Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare



# LUCRARE DE DIPLOMĂ

# Securizarea interacţiunilor online

**Conducător Științific:**
As. Dr. Ing. Mihai Bucicoiu

**Autor:**
Bucur Florin

București, 2015

Maecenas elementum venenatis dui, sit amet vehicula ipsum molestie vitae. Sed porttitor urna vel ipsum tincidunt venenatis. Aenean adipiscing porttitor nibh a ultricies. Curabitur vehicula semper lacus a rutrum.

Quisque ac feugiat libero. Fusce dui tortor, luctus a convallis sed, lacinia sed ligula. Integer arcu metus, lacinia vitae posuere ut, tempor ut ante.

# Abstract

Here goes the abstract about MySuperProject. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristique dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

# Cuprins

# Listă de figuri

# Listă de tabele

# Capitolul 1

# Introducere

**This is just a demo file. It should not be used as a sample for a thesis.**

**TODO:**
Remove this line (this is a TODO)

## 1.1 Scopul proiectului

This thesis presents the **MySuperProject**.

This is an example of a footnote [1]. You can see here a reference to Section 1.2.

Here we have defined the CS abbreviation. and the UPB abbreviation.

The main scope of this project is to qualify xLuna for use in critical systems.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

## 1.2 Obiectivele proiectului

We have now included Figure **??**.

[1] www.google.com

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

We can also have citations like [**?**].

# Capitolul 2

# Aplicaţii similare

## 2.1 OTR

Criptarea OTR (Off-the-Record), este un protocol criptografic folosit pentru securizarea canalelor de comunicaţie. Este întâlnit în special în aplicaţiile care presupun schimb de mesaje. Acest model de criptare poate fi folosit peste orice protocol de mesagerie, precum Yahoo Messenger, Facebook Messenger, Google Talk.

OTR foloseşte chei de criptare simetrice, aceeaşi cheie şi pentru criptare şi pentru decriptare. Procesul este reprezentat în figura 2.1. Datorită faptului că, nu se pot trimite cheile pe un canal considerat nesigur, se aplică algoritmul Diffie-Hellman pentru schimbul de chei. Astfel schimbul de chei se face în aşa fel încâ un eventual atacator care interceptează mesajele de pe canalul de comunicaţie, nu poate descifra mesajul pentru ca nu are acces la cheile folosite pentru criptarea mesajelor [7].

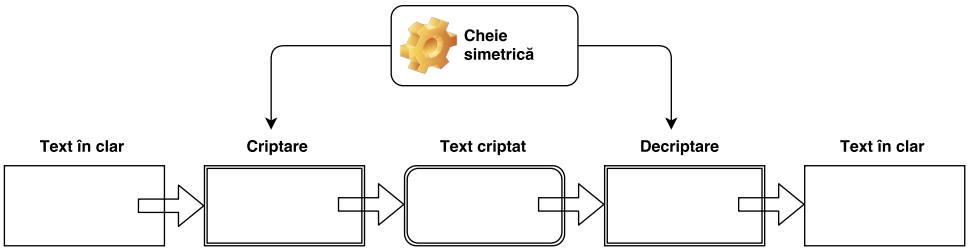


Figura 2.1: Criptarea OTR

Un alt aspect bun, din punct de vedere al securităţii, este faptul că pentru fiecare mesaj nou transmis, cheia de criptare este schimbată. Deci, chiar dacă cineva ar avea acces la schimbul de mesaje, pentru acesta ar fi imposibil să descifreze conţinutul.

O problemă a protocolului o reprezintă faptul că nu se folosesc semnături digitale, astfel este imposibil de demonstrat dacă cineva a reuşit să falsifice nişte mesaje. Această problemă se poate rezolva prin introducerea conceptului de autentificare a utilizatorilor. Folosind autentificarea, se pot găsii mesajele falsificate [7].

Această criptare este disponibilă în mai multe aplicaţii de chat. Una foarte importantă este Pidgin, aceasta fiind disponibilă pe Linux, Windows dar şi pe OS X. Pentru dispozitivele mobile, cum ar fi cele care folosesc ca sistem de operare Android, există ChatSecure. Astfel tot ce trebuie făcut este să se instaleze aceste aplicaţii în funcţie de platformele folosite, activată opţiunea de utilizare OTR şi bineînţeles autentificarea utilizatorului. Alte aplicaţii Android care folosesc criptarea OTR pentru schimbul de mesaje sunt Xabber şi Beem. Amândouă pot fi descărcate şi folosite gratis.

### 2.1.1 Xabber

Xabber, este o aplicaţie Android care foloseşte pe partea de comunicaţie protocolul XMPP, protocol ce este conceput în special pentru schimbul de mesaje şi este bazat pe limbajul XML. XMPP este un proiect open-source, care este standardizat şi dă posibilitatea oricui să dezvolte aplicaţii bazate pe XMPP.

Această aplicaţie vine însoţită de servicii preconfigurate ce folosesc XMPP şi oferă posibilitatea integrării contactelor din sistemul de operare cu orice cont XMPP. De asemenea este compatibil cu mai multe standarde, precum servere Ejabberd sau Openfire. Schimbul de mesaje se face similar cu Internet Relay Chat(IRC), iar mesajele pot fi criptate folosind OTR [1].

### 2.1.2 Beem

Beem, este un proiect open-source care îsi doreşte să ofere posibilitatea de a folosi toate funcţionalităţile puse la dispoziţie de Jabber şi XMPP. Este compatibil cu orice standard bazat pe XMPP(Google Talk, Prosody, Openfire).

Această aplicaţie oferă posibiliatea de a schimba mesaje cu diferite persoane din lista de contacte. Mesajele pot fi criptate folosind OTR [2].

## 2.2 PGP

Criptarea PGP sau criptarea Pretty Good Privacy are la baza un standard numit OpenPGP, care este folosit pentru criptarea de date si semnături digitale. PGP foloseşte atât chei publice, cât şi chei private.

Cu ajutorul cheii publice, se criptează mesajul, iar cu ajutorul cheii private, se decriptează mesajul. Astfel, fiecare utilizator generează o cheie publică pe care o pune la dispoziţie altor utilizatori, iar cheia privată este ţinută secretă, deci doar cel care deţine cheia privataă poate decripta mesajele. Procesul este reprezentat în figura 2.2

Un mare avantaj al folosirii de chei publice este posibilitatea de a utiliza semnături digitale. Cu ajutorul acestora, se pot semna datele transmise, astfel încât, orice încercare de falsificare a datelor sau de modificare a datelor este depistată foarte uşor. Semnăturile digitale sunt create cu ajutorul unor algoritmi matematici care combină cheia privată

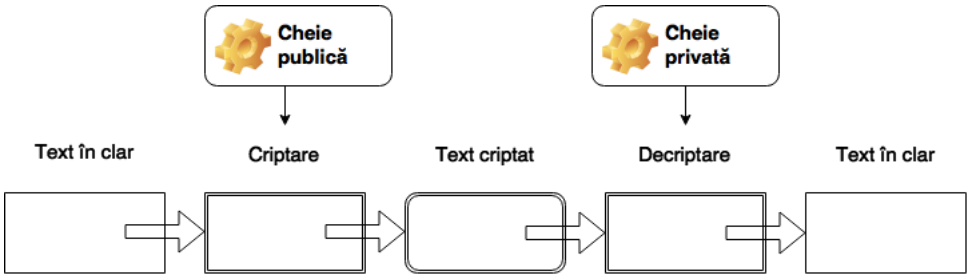


Figura 2.2: Criptarea PGP

cu datele care urmează să fie transmise. Ulterior, dacă cineva doreşte să verifice autenticitatea datelor primite sau integritatea lor, va aplica pe datele recepţionate, cheia publică a transmiţătorului.

Pentru a folosi criptarea PGP, în primul rând trebuie să generăm un certificat, care poate să conţină date extra, cum ar fi numele, adresa de email sau alte semnături digitale ale altor utilizatori. Un certificat se poate genera pe orice sistem de operare, iar acesta poate fi pus la dispoziţia altor oameni, facându-le publice pe anumite servere dedicate cum ar fi, serverul pentru chei publice al celor de la MIT.

Astfel, dacă cineva intenţionează să comunice cu o anumită persoană, trebuie să îi caute cheia publică pe aceste servere dedicate, dacă nu cumva o deţine deja. Acum poate cripta mesajul şi îl poate trimite.

Decriptarea presupune cunoaşterea unei parole care securizează cheia privată. Deci mesajele recepţionate sunt decriptate folosind cheia privată deţinută.

Pentru schimbul de chei publice există mai multe soluţii. O soluţie este de a obţine cheia publică de la partener, chiar în persoană, dar nu este cea mai comodă posibilă. Există conceptul de fingerprint, care reprezintă un cod hash al certificatului în hexazecimal, care poate fi verificat telefonic. O altă modalitate este de a obţine cheile publice folosind mai multe surse, verificând dacă toate coincid.

Această criptare este disponibilă în mai multe aplicaţii care presupun schimb de mesaje. Ca de exemplu, pentru Android există APG, PGP SMS, Symantec PGP Viewer.

### 2.2.1 Android Privacy Guard

APG este un proiect gratuit şi open-source, ce permite criptarea şi decriptarea de fişiere şi mesaje din email-uri. Pe lângă criptare şi decriptarea folosind chei publice şi chei private, oferă şi posibilitatea de a semna digital fişierele şi mesajele. De asemenea, se poate folosi şi criptarea pe baza de chei simetrice. Criptarea bazată pe chei publice - private, se realizează folosind protocolul OpenPGP [3].

### 2.2.2 PGP SMS

Această aplicaţie foloseşte criptarea PGP utilizând chei publice pentru criptare şi chei private pentru decriptare. Cheia publică este trimisă persoanei care doreşte să trimită

un mesaj criptat. Aceasta criptează mesajul folosind cheia primită şi transmite mesajul. La receptţie, mesajul este decriptat cu ajutorul cheii private [4].

### 2.2.3 PGP Viewer

PGP Viewer permite decriptarea mesajelor din email-uri, fişierelor sau a ataşamentelor din emai-uri. Rezultatul decriptării este copiat în directorul de intrăre a email-ului. Clientul de email folosit este cel standard al sistemului de operare Android. Astfel nu este necesară instalarea de noi aplicaţii.

Informaţiile criptate din email sunt disponibile, chiar dacă utilizatorul nu este conectat la Internet. Aplicaţia mai oferă şi posibilitatea de a verifica semnăturile digitale ale datelor, pentru a valida faptul că datele nu sunt falsificate [5].

# Capitolul 3

# Tehnologii folosite

În acest capitol se introduce sistemul de operare Android, arhitectura sa şi funcţionalităţile puse la dispoziţie de acesta. De asemenea sunt prezentate şi detalii referitoare la librăria oferită de Facebook pentru dezvoltarea de aplicaţii Android care interacţionează cu reţeaua de socializare. Tot în acest capitol se vor regăsii amănunte legate de paşii ce trebuie urmaţi pentru creare unei aplicaţii Facebook.

## 3.1 Android

### 3.1.1 Prezentare generală

Android este un sistem de operare creat pentru dispozitivele mobile, cum ar fi smartphone-uri şi tablete. Acest sistem este creat şi promovat de Google. De asemenea acest sistem de operare este disponibil pe mai multe tipuri de dispozitive, produse de un numar mare de companii [6].

Android este bazat pe o versiune modificata de Linux, împrumutând tehnologii care se ocupă de gestiunea componentelor hardware, a memoriei sau a proceselor. Tot odată foloseşte biblioteci Java pentru partea de grafică, telefonie sau pentru partea de conectivitate.

Un aspect important al acestui sistem de operare este faptul că este un produs open-source, deci producătorii de dispozitive mobile, pot modifica sistemul de operare astfel încât, să se poată folosi extensii ale hardware-ului, deşi sistemul de operare de bază nu oferă suport pentru aceste extensii.

În această secţiune se vor introduce mai întâi arhitectura Android şi apoi se vor detalia componentele sale utilizate.

### 3.1.2 Arhitectura Android

Arhitectura Android este compusă din patru elemente importante: Linux Kernel, Librării, FrameWorks şi Aplicaţii. În continuare vom detalia fiecare nivel al arhitecturii

folosindu-ne şi de Figura 3.1.

Primul nivel este Linux Kernel, care este folosit pentru comunicaţia dintre componentele hardware şi componentele software.

Următorul nivel al arhitecturii este reprezentat de librăriile de bază. Aceste librării sunt scrise în limbajele de programare C/C++. Aici găsim librăriile pentru grafică, pentru baze de date, pentru browser-ul WEB.

Android RunTime este compus din maşina virtuală Dalvik şi de librării ale procesorului.

Frameworks, este scris în totalitate în limbajul Java şi este folosit pentru dezvoltarea de aplicaţii Android.

Ultimul nivel este reprezentat de aplicaţiile Android. Acesta este nivelul cel mai înalt, aici găsindu-se aplicaţii precum cele pentru contacte, jocuri sau browser-ul.

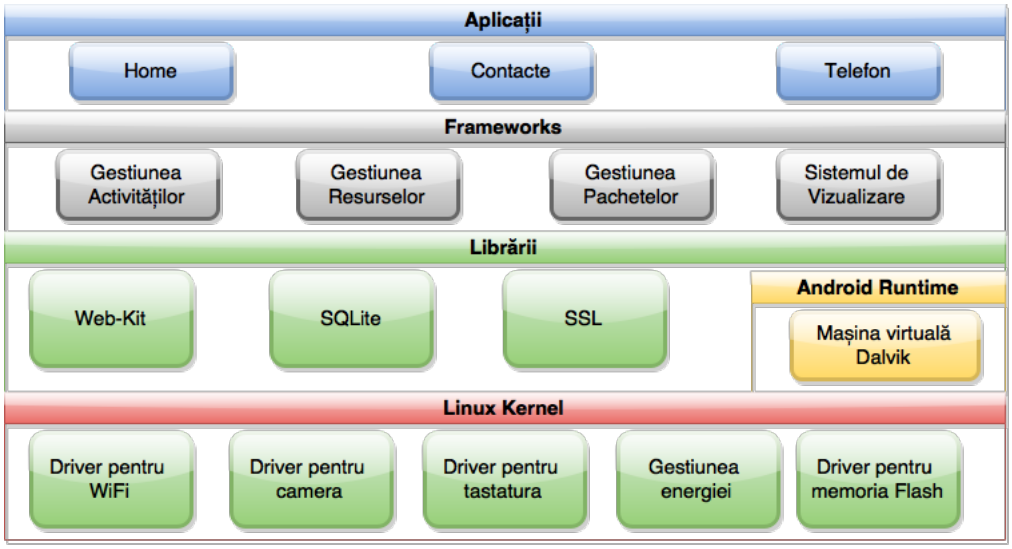


Figura 3.1: Arhitectura Android

### 3.1.3 Event Bus

Event Bus oferă o modalitate optimizată şi simplificată de a comunica între activităţi, fragmente, fire de execuţie sau servicii Android. Folosind această librărie se optimizează procesul de dezvoltare, iar calitatea comunicaţiei este mai bună decât în cazul comunicaţiei standard. Principiul de funcţionare Event Bus este descris în Figura 3.2.

### 3.1.4 Notification Listener Service

Notification Listener Service, este un serviciu folosit pentru gestionarea notificărilor. Cu ajutorul lui la primirea sau recepţionarea unor notificări se pot declanşa activităţi bazate pe datele care vin odată cu notificările.

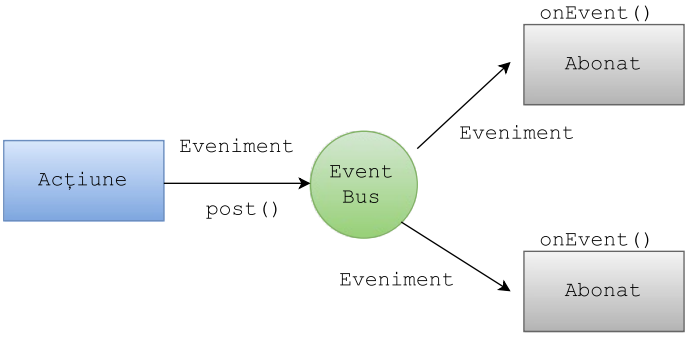


Figura 3.2: Principiul de funcţionare Event Bus

Acest serviciu se foloseşte de un obiect StatusBarNotification, prin intermediul căruia se primesc informaţiile necesare despre notificările existente.

Pentru a putea folosi acest serviciu, utilizatorul trebuie să activeze opţiunea de a permite aplicaţiei să se folosească de notificările primite în sistem. Opţiunea se găseşte în calea "Settings > Security > Notification access"

Comunicaţia dintre acest serviciu şi activitatea care prelucrează informaţiile primite odată cu notificările se realizează cu ajutorul librăriei Event Bus, prezentată mai sus.

### 3.1.5 Recycler View

RecyclerView reprezintă o versiune mult mai flexibilă faţă de ListView, folosită pentru a afişa pe ecran o listă cu un număr mare de elemente, dar într-o modalitate mult mai eficientă faţă de ListView.

Elementele listei nu sunt încărcate toate în memorie, ci doar cele afişate. Se pot adăga elemente sau şterge elemente foarte uşor şi într-un mod dinamic, existând în mod implicit activate animaţii pentru evenimentele de adăgare sau ştergere de elemente.

## 3.2 Facebook SDK

### 3.2.1 Prezentare generală

Facebook, este una dintre cele mai mari şi importante reţele de socializare din prezent. Platforma Facebook, este disponibilă pe toate tipurile de dispozitive, având suport pentru majoritatea sistemelor de operare.

Cu ajutorul Facebook SDK, se pot crea aplicaţii care implică autentificarea în reţeaua de socializare şi se pot folosii anumite funcţionalităţi ale platformei, direct din aplicaţiile care nu sunt create de Facebook. De exemplu, putem crea o aplicaţie care să ofere posibilitatea de a posta un mesaj pe wall-ul personal.

Pentru a folosi Facebook SDK, este necesar un cont de Facebook, în care se crează o nouă aplicaţie. Se alege un nume al aplicaţiei şi se generează un identificator unic al aplicaţiei care arată ca în Figura 3.3. Pentru ca aplicaţia să fie funcţională, acest id unic trebuie inclus in codul acesteia. După ce id-ul aplicaţiei este obţinut, trebuie să includem în proiect SDK-ul oferit de Facebook şi setat id-ul.


Figura 3.3: ID-ul aplicaţiei

Pe lângă toate acestea, este necesară o cheie pentru dezvoltare şi una pentru publicarea aplicaţiei. Acestea pot fi identice şi sunt reprezentate în Figura 3.4.

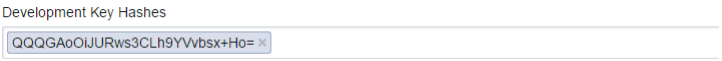


Figura 3.4: Cheia pentru dezvoltare

Fiecare sesiune deschisă, are asociat un token de acces. Acesta poate sa difere de la sesiune la sesiune, chiar dacă este vorba despre acelaşi user. Access token-ul este generat în funcţie de permisiunile cerute de aplicaţie sau de modificările apărute în profilul utilizatorului. Acest token este o modalitate de identificare, dar şi de securizare a sesiunilor. Un exemplu de token este prezent în Figura 3.5.

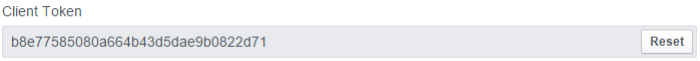


Figura 3.5: Token de acces

Un alt aspect important al SDK-ului de Facebook, este faptul ca s-au introdus două tipuri de id-uri ale utilizatorului. Astfel, pentru un utilizator, există un id, global sau real şi un id unic pentru fiecare aplicaţie cu ajutorul careia s-a autentificat. Cu aceste id-uri unice per aplicaţie se limitează accesul la anumite funcţionalităţi. De exemplu, deschiderea unui ferestre de Facebook Messenger, direct pentru un utilizator specific, nu este posibilă, pentru că id-ul acestuia nu este cel global, iar aplicaţia nativă de Facebook Messenger, nu va recunoaşte id-ul.

La începutul acestei practici, de a pune la dispoziţie dezvoltatorilor de aplicaţii, funcţionalităţile reţelei de socializare, a fost un lucru bine văzut de toată lumea. Cu timpul, securitatea datelor personale şi intimitatea utilizatorilor au reprezentat nişte puncte de reper foarte importante pentru evoluţia SDK-urilor. În momentul scrierii acestei lucrări de licenţă, cel mai recent SDK este Facebook SDK 4.2.0.

De anul acesta Facebook, a renunţat la ideea de a lăsa dezvoltatorii să işi facă propriile lor aplicaţii pentru schimbul de mesaje private. Prin aceste limitări, se pot evita aplicaţiile robot, care fac spam şi nu sunt utile cu nimic. O altă modalitate de a evita spam-ul este de a forţa folosirea aplicaţiilor native pentru anumite operaţii.

### 3.2.2 Graph API

Principala metodă de a obţine, şterge sau adăuga informaţii în reţeaua de socializare, este Graph API. Graph API este un API de nivel jos, bazat pe protocolul de comunicaţie, HTTP. Denumirea acestuia vine de la ideea de graf social, în care găsim elemente cum ar fi, noduri, muchii sau câmpuri de informaţii. Nodurile reprezintă lucruri de tipul utilizator, fotografie sau comentariu. Muchiile reprezintă conexiunile dintre noduri, ca de exemplu fotografiile dintr-o pagină. Câmpurile de informaţii reprezintă descrierea nodurilor, de exemplu ziua de naştere a unui utilizator sau numele unei pagini.

Graph API poate fi folosit cu orice limbaj de programare ce oferă suport pentru cereri HTTP. Pentru a obţine informaţii de pe platforma Facebook, trebuie creată o cerere GET. Pentru a adaugă mesaje, imagini, link-uri sau orice alte date, se foloseşte cereri de tip POST, iar pentru a şterge date, se foloseşte o cerere specială, DELETE. Fiecare cerere Graph API necesită un token de acces. Token-ul de acces este cel obţinut prin implementarea autentificării în reţeaua de socializare.

Versiunea actuală de Graph API este 2.3. Pentru că există mai multe versiuni, se poate specifica în cereri versiunea de API folosită. Totuşi, este indicat să se folosească ultima versiune, deoarece, funcţionalităţ din cele vechi pot dispărea sau se pot modifica.

O cerere GET folosind Graph API este de forma https://graph.facebook.com/v2.3/node-id/edge-name, unde node-id reprezintă identificatorul nodului, de exemplu id-ul unui prieten, iar edge-name poate fi "photos" pentru pozele acelui prieten. Datele întoarse de Facebook ajung în format JSON, putând fi parsate foarte uşor.

O cerere POST folosind Graph API, poate fi de forma https://graph.facebook.com/v2.3/node-id/edge-name, unde node-id poate fi identificatorul unui grup, iar edge-name poate fi şirul de caractere "members". Această cerere are nevoie să i se adauge un cămp extra, cum ar fi "member", acestui câmp având asociat ca valoare identificatorul unei persoane care va fi adăugată în grup.

În cazul în care se doreşte ştergerea unui element de pe reţeaua de socializare, cerearea arată similar cu cea de POST, dar în loc să se specifice metoda POST, se va seta ca metodă folosită cea de DELETE.

# Capitolul 4

# Proiectarea arhitecturii

Această aplicaţie are rolul de a aduce un plus de securitate interacţiunilor online. În acest caz este vorba de criptarea mesajelor private transmise prin intermediul unei reţele de socializare, dar şi a postărilor din anumite spaţii private de comunicare.

Datele criptate sunt transmise pe aceleaşi canale folosite şi de datele necriptate. Astfel, aplicaţia crează puncte de acces la platforma reţelei de socializare, dar în cazul în care securitatea este compromisă, informaţiile considerate de utilizator importante vor rămâne indescifrabile.

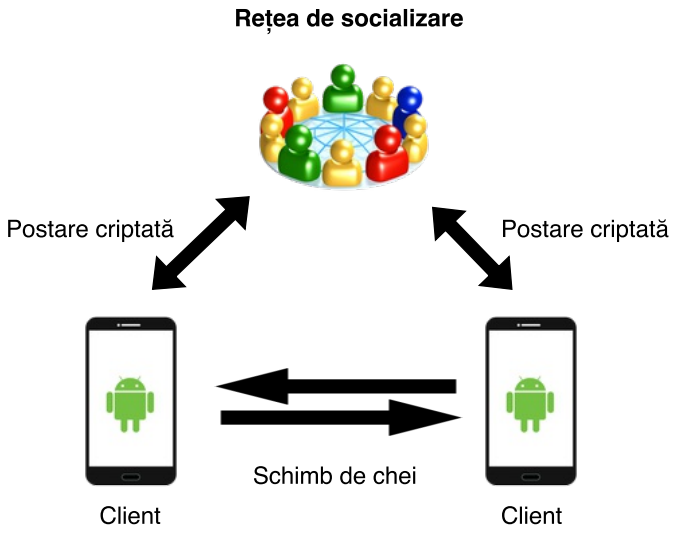


Figura 4.1: Arhitectură

## 4.1 Conectarea la reţeaua socială

Utilizarea aplicaţiei presupune ca utilizatorul să deţină un cont pe reţeaua de socializare, cu ale cărui credenţiale se poate realiza autentificarea. În cazul în care contul este setat deja în sistemul de operare, acesta se preia automat de către aplicaţie şi se autentifică utilizatorul. De asemenea, este necesar ca dispozitivul să aibă instalată aplicaţia nativă pusă la dispoziţie de platforma reţelei.

Procesul de autentificare printr-o altă aplicaţie decât cea nativă necesită permisiunea utilizatorului de a accesa anumite date din contul său. Dacă utilizatorul este de acord cu accesarea datelor de către aplicaţie, înseamnă ca procesul s-a terminat cu succes.

Dacă autentificare s-a încheiat cu succes, aplicaţia primeşte din partea platformei reţelei sociale un token de acces, care a fost decris în secţiunea 3.2.1. Cu acesta se vor deschide noi sesiuni de comunicaţie între aplicaţia client şi reţeaua de socializare.

Dacă autentificarea s-a încheiat cu succes, utilizatorul poate beneficia de funcţionalităţile disponibile. Pentru început se va afişa lista de prieteni care, de asemenea folosesc această aplicaţie. Prin selectarea unui prieten, se va deschide o nouă activitate de unde se pot iniţia conversaţii sau gestiona grupuri.

Pentru a iniţia o conversaţie este nevoie de aplicaţia nativă de Messenger a reţelei de socializare. Mesajul de iniţiere este "Hello", alt tip de mesaj este ignorat. În momentul în care mesajul ajunge la destinatar, apare mesajul Hello şi se ştie că se poate începe schimbul de mesaje criptate. Cel care a primit mesajul, poate răspunde prin completarea unui câmp text, ulterior apăsând pe un buton.

## 4.2 Grupuri protejate

## 4.3 Partajare chei

## 4.4 Conversaţii criptate

# Capitolul 5

# Implementare

În acest capitol se vor prezenta detalii despre modul de implementare a funcţionalităţilor aplicaţiei. Se vor oferii amănunte legate de interfaţa grafică, procesul de autentificare, gestionare a grupurilor, schimbul de mesaje private şi criptarea, respectiv decriptarea datelor transmise.

Pe scurt, aplicaţia oferă posibilitatea autentificării pe baza datelor de acreditare a unui cont pe reţeaua de socializare Facebook şi a folosirii anumitor funcţionalităţi puse la dispoziţie de Graph API. Gestionarea grupurilor se face cu ajutorul metodelor oferite de platforma Facebook, iar schimbul de mesaje, respectiv criptarea si decriptarea lor nu au suport pentru dezvoltarea de aplicaţii Android. Pentru partea de mesaje private este folosit un serviciu de gestionare a notificărilor primite în sistemul de operare Android. Criptarea datelor este de tip AES şi foloseşte chei simetrice.

## 5.1 Implementarea aplicaţiei

### 5.1.1 Interfaţă grafică

Dezvoltarea de aplicaţii Android este foarte flexibilă putând implementa un lucru folosind mai multe metode. Flexibilitatea se poate observa în faptul că partea de interfaţă grafică poate fi separată de partea de cod care implementează functionalităţile interfeţei grafice. Astfel pentru a dezvolta interfaţa grafică se crează fişiere de tip XML în care se definesc elementele grafice, cât şi proprietăţile lor, cum ar fi culoare, dimensiune sau poziţionarea sa pe ecran.

Pentru această aplicaţie am creat 6 fişiere XML, reprezentând cinci layout-uri ale interfeţei grafice: login_fragment.xml, activity_main.xml, activity_friends.xml, friend_row.xml, chat_activity.xml şi group_row.xml.

Primul dintre ele este login_fragment.xml, care este folosit pentru fereastra de autentificare în aplicaţie. Acesta este format dintr-un buton de autentificare, buton pus la dispoziţie de API-ul de Facebook, un element de tip TextView folosit pentru a afişa un mesaj de întâmpinare dupa ce procesul de autentificare se termină cu succes şi un obiect de tip ProfilePictureView, care este oferit tot de API-ul de Facebook, afişând imaginea

de profil a utilizatorului proaspă autentificat. Acestea sunt parte a unui LinearLayout cu orientarea verticală.

Următorul este activity_main.xml, în care este inclus fragmentul de autentificare. Conţine un framelayout care ajută la definirea şi includerea fragmentului de autentificare în interfaţa grafică.

Urmează activity_friends.xml, care conţine doar un element de tip RecyclerView şi este folosit pentru a afişa pe ecran o listă cu toţi prietenii care folosesc această aplicaţie. Un rând din listă este reprezentat printr-un layout compus dintr-un element de tip ProfilePictureView, care afişează imaginea de profil al prietenului şi un element de tip TextView folosit pentru afişarea numelui prietenului. Prin selectarea unui prieten din această listă se deschide o nouă activitate, care este reprezentată prin fişierul chat_activity.xml.

Layout-ul chat_activity.xml este compus dintr-o parte folosită pentru chat şi o parte folosită pentru gestionarea de grupuri. Partea de chat conţine un element ProfilePictureView pentru imaginea de profil al prietenului, un TextView în care apare un mesaj nou primit, un camp EditText în care se poate introduce un răspuns la un mesaj primit. Pe lângă acestea mai există doua elemente de tip buton. Unul este "Reply", care prin apăsarea lui se trimite răspunsul completat în EditText persoanei de la care a venit mesajul. Al doilea, "Initiate conversation", care prin apăsarea lui se deschide aplicaţia nativă de Facebook Messenger. Astfel se poate iniţia o conversaţie.

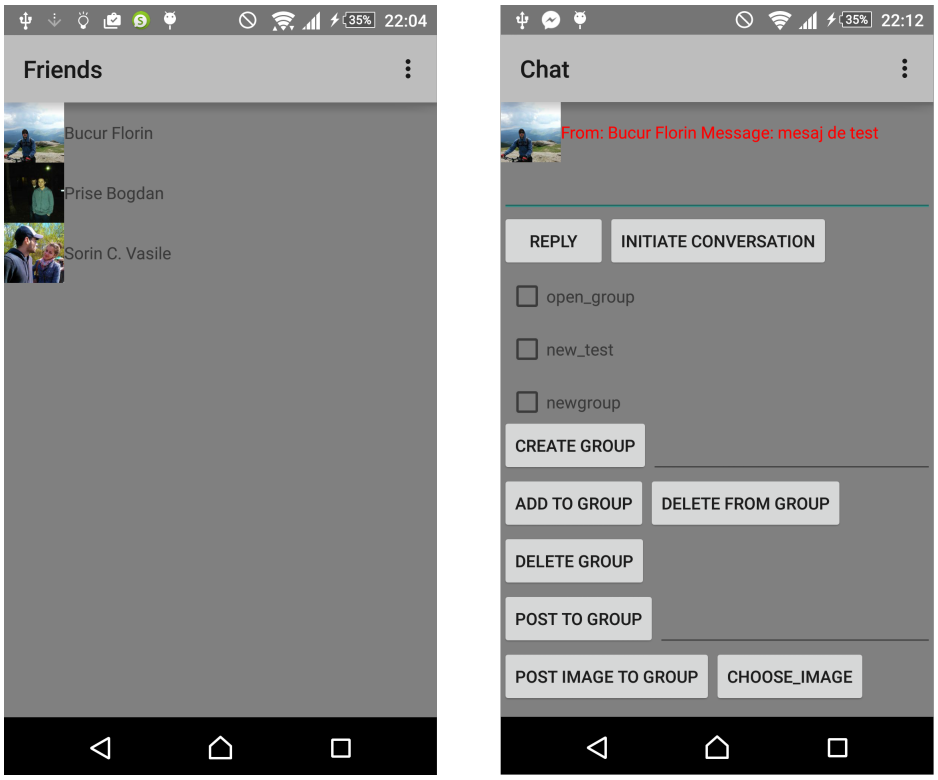


Figura 5.1: Interfaţă grafică

Partea de grupuri are în componenţă un buton "Create group", care are asociat un camp

de tip EditText. În momentul în care se apasă butonul se preia textul din câmpul text şi se crează un nou grup al aplicaţiei care va avea numele regăsit în EditText-ul asociat. De asemenea şi aici există un element de tip RecyclerView, folosit pentru afişarea unei liste de grupuri existente, care se poate actualiza în funcţie de operaţiile efectuate, de exemplu, în momentul în care se crează un grup nou, lista de grupui se actualizează imediat ce operaţia s-a terminat cu succes.

Fiecare rând din obiectul RecyclerView este reprezentat prin doua elemente de grafică. Un element de tip CheckBox, folosit pentru selectarea grupului asupra căruia urmează să se efectueze o acţiune şi un TextView pentru afişarea numelui grupului. Mai sunt prezente butoane pentru adăgare utilizator în grupul selectat sau ştergere din grupul selectat. Persoana care este adăugată sau eliminată din grup este cea selectată din lista de prieteni prezentată anterior. Butonul "Delete group" este folosit pentru ştergerea în întregime a unui grup. Butonul "Post to group" are asociat un câmp EditText, iar prin apăsarea butonului se preia textul tastat în acest câmp şi este postat în grupul selectat.

Pentru implementarea elementelor de tip RecyclerView este necesară crearea de mai multe clase. O clasă care va reprezenta un obiect de tip Adapter ce se va ocupa de iniţializarea listei din interfaţa grafică. La crearea acestui obiect trebuie pasate ca argumente o listă care conţine datele ce vor fi afişate şi contextul în care vor fi afişate. De asemenea, ascultătorii de tip apăsare pe un rând al listei sau apăsare a unui anumit element dintr-un rând, sunt definite tot în această clasă. O altă clasă reprezintă un obiect de tip ViewHolder, care va conţine elementele grafice folosite. Un rând din listă reprezintă un obiect care conttţine informaţiile ce vor fi completate pe acel rând.

### 5.1.2 Autentificare

Modulul de autentificare este reprezentat printr-o clasă Java, FacebookLoginFragment, care extinde clasa Fragment. Procesul de autentificare presupune, completarea cămpurilor user şi parolă din interfaţa grafică pusă la dispoziţie de Facebook SDK prin butonul de autentificare. În cazul în care autentificarea a fost efectuată anterior, numele de utilizator şi parola acestuia sunt preluate din sistemul de operare, mai exact din secţiunea "Accounts".

Dacă datele de autentificare sunt corecte, se va obţine un token de acces valid, căruia îi sunt asociate şi permisiunile necesare utilizării normale a aplicaţiei. Cum am spus şi în secţiunea 3.2.1, pentru ca procesul de autentificare să funcţioneze, trebuie să avem creată o aplicaţie pe platforma Facebook, inclus Facebook SDK în proiect, generat identificatorul aplicaţiei şi adăugat în codul aplicaţiei şi generată o cheie pentru profilul dezvoltatorului.

Autentificarea începe în momentul în care se apasă butonul din interfaţa grafică. Butonul acesta se ocupă şi de permisiuni, dar urmăreşte şi starea în care se află procesul. Astfel la terminarea procesului, se afişează un mesaj în funţie de rezultatul autentificării.

În metoda onCreate a fragmentului trebuie să se iniţializeze Facebook SDK şi un obiect de tip CallbackManager folosit pentru gestionarea operaţiilor de tip FacebookCallback. De asemenea, tot aici, datorită faptului că token-urile de acces nu sunt permanente, pot expira în orice moment, se verifică şi validitatea token-ului. Acest lucru se verifică

folosid un obiect de tip AccessTokenTracker şi unul de tip ProfileTracker. Astfel în cazul în care se schimbă token-ul de acces în timp ce aplicaţia rulează, obiectul AccessTokenTracker va informa ProfileTracker-ul de schimbare, astfel, codul din metoda suprascrisă, "onCurrentProfileChanged", actualizează id-ul utilizatorului. Deci având în orice moment token-ul de acces actualizat, nu se vor sesiza schimbările din spate.

Elementele de grafică şi parametrii pentru butonul de autentificare se iniţializează în metoda "onViewCreated". Se initializează obiectele de tip ProfilePictureView şi TextView asociindu-le identificatorii din layout-ul de autentificare. Pentru butonul de autentificare, se setează fragmentul curent, se setează un vector de permisiuni şi se înregistrează apelul cererii de autentificare. În momentul în care procesul de autentificare se finalizeazăcu succes, se verifică încă odată validitatea token-ul de acces şi se porneşte o nouă activitate, cea pentru lista de prieteni.

### 5.1.3 Listă de prieteni

Prietenii care folosesc aplicaţia sunt afisaţi într-o listă de tip RecyclerView şi este obţinută cu ajutorul unei interogări a platformei Facebook prin intermediul Graph API. În metoda onCreate a activităţii FriendsActivity se aşteaptă ca aceasta să fie invocată prin intermediul unui obiect de tip Intent. Dacă activitatea este invocată, se crează un obiect de tip GraphRequestBatch prin intermediul căruia se realizează comunicaţia între aplicaţie şi platforma Facebook.

Cererea pentru lista de prieteni este implementată în metoda newMyFriendsRequest, pusă la dispoziţie de către Graph API. Pentru ca solicitarea să fie validă această metodă primeşte ca parametru token-ul de acces al utilizatorului.

Informaţiile primite de la Facebook sunt în format JSON şi conţin numele prietenilor şi identificatorii acestora. În metoda onCompleted a cererii se parsează datele recepţionate şi se populează un obiect de tip List care va conţine perechi "nume", "identificator". În momentul în care datele sunt gata de afişat, se iniţializează obiectul RecyclerView cu lista de prieteni.

### 5.1.4 Grupuri

Grupurile de Facebook sunt create pentru a da posibilitatea persoanelor care au aceleaşi interese să se reunească într-o comunitate restrânsă de utilizatori. Acestea pun la dispoziţie membrilor un spaţiu special de comunicare, care poate fi accesat doar de cei care fac parte din grup. Grupurile pot fi de trei feluri. Pot fi publice, adică oricine poate deveni membru dacă face o cerere, la care poate să răspundă oricare membru sau dacă primeşte o invitaţie, trimisă tot de un membru oarecare. Conţinutul celor publice poate fi vizualizat şi de persoanele din afara grupului. Grupurile închise oferă aceleaşi privilegii ca şi cele publice, doar că, informaţiile împărtăşite pe grup nu se pot citi din afara acestuia. Al treilea tip de grup este cel secret. În grupurile secrete pot intra doar utilizatorii care primesc invitaţii de la cei care fac parte deja din grup. De asemenea conţinutul grupului nu poate fi văzut în afara lui.

O altă clasificare a grupurilor de Facebook este după modalitatea de creare. Ele pot fi create de aplicaţii care nu sunt dezvoltate de cei de la Facebook sau create de aplicaţiile native ale platformei. Aplicaţiile care nu sunt proprietatea platformei reţelei de socializare pot crea grupuri ale aplicaţiei sau grupuri pentru jocuri. Membrii acestor grupuri sunt gestionaţi doar de aplicaţia care a creat grupurile. Astfel, în aceast proiect, în lista de grupuri afişată se regăsesc doar acele grupuri care au fost create de această aplicaţie.

### Listarea grupurilor

Pentru a obţine aceste grupuri se face o cerere prin intermediul Graph API, folosind un obiect de tip GraphRequest. Acesta trebuie să primească parametrii precum token-ul de acces al aplicaţiei, metoda HTTP folosită, în acest caz HTTPMethod.GET, şi calea pentru informaţiile dorite. Calea este de forma "[identificatorul_aplicaţiei]/groups". Pentru că folosim token-ul de acces al aplicaţiei, nodul din graful reţelei de socializare cerut este reprezentat de identificatorul aplicaţiei, cel obţinut folosind paşii din secţiunea 3.2.1. În metoda onCompleted a cererii este primită lista de grupuri sub formă de JSON. Aceste informaţii sunt parsate şi este populat obiectul care reţine grupurile sub formă de perechi identificator_grup, nume_grup. Tot aici, dupa ce datele sunt gata de afişare, se actualizează interfaţa grafică.

### Crearea

Pentru a crea un nou grup, se trimite o solicitare de tip POST tot cu ajutorul unui obiect GraphRequest, folosind metoda oferită de Graph API, newPostRequest. Acestei metode i se pasează ca argumente, token-ul de acces al aplicaţiei şi calea din graful reţelei. Şi de această dată calea este de forma "[identificatorul_aplicaţiei]/groups", dar cererii îi mai este asociate şi alte date, precum numele pe care o să îl aibă noul grup, identificatorul administratorului, dacă se doreşte setat acest lucru şi informaţii despre tipul grupului, cum ar fi tipul public. Datele primite în urma acestei solicitări vin tot sub formaă de JSON şi conţin identificatorul noului grup creat. După parsarea datelor, se actualizează lista de grupuri din interfaţa grafică.

### Adăugarea de membri

Adăugarea de persoane în grup se realizează tot prin intermediul unei cereri, folosind Graph API. Pentru ca solicitarea şă fie validă, metodei newPostRequest îi trebuie transmise token-ul de acces al aplicaţiei şi calea din graful reţelei, care va fi un şir de caractere de forma "[identificatorul_grupului]/members". Acestei cereri, îi va fi ataşat parametrul "member" având ca valoare identificatorul persoanei care urmează să fie adăugată în grup.

### Ştergerea de membri

Ştergerea de persoane din grup se face asemănător cu adăugarea lor. Şi aici este nevoie de token-ul de acces al aplicaţiei, iar calea din graful reţelei este la fel, adică "[iden-

tificatorul_grupului]/members". Diferenţa vine din faptul că este specificată metoda HTTP folosită. În acest caz, HttpMethod.DELETE. Parametrul ataşat cererii este tot "member" şi are ca valoare tot identificatorul utilizatorului care urmează să fie şters.

#### Ştergerea de grupuri

Grupurile create de aplicaţiei pot fi şterse şi ele, dar condiţia este ca acestea să nu mai aibă niciun membru. Astfel, dacă un utilizator doreşte să şteargă un grup, trebuie mai întâi să elimine toţi membrii din acesta. Astfel printr-o cerere de tip GraphRequest, se obţine lista membrilor. Pentru această solicitare, este nevoie de token-ul de acces al aplicaţiei, iar calea din graful reţelei este "[identificator_grup]/members" şi de specificarea metodei folosite, adică HttpMethod.GET. Lista este întoarsă sub formă de JSON, este parsată şi pentru fiecare membru se foloseşte metoda de ştergere al unei persoane prezentată mai sus. După această operaţie se actualizează şi lista de grupuri din interfaţa grafică.

#### Postarea de mesaje text

Pentru postarea de mesaje text în grup se foloseşte din nou GraphRequest cu metoda newPostRequest, care va primi ca argumente, token-ul de acces al aplicaţiei şi calea din graful reţelei de socializare, "[identificatorul_grupului]/feed". Acestei cereri îi este ataşat parametrul "message" care va avea ca valoare mesajul text care urmează să fie postat. Aceste postări apar pe grup, având ca autor această aplicaţie.

#### Listarea postărilor

Postările regăsite pe grupuri pot fi preluate în aplicaţie folosind tot Graph API. Se trimite o solicitare de tip GraphRequest care primeşte token-ul de acces curent al aplicaţiei, calea din graful reţelei, "[identificatorul_grupului]/feed" şi metoda HTTP folosită, HttpMethod.GET. Datele primite în urma acestei cereri sunt sub formă de JSON. Acestea sunt parsate, căutându-se dupa cuvîntul cheie "message".

### 5.1.5 Trimitere - Recepţie mesaje

Schimbul de mesaje folosind infrastructura Facebook, nu se mai poate implementa în aplicaţiile care folosesc instrumentele puse la dispoziţie de această platformă. Această idee, de a bloca dezvoltatorii să creeze aplicaţii pentru schimbul de mesaje prin intermediul reţelei de socializare, a fost adoptată în acest an, începând cu 30 aprilie 2015. Decizia de a opri suportul pentru schimbul de mesaje private a fost luată datorită numărului foarte mare de aplicaţii robot care nu făceau decât să deranjeze prin trimiterea de mesaje. Astfel a trebuit să găsesc o modalitate de a evita folosirea SDK-ului de la Facebook.

Soluţia găsită se bazează pe gestionarea notificărilor din sistemul de operare Android. Toate notificările primite pe un anumit sistem, pot fi interceptate de o aplicaţie Android.

Acest lucru este posibil, doar dacă utilizatorul acordă privilegii de acces la notificări pentru aplicaţie. Pe lângă acest fapt, pentru o funcţionarea corectă este necesară prezenţa aplicaţiei native de Facebook Messenger pentru că iniţierea de conversaţii nu se poate face fără aceasta.

În momentul în care un utilizator doreşte să poarte o discuţie privată şi criptată cu un alt utilizator, acesta trebuie să iniţieze conversaţia din aplicaţia nativă de Facebook Messenger. Astfel, utilizatorul de la celălalt capăt al canalului de comunicaţie va primi o notificare de la "com.facebook.orca" care va conţine mesajul pe care trebuia sa îl primească. Având această notificare se poate răspunde la mesaj pe baza datelor primite prin intermediul notificării. Răspunsul va fi primit tot sub formă de notificare, deci se poate trimite înapoi un mesaj partenerului de comunicaţie tot pe baza notificărilor. Mesajul de iniţiere va fi întotdeauna necriptat şi va fi un mesaj de tip atenţionare, a faptului că cineva doreşte să poarte o conversaţie criptată.

Notificările din sistemul de operare sunt interceptate prin intermediul unui serviciu, NotificationListenerService. Acesta implementează o metodă de a extrage notificări dintr-un obiect de tip StatusBarNotification. Din acest obiect se poate obţine numele pachetului care a produs notificarea. În acest proiect ne interesează doar pachetul "com.facebook.orca". Acesta este folosit de Facebook la schimbul de mesaje private.

Astfel, având notificarea prezentă în bara de stare a sistemului, putem crea un obiect de tip WearableExtender. Acesta este folosit pentru a extrage din notificare informaţiile suplimentare, cum ar fi, acţiuni sau pagini existente în notificăre. Notificarea preluată din bara de acţiuni conţine un câmp extra în care regăsim informaţii precum mesajul text primit, numele transmiţătorului sau identificatorul notificării. Tot de aici putem extrage şi eticheta notificării sau conţinutul intenţiei care va trebui să gestioneze datele recepţionate.

În momentul în care o notificare apare în bara de acţiuni, aceasta se extrage şi se transmite mai departe cu ajutorul unui obiect EventBus. Acesta transmite informaţiile preluate, activităţii care gestionează acest tip de evenimente. În acest caz, activitatea se numeşte ChatActivity.

În ChatActivity se implementează metoda onEvent, care va executa codul doar atunci când o notificare a sosit. Aici se verifică numele pachetului pe care îl foloseşte notificarea. Acesta trebuie să fie "com.facebook.orca" pentru ca ne interesează doar datele primite de la Facebook Messenger. Deoarece, metoda onEvent rulează în fundal, pentru a actualiza interfaţa grafică trebuie să folosim metoda runOnUiThread pentru codul de actualizare. Mesajul primit se găseşte în câmpul "android.text", iar numele transmiţătorului în câmpul "android.title".

Pentru a răspunde unui mesaj, este nevoie de un obiect RemoteInput, ce ajută la specificarea datelor care vor fi preluate de la aplicaţie şi pasate unei intenţii. Deoarece, în cazul acestui proiect nu se crează notificări, ci doar se răspunde la ele pe baza datelor primite, nu trebuie să construim noi obiecte de tip RemoteInput, pentru că acestea vin odată cu notificările. Trebuie doar modificate datele transmise inteţiei.

Astfel, în câmpul extra al notificării se adaugă mesajul de răspuns. Când răspunsul este gata de transmis, se apelează metoda addResultsToIntent care va primi ca argumente, obiectul RemoteInput modificat, o nouă inteţie care va gestiona transmisia şi câmpul

extra care va conţine mesajul. După toate acestea, se poate trimite mesajul de răspuns, folosind metoda send a intenţiei care gestionează această notificare.

Acest serviciu, este pornit, doar dacă aplicaţia este în activitatea ChatActivity. Deci notificările sunt gestionate doar dacă ambii parteneri de conversaţie au activat serviciul. În metoda onResume a activităţii trebuie verificat dacă serviciul este activ sau nu. În cazul în care serviciul nu este activ, se încearcă activarea lui. În cazul în care activitatea este oprită, o să se opreasą şi serviciul. Acest lucru se face în metoda onDestroy a activităţii.

## 5.2 Integrarea criptării transmisiei de date

Criptarea mesajelor transmise foloseşte algoritmul AES, iar cheile sunt simetrice, adică şi criptarea şi decriptarea este realizată pe baza aceluiaşi şir de caractere. Pentru a obţine cheia secretă se utilizează două şiruri de caractere. Unul ce reprezintă cheia în clar şi unul care este folosit la iniţializarea unui obiect de tip Cipher.

Bibliotecile Java pun la dispoziţie metode de a implementa partea de criptare a datelor, uşurând foarte mult munca depusă pentru acest proces. Astfel, se poate folosi un obiect de tip Cipher pentru obţinerea unui cifru dintr-un şir de octeţi. Acesta nu poate fi instaţiat direct, ci printr-o metodă numită getInstance. Metoda primeşte ca argument un şir de caractere de forma "algoritm/mod/padding", în cazul acestei aplicaţii se foloseşte "AES/CBC/PKCS5Padding".

Algoritmul folosit este AES, care reprezintă un standard ce poate procesa blocuri de date de 128 de biţi, folosind chei simetrice de tip cifru de lungime 128, 192 sau 256 de biţi. Fiecare intrare şi ieşire a acestui algoritm reprezintă o secvenţă de 128 de biţi de 1 sau 0. [9]. Această aplicaţie utilizează o cheie simetrică de 128 de biţi.

Modul CBC, reprezintă înlănţuirea blocurilor de cifru. Pentru acest mod este necesar un vector de iniţializare. În criptarea care foloseşte CBC, primul bloc este format prin aplicarea funcţiei logice SAU-exclusiv între primul bloc de text clar şi vectorul de iniţializare. Primul bloc de text criptat este obţinut prin aplicarea algoritmului de criptare pe blocul rezultat din procesul anterior. Următorul bloc este obţinut prin aplicarea funcţiei SAU-exclusiv între primul bloc criptat şi următorul bloc de text în clar şi aşa mai departe [8]. Procesul de obţinere a blocurilor este prezentat în figura 5.2.

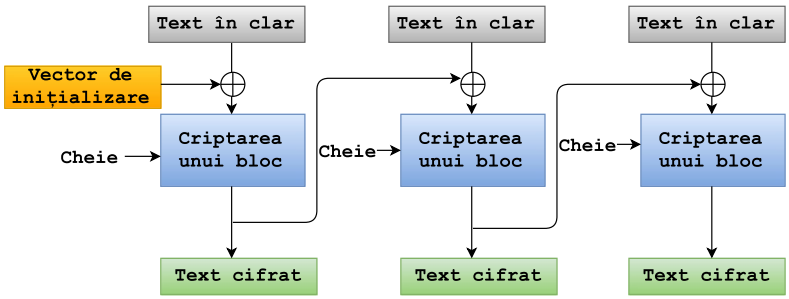


Figura 5.2: Criptarea folosind CBC

Decriptarea în modul CBC se face aplicând funcţia de decriptare a obiectului Cipher pe primul bloc de text criptat. Între blocul rezultat şi vectorul de iniţializare se aplică funcţa logică SAU-exclusiv, rezultând primul bloc de text clar. Următorul bloc este obţinut prin aplicarea funcţiei de decriptare pe al doilea bloc, iar între rezultat şi primul bloc criptat se aplică tot funcţia locică SAU-exclusiv [8]. Procesul de obţinere a blocurilor de text în clar este reprezentat în figura 5.3.

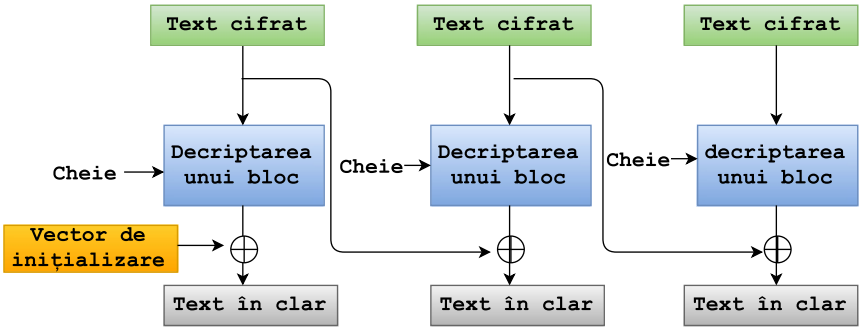


Figura 5.3: Decriptarea folosind CBC

PKCS5Padding, reprezintă o schemă de formatare a unui text, astfel încât lungimea acestuia să fie multiplu de opt. Trebuie să fie multiplu de opt, deoarece un bloc de text are lungimea de opt octeţi. Este necesară această metodă de completare a textului până dimensiunea lui este multiplu de opt, deoarece se foloseşte modul CBC, care se bazează pe criptarea înlănţuită a blocurilor de text.

### 5.2.1 Criptarea

Pentru criptare, se crează un obiect de tip Cipher prin metoda getInstance care primeşte ca argument şirul de caractere "AES/CBC/PKCS5Padding", după cum se poate observa în secţiunea de cod 5.1. Se instanţiază un obiect SecretKeySpec din cheia de criptare. Şirurile de caractere nu pot fi procesate direct, ele trebuie convertite la un vector de octeţi, acest lucru făcându-se cu ajutorul metodei getBytes, căruia se specifică setul de caractere, în acest caz "UTF-8".

```
1 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
2 SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes
    ("UTF-8"), "AES");
3 cipher.init(Cipher.ENCRYPT_MODE, key, new IvParameterSpec(IV
    .getBytes("UTF-8")));
4 byte[] encryptedBytes = cipher.doFinal(plainText.getBytes("
    UTF-8"));
5 String result = android.util.Base64.encodeToString(
    encryptedBytes, android.util.Base64.DEFAULT);
```

Listing 5.1: Cod folosit pentru criptare

Cifrul trebuie iniţializat, acest lucru realizându-se prin metoda init. Aceasta primeşte ca argumente, modul de iniţializare, aici ENCRYPT_MODE, obiectul SecretKeySpec

şi un obiect de tipul IvParameterSpec ce reprezintă vectorul de iniţializare. Este nevoie de acest vector pentru că se foloseşte modul CBC, care necesită un astfel de obiect pentru a procesa primul bloc din text. În această aplicaţie vectorul de iniţializare are valoarea "AAAAAAAAAAAAAAAA", dar poate avea oricare altă valoare.

Rezultatul criptării este un vector de octeţi, returnat de metoda doFinal a cifrului. Această metoda primeşte ca argument textul care urmează a fi criptat, dar sub formă de vector de octeţi. Pentru a obţine un format de afişare a vectorului de octeţi se foloseşte codificare Base64.

Codificarea Base64 este utilizată deoarece cereririle care folosesc protocolul HTTP permit trimiterea de date sub formă de şir de caractere, nu sub formă de vectori de octeţi.

### 5.2.2 Decriptarea

Pentru ca decriptarea să funcţioneze, trebuie să se primească un şir de caractere care reprezintă criptarea unui text clar, dar folosind aceeaşi cheie de criptare, acelaşi algoritm şi aceeaş metodă de codificare, adică Base64. Codul folosit pentru decriptare este prezentat în secţiunea de cod 5.2.

```
1 byte[] cipherText = android.util.Base64.decode(text.getBytes
    ("UTF-8"), android.util.Base64.DEFAULT);
2 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
3 SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes
    ("UTF-8"), "AES");
4 cipher.init(Cipher.DECRYPT_MODE, key,new IvParameterSpec(IV.
    getBytes("UTF-8")));
5 String result = new String(cipher.doFinal(cipherText));
```

Listing 5.2: Cod folosit pentru decriptare

Aplicaţia primeşte un şir de caractere ce trebuie decodificat şi convertit la un vector de octeţi prin metoda Base64, folosind setul de caractere UTF-8. După această decodificare se crează obiectul de tip Cipher tot prin getInstance("AES/CBC/PKCS5Padding"). Şi aici este nevoie de SecretKeySpec, format pe baza aceleiaşi chei, ca la criptare. Se iniţializează cifrul, la fel ca la partea de criptare, doar ca aici metoda specificată este DECRYPT_MODE.

Rezultatul este întors tot sub formă de vector de octeţi, deci este necesară conversia la şir de caractere, pentru a putea fi afişat în interfaţa grafică.

# Capitolul 6

# Testarea şi evaluarea soluţiei

## 6.1 Testarea funcţionalităţilor

### 6.1.1 Login

### 6.1.2 Trimitere - Recepţie mesaje

### 6.1.3 Criptare - Decriptare

## 6.2 Evaluarea rezultatelor

# Capitolul 7

# Concluzii

## 7.1 Activităţi viitoare

# Anexa A

# Project Build System Makefiles

## A.1 Makefile.test

```
# Makefile containing targets specific to testing

TEST_CASE_SPEC_FILE=full_test_spec.odt
API_COVERAGE_FILE=api_coverage.csv
REQUIREMENTS_COVERAGE_FILE=requirements_coverage.csv
TEST_REPORT_FILE=test_report.odt


# Test Case Specification targets

.PHONY: full_spec
full_spec: $(TEST_CASE_SPEC_FILE)
        @echo
        @echo "Generated␣full␣Test␣Case␣Specification␣into␣
           \"$^\""
        @echo "Please␣remove␣manually␣the␣generated␣file."

.PHONY: $(TEST_CASE_SPEC_FILE)
$(TEST_CASE_SPEC_FILE):
        $(TEST_ROOT)/common/tools/generate_all_spec.py --
           format=odt -o $@ $(TEST_ROOT)/functional-tests $(
           TEST_ROOT)/performance-tests $(TEST_ROOT)/
           robustness-tests
#

# ...
```

Listing A.1: Testing Targets Makefile (Makefile.test)

# Bibliografie


[1] http://www.xabber.com.

[2] http://beem-project.com/projects/beem/wiki.

[3] http://www.thialfihar.org/projects/apg.

[4] https://play.google.com/store/apps/details?id=com.woodkick.pgpsms&hl=en.

[5] https://play.google.com/store/apps/details?id=com.symantec.pgpviewersymantec&hl=en.

[6] http://www.androidcentral.com/best-android-phones.

[7] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, pages 77–84, New York, NY, USA, 2004. ACM.

[8] Morris J. Dworkin. Sp 800-38a 2001 edition. recommendation for block cipher modes of operation: Methods and techniques. Technical report, Gaithersburg, MD, United States, 2001.

[9] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Advanced Encryption Standard*. Alpha Press, 2009.