MINISTERUL EDUCAȚIEI ȘI CERCETĂRII

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

MeetUp - Aplicație de socializare pe platforma Android

LUCRARE DE LICENȚĂ

Absolvent: **Aura-Luiza GEORGESCU**

Coordonator: **Șef lucr. ing. Cosmina IVAN**

2011

Cuprins

1. Introducere – Contextul proiectului

1.1. Motivare

Apariția aplicațiilor de tip rețele sociale, și în special a site-urilor dedicate promovării interacțiunilor sociale, au cauzat un fenomen social pe Internet. Milioane de utilizatori, dintre care unii nu au deloc sau au puțină experiență anterioare, apelează la site-uri web precum Facebook, Twitter sau Orkut, numai câteva din numărul mare existent la ora actuală, pentru a păstra legaătura cu prietenii lor, pentru a se alătura grupurilor cu interese comune și de a-și publica gandurile, opiniile, recomandările și informații actualizate despre ei.

Pe măsură ce tot mai mulți oameni se alătură și folosesc rețelele sociale, comunitățile virtuale de dezvoltă și interacțiunea socială onlie crește.

În același timp, popularizarea crescândă a dispozitivelor mobile cu tot mai multe caracteristici, în special smartphone-urile, au cauzat un fenomen comparabil în lumea fizică. Pe măsură ce dezvoltarea de programe generalizate devine o realitate în cadrul aplicațiilor conștiente de context care ajută la estomparea liniei dintre lumea fizica și cea virtuală, oamenii se găsesc conectați într-un număr cât mai mare, chiar și atunci când se află în mișcare.

În timp ce ambele lumi cresc împreună cu aplicații precum Foursquare, Gowalla sau Facebool Places, comunitățile sociale online, devin accesibile oferind o platformă pentru interacțiunea cu contactele din rețea, atât timp real cât și fictiv. Astfel, dezvoltarea aplicațiilor sociale se orientează pe dispozitive mobile.

Pentru a construi aplicații sociale care să beneficieze de tehnologiile mobile și de funcționalitățile bazate pe locație, trebuie știut ce tip de informații sociale pot fi explorate într-un mediu mobil, cum acestea pot fi strânse și cum pot fi compuse într-un context social care pot fi utilizate apoi de aplicații pentru a spori experiența socială a utilizatorului.

1.2. Descrierea problemei

Tema lucării de față o reprezintă crearea unei aplicații de interacțiune socială, care să beneficieze de avantajele oferite de serviciile de localizare oferite de Google Maps.

Folosind această aplicație utilizatorul va avea posibilitatea de a păstra legătura cu prietenii săi prin schimb de mesaje, vizualizare de fotografii, adăugarea comentariilor la fotografii.

Un utilizator va putea de asemenea să adauge noi prieteni, după o căutare efectuată în prealabil pe baza numelui. Alegerea utilizatorului din lista returnată va putea fi efectuată cu ajutorul fotografiei de profil a persoanelor returnate în urma interogării.

Fiecare utilizator va putea vizualiza profilul si fotografiile prietenilor și va putea edita profilul propriu precum și fotografiile.

O altă funcționalitate importantă a aplicației este bazată pe folosirea API-ului Google Maps și este reprezentată de localizarea pe hartă a utilizatorilor.

Astfel, un utilizator va putea sa vizualizeze atat locația sa curentă cât și pe cea a prietenilor săi. Un utilizator va putea opta pentru modificarea automată prin GPS sau manuală a poziției sale pe hartă. Orice utilizator va putea să se plimbe pe hartă, să micșoreze sau să mărească vederea hărții și va putea să afle anumite obiective sau străzi prin simpla atingere a ecranului în locația dorită.

Pentru dezvoltarea interacțiunii cu utilizatorul, aplicația mai oferă și funcționalitatea de trimitere de notificări în momentul în care un tuilizator își modifică poziția pe hartă.

Actualizările se vor putea face automat sau manual către prietenii utilizatorului. Se va putea selecta de asemenea și trimiterea de notificări către anumiți utilizatori care se află într-o rază de 1km de poziția actualizată.

1.3. Conținutul lucrării

Conținutului lucrării este organizat după cum urmează: Capitolul 2 va descrie obiectivele generale și specificațiile sistemului, după care în Capitolul 3 va fi prezentat studiul bibliografic efectuat pentru familizarizarea cu tema aplicației. În Capitolul 4 se va analiza sistemul și se vor prezenta tehnologiile folosite pentru dezvoltare, iar în Capitolul 5 se va trece la proiectarea și implementarea sistemului. În cadrul Capitolului 6 se vor enumera testele efectuate pentru validarea sistemului, după care Capitolul 7 va descrie un manual de instalare și utilizare. În Capitolul 8 se vor prezenta concluziile lucrării care vor cuprinde descrierea realizărilor și prezentarea unor posibile dezvoltări ulterioare.

2. Obiectivele proiectului

2.1. Obiective generale

Obiectivele urmărite în această lucrare sunt:

- *Oferirea unei interfețe plăcute și ușor de utilizat*
  Se va crea o interfață simplă, ușor de învățat, utilizat și memorat.
- *Asigurarea interconectării utilizatorilor*
  Utilizatorii vor putea păstra legătura în orice moment prin trimitere de mesaje, comentarii, notificări.
- *Păstrarea datelor consistente*
  Datele din baza de date se vo afla într-o stare consistentă și aduse la zi, iar în cazul apariției unei erori în sistem utilizatorul va fi informat referitor la problemele apărute.
- *Efectuarea de notificări corecte și în timp real*
  Aplicația va oferi servicii de notificar utilizatorului asupra modificării locației prietenilor.
- *Integrarea serviciilor de localizare*
  Pe baza facilităților oferite de telefoanele inteligente cu sistem de operare Android, aplicația va putea oferi informații despre locația curentă a utilizatorului și îl va putea îndruma pe acesta în traseul de urmat până în locația dorită.

2.2. Specificațiile sistemului

2.2.1. Cerințe funcționale

Cerințele funcționale ale sistemului exprimă serviciile oferite de aplicație, si comportamentul sistemului in anumite situații. Pentru îndeplinirea obiectivelor, aplicația trebuie sa permită urmatoarele funcționalitati:

- Înregistrarea noilor utilizatori și trimiterea unui email de confirmare în cazul efectuării cu succes a operației.
- Autentificarea în sistem a utilizatorilor.
- Căutarea prietenilor după numele de utilizator al acestora.
- Adăugarea de noi prieteni.
- Vizualizarea profilurilor prietenilor.
- Vizualizarea si editarea profilului propriu.
- Încărcarea sau ștergerea de fotografii.
- Adăugarea de comentarii atat la propriile fotografii cât și la cele ale prietenilor.
- Trimiterea de mesaje către prietenii din listă.
- Modificarea poziției pe hartă.
- Selectarea modalității de trimitere a notificărilor referitoare la poziția actuală: automat sau manual.
- Selectarea grupului de persoane către care se dorește trimiterea notficării.

2.2.2. Cerințe non-funționale

Cerințele non-funcționale reprezintă calități pe baza cărora este evaluat sistemu. Acestea pot fi împărțite în două categorii:

2.2.2.1. Calități de execuție

- **Utilizabilitate**: sistemul trebuie să poată fi ușor învătat prin intermediul unei interfețe simple care să permită utilizatorilor o navigare predictibilă in cadrul aplicației.
- **Disponibilitate**: sitemul trebuie sa fie accesibil oricând, punând la dispoziția utilizatorului toate informațiile necesare și intreaga funcționalitate a aplicației.
- **Securitate**: datele transmise în cadrul aplicației sunt sigure, iar intrarea în cadrul sistemului se poate face doar pe baza autentificării.
- **Acces concurent**: sistemul trebuie să permită accesul la servicii mai multor utilizatori în același timp.
- **Timp mic de raspuns**: cea mai complexă operație din sistem nu trebuie să depășească 5 sec.

2.2.2.2. Calități de evoluție

- **Testabilitate**: sistemul poate fi ușor testat în vederea dezvoltării ulterioare.
- **Mentenanță**: ușurința cu care se pot executa servicii de mentenanță asupra sistemului, permițând corectarea defecțiunilor in timp ce sunt îndeplinite cerințele funcționale.
- **Extensabilitate**: sistemul permite adăugarea de noi funcționalități cu ușurință fără a afecta serviciile existente.
- **Scalabilitate**: sistemul permite accesul in cadrul aplicației unui număr mare de utilizator fără a afecta calitățile sistemului cum ar fi timpul de raspuns sau securitatea.
- **Utilizare redusă a memoriei**: având în vedere că aplicația ruleaza pe un dispozitiv mobil, trebuie optimizat consumul de memorie prin delegarea efectuării operațiilor complexe către sever.

2.2.3. Cerințe de dispozitiv

În timp ce Android este proiectat să asigure suport pentru o varietate de platforme și configurații hardware, această secțiune enumeră crințele minime recomandate de dsipozitiv.

Caracteristică	Cerințe minime	Observații
Chipset	Bazat pe ARM	Pentru prima lansare, Android este îndreptată în principal către porțiuni ale platformei, cum ar fi Dalvik VM de procesare grafică, presupunând o arhitectura ARM.
Memorie	128 MB RAM; 256 MB Flash Extern	Android poate boota și rula în configurații cu memorie mai puțină, dar nu este recomandat.

Stocare	Mini sau Micro SD	Nu este necesar pentru operații de bază, dar este recomandat.
Ecran	QVGA TFT LCD sau mai mare, culori pe 16 biți sau mai mare	Interfața curentă Android se axează pe un ecran tactil HVGA, cu o interfață de atingere de dimensiune mai mare decât 2.8 inchi.
Taste de navigare	5 taste aplicație cu 5 direcții, contoale pentru cameră, putere și volum	
Cameră	2MP CMOS	Nu e necesară pentru fucnționalitatea de bază.
USB	Interfață Standard mini-B USB	Android folosește interfața USB pentru flash-ul imaginilor dispozitivului.
Bluetooth	1.2 or 2.0	Nu e necesar pentru funcționalitate de bază.

Tabelul 2.1 Cerințe de dispozitiv

Pentru aplicația curentă dispozitivul Android mai trebuie sa dispună de următoarele caracteristici:

- Tastatura querty
- WiFi – pentru a accesa serviciile web
- GPS - pentru a beneficia de opțiunea de actualizare automată a poziției.

3. Studiu bibliografic

3.1. Rețele de socializare

O rețea de socializare reprezintă o structura sociala formată din indivizi sau organizații numite noduri, interconectate pe baza tipului de relație dintre acestea.

Transpunerea conceptului de retea socială în cadrul aplicațiilor software presupune utilizarea unor unelte interactive și de comunicare.Uneltele de comunicare gestioneaza captarea, depozitarea și prezentarea comunicării, atât scrisă cât și audio sau video. Partea interactivă a aplicației se ocupă de inteacțiunile din cadrul unei perechi sau grup de utilizatori. Uneltele interactive se axează pe stabilirea si menținerea conexiunii între utilizatori, facilitand procesul de conversație.

Capabilitățile de sporire ale software-ului social au fost demonstrate încă din aplicațiile de început ale internetului în vederea comunicării, precum e-mail, gruprui de știri, groupware, comunități virtuale etc. În etapa curentă de evoluție a acestor sisteme, aceste unelte colaborative adaugă o capabilitate care agregă acțiunile utilizatorilor conectați în rețea. Acesta ne îndreaptă spre o dinamică puternică care distinge software-ul social de alte unelte de colaborare și spre o componentă puternică a tehnologiei Web 2.0. În faza următoare, experții academici, Constructivismul Social si mișcarea software-ului social open-source se așteaptă să aibă o influență notabilă.

Clay Shirky, un scriitor american ce predă și studiază efectele economice si sociale ale tehnologiilor Internet la Universitatea din New York (NYU), plasează originea termenului de *software social*în discursul lui Eric Drexler (inginer american cunoscut pentru popularizarea nanotehnologiei moleculare) din 1987 despre Sisteme de Publicare Hypertext și despre cum sistemele de acest tip pot asigura software-ul necesar pentru discuții publice, dezvoltare colaborativă, luarea de decizii în grup, filtrare colaborativă etc.

Tehnologiile sociale (conversaționale) reprezintă un termen folosit în cadrul organizațiilor și descriu tehnologiile care permit stocarea si crearea de cunoștiințe prin intermediul scrierii colaborative.

Uneltele de comunicare sunt în general asincrone. Prin contrast, uneltele interactive sunt sincrone, permițând utilizatorilor să comunice in timp real (telefon, video chat) sau aproape sincrone (IM, text chat).

Comunicarea implică acțiuni de vorbire sau scriere, în timp ce interacțiunea presupune interesul utilizatorilor unul față de celălalt ca și indivizi.

Tehnologii emergente

- *Rețele sociale peer-to-peer*

Rețele sociale hibride bazate pe web permit în general utilizatorilor sa împartă blog-uri, fișiere și mesaje instant. Câteva exemple enumeră: Imeem, SpinXpress, Bouillon, Wirehog, Soulseek. De asemenea pot fi amintite și Groove, Collanos, WiredReach si Kerika, ce au funcționalități asemănătoare, dar au la bază relații de colegialitate între utilizatori.

- *Prezența virtuală*

Destul de răspândită, prezența virtuală sau teleprezența înseamnă posibilitatea de a fi prezent în anumite locuri prin intermediul diferitelor tehnologii, cum ar fi radio, telefon, televiziune sau internet.

Într-un context mai restrâns, termenul de prezență virtuală poate desemna locații in cadrul *World Wide Web*, identificabile prin URL. Cei care accesează un site web sunt considerați a fi prezențe virtuale în locații web. Prezența virtuală este un software social în sensul în care oamenii se întâlnesc pe internet din întâmplare sau în mod intenționat.

Comunicarea ubicuuă (în spațiul web) transferă pattern-uri de comportament specifice lumii reale și lumii virtuale către lumea web-ului.

Tipuri de aplicații ce permit socializarea prin intermediul internetului:

- Mesageria instant- permite comunicarea cu alte persoane în timp real.
- Text chat- Utilizatorii se pot alătura unui chat room existent sau poate creea unul nou cu alt topic de discuție, dupa care poate scrie mesaje vizibile celorlalți. De asemenea poate invita alte persoane sa se alăture grupului.
- Forumuri- permit utilizatorilor sa posteze topicuri pe care alții sa le poată revizui. Alți utilizatori pot vizualiza topicul si pot scrie propriile comentarii in mod secvențial. Cele mai multe forumuri sunt publice, permițand înscrierea oricui, dar există și câteva private, care cer membrilor o taxă de înscriere.
- Blog-uri –jurnale online. Utilizatorul va scrie mesaje periodic, permițând vizitatorilor sa adauge comentarii. Temele de discuție includ deseori viața de zi cu zi, pareri politice, subiecte de interes particular sau general.
- Editoare colaborative de timp real – permit editarea simultană de text sau de fișiere media de către diferiți participanți.
- Piețe de predicție.
- Serviciile rețelei sociale – permit utilizatorilor să se întâlneasca online având interese sau hobby-uri comune.
- Motoare de căutare ale rețelelor sociale – reprezintă o clasă a motoarelor de căutare care utilizează rețelele sociale pentru prioritizarea, organizarea sau filtrarea rezultatelor. Există două subclase ale motoarelor de căutare sociale: cele care utilizează rețele sociale în mod explicit, sau care le folosesc în mod implicit.
- Ghiduri sociale – recomandă locuri de vizitat sau conține informații despre anumite locații de interes.
- Librării sociale – permite utilizatorilor să își urmărească obiecte de colecție, cărți si DVD-uri. Utiliatorii își pot împărți colecțiile sau pot face recomandări.

3.2. Analiză comparativă cu sisteme similare

Vor fi studiate trei aplicații mobile: Facebook mobile, Foursquare, Brightkite.

Facebook mobile este o variantă simplificată a rețelei de socializare Facebook care permite utilizatorilor să își creeze un profil, adăugarea altor utilizatori ca prieteni, schimbul de mesaje, incluzând notificarea automată atunci când apare o modificare a profilului. Utiliatorii Facebook trebuie sa se inregistreze inainte de utilizarea site-ului. Ca funcționalități adiționale, utilizatorii se pot alătura grupurilor de interes comun, organizate pe diferite categorii, cum ar fi locul de muncă, școala, sau alte caracteristici. Facebook permite oricărui utilizator cu vârsta peste 13 ani sa devină membru.

Foursquare este o rețea socială bazată pe locație destinată utilizării pe dispozitive mobile. Acest serviciu este disponibil utilizatorilor ce posedă dispozitive care integrează GPS. Această aplicație permite utilizatorilor înregistrați sa comunice cu prietenii lor si să-și modifice locația pe hartă. Utilizatorii primesc puncte sau insigne atunci când ajung într-un punct de interes și pot alege de asemenea să facă publice informații referitoare la poziția lor pe contul de Twitter sau Facebook. Aplicația oferă și funcționalitatea creării unei liste "To Do" pentru utilizare privată, precum și adăugarea de sugestii referitoare la o anumită locație ce poate fi vizualizată de ceilalți utilizatori ai aplicației.

Brightkite este o altă aplicație care permite utilizatorilor să descopere unde se află prietenii din cadrul rețelei în orice moment. Din moment ce nu necesită funcția de GPS, utilizatorii pot trimite updatări serviciului prin mesaje text sau e-mail, pentru a-și modifica profilul cu noua locație, poze și notări. Având setări de securitate pentru a preveni orice formă de urmărire, tot mai mulți utilizatori de Twitter folosesc această rețea pentru își înștiința prietenii de noua locație.

Spre deosebire de Facebook mobile, aplicația de față oferă și servicii de localizare și notificare, dar aplicația Facebook beneficiază de o interfață mai avansată, fiind preluată de la aplicația desktop.

În comparație cu Foursquare, aplicația MeetUp beneficiază de o componentă de socializare mai dezvoltată, în strânsă legătură cu serviciile de localizare și notificare.

Aplicația de care MeetUp se apropie cel mai mult este Brightkite, ambele având ca temă socializarea în cadrul unui grup de prieteni, serviciul de localizare, trimiterea de notificări referitoare la poziția pe hartă a prietenilor, precum și schimbul de poze.

4. Analiză şi fundamentare teoretică

În cadrul acestui capitol este descrisă arhitectura aleasă pentru implementarea aplicației și justificarea alegerii făcute, iar în secțiunile ce urmează analizei sunt prezentate cele mai importante tehnologii utilizate pentru dezvoltare, respectiv platforma Android, serviciile Web și serviciul Java pentru mesaje.

4.1. Arhitectura sistemului

Aplicația este construită pe baza unei arhitecturi client-server. Modelul client-server reprezintă o structură distribuită care împarte atribuțiile între serviciile oferite de server și cererile venite din partea clienților. Deseori, clientul și serverul comunică prin intermediul rețelei, aflându-se pe componente hardware diferite, dar pot exista și în cadrul aceluiași sistem.

Maşina care găzduieşte serverul îşi face publice o parte din resurse, în timp ce clientul face o cerere către server fără să își împartă resursele. Astfel, clienții sunt cei care inițiază sesiunea de comunicare, iar serverul așteaptă primirea cererilor de servicii.

Am ales o arhitectură client-server deoarece:

- Permite stocarea datelor pe server, ținând cont de capacitățile limitate ale dispozitivelor mobile, oferind un control mai bun al securității, permițând accesul la date numai utilizatorilo autorizați.
- Stocarea centralizată a datelor asigură o performanță sporită, datele fiin administrate mai ușor și mai rapid decât în cazul unei aplicații peer-to-peer, in care modificările în cadrul rețelei sunt mari consumatoare de timp.
- Se realizează o delimitare clară a responsabilităților fiecărei componente.
- Se asigură posibilități de extensibilitate și mentenanță, serverul putând fi înlocuit cu ușurință fără a fi nevoie de efectuarea de modificări asupra clienților. De asemenea permite existența unor clienți dezvoltați pe platforme diferite, neexistând o dependență între aceștia.

Această arhitectură are și dezavantaje:

- Congestionarea traficului.
- Creșterea numărului de cereri simultane poate duce la supraîncărcarea serverului.
- În cazul în care serverul nu poate accesa sau executa un serviciu, cererea clientului nu poate fi îndeplinită; pentru această problemă există soluții precum loadbalancing sau servere de replicare.

În cadrul aplicației de față, clientul este reprezentat de un dispozitiv mobil, în timp ce componenta de server este server-ul de aplicație care găzduiește serviciile web. Serviciile sunt apelate de către aplicația client și răspund acesteia cu setul de date cerut. Tot responsabilitatea serverului este de a se ocupa de accesul la baza de date și de a efectua interogările necesare pentru a furniza un răspuns consistent.

Cererile către serviciile web, precum și răspunsul acestora către aplicația client se realizează cu ajutorul mesajelor SOAP. În cadrul aplicației curente, acest lucru a fost posibil prin utilizarea librăriei ksoap2-android, o specializare a librăriei ksoap2 pe platforma Android. Ksoap2 este o librărie folosită pe partea de client pentru servicii SOAP, folosită în medii de dezvoltare Java, precum applet-uri sau J2ME.

Cele două componente ale aplicației sunt reprezentate de două aplicații independente ce pot rula pe dispozitive hardware diferite. Limbajul de programare folosit pentru

implementare este Java. Aplicația client rulează pe un telefon mobil, în timp se serviciile web rulează în cadrul unui server de aplicație, în cazul de față fiind folosit Glassfish3.

Fluxul datelor este următorul: aplicația client apelează serviciul web dorit, iar servciul web se conectează mai departe la o bază de date MySQL, pentru a colecta datele cerute de către client, pe care le trimite apoi aplicației client Android. Acest lucru poate fi observat și in Figura 4.1, care ilustrează componentele aplicației.

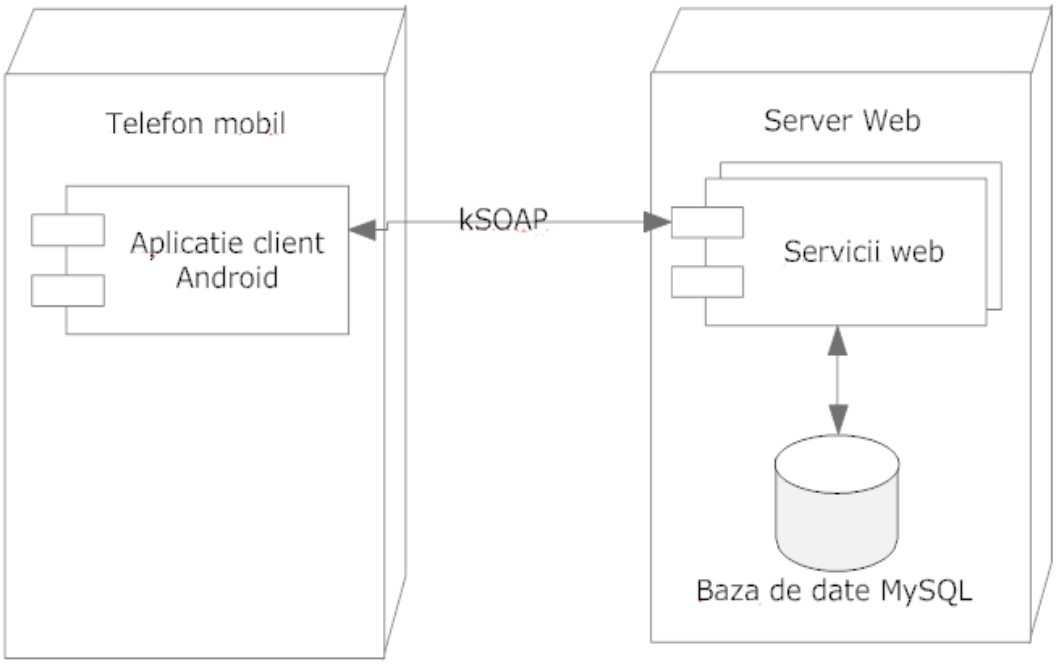

Figura 4.1 Componentele aplicației

4.2. Android

Android este o stivă software pentru dispozitive mobile care include un sistem de operare, un middleware și aplicații cheie. Kitul de de dezvoltare software Android oferă unelte și API-uri necesare dezvoltării de aplicații pe platforma Android, folosind limbajul de programare Java.

4.2.1. Caracteristici

- *Framework de aplicație* – permite reutilizarea și interschimbarea componentelor;
- *Mașina virtuală Dalvik* – optimizată pentru dispozitive mobile;
- *Browser integrat* –bazat pe motorul WebKit open source;
- *Grafică optimizată* – susținută de o librărie de grafică 2D; grafică 3D bazată pe specificația OpenGL ES 1.0 (accelerare hardware opțională);
- *SQLite* – pentru stocarea structurată de date;

- *Suport Media* – pentru formate audio , video și imagini (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF);
- *Telefonie GSM* – dependent de hardware;
- *Bluetooth, EDGE, 3G și WiFi* – dependent de hardware;
- *Cameră, GPS, compass și accelerometru* – dependent de hardware;
- *Mediu de dezvoltare bogat* – inclusiv un emulator de dispozitiv, unelte pentru debugging, profiluri de memorie și performanță, plugin pentru Eclipse IDE.

Aplicatiile Android sunt scrise in Java, iar codul este compilat, alaturi de alte fisiere de date si resurse, intr-un pachet Android, o arhiva cu sufixul .apk. Codul din cadrul unui singur fisier .apk este considerat ca apartinand unei singure aplicatii si reprezinta fisierul pe care dispozitivele care ruleaza pe Android il instaleaza pentru a porni aplicatia.

Odata instalata pe un dispozitiv, fiecare aplicatie Android are o identitate proprie:

- Sistemul de operare Android reprezinta o structura Linux multi-utilizator in care fiecare aplicatie este reprezentata de un utilizator distinct.
- In mod implicit, sistemul asigneaza fiecarei aplicatii un ID utilizator unic (ID-ul este utilizat numai de catre aplicatie si nu este cunoscut de catre aplicatie). Sistemul seteaza permisiuni pentru toate fisierele aplicatiei astfel incat numai utilizatorul cu ID-ul respectiv sa le poata accesa.
- Fiecare proces are propria masina virtuala (VM), astfel incat codul aplicatiei sa ruleze izolat de alte aplicatii.
- Implicit, fiecare aplicatie ruleaza in propriul proces Linux. Android incepe procesul atunci cand o componenta a aplicatiei trebuie sa fie executata, apoi opreste procesul atunci cand nu mai este necesar sau atunci cand sistemul trebuie sa aloce memorie pentru alte aplicatii.

Astfel, sistemul Android implementeaza principiul celui mai putin privilegiat. Aceasta inseamnca ca fiecare aplicatie poate accesa numai componentele de care are nevoie pentru a-si atinge obiectivul si nimic mai mult. Se creaza astfel un mediu foarte sigur in care o aplicatie nu poate accesa parti ale sistemului pentru care nu are permisiune.

Oricum, exista cateva moduri in care o aplicatie sa partajeze date cu alte aplicatii si sa acceseze serviciile sistemului:

- Se poate atribui acelasi ID utilizator pentru doua aplicatii, caz in care fiecare poate accesa fisierele celeilalte. Pentru a pastra resursele sistemului, aplicatiile cu acelasi ID pot rula in acelasi proces Linux, si pot imparti aceeasi masina virtuala.
- O aplicatie poate cere permisiunea de a accesa date precum contactele utilizatorului, mesaje SMS, cardul SD, camera, Bluetooth si altele. Toate permisiunile trebuie acordate de utilizator in momentul instalarii.

4.2.2. Arhitectura platformei Android

Următoarea diagramă arată cele mai importante componente ale sistemului de operare Android. Fiecare secțiune va fi mai apoi descrisă detaliat.

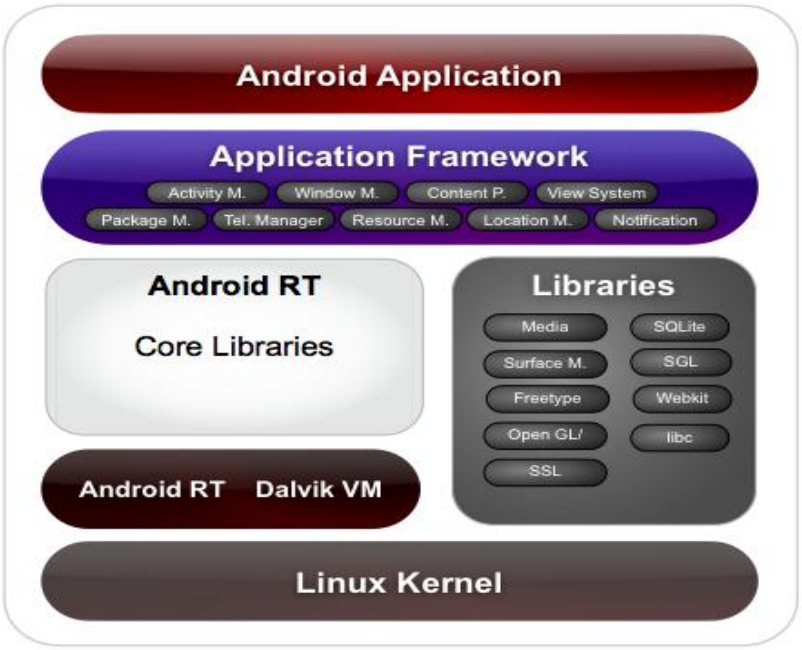

Figura 4.2 Arhitectura platformei Android

Aplicații

Platforma Android dispune de un set de bază de aplicații , incluzând un client de e-mail, program SMS, calendar, hărți, browser, contacte și altele. Toate aplicațiile sunt scrise în limbajul de programare Java.

Framework de aplicație

Punând la dispoziție o platformă de dezvoltare deschisă, Android oferă dezvoltatorilor abilitatea de a construi aplicații extrem de bogate și inovative. Dezvoltatorii pot beneficia de hardware de dispozitiv, informații de acces la locație, rularea serviciilor în background, setarea de alarme, adăugarea notificărilor în bara de stare și multe altele.

Dezvoltatorii au acces total la același API folosit de aplicațiile de bază. Arhitectura aplicației este proiectată în scopul simplificării reutilizării componentelor; orice aplicație își poate publica diferitele capabilități, iar orice altă aplicație se poate folosi apoi de acele funcționalități (subiect al constrângerilor de securitate impus de framework). Același mecanism permite utilizatorilor înlocuirea componentelor.

La baza tuturor aplicațiilor stă un set de servicii și sisteme, incluzând:

- Un set bogat și extensibil de vederi care sunt folosite în construirea aplicației, incluzând liste, grid-uri, căsuțe de text, butoane și un bowser încorporat.
- Furnizorii de conținut care permit aplicațiilor să acceseze date din cadrul altor aplicații (precum contacte) sau să-și împărtășească propriile date.
- Un manager de resurse, ce oferă acces resurselor lipsite de cod, precum string-uri localizate, grafice și fișiere de layout.

- Un manager de notificări care permite tuturor aplicațiilor să afișeze alertele proprii in bara de stare.
- Un manager de activități care gestionează ciclul de viață al aplicațiilor și oferă o stivă de navigație comună.

Librării

Android include un set de librării C/C++ folosite de diferite componente ale sistemului. Aceste capabilități sunt expuse dezvoltatorilor prin intermediul framework-ului de aplicație Android. Câteva din librăriile de bază sunt enumerate mai jos:

- Librăria C de sistem – o implementare derivată BSD (Berkely Software Distribution –distribuție UNIX) a librăriei C de sistem (libc), personalizată pentru dispozitive încorporate bazate pe Linux.
- Librării media – bazate pe OpenCORE de la PacketVideo; librăriile suportă playback și înregistrări pentru multe formate audio și video, precum și pentru imagini, inclusiv MPEG4, H.264, MP3, AAC, AMR, JPG și PNG.
- Manager de suprafață – gestionează accesul la subsistemul de afișare și compune straturile grafice 2D și 3D de la multiple aplicații.
- LibWebCore – un motor de web browser modern care sprijină atât browser-ul Android cât și vederile web încorporate.
- SGL – motorul grafic 2D de bază.
- Librării 3D – o implementare bazată pe API-ul OpenGL ES; librăriille folosesc fie accelerarea 3D hardware (unde este disponibil), fie un rasterizator software 3D optimizat.
- FreeType – redarea vectorilor de font și a imaginilor.
- SQLite – un motor de baze de date relaționale puternic și disponibil tuturor aplicațiilor.

Runtime-ul Android

Platforma Android include un set de librării de bază care oferă o mare parte din funcționalitatea disponibilă în librăriile de bază ale limbajului de programare Java.

Fiecare aplicație Android rulează în cadrul propriului proces, cu propria instanță a mașinii virtuale Dalvik. Dalvik a fost scris astfel încât un dispozitiv să poată rula mai multe mașini virtuale în mod eficient. Mașina virtuală Dalvik execută fișierele în formatul *.dex* care sunt optimizate pentru consum minim de memorie. Mașina virtuală este bazată pe regiștri și rulează clase compilate de un compliator Java care au fost transformate în formatul *.dex* cu ajutorul uneltei incluse “dx”.

Mașina virtuală Dalvik se bazează pe nucleul Linux care asigură funcționalități precum lucrul cu thread-uri și gestionarea memoriei de nivel jos.

Nucleul Linux

Android se bazează pe versiunea 2.6 a sistemului de operare Linux pentru servicii sistem de bază precum securitate, management-ul memoriei, gestionarea proceselor, stiva rețelei și drivere. Nucleul se comportă de asemenea și ca un nivel de abstractizare între hardware și restul stivei software.

4.2.3. Componentele unei aplicații Android

Componentele sunt blocuri esențiale ale aplicației Android. Fiecare componentă reprezintă un punct distinct prin care sistemul poate accesa aplicația, şi joacă un rol specific ajutând astfel la definirea comportamentului aplicației.

Există șase tipuri diferite de componente. Fiecare tip are un scop specific și un ciclu de viață care definește cum este creată și distrusă respectiva componentă.

Activități

Activităţile reprezintă partea de prezentare într-o aplicaţie. Fiecare ecran din aplicaţie va fi o extensie a clasei *Activity*. O aplicaţie poate fi alcătuită dintr-o singură activitate sau poate să conţină mai multe în funcţie de design-ul conceput de dezvoltator. De obicei, una dintre activităţi este marcată ca fiind prima activitate care să fie prezentată utilizatorului. Trecerea de la o activitate la alta se realizează prin apelul explicit in cadrul primei activităti a rutinei care o porneşte pe cea de a doua.

Cele mai multe activităţi sunt proiectate pentru a ocupa întregul ecran, dar se pot crea activităţi care sunt semitransparente, plutitoare, sau dialog-uri. Fiecare actvitate este alcătuită dintr-o structură de *vederi* . Vederile reprezintă elementele de interfaţă de bază in Android. De exemplu, o vedere poate prezenta o imagine şi să reacţioneze cand utilizatorul apasă pe ea.

Pentru a crea o activitate, dezvoltatorul trebuie să extindă clasa de bază *Activity* şi să constiuască in interiorul ei interfaţa şi comportamentul dorit.

Componentele unei aplicaţii au un ciclu de viaţă : un început când Android le instanţiază pentru a răpunde la anumite intenţii şi un sfarşit atunci când instanţele sunt distruse. Starea fiecărei activităţi este determinată de poziţia sa cu privire la stiva de activităţi, o colecţie *FIFO* (first in first out) ce conţine toate activităţile care rulează la un moment dat în sistem. Când o nouă activitate este începută, aceasta este mutată în capul stivei. În cazul în care utilizatorul navighează înapoi folosind butonul dedicat sau cand activitatea este închisă, activitatea imediat următoare este împinsă pe stivă şi devine activă. Această stivă este folosită si de manager-ul de memorie pentru determinarea priorităţii aplicaţiei şi eliberarea resurselor de memorie.

O activitate are, în esenţă, trei stări:

- Este activă sau se execută atunci când este în prim planul ecranului (se situeaza în capul stivei de activităţi). Aceasta este activitatea cu care interacţionează utilizatorul la un moment dat.
- Este întreruptă în cazul în care şi-a pierdut focus-ul, dar este încă vizibilă pentru utilizator. O altă activitate se află peste activitatea întreruptă dar o parte a activităţii întrerupte este incă accesibilă utilizatorului. O activitate întreruptă este menţinută în viaţă (îsi menţine starea în intregime, informaţiile despre membrii şi rămâne ataşat la managerul de ferestre), dar poate fi oprită de sistem în situatii de utilizare excesivă a memoriei.
- Este oprită în cazul în care este complet acoperită de o altă activitate. Informaţiile despre stare sunt menţinute, dar activitatea respectivă nu mai este vizibilă pentru utilizator şi va fi distrusă cand va fi nevoie să se elibereze memoria.

Ciclul de viață al unei activități este prezentat în figura următoare:

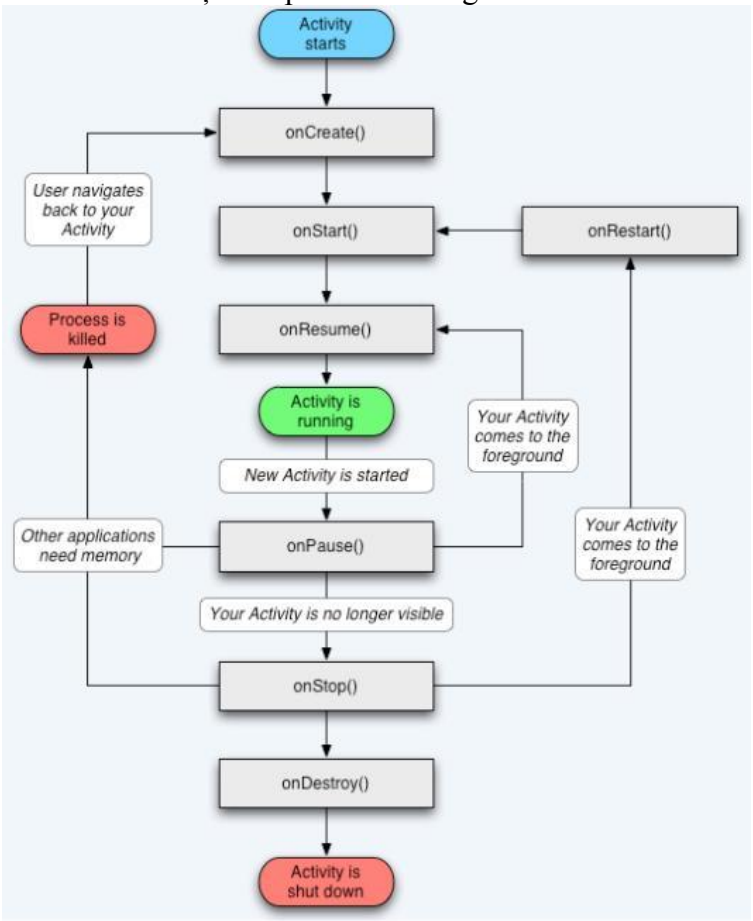

Figura 4.3 Ciclul de viață al unei activități [10]

Intenții

Intent-urile reprezintă un mecanism de mesaje ce permite declararea intenţiei ca o acţiune să fie efectuată, de obicei, cu (pe) un anumit set de date. Intenţiile suportă interacţiunea dintre două componente diverse aflate in aceiaşi aplicaţie sau în aplicaţii diferite. Ele sunt responsabile cu transformarea unei colecţii de componente independente într-un singur sistem interconectat.

Una dintre cele mai comune utilizări ale intenţiilor îl reprezintă pornirea unei activităţi, fie în mod explicit (prin specificarea clasei activităţii care se doreşte a fi încărcată), fie implicit (prin cererea ca o acţiune să fie efectuată pe un set de date). O aplicaţie poate înregistra un broadcast receiver care să asculte şi să reacţioneze la aceste Intent-uri. Android foloseşte intent-uri broadcast pentru a anunţa evenimente de sistem , cum ar fi schimbarea stării conexiunii de internet sau nivelul bateriei telefonului. Aplicaţiile native Android , cum ar fi SMS manager-ul sau manager-ul de telefonie, înregistrează componente care ascultă aceste intent-uri broadcast şi reacţionează corespunzător prin alertarea utilizatorului cu privire la schimbările din sistem.

Pentru a începe în mod explicit o activitate trebuie creat un nou intent specificând contextul curent al aplicaţiei, precum şi clasa de activitate ce trebuie lansată. Această intenţie trebuie trimisă ca parametru metodei *startActivity*, aşa cum se arată în următorul fragment de cod:

```
Intent intent = new Intent(MyActivity.this, MyOtherActivity.class);
startActivity(intent);
```

Servicii

Un serviciu este o componentă care rulează în background pentru efectuarea de operații de durată sau pentru a permite accesul la distanță al proceselor. Un serviciu nu dispune de o interfață utilizator. De exemplu, un serviciu poate poate efectua un proces in background, în timp ce utilizatorul se află în cadrul altei aplicații. O altă componentă, precum o activitate, poate incepe serviciul, îl poate lăsa să ruleze sau îl poate îngloba pentru a interacționa cu acesta.

Un serviciu este implementat ca o subclasă a clasei Service.

Furnizori de conținut

Un furnizor de conținut gestioneaza un set de date partajate. Datele pot fi stocate intr-un fisier, o baza de date SQLite, pe web sau orice alta locatie persistenta ce poate fi accesata de aplicatie. Prin intermediul furnizorului, alte aplicatii pot interoga sau modifca datele.

De exemplu, sistemul Android ofera un furnizor de conținut care gestioneaza infromatiile de contact. Asrfel, orice aplicatie care detine permisiunea poate interoga o parte din continut pentru scrierea si citirea de informatii referitoare la o anumita persoana.

Un furnizor de conținut este implementat ca si subclasa a clasei ContentProvider si trebuie sa implementeze un set de API-uri standard care permit altor aplicatii efectuarea de tranzactii.

Furnizorii de conţinut expun datele lor sub forma unui tabel dintr-un model de baze de date, fiecare rând reprezintă o înregistrare şi fiecare coloană reprezintă o dată de un anumit tip. Fiecare furnizor de conţinut expune un URI public care identifică în mod unic datele sale. Un furnizor de conţinut care controlează mai multe seturi de date (mai multe tabele) trebuie să expună un URI separat pentru fiecare dintre ele.

Pentru a interoga un furnizor este nevoie de trei informaţii:

- un URI care identifică furnizorul;
- numele câmpurilor de date care se doresc accesate;
- tipurile de date pentru aceste campuri;

Receptori de broadcast

Un receptor de broadcast este o componenta care raspunde anunturilor de broadcast ale sistemului. Multe broadcasturi isi au originea in cadrul sistemului, de exemplu, un anunt broadcast legat de inchiderea ecranului, terminarea bateriei, sau efectuarea unei fotografii. Aplicatiile pot initia si ele broadcast-uri, precum anuntarea altei aplicatii de faptul ca anumite date au fost descarcate si se afla pe dispozitiv disponibile spre utilizare. Cu toate ca receptorii de broadcast nu afiseaza o interfata utilizator, pot crea o bara de notificare a status-ului pentru a instiinta utilizatorul de momentul aparitiei unui eveniment de broadcast.

Un receptor de broadcast este implementat ca si subclasa a clasei BroadcsatReceiver si fiecare broadcast este livrat ca un obiect Intent.

Fişierul manifest Android

Fiecare proiect Android include un fişier manifest, *AndroidManifest.xml*, stocat în directorul rădăcină al proiectului. In acest fişier sunt descrise componentele folosite în aplicaţie şi alte informaţii referitoare la permisiuni sau librării ce trebuie legate de aplicaţia curentă. Se poate defini versiunea de Android folosită de aplicaţie, precum şi o temă preferată pentru afişarea interefeţei. Structura acestui fişier impune prezenţa unui tag rădacină <*manifest*>urmat de tag-ul <*application*>în interiorul căruia vor fi descrise componentele care alcătuiesc aplicaţia. Pentru definirea de permisiuni se va folosi tag-ul <*uses-permission*>, urmat de numirea permisiunii dorite.

Deoarece sistemul rulează fiecare aplicație într-un proces separat cu fișiere de permisiuni care restricționează accesul la alte aplicații, aplicația nu poate acționa direct o componentă din cadrul altei aplicații. Sistemul Android, în schimb, poate. Astfel, pentru a activa o componentă a altei aplicații trebuie trimis un mesaj sistemului care să specifice inenția (*Intent*) de a începe o anumită componentă. Sistemul activează apoi componenta respectivă.

4.2.4. Servicii de localizare, hărți, geo-coding

Una dintre caracteristicile definitorii ale telefoanelor mobile este portabilitatea lor, astfel că nu este de mirare faptul că unele dintre cele mai atractive caracteristici Android sunt serviciile care permit localizarea, contextualizarea şi maparea locaţiile fizice.

Hărţile şi servicii bazate pe locaţie au nevoie de latitudine şi longitudine pentru a indica locaţiile geografice, dar există şi posiblitatea de a descoperi aceste coordonate pe baza unei adrese printr-un proces numit *GeoCoding.* Prin acest proces, dezvoltatorul are posibilitatea să transforme o adresă reală în coordonate geografice şi invers.

Pentru a determina locaţia curentă, Android oferă un serviciu denumit *LocationManager*. Prin intermediul lui, dezvoltatorul poate citi coordonatele la care se găseste sistemul la un moment dat, poate asculta şi înregistra traseele urmate şi poate identifica proximitatea cu o locaţie predefinită.

Pentru a prezenta aceste informaţii utilizatorului, este folosită o librărie externă *Google Maps*. Principala componentă din acestă librărie este vederea *MapView*. *MapView* afişează o hartă cu datele obţinute de la serviciul Google Maps. Ea este capabilă să interaţioneze cu utilizatorul şi permite capturarea evenimentelor de apăsare a ecranului, ajustarea manuală a zoom-ului şi completarea automată cu hărţi aditionale. Programatorul poate folosi acest API şi pentru a trasa diferite rute pe hartă, prin intermediul unor obiecte de tip *Overlay*. Serviciile Google Maps nu fac parte din pachetul standard Android, dezvoltatorul fiind nevoit să obţină licenţe separate penrtru utilizarea lor.

4.2.5. Emulatorul

Emulatorul Android imită toate caracteristicile hardware şi software ale unui dispozitiv mobil tipic, cu excepţia faptului că nu se pot primi sau efectua apeluri telefonice reale şi nu se poate simula folosirea camerei foto încorporate. Se pune la dispoziţia utilizatorului o tastatură şi un ecran care se pot accesa folosind mouse-ul sau tastatura calculatorului. Se pot afişa şi utiliza toate aplicaţiile disponibile în sistemul Andorid, se pot utiliza configuraţii hardware şi versiuni de platformă diferite. Odată pornit, emulatorul poate invoca alte aplicaţii sau servicii, poate reda conţinut media şi poate să se conecteze la Internet.

4.3. Servicii web

4.3.1. Noțiuni generale

Termenul de servicii web descrie un mod standardizat de integrare a aplicațiilor bazate pe Web utilizându-se standardele deschise XML, SOAP, WSDL și UDDI prin intermediul unei magistrale de protocoale Internet.XML este utilizat pentru etichetarea datelor, SOAP este utilizat pentru transferul datelor, WSDL este utilizat pentru descrierea serviciilor disponibile, iar UDDI este utilizat pentru afișarea listei serviciilor disponibile.

Utilizate inițial ca mijloc prin care afacerile comunică între ele și cu clienții lor, serviciile Web permit organizațiilor să comunice date fără să fie necesar să-și cunoască unele altora tehnologiile utilizate. Spre deosebire de modelele tradiționale client-server, precum este modelul server Web/ sistem de pagini Web, serviciile Web nu oferă utilizatorului o interfață grafică cu utilizatorul (GUI).În schimb, serviciile Web partajează logica afacerii, date și procese prin intermediul unei interfețe de program de-a lungul unei rețele. Interacțiunea are loc între aplicații, nu între utilizatori. Dezvoltatorii pot apoi să adauge serviciul Web la o interfață grafică cu utilizatorul (de exemplu o pagină web sau un program executabil) pentru a oferi utilizatorilor funcționalități specifice.

Serviciile Web permit unor aplicații diferite provenite din surse diferite să comunice între ele fără să se piardă timpul cu efectuarea unei codificări și, deoarece toate comunicațiile sunt în XML, serviciile Web nu depind de un anumit sistem de operare sau limbaj de programare. De exemplu, Java poate comunica cu Perl, aplicațiile Windows pot comunica cu aplicațiile UNIX.

Serviciul Web poate fi descris prin folosirea unui limbaj de descriere a serviciului, poate fi publicat într-un registru de servicii, poate fi descoperit printr-un mecanism standard (în etapa de execuție sau în etapa de proiectare), poate fi invocat printr-o interfață de programare a aplicației (API) declarată (de obicei pe o rețea) și poate fi compus cu alte servicii.Nu este obligatoriu ca un serviciu Web să existe pe World Wide Web, acesta putând exista oriunde pe o rețea, Internet sau intranet. De asemene, unele servicii pot fi invocate printr-o simplă apelare de metodă în același proces al sistemului de operare sau prin utilizarea memoriei partajate între procesele puternic cuplate care rulează pe aceeași mașină.

Detaliile specifice platformei de implementare și utilizare ale unui serviciu Web nu sunt relevante pentru un program care invocă serviciul, un serviciu Web fiind disponibil printr-o interfață API și un mecanism de invocare declarate (protocol de rețea, scheme de codificare date etc.).Browserul Web nu este influențat de tipul serverului de aplicație (de exemplu Apache, Tomcat, Microsoft IIS sau IBM Websphere).Atât browserul Web cât și serverul de aplicație utilizează HTTP și HTML sau folosesc un set limitat de tipuri MIME.Serverul de aplicație Web poate fi folosit de tipuri diferite de browsere Web sau chiar de clienți non- browser. Această înțelegere comuna minimală dintre componente permite serviciilor Web să formeze un sistem de componente slab conectate.

4.3.2. Arhitecturi, tehnologii, tehnici și limbaje orientate Java

4.3.2.1. Arhitectură orientată pe servicii

Arhitectura orientată pe servicii (SOA) atunci când mai multe aplicații eterogene trebuie să comunice într-un mediu distribuit. În cadrul SOA, există trei parteneri cu roluri distincte: Provider de servicii, Broker de servicii și Client de servicii.

Provider-ul de servicii crează un serviciu Web și îl publică într-un registru al broker-ului de servicii. Broker-ul de servicii este un registru de servicii care publică serviciile disponibile oricărui client. Clientul localizează serviciile în cadrul registrului și apoi se leagă la provider-ul de servicii pentru invocarea corectă a serviciilor Web.

Interacțiunile dintre cei trei parteneri sunt efectuate cu ajutorul operațiilor de publicare, căutare și legare.

Figura următoare ilustrează cele trei entități si legăturile dintre ele.

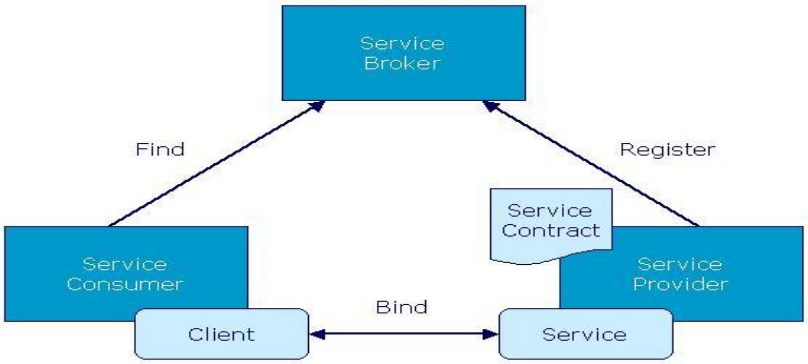

Figura 4.4 Entitățile SOA [15]

Pentru implementarea unei arhitecturi orientate pe servicii Web sunt utilizate numeroase tehnologii și instrumente Java, numărul acestora fiind într-o continuă creștere datorită progresului tehnologic.

Cele mai utilizate tehnologii și instrumente Java sunt prezentate structurate pe niveluri in figura 4.5. Fiecare nivel din figură conține tehnologii și/sau instrumente care interacționează cu tehnologiile și respectiv instrumentele din nivelurile vecine.

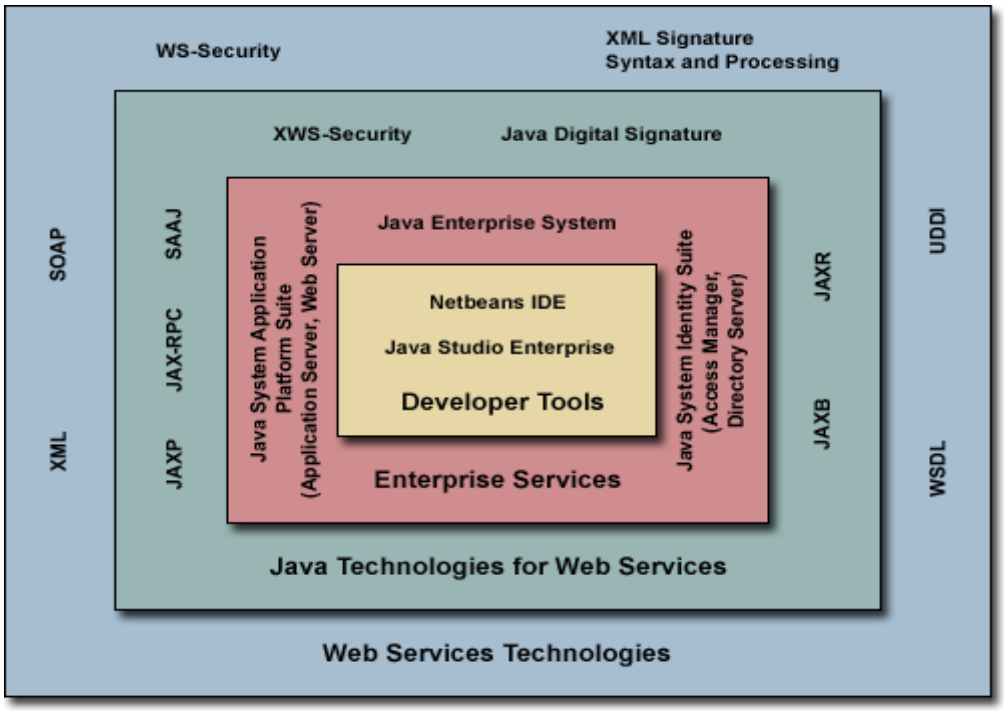

Figura 4.5 Tehnologii și instrumente Java orientate spre servicii Web [16]

4.3.2.2. Protocoale, limbaje și tehnologii utilizate în platforma Java

Dezvoltarea serviicilor Web se bazează pe standarde acceptate și utilizate pe scară largă. Clienții și serviciile pot să comunice și să deruleze afaceri indiferent de platforma și limbajul de programare utilizate. În continuare sunt prezentate protocoale, limbaje și tehnologii utilizate ca standarde pentru dezvoltarea de servicii Web pe platforma Java.

1. **Limbajul de marcare extins (XML)**

XML este standardul dedicat descrierii datelor care fac obiectul schimburilor de date prin intermediul Web-ului. XML utilizează etichete care "marchează" și descriu conținutul unui document. O etichetă XML identifică informațiile dintr-un document și structura acestora.

Un exemplu de informații identificate utilizând sistemul de marcare al limbajului XML este prezentat în listingul 1. În acest listing este descrisă o structură *bookshelf*(raft de cărți) care conține un articol subordonat book (carte). Articolul book conține trei asticole subordonate: *title* (titlu), *author* (autor) și *price* (preț).

```
<bookshelf>
<book>
        <title>My Life and Times</title>
        <author>Felix Harrison</author>
        <price>39.95</price>
</book>
</bookshelf>
```

Listingul 1. Exemplu de marcare XML

Etichetele XML, precum <*book*>sau <*title*>, nu conferă o semnificație informațiilor pe care le identifică. Aceste informații XML au o semnificație numai dacă utilizatorii asociază unei etichete o anumiya semnificație. Astfel, în exemplul prezentat

anterior, eticheta <*book*>semnifică o carte, iar etichetele <*title*>,<*author*>și <*price*>semnifică titlul, autorul și prețul cărții. În aplicații se poate face schimb de documente XML și se pot prelucra informațiile din aceste documente în conformitate cu semnificațiile și organizarea etichetelor XML. Documentele XML au o structură bine definită. Pentru fiecare etichetă XML există un marcaj de început și un marcaj de sfârșit. Dacă acestea se află în interiorul altei etichete trebuie să fie incluse între marcajele de început și de sfârșit ale etichetei care o conține. Ca urmare, marcajele de început ale etichetelor <*title*>, <*author*>și <*price*> sunt scrise în această ordine după marcajul de început al eticheti <*book*>, iar marcajele de sfârșit ale etichetelor </*title*>, </*author*> și </*price*>sunt scrise ân această ordine înainte de marcajul de sfârșit al etichetei </*book*>.

Un document XML este de obicei asociat cu o schemă care îi definește regulile de gramatică. În această schemă sunt specificate etichetele permise în document, structura lor și alte reguli referitoare la acestea. Deoarece documentele XML valabile trebuie să fie bine structurate și conforme cu schema asociată, prelucrarea documentelor XML este relativ ușoară. În consecință, XML a fost adoptat ca limbaj de marcare a datelor pentru serviciile Web.

2. Protocolul simplu de acces la obiecte (SOAP)

Pentru realizarea unui schimb eficient de date prin intermediul Web utilizând XML, pe lângă atribuirea unei semnificații etichetelor și agregarea structurii acestora este necesar și un protocol pentru formatarea documentului XML. Acest protocol trebuie să permită destinatarului să înțeleagă care este partea principală a mesajului primit și care este partea cu instrcțiuni sau conținut suplimentar. În acest scop, este utilizat protocolul simplu de acces la obiecte (SOAP). SOAP este un protocol bazat pe XML utilizat pentru schimbul de date într-un mediu distribuit.

SOAP asigură un format general pentru mesaje având ca scop efectuarea cu succes a schimbului de date între clienți și servicii.

Elementul de bază al transmisiei bazate pe SOAP este un mesaj SOAP constituit dintr-un plic SOAP obligatoriu, un antet SOAP opțional și un corp SOAP obligatoriu (Figura 2). Plicul conține spațiul de nume XML și un stil de codificare. Spațiuld e nume XML indică numele care pot fi utilizate în mesajul SOAP. Spațiile de nume sunt astfel proiectate încât sunt evitate conflictele de nume. Același nume poate fi utilizat pentru diferite elemente, cu condiția ca numele să se afle în spații de nume diferite.

Stilul de codificare identifică tipurile de date recunoscute de mesajul SOAP. Includerea unui antet are ca rezultat extinderea modulară a mesajului.

Un mesaj SOAP se transmite de la un client la un servciu și invers, iar pe parcursul acestei transmisii poate să treacă printr-un set de noduri intermediare. Fiecare nod este o aplicație care poate primi și trimite mai departe mesaje SOAP. Un nod intermediar poate oferi serviici suplimentare, precum transformarea datelor din mesaj sau efectuarea de operații privinde securitatea.

Antetul SOAP poate fi utilizat pentru indicarea unei prelucrări suplimentare într-un nod intermediar, adică a unei prelucrări independente de prelucrarea efectuată la destinația finală. De obicei, antetul SOAP este utilizat pentru transportul informațiilor referitoare la securitate care urmează să fie peclucrate de componentele procesului de execuție.

Corpul mesajului SOAP conține partea principală a mesajului și anume partea care ajunge la destinatarul final al mesajului SOAP.

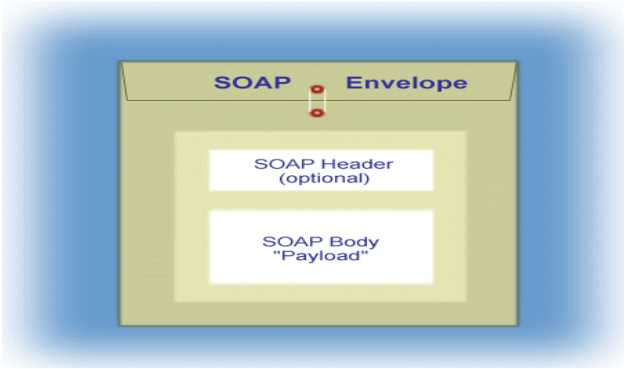

Figura 4.6 Structura conceptuală a unui mesaj SOAP

SOAP Messages With Attachments este un standard care specifică formatul unu mesaj SOAP ce conține atașamente (de exemplu imagini). Mesajele SOAP, care pot conține sau nu atașamente sunt independente de sistemul de operare sau de platformă și pot fi transportate utilizând diferite protocoale de comunicație, precum HTTP sau SMTP.

3. Limbajul de descriere a serviciilor Web (WSDL)

Informațiile referitoare la formatul care poate fi utilizat de un client atunci când efectuează o cerere către un serviciu precum și informațiile referitoare la semnificația cererii sunt incluse într-un document XML, denumit WSDL, ce conține o descriere a interfeței serviciului Web.

Pentru a descoperi un serviciu Web, clientul trebuie să găsească documentul WSDL al serviciului. Cel mai utilizat mod de efectuare a acestei acțiuni este cel în care clientul găsește un "pointer" către documentul WSDL în informațiile de înregistrare ale unui serviciu Web aflate într-un registru UDDI sau într-un registru/ depozit ebXML.

Un document WSDL descrie un serviciu Web ca fiind o colecție de elemente abstracte denumite "porturi" sau "puncte terminale". Documentul WSDL definește, într-un mod abstract, acțiunile efectuate de un serviciu Web și datele transmise acestor acțiuni. Acțiunile sunt reprezentate prin "operații", iar datele sunt reprezentate prin "mesaje". O colecție de operații corelate este denumită "tip de porturi". Un tip de porturi constituie colecția de acțiuni oferite de un serviciu Web. O "legătură" transformă descrierea WSDL dintr-o descriere abstractă într-una concretă. O legătura indică protocolul rețelei și specificațiile formatului mesajelor pentru un anumit tip de porturi. Un port este definit prin asocierea unei adrese de rețea unei legături. Un client poate apela operațiile serviciului conform protocolului și formatului mesajelor specificate dacă localizează un document WSDL și găsește legătura și adresa de rețea pentru fiecare port.

4. Descrierea universală, descoperirea și integrarea (UDDI)

Specificațiile de descriere, descoperire și integrare universale (UDDI) descriu modul de publicare și descoperire a informațiilor referitoare la servicii într-un registru conform cu UDDI. Astfel, specificațiile definesc o schemă UDDI și o interfață de programare a aplicației (API) a UDDI.

Schema UDDI identifică tipurile structurilor de date XML care conțin o intrare în registrul unui serviciu. Un registru UDDI oferă infomații despre un serviciu, precum numele serviciului, o scurtă descriere a acestuia, adresa la care poate fi accesat și descrierea interfeței pentru accesare.

Fiecare descriptor de servicii conține informația necesară de către serviciul apelant pentru a găsi și a se lega de un anumit serviciu Web. UDDI permite:

- Publicarea serviciului – UDDI definește operații care permit organizațiilor să-și expună serviciile Web.
- Interogări – UDDI definește operații care permit utilizatorilor să extragă informații despre servicii din registrul UDDI.
- Clasificări – UDDI definește operații care permit clasificarea bussines-ului și serviciilor conform standardelor.

UDDI constă din trei elemente numite pagini, în cadrul cărora sunt descrise serviciile:

- Pagini albe – conțin informații de bază de contact pentru fiecare provider de servicii Web. Include o descriere de bază a companiei care expune serviciul.
- Pagini galbene – conțin mai multe detalii despre companie și include descrierea funcționalităților pe care compania le poate oferi unui potențial partener.
- Pagini verzi – conțin informația de legare a serviciului Web. Include diferite interfețe, locații URL, informații de descoperire și date similare necesare pentru găsirea și invocarea unui serviciu.

Secțiunile descriptive ale UDDI sunt numite Listări și constau din descrierea interfeței serviciului. Descrierile sunt create utilizând WSDL și sunt stocate într-un registru UDDI. UDDI va permite registrelor să facă schimb de Listări, astfel încât să fie posibilă replicarea pe mai multe registre UDDI.

UDDI definește un set de tipuri de date descrise mai jos:

- businessEntity – structura de nivel înalt; descrie un business sau o altă entitate pentru care este înregistrată informația.
- businessService –descrierea unui set de servicii ce poate conține una sau mai multe template-uri de legare (binding template).
- bindingTemplate – informația necesară pentru invocarea serviciilor specifice care pot implica legături cu unul sau mai multe protocoalem precum HTTP sau SMTP.
- tModel – reprezintă amprenta tehnică pentru un serviciu dat, care poate funcționa de asemenea ca un spațiu de nume pentru identificarea altor entități, inclusiv tModel.

Pașii necesari pentru înscrierea regisrtului UDDI în contextul arhitecturii SOA sunt explicați în cele ce urmează:

- Un provider de servicii Web publică informația despre serviciu (luată din fișierul WSDL) în registrul UDDI.
- Cientul caută în UDDI pentru a găsi un serviciu care să se potrivească cerințelor.
- UDDI trimite descrierea clientului.
- Clientul se conectează la provider-ul serviciului Web folosind protocolul SOAP pentru a obține fișierul WSDL ce descrie operațiile serviciului.
- Operațiile serviciului Web sunt invocate prin intermediul SOAP.

4.4. Serviciul Java pentru mesaje (JMS)

Serviciul Java pentru mesaje (JMS) este serviciul standard petru transmiterea mesajelor întreprinderii. Transmiterea mesajelor întreprinderii, cunoscută sub numele de middleware orientat spre transmiterea de mesaje (MOM), este un instrument esențial pentru crearea de aplicații pentru întreprindere. Prin combinarea tehnologiei Java cu transmiterea mesajelor, API-ul JMS furnizează un instrument care permite întreprinderii să-și rezolve problemele organizaționale.

Transmiterea mesajelor oferă un serviciu fiabil și flexibil pentru schimbul asincron al datelor și evenimentelor de afaceri în interiorul unei întreprinderi. API-ul JMS oferă și un cadru de lucru prentrun furnizori care permit dezvoltarea, în limbajul de programare Java, de aplicații portabile bazate pe mesaje.

API-ul JMS mărește productivitatea programatorilor prin definirea unui set de concepte privind transmiterea de mesaje și de strategii de programare care sunt acceptate de toate sistemele de transmitere de mesaje conform cu tehnologia JMS.Serviciul Java pentru mesaje este parte integrantă a platformei J2EE. Dezvoltatorii de aplicații pot transmite mesaje prin intermediul componentelor care utilizează API-urile J2EE.

Caracteristicile API-ului JMS din platforma J2EE 1.4 sunt:

- Componentele "bean" bazate pe mesaje permit consumarea asincronă a mesajelor JMS.
- Trimiterea și primirea de mesaje pot participa la tranzacțiile JTA.
- Interfețele arhitecturii conectorilor J2EE permit conectarea diferitelor implementări JMS la un server de aplicații J2EE 1.4.

Utilizarea API- ului JMS în platforma J2EE simplifică activitatea de dezvoltare a întreprinderii, permițând interacțiuni slab cuplate, fiabile și asincrone între componentele J2EE și sistemele moștenite capabile să transmită mesaje. Arhitectura containerelor EJB a platformei J2EE îmbunătățește API-ul JMS prin:

- Facilitarea consumării concurente a mesajelor;
- Asigurarea suportului pentru tranzacții distribuite, astfel încât actualizările bazelor de date, prelucrarea mesajelor și conectorii la sistemele EIS (Sisteme informatice de întreprinderi) care utilizează arhitectura J2EE a conectorilor să poată participa în același context de tranzații.

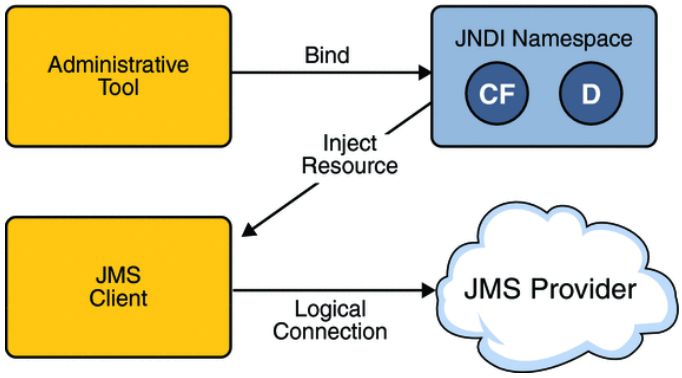

Figura 4.7 Arhitectura JMS

4.4.1. Elemente și modele JMS

În continuare sunt prezentate elementele și modelele JMS.

1. ***Elementele JMS sunt următoarele:***
   - Furnizorul JMS: este o implementare a interfeței JMS pentru MOM. Furnizorul este implementat fie prin intermediul Java JMS, fie ca un adaptor pentru un MOM care nu se bazează pe Java;
   - Clientul JMS: este o aplicație/ un obiect care se bazează pe Java și produce și/sau consumă mesaje;
   - Producătorul JMS: este un client JMS care trimite mesaje;
   - Consumatorul JMS: este un client JMS care primește mesaje;
   - Mesajul JMS: este un obiect care conține datele tranferate între clienți JMS;
   - Coada JMS: este o zonă ce conține mesaje trimise și care așteaptă să fie citite. Mesajele sunt livrate în ordinea trimiterii și sunt șterse din coadă după ce au fost citite;
   - Topica JMS: este un mecanism de distribuție pentru publicarea mesajelor livrate mai multor abonați.
2. ***API-ul JMS asigură suportul pentru două modele:***
   - Modelul punct-la-punct sau trimitere în coadă;
   - Modelul de publicare și abonare.

În modelul punct-la-punct sau de trimitere în coadă, producătorul cunoaște destinația mesajelor pe care le trimite direct la coada de mesaje a consumatorului de unde sunt citite de către consumator. Acest model are următoarele caracteristici:

- Mesajul este recepționat de către un singur consumator;
- Nu este necesar ca producătorul să lucreze în momentul în care destinatarul consumă mesajul și nici ca destinatarul să lucreze în momentul în care este trimis mesajul;
- Prelucrarea cu succes a fiecărui mesaj este confirmată de destinatar.

Modelul de publicare și abonare asigură suportul pentru publicarea mesajelor pe un anumit topic de mesaje. Numărul abonaților care se pot înregistra pentru a primi mesaje pe un anumit topic poate fi mai mare sau egal cu zero. În acest model, nici cel care publică și nici abonatul nu știu nimic unul despre celălalt. Modelul de publicare și abonare are următoarele caracteristici:

- Mesajul este recepționat de mai mulți consumatori;
- Între cei care publică și abonați există o condiție de sincronizare;
- Abonatul care și-a creat un abonament pe termen lung, va primi în momentul reconectării sale mesajele publicate pe durata de timp în care nu a fost conectat;
- Abonatul care nu și-a creat un abonament trebuie să rămână activ în permanență pentru a primi mesaje.

Utilizând Java, JMS oferă un mod de separarae a aplicației de stratul de transport destinat furnizării datelor. Se pot utiliza aceleași clase Java pentru comunicarea cu diferiți furnizori JMS utilizând informațiile JNDI pentru furnizorul luat în considerare. La început, clasele utilizează o "fabrică" de conexiuni pentru conectarea la coadă sau la topic, iar în continuare este realizată popularea mesajelor și trimiterea sau publicarea acestora. La destinație, clienții primesc mesaje sau se abonează la mesaje.

4.4.2. API-ul JMS

API- ul JMS este furnizat în pachetul *javax.jms*. În continuare sunt prezentate componenetele API- ului JMS.

- Interfața *ConnectionFactory*- este un obiect utilizat de un client pentru crearea unei conexiuni la furnizorul JMS. Clienții JMS accesează "fabrica" de conexiuni prin intermediul unor interfețe portabile nefiind necesară modificarea codului daca se schimbă implementarea considerată. Administratorii configurează "fabrica" de conexiuni în spațiile de nume JNDI astfel încât clienții JMS să poată efectua căutări. În funcție de tipul mesajului, utilizatorii pot folosi fie o "fabrica" de conexiuni la cozi de mesaje, fie o "fabrica" de conexiuni la topic-uri.
- Interfața *Connection* - *d*upă obținerea unei "fabrici" de conexiuni, se poate crea o conexiune la un furnizor JMS. O conexiune este o legătură de comunicație între aplicație și serverul de transmitere a mesajelor. În funcție de tipul conexiunii, conectorii permit utilizatorilor să creeze sesiuni penru trimiterea și primirea mesajelor dintr-o coadă sau dintr-un topic.
- Interfața *Destination* - este un obiect administrat ce încapsulează identitatea unei destinații a mesajului. Destinația mesajului este locul unde sunt livrate și consumate mesajele. Obiectul poate fi o coadă sau un topic. Administratorul JMS crează aceste obiecte, iar utilizatorii le descoperă utilizând JNDI. La fel ca în cazul "fabricii" de conexiuni, administratorul poate crea două tipuri de destinații: cozi de mesaje pentru modelul punct-la-punct și topicuri pentru modelul de publicare și abonare.
- Interfața *MessageConsumer*- este un obiect creat în timpul unei sesiuni care primește mesaje trimise la o destinație. Consumatorul poate primi mesaje în mod sincron sau în mod asincron pentru transmiterea de mesaje la coadă și pe un anumit topic.
- Interfața *MessageProducer*- este un obiect creat în timpul unei sesiuni care trimite mesaje la o destinație. Utilizatorul poate crea un expeditor către o anumită destinație sau poate crea un expeditor generic care indică destinația în momentul trimiterii mesajului.
- Interfața *Message*- este un obiect trimis între consumatori și producători, adică de la o aplicație la alta.
- Interfața *Session*- reprezintă un context pe un singur fir pentru trimiterea și primirea de mesaje.O sesiune se realizează pe un singur fir astfel încât mesajele sunt serializate, ceea ce înseamnă că mesajele sunt primite unul câte unul în ordinea trimiterii. Avantajul oferit de o sesiune este că asigură suport pentru tranzacții. Dacă utilizatorul selectează suportul tranzacției, contextul sesiunii păstrează un grup de mesaje până se efectuează transmiterea și, după aceea, livrează mesajele. Înainte de transmiterea tranzacției, utilizatorul poate să renunțe la mesaje utilizând o operație de trimitere înapoi (rollback). O sesiune permite utilizatorilor să creeze mesaje, producătorilor să trimită mesaje și consumatorilor de mesaje să primească mesaje.

5. Proiectare de detaliu şi implementare

5.1. Cazuri de utilizare

Scopul urmărit în acest tip de modelare este descrierea funcţionalităţii sistemului aşa cumaceasta este văzută din exterior de un număr de actori şi a conexiunilor acestora lacazurile de utilizare furnizate de sistem. Aşadar, pentru crearea modelului cazurilor deutilizare vor trebui identificaţi actorii, cazurile de utilizare, relaţiile dintre acestea cât şirelaţiile cu actorii.

Scopul principal al construirii acestui tip de diagramă este:

- Să decidă şi să descrie cerinţele funcţionale ale sistemului, cerinţe care au fost dedusedupă o discuţie între client şi/sau utilizatorii sistemului şi viitorii dezvoltatori ai acestuia.
- Să ofere o descriere clară şi consistentă a ce va trebui să facă sistemul.
- Să constituie o bază pentru realizarea testelor de verificare dacă funcţionalitatea finală a sistemului concordă cu cerinţele iniţiale ale acestuia.
- Să permită o transformare uşoară a cerinţelor funcţionale în viitoare clase şi operaţii.

5.1.1. Diagrama cazurilor de utilizare

Un model use case este descris folosind una sau mai multe diagrame use case.Acestea conţin următoarele elemente de modelare: actori, cazuri de utilizare şi diferite relaţii între aceste elemente: generalizare, asociere, dependenţă.

Figura 5.1 Diagrama cazurilor de utilizare

5.1.2. Descrierea cazurilor de utilizare

În acest paragraf se vor descrie pe scurt cazurile de utilizare, iar unul mai complex va fi prezentat în amănunt.

- Specificație de caz de utilizare:*Înregistrare*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul a deschis aplicația și a selectat opțiunea de înregistrare prin apasarea butonului *Register* din prima fereastră.
    1. Utilizatorul introduce numele, prenumele, numele de utilizator, parola, data nașterii și e-mail-ul.
    2. Apasă butonul de înrgistrare.
    3. În cazul în care toate datele au fost introduse, numele de utilizator este disponibil si e-mail-ul valid, inregistrarea se efectuează cu succes.
    4. La adresa specificată se trimite un e-mail de confirmare a înregistrării în care se specifică numele utilizatorului și parola.
    5. Utilizatorul este direcționat spre meniul principal având acces la toate funcțiile aplicației.
    6. În cazul în care datele introduse nu sunt complete sau valide, utilizatorului îi este afișat un mesaj care specifică ce trebuie modificat.
    7. Utilizatorul poate alege să modifice datele pentru a continua înregistrarea sau poate renunța prin apăsarea butonului *Back.*

  Sfârșitul cazului de utilizare
  Utilizatorul poate părăsi acest caz de utilizare în caz de succes sau poate să îl reia în caz de eșec începând cu pasul 1.
- Specificație de caz de utlizare: *Login*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul a deschis aplicația.
    1. Utilizatorul introduce numele de utilizator și parola.
    2. Apasă butonul de login.
    3. În cazul în care datele introduse sunt corecte, utilizatorul este logat în sistem, fiind direcționat spre meniul principal.
    4. În cazul în care combinația utilizator-parolă nu este corectă, utilizatorul are posibilitatea de a introduce din nou datele, de a cere să primească pe e-mail datele de autentificare sau poate să se înregistreze în cazul în care nu este deja înregistrat.

  Sfârșitul cazului de utilizare
  Dacă operațiile au fost efectuate cu succes utilizatorul părăsește acest caz de utilizare și este direcționat spre meniul principal. În caz de eșec, utilizatorului îi sunt afișate mesaje ajutătoare pentru a putea finaliza operația de logare.
- Specificație de caz de utilizare: *Vizualizare prieteni*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem.
    1. Utilizatorul introduce numele de utilizator și parola.
    2. Utilizatorul selectează din meniul principal opțiunea de vizualizare a prietenilor.

Sfârşitul cazului de utilizare

Cazul de utilizare ia sfârşit după selectarea opțiunii.

- Specificație de caz de utilizare: *Adăugare prieteni*

Începutul cazului de utilizare

Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem.

1. Utilizatorul introduce numele de utilizator şi parola.
2. Utilizatorul selectează din meniul principal opțiunea de adăugare prieten.
3. Utilizatorul poate adăuga un nou prieten după o căutare facută pe baza numelui complet al utlilizatorului.

Sfârşitul cazului de utilizare

Acest caz de utilizare ia sfârşit dupa trimiterea cererii de prietenie.

- Specificație de caz de utilizare: *Vizualizarea profilului propriu*

Începutul cazului de utilizare

Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem.

1. Utilizatorul introduce numele de utilizator şi parola.
2. Utilizatorul selectează din meniul principal opțiunea de vizualizare a profilului.

Sfârşitul cazului de utilizare

Cazul de utilizare ia sfîrşit odată cu selectarea opțiunii.

- Specificație de caz de utilizare: *Modificarea poziției pe hartă*
- Specificație de caz de utilizare: *Vizualizarea profilurilor prietenilor*

Începutul cazului de utilizare

Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem şi a selectat unul din prieteni.

1. Utilizatorul introduce numele de utilizator şi parola.
2. Utilizatorul selectează din meniul principal opțiunea de vizualizare a prietenilor.
3. Utilizatorul selectează prietenul căruia dorește să-i vizualizeze profilul.
4. Utilizatorul poate vedea apoi fotografiile, comentariile sau poate adăuga propriile comentarii la fotografii.

Sfârşitul cazului de utilizare

Cazul de utilizare ia sfârşit atunci când utilizatorul părăsește profilul prietenului.

- Specificație de caz de utilizare: *Adăugare comentarii*

Începutul cazului de utilizare

Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem şi a selectat vizualizarea fotografiilor unui prieten.

1. Utilizatorul introduce numele de utilizator şi parola.
2. Utilizatorul selectează din meniul principal opțiunea de vizualizare a prietenilor.
3. Utilizatorul selectează profilul unui prieten.
4. Utilizatorul selectează vizualizarea pozelor de pe profilul accesat.
5. Utilizatorul adaugă comentarii la poze.

Sfârşitul cazului de utilizare

Cazul de utilizare se sfârșește atunci când utilizatorul a terminat de postat comentariile.

- Specificație de caz de utilizare: *Trimitere de mesaje*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem și a selectat profilul unui prieten.
  1. Utilizatorul introduce numele de utilizator și parola.
  2. Utilizatorul selectează din meniul principal opțiunea de vizualizare a prietenilor.
  3. Utilizatorul selectează profilul unui prieten.
  4. Utilizatorul apasă butonul de trimitere a unui mesaj.
  5. Utilizatorul scrie mesajul în spațiul alocat.
  6. Utilizatorul apasă butonul *Send*.

  Sfârșitul cazului de utilizare
  Cazul de utilizare ia sfîrșit dupa apăsarea butonului *Send*.
- Specificație de caz de utilizare: *Căutare prieteni*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem.
  1. Utilizatorul introduce numele de utilizator și parola.
  2. Utilizatorul selectează din meniul principal opțiunea de căutare a unui prieten.
  3. Utilizatorul introduce numele prietenului pe care dorește să-l găsească.

  Sfârșitul cazului de utilizare
  Cazul de utilizare ia sfârșit în momentul în care utilizatorul a găsit profilul prietenului sau în cazul în care sistemul nu reușește să-i ofere informația căutată deoarece numele introdus nu este corect, utilizatorul poate decide în orice moment să renunțe la căutare.
- Specificație de caz de utilizare: *Editarea profilului*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem.
  1. Utilizatorul introduce numele de utilizator și parola.
  2. Utilizatorul selectează din meniul principal opțiunea de editare a profilului, primind ca răspuns din partea sistemului afișarea propriului profil.
  3. Utilizatorul poate modifica statusul sau fotografiile.

  Sfârșitul cazului de utilizare
  Cazul de utilizare ia sfârșit în momentul în care utilizatorul a terminat de modificat informațiile profilului.
- Specificație de caz de utilizare: *Editarea fotografiilor*
  Începutul cazului de utilizare
  Acest caz de utlizare începe în momentul în care utilizatorul s-a autentificat cu succes în sistem și a selectat editarea profilului.
  1. Utilizatorul introduce numele de utilizator și parola.
  2. Utilizatorul selectează din meniul principal opțiunea de editare a profilului.
  3. Utilizatorul selectează opțiunea de adăugare sau de ștergere a unei fotografii.

Sfârşitul cazului de utilizare

Cazul de utilizare ia sfârşit în momentul în care utilizatorul a finalizat operațiile de ştergere sau adăugare de fotografii.

În continuare va fi prezentat în detaliu un caz de utilizare mai complex, ce necesită parcurgerea mai multor paşi pentru îndeplinirea cu succes a operației.

Specificație de caz de utilizare: *trimiterea notificarilor*

Descriere generală

Scopul este de a oferi o descriere clară a secvenței de interacțiuni ce are loc între aplicație si utilizator, în cazul in care acesta dorește să trimită in mod explicit notificări referitoare la poziția actuală pe hartă.

Actorul principal*:* utilizatorul.

Beneficiarii, utilizatorii şi interesele acestora

Se vor enumera, pe rând, beneficiarii şi utilizatorii sistemului şi principalele lor interese în ceea ce priveşte cazul de utilizare analizat.

- Liderii (conducerea şi personalul administrativ): urmăresc realizarea unei operații de notificare cât mai uşor de realizat, astfel încat fluxul evenimentelor să fie cât mai clar.
- Utilizatorii : doresc o aplicație care să le permită trimiterea de notificări într-o manieră simplă si rapidă; de asmenea, ei pot selecta opțiunea de trimitere automată de notificari.

Fluxul evenimentelor

1. Fluxul principal

Începutul cazului de utilizare

Acest caz de utilizare începe în momentul în care utilizatorul trece cu succes de partea de autentificare.

1.1 Utilizatorul introduce numele utilizatorului şi parola.

1.2 Utilizatorul selectează din meniul principal opțiunea de vizualizare a hărții.

1.3 Utilizatorul selectează apoi opțiunea de trimitere de notificări. Vezi extensia 1.

1.4 Utilizatorul specifică destinatarii notificării. Vezi extensia 2.

1.5 Aplicația răspunde prin afişarea unui mesaj de succes sau de eroare în cazul în care cererea a eşuat.

1.6 În caz de succes este afişată harta updatată, în caz de insucces utilizatorul are opțiunea de a relua operația de trimitere de notificare sau de a reveni la meniul principal.

Sfârşitul cazului de utilizare

Utilizatorul poate părăsi acest caz de utilizare în caz de succes sau poate să il reia în caz de eşec, începând cu pasul 1.3.

2. Fluxuri alternative

2.1. Anularea operației de căutare

Începutul fluxului alternativ

Acest flux alternativ poate apărea în pasul 1.3 al fluxului principal.

2.1.1. Utilizatorul execută o comandă (apasă un buton) care reprezintă faptul că se anulează trimiterea.

2.1.2. Sistemul răspunde prin afişarea unui mesaj corespunzator, şi prin afişarea hărții.

Sfârşitul fluxului alternativ

2.2. Trimiterea automată de notificări

Începutul fluxului alternativ

Acest flux alternativ poate apărea în pasul 1.2 al fluxului principal.

2.2.1. Utilizatorul intră în meniul de setări.

2.2.2. Utilizatorul specifică în meniu faptul că doreşte trimiterea automată de notificări si destinatarii acesteia.

Sfârşitul fluxului alternativ

2.3. Căderea sistemului

Începutul fluxului alternativ

Acest flux alternativ poate apărea în oricare dintre paşii fluxului principal sau a fluxurilor alternative.

2.3.1. La căderea sistemului este necesară o nouă autentificare.

Sfârşitul fluxului alternativ

Cerințe speciale

Cerințele speciale prevăzute pentru acest caz de utilizare sunt urmatoarele:

- Timpul de răspuns

Se cere ca timpul de răspuns (perioada dintre lansarea operației de trimitere de notificări și afișarea hărții) să fie cât mai mic posibil, o limită maximă admisă fiind setată la 5 secunde.

- Utilizabilitatea

Se cere ca interfața grafică să fie ușor de folosit, să aibă elemente vizuale intuitive și amplasate în locuri optime.

Precondiții

Pentru ca acest caz de utilizare (*trimiterea notificărilor*) să poată avea loc, sunt necesare următoarele precondiții:

- Autentificare

Utilizatorul trebuie să fie autentificat cu succes în sistem.

- Funcționalitate integrală

Toate componentele sistemului trebuie să fie funcționale (baza de date, harta, GPS).

- Validitatea resurselor

Toate resursele din baza de date vor conține informații valide (nu vor fi corupte).

Postcondiții

Postcondițiile acestui caz de utilizare sunt urmatoarele:

Afișarea hărții cu locația actualizată

Puncte de extensie

Punctele de extensie reprezintă zone detaliate din specificația cazului de utilizare, create în scopul de a nu încărca prea mult fluxul principal. Punctele de extensie specificate mai jos sunt referite din fluxul principal și/sau din fluxurile alternative.

1. Selectarea opțiunii de trimitere a notificării (extensia punctului 1.3)

Utilizatorul selectează prin intermediul meniului de Options, trimiterea de notificări care va accesa o nouă fereastră pentru specificarea destinatarilor.

2. Specificarea destinatarilor (extensia punctului 1.4)

Utilizatorul selectează din meniul de trimitere a notificărilor destinatarii, prin bifarea unui checkbox:

- Către toți prietenii;
- Către prietenii care se află în zonă;
- Către anumiți prieteni;
- Către un grup de prieteni.

5.2. Aplicația client Android

5.2.1. Descriere de ansamblu

Aplicația client a fost dezvoltată cu ajutorului kitului pus la dispoziție de Android. Interfața cu utilizatorul este compusă din diverse activități implementate prin extinderea clasei de baza Activity. Acestea tratează partea de interacțiune cu utilizatorul precum și apelul către serviciile web.

Alături de partea de interfață, aplicația client oferă și servicii de notificare în momentul în care un utilizator și-a modificat poziția pe hartă, precum și tratarea excepțiilor ce pot apărea în momentul în care aplicația nu funcționează corect din cauza datelor introduse de către utilizator.

Toate aceste sunt încorporate în diagrama de pachete prezentată mai jos.

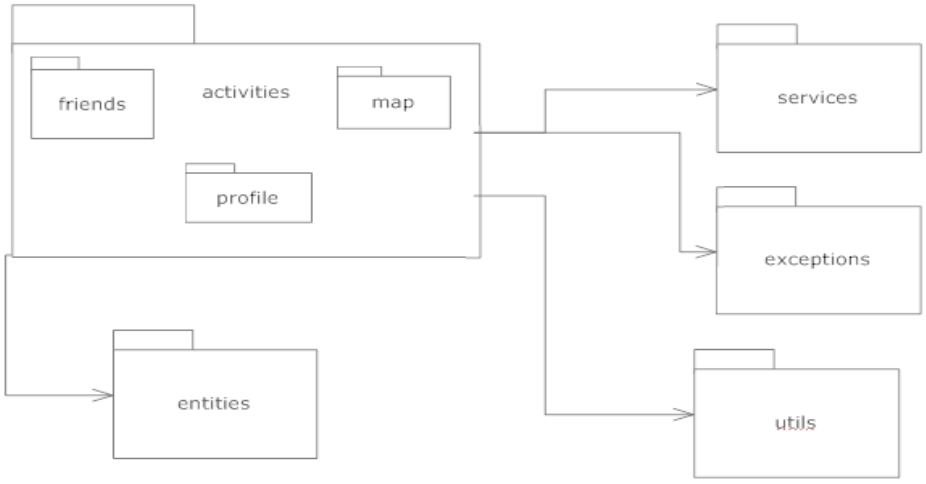

Figura 5.2 Diagrama de pachete a aplicației client

5.2.2. Implementarea interfeței cu utilizatorul

Interfața aplicației este construită pe baza modelului oferit de kitul de dezvoltare Android. Layout-urile sunt construite în fișiere xml, iar asocierea dintre aceste fișiere si date se face în cadrul claselor Activity. Această abordare permite o separare clară a modului de afișare a componentelor de partea de controller.

Aplicația permite diferite tipuri de interacțiuni cu utilizatorul. Astfel, există diferite tipuri de ecrane: obișnuit care oferă posibilitatea introducerii de date prin intermediul csuțelor de text și accesarea diferitelor funcționalități prin intermediul butoanelor, desfășurabile, de