Mobile Computing - EIC0050

Assignment #1 - Train Ticket System - **DRail**

Professor António Miguel Pontes Pimenta Monteiro

Ruben Fernando Pinto Cordeiro - ei11097
Duarte Nuno Pereira Duarte - ei11101

Context

The proposed project consists in the implementation of a railway management platform. The distributed system is composed by two distinct mobile Android applications: one for ticket purchasers and another for ticket inspectors.

The Android application for ticket purchasers allows end users to consult information about daily timetables and ticket prices as well as buying tickets.

The Android application for ticket inspectors allows the end user to validate the emitted tickets for a specific trip.

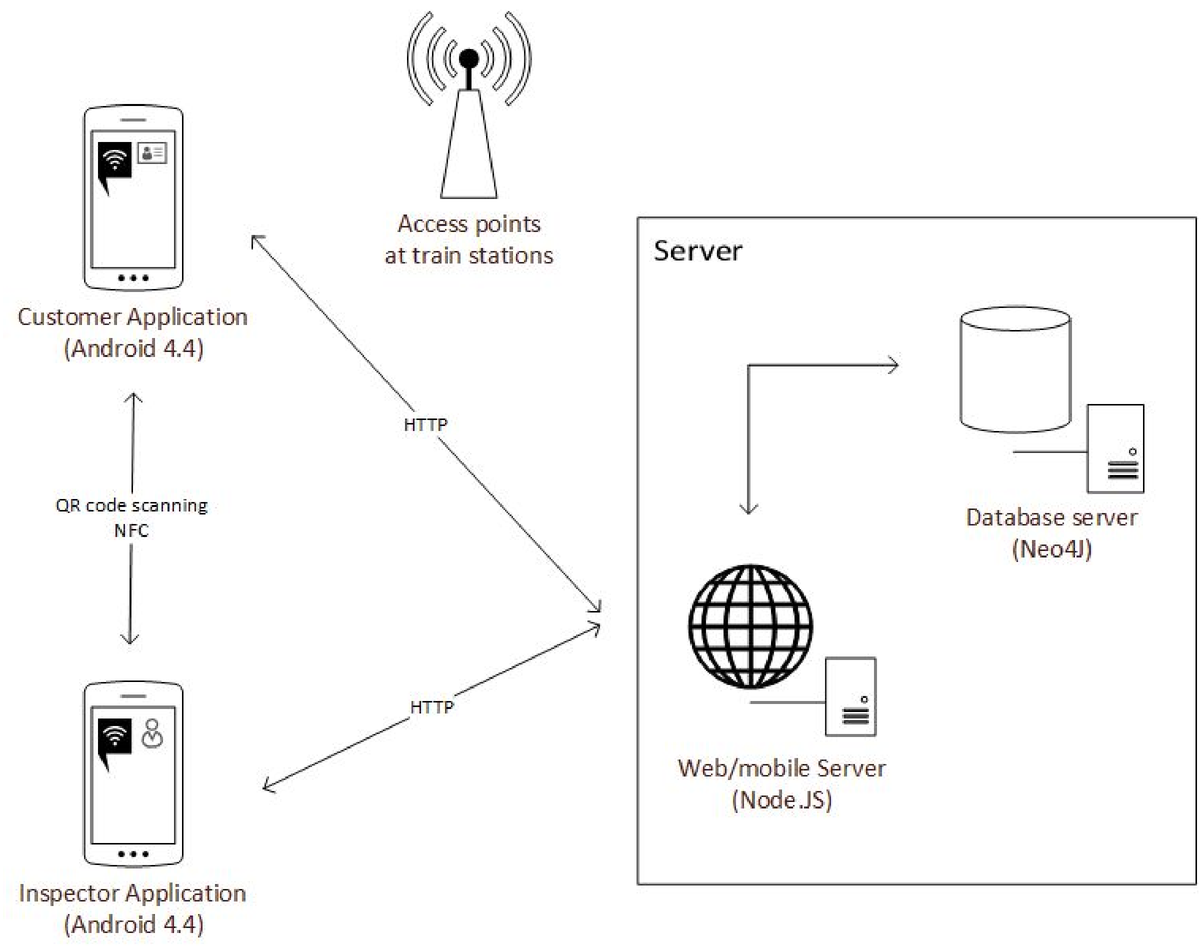
Architecture

The railway management platform follows client-server architecture composed by three layers: data, business logic and presentation layer.

The data layer consists of a Neo4j database instance. Neo4j is an open-source NoSQL graph database.

The business logic layer consists of a Node.js server. It exposes a RESTful API and communicates with the Neo4j instance in order to retrieve and update all the information related to the railway platform.

The presentation layer is represented by the two mobile Android applications.



Data storage

Why Neo4j

The railway topology matches the graph model perfectly: each station is a node and the trips between stations are represented as edges. Each edge holds specific properties: id of the train, departure date, arrival date and distance. Users are also nodes in the graph.

Relationships are first-class citizens of the graph data model, unlike other database management systems, which require the inferral of connections between entities using special properties such as foreign keys, or out-of-band processing like map-reduce. A relational database would most certainly require various self referential joins to compute a route between stations separated by a number of hops. By assembling the simple abstractions of nodes and relationships into connected structures, a graph database enables the creation of a sophisticated model that maps closely to our problem domain.

Data “schema”

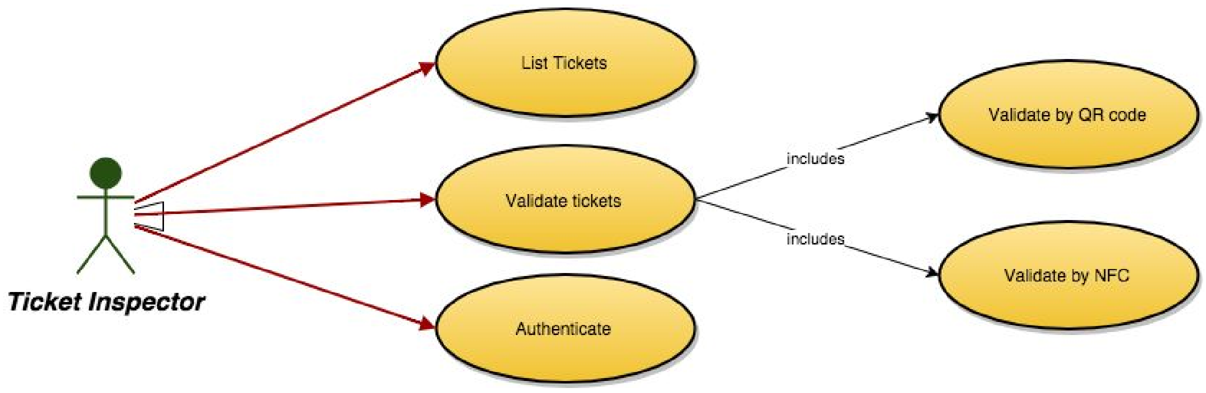
Node labels

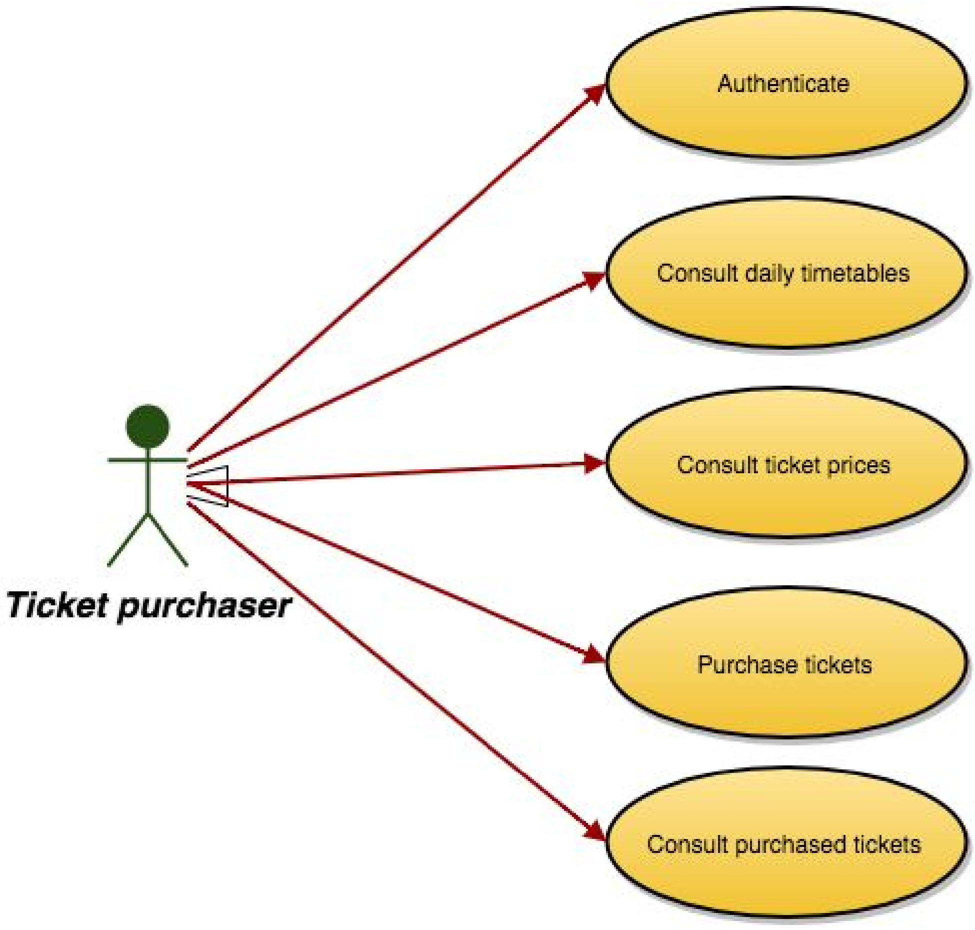
user *id*: integer *name*: string *username*: string *password*: string *email*: string *role*: passenger|inspector	**station** id: integer *name*: string *isCentral*: boolean *latitude*: float *longitude*: float	**credit_card** *id*: integer *number*: string *expireDate*: timestamp
train *id*: integer *name*: string	**ticket** *id*: integer *creationDate*: timestamp *trips*: array of integers status: pending|noShow|validated	

Relationship types

trip (station ↔ station) *departureDate*: timestamp *arrivalDate*: timestamp *trainId*: integer *distance*: float	**owns (user → credit_card)** *id*: integer	**bought (user → ticket)** *id*: integer

Use cases





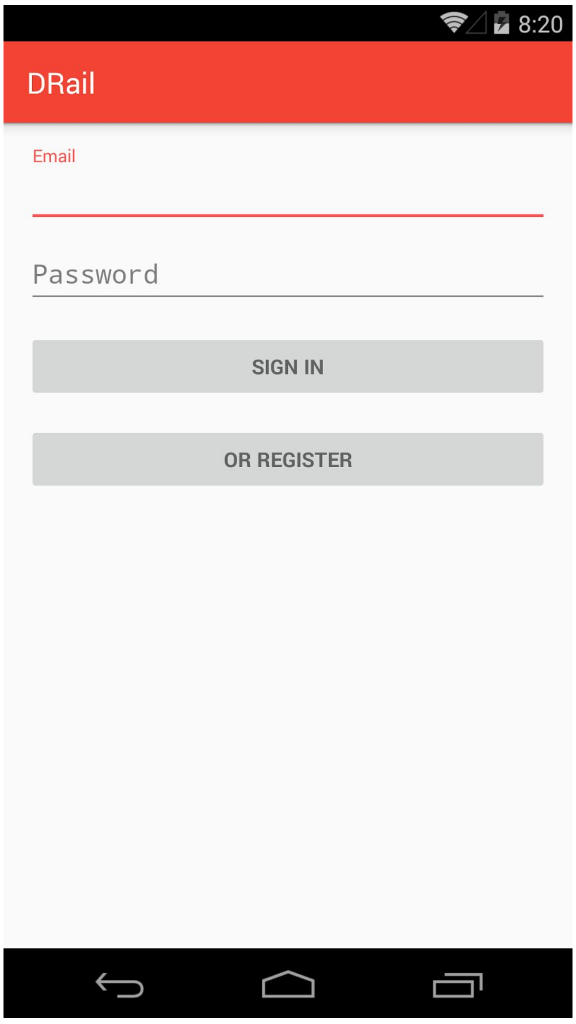
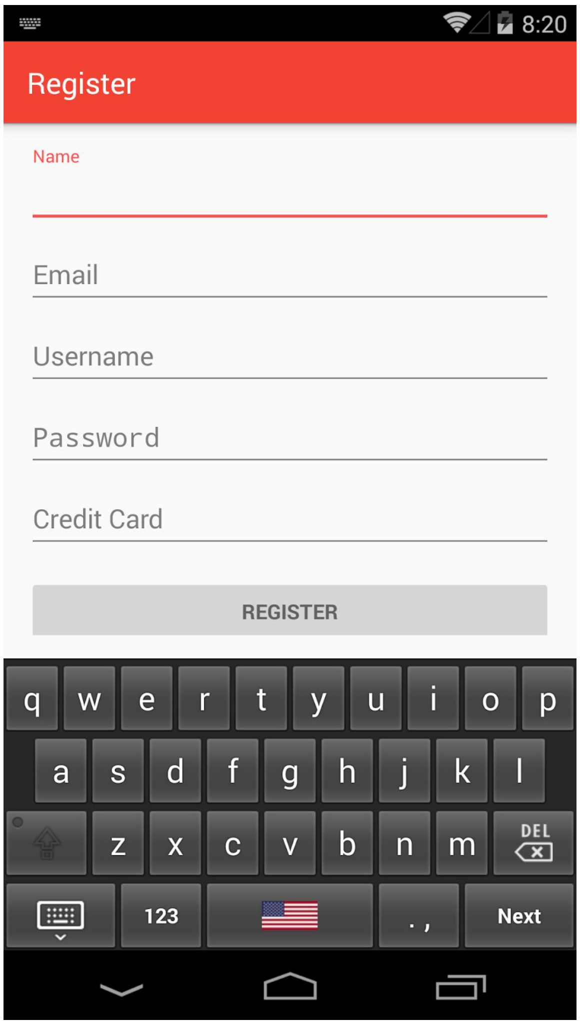
Security

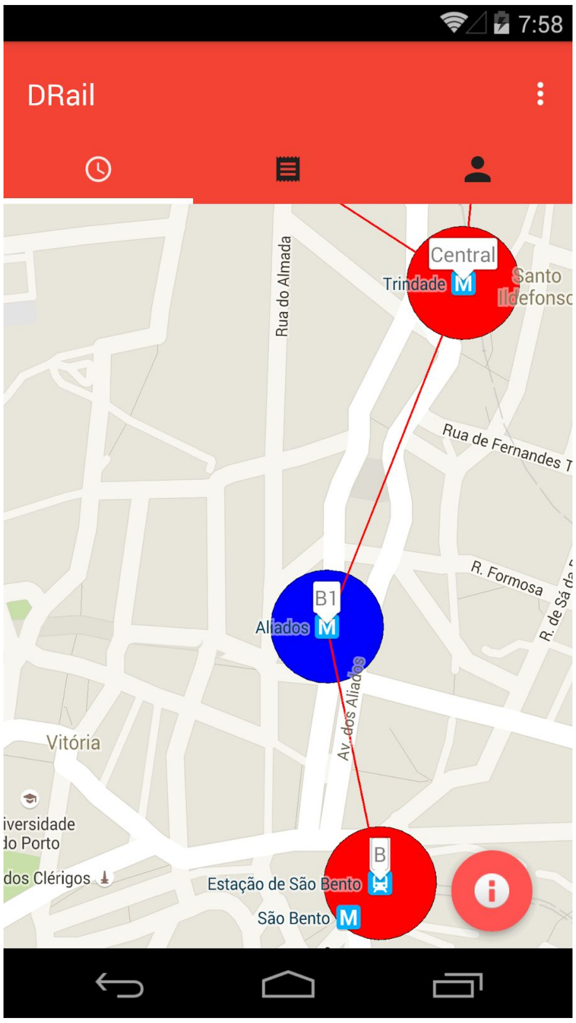
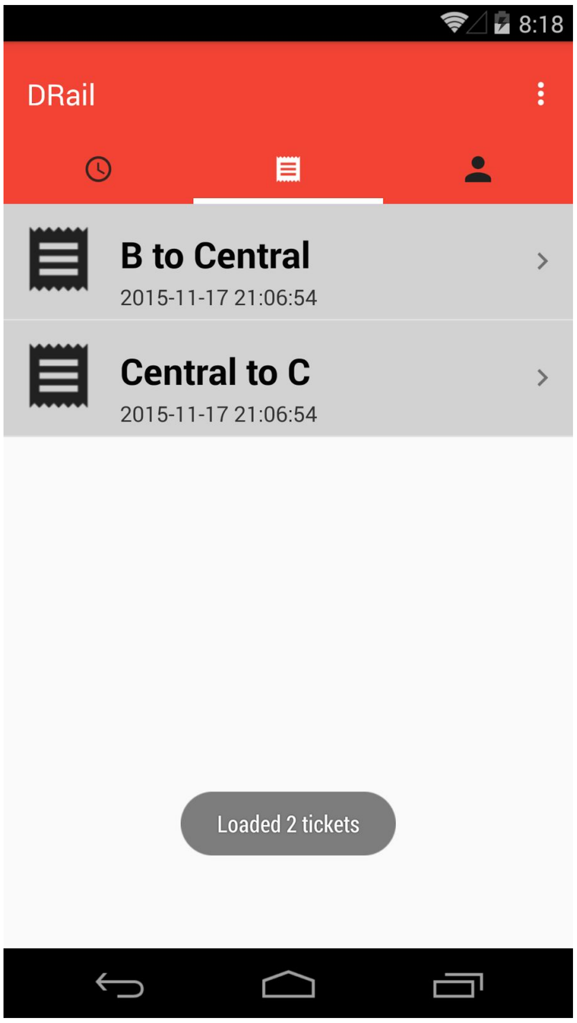
All HTTP requests (except POST /login and POST /register) require the HTTP header Bearer to be set. This token is generated on login or on register using JSON Web Token (JWT, RFC 7519).

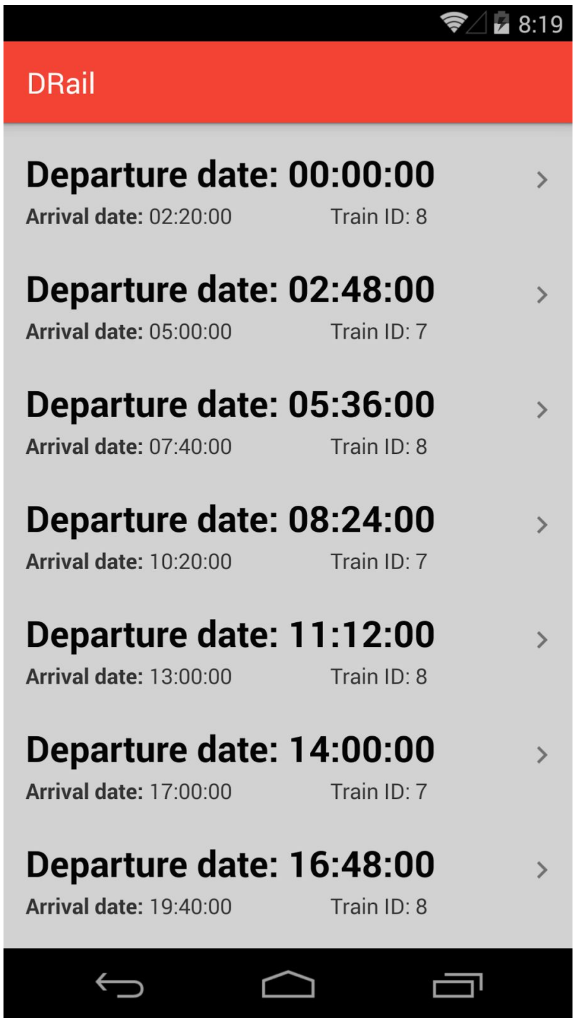
To verify the authenticity of the tickets, a digital signature scheme is used. More specifically, we are using SHA1 with RSA in the following manner:

1. When a ticket is created, the server generates the signature with a private key using the identifier and creation date of the ticket.
2. The signature and other ticket information is sent to both Android clients (when requested to do so).
3. The customer app generates a QR token including the id, creation date and signature of the ticket.
4. The inspector app scans the QR and verifies that the id and date are valid (not tempered with and not duplicated) using the public key.
5. If the signature is valid, the ticket is marked as valid and used.

Application interface

	 8:20 Register Name Email Username Password Credit Card REGISTER 123 Next
Screen 1 - Customer app - Login view	Screen 2 - Customer app - Register view

 7:58 DRail Central Trindade Santo Ildefonso Rua do Almada Rua de Fernandes T R. Formosa R. de Sá da B1 Aliados Av. dos Aliados Vitória iversidade do Porto dos Clérigos B Estação de São Bento São Bento	 **B to Central** 2015-11-17 21:06:54 **Central to C** 2015-11-17 21:06:54 Loaded 2 tickets
Screen 3 - Customer app - Scheduling view	Screen 4 - Customer app - Ticket list view


8:19

DRail

Departure date: 00:00:00
Arrival date: 02:20:00 Train ID: 8

Departure date: 02:48:00
Arrival date: 05:00:00 Train ID: 7

Departure date: 05:36:00
Arrival date: 07:40:00 Train ID: 8

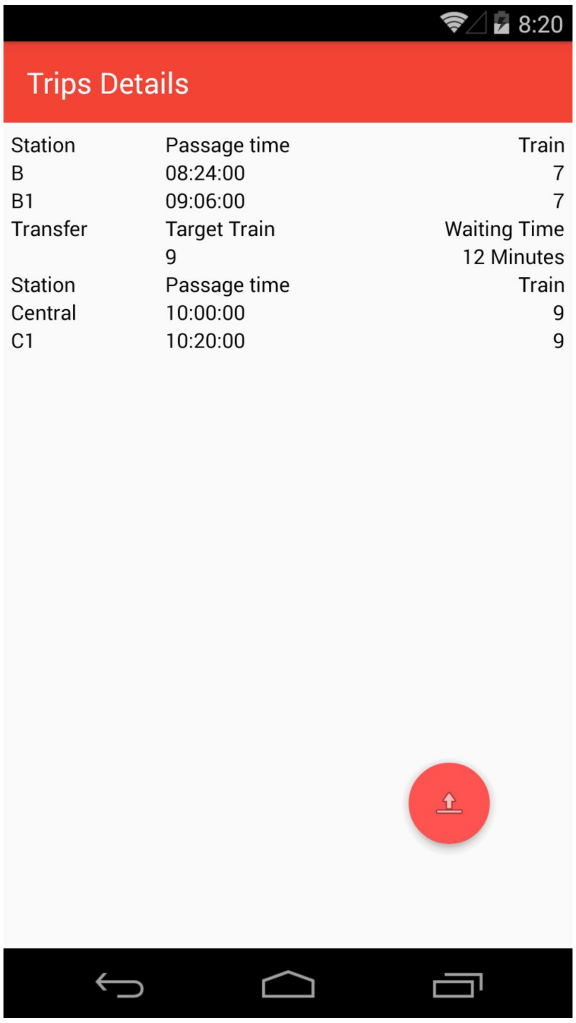
Departure date: 08:24:00
Arrival date: 10:20:00 Train ID: 7

Departure date: 11:12:00
Arrival date: 13:00:00 Train ID: 8

Departure date: 14:00:00
Arrival date: 17:00:00 Train ID: 7

Departure date: 16:48:00
Arrival date: 19:40:00 Train ID: 8

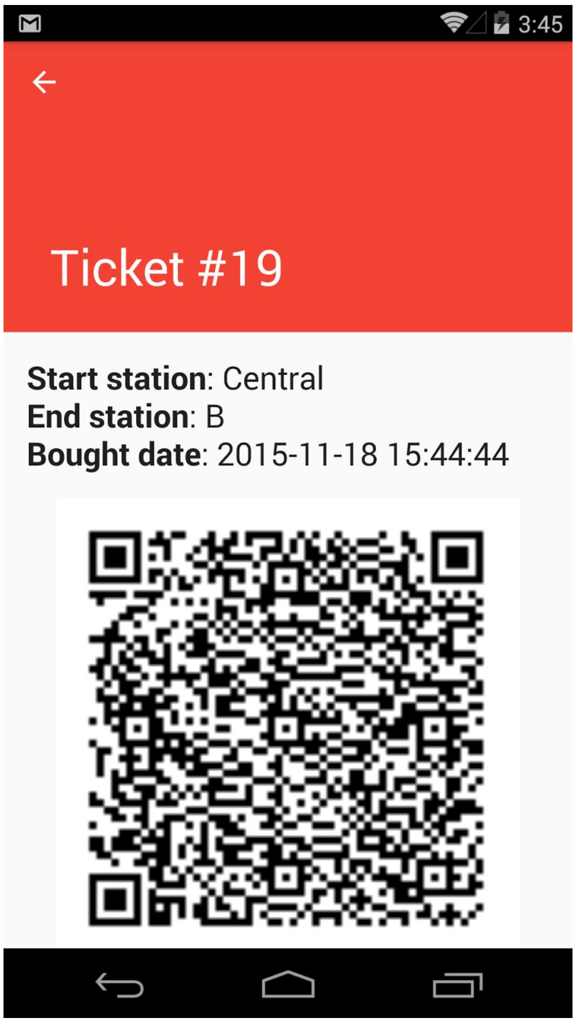
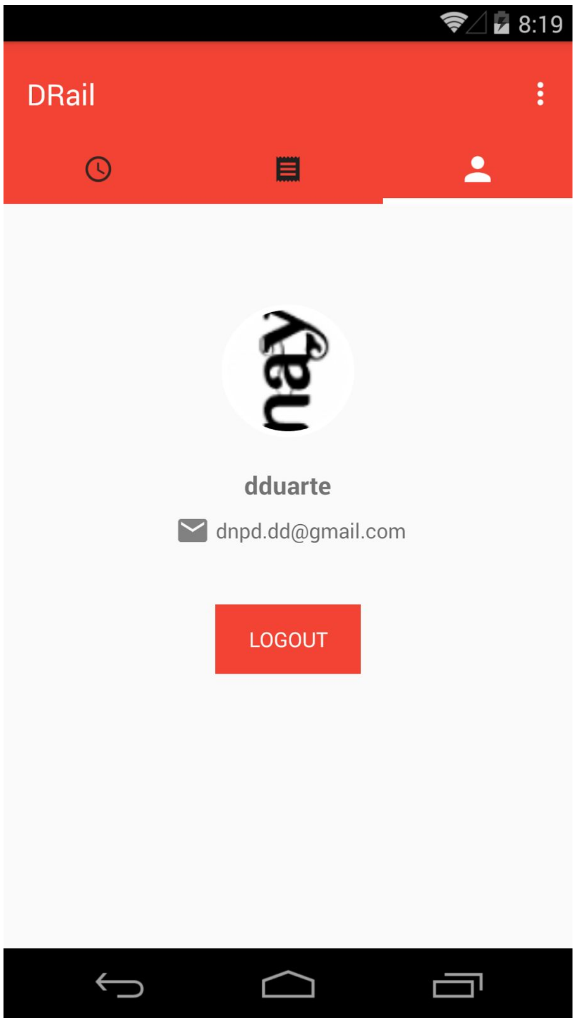
Screen 5 - Customer app - Trip list view

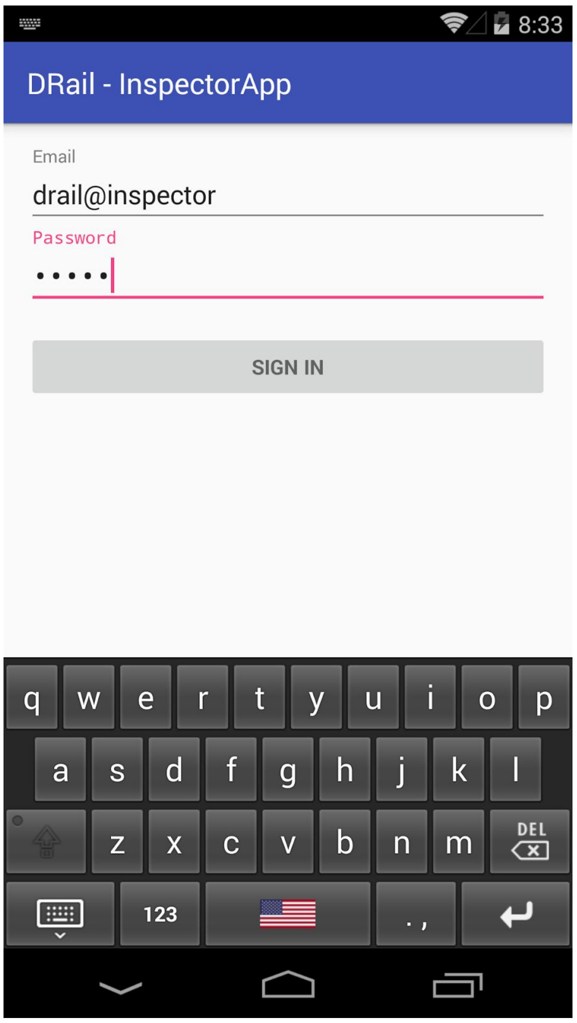
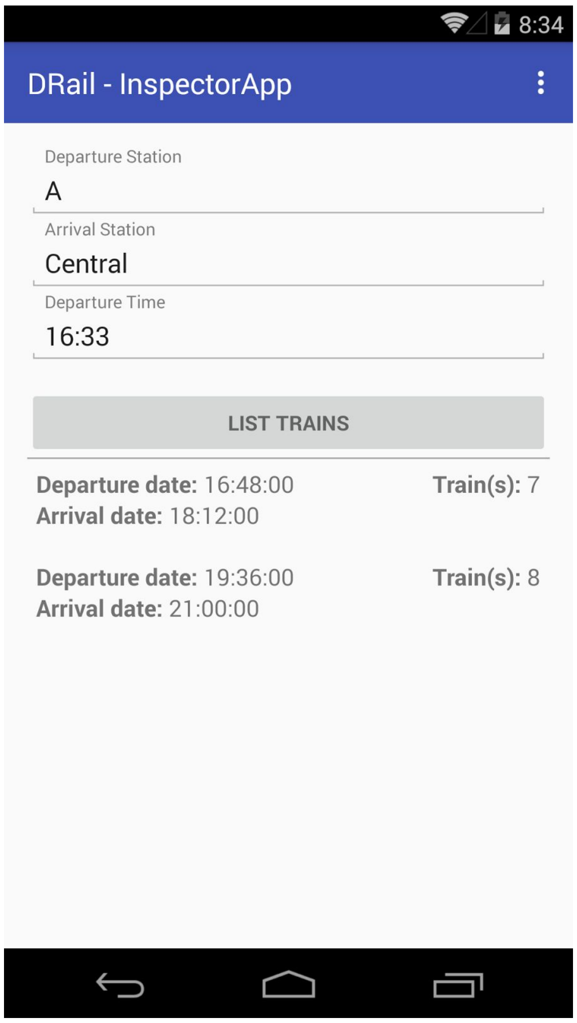

8:20

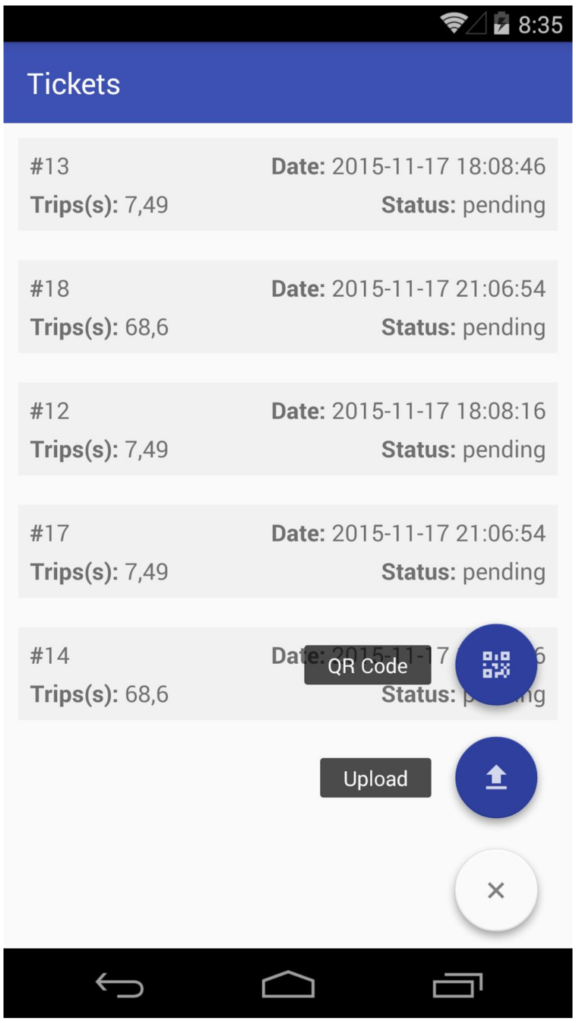
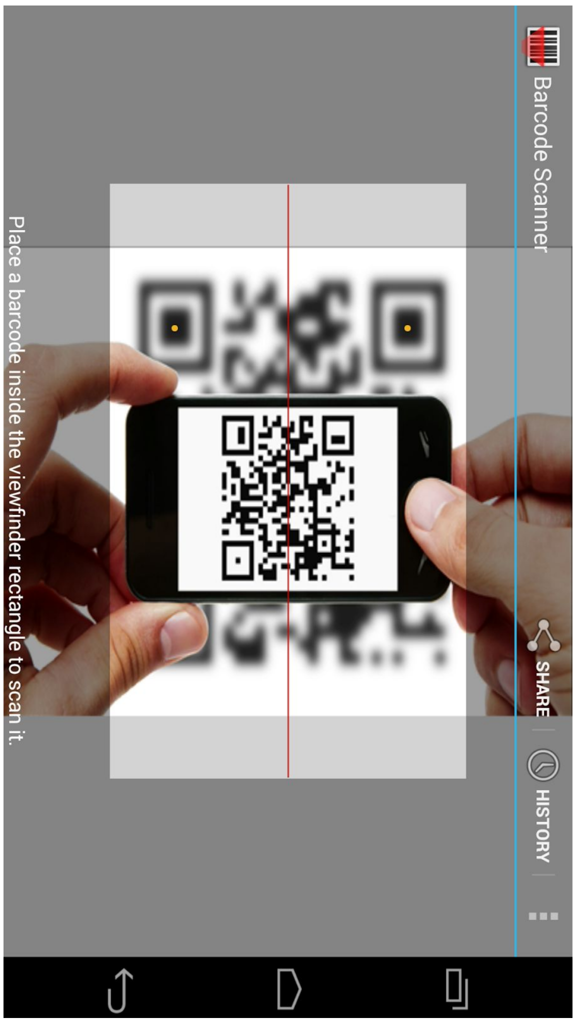
Trips Details

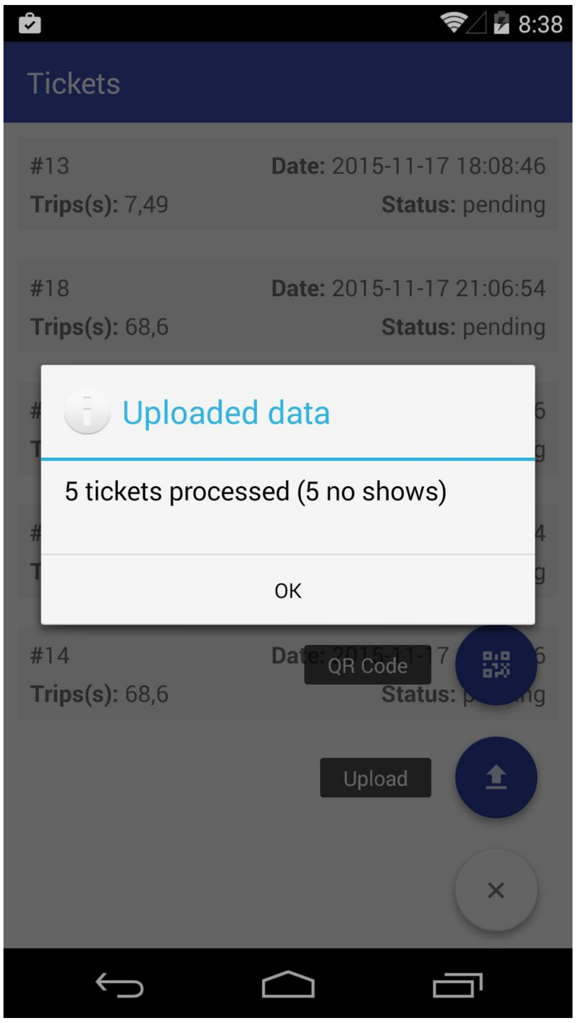
Station	Passage time	Train
B	08:24:00	7
B1	09:06:00	7
Transfer	Target Train	Waiting Time
	9	12 Minutes
Station	Passage time	Train
Central	10:00:00	9
C1	10:20:00	9

Screen 6 - Customer app - Trip details view

 3:45 Ticket #19 **Start station**: Central **End station**: B **Bought date**: 2015-11-18 15:44:44	 8:19 DRail dduarte dnpd.dd@gmail.com
Screen 7 - Customer app - Ticket view	Screen 8 - Customer app - User info view

 8:33 DRail - InspectorApp Email drail@inspector Password ••••• SIGN IN	 8:34 DRail - InspectorApp Departure Station A Arrival Station Central Departure Time 16:33 LIST TRAINS **Departure date:** 16:48:00 **Train(s):** 7 **Arrival date:** 18:12:00 **Departure date:** 19:36:00 **Train(s):** 8 **Arrival date:** 21:00:00
Screen 9 - Inspector app - Login view	Screen 10 - Inspector app - Trip select. view

 8:35 Tickets #13 Date: 2015-11-17 18:08:46 Trip(s): 7,49 Status: pending #18 Date: 2015-11-17 21:06:54 Trip(s): 68,6 Status: pending #12 Date: 2015-11-17 18:08:16 Trip(s): 7,49 Status: pending #17 Date: 2015-11-17 21:06:54 Trip(s): 7,49 Status: pending #14 Date: 2015-11-17 6 Trip(s): 68,6 Status: p ng QR Code Upload	 Barcode Scanner Place a barcode inside the viewfinder rectangle to scan it. SHARE HISTORY
Screen 11 - Inspector app - Ticket list view	Screen 12 - Inspector app - QR scanning (placeholder)

 8:38 Tickets #13 **Date:** 2015-11-17 18:08:46 **Trips(s):** 7,49 **Status:** pending #18 **Date:** 2015-11-17 21:06:54 **Trips(s):** 68,6 **Status:** pending Uploaded data 5 tickets processed (5 no shows) OK #14 **Trips(s):** 68,6 QR Code Upload	
Screen 13 - Inspector app - Upload ticket info view	

Basic application usage

Customer

1. The user should login or register (first visit) (screen 1 or 2)
2. In the scheduling view (screen 3), the user can select two stations (red) to browse all the trips between them.
3. After selecting the trip (screen 5), the user is presented with the trip details (screen 6) and a button so that the trip can be bought.
    a. If the credit card is not deemed valid when buying the ticket, the user has to redo the process.
4. The user can select the ticket in the ticket list view (screen 4).

5. Once in the ticket view (screen 7), the QR Code can be shown to the inspector or connect to the Inspector App using NFC.

Inspector

1. The inspector can login with the credentials given by the company (e.g username drail@inspector and password drail) (screen 9).
2. After selecting the departure station, arrival station and departure station, a list of trips is shown (screen 10).
3. Once the desired trip is selected, a list of tickets is shown (screen 11) and three options are enabled:
   - Validating tickets by NFC (not shown, get two devices together)
   - Validating tickets by QR Code (screen 12)
   - Upload the state of the tickets once the trip is finished (screen 13).