



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

IZJAVA O AUTORSTVU

Ja, **Malik (Zijad) Koljenović**, student Elektrotehničkog fakulteta, Univerziteta u Sarajevu, pod punom moralnom, materijalnom i krivičnom odgovornošću **izjavljujem** da je ovaj rad i pripadajući programski kod, pod naslovom “APLIKACIJA ZA EVIDENTIRANJE PRISUSTVA” u potpunosti rezultat vlastitog istraživanja, gdje su korišteni sadržaji drugih autora (tekst, ilustracije, tabele etc.) jasno označeni pozivanjem na izvor.

MALIK KOLJENović, 984/2015

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 technologies 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	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 (<i>en. near-field communication</i>)	19
4.1.1 NXP NTAG216	19
4.1.2 NDEF (<i>en. NFC Data Exchange Format</i>)	21
4.1.3 HCE (<i>en. Host card emulation</i>)	21
4.2 Ostalo	22
5 Izdvojeni detalji implementacije	23
5.1 Podatkovni i kriptografski primitivi	23
5.1.1 SPIM paket	23
5.1.2 SESS paket	25

6 Zaključak	26
Dodatak	27
A Korisničko uputstvo	28
A.1 Instruktorski način rada	30
B Izvorni kod	31
B.1 LAPI izvorni kod	31
B.1.1 <code>__init__.py</code>	31
B.2 Android izvorni kod	34
B.2.1 <code>AndroidManifest.xml</code>	35
B.2.2 <code>model/Attendance.java</code>	36
B.2.3 <code>model/Place.java</code>	41
B.2.4 <code>model/Session.java</code>	43
B.2.5 <code>model/User.java</code>	44
B.2.6 <code>api/LogitService.java</code>	45
B.2.7 <code>AttendanceActivity.java</code>	46
B.2.8 <code>AttendanceAdapter.java</code>	60
B.2.9 <code>LogitApuService.java</code>	61
B.2.10 <code>LogitApplication.java</code>	70
B.2.11 <code>MainActivity.java</code>	71
Bibliografija	76

Popis slika

3.1	Dijagram interakcije - uspješna registracija 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
4.1	NTAG216 organizacija memorije[1]	20
4.2	NFC HCE virtuelnog sigurnog elementa (SE)[2]	22
5.1	Dijagram klasa SPIM i SESS objekata	24
A.1	Zaglavlje aplikacije prikazuje aktivnog korisnika	30
A.2	Glavni izbornik, opisi funkcionalnosti u nastavku	30
A.3	Trenutno zabilježena lokacija korisničkog uređaja	30
A.4	Ordinalno numerisan spisak prisutnih studenata	30

Pojmovnik

ATTN repozitorij potpisanih prisustva spremljen na LAPI.

BUMP približavanje mobilnih uređaja, otvara jednosmjerni komunikacijski kanal u smjeru 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 (*en. International Organization for Standardization*) - Međunarodna organizacija za standardizaciju u Ženevi, Švicarska.

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

JSON (*en. Java Simple Object Notation*) - vrlo jednostavna šema serijalizacije objekata u tekstualni oblik.

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 (*en. near-field communication*) - skup komunikacijskih protokola omogućava uspostavu komunikacijskog kanala između dva uređaja koji se nalaze u neposrednoj blizini.

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 suprotstavljenim komercijalnim interesima sa druge strane. Ukoliko se u tom kontekstu posmatra aktuelna EU uredba o zaštiti podataka[3] (*en. GDPR*) postaje jasno da su digitalna tehnologija i komunikacije postale integralni dio društvene i emocionalno-psihološke realnosti[4], 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[5] 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[6], evidenciju prisustva u kontekstu naprednog elektronskog registra 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 elektronske 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 beskontaktnu 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.

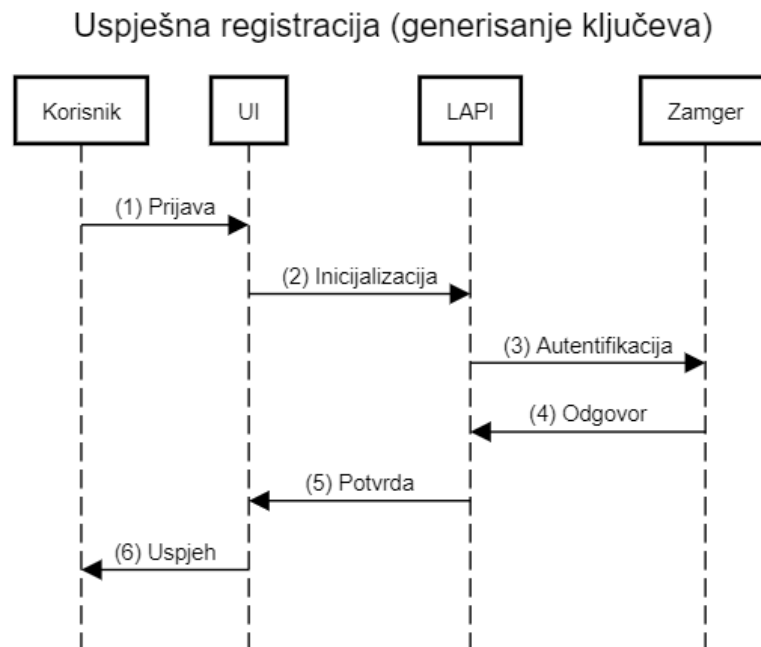
3 Prijedlog rješenja

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[7]. 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.[8]

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



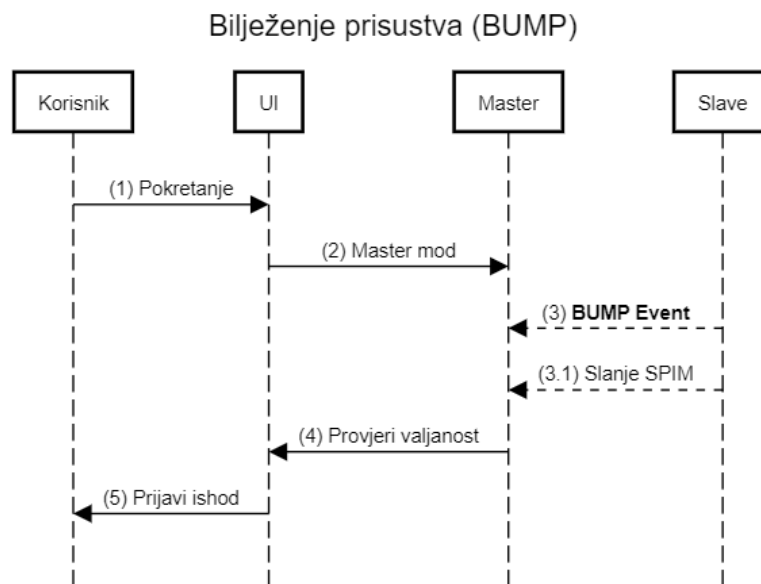
Slika 3.1: Dijagram interakcije - uspješna registracija i generisanje ključeva

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 uspješne 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

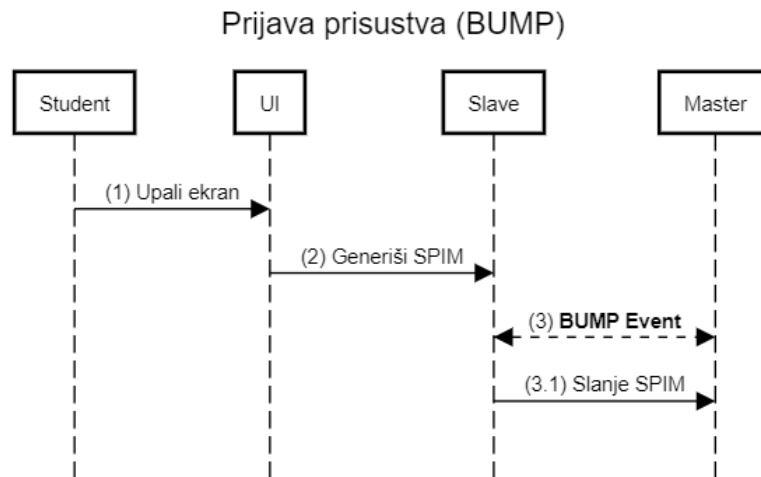
Bilježenje prisustva studenata od strane predmetnog nastavnika izvodi se u Master (M) modu funkcionisanja aplikacije, aplikacija se pri samom pokretanju i nakon uspješno obavljene registracija automatski stavlja u takav mod operacije i u njemu ostaje sve dok je upaljen ekran korisničkog uređaja (DEVICE) 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

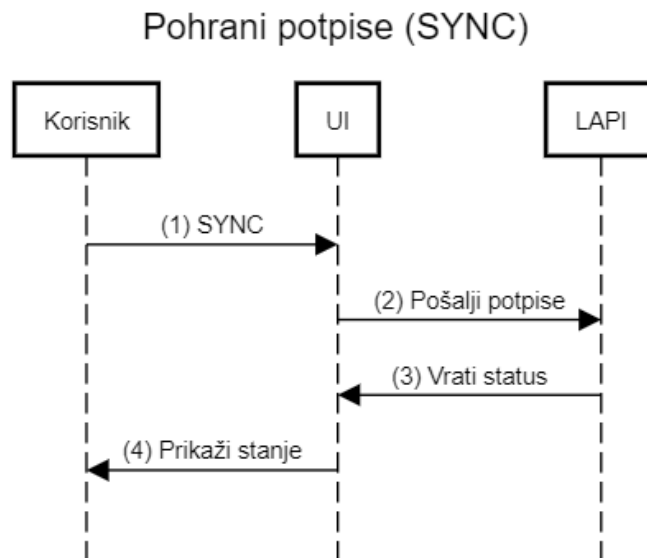


Slika 3.3: Dijagram interakcije - prijava prisustva studenta (Slave BUMP)

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 integriranim 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[9] koji u širem smislu u kontekstu poredenog 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 pohranjivanje 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).

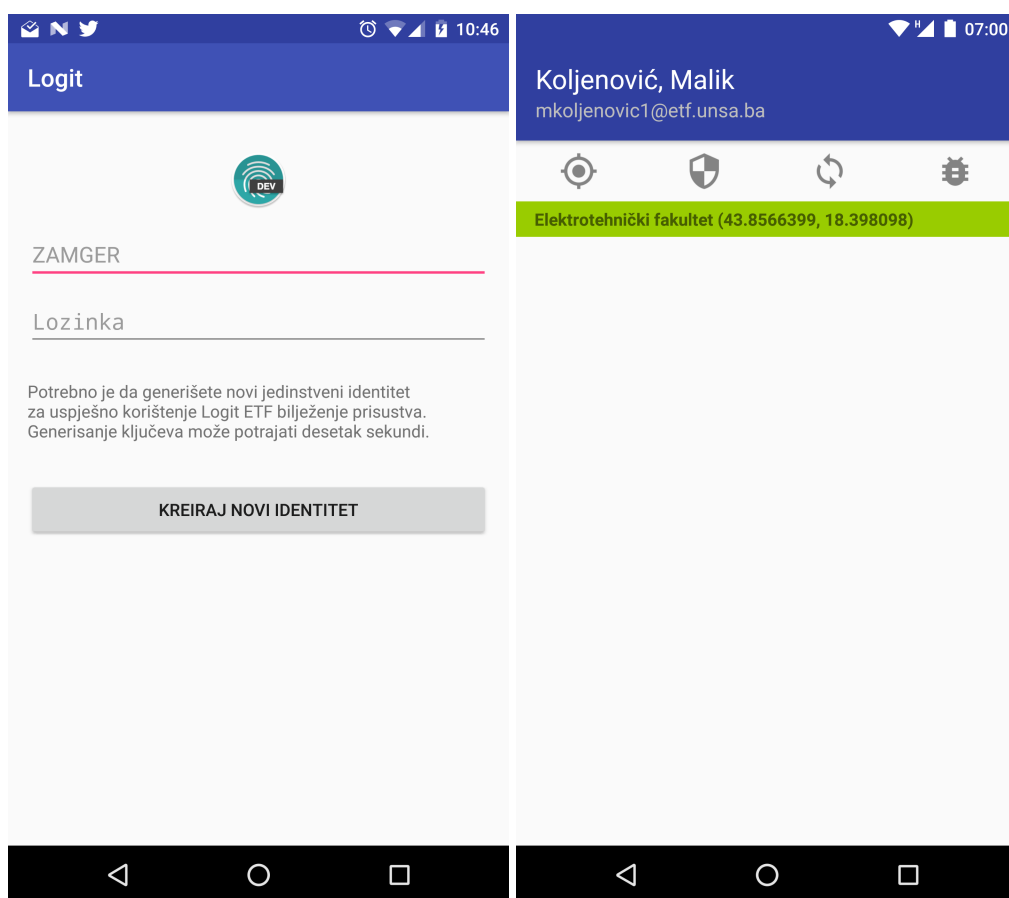
Da bi se osigurala jednostavnost korištenja aplikacije odabrana je implementacija HCE emulacijskog načina rada NFC komunikatora koji omogućava korisniku da izvrši komunikaciju sa drugim uređajem bez potrebe da pokreće aplikacijski prozor na svom uređaju, dovoljno je da upali ekran svoj uređaja i prinese 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 smjeru 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)[10].

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 zadovoljava 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 sesiji (SESS) od strane malicioznih aktera bez da naruši integritet SHA256 vrijednosti.

Uzimajući u obzir bitnost rješenja i visoku vjerovatnoću svakodnevne primjene kod ciljane korisničke grupe, te izazove koje takav slučaj korištenja predstavlja 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 (*en. near-field communication*)

NFC skup komunikacijskih protokola omogućava uspostavu komunikacijskog kanala između dva uređaja koji se nalaze u neposrednoj blizini jedan drugog (1-4 cm) i razmjenu podataka između njih[11]. Komunikacija se odvija na način da MASTER uređaj osluškuje signal na prijemniku i u slučaju detektovanje SLAVE signala uređaja pošiljaoca, propisanog istim standardom zaprima podatke i vrši obradu nad njima, komunikacija se u većini slučajeva odvija jednosmjerno kratkim standardiziranim porukama (NDEF), no moguće je ostvariti i half-duplex komunikaciju između uređaja, kao i razmjenu nestandardnih poruka, u kojem slučaju se sam korisnik mora pobrinuti za implementaciju kompletnog komunikacijskog protokola. Potpuni detalji implementacija dati su u referencama relevantnih standarda u nastavku tehničkog pregleda, odličnu sintezu detalja i implementacije daju Igoe, Coleman i Jepson[12].

4.1.1 NXP NTAG216

U cilju zadovoljenja postavljenih funkcionalnih zahtjeva bilo je neophodno odabrati NFC Tag platformu koja će odrediti relevantne standarde pohrane binarnih podataka na uređajima kao i pripadajuće komunikacijske protokole, također dodatno su postavljeni zahtjevi ekonomičnosti implementacije i kompatibilnosti sa postojećim čitačima. Uzimajući u obzir nabrojane kriterije odabrana je platforma NTAG216 proizvođača NXP Semiconductors[1] bazirana na NFC Forum Tag tipu 2 i ISO/IEC14443 Tip A specifikaciji[13][14].

Mogućnosti navedene platforme dostatne su za ispunjenje navedenih funkcionalnih uslova, a pružaju i neke dodatne sigurnosne mehanizme - poput neizmjenjivog jedinstvenog serijskog broja svakog taga (Tag UID) potpisanog kriptografskim ključem proizvođača, navedena funkcionalnost nije implementirana u predstavljenom rješenju jer se bazira na zaštićenoj NXP tehnologiji i nije kompatibilna sa HCE emulacijom, no umnogome može doprinijeti ukupnoj sigurnosti fizičkih Tag čipova u slučaju produkcijske implementacije rješenja. U nastavku je data proizvođačka lista izdvojenih funkcionalnosti NTAG216:

- 7-bajtni UID programiran od strane proizvođača za svaki tag
- mogućnost jednokratnog programiranja i zaključavanja taga za dalje izmjene
- mogućnost read-only zaključavanja taga
- potpis originalnosti baziran na kriptografiji eliptičnih krivih
- zaštita memorijskih operacija 32-bitnom lozinkom

Page Adr		Byte number within a page				Description
Dec	Hex	0	1	2	3	
0	0h	serial number				Manufacturer data and static lock bytes
1	1h	serial number				
2	2h	serial number	internal	lock bytes	lock bytes	
3	3h	Capability Container (CC)				Capability Container
4	4h	user memory				User memory pages
5	5h					
...	...					
224	E0h					
225	E1h	dynamic lock bytes				Dynamic lock bytes
226	E2h					
227	E3h	CFG 0				Configuration pages
228	E4h	CFG 1				
229	E5h	PWD				
230	E6h	PACK		RFUI		

aaa-008089

Slika 4.1: NTAG216 organizacija memorije[1]

4.1.2 NDEF (*en. NFC Data Exchange Format*)

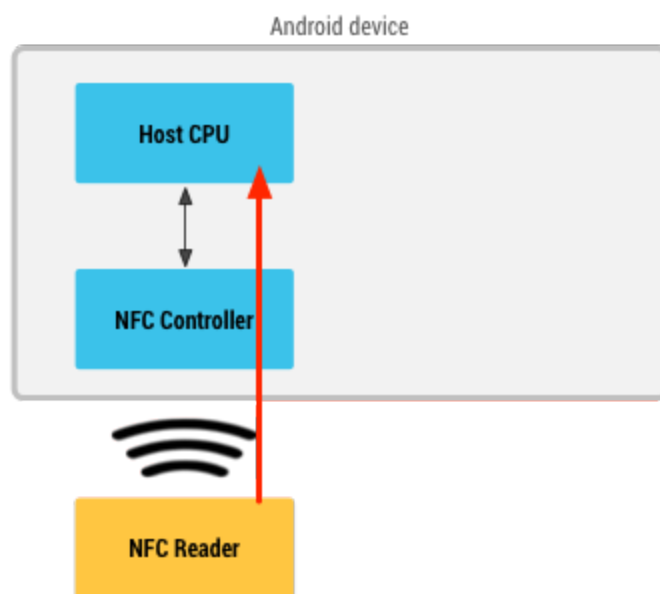
NDEF specifikacija definiše **format enkapsulacije poruke** za razmjenu informacija između dva NFC uređaja. NDEF je lagan binarni format poruke i može se koristiti za enkapsulaciju jednog ili više aplikacijski-definiranih tereta (*en. payload*) raznih vrsta i veličina unutar jedne NDEF poruke. Svaki teret opisan je od strane tipa, dužine i opcionalnog identifikatora. Identifikatori tipa mogu biti URI, MIME media tipovi, ili NFC-specifični tipovi. NDEF je striktno **format** poruke i ne poznaje pojam konekcije ili logičkog kola.[15]

Neki od ciljeva koje NDEF nastoji da ispuni:

- enkapsulacija dokumenata i binarnih objekata, slika etc.
- enkapsulacija podataka nepoznate dužine, npr. stream-a podataka
- agregacija srodnih sadržaja unutar jedne poruke
- kompaktna enkapsulacija malih datagrama

4.1.3 HCE (*en. Host card emulation*)

HCE je metod emuliranja virtuelnog identifikacijskog modula korisnika, u osnovi to je način zaobilaska hardverskih ograničenja (*en. hack, workaround*) koja onemogućavaju direktan pristup SIM (*en. Subscriber Identification Module*) modulu kod mobilnih telefonskih uređaja[16], ovakvo rješenje vuče korijene iz ekonomske realnosti sektora mobilnih komunikacija i kartičnog plaćanja, koja se najpreciznije može okarakterisati kao oligopol, naime Google je nastojao integrisati SIM karticu unutar Android operativnog sistema u vidu eSE (*en. embedded Secure Element*) korištenjem već postojeće SIM kartice operatera a u svrhu razvoja Google Wallet rješenja, no to nije odgovaralo operaterima i odbili su suradnju, nakon toga Google iznalazi alternativne načine rješenja problema poput HCE[17].



Slika 4.2: NFC HCE virtuelnog sigurnog elementa (SE)[2]

HCE na Android OS radi, kako i naziv govori u CE (*en. Card Emulation*) SLAVE modu, gdje se na svaki BUMP sa NFC čitačem odašilju pripremljeni podaci. Podaci koji se pri tome šalju moraju pratiti standard kartice koju žele emulirati i najčešće se vrši prijenos dokumenta ili datagrama unutar jednog ili više NDEF paketa. Logit koristi pristup prijenosa JSON formatiranog SPIM objekta plain/text unutar jednog NDEF paketa. Radi se o konceptu sa mnoštvom implementacijskih detalja, te zbog sažetosti za više detalja konsultujte oficijelnu dokumentaciju <https://developer.android.com/guide/topics/connectivity/nfc/hce>, dok je izvrstan logički prikaz sa primjerima dao Elenkov[18].

4.2 Ostalo

Dodatno koristi se Android arhitektura za dobavljanje geolokacije[19] uz reverzno geokodiranje od strane OpenStreetMap Nominatim projekta[20]. Za potpisivanje i enkripciju korisničkih podataka koristi se RSA[21] kriptografija sa 2048 bit ključem. LAPI je Python flask API i koristi UPnP, kompletan listing koda prikazan je u dodatku.[22]

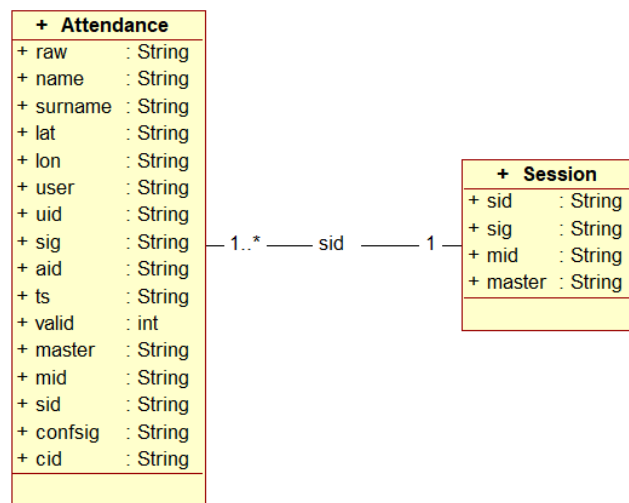
5 Izdvojeni detalji implementacije

Detalji izdvojeni u ovom poglavlju ključni su za razumijevanje sigurnosnog modela aplikacije, sa tog aspekta posebno su zanimljiva dva objekta, *Attendance* i *Session*, koji u osnovi predstavljaju proširene kriptografski potpisane SPIM i SESS objekte.

5.1 Podatkovni i kriptografski primitivi

5.1.1 SPIM paket

Osnovni SPIM objekat sastoji se iz korisničkog imena studenta, geografske širine, geografske dužine i trenutnog vremena na studentovom mobilnom uređaju. Vrijednosti tih stringova se lančaju u jedan string izloženim redoslijedom i takav string se potpisuje korištenjem RSA kriptografije, tako potpisan paket u obliku JSON objekta šalje se na profesorski master uređaj, gdje se dodaju podaci sesije, u vidu jedinstvenog identifikatora sesije (SID), te se potpisano studentsko prisustvo obilježava jedinstvenim heksadecimalnim identifikatorom AID izvedenim iz potpisa prisustva putem SHA256 hash funkcije, navedene vrijednosti, SID i AID se lančaju u jedan binarni string i potpisuju od strane profesora (CONFSIG), naknadno se na iz SHA256 hash vrijednosti CONFSIG profesorskog potpisa formira finalni identifikator potvrde prisustva CID, time se završava kriptografsko osiguravanje valjanosti prisustva u smislu SPIM objekta.



Slika 5.1: Dijagram klasa SPIM i SESS objekata

- raw** serijalizirana string JSON verzija objekta
- name** ime studenta
- surname** prezime studenta
- lat** geografska širina student
- lon** geografska dužina student
- user** ZAMGER korisničko ime studenta
- uid** User ID - SHA2 hex hash javnog ključa studenta
- sig** hex potpis SPIM-a (user:lat:lon:ts)
- aid** Attendance ID - hex SHA2 hash sig potpisa
- ts** vrijeme na studentovom uređaju
- valid** validation cache
- master** ZAMGER korisničko ime profesora
- mid** Master ID - SHA2 hex hash javnog ključa profesora
- sid** Session ID - identifikator profesorove sesije
- confsig** Master Conf. profesorov hex potpis (sid:aid)
- cid** Confirmation ID - SHA2 hex hash confsig-a

5.1.2 SESS paket

Dodatno se za SESS objekat prilikom finaliziranja sesije na LAPI serveru vrši prikupljanje svih CID potpisa koji pripadaju datoj sesiji, te se CID vrijednosti ulančane hronološkim redoslijedom potpisuju LAPI ključem koji se nalazi samo na LAPI hardverskom uređaju, stoga je sigurnost LAPI servera od ključne važnosti za sigurnost ukupnog sistema. Ovako potvrđena sesija ne može biti naknadno mijenjana, lažirana ili porečena izvan LAPI izvršnog okruženja.

6 **Zaključak**

Dodaci

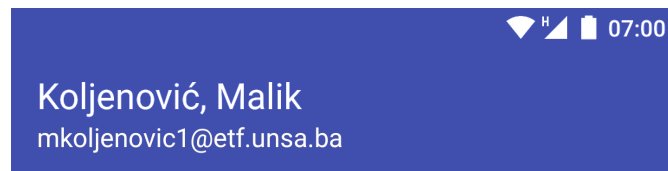
A Korisničko uputstvo

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 sigurnosne 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

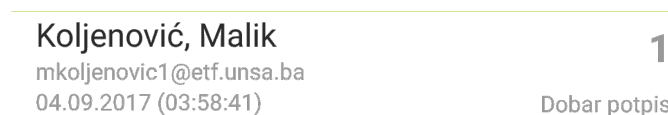
Dugme 2 koristite za provjeru valjanosti ključeva korištenih pri potpisu

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

Pripremljeno za štampanje uz pomoć \LaTeX -a i dopunskog paketa minted, Python interpretera i Pygments sintaksičkog parsera. Puni \LaTeX izvorni kod dostupan na <https://github.com/koljenovic/logit-tex>.

B.1 LAPI izvorni kod

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

```
1 .
2 |— Logit
3 |   |— Logit
4 |   |   |— __init__.py
5 |   |   |— logit.db
6 |   |   |— static
7 |   |— logit.wsgi
8 |— README.md
9 |— README.md~
```

B.1.1 `__init__.py`

```
1 from __future__ import print_function
2 from bs4 import BeautifulSoup
3 from flask import Flask, request
4 from asn1crypto.x509 import Certificate
5 from Crypto.PublicKey import RSA
6 from Crypto.Signature import PKCS1_v1_5
7 from Crypto.Hash import SHA256
8 import os, sys, sqlite3, requests, json, binascii
9
10 app = Flask(__name__)
11 db_name = 'logit.db'
12 dbcon = None
```



```

13
14 def init_db(dbcon):
15     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     passwd = 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': passwd})
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",
60             ↵ (attn['user'],))
61         certId = c.fetchone()
62         c.execute("SELECT * FROM Users WHERE id = ?", (certId[0],))
63         user = c.fetchone()
64         cert = Certificate.load(binascii.unhexlify(user[5]))
65         n = cert.public_key.native['public_key']['modulus']
66         e = cert.public_key.native['public_key']['public_exponent']
67         package = attn['user'] + ':' + attn['lat'] + ':' + attn['lon'] + ':' +
68             ↵ attn['ts']
69         digest = SHA256.new()
70         digest.update(package)
71
72         public_key = RSA.construct((n, e))
73
74         verifier = PKCS1_v1_5.new(public_key)
75         verified = verifier.verify(digest, binascii.unhexlify(attn['sig']))
76
77         attn['valid'] = 1 if verified else -1
78         attn['raw'] = json.dumps(attn)
79         attn['name'] = binascii.unhexlify(attn['name'])
80         attn['surname'] = binascii.unhexlify(attn['surname'])
81         result.append(attn)
82         # print(verified, file=sys.stderr)
83
84     return json.dumps(result)
85
86 @app.route("/sync/", methods=['POST'])
87 def sync():
88     data = request.data
89     session = json.loads(data)
90     c = dbcon.cursor()
91     c.execute("SELECT * FROM Users WHERE uid = ?", (session['mid'],))
92     master = c.fetchone()
93     cert = Certificate.load(binascii.unhexlify(master[5]))
94     n = cert.public_key.native['public_key']['modulus']
95     e = cert.public_key.native['public_key']['public_exponent']
96
97     hash_package = ''.join(sorted([attn['cid'] for attn in session['attns']]))
98     digest = SHA256.new()
99     digest.update(hash_package)
100
101     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>

```

1  .
2  ├── AndroidManifest.xml
3  ├── ic_launcher-web.png
4  ├── java
5  │   ├── ba
6  │   │   ├── unsa
7  │   │   │   ├── etf
8  │   │   │   │   ├── logit
9  │   │   │   │   │   ├── api
10 │   │   │   │   │   │   ├── LogitService.java
11 │   │   │   │   │   ├── AttendanceActivity.java
12 │   │   │   │   │   ├── AttendanceAdapter.java
13 │   │   │   │   │   ├── LogitApduService.java
14 │   │   │   │   │   ├── LogitApplication.java
15 │   │   │   │   │   ├── MainActivity.java
16 │   │   │   │   │   └── model
17 │   │   │   │   │       ├── Attendance.java
18 │   │   │   │   │       ├── Place.java
19 │   │   │   │   │       ├── Session.java
20 │   │   │   │   │       └── User.java
21 └── res

```

B.2.1 AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="ba.unsa.etf.logit">
4
5     <uses-sdk android:targetSdkVersion="19" />
6
7     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
8     <uses-permission android:name="android.permission.NFC" />
9     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
10
11     <uses-permission android:name="android.permission.READ_PHONE_STATE" />
12     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
13     <uses-permission android:name="android.permission.INTERNET"/>
14
15     <uses-feature
16         android:name="android.hardware.nfc.hce"
17         android:required="true" />
18     <uses-feature android:name="android.hardware.location.gps" />
19
20     <!-- To auto-complete the email text field in the login form with the
21          user's emails -->
22     <uses-permission android:name="android.permission.GET_ACCOUNTS" />
23     <uses-permission android:name="android.permission.READ_PROFILE" />
24     <uses-permission android:name="android.permission.READ_CONTACTS" />
25     <uses-permission android:name="android.permission.READ_CALL_LOG" />
26
27     <application
28         android:name=".LogitApplication"
29         android:allowBackup="false"
30         android:icon="@mipmap/ic_launcher"
31         android:label="@string/app_name"
32         android:supportsRtl="true"
33         android:theme="@style/AppTheme">
34         <service
35             android:name=".LogitApduService"
36             android:permission="android.permission.BIND_NFC_SERVICE">
37             <intent-filter>
38                 <action
39                     android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE" />
40
41                 <category android:name="android.intent.category.DEFAULT" />
42             </intent-filter>
43
44             <meta-data
45                 android:name="android.nfc.cardemulation.host_apdu_service"
46                 android:resource="@xml/apduservice" />
47         </service>
48     </application>
49 </manifest>
```

```

44     </service>
45
46     <intent-filter>
47         <action android:name="android.nfc.action.NDEF_DISCOVERED" />
48
49         <category android:name="android.intent.category.DEFAULT" />
50
51         <data android:mimeType="application/octet-stream" />
52     </intent-filter>
53
54     <activity android:name=".MainActivity"
55         android:noHistory="true"
56         android:screenOrientation="portrait">
57         <intent-filter>
58             <action android:name="android.intent.action.MAIN" />
59
60             <category android:name="android.intent.category.LAUNCHER" />
61         </intent-filter>
62     </activity>
63     <activity android:name=".AttendanceActivity"
64         android:theme="@style/AppTheme.NoActionBar"
65         android:screenOrientation="portrait">
66         <intent-filter>
67             <action android:name="android.nfc.action.NDEF_DISCOVERED" />
68         </intent-filter>
69     </activity>
70 </application>
71
72 </manifest>

```

B.2.2 model/Attendance.java

```

1 package ba.unsa.etf.logit.model;
2
3 import org.json.JSONObject;
4
5 import java.text.DateFormat;
6 import java.text.SimpleDateFormat;
7 import java.util.Date;
8
9 import ba.unsa.etf.logit.LogitApplication;
10
11 public class Attendance {
12     // Raw JSON version of this object
13     public String raw;
14     // Attendee name
15     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:lon: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
186             ↵ String(LogitApplication.fromHext(jResult.getString("name")), "UTF-8");
187             this.surname = new
188             ↵ String(LogitApplication.fromHext(jResult.getString("surname")), "UTF-8");
189             this.lat = jResult.getString("lat");
190             this.lon = jResult.getString("lon");
191             this.user = jResult.getString("user");
192             this.uid = jResult.getString("uid");
193             this.sig = jResult.getString("sig");
194             this.ts = jResult.getString("ts");
195             this.valid = jResult.has("valid") ? (short)jResult.getInt("valid")
196             ↵
197             ↵ : 0;
198         } catch (Exception e) {
199             e.printStackTrace();
200         }
201     }
202
203     public short getValid() {
204         return valid;
205     }
206
207     public void setValid(short valid) {
208         this.valid = valid;
209     }
210
211     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() + ":" +
225         ↵ + this.getTs();
226     }
227
228     public String getConfSigPkg() {
229         return this.getMaster() + ":" + this.getSid() + ":" + this.getAid();
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 /SAPER 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.AdapterView;
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.Enumuration;
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") + ", " +
98 ↵ userData.getString("name", "Unknown"));
99
100         topToolbar.setSubtitle(userData.getString("user", "unknown") +
101 ↵ "@etf.unsa.ba");
102
103         listview = (ListView) findViewById(R.id.prisutni);
104
105         // Session ID control sequence
106         if(!userData.contains("sid")) {
107             SharedPreferences.Editor editor = userData.edit();
108             try {
109                 MessageDigest md = MessageDigest.getInstance("SHA-256");
110                 byte [] sidPayload = (userData.getString("user", "unknown") +
111 ↵ System.currentTimeMillis()).getBytes();
112                 md.update(sidPayload, 0, sidPayload.length);
113                 editor.putString("sid", LogitApplication.toHex(md.digest()));
114                 editor.apply();
115             } catch (Exception e) {
116                 Toast.makeText(that, "Greška: neispravan CRYPT zahtjev.",
117 ↵ Toast.LENGTH_LONG).show();
118             }
119         }
120
121         if(!userData.contains("attns")) {
122             SharedPreferences.Editor editor = userData.edit();
123             editor.putStringSet("attns", Collections.synchronizedSet(new
124 ↵ 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()) ||
248                             tmpAttn.getUser().equals(a.getUser())) {
249                             // if (tmpAttn.getUid().equals(a.getUid())) {
250                                 Toast.makeText(that, "Student potpisan.",
251                                     Toast.LENGTH_LONG).show();
252                                 return;
253                             }
254                         }
255                     }
256                     final long timediff = System.currentTimeMillis() / 1000 -
257                         Long.parseLong(tmpAttn.getTs());
258                     final Location userLocation = new Location("MOCK");
259                     userLocation.setLatitude(Double.valueOf(tmpAttn.getLat()));
260                     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 mFusedLocationClient =
↳ LocationServices.getFusedLocationProviderClient(that);
262
↳ mFusedLocationClient.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                             attns.add(0, tmpAttn);
277                             Attendance[] attnsArray = (new
↳ Attendance[attns.size()]);
278                             attns.toArray(attnsArray);
279                             final ArrayAdapter adapter = new
↳ AttendanceAdapter(that, attnsArray);
280                             listview.setAdapter(adapter);
281                             } else {
282                                 Toast.makeText(that, "Greška:
↳ lokacije udaljene " + locdiff.intValue() + " metara.",
↳ Toast.LENGTH_LONG).show();
283                             }
284                             } 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,
330     ↵ NfcAdapter adapter) {
331         adapter.disableForegroundDispatch(activity);
332     }
333
334     @Override
335     public void onConnected(Bundle connectionHint) {
336         LocationRequest mLocationRequest = new LocationRequest();
337         mLocationRequest.setFastestInterval(10000);
338         mLocationRequest.setNumUpdates(3);
339         mLocationRequest.setSmallestDisplacement(1);
340         mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
341
342         if (Build.VERSION.SDK_INT >= 23
343             && ContextCompat.checkSelfPermission(this,
344             ↵ android.Manifest.permission.ACCESS_FINE_LOCATION ) ==
345             ↵ PackageManager.PERMISSION_GRANTED
346             && ContextCompat.checkSelfPermission(this,
347             ↵ android.Manifest.permission.ACCESS_COARSE_LOCATION ) ==
348             ↵ PackageManager.PERMISSION_GRANTED
349             || Build.VERSION.SDK_INT < 23) {
350             ↵ LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
351             ↵ mLocationRequest, new LocationListener() {
352                 @Override
353                 public void onLocationChanged(Location location) {
354                     Log.d("LOCATION", Double.toString(location.getLatitude()));
355                 }
356             });
357         }
358     }
359
360     @Override
361     public void onConnectionSuspended(int i) {
362     }
363
364     public void onSyncButton(View v) {
365         if (attns.size() > 0) {
366             final ProgressBar validateProgress = (ProgressBar)
367             ↵ findViewById(R.id.validate_progress);
368             validateProgress.setVisibility(View.VISIBLE);
369             final SharedPreferences userData = getSharedPreferences("UserData",
370             ↵ Context.MODE_PRIVATE);
371             final SharedPreferences.Editor editor = userData.edit();
372
373             try {
374                 final MessageDigest md = MessageDigest.getInstance("SHA-256");
375                 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
391             attns.get(i).setMid(userData.getString("uid", "unknown"));
392             md.update(attns.get(i).getSig().getBytes());
393             attns.get(i).setAid(LogitApplication.toHex(md.digest()));
394             attns.get(i).setSid(userData.getString("sid", "unknown"));
395
396             // Sign the confsig logit confirmation package
397             s.update(attns.get(i).getConfSigPkg().getBytes());
398             confSig = s.sign();
399
400             attns.get(i).setConfsig(LogitApplication.toHex(confSig));
401             md.update(confSig);
402             attns.get(i).setCid(LogitApplication.toHex(md.digest()));
403             cidHashes.add(attns.get(i).getCid());
404         }
405         Collections.sort(cidHashes);
406
407         StringBuilder builder = new StringBuilder();
408         for (String cidHash : cidHashes) {
409             builder.append(cidHash);
410         }
411         String hashPackage = builder.toString();
412         s.update(hashPackage.getBytes());
413         byte [] rootSignature = s.sign();
414         String rootHash = LogitApplication.toHex(rootSignature);

```



```

415         Session session = new Session();
416         session.setSid(userData.getString("sid", "unknown"));
417         session.setSig(LogitApplication.toHex(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.toHex(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.toHex(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
574             ↵ String[]{"mkoljenovic1@etf.unsa.ba"});
575         emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Logit bug
576             ↵ @" + userData.getString("user", "unknowns") + ":" + userData.getString("uid",
577             ↵ "unknown"));
578
579         this.startActivity(Intent.createChooser(emailIntent, "Prijavite grešku
580             ↵ putem e-maila ..."));
581     }
582
583     public void onGeoButton(View v) {
584         refreshLocation();
585     }
586 }

```

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 LogitApuService.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.HostApuService;
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 LogitApuService extends HostApuService {
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, // CLEN - 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 = "LogitApuService";
64     private final static String ALL = "AllLogitApuService";
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 LogitApuService() {
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
94                                 String ts = tsLong.toString();
95                                 byte[] signature;
96
97                                 // Instantiate and load a Android KeyStore
↳ object
98                                 KeyStore ks =
↳ KeyStore.getInstance("AndroidKeyStore");
99                                 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 +

```

```

138         "\", \"sig\\\":\\\"\" +
↳ LogitApplication.toHext(signature) +
139         "\", \"uid\\\":\\\"\" + pubKeyHashString +
140         "\", \"name\\\":\\\"\" +
↳ LogitApplication.toHext(userData.getString("name",
↳ "unknown").getBytes("UTF-8")) +
141         "\", \"surname\\\":\\\"\" +
↳ LogitApplication.toHext(userData.getString("surname",
↳ "unknown").getBytes("UTF-8")) +
142         "\", \"user\\\":\\\"\" +
↳ userData.getString("user", "unknown") + "\\}";
143
144         // Create a NDEF message from the tx package
145         NdefMessage ndef =
↳ createMessage(msg.getBytes("UTF-8"));
146         byte[] ndefarray = ndef.toByteArray();
147
148         // Prepare a NDEF file for HCE Tag emulation
149         mNdefFile = new byte[ndefarray.length + 2];
150
151         // Append length bytes as per NDEF NDFILE
↳ specification
152         mNdefFile[0] = (byte) ((ndefarray.length &
↳ 0xff00) >> 8);
153         mNdefFile[1] = (byte) (ndefarray.length &
↳ 0x00ff);
154
155         // Copy the NDEF message into the NDEF file
156         System.arraycopy(ndefarray, 0, mNdefFile, 2,
↳ ndefarray.length);
157
158         logitApp.setMessage(mNdefFile);
159     } catch (Exception e) {
160         e.printStackTrace();
161     }
162 }
163 }
164 });
165 }
166 } catch (Exception e) {
167     e.printStackTrace();
168 }
169 }
170
171 @Override
172 public void onDeactivated(int reason) {
173     Log.d(TAG, "onDeactivated");
174     mCardSelect = CardSelect.SELECT_NONE;
175     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) |
189 ↵ (commandApdu[APDU_P2] & 0xff);
190         Log.d(TAG, "offset: " + Integer.toString(offset));
191         int Le = commandApdu[APDU_READ_LE] & 0xff;
192         byte [] retData = new byte[Le + RET_COMPLETE.length];
193
194         // Copy payload data into R-APDU
195         System.arraycopy(src, offset, retData, 0, Le);
196         // Add terminator to R-APDU
197         System.arraycopy(RET_COMPLETE, 0, retData, Le, RET_COMPLETE.length);
198
199         Log.d(TAG, "*****");
200         for (byte ch : retData) {
201             Log.d(TAG, Integer.toHexString(ch & 0xff));
202         }
203         Log.d(TAG, "*****");
204
205         return retData;
206     }
207
208     @Override
209     public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
210         for (int i = 0; i < commandApdu.length; i++) {
211             Log.d(ALL, Integer.toHexString(commandApdu[i] & 0xff));
212         }
213
214         byte [] retData = RET_NONDEF;
215
216         switch (commandApdu[APDU_INS]) {
217             case INS_SELECT:
218
219                 switch (commandApdu[APDU_P1]) {
220                     case P1_SELECT_BY_NAME:
221                         Log.d(TAG, "select : name");
222                         // 1. NDEF Tag Application Select
223                         if (memCmp(commandApdu, DATA_OFFSET, DATA_SELECT_NDEF,
224 ↵ 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 <
237                 ↵ commandApdu[APDU_SELECT_LC]; loop++) {
238                 file_id <= 8;
239                 file_id |= commandApdu[DATA_OFFSET + loop] &
240                 ↵ 0xff;
241
242                 switch (file_id) {
243                     case FILEID_CC:
244                         Log.d(TAG, "select CC file");
245                         mCardSelect = CardSelect.SELECT_CCFILE;
246                         retData = prepareRetData(commandApdu);
247                         break;
248
249                     case FILEID_NDEF:
250                         Log.d(TAG, "select NDEF file");
251                         mCardSelect = CardSelect.SELECT_NDEFFILE;
252                         retData = prepareRetData(commandApdu);
253                         break;
254
255                     default:
256                         Log.e(TAG, "select: unknown file id : " +
257                         ↵ file_id);
258                         break;
259                 }
260             } else {
261                 Log.e(TAG, "select: not select NDEF app");
262             }
263             break;
264
265     default:
266         Log.e(TAG, "select: unknown p1 : " +
267         ↵ commandApdu[APDU_P1]);
268         break;
269     }
270     break;

```

```

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,
281                             ↵ logitApp.getMessage());
282                         break;
283                 }
284             } else {
285                 Log.e(TAG, "read: not select NDEF app");
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 ↵
304     ↵ offset2, int cmpLen) {
305         final int len = p1.length;
306         if ((len < offset1 + cmpLen) || (p2.length < offset2 + cmpLen)) {
307             Log.d(TAG, "memCmp fail : " + offset1 + " : " + offset2 + " (" +
308             ↵ cmpLen + ")");
309             Log.d(TAG, "memCmp fail : " + len + " : " + p2.length);
310             return false;
311         }
312
313         boolean ret = true;
314         for (int loop = 0; loop < cmpLen; loop++) {
315             if (p1[offset1 + loop] != p2[offset2 + loop]) {

```

```

314         Log.d(TAG, "unmatch");
315         ret = false;
316         break;
317     }
318 }
319
320 return ret;
321 }
322
323
324 ↪ https://github.com/bs-nfc/WriteRTDUri/blob/master/src/jp/co/brilliantservice/android/writertdu
325 private NdefMessage createUriMessage(int index, String uriBody) {
326     try {
327         byte[] uriBodyBytes = uriBody.getBytes("UTF-8");
328         byte[] payload = new byte[1 + uriBody.length()];
329         payload[0] = (byte) index;
330         System.arraycopy(uriBodyBytes, 0, payload, 1, uriBodyBytes.length);
331         return new NdefMessage(new NdefRecord[]{
332             ↪ new NdefRecord(NdefRecord.TNF_WELL_KNOWN,
333             ↪ NdefRecord.RTD_TEXT, new byte[0], payload)
334         });
335     } catch (UnsupportedEncodingException e) {
336         throw new RuntimeException(e);
337     }
338 }
339
340 private NdefMessage createMessage(byte [] body) {
341     try {
342         NdefRecord r0 = NdefRecord.createMime("application/octet-stream",
343         ↪ body);
344         return new NdefMessage(r0);
345     } catch (Exception e) {
346         e.printStackTrace();
347         throw new RuntimeException(e);
348     }
349 }
350
351 @Override
352 public void onCreate() {
353     super.onCreate();
354     logitApp = ((LogitApplication) this.getApplication());
355     generateSignature();
356 }
357
358 enum CardSelect {
359     SELECT_NONE,
360     SELECT_CCFILE,
361     SELECT_NDEFFILE,
362 }

```

360 }

B.2.10 LogitApplication.java

```
1 package ba.unsa.etf.logit;
2
3 import android.app.Application;
4
5 public class LogitApplication extends Application {
6     private byte[] message;
7     public static final String SERVICE_URL = "https://logit.mine.nu:5000";
8
9     public static String toHext(byte [] data) {
10         StringBuilder buf = new StringBuilder();
11         for (byte b : data) {
12             int halfbyte = (b >> 4) & 0x0F;
13             int two_halfs = 0;
14             do {
15                 buf.append((0 <= halfbyte) && (halfbyte <= 9) ? (char) ('0' +
16 halfbyte) : (char) ('a' + (halfbyte - 10)));
17                 halfbyte = b & 0x0F;
18             } while (two_halfs++ < 1);
19         }
20         return buf.toString();
21     }
22
23     public static byte[] fromHext(String sData) {
24         int len = sData.length();
25         byte[] data = new byte[len / 2];
26         for (int i = 0; i < len; i += 2) {
27             data[i / 2] = (byte) ((Character.digit(sData.charAt(i), 16) << 4)
28 + Character.digit(sData.charAt(i + 1), 16));
29         }
30         return data;
31     }
32
33     public void setMessage(byte[] message) {
34         this.message = new byte[message.length];
35         System.arraycopy(message, 0, this.message, 0, message.length);
36     }
37
38     public byte[] getMessage() {
39         return this.message;
40     }
41 }
```

B.2.11 MainActivity.java

```
1 package ba.unsa.etf.logit;
2
3 import android.Manifest;
4 import android.content.Intent;
5 import android.content.SharedPreferences;
6 import android.content.pm.PackageManager;
7 import android.os.Build;
8 import android.security.KeyPairGeneratorSpec;
9 import android.support.v4.app.ActivityCompat;
10 import android.support.v4.content.ContextCompat;
11 import android.support.v7.app.AppCompatActivity;
12 import android.os.Bundle;
13 import android.util.Log;
14 import android.view.View;
15 import android.widget.ArrayAdapter;
16 import android.widget.EditText;
17 import android.widget.ListView;
18 import android.widget.ProgressBar;
19 import android.widget.Toast;
20
21 import java.math.BigInteger;
22 import java.security.KeyPair;
23 import java.security.KeyPairGenerator;
24 import java.security.KeyStore;
25 import java.security.MessageDigest;
26 import java.security.cert.Certificate;
27 import java.util.Calendar;
28 import java.util.Collections;
29 import java.util.Enumeration;
30 import java.util.List;
31
32 import javax.security.auth.x500.X500Principal;
33
34 import ba.unsa.etf.logit.api.LogitService;
35 import ba.unsa.etf.logit.model.User;
36 import retrofit2.Call;
37 import retrofit2.Callback;
38 import retrofit2.Response;
39 import retrofit2.Retrofit;
40 import retrofit2.converter.gson.GsonConverterFactory;
41
42 public class MainActivity extends AppCompatActivity {
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46         super.onCreate(savedInstanceState);
```



```

47         setContentView(R.layout.activity_main);
48         try {
49             if (Build.VERSION.SDK_INT >= 23 &&
50                 ContextCompat.checkSelfPermission(this,
51                     android.Manifest.permission.ACCESS_FINE_LOCATION ) !=
52                     PackageManager.PERMISSION_GRANTED ) {
53                 ActivityCompat.requestPermissions(this, new String[] {
54                     Manifest.permission.ACCESS_FINE_LOCATION }, 1337);
55             }
56         } catch (Exception e) {
57             e.printStackTrace();
58         }
59
60         SharedPreferences userData = getSharedPreferences("UserData", 0);
61
62         if(userData.contains("uid")) {
63             Intent attnIntent = new Intent(this, AttendanceActivity.class);
64             startActivity(attnIntent);
65         }
66
67         public void onNewKeyButton(View v) {
68             final ProgressBar progressBar = (ProgressBar)
69                 findViewById(R.id.progressBar);
70             progressBar.setVisibility(View.VISIBLE);
71
72             Long tsLong = System.currentTimeMillis() / 1000;
73             String ts = tsLong.toString();
74             String certDer = null;
75             String pubKeyHashString = null;
76             final MainActivity that = this;
77
78             try {
79                 KeyPairGenerator kpg = KeyPairGenerator.getInstance(
80                     "RSA", "AndroidKeyStore");
81                 Calendar start = Calendar.getInstance();
82                 Calendar end = Calendar.getInstance();
83                 end.add(Calendar.YEAR, 1);
84
85                 KeyPairGeneratorSpec spec =
86                     new
87                     KeyPairGeneratorSpec.Builder(this).setAlias("etf_logit_" + ts)
88                         .setKeySize(2048)
89                         .setSubject(new X500Principal("CN=users.etf.ba"))
90                         .setSerialNumber(BigInteger.valueOf(tsLong))
91                         .setStartDate(start.getTime()).setEndDate(end.getTime()).build();
92
93                 kpg.initialize(spec);

```

```

90
91      // Ref: Android Security Internals: An In-Depth Guide to Android's
92      ↪ Security Architecture By Nikolay Elenkov
93
94      KeyPair kp = kpg.generateKeyPair();
95
96      KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
97      ks.load(null);
98
99      //      List<String> aliasesList = Collections.list(aliases);
100     //      ListView existingCredList = (ListView)
101     ↪ findViewById(R.id.existingCredList);
102     //      existingCredList.setAdapter(new ArrayAdapter<String>(this,
103     ↪ android.R.layout.simple_list_item_1, aliasesList));
104
105     // Get the most recent user secure entry element
106     Enumeration<String> aliases = ks.aliases();
107     String alias = aliases.nextElement();
108     KeyStore.ProtectionParameter pp = new
109     ↪ KeyStore.PasswordProtection(null);
110     KeyStore.Entry entry = ks.getEntry(alias, pp);
111
112     // Instantiate a digest object for hashing
113     MessageDigest md = MessageDigest.getInstance("SHA-256");
114
115     // Read in the user certificate
116     Certificate c = ks.getCertificate(alias);
117
118     // Generate public key hash as user identifier
119     byte [] pubKey = c.getPublicKey().getEncoded();
120     md.update(pubKey, 0, pubKey.length);
121     byte [] pubKeyHash = md.digest();
122     pubKeyHashString = LogitApplication.toHex(pubKeyHash);
123
124     certDer = LogitApplication.toHex(c.getEncoded());
125     Log.d("DER", certDer);
126 } catch (Exception e) {
127     Log.d("logit", Log.getStackTraceString(e));
128 }
129
130 Retrofit retrofit = new Retrofit.Builder()
131     .baseUrl(LogitApplication.SERVICE_URL)
132     .addConverterFactory(GsonConverterFactory.create())
133     .build();
134
135 final EditText usernameBox = (EditText) findViewById(R.id.usernameBox);
136 final String usernameValue = usernameBox.getText().toString();
137 EditText passwordBox = (EditText) findViewById(R.id.passwordBox);
138 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,
139         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 =
146                     getSharedPreferences("UserData", 0);
147                     SharedPreferences.Editor editor = userData.edit();
148                     editor.putString("uid", uid);
149                     editor.putString("user", usernameValue);
150                     editor.putString("name", user.getName());
151                     editor.putString("surname", user.getSurname());
152                     editor.apply();
153
154                     Intent attnIntent = new Intent(that,
155                     AttendanceActivity.class);
156                     startActivity(attnIntent);
157                 } else {
158                     try {
159                         KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
160                         ks.load(null);
161                         Enumeration<String> aliases = ks.aliases();
162                         List<String> aliasesList = Collections.list(aliases);
163                         for (String a : aliasesList) {
164                             ks.deleteEntry(a);
165                         }
166                         Toast.makeText(that, "Došlo je do greške, pokušajte
167                         ponovo.", Toast.LENGTH_LONG).show();
168                     } catch (Exception e) {
169                         e.printStackTrace();
170                         progressBar.setVisibility(View.INVISIBLE);
171                     }
172                 }
173                 progressBar.setVisibility(View.INVISIBLE);
174             }
175
176             @Override
177             public void onFailure(Call<User> call, Throwable t) {
178                 Log.d("RESP", "err");
179                 Toast.makeText(that, "Došlo je do greške, pokušajte ponovo.",
180                 Toast.LENGTH_LONG).show();
181             }
182         });

```

179 }

180

181 }

Bibliografija

- [1] NXP Semiconductors, "NTAG213/215/216 NFC Forum Type 2 Tag compliant IC with 144/504/888 bytes user memory," Tech. Rep. October, NXP Semiconductors, 2011.
- [2] Google. - <https://developer.android.com/guide/topics/connectivity/nfc/hce>, 2018. Pristupano: 2018-09-14.
- [3] 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.
- [4] J. R. Searle, S. Willis, *et al.*, *The construction of social reality*. Simon and Schuster, 1995.
- [5] 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.
- [6] ISO/IEC, "27001: 2013," standard, International Organization for Standardization, 2013.
- [7] "Android statistics dashboard." - <https://developer.android.com/about/dashboards/>. Pristupano: 2017-09-01.

- [8] "Nfc world - adoption statistics." - <https://nfcworld.com/2014/02/12/327790/two-three-phones-come-nfc-2018/>, 2014. Pristupano: 2017-09-11.
- [9] 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.
- [10] B. Sterling and L. Wild, *Shaping things*. The MIT Press, 2005.
- [11] NFC Forum, "NFC Digital Protocol Technical Specification," standard, NFC Forum, 2010.
- [12] T. Igoe, D. Coleman, and B. Jepson, *Beginning NFC*. O'Reilly, 2014.
- [13] NFC Forum, "Type 2 Tag Operation Specification," standard, NFC Forum, 2011.
- [14] ISO/IEC, "ISO/IEC 14443-1:2018(en) 1-4 - Cards and security devices for personal identification," standard, International Organization for Standardization, Geneva, CH, 2018.
- [15] NFC Forum, "NFC Data Exchange Format (NDEF)," standard, NFC Forum, 2006.
- [16] N. Elenkov, "Accessing the embedded secure element in android 4.x." - <https://nelenkov.blogspot.com/2012/08/accessing-embedded-secure-element-in.html>, 2012. Pristupano: 2018-09-14.
- [17] C. Paya. - <https://randomoracle.wordpress.com/2014/08/12/fakeid-android-nfc-stack-and-google-wallet-part-i/>, 2014. Pristupano: 2018-09-14.
- [18] N. Elenkov, *Android Security Internals*. no starch press, 2015.
- [19] Google. - <https://developer.android.com/guide/topics/location/strategies>, 2018. Pristupano: 2018-09-14.
- [20] OpenStreetMaps. - <https://wiki.openstreetmap.org/wiki/Nominatim>, 2018. Pristupano: 2018-09-14.

- [21] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [22] D. E. Knuth, "Literate programming," *The Computer Journal*, vol. 27, no. 2, pp. 97–111, 1984.