UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Luka Finžgar

Implementacija elektronske vozovnice na mobilnem telefonu

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mira Trebar

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01699/2010
Datum: 01.09.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **LUKA FINŽGAR**

Naslov: **IMPLEMENTACIJA ELEKTRONSKE VOZOVNICE NA MOBILNEM TELEFONU**

IMPLEMENTATION OF AN ELECTRONIC TICKET ON MOBILE PHONE

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V diplomskem delu naj kandidat analizira različne izvedbe elektronske vozovnice v javnem potniškem prometu. Za predlagano rešitev naj upošteva široke možnosti uporabe spletnih tehnologij, brezžične komunikacije NFC (Near Field Communication) in operacijskega sistema Android za mobilne telefone. V implementaciji aplikacij na spletnem strežniku in mobilnih telefonih naj bodo za uporabo elektronske vozovnice na vlaku upoštevane naslednje funkcije: vstop, izstop, preverjanje veljavnosti vozovnice in analizo podatkov o številu potnikov in obremenjenosti prog. Delovanje sistema naj kandidat praktično preveri v laboratorijskem okolju.

Mentor:

doc. dr. Mira Trebar

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Luka Finžgar,

z vpisno številko 63030155,

sem avtor diplomskega dela z naslovom:

Implementacija elektronske vozovnice na mobilnem telefonu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mire Trebar
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 6.12.2010 Podpis avtorja:

Zahvala

Zahvaljujem se družini za dragoceno podporo in spodbudo v času študija. Zahvala gre tudi mentorici doc. dr. Miri Trebar za nasvete in pomoč pri izdelavi diplomske naloge.

Diplomsko delo posvečam prijateljem iz smeri računalniški sistemi.

Kazalo

Seznam uporabljenih kratic in simbolov

CDC – Connected Device Configuration
CLDC – Connected Limited Device Configuration
CSS – Cascading Style Sheets
DDMS – Dalvik Debug Monitor Server
GWT – Google Web Toolkit
HF – High Frequency
HTTP – Hypertext Transfer Protocol
JDBC – Java Database Connectivity
LF – Low Frequency
MIDP – Mobile Information Device Profile
MIME – Multipurpose Internet Mail Extensions
NDEF – NFC Data Exchange Format
NFC – Near Field Communication
NVRAM – Non-Volatile Random Access Memory
OHA – Open Handset Alliance
PDF – Portable Document Format
PNG – Portable Network Graphics
QR CODE – Quick Response code
RFID – Radio-frequency identification
RPC – Remote procedure call
SQL – Structured Query Language
SSID – Service Set Identifier
UHF – Ultra High Frequency
UID – Unique identifier
URL – Uniform Resource Locator
XML – Extensible Markup Language

Povzetek

Nakup vozovnice za javni potniški prevoz zelo pogosto podaljša čas potovanja, zato bi bilo primerno imeti dober elektronski sistem, ki bi to opravilo izvedel čim enostavneje in brez dodatnih operacij. Splošna dostopnost in uporabnost mobilnega telefona se zaradi svojih zmogljivosti in možnosti dostopa do interneta pokaže kot ena od primernih rešitev.

V diplomskem delu sem opisal izvedbo sistema, v katerem mobilni telefon predstavlja elektronsko vozovnice javnega prevoza. Pri tem sta bili upoštevani dve sodobni tehnologiji, ki omogočata razvoj aplikacij za mobilne telefone. V prvem primeru je bil uporabljen mobilni telefon z operacijskim sistemom Android, v drugem pa brezkontaktna komunikacijska tehnologija v bližnjem polju (NFC-Near Field Communication) za mobilni telefon Nokia 6212. Potnik se s svojim telefonom registrira na vstopni postaji in si s tem zagotovi nakup enkratne vozovnice ali preverjanje mesečne vozovnice. Veljavno vozovnico pridobi v obliki dvodimenzionalne črtne kode QR z aplikacijo, ki se izvaja na spletnem strežniku. Na izstopni postaji se potnik znova identificira, kar pomeni, da je njegovo potovanje končano in se mu izstavi obračun. Za hitro identifikacijo vstopne in izstopne postaje se uporabijo črtne kode QR ali značke RFID. Z uporabo mobilne aplikacije lahko uporabnik pridobi še prihodne čase vlakov in pregleda svojo porabo. V mobilni aplikaciji je za potrebe sprevodnikov implementirana tudi funkcija preverjanja vozovnic.

Poleg možnosti nakupa elektronske vozovnice je bila izdelana tudi spletna stran, ki omogoča registracijo novih uporabnikov. Administratorji lahko pregledujejo določene podatke o opravljenih potovanjih. Sistem je zasnovan tako, da uporabnik enkrat mesečno na svoj elektronski naslov prejme račun za pretekla potovanja.

Cilj diplomske naloge je bil raziskati in implementirati možnost uporabe RFID značk in QR črtnih kod v aplikaciji z mobilnimi telefoni za uporabo v javnem prometu.

Kombinacija se je izkazala za zelo posrečeno, kar se odraža tudi v njeni uporabi v nekaterih državah.

Ključne besede:

Elektronska vozovnica, javni prevoz, Android, QR črtna koda, RFID

Abstract

Buying a public transport ticket often makes the whole journey longer, so it would be convenient to have a electronic system at our disposal, that could make this task as easier as possible and without additional steps for the passenger. The wide use of mobile phones, their computational capabilities and their ability to connect to the Internet makes them suitable for solving this problem.

This thesis describes the implementation of a system, which enables the use of mobile phones for acquiring public transport ticket. The development of mobile applications was done using two modern technologies, mobile operating system Android and Near Field Communication (NFC) for Nokia 6212. The passenger registers himself at the start of the journey with his mobile phone and receives an electronic ticket. The ticket is sent to the mobile phone in the form of QR barcode from the application on the server. The passenger registers himself again at the end of the journey, and the system finalizes his trip and calculates the fare. QR barcodes and RFID tags are used for registering passenger at the beginning and at the end of his journey. Mobile application user can also display train arrivals and see the amounts charged. Train conductor can use the Android application to check the validity of the tickets.

A website was built, where new users of this system can register. Administrators can see the data on the journeys of users on another website. The system is designed in a way that users receive the receipt for their past journeys in their email once a month.

The aim of this thesis was to explore the possibility of using RFID tags and QR barcodes along with mobile phones in public transport. This combination was found out to be very promising, what can also be seen in current use of similar systems in some countries.

Key words:

Electronic ticket, Public transport, Android, QR barcode, RFID

1 Uvod

Uporabnik javnega prevoza vsako potovanje začne z nakupom vozovnice. Tu se v zadnjem času uporablja kar nekaj tehnologij. Po svetu in vse več tudi v Sloveniji so uveljavljene naslednje rešitve: nakup prek spleta in samostojno tiskanje vozovnice (npr. pri avstrijskih železnicah [1]), nakup na avtomatu, nakup na okencu postaje in plačevanje z mobilnim telefonom ali brezkontaktno kartico. Pri nas je najbolj znan in razširjen sistem za plačevanje z mobilnim telefonom Moneta, s katero je med drugim mogoče plačevati tudi storitve ljubljanskega potniškega prometa [2], v tujini pa na primer sistem nemških železnic Touch&Travel [3]. Primer sistema plačevanja z brezkontaktno kartico v Sloveniji je Urbana [4], med bolj znanimi v tujini pa velja omeniti londonsko kartico Oyster [5].

Dober sistem prodaje vozovnic mora biti za uporabnika enostaven in čim manj zamuden. Nakup vozovnice na okencu ali avtomatu lahko namreč ob gneči zelo podaljša potovanje, zato so se pojavili manj zamudni sistemi. Ko sem med študijem uporabljal javni prevoz, sem bil večkrat priča gneči in zastarelemu načinu prodaje vozovnic. Ker sem že nekoliko poznal sisteme v tujini, sem tako dobil idejo za diplomsko nalogo.

Mobilni telefoni so danes ves čas z nami in ker postajajo vse zmogljivejši, so nadvse primerni, da se jih uporablja kot plačilno sredstvo in način identifikacije.

Cilj diplomske naloge je bil razviti sistem, ki bi omogočal uporabo mobilnega telefona za pridobitev in plačilo vozovnice javnega prevoza. Omogočal naj bi tudi registracijo uporabnikov prek spletnega vmesnika in prikaz nekaterih podatkov o opravljenih prevozih za administratorje.

Za razliko od sistemov [4,5], naš sistem na vstopni/izstopni postaji ne potrebuje čitalcev pametnih kartic in druge strojne opreme. Dovolj so oznake postaj na QR črtnih kodah ali RFID značkah ter mobilni telefon s posebno aplikacijo in možnostjo dostopa do interneta. Vse ostale dejavnosti kot avtorizacija, beleženje potovanj, zaračunavanje itd. se izvajajo na centralnem strežniku.

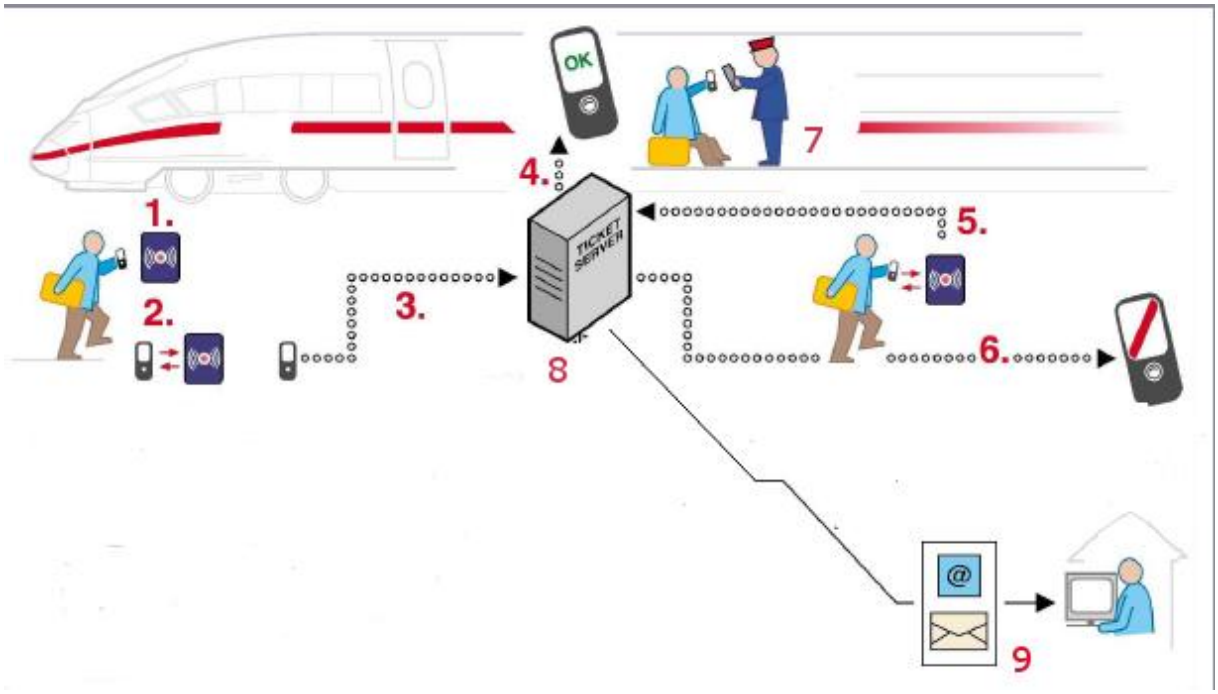
2 Elektronska vozovnica na mobilnem telefonu

Potnik lahko z uporabo aplikacije na mobilnem telefonu pridobi elektronsko vozovnico, pri čemer se mora prijaviti na vstopni postaji (ang. check in) in odjaviti na izstopni postaji (ang. check out). Potrebno se je bilo odločiti:

- na kakšen način naj mobilni telefon iz okolice dobi informacijo o vstopni/izstopni postaji,
- katere mobilne telefone naj se uporabi,
- kako naj bo predstavljena vozovnica.

Za identifikacijo mesta vstopa in izstopa sem se odločil uporabiti QR črtne kode ter RFID značke. QR črtno kodo lahko uporabimo za mobilne telefone s kamero (izbral sem mobilni

telefon HTC Hero z OS Android). Značke RFID pa lahko uporabimo za mobilne telefone, ki imajo vgrajen modul za komunikacijo NFC (ang. Near Field Communication). V ta namen sem uporabil mobilni telefon Nokia 6212. Tudi vozovnico predstavimo s QR črtno kodo, saj nam to omogoča enostavno in hitro preverjanje veljavnosti vozovnic z branjem kode z zaslona potnikovega mobilnega aparata. Predlog uporabe sistema je opisan na sliki 1.



Slika 1. Sistem elektronske vozovnice

Postopek je sestavljen iz naslednjih možnosti:

Začetek potovanja

1. Uporabnik približa mobilni telefon RFID znački ali QR črtni kodi, ki se nahajata na postaji.
2. Program na mobilnem telefonu prebere ime vstopne postaje.
3. Program na strežnik pošlje zahtevo za vozovnico.
4. Program na strežniku avtorizira uporabnika in mu pošlje vozovnico v obliki QR črtne kode ter zabeleži podatke o potniku.

Konec potovanja

5. Iz RFID značke ali QR črtne kode se z mobilnim telefonom prebere ime izstopne postaje in ta podatek se pošlje na strežnik.
6. Strežnik zaključi potovanje.

Preverjanje vozovnic

7. Sprevodnik z zaslona potnikovega mobilnega telefona prebere vozovnico in preveri njeno veljavnost.

Zaračunavanje

8. Strežnik za vstopno in izstopno postajo izračuna ceno potovanja.
9. Račun za potovanja v preteklem mesecu uporabnik prejme prek elektronske pošte.

Za izvedbo tako zasnovane rešitve sem razvil aplikacije na spletnem strežniku, za mobilni telefon z OS Android in za mobilni telefon Nokia 6212. Sledi opis razvite funkcionalnosti:

Spletni strežnik

Uporabnik se prek spletne strani registrira in pri tem izbere želeno uporabniško ime, geslo, način plačevanja in vrsto svojega mobilnega telefona. Po uspešni registraciji prejme elektronsko pošto s potrebnimi podatki za uporabo in aplikacijo za svoj mobilni telefon.

Na strežniku se izvaja tudi aplikacija, ki skrbi za naslednje funkcionalnosti:

- na zahtevo mobilnih odjemalcev pošilja vozovnice v obliki QR črtne kode,
- beleži vstopno in izstopno postajo določenega uporabnika ter računa stroške opravljenega potovanja,
- na zahtevo uporabnikov jim pošilja njihovo porabo in odhodne čase vlakov za izbrano vstopno postajo,
- omogoča zagon programa, ki za pretekli mesec izračuna porabo za vse registrirane uporabnike in jim to pošlje v obliki PDF-priponke na njihov elektronski naslov.

Aplikacija za mobilni telefon z operacijskim sistemom Android

Aplikacija omogoča prijavo na določeno vstopno postajo z branjem QR črtne kode s pomočjo fotoaparata mobilnega telefona. Po prijavi se pridobi tudi vozovnica v obliki QR črtne kode. Na enak način kot prijava poteka tudi odjava z določene izstopne postaje.

Po vstopu je mogoče prek aplikacije pridobiti prihodne čase vlakov za izbrano postajo. Uporabnik lahko pregleda tudi svojo porabo in med potovanjem prikaže svojo aktivno vozovnico sprevodniku.

Ob vnosu administratorskega imena in gesla je s programom mogoče preveriti veljavnost vozovnic drugih uporabnikov, in to z branjem vozovnic z zaslona njihovega mobilnega telefona. Ta funkcija je namenjena za potrebe sprevodnikov.

Aplikacija za mobilni telefon Nokia 6212

Aplikacija omogoča prijavo na določeno vstopno postajo in pridobitev vozovnice z branjem RFID značke s pomočjo NFC čitalca znotraj mobilnega telefona. Na enak način se vrši tudi odjava na izstopni postaji.

Kot pri programu za Android je po vstopu mogoče prek aplikacije pridobiti prihodne čase vlakov za izbrano postajo. Uporabnik lahko pregleda tudi svojo porabo in med potovanjem prikaže svojo aktivno vozovnico sprevodniku.

Spletna aplikacija za administratorje

Spletna stran je namenjena podjetju, ki izvaja prevoze. Administrator lahko po prijavi pregleduje naslednje podatke:

- število potnikov na določenih progah (tudi grafično),
- število potnikov na določeni progi ob določenem času,
- razmerje med uporabniki, ki plačujejo po porabi, in tistimi, ki plačujejo pavšalne mesečne zneske (tudi grafično),
- število novih uporabnikov za zadnje tri mesece (tudi grafično).

Poleg prikaza podatkov lahko vnašajo tudi odhodne čase za določeno postajo in nove cene mesečne vozovnice.

3 Uporabljena orodja, metode, tehnologije

Razvoj spletne strani, »servleta« in aplikacij za mobilne telefone je potekal v razvojnem okolju Eclipse [6] z uporabo več vtičnikov (ang. plugin) Sledi opis uporabljenih tehnologij, orodij in potek razvoja.

3.1 Android

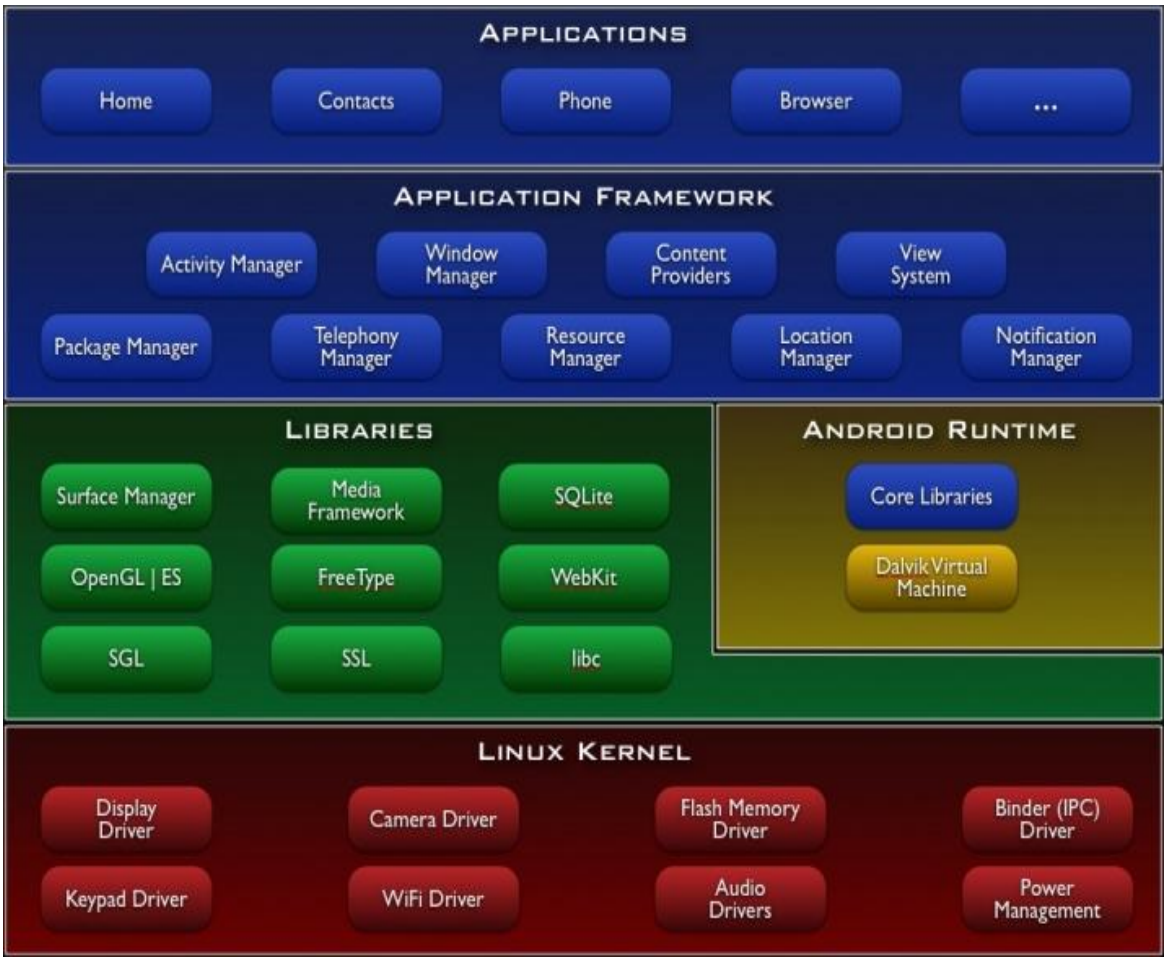
Android [7] je programska platforma in operacijski sistem za pametne mobilne telefone, ki temelji na Linux jedru. Razvija ga Google v sodelovanju s podjetji združenja Open Handset Alliance (OHA), ki si prizadeva za skupen razvoj odprtih standardov na področju prenosnih naprav. Android je v prvi vrsti namenjen mobilnim telefonom, v zadnjem času pa vedno več proizvajalcev napoveduje tudi tablične računalnike s tem operacijskim sistemom. Večina kode je izdane kot odprtokodne pod Apache 2.0 licenco.

Odprtokodnost Androida prinaša naslednje prednosti:

- za potrošnike to pomeni, da bodo imeli na voljo cenejše in bolj funkcionalne mobilne naprave in večje število inovativnih storitev,
- proizvajalcem mobilnih telefonov omogoča znižanje stroškov razvoja programske opreme in hitrejšo realizacijo svojih konceptov, prav tako pa jim ob fleksibilnosti sistema še vedno prinaša možnost krojenja lastne identitete z razvojem samosvojih rešitev,
- podjetja, ki se ukvarjajo s programsko opremo, bodo odslej enostavneje in ceneje izdelovala programske komponente, ki bodo na voljo velikemu številu uporabnikov.

S stališča razvijalcev je zelo dobrodošla možnost zamenjave vseh komponent sistema, od programa za klicanje do programa za pošiljanje SMS-sporočil. Poleg tega se je okrog sistema razvila že zelo velika skupnost, ki nam je pripravljena priskočiti na pomoč pri vsaki težavi v razvoju naše aplikacije. Kljub začetnim težavam pa je na zelo visoki ravni tudi že uradna dokumentacija.

Glavne komponente operacijskega sistema Android so prikazane na sliki 2.



Slika 2. Glavne komponente OS Android.

Android temelji na Linux jedru verzije 2.6, ki ga uporablja za varnost, upravljanje s pomnilnikom, razvrščanje, mrežni sklad ter gonilnike. Za Android so na voljo tudi C/C++ knjižnice, ki jih potrebujejo razni deli sistema. Razvijalec lahko do njih dostopa prek programskega vmesnika. Android podpira večopravilnost in omogoča, da aplikacije tečejo v ozadju.

Vsaka Android aplikacija teče znotraj svojega procesa in znotraj lastnega Dalvik navideznega stroja (ang. Dalvik virtual machine). Dalvik je optimiziran navidezni stroj, ki omogoča tudi manj zmogljivim napravam (pomnilniško ali procesorsko) izvajanje več navideznih strojev Dalvik hkrati. Datoteke vrste »class«, dobljene z Java prevajalnikom, se pretvori v format »dex« (Dalvik izvršljiva koda) z orodjem »dx« in nato požene v Dalvik VM. Dalvik VM je z izdajo Android 2.2 veliko pridobil na hitrosti zaradi uvedbe JIT prevajanja (ang. just in time).

Za razvoj programov za Android imamo na voljo Android SDK [8], ki vsebuje potrebna orodja za emulacijo in razhroščevanje ter programske vmesnike (API). Razvoj poteka v programskem jeziku Java ali C (za časovno kritične dele kode z uporabo Android NDK [9]).

V veliko pomoč mi je bil vtičnik za Eclipse Android development tools (ADT), ki zelo poenostavi programiranje. Na voljo imamo emulator, ki omogoča testiranje naše aplikacije pred nalaganjem na mobilni telefon. Emulator je zelo zmogljiv, saj zna med drugim simulirati

prejete klice, SMS sporočila in spremembo geografskih koordinat (za testiranje lokacijsko odvisnih aplikacij). Poganjanje aplikacije na emulatorju je prikazano na sliki 3.

Slika 3. Zagon aplikacije na emulatorju.

Na voljo imamo tudi zmogljiv razhroščevalnik DDMS (ang. Dalvik Debug Monitor Service), ki omogoča uporabo zastavic (ang. watches) in prekinitvenih točk (ang. breakpoints), kot smo tega vajeni pri programiranju v skorajda vseh naprednejših razvojnih okoljih (IDE). Za naprednejše razvijalce je na voljo še izpis kopice, niti ter procesov na emulatorju in ročni zagon pobiralca smeti (ang. garbage collector).

Oblikovanje grafičnega vmesnika pri Androidu poteka z urejanjem XML datotek, kar je zelo podobno urejanju spletnih strani s spreminjanjem HTML datotek.

Pri izdelavi aplikacij za Android ne moremo mimo mehanizma, imenovanega namen (ang. intent). Namen je podoben komunikaciji med procesi (IPC). Gre za podatkovno strukturo, ki vsebuje opis akcije, ki se mora izvesti. Glavni dve informaciji, ki jih vsebuje, sta akcija (na primer prikaz »ACTION_VIEW«) in podatek, nad katerim naj se ta akcija izvede (na primer telefonska številka »tel041922674«). Namen se najpogosteje uporablja za zagon aktivnosti in servisov.

Pri razvoju za Android je zelo pomembna tudi datoteka AndroidManifest.xml, brez katere ne deluje nobena aplikacija. S to datoteko se aplikacija predstavi OS Android in med drugim vsebuje naslednje informacije:

- ime java paketa, ki služi kot unikatni identifikator,

- seznam glavnih komponent sistema (aktivnosti, servisi, ponudniki podatkov, …),
- seznam dovoljenj, ki so potrebni za delovanje aplikacije,
- minimalno verzijo Android OS-a, potrebnega za delovanje aplikacije.

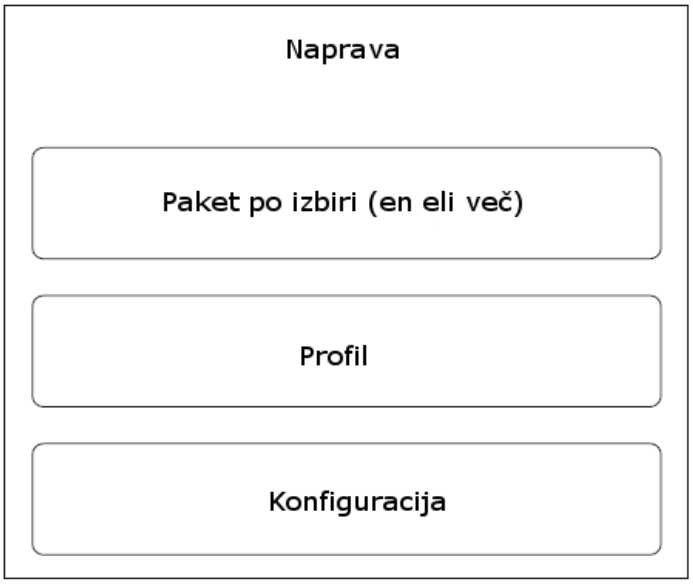
Preden lahko našo aplikacijo uporabimo na telefonu, ali jo pošljemo v Google Android Market, jo moramo digitalno podpisati. Zadostuje že podpis z lastnim certifikatom (ang. self-signed), kar nam omogoča tudi čarovnik znotraj orodja Eclipse.

3.2 Java ME

Java Platform, Micro Edition (Java ME), je Java platforma, razvita za uporabo na mobilnih napravah in vgradnih sistemih (ang. embedded systems). Razvili so jo pri podjetju Sun Microsystems. Trenutno je prisotna že v več kot dveh milijardah mobilnih telefonov, dlančnikih in tv komunikatorjih (ang. set-top box).

Funkcionalnost Java ME temelji na treh konceptih, ki definirajo programske sposobnosti določene naprave (slika 4):

- konfiguraciji (ang. configuraton), ki definira množico knjižnic in vrsto Java navideznega stroja za širšo skupino naprav,
- profilu (ang. profile), ki definira množico programskih vmesnikov za ožjo skupino naprav,
- paketu (ang. package), ki definira opcijske programske vmesnike, ki se nanašajo na določeno tehnologijo, ki je navzoča le v posameznih napravah (npr. NFC, Bluetooth).



Slika 4. Razmerje med konfiguracijo, profilom in paketom pri Java ME.

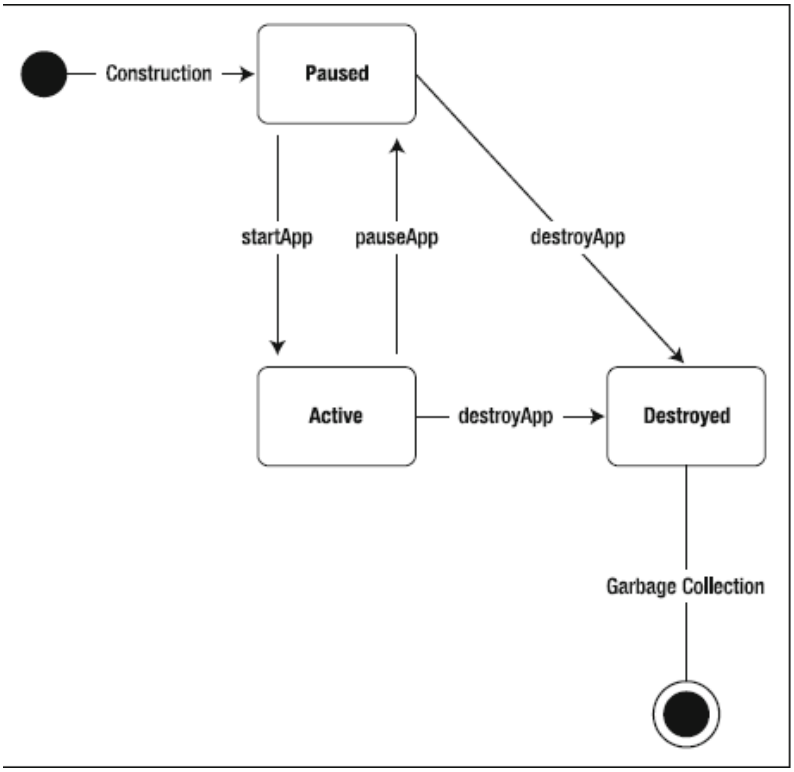
Trenutno obstajata dve konfiguraciji znotraj Java ME, »the Connected Limited DeviceConfiguration (CLDC)« in »Connected Device Configuration« (CDC). CLDC vsebuje majhno podmnožico Java SE knjižnic (ni zbirk in naprednejših podatkovnih struktur). Namenjen je napravam z omejenim pomnilnikom in/ali omejeno procesorsko močjo. Prevedena koda (.class datoteka) se izvaja na K virtualnem stroju, ki je spremenjen JVM, optimiziran za poganjanje na napravah z baterijskim napajanjem, minimalno 16 bitnim procesorjem in minimalno 192 KB delovnega pomnilnika (RAM). CDC vsebuje bogatejšo

podmnožico knjižnic kot CLDC, saj vsebuje mnogo paketov razredov, kot jih poznamo iz Java SE (java.lang, java.io, java.net, java.security, java.text, java.util). Namenjen je zmogljivejšim napravam (dlančniki, tv komunikatorji, avtomobilske navigacijske naprave …) z minimalno 32-bitnim procesorjem, 2 MB RAM in 2.5 MB ROM. Prevedena koda (.class datoteka) se izvaja v JVM kot pri Java SE. V kombinaciji s konfiguracijama je mogoče veliko profilov in še več paketov. Najbolj znan je profil »Mobile Information Device Profile« (MIDP), ki je namenjen uporabi v mobilnih telefonih. Definira več vmesnikov, ki nam omogočajo razvoj MIDletov; aplikacij, ki tečejo nad CLDC. MIDP omogoča uporabo naslednjih funkcionalnosti:

- mrežno povezovanje prek standardnih komunikacijskih protokolov,
- shranjevanje podatkov na pomnilnik mobilnega telefona,
- uporabo multimedijskih funkcij (predvajanje zvoka, prikaz slik, …),
- uporabo časovnikov,
- gradnjo grafičnega uporabniškega vmesnika (GUI),
- možnost kriptirane mrežne povezave preko HTTPS,
- OTA (ang. over the air) nalaganje aplikacij.

V MIDP so definirana tudi dovoljenja, s katerimi MIDlet lahko dostopa do določenega omejenega programskega vmesnika. Dovoljenje je v bistvu ime java paketa, ki ga potrebujemo za pravilno delovanje naše aplikacije (na primer za uporabo mrežne povezave mora imeti MIDlet »javax.microedition.io.Connector.socket.« dovoljenje). Vsa dovoljenja morajo biti vpisana v opisni JAD datoteki (več o tem v nadaljevanju).

MIDleti so lahko v treh stanjih: pavzi (ang. paused), aktivni (active) in uničeni (ang. destroyed). To se pogosto poimenuje tudi kot življenjski cikel MIDleta. Aplikacija v pavzi ne sprejema dogodkov od sistema in čaka na aktivacijo ali uničenje. Aktivna aplikacija je v interakciji z uporabnikom in sprejema sistemske dogodke. Uničena aplikacija je končala svoje izvajanje in čaka na pobiralca smeti (ang. garbage collector) virtualnega stroja. Prehod med stanji izvaja program za upravljanje z aplikacijami (ang. application management software ali AMS), katerega potek prikazuje slika 5.



Slika 5. Življenjski cikel MIDlet aplikacije.

Tipičen primer, ko AMS MIDlet prestavi iz aktivnega stanja v pavzo, je ob sprejemu klica ali sprejemu SMS-sporočila.

Za razvoj aplikacij za telefon Nokia 6212 classic potrebujemo Series 40 Nokia 6212 NFC SDK [10], ki vsebuje emulator mobilnega telefona Nokia 6212, programske vmesnike (API), vzorce izvorne kode in tekstovne datoteke s pomočjo in primeri.

Za hitrejše delo sem namestil še Eclipse ME, vtičnik za razvojno okolje Eclipse. Ta omogoča razhroščevanje, prevajanje in zagon naših aplikacij v emulatorju neposredno iz Eclipse.

Po končanem razvoju nam Eclipse ME omogoča tudi pakiranje programa v datoteki s končnico .jar (arhiv .class datotek in ostalih virov kot slike, zvoki, …) in .jad (opis aplikacije) ter digitalni podpis aplikacije, s čimer se zagotovi določena stopnja zaupanja v ta program. V .jad datoteki najdemo ime aplikacije, ikono, ime razvijalca in dovoljenja za uporabo nekaterih programskih vmesnikov. V jad datoteki so atributi zapisani kot pari »ime : vrednost«.

Primer jad datoteke:

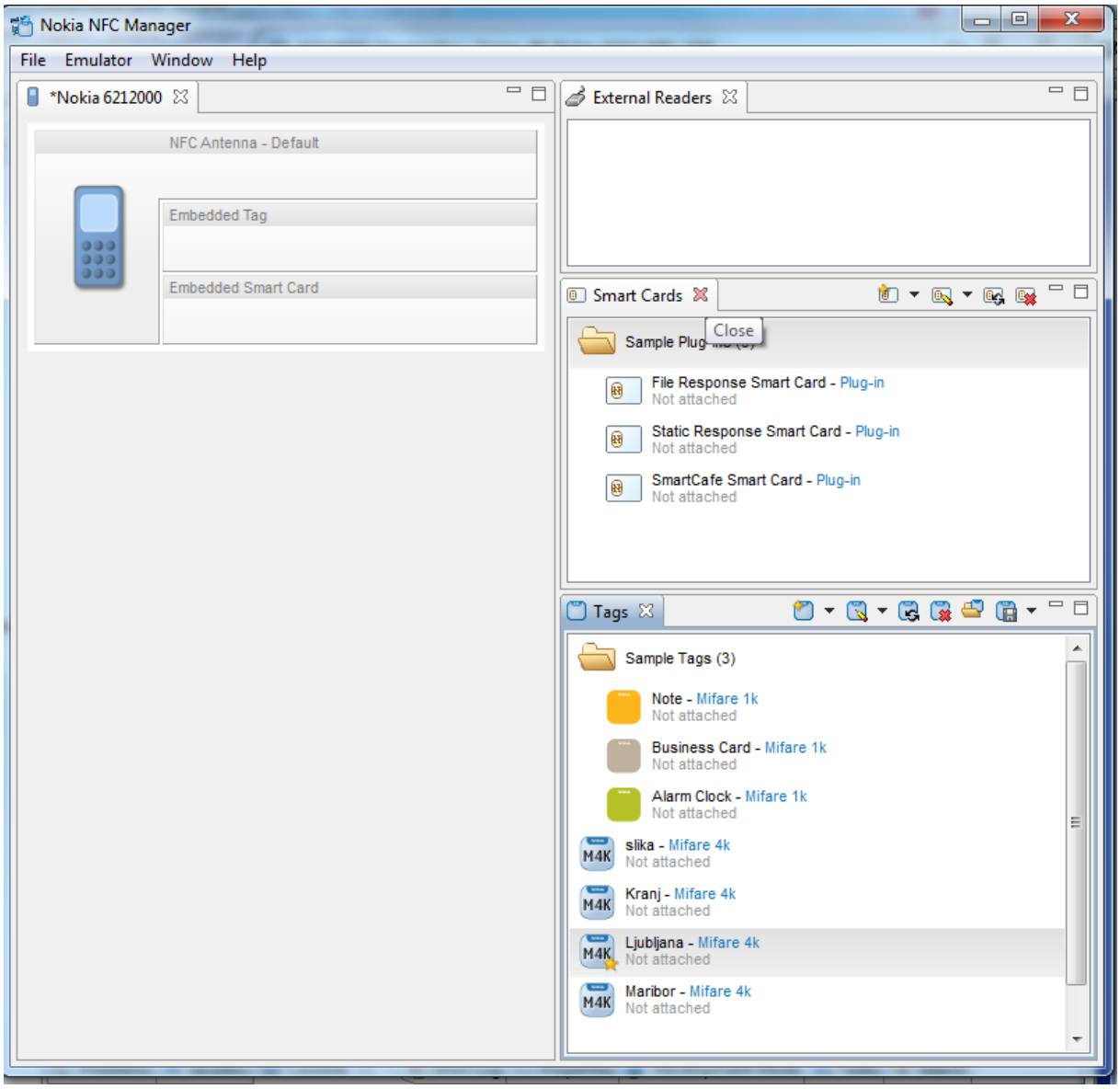
MIDlet-1: Weather,,com.apress.rischpater.weatherwidget.WeatherWidget
MIDlet-Jar-Size: 2858
MIDlet-Jar-URL: WeatherWidget.jar
MIDlet-Name: WeatherWidget
MIDlet-Vendor: Ray Rischpater
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.1

Eclipse ME nam omogoča urejanje jad datoteke prek grafičnega vmesnika, tako da ni potrebno spreminjati tekstovnih datotek. Razvoj in testiranje aplikacije lahko izvedemo skorajda brez mobilnega telefona, saj skupaj z Series 40 Nokia 6212 NFC SDK dobimo tudi program Nokia NFC manager. Ta omogoča komunikacijo med Nokia 6212 emulatorjem in zunanjim NFC čitalcem (ali emulatorjem značk). Poleg tega je mogoče emulirati več vrst RFID značk, kar nam s programom na emulatorju mobilnega telefona omogoča branje emuliranih značk (za razvoj tako ne potrebujemo ne pravega telefona ne pravih značk).
Podprta je simulacija naslednjih tipov značk:

- NFC Forum Type 1,
- NFC Forum Type 2,
- Mifare Standard 1k,
- Mifare Standard 4k.

Vsebino teh značk je mogoče spreminjati v posebnem urejevalniku, kjer lahko dodajamo, brišemo in formatiramo njihovo vsebino.
Emulator lahko simulira tudi prihodne klice, izgubo GSM-signala, beleži statistiko prejetih in poslanih podatkov, sprejetih in poslanih SMS-sporočil ter celotne Bluetooth komunikacije. Uporaba emulatorja mobilnega telefona in programa Nokia NFC Manager je prikazana na slikah 6 in 7.

Slika 6. Emulator Nokie 6212.



Slika 7. Emulator brezkontaktnih značk Nokia NFC Manager.

3.3 Google Web Toolkit

Google Web Toolkit (GWT) je zbirka odprtokodnih orodij, ki spletnim razvijalcem omogočajo gradnjo in vzdrževanje kompleksnih AJAX spletnih aplikacij v programskem jeziku Java [11]. To prinaša mnogo prednosti, saj je trenutno na voljo veliko dobrih orodij za razvoj v Javi. Naše delo tako postane podobno razvoju namiznih aplikacij, lažje je odkrivanje napak v kodi in poveča se število programerjev spletnih aplikacij (na svetu je več programerjev z znanjem Jave kot tistih z znanjem JavaScripta). Velika prednost je tudi pisanje kode v istem jezika za odjemalca in za strežnik (pisanje »servleta« z Javo EE).

GWT SDK ponuja javanske programske vmesnike, s katerimi lahko zgradimo logiko in grafični del naše aplikacije. GWT prevajalnik nato kodo prevede v več JavaScript datotek, pri čemer se uporabi optimizacija. Odstrani se neuporabljena koda, klici metod se menjajo s kodo teh metod (ang. method inlining). Na koncu dobimo JavaScript kodo, ki deluje v vseh

pomembnejših brskalnikih. Tako odpade veliko testiranja v različnih spletnih brskalnikih. Tipičen postopek razvoja z Google GWT je prikazan na sliki 8.

Slika 8. Osnove razvoja spletnih aplikacij z GWT.

Razvoj z GWT je najbolj preprost z uporabo vtičnika za Eclipse, saj nam ta omogoča ustvarjanje projektov, zagon GWT prevajalnika, označevanje napačne sintakse itd. Pri izdelavi novega projekta se ustvari tudi HTML datoteka ter stilske datoteke CSS, ki definirajo videz HTML elementov.

GWT aplikacije lahko zaženemo v razvojnem in spletnem načinu. V razvojnem načinu se koda ne prevaja v JavaScript, ampak se prevedena Java koda izvaja v javanskem virtualnem stroju in pošilja podatke za prikaz prek vtičnika brskalniku Google Chrome. Na ta način lahko uporabljamo vse možnosti razhroščevalnika za iskanje napak v kodi za odjemalca in v kodi, ki bo tekla na strežniku.

Ko smo zadovoljni z delovanjem naše spletne aplikacije v razvojnem načinu, jo po navadi zaženemo v spletnem načinu, s čimer se požene prevajalnik in ustvari optimizirane JavaScript datoteke. S tem je naša aplikacija pripravljena za prenos na strežnik. Zagon GWT aplikacije sestavljajo naslednji koraki:

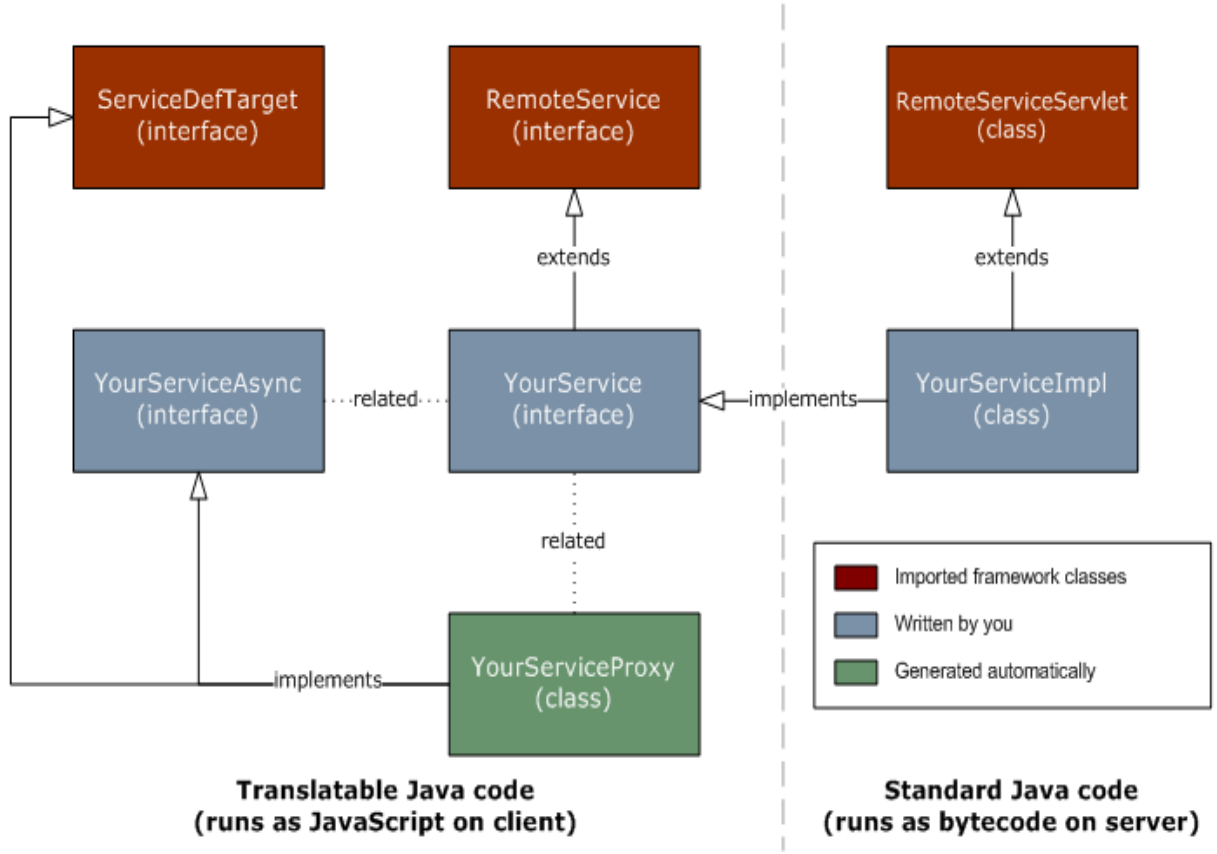
1. Brskalnik naloži HTML datoteko.
2. Ko brskalnik znotraj HTML datoteke najde <script src="<ime modula>.nocache.js"> oznako, k sebi prenese JavaScript kodo in jo izvede.
3. Datoteka nocache.js vsebuje kodo, ki med drugim zazna vrsto brskalnika in s pomočjo iskalne tabele, ki jo je naredil GWT prevajalnik, izbere ustrezno .cache.HTML datoteko.
4. JavaScript koda v nocache.js datoteki zgradi iframe (inline frame) HTML gradnik in ga vstavi v DOM gostujoče strani, v iframe pa vstavi .cache.html datoteko.
5. Datoteka .cache.html vsebuje programsko logiko GWT aplikacije.

Vsaka resna spletna aplikacija na določeni točki potrebuje komunikacijo s strežnikom (na primer pri branju podatkov iz podatkovne baze za prikaz podatkov na spletni strani). Pri GWT za komunikacijo s strežnikom prek HTTP uporabljamo GWT RPC. Implementacija klicev strežniške kode temelji na »servlet« tehnologiji. GWT RPC dovoljuje pošiljanje Java objektov med odjemalcem in strežnikom, pri čemer vsa serializacija (pretvorba objektov v zaporedje bitov, ki jih nato lahko shranimo v pomnilniku ali pošljemo preko omrežja) poteka avtomatsko. Tu je treba opozoriti, da ne gre za serializacijo, kot smo jo vajeni v Javi, ampak za posebno GWT izvedbo. GWT RPC klici so asinhroni (neblokirajoči), kar nam zagotavlja odzivnost aplikacije tudi po klicu strežniške metode.

Za klic strežniške metode je treba definirati tri stvari:

- vmesnik naše storitve, ki deduje od RemoteService in ima prototipe vseh RPC metod,
- razred, ki implementira strežniško kodo in deduje od RemoteServiceServlet ter implementira vmesnik iz zgornje točke ,
- asinhron vmesnik, ki se ga kliče iz odjemalčeve kode.

Slika 9 prikazuje povezave med vmesniki in razredi, ki so potrebni pri GWT RPC.



Slika 9. Potrebni vmesniki in razredi za uporabo GWT RPC mehanizma.

3.4 Servlet

Osnovna naloga spletnih strežnikov je pošiljanje statičnih dokumentov (spletnih strani) na zahtevo odjemalcev. Če želimo ponuditi dinamične vsebine, kjer se dokument, poslan odjemalcu, razlikuje glede na parametre zahteve, potrebujemo določeno aplikacijo, ki je v pomoč našemu spletnemu strežniku. Spletni strežnik lahko zahtevo posreduje tej aplikaciji, ki izračuna vse potrebno in strežniku vrne ustvarjen dokument, ta pa ga nato vrne odjemalcu. Aplikaciji, ki je v pomoč spletnemu strežniku, so včasih rekli CGI (ang. Common Gateway Interface) skripta. Navadno je bila napisana v programskem jeziku Perl. CGI skripte so imele veliko pomanjkljivosti, med najbolj motečimi sta bili platformna odvisnost in slaba skalabilnost. To je tudi razlog, zakaj je bila ustvarjena Java Servlet tehnologija.

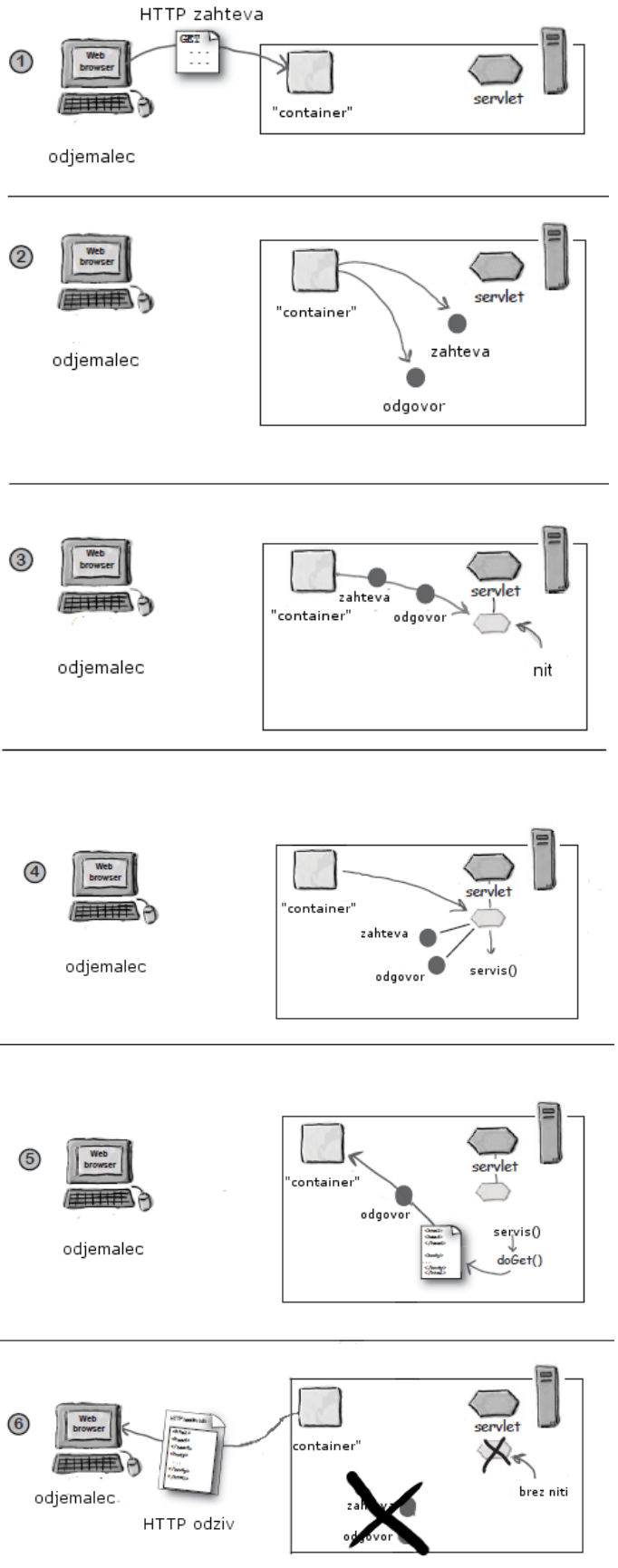
Servlet je razred v programskem jeziku Java, skladen s programskim vmesnikom Java Servlet (Java Servlet API). Gre za protokol, po katerem lahko Java razredi odgovarjajo na HTTP zahteve [12].

Paketa javax.servlet and javax.servlet.http vsebujeta vmesnike in razrede za pisanje »servletov«. Vsak »servlet«, ki ga programiramo, mora implementirati javanski vmesnik servlet.

»Servleti« za svoje delovanje potrebujejo »container«, ki je prav tako napisan v Javi. Tipičen primer »containerja« sta Tomcat in Jetty. Ko spletni strežnik dobi zahtevo za »servlet« (v nasprotju z zahtevo po določeni statični strani), strežnik to zahtevo posreduje »containerju«, šele zadnji pa kliče tipične servlet metode, na primer »doGet« ali »doPost«. »Container« je kot dodaten člen koristen, saj za nas ureja življenjski cikel servletov (nalaganje razredov, instanciranje objektov), nudi podporo večnitenju (za vsako zahtevo se ustvari nova nit) in skrbi za vso komunikacijo med »servleti« in spletnim strežnikom. Tako ni treba več pisati vtičnic in odpirati tokove za pisanje datotek.

Tipičen potek od prispele zahteve po »servletu« do odgovora odjemalcu (glej tudi sliko 10) je:

1. Uporabnik klikne na povezavo, ki kaže na »servlet« namesto na statično HTML stran.
2. »Container« vidi, da gre za zahtevo po »servletu« in naredi dva objekta, HttpServletResponse in HttpServletRequest.
3. »Container« najde ustrezen servlet glede na URL zahteve, ustvari nit za to zahtevo in niti »servleta« preda objekt zahteve in objekt odgovora.
4. »Container« pokliče metodo servis() znotraj »servleta«. Servis() metoda nato glede na tip zahteve pokliče metodo doGet() ali doPost(). Za ta primer predpostavimo, da gre za HTTP GET zahtevo.
5. Metoda doGet() ustvari dinamično stran in jo vstavi v objekt odgovora.
6. Nit se konča, »container« spremeni objekt odgovora v HTTP odgovor in ga pošlje odjemalcu. Nato se zbrišeta še objekta zahteve in odgovora.



Slika 10. Potek zahteve po dinamični strani pri »servletih«.

Da »container« izbere pravilen »servlet«, pogleda v datoteko web.xml. Tam so zapisi podani v naslednji obliki:

```
<servlet-mapping>
<servlet-name>catalogBrowse</servlet-name>
<url-pattern>/Catalog/Various/</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>catalogSearch</servlet-name>
<url-pattern>/Catalog/search/*</url-pattern>
</servlet-mapping>
```

Če pride zahteva z URL naslovom »www.primer.com/Catalog/search?item=coat«, bi »container« glede na zgoraj zapisano web.xml datoteko izbral »servlet« z imenom catalogSearch.

3.5 MySQL

MySQL je sistem za upravljanje s podatkovnimi bazami [13]. Natančneje gre za odprtokodno implementacijo relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. Napisana je v programskem jeziku C in C++. Deluje na več platformah, vključno z Microsoft Windows, FreeBSD, HP-UX, i5/OS, Linux, Mac OS X, NetBSD, Novell NetWare, OpenBSD, OS/2 Warp, Solaris, Symbian, SunOS,SCO UnixWare, Sanos in Tru64.

Za veliko večino programskih jezikov obstajajo knjižnice, s katerimi lahko dostopamo do MySQL baze neposredno iz programske kode. Administracija baze se tipično vrši prek ukazne vrstice, lahko pa se uporabi tudi kakšen grafični vmesnik, ne primer PhpMyAdmin ali MySQL Workbench, ki ga priporočajo MySQL razvijalci. MySQL Workbench uporabniku poleg administracije omogoča tudi grafično načrtovanje in modeliranje podatkovne baze.

3.6 Apache Tomcat

Apache Tomcat ali krajše Tomcat je odprtokodni »servlet container«, ki ga je razvila Apache Software Foundation (ASF) [14]. Tomcat je skladen z Java Servlet in JavaServerPages specifikacijami podjetja Sun Microsystems in ponuja *čisti* Java HTTP spletni strežnik za poganjanje Java kode.

Konfiguracija strežnika poteka prek XML datotek, lahko pa uporabimo tudi namenska orodja, ki so v paketu skupaj s strežnikom.

Razvoj Apache programske opreme in pomoč uporabnikom potekata v glavnem prek dveh poštnih seznamov. Na razvijalskem poštnem seznamu se razpravlja o poteku razvoja in testiranja, medtem ko se na uporabniškem poštnem seznamu uporabniki obračajo po pomoč k razvijalcem in drugim uporabnikom. V primeru pomoči sta nepogrešljivi tudi spletni strani Tomcatexpert.com in Apache Tomcat Resource Cente.

3.7 Radiofrekvenčna identifikacija RFID

Pojem radiofrekvenčne identifikacije (ang. Radio Frequency Identification ali RFID) se uporablja za opis sistemov, ki brezžično prenašajo identiteto določenega objekta ali osebe z uporabo radijskih valov [15].

Tipičen RFID sistem je sestavljen iz RFID značke (ang. RFID tag), čitalca in programske opreme, ki prebrano oznako pretvori v uporabniku koristno informacijo. RFID značka je pritrjena na objekt, ki ga želimo identificirati. Prikaz takega sistema je na sliki 11.

Slika 11. Prikaz osnovnega RFID sistema.

Večina RFID značk je sestavljena iz dveh delov. Prvi del je integrirano vezje za hranjenje podatkov, procesiranje, modulacijo in demodulacijo radijskih signalov ter ostale specializirane funkcije. Obstaja več različic pomnilnika, ki omogočajo večkratno branje in pisanje (ang. read-write), samo branje (ang. read-only) in enkratno pisanje z večkratnim branjem (ang. write once, read many ali WORM). Drugi del je antena za sprejem in oddajo radijskega signala.

V splošnem poznamo dve vrsti RFID značk, aktivne ter pasivne:

a) Aktivne RFID značke vsebujejo baterijo in so sposobne samostojnega oddajanja (za svoje delovanje ne potrebujejo zunanjega vira elektromagnetnega valovanja-radijskih valov). Prisotnost baterije poveča velikost značke, a jim obenem omogoča večjo moč oddajanja in s tem večji domet signala ter posledično zanesljivejše delovanje v bližini materialov, ki radi ovirajo radijsko komunikacijo (tekočine, kovine, …).
b) Pasivne RFID značke so brez lastnega vira napajanja in za svoj vklop in delovanje potrebujejo zunanji vir elektromagnetnega valovanja (radijske valove). Čitalec s svojim radijskim signalom inducira električni tok in s tem *zbudi* RFID značko, ki nato pošlje svoje podatke čitalcu.

RFID sistemi tipično delujejo v štirih frekvenčnih območjih: nizkofrekvenčnih (LF, 135kHz), visokofrekvenčnih (HF, 13,56MHz), ultravisokofrekvenčnih (UHF, 868MHz) in v področju mikrovalov (5.8GHz). Od področja delovanja je odvisen domet čitalca, kar je prikazano v tabeli 1.

Frekvenca	Maksimalen domet za pasivne RFID značke	Področje uporabe
LF	50 cm	Identifikacija hišnih ljubljenčkov
HF	Do 3 m	Kontrola dostopa v zgradbe
UHF	9 m	Sledenje palet, škatel
MIKROVALOVI	>10 m	Identifikacija vozil

Tabela 1. Domet RFID čitalcev glede na frekvenco delovanja.

RFID značke imajo svojo serijsko ali identifikacijsko številko (ang. Unique Identifier ali UID). UID je po navadi dolžine štirih ali sedmih bajtov [15].

Z razvojem polprevodniške industrije in s stalnim pomanjševanjem elektronskih elementov se ves čas nižajo cene in dimenzije RFID značk. Skladno s tem se širi tudi področje uporabe. RFID značke se trenutno uporabljajo na naslednjih področjih:

- Plačilo cestnine.
- Plačilo javnega prevoza.
- Sledenje produktom.
- Transport in logistika.
- Sledenje živalim.
- Označevanje inventarja v operacijski sobi (na primer za preprečevanje tragičnih dogodkov, ko se del inventarja med operacijo pozabi v pacientu).
- Nadomeščanje črtnih kod v trgovini, knjižnici.
- Potni listi (shranjevanje biometričnih podatkov na RFID značke).
- Sledenje ogroženim ljudem (npr. državnim tožilcem)

Z miniaturizacijo in padanjem cen se v prihodnosti napovedujejo ogromna brezžična senzorska omrežja z več tisoč vozlišči (*internet stvari*, ang. internet of things), ki bi pošiljala podatke o temperaturi, vlagi, potresnih sunkih, nevarnih plinih, pretoku vode, vodnem pritisku itd. ter na ta način lahko preprečila marsikatero nesrečo [16, 17].

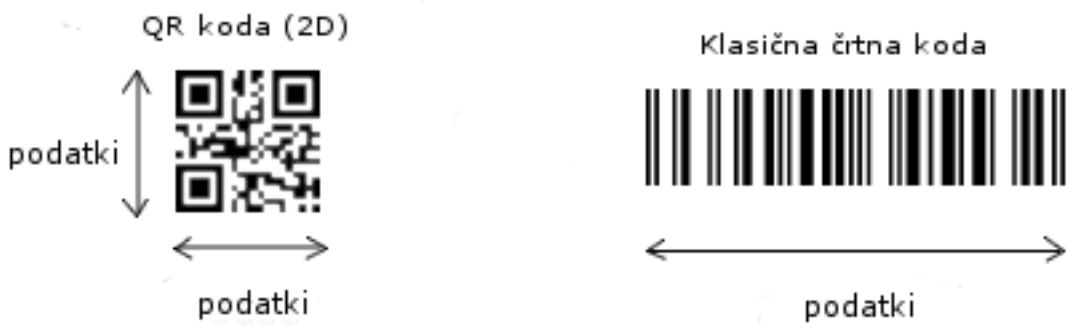
Kljub velikim prednostim uporabe RFID tehnologije pa ta s seboj prinaša tudi skrbi zaradi varnosti (podjetij, države) in zasebnosti.

Komunikacija v bližnjem polju NFC (ang. Near Field Communication) je sorodna tehnologiji RFID. NFC ima domet komunikacije do 20cm, kar zagotavlja višjo stopnjo varnosti pri npr. denarnih transakcijah in jo dela zelo primereno za uporabo v gneči. NFC je kompatibilen z obstoječo RFID (13.56 MHz ISO/IEC 18000-3) infrastrukturo. Omogoča zelo kratek čas vzpostavitve povezave (<0.1s). Hitrost prenosa podatkov je maksimalno 424 kbit/s. NFC tehnologija se trenutno največ uporablja v sistemih za plačevanje manjših zneskov ter v sistemih javnega prevoza, kjer mobilni telefoni z NFC modulom nadomeščajo pametne kartice.

3.8 QR črtne kode

QR (ang. quick response) koda je dvodimenzionalna črtna koda, ki jo lahko beremo z namenskimi QR čitalci in s kamero na mobilnih telefonih [18]. Razvita je bila leta 1994 v Toyotinem hčerinskem podjetju Denso Wave. Podjetje je želelo izdelati kodo, ki jo bo mogoče čim hitreje prebrati. Koda je standardizirana pri mnogih svetovnih standardnih organizacijah, med drugim je pridobila tudi mednarodni standard ISO/IEC18004.

Podatki so zapisani v horizontalni in vertikalni smeri (pri klasični črtni kodi le v eni smeri), kar omogoča veliko večjo gostoto zapisanih podatkov. Primerjava QR in običajne 1D črtne kode je na sliki 12.



Slika 12. Primerjava QR in 1D črtne kode.

QR kodo sestavljajo črne in bele točke, imenovane moduli. Velikost in število modulov določata velikost QR kode in količino podatkov, ki jih je mogoče shraniti vanjo (maksimalno 4296 alfanumeričnih znakov). Branje kode je mogoče iz katerekoli strani, saj se za pozicioniranje uporabljajo simboli na treh robovih. QR kode za korekcijo napak uporabljajo postopek imenovan Reed–Solomon, kar omogoča uspešno branje kljub zamazanosti ali delni uničenosti kode (možnost uporabe v umazanih okoljih, na primer v halah, delavnicah).

Najprej so se kode uporabljale za sledenje avtomobilskim delom med proizvodnjo, danes pa v njih po navadi shranjujemo URL naslove, vizitke, koledarske dogodke, geografske lokacije in podatke o WI-FI omrežju, ki jih potrebujemo za uspešno povezavo.

Za skorajda vse današnje mobilne telefone s kamerami obstajajo aplikacije, s katerimi lahko beremo QR črtne kode. Poleg tega na spletu najdemo tudi strani, na katerih lahko sami ustvarimo kodo poljubne velikosti s poljubno vsebino.

4 Implementacija elektronske vozovnice

V tem poglavju so opisane tehnične podrobnosti vseh razvitih aplikacij. Pomembnejši algoritmi so za lažje razumevanje predstavljeni v psevdokodi ali z diagramom poteka. Poleg tega je prikazanih tudi nekaj zaslonskih mask razvitih aplikacij, s čimer si bralec lahko predstavlja tipičen način uporabe.

4.1 Aplikacija za mobilni telefon z OS Android

Aplikacija je sestavljena iz petih aktivnosti (ang. activity): Kartar, Vstop, Izstop, Info ter Preveri. Java razred Activity (od katerega deduje vsaka aktivnost) skrbi za gradnjo okna, v katerega lahko položimo elemente grafičnega vmesnika in prek katerega poteka vsa interakcija z uporabnikom. Grafični vmesnik vseh aktivnosti je opisan v XML datotekah (s tem je dosežena ločitev videza aplikacije in logike). V nadaljevanju je opisano delovanje vseh petih aktivnosti.

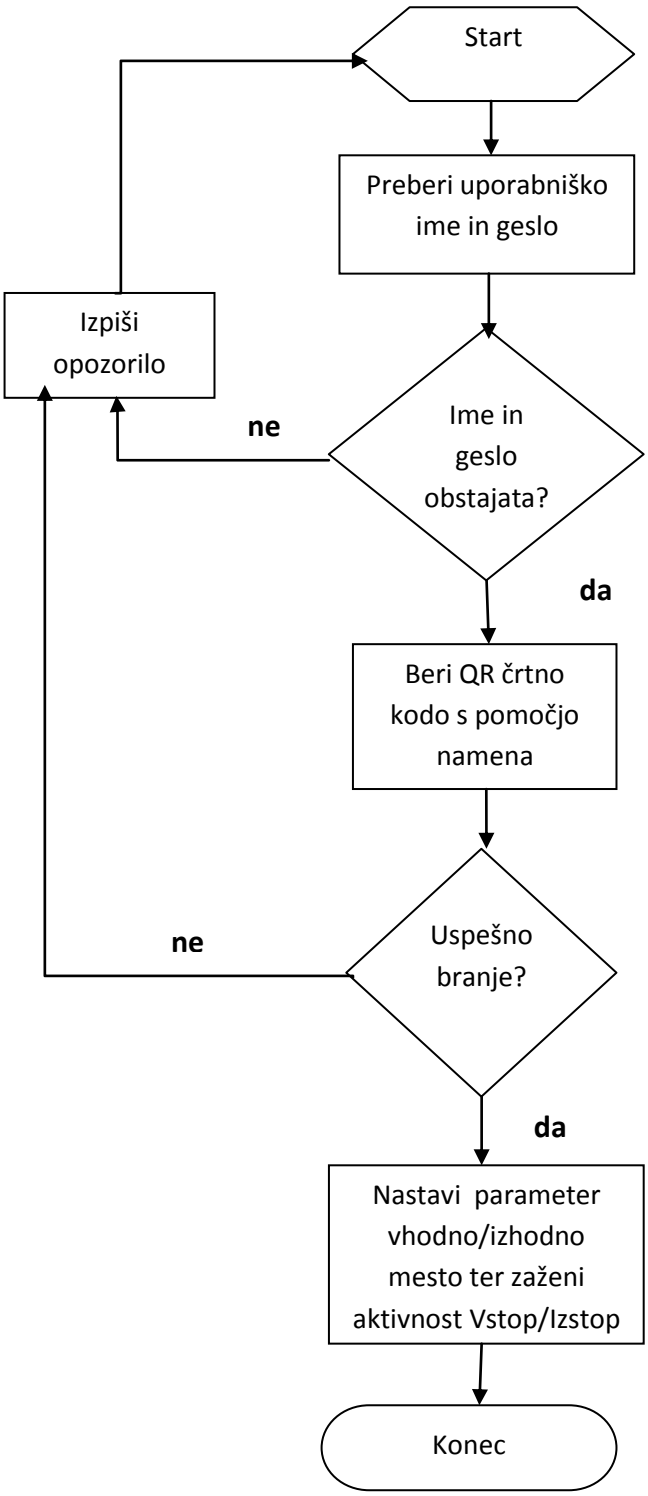
4.1.1 Aktivnost Kartar (uvodni zaslon aplikacije)

Ta aktivnost se pojavi takoj po zagonu aplikacije. Grafični vmesnik je sestavljen iz več gumbov, pri čemer je gumb za zagon preverjanja vozovnic skrit s pomočjo atributa »android:visibility="gone"« v datoteki main.xml, ki opisuje grafični vmesnik. Gumb postane viden, če v nastavitvah vpišemo administratorsko uporabniško ime in geslo (to je izvedeno s klicem metode setVisibility(View.VISIBLE)). Na sliki 13 je prikazan grafični vmesnik, ki uporabniku prek gumbov omogoča izvedbo ostalih aktivnosti.

Slika 13. Osnovna zaslonska maska.

Uporabnik lahko prek gumbov zaganja ostale aktivnosti. Za vsak gumb je nastavljen poslušalec, ki ob kliku nanj zažene izbrano aktivnost (vstop, izstop, informacije, preverjanje vozovnice) s klicem metode »startActivityForResult«. Ta metoda po koncu izvajanja vedno kliče metodo »onActivityResult«, v kateri lahko obdelamo podatke, ki nam jih je vrnila klicana metoda. Pritisku na gumb vstop ali izstop sledi postopek, ki je opisan na sliki 14.

Program uporabniku ne dovoli novega vstopa ali izstopa, če pred tem še ni izstopil oziroma vstopil.



Slika 14. Diagram poteka procedure vstopa, izstopa.

Branje QR črtne kode poteka z uporabo ZXing knjižnice (ang. zebra crossing) [19], ki je del aplikacije Barcode Scanner (ta mora biti nameščena na mobilnem telefonu). Zxing je odprtokodna knjižnica v programskem jeziku Java, ki podpira branje različnih formatov črtnih kod, tako 1D kot 2D. Branje črtne kode poženemo z uporabo namena

»com.google.zxing.client.android.SCAN«. V osnovi se klic aktivnosti, ki začne brati črtno kodo, izvede z naslednjo programsko kodo:

```
public Button.OnClickListener mScan = new Button.OnClickListener() {
public void onClick(View v) {
        Intent intent = new Intent("com.google.zxing.client.android.SCAN");
        intent.putExtra("SCAN_MODE", "QR_CODE_MODE"); //izberemo format kode
         startActivityForResult(intent, 0);
         }
};
```

4.1.2 Aktivnost Vstop

Aktivnost se zažene takoj po uspešnem branje QR črtne kode. V tej aktivnosti je najpomembnejša metoda »dobiKarto(final String mesto)«. Ta metoda naredi HTTP GET zahtevo na strežnik, kjer teče aplikacija za izdelavo in pošiljanje vozovnic. Parametri te zahteve so uporabniško ime (spremenljivka user v kodi), geslo (pass), mesto vstopa, tip ukaza (ukaz) ter UID RFID značke (strežniška aplikacija tega podatka pri zahtevah Android odjemalcev ne uporabi). URL naslov HTTP GET zahteve je naslednje oblike:

```
URLnaslov=newURL("http://lukaf.mine.nu:8080/webui/karta?user="+uporabnik+"&mesto="+mesto+
"&pass= "+geslo+"&ukaz=1&uid=0");
```

Zahteva se naredi v svoji niti, s čimer grafični vmesnik ostane odziven. Komunikacija med nitmi poteka z uporabo objekta razreda Message, v katerega se spravi sporočilo, namenjeno glavni niti. Glavna nit ta sporočila prebere v metodi »handleMessage(Message msg)«. Uporabniku se prikaže tudi pogovorno okno, ki ga obvesti, da poteka pridobivanje vozovnice. Nastavi se tudi čas, po katerem se v primeru, da ni odziva strežnika, proži izjema tipa »SocketTimeoutException«. Ob odzivu strežnika na našo zahtevo pa metoda »dobiKarto« deluje po naslednjem algoritmu, napisanem v psevdokodi:

```
If ( vrsta odgovora je tekst kodnega nabora UTF-8) //gre za napako, ker ne dobimo vstopnice
        preberiTekst();
        naredi SporočiloZaGlavnoNit();
else if (vrsta odgovora je slika formata PNG)
        preberiSliko();
        naredi SporočiloZaGlavnoNit();
        zabeleži ČasVstopa();
```

Strežnik vrsto odgovora sporoči v glavi HTTP odgovora kot »Content-Type«, znano tudi kot MIME tip (ang. MIME type). MIME tip uporabljajo brskalniki, da vedo, kako prikazati sprejete podatke. Ko glavna nit v primeru uspešnega sprejetja vstopnice prebere sporočilo delavske niti (ang. worker thread) v metodi »handleMessage«, se prikaže prejeto vozovnico in nastavi sporočilo v statusno vrstico mobilnega telefona (prikaz opozorila v statusni vrstici je na sliki 16). Na ta način se uporabnika ves čas opominja, da naj na izstopni postaji ne pozabi narediti odjave. Vstopnica se shrani tudi na telefonu, da jo uporabnik lahko pozneje prikaže sprevodniku med preverjanjem vozovnic. Na sliki 15 je prikazana zaslonska maska aplikacije ob pridobitvi vozovnice.

Slika 15. Zaslonska maska ob pridobitvi vozovnice.

Ko je uporabnik pridobil vozovnico, lahko s pritiskom na gumb pokliče metodo »dobiPrihode«, ki prikaže prihode naslednjih treh vlakov za uporabnikovo vstopno postajo. Ta metoda izvede HTTP GET zahtevo na strežnik z dvema parametroma, mestom vstopa in tipom ukaza (ukaz=5). URL naslov HTTP GET zahteve je naslednje oblike:

URL naslov=new URL("http://lukaf.mine.nu:8080/webui/karta?mesto="+*mesto*+"&ukaz=5");

HTTP GET zahteva je tudi tu narejena v svoji niti, prav tako se za komunikacijo med nitmi uporablja objekt vrste Message.

Prihode iz strežnika dobimo v obliki »1 Ljubljana: 10:15:00/ 2 Maribor: 12:00:00 /3 Ljubljana: 14:32:15«. Zato se pred prikazom podatkov kliče še metoda »parsej«, ki podatke pripravi za primeren prikaz (glede na poševnico jih loči na več delov). Prikaz pridobljenih prihodnih časov je na sliki 16.

Slika 16. Prikaz prihodnih časov na mobilnem telefonu.

4.1.3 Aktivnost Izstop

Aktivnost se zažene po uspešnem branju QR črtne kode in uporabnika na strežniku odjavi na določeni postaji. HTTP GET zahteva s parametri uporabniško ime (spremenljivka user v kodi), geslo (pass), mestoizstopa (mesto), tip ukaza (ukaz=2), način izstopa (nacin izstopa) ter UID RFID značke (uid) se izvede v svoji niti. Parameter »nacinizstopa« ima lahko vrednost h (hitro) ali p (počasi), odvisno od tega, ali se je izstop zgodil manj kot 30 ali več kot 30 sekund po vstopu (vrednost 30s izbrana le za testiranje, v uporabi bi morala biti višja). Na ta način uporabnik lahko izstopi takoj po vstopu brez stroškov.

URL naslov HTTP GET zahteve je v primeru hitrega izstopa naslednje oblike:

```
URL naslov=new
URL("http://lukaf.mine.nu:8080/webui/karta?user="+uporabnik+"&mesto="+mesto+"&pass="+geslo
+"&ukaz=2&nacinizstopa=h&uid=0");
```

Po uspešnem izstopu se prebere še odziv strežnika, v katerem dobimo ceno opravljenega potovanja in jo uporabniku prikažemo tudi na zaslonu. Iz statusne vrstice telefona se odstrani obvestilo, poleg tega pa se uporabniku ponovno dovoli vstopiti.

4.1.4 Aktivnost Info

Če smo vstopili na vlak, se nam v tej aktivnosti prikažeta vozovnica in aktualna poraba (mesečna, skupna), sicer pa se prikaže le poraba. Vozovnica se bere s pomočjo metode »preberiKarto«, ki vozovnico v PNG formatu iz SD kartice prebere z naslednjo programsko kodo:

```
FileInputStream in = new FileInputStream("/sdcard/Karte/karta.png");
```

```
BufferedInputStream buf = new BufferedInputStream(in);
Bitmap img = BitmapFactory.decodeStream(buf);
```

Porabo se dobi iz strežnika s pomočjo HTTP GET zahteve s parametri uporabniško ime (spremenljivka user v kodi), geslo (pass) ter tip ukaza (ukaz=4). Zahteva je izvedena znotraj metode »dobiPorabo« kot nova programska nit. URL naslov HTTP GET zahteve je naslednje oblike:

```
URL naslov=new
URL("http://lukaf.mine.nu:8080/webui/karta?user="+uporabnik+"&pass="+geslo+"&ukaz=4");
```

Strežnik se odzove s tekstom kodnega nabora UTF-8, ki ga razčlenimo in prikažemo na zaslonu mobilnega telefona. Prikaz vozovnice in porabe je prikazan na sliki 17.

Slika 17. Prikaz aktivnih vozovnic.

4.1.5 Aktivnost Preveri

Aktivnost omogoča branje vozovnice v obliki QR črtne kode z zaslona drugega mobilnega telefona in preverjanje njene veljavnosti s klicem ustrezne metode na strežniku. Iz črtne kode se dobi uporabniško ime, čas vstopa in varnostno kodo, ki se potem kot parametri HTTP GET zahteve pošljejo na strežnik. URL naslov zahteve je:

```
URL naslov=new
URL("http://lukaf.mine.nu:8080/webui/karta?user="+user+"&koda="+koda+"&ukaz=3&datum="+cas
Vstopa);
```

To se naredi v metodi »preveriKarto«, ki po izvedeni HTTP GET zahtevi prebere odziv strežnika in izpiše, ali gre za veljavno ali neveljavno vozovnico. S tem se morebitnim goljufivim potnikom prepreči, da bi sprevodniku v lastni aplikaciji podobnega videza kazali črtne kode, ki bi jih ustvarili sami.

4.2 Aplikacija za mobilni telefon Nokia 6212 classic

MIDlet v datoteki Kartar.java zazna vstopno (izstopno) postajo z branjem RFID značke, nato pa prek strežnika pridobi ustrezno vozovnico. Aplikacija ob zagonu zgradi enostaven meni ter registrira poslušalca »CommandListener«, ki se na določene ukaze (nazaj, izhod, Ok) odziva v metodi »commandAction«. Kako je videti glavni meni, je prikazano na sliki 18.

Slika 18. Glavni meni aplikacije za Nokio 6212.

Upošteva se tudi dobro prakso programiranja MIDletov, saj se v metodi »pauseApp« vrednosti vseh zaslonskih mask (ang. forms) nastavi na null. Na ta način se sprostijo vsi sistemski viri.

4.2.1 Vstop in izstop

Pri izbiri vstopa in izstopa v glavnem meniju se zažene naslednji algoritem:

1. Registrira se poslušalca za določen tip NDEF (ang. NFC Data Exchange Format) zapisa znotraj RFID značk (text/plain; FORMAT=FIXED).
2. Zastavica (spremenljivka tipa boolean) vstopam (izstopam) se postavi na 1 (true).
3. Izpiši se poziv uporabniku, naj telefon približa RFID znački.
4. Ob zaznani RFID znački se proži metoda »targetDetected«, kjer shranimo UID značke.
5. Ob zaznanem NDEF zapisu se proži metoda »recordDetected«.
6. Odstrani se poslušalca za RFID značke in NDEF zapise.
7. Zastavica vstopam/izstopam se postavi na 0 (false).
8. V metodi »recordDetected« iz značke preberemo vstopno (izstopno) mesto, iz telefona preberemo uporabniško ime in geslo ter zgradimo URL naslov za HTTP GET zahtevo.
9. Izvedemo HTTP GET zahtevo za vstop (izstop).
10. Preberemo odziv strežnika, nato pa v primeru vstopa izrišemo vozovnico (slika 19), v primeru izstopa pa izpišemo ceno opravljenega potovanja.

Spletni naslov zahtev je enak kot pri aplikaciji za Android, le da se tu na strežnik poleg ostalih parametrov pošilja tudi UID prebrane RFID značke. Uporabnik ne more zagnati izstopa, če pred tem še ni vstopil. Opozori se ga z uporabo metode »showAlert«, ki izriše alarm (ang. alert). Alarm je okno, ki na zaslonu prikaže besedilo (po možnosti tudi sliko) in po nastavljenem času izgine.

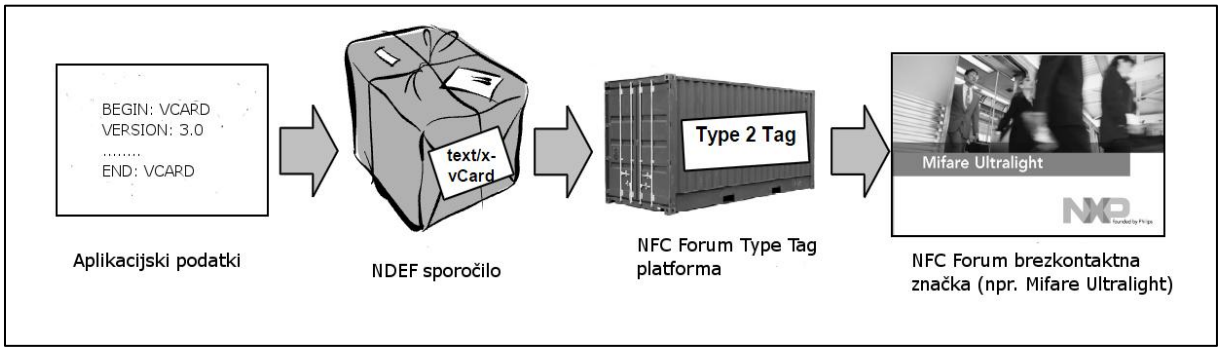
Za komunikacijo in izmenjavo podatkov z brezkontaktnimi značkami se uporablja programski vmesnik (API) za brezkontaktno komunikacijo JSR-257 (ang. The Contactless Communication API) [20]. Ta programski vmesnik sestavlja 5 Java paketov, ki so opisani v tabeli 2.

Java paket	Lastnosti
javax.microedition.contactless	Obvezen paket z razredi, ki so v uporabi pri vseh značkah.
javax.microedition.contactless.ndef	Izbirni paket z razredi za komunikacijo z značkami, ki vsebujejo podatke v NDEF formatu.
javax.microedition.contactless.rf	Izbirni paket z razredi za komunikacijo z RFID značkami, kjer pa podatki niso v formatu NDEF.
javax.microedition.contactless.sc	Izbirni paket z razredi za komunikacijo s pametnimi karticami.
javax.microedition.contactless.visual	Izbirni paket z razredi za branje in ustvarjanje črtnih kod.

Tabela 2. Seznam paketov, ki sestavljajo programski vmesnik za brezkontaktno komunikacijo.

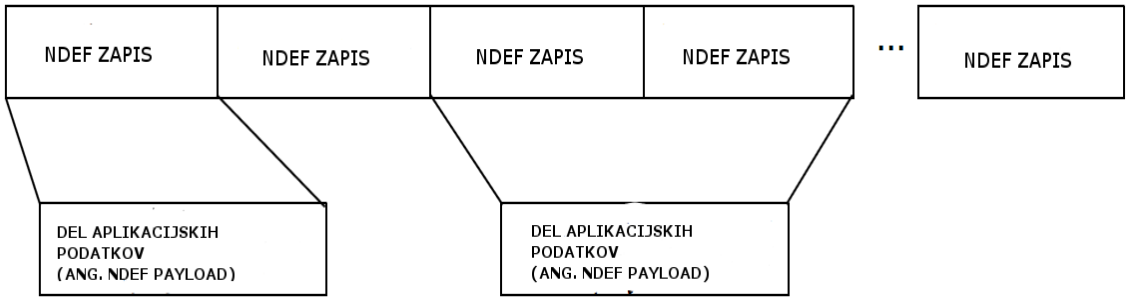
Slika 19. Prikaz pridobljene vozovnice.

Pred opisom NDEF formata je treba pogledati, kako so aplikacijski podatki shranjeni znotraj značk NFC Foruma. Podatki so najprej oviti (ang. encapsulated) v NDEF sporočilo, nato pa še v podatkovno strukturo, ki je določena s strani NFC Forum Type Tag Platforme [21]. Ti dve podatkovni ovijanji se uporabita za identifikacijo vrste podatkov (npr. URL, JPEG slikovna datoteka, v-Card …) ter za zagotovitev interoperabilnosti med različnimi aplikacijami. Postopek ovijanja je prikazan na sliki 20.



Slika 20. Podatkovno ovijanje aplikacijskih podatkov na značkah NFC Foruma.

NDEF format je binarni format, ki omogoča ovijanje aplikacijskih podatkov poljubnega tipa in poljubne dolžine (omejeno le s količino pomnilnika na brezkontaktni znački) v strukturo, imenovano NDEF sporočilo. NDEF sporočilo je lahko sestavljeno iz enega ali več NDEF zapisov (slika 21). NDEF je konceptualno zelo blizu MIME tipom.



Slika 21. Zgradba NDEF sporočila.

Poslušalca za NDEF sporočila določene vrste znotraj RFID značke se registrira z naslednjo kodo (za uporabo te kode mora razred implementirati vmesnika TargetListener in NDEFRecordListener):

```
DiscoveryManager dm = DiscoveryManager.getInstance();
// Registracija poslušalca za NDEF značke (mogoči so še ISO14443_CARD,   //RFID_TAG and
VISUAL_TAG)
 dm.addTargetListener(this, TargetType.NDEF_TAG);
 // ..dodaj plain text mime tip k NDEF poslušalcu
 dm.addNDEFRecordListener(this, new NDEFRecordType(
 NDEFRecordType.MIME, "text/plain; FORMAT=FIXED"));
```

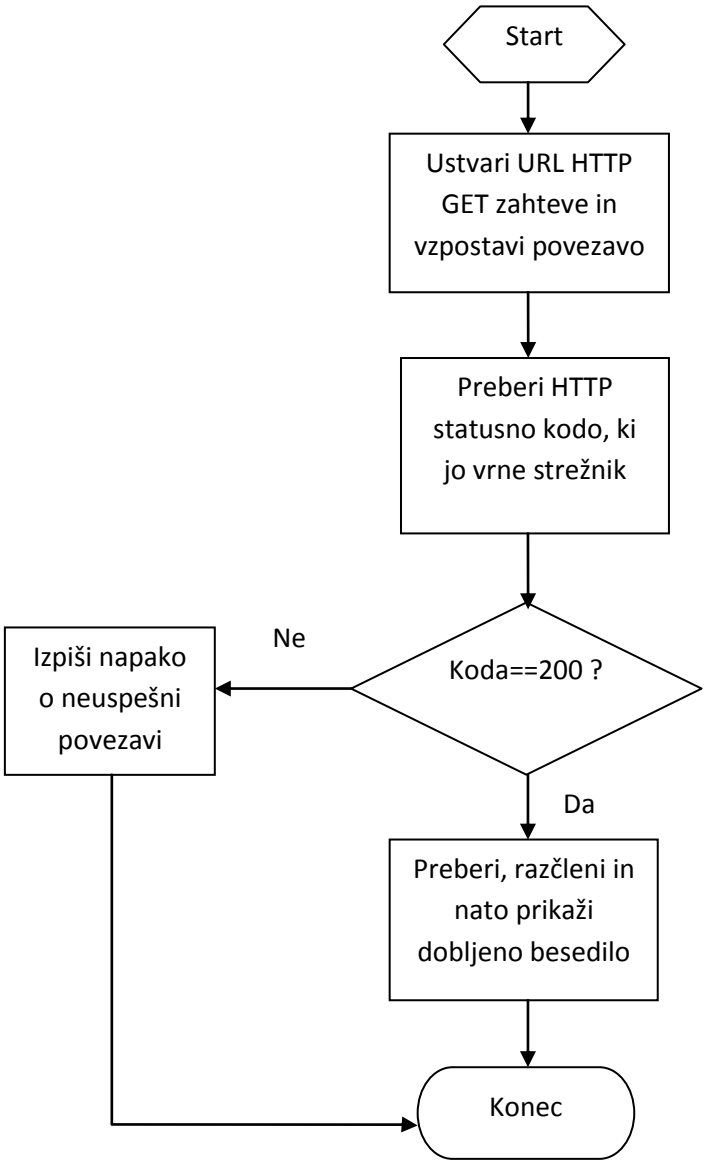
Odstrani pa se jih na naslednji način:

```
dm.removeTargetListener(this, TargetType.NDEF_TAG);
dm.removeNDEFRecordListener(this, new NDEFRecordType(
 NDEFRecordType.MIME, "text/plain; FORMAT=FIXED"));
```

Vozovnico po prejemu shranimo v obstojni pomnilnik mobilnega telefona, da jo uporabnik pozneje lahko pokaže sprevodniku. Shranjujeta se tudi uporabniško ime in geslo, ki se ga vpiše v nastavitvah. Za shranjevanje podatkov se pri MIDletih uporablja koncept skladišča zapisov (ang. record store). Skladišče zapisov shranjuje podatke podobne vrste in je sestavljeno iz zapisov (records). Podatki se shranjuje na obstojni pomnilnik (NVRAM, danes je v mobilnih telefonih to tipično flash pomnilnik). MIDlet lahko uporablja eno ali več skladišč zapisov, pri čemer mora imeti vsako unikatno ime. Skladišča zapisov moramo pred uporabo odpreti z metodo »openRecordStore("imeSkladišča", true)«. Nato lahko beremo zapise. Vsak zapis je dejansko le en ali več bajtov (podatkovne strukture mora programer implementirati sam). Ko v skladišče vstavimo zapis, ta dobi indeks 1. Ko končamo s shranjevanjem zapisov, pokličemo metodo »closeRecordStore« in s tem sprostimo dragocene strojne resurse.

Za shranjevanje vozovnice (slike) v moji aplikaciji skrbi metoda »writeRecord(byte [] b)«, ki uporablja skladišče zapisov z imenom *REC_STORE2*. Metoda poskrbi tudi za odpiranje in zapiranje skladišča zapisov. Za shranjevanje uporabniškega imena in gesla pa se uporablja metoda »writeRecord(String str)«, ki uporablja skladišče REC_STORE. Za njuno branje se uporabi metoda »readRecords«.

Po pridobitvi vozovnice lahko s pritiskom na sredinski potrditveni gumb zaženemo metodo »prihodi«, ki pridobi čase prihodov naslednjih treh vlakov. Algoritem v obliki diagrama poteka je prikazan na sliki 22.



Slika 22. Diagram poteka metode »prihodi«.

Ker v Javi ME ni na voljo vseh razredov kot v Javi SE, nisem mogel uporabiti metode »split« za razčlenjevanje nizov, sem zato napisal funkcijo »parsej«, ki en niz s poševnicami kot ločilom loči na več nizov.

4.2.2 Informacije

Pri izbire informacij v meniju se kliče metoda »informacije«, ki prikaže aktivne vozovnice in porabo (slika 23). Metoda deluje po naslednjih korakih:

1. Če ima uporabnik aktivno karto, se jo prebere iz skladišča zapisov in prikaže na zaslonu.
2. Naredi se HTTP GET zahtevo na strežnik in prebere vrnjeno HTTP statusno kodo.
3. Ob statusni kodi 200 (OK) se prebere odziv strežnika.
4. Odziv se razčleni z metodo »parsej«.
5. Na zaslonu se prikaže uporabnikova poraba.

Slika 23. Prikaz aktivnih kart in porabe.

4.2.3 Nastavitve

Nastavitve so namenjene vpisu uporabniškega imena in gesla v za to namenjena tekstovna polja (ang. textfield). Ob kliku na *Ok* se obe vrednosti vpišeta v skladišče zapisov, nato pa se spet prikaže glavni meni.

4.3 Aplikacije na strežniku

Na strežniku tečeta spletni aplikaciji za registracijo uporabnikov ter za administracijo. Poleg tega je na strežniku tudi »servlet«, ki odgovarja na HTTP zahteve odjemalcev. Najprej si bomo ogledali podatkovno bazo, saj jo uporabljajo vse aplikacije na strežniku.

4.3.1 Podatkovna baza

Uporabil sem sistem za upravljanje z relacijsko podatkovno bazo MySQL. Z orodjem MySQL Workbench sem ustvaril bazo z imenom *test*. V njem sem ustvaril tabele:

- »users« v kateri so vsi registrirani uporabniki;
- »potovanja« v kateri je seznam potovanj registriranih uporabnikov;
- »kr«, »lj«, «mb«, ki vsebujejo povezave določenega kraja z drugimi ter cene potovanj na teh relacijah;
- »krodhodi«, »ljodhodi«, «mbodhodi«, ki vsebujejo odhodne čase vlakov iz določene postaje;
- »mesečna«, ki vsebuje ceno mesečne vozovnice.

Najpomembnejši sta tabeli »users« in »potovanja«. Tabela »users« vsebuje stolpce: username (uporabniško ime, ki je hkrati tudi primarni ključ), password (geslo), ime, priimek, nacinp (način porabe), email, vrsta (ali gre za navadnega uporabnika ali administratorja), casregistracije (čas registracije v formatu POSIX – številu sekund od 1.1.1970), telefon (vrsta

mobilnega telefona). Še več podatkov o tabeli je razvidnih iz SQL ukaza, s katerim sem jo ustvaril:

```
CREATE TABLE `users` (
  `username` varchar(45) NOT NULL,
  `password` varchar(45) DEFAULT NULL,
  `ime` varchar(45) DEFAULT NULL,
  `priimek` varchar(45) DEFAULT NULL,
  `nacinp` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  `vrsta` int(11) DEFAULT '0',
  `casregistracije` int(11) DEFAULT NULL,
  `telefon` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Tabela »potovanja« vsebuje stolpce: idpoti (primarni ključ), od (vstopna postaja), do (izstopna postaja), datumvstopa (čas vstopa v POSIX formatu), username (uporabniško ime potnika), cena (cena opravljenega potovanja), datumvstopa (čas vstopa v POSIX formatu), koda (ustvarjena zaščitna koda za varovanje vozovnic), uidVstopa (UID RFID vstopne značke) ter uidIzstopa (UID RFID izstopne značke). Primarne in tuje ključe ter indekse se vidi iz SQL ukaza, s katerim je bila tabela narejena:

```
CREATE TABLE `potovanja` (
  `idpoti` int(11) NOT NULL AUTO_INCREMENT,
  `od` varchar(45) DEFAULT NULL,
  `do` varchar(45) DEFAULT NULL,
  `datumvstopa` int(10) DEFAULT NULL,
  `username` varchar(45) DEFAULT NULL,
  `cena` int(11) DEFAULT NULL,
  `datumizstopa` int(10) DEFAULT NULL,
  `koda` varchar(45) DEFAULT NULL,
  `uidVstopa` varchar(45) DEFAULT NULL,
  `uidIzstopa` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idpoti`),
  KEY `usr_ind` (`username`),
  CONSTRAINT `usr_ind` FOREIGN KEY (`username`) REFERENCES `users` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=180 DEFAULT CHARSET=utf8
```

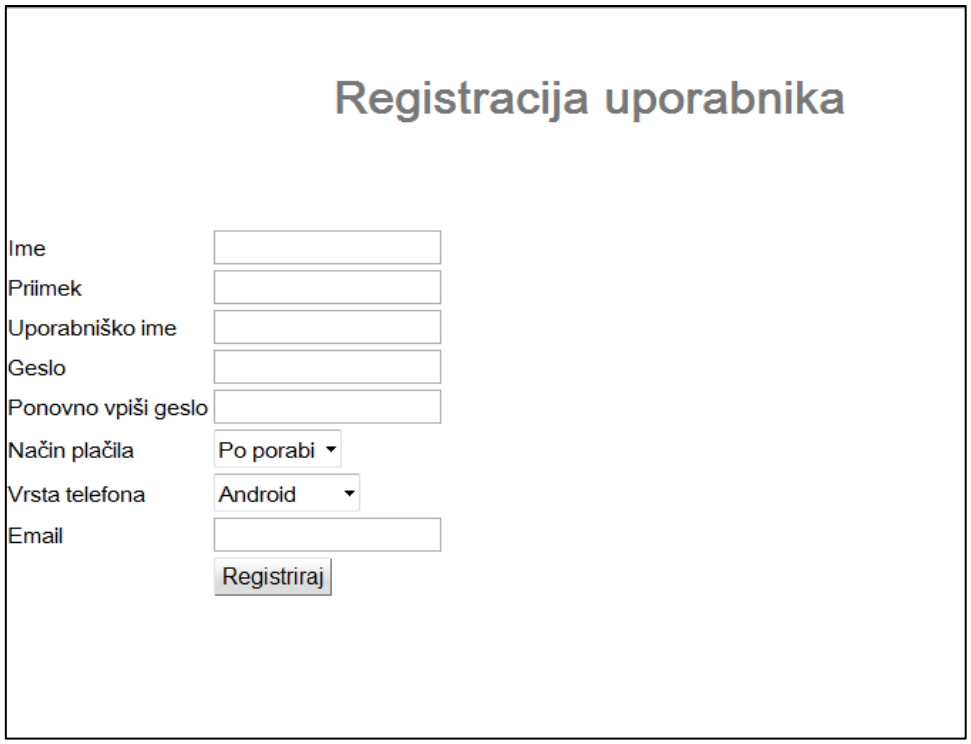
Tabela potovanja s podatki iz testiranja je prikazana na sliki 24.

idpoti	od	do	datumvstopa	username	cena	datumizstopa	koda	uidVstopa	uidIzstopa
165	kr	lj	1285341237	igor	12	1285341256	m3wif726wt	0	0
166	lj	kr	1285341482	igor	12	1285341502	lfa2apgt2w	0	0
167	kr	mb	1285343084	lukafin	20	1285343095	8b6o90oxa8	0	0
168	mb	lj	1285511015	lukafin	10	1285511364	n7xs68stdo	f3da3ea7	8871d5d1
169	lj	kr	1285857277	igor	12	1285857334	xf8lszo2xi	null	null
170	lj	kr	1285860828	lukafin	12	1285860977	fge52rko2p	0	null
171	lj	kr	1285861008	lukafin	12	1285861047	i6jnv0nkn2	0	0
172	lj	kr	1285861400	lukafin	12	1285861422	zywbm08...	0	0
173	kr	mb	1285861931	lukafin	20	1285861965	p71b1ez4o5	null	null
174	kr	lj	1285940447	lukafin	12	1285940460	z437a8u88k	22102cd1	8871d5d1
175	kr	lj	1286205476	igor	12	1286205575	0iyb5btv97	0	0
176	kr	lj	1286206057	igor	12	1286206225	tnhwuwc...	0	0
177	kr	lj	1286268441	lukafin	12	1286268514	u90n9a4jnu	22102cd1	8871d5d1
178	kr	lj	1286268524	lukafin	12	1286268557	09i4fbbgwk	22102cd1	8871d5d1
179	kr	mb	1286350959	lukafin	20	1286351005	1rz48p7wy1	22102cd1	f3da3ea7
180	kr	lj	1287476036	igor	12	1287476161	vwzntkwx2r	0	0
181	lj	kr	1289547908	igor	12	1289547917	uwjj3h07kr	null	null
182	kr	lj	1289548487	igor	12	1289548495	pt2eclctb5	null	null
183	kr	lj	1289548507	igor	12	1289548531	tcpivchh4i	0	0
184	Lj	Kr	1289554439	igor	12	1289554523	07wdgma...	4c3e5725	7cc55825
185	Kr	Lj	1289554572	igor	12	1289554598	hv686i0nyf	7cc55825	4c3e5725
186	Kr	Lj	1289555496	igor	12	1289555567	94isv26ke5	7cc55825	4c3e5725
187	Lj	Lj	1289556705	igor	0	1289556716	bzmp3fuvvn	4c3e5725	4c3e5725
188	Kr	Kr	1289556726	igor	50	1289556768	njjs5955jb	7cc55825	7cc55825
189	kr	lj	1289557803	lukafin	12	1289557838	ow9iw4c1ck	0	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Slika 24. Tabela potovanja s podatki iz testiranja.

4.3.2 Spletna aplikacija za registracijo uporabnikov

Spletna stran za registracijo novih uporabnikov je sestavljena iz več vnosnih polj ter menijev, v katera uporabnik vpiše zahtevane podatke. Stran je narejena v Javi s pomočjo GWT. V metodi »onModuleLoad« (vstopna metoda, podobno kot metoda main v namiznih Javanskih aplikacijah) se zgradijo elementi grafičnega vmesnika ter nastavi poslušalca za gumb »registriraj«. Zaslonska maska strani je prikazana na sliki 25.



Slika 25. Stran za registracijo uporabnikov.

Ob kliku gumba »Registriraj« se kliče metoda »sendNameToServer«, ki naredi naslednje:

1. Iz vnosnih polj prebere podatke.
2. Preveri ustreznost podatkov (dolžina gesla vsaj 7 znakov, ujemanje 2x vpisanega gesla, izpolnjenost vseh zahtevanih polj).
3. Asinhroni klic metode »greetServer« na strežniku s pomočjo GWT RPC mehanizma.
4. Ob uspešni izvedbi metode »greetServer« strežnik pokliče metodo »onSuccess«, ki uporabnika preusmeri na spletno stran »Uspeh.html« z obvestilom o uspešni registraciji.

Metoda »greetServer« v tabeli »users« preveri, če sta izbrano uporabniško ime in elektronski naslov že zasedeni. Nato prebere trenuten čas in vse podatke vstavi v tabelo »users«. Po vstavljanju se pokliče metoda »posljiMail«, ki uporabniku na njegov elektronski naslov pošlje registracijske podatke ter priponko z aplikacijo za njegov tip mobilnega telefona. Za pošiljanje se je uporabil moj Gmail račun.

Za dostop do podatkovne baze sem uporabil knjižnico (gonilnik) MySQL Connector/J [22], ki pretvarja JDBC (ang. Java Database Connectivity) klice v klice po protokolu MySQL podatkovne baze. Dostop do baze iz programske kode poteka po naslednjih korakih:

1. Nalaganje gonilnikov s kodo:
   »Class.forName("com.mysql.jdbc.Driver").newInstance();«.
2. Vzpostavitev povezave na določeno tabelo s kodo:
   »Connection conn = DriverManager.getConnection(url, user, pass);«.
3. Kreiranje Statement objekta, ki predstavlja SQL stavek s kodo:
   »Statement st = conn.createStatement();«.
4. Izvedba SQL stavka in branje vrnjenih podatkov s kodo:
   »ResultSet res = st.executeQuery("SELECT COUNT(*) FROM users WHERE

```
username = "'+data[2]+"';");
int i = res.getInt("COUNT(*)");«.
```

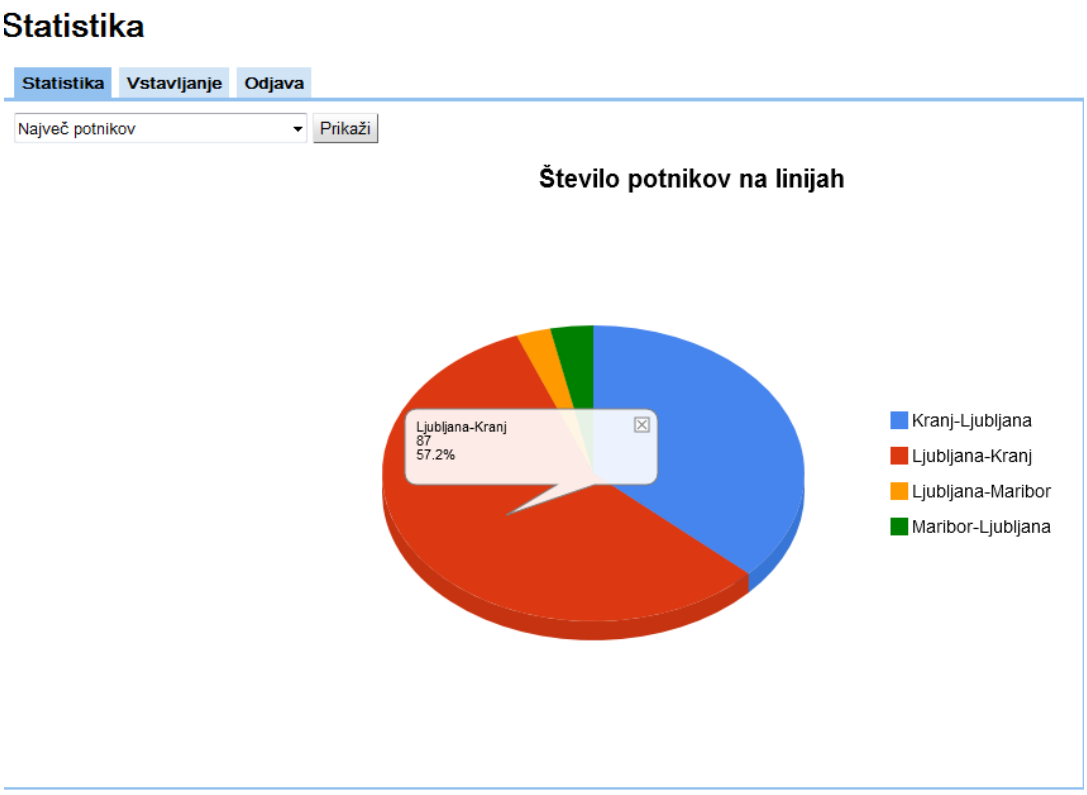
Za pošiljanje elektronske pošte iz Java programske kode sem uporabil programski vmesnik JavaMail [23]. Pri nastavitvah je treba vpisati URL naslov poštnega strežnika, uporabniško ime in geslo za dostop do njega.

4.3.3 Spletna aplikacija za administratorje

Aplikacija omogoča registriranim uporabnikom tipa administrator pregled raznih podatkov o prevozih, vstavljanje voznih redov in cene mesečne vozovnice. Stran za prijavo (DiplomaAdmin.html) in stran za pregled statističnih podatkov o prevozih (Stats.html) sta narejeni v Javi s pomočjo GWT in sicer kot dva modula.

Stran za prijavo prebere vpisano uporabniško ime in geslo ter s klicem metode »greetServer« (teče na strežniku, zato uporaba GWT RPC) v tabeli »users« preveri, če je uporabnik registriran in če je uporabnik tipa administrator (stolpec vrsta z vrednostjo 1). Ob uspešni prijavi se nastavi piškotek (ime: logiran, vrednost: da, čas poteka: teden dni) ter uporabnika preusmeri na stran »Stats.html«.

Spletna stran »Stats.html« je sestavljena iz treh zavihkov (slika 26).



Slika 26. Zaslonska maska aplikacije za pregled podatkov o prevozih.

Pri nalaganju strani v metodi »onModuleLoad« se najprej prebere piškotek, nato pa se v primeru prave vrednosti izriše celotna spletna stran. Poleg tega se nastavijo poslušalci za vse gumbe. Grafe sem izrisoval s pomočjo programskega vmesnika Google Visualization [24]. Ta

programski vmesnik omogoča uporabo najrazličnejših grafov (krožni, vrstični, …) in velik nadzor nad videzom (naslov, oblika, barve, …) v za to namenjenih metodah. Podatke za izris grafov sem dobil iz podatkovne baze (GWT RPC).

Sledi opis podrobnosti pri implementaciji posameznih grafov in drugih funkcij administratorske spletne strani:

a) Prikaz števila potnikov na posameznih progah je narejen s pomočjo krožnega grafa. Ob kliku na posamezen izsek grafa se izpišeta tudi točno število in odstotni delež. Podatki za posamezno progo (npr. Kranj-Ljubljana) se dobijo iz tabele »potovanja« s pomočjo SQL stavka:
"SELECT COUNT(*) FROM potovanja WHERE od='kr' AND do='lj'".

b) Prikaz števila uporabnikov, ki plačujejo po porabi, in tistih, ki plačujejo pavšalne mesečne zneske, je narejen z vrstičnim grafom. Podatki za uporabnike, ki na primer plačujejo po porabi, se dobijo iz tabele *users* s pomočjo SQL stavka: "SELECT COUNT(*) FROM users WHERE nacinp='Po porabi'".

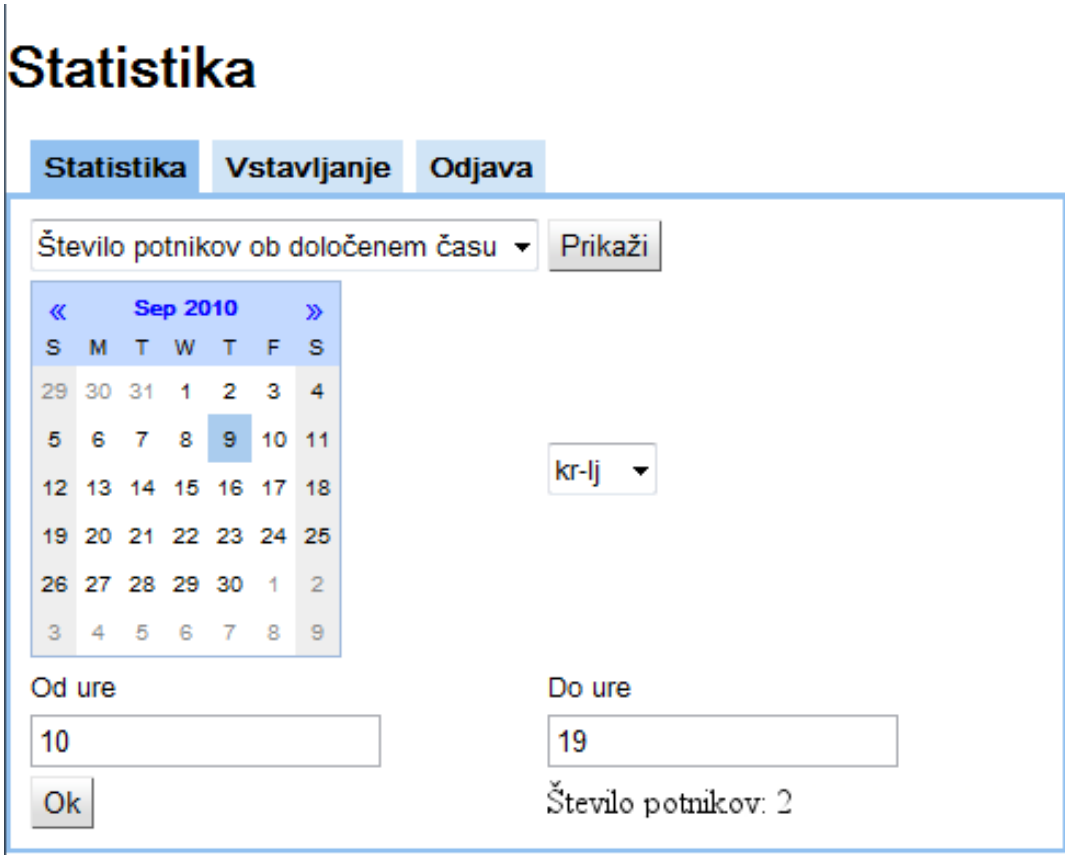
c) Število novoregistriranih uporabnikov za zadnje tri mesece je prikazano z vrstičnim grafom. Najprej se izračuna čas začetka (prvi) in konca (zadnji) zadnjih treh mesecev (POSIX format), nato pa se naredi SQL poizvedba: "SELECT count(*) FROM users WHERE casregistracije>=časZačetka AND casregistracije<=časKonca".

d) Mogoč je prikaz števila potnikov, ki so vstopili na določeno progo med dvema izbranima urama (slika 27). Uporabnik lahko na koledarju izbere določen dan, v polja vpiše začetno in končno uro ter prek menija izbere progo (spremenljivki »odK« in »doK«). Za ta dva datuma (spremenljivka »from« in »till«) se pridobi čas v POSIX formatu ter nato prebere podatke iz tabele »potovanja« s SQL stavkom: "SELECT COUNT(*) FROM potovanja WHERE datumvstopa>=from AND datumvstopa<=till AND od='odK' AND do='doK'".

e) Dodajanje odhodnega časa vlaka za določeno progo je mogoče v zavihku »Vstavljanje«. Prek menija se izbere začetno ter končno postajo, preko vnosnega polja pa se prebere čas odhoda. Metoda »vstaviOdhod«, ki teče ne strežniku, najprej preveri, če slučajno ta odhodni čas že obstaja. Če ne obstaja, ga vstavi v ustrezno tabelo (krodhodi, ljodhodi ali mbodhodi). Npr. za nov vlak Kranj–Ljubljana z odhodom ob 12.30 se izvede SQL stavek:
INSERT INTO 'krodhodi' VALUES ('12:30:00,'Ljubljana');".

f) Ceno mesečne vozovnice je mogoče spremeniti s klicem metode »dodajMesecno«, ki na strežniku pokliče metodo »vstaviMesečno«, ta pa izvede SQL stavek: "UPDATE mesecna set cena=novaCena".

g) Ob pritisku gumba »odjava« se pobriše uporabnikov piškotek, nato pa se ga preusmeri na prijavno stran »DiplomAdmin.html«.



Slika 27. Prikaz števila potnikov na določeni progi ob določenem času.

4.3.4 Servlet

V »servletu« (datoteka »KartaServer.java«) je najpomembnejša koda, ki skrbi za registracijo vstopov (ustvarjanje in pošiljanje vozovnic), registracijo izstopov, preverjanje veljavnosti vozovnic, računanje porabe, pošiljanje voznih redov in računanje mesečne porabe ter pošiljanje računov. Glavne funkcionalnosti so v metodi »doGet«, kjer se najprej iz HTTP GET zahteve prebere vrednost parametra »ukaz«. Glede na to vrednost se izvede naslednje:

a) Registracija vstopa in pošiljanje ustvarjene vozovnice (ukaz=1)

Delovanje tega algoritma je najlažje ponazoriti s psevdokodo in naknadno razlago nekaterih pomembnejših delov:

```
If (uporabniško ime in geslo obstajata) {
        vstaviZačetekPotovanjaVBazo();
        narediNaključenNizZnakov();
        ustvariVozovnico();
        pošljiVozovnico();
}
Else {
        VrniNizOdjemalcu(»LOGIN_NAPAKA«);
        }
```

V tabelo »potovanja« se doda nova vrstica s krajem vstopa, datumom vstopa, naključnim nizom in UID vrednostjo. Razen naključnega niza so vsi parametri HTTP GET zahteve, ki pridejo na strežnik iz mobilnih telefonov.

Naključen niz znakov se naredi z metodo »generateString«, kateri kot parameter podamo množico znakov (iz katerih se dela naključen niz) in želeno dolžino.

```
public static String generateString(Random rng, String characters, int length)
{
        char[] text = new char[length];
        for (int i = 0; i < length; i++)
        {
                text[i] = characters.charAt(rng.nextInt(characters.length()));
        }
        return new String(text);
}
```

Vozovnico v obliki QR črtne kode se ustvari s pomočjo metode »narediQr«. Ta v črtno kodo zapiše niz oblike »UporabniškoIme/ČasVstopa/NaključenNiz«. Kodo se ustvari s pomočjo Zxing Java knjižnic. Naključen niz se vsebini vozovnice dodaja kot varovalka pred kreiranjem lastnih vozovnic in posledičnim goljufanjem. Kreirano vozovnico v formatu PNG in velikosti manj kot 1 KB se nato shrani na strežniku.

Pred pošiljanjem vozovnice z uporabo tokov razreda OutputStream se nastavi še MIME tip HTTP odziva na »image/png« in dolžino odziva na velikost slike (vozovnice).

b) Registracija izstopa in pošiljanje cene opravljenega potovanja (ukaz=2)

Najprej se nastavi MIME tip odziva na »text/plain« kodnega nabora UTF-8, nato pa sledi postopek, ponazorjen s psevdokodo:

```
If  (uporabniško ime in geslo obstajata) {
        pridobiVstopnoPostajoUporabnika();
        if (vstopnaPostaja==IzstopnaPostaja && čas izstopa manj kot 30s po vstopu) {
                vpišiCenoPotovanjaVBazo(0); //zastonj
                vrniCenoPotovanjaOdjemalcu();
                }
        else if ( vstopnaPostaja==IzstopnaPostaja && čas izstop več kot 30s po
        vstopu) {
                vpišiCenoPotovanjaVBazo(50); //kazen
                vrniCenoPotovanjaOdjemalcu();
                }
        else {
                dobiCenoZaRelacijo();
                vpišiCenoPotovanjaVBazo();
                vrniCenoPotovanjaOdjemalcu();
```

```
                }
}
Else {
        VrniNizOdjemalcu(»LOGIN_NAPAKA«);
        }
```

Vstopno postajo začetega potovanja uporabniškega imena »u« dobimo iz tabele »potovanja« s SQL stavkom:

```
»SELECT od FROM `potovanja` WHERE datumizstopa IS NULL AND username='u'«
```

Če je bil izstop narejen manj ali več kot 30 s po vstopu (v primeru istega vstopnega in izstopnega mesta), se vidi iz parametra HTTP GET zahteve z imenom *nacinizstopa*. V prvem primeru ima vrednost *h*, v drugem pa *p*.

Ceno za na primer relacijo Kranj-Ljubljana dobimo z naslednjim SQL stavkom:

```
SELECT cena FROM `kr` WHERE ime='lj';
```

Ceno potovanja skupaj z mestom izstopa, časom izstopa in UID vrednostjo vpišemo z SQL stavkom:

```
UPDATE `test`.`potovanja` SET `do`='"+kraj+"', `cena`= "+cena+", `datumizstopa`=
"+cas+",uidIzstopa='"+uid+"' WHERE datumizstopa IS NULL AND username='"+u+"
```

Na koncu odjemalcu (mobilnemu telefonu) vrnemo še niz oblike »CheckOutOK/CenaOpravljenegaPotovnja« in zapremo povezavo.

c) Preverjanje veljavnosti vozovnic (ukaz=3)

Iz HTTP GET zahteve se prebere parametre uporabniško ime, naključen niz in čas vstopa. Nato se pokliče metodo »preveriKarto«, ki za uporabniško ime in čas vstopa v tabeli »potovanja« prebere naključen niz. SQL stavek za uporabnika Janez in datum 1286523357:

```
SELECT koda FROM `potovanja` WHERE username='Janez' AND datumvstopa='1286523357'
AND datumizstopa IS NULL).
```

Če ga najde in se prejeti naključni niz in tisti v bazi ujemata, gre za veljavno vozovnico. V tem primeru se odjemalcu vrne niz »Karta je veljavna«, v nasprotnem primeru pa »Karta ni veljavna« (MIME tip odziva je »text/plain« s kodnim naborom UTF-8).

d) Računanje porabe (ukaz=4)

Računanje in pošiljanje porabe v psevdokodi:

```
If (uporabniško ime in geslo obstajata) {
        preberiNačinPlačevanja();
        if (plačuje se pavšalno 1 x mesečno) {
```

```
                preberiCenoMesečne();
                vrniCenoOdjemalcu();
        }
        Else { //plačuje se po porabi
                dobiČasZačetkaMeseca();
                dobiČasKoncaMeseca();
                dobiPorabo();
                vrniPoraboOdjemalcu();
        }
}
Else {
        VrniNizOdjemalcu(»Niste registriran uporabnik!«);
}
```

Cena mesečne vozovnice se prebere iz tabele »mesecna«. Poraba v trenutnem mesecu pa se izračuna s pomočjo SQL stavka (npr. med POSIX časom 1285891200 in 1288569599 za uporabnika Janez):

```
SELECT sum(cena) AS poraba FROM `potovanja` WHERE username='Janez' AND datumizstopa BETWEEN 1285891200 AND 1285891200.
```

e) Pošiljanje voznega reda (ukaz=5)

Odjemalcu se za njegovo vstopno postajo pošlje prve tri odhode vlakov. Podatke se dobi iz tabele »krodhodi«, »ljodhodi« ali »mbodhodi«, odvisno od mesta vstopa. Branje iz tabele za mesto Kranj poteka s SQL stavkom:

```
SELECT cas,cilj FROM krodhodi WHERE cas>NOW() LIMIT 3.
```

Prebrano besedilo se nato oblikuje v niz oblike »1 Maribor: 14:32:00 /2 Maribor: 17:45:00 /3 Ljubljana: 17:55:00« in pošlje odjemalcu.

f) Računanje mesečne porabe in pošiljanje računov (ukaz=6)

Izračun porabe za pretekli mesec in pošiljanje porabe znotraj priponke elektronske pošte je prikazano v psevdokodi:

```
If (obračun je zagnal registriran uporabnik tipa administrator) {
        preberiVsaUporabniškaImena();
        preberiVseElektronskeNasloveUporabnikov();
        pridobiNačinPorabeZaVseUporabnike();
        for (vsak uporabnik) {
                if (plačuje se po porabi) {
                        zračunajPorabo();
                        narediPdfSPorabo();
                        pošljiEmailSPriponko();
                }
```

```
                    Else { //plačuje se mesečne pavšalne zneske
                        preberiCenoMesečne();
                        narediPdfSPorabo();
                        pošljiEmailSPriponko();
                    }

            }
        }
        Else {
            VrniNizOdjemalcu(»Omenjeno akcijo lahko začne le administrator!«);
        }
```

PDF-dokumente z vsebino (»Porabili ste za x €«) sem naredil z uporabo knjižnice iText [25]. Pošiljanje elektronske pošte s PDF-priponko pa sem naredil s pomočjo programskega vmesnika JavaMail. Med testiranjem sem zagon mesečnega obračuna opravil z uporabo razporejevalnika opravil (ang. task scheduler), ki je enkrat na mesec pognal spletni brskalnik in naredil HTTP GET zahtevo na naslov »http://localhost:8080/webui/karta?user=lukafin&pass=lukafin&ukaz=6«.

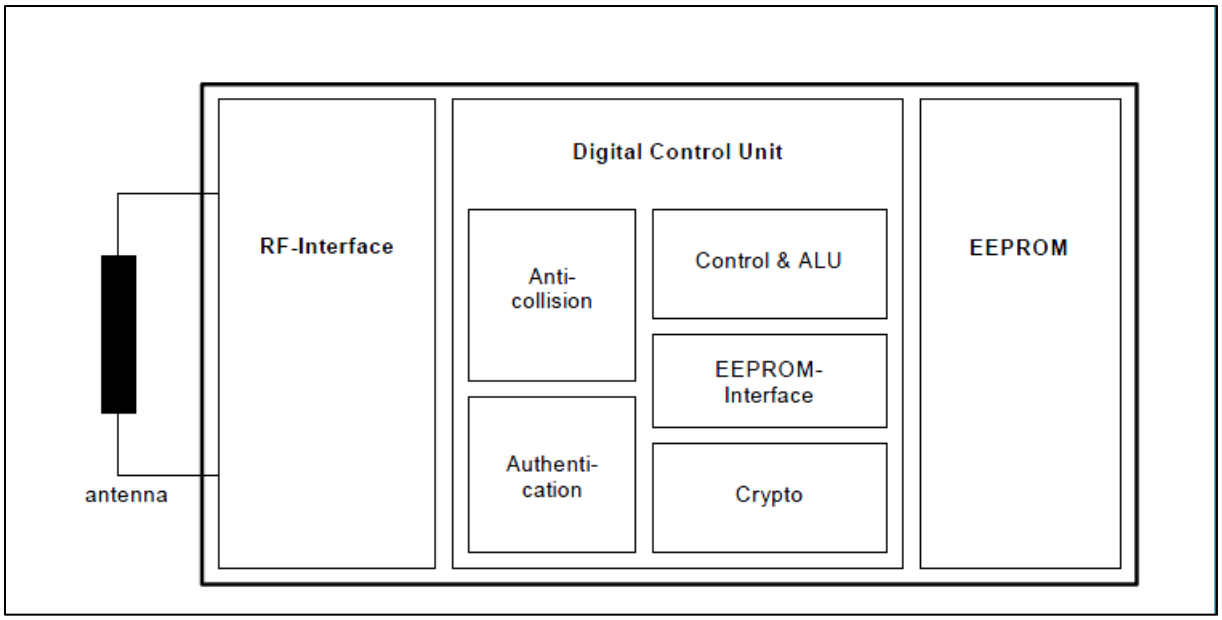
4.4 Zapis podatkov na RFID značke

Za potrebe testiranja sem za mobilni telefon Nokia 6212 napisal program, ki je na RFID znački ustvaril NDEF zapis tipa »text/plain; FORMAT=FIXED«. V tem zapisu smo shranili ime vstopne/izstopne postaje. S tem se je preverilo, ali aplikacija za mobilni telefon deluje tudi z resničnimi RFID značkami in ne le v emulatorju.

Uporabili smo značko MIFARE MF1ICS50 podjetja NXP [26] z naslednjimi lastnostmi:

- Delovanje na razdalji do 100mm (odvisno od geometrije antene),
- Delovanje na frekvenci 13,56MHz,
- Hitrost prenosa podatkov 106 kbit/s,
- Zaznavanje napak pri prenosu s 16 bitnim CRC,
- 1KB EEPROM pomnilnika, razdeljenega na 16 sektorjev s 4 bloki,
- Obstojnost podatkov vsaj 10 let ali 100 000 pisalnih ciklov,
- Avtentikacija po standardu ISO/IEC DIS 9798-2 (dva ključa na sektor).

Bločni diagram značke je prikazan na sliki 28.



Slika 28. Bločni diagram RFID značke MF1ICS50.

5 Sklepne ugotovitve

V diplomskem delu je predstavljen sistem, ki uporabnikom mobilnih telefonov omogoča pridobitev elektronskih vozovnic za javni prevoz. Sistem vrši tudi zaračunavanje vozovnic uporabnikom in upravljalcem sistema omogoča ogled nekaterih statističnih podatkov o opravljenih prevozih. Razvite aplikacije omogočajo preprosto in hitro registracijo potnikov na vstopnih in izstopnih postajah.

Med izdelavo diplomskega dela sem se seznanil s tehnologijami in orodji za razvoj spletnih aplikacij ter programov za mobilne naprave. Spoznal sem se tudi z različnimi pristopi beleženja in obračunavanja potnikov doma in po svetu.

Ugotovil sem, da je za prenos manjše količine podatkov iz okolice (fizičnega sveta) v elektronske naprave (virtualni svet) zelo smiselna uporaba mobilnega telefona s kamero in uporaba QR črtnih kod (ali RFID značk). Ta način se zdi zelo primeren za identifikacijo vstopne (izstopne) postaje javnega prometa. Mobilni telefoni s kamero so zelo razširjeni tudi v nižjem cenovnem razredu, na drugi strani pa si QR črtno kodo lahko brezplačno izdelamo in jo za majhen strošek tudi natisnemo. Prav tako v zadnjih letih vztrajno padajo cene RFID značk. Tu je za zdaj problem le v majhnem številu mobilnih telefonov, ki podpirajo branje RFID značk (to se bo najverjetneje spremenilo v letu 2011, ko bo po napovedih NFC tehnologija v mobilnih telefonih z OS Android, OS Apple iOS ter v Nokijinih pametnih mobilnikih) [27].

Izdelan sistem nudi glavne funkcionalnosti uporabe v prometu, zato je v nadaljevanju predlaganih še nekaj idej za izboljšavo:

- Sistem bi uporabnika, ki se že zelo dolgo ni odjavil (vstopil, a ni izstopil), o tem opozoril prek SMS-sporočila ali z uporabo servisa na mobilnem telefonu. Po določenem času pa bi mu zaračunaval ceno najdaljšega mogočega potovanja. Na ta način bi preprečili zlorabe in omogočili pravilno uporabo sistema.
- Ob vstopu bi uporabnik od prevoznika na mobilni telefon prejel informacije o zamudah, izrednih dogodkih, akcijskih cenah prevozov in ostalih aktualnih ugodnostih.
- Na QR črtno kodo, ki služi kot vozovnica, bi lahko zapisali tudi podatke (SSID, geslo), potrebne za priklop na brezžično omrežje na vlaku.
- Na postajah bi bilo vzpostavljeno brezžično internetno omrežje (WI-FI), prek katerega bi aplikacije na mobilnih telefonih pridobivale vozovnice. S tem uporabnik ne bi imel stroškov prenosa podatkov.
- Registriran uporabnik bi se lahko prijavil na spletno stran, kjer bi lahko pregledoval opravljena potovanja. Sistem bi mu glede na te podatke lahko predlagal, ali je zanj ugodneje, da potuje z mesečno vozovnico ali plačuje po porabi.
- Pri spletni strani s statistiko je mogočih veliko izboljšav. Pomembno bi bilo opozarjanje administratorjev v primeru, če so vlaki na določenih progah preveč zasedeni. Na ta način bi lahko prilagajali vozne rede in velikosti posameznih vlakovnih kompozicij.

Sistem bi bilo mogoče povezati tudi s spletno stranjo ali mobilno aplikacijo turističnega društva. Na ta način bi uporabnik lahko takoj po prihodu na cilj potovanja prejel informacije o mestnih znamenitostih (turistični vodič) in podatke o aktualnih dogodkih (športnih, kulturnih).

Literatura

[1] (2010) OEE On-line-Ticket. Dostopno na: http://www.oebb.at/de/Fahrkarten/Online-Ticket/index.jsp

[2] (2010) Moneta predstavitev. Dostopno na: http://www.moneta.si/predstavitev/postopek_placevanja/lpp

[3] (2010) Touch and Travel. Dostopno na: http://www.touchandtravel.de/site/touchandtravel/de/start.html

[4] (2010) Enotna mestna kartica Urbana. Dostopno na: http://www.jhl.si/holding/urbana

[5] (2010) Oyster ticket. Dostopno na: https://oyster.tfl.gov.uk/oyster/entry.do

[6] (2010) Eclipse IDE. Dostopno na: http://www.eclipse.org/

[7] (2010) What is Android. Dostopno na: http://developer.android.com/guide/basics/what-is-android.html

[8] (2010) Android SDK. Dostopno na: http://developer.android.com/sdk/index.html

[9] (2010) Android NDK. Dostopno na: http://developer.android.com/sdk/ndk/index.html

[10] (2008) Series 40 Nokia 6212 NFC SDK. Dostopno na: http://www.forum.nokia.com/info/sw.nokia.com/id/5bcaee40-d2b2-4595-b5b5-4833d6a4cda1/S40_Nokia_6212_NFC_SDK.html

[11] (2010) Google Web Toolkit Overview. Dostopno na: http://code.google.com/webtoolkit/overview.html

[12] Bryan Basham, Kathy Sierra, Bert Bates, Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam, O'Reilly Media, maj 2008

[13] (2010) The world's most popular open source database. Dostopno na: http://www.mysql.com/

[14] (2010) Apache Tomcat. Dostopno na: http://tomcat.apache.org/

[15] Bill Glover, Himanshu Bhatt, RFID Essentials (Theory in Practice), O'Reilly Media, januar 2006

[16] HP Labs' Central Nervous System for the Earth project aims to build a planetwide sensing network: HP Labs Feature Article (November 2009). Dostopno na: http://www.hpl.hp.com/news/2009/oct-dec/cense.html

[17] (2010) The Internet of things. Dostopno na: http://asmarterplanet.com/blog/2010/03/the-internet-of-things.html

[18] (2010) About QR code. Dostopno na: http://www.denso-wave.com/qrcode/index-e.html

[19] (2010) Zebra Crossing Java library. Dostopno na: http://code.google.com/p/zxing/

[20] (2010) JSR 257: Contactless Communication API. Dostopno na: http://jcp.org/en/jsr/detail?id=257

[21] (2009) NFC Forum type tags. Dostopno na: http://www.nfc-forum.org/resources/white_papers/NXP_BV_Type_Tags_White_Paper-Apr_09.pdf

[22] (2010) Using MySQL With Java. Dostopno na: http://dev.mysql.com/usingmysql/java/

[23] (2010) JavaMail. Dostopno na: http://www.oracle.com/technetwork/java/index-jsp-139225.html

[24] (2010) Getting Started Using the Google Chart Tools with GWT. Dostopno na: http://code.google.com/p/gwt-google-apis/wiki/VisualizationGettingStarted

[25] (2010) iText - Free / Open Source PDF Library for Java and C#. Dostopno na: http://itextpdf.com/

[26] (2008) MF1ICS50 functional specifications. Dostopno na: http://www.nxp.com/acrobat_download2/other/identification/M001053_MF1ICS50_rev5_3.pdf

[27] (2010) Google CEO: Android 2.3 due in weeks with NFC, Nexus S shown. Dostopno na: http://www.electronista.com/articles/10/11/15/googles.schmidt.demos.nexus.s.and.gingerbread.nfc/