Cuprins

Capitolul 1. INTRODUCERE

1.1 Context proiect

Într-o lume în care avansul tehnologic a ajuns la stadiul unei creșteri exponențiale, disponibilitatea noilor tehnologii (precum tehnologiile Web, multimedia, dar în special tehnologii de comunicație) și a dispozitivelor digitale portabile se află în continuă creștere. În prezent, omul este dependent de tehnologie și telecomunicații, dedicându-se evoluției acestora, dar mai ales utilizării lor, în viața de zi cu zi. Ca urmare, se observă rapiditatea cu care modelele digitale au înlocuit sistemele analogice în domenii din diferite sfere de activitate: comunicație (telefonie, televiziune, radiodifuziune), divertisment (fotografie, muzică, film), medicină, educație, agricultură, industrie etc.

Tot efortul depus, de-a lungul timpului, pentru implementarea unei noi viziuni asupra aspectelor esențile ale vieții se fundamentează pe necesitatea simplificării sarcinilor care revin fiecărei persoane în parte și se manifestă prin încercarea de a sprijini sau chiar de a înlocui o serie din atribuțiile personale. Aceste lucruri sunt posibile datorită interesului major acordat evoluției tehnologiei în toate ariile de interes ale populației, în direcția unei asistențe fidele din partea noilor tendințe și concepte.

O atenție sporită este acordată dezvoltării și eficientizării sistemelor de comunicație. Aceasta reprezintă o condiție a evoluției și diversificării continue a funcțiilor dispozitivelor de comunicație, în conformitate cu cererile din ce în ce mai ample. Se observă o tendință tot mai mare de a reduce sau chiar de a elimina limitele dintre diferite modalități de mediatizare.

Secolul XX a marcat evoluția telefonului mobil de la un privilegiu, la un mijloc de comunicare indispensabil. De asemenea, sfârșitul secolului XX a marcat trecerea de la tehnologia analogică la cea digitală, corespunzătoare unei noi ere a telefoniei mobile. Se trece de la simplele telefoane mobile, capabile de efectuarea apelurilor telefonice si schimbului de mesaje, la telefoane din ce în ce mai performante, prin adăugare de noi funcții și îmbunătățirea funcțiilor existente.

Rezultatul acestei evoluții este reprezentat de telefoanele inteligente (**smartphone**-urile) și alte dispozitive mobile, dezvoltate pe baza unui sistem de operare mobil. Dispozitivele multifuncționale pot fi utilizate atât pentru a efectua apeluri telefonice și a realiza schimbul de mesaje, cât și pentru a face fotografii, a înregistra filme scurte, a interpreta fișiere multimedia etc. Telefoanele inteligente au, pe lângă toate aceste competențe, volum mai mare de memorie, o mai bună capacitate de conectare și procesare, precum și disponibilitatea unor aplicații, cu diferite funcționalități, instalate inițial. Cel mai mare avantaj adus de aceste dispozitive este acela că noi aplicații pot fi instalate prin descărcarea lor de pe Internet sau prin implementarea unor aplicații personalizate, de către utilizatorii dispozitivului.

1.2 Obiective și specificații

În prezent, se pune problema economisirii timpului, care reprezintă cea mai prețioasă resursă de care dispunem. Toată lumea a întalnit, cel puțin o dată, situații în care a fost nevoită să ajungă în multe locuri, într-un timp relativ scurt. În general, aceste deplasări se bazează pe mijloacele de transport, în special în cazul unor distanțe mai mari. Multe persoane sunt dependente de mijloacele de transport în comun, întrucât nu dispun de automobile personale. Însă, aceste mijloace de transport în comun (tramvaie, troleibuze, autobuze, trenuri, minibus-uri, autocare), deși au un orar de funcționare, nu asigură o respectare strictă a acestuia, deoarece depind foarte mult de trafic, având un traseu preastabilit, pe care nu au dreptul să îl modifice. Numărul foarte ridicat al automobilelor implicate în trafic, în special la ”orele de vârf”, determină nesiguranța promptitudinii în cazul transportului în comun. Așadar, se ajunge la considerarea unei soluții alternative, și anume taxiurile, care sunt disponibile la orice oră din zi și din noapte, care pot urma trasee la cerere, pot ajunge în orice locație pentru a prelua comanda, iau în considerare cele mai scurte rute, pot schimba, în orice moment, traiectoriasau pot opri la cerere.

Rezervarea unui taxi poate fi efectuată la întâmplare, în momentul întâlnirii unui taxi disponibil, în drum spre destinația propusă, prin apel telefonic la dispeceratele companiilor de taxi-uri sau, cea mai nouă și mai simplă metodă, pentru posesorii dispozitivelor cu sistem de operare mobil (smartphone, tabletă etc.), prin accesarea aplicațiilor de comandă taxi.

Prima metodă se bazează pe un noroc întâmplător, depind de circumstanțele de moment, care nu pot fi controlate în vreun fel. Metoda clasică de rezervare a taxi-urilor implică efectuarea unui apel telefonic de către o persoană care își explimă dorința de a se deplasa cu taxi-ul spre o anumită destinație. Dispecerul cere informații legate de locația clientului și de destinația dorită, iar apoi, prin stații emisie-recepție, comunică tuturor taximetriștilor noua cerere efectuată, cu toate informațiile de la client. Cel care se consideră cel mai apropiat de locația actuală a clientului și este disponibil, preia comanda și pornește spre acesta, estimând timpul necesar pentru a ajunge în locul respectiv. Dispecerul înștiințează, ulterior, clientul despre timpul de așteptare a taxiului.

A treia metodă, de actualitate și aflată în plină dezvoltare, reprezintă o îmbunătățire a eficienței și eficacității companiilor de taxiuri, în ceea ce privește timpul și cheltuielile companiei, dar și mai mult, timpul clientului. Aceasta vine ca o alternativă îmbunătățită a felului clasic în care se proceda pentru a rezerva taxiuri. Aplicațiile pentru comandă taxi sunt ușor accesibile persoanelor care dispun de dispozitive cu sisteme de operare mobile, prin simpla lor descărcare de pe Internet. Având în vedere faptul că numărul utilizatorilor de astfel de dispozitive (în special smartphone-uri) este în continuă creștere, se justifică interesul pentru diversificarea și îmbunătățirea acestor sisteme.

Tema acestei lucrări are în vedere dezvoltarea unei aplicații de rezervare a taxiurilor, pentru sistemul de operare Android, disponibil pe o mare varietate de dispozitive mobile. Scrierea aplicațiilor în Android este, de asemenea, gratuită, spre deosebire de alte sisteme de

operare mobile (iOS – principalul rival), lucru care asigură portabilitatea și accesibilitatea ridicată a aplicațiilor construite pentru acest sistem de operare (OS).

Principalele obiective ale acestei lucrări sunt bazate pe:

- Analiza aplicabilității și necesității, în viața cotidiană, a acestui tip de sitem;
- Studierea facilităților oferite de aplicațiile de rezervare a taxiurilor, construite pentru dispozitive mobile, precum și cercetarea câtorva sisteme existente și a funcționalităților acestora;
- Conceperea unei arhitecturi fundamentale, aplicabilă unui sistem de gestiune a comenzilor efectuate de utilizatori, cu modulele participante la proces;
- Proiectarea și implementarea unei aplicații care să rapundă cererilor utilizatorilor, prin rezervarea celui mai apropiat taxi, asigurând o promptitudine ridicată și o înștiințare cât mai exactă a clientului asupra informațiilor comenzii;
- Testarea aplicației și evaluarea rezultatelor obținute. Studierea eventualelor posibilități de îmbunătățire a soluției software alese.

Capitolul 2. STUDIU BIBLIOGRAFIC

2.1 Telefoane mobile – scurt istoric

Pentru a putea înțelege ce anume face ca Android-ul să aibă un efect atât de puternic și aproape irezistibil asupra utilizatorilor de pretutindeni, este bine să facem o scurtă călătorie în trecutul telefoanelor mobile, pentru a observa evoluția acestora, în timp, odată cu apariția sistemelor de operare mobile.

În primul rând, ar trebui să ne amintim de vremurile în care telefonul era ceva grandios, rar întâlnit, doar în familiile înstărite; telefonul fix cu manivelă sau cu disc, ulterior cel cu tastatură erau un lux pe care doar cei mai norocoși oameni și-l permiteau. Erau zilele în care trebuiau făcute liste de cumpărături, pentru a evita reîntoarcerea la magazin, în cazul în care s-a omis ceva, întâlnirile trebuiau programate cu mare rigurozitate, pentru a evita așteptarea în zadar sau posibilitatea de a înțelege greșit locul de întâlnire și multe alte lucruri asemănătoare puteau constitui o adevărată aventură și mult timp pierdut.

În zilele noastre, însă, toate aceste întâmplări pot fi evitate cu mare ușurință, totul rezolvându-se printr-un singur apel dat cu ajutorul telefoanelor mobile; acestea ne țin conectați cu lumea înconjurătoare, nu doar prin faptul că putem păstra o legatură strânsă cu prietenii, familia și colegii, ci și pentru că ne ajută în anumite situații prin funcționalitățile și aplicațiile disponibile: calculator, calendar, alarmă, cronometru, cameră foto, jocuri, radio, sistem de navigație prin GPS (Global Positioning System), aplicații pe Internet etc.

Evoluția telefoanleor mobile a întâlnit multe piedici și situații dificile, de-a lungul istoriei dezvoltării lor la nivel software și chiar hardware, însă, în ciuda tutror faactorilor nefavorabili, tehnologia a avut un puternic impact asupra acestui domeniu. Primul telefon celular disponibil a fost Motorola DynaTAC 8000X, fabricat în 1984 și poreclit ”cărămida”, datorită dimensiunilor specifice, comparabile cu cele ale unei cărămide. Această primă genrație de telefoane era extrem de scumpă, deși bateria dura cât pentru jumătate de converație. Singurii care își permiteau acest lux erau oamenii de afaceri care călătoreau mult, personalul de securitate și oamenii bogați. În timp, prețul telefoanelor a început să scadă, bateriile s-au îmbunătățit, ariile de recepție s-au lărgit și din ce în ce mai mulți oameni au început să le folosească. Astfel, au crecut tot mai mult cerințele pentru ca telefoanele mobile să satisfacă toți utilizatorii: îmbunătățiri aduse comportamentului existent, adăugare de noi activități, reducerea dimensiunilor etc.

În curând au apărut alte modele de telefoane: cu clapetă (Motorola StarTAC – 1996), cu slider/cursor (Nokia 8110 – 1996, cu porecla ”The Matrix Phone”, pentru că era foarte des utilizat în filme) și alte modele de tip bloc, precum ”cărămida”. Toate noile generații au adus îmbunătățiri, noi tehnologii și au început să reducă prețurile, deoarece se observa o creștere în numărul persoanelor interesate de procurarea unui astfel de dispozitiv, acestea conștientizând

utilitatea lor. Aceste prime modele aveau ecrane cu rezoluție mică, spații limitate de depozitare și capacitate redusă de procesare. Figura 2.1.1 prezintă evoluția primelor telefoane apărute.

Figura 2.1.1 Evoluția primelor generații de telefoane mobile (celulare)

În continuare, evoluția telefoanelor mobile a fost accelerată, apărând tot mai multe mărci producătoare: Alcatel, Sony Ericson, Samsung, LG și s-a pus un accent tot mai mare pe disponibilitatea rețelei de Internet pe mobil, însă modelele clasice nu puteau face față operațiilor intensive de date necesare navigatoarelor web tradiționale.

Astfel, a apărut standardul WAP (Wireless Application Protocol), o versiune simplificată a HTTP (HyperText Transmission Protocol), protocol de bază al Internetului. Protocolul WAP a fost proiectat să ruleze pe dispozitive cu constrângeri de memorie și de lățime de bandă, precum telefoanele din vremea respectivă. Comercializarea aplicațiilor WAP era dificilă, unele din cele mai populare fiind simplele poze de fundal (wallpapers) și tonuri de apel (ringtones) cu care utilizatorii își puteau personaliza, pentru prima oară, telefoanele. Însă, aceste abilități au reprezentat doar începutul unei ere în care cerințele utilizatorilor devin din ce în ce mai intense și pretențiile din ce în ce mai mari.

După ce au fost parcurși mai mulți pași în evoluția telefoanelor mobile (rularea unor dispozitive pe sistemul de operare Palm sau RIM BlackBerry, Windows CE/Pocket PC, iOS – anul 2008), au apărut smartphone-urile (anul 2001) – telefoane mobile proiectate cu un sistem de operare mobil, cu capacități de calcul și conectivitate mai avansate decât telefoanele mobile tradiționale. Primul smartphone, apărut în 1994, a dispus de tastatură reală sau virtuală, datorită unui display touchscreen, și a combinat funcțiile unui PDA (Personal Digital Assistant) – dispozitiv mobil utilizat pe post de manager al informațiilor personale – cu cele ale telefonului mobil. În figura 2.1.2 este ilustrat primul smartphone din lume, caruia i s-a dat numele Simon.

Figura 2.1.2 Primul smartphone numit Simon

Modelele ulterior apărute, au adăugat pe lista funcționalităților player-e media portabile, camere digitale, camere video pocket, sisteme GPS înglobate pentru orientare etc. Multe modele de smartphone moderne sunt concepute cu touchscreen și navigatoare de web care permit afișarea unor pagini web standard. Sistemele de operare mobile folosite de smartphon-urile moderne sunt Android de la Google, iOS de la Apple, Symbian de la Nokia, BlackBerry OS, Windows Phone de la Microsoft etc.

Inițial, corporația multinațională Google, apoi toate companiile membre ale OHA (Open Handset Alliance): Sangung, HTC, Motorola, LG, precum și companiile de semiconductori ca Intel, Texas Instruments, NVIDIA, au ajutat la proiectarea primei generații de telefoane Android. Primul telefon cu Android expeiat (T-Mobile G1) a fost dezvoltat de producătorul HTC, în luna octombrie a anului 2008. Următoarele modele de telefoane cu Android au fost lansate în anii 2009 și 2010. Platforma a luat amploare relativ repede, ficare nou dispozitiv cu Android fiind din ce în ce mai puternic și mai captivant. În mai puțin de doi ani au apărut în jur de 60 de modele diferite de telefoane cu Android, în 48 de țări din întreaga lume, urmând ca, în luna iunie a anului 2010, Google să anunțe că mai mult de 160000 de dispozitive cu Android au fost activate în fiecare zi, ducând la o rată de aproape 60 de milionae de dispozitive anual [1].

2.2 Ce este Android-ul?

Android-ul este o platformă software și un sistem de operare pentru dispozitive și telefoane mobile, bazat pe o versiune modificată a nucleului Linux. Ca parte a strategiei de a se lansa în spațiul mobil, în anul 2005, compania Americană multinațională specializată pe produse și servicii asociate rețelei de Internet, Google, a susținut financiar și apoi a achiziționat Android, preluând și activitatea sa de dezvoltare, devenind astfel Android Inc. Apoi, în anul 2007, odată cu fondarea OHA, alianță formată din companii de software, hardware și telecomunicații, hotărâte să avanseze standardele dispozitivelor mobile, Android-ul a devenit cunoscut și a început să își câștige renumele.

Google și-a dorit ca Android-ul să fie gratuit. Așadar, framework-ul Android-ului a fost lansat sub Licența Apache, care este open source, adică orice dezvoltator de aplicații îl poate folosi gratuit, prin descărcarea completă a codului sursă Android. Trusa de dezvoltare software (SDK = Software Development Kit), precum și instrumentele (ADT = Android Develeopment Tools) sunt, de asemenea, gratuite pentru utilizatorii care construiesc aplicații (”apps”) pentru a extinde funcționalitatea dispozitivelor mobile, inițial scrise într-o variantă personalizată a limbajului de programare Java.

Mai mult decât atât, furnizorilor, în general fabrici de componente harware, le este permis să adauge propriile extensii și să personalizeze Android-ul după bunul lor plac, astfel încât să diferențieze produsul lor de cel al altor companii. Acest lucru face ca Android-ul să fie foarte atractiv, stârnind interesul multor furnizori. Acest lucru s-a adeverit, în special, în cazul companiilor afectate de fenomenul iPhone de la Apple, un produs de foarte mare succes, care a revoluționat industria telefoanelor smartphone. Aici sunt incluse companii precum Motorola și Sony Ericsson, care pentru mulți ani au dezvoltat propriile lor sisteme de operare. Când a fost lansat iPhone-ul, mulți din acești producători au trebuit să se zbată pentru a găsi noi metode a readuce la viață produsul lor, având în vedere concurența acerbă. În cele din urmă, producătorii au considerat Android-ul soluția potrivită, continuând să proiecteze propriile lor produse, dar să folosească sistemul de operare Android.

Având în vedere faptul că, în lumea smartphone-urilor, aplicațiile reprezintă cea mai importantă parte a unui succes înlănțuit, principalul avantaj în adoptarea sistemului de oparare Android este acela că oferă o abordare unitară în dezvoltarea aplicațiilor. Dezvoltatorii trebuie doar să producă pentru Android, iar aplicațiile lor ar trebui să poată rula pe numeroase dispozitive diferite, atât timp cât pe acestea rulează sistemul de operare Andorid. Astfel, producătorii consideră Android-ul ca fiind cea mai potrivită ”armă” în lupta cu sistemul iPhone, care deja comandă o mare diversitate de aplicații [2].

Întrucât telefoanele cu Android și aplicațiile asociate au devenit foarte ușor accesibile, mulți alți operatori de telefonie mobilă și producători de telefoane mobileau profitat de ocazia de a vinde telefoane cu Android abonaților lor, mai ales ținând cont de beneficiile legate de cost, în comparație cu propriile platforme.

Așadar, Android-ul este cea mai populară platformă din lume. Cu Android, pot fi folosite toate aplicațiile Google deja cunoscute, iar pe lângă acestea mai sunt disponibile peste 600000 de aplicații și jocuri, pe Google Play, împreună cu filme, muzică și cărți. Dispozitivele cu Android sunt deja deștepte (en. ”smart”), dar pot deveni mult mai deștepte, prin adăugarea unor caracteristici care nu există la alte platforme, permițând utilizatorilor să se concentreze asupra a ceea ce e important pentru ei, determinându-i să preia controlul asupra experienței telefonului mobil [1] [3].

2.3 Versiuni de Android

De la lansare, Android-ul a trecut prin multe etape de actualizare, iar de fiecare dată când s-a modificat versiunea anterioară, i-a foat atribuit un alt nume noii versiuni. În tabelul 2.3.1 sunt enumerate versiunile disponibile de Android, împreună cu numele lor de cod. Acestea sunt reprezentate în ordine crescătoare după nivelul lor API (API-level).

Tabel 2.3.1 Versiuni de Android

Versiunea	Data apariției	API-level	Denumirea
Android 1.0	23 Septembrie 2008	1	
Android 1.1	9 Februarie 2009	2	
Android 1.5	30 Aprilie 2009	3	Cupcake
Android 1.6	15 Septembrie 2009	4	Donut
Android 2.0	26 Octombrie 2009	5	Eclaire
Android 2.01	3 Decembrie 2009	6	Eclaire
Android 2.1	12 Ianuarie 2010	7	Eclaire
Android 2.2	**20 Mai 2010**	**8**	**Froyo**
Android 2.2.1	18 Ianuarie 2011	8	Froyo
Android 2.2.2	22 Ianuarie 2011	8	Froyo
Android 2.2.3	21 Noiembrie 2011	8	Froyo
Android 2.3	6 Decembrie 2010	9	Gingerbread
Android 2.3.1	6 Decembrie 2010	9	Gingerbread
Android 2.3.2	Ianuarie 2011	9	Gingerbread
Android 2.3.3	9 Februarie 2011	10	Gingerbread
Android 2.3.4	28 Aprilie 2011	10	Gingerbread
Android 2.3.5	25 Iulie 2011	10	Gingerbread
Android 2.3.6	2 Septembrie 2011	10	Gingerbread
Android 2.3.7	21 Septembrie 2011	10	Gingerbread
Android 3.0	22 Februarie 2011	11	Honeycomb
Android 3.1	10 Mai 2011	12	Honeycomb
Android 3.2	15 Iulie 2011	13	Honeycomb
Android 3.2.1	20 Septembrie 2011	13	Honeycomb
Android 3.2.2	30 Septembrie 2011	13	Honeycomb
Android 3.2.3	30 August 2011	13	Honeycomb
Android 3.2.4	Decembrie 2011	13	Honeycomb
Android 3.2.5	Ianuarie 2012	13	Honeycomb
Android 3.2.6	Februarie 2012	13	Honeycomb
Android 4.0	19 Octombrie 2011	14	Ice Cream Sandwich
Android 4.0.1	21 Octombrie 2011	14	Ice Cream Sandwich
Android 4.0.2	28 Noiembrie 2011	14	Ice Cream Sandwich
Android 4.0.3	6 Decembrie 2011	15	Ice Cream Sandwich
Android 4.04	29 Martie 2012	15	Ice Cream Sandwich
Android 4.1.	9 Iulie 2012	16	Jelly Bean

Android 4.1.1	23 Iulie 2012	16	Jelly Bean
Android 4.1.2	9 Octombrie 2012	16	Jelly Bean
Android 4.2.	13 Noiembrie 2012	17	Jelly Bean
Android 4.2.1	27 Noiembrie 2012	17	Jelly Bean
Android 4.2.2	11 Februarie 2013	17	Jelly Bean
Android 4.3	Ongoing	18	Jelly Bean

Începând cu anul 2008, versiunilor de Android le-au fost atribuite nume de cod, inspirate din denumirile unor deserturi. În funcție de data apariției, versiunile au fost lansate în ordine alfabetică: **cupcake**, **donut**, **eclair**, **froyo**, **gingerbread**, **honeycomb**, **ice cream sandwich**, **jelly bean**, iar în viitor, odată cu versiunile Android 4.3, 4.4 sau 4.5, va fi lansat și numele de cod **key lime pie**. Figura 2.3.1 prezintă această trecere de la o versiune la alta, prin intermediul logo-urilor atribuite fiecărui nume de cod [1] [2] [4].

Figura 2.3.1 Evoluția versiunilor de Android, împreună cu logo-urile lor

2.4 De ce Android?

Platforma Android este recunoscută ca fiind prima platformă mobilă completă, ”deschisă” și gratuită (en. complete, open and free).

- **Completă:** Proiectanții au adoptat o abordare cuprinzătoare când au dezvoltat platforma Android. Au început cu un sistem de operare sigur, peste care au constuit un cadru (framework) software robust, care oferă numeroase oportunități de dezvoltare a aplicațiilor.
- **”Deschisă”:** Platforma Android este furnizată prin licență open source. În timpul dezvoltării aplicațiilor, programatorii au acces la caracteristicile telefonului, cum nu au mai avut până în momentul apariției platformei.
- **Gratuită:** aplicațiile pot fi dezvoltate gratuit, datorită faptului că nu există licențe sau drepturi de autor care necesită să fie plătite. De asemenea, nu sunt cerute taxe de membru, de certificare, de testare sau alte tipuri de taxe.

Dezvoltatorii sunt capabili să scrie o aplicație care să înlocuiască oricare aplicație deja existentă sau să aducă îmbunătățiri acestora sau doar informații suplimentare. Acest lucru este posibil datorită faptului că nu există limite pentru accesarea informațiilor particulare. Aceștia sun, practic, nelimitați și pot face absolut tot ceea ce doresc cu Android-ul. Așadar, Android-ul deschide oportunități nelimitate dezvoltatorilor [1] [2].

2.5 Arhitectura sistemului Android

Pentru a înțelege cum funcționează platforma Android, figura 2.5.1 prezintă arhitectura sistemului, care cuprinde straturile (layers) diferite care compun sistemul de operare.

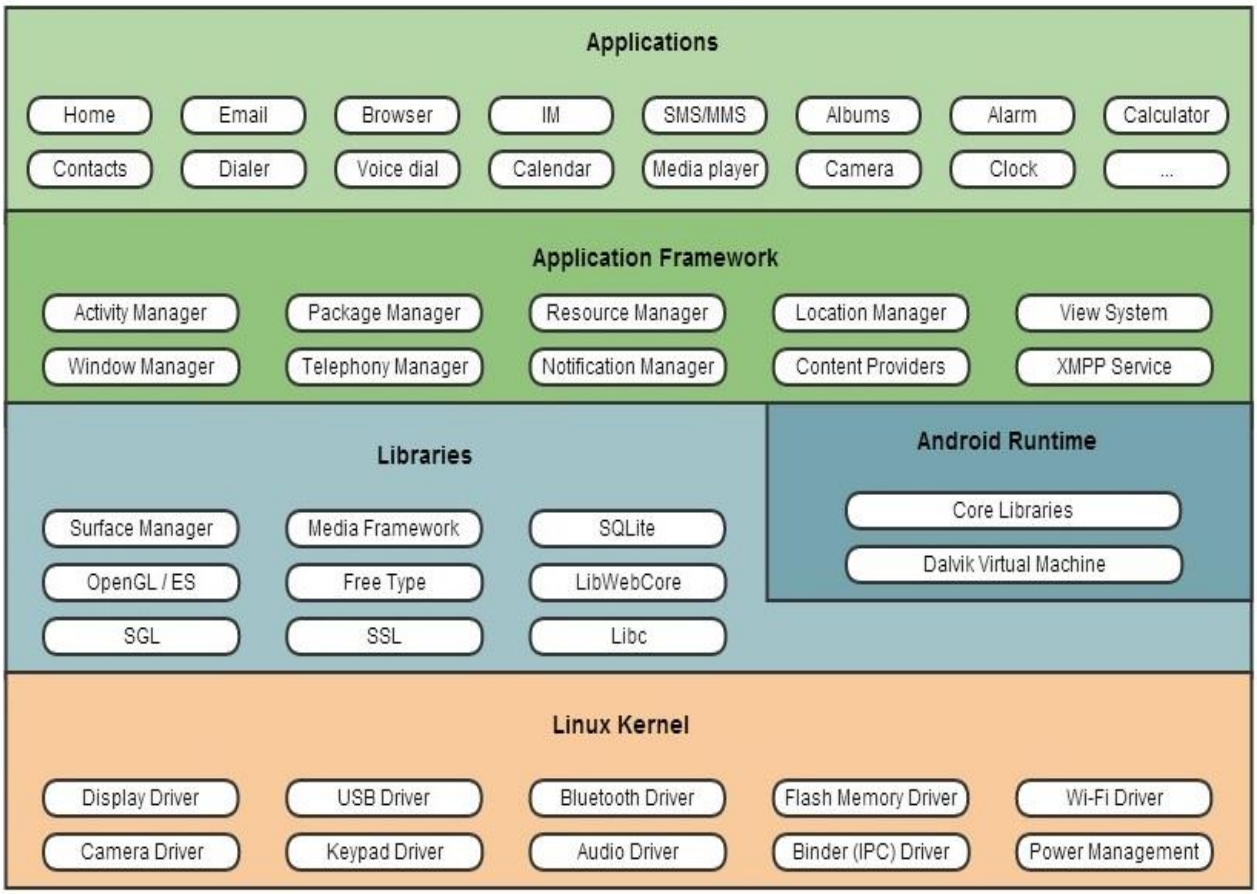


Figura 2.5.1 Arhitectura sistemului Android pe layere

Pentru o mai bună înțelegere, diagrama trebuie parcursă de la bază în sus. Sistemul de operare Android este împărțit în în cinci secțiuni, grupate în patru layere principale:

- **Linux kernel** (nucleul Linux):Android-ul se bazează pe versiunea de Linux 2.6 pentru servicii de bază ale sistemului, precum securitatea, gestionarea memoriei, procesul de management, stiva de rețea și toate driverele pentru dispozitive de la cel mai jos nivel, folosite pentru diferite componente hardware ale echipamentului bazat pe Android. Nucleul se comportă ca un layer de abstractizare între hardware și stiva software.

- **Libraries**: acest layer permite dispozitivului să manipuleze diferite tipuri de date, cuprinzând un set de biblioteci scrise în limbajul C/C++, care sunt folosite de diverse componente ale sistemului Android. Aceste capcaități sunt dezvăluite dezvoltatorilor prin intermediul layer-ului application fraamework. Câteva din bibliotecile de bază sunt:

    - **Surface Manager:** crearea interfeței utilizator (butoane, câmpuri text, etc.) și combină layerele grafice 2D și 3D ale mai multor aplicații;

    - **Media Libraries/Framework:** bazat pe OpenCORE, bibliotecile acordă sprijinul și înregistrarea mai multor tipuri cunoscute de formate media: video, audio (MPEG4, H.264, MP3, AAC), fișiere imagine (JPEG, PNG);

- **SQLite:** motorul bazei de date utilizat pentru a stoca informația utilă din aplicații;
- **SGL:** motorul grafic de bază 3D;
- **SSL:** comunicații în rețea;
- **WebKit:** motorul navigatorului utilizat pentru a afișa conținut HTML;
- **OpenGL:** folosit pentru a reda conținut grafic 2D sau 3D, pe ecran.

- **Android Runtime**: situat pe același layer ca și bibliotecile, Android Runtime furnizează un set de biblioteci pe baza cărora dezvoltatorii pot scrie aplicații Android folosind limbajul de programare Java. Această componentă cuprinde și **mașina virtuală Dalvik**, care permite fiecărei aplicații Android să ruleze în cadrul unui proces propriu. Cealaltă componentă este reprezentată de **Core Java Libraries**, care furnizează majoritatea funcționalităților definite în bibliotecile SE din Java, deși sunt diferite de acestea.

- **Application framework**: acesta este unul din blocurile cu care utilizatorul interacționează direct. Asigură accesul utilizatorilor la diverse capacități ale sistemului pentru a se putea folosi de ele în aplicațiile personale. Unele din cele mai importante blocuri sunt:
  - **Activity Manager:** gestionează ciclul de viață al activității unei aplicații;
  - **Telephony Manager:** gestionează toate apelurile telefonice (voice calls);
  - **Location Manager:** administrează locațiile, foflosind GPS-ul sau alte sisteme;
  - **Resource Manager:** administrează diferite tipuri de resurse folosite în aplicație;
  - **Content Providers:** gestionează schimbul de date din cadrul aplicației.

- **Applications**: etse layer-ul superior din structura arhitecturală a aplicației, în care se găsesc aplicațiile standard care sunt pre-instalate pe fiecare dispozitiv cu Android (Phone, Dialer, Browser, Contacts), dar și aplicații descărcate [2].

2.6 Mașina Virtuală Dalvik vs. Mașina Virtuală Java

O **mașină virtuală** este o abstractizare software a componentelor hardware ce stau la baza aplicației, fiind componentă a layer-ului Android Runtime al sistemului. Mașinile virtuale se pot baza pe specificații ale unui calculator ipotetic sau pot simula arhitectura și funcțiile unui calculator real. O aplicație de sine stătătoare este compilată și rulează pe o mașină virtuală.

2.6.1 Mașina Virtuală Java (JVM)

JVM este un program care permite executarea anumitor programe, și anume al celor care conțin instrucțiuni Java bytecode. În general, mașinile virtuale Java sunt implementate să ruleze pe sisteme de operare existente, dar pot fi realizate astfel încât să ruleze direct pe componente hardware. JVM folosește **arhitectura de stivă**.

Aplicațiile scrise în Java pot fi rulate de pe diferite platforme (Win32, MacOS, Linux, Sun), fără a fi necesară recompilarea lor pentru fiecare platformă în parte. Acest lucru atrage după sine faptul că aplicațiile Java sunt independente de platformă. Implementarea practică a conceptului independenței de platformă este realizată prin folosirea unui calculator virtual pe care rulează, de fapt, aplicațiile acopilate Java.

Pașii parcurși pentru interpretarea aplicațiilor Java: Codul scris reprezintă fișierele sursă ale aplicație (fișiere .java). În urma complării fișierelor sursă Java, nu se va genera cod executabil, ci un cod intermediar, numit cod de octeți (fișiere .class). Pentru a rula un cod executabil, este nevoie de un program special care să interpreteze codul de biți; acest program este mașina virtuală Java, prin încărcarea fișierelor într-un subsistem (class loader) cre se află în permanentă legătură cu motorul de execuție și cu ariile cu datele de rulare [5].

2.6.2 Mașina Virtuală Dalvik (DVM)

În Android, interpretarea fluxului de biți se face prin intermediul mașinii virtuale Dalvik, care, practic, înlocuiește JVM. Dalvik VM este o mașină virtuală optimizată și este proiectată, luând în considerare limitările specifice dispozitivelor mobile (memorie, putere de calcul, energie). DMV folosește **arhitectura de registru**.

Dalvik este mașina virtuală concepută special pentru sistemul de operare Android de la Google. Este software-ul pe care rulează aplicațiile dispozitivelor mobile cu Android (telefoane mobile, tablete sau, mai rcent, pe dispozitive încorporate precum televizoare inteligente sau fluxuri mass-media).

Pașii parcurși pentru interpretarea aplicațiilor Android: codul sursă Java este compilat din fișier .java în fișier .class, cu ajutorul compilatorului Java. Apoi compilatorul dex (dexer) al SDK-ului Android transformă fișierele .class în fișiere cu format specific (.dex = Dalvik Executable files), care conțin cod de biți Dalvik [1] [4].

În figura 2.6.2.1 este reprezentată structura celor doua mașini virtuale, componentele și etapele de evoluție ale codului sursă, până la obținerea unui cod executabil, astfel putându-se observa diferențele enunțate mai sus.

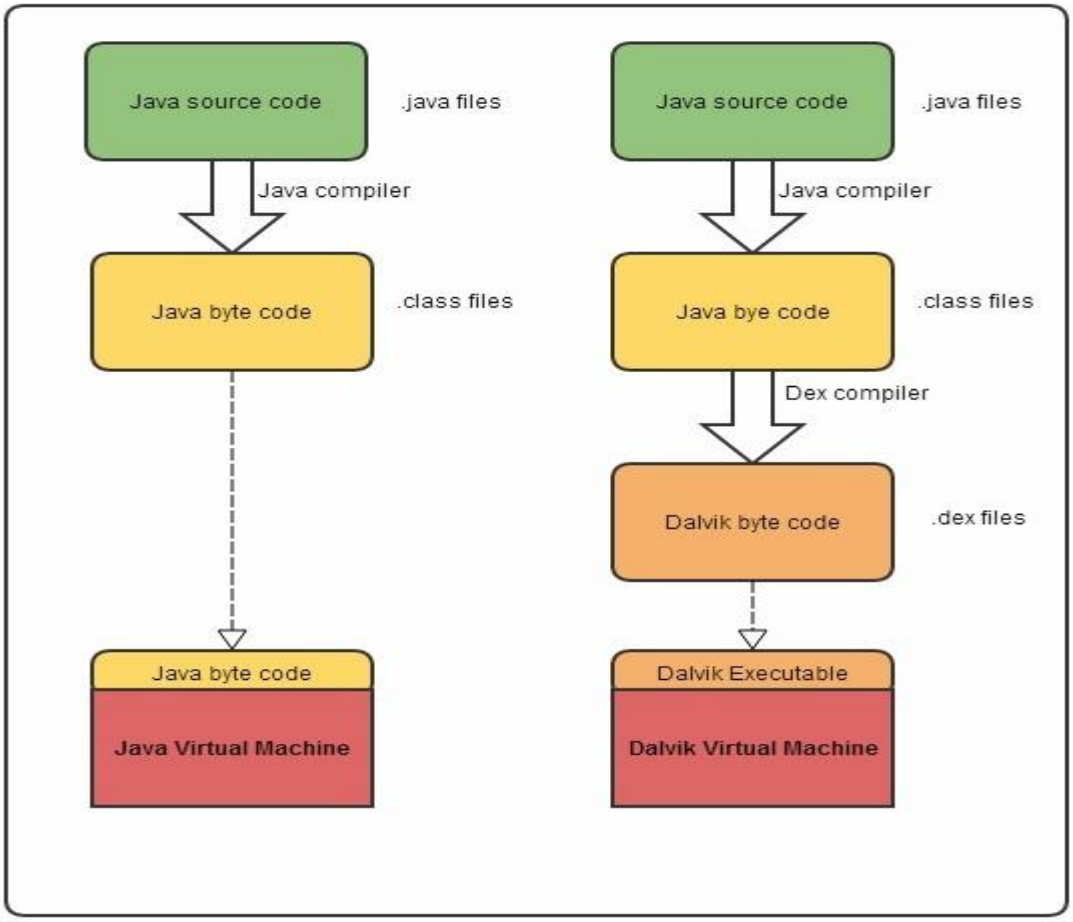


Figura 2.6.2.1 JVM vs. DVM

2.7 Activitățile Android

În Android, o activitate reprezintă interfața grafică prin intermediul căreia utilizatorul interacționează cu aplicația. În general, aplicațiile Android sunt multi-proces (compuse din mai multe activități), iar sistemul de operare Android permite ca mai multe aplicații să ruleze simultan, în funcție de memoria cu care e prevazut dispozitivul și puterea sa de procesare.

2.7.1 Ciclul de viață al unei activități Android

Aplicațiile pot avea procese care rulează în fundal (background), pot fi întrerupte sau oprite când au loc evenimente precum apeluri telefonice. Doar o singuă aplicație activă poate fi vizibilă pentru utilizator, la un moment dat, adică, o singură activitate poate rula în prim plan.

Aplicațiile Android sunt responsabile de gestionarea stării, a memoriei , resurselor și datelor lor. Trebuie să se oprească și să repornească în mod desăvârșit. Înțelegerea diferitelor stări

din ciclul de viață al activității constituie primul pas în proiectarea și dezvoltarea unei aplicații Android. Aceste stări sunt reprezentate grafic în figura 2.7.1.1 [1].

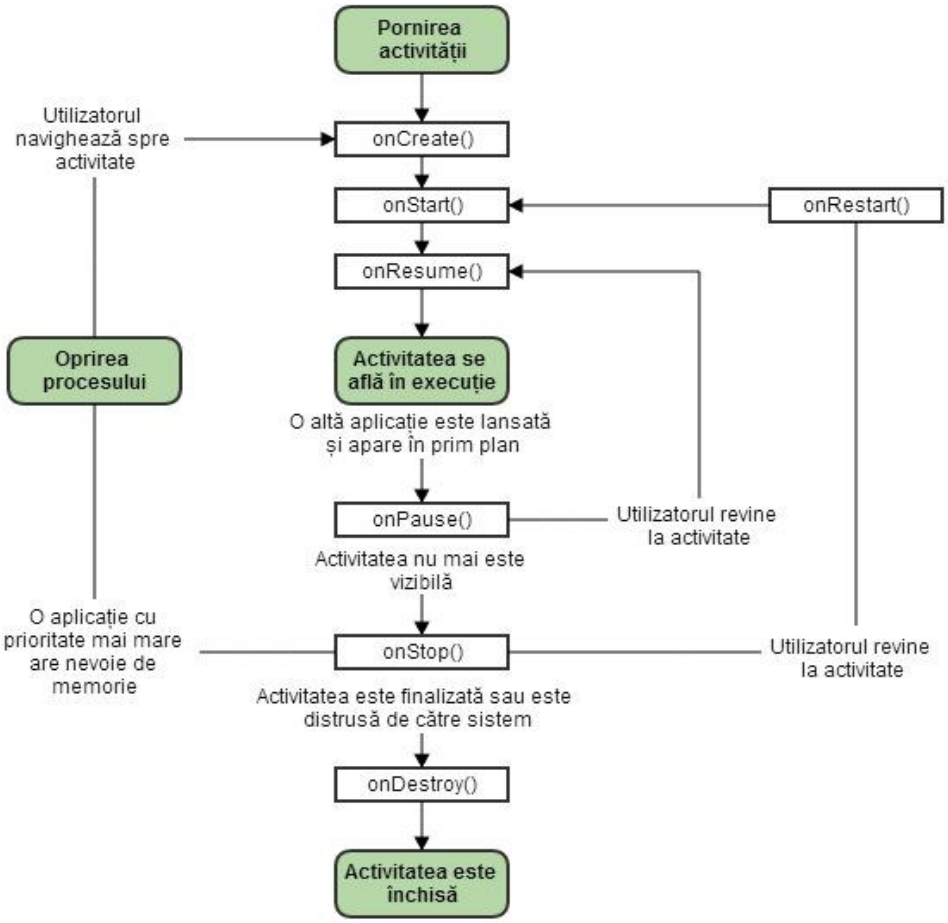


Figura 2.7.1.1 Ciclul de viață al activităților Android

Metodele reprezentate în figură sunt moștenite din clasa Activity și pot fi suprascrise.

- **onCreate()** – este implementată de toate activitățile Android și e apelată atunci când activitatea este creată pentru prima dată; este metoda în care se creează/instanțiază obiectele din aplicație. Obiectul Bundle din semnătura metodei conține starea anterioară a activității, dacă există.
- **onStart()** – este apelată când activitatea devine vizibilă pe ecran.
- **onResume()** - este apelată atunci când activitatea începe să interacționeze cu utilizatorul prin intermediul elementelor grafice.

- **onPause()** – este apelată în momentul în care activitatea curentă intră în starea de așteptare, iar cea anterioară este reluată/continuată.
- **onStop()** – este apelată atunci când activitatea nu mai este vizibilă pe ecran.
- **onDestroy()** – este apelată înainte ca activitatea să fie distrusă de către sistemul de operare.
- **onRestart** – este apelată în momentul când activitatea a fost oprită și apoi reîncepută.

2.8 Android SDK și instrumentele ADT

Aplicațiile sunt scrise, în general, în limbajul de programare Java, folosind trusa de dezvoltare software pentru Android, dar sunt disponibile și alte instrumente de dezvoltare a aplicațiilor. Android **SDK**oferă o varietate de biblioteci API (Application Programming Interface), suportând o mare parte din bibliotecile din Java SE, însă pune la dispoziția programatorilor o platformă nouă (diferită de Swing și AWT), pentru definirea interfețelor utilizator. Aceste biblioteci oferă suport pentru:

- Crearea interfeței utilizator (butoane, meniuri, câmpuri text)
- Baze de date (SQLite)
- Fișiere audio, video, imagini (MPEG4, MP3, AAC, JPG, PNG, etc.)
- Conectivitate Bluetooth, Wi-Fi, GSM/EDGE, WiMAX, etc.
- Navigare pe internet (WebKit)
- Acces la componente hardware (senzori, GPS, cameră video)
- Reprezentări grafice 2D și 3D (OpenGL)
- Mesagerie SMS și MMS
- Comunicații în rețea (SSL)

Instrumentele de dezvoltare Andorid **ADT (Android Development Toolkit)**, pentru Eclipse, oferă un mediu de dezvoltare cu grad profesional, pentru constrirea aplicațiilor în Androd. Este un mediu de dezvoltare Java integrat (IDE = Integrated Development Environment) cu caracteristici avansate, care ajută la construirea, testarea, depanarea și împachetarea aplicațiilor Android. Pachetul ADT Bundle livrează tot ceea ce este necesar pentru a începe construirea aplicațiilor, inclusiv o versiune a mediului ADT, pentru eficientizarea dezvoltarea aplicațiilor [3].

2.9 Dispozitive cu sistem de operare Android

Dispozitivele pe care rulează sistemul de operare Android au apărut în toate formele și dimensiunile. Cele mai cunoscute și răspândite astfel de dispozitive sunt telefoanele inteligente (smatphone). Pe lângă acestea, pe piață mai există și alte modele, precum: tablete, dispozitive e-reader, aparat pentru accesarea fișierelor cu extensia MP4 (MP4 playere), laptopuri tip netbook, ceas inteligent (smartwatch), calculatoare de bord pentru mașini, console de jocuri și altele.

Pe baza studiilor, cercetărilor și observațiilor realizate în domeniu, se preconizează, în viitor, o creștere însemnată a numărului și diversității dispozitivelor care funcționează cu sistemul de operarea Android, până la a depăși numarul dispozitivelor cu sistem de operare Windows, care în momentul de față sunt cele mai răspândite, dar și mai mare decât numărul dispozitivelor cu iOS, principalul rival de pe piață al Android-ului.

Figura 2.9.1 Concurența acerbă Android – Windows - iOS

Capitolul 3. ANALIZĂ ȘI FUNDAMENTARE TEORETICĂ

3.1 Concepte generale

În ultimul timp, numărul telefoanelor inteligente și al dispozitivelor mobile, dezvoltate pe baza unuo sisteme de operare, a crescut semnificativ, preconizându-se o cerere din ce în ce mai mare pentru acest tip de echipamente. Printre cele mai răspândite astfel de dispozitive, un loc fruntaș îl ocupă telefoanele inteligente, motiv pentru care se pune tot mai mare accent pe dezvoltarea aplicațiilor accesibile pentru smartphone. Acestea sunt menite să aducă o serie de avantaje și să ușureze munca pe care oamenii, înainte, erau nevoiți să o realizeze prin metode clasice și poate chiar rudimentare. Datorită disponibilității rețelelor mobile (2G, 3G, 4G, HSPA+), utilizatorii dispozitivelor mobile au acces la Internet. În momentul de față este posibil pentru aceștia să efectueze diferite operații și activități, prin simpla descărcare, gratuită sau nu, a aplicațiilor diponibile în rețea. În cazul dispozitivelor cu sistem de operare Android, majoritatea aplicațiilor sunt disponibile gratuit

Există o mare diversitate de aplicații proiectate pentru telefoanele inteligente, realizate pe baza sistemului GPS încorporat. Orice telefon inteligent, dotat cu sitem de operare mobil și un modul GPS, poate fi folosit și ca un veritabil GPS, în sine. Împreună cu servicii utile precum Google Maps, funcționalitatea GPS-ului convinge foarte mulți șoferi, și chiar pietoni, să utilizeze propriul telefon pe post de GPS, în detrimentul sistemului original GPS. Fie că sunt simple aplicații de orientare, de alegere a unei rute ocolitoare, de găsire a celui mai scurt traseu etc., aceste sisteme software sunt foarte des utilizate, motiv pentru care sunt și ușor accesibile. Dintre sistemele existente, câteva exemple sunt prezentate, pe scurt, în cele ce urmează.

CoPilot GPS, disponibilă pentru dispozitive cu sisteme de operare Android și iOS, este o aplicație care oferă unelte de navigare pe bază de hărți, fiind posibilă vizualizarea unor rute principale, a unor rute alternative și a unei liste cu principalele obiective turistice ale marilor orașe. Această aplicație funcționează în zone în care nu există conexiune la Internet, nefiind nevoie de conectare prin Wi-Fi la rețeaua globală [6].

Aplicația **Waze**, disponibilă pe orice dispozitiv Android, dar nu și pe orice dispozitiv iOS, citește locația utilizatorului, cere introducerea unei destinații, calculând, ulterior ruta dintre cele două locații, precizând informații depre eventuale zone problematice (zone aglomerate sau în care au avut loc coliziuni sau sunt create blocaje datorate unor diverși factori externi) [7].

HotSpot, disponibilă în Android și iOS, este o aplicație concepută, pentru pietoni, în special turiști. Aceasta preia coordonatele utilizatorului, care introduce destnația dorită, după care calculează rutele cele mai scurte, oferind informații legate de acestea, dar și despre posibilitățile

de transport în comun, ajutându-i pe cei mai puțin coscători să ajungă rapid și ușor la destinație [8].

Există o aplicație care atenționează șoferii în legătură cu zonele riscante, în special pentru vitezomani, în ceea ce privește prezența mașinii de poliție , a radarelor, date despre zonele cu camere foto și videp instalate în scopul monitorizării traficului, dar și informații suplimentare legate de accidente sau alte situații de creare a ambuteiajelor în trafic. Această aplicație se numește **Trapster** și este disponibilă atât pe dispozitive cu Android, cât și pentru cele cu iOS [9].

Aplicația **Where2Boss**, disponibilă doar pentru dispozitive cu iOS, a fost dezvoltată, în primul rând, pentru turiști, având ca scop determinarea unor rute cu care e comparată cea propusă de taximetrist. Astfel, utilizatorul va ști dacă este o posibilă victimă a unei capcane a șoferului de taxi, putând evita, din timp, acest aspect neplăcut. Această aplicație permite și compararea timpului în care clientul a ajuns la destinație cu taxiul și timpul în care ar fi ajuns dacă alegea un mijloc de transport în comun [10].

Se acordă o atenție deosebită aplicațiilor pentru comanda taxiurilor prin intermediul telefoanelor inteligente, deoarece se constată o îmbunătățire a economisirii timpului și a banilor și sunt mai eficiente decât metodele clasice de rezervare a taxiurilor, prin efectuarea apelurilor telefonice dispeceratelor companiilor de taximetrie. Cheia evoluției este reprezentată de coordonatele GPS, întrucât, aproape fiecare telefon inteligent și dispozitiv mobil cu sistem de operare este dotat cu sistem GPS. Acesta permite dispecerului să cunoască locația exactă a clientului care efectuează comanda, făcând posibilă determinarea cu ușurină a celui mai apropiat taxi de locația respectivă.

Câteva sisteme existente, de comandă a taxiurilor cu ajutorul telefonului inteligent sunt amintite în continuare, unele fiind funcționale în străinătate, iar unele putând fi utilizate și în țară. Fiecare aplicație în parte are caracteristici proprii, care le diferențiază de celelalte, dar în fond, toate au la bază același principiu de funcționare.

Aplicația **Taxi Magic**, disponibilă în New York și alte state din Statele Unite ale Americii, permite utilizatorilor să găsească cel mai apropiat taxi, raportat la poziția lor geografică, să comande taxiul parcurgând câțiva pași , pot urmări taxiul până când ajunge la locația clientului, iar, în final, costul călătoriei este achitat automat de cartea de credit a clientului, aplicația dispunând de datele necesare ale clientului pentru efectuarea operațiunilor necesare [11].

Aplicația **mytaxi-The Taxi App**, disponibilă în peste patruzeci de țări, din S.U.A. și câteva țări europene (Germania, Spania, Polonia, Austria, Elveția), localizează clientul în momentul depunerii comenzii pentru taxi, cere, din paretea clientului, o confirmare a locației curente și a destinației, dupăcare confirmă comanda [12].

Cabbie – Taxi Cab Booking, disponibilă în Singapore, este o aplicație care permite determinarea poziției exacte a clientului care efectuează comanda, folosind coordonatele GPS furnizate de sistemul GPS încorporat în dispozitivul mobil de pe care s-a realizat comanda. De asemenea, poate determina o poziție aproximativă a clientului, potrivit locației unei surse de rețea mobilă. Această aplicație poate memora adresele des utilizate de către un anumit client, pentru a face posibilă trimiterea ulterioară a unui simplu mesaj, pentru adresele deja salvate [13].

StarTaxi este o aplicație disponibilă în România, care detectează în mod automat locația utilizatorului care trimite comanda de taxi, după ce a confirmat adresa destinației. După determinarea celui mai apropiat taxi, comanda este confirmată, iar în timpul cursei, clientul beneficiază de acces gratuit la Internet, putând lăsa comentarii depsre impresiile sale referitoare la condițiile de călătorie și poate citi comentarile lăsate de alți clenți. În final, clientul poate plăti cursa cu bani sau prin intermediul aplicației, diorect din contul bancar [14].

O altă aplicație disponibilă în țara noastră, este **Clever Taxi**, care, la fel ca și cealaltă, citește locația utilizatorului care trimite cerer e de taxi, și caută cel mai apropiat taxi disponibil. Oferă posibilitatea alegerii între mai multe taxiuri, ale diferitelor companii, putând transmite mai multe comenzi. Taxiul asignat poate fi monitorizat de către client, respectiv, utilizatorul poate vizualiza timpul rămas până în momentul în care vehiculul ajunge la locația sa. De asemenea, este posibilă salvarea unui istoric al comenzilor și, în final, poate fi acordată o notă șoferului, pentru a exprima în mod direct aprecierea față de sistemul de rezervare al taxiurilor [15].

În lucrarea de față este prezentată o aplicație software, menită să vină în ajutorul persoanelor care se bucură de utilitatea unui telefon inteligent cu sistemul de operare Android și care se află în drum spre anumite destinații. În cazul unei constrângeri temporale a deplasării până la destinație, individul poate alege să rezerve un taxi, prin intermediul unei interfețe prietenoase și ușor de utilizat, metodă care îmbunătățește eficiența și eficacitatea în rândul companiilor de taxi, în ceea ce privește cheltuielile pricinuite de o slabă administrare a resurselor și timpul deplasării vehiculelor între comenzi. Acestă metodă aduce o serie de beneficii și în cazul clienților/utilizatorilor, referitor la timpul necesar așteptării taxiului alocat pentru onorarea comenzii, datorită faptului că aplicația este capabilă să calculeze distanțele între locația clientului și locația mai multor taxiuri disponibile, bazându-se pe coordonatele GPS ale acestora și determinând, astfel, cel mai apropiat taxi, care va fi alocat ulterior clientului.

Concepută pe același principiu cu sistemele existente, aplicația prezentată în această lucrare permite utilizatorilor crearea unei comenzi de taxi pentru a ajunge la o anumită destinație, asigurând alocarea celui mai apropiat vehicul, în cel mai scurt timp. Fiecare sistem amintit are, pe lângă trăsăturile comuni de pricipiu, cel puțin câte o caracteristică prin care se afirmă în mod deosebit față de celelalte. În cazul modelului de față, clientului îi este permisă monitorizarea prin trafic a taxiului alocat, respctiv, afișarea timpul rămas până în momentul sosirii la locația curentă

a utilizatorului, precizând și costul estimat al călătoriei clientului, raportat la distanța din poziția inițială de la care este preluat clientul, până la locația destinației.

3.2 Tehnologii utilizate

Pentru realizarea sistemului descris în prezenta lucrare de diplomă s-a folosit o serie de tehnologii care, prin îmbinarea armonioasă, au dus la construirea funcționalității aplicației. Sunt folosite câteva tehnologii de programare, bazate pe POO (Programare Orientată pe Obiecte), modelul clinet-server cu comunicația realizată prin socket-uri (clienți Android și server Java). Tehnologia GPS este folosită pe partea de clienți (atât utilizatori, cât și taxiuri), care au acces la Internet prin serviciile GPRS și Wi-Fi. Serverul central al aplicației este conectat permanent la o bază de date MySQL, prin interfața JDBC. În figura 3.2.1 este ilustrată relația stabilită între aceste tehnologii pe parcursul dezvoltării aplicației.

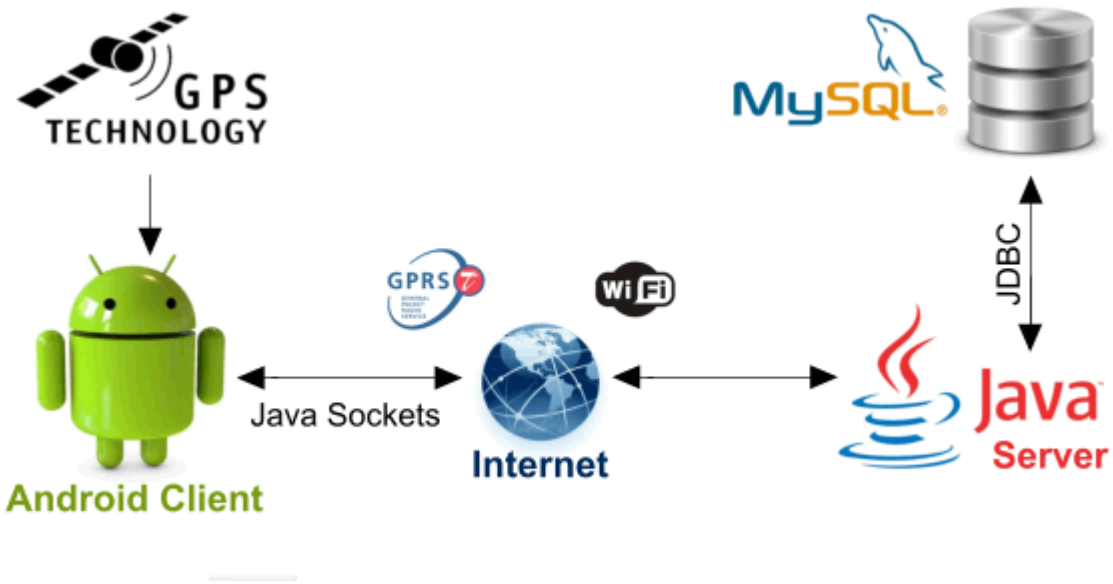


Figura 3.2.1 Tehnologii de implementare

3.2.1 Tehnologii de programare

3.2.1.1 Noțuni de programare Orientată pe Obiecte

De-a lungul timpului, tehnicile de programare au evoluat, trecând prin mai multe etape, cunoscute sub diferite nume: **programare nestructurată** (program simplu în care sunt folosite doar variabile globale, întâmpinând dificultăți în cazul unor aplicații mai ample, cu date numeroase), **programarea procedurală** (program principal structurat pe bază de subprograme cu parametri formali, variable locale și apel de subprograme cu parametri efectivi, probleme

apărând în momentul în care doi sau mai mulți programatori sunt asignați la program, dar nu pot accesa deodată un anumit fișier), **programare modulară** (program principal caracterizat de mai multe module, formate prin gruparea unor subprograme cu funcționalități similare; modulele sunt implementate și depanate separat – independență și încapsulare), **programare orientată obiect**.

Programarea orientată pe obiecte se bazează pe programe cu tipuri noi de date, cât și de metode asociate creării, prelucrării și distrugerii datelor respective. Principiile fundamentale care stau la baza abordării Orientate Obiect sunt **abstractizarea** și **încapsularea**.

Abstractizarea permite considerarea unor idei complexe, ignorând anumite detalii irelevante care ar putea aduce confuzii; programele devin ansamble de obiecte caracterizate de proprietăți și comportament specific cu ajutorul carora interacționează în cadrul programului. Un tip de date abstract (ADT) este o entitate caracterizată printr-o structură de date și un ansamblu de operații aplicabile acestor date. Tipul de date obiectual este un tip de date care implementează tipul de date abstract. Abstractizarea este exemplificată în figura 3.2.2.1.1, unde clasa abstractă FormeGeometrice conține, pe lângă metode implementate, metodele abstracte calculArie() și afișare(), neimplemenetate, care ulterior vor fi implementate cu comportament concret, specific fiecărei clase care extinde clasa abstractă (Triunghi, Cerc, Dreptunghi).

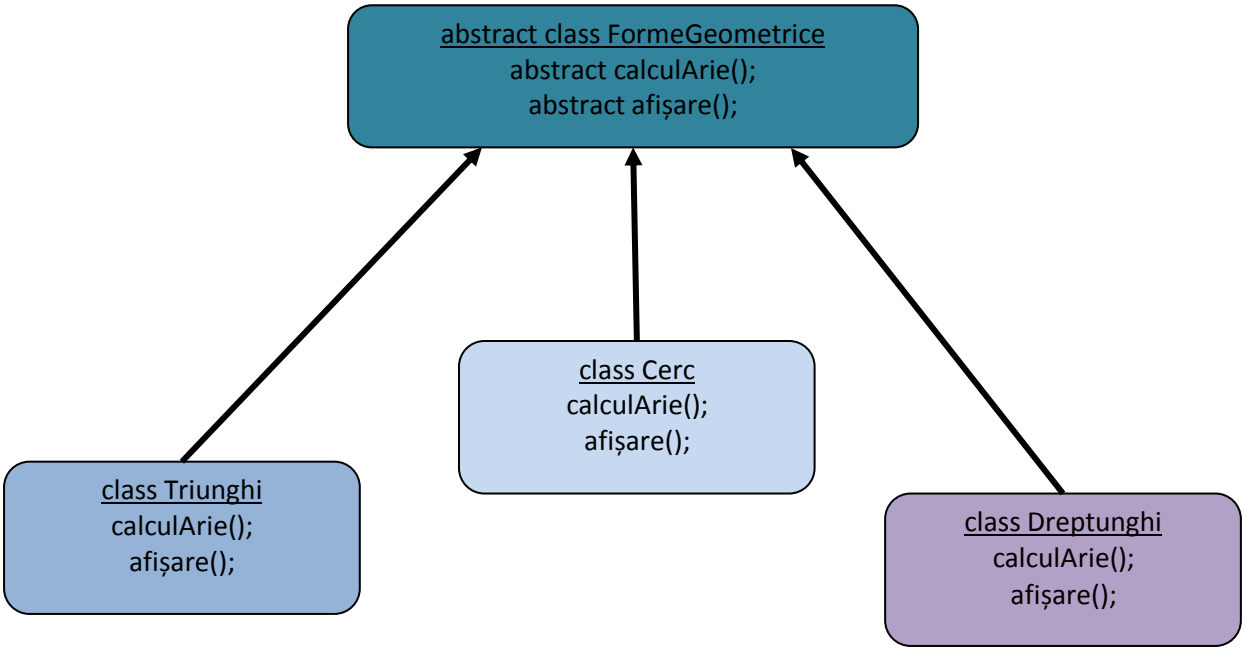


Figura 3.2.1.1.1 Exemplu de abstractizare în POO

Încapsularea permite focalizarea atenției asupra a lucrului care se face, fără a se lua în considerare complexitatea felului în care acel lucru se face. Se recomandă definirea completă și corectăa datelor și operațiilor, precum și denumirea sugestivă a acestora, pentru a ușura interacțiunea utilizatorului cu programul, acetsa nefiind obligat să țină minte detalii legate de implementare. Datele vor fi accesate de el prin intermediul proprietăților și vor fi efectuate operațiile specifice prin intermediul metodelor de care dispune tipul de date obiectual definit.

Pe lângă abstractizare și încapsulare, programarea orientată pe obiecte mai are la baze alte două principii fundamentale: **moștenirea** (extinderea sau specializarea unei clase existente prin definirea unei clase noi care moștenește stările și comportamentul clasei de bază) și **polimorfismul** (posibilitatea mai multor obiecte aparținând unei clase dintr-o ierarhie de clase de a folosi același nume pentru metode care manoifestă un comportament diferit, prin redefinirea lor).

La baza Programării Orientată-Obiect stau entități precum **clasele** și **obiectele**. Clasele sunt modele software care descriu proprietâți generale ale unor entități cu care lucrează sistemul software. Obiectele reprezintă instanțe ale clasei de obiecte în care se încadrează, o clasă fiind, de fapt, un model, o schiță după care sunt create obiectele. Acestea din urmă sunt caracterizate de stare și comportament (starea este stocată în câmpuri, iar comportamentul este exprimat prin metode).

Principalele beneficii aduse de POO, în plus față de celelalte tipuri de programări, sunt reprezentate de o mai bună abstractizare (modelarea simultană a informației și comportamentului), o mentenanță mai bună (claritate mai ridicată, fragilitate scăzută), reutilizare mai bună (clasele considerate ca fiind componente încapsulate) [16] [17] [18] [19].

3.2.1.2 Limbajul Java

Asemenea limbajului uman, Java reprezintă o cale de a exprima anumite concepte. Java este un limbaj de programare de uz general, bazat pe clase și orientat pe obiecte, proiectat astfel încât sa aibă cât mai puține dependențe posibile, legate de implementare. Limbajul este astfel conceput, înât să permintă dezvoltatorilor să realizeze aplicații pe o anumită platformă, care apoi să poată rula și pe alte platforme (Linux, Win32, Sun, MacOS), fără a fi necesară recompilarea pentru fiecare în parte.

Un program, scris în limbajul de programare Java, este compilat și rulează pe o mașină virtuală Java (JVM), lucru care face posibilă independența de platformă a aplicațiilor Java. Această mașină virtuală interpretează codul de octeți (fișiere .class) obținut în urma compilării fișierelor sursă (fișiere .java) [5].

3.2.1.3 Modelul client-server

Ideea de bază a aunui sistem client – server este existența unui depozit central de informații (date stocate, de obicei, în baze de date), care se doresc a fi distribuite la cerere unor clienți sau mașini. Cheia conceptului client – server este că depozitul de informații este localizat central, asfel încât eventualele modificări aduse acestor informații să fie propagate și clienților.

Astfel, în modelul client – server, serverul este reprezentat de ansamblul format din depozitul de informații, software-ul care distribuie informațiile și sistemul în care acestea sunt stocate. Clientul este reprezentat de programul software aflat pe mașina consumatorului, care comunică cu serverul, extrage informații, le prelucrează și le afișează pe stația utilizatorului [5] [20].

3.2.1.4 Programarea în rețea prin socket-uri

Calculatoarele conectate într-o rețea comunică între ele, la baza niveleului Transport de comunicare în rețea, prin intermediul protocoalelor de comunicație **TCP (Transport Control Protocol)** și **UDP (User Diagram Protocol)**, după cum se poate observa și în reprezentarea din tabelul 3.2.1.4.1.

Tabel 3.2.1.4.1 Nivele de comunicare în rețea

Nivelul Aplicație HTTP, FTP, DNS, Telnet, SMTP
Nivelul Transport TCP, UDP
Nivelul Rețea (Internet) IP
Nivelul Acces la Rețea Conectica, semnale, drivere de rețea

Protocolul TCP oferă un canal de comunicare punct-la-punct, de încredere, pe care aplicațiile client-server, bazate pe Internet, le utilizează pentru a realiza comunicarea între entități (client și server). Este un protocol orientat pe conexiune, de nivel scăzut, dar care asigură transmiterea unui flux de octeți, fără erori, de pe o mașină pe altă mașină din rețea. Fluxul de biți este fragmentat în mesaje discrete, iar fiecare mesaj este păstrat la baza nivelului Internet.

Protocolul UDP utilizează un model de transmisie simplu și face posibilă livrarea mesajelor într-o rețea. Acesta este un protocol de nivel înalt, nesgiur, fără conexiuni, care nu garantează sosirea datelor la destinație, deoarece nu dispune de mecanisme de confirmare.

Ordinea de sosire a datagramelor la destinație este, de asemenea, nesigură, aceasta putând să difere de cea în care au fost transmise. Viteza de transmisie a mesajelor este, însă, mult mai mare decât cea a protocolului TCP.

Adresele IP, în cazul sistemului de operare Windows, pot fi alocate manual, dinamic, de către serverele DHCP (Dynamic Host Configuration Protocol) sau automat, prin mecanismul APIPA (Automatic Private IP Addressing).

Un **soclu (socket)** reprezintă un punct de conexiune într-o rețea TCP/IP, un capăt de comunicație, o formă de comunicare bidirecțională între două procese (IPC - Inter Process Communication) care rulează pe aceeași mașină sau pe mașini diferite conectate printr-o rețea. Pentru a realiza comunicarea, ambele procese/programe trebuie să lege câte un socket la capătul conexiunii. O pereche de socket-uri conectate realizează o interfață de comunicare între două procese. Porturile (port sursă și port destinație) permit identificarea unică a unui punct de acces la servicii (SAP – Service Access Point), de nivel transport [21].

Principalele operații realizate pde socket-uri sunt:

- Conectare la un alt socket
- Trimitere date
- Recepție date
- Acceptare conexiuni
- Închidere conexiuni

Realizarea comunicării în rețea prin intermediul scolurilor (socket-urilor), pentru o aplicație de tip client-server, este ilustrată în figura 3.2.1.4.1.

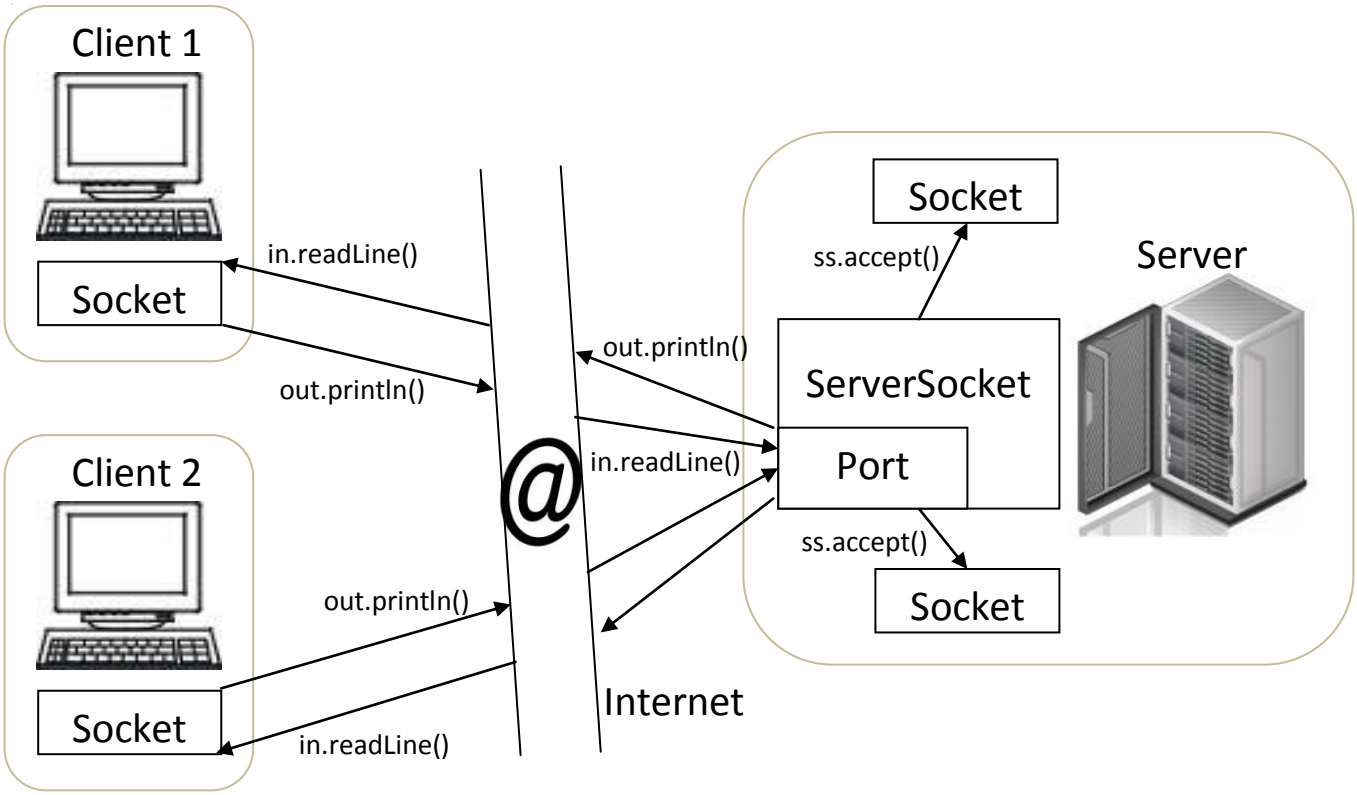


Figura 3.2.1.4.1 Comunicarea în rețea prin socket-uri

3.2.2 Tehnologia GPS

3.2.2.1 Sisteme de navigație bazate pe sateliți

Sistemele GNSS (**Global Navigation Satellite Systems**), cunoscute și sub numele de Sisteme de poziționare globală prin satelit, sunt recunoscute ca fiind elemente cheie în domeniul **comunicațiilor**, **navigației**, **orientării**, **dirijării traficului terestru**, **supravegherii și administrării traficului aerian**, precum și un fundament pe baza căruia pot fi oferite servicii îmbunătățite de **navigație aero-nautică**. Acestea reprezintă, de fapt, sistemele de navigație care utilizează tehnica poziționării prin intermediul sateliților, având scopul de a furniza informații precise legate de navigație (poziție momentană și timp) a receptorului unui utilizator, aflat în mișcare sau repaus.

Sistemele de navigație bazate pe sateliți folosesc tehnica **triangulației** pentru localizarea unui obiect, oriunde pe glob, prin intermediul unor calcule realizate pe baza unor informații provenite simultan de la cel puțin trei sateliți, ducând la obținerea unor coordonate, într-un anumit sistem de referință. Fiecare satelit tarnsmite semnale codate la intervale prestabilite de timp. Receptorul convertește informația recepționată în poziție, viteză și estimări de timp. Pe baza acestor informații, orice receptor de pe suprafața sau din apropierea suprafeței Pământului

poate calcula cu exactitate poziția satelitului de la care a primit semnalul, precum și distanța dintre ei, pe baza întârzierii transmisiunii. Coordonarea datelor actuale, recepționate de la trei sau mai mulți sateliți, permit receptorului să își determine propria poziție [22] [23].

Există două sisteme GNSS care, momentan, sunt complet operaționale la nivel mondial: Sistemul de Poziționare Globală al Statelor Unite (GPS = Global Positioning System) și Sistemul Global de Navigație bazat pe Sateliți Orbitali al Rusiei (GLONASS = Global Orbiting Navigation Satellite System). Alte două sisteme de acest tip, care se află în curs de dezvoltare și extindere teritorială, sunt sistemul de poziționare Galileo, al Uniunii Europene și sistemul de navigație Compass (BeiDou 1), dezvoltat de China, programate să devină complet funcționale în jurul anului 2020.

Potrivit Corporației Aerospațiale, tehnologia GNSS este descrisă de trei segmente, care îi intră în componență: segmentul spațial, segmentul de control și segmentul utilizator.

Segmentul spațial este format dintr-o constelație de sateliți, în general 24 sateliți orbitali (câte opt, organizați pe trei orbite circulare, ulterior structurate în șase orbite, fiecare având câte patru sateliți). Orbitele își au centrul în interiorul Pământului, dar nu se rotesc odată cu mișcările planetei, ci rămân fixe, astfel încât să respecte un aliniament corect, adică toți cei patru sateliți să fie vizibili pentru un anumit punct timp de câteva ore, în fiecare zi. Fiecare satelit din constelație emite semnale de radiofrecvență (RF), modulate prin coduri și mesaje de navigație.

Segmentul de control are în structură patru componente: un post central de comandă, un post alternativ de control, patru antene dedicate la sol și cinci stații de monitorizare dedicate, aflate pe Pamânt, care gestionează buna funcționare a sateliților, prin supravegherea lor și actualizarea mesajelor de navigație, calculând pozițiile spațio-temporale ale acetora.

Segmentul utilizator este format din receptoarele de radionavigație GPS ale comunității utilizatorilor: numerosi utilizatori militari, deopotrivă cu utiliaztoricivili, comerciali și științifici. Receptoarele sunt specializate în recepția decodarea și procesarea codurilor și a mesajelor de navigație, convertind semnalele vehiculelor spațiale (SV) în poziție, viteză și estimări de timp. Aceste receptoaqre sunt alcătuite din mai multe componente, printre cele mai importante fiind: sistemul de alimentare cu energie, antena cu amplificatorul de semnal, microprocesorul, oscilatorul de înaltă frecvență, unitatea de control și memoria pentru stocarea datelor. În funcție de calitățile receptorului și ale antenei, este determinată acuratețea preciziei de poziționare sau a elementelor de navigație [24] [25] [26] [27].

3.2.2.2 Ce este GPS-ul?

Sistemul de Poziționare Globală (en. **Global Positioning System = GPS**) a fost proiectat, exploatat și dezvoltat de Departamentul Apărării al Statelor Unite ale Americii (eng. US D.O.D. = United States' Department of Defense). Acesta este un sistem mondial de

radionavigație format dintr-o rețea de 24, până la 32 de sateliți și stațiile aflate pe suprafața Pamântului. Sateliții se deplasează cu o viteză de aproximativ 3.9 km/s și sunt plasați pe orbite geostaționare ale Pământului, aflate la distanțe de aproximativ 20180 km de suprafață, care transmit semnale sub formă de microunde. După recepționarea, prelucrarea și interpretarea semnalelor, acest sistem este capabil să furnizeze informații precise legate de timpul și poziția geografică (latitudine, longitudine, altitudine) a unui obiect, indiferent de condițiile meteorologice, oriunde pe sau în apropierea Pământului, unde există o linie sau un orizont liber de vedere a trei sau mai mulți sateliți GPS.

Sistemul GPS, cunoscut și ca NAVSTAR (NAVigation System with Timing And Ranging), a fost propus de către Departamentul american al apărării, încă de la inceputul anului 1970. Acesta a fost destinat aplicațiilor militare și a avut drept scop central furnizarea poziției unei persoane, aflate în orice punct geografic de pe suprafața Terrei [28] [29].

Fiind cel mai cunoscut și utilizat sistem de navigație bazat pe sateliți (GNSS), sistemul GPS moștenește numeroase trăsături și atribuții ale acestora, dezvoltând și alte însușiri noi, caracteristice doar sistemelor GPS, însușiri prin care se diferențiază de celelalte sisteme GNSS. Așadar, un sistem GPS este caracterizat de cele trei segmente: **segmentul spațial** (format din constelația de 24-32 sateliți orbitali care emit semnale RF), **segmentul de control** (care constă dintr-o rețea globală de stații specializate - în general cinci – care sunt dispuse aproximativ uniform de-a lungul zonei ecuatoriale, avand rolul de a urmărisateliții GPS, de a monitoriza transmisiile lor, de a efectua analize, trimite acestora comenzi și date) și **segmentul utilizator** (format din totalitatea utilizatorilor care dețin receptoare GPS, cărora li se calculează poziția) [24] [25].

GPS-ul a fost inițial destinat aplicațiilor militare, dar în anii 1980, guvernul a facut sistemul disponibil pentru a putea fi folosit și în scop civil. Din motive strategice, Segmentul de Control a fost responsabil cu bruiajul semnalelor GPS (politica DS – Disponibilitate Selectivă) pentru a scădea precizia sistemelor de poziționare folosite în scopuri civile. În anul 2000 restricțiile DS au fost ridicate, iar precizia poziționării a ajuns pentru prima dată sub limita de 100m [30].

Sistemul GPS crește productivitatea în multe ramuri ale economiei, în agricultură, construcții, minerit, topografie, sisteme de livrare a pachetelor, precum și managementul logistic al lanțurilor de aprovizionare. Mari rețele de comunicații, sisteme bancare, piețe financiare, precum și rețele electrice depind în mare măsură de GPS, pentru sincronizare în timp real. Unele servicii wireless nu pot funcționa fără acest tip de sistem.

În prezent, sistemul GPS rămâne esențial pentru securitatea națională a S.U.A., iar aplicațiile sale sunt incluse în aproape fiecare aspect al operațiunilor militare. Aproape toate bunurile militare noi, de la vehicule la muniții, sunt echipate cu GPS [31].

3.2.2.3 Determinarea poziției prin metoda triangulației

Semnalele transmise de GPS sunt codate și recepționate simultan de la cel puțin 3 sateliți, cei mai „vizibili”, pe principiul trinagulației, specific sistemelor de navigație bazate pe sateliți, pentru poziționarea într-un sistem de coordonate X, Y, Z (corespunzătoare latitudinii, longitudinii și altitudinii), concomitent cu datele „clock”, reprezentând timpul unic al rețelei de sateliți. Principiul triangulației este prezentat în figura 3.2.2.3.1.

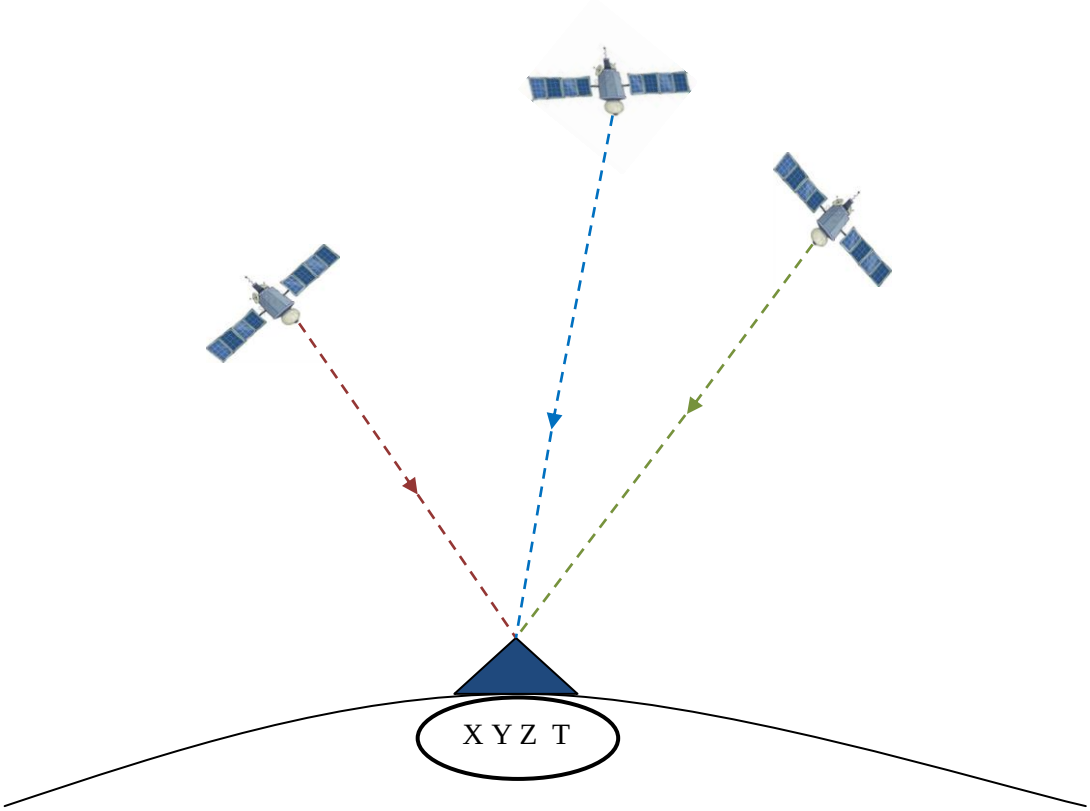


Figura 3.2.2.3.1 GPS – principiul triangulației
Poziția celor 3 sateliți și sincronizarea cu timpul universal (UTC) sunt necesare pentru determinarea coordonatelor X, Y, Z și T.

Acest principiu se bazează pe estimarea cât mai precisă a distanței dintre receptorul GPS și cei trei sateliți. Pentru a putea calcula o locație de pe glob, mai este necesară cunoașterea coordonatelor exacte ale celor trei puncte de referință din spațiu.

Distanța (d) dintre un satelit și un receptor de pe Pământ se calculează cu formula (1):

$$d = \Delta t \cdot c \qquad (1)$$

unde Δt reprezintă întârzierea semnalului recepționat de la satelit, iar c reprezintă viteza luminii.

3.2.2.4 Formatul mesajelor GPS

Sateliții GPS transmit două tipuri de semnale radio de putere redusă, și anume L1 și L2. Sistemul GPS civil utilizează frecvența L1 de 1575,42 MHz, în bandă UHF (Ultra High Frequency). Semnalele transmise sunt capabile să treacă prin nori, sticlă și plastic, dar nu vor putea trece prin majoritatea corpurilor solide, cum ar fi clădiri sau munți.

Fiecare satelit transmite, în mod continuu, spre Pământ, un mesaj de navigație, cu viteza de 50 b/s, care cuprinde trei biți de informație diferiți: un **cod pseudoaleator** (identifică satelitul care transmite informația), **date ephemeris** (transmit informații importante despre starea satelitului, timpul sistemului și valorile de corecție ale ceasului) și **date almanac** (indică poziția fiecărui satelit GPS și a celorlalți sateliți care fac parte din sistem, la un moment dat).

Mesajul de navigație este necesar pentru determinarea întârzierii semnalului recepționat și a poziției exacte a unui satelit în spațiu. Un întreg mesaj de navigație este compus din 25 de cadre (pagini) independente, cu lungimea de 1500 de biți fiecare. Aceste cadre, la rândul lor, sunt divizate în cate 5 sub-cadre de 300 biți. Fiecare sub-cadru este compus dintr-un antet (TLM, HOW) care are un format comun pentru toate sub-cadrele transmise. Acest antet este utilizat de către receptoarele GPS la identificarea sub-cadrelor și, de asemenea, la verificarea parității datelor.

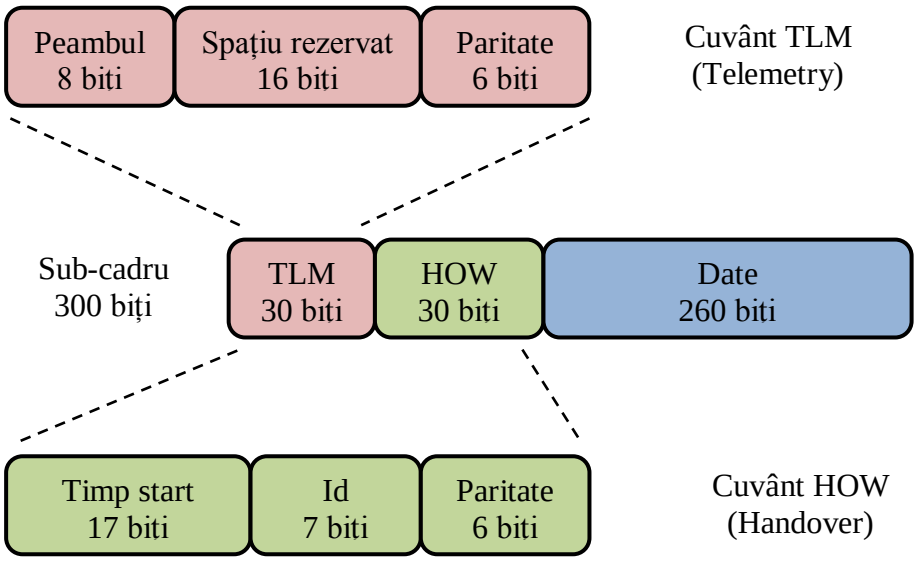


Figura 3.2.2.4.1 Structura unui sub-cadru GPS

În figura 3.2.2.4.1 este ilustrată structura unui sub-cadru al unui cadru din mesajul de navigație. Primul cuvânt al fiecărui sub-cadru, cunoscut ca și TLM, este compus dintr-un preambul de 8 biți (100001011) utilizat pentru sincronizarea cu GPS, urmat de un segment de 16 biți, rezervat utilizatorilor autorizați. Fiecare cuvânt dintr-un sub-cadru se termină cu 6 biți de paritate utilizați pentru verificarea datelor recepționate. În cazul în care biții de paritate nu corespund datelor din cuvântul TLM, receptorul va trece la următorul preambul [28] [41].

3.2.2.5 Acuratețea sistemului GPS

GPS-ul este adesea folosit în scop civil pe post de sistem de navigare. Astfel, rezultatul, obținut în urma operației realizate pe principiul triangulației, este furnizat receptorului sub forma poziției geografice (latitudine și longitudine), în general, cu o acuratețe de 10, până la 100 metri. Așadar, aplicațiile software pot utiliza coordonatele pentru a furniza instrucțiuni de orientare.

Stabilirea unei legături (blocare) de către receptoarele GPS de la nivelul suprafeței Pământului durează, de obicei, o perioadă de timp, în special dacă receptorul este un vehicul mobil sau se găsește în zone urban dense. Timpul inițial necesar pentru o blocare GPS depinde, în general, de modul în care pornește receptorul GPS. Există trei tipuri de pornire:

- **pornire fierbinte** (cea mai rapidă metodă de blocare GPS) – când dispozitivul GPS își amintește ultima poziție calculată și satelitul de conexiune, informații almanac, precum și timpul universal, încercând să se blocheze pe același satelit, calculând noi poziții pe baza informațiilor anterioare. Această metodă este posibilă doar dacă receptorul se găsește în aceeași poziție în care a fost în momentul opririi GPS-ului.
- **pornire caldă** (mai lentă decât pornirea fierbinte) – când dispozitivul GPS își amintește ultima poziție calculată, datele almanac folosite și timpul universal, dar nu și satelitul cu care a comunicat. Așadar, așteaptă săobțină semnale de la satelit și calulează o nouă poziție.
- **pornire rece** (cea mai lentă metodă de blocare GPS) – când dispozitivului GPS îi lipsesc toate informațiile anterioare, acesta așteptând să localizeze sateliții și apoi calculează blocarea GPS [29].

În prezent, receptoarele GPS sunt caracterizate de o acuratețe extrem de bună, datorită designului multi-canal paralel, ajungând la o precizie de până la 15 metri și menținând o blocare puternică, chiar și în condiții nefavorabile.

3.2.2.6 Surse de erori ale semnalului GPS

Acuratețea receptoarelor GPS este afectată de anumiți factori atmosferici și alte surse de erori, care se grupeaza in erori datorate zgomotelor, erori datorate interferențelor și erori datorate greșelilor.

a) **Erorile cauzate de zgomote** sunt o combinație între efectul zgomotului pseudo-aleator (PRN = PseudoRandom Noise) generat de cod (aproximativ 1 metru) și zgomotul produs de receptor (aproximativ 1 metru).

b) **Erorile cauzate de interferențe** sunt datorate disponibilității selective și altor surse posibile:

- **Disponibilitatea selectivă** (eng. SA = Selective Availability) este o degradare intenționată a semnalului provenit de la sateliți, de către Departamentul Apărării (DOD), pentru a limita acuratețea sistemului GPS aparținând utilizatorilor din afara aramtei sau a Guvernului S.U.A. Această măsură a fost luată pentru a împiedica adversarii militari să folosească acuratețea ridicată a semnalului GPS. Pentru fiecare vehicul spațial, interferențele pe baza disponibilității selective sunt aplicate diferit, deci soluția poziției rezultate este, de fapt, o funcție care combină interferențele aplicate fiecărui satelit din rețeaua de navigație. Astfel, se reduce acuratețea de la 30 metri la 100 metri. În anul 2000, Guvernul a oprit funcționarea acestui tip de degradare a semnalelor transmise, îmbunătățind semnificativ acuratețea receptoarelor GPS ale utilizatorilor civili.

- **Alte surse de erori cauzate de interferențe**

- erori de ceas ale receptoarelor GPS, deoarece ceasul disponibil pe acestea nu este la fel de precis ca și ceasurile atomice construite pe sateliți. Astfel, se pot produce mici erori datorate măsurării greșite a timpului, ducând la o eroare de până la 1 metru.

- erori în cazul datelor ephemeris (erori orbitale), lipsa de precizie a stabilirii poziției satelitului de comunicație, producând erori de aproximativ 1 metru.

- distribuția semnalului (greu de detectat și, uneori, greu de evitat) cauzată de reflecția semnalului GPS de către obictele și suprafețe aflate în apropierea receptorului (clădiri înalte sau munți) care pot interfera cu sau pot produce erori sau întârzieri de transmisie ale semnalului respectiv de la satelit la receptor; erorile produse ajung la 0.5 metri.

- întârzieri cauzate de ionosferă (stratul cu aer ionizat, pentru care pot fi eliminate cel puțin jumătate din întreaga cantitate) produc erori de până la 10 metri; pentru înlăturarea acestui tip de eroare, se înglobează în GPS un modul care să calculeze o sumă medie a întârzierii.

- întârzieri cauzate de troposferă – cea mai joasă parte a atmosferei – în care se petrec schimbările de temperatură, presiune și umiditate, asociate cu schimbările vremii. În acest caz se produc erori de precizie de aproximativ 1 metru; pentru o precizie mai bună, se recomandă un model care să estimeze sau să măsoare acești parametri care se modifică în timp.

- geometria/umbrirea vehiculelor spațiale, care se referă la poziția relativă a sateliților la orice moment de timp; geometria ideală a sateliților este întâlnită atunci când aceștia

sunt localizați în unghiuri largi unul față de altul; o geometri deficitară este atunci când sateliții sunt situați în linie dreaptă sau în grupuri strânse.

c) Erori datorate diferitelor tipuri de greșeli

- Greșeli la nivelul segmentului de control, datorat sistemului de calcul sau erori ale oamenilor care produc erori de precizie de la 1 metru la sute de kilometri.
- Greșeli ale utilizatorilor, inclusiv selecția geodezică greșită, care poate produce erori de precizie de la 1 metru la sute de metri.
- Erori la nivelul receptorului GPS, datorate unor defecțiuni ale componentelor software sau hardware, care pot duce la apariția unor erori de orice dimensiune [24] [28].

3.2.3 Serviciile GPRS și Wi-Fi

Tehnologiile **GPRS (General Packet Radio Service)** și **Wi-Fi** sunt cele care asigură conxiunea dintre un dispozitiv digital, precum telefonul inteligent (smartphone), și rețeaua de calculatoare (Internet).

3.2.3.1 GPRS

General Packet Radio Service (GPRS) este un serviciu mobil de date orientat pe pachete, de generația 2,5G, prin care se permite transmisia, cu viteză sporită, a datelor și repartizarea acestora în fișiere mici, care sunt transmise separat, destinatarul cunoscând modul în care acestea trebuie reasamblate. Generația 2,5G reprezintă o tehnologie combinată între generația 2G și 3G, și anume faptul că sistemul GPRS permite rețelelor mobile 2G și 3G să acceseze serviciile on-line ale rețelelor externe, precum Internet-ul.

Acest serviciu este o extensie a sistemului de comunicții mobile standard (GSM = Global System for Mobile communication), obținut prin dezvoltarea acestor rețele. Se bazează pe comutarea pachetelor, iar transmisia datelor se constituie pe baza modului de folosire al intervalelor de timp ale GSM, utilizând protocoale de transmisie a pachetelor X.25 și TCP/IP. Astfel, în cazul în care sunt utilizate toate cele opt intervale de timp ale canalelor GSM, viteza maximă de transmisie a pachetelor, prin GPRS, poate ajunge, teoretic, la 171,2 kbit/s. Viteza medie este de 40 kbit/s, comparabilă cu cea a unu modem prin cablul de telefonie (dialup modem), însă GPRS-ul oferă posibilitatea unei conexiuni din aproape orice locație [32] [33] [34].

În figura 3.2.3.1.1 este reprezentat schematic un sistem în care se realizează transmisia datelor prin pachete GPRS.

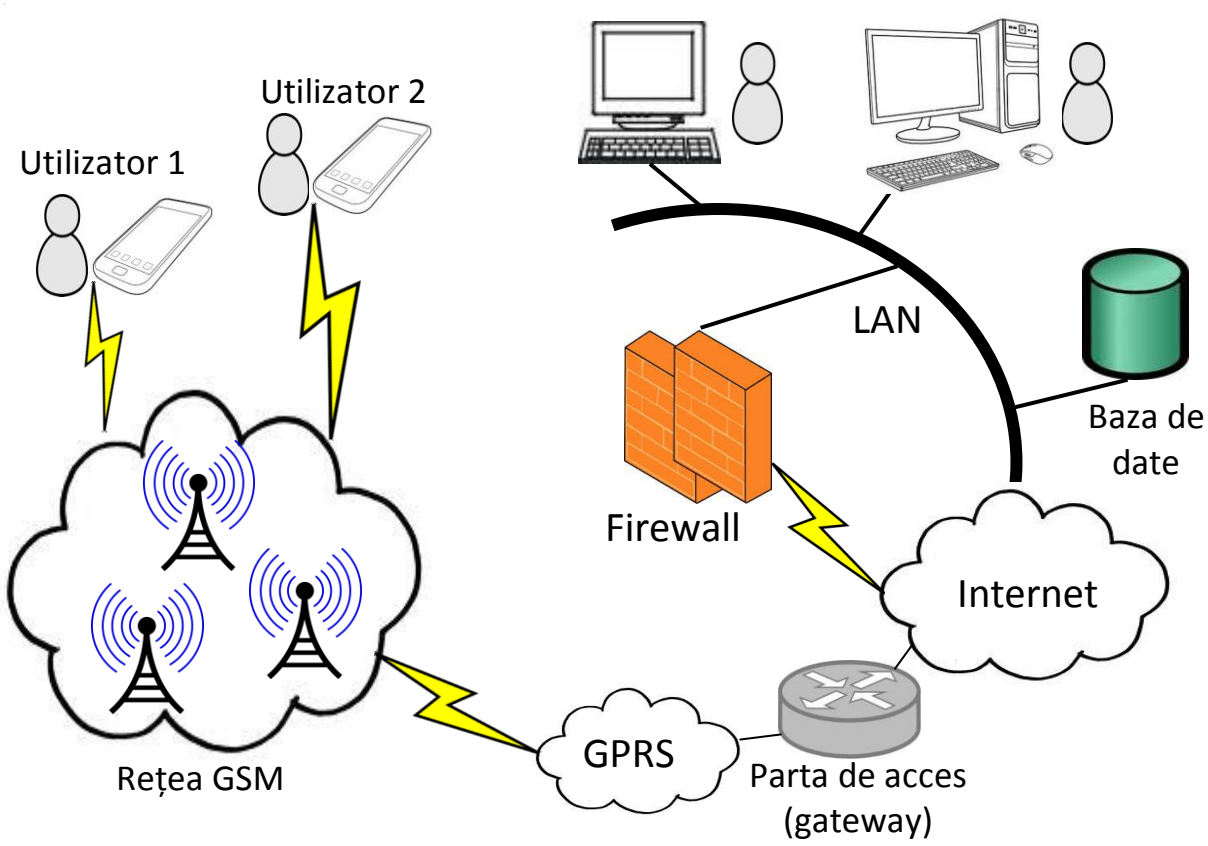


Figura 3.2.3.1.1 Sistem cu comunicare prin GPRS

3.2.3.2 Wi-Fi

Wi-Fi (Wireless Field) este numele unei tehnologii renumite de rețele fără fir (wireless), care folosește unde radio pentru a furniza Internet de mare viteză și conexiuni de rețele. Această tehnologie permite unui dispozitiv electronic, mobil să schimbe date, prin intermediul undelor radio, într-o rețea de calculatoare, inclusiv conexiuni la Internet.

Wi-Fi administrează conexiunea dintre emițător și receptor, printr-o rețea fără legături fizice prin fire, ci folosind tehnologia de radio-frecvență (RF) , cu o frecvență din spectrul electromagnetic, asociat cu propagarea undelor radio.

Alianța Wi-Fi (Wi-Fi Alliance) definește Wi-Fi ca fiind un produs de rețea locală fără fir (WLAN = Wireless Local Area Network) care se bazează pe standardele IEEE (Institutul de Inginerie Electrică și Electronică). Datorită faptului că majoritatea rețelelor WLAN au la bază aceste standarde, în limba engleză, termenul ”Wi-Fi” este folosit ca și sinonim pentru WLAN.

Mijlocul efectiv prin care este asigurat schimbul de date între dispozitiv și resursa de rețea este punctul de acces la rețeaua fără fir (hotspot). Un astfel de punct de acces are o arie de acoperire de aproximativ 20 metri în interior (în clădiri) și o arie mai largă în mediul exterior.

Acoperirea poate fi asigurată și pe arii restrânse precum o cameră cu pereți care blochează undele radio sau pe arii întinse de mulți kilometri pătrați, ultima situație fiind rezolvată prin suprapunerea mai multor puncte de acces [35] [36].

Structura unui sistem cu comunicare prin Wi-Fi este prezentat în figura 3.2.3.2.1.

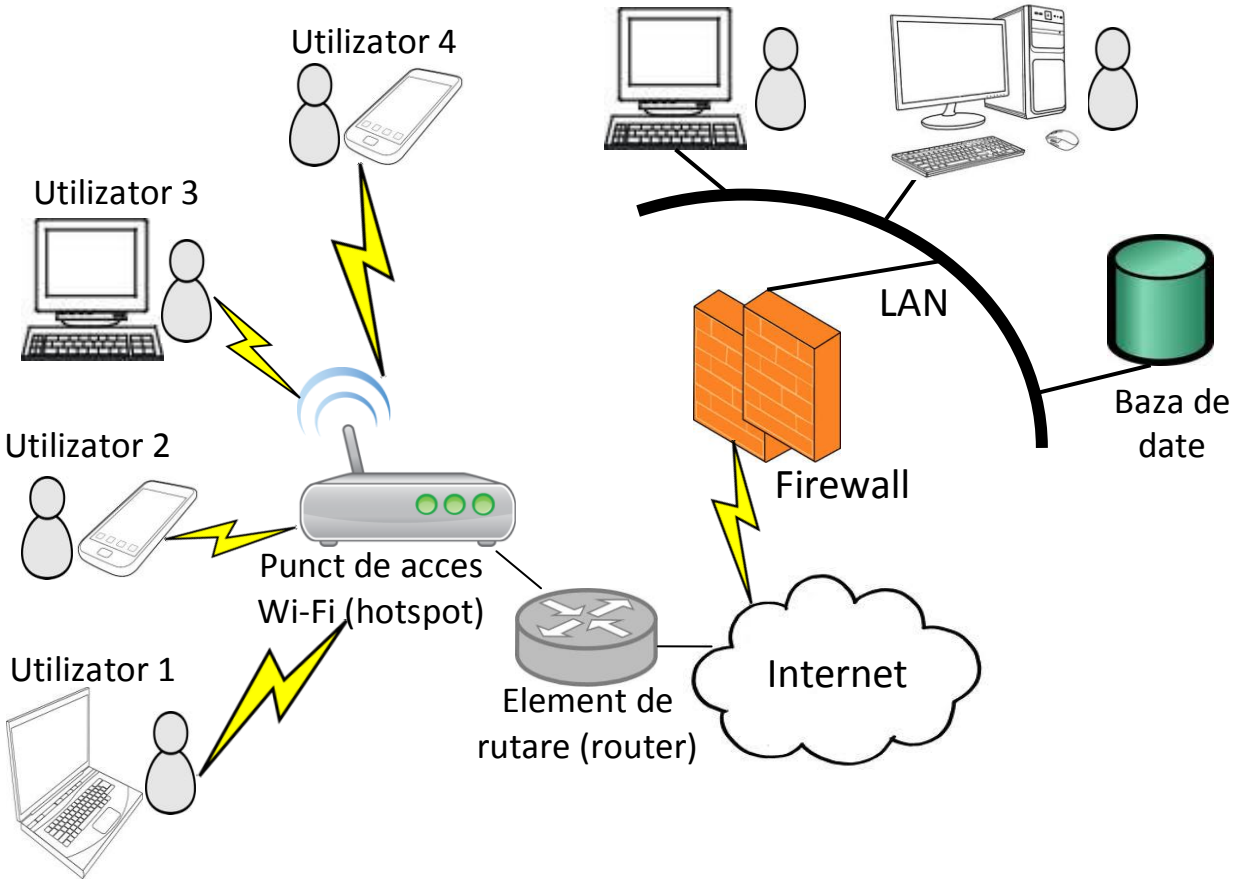


Figura 3.2.3.2.1 Sistem cu comunicare prin Wi-Fi

3.2.4 Sistemul MySQL de administrarea al bazelor de date

MySQL este cel mai popular sistem de gestiune a bazelor de date relaționale (**RDBMS**). Acesta este dezvoltat, distribuit și susținut de Corporația Oracle și este disponibil gratuit (Open Source SQL). Sistemul funcționează ca și un server care asigură accesul mai multor utilizatori la un număr variat de baze de date.

Faptul că MySQL este Open Source, înseamnă că oricine are posibilitatea de a folosi și modifica acest program, descărcându-l de pe Internet, fără a fi supuși unor plăți, în schimb. Modificările pe care utilizatorii doresc să le efectueze asupra codului, pentru a se adapta la nevoile personale, sunt limitate, însă, de licența GPL, care specifică diferite situații în care pot fi aduse modificări și ce anume poate sau nu poate fi modificat.

O bază de date este o colecție structurată de date, de orice tip (numere întregi, reale; șiruri de caractere, date, etc.), iar pentru a accesa, adăuga și procesa datele stocate într-o bază de date a unui calculator, este nevoie de un sistem de management a bazei de date precum MySQL.

O bază de date relațională se bazează pe evitarea depozitării informațiilor într-o magazie masivă, prin stocarea datelor în tabele separate, pentru care se stabilesc un set de reguli care dirijează legăturile dintre câmpurile tabelelor. Aceste reguli pot fi de tipul ”unu-la-unu”, ”unu-la-mai mulți”, unică, obligatorie, opțională sau pointeri între diferite tabele, asigurând consistența și integritatea aplicației care folosește base de date.

Sistemul MySQL este ușor de folosit, foarte rapid și sigur, putând rula pe un calculator, împreună cu alte aplicații, fără a necesita o atenție deosebită.

SQL este cel mai popular limbaj standardizat folosit pentru comunicarea cu bazele de date. În funcție de mediul de programare utilizat, pot fi folosite variate metode de accesare a bazei de date: SQL în mod direct (ex. generare de rapoarte), încadrarea unor linii specifice sintaxei SQL, intr-un cod scris cu un limbaj de programare diferit sau prin folosirea unei interfețe de programare a aplicațiilor (API), care are la bază sintaxa SQL (ex. JDBC).

JDBC este o interfață Java, care poate accesa orice tip de date, în special cele stocate în baze de date relaționale. Aceasta ajută la implementarea aplicațiilor care gestionează trei principale activități:

- Conexiunea la baza de date
- Realizarea unor interogări pe baza de date (acces, adăugare, ștergere, actualizare)
- Preluarea și prelucrarea rezultatelor primite de la baza de date, ca și răspuns la interogările anterior efectuate [16] [37]

În figura 3.2.4.1 este reprezentată arhitectura JDBC pe trei nivele pentru accesul datelor.

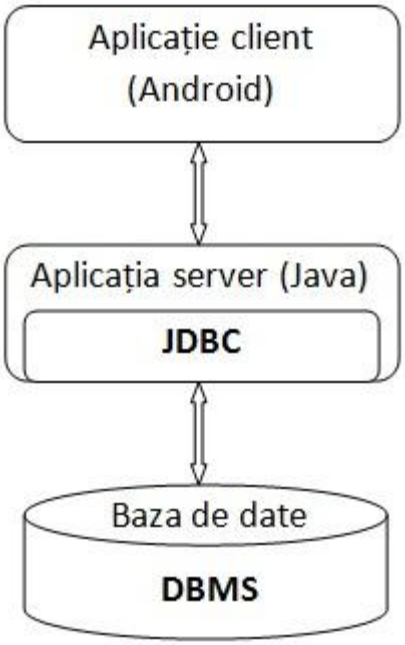


Figura 3.2.4.1 Arhitectura JDBC pe trei nivele

Capitolul 4. PROIECTAREA ȘI IMPLEMENTAREA SISTEMULUI ATBA (Android Taxi Booking Application)

4.1 Arhitectura aplicației

Lucrarea de față prezintă o aplicație care permite utilizatorilor/clienților să rezerve cel mai apropiat taxi de locația lui actuală, prin considerarea coordonatelor geografice ale poziției. Sistemul de rezervare propus constă din trei componente principale:

- Modulul de efectuare a solicitării unui taxi;
- Modulul de urmărire a vehiculelor;
- Modulul de control al taxiurilor și de administrare a comenzilor.

Soluția software permite utilizatorilor să realizeze următoarele lucruri, prin interediul unor intefețe utilizator:

- Autentificarea pe un server web pentru a putea realiza comenzi de taxiuri;
- Vizualizarea informațiilor de timp real cu privire la statutul comenzii: locația vehiculului în trafic (timp de așteptare, astimarea duratei și a distanței călătoriei).

În figura 4.1.1 este sintetizată arhitectura sistemului software, descrisă anterior prin intermediul modulelor componente.

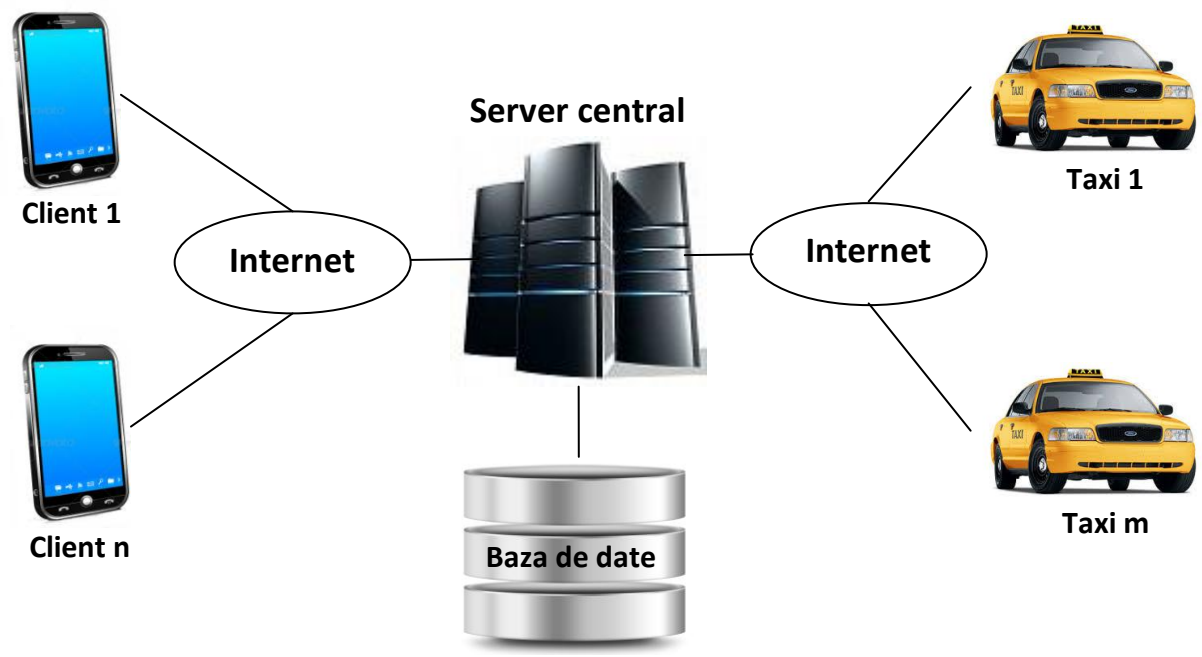


Figura 4.1.1 Arhitectura sistemului

Se observă un număr variat de clienți/utilizatori care se pot conecta la serverul aplicației, alături de un număr diferit de taxiuri. Comunicația dintre acestea și serverul central este realizată prin Internet, iar serverul dispune de o permanentă legătură cu baza de date a sistemului, în care sunt stocate informațiile referitoare la clienți și taxiuri, pe baza cărora sunt îndeplinite funcționalitățile aplicației.

4.2 Cazuri de utilizare

Cazurile de utilizare reprezintă o tehnică de modelare utilizată prentu a reda funcționalitatea unui sistem existent, dar, în cele mai multe situații, se folosește pentru a descrie un nou sistem, înainte de a fi implementat.

Diagramele de utilizare (Use Case Diagrams) sunt diagrame de comportament, care scot în evidență tipul utilizatorilor disponibili în aplicație și interacțiunile dintre aceștia și sitem. Acestea sunt realizate din perspectiva *actorilor* (utilizatorilor): clienți și taxiuri și specifică modul de comunicare între ei și relațiile dintre ei și partea de control a sistemului (serverul central). În figura 4.2.1 este ilustrată diagrama cazutirlo de utilizare, prezentând principalele operații pe care modulele sistemului le pot efectua.

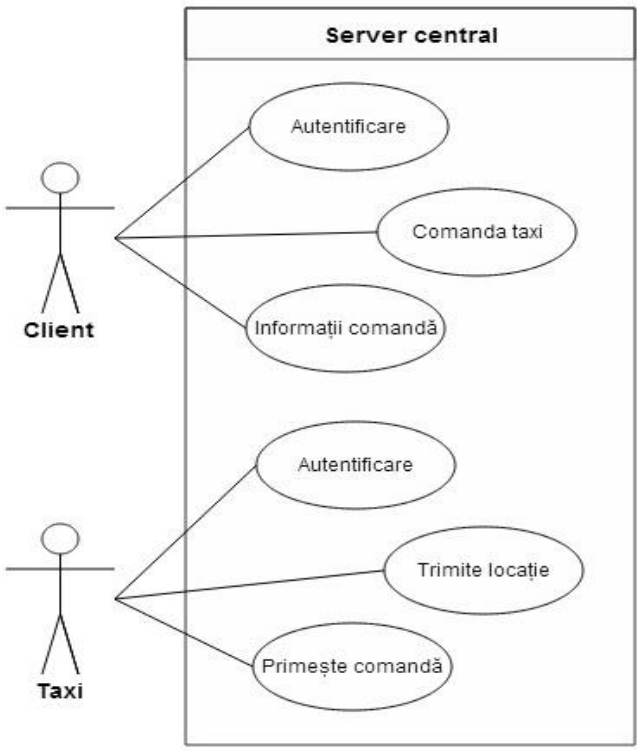


Figura 4.2.1 Diagrama Use Case

Astfel, clientul accesează aplicația, iar în cazul unei autentificări realizate cu succes, acesta poate efectua comanda unui taxi, către o destinație aleasă de el, de pe hartă. În momentul în care serverul alocă un taxi disponibil pentru a onora comanda respectivă, clientul primește informații referitoare la comandă: distanța în kilometri și durata în timp (minute) a cursei, calculată pe baza coordonatelor GPS ale locației curente a clientului și destinației.

Pe de altă parte, taxi-ul, se autentifică, la rândul său, după care începe să trimită periodic, serverului, coordonatele GPS ale locației actuale. Ca și răspuns de la server, acesta primește un mesaj în care apare specificat dacă are sau nu vreo comandă asignată.

Serverul central este principala componentă care asigură comunicația și controlul celorlalte module, prin intermediul bazei de date.

Scopul principal al diagramelor de utilizare este de a oferi o prezentare generală a modului în care sistemul va fi utilizat, de a ilustra modul de interacțiune între sistem și actori, oferind o privire de ansamblu asupra funcționalităților urmărite a fi implementate în sistemul nou dezvoltat.

4.3 Diagrame UML

La fel cum pentru un arhitect schițele reprezintă proiectul detaliat al unei clădiri, diagramele UML (Unified Modeling Language) permit modelelor software să fie construite, obervate și manipulate în timpul analizării și al proiectării. Modelarea este o tehnică folosită în multe discipline, în special în inginerie, întrucât ajută înțelegerea anumitor procese mai complexe din sistem și permit evaluarea proiectului în raport cu diferite criterii, cum ar fi flexibilitatea, securitatea, etc.

UML este un limbaj care realizează descrierea grafică a subiectelor tratate de OOAD (Object-Oriented Analysis and Design), prin care aplicațiile software pot fi documentate, specificate, construite și observate. După examinarea atentă a modelului, pot fi determinate eventuale deficiențe sau disfuncționalități ale sistemului, putând fi mai ușor de stabilit eventuale posibilități de soluționare a problemelor sau de îmbunătățire a stării sitemului.

Scopul final al acestor diagrame este de a deriva codul limbajului de programare. Generarea codului din modele UML este cunoscut sub numele de inginerie avansată (forward engineering).

Există mai multe tipuri de diagrame UML folosite pentru descrierea sistemelor software, cum ar fi: diagrama secvențială, digrama clsaselor, diagrama obiectelor, diagrama de colaborare etc.

În cazul aplicației de față, au fost realizate două diagrame secvențiale, pentru a exprima fluxul de control și schimbul de date dintre componentele modelului (servere, clineți, taxiuri) și o diagramă a claselor, pentru evidențierea structurii logice a sistemului implementat [38].

4.3.1 Diagrame secvențiale

Diagramele secvențiale reprezintă un tip de diagramă de interacțiuni care exprimă relațiile și interacțiunile dintre procesele implicate în sistem, precum și ordinea acțiunilor care au loc la nivelul sistemului. Interacțiunile se bazează pe secvențele de mesaje transmise de la un obiect la altul. Cu acest tip de diagramă se scoate în evidență și durata de viață a fiecărui proces, în concordanță cu durata mesajelor care conectează procesele, de la cerere, până la răspuns.

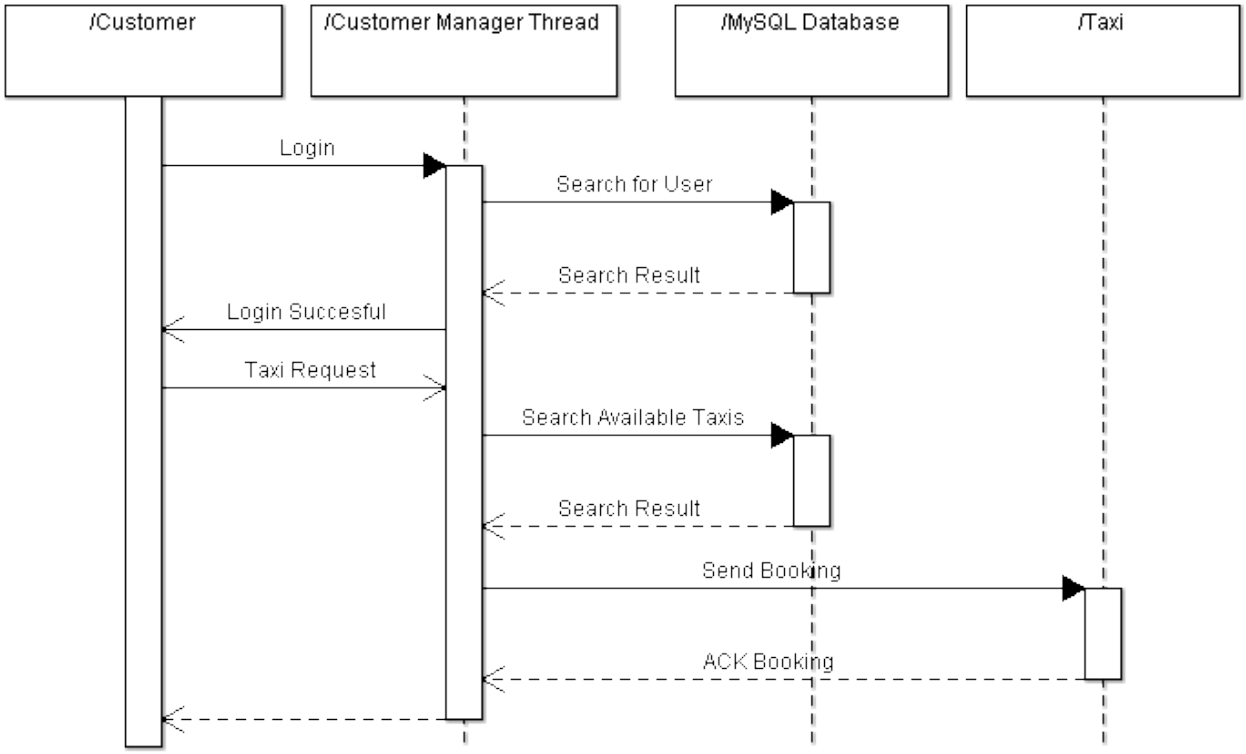


Figura 4.3.1.1 Diagrama secvențială Client-Server

În figura 4.3.1.1 este reprezentată relația dintre client și server: după accesarea aplicației, clientul încearcă să se autentifice; serverul caută, în baza de date, acel utilizator particular, iar în momentul în care este găsit, clientul primește permisiunea de a continua cu comanda unui taxi. Odată efectuată comanda, prin specificarea destinației dorite, serverul o preia și o procesează, verificând în baza de date taxiurile disponibile și calculând distanțele dintre acestea și client. Când cel mai apropiat taxi este determinat, serverul trimite clientului un răspuns pozitiv, prin afișarea unei hărți cu taxiurile și a duratei și distanței călătoriei cu taxiul.

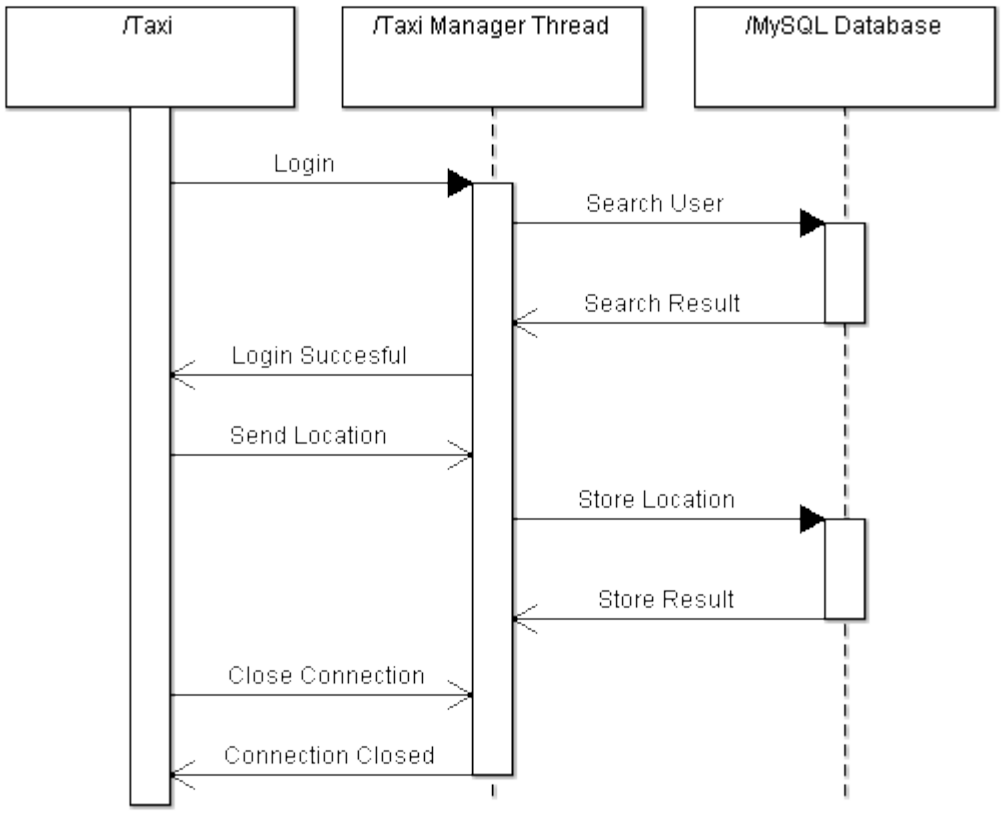


Figura 4.3.1.2 Diagrama secvențială Taxi-Server

În a doua diagramă secvențială (figura 4.3.1.2), sunt ilustrate relațiile dintre taxi și server: taxiul se autentifică, pentru a putea trimite, ulterior, informații serverului. În momentul confirmării autentificării, taxiul începe transmiterea periodică acoordonatelor sale GPS pentru poziția actuală, aflându-se în mișcare. Aceste date sunt preluate de server și actualizate în baza de date, iar ca și răspuns pentru a confirma recepția coordonatelor, serverul verifică existența unor eventuale comenzi efectuate pentru taxiul respectiv, trimițând taxiului un mesaj în care specifică daca are sau nu comandă de preluat.

4.3.2 Diagrama claselor

Diagrama claselor este pricipala schemă bloc a modelării orientate-obiect. Este o diagramă statică, care descrie structura sistemului, prin intermediul claselor componente, al atributelor , operațiilor (metodelor) acestora și evidențiază relațiile dintre ele (asociere, agregare, imbricare, derivare, extindere) [39] [40].

Pentru aplicația descrisă în lucrarea de față, a fost realizată diagrama de clase, împărțită în două: prima parte referitoare la serverele Java (figura 4.3.2.1), iar a doua parte la clienții Android (figura 4.3.2.2).

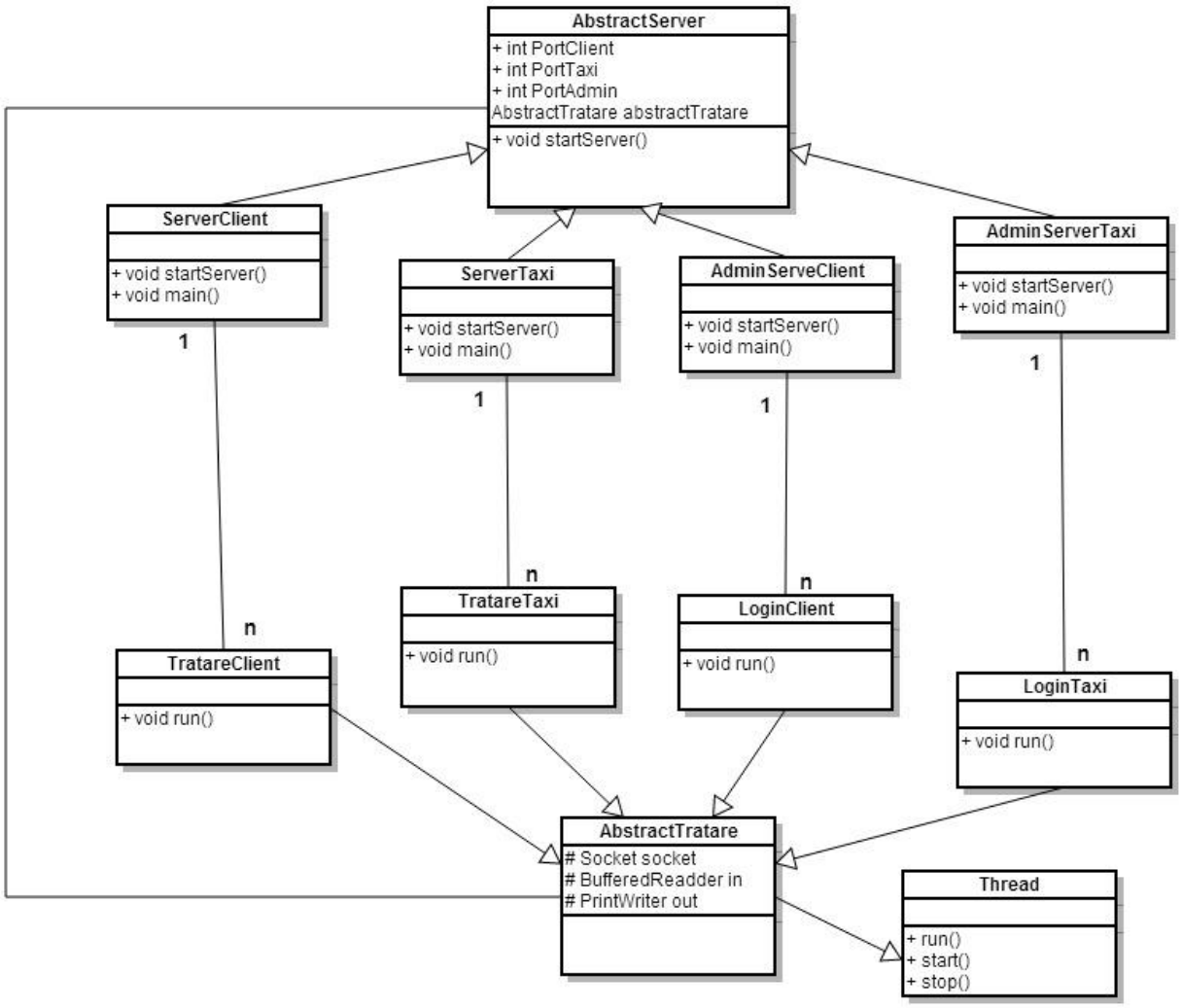


Figura 4.3.2.1 Diagrama claselor pentru serverele Java

Clasa *AbstractServer* e construită pentru declararea porturilor de acces pe servere, iar clasa *AbstractTratare* pentru declararea socketului și a bufferelor de intrare ieșire necesare serverului. Clasa *AdminServerClient*, împreună cu *LoginClient*, asigură verificarea autentificării clientului, în baza de date. Clasele *AdminServerTaxi* și *LoginTaxi* asigură verificarea autentificării taxiurilor. Clasa *ServerClient* reprezintă serverul care se ocupă de acțiunile clientului prin intermediul ”manipulatorului” *TratareClient*, iar *ServerTaxi*, alături de *TratareTaxi*, controlează acțiunile taxiurilor.

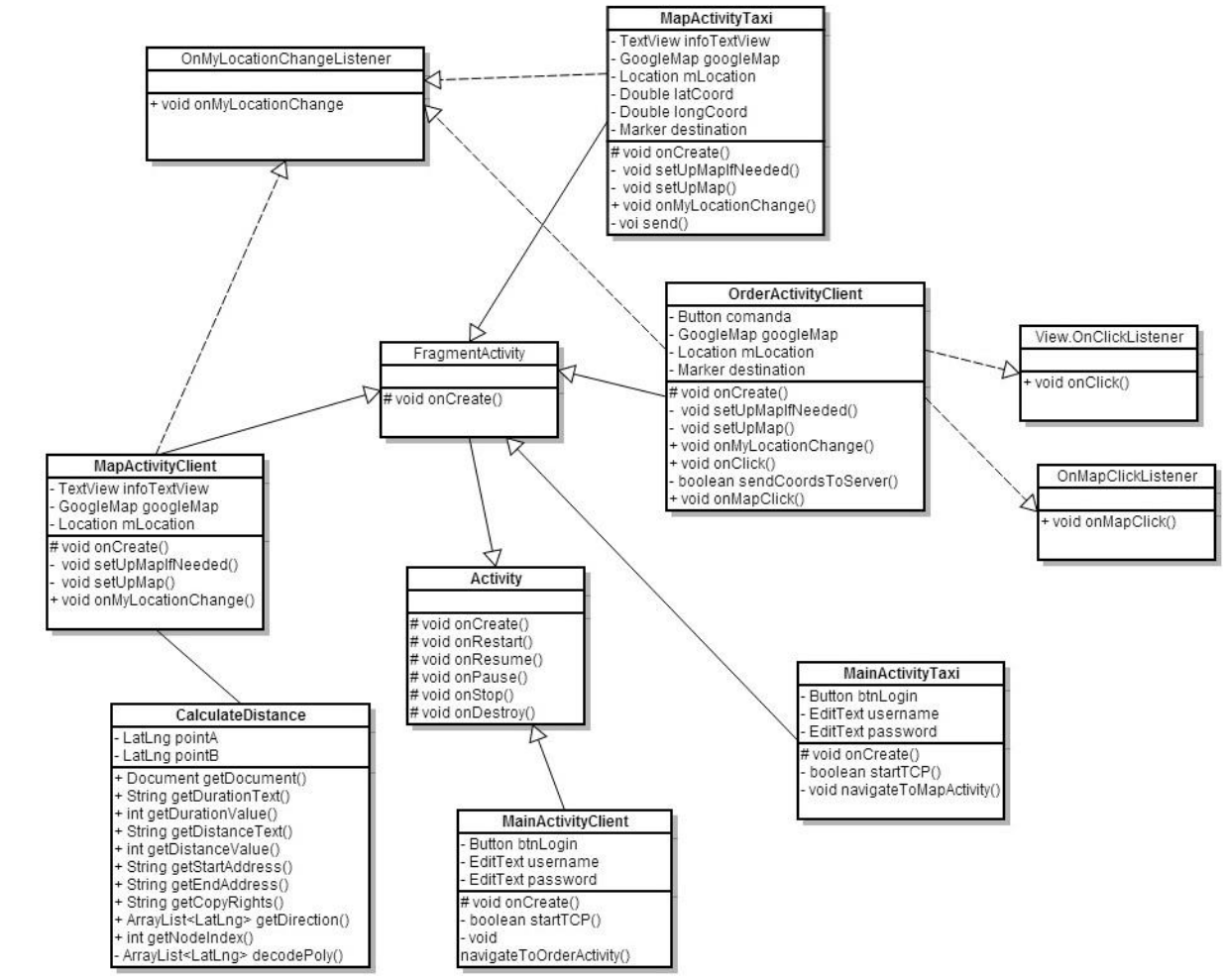


Figura 4.3.2.2 Diagrama claselor pentru clienții Android

Clasa MainActivityClient realizează prima interfață corespunzătoare clientului, și anume cea de autentificare; din această clasă se face trecerea la clasa OrderActivityClient, prin intermediul unui intent, aceasta fiind a doua interfață a utilizatorului, cea în care selectează destinația și plasează comanda. MapActivityClient este clasa care reprezintă a treia interfață, la care se trece printr-un intent, din interfața anterioară. Este intefața în care clientul primește informații legate de comandă.

Clasa MainActivityTaxi reprezintă interfața de autentificare a taxiului, iar clasa MapActivityTaxi realizează a doua interfață corespunzătoare taxiului în care acesta primește mesajul de comandă, împreună cu coordonatele destinației și locația clientului pe hartă.

În clasa CalculateDistance este construit algoritmul de calculare a distanței între două puncte pe harta Google Maps, în funcție de drum.

4.4 Implementare

Sistemul tratat în prezenta lucrare de diplomă, este construit pe baza modelului de comunicare client-server, prin Internet, folosind protocolul TCP/IP, comunicarea propriu-zisă realizându-se prinsocketuri, pe anumite porturi specificate de server. Modulele aplicației sunt implementate în mod diferit unul față de celălalt, astfel: serverul central, format din două servere multifir (ServerClient și ServerTaxi), care se ocupă separat de tratarea fiecărui tip de client în parte, este dezvoltat în limbajul de programare Java, iar clienții (utilizatori și taxiuri) sunt implementați cu ajutorul platformei de dezvoltare Android, caracterizată printr-un aport de biblioteci suplimentare față de Java JDK. Mediul de dezvoltare utilizat este Eclipse Juno, care, pentru a putea realiza anumite operații necesare obținerii unor funcții speciale și pentru a putea utiliza platforma de programare Android, a avut nevoie de câteva configurări.

4.4.1 Configurarea mediului de dezvoltare

Pentru a putea dezvolta aplicații cu platforma Android, este necesară descărcarea pachetului Bundle ADT [40] , care include componente esențiale ale SDK-ului Android și o versiune a Eclipse IDE (Integrated Development Environment) de echipamente ADT înglobate, pentru a eficientiza dezvoltarea aplicațiilor Android. Arhiva descărcată se salvează în directorul de lucru, se lansează în execuție eclipse, din directorul eclipse al pachetului bundle. În continuare se descarcă și se instalează o serie de biblioteci SDK, cu ajutorul Android SDK Manager, din eclipse (se bifează bibliotecile necesare, apoi se apasă butonul *Install x packages*):

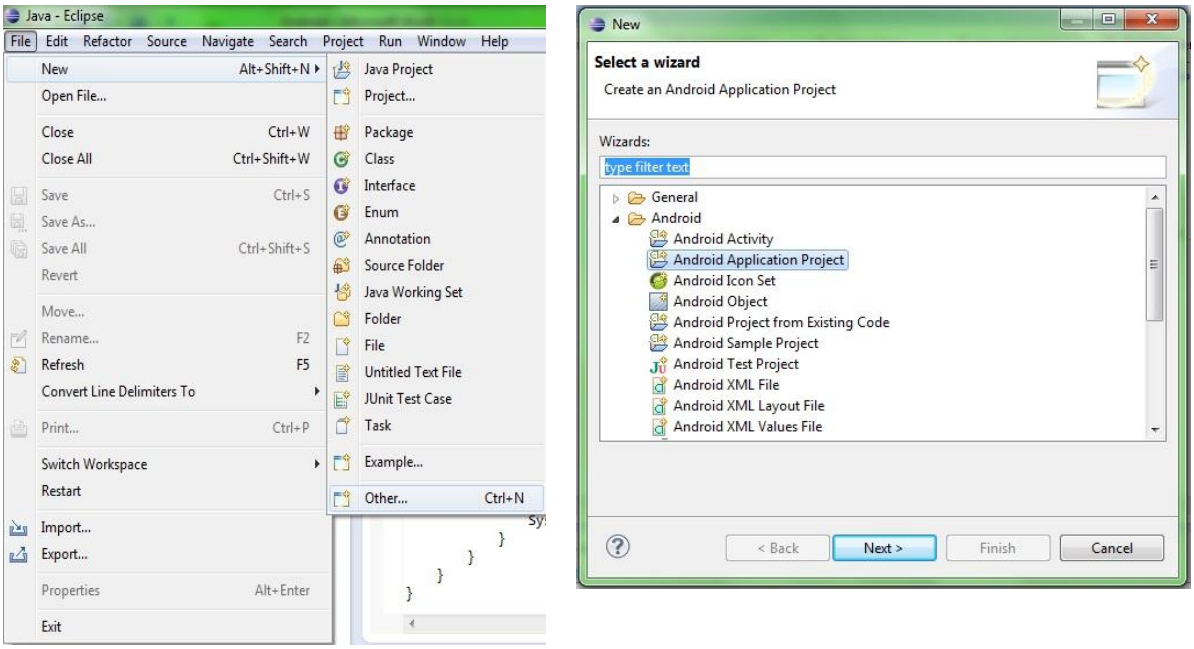
- Din directorul tools, se aleg bibliotecile *Android SDK Tools* și *Android SDK Platform-tools* care conțin principalele instrumente de lucru pentru platforma Android;
- Din directorul corespunzător variantei de Android alese pentru implementarea aplicației (în cazul de față, Android 2.2 – API 8), se selectează bibliotecile *SDK Platform*, *Samples for SDK* – cu exemple de aplicații pentru începători, *Google APIs* – set de interfețe care permit interacțiunea cu serviciile Google;
- Din directorul extras, se alege biblioteca *Android Support Library*, care furnizează un fișier JAR (Java ARchive) care conține o bibliotecă de interfețe (API) care permit folosirea, în cadrul aplicației personale care rulează pe o versiune de Android mai veche, a unor interfețe API Android mai recente și biblioteca *Google USB Driver*, care permite rularea aplicațiilor pe dispozitive cu sistem de operare Android, prin conectarea prin USB la calculator.

Odată ce aceste biblioteci sunt instalate, se poate începe crearea unui proiect Android.Pentru acest lucru, trebuie parcurși câțiva pași simpli:

1. Din meniul mediului de programare eclipse se alege *File* → *New* → *Other ()*.

2. Se extinde directorul *Android*, se selectează *Android Application Project*, apoi se apasă butonul *Next*.

Pașii 1 și 2 sunt ilustrați în figura 4.4.1.1.



Fihura 4.4.1.1 Pașii 1 și 2. Creare proiect Android

3. În fereastra nouă, se completează numele aplicației (ex. ”App1”), iar câmpul pentru numele pachetului se autocompletează cu textul ” com.example.App1”, după numele aplicației. Îm următoarele câmpuri se alege varianta de Android dorită, în cazul proiectului de lincență a fost folosită varianta API 8: Android 2.2 (Froyo). După completarea conform figurii 4.4.1.2, se apasă butonul Next.

4. Opțiunile pentru configurarea proiectului pot fi păstrate cele predefinite. Se apasă Next.

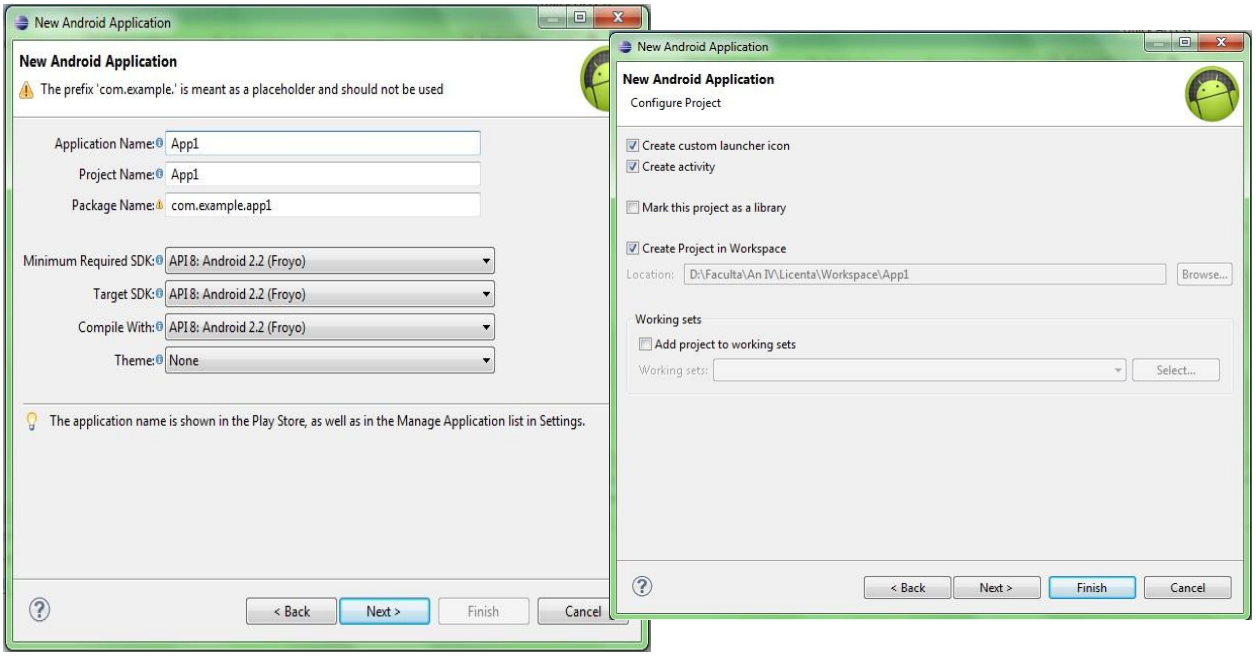


Figura 4.4.1.2 Pașii 3 și 4. Denumire și configurare proiect

5. Se alege un logo pentru aplicație.

6. Se selectează opțiunea *Create Activity* → *BlankActivity*. Acești doi pași sunt ilustrați în figura 4.4.1.3.

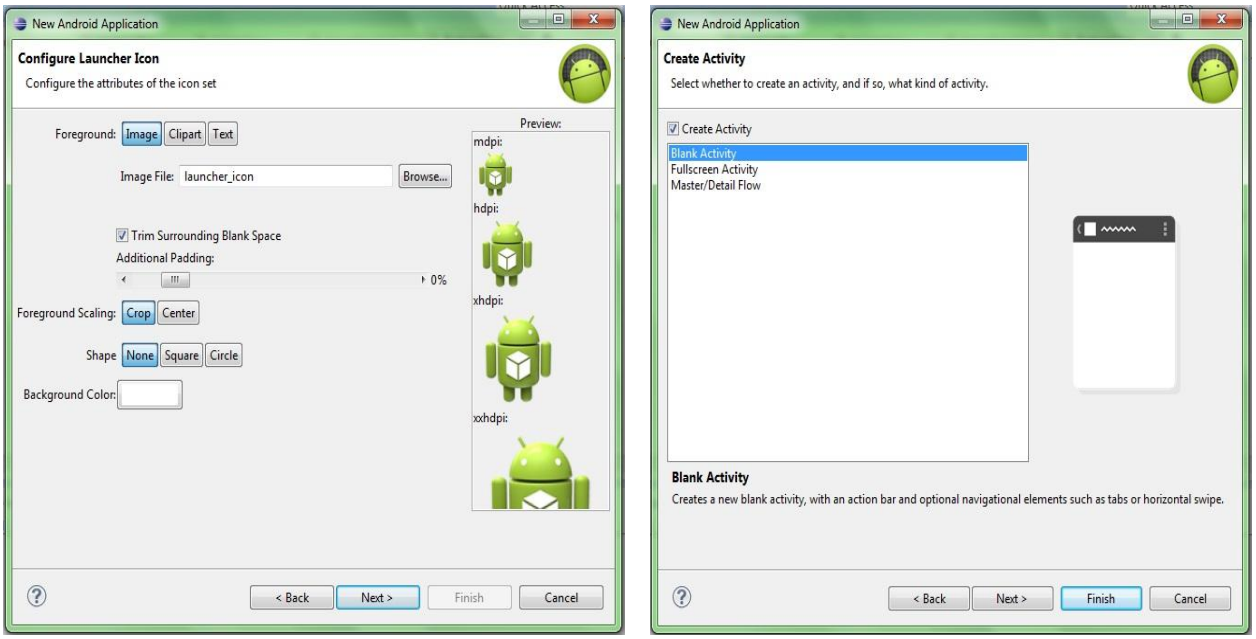


Figura 4.4.1.3 Pașii 5 și 6. Selectare logo și crearea unei noi activități

7. Se alege o denumire pentru activitate, apoi se apasă butonul *Finish* (figura 4.4.1.4).

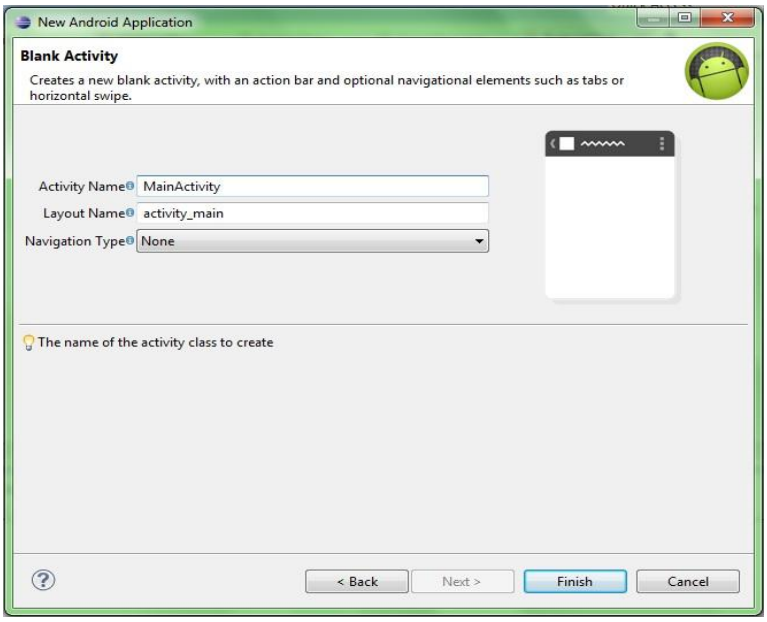


Figura 4.4.1.4 Pasul 7. Denumire activitate

În acest moment, noul proiect Android este creat și se deschide fișierul *res* → *layout* → *activity_main.xml*, împreună cu *Graphical Layout*, clase din fișierul de resurse, cu ajutorul cărora se definește interfața utilizator, ca în figura 4.4.1.5. În partea stângă a figurii se observă structura proiectului, cu toate fișierele și bibliotecile înglobate.

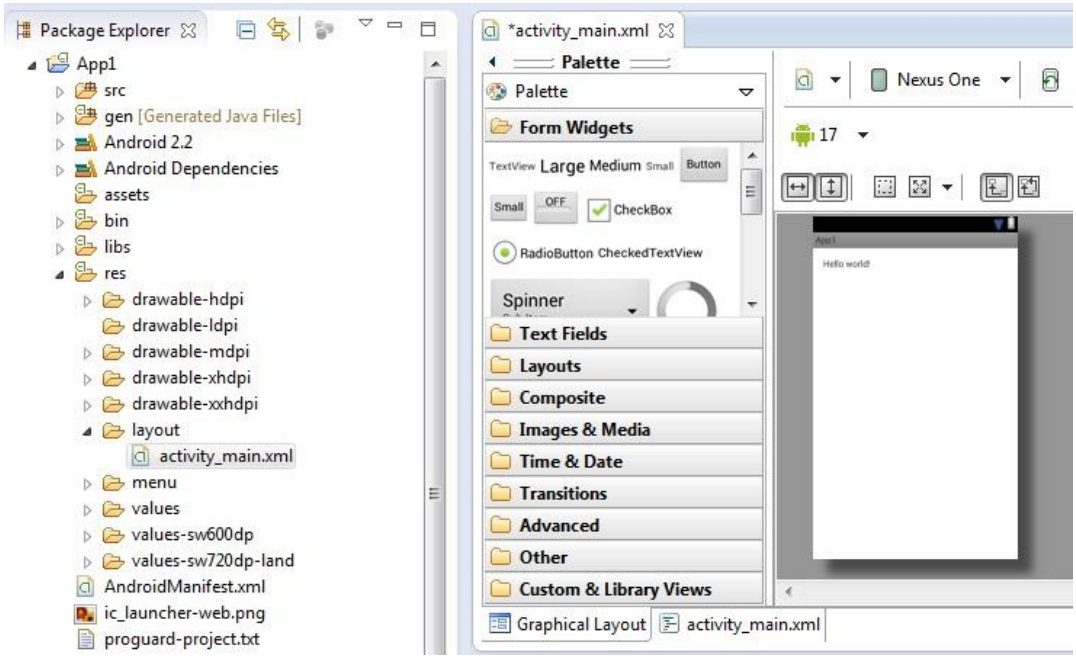


Figura 4.4.1.5 Aspectul inițial al fișierului Graphical Layout

Conținutul predefinit al fișierului activity_main.xml este afișat în figura 4.4.1.6.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Figura 4.4.1.6 Conținutul fișierului activity main.xml

Un alt fișier important din cadrul unui proiect Android este AndroidManifest.xml. Acest fișier cuprinde informații esențiale despre aplicația Android, informații pe care sistemul trebuie să le dețină înainte să ruleze orice parte de cod din aplicație. Unele din cele mai importante atribuții ale fișierului sunt prezentate în figura 4.4.1.7 și enunțate mai jos:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.app1"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="8" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.app1.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Figura 4.4.1.7 Structura de bază a fișierului AndroidManifest.xml

Principalele sarcini asigurate de fișierul AndroidManifest.xml:

- denumește pachetul Java al aplicației;
- descrie componentele aplicației;
- specifică permisiunile aplicației necesare pentru a accesa părți protejate ale API (de exemplu accesul la Internet) și pentru a interacționa cu alte aplicații;
- declară nivelul minim al interfeței Android necesar pentru aplicație;
- menționează lista de biblioteci de care aplicația în cauză trebuie să se lege;

informații legate de activitate în sine.

După ce aceste fișiere au fost configurate, aplicația poate fi scrisă în fișierul MainActivity.java, în figura 4.4.1.8 putând fi observată locația acestuia în proiect.

Figura 4.4.1.8 Fișierul destinat scrieii codului sursă

În componența acestui fișier este deja implementată metoda onCreate(), despre care se știe că este implementată de toate aplicațiile Android, care va fi suprascrisă, astfel încât să fie instanțiate/create obiectele din aplicație.

Pe partea de client Android, aplicația este realizată cu interfețe care afișează Google Maps, pentru selectare și vizualizare locații, în funcție de coordonatele GPS. Pentru a permite folosirea acesteia, este necesară configurarea Google Maps API, pentru proiectele care vor apela funcții ale hărții web. Acest lucru se realizează prin parcurgerea câtorva pași, descriși în tutorialul [41]:

1. Instalare Google Play services, în Android SDK Manager

Pentru început, se cere instalarea bibliotecii Google Play services, din Window -> Android SDK Manager. Se bifează în fișierul Extras, în dreptul pachetului Google Play services, apoi se apasă butonul Install 1 package, conform figurii 4.4.1.9.

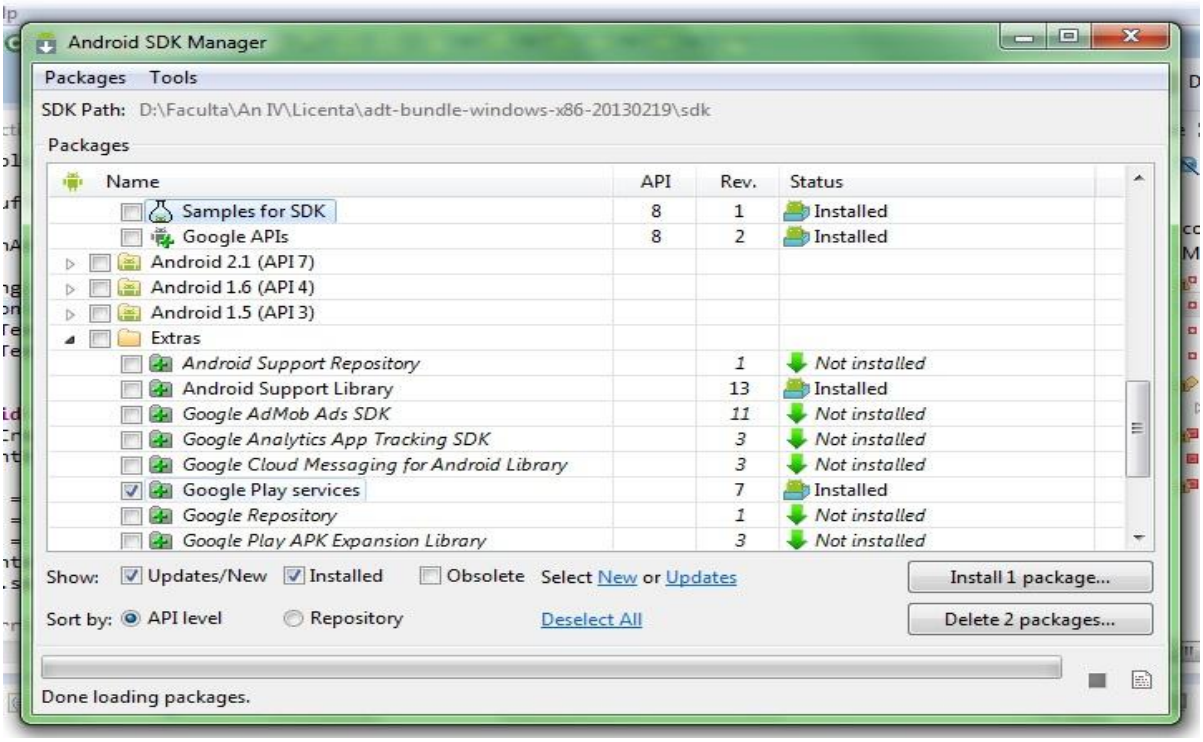


Figura 4.4.1.9 Instalare pachet Google Play service

2. Obținerea cheii

La pasul următor este nevoie de o cheie validă a Google Maps API, care se obține online, de pe Google APIs Console, prin specificarea cheii de semnătură a aplicației si a numelui pachetului. Cheia de depanare, pentru subscrierea aplicației, se găsește în fișierul *users/.android/debug.keystore*.

Se cere crearea unei amprente SHA-1. Pentru aceasta, se scriu în CMD comenzile următoare:

```
keytool –list –v – alias androiddebuigkey\
-keystore <path_to_debug_keystore>debug.keystore \
-storepass android –keypass android
```

unde path_to_debug_keystore este calea spre fișierul keytool, din fișierul bin al JDK-ului din Java. Rezultatul obținut se înscrie într-un fișier text key.txt. Se deschide acest fișier, de unde se copiază șirul de caractere care urmează dupa SHA-1, în cazul aplicației prezentate acest șir este: 60:74:9D:17:E0:5D:F5:7A:E0:DA:B8:58:74:98:B7:0F:8D:07:11:26.

Este necesară înregistrarea pe Google APIs Console, unde trebuie parcurși mai mulți pași pentru a putea folosi Google Maps pentru Android:

- Se selectează câmpul *Services* (se observă că toate serviciile sunt oprite - OFF).

- Se activează *Google Maps Android API v2*. (ON)
- Se selectează câmpul *API Access*
- Se apasă butonul *Create new Android key...*
- Se introduce amprenta SHA-1 și numele pachetului aplicației (*com.example.tcpclient*), separate prin caracterul ”;”
- Se apasă butonul *Create*
- Google APIs Console crează, ulterior, cheia pentru aplicația Android (*Key for Android apps(with certificates)*): AIzaSyDYZiFMD4ihAOLzwTw26shcuPQtE7APA90
- Se copiază valoarea chei, pentru a fi folosită mai departe.

3. Adăugarea cheii în aplicație

Pentru a beneficia de utilizarea Google Maps, în aplicație, trebuie inserată cheia în fișierul AndroidManifest.xml, prin introducerea textului următor, chiar înainte de a închide eticheta </application>:

```
<meta-data
      android:name="com.google.android.maps.v2.API_KEY"
      android:value="API_KEY" />
```

unde API_KEY trebuie înlocuită cu valoarea generată la pasul anterior, în Google APIs Console.

4. Specificarea permisiunilor

Ulterior, pentru a putea utiliza Google Maps Android API, se adaugă o serie de prmisiuni pentru:

- Accesul pachetului la harta Google maps

```
<permission
  android:name="com.example.tcpclient.permission.MAPS_RECEIVE"
  android:protectionLevel="signature" />

<uses-permission android:name="com.example.tcpclient.permission.MAPS_RECEIVE"/>
```

- Internet, pentru acces la serverele Google Maps

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Verificarea stării conexiunii pentru a putea stabili dacă datele pot fi descărcate

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- Autorizarea stocării fragmentelor de hartă Google Map în memoria externă a dispozitivului

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Accesul interfeței la serviciul web Google

```
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

- Accesul interfeței la rețele mobile sau fără fir pentru a determina locația dispozitivului

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

- Accesul interfeței la sistemul GPS pentru determicarea locației dispozitivului pe o suprafață mică

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

5. Solicitarea limbajului OpenGL, versiunea 2

Google Maps Android, versiunea 2 are nevoie de versiunea 2 a limbajului OpenGL, așadar este necesară adăugarea în *AndroidManifest.xml* a elementelor:

```
<uses-feature
     android:glEsVersion="0x00020000"
     android:required="true" />
```

6. Adăugarea hărții în interfețele aplicației

Se adaugă proiectul *google-play-services_lib* în spațul de lucru (Package Explorer) al mediului Eclipse, iar apoi, în proiectele Android care implmentează interfețe grafice cu Google Maps, acesta se adauga ca și bibliotecă, parcurgân următorii pași:

a) Click dreapta pe proiectul care se dorește să integreze o hartă în interfața grafică; se alege opțiunea Properties, conform figurii 4.4.1.10:

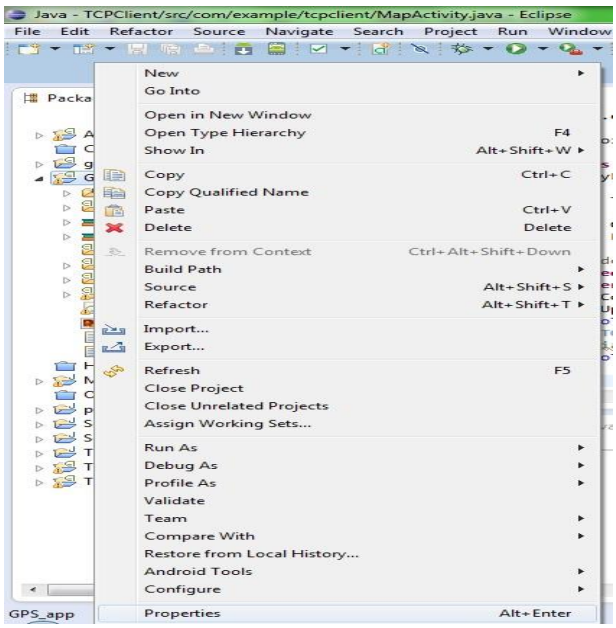


Figura 4.4.1.10 Pas a). Listare bibliotecă

b) Se alege câmpul *Android* din meniul din stânga al ferestrei deschise; se apasă butonul *add*
c) Se selectează din noua fereastră biblioteca *google-play-services_lib*. Cei doi pași sunt ilustrați în figura 4.4.1.11.

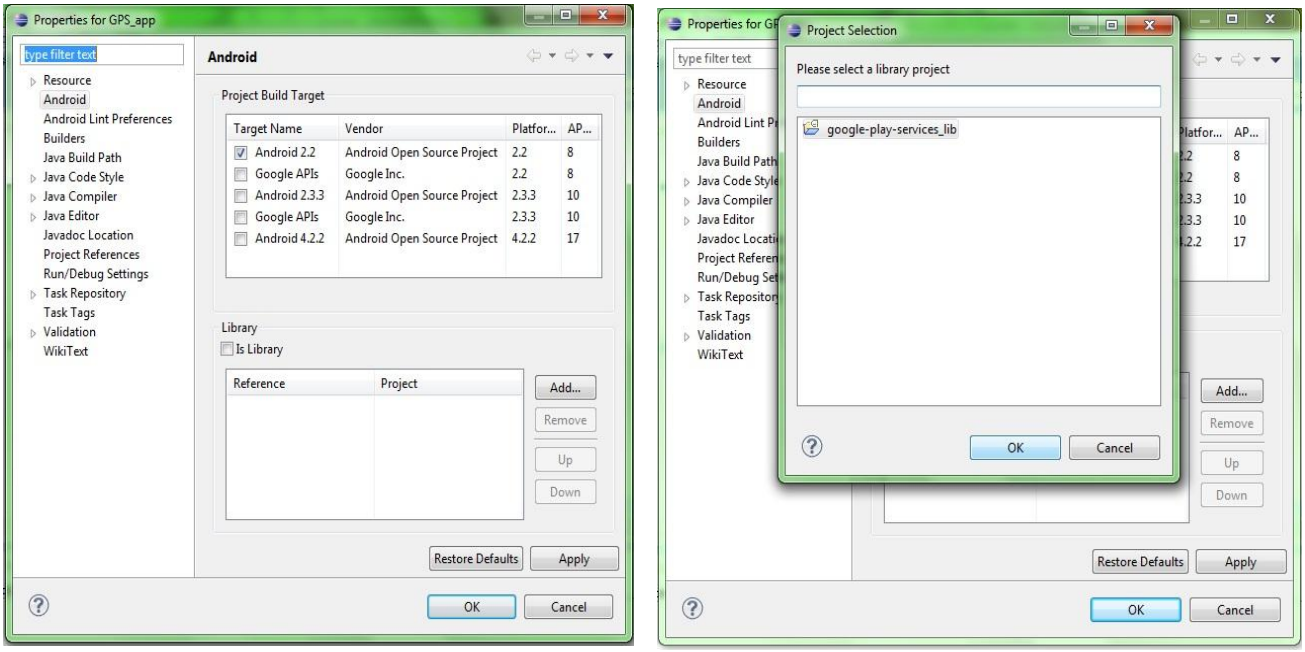


Figura 4.4.1.11 Pașii b) și c). Inserare bibliotecă

Pentru a adăuga harta în aplicație, trebuie modificate fișierele *activity_main.xml* și *MainActivity.java*. În fișierul *activity_main.xml*, trebuie adăugat un fragment, în care va fi încadrată harta Google Maps, pe interfața utilizatorului:

```
<fragment
     android:id="@+id/map"
     android:layout_above="@id/info"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     class="com.google.android.gms.maps.SupportMapFragment" />
```

În fișierul *MainActivity.java*, trebuie suprascrisă metoda *onCreate()*, unde se adaugă referința la fișierul *activity_main.xml*, în care s-a definit fragmentul pentru hartă:

```
@Override
    protected void onCreate(Bundle arg0) {
          super.onCreate(arg0);
          setContentView(R.layout.activity_main);
    }
```

Pentru a permite lucrul cu baze de date, este necesară adăugarea unui pachet JAR în proiectul serverului, care realizează legătura la baza de date. Acest conector de base de date (mysql-connector-java-5.1.25) se descarcă și se dezarhivează în directorul de lucru (opțional).

În continuare trebuie introdus JAR-ul în proiectul Java care se leagă la baza de date. Se selectează cu click dreapta proiectul din spațiul de lucru al mediului eclipse, se alege din meniu câmpul *Build Path* → *Configure Build Path*, iar din fereastra deschisă, se selelctează tab-ul *Libraries*, se apasă butonul *Add External JARs...* și se caută locația unde a fost dezarviat conectorul de baze de date, din care se selectează fișierul de tip Executable Jar (mysql-connector-java-5.1.24-bin) și se apasă butonul *Open*, iar apoi butonul *OK* al ferestrei anterioare, în care trebuie să apară fișierul JAR importat. Pașii descriși în cuvinte sunt ilustrați în figurile 4.4.1.12 și 4.4.1.13.

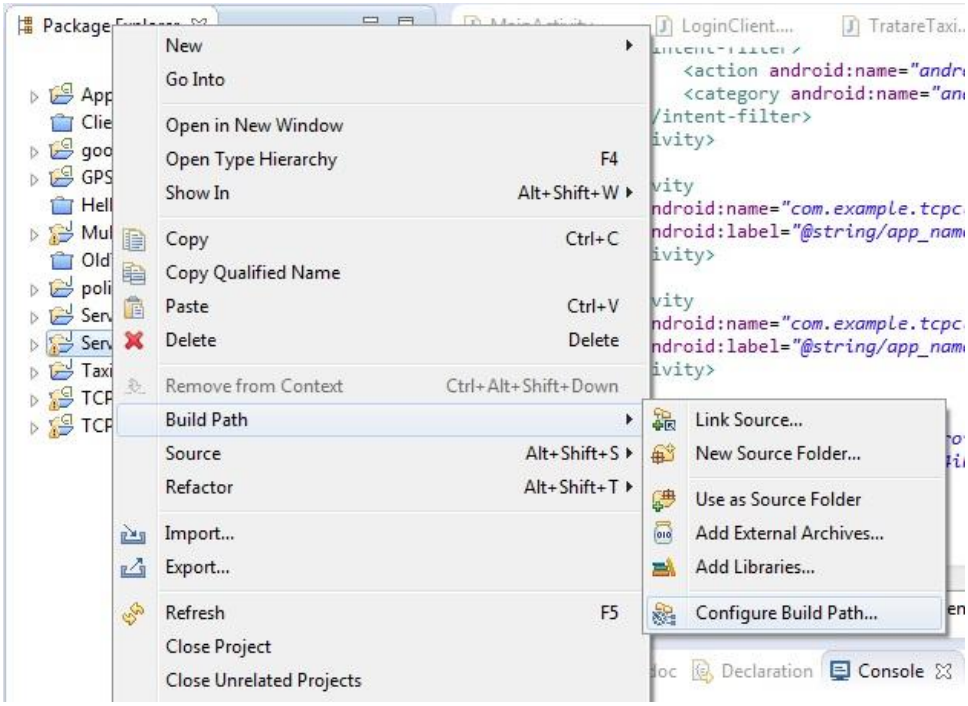


Figura 4.4.1.12 Selectarea mediului prin care se introduce un JAR în proiect

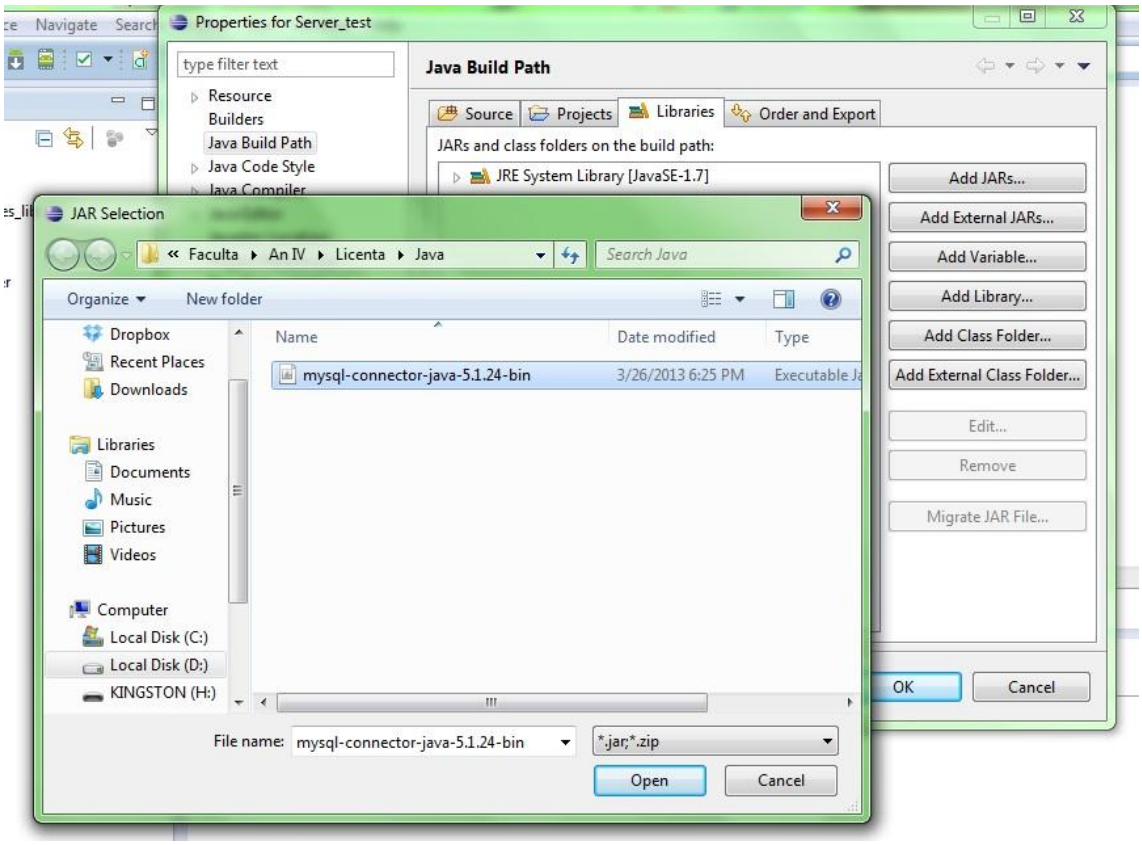


Figura 4.4.1.13 Inserarea în proiect a JAR-ului conector de baze de date

4.4.2 Modelarea algoritmului cu Rețele Petri

Modelul realizat în această lucrare poate fi categorisi ca și **sistem distribut** (pentru că acțiunile care pot fi realizate sunt divizate părților diferitelor procese care intră în componența sistemului: clienți, taxiuri, server) și ca **sistem cu evenimente discrete**, datorită comportării sale dinamice realizată de producerea unor evenimente și care nu sunt dependente de timp, precum și datorită faptului că unele variabile folosite în implementarea sistemului sunt cuantificabile (exemplu, în această aplicație, starea taxiurilor ocupat sau liber).

Această categorisire permite modelarea algoritmului de bază al implememtării cu ajutorul **Rețelelor Petri**, stările proceselor fiind exprimate prin intermediul **locațiilor**, iar evenimentele care au loca între stări și care produc modificarea stărilor, sunt reprezentate de **tranziții**, iar interacțiunea dintre stări și evenimente este asigurată de **arce**.

Prima Rețea Petri, reprezentată în figura 4.4.2.1, arată evoluția pas cu pas a comunicației și interacțiunii dintre client și server, iar tabelul 4.4.2.1 detaliază semnificația fiecărei locații și

tranziții reprezentate în diagramă (celulele verzi se referă la acțiunile realizate de server, iar cele mov descriu comportamentul clientului).

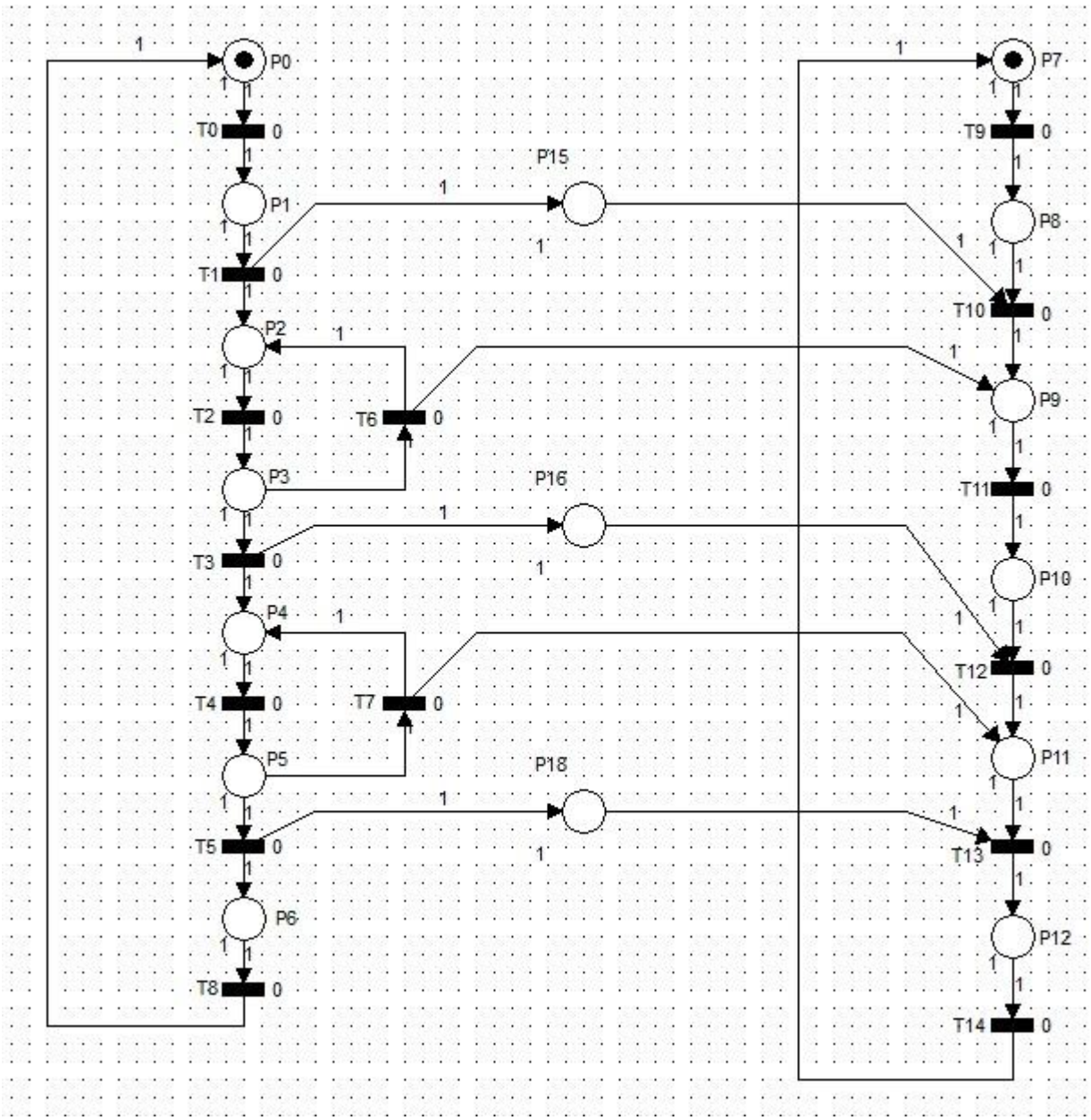


Figura 4.4.2.1 Modelul Client-Server cu Rețea Petri

Tabel 4.4.2.1 Descriere Rețea Petri pentru Client-Server

Locații	Semnificație	Tranziții	Semnificație
P0	Stare inițială server (așteptare după conexiuni)	T0	Are loc o nouă conexiune
P1	Procesare conexiune	T1	Trimitere confirmare conexiune
P2	Așteptare pentru autentificare utilizator	T2	Primire eveniment autentificare
P3	Căutare în BD (validare utilizator)	T3	Acordare permisiune pentru utilizator
P4	Așteptare pentru cerere de taxi	T4	Primire cerere taxi
P5	Căutare în BD (disponibilitate taxi)	T5	Taxi găsit
P6	Adăugare cerere în BD Trimite mesaj la taxi despre noua cerere	T6	Suprimare permisiune pentru utilizator
P7	Stare inițială client	T7	Taxi nu a fost găsit
P8	Așteptare mesaj de permisiune de la server	T8	Răspuns pozitiv
P9	Așteptare pentru acces la interfața de autentificare	T9	Accesare aplicație
P10	Așteprare confirmare de la server pentru autentificare	T10	Primire confirmare conexiune
P11	Așteptarea taxiului	T11	Apăsare buton de autentificare
P12	Așteptare pentru închidere	T12	Trimitere cerere taxi
P15	Confirmare conexiune	T13	Vizualizare informații comandă
P16	Utilizator valabil	T14	Revenire din aplicație
P18	Taxi găsit		

A doua Rețea Petri, ilustrată în figura 4.4.2.2, furnizează informații utile legate de relația dintre taxi și server, bazată pe interschimbarea mesajelor, iar tabelul 4.4.2.2 prezintă semnificația locațiilor și tranzițiilor din rețea (celulele de culoare verde descriu comportamentul serverului, iar cele mov precizează acțiunile tipice taxiului).

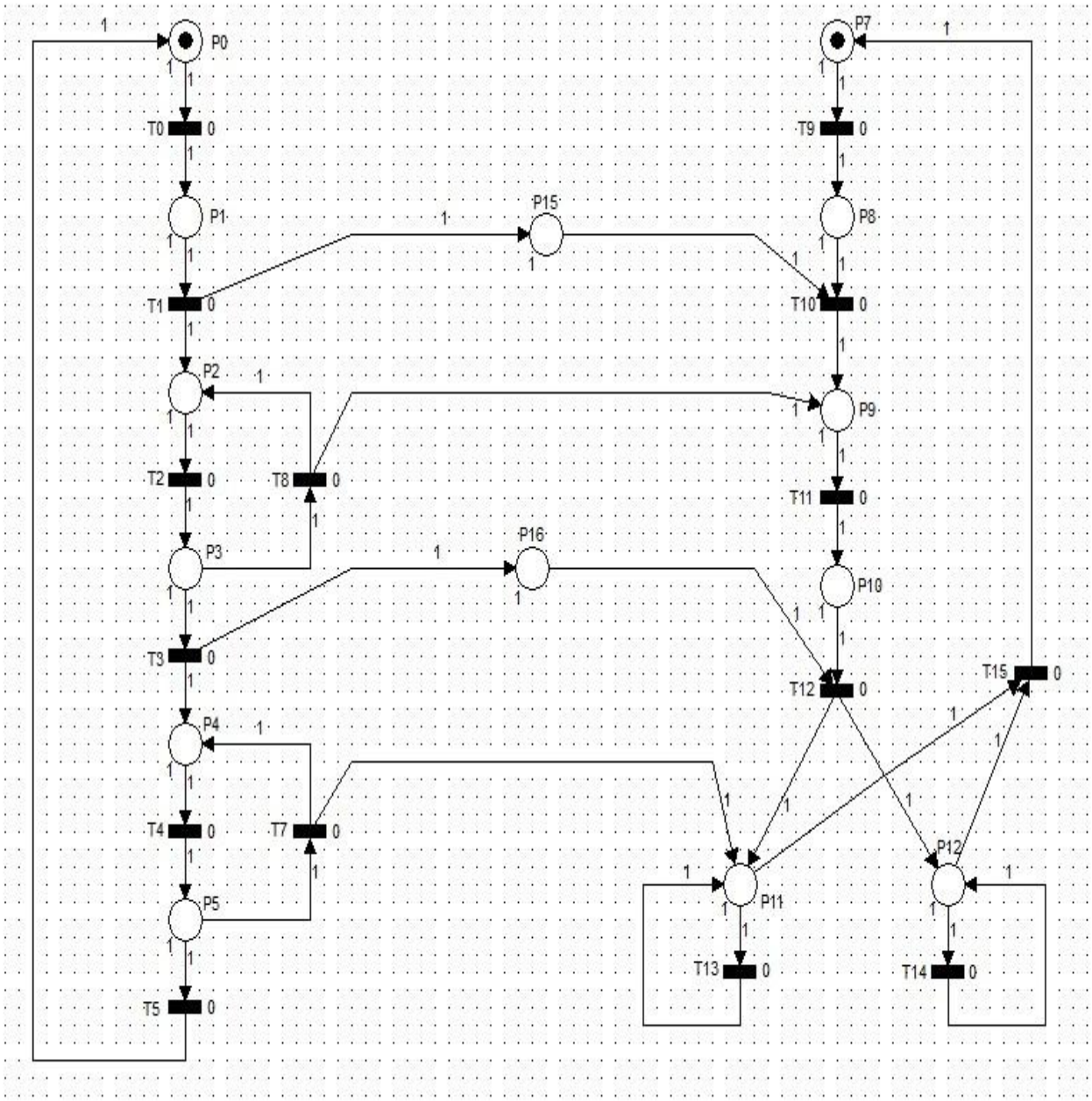


Figura 4.4.2.2 Modelul Taxi-Server cu Rețea Petri

Tabel 4.4.2.2 Descriere Rețea Petri pentru Taxi-Server

Locații	Semnificație	Tranziții	Semnificație
P0	Stare inițială server (așteptare după conexiuni)	T0	Are loc o nouă conexiune
P1	Proceasare conexiune	T1	Trimitere confirmare conexiune
P2	Așteptare pentru autentificare taxi	T2	Primire eveniment autentificare
P3	Căutare în BD (validare taxi)	T3	Acordare permisiune pentru taxi
P4	Așteptare pentru coordonatele GPS ale taxi-ului	T4	Primire coordonate taxi
P5	Actualizare tabel taxi_info în BD Căutare în BD a unei comenzi pentru taxiul actual	T5	Trimite comandă taxi
P7	Stare inițială taxi	T7	Nicio comandă pentru taxiul actual
P8	Așteptare mesaj de confirmare de la server	T8	Suprimarea permisiunii pentru taxi
P9	Așteptare pentru interfața de autentificare	T9	Accesare aplicație
P10	Așteptare confirmare autentificare de la server	T10	Primire confirmare conexiune
P11	Așteptare informații despre comandă de la server	T11	Apăsare buton de autentificare
P12	Așteptare confirmare de la server	T12	Trimitere coordonate GPS
P15	Confirmare de conexiune	T13	Actualizare
P16	Taxi valabil	T14	Reîmprospătare
		T15	Close connection

Rețelele Petri au fost proiectate și simulate cu ajutorul mediului *HPSim*. Rezultatele simulărilor demonstrează faptul că Rețelele Petri sunt viabile și mărginite, fapt care vine în sprijinul algoritmului de implementare.

Pentru realizarea comunicației prin protocolul TCP/IP, prin socketuri, în cazul clienților, este utilizată o secvență de cod în care se crează socketul de comunicare, bufferele pentru interschimbarea mesajelor între client și server, citire mesaj de la client prin bufferul de intrare și,

după procesare, trimitere răspuns clientului prin bufferul de ieșire, după care acestea sunt golite și închise.

```
private Socket socket;
public void run() {
	try {
		BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
		PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())), true);
		System.out.println("IO buffers created, waiting for transmission...");

		String line = "";
		line = in.readLine();
		try {
			...
			out.println("true");
		} catch (Exception ex) {
			ex.printStackTrace();
		}
		out.println("ECHO " + line);
		out.println("END");
		out.flush();
		out.close();
		in.close();
		} catch (Exception e) {
			e.printStackTrace();
		} finally {
			try {
				socket.close();
			} catch (IOException e) {
				System.err.println("Socket not closed");
			}
		}
	}
```

Pe partea de server, realizarea conexiunilor de comunicație prin socketuri este realizată în mod diferit față de cea pentru clienți, fiind necesară crearea obiectului serverSocket prin precizarea portului de comunicare, așteptarea conexiunilor pe socketul realizat, citire și scriere mesaje prin bufferele (zonele de memorie tampon) de intrare-ieșire create.

```
ServerSocket ss=null;
Socket socket=null;

  try{
      String line="";
      ss = new ServerSocket(PORT);
      socket = ss.accept();

      BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
```

```
      PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())),true);

      while(!line.equals("END")){
        line = in.readLine();

        out.println("ECHO "+line);
      }
      out.println("END");
      out.flush();
      out.close();
      in.close();

       }catch(Exception e){e.printStackTrace();}
       finally{
             ss.close();
             if(socket!=null) socket.close();
             }
       }
```

Conectarea la baza de date, prin JDBC, este, realizată într-o secvență de cod în care se efectuează încărcarea driverului pentru baza de date (pachetul com.mysql.jdbc.Driver) și conectarea la baza de date a aplicației, după care pot fi realizate interogări pe tabelele din baza de date, prin intermediul unui statement.

```
String[] token = line.split(" ");
try {
       Class.forName("com.mysql.jdbc.Driver");
       Connection conn = DriverManager
       .getConnection("jdbc:mysql://localhost:3306/aplicatiedb?user=root");

       Statement stat = conn.createStatement();
       ResultSet result = stat.executeQuery("SELECT * FROM client where UserName=\'"
                          + token[0] + "\' and Password=\'" + token[1] + "\';");
       ...
       }
       conn.close();
       } catch (Exception ex) {
             ex.printStackTrace();
       }
```

În cazul clienților, trecerea de la o interfață la alta reprezintă, de fapt, trecerea de la o activitate la alta. Activitățile specifice unui client sunt înmagazinate în același pachet, iar pentru a se efectua trecerea de la una la alta, în implementarea clasei unei activități trebuie introdusă o metodă care să utilizeze un **intent**. Scopul intent-ului este de a realiza trecerea la următoarea activitate, prin apelul metodei *startActivity()*.

```
private void navigateToMapActivity() {
       Intent i = new Intent(this, MapActivity.class);
       startActivity(i);
}
```

Pe interfețele utilizatorilor și taximetriștilor apar butoane, căsuțe de text, fragmente de hartă, etc. Acestea sunt declarate în fișierele activity_main.xml și sunt apoi folosite în clasele MainActivity.java .

Declararea unui obiect folosit pe interfața utilizatorului în fișierul activity_main.xml arată în felul următor:

```
<Button
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="16dp"
        android:layout_marginTop="18dp"
       android:text="Login" />
```

Referința butonului din cadrul fișierului MainActivity.java este realizată cu ajutorul metodei *findViewById()*:

```
butLogin = (Button) findViewById(R.id.login_button);
```

Pentru implementarea operațiilor care se execută la apăsarea butonului, se declară un **listener**, care tratează evenimentul de apăsare al butonului:

```
btnLogin.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View arg0) {
            boolean succesfulLogin = startTCP();
            if(succesfulLogin) {
            navigateToOrderActivity();
            }
      }
});
```

Capitolul 5. TESTARE ȘI REZULTATE

5.1 Metode de testare

5.1.1 Emulatorul Android

Android SDK conține un emulator pentru dispozitivele mobile, adică un dispozitiv mobil virtual care rulează pe calculatorul dezvoltatorului. Acest emulator permite dezvoltarea și testarea aplicațiilor Android, fără a fi nevoie de folosirea unui dispozitiv fizic cu sistem de operare Android.

Emulatorul Android imită toate trăsăturile hardware și software ale unui dispozitiv mobil tipic, exceptând faptul că nu se pot realiza apeluri telefonice. Asigură o varietate de taste de navigare și control, care sunt apăsate prin intremediul mouse-ului sau al tastaturii calculatorului, generând evenimente pentru aplicație. Dispune, de asemenea, de un ecran, pe care e afișată aplicația, împreună cu orice altă aplicație Android activă. În figura 5.1.1.1 este reprezentat un model de emulator Android.

Figura 5.1.1.1 Emulatorul Android

Pentru a permite modelarea și testarea mai ușoară a aplicației, emulatorul utilizează configurații ale unui dispozitiv virtual (AVD = Android Virtual Device). Acestea permit definirea unor anumite aspecte hardware ale telefonului emulat și permite utilizatorului să creeze mai multe configurații pentru a testa platforme Android și permutări hardware. Odată ce aplicația rulează pe emulator, aceasta se poate folosi de platforma Android pentru a invoca alte aplicații, pentru a accesa rețeaua, interpreta fișiere audio și video, stoca și recupera date, notifica utilizatorul și a reda transmisii și teme grafice [1].

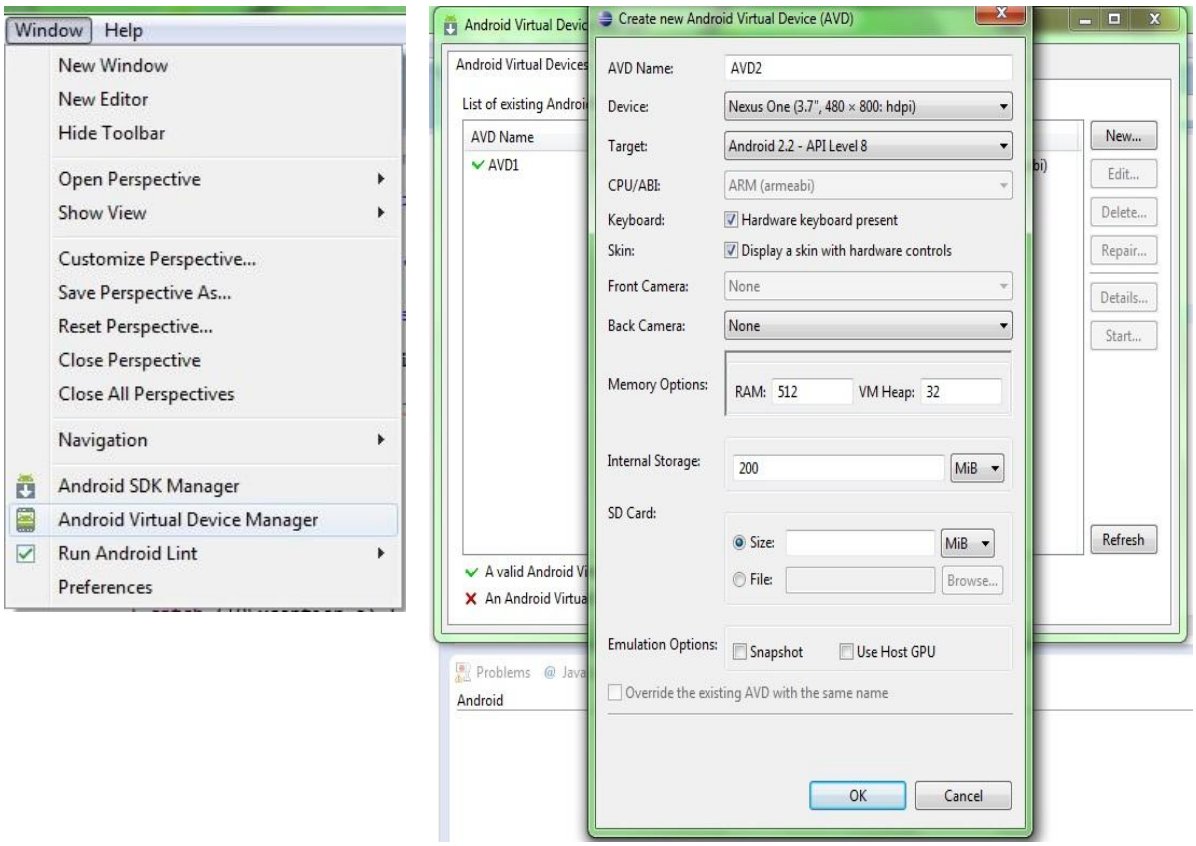


Figura 5.1.1.2 Crearea unui nou AVD pentru simulare

În mediul de programare Eclipse, crearea unui nou AVD se realizează conform figurii 5.1.1.2: Window → Android Virtual Device Manager → New. În fereastra Create New AVD, se completează câmpurile corespunzătoare numelui AVD, tipul dispozitivului (Nexus, WVGA etc.) și ținta (varianta de Android – în cazul proiectului de față s-a folosit versiunea Android 2.2 API 8 Froyo). Odată creat și confingurat AVD-ul, interfața Android poate fi rulată cu ajutorul emulatorului.

Această metodă de testare a funcționalității programului a fost utilizată în prima parte a implementării sistemului, pentru verificarea corectitudinii conexiunilor realizate între modulele aplicației (comunicarea prin socketuri între clienți, taxiuri și serverul central, precum și legătura

stabilită între server și baza de date prin JDBC). Astfel, se trimiteau mesaje de confirmare a conexiunilor și realizării fluxurilor de trimitere a mesajelor, vizibile atât pe interfața emulatorului, aparținând clienților, cât și pe consola mediului eclipse, corespunzătoare serverului. Principalele dezavantaje ale emulatorului sunt greutatea ce care se încarcă, mai ales când aplicația este rulată pentru prima dată și faptul că interfața Google Maps Api v2 nu poate fi încărcată pe emulator, afișând mesajul din figura 5.1.1.3. Din acest motiv se utilizează dispozitivele mobile cu sistem de operare Android.

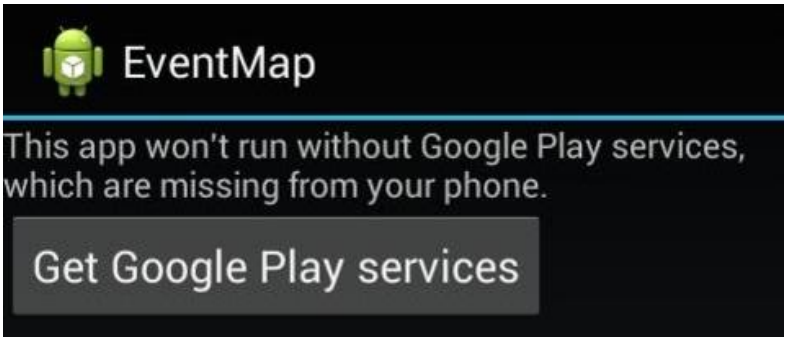


Figura 5.1.1.3 Eroare de încărcare a Google Maps API pe emulator

5.1.2 Dispozitiv mobil

Pentru a putea rula întreaga aplicație, care are clienții implementați cu interfețe Android ce folosesc Google Maps API, este nevoie de folosirea dispozitivelor mobile cu Android OS. Dispozitivul folosit pentru testarea aplicațiilor Android din acest proiect este smartphone HTC Rhyme Gold, pentru care este necesară descărcarea unui driver și realizarea unor pași pentru configurare, pași prezentați în tutorialul [40].

1. Se descarcă driverul, se rulează și se instalează pe calculator, pentru a fi compatibil cu dispozitivul cu care va intra în contact prin USB.
2. Pe telefon, se activează modul de depanare USB, parcurgând calea: →*USB debugging*, se bifează opțiunea, se apasă butonul OK, iar apoi trebuie să apară mesajul *USB debugging connected.*
3. După ce instalarea driver-ului a luat sfârșit, dispozitivul poate fi conectat prin USB, iar când aplicația Android va fi startată în eclipse, acesta va recunoaște automat dispozitivul și va rula programul pe telefon, în loc să îl ruleze pe emulator. În figura xx este reprezentat felul în care arată meniul de notificări când e activ USB debugging mode.

Figura 5.1.2.1 Modul USB debugging activ

O altă condiție necesară ca aplicația să poată fi rulată pe dispozitivul mobil, este ca atât calculatorul, cât și dispozitivul/dispozitivele implicate să fie conectate la aceeași sursă de Internet, și anume la același router wireless. În fiecare rețea, IP-ul calculatorului se schimbă, iar pentru că serverul rulează pe caclulator, acesta îi preia IP-ul, fiind necesară schimbarea în cod a acestui parametru, pentru ase putea realiza conexiunea între module.

```
Socket client = null;
client = new Socket("192.168.1.123", 1902);
```

5.2 Cazuri de testare

După ce aplicația a fost descărcată în telefon, aceasta poate fi rulată ori de câte ori e nevoie, atâta timp cât dispozitivele clientului și al taxiului sunt conectate în rețea cu mașina serverului.

Așadar, utilizatorul accesează aplicația pentru client (TCPClient), de unde trebuie să completeze câmpurile UserName și Password, iar după ce acestea sunt introduse, se apasă butonul *Login*. În cazul în care s-a introdus parola sau numele utilizatorului greșit, aplicația nu permite trecerea mai departe spre comanda taxiului. Interfața pentru autentificarea clientului este prezentată în figura 5.2.1.

Figura 5.2.1 Interfața autentificare client

În următoarea interfață, clientul trebuie să selecteze un punct pe hartă care să reprezinte destinația dorită, iar apoi să apese butonul *Plasati Comanda*. Dacă nu a fost selectată niciun punct pe hartă, aplicația consideră că nu s-a ales destianția, astfel că nu va permite clientului să depuna cererea de taxi, afișându-i mesajul ”Nu avem destinația ta!”, iar în cazul în care GPS-ul dispozitivului nu trimite coordonatele GPS (latitudinea și longitudinea), serverul nu permite din nou clientului să plaseze comanda, afișând mesajul ”Nu avem locația ta!”. Interfața pentru plasarea comenzii este reprezentată în figura 5.2.2.

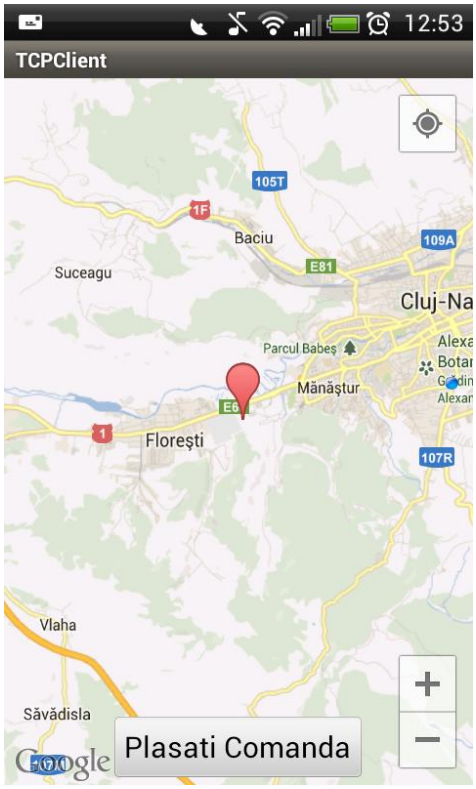


Figura 5.2.2 Interfața plasare comandă

În cazul plasării cu succes a comenzii, clientului îi apare următoarea interfață, în care primește informații legate de comandă, și anume timpul și distanța pe care o va parcurge cu taxiul pentru a ajunge la destinație. Această ultimă interfață este afișată în figura 5.2.3.

Figura 5.2.3 Interfața informații comandă pentru client

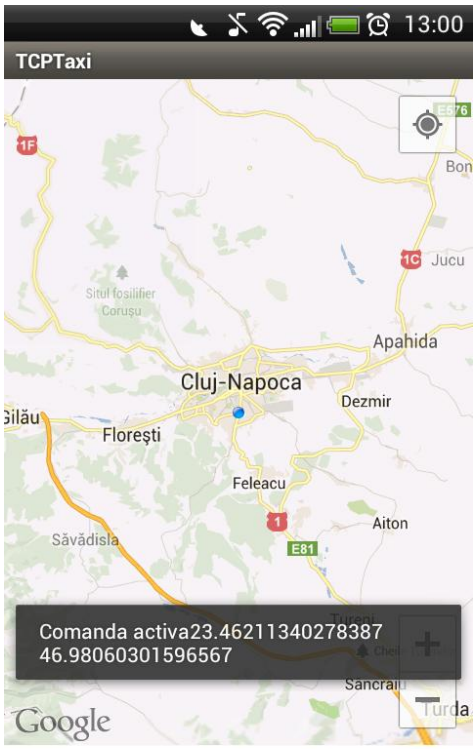
Pe consola din eclipse apar, de asemena, mesaje despre conexiunea serverului cu clientul: mesajul ”Server on! Waiting for connections...” – apare pentru a arăta că serverul a pornit și așteaptă conexiuni de la clienți. În momentul în care un client accesează aplicația, apare un mesaj de conectare a clientului, și adresa socketul pe care acesta comunică cu serverul, respectiv portul de comunicare. Când a fost efectuată comanda, id-ul clientului (al dispozitivului, generat în cod) este trimis serverului de tratare a clientului, împreună cu coordonatele locației utilizatorului și a coordonatelor destinației alese (latitudine și longitudine). Aceste mesaje sunt reprezentate în firgura 5.2.4, fiind o captură a consolei serverului de tratare a clientului.

Figura 5.2.4 Consola Server tratare client

În ceea privește taxiurile, și ele trebuie sa acceseze aplicația corespunzătoare lor (TCPTaxi), să se autentifice, completând câmpurile UserName și Password, apăsându-se, ulterior, butonul *Login*. La fel ca și în cazul clienților, aplicația nu permite trecerea mai departe, decât în momentul unei autentificări realizate cu succes. Interfața de autentificare a taxiului este asemănătoare cu cea a clientului și este ilustrată în figura 5.2.5.

Figura 5.2.5 Interfață autentificare taxi

După realizarea cu succes a autenrificării, taxiul accesează a doua interfață, în care are o hartă pe care îi este specificată, printr-un marker, poziția clientului care a realizat comanda, apărandu-i și un mesaj ”Comanda on”, alături de un pop-up (toast), cu mesajul ”Comanda activă”, urmată de coordonatele GPS ale destinației alese de client. Această interfață este reprezentată în figura 5.2.6.



Comanda on

Figura 5.2.6 Interfața informații comandă pentru taxi

Și în cazul taxiului apar mesaje de conxiune pe consola serverului, și anume mesajul de înștiințare că serverul e pornit și așteaptă conexiuni (”Server on! Wainting for connections...”). În momentul în care un taxi accesează aplicația, apar mesaje de conectare a unui taxi (client pentru propriul sau server): adresa soketului și numărul portului, împreună cu mesajul de creare al bufferelor de interschimbare a mesajelor (InputStream și OutputStream).

Taxiul începe să își trimită continuu coordonatele serverului, pentru a le stoca în baza de date, acestea fiind afișate și în consolă, împreună cu id-ul taxiului, apărând mesajul că taxiul cu respectivul id s-a conectat la server. În continuare mai apare un mesaj care specifică id-ul taxiului care a acceptat comanda, și anume taxiul aflat cel mai aproape de locația clientului care a realizat comanda. Consola serverului de taxi este afișată în figura 5.2.7.

Figura 5.2.7 Consola Serverului tratare taxi

Capitolul 6. CONCLUZII

6.1 Rezumat contribuții

Prezenta lucrare de diplomă e concentrează pe implementarea unei aplicații proiectată pentru dispozitivele cu Android SO. Pentru o mai simplă abordare a problematicii studiate pentru realizarea acestui sistem, descrierea funcționalității sitemului a fost realizată prin intermediul diagramelor de utilizare, în starea incipientă a proiectului. Pe parcurs s-a putut obține o modelare a algoritmului general al aplicației, cu ajutorul Rețelelor Petri, simulate și testate în mediul HPSim, rezultând a fi o soluție viabilă. Odată cu atingerea anumitor puncte din dezvoltarea sistemului, au putut fi realizate diagramele secvențiale, care descriu fluxul de mesaje schimbate între modulele componente, iar în final, a fost trasată diagrama claselor, cu descrierea rolului fiecărei clase din structura diagramei.

6.2 Realizarea obiectivelor

Aplicația realizată vine ca și o soluție alternativă a metodelor clasice de rezervare a taxiurilor. Permite conectarea clientului la serverul central, plasarea unei cereri către o anumită destinație, asigurând asignarea celui mai apropiat taxi pentru a onora comanda.

În urma testării aplicației, se observă ca ea asigură o conexiune bine realizată între clienți și serverul central, respectiv o bună comunicare, clientul putând realiza comenzi oricând, atâta timp cât e conectat la aceeași rețea cu calculatorul pe care rulează serverul. Comenzile sunt preluate de server, analizate, iar în momentul în care clientului i se oferă confirmarea pentru comandă, primește și informații suplimentare despre comandă. Taxiurile, la rândul lor, în urma trimiterii coordonatelor GPS ale poziției actuale, primesc răspuns un mesaj în care sunt înștiințate dacă sunt sau nu comezi rezervate pentru respectivul taxi.

6.3 Posibilități de dezvoltare ulterioară

Proiectul propus pentru această lucrare de diplomă are multe avantaje, însă, întotdeauna vor exista funcții care pot fi aduse în plus sau aspecte care pot fi îmbunătățite, algoritmi care pot fi eficientizați etc., care ar duce la o sporire a eficacității aplicației.

În viitor, se urmărește ca aplicația să fie dezvoltată din mai multe puncte de vedere, în primul rând prin afișarea câtor mai multe informații exacte pe interfețele care utilizează harta Google Maps, alături de date numerice suplimentare: vizualizarea pozițiilor clientului și a taxiului, estimarea unui timp cât mai exact de așteptare a taxiului, trasarea pe hartă a întregii rute parcurse de utilizator cu taxiul, memorarea traseelor parcurse, pentru a fi capabil să determine ruta cea mai scurtă, etc.

Ar putea fi propus un studiu serios bazat pe comparația între sistemul clasic de comandă a taxiurilor și comanda prin intermediul aplicațiilor mobile, cu ajutorul căruia să fie observate și alte beneficii ale utilizării unor astfel de siteme, pe lângă economisirea banilot și a timpului.

Ca și funcționalități suplimentare, ar putea fi luate în calcul posibilitatea de plată a prețului cursei cu cardul de credit al clientului, în cazul în care acesta își divulgă informațiile necesare pentru realizarea tranzacțiilor. Ar mai putea fi propuse multe alte funcționalități noi, cum ar fi: creșterea numărului de stări ale unui taxi (liber, ocupat, alocat, stricat), posibilitatea urmăririi tuturor taxiurilor pe harta disponibilă în aplicație, a alegerii unui taxi dintr-o listă, în funcție de ce parametri se doresc (preț, companie, condiții, timp etc.), a anulării unei comenzi, în cazul în care se realizează mai multe rezervări de taxi și multe alte propuneri asemănătoare. Așadar, se observă că această aplicație poate deveni foarte complexă, dar mai ales utilă și convenabilă, datorită economisirii multor resurse prețioase din viața noastră, printre care timpul și banii.

BIBLIOGRAFIE

[1] Shane Conder, Lauren Darcey – Android Wireless Application Development, Addison-Wesley Professional Publishing; second edition, December 25 2010, ISBN 9780321619662

[2] Wei-Meng Lee - Beginning Android 4 Application Development, John Wiley & Sons Publishing, 8 April 2011. ISBN 9781118017111, http://it-ebooks.info/book/1313/

[3] http://developer.android.com

[4] Satya Komatineni, Dave MacLean – Pro Android 4, Apress Publishing, 1 edition, 6 Martie 2012, ISBN 9781430239307, http://it-ebooks.info/book/860/

[5] Bruce Eckel - Thinking in Java, Prentice Hall Publishing, Fourth Edition, 2006, ISBN 0131872486, http://mindview.net/Books/TIJ4

[6] http://www.hit.ro/stiinta-generala/5-aplicatii-excelente-pentru-navigatie-GPS-foto-si-video/1/?

[7] http://www.hit.ro/stiinta-generala/5-aplicatii-excelente-pentru-navigatie-GPS-foto-si-video/2/?

[8] http://www.hit.ro/stiinta-generala/5-aplicatii-excelente-pentru-navigatie-GPS-foto-si-video/3/?

[9] http://www.hit.ro/stiinta-generala/5-aplicatii-excelente-pentru-navigatie-GPS-foto-si-video/4/?

[10] http://www.hit.ro/stiinta-generala/5-aplicatii-excelente-pentru-navigatie-GPS-foto-si-video/5/?

[11] https://taximagic.com/en_US

[12] http://www.mytaxi.com/en/home.html

[13] https://play.google.com/store/apps/details?id=net.antrix.android.cabbie&hl=en

[14] http://www.startaxi.ro/

[15] http://www.clevertaxi.com/ro/

[16] http://docs.oracle.com/javase/tutorial/

[17] K. Barclay, J. Savage – Object-Oriented Design with UML and Java, Elsevier Publishing, Publicat Decembrie 2003, ISBN 9780750660983, http://dx.doi.org/10.1016/B978-075066098-3/50000-3

[18] Simon Kendal – Object Oriented Programming using Java, Editura Bookboon, Publicat 2009, ISBN 9788776815011, http://books.google.ro/books?id=HWC4ZjXs1V8C.

[19] Dung Nguyen, Stephen Wong, Mark Husband – Principles of Object-Oriented Programming, Editura Connexions, Publicat 2008, ISBN 9781616100629, http://www.cd3wd.com/data/133/Principles_of_Object_Oriented_Programming_cmptr_cnx_x10213_.pdf

[20] Mary C. Lacity, Leslie P. Willcock, Ashok Subramanian, A strategic client/server implementation: new technology, lessons from history, The Journal of Strategic Information Systems, Volume 6, Issue 2, June 1997, Pages 95-128, ISSN 0963-8687, http://dx.doi.org/10.1016/S0963-8687(97)00009-7.

[21] http://control.aut.utcluj.ro/

[22] http://searchnetworking.techtarget.com/definition/GNSS

[23] http://www.icao.int/Meetings/PBN-Symposium/Documents/9849_cons_en[1].pdf

[24] http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

[25] Cornel Păunescu, Sorin Dimitriu, Victor Mocanu – Sistemul Global dePoziționare (GPS): curs, Editura Universității din București, Publicat 2006, ISBN 9737371208, http://www.scribd.com/doc/129703543/Cornel-Paunescu-Sorin-Dimitriu-Victor-Mocanu-%E2%80%93-Curs-GPS

[26] Norman Bonnor – A brief History of Global Navigation Satellite Systems. Journal of Navigation – Cambridge University Press, 65:1-14, Publicat Ianuarie 2012, ISSN 0373-4633, http://dx.doi.org/10.1017/S0373463311000506

[27] Tracy M. L. Brown, Steven A. McCabe, and Charles Wellford - Global Positioning System (GPS) Technology for Community Supervision: Lessons Learned. NOBLIS Technical Report, Aprilie 2007, Număr document 219376, http://www.in.gov/idoc/files/B_M_W_2007_GPS1.pdf

[28] http://www8.garmin.com/aboutGPS/

[29] http://www.gsmarena.com/glossary.php3?term=gps

[30] Jean-Marie Zogg – GPS Essentials os Satellite Navigation : Compedium: Theorie and Priciples of Sattelite Navigation, Overview of GPS/GNSS Systems and Applications, U-blox Online Publication, 2009, ISBN 9783033021396, http://books.google.ro/books?id=F892KQEACAAJ

[31] http://www.gps.gov/applications/

[32] http://www.gsma.com/aboutus/gsm-technology/gprs

[33] http://www.gsmarena.com/glossary.php3?term=gprs

[34] Brahim Ghribi, Luigi Logrippo – Understanding GPRS: the GSM packet radio service, Computer Networks, Volume 34, Issue 5, November 2000, Pages 763-779, ISSN 1389-1286, http://dx.doi.org/10.1016/S1389-1286(00)00127-4

[35] http://www.webopedia.com/TERM/W/Wi_Fi.html

[36] William Lehr, Lee W. McKnight – Wireless Internet access: 3G vs. WiFi?, Telecomunication Policy, Volume 27, Issues 5–6, June–July 2003, Pages 351-370, ISSN 0308-5961, http://dx.doi.org/10.1016/S0308-5961(03)00004-1.

[37] David Axmark and Michael "Monty" Widenius - MySQL 5.6 Reference Manual, August 2010, Revision 22474 Publisher, http://dev.mysql.com/doc/refman/5.6/en/

[38] James Rumbaugh, Ivar Jacobson, Grady Booch – The Unifierd Modeling Language Reference Manual, second edition, July 29 2004, Addison Wesley Publsihing, ISBN 9780321718952, http://msdl.cs.mcgill.ca/people/tfeng/docs/The%20Unified%20Modeling%20Language%20Reference%20Manual.pdf

[39] Grady Booch - The Unified Modeling Language User Guide, October 20 1998, Addison-Wesley Publishing, ISBN 0-201- 57168-4, http://meusite.mackenzie.com.br/rogerio/the-unified-modeling-language-user-guide.9780201571684.997.pdf

[40] Martin Fowler and Kendall Scott - UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd Edition, 1999, Published by Addison-Wesley, ISBN 0- 201-65783-X, http://udp.vilaboa.cl/ramos/dsi2/cd/(eBook)%20Martin%20Fowler%20-%20UML%20Distilled.pdf.

[41] Jinsoo Kim, Seongkyu Lee, Hoyong Ahn, Dongju Seo, Doochun Seo, Jongchool Lee, Chuluong Choi - Accuracy evaluation of a smartphone-based technology for coastal monitoring, Measurement, Volume 46, Issue 1, January 2013, Pages 233-248, ISSN 0263-2241, http://dx.doi.org/10.1016/j.measurement.2012.06.010

LISTĂ DE TABELE

LISTĂ DE FIGURI

ACRONIME

OS – Operating System (Sistem de Operare)

iOS – iPhone OS (Sistem de Operare mobil dezvoltat de Apple Inc.)

GPS – Global Positioning System (Sistem de Poziționare Globală)

WAP – Wireless Application Protocol (Protocol de Aplicații fără Fir)

HTTP – HyperText Transmission Protocol (Protocol de Transmisie pentru Hipertext)

RIM BlackBerry – Research In Motion BlackBerry (Sistem de operare specific mărcii de telefoane BlackBerry)

Windows CE – Windows Embedded Compact (Versiune de sistem de operare de la Microsoft)

Pocket PC – Pocket Personal Computer (Calculator Personal de Buzunar = dispozitiv mobil cu sistem de operare)

PDA - Personal Digital Assistant (Dispozitiv utilizat pe post de Asistent Digital Personal)

OHA – Open Handset Alliance (Alianța de dezvoltare a dispozitivelor mobile)

HTC – High Tech Computer (Corporație de echipamente de telecomunicații)

G1 – First Genereation (Prima Generație de rețea mobilă)

Inc. – Incorporation (Corporație)

SDK – Software Development Kit (Trusa de Dezvoltare Software)

ADT – Android Develeopment Tools (Instrumente de Dezvoltare în Android)

2D, 3D – 2 Dimension, 3 Dimension (reprezentare grafică în 2, respectiv 3 dimensiuni)

MPEG4 – Moving Picture Experts Group (Tip de format video)

AAC – Advanced Audio Coding (Tip de format audio)

JPEG – Joint Photographic Experts Group (Tip de format pentru imagini)

PNG – Portable Network Graphics (Tip de format pentru fișiere imagine)

SGL – Scalable Graphic Library (Bibliotecă în Android pentru reprezentări 3D)

SSL – Secure Sockets Layer (Bibliotecă în Android pentru comunicații în rețea)

OpenGL – Open Graphic Library (Bibliotecă în Android pentru redarea conținutului 2D și 3D pe ecran)

HTML – HyperText Markup Language (Limbaj de Marcare pentru Hipertext)

SE – Standard Edition (Ediție Standard)

JVM – Java Virtual Machine (Mașina Virtuală Java)

DVM – Dalvik Virtual Machine (Mașina Virtuală Dalvik)

DEX – Dalvik Executable files (Formatul fișierelor executabile în Android)

API – Application Programming Interface (Interfață de Programare a Aplicațiilor)

AWT – Abstract Window Toolkit (Trusă cu instrumente pentru lucrul cu interfețe în Java)

Wi-Fi – Wireless Field (Zonă de acces la rețele fără fir)

GSM – Global System for Mobile Communication (Sistem Global de Comunicație între dispozitive Mobile)

EDGE – Enhace Data range for GSM Evolution (Tehnologie de evoluție a rețelei GSM)

WiMAX – Worldwide Interoperability for Microwave Access (Standard de comunicare prin rețele fără fir)

SMS – Short Message Sent (Mesaj Scurt Trimis prin telefonul mobil)

MMS – Multimedia Messaging Service (Serviciu de trimitere a Mesajelor cu conținut Multimedia)

AVD - Android Virtual Device (Dispozitiv Virtual în Android = Emulator)

WVGA – Wide Video Graphics Array (Tip de dispozitive mobile)