Moderni upravitelj zaporkama

Radošić, Leo-Siniša

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:752482

Rights / Prava: Attribution 3.0 Unported/Imenovanje 3.0

Download date / Datum preuzimanja: **2024-04-24**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Leo Siniša Radošić

Modreni upravitelj zaporkama

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Leo Siniša Radošić

Matični broj: 43296/14–R

Studij: Informacijski sustavi

Modreni upravitelj zaporkama

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Miroslav Bača

Varaždin, rujan 2018.

Leo Siniša Radošić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Podizanje razine sigurnosti korisnika novih tehnologija počinje s dobrom zaporkom. Mnogi korisnici danas imaju previše jednostavne zaporke, a osim toga za različite stranice koriste iste zaporke. Kako bi riješili taj problem moramo korisniku prezentirati upravitelj zaporkama koji će biti jednostavan za koristiti. Moderni upravitelj zaporkama nazvan MyPass korisnicima nudi jednostavno i moderno rješenje za njihovo upravljanje zaporkama. Ideja modernog upravitelj zaporkama je pretvoriti nešto što korisnik koristi svaki dan, a to su ključevi odnosno privjesak za ključeve, u digitalni format kako bi se to njemu dodatno približio. Tako korisnik kreira ključeve koje štiti koristeći najnovije značajke pametnih telefona poput otiska prsta, prepoznavanja lica i sl., a onda na jednostavan način koristi te ključeve te su njegovi računi dodatno zaštićeni.

Ključne riječi: upravitelj zaporkama, prepoznavanje lica, otisak prsta, mobilna aplikacija, web aplikacija

Sadržaj

Sadržaj iii
1. Uvod 1
2. Metode i tehnike rada 2
3. Sigurnost 3
   3.1. Mobilna aplikacija 3
   3.2. Web aplikacija 5
   3.3. Ekstenzija za preglednik i API 5
   3.4. Websocketi 5
4. API 7
   4.1. /api/Login PUT 7
   4.2. /api/Login POST 8
5. Web aplikacija 9
   5.1. Prijava / registracija na sustav 10
6. Mobilna aplikacija 12
7. Zaključak 14
8. Literatura 15
9. Popis slika 16

1. Uvod

Zbog opće loše ocjene sigurnosti korisnika koji nisu izrazito informatički pismeni te se često na internetu ostavljaju na milost i nemilost raznih napada to jest napadača koji to iskorištavaju kako bi izvukli od njih javlja se potreba za jednostavnim i modernim upraviteljem zaporki kako bi pokušali dodatno poboljšati barem taj korisnički dio to jest prebaciti tu „lopticu“ sigurnosti na one koji se time bave to jest koji prate sigurnosne preporuke, pa na taj način smanjiti mogućnost napada. Jedan upravitelj zaporkama korisniku mora omogućiti kreiranje zaporki koje će se teško probijati, biti dovoljno dugačke i teške kako ih napadač ne bi mogao otkriti, a onda je ostatak sigurnosti na samom proizvođaču aplikacije koju korisnik koristi.

Glavna ideja modernog upravitelj zaporkama nazvanog „MyPass“ je uzeti nešto što korisnik koristi stalno i prenijeti to u digitalni format. Ono što MyPass uzima je ideja ključa i privjesaka za ključeve. Dakle korisnik bi za svaku prijavu koristio određeni ključ kao što bi za razna vrata u svome životu koristio razne ključeve. Također korisnik te ključeve nosi na nekakvom privjesku što mu omogućava da si na neki način kategorizira ključeve.

Osim samo jednog pojedinačnog korisnika u tvrtkama se javlja potreba za dijeljenjem zaporki što raznih računa za određene usluge / aplikacije koje tvrtka koristi što za neke interne procese koje tvrtka sama definira. Tu se javlja potreba za upraviteljem zaporki koji će jednostavno moći podijeliti zaporku s nekim zaposlenikom to jest jednostavni ju prestati dijeliti ukoliko za to nastane potreba. Osim toga sam zaposlenik ne treba znati što je točno zaporka.

2. Metode i tehnike rada

Upravljanje zaporkama će se odvijati u par koraka. Prvi od tih koraka je mobilna aplikacija koje je u ovom slučaju aplikacija napisana za android sustav u Java programskom jeziku. Mobilna aplikacija korisniku omogućava sigurno kreiranja zaporki koje ona čuva u korisniku ne čitljivom obliku odnosno šifriranom obliku. Svi se podaci koje aplikacija čuva šifriraju nekim od biometrijskim mogućnostima mobitela primjerice otiskom prsta tako su oni vezani samo uz tog korisnika odnosno nitko osim njega ne može otključati ključeve (podatke) osim ako mu to ne omogući sam korisnik.

Drugi korak bi bio otključavanje računa to jest korištenje napravljenih ključeva. Ono se odvija u potpunosti unutar same aplikacije koju korisnik koristi a kako bi korisniku omogućili da MyPass može koristiti unutar svake aplikacije omogućavamo dva načina otključavanja. Prvi način je putem ekstenzije za preglednik u ovom slučaju ekstenzije za chrome preglednik koja je napisana u javascript programskom jeziku. Drugi način je da sam proizvođač aplikacije iskoristi MyPass API koji će mu omogućiti direktnu integraciju MyPass-a u njegovu aplikaciju a korisniku dodatnu razinu sigurnosti. MyPass api je napisan i C# programskom jeziku a koristi moderne tehnologije poput REST-a kako bi i proizvođaču omogućio jednostavno korištenje API-a.

Ono što ostaje za pojasniti je na koji način će aplikacije komunicirati te gdje tu dođe još i upravljanje zaporkama za tvrtke. Aplikacije komuniciraju putem websocketa koji je predstavljen u sklopu HTML5-a koji omogućuje otvaranje „tunela“ između njih kako bi oni mogli komunicirati. Ono što nam REST i websocketi omogućuju na području zaštite podataka će biti pojašnjeno u idućem poglavlju. Websocket je omogućen uz sami API a njegovo korištenje aplikacijama je vrlo jednostavno.

Tvrtke koje se odluče za koristiti MyPass imaju posebno web sučelje koje se nadovezuje na REST API koji je prethodno objašnjen. Sučelje je napravljeno koristeći tehnologije poput ASP.NET-a koji koristi C#, HTML, Javascript, CSS i sl. Njihove se zaporke prvobitno spremaju na server a onda kasnije sinkroniziraju na sve mobitele to jest privjeske radnika koji imaju pristup tim zaporkama. Sinkronizacija se odvija putem websocketa.

3. Sigurnost

Kako se radi o upravitelju zaporki sigurnost mora biti na visokoj razini. To je možda i jedan od najvažnijih segmenata samog sustava. Način na koji je aplikacija osigurana je tako da se svaki pojedini dio sustava osigurava sam za sebe što znači da ukoliko bilo koji dio sustava bude kompromitiran drugi dijelovi su sigurni. Što se tiće nekih protokola i slično unutar same aplikacije se ne koriste nikakva novo napisana rješenja, već protokoli koji su se već godinama pokazali kao sigurni te koji se redovito ažuriraju.

3.1. Mobilna aplikacija

Podaci koji se spremaju unutar mobilnih aplikacija spremaju se u bazu podataka koju nam taj sustav omogućuje, za android je to SQLite. Svi podaci su šifrirani to jest kriptirani sinkrono. Ključ za otključavanje podataka se čuva unutar sistemske baze ključeva koja se zove „Android key store“. Kako nije sigurno ključeve čuvati unutar aplikacije to jest morali bi ključ od pojedine aplikacije čuvati negdje na sustavu kao tekstualnu datoteku android unutar svog sustavu implementira key store koji nam omogućava sigurno čuvanje ključeva. Svaka aplikacija sprema svoj ključ a uz njega dodaje i određene atribute kojima onda opisuje što se sve može raditi s tim ključem.

Ključ generiramo na sljedeći način:

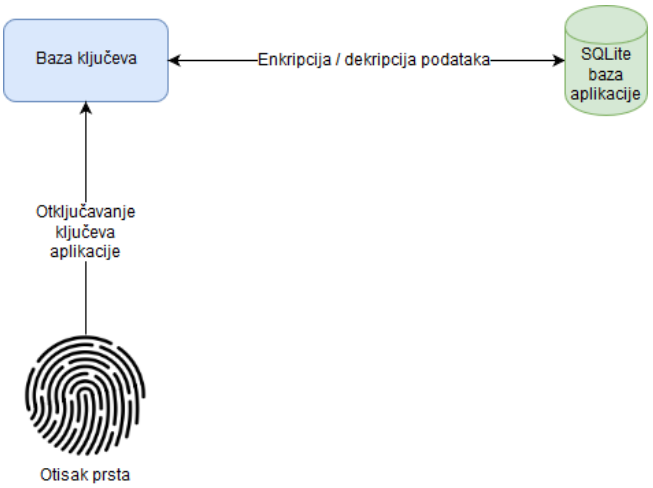
```
final KeyGenParameterSpec keyGenParameterSpec = new
KeyGenParameterSpec.Builder(alias,
        KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT)
        .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
       .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
        .build();
```

Slika 1. Isječak koda generiranja ključ [1]

Pomoću klase „KeyGenParameterSpec“ generiramo novi ključ tako da mu dodamo ime (varijabla alias na slici), nakon toga mu dodamo što ćemo raditi s tim ključem. Ostale postavke možemo ostaviti kakve jesu te nakon toga pomoću „.build()“ metode generirati ključ.

Ono što je nama zanimljivo prilikom postavljanja postavki ključa možemo mu dodati još jednu nama jako bitnu postavku, a to je postavka koja omogućuje korištenje ključa isključivo ako se korisnik prethodno prijavio putem nekog od biometrijskih mogućnosti mobitela. Tako da ukoliko dođe do neke manipulacije ili napada na mobitel androidov sustav zaštite neće

omogućiti nikome da dođe do ključeva, a na taj način čuva naše podatke to jest zaporke sigurnima.



Slika 2. Način zaštite korisnikovih podataka

Postavku koju koristimo za dodatnu zaštitu otiskom prsta je „setUserAuthenticationRequired“ koja onda govori programu da se taj ključ može otključati samo korisnik.

```
final KeyGenParameterSpec keyGenParameterSpec = new KeyGenParameterSpec.Builder(this.alias,
        KeyProperties.PURPOSE_ENCRYPT | KeyProperties.PURPOSE_DECRYPT)
        .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
        .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
        .setUserAuthenticationRequired(true)
        .build();
```

Slika 3. Isječak koda / Zaključaj ključ s otiskom prsta.

Mobilna aplikacija također sprema i podatke o konekcijama popit jedinstvenog identifikatora koji se sastoji od 6 parova znamenki zapisanih u heksadecimalnom obliku na primjer: 1A:2B:3C:4D:5E:6F te pina i za tim podacima postoji potreba da se čuvaju zaštićeno. Osim gore navedenih podataka mobilna aplikacija čuva i podatke o privjescima.

3.2. Web aplikacija

Podaci koje web aplikacija ima su zapravo svi vezani uz tvrtke i njihove zaporke. Svi se podaci formatiraju u JSON tip stringa a nakon toga šifrirani spremaju u .json datoteke. Za svaku firmu se radi zasebno šifriranje te su svi podaci od pojedini firme odvojeni. Također ti podaci imaju jednu dodatnu razinu sigurnosti pošto se oni nalaze na našem serveru, a s obzirom da se radi o web aplikaciji ključeve možemo čuvati u privremenoj memoriji te i koristiti po potrebi.

Svi podaci unutar web aplikacije se kriptiraju pomoću posebno napisane klase za spremanje podataka unutar koje možemo vrlo jednostavno mijenjati standarde koje koristimo za enkripciju i dekripciju. Klasa po nazivom „Database“ trenutno implementira dvije metode koje zapisuju json i čitaju json iz datoteka.

```
public static class Database
{
    public static string BasePutanja { get; set; } = HttpContext.Current.Server.MapPath("~") + "Data/";


    public static string ReadAllText(string putanja)
    {
        //TODO: implementirati zastitu
        //TODO: implementirati otvaranje za citanje/pisanje i sl.ž
        return File.ReadAllText(putanja);
    }

    public static void WriteAllText(string putanja, string text)
    {
        //TODO: implementirati zastitu
        //TODO: implementirati otvaranje za citanje/pisanje i sl.ž
        File.WriteAllText(putanja, text);
    }
}
```

Slika 4. Isječak koda / prikaz klase "Database"

3.3. Ekstenzija za preglednik i API

Ekstenzija za preglednik i API također trebaju čuvati podatke, oni čuvaju samo podatke o konekciji sa određenim privjeskom kao i mobilna aplikacija. Ti podaci nisu nikako posebno šifrirani ali postoje dodatni mehanizmi koji osiguravaju korisnika.

3.4. Websocketi

Kao što je već prethodno spomenuto za komunikaciju između korisnika i pretraživača se koriste websocketi. Brojne su prednosti korištenja websocketa a ona osnovna bi bila ta da nema potrebe za otvaranjem bilo kakvih dodatnih portova ili slično što bi bio slučaj kada bi se odlučili za druge http metode. Websocketi otvaraju tunel između MyPass mobilne aplikacije i ekstenzije za pretkazivače ili API-a i to tako da se i mobilna aplikacija i ekstenzija spoje na server. Kako oni oboje imaju svoj jedinstven broj server to prepoznaje te ih spaja i programski

radi tunel samo između njih dvoje. Tu dolazi jedan sigurnosni mehanizam koji je gore spomenut a to je da zapravo tunel može biti stvoren samo između dvije strane ukoliko se u taj proces uključi još jedna strana sustav to automatski prepoznaje te od korisnika traži unos novog pina na svojem mobitelu i na hostu te ponovnu prijavu.

Tu se može postaviti pitanje zašto baš websocketi, a osim jednostavnosti odgovor bi bio i sigurnost. Websocketi su dio http protokola te koriste port 80 i na većini poslužitelj s tim djelom ne bi trebalo biti problema. Osim toga websocketi mogu koristiti i ssl tako da svaki upit i podatak koji se šalje kanalom je šifriran a kako su zapravo websocketi dio sedmog sloja OSI modela internetskog protokola ispod njih je riješena zapravo sva sigurnost. Također tu što se samog httpa tiće važno je za napomenuti kako je to jedan od najkorištenijih protokola tako da on ima i dosta ažuriranja te ako se pojavi nekakav problem vezan uz njegovu zaštitu ažuriranje i rješavanje problema će vjerojatno biti u vrlo kratkom roku.

OSI Model			
Layer		Protocol data unit (PDU)	Function[3]
Host layers	7. Application	Data	High-level APIs, including resource sharing, remote file access
	6. Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5. Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4. Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2. Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1. Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

Slika 5. OSI model internet protokola [2]

4. API

Funkcionalnosti i API-a i web aplikacije su već prethodno opisani dakle nema ih potrebe ponovo opisivati. Ono što je važno za napomenuti su detalji vezani uz aplikaciju. Svaki dio API-a prima upite s tijelom koje je ili XML ili JSON koji se šalju uz odgovarajuća zaglavlja application/xml ili application/json.

API dio aplikacije koristi REST metode koje se slobodno pozivaju. Kreirane REST metode koje se mogu koristiti su:

- /api/Login PUT
- /api/Login POST

4.1. /api/Login PUT

Metoda koja se koristi za kreiranje nove konekcije. Dakle mobitel korisniku na ekran ispisuje pin koje korisnik upisuje u ekstenziju za pretraživače, nakon toga ekstenzija generira jedinstveni broj konekcije te pokušava registrirati konekciju na server. Ukoliko server nema već prije registriranu konekciju s tim jedinstvenim broj on sprema taj broj i pin te vraća da je uspješno spremio konekciju. Ukoliko taj broj postoji API će vratiti da nije uspio spremiti konekciju te će nakon toga ekstenzija generirati novi jedinstveni broj te pokušati ponovo. Još samo preostaje u mobitel unesti jedinstveni broj konekcije kako bi mogao započeti s radom.

Slika 6. Primjer tijela PUT upita u json sintaksi.

4.2. /api/Login POST

Kako bi smanjili mogućnost spajanja nepoznatih na websocket svaki član neke konekcije se mora prvo prijaviti kako bi dobio token kojim će se kasnije identificirati. Taj token može dobiti tako da se prijavi s jedinstvenim brojem konekcije i pinom te kao odgovor dobije token te do kada token vrijedi.


```
{
  "ConnectionID": {
    "P1": "66",
    "P2": "ED",
    "P3": "9B",
    "P4": "F1",
    "P5": "F4",
    "P6": "F2"
  },
  "PIN": 0,
  "Token":
"b8cd19ba9253d772a1589d5f40466f38ae9b9af8b6c225eb1b314648433629c23cfba919f53de90014db59c77d84a14941a4a
66a31a4729271c32874d932b9a0",
  "ValidUntil": "2018-08-10T14:54:06.4592839+02:00"
}
```


Slika 7. Primjer odgovora na /api/Login post upit u json formatu.

5. Web aplikacija

Kako je već gore opisano web aplikacija je napisana koristeći ASP.NET. Za sam dizajn aplikacije se koristi ručno napisan HTML i CSS dok za onu dinamičnu stranu kod klijenta odrađuje jQuery.

Korisnik se na Web aplikaciju prijavljuje putem API-a te na taj način prijavljuje na sustav nakon čega dobiva sve informacije koje su mu dostupe ovisno o tome što mu je administrator sustava dozvolio.

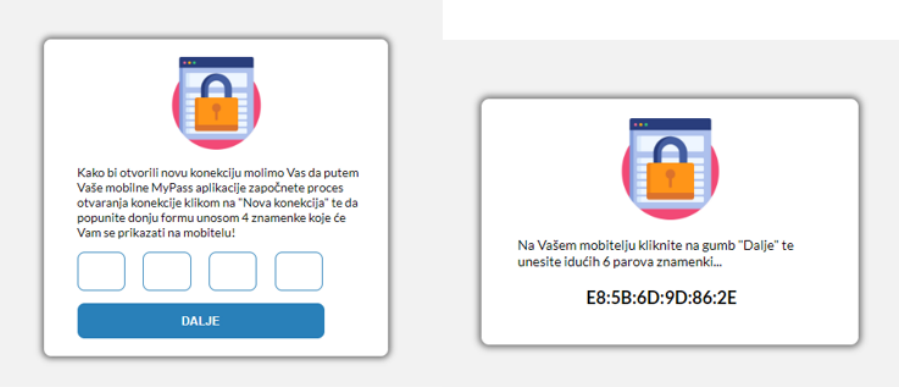
Ukoliko je korisnik administrator tvrtke on može mijenjati dodavati dodatne moderatore koji će mu pomoći u upravljanju zaporkama. Naravno on te moderatore dodaje pomoću njihovih privjesaka. Kada je administrator dodao moderatora on može upravljati svime kao što može i administrator samo ne može obrisati drugog moderatora i administratora. Njegova bi primarna zadaća bila dodavanje timova te postavljanje lidera tih timova.

Kada je određeni privjesak dobio privilegiju lidera tima on može dodavati druge korisnike u tim te kreirati nove zaporke koje će njegovi timski kolege koristiti. On bi trebao biti odgovoran za sve ključeve unutar njegovog tima, a određeni tim može imati više voditelja.

Običan korisnik u toj hijerarhiji ima najjednostavniji zadatak a taj je koristiti ključeve. On kada je dodan u tim prvom prilikom automatskom sinkronizacijom dohvaća sve ključeve te ih može koristiti za prijavu na račune koji su mu potrebni.

5.1. Prijava / registracija na sustav

Proces registracije započinje unosom PIN-a. PIN se dobije putem mobilne aplikacije te se nakon toga generira jedinstveni broj konekcije koji onda moramo unijeti na mobitelu.



Slika 8. Koraci registracije

Prilikom registracije aplikacija radi REST PUT upit na api/Login koji onda provjera postoji li prijava s tim brojem konekcije i PIN-om te ukoliko ne postoji sprema prijavu, a ukoliko postoji vraća false.

```
// PUT: api/Login
0 references | 0 changes | 0 authors, 0 changes | 1 request | 0 exceptions
public bool Put([FromBody]Login value)
{
    List<Login> list = (new Login()).Get();
    if (list.Where(l => l.ConnectionID.ToString() == value.ConnectionID.ToString()).ToList().Count == 0)
    {
        value.Add();
        return true;
    }
    else
    {
        return false;
    }
}
```

Slika 9. Isječak koda / Registracija konekcije

Jednom kada su se korisnik i aplikacija registrirali dalje se oboje moraju prijaviti kako bi dobili token koji će koristiti za identifikaciju unutar samog websocket-a. Taj token je ispravan samo 24 sata te nakon toga se obje aplikacije moraju ponovo prijaviti.

```
// POST: api/Login
0 references | 0 changes | 0 authors, 0 changes | 1 request | 0 exceptions
public Login Post([FromBody]Login value)
{
    List<Login> list = (new Login()).Get();
    if(list.Where(l => l.ConnectionID.ToString() == value.ConnectionID.ToString() && l.PIN == value.PIN).ToList().Count != 1)
    {
        return new Login();
    }
    else
    {
        value.ValidUntil = DateTime.Now.AddDays(1);
        value.Token = Hash.HashSHA512(value.ConnectionID.ToString() + DateTime.Now + "++MYPASS");
        value.Update();
        value.PIN = 0;
        return value;
    }
}
```

Slika 10. Isječak koda / Prijava na sustav

Token koji se dobije prijavom se dodaje u zaglavlje upita kojim se povezuje na websocket i tako da se postavi kolačić naziva „token“ kojemu se pridruži vrijednost tokena. Nažalost unutar samih websocketa ne postoji elegantnije rješenje ovog problema ali i ovo je dovoljno sigurno. Prilikom konekcije web aplikacija provjerava o kojem se tokenu radi te koliko je korisnika već unutar mreže s tom konekcijom tako da je to još jedan dodatni mehanizam zaštite korisnika od krađe podataka.

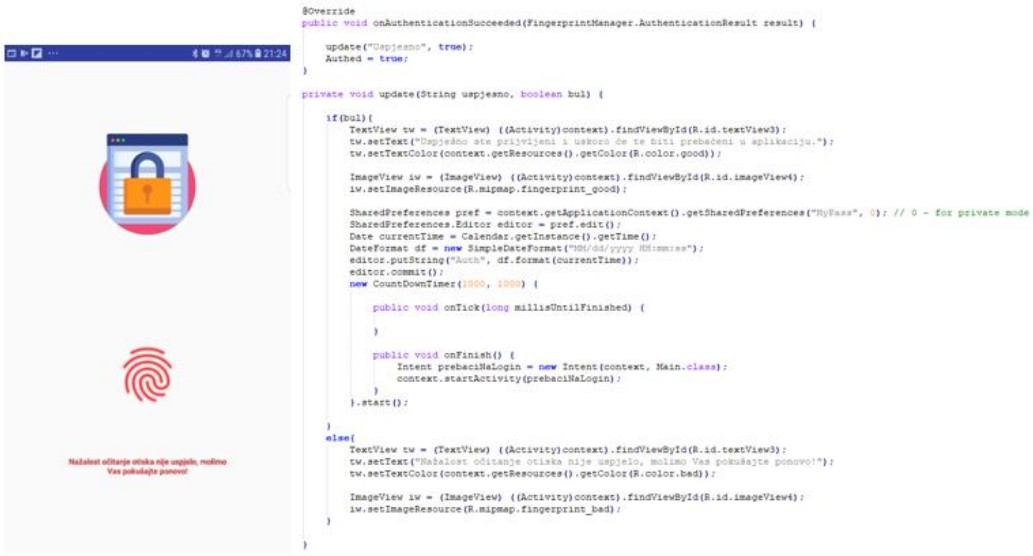
```
0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
public void ProcessRequest(HttpContext context)
{
    if (context.IsWebSocketRequest)
    {
        NameValueCollection parametri = context.Request.Params;

        string connectionType = "";
        string token = "";
        string id = "";

        try
        {
            connectionType = parametri["connection_type"];
            token = parametri["token"];
            id = parametri["id"];
        }
        catch { }

        if (!string.IsNullOrEmpty(connectionType) && !string.IsNullOrEmpty(token))
        {
            if (connectionType == "host" || connectionType == "client")
            {
                List<Login> logins = new Login().Get();
                Login login = logins.Where(l => l.Token == token).FirstOrDefault();

                if (login != null)
                {
                    WebSocketConnectionInfo info = new WebSocketConnectionInfo();
                    info.ConnectionID = login.ConnectionID;
                    info.ID = id;
                    if (connectionType == "host") { info.Host = true; } else { info.Host = false; }

                    context.AcceptWebSocketRequest(new MicrosoftWebSockets(info));
                }
            }
        }
    }
}
```

Slika 11. Isječak koda / Prihvaćanje websocket upita, spajanje na websocket

6. Mobilna aplikacija

Mobilna android aplikacija pisana u JAVA programskom jeziku koji pišemo nativno to jest ne koristimo nikakve hibridne programske jezike koji nam omogućuju pisanje aplikacija za više operativnih sustava iz razloga zato što nam nativni jezici omogućuju korištenje raznih prednosti samog sklopovlja mobitela.

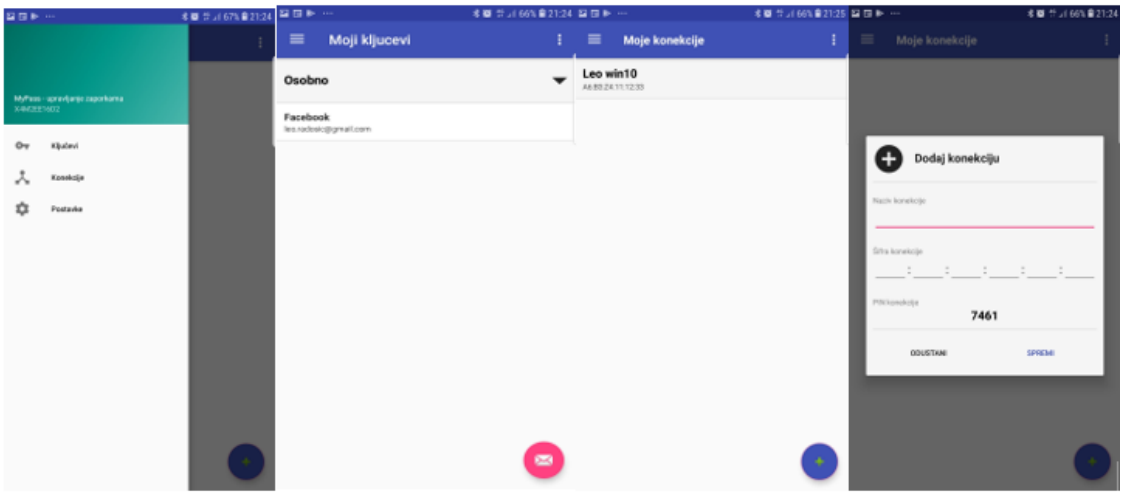
Prilikom prvog pokretanja aplikacije aplikacija od korisnika traži prijavu putem otiska prsta odnosno otključavanje ključeva za enkripciju i dekripciju. Jednom kada se korisnik uspješno prijavi sustav mu pamti prijavu idućih 15 minuta te ga prebacuje na drugu aktivnost koja je zapravo glavni ekran aplikacije.




```
@Override
public void onAuthenticationSucceeded(FingerprintManager.AuthenticationResult result) {

    update("Uspjesno", true);
    Authed = true;
}

private void update(String uspjesno, boolean bul) {

    if(bul){
        TextView tw = (TextView) ((Activity)context).findViewById(R.id.textView3);
        tw.setText("Uspješno ste prijavljeni i uskoro ce te biti prebaceni u aplikaciju.");
        tw.setTextColor(context.getResources().getColor(R.color.good));

        ImageView iw = (ImageView) ((Activity)context).findViewById(R.id.imageView4);
        iw.setImageResource(R.mipmap.fingerprint_good);

        SharedPreferences pref = context.getApplicationContext().getSharedPreferences("MyPass", 0); // 0 - for private mode
        SharedPreferences.Editor editor = pref.edit();
        Date currentTime = Calendar.getInstance().getTime();
        DateFormat df = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
        editor.putString("Auth", df.format(currentTime));
        editor.commit();
        new CountDownTimer(1000, 1000) {

            public void onTick(long millisUntilFinished) {

            }

            public void onFinish() {
                Intent prebaciNaLogin = new Intent(context, Main.class);
                context.startActivity(prebaciNaLogin);
            }
        }.start();

    }
    else{
        TextView tw = (TextView) ((Activity)context).findViewById(R.id.textView3);
        tw.setText("Nažalost očitanje otiska nije uspjelo, molimo Vas pokušajte ponovo!");
        tw.setTextColor(context.getResources().getColor(R.color.bad));

        ImageView iw = (ImageView) ((Activity)context).findViewById(R.id.imageView4);
        iw.setImageResource(R.mipmap.fingerprint_bad);
    }
}
```

Slika 12. Slika ekrana i isječak koda prijave.

Za navigiranje po aplikaciji koristi se meni koji se može otvoriti klikom na ikonu u gornjem lijevom kutu aplikacije. Izbornik nudi tri opcije, prva opcija je „Ključevi“ kojom otvaramo dio aplikacije koji služi za kreiranje zaporki odnosno upravljanje našim ključevima i privjescima. Druga opcija je „Konekcije“ unutar koje upravljamo konekcija koje imamo s raznim aplikacijama. Treća je opcija postavke a ona bi bila postavke aplikacije.



Slika 13. Snimke zaslona mobilne aplikacije

Za sam dizajn aplikacije koriste se neke postojeće a i neke posebno napravljene strukture dizajna. Posebno napravljene struktura se mogu pronaći unutar prikaza i dodavanja ključeva i konekcija. Njih smo posebno radili kako bi korisniku omogućili jedno jednostavno ali opet vizualno privlačno korisničko sučelje, iako to naravno korisnik mora i sam procijeniti.

Upravljanje bazom podataka se radi pomoću relativno novog sustava odnosno sučelja za upravljanje baza koje se zove ROOM. Room na omogućuje kreiranje baze tako da prvo kreiramo entite u obliku klasa te nakon toga on na temelju tih klasa kreira bazu. Također za unos i dohvaćanje podataka koristimo room. Ono što nam je bitno je da za korištenje samog room-a u određenom trenutku imamo samo jednu instancu klase pokrenutu pa smo tako morali kreirati jednu posebnu klasu koja će se moći instancirati samo jednom te na taj način zabranjujemo korištenje room-a u više od jednoj instanci.

```
public class DatabaseSingleton {

    private static DatabaseSingleton single_instance = null;
    public DB db;


    private  DatabaseSingleton(Context ctx){
        db = Room.databaseBuilder(ctx, DB.class, "MyPassDatabse").allowMainThreadQueries().build();
    }

    public static DatabaseSingleton getInstance(Context ctx){
        if(single_instance == null)
            single_instance = new DatabaseSingleton(ctx);

        return single_instance;
    }
}
```

Slika 14. Isječak koda / klasa za rad s bazom podataka.

7. Zaključak

U današnjem informatičkom svijetu potrebno je naći što bolje rješenje za zaštitu naših osobnih podataka među kojima su možda jedne od najvažnijih naše zaporke to jest naši korisnički računi. Veliki se problem javlja u tome što su korisnici najčešće manje informatički obrazovani pa se oni i ne znaju sami suprotstaviti onima koji ih žele iskoristiti. Trenutna rješenja su previše kompliciranа i ne primjenjuju najnovije i najpoželjnije tehnologije to jest tehnologije koje korisnici koriste na dnevnoj bazi a to ih odvaja od korištenja to jest od zaštite. Pri tome se javlja potreba za jednim jednostavnim a opet modernim upraviteljem zaporki što MyPass i je. Glavna premisa MyPassa je na modernosti i jednostavnosti što je i postignuto jednostavnim korisničkim sučeljem i jednostavnošću korištenja iako je sam proces u pozadini dosta kompliciran ali to ionako i ovako nije korisnikov posao. MyPass odvaja korisnika od kompliciranih stvari, on zapravo korisniku u neku ruku omogućava da ima svoje vlastite sigurnosne konzultante koji za njega paze na njegove račune i to na način da koriste najnovije tehnologije i na vrijeme krpaju sve moguće sigurnosne propuste.

8. Literatura

[1] Google developers „Androd keystor system“ [Dokumentacija] Dostupno: https://developer.android.com/training/articles/keystore [pristupano: 14.08.2018]

[2] „OSI model“ (bez dat.). u Wikipedia, the Free Encyclopedia, Dostupno: https://en.wikipedia.org/wiki/OSI_model [pristupano: 18.08.2018]

.

9. Popis slika

Slika 1. Isječak koda generiranja ključa] 3
Slika 2. Način zaštite korisnikovih podataka 4
Slika 3. Isječak koda / Zaključaj ključ s otiskom prsta. 4
Slika 4. Isječak koda / prikaz klase "Database"..... 5
Slika 5. OSI model internet protokola [1] 6
Slika 6. Primjer tijela PUT upita u json sintaksi. 7
Slika 7. Primjer odgovora na /api/Login post upit u json formatu. 8
Slika 8. Koraci registracije 10
Slika 9. Isječak koda / Registracija konekcije.....10
Slika 10. Isječak koda / Prijava na sustav 10
Slika 11. Isječak koda / Prihvaćanje websocket upita, spajanje na websocket 11
Slika 12. Slika ekrana i isječak koda prijave. 12
Slika 13. Snimke zaslona mobilne aplikacije.....13
Slika 14. Isječak koda / klasa za rad s bazom podataka. 13