pode fazer download dos bilhetes para os clientes a bordo.
TICKETING			
GET	/tickets		[Requer autenticação de utilizador] Retorna os bilhetes não validados de um utilizador.
GET	/downloadtickets	from, to, date, time	[Requer autenticação de inspetor] Retorna os bilhetes da viagem identificada pelos parâmetros.
POST	/tickets/purchase	from, to, date, time	[Requer autenticação de utilizador] Compra bilhete(s) da viagem identificada pelos parâmetros.
POST	/tickets/upload	tickets	[Requer autenticação de inspetor] Faz *upload* dos bilhetes validados para o servidor.
STATISTICS			
GET	/statistics		[Requer autenticação de inspetor] Retorna as estatísticas associadas ao inspetor autenticado.
POST	/statistics/upload	uploaded_routes, uploaded_tickets, validated_tickets, fraudulent_tickets, no_shows	[Requer autenticação de inspetor] Atualiza as estatísticas associadas ao inspetor autenticado.

Aplicação do utilizador

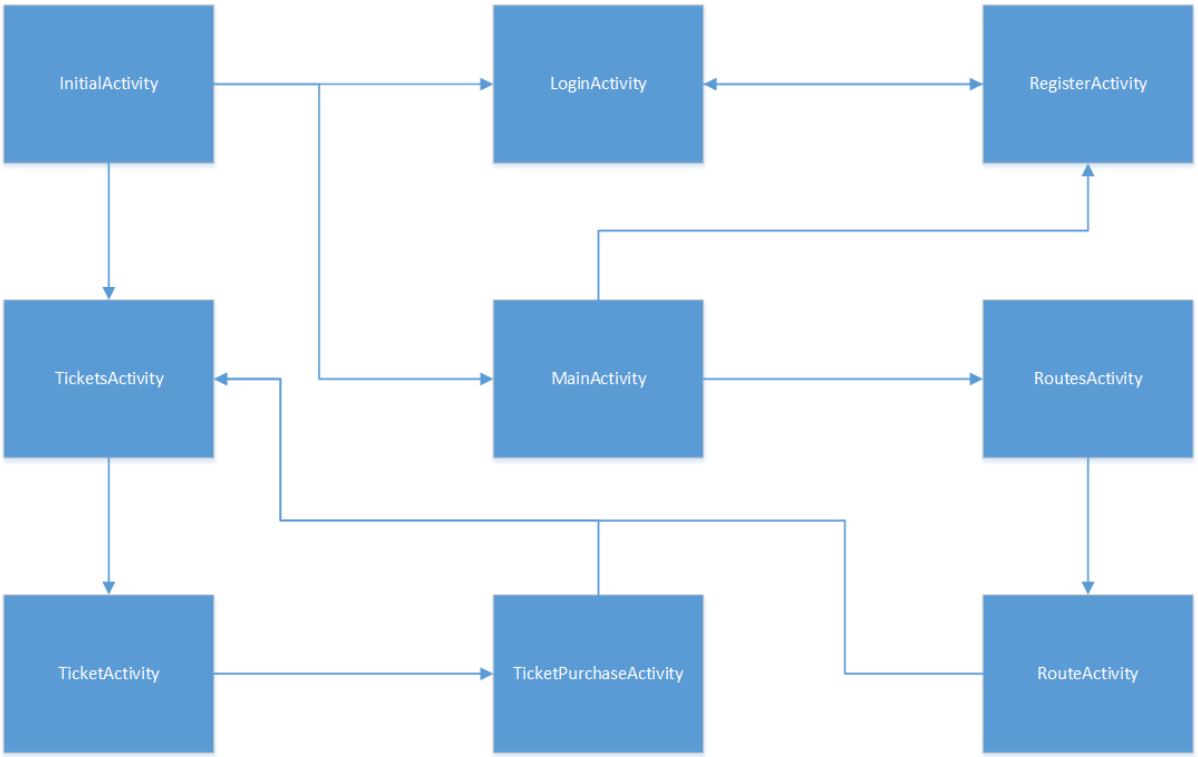


Fig. 4 - Workflow da aplicação do utilizador

Como as aplicações recorrem fortemente ao funcionamento em modo offline e armazenamento local, em algumas *activities* recorre-se à implementação de *BroadcastReceiver* para identificar mudanças na rede e perceber se o utilizador está conetado à internet. Na *InitialActivity*, se o utilizador estiver em modo offline, apenas lhe permite ver os seus bilhetes armazenados localmente, caso esteja devidamente *logged in*. Caso esteja em modo online, tem acesso às restantes funcionalidades da aplicação. Outro caso de uso deste *BroadcastReceiver* nesta aplicação é na *activity MainActivity*. Caso esteja em modo offline, apenas serão mostrados os bilhetes armazenados localmente. Se o utilizador se reconectar à internet, passam a ser mostrados todos os seus bilhetes não usados.

Um utilizador não autenticado pode utilizar a aplicação para pesquisa de todos os horários em todas as rotas, podendo escolher e analisar todas as viagens em detalhe. A autenticação é apenas requerida no ato de compra, permitindo o acesso aos bilhetes comprados.

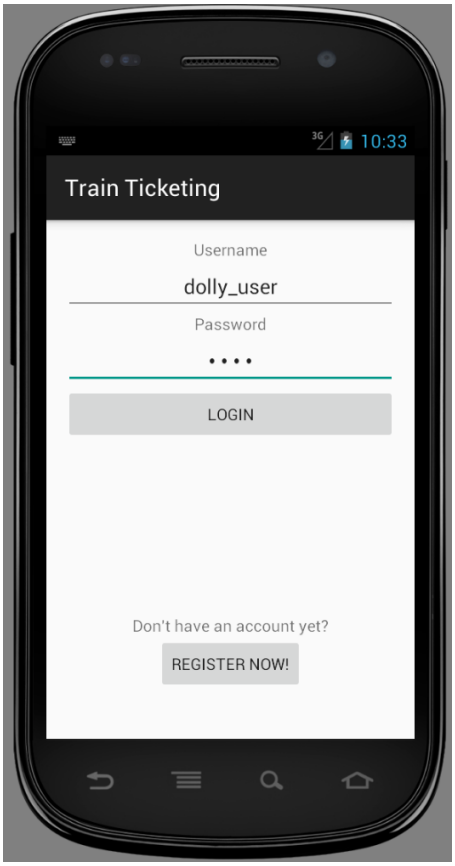
Funcionalidades e testes realizados

Todas as *features* requeridas no enunciado do projeto foram concluídas com sucesso.

Feature [Activity]	**Teste realizado**	**Resultado**
Aceder a *features* offline. [InitialActivity]	Entrar na aplicação sem estar conetado à internet, estando previamente *logged in*.	O utilizador pode ver os bilhetes armazenados localmente.
Aceder a *features* offline. [InitialActivity]	Entrar na aplicação sem estar conetado à internet, não estando previamente *logged in*.	O utilizador não pode ver nada. Na atividade inicial, é informado que se deve conectar à internet para poder efetuar o *login*.
Registar conta. [RegisterActivity]	Registar conta sem preencher todos os campos	A aplicação indica os campos que faltam ser preenchidos.
Registar conta. [RegisterActivity]	Registar conta preenchendo campos com informação inválida (número de cartão de crédito inválido ou confirmação de password diferente)	Os campos em causa são *highlighted* e é mostrada informação do erro.
Registar conta. [RegisterActivity]	Registar conta preenchendo todos os campos com informação válida.	O utilizador é criado com sucesso e redirecionado para a página de *Login*.
Login. [LoginActivity]	Fazer login sem preencher campos ou utilizador inválido.	É mostrado um *Toast* com a informação do erro.
Login. [LoginActivity]	Fazer login preenchendo campos com informação de utilizador válido.	O utilizador é redirecionado para a MainActivity.
Consultar horários. [MainActivity]	Pesquisar horários sem inserir data.	O utilizador é informado que deve inserir uma data.
Consultar horários. [MainActivity]	Pesquisar horários, inserindo data e locais de partida e chegada pretendidos.	O utilizador é redirecionado para a rota e data definida.
Consultar horários. [MainActivity]	Na rota e data definidas, verificar que são identificadas viagens com lotação esgotada.	Diminuindo a capacidade de um comboio na base de dados, e comprando bilhete para um dos troços dessa rota, verificar que essa viagem surge como esgotada. Estas viagens não permitem prosseguir para a compra de bilhete.
Consultar horários. [MainActivity]	Visualização correta das estações, hora e tempo de espera de uma viagem.	Para cada viagem listada, mostra corretamente as estações em que o comboio passa. Em caso de ter de trocar na estação central, informa também o tempo de

		espera na estação central. É possível prosseguir para a compra de bilhete.
Compra de bilhete. [TicketPurchaseActivity]	Compra de bilhete repetido com o mesmo utilizador.	O utilizador é informado que já comprou um bilhete para essa viagem.
Compra de bilhete. [TicketPurchaseActivity]	Compra de bilhete de viagem sem troca na estação central.	O utilizador é redirecionado para a atividade dos bilhetes, com o novo bilhete na lista.
Compra de bilhete. [TicketPurchaseActivity]	Compra de bilhete de viagem com troca na estação central.	O utilizador é redirecionado para a atividade dos bilhetes, com os novos bilhetes na lista.
Listagem de bilhetes. [TicketsActivity]	Listar bilhetes sem estar conectado à internet.	Apenas os bilhetes guardados localmente de data igual ou superior ao dia atual são mostrados.
Listagem de bilhetes. [TicketsActivity]	Listar bilhetes conectado à internet.	Todos os bilhetes de data igual ou superior ao dia atual são mostrados.
Armazenar/remover bilhete localmente. [TicketsActivity]	Clicar no botão para armazenar/remover um bilhete ou todos os bilhetes localmente.	Os bilhetes mudam o seu símbolo, que representa o seu armazenamento local. Caso o utilizador não esteja ligado à internet, é informado que se remover o bilhete, não terá mais acesso a este enquanto não se reconectar à internet.
Visualização de um bilhete. [TicketActivity]	Clicar num dos bilhetes listados.	É apresentada mais informação do bilhete e o seu *QR code*.
Validação de bilhete por *NFC*. [TicketActivity]	Clicar em *Send NFC*, aproximando o *smartphone* ao do inspetor.	O bilhete é validado/recusado na aplicação do inspetor.

Atividades e interações



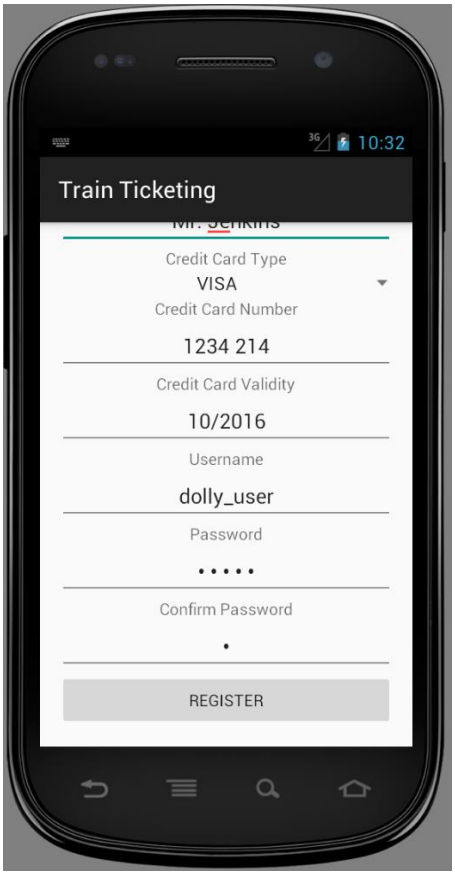
Login activity

Descrição:

Este ecrã potencia a utilização da aplicação, fornecendo ao utilizador novas funcionalidades. Esta operação é apenas necessária uma vez, uma vez que o *token* fica registado localmente para maior comodidade

Funcionalidades:

- Escrever o *username*
- Escrever a *password*
- Clicar em *Login* (redirecionado para *MainActivity* ou mostrada mensagem de erro)
- Clicar em *Register*
- Voltar para *MainActivity* (*back button*)



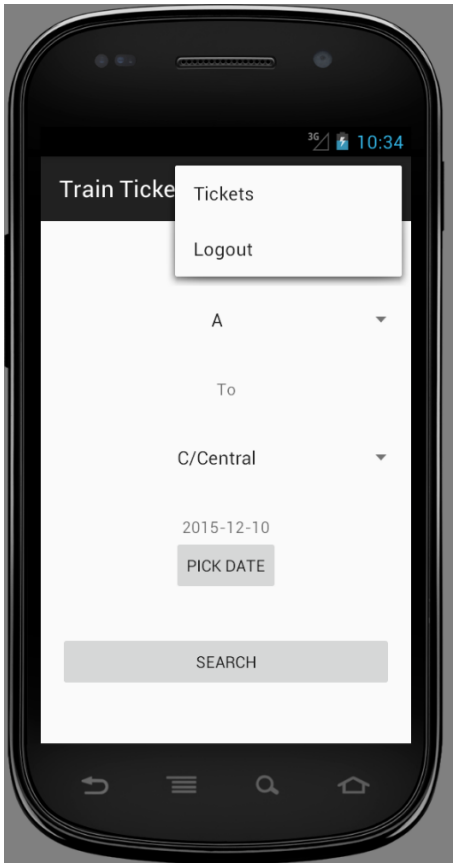
Register activity

Descrição:

Este ecrã permite ao utilizador o registo na base de dados, autenticando-se para poder aceder a outras funcionalidades da aplicação.

Funcionalidades:

- Escrever *name*
- Escolher *credit card type*
- Escrever *credit card number*
- Escolher *credit card validity*
- Escrever *username*
- Escrever *password* e sua confirmação
- Clicar em *Register* (redirecionado para *LoginActivity* ou mostrada mensagem de erro e indicados os campos errados ou em falta)
- Voltar para *LoginActivity* (*back button*)



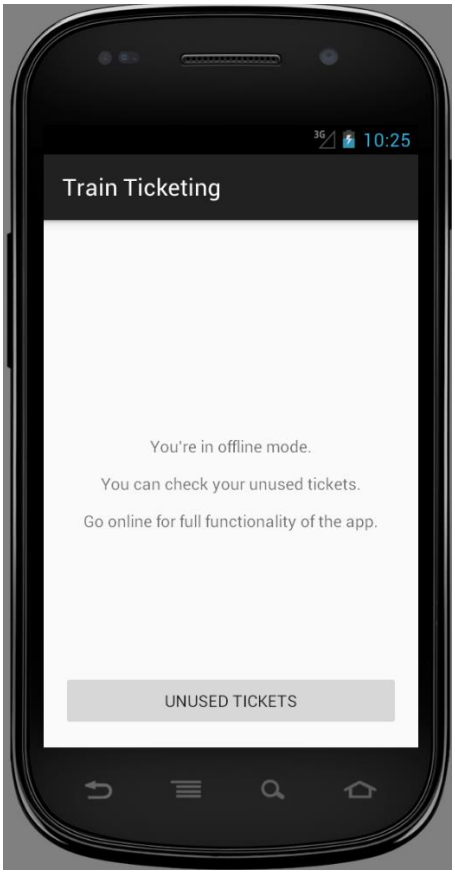
Main activity

Descrição:

Esta atividade permite ao utilizado pesquisar viagens, através da estação de partida, chegada e a data. A partir daí, será redirecionado para as várias viagens disponíveis nessa data.

Funcionalidades:

- Verificar os seus bilhetes
- *Logout*
- Selecionar estação de partida
- Selecionar estação de destino
- Selecionar data
- Clicar *Search*
- Sair da aplicação (*back button*)



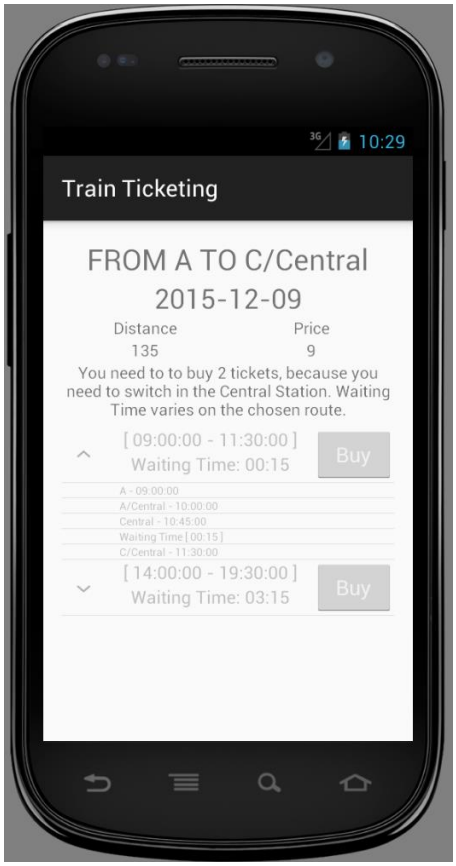
Initial activity (*offline* but *logged in*)

Descrição:

Este ecrã é apresentado imediatamente antes da *MainActivity* e serve para testar a conexão do utilizador à internet e ao servidor. Caso o utilizador se encontre em modo offline, poderá ver os seus bilhetes armazenados localmente, caso se encontre já *logged in* na aplicação, permitindo validar o seu bilhete durante uma viagem. Caso o utilizador se reconecte à internet, é automaticamente redirecionado para a *MainActivity*.

Funcionalidades:

- Ver os seus bilhetes (armazenados localmente)
- Sair da aplicação (*back button*)



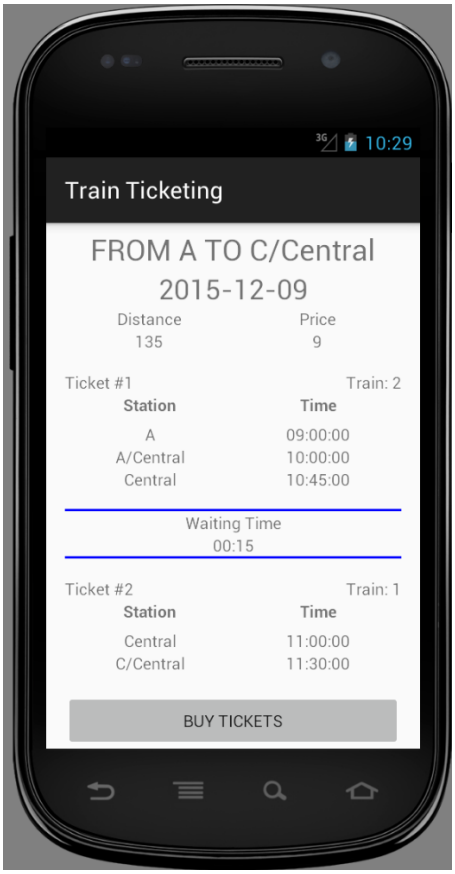
Routes activity

Descrição:

Nesta actividade é possível visualizar todas as viagens disponíveis para a data selecionada. Para cada viagem, é possível ainda ver a que horas passa em cada estação e, caso seja uma viagem que necessite de trocar de comboio na estação central, indica também o tempo de espera nesta. Indica também se a viagem se encontra com lotação esgotada, não permitindo compra de mais bilhetes.

Funcionalidades:

- Verificar distância e preço da viagem
- Verificar estações de partida e chegada da viagem
- Para cada horário disponível, verificar tempo de passagem em cada estação
- Clicar em *Buy*, escolhendo o horário pretendido
- Voltar para *MainActivity* (*back button*)



Ticket Purchase Activity

Descrição:

Este ecrã mostra todos os detalhes da viagem, já previamente mostrados, mas de forma mais organizada, incluindo também o identificador do comboio para cada viagem. Faz a divisão dos bilhetes necessários, caso seja necessário trocar na estação central.

Funcionalidades:

- Verificar todas as paragens e horas da viagem
- Verificar tempo de espera na estação central, se for o caso
- Verificar comboio(s) da viagem
- Verificar distância da viagem
- Verificar preço da viagem
- Clicar em *Buy Ticket(s)* para comprar o(s) bilhete(s)
- Voltar para *RoutesActivity* (*back button*)

Tickets Activity

Descrição:

Esta atividade mostra os bilhetes do utilizador que não tenham sido validados. No caso de não terem sido validados, mas a data ser passada (do dia anterior), também não são mostrados, considerando-se que o utilizador não validou o bilhete. Bilhetes que estejam guardados localmente também são apagados caso não sejam validados e se apresentem de uma data anterior à presente. O utilizador tem possibilidade de adicionar e remover livremente os bilhetes armazenados localmente. A actividade responde automaticamente às alterações de rede, mostrando apenas os bilhetes armazenados localmente caso o utilizador se desconecte da internet.

Funcionalidades:

- Ver todos os bilhetes (*online*)
- Clicar no botão "+" para descarregar todos os bilhetes localmente
- Clicar no botão "-" para remover todos os bilhetes localmente
- Carregar no botão "+" ou "-" individualmente num bilhete para o

Ticket Activity

Descrição:

Aqui o utilizador deve validar o seu bilhete com o inspetor, de uma das duas formas:

1. por *QR code*, permitindo ao inspetor ler o seu código e validar o bilhete.
2. por *NFC*, caso o seu *smartphone* possua esta funcionalidade, enviando os dados do bilhete para validação para a aplicação do inspetor.

Funcionalidades:

- Verificar detalhes da viagem
- Ver o *QR code* correspondente ao bilhete
- Clicar Send NFC para enviar informação do bilhete por NFC
- Voltar para *TicketsActivity* (*back button*)

adicionar ou remover localmente, respetivamente

- Clicar numa viagem para ver o bilhete em detalhe e o validar com o inspetor
- Voltar para *MainActivity* (*back button*)

Aplicação do inspetor

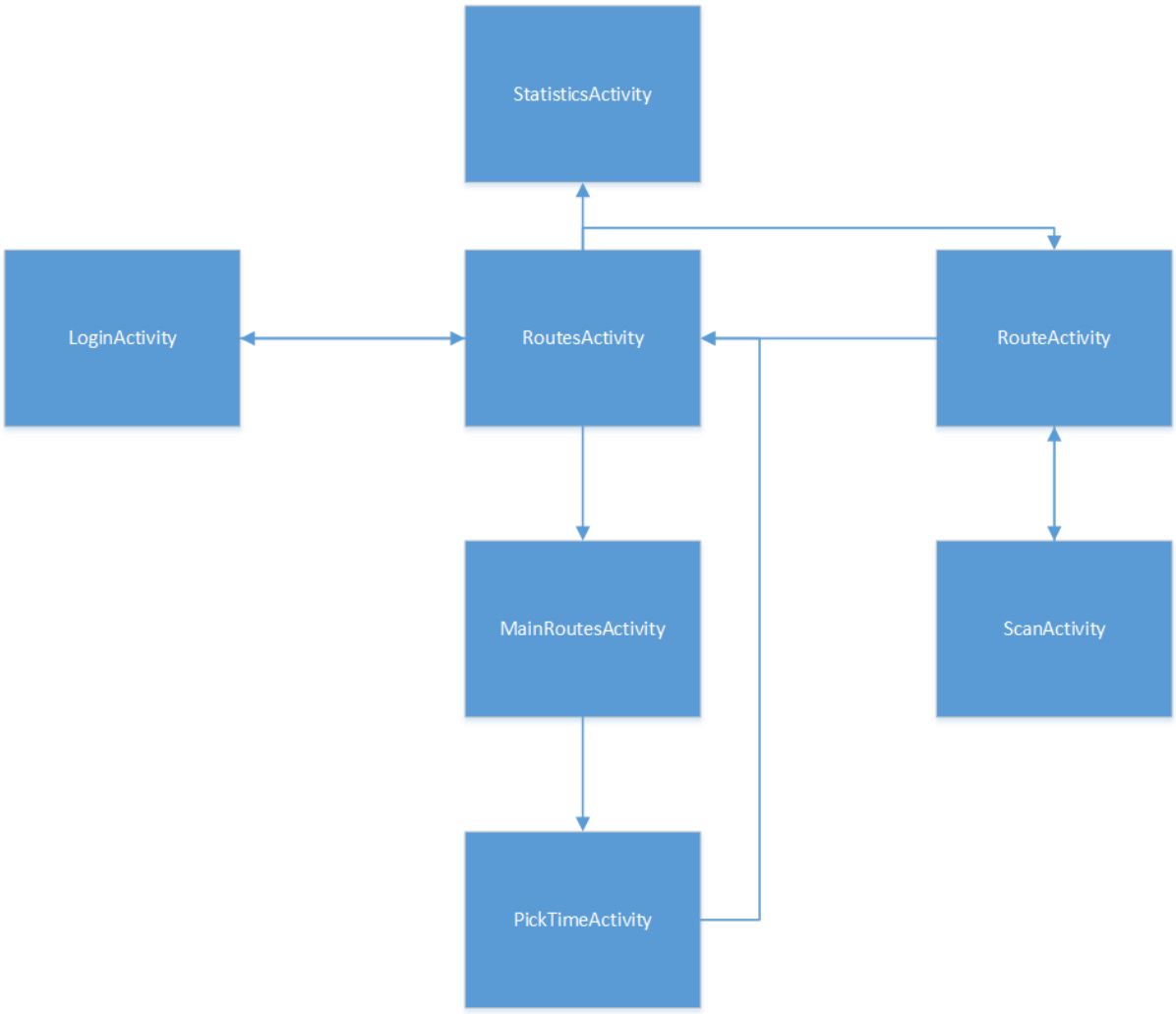


Fig. 5 - Workflow da aplicação do inspetor

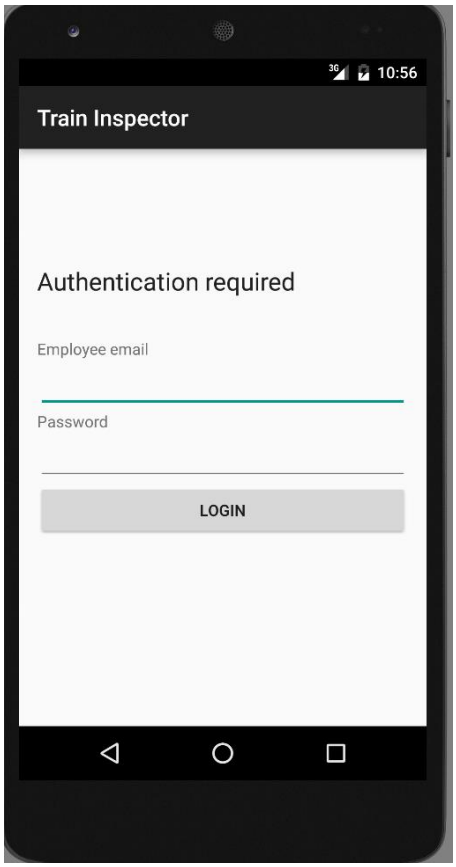
Funcionalidades e testes realizados

Todas as *features* requeridas no enunciado do projeto foram concluídas com sucesso.

Feature [Activity]	**Teste realizado**	**Resultado**
Login [LoginActivity]	Entrar na aplicação já estando *logged in*.	O inspetor é automaticamente redirecionado para a listagem das viagens que o inspetor fez download.
Login [LoginActivity]	Fazer *login* sem preencher campos ou utilizador inválido.	É mostrado um *Toast* com a informação do erro.
Login [LoginActivity]	Fazer *login* preenchendo campos com informação de utilizador válido.	O utilizador é redirecionado para a listagem das viagens que fez download.

Listagem das rotas descarregadas. [RoutesActivity]	O inspetor desconeta-se da internet.	O botão que permite descarregar novas rotas e as estatísticas estão *disabled*. As rotas permanecem listadas, pois encontram-se armazenadas localmente.
Ver rota descarregada. [RouteActivity]	Clicar numa das rotas descarregadas, estando desconectado da internet.	É apresentada mais informação da rota descarregada. O botão de *upload* encontra-se *disabled*, enquanto o inspetor não está conetado à internet.
Validar bilhete por *QR code*. [RouteActivity + ScanActivity]	Clicar em *Scan* e ler *QR code*.	Caso o bilhete seja válido, surge uma mensagem de sucesso e indica entre que estações o bilhete é válido. Caso seja válido, mas já tenha sido validado, surge essa informação ao inspetor. Caso a informação do código tenha sido adulterada, surge uma mensagem de erro. A contagem de bilhetes validados é atualizada em casa de sucesso.
Validar bilhete por NFC. [RouteActivity]	Aproximar *smartphone* do utilizador, enquanto mantém aberta a rota em que se encontra. O utilizador clica em *Send NFC*.	A informação apresentada é a mesma que a validação por *QR code*.
Upload de rota. [RouteActivity]	Clicar em *Upload*.	A rota deixa de estar listada nas rotas descarregadas. As estatísticas são atualizadas. Redireciona para a listagem de rotas descarregadas.
Ver estatísticas. [StatisticsActivity]	Na atividade de listagem de rotas descarregadas, clicar em *Statistics* no menu (tem de estar conetado à internet).	Surgem as estatísticas relativamente a rotas e bilhetes validados, para o inspetor autenticado.

Atividades e interações



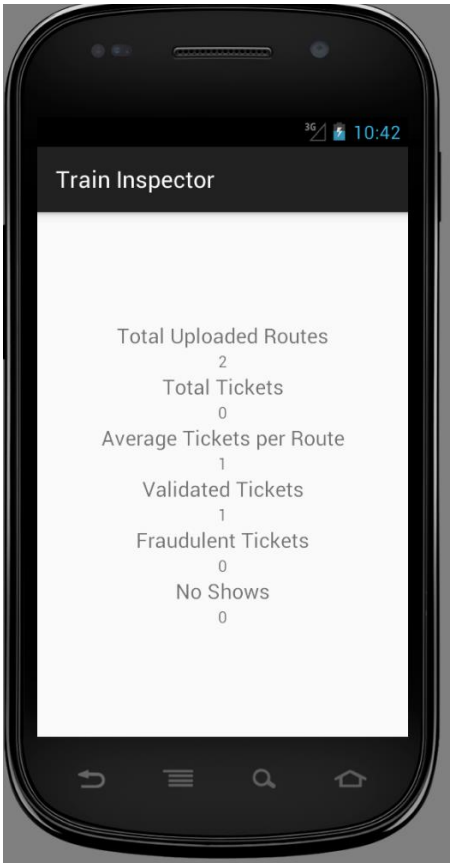
Login activity

Descrição:

Este passo é estritamente necessário para utilizar a aplicação, pois sem a autenticação confirmada de funcionário não é possível aceder às restantes funcionalidades. Esta operação é apenas necessária uma vez, uma vez que o *token* fica registado localmente para maior comodidade.
É também recebida neste momento a chave pública a partir do servidor.

Funcionalidades:

- Introduzir *email*
- Introduzir *password*
- Clicar no botão *Login* e avançar para *MainActivity*
- Sair da aplicação (*back button*)



Statistics Activity

Descrição:

Esta atividade permite a qualquer funcionário receber informações do servidor sobre os bilhetes validados por si. Esta informação é atualizada quando é feito o *upload*, no final de cada viagem.

Funcionalidades:

- Ver total de viagens enviadas para o servidor
- Ver total de bilhetes deste dispositivo
- Ver média de bilhetes por rota
- Voltar para *MainActivity* (*back button*)

Main Routes Activity

Descrição:

Este passo é necessário para selecionar qual das quatro viagens simples o inspetor vai apanhar, viajando toda a viagem do comboio. Estes botões são adicionados dinamicamente a partir das informações enviadas pelo servidor sobre cada viagem.

Funcionalidades:

- Selecionar viagem a descarregar
- Voltar para *MainActivity* (*back button*)

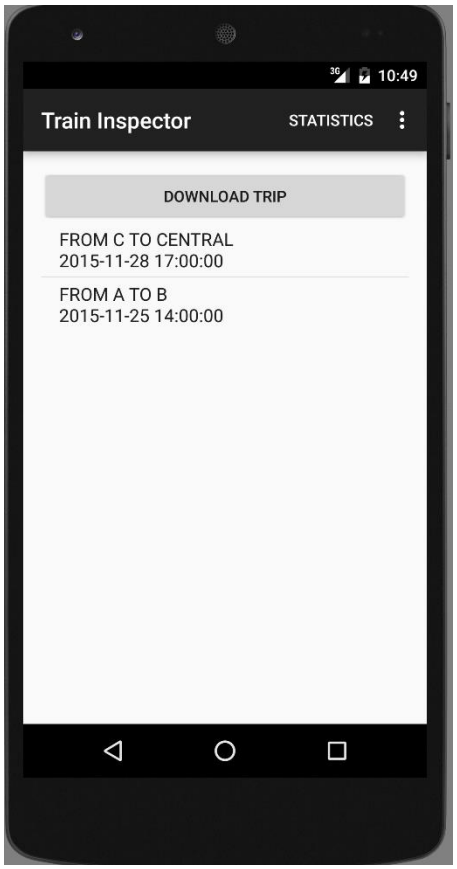
Pick Time Activity

Descrição:

Esta atividade permite escolhe, para a viagem selecionada, qual dos três horários (incluindo o número do comboio) vai querer descarregar. Não é feito novo pedido para o servidor, uma vez que esta informação já é recebida no ecrã anterior.

Funcionalidades:

- Selecionar horário através dos *radio buttons*
- Selecionar data da viagem
- Voltar para *MainRoutesActivity* (*back button*)



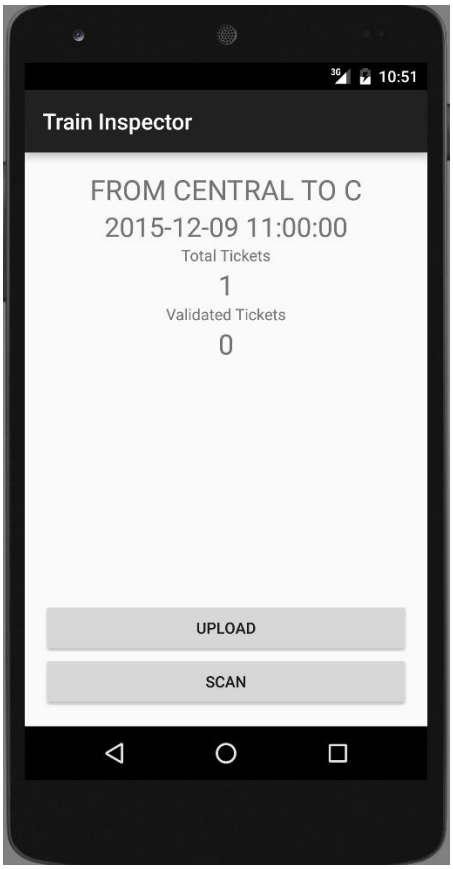
Main Activity

Descrição:

Este é o ecrã principal, onde o inspetor pode ver as viagens para as quais descarregou informação para poder validar os bilhetes. A partir deste ecrã. ele pode descarregar novas viagens ou preparar o download de uma em particular. Esta informação está guardada localmente, e não é atualizada a partir do servidor.

Funcionalidades:

- Utilizar o botão *Download Trip* para navegar para Main Routes Activity
- Selecionar uma viagem a partir da list-view para realizar o scan



Trip Activity

Descrição:

Este ecrã é aquele em que o inspetor se situa enquanto circula pelo comboio, abordando todos os passageiros. A partir deste ecrã. o inspetor pode encostar o dispositivo a um cliente que transmita o bilhete por nfc, e irá obter a validação. É também neste ecrã que são contabilizados os bilhetes restantes. A partir daqui, o inspetor pode abrir a câmara para ler os códigos *QR*, ou fazer o upload das viagens quando concluir todo o processo.

Funcionalidades:

- Ligar a câmara para ler *QR code*
- Fazer o upload de bilhetes
- Ver resultado da receção do bilhete

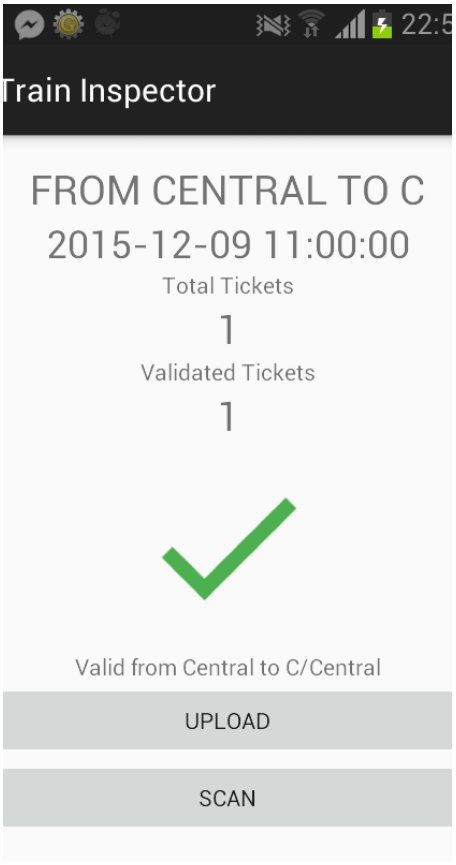
Scan Activity

Descrição:

Esta atividade utiliza a biblioteca https://github.com/dm77/barcodescanner, uma abstração à mais conhecida biblioteca zxing, para gerar uma vista que inclui a câmara do dispositivo, e uma interface que indica a área de leitura. Após reconhecer corretamente a imagem, é feita a validação do bilhete assinado pelo servidor, utilizando *Sha1withRSA,* a partir do suporte da linguagem Java.

Funcionalidades:

- Clicar no botão *Download Trip* para navegar para *MainRoutesActivity*
- Selecionar uma viagem a partir da list-view para realizar o *scan*



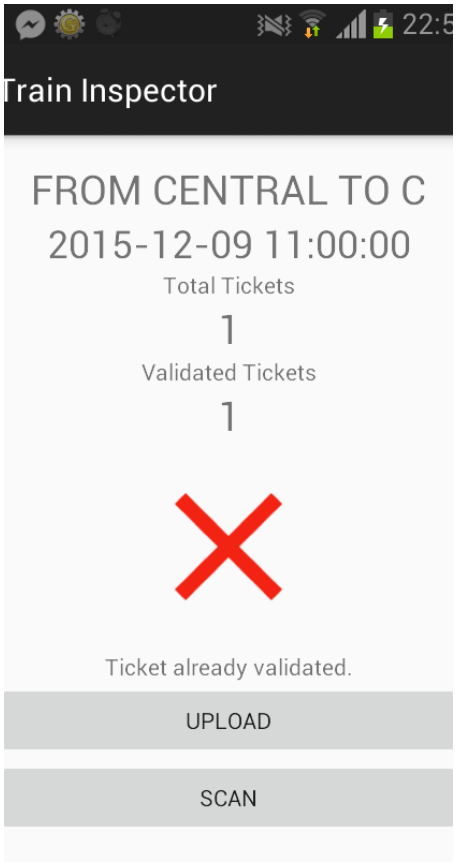
Trip Activity (success)

Descrição

Esta vista é atualizada, tendo em conta o resultado da vista *Scan* ou da receção de uma mensagem NFC. No caso de sucesso, o bilhete é validado, e é apresentado ao inspetor a parte do percurso para o qual é válido. Um bilhete fica assim registado, e se for visto novamente é assinalado como repetição.

Funcionalidades:

- Validar bilhete



Trip Activity (erro)

Descrição:

No caso de erro na validação do código, é apresentada uma mensagem indicando o motivo da falha. A leitura de um código pode falhar no caso de um bilhete ser lido repetidamente, não estar na lista de bilhetes para uma viagem, ou conter dados errados / fraudulentos. Este último caso é assinalado e tido em conta nas estatísticas.

Funcionalidades:

- Validar bilhete

Segurança

Prevenção de falsificação de bilhetes

Os bilhetes são assinados do lado do servidor, aquando da sua compra, tendo em conta todos os dados que o identificam, juntamente com um identificador único de 16 *bytes*. Nos *QR codes* gerados, é apenas enviado este identificador e a respetiva assinatura gerada através do algoritmo "*SHA1WithRSA*". O inspetor tem na aplicação também os restantes dados do bilhete localizando-os através do identificador único. Desta forma, a aplicação de validação utiliza a chave pública do servidor para confirmar a validade do bilhete, utilizando os dados que o identificam de forma única, a chave e a assinatura. Assim sendo, um intruso apenas tem acesso a 2 dados através do *QR code*, o identificador e a assinatura.

Modificando o identificador, não vai ser possível encontrar o respetivo bilhete e gerará um erro. Num caso extremo, poderia mudar para um identificador existente, mas a assinatura não corresponderia. Alterando a assinatura também fará com que não seja feito o match, invalidando o bilhete. Mesmo tendo acesso a todos os dados identificadores do bilhete, o utilizador teria de saber o formato exato dos dados sobre o qual foi gerada a assinatura, e chave privada do servidor.

Prevenção de cópias de bilhetes

Só é permitido, para um utilizador, a compra de um bilhete para a mesma viagem/data. Na leitura dos códigos, previne-se a validação de 2 bilhetes iguais, avisando o utilizador que esse bilhete já foi previamente validado. Nesta situação pode-se verificar uma pequena falha de segurança. Caso um intruso detete os dados do *QR code*, poderá gerar um novo *QR code* com os mesmos dados, tendo um código válido (tirar uma foto à aplicação de outro utilizador e apresentar ao inspetor). O bilhete que for apresentado primeiro será validado, enquanto que os seguintes serão apresentados como repetidos. Isto apenas é possível no caso extremo de ser roubado o telemóvel. De qualquer das formas, acaba por ser detetado a fraude, podendo causar distúrbios ao cliente que foi "roubado".

Conclusão

Este projeto pareceu-nos bastante interessante e desafiante. O esquema de dados para um modelo de negócio deste tipo, mesmo sendo em pequena escala, proporcionou um trabalho bastante desafiante e complexo.

A grande diversidade de caraterísticas e tamanhos dos diversos dispositivos adiciona complexidade no que toca ao desenvolvimento para sistema *Android*. Pode ser relevante definir diferentes *layouts* ou ações para diferentes dispositivos pelo que o trabalho adicional para manter uma aplicação o mais abrangente possível pode ser elevado, sendo importante o seu teste no maior tipo de dispositivos possível, e não só no emulador. Neste projeto, sendo o tempo de desenvolvimento relativamente reduzido, não foi feita esta adaptação a grande detalhe. No entanto, foi tido em conta um *design* responsivo, não usando medidas fixas, estando adaptado a qualquer tamanho de dispositivo.