DVOSTRUKA AUTENTIFIKACIJA POMOĆU PREPOZNAVANJA LICA

Zelić, Filip

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

DVOSTRUKA AUTENTIFIKACIJA POMOĆU PREPOZNAVANJA LICA

Filip Zelić

Zagreb, rujan 2019.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada.

U Zagrebu, datum.

Ime Prezime

Zahvaljujem mentoru Aleksanderu Radovanu, prof., na ukazanom povjerenju i stručnim savjetima. Također zahvaljujem svojoj obitelji (supruzi Kristini, majci Silviji, ocu Milanu, sestri Andrei) na danoj podršci.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Svakodnevni *cyber*-napadi i krađa podataka pokazuju koliko treba uložiti napora u dodatnu sigurnost informatičkih sustava. Zaštite postoje raznolike, od autentifikacije korisnika (provjera identifikacije korisnika uz predočenje akreditiranih podataka) do autorizacije (pravo pristupa određenom resursu) te same zaštite čuvanja podataka. Ovaj rad se fokusira na autentifikaciju korisnika, iznimno važan element sigurnosti budući da predstavlja prvi korak prijave korisnika u određeni informatički sustav. Dosadašnja praksa korištenja samo korisničke identifikacije i lozinke tijekom autentifikacije nije dovoljna [9], te se u zadnje vrijeme sve više primjenjuje tehnologija zvana autentifikacija u dva koraka. U sklopu rada razvijena je biblioteka za Django platformu pisana u Python programskom jeziku koja nudi gotovo rješenje dvostruke autentifikacije pomoću prepoznavanja lica. Sama biblioteka generira forme i sve elemente (korisničko ime, lozinku te funkcionalnost za slikanje lica uz pomoć *web*-kamere) potrebne za registraciju i prijavu u *web*-sustav te svu pozadinsku funkcionalnost za detekciju i identifikaciju lica na slici i autentifikaciju.

Uz samu biblioteku priloženi su dokumentacija korištenja, primjer implementacije biblioteke u novi *web*-aplikacijski sustav i prezentirani rezultati uspješnosti u detekciji i identifikaciji lica.

Ključne riječi: dvostruka autentifikacija pomoću prepoznavanja lica, računalna biblioteka, *web*, Python, Django

Daily cyber attacks and data breaches shows additional effort that is needed for extra security of information systems. There are different type of safeguard, from authentification (verifying identity of a user accredited data) to authorization (to determine privileges or access levels related to system resources) and data protection itself. This thesis will focus on authentification, extremely important element of cyber security since it is first element of sign-in to any information system. Current practice of using only username and password during authentication is insufficient, and the two-factor authentication technology is increasingly being applied.

As a part of this thesis, library is developed for Django framework written in Python language, it offers out-of-the-box solution for two-factor authentication with facial recognition. The library generates forms and all required elements (username, password and all functionality to take face image using web camera) for registration and sign-in to web system along with all backend functionality needed for face detection and identification and authentication to system.

Along with library itself, enclosed you can find the documentation, code examples of library implementation within new web application and some test results of face detection and identification accuracy.

Keywords: two-factor authentication with face recognition, computer library, web, Python, Django

Sadržaj

1. Uvod

Statistički podaci pokazuju sve veći broj *cyber*-napada i krađe podataka (engl. *data breach*) informatičkih sustava [9]. Također, trend korištenja istih korisničkih podataka za različite sustave ili jednostavnost probijanja lozinki uz sve jaču računalnu snagu sugerira da se trebaju uložiti novi napori u dodatnu zaštitu informatičkih sustava. Autentifikacija kao prvi korak sigurnosti predstavlja pravi izazov kako podići sigurnost na višu razinu. Kao odgovor na taj izazov razvila se tehnologija zvana dvostruka autentifikacija (engl. *two-factor authentication*) te postoji više izvedbi samog rješenja. Kao dodatni korak koristi se prepoznavanje otisaka prstiju, prepoznavanje lica te generiranje jednokratnih lozinki.

U *web*-svijetu se kao svojevrsni standard dodatne zaštite koristi generiranje jednokratnih lozinki. Korisnik pristupi sustavu, unese korisničko ime i lozinku te sam sustav šalje jednokratnu lozinku na elektroničku poštu ili SMS. No, što ako je pretinac elektroničke pošte ili mobilni uređaj kompromitiran? Kao rješenje tog problema bi bilo raspoznavanje fizičkih karakteristika korisnika poput lica.

Budući da je izrada takvog rješenja kompleksna te ne postoji gotovo rješenje na tržištu (za programski okvir Django), ishod ovog rada će biti računalna biblioteka koja nudi gotovo rješenje dvostruke autentifikacije pomoću prepoznavanja lica.

U drugom i trećem poglavlju općenito je opisan proces prepoznavanja lica te strojnog učenja, njihov razvoj kroz povijest te razlog korištenja. Kroz četvrto poglavlje detaljno je opisan proces prepoznavanja i pronalaska lica na slikama te njihova usporedba. Također su opisane osnovne značajke i koncepti korištenog programskog jezika, programskih okvira i biblioteka. Slijedi detaljan opis implementacije računalne biblioteke završnog rada, njena konfiguracija i korištenje te prikaz korištenja u produkcijskom okruženju uz horizontalno skaliranje.

2. Prepoznavanje lica

Sustav prepoznavanja lica je tehnologija koja može prepoznati i identificirati lice iz digitalne slike ili videoizvora. Postoje razni načini implementacije sustava, no najčešće se temelje na prepoznavanju lica na različitim slikama te njihovoj usporedbi.

Prepoznavanje lica putem računala predstavljalo je jedan od najvećih računalnih izazova par desetljeća [10]. Problem nije bio rješiv klasičnim setom instrukcija računalu već se razvila tehnologija zvana strojno učenje (engl. *machine learning*). U početku je primjena prepoznavanja lica korištena samo u računalnom obliku, u posljednje je vrijeme sve veći trend korištenja i u drugim oblicima poput mobilnih telefona, robotike i sl.[11]

Pouzdanost i preciznost prepoznavanja lica temelji se na kvaliteti same snimke lica, računalnoj snazi te korištenom modelu strojnog učenja. Dugo vremena problem je predstavljao manjak velikog skupa podataka te sami modeli nisu bili dobro trenirani [12]. No, u međuvremenu su se pojavili otvoreni skupovi podataka s par milijuna slika lica te modeli postaju sve precizniji [13].

3. Strojno učenje

U prošlosti računala su mogla raditi samo ono za što su bila programirana te su i za relativno malu funkcionalnost trebale biti napisane velike količine koda [14], odnosno instrukcija kako bi se pokrio određeni slučaj i dobio neki rezultat. S vremenom stručnjaci su prepoznali da je taj način vrlo ograničavajući te da bi idući korak u razvoju bio pokušaj oponašanja rada ljudskog mozga. Iz toga se rodila ideja o umjetnoj inteligenciji (engl. *artificial intelligence*), odnosno strojnog učenja na temelju prošlih iskustava, tj. podataka.

Strojno učenje je znanstveno proučavanje algoritama i statističkih modela koja računalni sustavi koriste za učinkovito obavljanje određenog zadataka bez eksplicitnih uputa, oslanjajući se na obrasce i zaključke [14]. Algoritmi bazirani na strojnom učenju grade matematičke modele uzoraka podataka (engl. *training data*) kako bi se predvidjela ili donijela odluka bez eksplicitnog programiranja. Primjene su jako široke, od filtriranja elektroničke pošte (engl. *e-mail*), prepoznavanja govora/slika, samovozećih automobila (engl. *self-driving car*), predikcije vremenskih uvjeta/financijske burze, dijagnosticiranja bolesti i sl.

Glavni razlozi sve veće popularnosti i korištenja tehnologije strojnog učenja su sve veća dostupnost ogromne količine podatka (engl. *big data)* i sve jača i jeftinija računalna snaga [14].

Rad se fokusira na prepoznavanje lica na slikama te će se u tu svrhu koristiti podvrstu strojnog učenja zvanu duboko učenje (engl. *deep learning*), a kako radi, opisano je niže.

4. Autentifikacija u dva koraka

4.1. Autentifikacija u dva koraka pomoću prepoznavanja lica

Autentifikacija u dva koraka pomoću prepoznavanja lica je vrlo izazovan problem. Korisnik prilikom prijave u sustav unosi svoje korisničko ime i lozinku te pomoću *web*-kamere uzima se trenutna slika i šalje na server na obradu. Prvi korak je pronalazak korisnika u bazi podataka, ako korisničko ime odgovara, dohvaćaju se dotični podaci. Potom upisana lozinka se provlači kroz algoritam jednosmjernog raspršivanja (engl. *one-way hashing algorithm*) te rezultat se uspoređuje s podatkom iz baze. Ovaj korak je izuzetno bitan za sigurnost ako dođe do krađe podataka iz baze, budući da su lozinke kriptirane te je napadaču otežano pristupiti izvornom podatku.

Idući korak je proces usporedbe lica. Ukratko, na slici se prvo pronalazi lice, kalkuliraju se mjerne jedinice za usporedbu te se uspoređuju s brojkama iz baze (detaljan proces opisan u nastavku).

Ako sva tri podatka odgovaraju (korisničko ime, lozinka i lice) korisnik se autentificira, kreira se sesija (engl. *session)* te se preusmjerava na ograničenu rutu.

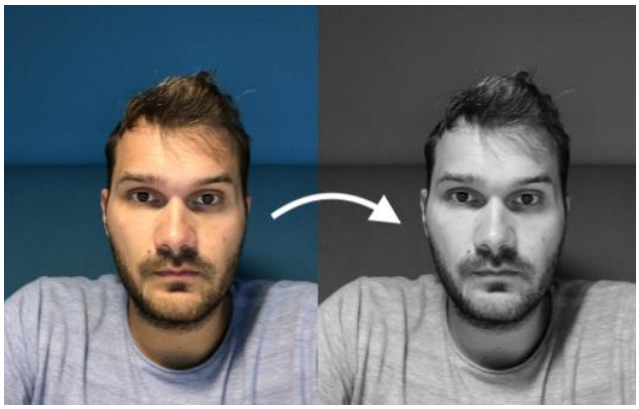
4.2. Proces prepoznavanja lica

S obzirom na to da je sam proces vrlo složen, rješenje problema se dijeli u četiri koraka. Tijekom svakog koraka koristi se drugi algoritam strojnog učenja te su objašnjene ključne ideje iza svakoga.

4.2.1. Prvi korak: pronalaženje lica na slici

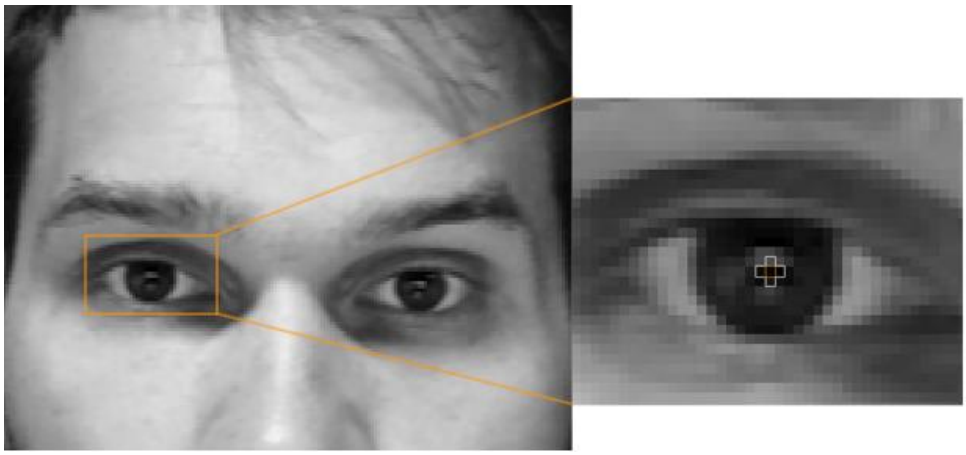
Prvi korak procesa je pronalazak jednog ili više lica na slici. Svejedno je radi li se o slici ili snimci, proces je isti. Tijekom ovog procesa cilj je pronaći dijelove slike gdje se nalaze lica.

Započinje pretvorba slike u crno-bijelu, budući da boja nije potrebna za pronalazak lica.



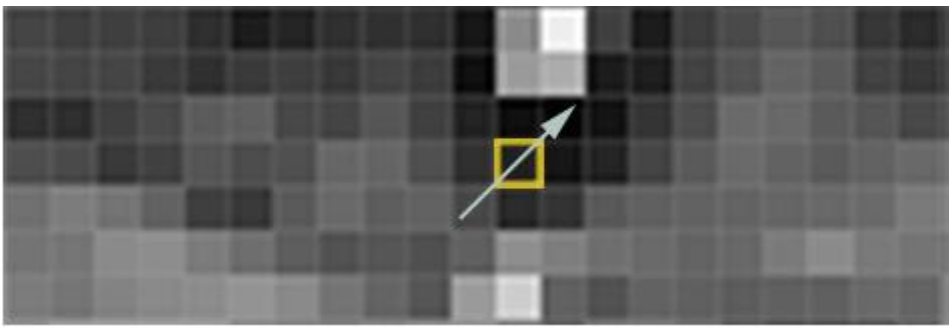
Slika 1. Transformacija slike u crno-bijelu

Nakon toga slijedi pregledavanje svakog pojedinačnog piksela na slici. Za svaki piksel gleda se na one koji ga direktno okružuju.



Slika 2. Pregled piksela i usporedba s okružujućim

Cilj je shvatiti koliko je trenutni piksel taman u odnosu na piksele koji ga okružuju, zatim se crta strelica koja pokazuje u kojem smjeru slika postaje tamnija.



Slika 3. Crtanje strelica u smjeru gdje slika postaje tamnija

Postupak se ponavlja za svaki pojedinačan piksel na slici dok svaki nije zamijenjen strelicom. Strelice se nazivaju gradijenti i pokazuju protok od svijetle do tamne strane na cijeloj slici. Zatim se grupira piksele u kvadrate veličine 16 x 16 te će se taj kvadrat na slici zamijeniti smjerom strelica koje su bile najjače. Kranji rezultat pretvara originalnu sliku u vrlo jednostavnu reprezentaciju koja otkriva osnovnu strukturu lica – metoda se naziva HOG.

Slika 4. HOG reprezentacija lica

Za pronalazak lica na HOG slici se koristi istrenirani uzorak lica na milijunima slika, uzorak uspoređujemo svakim dijelom naše slike dok ne odgovara.

4.2.2. Drugi korak: pozicioniranje i projiciranje lica

Sam korak prepoznavanja lica je gotov, no lice također treba identificirati odnosno usporediti sliku *web*-kamere postojećom slikom/uzorkom. Izolacijom slike lica, često se javlja problem da pozicija glave odnosno smjer ne odgovara.

Slika 5. Ljudi vrlo lako mogu prepoznati da se radi o istoj osobi, no računala ih vide kao dvije različite

Rješenje tog problema je da uvijek oči, nos i usta budu maksimalno centrirani te u većini slučajeva sliku moramo iskriviti. Koristit će se algoritam naziva *face landmark estimation*, sam algoritam na temelju strojnog učenja pronalazi 68 specifičnih točaka koje postoje na svakom ljudskom licu – vrhu brade, vanjskog ruba oka, unutarnjeg ruba svake obrve i sl.

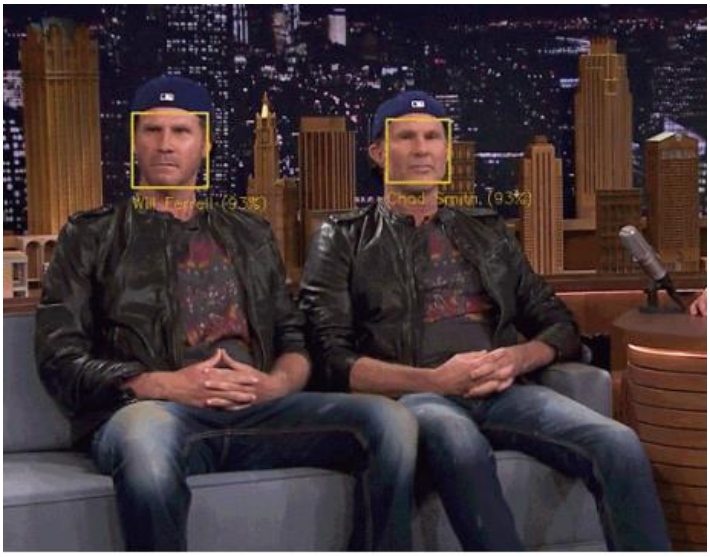
Slika 6. 68 specifičnih točaka na licu

Znajući lokaciju očiju, usta i sl., koriste se osnovne slikovne transformacije poput rotacije, skaliranja i pomicanja kako bismo maksimalno centrirali sliku lica.

4.2.3. Treći korak: *encoding* lica

Za usporedbu lica s postojećim, treba za svako lice izvući nekoliko osnovnih mjernih parametara. Uspostavilo se da mjere poput boje očiju, širine i dužine nosa/usta i sl. nemaju smisla računalu kada uspoređuje piksele. Metoda dubokog učenja (engl. *deep learning*)

može daleko točnije utvrditi koji su dijelovi lica važni za mjerenje. Kao dokaz tome na sljedećoj slici je prikazano kako računalo s lakoćom i preciznošću većom od 90 % može utvrditi razliku između dviju gotovo identičnih osoba.



Slika 7. Računalna detekcija dviju različitih osoba

Koristeći istrenirani model za svaku sliku lica se dobiva 128 mjernih jedinica koje se potom koriste za usporedbu. Model dubokog učenja koji se koristi ima na skupu podataka od 3 milijuna slika točnost od 99,38 % identifikacije lica.

4.2.4. Četvrti korak: usporedba mjernih jedinica

Posljednji korak je usporedba generiranih mjernih jedinica. Može se koristiti bilo koji osnovni algoritam za klasifikaciju strojnog učenja. Sve što treba učiniti je trenirati klasifikator koji može uzeti mjerenje sa slike *web*-kamere te usporediti s već postojećom slikom.

4.3. Koncepti biblioteke django-two-factor-face-auth

4.3.1. Najvažnije značajke programskog jezika Python

Python je programski jezik opće namjene, interpretiran i visoke razine [15]. Dopušta korištenje nekoliko stilova programiranja: objektno orijentirano, strukturno i aspektno orijentirano.

Ima dinamički sustav i automatsko upravljanje memorijom te ima veliku i sveobuhvatnu biblioteku.

Python koristi uvlačenje kao metodu razlikovanja programskih blokova, tj. ne koristi vitičaste zagrade ili ključne riječi kao većina programskih jezika. Povećanje uvlačenja znači da dolazi novi, ugniježđeni blok, dok smanjenje označava kraj trenutnog bloka.

U Pythonove ključne riječi spadaju: `if`, `for`, `while`, `try`, `class`, `def`, `assert`, `import`.

4.3.2. Osnovni koncepti rada programskog okvira Django

Programski okvir Django je temeljen na Python programskom jeziku i otvorenog je koda (engl. *open-source)* [16]. Prati arhitektonski uzorak MVT te ga održava zajednica Django Software Foundation.

Primarni cilj Djanga je olakšati stvaranje složenih *web*-stranica koje se temelje na bazama podataka. Programski okvir naglašava mogućnost ponovne upotrebe modula, manje koda, brzi razvoj i prati načelo da nema ponavljanja. Django također pruža opcionalno administrativno kreiranje, čitanje, ažuriranje i brisanje sučelja koje se generira dinamički kroz introspekciju i konfigurirano putem administracijskog modela.

4.3.3. Računalne biblioteke Dlib i face_recognition

Dlib je *cross-platform* računalna biblioteka opće namjene napisana u programskom jeziku C++ [17]. Sadrži komponente za rad sa strojnim učenjem, procesuiranjem slika, podatkovnog rudarenja, linearne algebre, podatkovnih struktura, tekstualnih procesuiranja i sl. Često se upotrebljava u akademskoj zajednici i industriji u širokom rasponu domena, od robotike, mobilnih uređaja i računalnih okruženja visokih performansi.

Face_recognition je računalna biblioteka bazirana na dlib-u napisana u Python programskom jeziku [18]. Biblioteka omogućava olakšan rad i manipulaciju slika, funkcionalnosti za prepoznavanje i identifikaciju lica.

4.3.4. Računalna biblioteka django.contrib.auth

Biblioteka django-two-factor-face-auth se oslanja i proširuje osnovnu biblioteku django.contrib.auth koja omogućuje autentifikaciju i autorizaciju. Biblioteka je bazirana oko

objekta korisnik (engl. *User*) koji obično predstavlja osobu koja komunicira s *web*-stranicom. Koristi se za omogućavanje stvari kao što su ograničavanje pristupa, registracija korisničkih profila, povezivanje sadržaja s kreatorom i slično. Primarni atributi objekta korisnik su:

- korisničko ime
- lozinka
- *e-mail* adresa
- ime
- prezime.

4.4. Implementacija biblioteke django-two-factor-face-auth

Glavna značajka biblioteke django-two-factor-face-auth su jednostavnost i mogućnost integracije u bilo koje novo ili postojeće rješenje bazirano na Django tehnologiji. Vodeći se tim značajkama biblioteka je dizajnirana na način da se lako može proširiti ili dio funkcionalnosti *overrideati*.

Biblioteka prati preporučenu strukturu po standardima Django programskog okvira [19], pisana je i testirana za Linux i Mac operacijske sustave, no ne isključuje niti Windows.

Budući da je biblioteka pisana u Python jeziku i ovisi o drugim računalnim bibliotekama, minimalni zahtjevi su:

```
'Python>=3.5',
'Django>=2.0',
'face_recognition>=1.2.3',
'dlib>=19.7',
```

Kod 1. Minimalni zahtjevi biblioteke django-two-factore-face-auth

Biblioteka je licencirana pod MIT licencom, koja dopušta besplatno korištenje koda i popratne dokumentacije bez ograničenja.

4.4.1. Struktura biblioteke django-two-factor-face-auth

Struktura biblioteke prati najbolje prakse i pravila Django programskog okvira [20]. Vodeći se tim pravilima omogućeno je njegovo lako proširenje te premošćivanje.

```
[django-two-factor-face-auth]
|---migrations/
|   |--0001_initial.py
|   |__init__.py
|---static/
|   |---authentication/
|   |   |---js/
|   |   |   |---login.js
|   |   |   |---register.js
|---templates/
|   |---django_two_factor_face_auth/
|   |   |---login.html
|   |   |---register.html
|---__init__.py
|---authenticate.py
|---forms.py
|---models.py
|---urls.py
|---utils.py
|---views.py
|---wsgi.py
```

Kod 2. Struktura biblioteke

4.4.2. Model biblioteke

Datoteka `models.py` predstavlja model biblioteke. Sadrži osnovna polja, svojstva i ponašanja podataka s kojom biblioteka radi. Općenito, svaki model se preslikava u jednu tablicu baze podataka.

```
class UserFaceImage(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    image = models.ImageField(upload_to=content_file_name, blank=False)
```

Kod 3. Model UserFaceImage

U primjeru je definirana klasa `UserFaceImage` s dva svojstva/atributa `user` i `image`. Budući da biblioteka proširuje modul `django.contrib.auth` svojstvo `user` je svojevrsna poveznica između modela korisnik i definiranog modela slike lica. Svojstvo `image` definira putanju do datoteke slike lica. Same datoteke slika spremljene su na tvrdi disk unutar direktorija `content/*`.

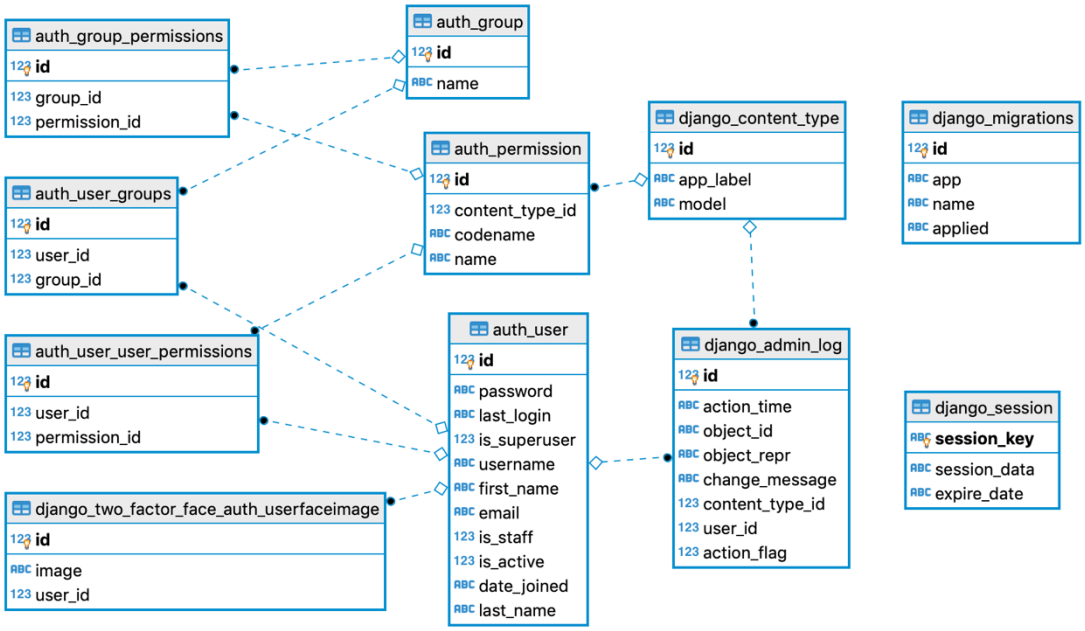
4.4.3. Migracija baze podataka

Migracija baze podataka u Django programskom okviru je način propagiranja promjene modela u shemu baze podataka. Za izradu migracije koristi se komanda `python manage.py makemigrations` koja kreira datoteku s promjenama (unutar direktorija `migrations/`). Kako bi se izvršila sama migracija kreiranih datoteka, koristi se komanda `python manage.py migrate` (detaljnije opisano u nastavku). Komanda će dotične migracijske datoteke pretvoriti u upite baze podataka te izvršiti na samoj bazi. Datoteke se izvršavaju slijedno po rednom broju.

```
SELECT column1, column2 FROM table1, table2 WHERE
column2='value';
```

Kod 4. Primjer upita na bazu podataka

Primjer upita na bazu kreiran na temelju `UserFaceImage` modela. Tijekom migracije baze podataka tablice s nazivima `auth_group`, `auth_group_permissions`, `auth_permission`, `auth_user`, `auth_user_groups`, `auth_user_user_permissions`, `auth_user_face_image`, `django_admin_log`, `django_content_type`, `django_session` te `django_site` će biti kreirane.



Slika 8 - ER model baze podataka

4.4.4. *URL* putanje

U datoteci `urls.py` su definirane sve putanje s kojima biblioteka raspolaže, odnosno sadrži listu *URL* izraza te ih mapira u *Python* funkcije.

```
urlpatterns = [
            path('accounts/register/', views.register, name='register'),
            path('accounts/login/', views.face_login, name='login'),
            path('accounts/', include('django.contrib.auth.urls')),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Kod 5. URL putanje

Biblioteka django-two-factor-face-auth ima definirane dvije putanje, jednu za registraciju novog korisničkog računa sa slikom lica te drugu putanju za prijavu korisnika u sustav s pomoću korisničkih podataka i prepoznavanje slike lica. Također, biblioteka proširuje modul `django.contrib.auth` koji omogućuje provjeru autentičnosti i autorizacije te gotovih značajki poput prijave/odjave korisnika iz sustava, kreiranja sesije, promjene i ponovnog postavljanja lozinke. Biblioteka pomoću funkcije `include('django.contrib.auth.urls')` uključuje sljedeće putanje:

```
accounts/logout/
accounts/password_change/
accounts/password_change/done/
accounts/password_reset/
accounts/password_reset/done/
accounts/reset/<uidb64>/<token>/
accounts/reset/done/
```

4.4.5. Pomoćni modul `utils.py`

Datoteka `utils.py` sadrži pomoćne funkcije biblioteke django-two-factor-face-auth. Većina funkcija unutar modula dizajnirana je za unutarnju upotrebu. Trenutno definirane funkcije koriste se za manipulaciju slika poput dekodiranja Base64 formata. Kodiranje u Base64 formatu obično se koristi kada postoji potreba za kodiranjem binarnih podataka koje je potrebno pohraniti i prenijeti preko medija koji su dizajnirana da rade s tekstualnim podacima. Time se osigurava da podaci ostaju netaknuti (bez izmjena) tijekom prijenosa podataka.

4.4.6. Implementacija prednjeg sučelja (*frontend*)

Logika funkcionalnosti prednjeg sučelja pisana je u programskom jeziku JavaScript te je podijeljena u dvije datoteke `login.js` (logički kod vezan za prijavu) i `register.js` (logički kod vezan za registraciju). Kako bi se zadovoljila kompatibilnost i funkcionalnost, sami predlošci moraju sadržavati HTML elemente poput `form`, `input`, `button`, `video` i sl. (detaljno opisano u sekciji 4.6.1. Prednje sučelje (*frontend*) – predlošci dizajna).

```
const constraints = {
  video: true
};
```

Kod 6. Najava korištenja *web*-kamere

U prvoj liniji koda nalazi se najava internet-pregledniku *web*-stranice za korištenje *web*-kamere. U mnogim izvedbama internet-preglednika korisnik je obaviješten o namjeri te mora na izričit način dopustiti korištenje.

```
const captureVideoButton = function() {
    document.querySelector('#video-button');
    const registerButton = document.querySelector('#register-
                                                    button');
    const video = document.querySelector('#screenshot-
                                            video');
    const image_hidden = document.querySelector('#id_image');
};
```

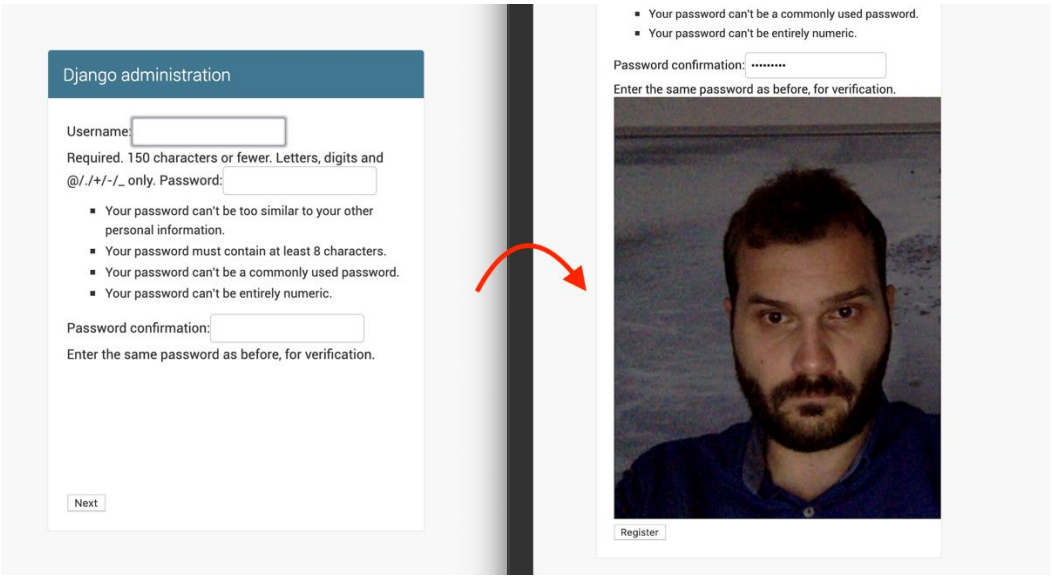
Kod 7. Dohvat HTML elemenata

Idući odjeljak koda dohvaća kreirane elemente predloška kako bi mogao manipulirati njima.

```
captureVideoButton.onclick = function() {
    captureVideoButton.setAttribute('style','display:
                                        none;');
    registerButton.removeAttribute("style");
    navigator.mediaDevices.getUserMedia(constraints).
    then(handleSuccess).catch(handleError);
};
```

Kod 8. Dohvat slike s *web*-kamere

Tijekom registracije unosi se korisničko ime (slika 9), lozinka te potom odabirom opcije nastavi se pozivna funkcija `captureVideoButton.onclick`. Ona omogućuje otvaranje *overlaya* u kojem se pojavljuje video s *web*-kamere.



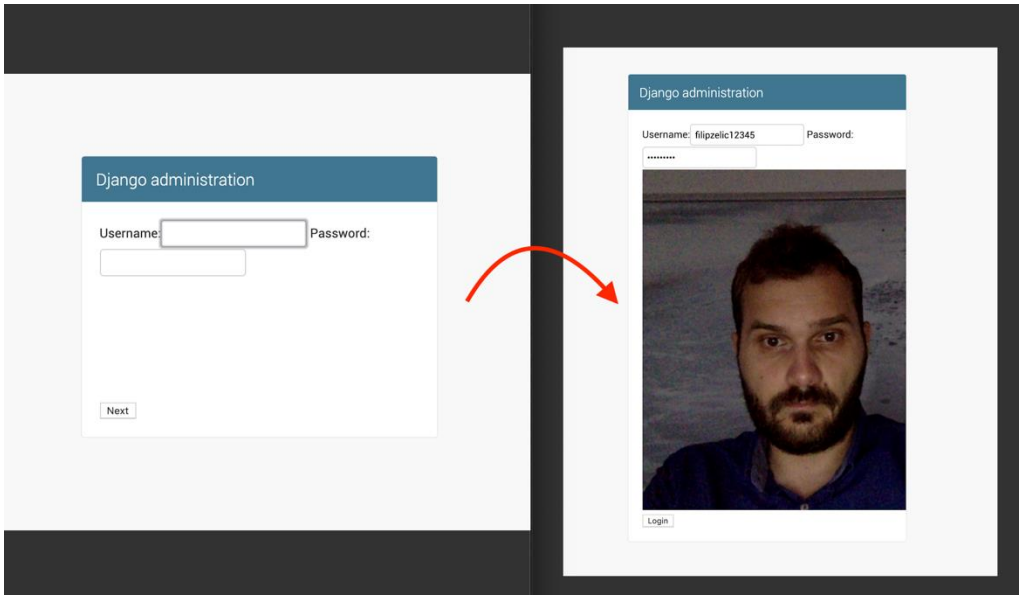
Slika 9 - Registracija na sustav

```
registerButton.onclick = video.onclick = function() {
    var canvas = document.createElement("canvas");
    canvas.getContext('2d').drawImage(video, 0, 0,
                                      canvas.width, canvas.height);
    image_hidden.value = canvas.toDataURL();
    document.forms["face-register-form"].submit();
};
```

Kod 9. Slanje ispunjene forme na poslužitelj

Ako je korisnik zadovoljan viđenim, odabirom opcije *nastavak* poziva se funkcija `registerButton` koja dohvaća element `canvas`, iscrtava sliku uhvaćenu *web*-kamerom te pozivom `canvas.toDataURL()` funkcija vraća podatkovnu URI reprezentaciju slike. Zadnji poziv funkcije `submit()` svi podaci (korisničko ime, lozinka te slika lica) prenose se na poslužitelj (engl. *server*).

Sama prijava na sustav (slika 9) je vrlo slična kao registracija, no informacije tokom unosa (korisničko ime, lozinka te slika lica) se prenose na poslužitelj, uspoređuju se s postojećima te ukoliko odgovaraju korisnik se autentificira.



Slika 10 - Prijava na sustav

4.4.7. Ulazna točka biblioteke - `views.py`

Modul `views.py` služi kao ulazna točka biblioteke. Sadrži funkcije koje primaju *web*-zahtjeve i vraćaju *web*-odgovore. *Web*-zahtjevi mogu biti bilo koje vrste podataka koje poslužitelj (engl. *server*) ovjerava i koristim tijekom izvršavanja te odgovori tipa HTML,

greške, slike, preusmjeravanja i sl. Definirane su dvije funkcije (registracija i prijava) opisane u nastavku:

```
def register(request):
      if  request.method == 'POST':
            form = UserCreationForm(request.POST, request.FILES)

            if form.is_valid():
                  form.save()
                  username = form.cleaned_data['username']
                  password = form.cleaned_data['password2']
                  user = authenticate(username=username,
            password=password)
                  login(request, user)
                        return redirect(settings.LOGIN_REDIRECT_URL)
                else:
                    form = UserCreationForm()
      context = {'form': form}
      return render(request, 'django_two_factor_face_auth/register.html',
context)
```

Kod 10. Ulazni modul biblioteke

Funkcija `register` kao ulazni parametar prima objekt `request` koji sadrži stanje samog zahtjeva. Odnosno, kada korisnik tijekom registracije unese željeno korisničko ime, lozinku te putem *web*-kamere unese sliku lica putem *`HTTP POST`* zahtjeva, informacije se prenose do samog poslužitelja. Jedna od prvih stvari koje poslužitelj pokreće je poziv prema Django programskom okviru, te zahtjev slijedom događaja dolazi do modula `views.py` i funkcije `register`. Veze između putanja i funkcija definirane su u modulu `urls.py`. U samoj funkciji se ovjeri jesu li poslani podaci važeći te da sadrže CSRF token. Ako podaci nisu važeći, programski okvir Django automatski vraća odgovor s porukama o greškama. S važećim podacima se pozivom funkcije `form.save()` podaci spremaju u bazu podataka te slika na disk (opisano u nastavku). Kada je sve uspješno izvršeno, korisnik se preusmjerava na putanju definiranu u postavkama.

```
def face_login(request):
      if request.method == 'POST':
            form = AuthenticationForm(request, request.POST)

            if form.is_valid():
                  username = form.cleaned_data['username']
```

```
                password = form.cleaned_data['password']
                face_image = prepare_image(form.cleaned_data['image'])

                face_id = FaceIdAuthBackend()
                user = face_id.authenticate(username=username,
                                password=password, face_id=face_image)
                if user is not None:
                      login(request, user)
                      return redirect(settings.LOGIN_REDIRECT_URL)
                else:
                      form.add_error(None, "Username, password or face
          id didn't match.")
              else:
                  form = AuthenticationForm()

context = {'form': form}
return render(request, 'django_two_factor_face_auth/login.html', context)
```

Kod 11. Ulazna funkcija face_login

Funkcija `face_login` koristi se za samu prijavu registriranih korisnika u sustav. Kao i tijekom registracije, korisnik unosi korisničko ime, lozinku te sliku lica putem *web*-kamere. Podaci se ovjere te ako su važeći pozivom na funkciju `face_id.authenticate` prosljeđuje se korisničko ime, lozinka te slika lica. Ako korisnik postoji u bazi podataka, lozinka odgovara te se slika lica podudara sa slikom lica dodatno tijekom registracije, korisnik je autentičan. Pozivom na funkciju `login(request, user)`, Django programski okvir koristi objekt `request` kako bi kreirao sesiju (engl. *session*) koja sadrži korisnički identifikacijski broj, metodu autentifikacije, datum i vrijeme isteka sesije i ostale podatke potrebne za provjeru autentifikacije. Time je omogućeno da se svakim novim zahtjevom korisnik identificira te olakša korištenje.

4.4.8. Autentifikacijske forme

Na isti način kako modeli opisuju logičku strukturu podataka nekog objekta i njihovo ponašanje, klasa `forms.py` opisuje kako obrazac izgleda, koja polja sadrži te kako djeluje. Polja obrasca imaju direktnu poveznicu na *HTML* kod, odnosno elemente `form`, `input` i sl. Polja obrasca su svojevrsne klase, upravljaju podacima obrasca, izvršavaju provjeru valjanosti kada je obrazac poslan.

```
class UserCreationForm(UserCreationForm):
```

```
image = forms.CharField(widget=forms.HiddenInput())

class Meta:
      model = User
      fields = ("username", "password1", "password2")

def save(self, commit=True):
      if not commit:
            raise NotImplementedError("Can't create User and
      UserFaceImage without database save")
      user = super(UserCreationForm, self).save(commit=True)
      image = base64_file(self.data['image'])
      face_image = UserFaceImage(user=user, image=image)
      face_image.save()
      return user
```

Kod 12. Forma za kreiranje korisnika

Klasa obrasca za kreiranje korisničkog računa proširuje Django obrazac kreiranja korisničkog računa u kojem su definirana osnovna polja korisničkog imena, lozinke, potvrde lozinke i sl. U proširenju se dodaje polje za sliku lica. Također, u klasi je definirano kako se obrazac ponaša u funkciji spremanja. Ako su svi podaci valjani, prvo se spremaju podaci korisničkog imena i lozinke u bazu podataka. Lozinka se prije spremanja promijeni, odnosno prikrije koristeći *hash* algoritam. *Hashing* omogućuje transformaciju lozinke u jednom smjeru, pretvarajući je u dugi niz raspršenih znakova. Iz sigurnosnih razloga lozinka se ne sprema u izvornom obliku ako dođe do neovlaštenog pristupa bazi podataka. Zatim se BASE64 dekodiranjem (korištenjem pomoćnog modula `utils.py`) slika pretvara nazad u binaran format te pomoću funkcije `face_image.save()` sprema na disk poslužitelja. Putanja slike se također sprema u bazu podataka za buduće korištenje.

```
class AuthenticationForm(AuthenticationForm):
          image = forms.CharField(widget=forms.HiddenInput())
```

Kod 13. Forma za autentifikaciju

Kao i u prethodnom primjeru, klasa obrasca za autentifikaciju korisnika proširuje Django obrazac te dodaje polje za sliku lica. Tijekom prijave korisnik unosi uz korisničko ime i lozinku sliku lica pomoću *web*-kamere.

4.4.9. Implementacija autentifikacije pomoću prepoznavanja lica

Modul authenticate.py sadrži svu logiku potrebnu za autentifikaciju pomoću prepoznavanja lica. U nastavku opisana funkcija se može pozvati iz bilo kojeg dijela koda kako bi se provjerila sama autentičnost korisnika, no dodatno treba paziti na spremanje stanja (engl. *session*) kako bi korisnik nakon provjere valjanosti nesmetano nastavio koristiti sustav.

```
def authenticate(self, username=None, password=None, face_id=None,
**kwargs):
      try:
            user = User.objects.get(username=username)
            if user.check_password(password) and
            self.check_face_id(face_id=user.userfaceimage.image,

                                    uploaded_face_id=face_id):
                  return user
      except User.DoesNotExist:
            User().set_password(password)
```

Kod 14. Glavna funkcija autentifikacije

Definirana funkcija `authenticate` prihvaća tri glavna parametra: korisničko ime, lozinku te sliku lica. U prvom pozivu se provjerava postoji li korisnik pomoću upita na bazu podataka (tablica `auth_user`) definiran pomoću korisničkog imena. Ako podatak postoji, provjerava se valjanost lozinke. Sama lozinka se promijeni u jednom smjeru (engl. *hashing*) istim *hash* algoritmom kojim je izvorno kreirana te se dodaje *salt* (proizvoljno generiran podatak zbog dodatne sigurnosti). Ako *hash* lozinka odgovara onoj spremljenoj u bazi podataka, se nastavlja s provjerom autentičnosti slike lica.

```
<algoritam>$<iteracija>$<salt>$<hash-lozinke>
```

Kod 15. Struktura raspršene lozinke

Primjer spremljene lozinke u bazi podataka: podaci su odvojeni znakom „$“ te se sastoje od algoritma raspršivanja, broja iteracije algoritma, salta i rezultata *hash* lozinke. Django programski okvir po zadanim postavkama koristi algoritam PBKDF2, no prema potrebi može biti zamijenjen u datoteci `settings.py`.

```
def check_face_id(self, face_id=None, uploaded_face_id=None):
      confirmed_image = face_recognition.load_image_file(face_id)
      uploaded_image = face_recognition.load_image_file(uploaded_face_id)
```

```
face_locations = face_recognition.face_locations(uploaded_image)
      if len(face_locations) == 0:
              return False

confirmed_encoding =     face_recognition.face_encodings…
unknown_encoding = face_recognition.face_encodings…

results = face_recognition.compare_faces([confirmed_encoding],
                                          unknown_encoding)

if results[0] == True:
      return True

return False
```

Kod 16. Provjera valjanosti slike lica

Sama provjera valjanosti slike lice odvija se u funkciji `check_face_id`. Prvi korak je učitavanje slika lica u biblioteku face_recognition pomoću poziva funkcije `face_recognition.load_image_file`. Potom funkcija `face_recognition.face_locations` pronalazi i vraća polje koordinata položaja lica na slici. Vrlo je važno da se provjera obavi u ranoj fazi izvršavanja koda kako bi se oslobodilo korištenje računalnih resursa ako, slika preuzeta s *web*-kamere ne sadrži lice. Kako bi se potvrdilo da su lica na slikama identična, treba za svako lice izvući osnovnih par mjernih parametara, proces naziva *encoding*. Koristeći istrenirani model, za svaku sliku lica dobit će se 128 mjernih jedinica koji se potom koriste za usporedbu. Ako lica odgovaraju, vraća se vrijednost nazad.

4.5. Instalacija i konfiguracija računalne biblioteke

Instalacija i korištenje biblioteke django-two-factor-face-auth je jednostavna. Prvi korak je automatska instalacija biblioteke na sam operacijski sustav pomoću PIP-a. PIP je sustav za upravljanje koji se koristi za instaliranje programskih paketa napisanih u Pythonu.

```
pip install django-two-factor-face-auth
```

Kod 17. Instalacija računalne biblioteke django-two-fator-face-auth

Potom je potrebno dodati samu referencu na računalnu biblioteku unutar datoteke `settings.py`. Time Django registrira novi biblioteku u *web*-aplikaciju/sustav.

```
INSTALLED_APPS = [
    ...
    'django_two_factor_face_auth',
]
```

Kod 18. Dodavanje referenci

Nakon toga je potrebno registrirati nove rute koje računalna biblioteka pruža. Rute koje će biti automatski generirane su: `accounts/register`, `accounts/login`, `accounts/logout`.

```
path('/', include('django_two_factor_face_auth.urls')),
```

Kod 19. Registracija novih ruta

Pokrenuti komandu za kreiranje modela i migriranje baze podataka.

```
python manage.py migrate
```

Kod 20. Migracija baze podataka

4.5.1. Opcionalna konfiguracija

Biblioteka django-two-factor-face-auth omogućuje dodatnu opcionalnu konfiguraciju:

- STATIC_URL – URL za dohvat statičnih datoteka (*javascript*, *css*)
- STATIC_ROOT – mjesto na disku poslužitelja gdje su spremljene statične datoteke
- LOGIN_REDIRECT_URL – URL za preusmjeravanje korisnika nakon uspješne autentifikacije na sustav
- IMAGE_PATH – mjesto na disku poslužitelja gdje se spremaju slike lica korisnika.

4.6. Korištenje računalne biblioteke

4.6.1. Prednje sučelje (engl. *frontend*) – predlošci dizajna

Sama biblioteka generira dva predloška stranice za prijavu i registraciju korisnika. Strogo je preporučeno kreirati vlastite predloške unutar novog *web*-sustava. Nova struktura direktorija mora biti kreirana na sljedeći način:

```
templates/django_two_factor_face_auth/login.html
                                        /register.html
```

Kod 21. Struktura dizajn predložaka

Predložak za prijavu (login.html) mora sadržavati sljedeće elemente:

```
form id='face-login-form' method="post" action="{% url
'login' %}" enctype="multipart/form-data">
    {% csrf_token %}

    {{ form }}

    <input type="hidden" name="next" value="{{ next }}">
</form>
<button id="video-button">Next</button>
<video id="screenshot-video" autoplay></video>
<button id="login-button" style="display:
none;">Login</button>
```

Kod 22. Predložak za prijavu korisnika

Predložak za registraciju (register.html) mora sadržavati sljedeće elemente:

```
<form id='face-register-form' method="post" action="{% url
'register' %}" enctype="multipart/form-data">
    {% csrf_token %}

    {{ form }}

</form>
<button id="video-button">Next</button>
<video id="screenshot-video" autoplay></video>
<button id="register-button" style="display:
none;">Register</button>
```

Kod 23. Predložak za registraciju korisnika

Napomena, identifikacijski nazivi se ne smiju mijenjati. Sav potreban kod za funkcionalnost prednje *web*-stranice nalazi se unutar direktorija `django_two_factor_face_auth/static/authentication/js/` te ga se može proširiti po potrebi.

4.6.2. Zadnje sučelje (engl. *backend*)

Biblioteka sadržava svu potrebnu funkcionalnost za registraciju i prijavu korisnika pomoću prepoznavanja lica. Biblioteka koristi postojeću bazu podataka koju korisnik definira u postavkama te kreira nove tablice `auth_user, auth_group, auth_permission, auth_user_groups, auth_group_permissions, auth_user_images`.

Tijekom registracije novih korisnika, biblioteka automatski kreira direktorij sa slikama u sljedećem formatu (slike nisu javno dostupne):

```
'content'/{username}/{timestamp + png/jpg/jpeg}
```

Kod 24. Direktorij sa slikama

4.6.3. Produkcijsko okruženje

Cijeli proces prepoznavanja i usporedba lica je vrlo složen i zahtijeva korištenje mnogo računalnih resursa. Budući da u stvarnom svijetu dolazi do višestrukih paralelnih zahtjeva na poslužitelj (istovremenih prijava ili registracija na sustav), dolazi do izazova kako podržati tolike zahtjeve i omogućiti nesmetan rad. Kao odgovor na taj izazov, u sklopu rada je izrađeno virtualno rješenje koje sadrži samo nužne komponente i računalne biblioteke bazirano na Alpine Linuxu. Operacijski sustav Alpine je dizajniran za sigurnost, jednostavnost i učinkovitost resursa.

Kako bi se omogućilo da rješenje radi neovisno o platformi (Linux, Windows, Mac OS i sl.) bazirano je na tehnologiji zvanoj Docker [21]. Docker koristi virtualizaciju na nivou operativnog sustava za isporuku softvera u paketima zvanim spremnik (engl. *container)* [21]. Spremnici su izolirani jedni od drugih i sadrže sav potreban softver, računalne biblioteke i konfiguracijske datoteke. Sve spremnike pokreće jedan *kernel* operacijskog sustava i stoga su lakši, brži i optimiziraniji u odnosu na klasične virtualne sustave. Budući da je cijelo rješenje zapakirano u spremniku, omogućuje vrlo lako horizontalno skaliranje.

```
FROM python:3.6

RUN apt-get -y update
RUN apt-get install -y --fix-missing \
    build-essential \
    cmake \
    gfortran \
    git \
```

```
    wget \
    curl \
    graphicsmagick \
    libgraphicsmagick1-dev \
    libatlas-dev \
    libavcodec-dev \
    libavformat-dev \
    libgtk2.0-dev \
    libjpeg-dev \
    liblapack-dev \
    libswscale-dev \
    pkg-config \
    python3-dev \
    python3-numpy \
    software-properties-common \
    zip \
    && apt-get clean && rm -rf /tmp/* /var/tmp/*
```

Kod 25. Popis biblioteka unutar Dockerfilea

Datoteka Dockerfile sadrži svojevrstan recept izgradnje. U prvoj liniji `FROM python:3.6` je definirana osnovna slika koja najčešće sadrži upute koji osnovni operacijski sustav i verzija se koristi. U nastavku su definirane sve računalne biblioteke koje moraju biti instalirane kako bi se zadovoljili zahtjevi računalne biblioteke django-two-factor-face-auth.

```
RUN cd ~ && \
    mkdir -p dlib && \
    git clone -b 'v19.9' --single-branch
https://github.com/davisking/dlib.git dlib/ && \
    cd  dlib/ && \
    python3 setup.py install --yes USE_AVX_INSTRUCTIONS
```

Kod 26. Kompajliranje dlib biblioteke

U nastavku je definirano preuzimanje biblioteke dlib te njena konfiguracija i kompiliranje unutar samog spremnika. Komanda `RUN` se izvršava samo kada je spremnik pokrenut u virtualnom okruženju.

```
RUN pip install django pillow face_recognition
```

Kod 27. Konfiguracija django i face_recognition biblioteke

U nastavku će se tijekom izvršavanja instalirati i sama računalna biblioteka django zajedno s bibliotekom face_recognition.

```
docker build -t django-two-factor-face-auth:v1 .
```

Kod 28. Izgradnja Docker slike

Naredba `docker build -t` gradi Docker sliku (engl. *image*) iz datoteke Dockerfile. Kontekst gradnje predstavlja skup datoteka koje se nalaze u trenutnom direktoriju.

```
docker run --rm -it -p 8000:8000 -v {LOKALNA-
PUTANJA}:{PUTANJA-U-VIRTUALNOM-POSLUŽITELJU} django-two-
factor-face-auth:v1
```

Kod 29. Pokretanje Docker spremnika

Naredba `docker run` pokreće virtualni sustav u izoliranom spremniku s vlastitim datotečnim sustavom, vlastitom mrežom te vlastitim stablom procesa odvojeno od glavnog računala (engl. *host*).

Zaključak

Dvostruka autentifikacija postaje svojevrsni standard u komercijalnoj primjeni velikih sustava najčešće implementiranih od strane većih internacionalnih kompanija. Zbog dugotrajnog razvoja i većeg troška izrade sličnih modula češće se odlučuje na jednostavnija gotova rješenja bez dvostruke autentifikacije. Rad se temelji na treniranom modelu uz točnost od 99,38 % prepoznavanja lica iz skupa podataka (engl. *data set*) od 3 milijuna slika što daje veliku pouzdanost ovom rješenju.

Rad prati preporučene prakse i strukturu baziranu na standardima Python programskog jezika i Django programskog okvira. Kao takva, biblioteka je dizajnirana na način da se funkcionalnost lako može proširiti i prilagoditi u svako postojeće rješenje.

Ovim radom želja je gotovim modulom omogućiti širu primjenu tehnologije dvostruke autentifikacije pomoću prepoznavanja lica. Također, samo rješenje je pisano u Python programskom jeziku, koji je interoperabilan.

Popis kratica

CSRF – *Cross-Site Request Forgery*, vrsta napada na *web*-stranicu koja se izvodi posredstvom klijentskog programa

HOG – *Histogram of Oriented Gradients*, histogram orijentiranih gradijenata

HTML – *HyperText Markup Language*, prezentacijski jezik za izradu *web*-stranica

HTTP – *HyperText Transfer Protocol*, glavna metoda prijenosa informacija na *webu*

MVT – *Model-View-Template*, Model-Pogled-Predložak

PIP – *Pip Installs Packages*, sustav upravljanja paketima

SMS – *Short message service*, kratka tekstualna poruka

URI – *Uniform Resource Identifier*, usklađeni identifikator sadržaja

URL – *Uniform Resource Locator*, usklađeni lokator sadržaja

Popis slika

Popis kodova

Literatura

[1] SAMARTH BRAHMBHATT, *PRACTICAL OPENCV*, 2013.

[2] RAFAEL C. GONZALEZ, RICHARD E. WOODS, *DIGITAL IMAGE PROCESSING*, 1997.

[3] JAN ERIK SOLEM, *PROGRAMMING COMPUTERVISION WITH PYTHON*, 2012.

[4] BRINK, H., J. RICHARDS, AND M. FETHEROLF, *REAL-WORLD MACHINE LEARNING*, 2014.

[5] P. VIOLA, M. JONES, *ROBUSST REAL-TIME OBJECT DETECTION*, 2010.

[6] WILLIAM S. VINCENT, *DJANGO FOR PROFESSIONALS*, 2019.

[7] ADRIAN ROSEBROCK, *FACE RECOGNITION WITH OPENCV, PYTHON, AND DEEP LEARNING URL HTTPS://WWW.PYIMAGESEARCH.COM/2018/06/18/FACE-RECOGNITION-WITH-OPENCV-PYTHON-AND-DEEP-LEARNING/*, 2018.

[8] ERIC BIDELMAN, *CAPTURING AUDIO & VIDEO IN HTML5 URL HTTPS://WWW.HTML5ROCKS.COM/EN/TUTORIALS/GETUSERMEDIA/INTRO/*, 2016.

[9] JULIANA DE GROOT, URL HTTPS://DIGITALGUARDIAN.COM/BLOG/HISTORY-DATA-BREACHES

[10] SONIA OHLYAN, SUNITA SANWAN, TARUN AHUJA, A SURVEY ON VARIOUS PROBLEMS & CHALLENGES IN FACE RECOGNITION, URL HTTPS://WWW.IJERT.ORG/RESEARCH/A-SURVEY-ON-VARIOUS-PROBLEMS-CHALLENGES-IN-FACE-RECOGNITION-IJERTV2IS60850.PDF

[11] FACIAL RECOGNITION SYSTEM, URL HTTPS://EN.WIKIPEDIA.ORG/WIKI/FACIAL_RECOGNITION_SYSTEM

[12] EVOLUTION OF MACHINE LEARNING, URL HTTPS://WWW.SAS.COM/EN_IE/INSIGHTS/ANALYTICS/MACHINE-LEARNING.HTML

[13] LABELED FACES IN THE WILD, URL HTTP://VIS-WWW.CS.UMASS.EDU/LFW/

[14] MACHINE LEARNING, URL HTTPS://EN.WIKIPEDIA.ORG/WIKI/MACHINE_LEARNING

[15] PYTHON PROGRAMMING LANGUAGE, URL HTTPS://EN.WIKIPEDIA.ORG/WIKI/PYTHON_(PROGRAMMING_LANGUAGE)

[16] DJANGO WEB FRAMEWORK, URL HTTPS://EN.WIKIPEDIA.ORG/WIKI/DJANGO_(WEB_FRAMEWORK)

[17] DLIB C++ LIBRARY, URL HTTP://DLIB.NET/

[18] FACE RECOGNITION LIBRARY, URL HTTPS://GITHUB.COM/AGEITGEY/FACE_RECOGNITION

[19] DJANGO BEST PRACTICES, URL HTTPS://STACKOVERFLOW.COM/QUESTIONS/22841764/BEST-PRACTICE-FOR-DJANGO-PROJECT-WORKING-DIRECTORY-STRUCTURE

[20] DJANGO CODING STYLE, URL HTTPS://DOCS.DJANGOPROJECT.COM/EN/DEV/INTERNALS/CONTRIBUTING/WRITING-CODE/CODING-STYLE/

[21] DOCKER, URL HTTPS://WWW.DOCKER.COM/WHY-DOCKER

Prilog

Završni rad može imati priloge, ali se oni ne prilažu uz pisanu verziju završnog rada već se mogu priložiti na završnom ispitu ukoliko povjerenstvo na završnom ispitu tako odluči. Važno je čuvati svu poratnu dokumentaciju koja je nastala pri izradi završnog rada.

S unutarnje strane na zadnjim koricama originala, kao i svake kopije završnog rada, pričvršćuje se CD s kompletnim završnim radom u izvornom formatu (npr. .doc) i .pdf formatu sa svom popratnom dokumentacijom i programima. Pri čemu je obvezno da na tom CD- u postoji i dokument koji opisuje kako se rezultat njegova diplomskog rada (softver ili hardver) koristi (ili kako se npr. izvode mjerenja koja je opisao u radu). Ako se radi o softveru nužno je opisati i kako se programska podrška instalira.

ALGEBRA
VISOKO UČILIŠTE

DVOSTRUKA AUTENTIFIKACIJA POMOĆU PREPOZNAVANJA LICA

Pristupnik: Filip Zelić, 0229033456

Mentor: prof. Aleksander Radovan