MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

SISTEM DE LOCALIZARE A MIJLOACELOR DE TRANSPORT ȘI EMITERE DE BILETE PE DISPOZITIVE MOBILE

LUCRARE DE LICENŢĂ

Absolvent: **Bogdan Vasile LUNGU**

Coordonator ştiinţific: **Șef lucrări ing. Cosmina IVAN**

2011

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DECAN,
Prof. dr. ing. Sergiu NEDEVSCHI

VIZAT,
ŞEF CATEDRĂ,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Bogdan Vasile LUNGU**

SISTEM DE LOCALIZARE A MIJLOACELOR DE TRANSPORT ȘI EMITERE DE BILETE PE DISPOZITIVE MOBILE

1. **Enunţul temei:** *Aplicația implementată va permite localizarea mijloacelor de transport în comun,vizualizarea acestora împreună cu stațiile corespunzătoare pe harta Google Maps, calcularea și vizualizarea pe hartă a unei rute la locația clientului la o stație, precum și emiterea de bilete pe dispozitivul mobil.*

2. **Conţinutul lucrării:** *Introducere, Obiectivele proiectului, Studiu Bibliografic, Analiză și fundamentare teoretică, Proiectare de detaliu și implementare, Testare și validare, Manual de instalare și utilizare, Concluzii*.

3. **Locul documentării: UTCN, biblioteca UTCN, biblioteca catedrei de Calculatoare**

4. **Consultanţi:**

5. **Data emiterii temei:** 1 noiembrie 2010

6. **Data predării:** 24 Iunie 2011

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAŢIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Declaraţie

Subsemnatul *Bogdan Vasile Lungu*, student al Facultăţii de Automatică şi Calculatoare, Universitatea Tehnică din Cluj-Napoca, declar că ideile, analiza, proiectarea, implementarea, rezultatele şi concluziile cuprinse în această lucrare de licenţă constituie efortul meu propriu, mai puţin acele elemente ce nu îmi aparţin, pe care le indic şi recunosc ca atare.

Declar de asemenea că, după ştiinţa mea, lucrarea în această formă este originală şi nu a mai fost niciodată prezentată sau depusă în alte locuri sau alte instituţii decât cele indicate în mod expres de mine.

Data: 24 Iunie 2011

Absolvent: **Bogdan Vasile LUNGU**
Număr matricol: 21010653
Semnătura:_____

Cuprins

Capitolul 1. Introducere

1.1 Contextul proiectului

Telefonul nu mai este de mult timp un simplu mijloc de comunicare care transmite și recepționează sunete la distanță. Accesul la Internet direct de pe telefonul mobil, vitezele de download de până la 14.4Mbps, site-urile pentru mobil, plățile facturilor si reclamele pe telefonul mobil fac parte din viața noastră de zi cu zi. Se estimează un număr de 5,28 miliarde de utilizatori de telefonie mobilă până în 2013, dintre care peste 1,5 miliarde vor deveni utilizatori de Internet mobil de mare viteză.

Dezvoltarea tehnologiei telefoniei mobile a suferit numeroase modificări de-a lungul timpului. Tehnologiile cheie utilizate in sistemele radio de comunicare mobila includ reutilizarea frecvenței mobile, sisteme analogice (Generatia 1), sisteme mobile digitale (Generatia 2), sisteme radio digitale bazate pe pachete (Generatia 2 ½) si sisteme bazate pe lătime de bandă mare (Generatia 3).

Tehnologia telefoniei mobile se află astăzi în plină evoluție. Aceasta constă atât în creșterea numărului de abonați și de posesori de telefoane, cât și în creșterea numărului de servicii și opțiuni realizabile cu ajutorul telefonului, bazate pe Internet și GPS, cum ar fi ghidaj pentru pietoni (de ex. Gasirea unei rute pentru a ajunge la o destinație dorită), informații locale (de ex. Localizarea celui mai apropiat hotel), sistem de plată (portmoneu electronic), modalitate de acces securizat în zone speciale și multe altele.

În prezent, pe lângă preocupările pentru introducerea sistemelor 3G în funcţiune, au început lucrări experimentale pentru o nouă generaţie de sisteme de comunicaţii mobile digitale, 4G, pentru care se prevede realizarea unor viteze de transmisie de utilizator de până la 100 Mbit/s. Caracteristica principală a 4G va fi reprezentată de controlul exercitat de utilizator asupra serviciilor, pe care le va gestiona în funcţie de pachetul de servicii la care s-a abonat. Deci utilizatorul va avea libertatea de a selecta serviciul dorit, cu un indice de calitate dorit, la un preţ acceptabil, oriunde şi oricând.

Tipurile de servicii oferite de sistemele 3G se diversifică faţă de oferta realizată de 2G sau de sistemele de generaţia 1, sistemele 3G fiind capabile să ofere:

- multimedia înalt interactiv (videoconferinţe, lucru în colectiv şi teleprezenţă);
- multimedia de viteză mare (acces rapid LAN şi Internet/Intranet, videoclipuri la cerere, cumpărături în direct etc.);
- multimedia de viteză medie (acces LAN şi Internet/Intranet, jocuri interactive, mesaje radiodifuzate şi informaţii publice complexe, jocuri interactive etc.);
- date comutate (acces LAN de viteză redusă, acces Internet/Intranet, fax etc.)
- mesagerie simplă (serviciu de mesaje scurte, e . mail, radiodifuzare şi mesagerie de informaţii publice, comenzi / plăţi pentru comerţul electronic simplu etc.);
- transmisii vocale (comunicare bilaterală, conferinţe, poştă vocală etc).

De-a lungul timpului s-au dezvoltat nenumărate aplicații având rolul de a ajuta societatea să se dezvolte intr-un ritm accelerat. Cele mai noi și în continuă dezvoltare sunt cele pentru dispozitive mobile, acestea având un succes enorm în aceasta perioadă, chiar dacă tot aplicațiile desktop sunt în continuare cele mai folosite.

Principalele caracteristici ale aplicațiilor mobile sunt urmatoarele:

- Comoditatea: Un design bun şi o manipulare simplă (cu o singură mână), garantează o stare de comoditate in rândul utilizatorilor. O aplicație bună își poate face treaba în diferite contexte şi situaţii diferite extrem de rapid (schimbare de lumină şi zgomot de mediu, miscare instabilitate a aparatului, etc).
- Localizarea: Localizarea şi posibilitatea de a oferi informatii bazate pe locaţie este un element cheie care face ca lumea mobilă să fie vie şi practică. Această caracteristică încorporează aplicația în contextul utilizatorilor.
- Accesibilitatea: Accesibilitatea acoperă un atribut mai social dat de natura de aplicaţii mobile în sine. O bună aplicaţie poate fi utilizată într-adevăr - şi mai important logic - oriunde în orice moment.
- Securitatea: Datele transferate prin reţea trebuie să fie criptate prin intermediul reţelei de transport. Deoarece unele aplicaţii de date sunt sincronizare cu aplicatii on-line, bazate pe web, stocarea acestor date pe server trebuie să fie, de asemenea, asigurată. Un alt aspect se referă la datele de pe dispozitiv.
- Personalizarea: Crearea de conţinut personalizat bazat pe utilizarea individuală sau în al context este o alta caracteristica. Aceasta se bazează pe toate caracteristicile anterioare și putem spune că se „mulează" pefect peste acestea.

1.2 Domeniul temei

CIVITAS (Cleaner and better transport in cities) Initiative este o acţiune europeană care susţine oraşele în punerea în aplicare a unei politici de transport integrate sustenabile, curate şi eficiente din punct de vedere energetic. În cadrul CIVITAS II (2005–2009), s-au pus în aplicare diferite măsuri în care au fost dezvoltate sisteme inovatoare de plată şi emitere a biletelor pentru transportul public, pentru a spori atractivitatea acestui mod de transport şi creşte proporţia de călători care îl folosesc.

Pentru a creşte gradul de utilizare a transportului public, oraşele trebuie să își propună să facă din sistemul de emitere a biletelor unul atractiv şi uşor de înţeles pentru toţi. Sistemul de stabilire a preţurilor trebuie să fie coerent şi simplu, cu un număr suficient de bilete care să ţină cont de nevoile utilizatorilor. Baza pentru costul biletelor trebuie să fie transparentă şi uşor de înţeles. Biletele şi sistemele de plată trebuie să fie disponibile pe scară largă, de exemplu:

- La puncte de vânzare răspândite în întregul oraş
- La automate de vânzare a biletelor din diverse locuri (de exemplu, în staţiile de parcare de tip „park and ride", în principalele staţii de autobuz sau în vehicule)
- Pe internet (de exemplu, abonament pentru deţinătorii de carduri inteligente (smart cards))
- Prin telefoanele mobile

Trebuie oferite politici integrate de emitere a biletelor şi tarifare între diferiţi operatori de transport public (de exemplu, transportul public local şi căile ferate naţionale) pentru a asigura valabilitatea biletelor pentru toate modurile de transport public şi pentru o întreagă regiune. Trebuie oferite metode de plată simple şi atractive. De exemplu, se pot pune în aplicare sisteme inovatoare de carduri inteligente care pot fi utilizate pentru plata fără contact a biletelor integrate. De asemenea, acestea pot servi drept element important de marketing al transportului public.

Plăţile inteligente pot, de asemenea, furniza date valoroase referitoare la comportamentul şi tiparele de mobilitate ale utilizatorilor.

Exista numeroase beneficii ale utilizării acestor sisteme inovatoare. Ele se impart in e categorii:

- Pentru public: Uşurinţa şi caracterul practic al achiziţiei premise de sistemele inovatoare de emitere a biletelor dintr-un oraş ar trebui să atragă mai mulţi pasageri în transportul public, conducând la un număr mai mic de automobile particulare care intră în zona urbană şi un grad ridicat de satisfacţie al pasagerilor. Accesibilitatea transportului public, în general, este îmbunătăţită prin introducerea unui bilet valabil pentru toate tipurile de servicii şi vehicule.
- Pentru persoanele fizice: Fiecare utilizator al transportului public poate beneficia de un nou sistem de emitere a biletelor deoarece noile oferte sunt mai bine adaptate nevoilor şi tiparelor de călătorie ale fiecărei persoane. La folosirea unui card inteligent sau a unui telefon mobil, pasagerii din transportul public pot economisi bani deoarece se calculează automat cel mai bun preţ pentru călătorii (de exemplu, după un anumit număr de călătorii, pasagerii obţin o reducere de preţ). Dacă sunt prevăzute automate de vânzare a biletelor în staţiile de autobuz sau în vehicule, timpul de îmbarcare se micşorează, iar fiabilitatea şi eficienţa serviciilor de transport public cresc datorită faptului că biletele nu se mai cumpără de la şofer. Un aspect important este, de asemenea, disponibilitatea punctelor de vânzare pentru diferite grupuri de utilizatori (de exemplu, persoane în vârstă sau persoane cu mobilitate redusă).
- Pentru companii Companiile private şi angajaţii lor pot profita de noile sisteme atunci când vânzarea şi subvenţionarea biletelor la serviciile de transport public pentru angajaţi se simplifică. Companiile de transport public beneficiază în special de pe urma acestei măsuri printr-un număr crescut de pasageri generat de serviciu. Prin oferirea de bilete personalizate pentru anumite grupuri de utilizatori, ar putea fi dezvoltate noi pieţe. Sunt create surse suplimentare de informare despre clienţi, oferind date valoroase pentru o analiză suplimentară companiilor de transport public.

La Rochelle (Franța), Preston (Marea Britanie) şi din România Ploiești reprezintă bine oraşul de mărime medie din Europa, care se confruntă cu probleme specifice de transport: suprafaţă şi volum mic, care implică un mai mare amestec şi o mai mare interconexiune a activităţilor; o lipsă a fondurilor, ceea ce poate fi o barieră pentru implementarea tehnologiilor sofisticate; o nevoie de a adopta soluţii politice rapide, lipsa de familiarizare cu proiectele europene de o mare complexitate, periodicitatea utilizării transportului. Aceste oraşe au decis să folosească ultimele tehnologii în ceea ce priveşte autovehiculele curate împreună cu alte măsuri care oferă locuri unde cetăţeni se pot bucura de un mediu de o înaltă calitate, şi unde pot călători uşor şi în siguranţă. De asemenea au decis să înfiinţeze parteneriate locale pentru a aborda problemele cu privire la durabilitatea transportului, să dezvolte sisteme de control eficiente şi să privească transportul urban dintr-o altă perspectivă. [10]

Printre altele aceste măsuri introduse vor demonstra că, folosind un pachet ambiţios în ceea ce priveşte măsurile pentru transport şi pentru dirijarea traficului, vor apărea rezultate pozitive în legătură cu transportul modern şi politica pentru energie: aproximativ 20 km² de zone interzise circulaţiei autovehiculelor şi de zone pentru pietoni în centrele oraşelor; se vor testa noi concepte şi idei pentru distribuirea bunurilor locale, implicând o organizare şi o administrare urbană nouă din punct de vedere logistic; sisteme automate vor fi introduse pentru a oferi

călătorilor informaţii noi, actualizate (folosind date de baze integrate moderne sau prin sistemele GPS) şi introducând strategii integrate de preţ pentru sistemele locale de emitere a biletelor.

Capitolul 2. Obiectivele proiectului

2.1 Obiective proiectului

Se doreşte realizarea unei aplicaţii, care are ca obiective principale emiterea de bilete pentru mijloacele de transport în comun precum şi localizarea acestora pe hartă prin intermediul GPS-ului.

Este un proiect cu un mare caracter practic şi de actualitate, fiind foarte util pentru toate regiile autonome de transport din România, precum şi pentru toate firmele care prestează servicii asemănătoare. Chiar dacă Uniunea Europeană a venit cu o serie de acţiuni, precum CIVITAS (Cleaner and better transport in cities) Iniţiative, care susţin oraşele în punerea în aplicare a unei politici de transport integrate sustenabile, curate şi eficiente din punct de vedere energetic, şi au dezvoltat sisteme inovatoare de plată şi emitere a biletelor pentru transportul public, în România nu există practice încă un astfel de sistem.

Astfel, am decis realizarea unei aplicaţii destinată dispozitivelor mobile care să vină în ajutorul personelor care folosesc aceste mijloace de transport.

Programul trebuie conceput astfel încât să fie o aplicaţie cu o interfaţă prietenoasă şi uşor de folosit de către utilizatori, având ca scop principal permiterea emiterii de bilete pentru mijloacele de transport în comun precum şi ajutorul acordat utilizatorului în localizarea pe hartă a mijlocului de transport dorit.

Bazat pe obiectivele descrise mai sus, precum şi pe funcţionalităţile necesare aplicaţiei, am decis ca sistemul să fie împărţit în patru componente: MobileTicketingAndLocalization Client System (MTL Client System), MobileTicketingAndLocalization WebService (MTL WebService), Ticket Server şi GPS Server.

- MobileTicketingAndLocalization Client System : Este un sistem care prezintă o interfaţă grafică care permite utilizatorului să interogheze şi să obţină un bilet sau informaţii despre mijloacele de transport în comun.
- MobileTicketingAndLocalization WebService : Acest serviciu are responsabilitatea de a furniza servicii web pentru a satisface cerinţele utilizatorului.
- Ticket Server : Server la care se conectează prin Bluetooth aplicaţia clientului. Are responsabilitatea de a interoga MTL WebService şi de a returna un bilet utilizatorului.
- GPS Server : Server care trimite la MTL WebService la scurte perioade de timp locaţia mijlocului de transport în comun.

Astfel, un alt obiectiv al acestei lucrări este proiectarea şi implementarea acestor componente astfel încât acestea să funtioneze conform cerinţelor.

2.2 Specificația proiectului

2.2.1. Cerințe funcționale

Cerințele functionale descriu servicii furnizate către utilizator sau către alte sisteme. Cerințele funcționale pot sa descrie:

- Ce intrări trebuie sa accepte sistemul
- Ce ieșiri trebuie să producă sistemul
- Ce date (care pot fi utilizate de alte sisteme) trebuie sa fie stocate de sistem

- Ce calcule trebuie sa efectueze sistemul
- Temporizarea sau sincronizarea serviciilor (în special in cazul sistermelor de timp real)

După cum s-a specificat si în obiectivele proiectul, aplicația este destinată utilizatorilor de mijloace de transport în comun si are ca cerințe funcționale principale emiterea de bilete, precum si localizarea pe hartă a mijloacelor de transport în comun dorite.

Având în vedere împărțirea întregului sistem în cele patru componente : MTL Client System, MTL WebService, Ticket Server și GPS Server, vom detalia în continuare cerințele funcționale specifice fiecărei componente.

MTL Client System este componenta principală si cea mai semnificativă a sistemului, având in vedere faptul că acesta comunică direct cu clientul acestui întreg sistem. Acesta component este mai exact o aplicație destinată dispozitivelor mobile care ajută clientul în realizarea diferitelor cerinte ale acestuia. Astfel există o serie de cerințe functionale pe care MTL Client System trebuie sa le indeplinescă.

Ca și orice aplicație care folosește datele personale ale clientului, aplicația de față necesită obligatoriu un proces de înregistrare precum și unul de autentificare in sistem. Înregistrarea se realizează când clientul utilizeaza pentru prima dată aplicatia, acest proces fiind necesar o singură data, spre deosebire de autentificare care este necesară de fiecare dată cand un client înregistrat accesează aplicația.

Fiind vorba de o aplicație de emitere de bilete aceasta trebuie să permită clientului să interogheze si să primescă un bilet electronic de fiecare dată cand acesta are nevoie. Acest lucru se realizează prin conectarea prin Bluetooth a aplicației client la server-ul de bilete

Printre alte funcționalități de care dispune acest sistem se numără si aceea de localizare pe harta Google Maps a mijloacelor de transport în comun. Acest lucru se va realiza alegând o linie de mijloace de transport în comun si se vor afișa pe hartă doar autovehiculele corespunzatoare acelei linii.

Pentru a putea realiza funcționalitățile dorite componenta de client interacționează direct sau indirect cu restul componentelor astfel:

- Interacțiunea cu MTL WebService se realizează atât direct cât și indirect. Interacțiunea directă se realizează la operațiile de înregistrare si autentificare în sistem, aplicația apelând serviciul web pentru a introduce datele de înregistrare în baza de date sau prentru a cere o autentificare in sistem interogând baza de date. Interacțiunea indirectă se realizează la emiterea unui bilet, atunci aplicația client trimițand o cerere serviciului web prin intermediul server-ului de bilete.
- Interacțiunea cu Serverul de bilete se realizează direct, aplicația client conectându-se la server si preluând direct biletul electronic.
- Interacțiune cu Serverul GPS nu există, aplicația client preluând datele despre locația mijloacelor de transport în comun din baza de date prin intermediul serviciului web.

Ca o concluzie, acestă componentă importantă a sistemului trebuie să permită următoarele:

- Înregistrarea în sistem: un utilizator de tip „guest” se poate înregistra în sistem prin alegerea unui „username” și a unei parole precum și introducerea unor date personale: nume și prenume.

- Autentificarea în sistem: Acest lucru se realizează cu ajutorul unui „usernme” si a unei parole alese la înregistrarea în sistem.
- Conectarea prin intermediul mobilului la serverul de bilete si preluarea unui bilet electronic.
- Vizualizarea unei liste cu toate mijloacele de transport în comun precum și a unor detalii despre acestea.
- Vizualizarea pe harta Google Maps a locațiilor în care se află la momentul respectiv toate mijloacelor de transport în comun aparținând unei anumite linii.
- Vizualizarea unei liste cu toate stațiile aparținând unei anumite linii de mijloacele de transport în comun.
- Vizualizarea pe harta Google Maps a tuturor stațiilor aparținând unei anumite linii de mijloacele de transport în comun.
- Calcularea și vizualizarea pe harta Google Maps a locației luate prin GPS a clientului care folosește aplicația.
- Calcularea și vizualizarea pe harta Google Maps a unei rute de la locația clientului care folosește aplicația până la o stație selectată de acesta anterior.
- Posibilitatea realizării operațiilor de „move” si „zoom” pe harta Google Maps.

MTL WebService este componenta care reprezintă serviciul web care are responsabilitatea să ofere informații necesare la restul componentelor care îl accesează, precum si de a efectua diferite servicii interogate de componenetele menționate.

Comunicarea cu componenta principală MTL Client System, reprezentând partea de client, impune componentei de servicii web realizarea unor servicii diferite precum: inregistrarea in sistem, autentificarea in sistem, returnarea informațiilor despre stații, returnarea informațiilor despre mijloacele de transport in comun .

Comunicarea cu componenta Ticket Server are ca si scop realizarea operației de emitere de bilete. Astfel serviciul web apelat de catre client prin intermediul server-ului de bilete trebuie sa acceseze baza de date pentru a actualiza datele clientului, precum și returnarea clientului, de asemenea prin intermediul server-ului de bilete, unui bilet electronic.

Acesta componenta de servicii web comunică și cu Server-ul de GPS instalat pe fiecare mijloc de transport care trebuie minitorizat. Acesta comunicare constă în trimiterea locației in care se află mijlocul de transport în acel moment de către serverul GPS, preluarea acestei informații de către serviciul web și actualizarea datelor respectivului mijloc de transport în baza de date.

În concluzie, acestă componentă accesată de toate celelalte trei module ale sistemului trebuie să permită următoarele:

- Preluarea datelor de înregistrare de la client și introducerea lor în baza de date.
- Preluarea datelor de autentificare de la client, interogarea bazei de date si trimiterea unui răspuns corespunzător clientului.
- Preluarea unei interogari de la client referitoare la stații sau la mijloace de transport în comun si returnarea unui răspuns corespunzător.
- Preluarea datelor unui client de al server-ul de bilete si returnarea acestuia un bilet electronic.
- Preluarea datelor despre locația mijloacelor de transport si actualizarea acestora în baza de date.

Serverul de bilete este componenta care interactioneaza direct cu aplicația client in procesul de emitere de bilete. Acest server este un fel de componentă intermediară intre aplicația client și serviciul web care emite biletul respectiv.

Comunicarea cu aplicația client se realizează prin intermediul Bluetooth-ului. Odată conectat la server-ul de bilete clientul nu are de făcut nimic decât sa aștepte preluarea biletului electronic. Astfel, odată ce un client s-a conectat, serverul preia date esențiale ale clientului si le trimite la serviciul web. După acest pas serverul așteaptă răspunsul serviciului web, iar cand acesta vine, clientul primește biletul electronic, bineînțeles prin intermediul componentei reprezentând serverul de bilete.

Astfel, această componentă necesară procesului de emitere de bilete trebuie sa permită următoarele:

- Preluarea datelor esențiale de la aplicația client câand acesta se conecteză la server.
- Apelarea serviciului web, preluarea datelor de răaspuns de la acesta (bilet electronic) și trimiterea răspunsului obținut către clientul respectiv.

Serverul GPS este cea de-a patra componentă a întregului sistem de emitere de bilete si localizare a mijloacelor de transport în comun. Acesta este instalat pe mijloacele de transport și are rolul de a oferi informații despre localizarea acestuia din urmă.

Acestă componentă comunică doar cu MTL WebService în vederea actualizării in baza de date a poziției mijlocului de transport în comun. Astfel odată la o periadă scurtă de timp server-ul GPS apelează serviciul web trimițându-i acestuia datele privind locația sa. Serviciul web preia datele și actualizează baza de date.

2.2.2. Cerințe non-funcționale

Portabilitate: J2ME aduce conceptul de portabilitate la nivelul dispozitivelor mobile, oferind un mediu optimizat, bazat pe Java, de dezvoltare a aplicațiilor pentru acest segment software. Astfel aplicația va rula pe majoritatea dispozitivelor mobile, indiferent daca acestea sunt smatphone-uri sau telefoane mobile normale care au în specificație Java ME.

Utilizare uşoară: Acest lucru nu înseamnă doar interfaţă utilizator prietenoasă. Aplcația software client interacționează cu alte sisteme software (server de bilete). Utilizarea uşoară se reflectă şi în uşurinţa de comunicare a sistemului şi de adaptare a acestuia la mediul software.

Fiabilitate: Reprezintă probabilitatea ca un sistem software să funcţioneze fără erori pentru o perioadă de timp precizată, într-un mediu dat. Sistemul de emitere de bielte trebuie să fie fiabil, acesta necesitând să funcționeze corect fără apariția unuei erori grave care ar avea consecințe serioase.

Robustețe: Un program robust se comportă „rezonabil”, chiar în condiţii care nu au fost anticipate la specificarea cerinţelor. Sistemul trebuie să fie pregătit pentru rezolvarea problemelor survenite în urma introducerii unor date de intrare incorecte sau în urma unei disfuncționalități hardware (de exemplu: defectarea unui hard disc).

Accesibilitate: Sistemul trebuie să răspundă în orice moment la cererile clientului, acesta din urmă necesită folosirea aplicației la orice oră în decursul unei zile. Indisponibilitatea sistemului va duce la consecințe neplăcute pentru client.

Capitolul 3. Studiu bibliografic

3.1 Emitere de bilete pe mobil

Conform articolului [3] pentru a se evita numeroasele dezavantaje ale sistemelor de emiterilor de bilete clasice existente, cum ar fi necesitatea printării acestora sau plătirea unui preț foarte ridicat pentru livrarea biletului in timp foarte scurt, s-a dezvoltat o noua tehnologie numită „mobile ticketing" sau emitere de bilete pe mobil. Biletele comandate vor fi livrate către client direct pe telefonul mobil al acestuia prin diverse moduri. Biletul va consta intr-un cod de bare care va fi scanat direct de pe telefonul mobil.

Acest sistem (emitere de bilete) este destinat utilizatorilor de mobile care au în specificația lor tehnologia MMS sau WAP activată. MMS sau „Multimedia Messaging Service" este varianta opusă „Short Messaging Service" (SMS), aceasta din urma fiind folosită pentru trimiterea mesajelor text simple. MMS-urile pot include text, imagini, poze, clipuri audio și chiar clipuri video, depinzând de dimensiunea acestora. WAP este prescurtarea de la „Wireless Application Protocol" și este o tehnologie standard care permite dispozitivelor fără fir să navigheze pe Internet sau sa ruleze aplicaţtii mobile. Telefoanele mai vechi cu WAP activat pot naviga pe paginile Web doar cele destinate telefoanelor mobile, în timp ce telefoanele de nouă generație pot vedea și naviga pe toate tipurile de pagini Web. Dacă telefonul are cameră foto atunci automat are capabilități de MMS, iar dacă acesta are opțiune de navigare pe Web atunci sigur există și tehnologia WAP inclusă. Un plus pentru biletele electronice este acela că acestea pot fi reprezentate fie prin coduri de bare, fie prin coduri alfanumerice (serii de cifre si litere). Astfel chiar dacă telefonul nu poate afișa imaginea cu codul de bare, cei care se ocupă de verificarea biletelor o pot face prin introducerea manuală a codului.

Tot în articolul [3] se prezintă o serie de avantaje ale acestui sistem inovativ. Biletele electronice pe mobil au potențialul de a putea fi folosite oriunde se pot folosi si biletele clasice actuale. Multe sporturi care dețin stadioane moderne sau la diferite concerte se folosesc deja cititoare de coduri de bare pentru a procesa biletele de hârtie, deci tehnologia este deja existentă. Emiterea de bilete pe telefoane mobile poate fi utilizată la evenimente sportive, concerte, cinema-uri, transport și altele. Cel mai mare avantaj al acestei tehnologii este comoditatea. Condiția necesară este posesia unui telefon cu WAP sau MMS, se trimite o comandă pentru un bilet, biletul se stochează în telefon și verificarea se face la evenimentul sau mijlocul de transport respectiv astfel evitându-se așteptările nedorite la rând.

De asemenea emiterea de bilete electronice pe mobil poate crește veniturile promotorilor de evenimente și a vânzătorilor de bilete. Aceștia pot vinde bilete și cu un minut înaintea începerii evenimentului deuarece livrarea pe mobil este instantanee. Biletele electronice reduc costul procesării la ambele părți. Vânzătorul nu trebuie să plătească pentru printarea si livrarea biletelor, acest lucru fiind valabil și pentru cumpărătorul biletului. Și în plus mai puțină hîrtie este un lucru foarte bun pentru mediul înconjurător. Nu în cele din urmă biletele electronice sun mai greu de falsificat, o serie de măsuri de securitate putând fi adăugate pentru a face ca orice tentativă de fraudă să fie aproape imposibilă. Biletul poate fi „blocat" pe telefonul clientului, astfel încât acesta să nu poată fi trimis mai departe. Numele si chiar poza clientului poate fi adăugată la bilet pentru confirmarea acestuia atunci cân acest lucru este necesar. Chiar dacă un bilet electronic este pierdut sau mesajul este șters din greșeală de pe telefon, este destul de ușor vânzătorului să anuleze vechiul bilet și să retrimită unul înlocuitor. În Japonia, există automate de Coca-Cola numite „cmode", care acceptă plata prin telefonul mobil utilizînd coduri de bare, și chiar unele taxiuri au început să accepte acest tip de plată.

În articolul [4] este prezentată o soluție pentru un sistem de emitere de bilete electronice pe mobil utilizând coduri de bare. Mai jos este ilustrat grafic un examplu de funcționare a unui astfel de sistem.

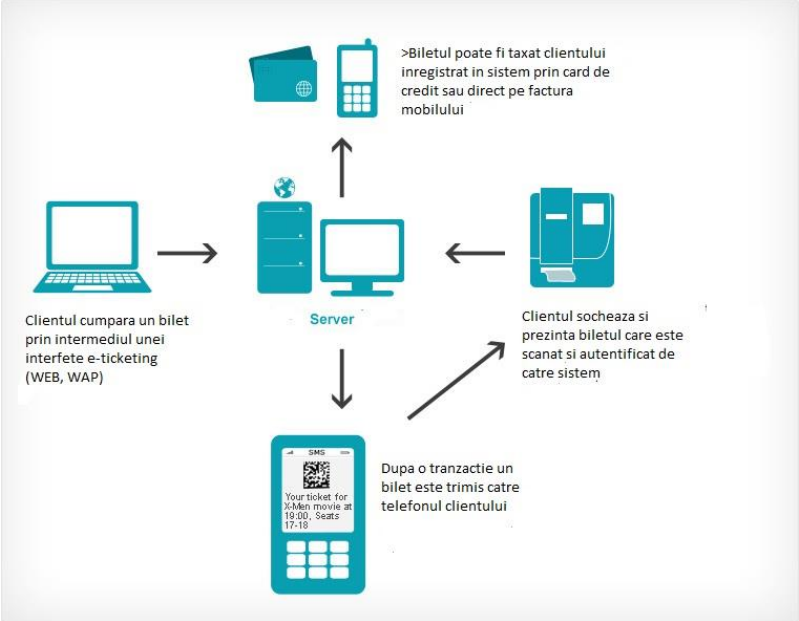


Figura 3-1 Sistem de emitere bilete pe mobil [4]

În [2] sunt prezentate o serie de tehnologi care ajută la comunicarea dintre un furnizor de servicii şi un client prin intermediul telefonului mobil. Există două soluții principale din care se poate alege: comunicarea prin Bluetooth sau comunicaera prin SMS\MMS.

Vom studia în continuare cele două tehnologii şi vom trage o serie de concluzii. Conform [3] la tehnologia Bluetooth informaţia poate fi împinsă spre utilizatorii care au activate dispozitivele lor pentru bluetooth si apoi transmisă de îndată ce utilizatorul a fost de acord să primească. Informaţii trimise pot fi text, audio, imagini sau video. 95% din telefoanele mobile vândute astăzi sunt cu tehnologia Bluetooth încorporată, procent care a crescut cu 5% începând cu anul 2008. Statisticile arată că peste 70% dintre consumatori lasă bluetooth-ul pornit pe tot parcursul zilei şi prin stimulente acest număr are un potenţial de creştere. Tehnologia Bluetooth poate fi utilizată atât în interior cât şi în aer liber cu același succes. Zone cum ar fi centrele comerciale, expoziţii, stadioane sportive şi zone publice sunt ideale, dar la fel de mult oamenii folosesc această soluție pentru reţele private şi de distribuţie a informaţiei. Bluetooth este gratis, prin urmare, o dată ce s-a investit în echipament, numărul de transmisii sunt gratuite indiferent dacă acesta este de 1 sau 1 milion de persoane l-a care se conectează. Prin tehnologia Bluetooth nu se pot trimite așa numitele „spam-uri” deurarece spre deosebire de SMS toate datele primite trebuie mai înainte acceptate de către cel care primește. Server-ul recunoaște si salvează orice telefon care a refuzat un transfer pentru a asigura că mesajul nu va fi retransmis către acesta. Se pot transmite fișsiere de până la 1MB cu viteză bună, fișierele mai mari transmițându-se mai greu. Transmisiile Bluetooth, când vine vorba despre distanță sunt împărțite în trei clase : Clasa 1 – până la 100 metri, Clasa 2 : până la 10 metri, Clasa 3 : până la 1 mentru. Se pot folosi de altfel antene care pot mări distanța de transmisie. [3]

Având în vedere argumentele aduse putem arăta o comparație intre tehnologia Bluetooth și cea prin SMS/MMS prezentând la fiecare avantajele si dezavantajele folosirii lor.

- Bluetooth:
  - Avantaje:
    - Standard folosit și acceptat de o gamă largă de public
    - Rată de transfer de date mare
    - Securitate (la nivelul protocolului)
    - Transmisii de date gratuite
  - Dezavantaje:
    - Limitat de distanțe scurte
- SMS/MMS:
  - Avantaje:
    - Flexibil (poate fi folosit oriunde)
    - Transfer de date (text/media) pe distanțe mari
  - Dezavantaje:
    - Rată de transfer de date mică
    - Transmisiile de date nu sunt de obicei gratuite
    - Succes mai mic în domeniul comercial

În concluzie singurul mare avantaj al tehnologiei SMS/MMS față de tehnologia Bluetooth este distanța superioră la nivelul transmiisiilor de date. Însă folosirea unei strategii mai bine orientate către client, este mai avantajoasă deuarece destinatarul este în zona corectă atunci când acesta primește datele, și astfel datele sunt păstrate relevante și în timp real. [3]

Conform [1] telefoanele mobile vor inlocui in urmatorii ani portofelele si orice fel de bilet de intrare sau de transport. Procesul va incepe chiar din acest an, fiind foarte posibil ca biletele de autobuz, cărtile de credit si cardurile de fidelitate sa fie incorporate in telefoanele mobile. Un raport publicat recent spune ca telefoanele mobile vor inlocui orice fel de bilet de hârtie, începand cu cele pentru cinematograf si terminând cu cele de avion. Potrivit studiului, companiile aeriene si cele de căi ferate vor lansa noi moduri de a emite bilete electronice pe telefoanele mobile, acestea urmand sa fie folosite prin sms, coduri de bare sau aplicatii care trebuie descărcate. Autorul raportului este de părere ca in anul 2014 vor fi emise 15 miliarde de bilete pe telefoanele mobile, față de doar 2 miliarde in 2010.. În prezent, aceasta metodă de a cumpara bilete este foarte populară in Japonia, unde tehnologia telefoanelor mobile este cea mai dezvoltată din lume. Un alt studiu pe aceeasi tema sustine ca, din acest an, telefoanele mobile vor inlocui portofelele, având ăn vedere că gările si magazinele au inceput deja să folosească o tehnologie wireless care permite comunicarea datelor necesare la distanta mica prin mobil.

3.2 Programe similare

3.2.1 Maxartists Mobile Ticketing Delivery System (MTDS)

Maxartists Technologies a fost infiintata in anul 2005 la Trivandrum, Kerala, cea mai recentă atragerea de IT Hub din India, pentru puţinii experţi ai acestei industrii de pe tot globul, cu o viziune de a deveni înaintaşii în tehnologiile emergente. [5]

Maxartists au dezvoltat sistem mobil de emitere de bilete bazat pe cerinţele clientului şi a fost implementat cu succes pentru diferite sporturi, concerte, teatre şi alte evenimente. Soluţia este destinată pentru distribuirea şi administrarea de bilete la diferite industrii cum ar fi

cinematografe,teatre,concerte,sport,festivităţi,etc. Mobile Ticketing Delivery System (MTDS) este o aplicaţie uşor de utilizat, în care mai mulți bilete furnizorii de bilete pot gestiona clienții lor și trimiterea biletelelor bazate pe coduri de bare 2D direct pe telefonul mobil ca paşaport de intrare pentru diverse evenimente.

Avantajele acestui sistem sunt urmatoarele:

- Poate fi folosit oriunde si oricand.
- Reduce cheltuielile.
- Creşte eficienţa de operare.
- Securitate pentru a ajuta la lupta împotriva fraudei.
- Permite cumpărătorilor de bilete să evite cozile lungi
- Îmbunătăţirea confortului consumatorilor.
- Reducerea costurilor de infrastructură.
- Elimină tipărirea şi distribuirea costurilor.
- Oferă servicii mai bune pentru clienţi.
- Biletele au evoluat de la biletele de hârtie, la bilete informatizate, care devin din ce în ce mai populare şi sunt mai uşor de utilizat.

Biletele electronice pe mobil sunt atractive în următoarele domenii:

- Transport: Toate transporturile publice majore, cum ar fi cel aerian, feroviar, autobuz, metrou, pe mare, Feribot, etc
- Evenimente sportive: Toate evenimentele majore profesionale sportive, cum ar fi fotbal, baseball, cricket, hochei, fotbal american, etc
- Distracţii & Evenimente: Include cele mai multe evenimente de divertisment live, cum ar fi concerte, teatru, cinema, muzee, galerii, expozitii, parcuri de distractii, etc

Mai jos sunt ilustrate principalele componenete ale sistemului Mobile Ticketing Delivery System (MTDS) și modul ăn care acestea interacționează între ele. [5]

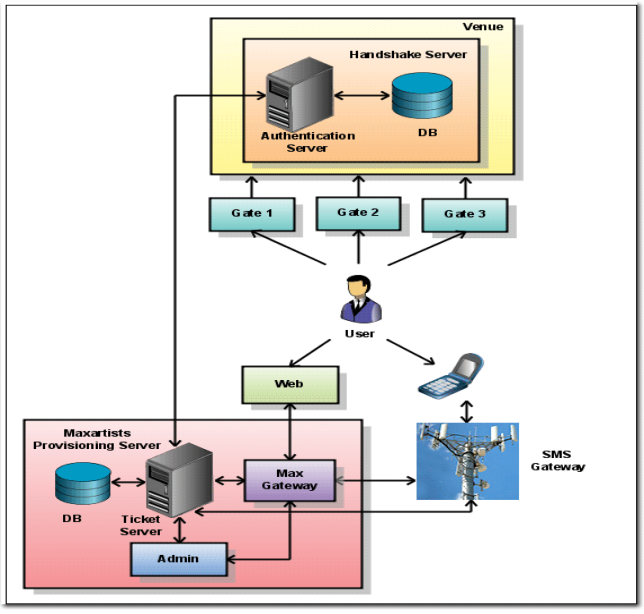


Figura 3-2 Componente MTDS [5]

De la interogarea sistemului de către client pentru a obține un bilet ți până la verificarea biletului există o serie de evenimente intermediare care se desfășoară în următoarea ordine:

- Utilizator trimite o solicitare pentru un bilet, prin intermediul Mobil / Web.
- Un cod de bare (1D / 2D) bilet este trimis ca SMS la telefonul utilizatorului final.
- Pentru a avea acces la locul de desfăşurare, utilizatorul arată telefonul cu biletul la un cititor electronic existent în locul respectiv.
- În cazul în care biletul este valabil, utilizatorul are acces la eveniment sau dovedește existența biletului, aceasta din urma în special în cazul transporturilor unde verificarea nu se face neapîrat la intrarea în mijlocul de transport.

Plata poate fi tratate în oricare dintre următoarele modalităţi existente bazate pe cerintele clientului:

- Operatori Telecom - clientul trebuie sa aiba o legătură cu un furnizor de servicii de telefonie. Astfel suma biletului vor fi deduse de către furnizorul de telecomunicaţii din contul utilizatorul dispozitivului mobil, atunci când utilizatorul primeşte un bilet bazat pe coduri de bare în telefonul său mobil.
- Carduri de plată - Utilizator se pot efectua plăţi utilizând orice din cărţile sale de plată, cum ar fi card de credit, card de debit, etc

Mobile Ticketing Delivery System suportă coduri de bare 1D și 2D. Un cod de bare 2D este similar cu cod de bare 1D, dar are mai mult capacitatea de reprezentare a datelor. Tipuri de coduri de bare utilizate în soluţia noastră sunt: Data Matrix, QR Code, Aztec, EAN 13, PDF 417, CODE 128, CODE 39, CODE 25i.

Mai jos este ilustrată arhitectura sistemului Mobile Ticketing Delivery System alcătuită din partea de Server, partea Web și partea de Mobil. [5]

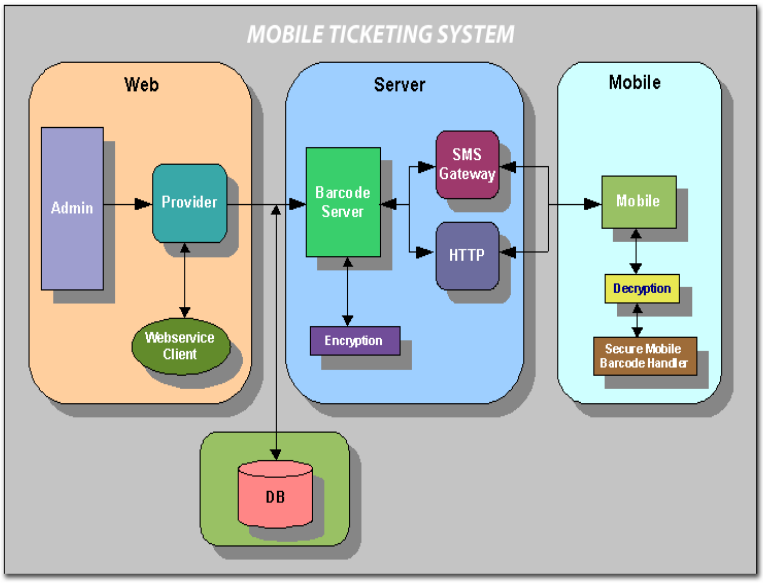


Figura 3-3 Arhitectura sistemului MTDS [5]

Maxartists au dezvoltat soluții mobile de emitere de bilete pentru clienți diferiți pe baza cerințelor clientului.Soluțiile de „ticketing”, pe care le poate oferi pentru client sunt:

- MTDS: Mobile Ticketing Delivery System: este partea de componente server pentru generarea şi trimiterea legitimaţiilor de transport prin SMS, MMS şi WAP Push, acestea incluzând un cod de bare 2D pentru telefoanele mobile
- MTDS Provider: sunt furnizori care gestionează clienţii lor şi le trimit biletele cu coduri de bare 2D direct pe telefonul mobil ca SMS sau MMS şi prin e-mail ca paşaport de intrare pentru diverse evenimente.
- MTS (Mobile Ticketing Shop) : este o interfaţă publică, în care utilizatorii se pot înregistra în site-ul web şi pot cumpăra bilete pentru un eveniment.
- Mobile Client: o aplicaţie „portofel” va fi instalată în telefon mobil al utilizatorului pentru manipularea biletelor bazate pe coduri de bare.

În continuare este ilustrată o comparație între MangoTix – Mobile Ticketing și sistemul de emitere de bilete si localizare de mijloace de transport implementat.

Tabel 3-1 Comparație MTDS – MobileTicketingAndLocalization System

Caracteristici	MTDS	MTL System
Mod preluare bilet	Web/SMS	Bluetooth
Bilet QRCode	Da	Da
Plată prin card/factură telefon	Da	Nu
Localizare mijloace de transport	Nu	Da
Vizualizare mijloace de transport pe hartă	Nu	Da

3.2.2 MangoTix Mobile Ticketing

Utilizarea platformei sale brevetate și rapide de dezvoltare a aplicaţiilor, JMango a dezvoltat o aplicaţie unică de mobile ticketing numită MangoTix. Cu o integrare a clientului perfectă şi fără efort, aplicaţia MangoTix este o soluţie completă de mobile ticketing, permiţând utilizatorilor să comande, plătească, primească şi să răscumpere bilete la evenimente.

Aplicaţia MangoTix poate funcţiona în cadrul unei game largi de clienti diferiti, de la mari case de ticketing la agenţiile de tip boutique, evenimente individuale, promotori şi oricine altcineva care caută o soluţie unică de ticketing. Soluţia este, de asemenea, ecologică, cu o opţiune sigură și informatizată de ticketing pentru cei „eco-minded”. La fel ca cele mai multe aplicaţii JMango, MangoTix poate fi simplu şi uşor de integrat pe serviciile existente de plăţi, permiţând integrarea imediată şi securizată a acesteia

Următoarea diagramă ilustrează procesul buclă închisă de în mobile ticketing. Acest proces este implicat in ambele situații pre și în timpul activităţilor evenimentului. [6]

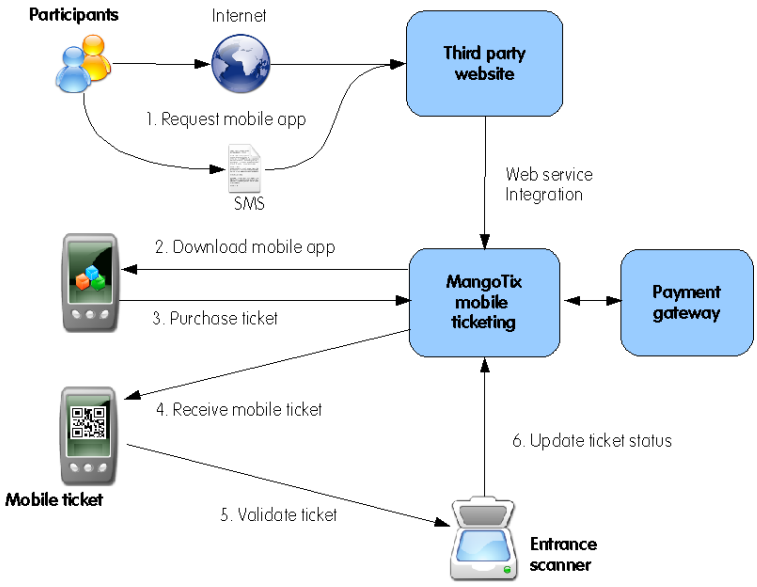


Figura 3-4 MangoTix - Proces de emitere de bilete [6]

Primul pas în acest proces este de a descărca aplicaţia mobilă pentru eveniment, care implică efectuarea unei cereri de către utilizator (1) pentru aplicaţie (fie prin intermediul site-ului sau prin mesaj SMS). MangoTix furnizează un API de integrare în sistem pentru a permite site-urilor Web să se conecteze direct la sistem şi să trimită aplicaţia, precum şi crearea şi emiterea de bilete. Un mesaj conţinând un link de descărcare este trimis la mobilul clientului, de unde acesta poate prelua bilete sau sa acceseze informaţii despre evenimente.

Odată ce aplicaţia este pe telefonul mobil, utilizatorii au posibilitatea de a achiziţiona bilete direct din meniul principal (3), folosind un card de credit care sunt prelucrate prin intermediul unui gateway de plată securizat. Odată ce tranzacţia este sa dovedită, o imagine 2D QRCode unică criptată va fi trimisă către telefonul mobil al utilizatorului (4), precum şi precum şi la telefoanele mobile ale altor clienţi pentru care utilizatorul a cumpărat bilete. Imaginea cu codul QRCode va fi stocată atât în aplicaţie cât şi în SMS.

În ziua evenimentului, utilizatorul va prezenta biletul cu codul unic QRCode şi vor trece acest cod la intrare (5) pentru un proces rapid de confirmare. Scanerul de la intrare verifică biletului şi actualizează starea acestuia (6), care poate fi apoi folosită pentru a declanşa conexiuni suplimentare, cum ar fi mesaje de bun venit.

Mai jos sunt ilustrate o serie ce „screenshoots” cu operaţiile de interogare si recepţionare a unui bilet. [6]

Figura 3-5 MangoTix – Interogare/Recepţionare bilet [6]

După operația de download a aplicației utilizatorul poate vedea o serie de informații despre prețtul biletelor, iar mai apoi sa aleagă să cumpere un bilet, selectând înainte detaliile de plată cu cardul. După acest prim proces utilizatorul poate vizualiza informații despre bilet precum și imaginea cu codul QRCode după cum se poate vedea in figura de mai jos. [6]

Figura 3-6 MangoTix – Vizualizare bilet QRCode [6]

În continuare este ilustrată o comparație între MangoTix – Mobile Ticketing și sistemul de emitere de bilete si localizare de mijloace de transport implementat.

Tabel 3-2 Comparație MangoTix – MobileTicketingAndLocalization System

Caracteristici	MangoTix	MTL System
Mod preluare bilet	Web/SMS	Bluetooth
Bilet QRCode	Da	Da
Plată prin card/factură telefon	Da	Nu
Localizare mijloace de transport	Nu	Da
Vizualizare mijloace de transport pe hartă	Nu	Da

Capitolul 4. Analiză şi fundamentare teoretica

4.1 Fundamentare teoretică

Acest subcapitol va prezenta o sinteză teoretică a technologiilor precum şi a design pattern-urilor folosite. Astfel se prezinta technologii alese şi folosite pentru dezvoltarea sistemului atât la nivel de structura a aplicaţiei , comunicare cu servicii Web cât si la nivelul interfeţei si legaturii dintre ele.

4.1.1 Tehnologii si resurse utilizate

NetBeans IDE 6.9.1 şi Java ME

Platforma Java Micro Edition (Java ME), cunoscută anterior sub numele de J2ME, oferă un mediu flexibil pentru aplicații destinate telefoanelor mobile şi a PDA-urilor. Java ME include interfețe utilizator flexibile, securitate si suport pentru aplicațiile online si offline. Aplicațiile bazate pe Java ME sunt portabile pe majoritatea dispozitivelor.

Daca privim tehnologia Java din punct de vedere statistic, 3 miliarde de telefoane mobile suporta platforma Java. În fiecare an, numarul de telefoane noi ce ruleaza aplicații Java este de 31 de ori mai mare decat cele de la Apple si cele cu Android împreună.

Mai jos avem ilustrată arhitectura platformei Java Micro Edition (Java ME).

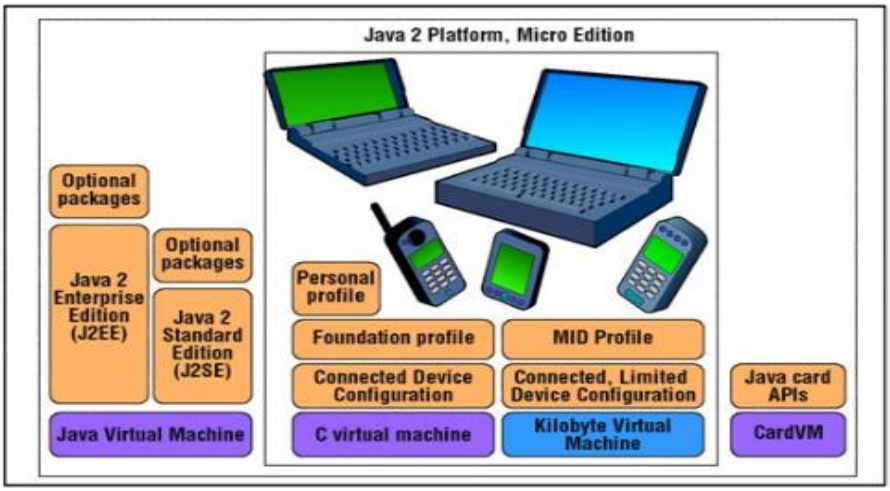


Figura 4-1 Arhitectura platformei Java ME

După cum putem observa arhitectura platformei J2ME este imparțită pe trei nivele:

- Nivelul Mașinii Virtuale
- Nivelul Configurărilor
- Nivelul Profilelor

O Configurare este o specificație care definește o funcționalitate Java Platform minima pentru o familie de dispozitive. Definește numărul minim de librării Java, capabilități VM si specificații de securitate.

- Connected Device Configuration (CDC)
  - Folosit de dispozitivele care sunt mai puternice decat dispovitivele low end precum telefoanele mobile (de exemplu: PDAs…)
  - 32 bit processor
  - >.5MB (ROM+RAM)
  - JVM suport
  - High bandwidth network connectivity
- Connected Limited Device Configuration (CLDC)
  - Folosit de dispozitivele cu resurse reduse (de exemplu: telefoane mobile…)
  - 16-32 bit processor
  - >160-512kB (ROM+RAM)
  - KVM suport
  - Limited power/battery, network, GUI, Java class libraries

Un Profil este o colecție de Java APIs care completează o Configurare pentru a furniza capabilitați pentru un anumit grup de dispositive.

- Mobile Information Device Profile (MIDP)
  - Furnizeaza o interfață pentru utilizator, funcționalitați de gaming si multimedia, securitate end-to-end, precum si conectivitate in rețea pentru telefoanele mobile.
- Foundation Profile
  - Un set de APIs care suporta dispozitive cu resurse limitate fără un sistem standard de GUI.
- Personal Profile
  - Împreună cu CDC si Foundation Profile, furnizeaza un mediu complet de aplicatii pentru piața high-end de PDAs. Personal Profile conține un set complet de AWT APIs , incluzând suport pentru applets si Xlets.

MIDP este o versiune a platformei Java bazata pe CLDC si KVM care țînteste aceasta categorie de dispozitive, in special telefoane mobile si pagere. MIDP este deja implementat si folosit de milioane de telefoane. Avantajele acestuia sunt:

- portabilitate - avantajul de a folosi Java față de alte instrumente pentru dezvoltarea aplicatiilor este portabilitatea. Se poate scrie cod si in alte limbaje, cum ar fi C/C++, dar rezultatul ar fi dependent de platformă. O aplicatie scrisă folosind API-urile MIDP poate fi rulată pe orice telefon cu MIDP. Acest principiu Java se numeste "Write Once, Run Anywhere"(WORA - "Scrie o data, rulează oriunde").
- securitate - un al doilea motiv important in favoarea platformei Java este securitatea. Platforma Java este bine cunoscută pentru abilitatea de a downloada si rula aplicatii de dimensiuni mici, cum ar fi applet-urile. De asemenea, aplicatiile nu pot accesa nimic din afara masinii virtuale, neinterferând cu alte aplicatii sau chiar cu sistemul de operare.

Un MIDlet este o aplicație MIDP sau o clasă care extinde clasa MIDlet si reprezintă interfata dintre instructiunile aplicației si mediul de execuție, care este controlat de managerul de aplicație. O clasă MIDlet trebuie sa conțină 3 metode abstracte ce sunt apelate de manager pentru a gestiona ciclul de viată al unui MIDlet. Aceste metode abstracte sunt:

- startApp(): apelată în momentul in care MIDlet-ul este pornit și conține instrucțiunile ce se vor executa de fiecare dată cand aplicația este lansata in execuție

- pauseApp(): apelată înainte ca managerul de aplicație sa întrerupă temporar MIDlet-ul. Managerul de aplicație restartează MIDlet-ul prin reapelarea metodei startApp().
- destroyApp(): apelată înainte de terminarea MIDlet-ului. Metoda publică fără valoare de retur. Are un parametru boolean care este setat la true (adevarat) dacă terminarea MIDlet-ului este necondiționată, si false (fals) dacă MIDlet-ul poate arunca o excepție MIDletStateChangeException.

Ciclul de viață al unui MIDlet este ilustrat in figura de mai jos împreună cu metodele sale obligatoriișsi modul lor de funcționare.

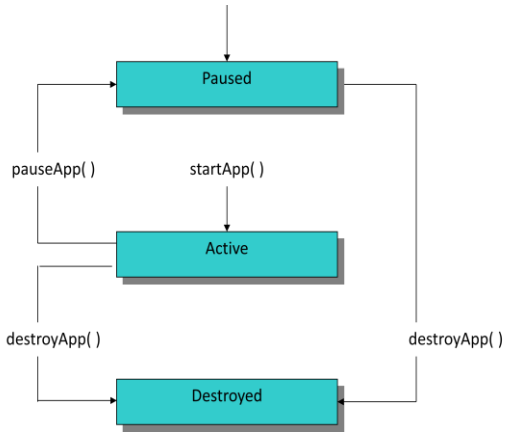


Figura 4-2 Ciclu de viață al unui Midlet

Având în vedere cele descrise mai sus putem spune că platforma Java ME este folosită pentru a crea aplicații mobile portabile. Acestea au in jur de 1MB si vor rula pe majoritatea telefoanelor existente.

Java ME Bluetooth 1.1 (JSR 82)

API-ul Java pentru Bluetooth este un pachet optional pentru J2ME definit de Java Community Process (JSR-82). Acest pachet opţional oferă o interfaţă comună pentru dezvoltarea Bluetooth. Figura de mai jos ilustrează relaţia dintre API Java pentru Bluetooth şi platforma J2ME, folosind stiva Mobile Device Profile Information (MIDP) şi Connected Limited Device Configuration (CLDC).

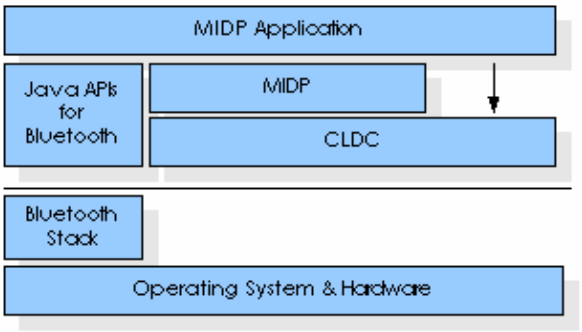


Figura 4-3 Relaţie API Java Bluetooth – J2ME

Utilizarea API-ului Java pentru Bluetooth constă în iniţializarea stivei Bluetooth, descoperirea dispozitivelor sau serviciilor care sunt în imediata vecinătate, deschiderea,

închiderea şi aşteptarea sau iniţierea unei conexiuni, şi de a efectua operaţii I / O. Figura de mai jos ilustrează diferitele operaţiuni Bluetooth care se pot efectua la cererea utilizatorului.

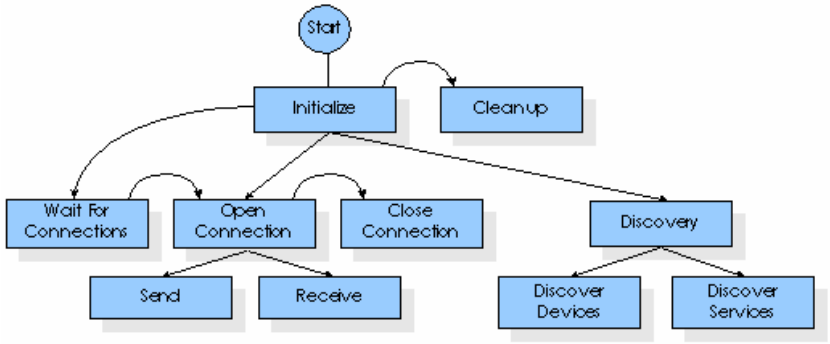


Figura 4-4 Operaţii Bluetooth

API Java pentru Bluetooth se bazează pe CLDC 1.0 Generic Connection Framework(GCF). Clasele şi interfeţele aparţinând acestui framework sunt următoarele:

- Clasa Conector care este o clasă de tip Connection factory. În sprijinul conexiunilor RFCOMM şi OBEX API-ul Bluetooth foloseşte tipurile de conexiuni GCF deja existente: streamConnection (si indirect ImputConnection si OutputConnection) şi StreamConnectionNotifier. Pentru conectivitatea L2CAP API-ul introduce două noi tipuri de conexiuni: L2CAPConnection şi L2CAPConnectionNotifier.
- StreamConnection este o subinterfaţă pentru ambele interfeţe InputConnection si OutputConnection. Acestea din urmă returnează fluxuri de date prin metode ce permit citirea şi scrierea datelor. StreamConnectionNotifier este o interfaţă care reprezintă o conexiune stream pe partea de server. StreamConnectionNotifier defineşte o singură metodă, acceptAndOpen() care asteaptă şi deschide conexiunile stream care vin.
- L2CAPConnection este o subinterfaţă a Conexiunilor şi metodelor de citire si scriere de date primare, şi de descoperire a Maximum Transmission Unit (MTU) trimis şi primit. MTU defineşte numărul maxim de octeţi care pot fi trimişi şi primiţi fără pierderi de date. L2CAPConnectionNotifier este similar cu StreamConnectionNotifier, dar în acest caz acesta reprezintă o parte de server pentru o conexiune de tip L2CAP.

Datele pentru utilizatori sunt de obicei transferat spre şi de la layerul Logical Link Control and Adaption Protocol (L2CAP) din stiva Bluetooth.

Datorită caracterului ad-hoc a reţelelor Bluetooth, dispozitivele Bluetooth se vor muta in şi în afara intervalului de acoperire frecvent. Astfel acestea din urma trebuie să aibă abilitatea de a detecta dispozitivele Bluetooth din apropiere. Cand un dispozitiv este gasit se initializeaza un "service discovery" pentru a determina ce serviciu oferă acel dispozitiv. Specificaţia Bluetooth asociaza operaţiunea de "device discovery" termenului "inquiry" (ancheta). În timpul procesului de "ancheta" dispozitivele Bluetooth anchetate" vor primi adresa si ceasul Bluetooth de la dispozitivele descoperite în apropiere putând astfel sa se sincronizeze cu acestea. Cele două procese amintite sunt descrise mai în detaliu mai jos:

- Device discovery (descoperire de dispozitive)

  Procesul de device discovery constă în apelarea a trei metode existente:

- startInquiry() : este apelată atunci când se pornește procesul de device discovery.
- deviceDiscovered() : este apelată când se găsește un dispozitiv.
- inquiryCompleted() : este apelată când se procesul de device discovery ia sfarșit.

Mai jos este ilustrat grafic cum se comportă aceste trei clase pe parcursul procesului de descoperire de dispozitive.

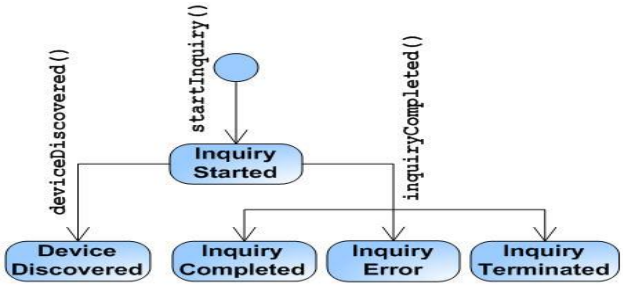


Figura 4-5 Descoperire de dispozitive Bluetooth

- Service discovery (descoperire de servicii)

  Procesul de service discovery constă de asemenea în trei metode asemănătoare cu cele de la device discovery.
  - searchServices() : este apelată atunci când se pornește procesul de service discovery.
  - serviceDiscovered() : este apelată când se găsește un serviciu.
  - serviceSearchCompleted() : este apelată când se procesul de device discovery ia sfarșit.

Mai jos este ilustrat grafic cum se comportă aceste trei clase pe parcursul procesului de descoperire de servicii.

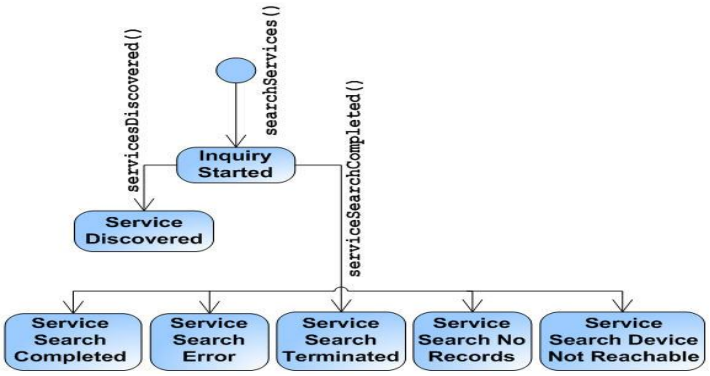


Figura 4-6 Descoperire de servicii Bluetooth

Având în vedere cele descrise mai sus putem spune că cei doi protagoniști ai unei aplicații Bluetooth Client – Server se vor comporta în felul următor:

- Server
  - Configurare setări de securitate
  - Creare serviciu Bluetooth
  - Așteptare conexiuni de la clienți
  - Trimitere/Recepționare date

- Client
  - Configurare setări de securitate
  - Căutare dispozitive Bluetooth cu serviciul dorit
  - Conectare la serviciu
  - Trimitere/Recepționare date

API de localizare pentru Java ME (JSR-179)

Scopul API-ului Location API definit de Java ME este de a permite dezvoltarea de aplicaţii bazate pe localizare de mobile. Având în vedere caracterul dispozitivelor mobile, API-ul API-ului Location API oferă o modalitate naturala de a utiliza informaţii bazate pe locaţie. În plus, acesta este un pachet compact de clase si interfețe care sunt ușor de utilizat. Cele trei caracteristici principale care API-ul le aduce programării pe mobile sunt:

- obţinerea de informaţii cu privire la amplasarea unui dispozitiv
- posibilitatea de a crea, edita, stoca, regăsi şi repere
- posibilitatea de a obţine orientarea unui dispozitiv

Location API are nevoie de o conexiune la o metodă de furnizare de locaţie, care genereaza locaţii. Metodele defurnizare Location API diferă între ele în multe feluri. De exemplu, utilizarea unor metode poate costa mai mult decât altele, şi preciziile suportate de individual de metodele de furnizare de locație variază. Cele mai comune metode sunt cele bazate pe dispozitiv (de exemplu, modulul GPS - o metodă bazată pe sateliţi Global Positioning System), bazate pe reţea (de exemplu, celule de origine - o metodă în care reţeaua determină locul unui utilizator), sau metode hibride (de exemplu, A-GPS : o metodă GPS care utilizează, de asemenea, informaţii bazate pe reţea pentru a accelera determinarea locului de amplasare).

Figura de mai jos prezintă o structura generalizată a unui Location API MIDlet API care utilizează un modul GPS ca o metodă de furnizare de locaţie. După ce MIDlet-ul dovedeşte că funcționează în mod corespunzător în mediul SDK, ar trebui să fie, de asemenea, testat în conformitate cu condiţiile din lumea reală. De obicei, lumea reală de testare înseamnă folosirea în aer liber a MIDlet-ului într-un dispozitiv care acceptă Location API.

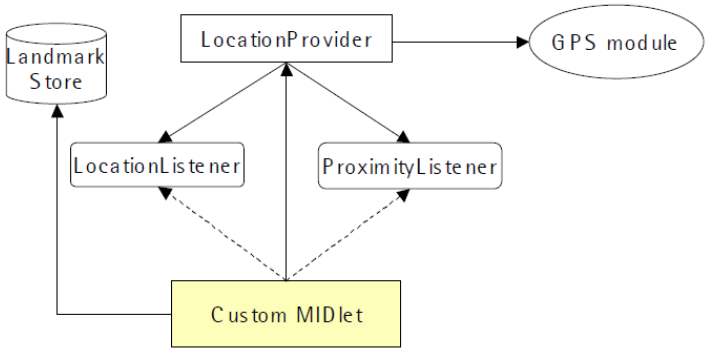


Figura 4-7 Midlet de tip Location API

Location API este definită în pachetul javax.microedition.location. Prezenţa API-ului poate fi descoperita cu ajutorul proprietății sistemului microedition.location.version. Valoarea acestei proprietăţi este versiunea specificaţiei API-ului implementate, de exemplu, "1.0".

Mobile Ajax pentru Java ME

Există multe Sun Microsystems tehnologii care folosesc Ajax [Ajax], şi mai mult de un mod de a folosi Ajax pe platforme mobile. De exemplu, aplicaţiile scrise folosind Java Platform, Enterprise Edition (Java EE, cunoscut anterior ca J2EE), pot genera XML, JSON [JSON], XHTML şi / sau ECMAScript destinate pentru browsere mobile.

Una dintre recentele progrese pe platforma Java, Mobile Edition (Java ME, cunoscut anterior ca J2ME) este Mobile Service Architecture [MSA]. MSA este un Java specification Request (JSR-248), care defineşte un set de API-uri pentru Java ME, care includ o varietate largă de caracteristici, de la Bluetooth la plăţii, API-uri multimedia si suport pentru grafică animată bogată.

Ajax este de obicei folosit în contextul de aplicatii web care rulează într-un browser şi foloseşte XMLHttpRequest de la ECMAScript pentru a prelua date XML sau JSON din servicii web RESTful (RESTful Web Services). Rezultatele sunt aplicate ca actualizări la pagina curentă DOM (Document Object Model [DOM]) a browserului.

Mobile Ajax pe platforma Java ME este utilizat pentru următoarele:

- Apel asincron la reţea (folosind Generic Connection Framework [GCF] al profilului Mobile Device Information Profile [MIDP]
- Utilizarea unui format de serie de date (cum ar fi XML sau JSON).
- Un layer de prezentare folosind DOM-based User Interface (cum ar fi XHTML sau SVG).

Figura de mai jos ilustrează o interacțiune tipică intre Mobile Ajax si platforma Java ME.

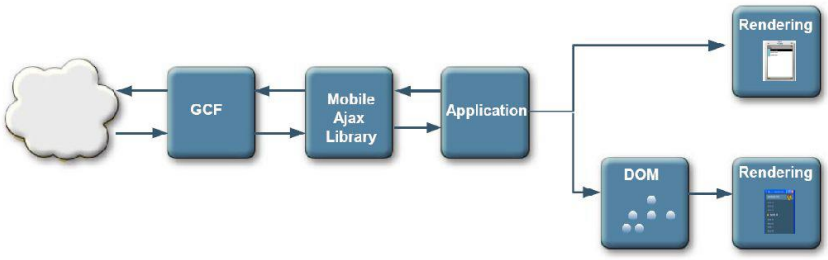


Figura 4-8 Interacțiune Mobile Ajax – J2ME [7]

Există multe motive pentru care utilizarea modelul Ajax este util în programele Java ME. Folosind o bibliotecă Ajax, aplicaţia este mai simplă şi are nevoie doar de implementarea apelurile sincrone sau rezolvarea apelurilor „callback", spre deosebire de lucrul cu complexitatea programelor multi-threaded. Biblioteca, de asemenea, conține parsere pentru datele cu format low-level interpretoare , care ajută la reducerea complexităţii aplicației, costurilor de întreținere şi depanare.

În scopul de a ajuta dezvoltatorii de Java ME a construi aplicaţii Web 2.0 mult mai uşor, a fost nevoie de o bibliotecă pentru a face faţă cu uşurinţă următoarelor cerințe:

- Manipularea asincronă de HTTP Get si Post.
- Rezolvarea apelurilor „callback" de progres.
- Autentificare HTTP Basic /Digest.
- Codare URL
- Codare Multi-part MIME

În concluzie pentru aplicaţii care au nevoia de a utiliza capacităţile unui dispozitiv mobil care nu sunt disponibile într-un browser pe telefonul mobil (cum ar fi camera foto, locul de amplasare, bluetooth, etc), o mică bibliotecă care oferă un model familiar de programare Ajax poate uşura sarcina unui dezvoltator Web.

Light Weight User Interface Toolkit 1.4

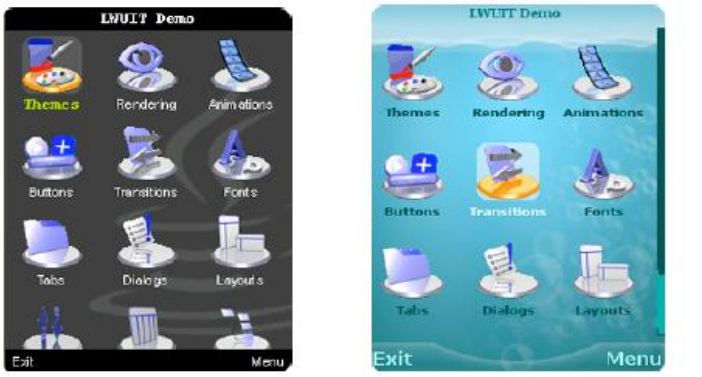
Având în vedere natura aplicațiile de telefoane mobile bazate pe tehnologia Java ™ Platform, Micro Edition (Java ME), crearea de aplicatii care pot rula cu uşurinţă pe toate dispozitivele poate fi o provocare. Prea adesea, presupunerile dezvoltatorul despre interfaţa utilizator şi aspect sunt diferite de la aparat la aparat. Pentru a ajuta la rezolvarea acestei probleme, Oracle a dezvoltat Lightweight User Interface Toolkit (LWUIT) pentru telefoanele care suportă Java ME.

LWUIT oferă capabilităţi UI si un API curat, care este inspirat din Swing. Pentru a adăuga „viaţa" în aplicaţiile lor, dezvoltatorii pot folosi tranziţii existente şi adăuga efecte vizuale la acestea. Dezvoltatorii pot construi, de asemenea, propriile lor componente UI sau pot folosi cele incluse pentru a oferi un aspect consistent şi convingătoar aplicațiilor lor, care pot fi rulate apoi într-o gamă largă de dispozitive și de furnizori. În plus, LWUIT suporta randare 3D prin intermediul JSR 184 şi suport pentru grafică vectorială scalabilă (Scalable Vector Graphics).

Aplicaţiile construite cu LWUIT pot prezenta o interfaţă de utilizator bogată şi omogenă pe mai multe dispozitive şi se poate adapta în mod automat şi profită de proprietăţile specifice ale dispozitivului, cum ar fi dimensiunea ecranului, capacităţi grafice, precum şi suport touch screen. Pentru a atinge aceste îmbunătăţiri de portabilitatea, LWUIT a fost construit folosind elemente „low-level" comune în MIDP 2.0 (sau similare cu capacităţi grafice de bază pe alte platforme). Managerul de aspect LWUIT, de asemenea, ajută în realizarea portabilității prin sprijinirea aplicațiilor în adaptarea cu uşurinţă la diferitele dimensiuni de ecran.

Componenta HTML introdusă în LWUIT 1.4 permite aplicaţiilor să randeze HTML cu uşurinţă în conformitate cu XHTML Mobile Profile 1.0. Cu această facilitate, dezvoltatorii pot include conținut web dinamic direct în aplicația lor, sau tehnologii de dezvoltare Web, inclusiv HTML si CSS pentru a putea proiecta cu mai multă uşurinţă interfeţe mobile client-side.

Prin aplicarea diferitelor seturi de resurse la o aplicaţie, tema sau aspectul de acesteia poate fi schimbat. Dezvoltatorii pot crea cu uşurinţă sau modifica teme prin editarea parametrilor cu ajutorul creatorului de teme. Mai jos sunt ilustrate aspectul unor astfel de teme.



LWUIT supports Theming and Pluggable Look and Feel (PLAF)

Figura 4-9 Teme LWUIT [8]

4.1.2 Șabloane arhitecturale

Model View Controller

MVC, sau Model-View-Controller este un şablon arhitectural folosit în industria de software development(inclusiv web development). Această modalitate de lucru reuşeşte cu succes izolarea părţii logice de interfaţa proiectului, rezultând în aplicaţii extrem de uşor de modificat. În organizarea MVC, modelul reprezintă informaţia (datele) de care are nevoie aplicaţia, viewerul corespunde cu elementele de interfaţă iar controller-ul reprezintă sistemul comunicativ şi decizional ce procesează datele informaţionale, făcând legătură între model şi view.

Mai jos avem o afișare grafică a componentelor unui Model View Controller și a interacțiunilor dintre acestea.

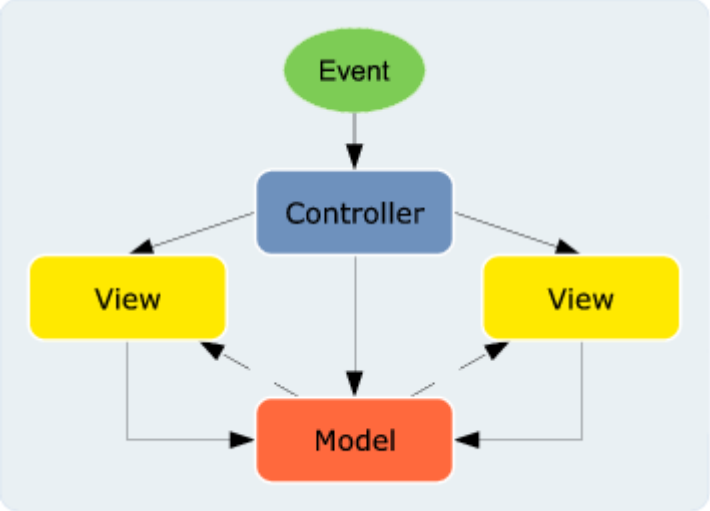


Figura 4-10 Șablonul MVC

Modelul reprezintă partea de hard-programming, partea logică a aplicaţiei. El are în responsabilitate acţiunile şi operaţiile asupra datelor, autentificarea utilizatorilor, integrarea diverselor clase ce permit procesarea informaţiilor din diverse baze de date.

View-ul se ocupă de afişarea datelor, practic această parte a programului va avea grijă de cum vede end-userul informaţia procesată de controller. O dată ce funcţiile sunt executate de model, viewului îi sunt oferite rezultatele, iar acesta le va trimite către browser. În general viewul este o mini-aplicaţie ce ajută la randarea unor informaţii, având la bază diverse template-uri.

Controller-ul reprezintă creierul aplicaţiei. Această face legătură între model şi view, între acţiunile userului şi partea decizionala a aplicaţiei. În funcţie de nevoile utilizatorului, controllerul apelează diverse funcţii definite special pentru secţiunea de program în care se află userul. Funcţia se va folosi de model pentru a prelucra (extrage, actualiza) datele, după care informaţiile noi vor fi trimise către view.

4.2 Analiză

4.2.1 Cazuri de utilizare

Diagramele cazurilor de utilizare descriu comportamentul sistemului, oferind o imagine de ansamblu asupra modului în care acesta este folosit din punctul de vedere al utilizatorilor. Cazurile de utilizare realizează o descriere, din punct de vedere funcţional, a sistemului, ansamblul tuturor cazurilor de utilizare şi utilizatorii acestora (actorii) formând modelul cazurilor de utilizare.

După cum s-a amintit și la specificația proiectului aplicația este împărțită în cele patru componente principale : MobileTicketingAndLocalization (MTL) Client System, MTL WebService, Ticket Server și GPS Server. Astfel ăn această secțiune include patru diagrame de cazuri de utilizare, câte una pentru fiecare componentă, care ajută la descoperirea cerințelor pe care acestea trebuie sa le îndeplinească din punctul de vedere al utilizatorului sistemului de localizare mijloace de transport si emitere de bilete.

GPS Server

Mai jos este ilustrată diagrama de cazuri de utilizare pentru componenta de furnizare de informații GPS din punctul de vedere al utilizatorului aplicatiei pe mobil.

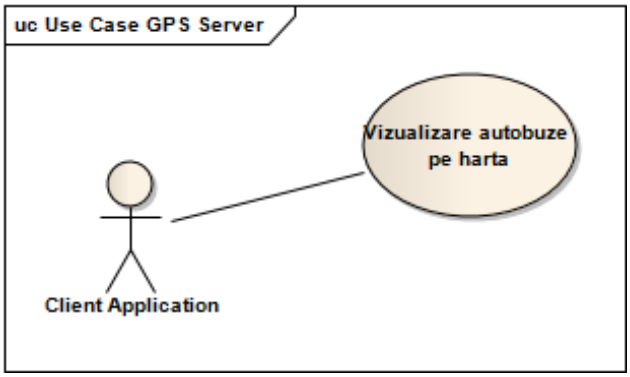


Figura 4-11 Server GPS - Use-Case

După cum se poate vedea în diagramă Serverul de GPS prezintă un singur caz de utilizare, din punctul de vedere al utilizatorului aplicației pe mobil, și anume „Vizualizare autobuze pe hartă”. Acest fapt se explică prin necesitatea clientului de a localiza mijloacele de transport în comun pe hartă. Componenta GPS Server ajută la realizarea acestui lucru prin trimiterea regulată a coordonatelor mijlocului de transport către baza de date cu ajutorul componentei de servicii Web MTL WebService, de unde vor fi preluate mai apoi de către componenta client și utilizate petru a afișa localizarea mijloacelor de transport.

Ticket Server

Mai jos este ilustrată diagrama de cazuri de utilizare pentru componenta de emitere de bilete din punctul de vedere al utilizatorului aplicatiei pe mobil.

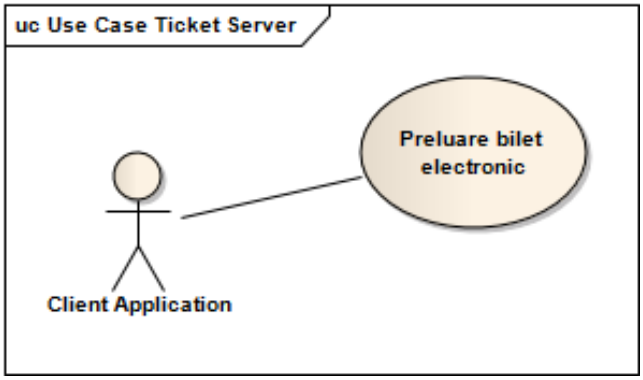


Figura 4-12 Server de bilete - Use-Case

La fel ca și la Serverul de GPS, cel de emitere de bilete (Ticket Server) prezintă deasemenea un caz de utilizare, din punctul de vedere al utilizatorului aplicației pe mobil, și anume „Preluare bilet electronic”. Acesta din urmă ajută clientul în procesul de cerere și obținere de bilet pentru călătoria cu mijlocul de transport. Componenta Ticket Server preia conexiuni simultane de la utilizatorii aplicaţtiei client pe mobil, comunică cu componenta de servicii Web MTL WebService și returnează clienților un bilet electronic.

MTL WebService

Mai jos este ilustrată diagrama de cazuri de utilizare pentru componenta de servicii Web (Mobile Ticketing and Localization WebService)din punctul de vedere al utilizatorului aplicatiei pe mobil.

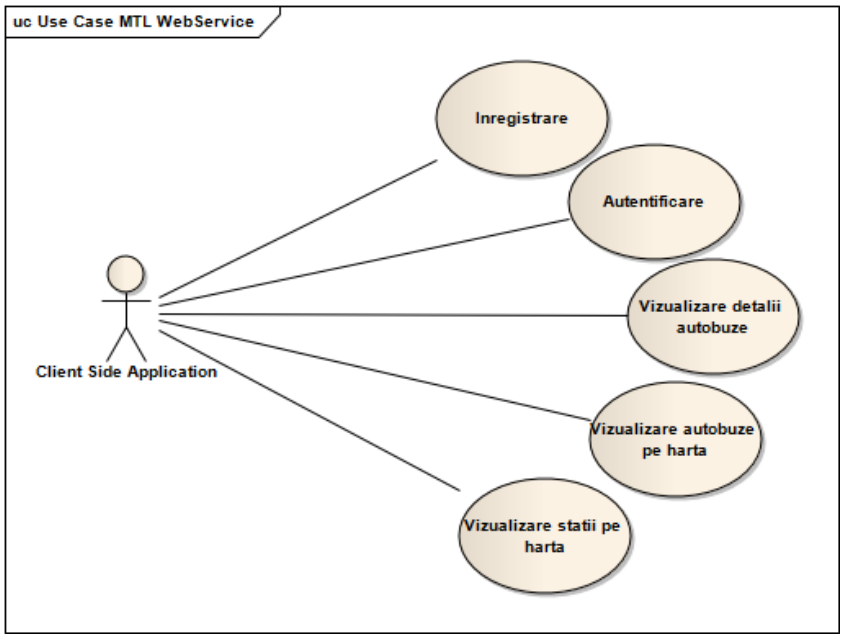


Figura 4-13 Componenta de servicii Web – Use-Case

Din diagrama de mai sus putem extrage principalele servicii Web pe care le oferă componenta MTL WebService. După cum se poate vedea există cinci cazuri de utilizare ale acestei componente deasemenea din puctul de vedere al clientului final al aplicației. Toate

cazurile de utilizare menționate au în spate o serie de servicii necesare realizării efectuării cu succes a acestora.

- Înregistrare : serviciu necesar înregistrarea în sistem, necesară la rândul ei realizarea autentificării ulterioare.
- Autentificare : serviciu necesar pentru autentificarea în sistem, necesară la rândul ei pentru preluarea de bilete și localizarea mijloacelor de transport.
- Vizualizare detalii autobuze : serviciu necesar pentru preluarea mai multor detalii despre mijloacele de transport.
- Vizualizare autobuze pe hartă : serviciu necesar pentru preluarea coordonatelor tuturor mijloacelor de transport
- Vizualizare stații pe hartă : serviciu necesar pentru preluarea stațiilor si a detaliilor despre acestea.

MTL Client System

Mai jos este ilustrata diagrama de cazuri de utilizare pentru aplicația pe mobil destinală clientului (Mobile Ticketing and Localization Client System) sistemului de emitere de bilete electronice si localizare a mijloacelor de transport în comun.

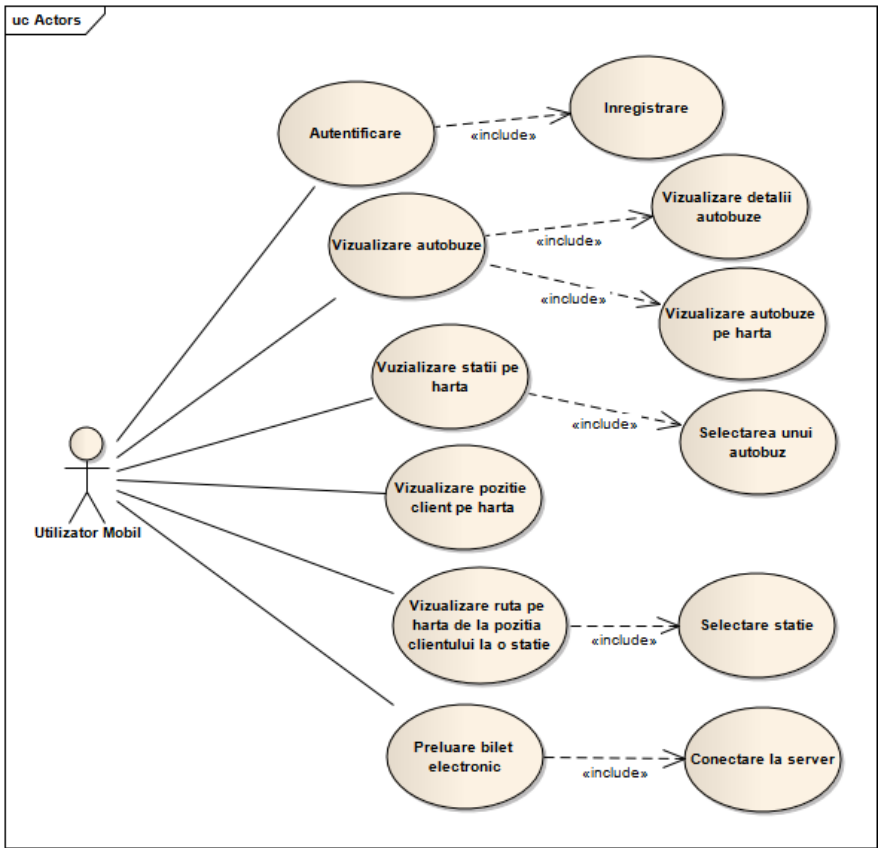


Figura 4-14 MobileTicketingAndLocalization – Use-Case

După cum se poate observa componenta MTL Client System realizează toate cerințele funcționale ale sistemului de emitere de bilete si localizare de mijloace de transport, acestea fiind reprezentate în diagrama de mai sus prin cazuri de utilizare.

4.2.2 Schema bloc a sistemului

Diagrama bloc este o diagrama a unui sistem, în care părţile principale sau funcţiile sunt reprezentate de blocuri conectate prin linii, acestea din urmă ilustrând relaţiile dintre blocuri.

Mai jos este reprezentată diagrama bloc a sistemului de localizare a mijloacelor de transport în comun si emitere de bilete MTL System (Mobile Ticketing and Localization System).

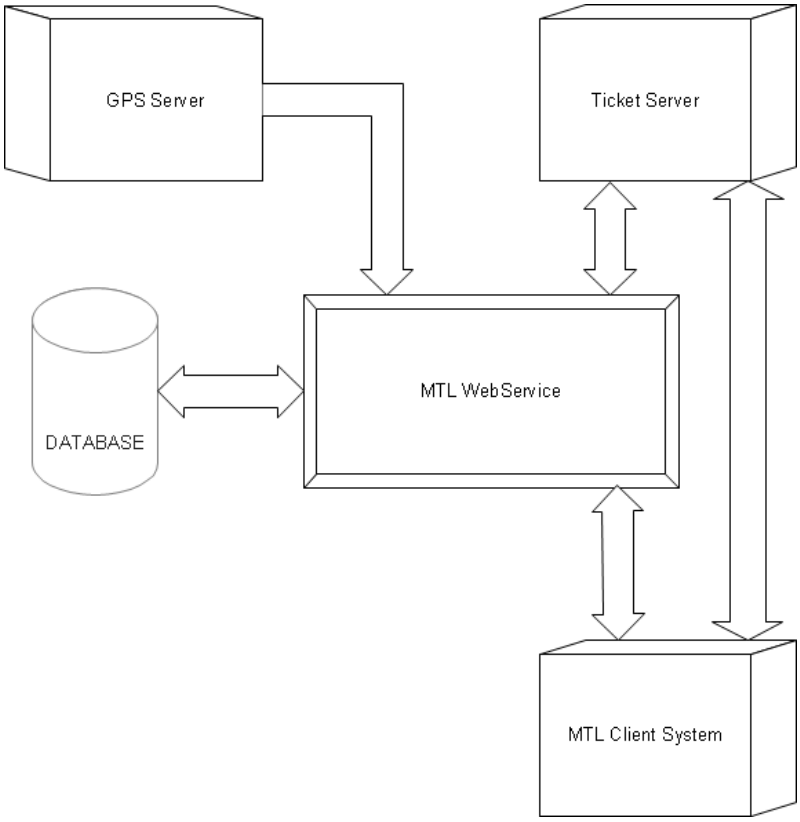


Figura 4-15 Schema bloc - sistem de emitere bilete și localizare

Principalele componenete ale sistemului sunt: aplicația client pentru dispozitive mobile MTL Client System, serverul care furnizează coordonatele mijlocului de transport în comun GPS Server, serverul care preia conexiuni de la clienți ți le returnează acestora bilete electronice ți nu în ultimul rând componenta de servicii Web MTL WebService care furnizează serviciile Web necesare pentru celelalte trei componente.

MTL Client System: după cum am precizat mai sus este chiar aplicația utilizată de client pe dispozitivul său mobil. Aceasta comunică cu serverul de bilete (Ticket Server) și cu MTL WebService în diferite scopuri. Conexiunea cu serverul de bilete este bidirecțională, acest fapt datorîndu-se nevoii clientului de a se conecta la server și totodată trimiterii unei chei specifice fiecărui utilizator, precum și nevoii clientului de a primi de la server biletul necesar. De asemenea aplicația client comunică bidirecțional și cu modulul de servicii Web, luând în considerarea atât trimiterea de date cât și recepționarea acestora, către și de la baza de date. Comunicarea cu serverul de GPS nu se realizează direct, acesta din urmă trimițând datele către baza de date prin servicii Web, iar aplicația client preluându-le tot prin același procedeu.

Serverul de GPS este tot o aplicație pe dispozitive mobile care ajută la localizarea mijloacelor de transport în comun. Acesta trimite datele despre locație (latitudine și longitudine)

către baza de date prin MTL WebService, de unde vor fi preluate ulterior de aplicația client. Conexiunea cu modulul de servicii Web este singura legatură directă a acestei componente, mai existând și una indirectă cu aplicația client descrisă mai sus.

Serverul de bilete este și el o aplicație server pentru dispozitive mobile care are rolul de a prelua conexiuni de la clienți, și de a emite bilete pentru aceștia. În acest context componenta Ticket Server comunică bidirecțional atât cu MTL Client System cât și cu MTL WebService. La conexinea unui cliet serverul de bilete preia id unic al acestuia il trimite către baza de date prin intermediul serviciului Web unde se realizează modificările necesare, iar mai apoi ii returnează clientului respectiv biletul interogat.

Componenta de servicii Web (MTL WebService) este o componentă esențială în sistem, aceasta comunicând cu toate celelalte componente. Aceste conexiuni au ca scop principal realizarea operațiunilor CRUD asupra bazei de date. Astfel putem spune că MTL WebService este un intermediar între aplicația client și baza de date, precum și intre servere (de bilete și GPS) și baza de date.

4.2.3 Arhitectura conceptuală

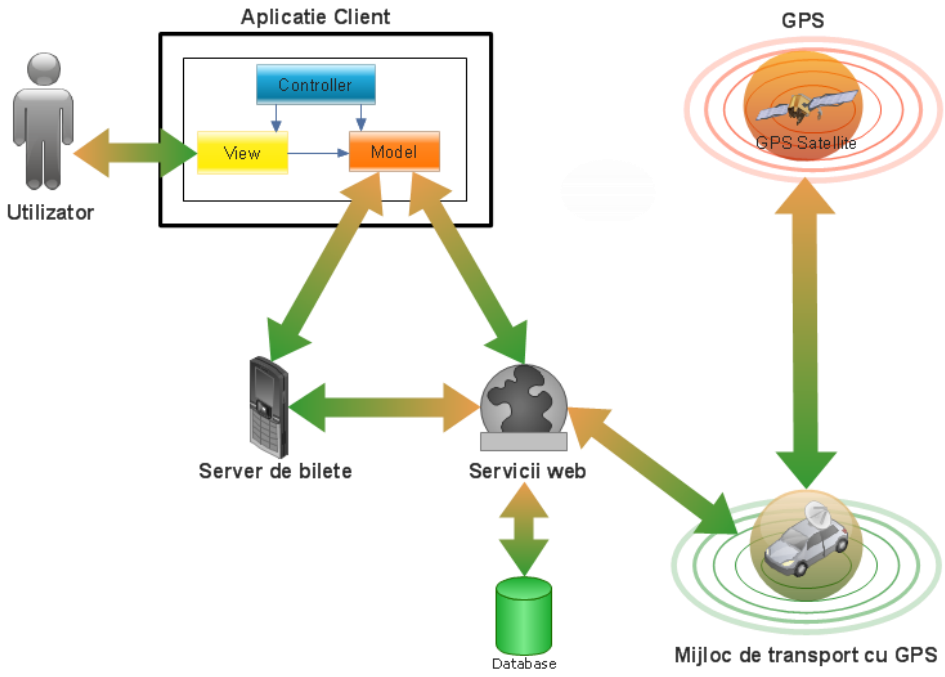


Figura 4-16 Arhitecura conceptuala a sistemului

Principalele elemente care intră în sistemul de emitere de bilete și localizare a mijloacelor de transport în comun sunt:

- Clientul
- Aplicație pe mobil destinată clientului
- Server de bilete
- Mijloc de transport în comun cu GPS încorporat
- Servicii Web
- Baza de date

Având în vedere aceste componente sistemul trebuie să îndeplinească cerințele funcționale descrise de acesta. În acest sens elementele amintite mai sus interacționează între ele furnizând o serie de servicii.

Aplicația clientului este împărțită și ea în trei module corespunzătoare șablonului arhitectural Model View Controller. Utilizatorul intră în contact direct cu componenta View a aplicației unde poate interoga sistemul pentru returnarea diferitelor servicii furnizate de către acesta. Controller-ul preia comenzile utilizatorului din View și le transmite spre procesare Model-ului. Acesta din urmă este componenta care interacționează cu servere-le și serviciile Web existente în sistem. Astfel Modelul se conectează la Serverul de bilete pentru cumpărarea uuui bilet, precum și la serviciile Web pentru a actualiza sau interoga baza de date.

Serverul de bilete preia conexiunile de la componentele Model ale clienților, actualizează baza de date și returnează un bilet clientului.

Furnizorul de coordonate de localizare preluate prin GPS este reprezentat de o aplicație existentă pe un dispozitiv mobil aflat în mijlocul de transport țn comun. Acesta actualizează datele referitoare la localizare pentru un automobil în baza de date prin intermediul serviciilor Web existente.

Capitolul 5. Proiectare de detaliu si implementare

5.1 Proiectare de detaliu

În acest capitol se vor prezenta în detaliu structurile celor patru componenete principale ale sistemului de localizare și emitere de bilete pentru mijloacele de transport în comun și anume: aplicația client pe dispozitive mobile **MobileTicketingAndLocalization**, serverul de bilete **TicketBluetoothServer**, serverul care furnizează detaiile despre localizare prin GPS **BusGPSProvider**, precum și componenta de servicii Web care face legătura între acestea **MobileTicketingAndLocalization_WebService**. Pe lângă toate acestea se mai adaugă și componenta reprezentând baza de date descrisă și ea în cele ce urmează.

5.1.1. Structura

Mai jos sunt descrise și ilustrate structurile celor patru component ale sistemului, precum și cea a bazei de date, cu ajutorul diagramelor de pachete.

MobileTicketingAndLocalization

Aplcația clientului este compusă din trei mari pachete și anume: **Model**, **View** și **Controller** reprezentând binențeles componentele șablonului architectural ModelViewController încorporat în aplicație. Figura de mai jos ilustrează cele trei pachete precum și legătura dintre acestea.

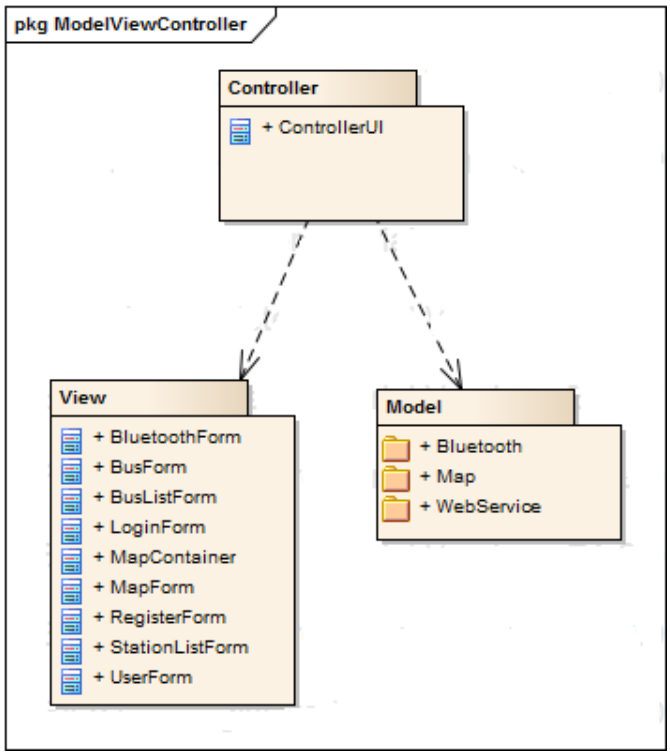


Figura 5-1 Diagrama de pachete - aplicație client

După cum am amintit și mai sus, cele trei pachete respectă regulile șablonului arhitectural ModelViewController.

- Pachetul **View** este reprezentat de componentele de interfață care interacționează direct cu utilizatorul aplicației, și sunt reprezentate prin Form-uri ce conțin elementa ca liste, butoane, comenzi și altele.
- Pachetul **Model** include clasele care realizează operații precum apelare de servicii Web, conectare la servicii Bluetooth și altele, precum și clasele entitate ale sistemului.
- Pachetul **Controller** are inclus în el doar clasa *ControllerUI* care preia cererile de la interfața cu utilizatorul si le transmite modelului. Acesta din urmă realizează operațiile necesare si transmite un raspuns controller-ului care modifică interfața utilizatorului în funcție de răspunsul primit.

Pachetul Model conține la rândul lui trei pachete și anume: **Bluetooth**, **WebService** și **Map**. Acest lucru se poate observa în figura de mai jos.

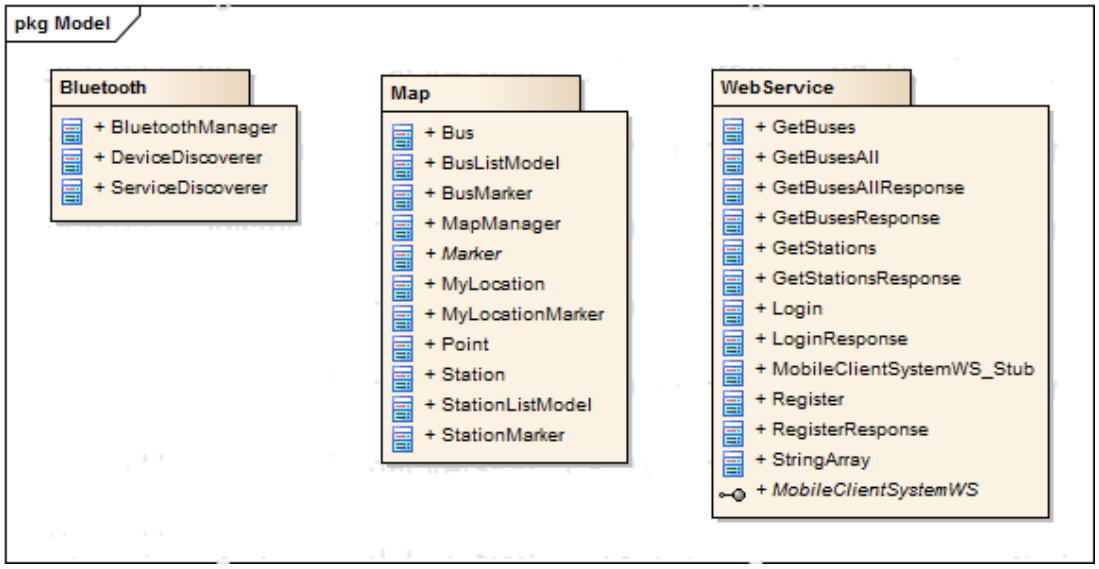


Figura 5-2 Diagrama de pachete - Model

- Pachetul **Bluetooth** încorporat în pachetul model prezintă trei clase necesare în realizarea conexiunii la serverul de Bluetooth pentru preluarea de bilete.
- Pachetul **Map** este alcătuit clase de tip entitate reprezentând autobuzele, stațiile, markerele, la acestea adăugându-se clasele ListModel pentru liste și nu în cele din urmă clasa *MapManager* care realizează principalele operații asupra hărții GoogleMaps integrată în aplicație.
- Pachetul **WebService** conține o serie de clase generate automat cu ajutotul unui tool (Wireless Toolkit) din fișierul de descriere WSDL al serviciului Web, cu ajutorul cărora aplicația poate apela serviciile Web dorite.

TicketBluetoothServer

Serverul de bilete este o aplicație server pe dispozitive mobile care preia simultan o serie de conexiuni prin Bluetooth de la clienți și ajută la realizarea operațiunii de emitere de bilete electronice. Aplicația este alcătuită din următoarele pachete principale: **Main**, **GPSServer** și **WebService**. Acestea din urmă sunt ilustrate în figura de mai jos.

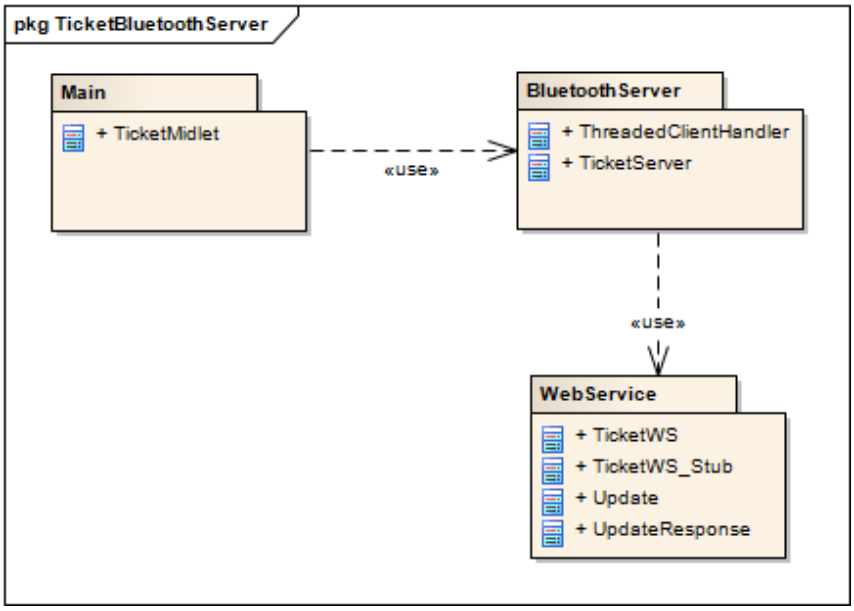


Figura 5-3 Diagrama de pachete – server de bilete

- Pachetul **Main** include clasa Midlet a aplicației necesară oricărui program destinat dispositive lor mobile și implementat in Java ME.
- **BluetoothServer** este pachetul care include clasele ce reprezintă serverul de Bluetooth creat (TicketServer), precum și clienții conectați la server (ThreadedClientHandler).
- Pentru realizarea operației de emitere de bilete mai este nevoie de clasele din pachetul **WebService** care, la fel ca și la aplicația client au fost generate cu tool-ul programului Wireless Toolkit, și folosesc la apelarea serviciilor Web necesare.

BusGPSProvider

Furnizorul de detalii despre localizarea mijloacelor de transport în comun este de asemenea o aplicație destinată dispozitivelor mobile. Acesta preia odată la câteva secunde prin GPS coordonatele de latitudine și longitudine ale mijlocului de transport și le transmite la o perioadă scurtă de timp către baza de date, de unde vor fi preluate ulterior de aplicația client și folosite pentru a localiza si vizualiza pe harta GoogleMaps mijlocul de transport în comun dorit. Figura următoare prezintă pachetele incluse în această aplicație.

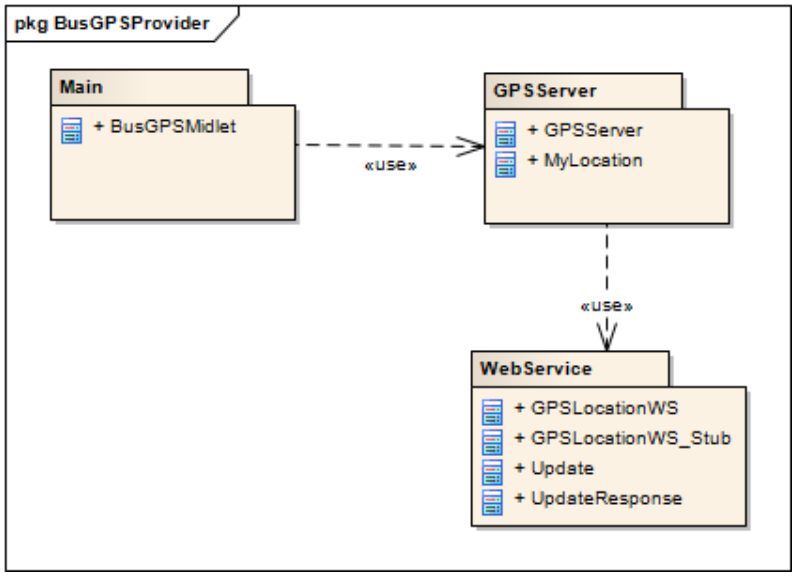


Figura 5-4 Diagrama de pachete – server GPS

Ca și la serverul de bilete, aplicația pentru furnizarea coordonatelor GPS are și ea un paachet Main în care este inclusă clasa Midlet denumită **BusGPSMidlet**. De asemenea există un pachet care înglobează clasele **GPSServer**, reprezentând serverul de GPS , acesta din urmă folosind clasa **MyLocation** pentru a lua odată la câteva secunde coordonatele respective. Pentru a putea trimite datele către baza de date, există inclus și pachetul WebService care include la fel ca și la primele două componente ale întregului sistem, clase auto-generate cu ajutorul tool-ului deținut de programul Wireless Toolkit, care ajută la apelarea serviciilor Web dorite pentru actualizarea bazei de date.

MobileTicketingAndLocalization_WebService

Componeta de servicii Web este o aplicație Web în care sunt încorporate servciile Web necesare și folosite de către restul componentelor sistemului pentru a putea interoga sau modifica baza de date. Serviciile incluse cuprind actualizarea datelor clientului în baza de date la procesarea unui bilet, actualizarea continuă a datelor reprezentând coordonatele mijloacelor de transport, înregistrarea unui client, autentificarea unui client, plus diferitele interogări asupra bazei de date. Principalele pachete și clase incluse în aplicație sunt ilustrate în diagrama de mai jos.

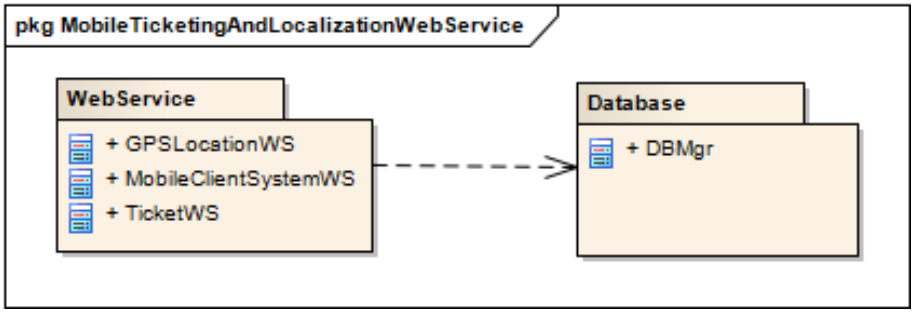


Figura 5-5 Diagrama de pachete – componenta de servicii Web

În pachetul WebService sunt descrise cele trei servicii Web implementate, care corespund, în funcție de operațiile efectuate celor trei componente: cea de client (MTL) și cele două servere (de bilete și de GPS) .

- Clasa **MobileClientSystemWS** reprezintă serviciul Web oferit aplicației client pentru realizarea diferitelor cerințe funcționale ale acesteia cum ar fi: înregistrarea în sistem, autentificarea în sistem, returnarea detaliilor despre stații din baza de date precum și returnarea detaliilor despre mijloacele de transport din baza de date.
- Clasa **TicketWS** este de fapt serviciul Web necesar și utilizat de către serverul de bilete pentru a actualiza tabelul clientului, în momentul procesării unui bilet electronic.
- Cea de-a treia clasă, și anume **GPSServerWS** reprezintă de asemenea un serviciu Web, necesar de această dată furnizorului de coordonate GPS pentru a putea trimite și actualiza tabelul mijlocului de transport în comun cu coordonatele de latitudine și longitudine actualizate.

DBMgr reprezintă clasa ce realizează conexiiunea la baza de date precum și operațiile asupra acesteia. Metodele de operații asupra bazei de date sunt apelate de către serviciile Web pentru a putea asigura cerințele funcționale descrise de acestea.

MobileTicketingAndLocalization_Database

Baza de date este formată din 5 tabele, structurate astfel încât să modeleze cât mai bine cerințele întregului sistem și să poată să răspundă cât mai bine la interogările principale la care va fi supusă.

Tabelele bazei de date sunt: users, buses, stations, bus_information și bus_station. Figura de mai jos prezintă cele 5 tabele și relațiile dintre acestea.

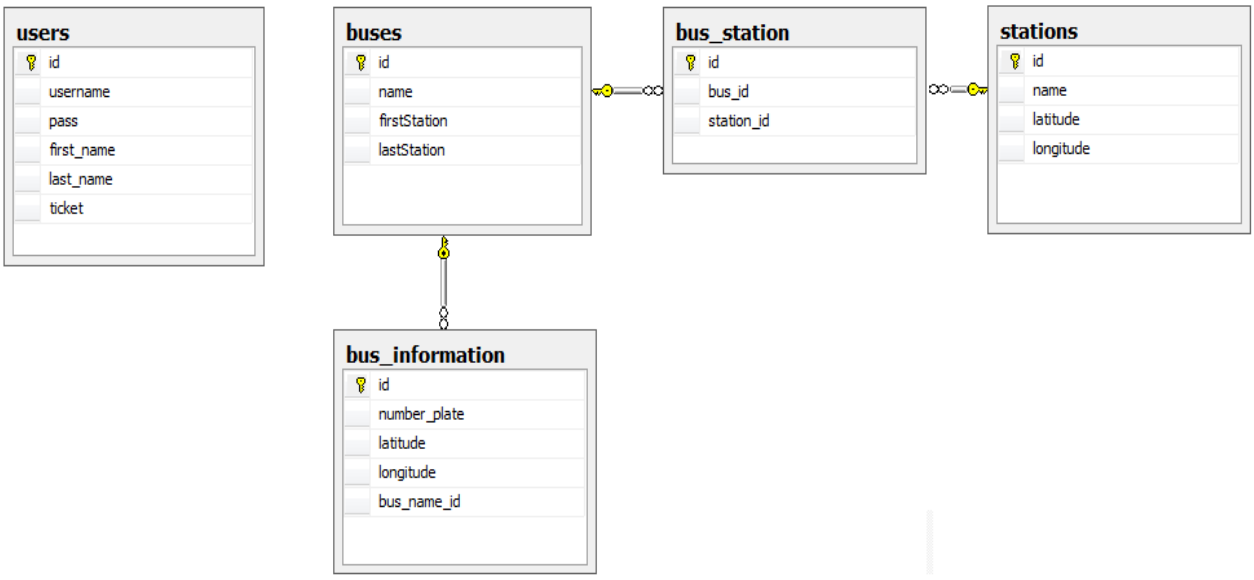


Figura 5-6 Structura Bazei de Date

Tabelele reprezintă entitățile din modelul entitate-relație (ER) pe baza caruia a fost modelată lumea reală a sistemului de emitere de bilete și localizare a mijloacelor de transport în comun, iar legăturile dintre tabele sunt constrângerile care apar intre entități.

După cum se poate observa în structura bazei de date există o serie de relații între tabele. Aceste relații sunt:

- buses – bus_information: acestă relație este una de tip „one-to-many”, adică unul la mai mulți. Acest lucru se datorează faptului că mai multe mijloace de transport în comun luate ca și o entități unice pot aparține unei clase de mijloace de transport (reprezentată de un număr și o rută speifică). Însă relația reciprocă nu mai este valabilă, un mijloc de transport în comun neputând avea mai multe rute sau numere.
- buses – stations: această relație este una de tip „many-to-many”, adică mai mulți la mai mulți, ceea ce inseamnă că o clasă de mijloace de transport în comun are mai multe stații, și se asemenea o stație poate aparține mai multor clase, mijloace de transport cu rute și numere diferite pot opri în aceeași stație

.

În continuare sunt prezentate pe rând toate tabelele bazei de date, insistându-se asupra atributelor și a tipurilor de date folosite, precum și asupra utilizabilitâții tabeluilui în sitemul de localizare a mijloacelor de transport în comun și emitere de bilete destinat utilizatorilor de dispozitive mobile.

Tabela „**users**” reprezintă ca și entitate un utilizator al sistemului care dorește prestarea serviciilor disponibile oferite de către sistem. Datele necesare pentru un utilizator sunt următoarele: un „username” și o parolă pentru autentificarea în sistem, numele și prenumele, precum și un câmp cu numărul biletelor disponible pentru a fi folosite. În acest sens tabelul „users” are următoarele câmpuri: id(PK, int, not null), username(varchar, null), pass(varchar, null), first_name(varchar, null), last_name(varchar, null), ticket(int, null). Tabela împreună cu câmpurile sale este ilustrată în figura de mai jos.

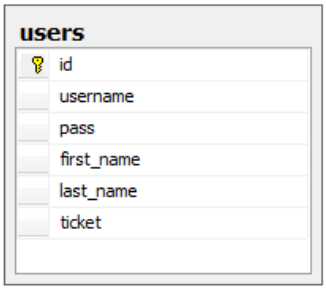


Figura 5-7 Tabel utilizatori

Tabela „**buses**” reprezintă tipul de rută a mijloacelor de transport în comun, mai exact numărul acestuia care definește ruta pe care se deplasează. Datele necesare pentru acest tabel sunt: stația de plecare, stația destinație și numărul mijlocului de transport. Astfel câmpurile tabelului „buses” sunt: id(PK, int, not null), name(varchar, null), fisrtStation(varchar, null), lastStation(varchar, null). Tabela împreună cu câmpurile acestuia este ilustrată în figura de mai jos.

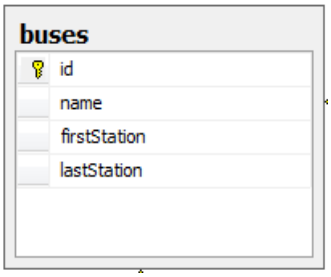


Figura 5-8 Tabel rută mijloc de transport

Tabela „**bus_information**” reprezintă ca și entitate un mijloc de transport în comun care poate fi localizat cu ajutorul sistemului implementat. Datele necesare pentru acest tabel sunt: numărul de înmatriculare, precum și coordonatele acestuia preluate prin GPS și anume latitudinea și longitudinea. În acest sens câmpurile tabelului „bus_information” sunt: id(PK, int, not null), number_plate(varchar, null), latitude(float, null), longitude(float, null), bus_name_id(FK, int, not null). Tabela împreună cu câmpurile sale este ilustrată în figura de mai jos.

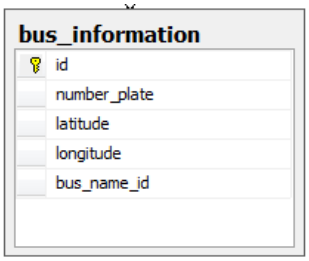


Figura 5-9 Tabel mijloc de transport

Tabela „**stations**” reprezintă o stație a unui mijloc de transport ţn comun. Datele necesare pentru acest tabel sunt:numele stației precum și coordonatele de latitudine și longitudine, necesare amplasîri acesteia pe hartă. Astfel câmpurile tabelului „stations” sunt: id(PK, int, not null), latitude(float, null), longitude(float, null). Tabela împreună cu câmpurile acestuia este ilustrată în figura de mai jos.

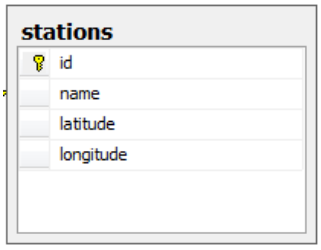


Figura 5-10 Tabel Stații

Tabela „**bus_station**” este o tabelă de legătura pentru a putea realiza legătura de „many-to-many”, adică mai mulți la mai mulți, existentă între tabela de stații și cea de clasă de mijloc de transport în comun.

Toate tabelele bazei de date au au chei primare surogate de tipul integer (tip de date numeric). O cheie surogat este o cheie primară unică generată de SGBD care nu este derivată din

nici o instanță a unei entități prezente în baza de date și a cărei singură semnificație este să aibă rolul de cheie primară. De asemenea pentru toate tabelele s-a folosit opțiunea auto-increment oferită de SQL Server 2008, deuarece în acest fel este mai ușoară numerotarea și gestiunea înregistrărilor, acestea organizându-se în ordinea în care au fost introduse. Majoritatea coloanelor de tip string, au fost declarate de tipul varchar, de diferite dimensiuni, evitându-se coloanele de tip date text. Acest lucru s-a făcut din considerente de eficiență și economisire a spațiului fizic.

5.1.2. Diagrame de clase

În această secțiune vor fi prezentate o serie de diagrame de clase pentru principalele și cele mai importante module ale sistemului de localizare și emitere de bilete pentru mijloacele de transport în comun.

MobileTicketingAndLocalization

În aplicația clientului proiectată pentru dispozitive mobile, există o serie de cerințe funcționale ce trebuie implementate precum înregistrarea sau autentificarea în sistem, preluarea unui bilet, vizualizarea stațiilor pe hartă sau localizarea si vizualizarea mijloacelor de transport în comun pe harta GoogleMaps. În acest sens sunt prezentate mai jos, cu ajutorul diagramelor, clasele ce ajută la realizarea acestor cerințe precum și relațiile dintre acestea.

Înainte de a realiza orice fel de operație, utilizatorul aplicației trebuie să se autentifice în sistem. Dacă nu este încă înregistrat, acest lucru este de asemenea posibil de realizat și necesar în vederea efectuării operațiilor dorite. Clasele care intervin în acest proces sunt: **MobileClientSystemWS_Stub** din pachetul WebService inclus în pachetul Model, precum și clasele din view (interfață utilizator): **LoginForm** și **RegisterForm**. Figura următoare prezintă clasele amintite mai sus, precum și legăturile dintre acestea.

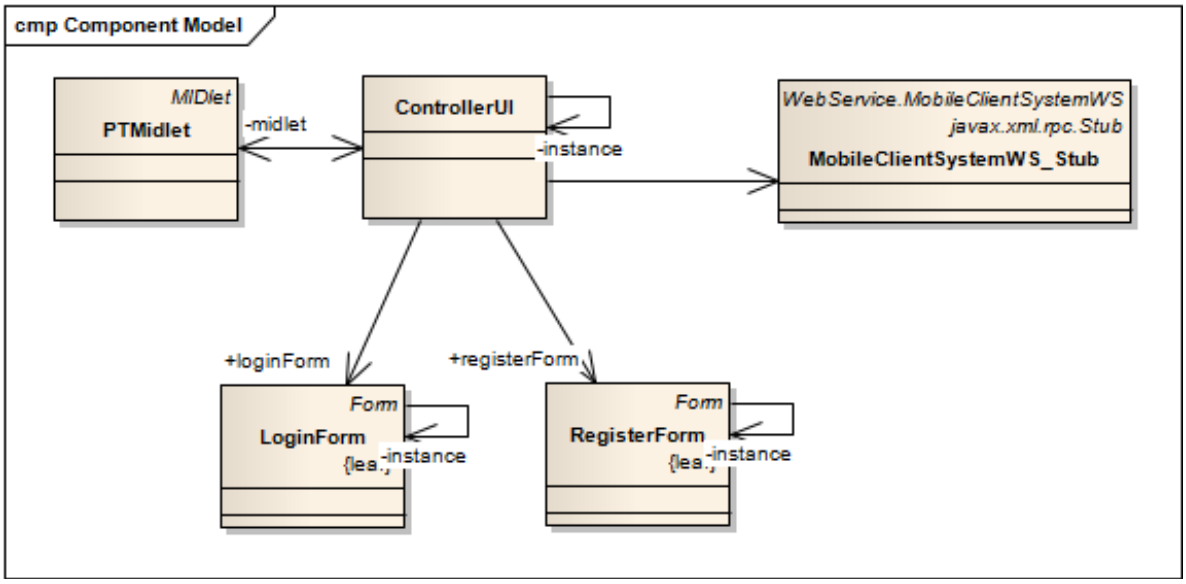


Figura 5-11 Diagrama de clase - autentificare

Dacă utilizatorul dorește să se înregistreze în sistem atunci el va apasa comanda register și va fi redirecționat către form-ul de înregistrare reprezentat de clasa **RegisterForm**. La fel se procedează și dacă clientul dorește autentificarea în sistem, în acest caz el fiind redirecționat către formul de autentificare reprezentat la rândul lui de clasa **LoginForm**. După completarea cîmurilor necesare operației alese se va putea executa comanda de înregistrare sau autentificare. Acest lucru va fi trimis către controller care va apela la rândul său serviciul Web necesar. Apelarea serviciului Web se face prin intermediul clasei **MobileClientSystemWS_Stub** care realizează conexiunea la serviciul Web și prezintă metodele necesare efectuării operațiilor dorite.

Pentru operațiunea de conectare la serverul de bilete și preluarea unui bilet există o serie de clase incluse în pachetul Bluetooth din interiorul pachetului Model. Aceste clase sunt: **BluetoothManager**, **DeviceDiscoverer** și **ServiceDiscoverer**. La nivelul interfeței cu utilizatorul clasa care descrie formul cu care acesta va intra in contact direct este **BluetoothForm**. Mai jos este ilustrată diagrama cu clasele care intră în realizarea acestei operații.

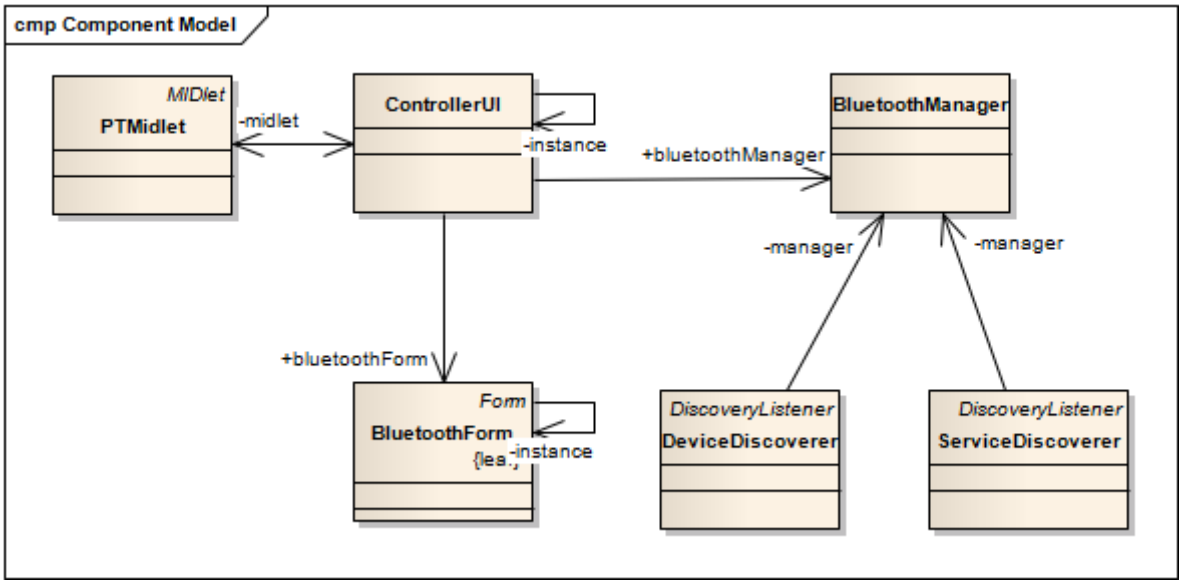


Figura 5-12 Diagrama de clase – conectare la several de bilete

După cum se poate poate observa în diagrama de mai sus clasa **ControllerUI** face legătură între model şi view, între acţiunile userului şi partea decizionala a aplicaţiei. În acest caz modelul este reprezentat de clasa **BluetoothManager** iar partea de view (interfață utilizator) este chiar clasa de tip form **BluetoothForm**. În cazul operației în cauză utilizatorul trimite o comandă de conectare la dispozitivul Bluetooth iar controllerul preia comanda și o trimite către managerul de Bluetooth. Acesta din urmă cu ajutorul metodelor de căutare de dispozitive si servicii Bluetooth incluse în clasele **DeviceDiscoverer** și **ServiceDiscoverer**, se va conecta la serverul Bluetooth pentru bilete urmând ca acesta sa emită și să returneze un bilet aplicației client.

O altă operație importantă a aplicației destinate clientului este cea de localizare și vizualizare pe harta GoogleMaps a mijloacelor de transport în comun. În acest sens există cateva clase descrise în pachetele Map și WebService din interiorul pachetului Model. Aceste clase sunt: **Bus**, **BusListModel**, **BusMarker**, **Marker**, **Point** și **MapManager** din Map, precum și

MobileClientSystemWS_Stub din WebService. Aceasta din urmă, reprezentând conectarea și apelarea la serviciului Web fiind auto-generată cu ajutorul unui „tool” din programul WirelessToolkit. În partea de view (interfață utilizator) clasele implicate sunt **BusListForm**, **MapForm** și **MapContainer**. Mai jos este ilustrată diagrama cu clasele care intră în realizarea operațiilor precizate anterior.

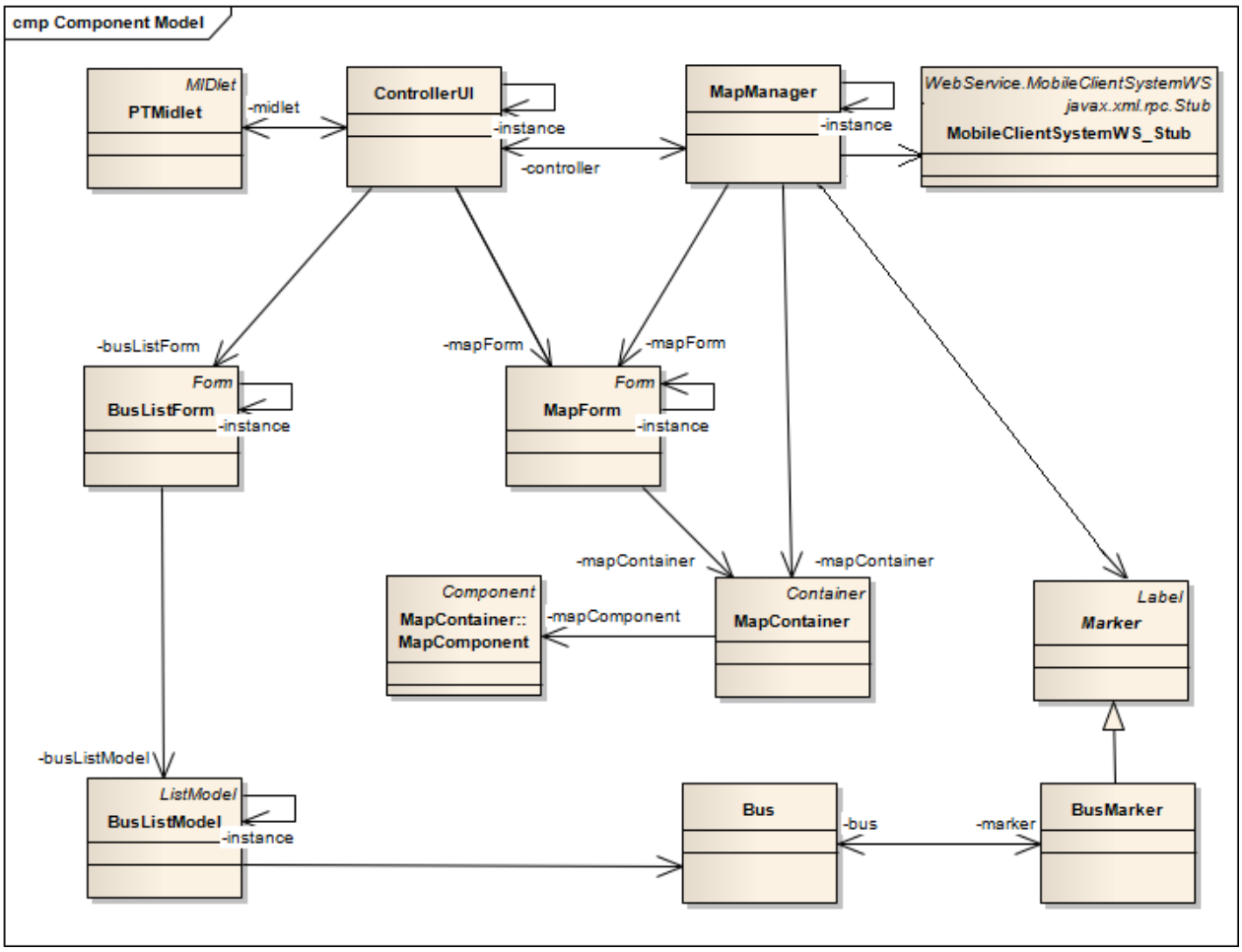


Figura 5-13 Diagrama de clase – operații pe hartă

Pentru a putea vizualiza mijloacele de transport în comun dorite pe harta GoogleMaps, utilizatorul trebuie mai întâi să selecteze o rută de autobuze pentru care dorește să realizeze localizarea. De aceea există clasa de tip form **BusListForm** în care există o listă de numere de mijloace de transport din care utilizatorul alege una. Această listă implementează modelul de listă **BusListModel** în care există încărcate obiectele necesare. După selectarea rutei dorite, utilizatorul va fi redirecționat către formul cu harta GoogleMaps reprezentat de clasa **MapForm**. Aici utilizatorul va putea să localizeze și să vizualizeze mijloacele de transport reprezentate pe hartă prin markere. Acest lucru este pozibil cu ajutorul clasei **MapManager** care prezintă metode de apelare a serviciului Web necesar și preluarea informațiilor necesare despre mijloacele de transport, precum și metode de adăugare și afișare a marker-elor. Conectarea la serviciul Web și apelarea metodelor acestuia este posibilă cu ajutorul clasei **MobileClientSystem_Stub**.

TicketBluetoothServer

Aplicația server pentru emiterea de bilete are ca cerințe funcționale principale preluarea simultană a conexiunilor Bluetooth de la mai mulți clienți, apelarea serviciului Web necesar pentru actualizarea tabelelor clienților, precum și returnarea biletului către aceștia. Astfel serverul prezintă o serie de clase care realizează aceste operații. Acestea sunt: **TicketServer**, **ThreadedClientHandler** și **TicketWS_Stub**. Clasele serverului Bluetooth de bilete precum și relațiile dintre acestea sunt prezentate în diagrama de mai jos.

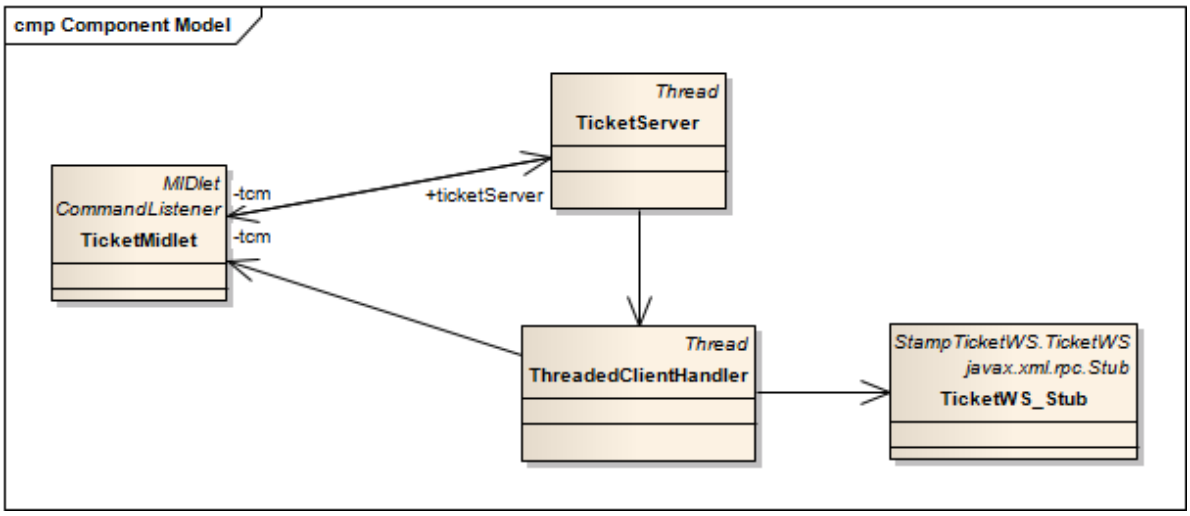


Figura 5-14 Diagrama de clase – server de bilete

Pentru a putea gestiona simultan conexiunile clienților, serverul de bilete prezintă o clasă TicketServer care este de fapt un thread care așteaptă conexiunile de la clienți atât timp cât serverul este pornit. La conectarea unui client se crează un nou thread separat, reprezentat prin clasa ThreadedClientHandler, care gestionează operațiile asupra clientului respectiv. Aceste operații constau în emiterea unui bilet și actualizarea tabelului corespunzător clientului în baza de date, acest lucru realizându-se prin apelarea servciului Web de actualizare a biletelor. Conectarea și apelarea serviciului Web se realizează cu ajutorul clasei TicketWS_Stub.

BusGPSProvider

Aplicația server pentru furnizare de coordonate pentru localizarea mijloacelor de transport în comun are ca cerințe funcționale principale preluarea coordonatelor de latitudine și longitudine a automobilului pe care este instalată și trimiterea acestora către baza de date. Transmiterea datelor către baza de date se realizează prin apelarea serviciului Web implementat în componenta de servicii, care actualizează câmpurile de latitudine și longitudine din tabela mijlocului de transport. Clasele necesare realizării acestor operațiuni sunt: **GPSServer**, **MyLocation** și **GPSLocationWS_Stub**. Acestea din urmă precum și relațiile dintre ele sunt prezentate în diagrama de clase de mai jos.

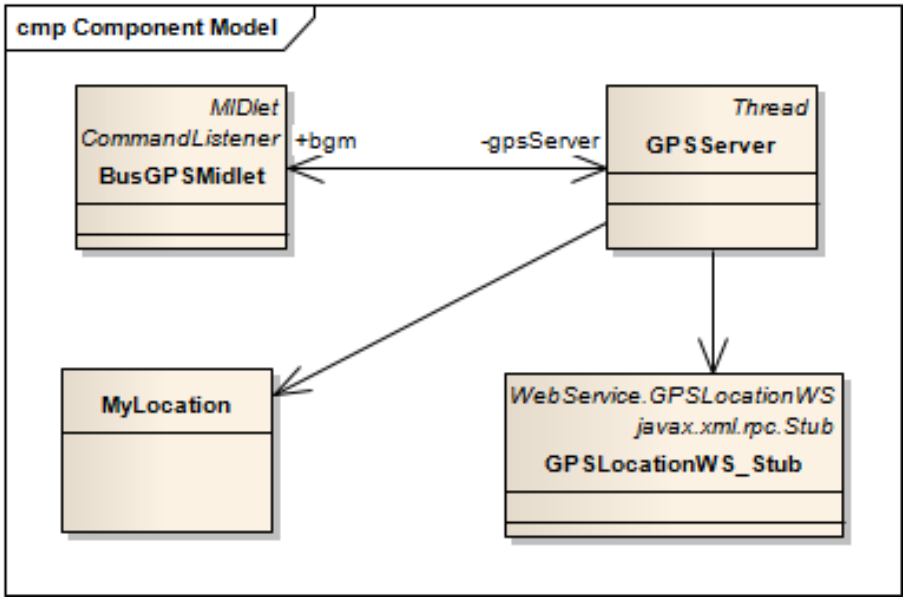


Figura 5-15 Diagrama de clase – server GPS

Pentru a putea prelua coordonatele prin GPS odată la cateva secunde aplicația prezintă clasa GPSServer, aceasta fiind de fapt un thread care după ce este pornit preia latitudinea și longitudinea prin GPS după care așteaptă câteva secunde și preia din nou coordonatele. Acest proces continuă până când este oprit serverul furnizor de coordonate. Preluarea coordonatelor GPS se realizează cu ajutorul clasei MyLocation. Trimiterea datelor actualizate în baza de date se face cu ajutorul clasei GPSLocationWS_Stub care realizează conexiunea la serviciul Web și furnizează metode de apelare al acestuia.

5.2 Implementare

5.2.1. Clase și componente

În continuare va fi prezentată organizarea fiecărei componente a sistemului, împreună cu cele mai importante aspecte legate de implementarea claselor şi comunicarea dintre acestea.

MobileTicketingAndLocalization

Această parte a sistemului, ținând cont că este utilizată și intră în contact direct cu clientul trebuie să rezolve principalele cerințe pe care acesta le transmite prin intermediul unei interfețe grafice. Astfel aplicația principală trebuie să implementeze și o interfașă grafică prietenoasă, ușor de utilizat și care să furnizeze principalele comenzi și elemente grafice pentru a putea efectua și vizualiza operațiile dorite.

În continuare sunt prezentate principalele clase ale aplicației precum și rolul lor în implementarea cerințelor funcționale ale sistemului. Clasele sunt împărțite în pachete în funcție de tipul operației pe care o implementează. Pachetele incluse în aplicație sunt: ***Main***, ***Model*** *(*cu subpachetele *Bluetooth*,*Map* și *WebService)*, ***View*** și ***Controller***.

Pachetul ***Main*** conține o singură clasă, mai precis clasa de tip Midlet a aplicației: *PTMidlet*. Figura următoare ilustrează acestă clasă precum și metodele și atributele corespunzătoare.

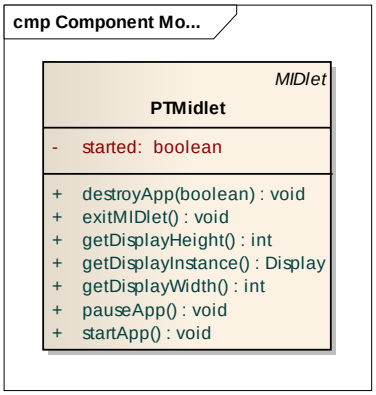


Figura 5-16 Clasa Midlet a aplicației client

Midlet-ul aplicației conține, ca și oricare midelt de altfel, clasele *startApp(), pauseApp()* și *distroyApp()* clase provenite din extinderea super-clasei MIDlet. Aceste metode descriu ciclul de viață al programului, adică în care dintre cele trei stări posibile (pornit, întrerupt sau terminat) se află acesta la un moment dat. Pe lângă aceste trei clase obligatorii, *PTMidlet* mai conține si clase de tip get pentru dimensiunile ecranului: *getDisplayHeight()* și *getDisplayWidth().*

Pachetul ***Controller*** conține de asemenea o singură clasă: *ControllerUI* care reprezintă chiar clasa de control rezultată din implementarea șablonului arhitectural ModelViewController. Metodele și atributele precum și clasele sale interioare pentru „prinderea" eventurilor din formurile interfeței grafice sunt ilustrate în imaginea de mai jos.

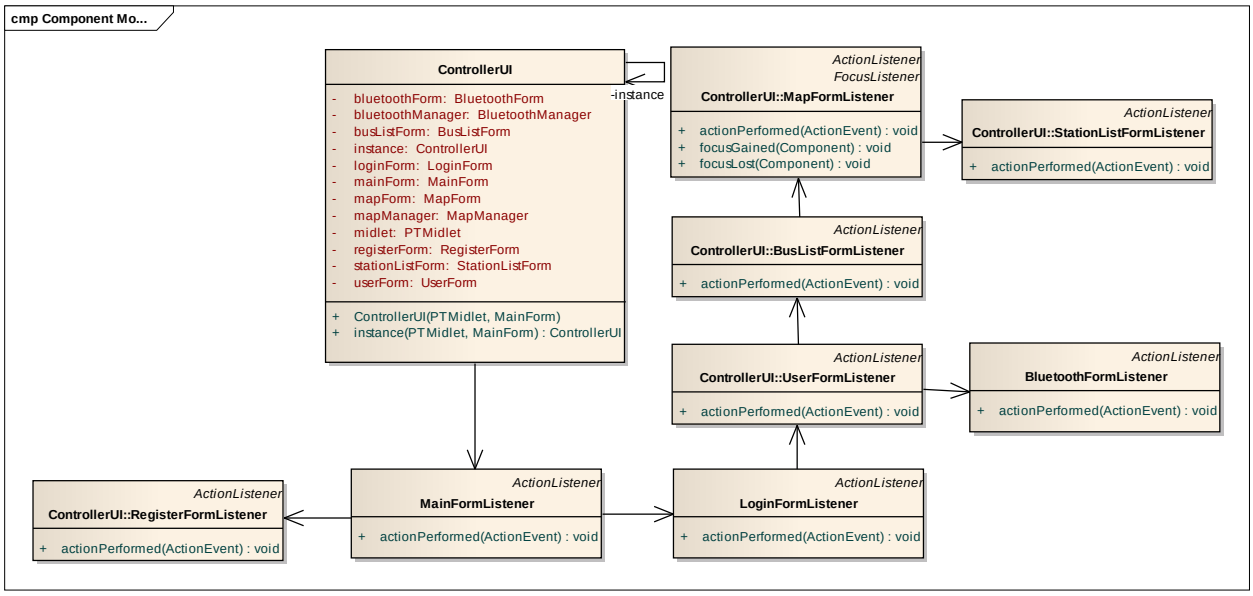


Figura 5-17 Clasa Controller

Clasa de control, după cum îi spune şi numele, trebuie să controleze interacţiunea dintre clasele din interfaţa grafică şi cele din model. Astfel *ControllerUI* descrie o serie de clase interioare care implementează interfeţe de tip *Listener*, fiecare clasă corespunzând unui form din interfaţa grafică. Cu ajutorul acestor clase „de ascultare" clasa de control va şti când şi unde s-a produs o modificare sau s-a apelat o comandă în interfaţa grafică şi va apela modelul pentru a realiza operaţiile necesare. În acest sens clasa *ControllerUI* are ca şi atribute toate clasele de interfaţă grafică (*MainForm, LoginForm, RegisterForm,UserForm, BluetoothForm, BusListForm, MapForm* şi *StationListForm*), precum şi principalele clase din model: *BluetoothManager* şi *MapManager*.

Pachetul ***Model*** realizaeză operaţiile trimise de către clasa de control. În funcţie de tipul operaţiei necesare modelul prezintă trei mari pachete care reprezintă cele trei tipuri de servicii oferite de aplicaţia client, şi anume:

- Conectarea prin Bluetooth la serverul de bilete: realizată cu ajutorul claselor din pachetul ***Bluetooth*** *(BluetoothManager,DeviceDiscoverer* şi *ServiceDiscoverer).*
- Operaţii asupra hărţii GoogleMaps: realizate de clasele din pachetul ***Map*** *(MapManger, MyLocation, Point, Station, Bus, BusListModel, StationListModel, Marker, BusMarker, StationMarker).*
- Apelarea serviciilor Web: realizată cu ajutorul claselor din pachetul ***WebService*** *(MobileClientSystemWS_Stub, GetBuses, GetBusesResponse, , GetBusesAll, GetBusesAllResponse, GetStations, GetStationsResponse, Login, LoginResponse, Register, RegisterResponse).*

Mai jos sunt prezentate şi ilustrate pe rând principalele clase din model, precum şi atributele şi metodele acestora. Vom începe prin prezentarea celor trei clase pentru conexiunea la serverul de Bluetooh. Acestea sunt ilustrate ilustrate în figura următoare.

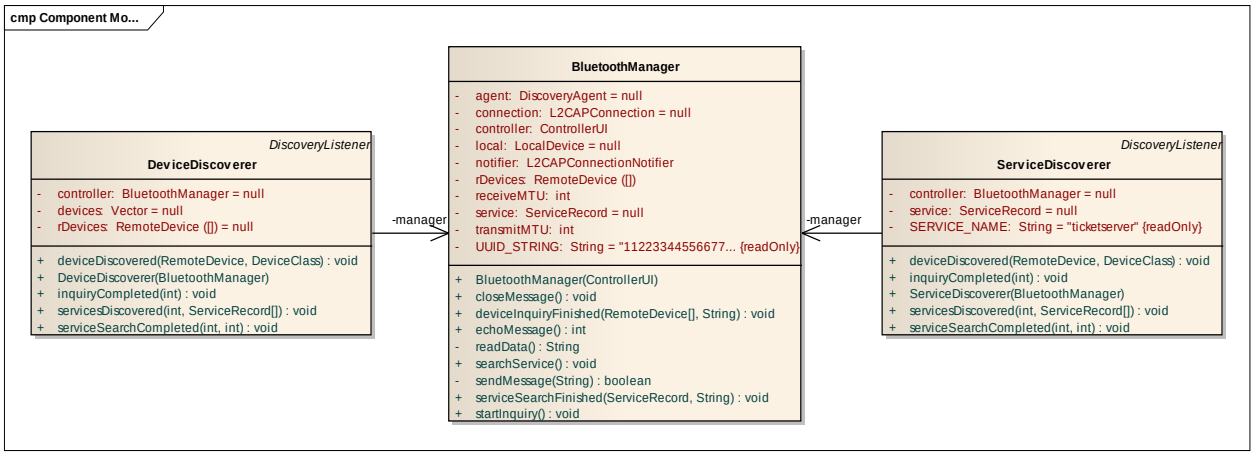


Figura 5-18 Clasele pentru Bluetooth

Pentru a folosi metodele legate de conectarea prin Bluetooth trebuie mai intâi obţinerea unei referinţe a obiectului *LocalDevice.* Acest lucru se face prin chemarea metodei *LocalDevice.getLocalDevice().* Cu ajutorul obiectului obtinut putem accesa proprietaţile Bluetooth ale dispozitivului găsit. Folosind acelasi obiect se obţine si agentul *DiscoveryAgent* folosit in procesele de device discovery si service discovery.

Astfel au fost declarate in clasa **BluetoothManager** obiectele de mai jos astfel:

```
private LocalDevice local = null;
private DiscoveryAgent agent = null;
```

Codul pentru obținerea acestor obiecte este următorul:

```
local = LocalDevice.getLocalDevice();
agent = local.getDiscoveryAgent();
```

Obiectul „agent" va fi folosit pentru a porni procesele de căutare de dispositive și căutare de servicii. Apelarea acestor metode se face astfel:

```
agent.startInquiry(DiscoveryAgent.GIAC, discoverer);
agent.searchServices(attrSet, uuidSet, rDevices[index], serviceDListener);
```

Procesul de Device Discovery este primul pas necesar pentru a găsi dispositive Bluetooth. În clasa DeviceDiscoverer sunt implementate metodele necesare acestui process. Codul acestor metode arată astfel:

```
public void deviceDiscovered(RemoteDevice remote,DeviceClass dClass)
{
        devices.addElement(remote);
}
public void inquiryCompleted(int descType)
{
         String message = "";

         switch(descType)
        {
            case DiscoveryListener.INQUIRY_COMPLETED:
                        message = "INQUIRY_COMPLETED";
                    break;
            case DiscoveryListener.INQUIRY_TERMINATED:
                        message = "INQUIRY_TERMINATED";
                    break;
            case DiscoveryListener.INQUIRY_ERROR:
                        message = "INQUIRY_ERROR";
                    break;
        }

         rDevices = new RemoteDevice[devices.size()];

          for(int i=0;i<devices.size();i++)
              rDevices[i] = (RemoteDevice)devices.elementAt(i);

        bluetoothManager.deviceInquiryFinished(rDevices,message);
        devices.removeAllElements();
}
```

Metoda *deviceDiscovered* se va apela cand un nou dispozitiv va fi găsit, iar acesta va fi adaugat in vectorul "devices". Când cautarea se termină, se verifică daca totul a decurs cum

trebuie si se afișează un mesaj utilizatorului cu ajutorul statement-ului „switch”. Apoi se adaugă dispozitivele in array-ul de RemoteDevice si se apelează metoda *deviceInquiryFinished* din BluetoothManager. Această metoda preia numele tuturor dispozitivelor găsite cu ajutorul metodei *getFriendlyName()* si le afișează pe ecran.

După terminarea procesului de device discovery, se poate afla ce servicii oferă dispozitivele găsite cu ajutorul procesului de service discovery aplicat pe dispozitivul dorit. În clasa ServiceDiscoverer sunt implementate metodele necesare acestui process. Codul acestor metode arată astfel:

```
public void servicesDiscovered(int transId,ServiceRecord[] services)
{
        for(int j=0;j<services.length;j++)
        {
                DataElement dataElementName = services[j].getAttributeValue(0x0100);
                String serviceName = (String)dataElementName.getValue();

                 if(serviceName.equals(SERVICE_NAME))
                                service = services[j];
                 break;
        }S
}
public void serviceSearchCompleted(int transId,int respCode)
{
    String message = "";

         switch(respCode)
         {
            case DiscoveryListener.SERVICE_SEARCH_COMPLETED:
                                message = "SERVICE_SEARCH_COMPLETED";
                        break;
            case DiscoveryListener.SERVICE_SEARCH_ERROR:
                                message = "SERVICE_SEARCH_ERROR";
                        break;
            case DiscoveryListener.SERVICE_SEARCH_TERMINATED:
                                message = "SERVICE_SEARCH_TERMINATED";
                        break;
            case DiscoveryListener.SERVICE_SEARCH_NO_RECORDS:
                                message = "SERVICE_SEARCH_NO_RECORDS";
                        break;
            case DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
                                message = "SERVICE_SEARCH_DEVICE_NOT_REACHABLE";
                        break;
         }
          controller.serviceSearchFinished(service,message);
}
```

Metoda *serviceDiscovered* caută la dispozitivul ales serviciul cu numele “ticketServer”. La fel ca si la device discovery când căutarea se termină, se verifică dacă totul a decurs cum

trebuie si se afișează un mesaj utilizatorului cu ajutorul statement-ului “switch”. Apoi se apelează metoda *serviceSearchFinished()* din BluetoothManager. Acestă metodă preia URL-ul serviciului gasit si realizează conexiunea cu dispozitivul server. Codul pentru aceste operații este prezentat mai jos:

```
url = service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT,false);
connection = (L2CAPConnection)Connector.open(url);
```

Un alt serviciu oferit de aplicația client și implementat în cadrul modelului este cel de localizare și vizualizare pe hartă a mijloacelor de transport în comun. Clasa care realizează management-ul operațiilor asupra hărții GoogleMaps este **MapManager** și este inclusă în cel de-al doilea pachet amintit la început, și anume **Map**. Printre operațiile importante realizate de clasa manager pentru hartă se află: apelarea serviciului Web oferit de GoogleMaps pentru a prelua o hartă statică, introducerea hărții obținute într-un container, precum și adăugarea diferitelor tipuri de markere pe hartă. Figura următoare prezintă atributele și metodele acestei clase importante:

Figura 5-19 Clasa MapManager

După cum am amintit și mai sus clasa MapManager realizează apelarea serviciului oferit de GoogleMaps pentru obținerea unui harți statice. URL-ul pentru acest serviciu este http://maps.google.com/maps/api/ plus o serie de parametri specifici, cum ar fi „staticmap” sau „directions”, acesta din urmă împreună cu un număr de parametri necesari returnează o hartă împreună cu direcțiile interogate.
Astfel au fost declarate în MapMnager următoarele string-uri:

```
private static String GOOGLE_MAP_BASE = "http://maps.google.com/maps/api/staticmap";
private static String GOOGLE_DIR_BASE = "http://maps.google.com/maps/api/directions/json";
```

Pentru a apela serviciul Web pentru returnarea unei hărți statice trebuie creat mai întâi un URL case să includă parametri necesari: coordonatele centrului, zoom, dimensiune și senzor și ruta dacă există.

Codul pentru apelarea serviciului și returnarea imaginii arată astfel:

```
String location = GOOGLE_MAP_BASE + "?" + "center=" +
                          Double.toString(loc.getLatitude()) + "," +
                          Double.toString(loc.getLongitude()) +
                          "&zoom=" + zoom +
                          "&size=" + Integer.toString(mapForm.getContentWidth()) + "x" +
                          Integer.toString(mapForm.getContentHeight()) +
                          "&sensor=false" +
                          pathParam;

HttpConnection imgConn;
imgConn = (HttpConnection) Connector.open(location);
imgConn.setRequestProperty("Accept", "image/png");

int totalToReceive = imgConn.getHeaderFieldInt(Arg.CONTENT_LENGTH, 0);
InputStream is = imgConn.openInputStream();
Image mapImage = Image.createImage(is);
imgConn.close();
is.close();
```

Pentru a putea apela serviciul GoogleMaps pentru preluarea hărții împreună cu o serie de direcții incluse, aplicația foloseste tehnologia Ajax pentru platforma Java Me. Acesta din urmă furnizează o librarie pentru gestionarea cererilor de tip HTTP care returneză fișiere de tip XML sau JSON de la serviciile Web. În acest sens, clasa MapManager include o metodă denumită *getMapWithDirections()* care trimite o cerere HTTP serviciului Web GoogleMaps după care preia răaspunsul și îl procesează. Codul pentru aceste operații este prezentat mai jos:

```
final Arg[] args = {
        new Arg("origin", Double.toString(mapForm.getMyLocation().getLatitude()) + "," +
           Double.toString(mapForm.getMyLocation().getLongitude())),
        new Arg("destination", Double.toString(station.getLocation().getLatitude()) + "," +
           Double.toString(station.getLocation().getLongitude())),
        new Arg("mode", "walking"),
        new Arg("sensor", "false")
        };
Response response;
response = Request.get(GOOGLE_DIR_BASE, args, null, null);
Result result = response.getResult();
```

Procesarea răspunsului se face foarte simplu, limbajul necesar folosind doar ”.” și ”[ ]” pentru a selecta un element dorit din răaspuns. Un exeplu de o astfel de procesare este prezentat mai jos:

```
String duration = result.getAsString("routes[0].legs[0].duration.text");
String distance = result.getAsString("routes[0].legs[0].distance.text");
```

Pentru adăugarea diferitelor tipuri de markere pe hartă, MapManager include o serie de metode necesare realizări acestui lucru. Markerul este reprezentat de o imagine inclusă intr-un label care va fi adaugată peste imaginea ce conține harta. Pentru a ști unde va fi poziționat markerul pe ecran va trebui să se transforme coordonatele de latitudine și longitudine ale acestuia

într-un punct reprezentat de un pixel pe ecran. Un exemplu pentru adaugarea unui marker pentru un mijloc de transport în comun este prezentat mai jos:

```
public void addBusMarker(BusMarker marker) {
      Point point = convertLatLongToPixel(marker.getBus().getLocation());
         if (point.getX() < 0 || point.getY() < 0 || point.getX() > mapForm.getContentWidth() ||
             point.getY() > mapForm.getContentHeight())
         {
                  return;
         }
      mapContainer.addBusMarker(marker, point);
}
```

Mai intâi se calculează cu ajutorul metodei *convertLatLongToPixel(Coordinate location)* preluată din [9], pozitia pixelului în funcție de central hărții, după care se verifică dacă pixelul este vizibil pe ecran, și daca este vizibil se adaugă în container-ul ce deține harta, deasupra acesteia.

Cel de-al treilea pachet amintit la începutul discuției despre implementarea aplicației client MobileTicketingAndLocalization este WebService. Acesta include clase ce fac posibilă apelarea serviciului Web necesar. Aceste clase au fost generate automat din fișierul WSDL al serviciului Web cu ajutorul unui instrument inclus în programul WirelessToolkit de la Oracle.

În continuare este prezentat un exemplu de generare a unor astfel de clase cu WirelessToolkit 2.5.2.

Primul pas este introducerea URL-ului corespunzător fișierului WSDL sau numele acestuia, după care trebuie selectată destinația claselor ce vor fi generate precum și pachetul în care acestea vor fi așezate.

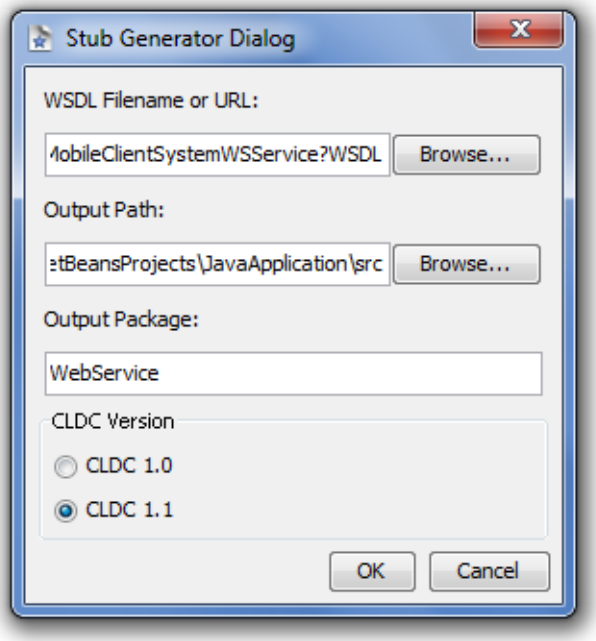


Figura 5-20 Instrument WirelessToolkit

După ce se execută butonul OK se vor genera clasele în directorul și pachetul selectat de unde vor putea fi folosite ulterior.

Mai jos este prezentat un exemplu de cod pentru apelarea unui serviciu pentru înregistrarea în sistem.

```
private boolean registerUser(String username, String pass, String fname, String lname) throws
RemoteException
{
      MobileClientSystemWS service = new MobileClientSystemWS_Stub();
      return service.register(username, pass, fname, lname);
}
```

TicketBluetoothServer

Acestă componentă a sistemului are datoria de a primi simultan un număr de conexiuni de la aplicația clientului și de a emite bilete pentru aceștia trimițând în același timp actualizările către baza de date prin apelarea serviciului Web necesar. Astfel aplicația prezintă două clase principale: **TicketServer** și **ThreadedClientHandler**. Ambele clase sunt clase care extind super-clasa Thread.

TicketServer crează un server Bluetooth. Cât timp este pornit acesta primește conexiuni de la clienți, creând câte un thread de tip ThreadedClientHandler pentru fiecare. Codul pentru crearea unei conexiuni L2CAP pentru server este afișat mai jos:

```
server = (L2CAPConnectionNotifier) Connector.open(
                        "btl2cap://localhost:" + UUID_STRING + ";name=" + SERVICE_NAME);
```

După crearea conexiunii, serverul așteaptă si acceptă conexiuni de la clienți, creând pentru fiecare un „handler”. Codul pentru această operație este următorul:

```
L2CAPConnection conn = server.acceptAndOpen();
      ThreadedClientHandler hand = new ThreadedClientHandler(conn, tcm);
        // create client handler
      handlers.addElement(hand);
      hand.start();
```

După ce a fost creat, obiectul de tip ThreadedClientHandler, preia datele trimise de client, apelează serviciul Web pentru actualizarea datelor clientului și îi trimite acestuia din urmă un bilet.

BusGPSProvider

Componenta furnizoare de coordonate GPS are ca și structură două clase pricipale: **GPSServer** și **MyLocation.** Clasa GPSServer este o clasă ce extinde super-clasa Thread, acest lucru datorându-se necesității serverului de GPS de a transmite periodic coordonatele de latitudine și longitudine ale mijlocului de transport în comun către baza de date. În acest sens, clasa thread GPSServer a fost implementată astfel încât, atâta timp cât thread-ul corespunzător este pornit, se va apela metoda de returnare a coordonatelor prin GPS din clasa MyLocation, după care coordonatele obținute vor fi trimise cu ajutorul unui serviciu web către baza de date, actualizând tabela corespunzătoare mijloacelor de transport în comun.

Codul pentru preluarea coordonatelor de latitudine și longitudine prin GPS este prezentat mai jos:

```
LocationProvider locProv;
Location location;
QualifiedCoordinates qCoord;
Coordinates myCoordinates = null;

try {
   locProv = LocationProvider.getInstance(null);
   location = locProv.getLocation(10);
   qCoord = location.getQualifiedCoordinates();
   if (qCoord != null) {
      myCoordinates = new Coordinates(qCoord.getLatitude(), qCoord.getLongitude(), 0);
   }
}
catch (LocationException locExc) {
   locExc.printStackTrace();
}
```

MobileTicketingAndLocalization

Având în vedere capabilitățile programului NetBeans, este foarte simplu de implementat un serviciu Web. Astfel, au fost implementate serviciile Web necesare sistemului cu ajutorul acestui instrument.
Mai jos este prezentat un exemplu de creare a unuei operații *„login”* pentru serviciul Web *MobileClientSystemWS*.

- Crearea unui serviciu Web nou numit MobileClientSystem

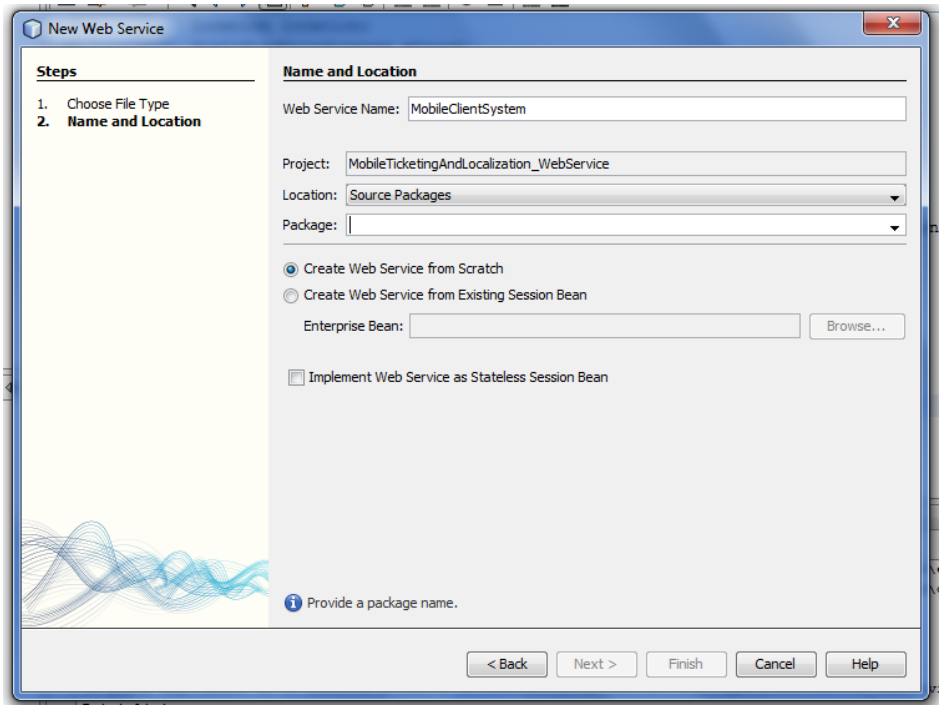


Figura 5-21 NetBeans – crearea unui serviciu Web – pasul 1

- Selectarea butonului „Add Operation”

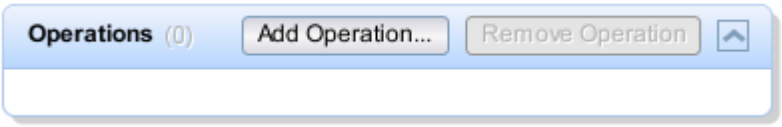


Figura 5-22 NetBeans – crearea unui serviciu Web – pasul 2

- Crearea unei operații *„login”*

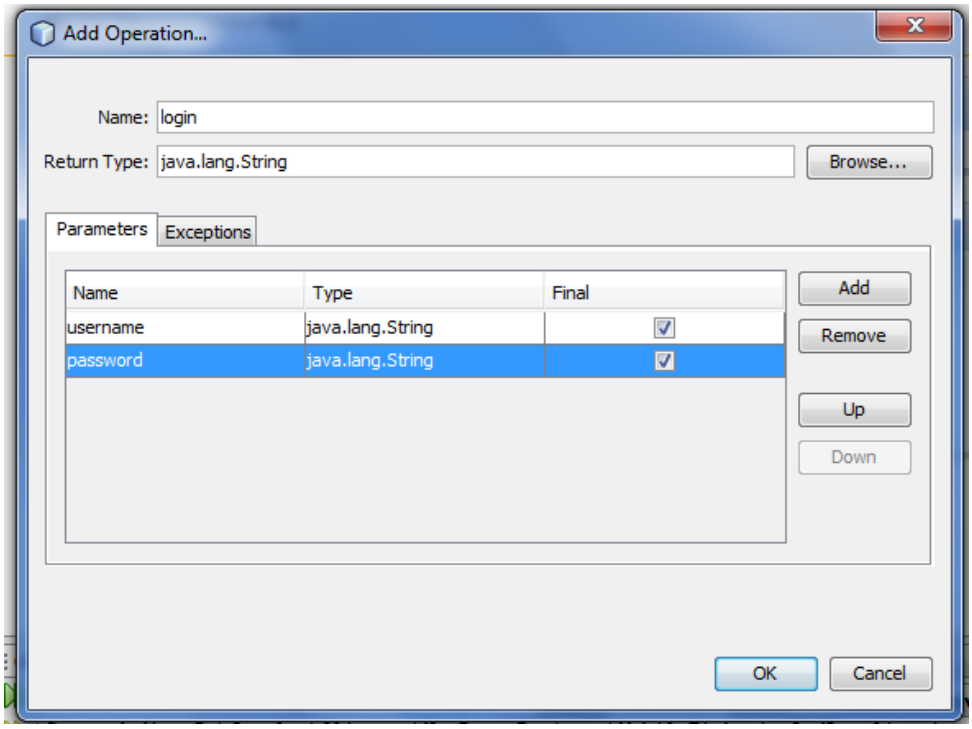


Figura 5-23 NetBeans – crearea unui serviciu Web – pasul 3

- După executarea butonului OK se va crea serviciul Web, care va putea executa operațiile implementate de programator în interiorul acestuia. Codul generat arată astfel:

```
@WebMethod(operationName = "login")
public String login(@WebParam(name = "username") final String username,
                    @WebParam(name = "password") final String password) {
        //TODO write your implementation code here:
    return null;
}
```

5.2.2. Interfață utilizator

O componentă importantă a sistemului de emitere de bilete și localizare a mijloacelor de transport în comun este interfața cu utilizatorul, deuarece acesta este cea care intră în contact direct cu acesta. O aplicație trebuie să aibă o interfață cât mai plăcută, simplă și ușor de folosit, având în vedere faptul că utilizatorul programului este un om simplu, care nu au cunoştinţe extraordinare despre calculatoare. După cum a fost amintit și în capitolele anterioare interfața cu utilizatorul se realizează cu ajutorul claselor de tip Form, care permit introducerea a numeroase elemente de grafică precum: butoane, label-uri, meniuri etc.

Singura aplicație din sistem care intră în contact direct cu clientul esre aplicația MobileTicketingAndLocalization. Clasele de tip Form incluse în aceasta sunt: **MainForm, UserForm, RegisterForm, LoginForm, MapForm, BusListForm, StationListForm.** Acestea prezintă o serie de elemente incluse cum ar fi: liste, câmpuri pentru text, comenzi etc. , care ajută utilizatorul să îndeplinească operațiile dorite asupra sistemului. Mai jos sunt ilustrate aceste Formuri împreună cu elementele incluse.

Figura 5-24 Form-uri: Main, Register, Login

Figura 5-24 prezintă formul inițial MainForm care are două comenzi pentru redirecționarea către formurile de înregistrare și autentificare. Acestea din urmă prezintă campurile necesate realizării acestor două acțiuni precum și o comandă care produce apelarea serviciilor Web pentru a putea interoga sau actualiza baza de date cu datele introduse de către client.

Alte formuri precum cele în care sunt afișate informațiile clientului, precum și cele pentru conectarea la serverele Bluetooth de bilete sunt prezentate mai jos.

Figura 5-25 Form-uri: User, Bluetooth

Figura 5-25 prezintă form-urile pentru detaliile client-ului,precum și cel pentru realizarea conexiunii la server-ul Bluetooth. Din UserForm utilizatorul poate alege comanda *TicketStamp* care îl va redirecționa către BluetoothForm unde se va putea conecta la un dispozitiv Bluetooth găsit. După realizarea conexiunii apăsând conamda *Select* va aparea un pop-up cu confirmarea preluării biletului.

De asemenea există o serie de formuri pentru gestionarea localizării mijloacelor de transport în comun pe hartă. Acestea sunt ilustrate în figura de mai jos.

Figura 5-26 Form-uri: BusList, Map, StationList

Figura 5-26 prezintă formurile pentru afișarea autobuzelor, pentru afișarea hărții GoogleMaps precum și cel pentru vizualizarea listei de stații. Utilizatorul va trbui mai întâi să aleagă o rută de autobuz, iar apoi acesta va fi redirecționat către hartă unde va putea vizualiza stațiile si autobuzele corespunzătoare rutei alese.

Capitolul 6. Testare şi validare

Testarea Software poate fi definită ca un proces de validare și verificare a faptului că un program/aplicație/produs software corespunde funcționale și non-funcționale care au ghidat proiectarea și implementarea lui. În acest sens programul trebuie să ruleaze și să se comporte corespunzător așteptărilor. Scopul principal pentru procesul de testare este identificarea erorilor de software, izolarea şi fixarea/corectarea defectelor care au cauzat aceste erori. Testarea nu poate demonstra cu certitudine de 100% ca produsul funţionează corect în orice condiţii; testarea doar poate demonstra că produsul nu funcţionează corect în anumite condiţii.

În cadrul procesului de testare s-a urmărit ca toate componentele sistemului să își îndeplinească în mod corect funcţionalităţile asignate. Pentru uşurarea procesului de dezvoltare si corectare rapidă și cât mai exactă a erorilor survenite în dezvoltare m-am folosit de facilităţile puse la dispoziţie de mediul de dezvoltare NetBeans, cum ar fi modul debug-ing sau vizualizarea conţinutului excepţiilor pentru tratarea acestora.

Pentru componenta de servicii a sistemului de localizare si emitere de bilete pentru mijloace de transport în comun, NetBeans prezintă o modalitate de testare a serviciilor Web implementate. Astfel, după ce serviciului Web a fost creat și i s-au implementat operațiile necesare acestuia, se poate executa click-dreapta pe acesta și selectarea operației „Test Web Service”. Figura de mai jos ilustrează această operațiune.

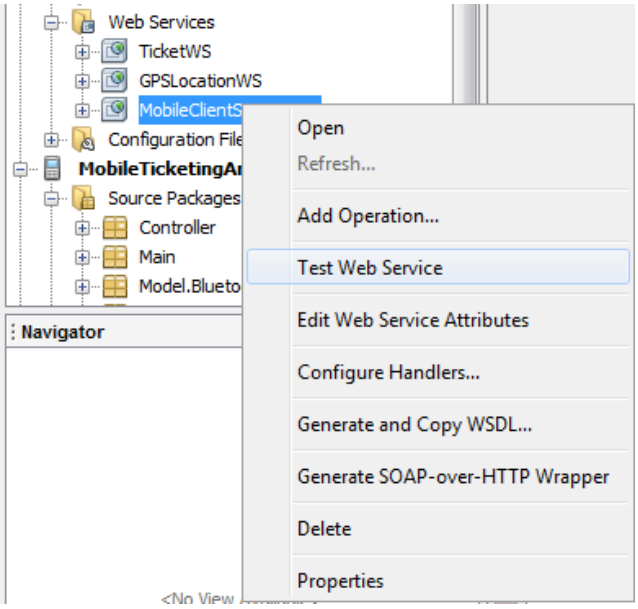


Figura 6-1 NetBeans – testare serviciu Web

După apăsarea click-ului programatorul va fi redirecționat către un browser unde va putea testa operațiile descrise de serviciul Web căruia i s-a realizat testarea. Pagina „Testet” ce va fi deschisă în browser conține câmpuri pentru text precum și butoane cu ajutorul cîrora se va realiza testarea. Componentele aceste pagini de testare sunt ilustrate în figura următoare.

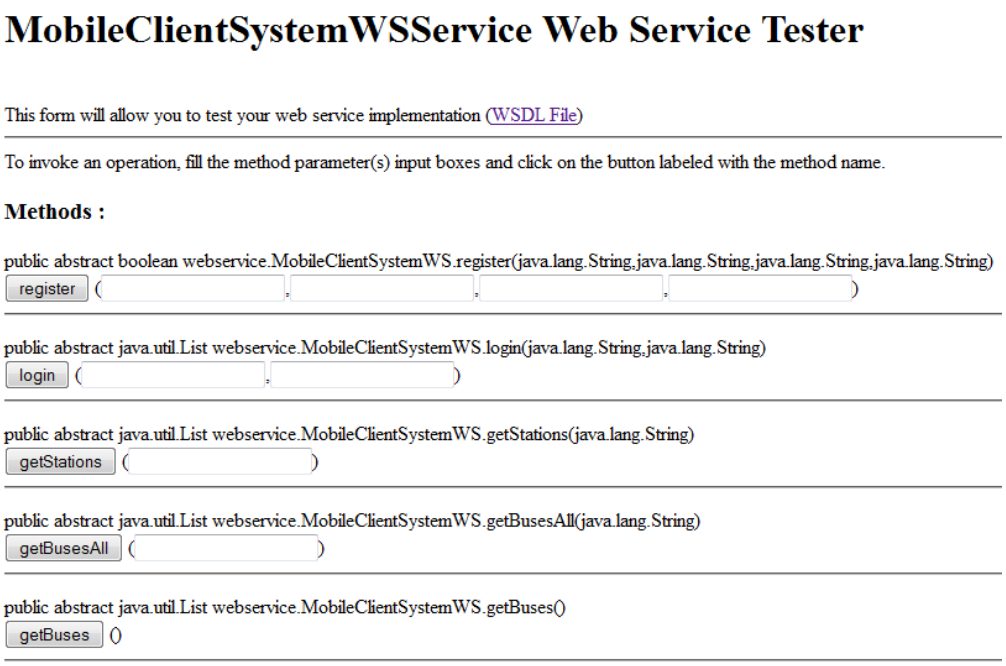


Figura 6-2 Pagina Web – testare serviciu Web

Un alt pas în testarea aplicației a fost reprezentat de testarea comunicării între componetele sitemului. A fost nevoie de testarea comunicări între:

- Aplicația clientului (MobileTicketingAndLocalizatio) și Serverul Bluetooth pentru bilete.
- Aplicația clientului și serviciile Web.
- Serverul Bluetooth pentru bilete și serviciile Web.
- Serverul furnizor de coordonate GPS și serviciile Web.

Scenariul de testare a funcționării comunicării între aplicații a fost următorul:

1. S-a pornit server-ul de furnizare de coordinate GPS.
2. S-a pornit serverul Bluetooth de emitere de bilete.
3. S-a pornit aplicația de servicii Web.
4. S-a pornit aplicația clientului (MobileTicketingAndLocalizatio).
5. Din cadrul acesteia din urmă s-a încercat autentificarea în sistem.
6. Apoi s-a încercat conectarea la serverul Bluetooth de bilete tot din cadrul aceleiași aplicații.
7. Un mesaj de confirmare a apărut pe ecran.
8. S-a selectat o rută de mijloace de transport în comun și s-a selectat comanda Map.
9. Harta GoogleMaps a fost afișată pe ecran împreună cu stațiile corespunzătoare rutei alese.
10. S-a ales apoi comanda pentru localizarea și afișarea mijloacelor de transport în comun.
11. Markere cu poziția mijloacelor de transport au fost afișate pe harta GoogleMaps vizibilă p ecran.

S-a constatat în acest mod funcționarea corectă a aplicației și comunicarea cu success între aplicații. În urma testelor efectuate s-a stabilit funcționarea corectă a sistemului si respectarea specificațiilor proiectului.

Capitolul 7. Manual de instalare si utilizare

Programul de emitere de bilete și localizare a mijloacelor de transport în comun vine cu un fișier de instalare (formatul *. jar) care trebuie să fie copiat și memorizat pe telefon. După copierea aplicației acesta trebuie instalată. Acest lucru se realizează prin executarea fișierului .jar. Un mesaj de confirmare va aparea după care daca răspunsul a fost afirmativ aplicația se va instala în interiorui dispozitivului mobil La diiferite tipuri de telefoane, instalarea are loc într-un mod diferit. Dacă aveţi probleme cu instalarea aplicației, consultaţi documentaţia cu privire la instalarea acesteia, furnizată de către producătorul telefonului. De obicei, informaţii detaliate cu privire la instalarea de aplicații sunt disponibile pe site-ul producătorului de telefon. După instalarea programului acesta va fi disponibil într-un anumit fișier din telefon, de obicei, o secţiune numită "Aplicații."

Există o serie de pași ce trebuie realizați pentru a putea utiliza cu succes serviciile oferite de program. Prima operațiune ce trebuie realizată este cea de autentificare/înregistrare. Pașii necesari realizării acestui serviciu sunt următorii:

- Înregistrare
  - Selectarea comenzii *Register* din form-ul inițial (Main)
  - Completarea câmpurilor
  - Executarea comenzii *Register*
- Autentificarea
  - Selectarea comenzii *Login* din form-ul inițial (Main)
  - Completarea câmpurilor
  - Executarea comenzii *Login*

Cele trei interfețe implicate în acest proces sunt ilustrate mai jos.

Figura 7-1 Operații: login/register

Pentru operația de preluare a unui bilet trebuie respectați următorii pași:

- Preluare bilet
  - Selectarea comenzii *Ticket Stamp* din interfața User
  - Selectarea unui dispozitiv
  - Executarea comenzii *Select*
  - Selectarea comenzii *Yes* pentru confirmare

Cele trei interfețe implicate în acest proces sunt ilustrate mai jos.

Figura 7-2 Operație: preluare bilet

Pentru operațiile asupra hărții GoogleMaps există o serie de pași descriși mai jos în funcție de serviciul ales.

- Vizualizare mijloace de transport în comun pe hartă
  - Selectarea comenzii *Buses* din interfața User
  - Selectarea unei rute de autobuz după care se execută comanda Map
  - Executarea comenzii *Show Buses* din interfața Map
- Calcularea și vizualizarea unei rute de la poziția clientului la o stație
  - Executarea comenzii My Location din intefața Map
  - Selectarea unei stații
  - Executarea comenzii *Directions* din interfața Map

Cele trei interfețe implicate în acest proces sunt ilustrate mai jos.

Figura 7-3 Operații: asupra hărții GoogleMaps

Capitolul 8. Concluzii

8.1 Realizări

Un proiect este rezultatul unui set de activităţi, propuse la începutul acestuia şi a celor apărute pe parcurs, propuneri care trebuie realizate pentru finalizarea proiectului.Pentru realizarea sistemului au fost îndeplinite următoarele activităţi principale: documentarea referitoare la subiectul care urmează a fi abordat, identificarea cerinţelor funcţionale şi non funcţionale, alegerea unui limbaj de programare pentru îndeplinirea cerinţelor definite anterior, analiza şi proiectarea arhitecturii sistemului, implementarea propriu-zisă, testarea, precum şi documentarea privind posibilităţile extinderi sistemului.

Într-un timp relativ scurt m-am acomodat cu noile tehnologii necesare dezvoltării aplicaţiei, în contextual în care limbajul orientat pe obiect îmi era familiar. Însă folosirea limbajului Java în programarea aplicaţiilor destinate dispozitivelor mobile a fost o adevărată provocare. Cu toate acestea mediul de dezvoltare NetBeans împreună cu emulatorul pentru programarea pe dispositive mobile Java(TM) Platform Micro Edition SDK 3.0, au oferit o serie de servicii care au uşurat considerabil munca de programare. La nivelul de studiu bibliografic am învăţat lucruri noi despre limbajul Java destinat programării pe mobile, precum şi despre funcţionalităţile pe care acesta le poate implementa, cum ar fi: apelarea serviciilor web, preluarea coordonatelor de latitudine şi longitudine sau conectarea la alte dispozitive folosind tehnologia bluetooth.

În final s-a realizat un sistem format din patru componente, care împreună comunică şi realizează operaţiile descrise în cerinţele proiectului.

8.2 Dezvoltări ulterioare

Obiectivelor prezentate la începutul proiectului au fost în mare parte realizate, însă bazat pe arhitectura de proiectare, unele servicii on-line, cum ar fi serviciul pentru trafic poate fi uşor introdus în proiect pentru a putea furniza servicii mai bune şi mai bogate de călătorie cu mijloacele de transport în comun. Dar din cauza timpului limitat, există, de asemenea, unele funcţionalităţi lăsate pentru implementare în viitor.

Prima înbunătăţire care trebuie să fie făcută în viitor este de a reduce dimensiunea de transfer de date. Acest lucru se datorează faptului că utilizatorii acestui sistem nu pot beneficia de conectarea la un mediu Wi-Fi gratuit tot timpul. Astfel, utilizatorii trebuie să utilizeze serviciile de internet inclus în reţeaua de telefonie mobilă la care sunt abonaţi (de exemplu Wap, 3G) care poate avea un cost ridicat pentru transferul datelor. Deci, activitatea de reducere a dimensiunii de transfer de date pentru a optimiza debitul este destul de semnificativă.

Următorul aspect care trebuie să fie luat în considerare în viitor este îmbunătăţirea securităţii sistemului. Acesta va fi realizată astfel:

- Îmbunătăţirea fiabilităţii parolei. Deoarece parola este selectată de către utilizatori, uneori nu este suficient de sigură. De exemplu, unele parole poate include numele utilizatorilor, sau ziua de naştere a acestora. Prin urmare, atunci când se creează o nouă parolă, sistemul trebuie să verifice dacă această parolă este suficient de "sigură". Standardul unei parole sigure va fi folosit în sistem acesta interzicând ca acesta să conţină numele utilizatorilor şi ziua de naştere, şi necesitând să aibă litere mari, litere mici, numere şi simboluri speciale (de exemplu, "%" sau "*").
- Criptarea datele sensibile. Pentru a proteja datele sensibile în baza de date, sistemul va cripta datele înainte de depozitare. Algoritmul de criptare se propus pentru a fi utilizat este

Advanced Encryption Standard (AES). Mai mult decât atât, atunci când se transmit date sensibile, pentru a preveni interceptarea, sistemul va folosi Transport Layer Security (TLS) protocolul pentru a asigura transferul de date.

Bibliografie

[1] Neate, R.,. *Mobiles to replace wallets and tickets.* Retrieved from The Telegraph (2010): http://www.telegraph.co.uk/technology/7138873/Mobiles-to-replace-wallets-and-tickets.html , ultima accesare 2 mai 2011

[2] WordPress., *Is Bluetooth better than SMS?* Retrieved from Is Bluetooth better than SMS Marketing?, from Businesses adopting Bluetooth Smartphone Marketing (2010): http://isendco.wordpress.com/2010/09/21/is-bluetooth-better-than-sms-marketing/ , ultima accesare 5 mai 2011

[3] Roos, D., *How Mobile Ticketing Works.* Retrieved from How Stuff Works (2011): http://communication.howstuffworks.com/how-mobile-ticketing-works.htm , ultima accesare 5 mai 2011

[4] Penna. , *M-Ticketing Solutions.* Retrieved from Penna Mobile Solution (2011): http://www.pennaltd.com/en/MTicketing.aspx , ultima accesare 25 mai 2011

[5] MAXArtists., *Mobile Ticketing Delivery solution* (2009): http://www.maxartists.com/images/media/Mobileticketing.pdf , ultima accesare 25 mai 2011

[6] JMango., *MangoTix - Mobile Ticketing* (2011): http://jmango.net/index , ultima accesare 25 mai 2011

[7] Akhil Arora, S. S, *Mobile Ajax for Java ME*. (2010): http://www.w3.org/2007/06/mobile-ajax/papers/sun.hardy.mobileAjaxJavaME.pdf , ultima accesare 2 iunie 2011

[8] ORACLE, LightWeight User Interface. (2011): http://www.oracle.com/us/technologies/java/lwuit-datasheet-167821.pdf , ultima accesare 2 iunie 2011

[9] Microsoft. *Upgrading to Bing Maps - Converting Coordinates from Pixels*. (2010): http://www.bingmapsupgrade.com/assets/files/MapPoint%20Upgrade%20Guide%20-%20Converting%20Coordinates%20from%20Pixels.pdf?eventID=90&custID=&indID= , ultima accesare 20 iunie 2011

[10] Civitas, U. ,*Civitas Succes*. Retrieved from CIVITAS: http://www.civitas-initiative.org/project_sheet?id=4&language=ro , ultima accesare 30 mai 2011

[11] Pamminger, H, *NFC: The New Dimension of Mobile Ticketing and Acces Control*.(2007): http://www.nfc-research.at/fileadmin/congress/2007/slides/07_Skidata_Herbert_Pamminger.pdf , ultima accesare 25 mai 2011

[12] Salomie, Ioan, *Cursuri: Tehnici de programare, Sisteme distribuite* (2011).

[13] Dadarlat, Vasile, *Cursuri: Rețele de calculatoare (2011).*

Anexa 1.

Listă de figuri

Glosar de termeni și acronime

MVC – șablonul arhitectural model-view-controller

LWUIT - Lightweight User Interface Toolkit: librărie pentru elemente de interfață grafică

GPS - Global Positioning System: sistem de poziționare pe glob

J2ME – Java to Micro Edition

MTDS – Mobile Ticketing and Delivery System

API - Application Programming Interface: interfață pentru programarea aplicațiilor

MTL Client System – Mobile Ticketing and Localization Client System

MTL WebService - Mobile Ticketing and Localization WebService

QRCode – prescurtarea de la *Quick Response code:* model de matrice de cod format din cuburi negre așezate pe fundal alb

CIVITAS - Cleaner and better transport in cities: inițiativă europeană pentru modernizarea trasportului în orașe

MMS – Multimedia Messaging Service: standard pentru trimiterea de mesaje pe telefon

SMS – Short Message Service: standard pentru trimiterea de mesaje pe telefon

WAP - Wireless Application Protocol: tehnologie standard care permite dispozitivelor fără fir să navigheze pe Internet