Aplikacija za evidentiranje prisustva

MSc završni rad

Autor: Malik Koljenović, BSc IT

Odsjek za računarstvo i informatiku

Elektrotehnički fakultet

Mentor: prof. dr. Saša Mrdović

Sarajevo, septembar 2018

Abstract This thesis addresses the problem of large scale electronic attendance taking in university setting by presenting an Android based attendance taking application, based on immutable and non repudiable location proofs backed by RSA cryptography, utilizing NFC and HCE for ease of use, emulating NFC Forum Tag Type 4 it is also compatible with existing reader infrastructures. It also presents a general overview of the utilized techologies and select implementation details.

Apstrakt Ova teza tretira problem masovnog elektronskog bilježenja prisustva u univerzitetskom okruženju izradom prijedloga aplikacija bazirane na Android platformi korištenjem neizmjenjivih i neporecivih vremensko-lokacijskih dokaza osiguranih korištenjem RSA kriptografije, te NFC i HCE tehnologija u cilju jednostavnosti upotrebe; emulirajući NFC Forum Tag Tip 4 kompatibilna je sa postojećim infrastrukturama čitača. Dat je i opšti pregled korištenih tehnologija i izdvojenih implementacijskih detalja.

MSC Primary 68P25; Secondary 94A60;

Keywords: NFC - near-field communication, HCE - host card emulation, security, Android, attendance, RSA, cryptography, NDEF, NTAG, geolocation, location proofs

Sadržaj

Popis slika **4**

Popis tablica **5**

Pojmovnik **6**

1 Uvod **8**

2 Postavka problema **10**

3 Prijedlog rješenja **11**

3.1 Logički model rješenja 11

3.2 Tehnički model rješenja 15

4 Pregled korištenih tehnologija **19**

4.1 NFC - Near-field communication 19

4.2 NDEF - NFC Data Exchange Format 19

4.3 HCE - Host card emulation 19

4.4 Kriptografske tehnologije 19

4.5 Android 19

4.5.1 GPS Geolokacija 19

4.5.2 Retrofit HTTP Client 19

4.5.3 GSON JSON Serializer 19

4.6 Python 19

5 Izdvojeni detalji implementacije **20**
5.1 Podatkovni i kriptografski primitivi 20
5.1.1 SPIM paket 20
5.1.2 SESS paket 20
5.2 NFC komunikacijski protokol 20
5.3 LAPI komunikacijski protokol 20

6 Zaključak **21**

Dodatak **22**

A Korisničko uputstvo **23**
A.1 Instruktorski način rada 25

B Izvorni kod **26**
B.1 LAPI izvorni kod 26
B.1.1 `__init__.py` 26
B.2 Android izvorni kod 29
B.2.1 `AndroidManifest.xml` 30
B.2.2 `model/Attendance.java` 31
B.2.3 `model/Place.java` 36
B.2.4 `model/Session.java` 38
B.2.5 `model/User.java` 39
B.2.6 `api/LogitService.java` 40
B.2.7 `AttendanceActivity.java` 41
B.2.8 `AttendanceAdapter.java` 55
B.2.9 `LogitApduService.java` 56
B.2.10 `LogitApplication.java` 65
B.2.11 `MainActivity.java` 66

Bibliografija **71**

Popis slika

3.1 Dijagram interakcije - uspješna registracija korisnika i generisanje ključeva 12
3.2 Dijagram interakcije - bilježenje prisustva studenata (Master BUMP) . . 13
3.3 Dijagram interakcije - prijava prisustva studenta (Slave BUMP) 14
3.4 Dijagram interakcije - pohranjivanje potpisa na LAPI (SYNC) 15
3.5 Logit UI Android prikaz korisničkog interfejsa 18

A.1 zaglavlje aplikacije prikazuje aktivnog korisnika 25
A.2 glavni izbornik, opisi funkcionalnosti u nastavku 25
A.3 trenutno zabilježena lokacija korisničkog uređaja 25
A.4 ordinalno numerisan spisak prisutnih studenata 25

Popis tablica

Pojmovnik

ATTN repozitorij potpisanih prisustva spremljen na LAPI.

BUMP približavanje mobilnih uređaja, otvara jednosmjerni komunikacijski kanal u smijeru od slave (S) prema master (M) uređaju.

CERT javni dio korisničkog kriptografskog ključa.

DEVICE korisnički Android uređaj.

HCE *(en. Host card emulation)* softverska arhitektura koja omogućava virtualnu emulaciju elektronskog identiteta.

ISO/IEC 14443 Tip A standard fizičkog sloja NFC komunikacijskog protokola.

KEYS jedinstveni set korisničkih RSA ključeva dužine 2048 bita.

LAPI Logit API, Python serverska aplikacija, komponenta LAPP platforme.

LAPP Logit višekomponentna aplikacijska platforma.

M *(en. master)* - Android UI komponenta pokrenuta na uređaju koji bilježi prisustvo.

NDEF vrsta standardizovanog paketa korištena za NFC komunikaciju između uređaja.

NDEFMSG NDEF poruka koja sadrži vremensko-lokacijski dokaz potpisan od strane korisnika.

NFC Forum Tag standardizovani format NFC taga.

S *(en. slave)* - Android komponenta koja se izvršava u pozadini na uređaju čije se prisustvo bilježi.

SPIM *(en. SPacetIME)* - lokacijski dokaz (JSON objekat, struktura podatka).

SSO *(en. Single Sign On)* - politika autentifikacije korištenjem jedinstvenog repozitorija.

UI Android komponente LAPP platforme.

1 Uvod

Prodor digitalnih računara i komunikacijskih tehnologija u sve sfere ljudskog života i djelovanja, te dramatično povećanje broja korisnika interneta u posljednjoj deceniji nametnulo je mnoštvo novih društvenih i tehničkih izazova. Društveni izazovi najbolje su uočljivi kroz višedecenijsku debatu o privatnosti i vlasništvu nad ličnim podacima, samim time zadiru duboko u diskusiju o ljudskim pravima i identitetu sa jedne i često suprostavljenim komercijalnim interesima sa druge strane. Ukoliko se u tom kontekstu posmatra aktuelna EU uredba o zaštiti podataka[1] (*en. GDPR*) postaje jasno da su digitalna tehnologija i komunikacije postale integralni dio društvene i emocionalno-psihološke realnosti[2], do te mjere da se digitalni tragovi smatraju dijelom nepovredivog identiteta osobe. Iz navedenog je jasno da se radi o institucionalizaciji jedne potpuno nove društveno-tehnološke paradigme unutar pravnih okvira Europske unije.

Sa tehničke strane, dostignuća na poljima kriptografije, teorije mreža i novih komunikacijskih tehnologija, te njihova široka prihvaćenost otvorila su mogućnosti izrade računarskih sistema spremnih da odgovore na novonastale društvene izazove u okviru opisane nove paradigme. Pomenuti računarski sistemi kao dodatno izvršno okruženje imaju društveno-pravnu realnost te se u tim okvirima izvršavaju masovno, dobrovoljno, distribuirano i interaktivno[3] van centralizovanog računarskog izvršnog okruženja u smislu Von Neumannove arhitekture. Opisani sistemi mogu se okarakterisati kao sistemi potpomaganja (*en. assist*), npr. kriptografski računarski sistem u domenu autentifikacije i autorizacija u novoj paradigmi postmatra se kao sistem računarski-potpomognutog povjerenja, ekvivalentno višem nivou apstrakcije.

Registri u kontekstu društvenih institucija su elementarni mehanizam sistema povjerenja, sigurnosne karakteristike takvih institucionalnih registara stoga čine osnov istraživačkog interesa u domenu institucionalne sigurnosi. Napredni elektronski registri izrađeni korištenjem kriptografskih tehnika i savremenih komunikacijskih protokola za prikupljanje i obradu podataka omogućavaju poboljšanje njihovih sigurnosnih osobina, otvarajući nove načine primjene i stvarajući uslove za viši nivo društvenog razvoja i institucionalne efikasnosti, uz to pružaju i adekvatan odgovor na novonastale društvene izazove. Stoga, ukoliko se obezbijede i ispoštuju preduslovi izrade sigurnog sistema[4], evidenciju prisustva u kontekstu naprednog elektronskog registara treba posmatrati i kao vremensko-prostorni dokaz određenog događaja, ovaj rad usmjeren je na izradu jednog takvog sistema računarski-potpomognutog povjerenja u obliku institucionalnog registra elektonske evidencije prisustva.

2 Postavka problema

Projektni zadatak ovog završnog rada je izrada aplikacije na Android platformi sa pripadajućom udaljenom komponentom, koje u cjelini treba da omoguće evidentiranje prisustva nastavnim aktivnostima na Elektrotehničkom fakultetu u Sarajevu. U skladu sa zadatim funkcionalnim zahtjevima, a iz razloga olakšanog korištenja i praktičnosti upotrebe neophodno je iskoristiti beskontaktne komunikacijske mogućnosti savremenih mobilnih telefona u vidu NFC komunikacijskog protokola.

Također neophodno je osigurati korisnike aplikacije od mogućih zloupotreba korištenjem dostupnih kriptografskih metoda i tehnologija, te stvoriti neophodne uslove za sticanje povjerenja u širi sistem bilježenja prisustva putem neporecivosti i neizmjenjivosti prethodno unesenih podataka. Poželjna mogućnost je jednostavna integracija sa postojećim sistemima, prvenstveno onim autentifikacijskim i autorizacijskim, te planiranje arhitekture za buduća proširenja u vidu omogućavanja integracije sa infrastrukturnim hardverskim čitačima i TAG karticama.

Potrebno je dokumentovati proces izrade i opisati korištene tehnologije, sa posebnim osvrtom na korištene kriptografske metode i tehnologije, te identifikovati otvorena pitanja na polju elektronskih registara prisustva, mogućnosti i izazove koje oni predstavljaju uz rješenja koja navedena aplikacija nudi u datom kontekstu.

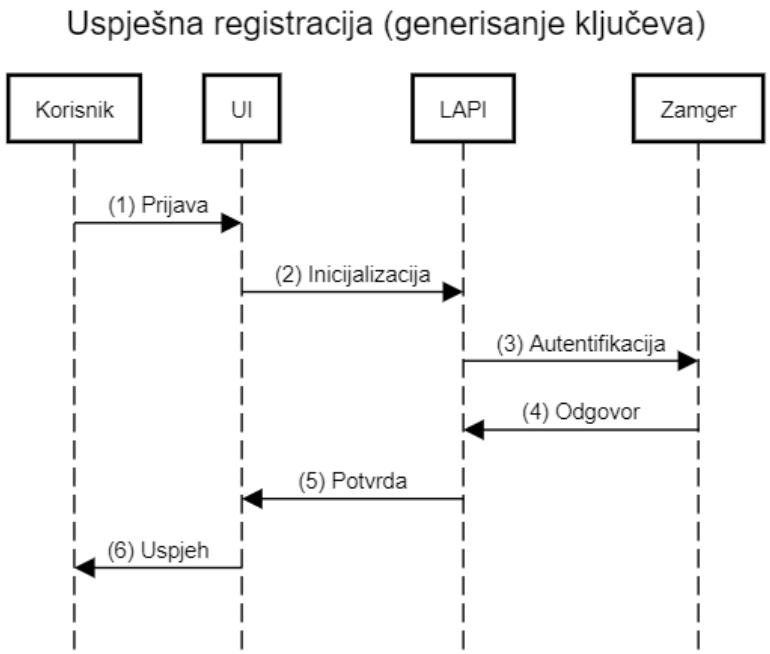
3 Prijedlog rješenja

U skladu sa datim zahtjevima predložena je izrada aplikacijske platforme pod nazivom Logit (LAPP), opisane u nastavku, sa detaljnim tehničkim detaljima u narednim poglavljima. Uzimajući u obzir data ograničenja, te funkcionalne i nefunkcionalne zahtjeve određeno je da se korisnička aplikacija izradi na Android platformi sa podrškom za Android API nivo počevši od nivoa 19 (4.4 KitKat), to je najniži nivo koji omogućava korištenja naprednih NFC i kriptografskih funkcionalnosti te osigurava dobru pokrivenost potencijalne korisničke baze sa ukupnom adopcijom od preko 90% za navedenu ili višu verziju[5]. Za uspješan rad aplikacije neophodno je da korisnički uređaj podržava i NFC funkcionalnosti, prema prognozama analitičke kuće IHS Technology, do 2020. godine svaki treći uređaj imati će podršku za NFC.[6]

3.1 Logički model rješenja

Priloženi dijagrami interakcije osnovnih funkcionalnosti Logit platforme i pripadajući opis imaju za cilj stvoriti opštu sliku sistema, te tako olakšati praćenje tehničkog modela rješenja datog u nastavku. Tehnički model opisuje dosta detaljniju sliku funkcioniranja sistema i može služiti kao svojevrstan uvod u kod platforme.

Registracija korisnika



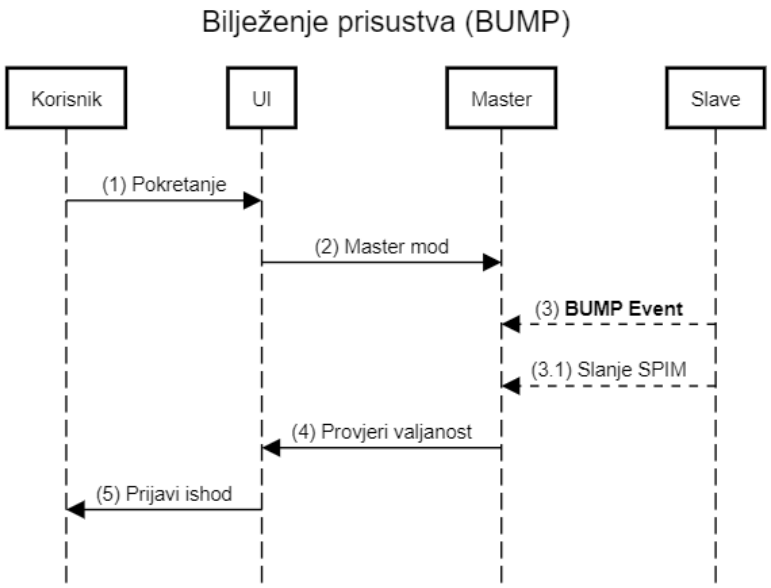
Slika 3.1: Dijagram interakcije - uspješna registracija korisnika i generisanje ključeva

Nakon uspješne instalacije aplikacije na korisnički Android uređaj (DEVICE) potrebno je obaviti proces registracije koji se izvršava u dva bitna koraka. Prvi korak sastoji se od unosa već postojećih autentifikacijskih podataka za ZAMGER sistem Elektrotehničkog fakulteta, korisnik se korištenjem datih podataka posredstvom LAPI servisa autentificira na ZAMGER sistemu, bitno je napomenuti da Logit platforma ne sprema korisničku lozinku ZAMGER sistema, navedeni podaci se koriste isključivo za povezivanje postojećeg identiteta i novogenerisanog para korisničkih RSA ključeva (KEYS), što je ujedno i drugi korak u procesu registracije na Logit platformu.

U slučaju uspješnje autentifikacije, korisnika se obavještava o završenoj registraciji te se preusmjerava na glavni ekran za bilježenje prisustva. Generisani javni ključ (CERT) i identifikacioni podaci korisnika spremaju se u LAPI direktorij korisničkih certifikata.

Bilježenje prisustva studenta

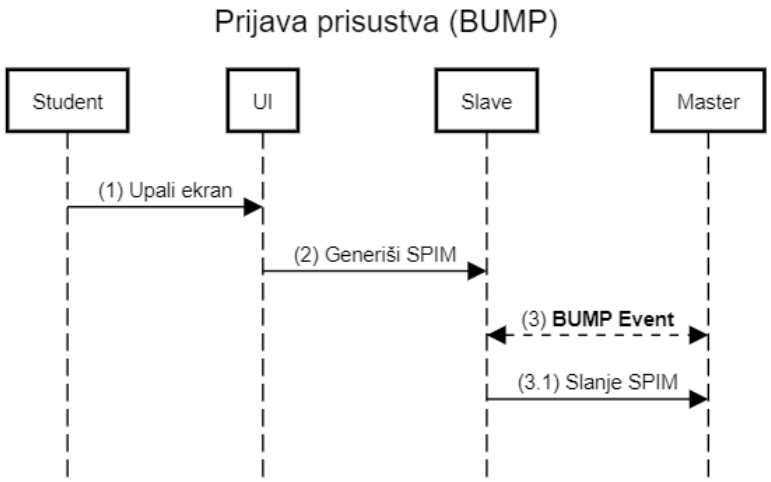
Bilježenje prisustva studenata od strane predmetnog nastavnika izvodi se u Master (M) modu funkcionisanja aplikacije, aplikacija se pri samom pokretanju i nakon uspješno obavljene registracija automatski stavlja u takav mod operacije i u njemu ostaje sve dok je upaljen ekran korisničkog uređaja (DEVICE) i Logit aplikacija (UI) se izvršava u prednjem planu *(en. foreground)*, navedene zahtjeve diktira sama Android platforma.



Slika 3.2: Dijagram interakcije - bilježenje prisustva studenata (Master BUMP)

Ukoliko su ispunjeni prethodno pobrojani zahtjevi, dovoljno je da student sa podešenom Logit aplikacijom na svom uređaju prinese slave (S) uređaj master (M) uređaju i da njegovo prisustvo bude zabilježeno i prikazano na ekranu M uređaja. Samu interakciju (BUMP) inicira studentski S uređaj. Prilikom ovog BUMP događaja dolazi do razmjene kriptografski potpisanih podataka o vremenu i lokaciji (SPIM) sa S na M, gdje M vrši validaciju primljenih podataka poredeći studentsko vrijeme i lokaciju sa vremenom i lokacijom na M uređaju, gdje se prisustvo odbija ukoliko se ustanovi pokušaj lažiranja podataka.

Prijava prisustva

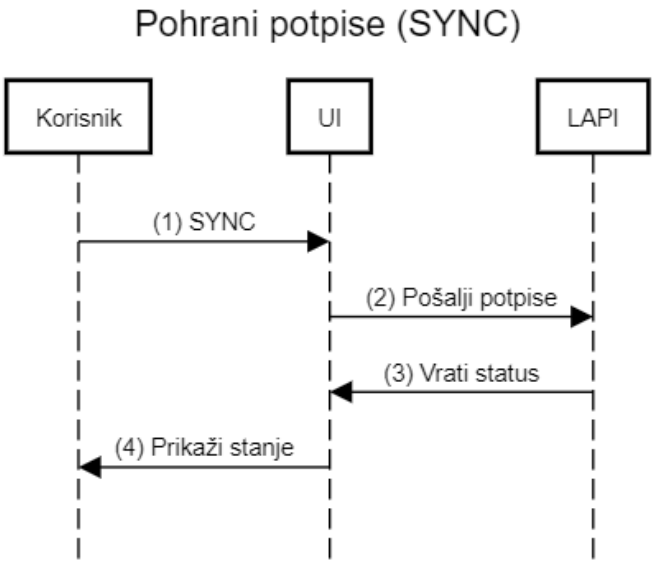


Slika 3.3: Dijagram interakcije - prijava prisustva studenta (Slave BUMP)

Studentski S uređaj i M uređaj nastavnog osoblja podešavaju se na isti način opisan iznad, jedina praktična razlika javlja se prilikom korištenja, gdje je za S uređaj čije se prisustvo bilježi dovoljno upaliti ekran uređaja da bi se mogla ostvariti BUMP interakcija prislanjanjem S na M. Ovo je moguće jer se NFC HCE emulator Logit aplikacije izvršava u pozadini Android sistema.

Pohranjivanje potpisa sesije na LAPI

Svako bilježenje prisustva unutar Logit Android UI odvija se unutar sesije (SESS) koja se automatski započinje prilikom prvog uspješno zabilježenog prisustva i traje sve dok korisnik ne izvrši pohranu navedene sesije na LAPI servis. Klikom na SYNC dugme prikupljeni podaci šalju se LAPI servisu, provjeravaju se jedinstveni potpisi studenata te potpis ukupne sesije od strane M uređaja, ukoliko se ne pronađu nepravilnosti navedeni podaci se pohranjuju u LAPI repozitorij potpisa, takvi podaci kriptografski su osigurani od naknadne izmjene.



Slika 3.4: Dijagram interakcije - pohranjivanje potpisa na LAPI (SYNC)

Potpisi pohranjeni u LAPI repozitoriju mogu dalje biti korišteni u integrisanim aplikacijskim rješenjima koja zahtijevaju ovakvu vrstu podataka pomoću ponuđenog LAPI REST interfejsa, te se mogu smatrati relevantnim i sigurnim dokazom prisustva.

3.2 Tehnički model rješenja

Uvodi se dodatno pojam lokacijskog dokaza[7] koji u širem smislu u kontekstu podređenog korisnika (en. slave), obuhvata kriptografski potpisan korisnički identitet, korisnički uređaj, vrijeme i GPS lokacijske podatke korisničkog uređaja. Za svrhu osiguranja jedinstvenosti identiteta i vjerodostojnosti potvrde lokacijskih dokaza odabrano je korištenje RSA asimetrične enkripcije, gdje se pri uspješnoj autentifikaciji generiše jedinstveni set ključeva za korisnički uređaj, privatnom dijelu ključa nije moguće pristupiti izvan aplikacije (SEC1), niti je moguće eksportovati ključ (SEC2), a u određenom vremenskom period može postojati samo jedan valjan set ključeva za jednog korisnika jer se raniji ključevi ne uzimaju u obzir ukoliko postoji noviji set (SEC3), sprječavajući tako replikaciju identiteta na više uređaja.

Pored Android komponente aplikacije (UI) izrađena je i serverska aplikacija u programskom jeziku Python (LAPI), čija je namjena posredovanje u komunikaciji sa autentifikacijskim agentom (ZAMGER), te pohranjivanja i održavanje javnih korisničkih kriptografskih ključeva (CERT) i njihovo povezivanje sa autentifikacijskim podacima korisnika, pored toga služi i kao repozitorij za potpisana prisustva (ATTN). Na ovu komponentu se može gledati kao na integrisani namjenski repozitorij korisničkih certifikata i domenski repozitorij neporecivih i neizmjenjivih lokacijskih dokaza (SPIM).

Budući da na Elektrotehničkom fakultetu u Sarajevu postoji SSO (en. Single-Sign On) politika autentifikacije, u serverskoj komponenti (LAPI) je implementiran autentifikacijski posrednik koji prilikom prvog pokretanja aplikacije prijavljuje korisnika koristeći postojeće pristupne podatke, tom prilikom u slučaju uspješne prijave generiše se i jedinstveni set RSA ključeva dužine 2048 bita (KEYS), koji se pohranjuju na korisničkom uređaju (DEVICE), a javni dio, tj. certifikat (CERT) se pohranjuje i u repozitorij ključeva (LAPI) sa poveznicom na korisnički identitet, kasnije se ti certifikati koriste za provjeru valjanosti potpisa lokacijskih dokaza (SPIM).

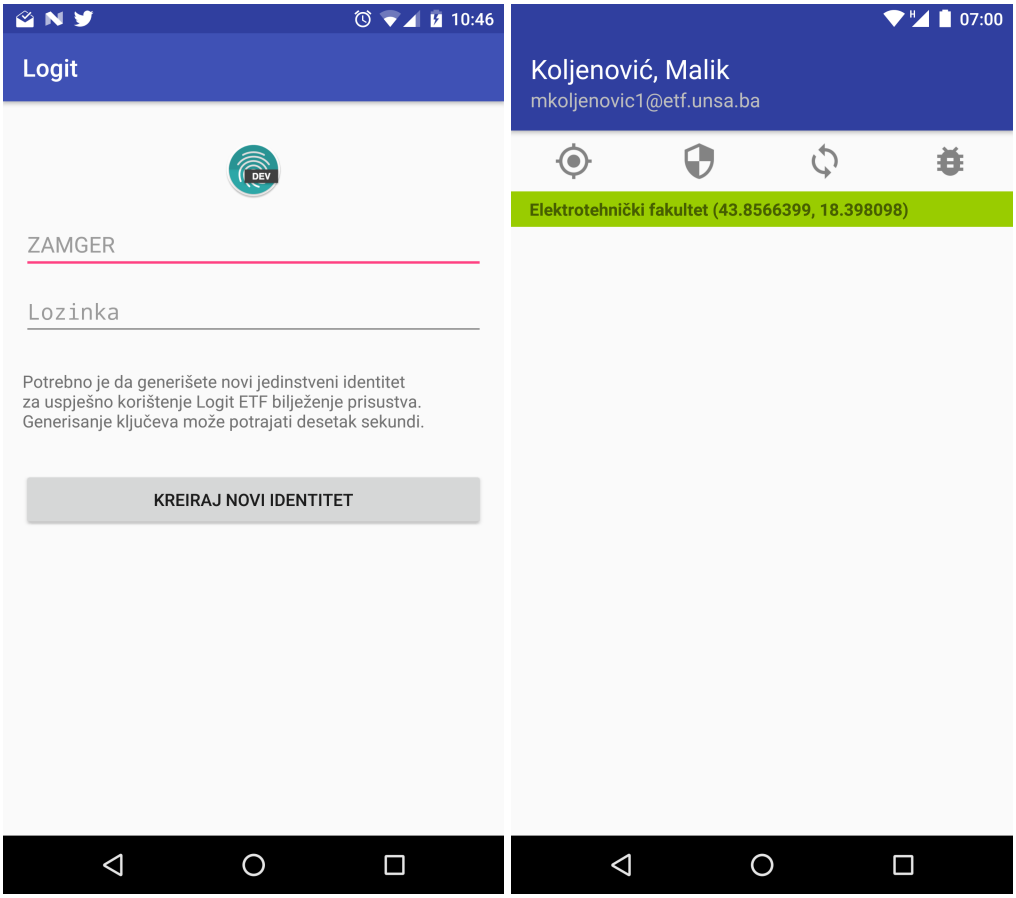
Da bi se osigurala jednostavnost korištenja aplikacije odabrana je implementacija HCE emulacijskog načina rada NFC komunikatora koji omogućava korisniku da izvrši komunikaciju sa drugim uređajem bez potrebe da pokreće aplikacijski prozor na svom uređaju, dovoljno je da upali ekran svoj uređaja i prinese ga master (M) uređaju koji prikuplja potpise, u ovom slučaju drugoj instanci Logit aplikacije na kojoj je pokrenuta aktivnost za prikupljanje potpisa (LAPP).

Približavanjem mobilnih uređaja (BUMP) otvara se jednosmjerni komunikacijski kanal u smijeru od slave (S) prema master (M) uređaju korištenjem ISO/IEC 14443 Tip A komunikacijskog protokola pri čemu se emulira NFC Forum Tag tipa 4 i putem NDEF Aplikacije prenosi jedna NDEF poruka (NDEFMSG) koja sadrži vremensko-lokacijski dokaz potpisan od strane korisnika, nadalje u tekstu označen kao SPIM (en. spime)[8].

Po primitku poruke nadređeni uređaj (en. master) koji osluškuje da mu se pridruže podređeni uređaji (en. slave) i ima pokrenutu Logit aplikaciju, tu poruku sprema u lokalni repozitorij potpisa ukoliko ona zadovolja uslove da očitana slave GPS lokacija nije udaljena više od 50 metara od očitane master GPS lokacije (VK1 - validacijski kriterij #1), te da podešena razlika satova master i slave uređaja nije veća od 300 sekundi (VK2), bez da nad SPIM objektom vrši ikakve izmjene, ukoliko SPIM objekat ne zadovoljava date validacijske kriterije odbija se i ispisuje se odgovarajuća poruka na master ekranu. Moguće je naknadno klikom na validacijsko dugme (ACTVAL) u korisničkom interfejsu izvršiti provjeru svih prikupljenih potpisa tokom jedne sesije (SESS), tom prilikom se, ukoliko postoji mrežna veza; svi potpisi pošalju Logit serveru (LAPI) na provjeru i vraća se stanje valjanosti potpisa za sve proslijeđene SPIM objekte.

Ukoliko master (M) želi da pohrani SPIM objekte iz jedne sesije (SESS) na Logit server (LAPI), klikom na sinhronizacijsko dugme u interfejsu (ACTSYNC), on vrši dodatno potpisivanje svakog SPIM objekta svojom komponentom privatnog ključa (MPRK), tako što potpiše hash (SHA256) vrijednost SPIM objekta (AID) sa dodatim svojim jedinstvenim master korisničkim imenom (MUSER) i jedinstvenim identifikatorom sesije (SID) i dodatno generiše SHA256 vrijednosti tih potvrda (CID), nakon čega objedinjuje sve CID vrijednosti i dodatno ih potpisuje svojim MPRK, sve te vrijednosti šalje Logit server (LAPI) na pohranjivanje, ovakvom procedurom se obezbjeđuje neporecivost i neizmjenjivost SPIM i SESS objekata, jer onemogućava izmjene pojedinačnih SPIM objekata, te brisanje ili dodavanje objekata u finaliziranoj sesiju (SESS) od strane malicioznih aktera bez da naruši integritet SHA256 vrijednosti.

Uzmimajući u obzir bitnost rješenja i visoku vjerovatnoću svakodnevne primjene kod ciljane korisničke grupe, te izazove koje takav slučaj korištenja predstvalja omogućena je i direktna e-mail komunikacija za prijavu grešaka ili slanje prijedloga sa glavnog korisničkog interfejsa (ACTBUG). Kako se radi o slojevitom i kompleksnom softverskom rješenju za više detalja referirati se na izvorni kod priložen u dodatku.



Slika 3.5: Logit UI Android prikaz korisničkog interfejsa

4 Pregled korištenih tehnologija

4.1 NFC - Near-field communication

4.2 NDEF - NFC Data Exchange Format

```
string title = "This is a Unicode in the sky"
/*
Defined as π = lim_{n→∞} P_n/d where P is the perimeter
of an n-sided regular polygon circumscribing a
circle of diameter d.
*/
const double pi = 3.1415926535
```

4.3 HCE - Host card emulation

4.4 Kriptografske tehnologije

4.5 Android

4.5.1 GPS Geolokacija

4.5.2 Retrofit HTTP Client

4.5.3 GSON JSON Serializer

4.6 Python

5 Izdvojeni detalji implementacije

5.1 Podatkovni i kriptografski primitivi

5.1.1 SPIM paket

5.1.2 SESS paket

5.2 NFC komunikacijski protokol

5.3 LAPI komunikacijski protokol

6 Zaključak

Dodaci

A Korisničko uputstvo

1. **Uspostavite internet konekciju** prema uputama vašeg instruktora.
   (a) potrebno je da na mreži bude dostupna Logit serverska aplikacija i certifikacijski repozitorij za uspješnu prijavu i korištenje, dostupnost možete provjeriti posjetom na https://logit.mine.nu:5000
2. **Uključite lokacijske usluge** vašeg Android mobilnog uređaja.
   (a) detaljno uputstvo možete pronaći na https://support.google.com/accounts/answer/3467281?hl=hr
3. **Omogućite NFC komunikaciju** na vašem Android mobilnom uređaju i **isključite Android Beam** uslugu za optimalan rad aplikacije.
   (a) `Settings > NFC > Enable`
   (b) `Settings > NFC > Android Beam > Disable`
   (c) više detalja za navedene postavke pročitajte na https://support.google.com/nexus/answer/2781895?hl=hr
4. Ukoliko niste, **omogućite sigurnosnu funkcionalnost zaključavanja vašeg ekrana**
   (a) Android OS nudi usluge integrisane sigurnosti mobilnih uređaja, te je Keystore funkcionalnost sigurnog pohranjivanja privatnih ključeva usko vezana za ostale sigurnostne postavke, stoga omogućite zaključavanje ekrana slijedeći uputstvo na https://support.google.com/nexus/answer/2819522?hl=hr

5. Prihvatite poziv za alpha testiranje posjetom na https://play.google.com/apps/testing/ba.unsa.etf.logit i nastavite na Play Store te **instalirajte aplikaciju**

   (a) ukoliko vaš mobilni uređaj nije izlistan kao podržan obratite se vašem instruktoru i biti će vam izdat jedinstveni NFC Certifikat, koji ćete koristiti za bilježenje prisustva

6. **Pokrenite Logit aplikaciju**

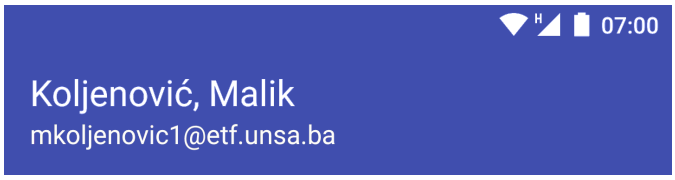
7. **Unesite vaše ZAMGER korisničke podatke**, ovi podatci koriste se jednokratno za provjeru valjanosti identiteta prije generisanja vašeg para ključeva, vaša lozinka ne ostaje pohranjena na Logit sistem i prenosi se https kanalom prema ZAMGER servisu

8. Aplikacija je spremna za korištenje i ne mora biti pokrenuta u prednjem planu za prijavu prisustva, **za optimalne rezultate** dovoljno je da upalite ekran vašeg Android uređaja na "lock screen" i prislonite na Android uređaj instruktora.

9. **Ukoliko želite koristiti aplikaciju u instruktorskom modu** i prikupljati prisustvo, dovoljno je da pokrenete Logit aplikaciju u prednjem planu te prislonite vaš uređaju studentskom uređaju u skladu sa korakom 8.

A.1 Instruktorski način rada

Pokretanjem glavnog prozora Logit aplikacije ulazite u mod za prikupljanje studentskih potpisa prisustva. Ovaj zaslon podijeljen je na četiri komponente, opisi kako slijedi u nastavku.



Slika A.1: zaglavlje aplikacije prikazuje aktivnog korisnika

Slika A.2: glavni izbornik, opisi funkcionalnosti u nastavku

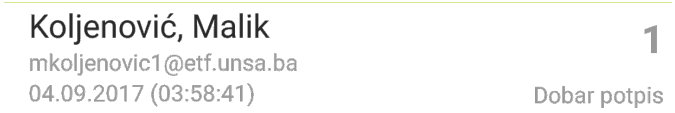
Dugme 1 služi za ručno osvježavanje trenutne lokacije

Dugme 2 koristite za provjeru valjanosti ključeva korištenih pri potpisu

Dugme 3 sinhronizacija trenutne sesije na Logit server, svi potpisi se pohranjuju u jednu sesijsku cjelinu i brišu sa mobilnog uređaja (kreira se nova sesija)

Dugme 4 otvara e-mail klijent po izboru korisnika u cilju lakše prijave grešaka

Slika A.3: trenutno zabilježena lokacija korisničkog uređaja



Slika A.4: ordinalno numerisan spisak prisutnih studenata

B Izvorni kod

B.1 LAPI izvorni kod

Repo: https://github.com/koljenovic/logit-node/

```
.
├── Logit
│   ├── Logit
│   │   ├── __init__.py
│   │   ├── logit.db
│   │   └── static
│   └── logit.wsgi
├── README.md
└── README.md~
```

B.1.1 __init__.py

```
from __future__ import print_function
from bs4 import BeautifulSoup
from flask import Flask, request
from asn1crypto.x509 import Certificate
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256
import os, sys, sqlite3, requests, json, binascii

app = Flask(__name__)
db_name = 'logit.db'
dbcon = None

def init_db(dbcon):
    dbcon.execute("CREATE TABLE Users (id INTEGER PRIMARY KEY autoincrement NOT NULL, uid TEXT, user TEXT, name TEXT, surname TEXT, cert TEXT, time TIMESTAMP DEFAULT current_timestamp NOT NULL)")
```

```
16      dbcon.execute("CREATE TABLE Attendances (id INTEGER PRIMARY KEY
    autoincrement NOT NULL, sid TEXT, mid TEXT, uid TEXT, lat TEXT, lon TEXT, ts
    TEXT, sig TEXT, aid TEXT, confsig TEXT, cid TEXT, time TIMESTAMP DEFAULT
    current_timestamp NOT NULL)")
17      dbcon.execute("CREATE TABLE Sessions (id INTEGER PRIMARY KEY autoincrement
    NOT NULL, sid TEXT, sig TEXT, master TEXT, time TIMESTAMP DEFAULT
    current_timestamp NOT NULL)")
18      dbcon.commit()
19
20  if not os.path.isfile(db_name):
21      dbcon = sqlite3.connect(db_name)
22      init_db(dbcon)
23  else:
24      dbcon = sqlite3.connect(db_name)
25
26  @app.route("/")
27  def main():
28      return 'Work'
29
30  @app.route("/auth/", methods=['POST'])
31  def auth():
32      user = request.form['user']
33      passw = request.form['pass']
34      cert = request.form['cert']
35      uid = request.form['uid']
36      s = requests.Session()
37      r = s.post('https://zamger.etf.unsa.ba/index.php', data={'loginforma':1,
    'login': user, 'pass': passw})
38      r = s.get('https://zamger.etf.unsa.ba/index.php?sta=common/profil')
39      soup = BeautifulSoup(r.text, 'html.parser')
40      nameTag = soup.find('input', attrs={"name": "ime"})
41      surnameTag = soup.find('input', attrs={"name": "prezime"})
42      name = nameTag['value'].encode('utf8')
43      surname = surnameTag['value'].encode('utf8')
44      dbcon.execute("INSERT INTO Users (uid, user, name, surname, cert) VALUES
    (?, ?, ?, ?, ?)", (uid, user, buffer(name), buffer(surname), cert))
45      dbcon.commit()
46      return json.dumps({ 'name': name, 'surname': surname})
47
48  @app.route("/validate/", methods=['POST'])
49  def validate():
50      data = request.data
51      attns = json.loads(data)
52      # print(attns, file=sys.stderr)
53      c = dbcon.cursor()
54
55      result = []
56
57      for attn_string in attns:
```

```
58          attn = json.loads(attn_string)
59          c.execute("SELECT max(id) id FROM Users WHERE user=? GROUP BY user",
    (attn['user'],))
60          certId = c.fetchone()
61          c.execute("SELECT * FROM Users WHERE id = ?", (certId[0],))
62          user = c.fetchone()
63          cert = Certificate.load(binascii.unhexlify(user[5]))
64          n = cert.public_key.native['public_key']['modulus']
65          e = cert.public_key.native['public_key']['public_exponent']
66          package = attn['user'] + ':' + attn['lat'] + ':' + attn['lon'] + ':' +
    attn['ts']
67          digest = SHA256.new()
68          digest.update(package)
69
70          public_key = RSA.construct((n, e))
71
72          verifier = PKCS1_v1_5.new(public_key)
73          verified = verifier.verify(digest, binascii.unhexlify(attn['sig']))
74
75          attn['valid'] = 1 if verified else -1
76          attn['raw'] = json.dumps(attn)
77          attn['name'] = binascii.unhexlify(attn['name'])
78          attn['surname'] = binascii.unhexlify(attn['surname'])
79          result.append(attn)
80          # print(verified, file=sys.stderr)
81
82      return json.dumps(result)
83
84  @app.route("/sync/", methods=['POST'])
85  def sync():
86      data = request.data
87      session = json.loads(data)
88      c = dbcon.cursor()
89      c.execute("SELECT * FROM Users WHERE uid = ?", (session['mid'],))
90      master = c.fetchone()
91      cert = Certificate.load(binascii.unhexlify(master[5]))
92      n = cert.public_key.native['public_key']['modulus']
93      e = cert.public_key.native['public_key']['public_exponent']
94
95      hash_package = ''.join(sorted([attn['cid'] for attn in session['attns']]))
96      digest = SHA256.new()
97      digest.update(hash_package)
98
99      public_key = RSA.construct((n, e))
100
101     verifier = PKCS1_v1_5.new(public_key)
102     verified = verifier.verify(digest, binascii.unhexlify(session['sig']))
103
104     if verified:
```

```
105         dbcon.execute("INSERT INTO Sessions (sid, sig, master) VALUES (?, ?, ?)", (session['sid'], session['sig'], session['master']))
106         for attn in session['attns']:
107             db_tuple = (attn['sig'], attn['mid'], attn['uid'], attn['lat'], attn['lon'], attn['ts'], attn['sig'], attn['aid'], attn['confsig'], attn['cid'])
108             dbcon.execute("INSERT INTO Attendances (sid, mid, uid, lat, lon, ts, sig, aid, confsig, cid) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", db_tuple)
109         dbcon.commit()
110         return "", 201 # Created
111     else:
112         return "", 401 # Unauthorized
113
114 if __name__ == "__main__":
115         app.run(host='0.0.0.0', port=5000)
```

B.2 Android izvorni kod

Repo: https://github.com/koljenovic/logit/tree/master/android/app/src/main

```
.
├── AndroidManifest.xml
├── ic_launcher-web.png
├── java
│   └── ba
│       └── unsa
│           └── etf
│               └── logit
│                   ├── api
│                   │   └── LogitService.java
│                   ├── AttendanceActivity.java
│                   ├── AttendanceAdapter.java
│                   ├── LogitApduService.java
│                   ├── LogitApplication.java
│                   ├── MainActivity.java
│                   └── model
│                       ├── Attendance.java
│                       ├── Place.java
│                       ├── Session.java
│                       └── User.java
└── res
    ├── ---
    │
```

B.2.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ba.unsa.etf.logit">

    <uses-sdk android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.NFC" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <uses-feature
        android:name="android.hardware.nfc.hce"
        android:required="true" />
    <uses-feature android:name="android.hardware.location.gps" />

    <!-- To auto-complete the email text field in the login form with the user's emails -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.READ_PROFILE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.READ_CALL_LOG" />

    <application
        android:name=".LogitApplication"
        android:allowBackup="false"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <service
            android:name=".LogitApduService"
            android:permission="android.permission.BIND_NFC_SERVICE">
            <intent-filter>
                <action android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>

            <meta-data
                android:name="android.nfc.cardemulation.host_apdu_service"
                android:resource="@xml/apduservice" />
```

```
        </service>

        <intent-filter>
            <action android:name="android.nfc.action.NDEF_DISCOVERED" />

            <category android:name="android.intent.category.DEFAULT" />

            <data android:mimeType="application/octet-stream" />
        </intent-filter>

        <activity android:name=".MainActivity"
            android:noHistory="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".AttendanceActivity"
            android:theme="@style/AppTheme.NoActionBar"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.nfc.action.NDEF_DISCOVERED" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

B.2.2 model/Attendance.java

```
package ba.unsa.etf.logit.model;

import org.json.JSONObject;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import ba.unsa.etf.logit.LogitApplication;

public class Attendance {
    // Raw JSON version of this object
    public String raw;
    // Attendee name
    public String name;
```

```
16      public String surname;
17      // Attendee latitude
18      public String lat;
19      // Attendee longitude
20      public String lon;
21      // Attendee username - zamger
22      public String user;
23      // Attendee User ID - hex hash of public certificate key
24      public String uid;
25      // Hex Signature String of attendee package data (lat:lot:ts)
26      public String sig;
27      // Attendance ID - hex hash of signature
28      public String aid;
29      // Attendance TimeStamp from attendees device
30      public String ts;
31      // Is the signature valid check performed remotely by Logit Service on
    demand
32      public short valid;
33      // Master username - zamger
34      public String master;
35      // Master ID - hex hash of masters public certificate key
36      public String mid;
37      // Attendance Session ID
38      public String sid;
39      // Master Confirmation Signature - hex signature string of (sid:aid)
40      public String confsig;
41      // Confirmation ID - hex hash of confsig
42      public String cid;
43
44      public String getMid() {
45          return mid;
46      }
47
48      public void setMid(String mid) {
49          this.mid = mid;
50      }
51
52      public String getSid() {
53          return sid;
54      }
55
56      public void setSid(String sid) {
57          this.sid = sid;
58      }
59
60      public String getConfsig() {
61          return confsig;
62      }
63
```

```
64      public void setConfsig(String confsig) {
65          this.confsig = confsig;
66      }
67
68      public String getCid() {
69          return cid;
70      }
71
72      public void setCid(String cid) {
73          this.cid = cid;
74      }
75
76      public String getMaster() {
77          return master;
78      }
79
80      public void setMaster(String master) {
81          this.master = master;
82      }
83
84      public String getAid() {
85          return aid;
86      }
87
88      public void setAid(String aid) {
89          this.aid = aid;
90      }
91
92      public boolean isValidBasic() {
93          if (this.getRaw() != null &&
94                  this.getUser() != null &&
95                  this.getUid() != null &&
96                  this.getLat() != null &&
97                  this.getLon() != null &&
98                  this.getTs() != null &&
99                  this.getSig() != null) {
100             return true;
101         } else {
102             return false;
103         }
104     }
105
106     public String getRaw() {
107         return raw;
108     }
109
110     public void setRaw(String raw) {
111         this.raw = raw;
112     }
```

```
113
114     public String getSurname() {
115         return surname;
116     }
117
118     public void setSurname(String surname) {
119         this.surname = surname;
120     }
121
122     public String getLat() {
123         return lat;
124     }
125
126     public void setLat(String lat) {
127         this.lat = lat;
128     }
129
130     public String getLon() {
131         return lon;
132     }
133
134     public void setLon(String lon) {
135         this.lon = lon;
136     }
137
138     public String getUser() {
139         return user;
140     }
141
142     public void setUser(String user) {
143         this.user = user;
144     }
145
146     public String getUid() {
147         return uid;
148     }
149
150     public void setUid(String uid) {
151         this.uid = uid;
152     }
153
154     public String getSig() {
155         return sig;
156     }
157
158     public void setSig(String sig) {
159         this.sig = sig;
160     }
161
```

```
162     public String getTs() {
163         return ts;
164     }
165 
166     public void setTs(String ts) {
167         this.ts = ts;
168     }
169 
170     public Date getDate() {
171         return new Date(Long.parseLong(this.ts) * 1000L);
172     }
173 
174     public String getDateString() {
175         DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy (HH:mm:ss)");
176         return dateFormat.format(this.getDate());
177     }
178 
179     public Attendance(String raw) {
180 
181         try {
182             this.raw = raw;
183             JSONObject jResult = new JSONObject(raw);
184 
185             this.name = new
    String(LogitApplication.fromHext(jResult.getString("name")), "UTF-8");
186             this.surname = new
    String(LogitApplication.fromHext(jResult.getString("surname")), "UTF-8");
187             this.lat = jResult.getString("lat");
188             this.lon = jResult.getString("lon");
189             this.user = jResult.getString("user");
190             this.uid = jResult.getString("uid");
191             this.sig = jResult.getString("sig");
192             this.ts = jResult.getString("ts");
193             this.valid = jResult.has("valid") ? (short)jResult.getInt("valid")
    : 0;
194         } catch (Exception e) {
195             e.printStackTrace();
196         }
197     }
198 
199     public short getValid() {
200         return valid;
201     }
202 
203     public void setValid(short valid) {
204         this.valid = valid;
205     }
206 
207     public String getMail() {
```

```
208             return this.user + "@etf.unsa.ba";
209         }
210 
211         public String getName() {
212             return name;
213         }
214 
215         public String getFullName() {
216             return surname + ", " + name;
217         }
218 
219         public void setName(String name) {
220             this.name = name;
221         }
222 
223         public String getSigPkg() {
224             return this.getUser() + ":" + this.getLat() + ":" + this.getLon() + ":"
    + this.getTs();
225         }
226 
227         public String getConfSigPkg() {
228             return this.getMaster() + ":" + this.getSid() + ":" + this.getAid();
229         }
230     }
```

B.2.3 model/Place.java

```
1   package ba.unsa.etf.logit.model;
2   
3   import java.util.List;
4   import com.google.gson.annotations.Expose;
5   import com.google.gson.annotations.SerializedName;
6   
7   public class Place {
8   
9       @SerializedName("place_id")
10      @Expose
11      private String placeId;
12      @SerializedName("licence")
13      @Expose
14      private String licence;
15      @SerializedName("osm_type")
16      @Expose
17      private String osmType;
18      @SerializedName("osm_id")
19      @Expose
20      private String osmId;
```

```
21      @SerializedName("lat")
22      @Expose
23      private String lat;
24      @SerializedName("lon")
25      @Expose
26      private String lon;
27      @SerializedName("display_name")
28      @Expose
29      private String displayName;
30      @SerializedName("boundingbox")
31      @Expose
32      private List<String> boundingbox = null;
33
34      public String getPlaceId() {
35          return placeId;
36      }
37
38      public void setPlaceId(String placeId) {
39          this.placeId = placeId;
40      }
41
42      public String getLicence() {
43          return licence;
44      }
45
46      public void setLicence(String licence) {
47          this.licence = licence;
48      }
49
50      public String getOsmType() {
51          return osmType;
52      }
53
54      public void setOsmType(String osmType) {
55          this.osmType = osmType;
56      }
57
58      public String getOsmId() {
59          return osmId;
60      }
61
62      public void setOsmId(String osmId) {
63          this.osmId = osmId;
64      }
65
66      public String getLat() {
67          return lat;
68      }
69
```

```
70      public void setLat(String lat) {
71          this.lat = lat;
72      }
73
74      public String getLon() {
75          return lon;
76      }
77
78      public void setLon(String lon) {
79          this.lon = lon;
80      }
81
82      public String getDisplayName() {
83          return displayName;
84      }
85
86      public void setDisplayName(String displayName) {
87          this.displayName = displayName;
88      }
89
90      public List<String> getBoundingbox() {
91          return boundingbox;
92      }
93
94      public void setBoundingbox(List<String> boundingbox) {
95          this.boundingbox = boundingbox;
96      }
97
98  }
```

B.2.4 model/Session.java

```
1   package ba.unsa.etf.logit.model;
2
3   import java.util.List;
4
5   public class Session {
6       public String sid;
7       public String sig;
8       public String mid;
9       public String master;
10      public List<Attendance> attns;
11
12      public String getMid() {
13          return mid;
14      }
15
```

```
16      public void setMid(String mid) {
17          this.mid = mid;
18      }
19
20      public String getMaster() {
21          return master;
22      }
23
24      public void setMaster(String master) {
25          this.master = master;
26      }
27
28      public String getSid() {
29          return sid;
30      }
31
32      public void setSid(String sid) {
33          this.sid = sid;
34      }
35
36      public String getSig() {
37          return sig;
38      }
39
40      public void setSig(String sig) {
41          this.sig = sig;
42      }
43
44      public List<Attendance> getAttns() {
45          return attns;
46      }
47
48      public void setAttns(List<Attendance> attns) {
49          this.attns = attns;
50      }
51  }
```

B.2.5 model/User.java

```
1   package ba.unsa.etf.logit.model;
2
3   public class User {
4       public String name;
5       public String surname;
6
7       public User(String name, String surname) {
8           this.name = name;
```

```
9           this.surname = surname;
10      }
11
12      public String getName() {
13          return name;
14      }
15
16      public void setName(String name) {
17          this.name = name;
18      }
19
20      public String getSurname() {
21          return surname;
22      }
23
24      public void setSurname(String surname) {
25          this.surname = surname;
26      }
27  }
```

B.2.6 api/LogitService.java

```
1   package ba.unsa.etf.logit.api;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   import ba.unsa.etf.logit.model.Attendance;
7   import ba.unsa.etf.logit.model.Place;
8   import ba.unsa.etf.logit.model.Session;
9   import ba.unsa.etf.logit.model.User;
10  import okhttp3.OkHttpClient;
11  import okhttp3.ResponseBody;
12  import retrofit2.Call;
13  import retrofit2.http.Body;
14  import retrofit2.http.Field;
15  import retrofit2.http.FormUrlEncoded;
16  import retrofit2.http.GET;
17  import retrofit2.http.Headers;
18  import retrofit2.http.POST;
19  import retrofit2.http.Path;
20  import retrofit2.http.Query;
21
22  public interface LogitService {
23      @FormUrlEncoded
24      @POST("auth/")
```

```
25      Call<User> auth(@Field("user") String user, @Field("pass") String pass,
    @Field("cert") String cert, @Field("uid") String uid);
26
27      @POST("validate/")
28      Call<List<Attendance>> validate(@Body List<String> attns);
29
30      @POST("sync/")
31      Call<ResponseBody> sync(@Body Session session);
32
33      @Headers({
34              "User-Agent: ETF Logit v1.0b /SAPERE AVDE/",
35              "Referrer: http://etf.unsa.ba/"
36      })
37      @GET("reverse")
38      Call<Place> getAddress(@Query("email") String email, @Query("format")
    String format, @Query("lat") double lat, @Query("lon") double lon,
    @Query("zoom") int zoom, @Query("addressdetails") int addressdetails);
39  }
```

B.2.7 AttendanceActivity.java

```
1   package ba.unsa.etf.logit;
2
3   import android.content.SharedPreferences;
4   import android.content.pm.PackageManager;
5   import android.location.Location;
6   import android.nfc.NdefMessage;
7   import android.nfc.NdefRecord;
8   import android.nfc.Tag;
9   import android.nfc.tech.Ndef;
10  import android.os.AsyncTask;
11  import android.os.Build;
12  import android.os.Bundle;
13  import android.support.v4.content.ContextCompat;
14  import android.support.v7.app.AppCompatActivity;
15  import android.support.v7.widget.Toolbar;
16  import android.util.Log;
17  import android.app.Activity;
18  import android.app.PendingIntent;
19  import android.content.Intent;
20  import android.content.IntentFilter;
21  import android.content.IntentFilter.MalformedMimeTypeException;
22  import android.nfc.NfcAdapter;
23  import android.view.View;
24  import android.widget.ArrayAdapter;
25  import android.widget.ListView;
26  import android.widget.ProgressBar;
```

```
27  import android.widget.TextView;
28  import android.widget.Toast;
29
30  import com.google.android.gms.common.api.GoogleApiClient;
31  import com.google.android.gms.location.FusedLocationProviderClient;
32  import com.google.android.gms.location.LocationListener;
33  import com.google.android.gms.location.LocationRequest;
34  import com.google.android.gms.location.LocationServices;
35  import com.google.android.gms.tasks.OnSuccessListener;
36  import com.google.gson.Gson;
37
38  import org.json.JSONObject;
39  import org.w3c.dom.Text;
40
41  import java.security.KeyStore;
42  import java.security.MessageDigest;
43  import java.security.Signature;
44  import java.security.cert.Certificate;
45  import java.util.ArrayList;
46  import java.util.Arrays;
47  import java.util.Collections;
48  import java.util.Date;
49  import java.util.Enumeration;
50  import java.util.HashSet;
51  import java.util.List;
52  import java.util.Set;
53
54  import ba.unsa.etf.logit.api.LogitService;
55  import ba.unsa.etf.logit.model.Attendance;
56  import ba.unsa.etf.logit.model.Place;
57  import ba.unsa.etf.logit.model.Session;
58  import ba.unsa.etf.logit.model.User;
59  import okhttp3.OkHttpClient;
60  import okhttp3.ResponseBody;
61  import retrofit2.Call;
62  import retrofit2.Callback;
63  import retrofit2.Response;
64  import retrofit2.Retrofit;
65  import retrofit2.converter.gson.GsonConverterFactory;
66
67
68  public class AttendanceActivity extends AppCompatActivity implements
    GoogleApiClient.ConnectionCallbacks {
69
70      public static final String MIME = "application/octet-stream";
71      public static final String TAG = "Logit";
72      private AttendanceActivity that = this;
73
74      private ListView listview;
```

```
75      private List<Attendance> attns = new ArrayList<Attendance>();
76
77      private GoogleApiClient mGoogleApiClient;
78      private NfcAdapter mNfcAdapter;
79
80      @Override
81      protected void onCreate(Bundle savedInstanceState) {
82          super.onCreate(savedInstanceState);
83          setContentView(R.layout.activity_attendance);
84
85          Toolbar topToolbar = (Toolbar) findViewById(R.id.top_toolbar);
86          setSupportActionBar(topToolbar);
87          getSupportActionBar().setDisplayShowTitleEnabled(false);
88          SharedPreferences userData = getSharedPreferences("UserData", 0);
89
90          mGoogleApiClient = new GoogleApiClient.Builder(this)
91                  .addConnectionCallbacks(this).addApi(LocationServices.API)
92                  .build();
93          mGoogleApiClient.connect();
94
95          refreshLocation();
96
97          topToolbar.setTitle(userData.getString("surname", "Unknown") + ", " +
    userData.getString("name", "Unknown"));
98          topToolbar.setSubtitle(userData.getString("user", "unknown") +
    "@etf.unsa.ba");
99
100         listview = (ListView) findViewById(R.id.prisutni);
101
102         // Session ID control sequence
103         if(!userData.contains("sid")) {
104             SharedPreferences.Editor editor = userData.edit();
105             try {
106                 MessageDigest md = MessageDigest.getInstance("SHA-256");
107                 byte [] sidPayload = (userData.getString("user", "unknown") +
    System.currentTimeMillis()).getBytes();
108                 md.update(sidPayload, 0, sidPayload.length);
109                 editor.putString("sid", LogitApplication.toHext(md.digest()));
110                 editor.apply();
111             } catch (Exception e) {
112                 Toast.makeText(that, "Greška: neispravan CRYPT zahtjev.",
    Toast.LENGTH_LONG).show();
113             }
114         }
115
116         if(!userData.contains("attns")) {
117             SharedPreferences.Editor editor = userData.edit();
118             editor.putStringSet("attns", Collections.synchronizedSet(new
    HashSet<String>()));
```

```
119             editor.apply();
120         } else {
121             HashSet<String> attnSet = (HashSet<String>) userData.getStringSet("attns", Collections.synchronizedSet(new HashSet<String>()));
122             for (String s : attnSet) {
123                 attns.add(0, new Attendance(s));
124             }
125             if (!attns.isEmpty()) {
126                 Attendance[] attnsArray = (new Attendance[attns.size()]);
127                 attns.toArray(attnsArray);
128 
129                 final ArrayAdapter adapter = new AttendanceAdapter(that, attnsArray);
130                 listview.setAdapter(adapter);
131             }
132         }
133 
134         mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
135 
136         if (mNfcAdapter == null) {
137             // Stop here, we need NFC
138             Toast.makeText(this, "This device doesn't support NFC.", Toast.LENGTH_LONG).show();
139             finish();
140             return;
141 
142         }
143 
144         if (!mNfcAdapter.isEnabled()) {
145             // @TODO
146         } else {
147 
148         }
149     }
150 
151     @Override
152     protected void onResume() {
153         super.onResume();
154 
155         setupForegroundDispatch(this, mNfcAdapter);
156     }
157 
158     @Override
159     protected void onPause() {
160         stopForegroundDispatch(this, mNfcAdapter);
161 
162         super.onPause();
163     }
```

```
164
165     @Override
166     protected void onNewIntent(Intent intent) {
167         handleIntent(intent);
168     }
169
170     private void handleIntent(Intent intent) {
171         String action = intent.getAction();
172         if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
173
174             String type = intent.getType();
175             if (MIME.equals(type)) {
176
177                 Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
178                 new NdefReaderTask().execute(tag);
179
180             } else {
181                 Log.d(TAG, "Wrong mime type: " + type);
182             }
183         } else if (NfcAdapter.ACTION_TECH_DISCOVERED.equals(action)) {
184
185             // In case we would still use the Tech Discovered Intent
186             Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
187             String[] techList = tag.getTechList();
188             String searchedTech = Ndef.class.getName();
189
190             for (String tech : techList) {
191                 if (searchedTech.equals(tech)) {
192                     new NdefReaderTask().execute(tag);
193                     break;
194                 }
195             }
196         }
197     }
198
199     private class NdefReaderTask extends AsyncTask<Tag, Void, String> {
200
201         @Override
202         protected String doInBackground(Tag... params) {
203             Tag tag = params[0];
204
205             Ndef ndef = Ndef.get(tag);
206             if (ndef == null) {
207                 // NDEF is not supported by this Tag.
208                 return null;
209             }
210
211             NdefMessage ndefMessage = ndef.getCachedNdefMessage();
212
```

```
213                 NdefRecord[] records = ndefMessage.getRecords();
214                 for (NdefRecord ndefRecord : records) {
215                     if (ndefRecord.getTnf() == NdefRecord.TNF_MIME_MEDIA) {
216                         return readText(ndefRecord);
217                     }
218                 }
219 
220                 return null;
221             }
222 
223             private String readHext(NdefRecord record) {
224                 byte[] data = record.getPayload();
225                 return LogitApplication.toHext(data);
226             }
227 
228             private String readText(NdefRecord record) {
229                 byte[] data = record.getPayload();
230                 String ret;
231                 try {
232                     ret = new String(data, "UTF-8");
233                 } catch (Exception e) {
234                     ret = "Error";
235                     e.printStackTrace();
236                 }
237                 return ret;
238             }
239 
240             @Override
241             protected void onPostExecute(String result) {
242                 if (result != null) {
243                     try {
244                         final Attendance tmpAttn = new Attendance(result);
245                         if (tmpAttn.isValidBasic()) {
246                             for (Attendance a : attns) {
247                                 if (tmpAttn.getUid().equals(a.getUid()) ||
↳ tmpAttn.getUser().equals(a.getUser())) {
248 //                                if (tmpAttn.getUid().equals(a.getUid())) {
249                                     Toast.makeText(that, "Student potpisan.",
↳ Toast.LENGTH_LONG).show();
250                                     return;
251                                 }
252                             }
253                             final long timediff = System.currentTimeMillis() / 1000
↳ - Long.parseLong(tmpAttn.getTs());
254                             final Location userLocation = new Location("MOCK");
255 
↳ userLocation.setLatitude(Double.valueOf(tmpAttn.getLat()));
256 
↳ userLocation.setLongitude(Double.valueOf(tmpAttn.getLon()));
```

```
257                         if (Build.VERSION.SDK_INT >= 23
258                                 && ContextCompat.checkSelfPermission(that,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
259                                 && ContextCompat.checkSelfPermission(that,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
260                                 || Build.VERSION.SDK_INT < 23) {
261                             FusedLocationProviderClient mFusedLocatiionClient =
    LocationServices.getFusedLocationProviderClient(that);
262
    mFusedLocatiionClient.getLastLocation().addOnSuccessListener(new
    OnSuccessListener<Location>() {
263                                 @Override
264                                 public void onSuccess(Location location) {
265                                     Float locdiff =
    userLocation.distanceTo(location);
266  //                                   if (Math.abs(timediff) < 300) {
267                                     if (true) {
268  //                                       if (locdiff < 100) {
269                                         if (true) {
270                                             SharedPreferences userData =
    getSharedPreferences("UserData", 0);
271                                             SharedPreferences.Editor editor =
    userData.edit();
272                                             HashSet<String> attnSet = new
    HashSet<String>((HashSet<String>) userData.getStringSet("attns",
    Collections.synchronizedSet(new HashSet<String>())));
273                                             attnSet.add(tmpAttn.getRaw());
274                                             editor.putStringSet("attns",
    attnSet);
275                                             editor.apply();
276
277                                             attns.add(0, tmpAttn);
278
279                                             Attendance[] attnsArray = (new
    Attendance[attns.size()]);
280                                             attns.toArray(attnsArray);
281
282                                             final ArrayAdapter adapter = new
    AttendanceAdapter(that, attnsArray);
283                                             listview.setAdapter(adapter);
284                                         } else {
285                                             Toast.makeText(that, "Greška:
    lokacije udaljene " + locdiff.intValue() + " metara.",
    Toast.LENGTH_LONG).show();
286                                         }
287                                     } else {
```

```
288                                         Toast.makeText(that, "Greška: vrijeme
    nije tačno ili je TAG zastario.", Toast.LENGTH_LONG).show();
289                                     }
290                                 }
291                             });
292                         } else {
293                             Toast.makeText(that, "Greška: lokacija nije
    dostupna.", Toast.LENGTH_LONG).show();
294                             return;
295                         }
296                     } else {
297                         Toast.makeText(that, "Greška: TAG nije valjan.",
    Toast.LENGTH_LONG).show();
298                     }
299                 } catch (Exception e) {
300                     e.printStackTrace();
301                 }
302             }
303         }
304     }
305
306     public static void setupForegroundDispatch(final Activity activity,
    NfcAdapter adapter) {
307         final Intent intent = new Intent(activity.getApplicationContext(),
    activity.getClass());
308         intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);
309
310         final PendingIntent pendingIntent =
    PendingIntent.getActivity(activity.getApplicationContext(), 0, intent, 0);
311
312         IntentFilter[] filters = new IntentFilter[1];
313         String[][] techList = new String[][]{};
314
315         // Notice that this is the same filter as in the manifest
316         filters[0] = new IntentFilter();
317         filters[0].addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
318         filters[0].addCategory(Intent.CATEGORY_DEFAULT);
319
320         try {
321             filters[0].addDataType(MIME);
322         } catch (MalformedMimeTypeException e) {
323             throw new RuntimeException("Check your mime type.");
324         }
325
326         adapter.enableForegroundDispatch(activity, pendingIntent, filters,
    techList);
327     }
328
```

```
329     public static void stopForegroundDispatch(final Activity activity,
    NfcAdapter adapter) {
330         adapter.disableForegroundDispatch(activity);
331     }
332
333     @Override
334     public void onConnected(Bundle connectionHint) {
335         LocationRequest mLocationRequest = new LocationRequest();
336         mLocationRequest.setFastestInterval(10000);
337         mLocationRequest.setNumUpdates(3);
338         mLocationRequest.setSmallestDisplacement(1);
339         mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
340
341         if (Build.VERSION.SDK_INT >= 23
342                 && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
343                 && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
344                 || Build.VERSION.SDK_INT < 23) {
345
    LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
    mLocationRequest, new LocationListener() {
346                 @Override
347                 public void onLocationChanged(Location location) {
348                     Log.d("LOCATION", Double.toString(location.getLatitude()));
349                 }
350             });
351         }
352     }
353
354     @Override
355     public void onConnectionSuspended(int i) {
356
357     }
358
359     public void onSyncButton(View v) {
360         if (attns.size() > 0) {
361             final ProgressBar validateProgress = (ProgressBar)
    findViewById(R.id.validate_progress);
362             validateProgress.setVisibility(View.VISIBLE);
363             final SharedPreferences userData = getSharedPreferences("UserData",
    0);
364             final SharedPreferences.Editor editor = userData.edit();
365
366             try {
367                 final MessageDigest md = MessageDigest.getInstance("SHA-256");
368                 byte[] confSig;
```

```
369
370                 KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
371                 ks.load(null);
372                 KeyStore.ProtectionParameter pp = new
    KeyStore.PasswordProtection(null);
373
374                 // Get the most recent master secure entry element
375                 Enumeration<String> aliases = ks.aliases();
376                 String alias = aliases.nextElement();
377                 KeyStore.Entry entry = ks.getEntry(alias, pp);
378
379                 // Read in the master certificate
380                 Certificate c = ks.getCertificate(alias);
381
382                 // Instantiate a signature object and obtain the private key
383                 Signature s = Signature.getInstance("SHA256withRSA");
384                 s.initSign(((KeyStore.PrivateKeyEntry) entry).getPrivateKey());
385
386                 // Generate a hash for each attendance signature to be used as
    unique ID
387                 ArrayList<String> cidHashes = new ArrayList(attns.size());
388                 for (int i = 0; i < attns.size(); i++) {
389                     attns.get(i).setMaster(userData.getString("user",
    "unknown"));
390                     attns.get(i).setMid(userData.getString("uid", "unknown"));
391                     md.update(attns.get(i).getSig().getBytes());
392                     attns.get(i).setAid(LogitApplication.toHext(md.digest()));
393                     attns.get(i).setSid(userData.getString("sid", "unknown"));
394
395                     // Sign the confsig logit confirmation package
396                     s.update(attns.get(i).getConfSigPkg().getBytes());
397                     confSig = s.sign();
398
399                     attns.get(i).setConfsig(LogitApplication.toHext(confSig));
400                     md.update(confSig);
401                     attns.get(i).setCid(LogitApplication.toHext(md.digest()));
402                     cidHashes.add(attns.get(i).getCid());
403                 }
404                 Collections.sort(cidHashes);
405
406                 StringBuilder builder = new StringBuilder();
407                 for (String cidHash : cidHashes) {
408                     builder.append(cidHash);
409                 }
410                 String hashPackage = builder.toString();
411                 s.update(hashPackage.getBytes());
412                 byte [] rootSignature = s.sign();
413                 String rootHash = LogitApplication.toHext(rootSignature);
414
```

```
415                     Session session = new Session();
416                     session.setSid(userData.getString("sid", "unknown"));
417                     session.setSig(LogitApplication.toHext(rootSignature));
418                     session.setMid(userData.getString("uid", "unknown"));
419                     session.setMaster(userData.getString("user", "unknown"));
420                     session.setAttns(attns);
421 
422                     Retrofit retrofit = new Retrofit.Builder()
423                             .baseUrl(LogitApplication.SERVICE_URL)
424                             .addConverterFactory(GsonConverterFactory.create())
425                             .build();
426                     final LogitService service =
    retrofit.create(LogitService.class);
427 
428                     Call<ResponseBody> sync = service.sync(session);
429                     sync.enqueue(new Callback<ResponseBody>() {
430                         @Override
431                         public void onResponse(Call<ResponseBody> call,
    Response<ResponseBody> response) {
432                             if (response.code() == 201) {
433                                 // Reset the session ID and clear the previous
    attendances list
434                                 byte[] sidPayload = (userData.getString("user",
    "unknown") + System.currentTimeMillis()).getBytes();
435                                 md.update(sidPayload, 0, sidPayload.length);
436                                 editor.putString("sid",
    LogitApplication.toHext(md.digest()));
437                                 editor.putStringSet("attns",
    Collections.synchronizedSet(new HashSet<String>()));
438                                 editor.apply();
439                                 attns.clear();
440                                 listview.setAdapter(null);
441                                 Toast.makeText(that, "Podaci uspješno pohranjeni.",
    Toast.LENGTH_LONG).show();
442                             } else if (response.code() == 401) {
443                                 byte[] sidPayload = (userData.getString("user",
    "unknown") + System.currentTimeMillis()).getBytes();
444                                 md.update(sidPayload, 0, sidPayload.length);
445                                 editor.putString("sid",
    LogitApplication.toHext(md.digest()));
446                                 editor.putStringSet("attns",
    Collections.synchronizedSet(new HashSet<String>()));
447                                 editor.apply();
448                                 attns.clear();
449                                 listview.setAdapter(null);
450                                 Toast.makeText(that, "Greška: loš potpis sesije.",
    Toast.LENGTH_LONG).show();
451                             } else {
```

```
452                             Toast.makeText(that, "Greška: neuspješan Logit zahtjev.", Toast.LENGTH_LONG).show();
453                         }
454                         validateProgress.setVisibility(View.INVISIBLE);
455                     }
456 
457                     @Override
458                     public void onFailure(Call<ResponseBody> call, Throwable t) {
459                         Toast.makeText(that, "Greška: Logit servis nedostupan.", Toast.LENGTH_LONG).show();
460                         validateProgress.setVisibility(View.INVISIBLE);
461                     }
462                 });
463             } catch (Exception e) {
464                 e.printStackTrace();
465                 Toast.makeText(that, "Greška: neispravan CRYPT zahtjev.", Toast.LENGTH_LONG).show();
466             }
467         }
468     }
469 
470     protected void refreshLocation() {
471         final ProgressBar validateProgress = (ProgressBar) findViewById(R.id.validate_progress);
472         validateProgress.setVisibility(View.VISIBLE);
473 
474         FusedLocationProviderClient mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
475         Retrofit retrofit = new Retrofit.Builder()
476                 .baseUrl("https://nominatim.openstreetmap.org/")
477                 .addConverterFactory(GsonConverterFactory.create())
478                 .build();
479         final LogitService service = retrofit.create(LogitService.class);
480 
481         if (Build.VERSION.SDK_INT >= 23
482                 && ContextCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION ) == PackageManager.PERMISSION_GRANTED
483                 && ContextCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED
484                 || Build.VERSION.SDK_INT < 23) {
485 
486             mFusedLocationClient.getLastLocation().addOnSuccessListener(new OnSuccessListener<Location>() {
487                 @Override
488                 public void onSuccess(final Location location) {
489                     if (location != null) {
```

```
490                         Call<Place> validate =
    service.getAddress("mkoljenovic1@etf.unsa.ba", "json", location.getLatitude(),
    location.getLongitude(), 18, 0);
491                         validate.enqueue(new Callback<Place>() {
492                             @Override
493                             public void onResponse(Call<Place> call,
    Response<Place> response) {
494                                 if (response.code() == 200) {
495                                     Place p =  response.body();
496                                     String [] address =
    p.getDisplayName().split(", ");
497                                     TextView geoText = (TextView)
    findViewById(R.id.geoText);
498                                     if (address.length > 0) {
499                                         geoText.setText(address[0] + " (" +
    location.getLatitude() + ", " + location.getLongitude() + ")");
500                                     }
501                                     if (location.getTime() -
    System.currentTimeMillis() > 600000) {
502
    geoText.setBackgroundResource(android.R.color.holo_orange_light);
503                                     }
504                                 } else {
505                                     Toast.makeText(that, "Greška: neispravan
    OSM zahtjev.", Toast.LENGTH_LONG).show();
506                                 }
507                                 validateProgress.setVisibility(View.INVISIBLE);
508                             }
509
510                             @Override
511                             public void onFailure(Call<Place> call, Throwable
    t) {
512                                 Toast.makeText(that, "Greška: OSM servis
    nedostupan.", Toast.LENGTH_LONG).show();
513                                 validateProgress.setVisibility(View.INVISIBLE);
514                             }
515                         });
516                     } else {
517                         Toast.makeText(that, "Greška: lokacija nije dostupna.",
    Toast.LENGTH_LONG).show();
518                         validateProgress.setVisibility(View.INVISIBLE);
519                     }
520                 }
521             });
522         }
523     }
524
525     public void onValidateButton(View v) {
```

```
526         final ProgressBar validateProgress = (ProgressBar)
    findViewById(R.id.validate_progress);
527         validateProgress.setVisibility(View.VISIBLE);
528         Retrofit retrofit = new Retrofit.Builder()
529                 .baseUrl(LogitApplication.SERVICE_URL)
530                 .addConverterFactory(GsonConverterFactory.create())
531                 .build();
532         LogitService service = retrofit.create(LogitService.class);
533         ArrayList<String> attnsRaw = new ArrayList<String>(attns.size());
534         for (Attendance attn : attns) {
535             attnsRaw.add(attn.getRaw());
536         }
537         Call<List<Attendance>> validate = service.validate(attnsRaw);
538         validate.enqueue(new Callback<List<Attendance>>() {
539             @Override
540             public void onResponse(Call<List<Attendance>> call,
    Response<List<Attendance>> response) {
541                 if (response.code() == 200) {
542                     attns.clear();
543                     for(Attendance a : response.body()) {
544                         attns.add(a);
545                     }
546                     Attendance[] attnsArray = (new Attendance[attns.size()]);
547                     attns.toArray(attnsArray);
548 
549                     final ArrayAdapter adapter = new AttendanceAdapter(that,
    attnsArray);
550                     listview.setAdapter(adapter);
551                 } else {
552                     Toast.makeText(that, "Greška: neispravan Logit zahtjev.",
    Toast.LENGTH_LONG).show();
553                 }
554                 validateProgress.setVisibility(View.INVISIBLE);
555             }
556 
557             @Override
558             public void onFailure(Call<List<Attendance>> call, Throwable t) {
559                 Log.d("validate", "error");
560                 Toast.makeText(that, "Greška: Logit servis nedosupan.",
    Toast.LENGTH_LONG).show();
561                 validateProgress.setVisibility(View.INVISIBLE);
562             }
563         });
564     }
565 
566     public void onBugButton(View v) {
567         final Intent emailIntent = new
    Intent(android.content.Intent.ACTION_SEND);
568 
```

```
569             getSupportActionBar().setDisplayShowTitleEnabled(false);
570             SharedPreferences userData = getSharedPreferences("UserData", 0);
571 
572             emailIntent.setType("plain/text");
573             emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, new
    String[]{"mkoljenovic1@etf.unsa.ba"});
574             emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Logit bug
    @" + userData.getString("user", "unknows") + ":" + userData.getString("uid",
    "unknown"));
575 
576             this.startActivity(Intent.createChooser(emailIntent, "Prijavite grešku
    putem e-maila ..."));
577         }
578 
579         public void onGeoButton(View v) {
580             refreshLocation();
581         }
582     }
```

B.2.8 AttendanceAdapter.java

```
1   package ba.unsa.etf.logit;
2   
3   import android.content.Context;
4   import android.support.annotation.NonNull;
5   import android.support.annotation.Nullable;
6   import android.view.LayoutInflater;
7   import android.view.View;
8   import android.view.ViewGroup;
9   import android.widget.ArrayAdapter;
10  import android.widget.TextView;
11  
12  import java.text.DateFormat;
13  import java.text.SimpleDateFormat;
14  import java.util.Date;
15  
16  import ba.unsa.etf.logit.model.Attendance;
17  
18  /**
19   * Created by koljenovic on 5/30/17.
20   */
21  
22  public class AttendanceAdapter extends ArrayAdapter<Attendance> {
23      private final Context context;
24      private final Attendance[] values;
25  
26      public AttendanceAdapter(Context context, Attendance[] values) {
```

```
27          super(context, R.layout.prisutni_row, values);
28
29          this.context = context;
30          this.values = values;
31      }
32
33      @NonNull
34      @Override
35      public View getView(int position, @Nullable View convertView, @NonNull
    ViewGroup parent) {
36          LayoutInflater inflater = (LayoutInflater) context
37                  .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
38          View rowView = inflater.inflate(R.layout.prisutni_row, parent, false);
39
40          TextView pName = (TextView) rowView.findViewById(R.id.pName);
41          TextView pMail = (TextView) rowView.findViewById(R.id.pMail);
42          TextView pDate = (TextView) rowView.findViewById(R.id.pDate);
43          TextView pSeq = (TextView) rowView.findViewById(R.id.pSeq);
44          TextView pMisc  = (TextView) rowView.findViewById(R.id.pMisc);
45
46          pName.setText(this.values[position].getFullName());
47          pMail.setText(this.values[position].getMail());
48
49          pDate.setText(this.values[position].getDateString());
50
51          pSeq.setText(Integer.toString(this.values.length - position));
52          short valid = this.values[position].getValid();
53          String msg = valid > 0 ? "Dobar potpis" : (valid < 0 ? "Loš potpis" :
    "");
54          pMisc.setText(msg);
55          return rowView;
56      }
57  }
```

B.2.9 LogitApduService.java

```
1   package ba.unsa.etf.logit;
2
3   import java.io.UnsupportedEncodingException;
4   import java.security.KeyStore;
5   import java.security.KeyStore.Entry;
6   import java.security.KeyStore.PrivateKeyEntry;
7   import java.security.MessageDigest;
8   import java.security.PublicKey;
9   import java.security.Signature;
10  import java.security.cert.Certificate;
11  import java.security.interfaces.RSAPublicKey;
```

```
12  import java.util.Enumeration;
13
14  import android.content.SharedPreferences;
15  import android.content.pm.PackageManager;
16  import android.location.Location;
17  import android.nfc.NdefMessage;
18  import android.nfc.NdefRecord;
19  import android.nfc.cardemulation.HostApduService;
20  import android.os.Build;
21  import android.os.Bundle;
22  import android.support.v4.content.ContextCompat;
23  import android.util.Log;
24
25  import com.google.android.gms.location.FusedLocationProviderClient;
26  import com.google.android.gms.location.LocationServices;
27  import com.google.android.gms.tasks.OnSuccessListener;
28
29
30  public class LogitApduService extends HostApduService {
31
32      final static int APDU_INS = 1;
33      final static int APDU_P1 = 2;
34      final static int APDU_P2 = 3;
35      final static int APDU_SELECT_LC = 4;
36      final static int APDU_READ_LE = 4;
37      final static int FILEID_CC = 0xe103;
38      final static int FILEID_NDEF = 0xe104;
39      final static byte INS_SELECT = (byte) 0xa4;
40      final static byte INS_READ = (byte) 0xb0;
41      final static byte INS_UPDATE = (byte) 0xd6;
42      final static byte P1_SELECT_BY_NAME = (byte) 0x04;
43      final static byte P1_SELECT_BY_ID = (byte) 0x00;
44      final static int DATA_OFFSET = 5;
45
46      final static byte[] DATA_SELECT_NDEF = {(byte) 0xd2, (byte) 0x76, (byte)
    0x00, (byte) 0x00, (byte) 0x85, (byte) 0x01, (byte) 0x01};
47      final static byte[] RET_COMPLETE = {(byte) 0x90, (byte) 0x00};
48      final static byte[] RET_NONDEF = {(byte) 0x6a, (byte) 0x82};
49      final static byte[] FILE_CC = {
50              (byte) 0x00, (byte) 0x0f,       // CCLEN - CC container size
51              (byte) 0x20,                    // Mapping version
52              (byte) 0x04, (byte) 0xff,       // MLe - max. read size
53              (byte) 0x08, (byte) 0xff,       // MLc - max. update size
54
55              // TLV Block (NDEF File Control)
56              (byte) 0x04,                    // Tag - Block type
57              (byte) 0x06,                    // Length
58              (byte) 0xe1, (byte) 0x04,       // File identifier
59              (byte) 0x04, (byte) 0xff,       // Max. NDEF file size
```

```
60              (byte) 0x00,                    // R permission
61              (byte) 0x00,                    // W permission
62      };
63      private final static String TAG = "LogitApduService";
64      private final static String ALL = "AllLogitApduService";
65      private CardSelect mCardSelect = CardSelect.SELECT_NONE;
66      private boolean mSelectNdef = false;
67      private byte[] mNdefFile = null;
68      private LogitApplication logitApp;
69      private FusedLocationProviderClient mFusedLocationClient;
70      protected String msg;
71
72      public LogitApduService() {
73          super();
74      }
75
76      private void generateSignature() {
77          try {
78              mFusedLocationClient =
    LocationServices.getFusedLocationProviderClient(this);
79
80              // App has to check if the user has granted an explicit permission
    to use the location
81              // This only applies to Android API level 23 and up
82              if (Build.VERSION.SDK_INT >= 23
83                      && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
    PackageManager.PERMISSION_GRANTED
84                      && ContextCompat.checkSelfPermission(this,
    android.Manifest.permission.ACCESS_COARSE_LOCATION) ==
    PackageManager.PERMISSION_GRANTED
85                      || Build.VERSION.SDK_INT < 23) {
86
87                  mFusedLocationClient.getLastLocation().addOnSuccessListener(new
    OnSuccessListener<Location>() {
88                      @Override
89                      public void onSuccess(Location location) {
90                          if (location != null) {
91                              try {
92                                  Long tsLong = System.currentTimeMillis() /
    1000;
93                                  String ts = tsLong.toString();
94                                  byte[] signature;
95
96                                  // Instantiate and load a Android KeyStore
    object
97                                  KeyStore ks =
    KeyStore.getInstance("AndroidKeyStore");
98                                  ks.load(null);
```

```
99                                  KeyStore.ProtectionParameter pp = new
  KeyStore.PasswordProtection(null);
100
101                                 // Get the most recent user secure entry
  element
102                                 Enumeration<String> aliases = ks.aliases();
103                                 String alias = aliases.nextElement();
104                                 Entry entry = ks.getEntry(alias, pp);
105
106                                 // Instantiate a digest object for hashing
107                                 MessageDigest md =
  MessageDigest.getInstance("SHA-256");
108
109                                 // Read in the user certificate
110                                 Certificate c = ks.getCertificate(alias);
111
112                                 // Generate public key hash as user identifier
113                                 byte [] pubKey = c.getPublicKey().getEncoded();
114                                 md.update(pubKey, 0, pubKey.length);
115                                 byte [] pubKeyHash = md.digest();
116                                 String pubKeyHashString =
  LogitApplication.toHext(pubKeyHash);
117
118                                 // Instantiate a signature object and obtain
  the private key
119                                 Signature s =
  Signature.getInstance("SHA256withRSA");
120                                 s.initSign(((PrivateKeyEntry)
  entry).getPrivateKey());
121
122                                 SharedPreferences userData =
  getSharedPreferences("UserData", 0);
123
124                                 // Prepare the logit data package to be signed
125                                 String sigPkg = userData.getString("user",
  "unknown") +
126                                         ":" + location.getLatitude() +
127                                         ":" + location.getLongitude() +
128                                         ":" + ts;
129
130                                 // Sign the logit data package
131                                 s.update(sigPkg.getBytes("UTF-8"));
132                                 signature = s.sign();
133
134                                 // Generate a tx package to be sent to master
135                                 msg = "{\"lat\":\"" + location.getLatitude() +
136                                         "\", \"lon\":\"" +
  location.getLongitude() +
137                                         "\", \"ts\":\"" + ts +
```

```
                                        "\", \"sig\":\"" +
LogitApplication.toHext(signature) +
                                        "\", \"uid\":\"" + pubKeyHashString +
                                        "\", \"name\":\"" +
LogitApplication.toHext(userData.getString("name",
"unknown").getBytes("UTF-8")) +
                                        "\", \"surname\":\"" +
LogitApplication.toHext(userData.getString("surname",
"unknown").getBytes("UTF-8")) +
                                        "\", \"user\":\"" +
userData.getString("user", "unknown") + "\"}";

                                    // Create a NDEF message from the tx package
                                    NdefMessage ndef =
createMessage(msg.getBytes("UTF-8"));
                                    byte[] ndefarray = ndef.toByteArray();

                                    // Prepare a NDEF file for HCE Tag emulation
                                    mNdefFile = new byte[ndefarray.length + 2];

                                    // Append length bytes as per NDEF NDFILE
specification
                                    mNdefFile[0] = (byte) ((ndefarray.length &
0xff00) >> 8);
                                    mNdefFile[1] = (byte) (ndefarray.length &
0x00ff);

                                    // Copy the NDEF message into the NDEF file
                                    System.arraycopy(ndefarray, 0, mNdefFile, 2,
ndefarray.length);

                                    logitApp.setMessage(mNdefFile);
                                } catch (Exception e) {
                                    e.printStackTrace();
                                }
                            }
                        }
                    });
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onDeactivated(int reason) {
            Log.d(TAG, "onDeactivated");
            mCardSelect = CardSelect.SELECT_NONE;
            mSelectNdef = false;
```

```
176     }
177
178     protected byte [] prepareRetData(byte [] commandApdu) {
179         return prepareRetData(commandApdu, null);
180     }
181
182     protected byte [] prepareRetData(byte [] commandApdu, byte [] src) {
183         if (src == null) {
184             Log.d(TAG, "return complete");
185             return RET_COMPLETE;
186         }
187
188         int offset = ((commandApdu[APDU_P1] & 0xff) << 8) | (commandApdu[APDU_P2] & 0xff);
189         Log.d(TAG, "offset: " + Integer.toString(offset));
190         int Le = commandApdu[APDU_READ_LE] & 0xff;
191         byte [] retData = new byte[Le + RET_COMPLETE.length];
192
193         // Copy payload data into R-APDU
194         System.arraycopy(src, offset, retData, 0, Le);
195         // Add terminator to R-APDU
196         System.arraycopy(RET_COMPLETE, 0, retData, Le, RET_COMPLETE.length);
197
198         Log.d(TAG, "******************************");
199         for (byte ch : retData) {
200             Log.d(TAG, Integer.toHexString(ch & 0xff));
201         }
202         Log.d(TAG, "******************************");
203
204         return retData;
205     }
206
207     @Override
208     public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
209         for (int i = 0; i < commandApdu.length; i++) {
210             Log.d(ALL, Integer.toHexString(commandApdu[i] & 0xff));
211         }
212
213         byte [] retData = RET_NONDEF;
214
215         switch (commandApdu[APDU_INS]) {
216             case INS_SELECT:
217
218                 switch (commandApdu[APDU_P1]) {
219                     case P1_SELECT_BY_NAME:
220                         Log.d(TAG, "select : name");
221                         // 1. NDEF Tag Application Select
222                         if (memCmp(commandApdu, DATA_OFFSET, DATA_SELECT_NDEF, 0, commandApdu[APDU_SELECT_LC])) {
```

```
223                             //select NDEF application
224                             Log.d(TAG, "select NDEF application");
225                             mSelectNdef = true;
226                             retData = prepareRetData(commandApdu);
227                         } else {
228                             Log.e(TAG, "select: fail");
229                         }
230                         break;
231 
232                     case P1_SELECT_BY_ID:
233                         Log.d(TAG, "select : id");
234                         if (mSelectNdef) {
235                             int file_id = 0;
236                             for (int loop = 0; loop < commandApdu[APDU_SELECT_LC]; loop++) {
237                                 file_id <<= 8;
238                                 file_id |= commandApdu[DATA_OFFSET + loop] & 0xff;
239                             }
240                             switch (file_id) {
241                                 case FILEID_CC:
242                                     Log.d(TAG, "select CC file");
243                                     mCardSelect = CardSelect.SELECT_CCFILE;
244                                     retData = prepareRetData(commandApdu);
245                                     break;
246 
247                                 case FILEID_NDEF:
248                                     Log.d(TAG, "select NDEF file");
249                                     mCardSelect = CardSelect.SELECT_NDEFFILE;
250                                     retData = prepareRetData(commandApdu);
251                                     break;
252 
253                                 default:
254                                     Log.e(TAG, "select: unknown file id : " + file_id);
255                                     break;
256                             }
257                         } else {
258                             Log.e(TAG, "select: not select NDEF app");
259                         }
260                         break;
261 
262                     default:
263                         Log.e(TAG, "select: unknown p1 : " + commandApdu[APDU_P1]);
264                         break;
265                 }
266                 break;
267 
```

```
268                 case INS_READ:
269                     Log.d(TAG, "read");
270                     if (mSelectNdef) {
271                         byte[] src = null;
272                         switch (mCardSelect) {
273                             case SELECT_CCFILE:
274                                 Log.d(TAG, "read cc file");
275                                 retData = prepareRetData(commandApdu, FILE_CC);
276                                 break;
277 
278                             case SELECT_NDEFFILE:
279                                 Log.d(TAG, "read ndef file");
280                                 retData = prepareRetData(commandApdu,
    logitApp.getMessage());
281                                 break;
282                         }
283                     } else {
284                         Log.e(TAG, "read: not select NDEF app");
285                     }
286                     break;
287 
288                 case INS_UPDATE:
289                     Log.d(TAG, "UPDATE not implemented");
290 
291                 default:
292                     Log.e(TAG, "unknown INS : " + commandApdu[APDU_INS]);
293                     break;
294             }
295 
296             if (retData == RET_NONDEF) {
297                 Log.d(TAG, "ret notdef");
298             }
299 
300             return retData;
301         }
302 
303         private boolean memCmp(final byte[] p1, int offset1, final byte[] p2, int
    offset2, int cmpLen) {
304             final int len = p1.length;
305             if ((len < offset1 + cmpLen) || (p2.length < offset2 + cmpLen)) {
306                 Log.d(TAG, "memCmp fail : " + offset1 + " : " + offset2 + " (" +
    cmpLen + ")");
307                 Log.d(TAG, "memCmp fail : " + len + " : " + p2.length);
308                 return false;
309             }
310 
311             boolean ret = true;
312             for (int loop = 0; loop < cmpLen; loop++) {
313                 if (p1[offset1 + loop] != p2[offset2 + loop]) {
```

```
                Log.d(TAG, "unmatch");
                ret = false;
                break;
            }
        }

        return ret;
    }


    //https://github.com/bs-nfc/WriteRTDUri/blob/master/src/jp/co/brilliantservice/android/writertdur
    private NdefMessage createUriMessage(int index, String uriBody) {
        try {
            byte[] uriBodyBytes = uriBody.getBytes("UTF-8");
            byte[] payload = new byte[1 + uriBody.length()];
            payload[0] = (byte) index;
            System.arraycopy(uriBodyBytes, 0, payload, 1, uriBodyBytes.length);
            return new NdefMessage(new NdefRecord[]{
                    new NdefRecord(NdefRecord.TNF_WELL_KNOWN, NdefRecord.RTD_TEXT, new byte[0], payload)
            });
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException(e);
        }
    }

    private NdefMessage createMessage(byte [] body) {
        try {
            NdefRecord r0 = NdefRecord.createMime("application/octet-stream", body);
            return new NdefMessage(r0);
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onCreate() {
        super.onCreate();
        logitApp = ((LogitApplication) this.getApplication());
        generateSignature();
    }

    enum CardSelect {
        SELECT_NONE,
        SELECT_CCFILE,
        SELECT_NDEFFILE,
    }
```

```
}
```

B.2.10 LogitApplication.java

```
package ba.unsa.etf.logit;

import android.app.Application;

public class LogitApplication extends Application {
    private byte[] message;
    public static final String SERVICE_URL = "https://logit.mine.nu:5000";

    public static String toHext(byte [] data) {
        StringBuilder buf = new StringBuilder();
        for (byte b : data) {
            int halfbyte = (b >>> 4) & 0x0F;
            int two_halfs = 0;
            do {
                buf.append((0 <= halfbyte) && (halfbyte <= 9) ? (char) ('0' + halfbyte) : (char) ('a' + (halfbyte - 10)));
                halfbyte = b & 0x0F;
            } while (two_halfs++ < 1);
        }
        return buf.toString();
    }

    public static byte[] fromHext(String sData) {
        int len = sData.length();
        byte[] data = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            data[i / 2] = (byte) ((Character.digit(sData.charAt(i), 16) << 4)
                    + Character.digit(sData.charAt(i + 1), 16));
        }
        return data;
    }

    public void setMessage(byte[] message) {
        this.message = new byte[message.length];
        System.arraycopy(message, 0, this.message, 0, message.length);
    }

    public byte[] getMessage() {
        return this.message;
    }
}
```

B.2.11 MainActivity.java

```java
package ba.unsa.etf.logit;

import android.Manifest;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.Build;
import android.security.KeyPairGeneratorSpec;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.Toast;

import java.math.BigInteger;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.KeyStore;
import java.security.MessageDigest;
import java.security.cert.Certificate;
import java.util.Calendar;
import java.util.Collections;
import java.util.Enumeration;
import java.util.List;

import javax.security.auth.x500.X500Principal;

import ba.unsa.etf.logit.api.LogitService;
import ba.unsa.etf.logit.model.User;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
47          setContentView(R.layout.activity_main);
48          try {
49              if (Build.VERSION.SDK_INT >= 23 &&
50                      ContextCompat.checkSelfPermission(this,
  android.Manifest.permission.ACCESS_FINE_LOCATION ) !=
  PackageManager.PERMISSION_GRANTED ) {
51                  ActivityCompat.requestPermissions(this, new String[] {
  Manifest.permission.ACCESS_FINE_LOCATION }, 1337);
52              }
53          } catch (Exception e) {
54              e.printStackTrace();
55          }
56
57          SharedPreferences userData = getSharedPreferences("UserData", 0);
58
59          if(userData.contains("uid")) {
60              Intent attnIntent = new Intent(this, AttendanceActivity.class);
61              startActivity(attnIntent);
62          }
63      }
64
65      public void onNewKeyButton(View v) {
66          final ProgressBar progressBar = (ProgressBar)
  findViewById(R.id.progressBar);
67          progressBar.setVisibility(View.VISIBLE);
68
69          Long tsLong = System.currentTimeMillis() / 1000;
70          String ts = tsLong.toString();
71          String certDer = null;
72          String pubKeyHashString = null;
73          final MainActivity that = this;
74
75          try {
76              KeyPairGenerator kpg = KeyPairGenerator.getInstance(
77                      "RSA", "AndroidKeyStore");
78              Calendar start = Calendar.getInstance();
79              Calendar end = Calendar.getInstance();
80              end.add(Calendar.YEAR, 1);
81
82              KeyPairGeneratorSpec spec =
83                      new
  KeyPairGeneratorSpec.Builder(this).setAlias("etf_logit_" + ts)
84                              .setKeySize(2048)
85                              .setSubject(new X500Principal("CN=users.etf.ba"))
86                              .setSerialNumber(BigInteger.valueOf(tsLong))
87
  .setStartDate(start.getTime()).setEndDate(end.getTime()).build();
88
89              kpg.initialize(spec);
```

```
90
91              // Ref: Android Security Internals: An In-Depth Guide to Android's
    Security Architecture By Nikolay Elenkov
92
93              KeyPair kp = kpg.generateKeyPair();
94
95              KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
96              ks.load(null);
97
98  //              List<String> aliasesList = Collections.list(aliases);
99  //              ListView existingCredList = (ListView)
    findViewById(R.id.existingCredList);
100 //              existingCredList.setAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, aliasesList));
101
102             // Get the most recent user secure entry element
103             Enumeration<String> aliases = ks.aliases();
104             String alias = aliases.nextElement();
105             KeyStore.ProtectionParameter pp = new
    KeyStore.PasswordProtection(null);
106             KeyStore.Entry entry = ks.getEntry(alias, pp);
107
108             // Instantiate a digest object for hashing
109             MessageDigest md = MessageDigest.getInstance("SHA-256");
110
111             // Read in the user certificate
112             Certificate c = ks.getCertificate(alias);
113
114             // Generate public key hash as user identifier
115             byte [] pubKey = c.getPublicKey().getEncoded();
116             md.update(pubKey, 0, pubKey.length);
117             byte [] pubKeyHash = md.digest();
118             pubKeyHashString = LogitApplication.toHext(pubKeyHash);
119
120             certDer = LogitApplication.toHext(c.getEncoded());
121             Log.d("DER", certDer);
122         } catch (Exception e) {
123             Log.d("logit", Log.getStackTraceString(e));
124         }
125
126         Retrofit retrofit = new Retrofit.Builder()
127                 .baseUrl(LogitApplication.SERVICE_URL)
128                 .addConverterFactory(GsonConverterFactory.create())
129                 .build();
130
131         final EditText usernameBox = (EditText) findViewById(R.id.usernameBox);
132         final String usernameValue = usernameBox.getText().toString();
133         EditText passwordBox = (EditText) findViewById(R.id.passwordBox);
134         String passwordValue = passwordBox.getText().toString();
```

```
135         final String uid = pubKeyHashString;
136
137         LogitService service = retrofit.create(LogitService.class);
138         Call<User> auth = service.auth(usernameValue, passwordValue, certDer,
    uid);
139         auth.enqueue(new Callback<User>() {
140             @Override
141             public void onResponse(Call<User> call, Response<User> response) {
142                 if (response.code() == 200) {
143                     User user = response.body();
144
145                     SharedPreferences userData =
    getSharedPreferences("UserData", 0);
146                     SharedPreferences.Editor editor = userData.edit();
147                     editor.putString("uid", uid);
148                     editor.putString("user", usernameValue);
149                     editor.putString("name", user.getName());
150                     editor.putString("surname", user.getSurname());
151                     editor.apply();
152
153                     Intent attnIntent = new Intent(that,
    AttendanceActivity.class);
154                     startActivity(attnIntent);
155                 } else {
156                     try {
157                         KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
158                         ks.load(null);
159                         Enumeration<String> aliases = ks.aliases();
160                         List<String> aliasesList = Collections.list(aliases);
161                         for (String a : aliasesList) {
162                             ks.deleteEntry(a);
163                         }
164                         Toast.makeText(that, "Došlo je do greške, pokušajte
    ponovo.", Toast.LENGTH_LONG).show();
165                     } catch (Exception e) {
166                         e.printStackTrace();
167                         progressBar.setVisibility(View.INVISIBLE);
168                     }
169                 }
170                 progressBar.setVisibility(View.INVISIBLE);
171             }
172
173             @Override
174             public void onFailure(Call<User> call, Throwable t) {
175                 Log.d("RESP", "err");
176                 Toast.makeText(that, "Došlo je do greške, pokušajte ponovo.",
    Toast.LENGTH_LONG).show();
177             }
178         });
```

```
179         }
180
181     }
```

Bibliografija

[1] EU, "Uredba (EU) 2016/679 Europskog parlamenta i Vijeća od 27. travnja 2016. o zaštiti pojedinaca u vezi s obradom osobnih podataka i o slobodnom kretanju takvih podataka te o stavljanju izvan snage Direktive 95/46/EZ (Opća uredba o zaštiti podataka)," 2016.

[2] J. R. Searle, S. Willis, *et al.*, *The construction of social reality*. Simon and Schuster, 1995.

[3] V. Cahill, E. Gray, J. M. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen, "Using trust for secure collaboration in uncertain environments," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 52–61, 2003.

[4] ISO/IEC, "Iso/iec 27001: 2013," *Information Technology-Security Techniques-Information Security Management Systems-Requirements*, 2013.

[5] "Android statistics dashboard." - https://developer.android.com/about/dashboards/index.html. Pristupano: 2017-09-01.

[6] "Nfc world - adoption statistics." - https://www.nfcworld.com/2014/02/12/327790/two-three-phones-come-nfc-2018/, 2014. Pristupano: 2017-09-11.

[7] R. Khan, S. Zawoad, M. M. Haque, and R. Hasan, "Who, when, and where? location proof assertion for mobile devices," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 146–162, Springer, 2014.

[8] B. Sterling and L. Wild, *Shaping things*. The MIT Press, 2005.