

Programsko inženjerstvo

Ak. god. 2021./2022.

IS2 – Digitalizacija

Dokumentacija, Rev. 2

Grupa: *Septabil*

Voditelj: *Dominik Jurinčić*

Datum predaje: *14. 01. 2022.*

Nastavnik: *Igor Stančin*

Sadržaj

1	Dnevnik promjena dokumentacije	3
2	Opis projektnog zadatka	6
3	Specifikacija programske potpore	10
3.1	Funkcionalni zahtjevi	10
3.1.1	Obrasci uporabe	12
3.1.2	Sekvencijski dijagrami	15
3.2	Ostali zahtjevi	19
4	Arhitektura i dizajn sustava	21
4.1	Baza podataka	22
4.1.1	Opis tablica	22
4.1.2	Dijagram baze podataka	25
4.2	Dijagram razreda	26
4.3	Dijagram stanja	28
4.4	Dijagram aktivnosti	29
4.5	Dijagram komponenti	30
5	Implementacija i korisničko sučelje	32
5.1	Korištene tehnologije i alati	32
5.2	Ispitivanje programskog rješenja	34
5.2.1	Ispitivanje komponenti	34
5.2.2	Ispitivanje sustava	40
5.3	Dijagram razmještaja	57
5.4	Upute za puštanje u pogon	58
5.4.1	Izgradnja aplikacije	58
5.4.2	Postavljanje aplikacije na Amazon App trgovinu	59
5.4.3	Postavljanje baze podataka na Amazon RDS-u	65
6	Zaključak i budući rad	70

Popis literature	72
Indeks slika i dijagrama	73
Dodatak: Prikaz aktivnosti grupe	74

1. Dnevnik promjena dokumentacije

Kontinuirano osvježavanje

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Dodani osnovni elementi arhitekture sustava i baze podataka.	Dominik Jurinčić, Borna Radojčić i Ivan Lovrić	14.11.2021.
0.2	Dodane manje modifikacije na modelu baze podataka.	Borna Radojčić i Ivan Lovrić	14.11.2021.
0.3	Dodan dio općeg opisa projektnog zadatka.	Marko Bunić	14.11.2021.
0.4	Promjene na dokumentaciji baze podataka, ostalih zahtjeva i vremena rada na projektu.	Borna Radojčić i Ivan Lovrić	17.11.2021.
0.5	Ispravljen dio dokumentacije vezan uz arhitekturu sustava.	Dominik Jurinčić	17.11.2021.
0.6	Dodan dijagram obrazaca uporabe.	Filip Martinović	17.11.2021.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.7	Postavljena cjelovita planirana baza podataka, pripadni relacijski dijagram i sekvencijski dijagrami.	Borna Radojčić, Ivan Lovrić i Jakov Prister	19.11.2021.
0.8	Dovršen opis projektnog zadatka.	Marko Bunić	19.11.2021.
0.9	Promijenjeni ostali zahtjevi, dodan dnevnik sastajanja i izbrisani nebitni dijelovi.	Marko Bunić i Dominik Jurinčić	19.11.2021.
0.10	Dodani dijagrami razreda.	Filip Martinović	19.11.2021.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	*	19.11.2021.
1.1	Popravljen dio dokumentacije vezan uz opću arhitekturu sustava.	Dominik Jurinčić	7.1.2022.
1.2	Ispravljeni sekvencijski dijagrami.	Jakov Prister i Borna Radojčić	10.1.2022.
1.3	Dodane upute za puštanje u pogon.	Borna Radojčić	11.1.2022.
1.4	Dodane dijagram razreda.	Filip Martinović	12.1.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.5	Dodan unaprijeđeni ER dijagram.	Ivan Lovrić i Borna Radojčić	13.1.2022.
1.6	Dodani dijagrami aktivnosti i razmještaja.	Jakov Prister	14.1.2022.
1.7	Dodane korištene tehnologije i alati.	Dominik Jurinčić	14.1.2022.
1.8	Dodan dijagram komponenti.	Ivan Lovrić	14.1.2022.
1.9	Dodan dijagram stanja.	Marko Bunić	14.1.2022.
1.10	Dodan zaključak.	Marko Bunić i Ivan Lovrić	14.1.2022.
2.0	Potpuna dokumentacija.	*	14.1.2022.

2. Opis projektnog zadatka

dio 1. revizije

Cilj ovog projekta je razviti prikladnu strukturu i programsku podršku za stvaranje mobilne aplikacije "DigiFyl". Aplikacija će biti dostupna i prikladna za korištenje svim računovodstvenim tvrtkama koje žele prijeći na novi i moderniji način skladištenja svojih dokumenata te im omogućiti brz i jednostavan prijelaz. U današnjem dobu gdje je digitalizacija sve jači faktor u svim granama proizvodnje i osobnog života, sustavi koji se njome ne koriste su zastarjeli. Još jedna značajna pozitivna strana korištenja DigiFyl digitaliziranog sustava skladištenja dokumenta je što se smanjuje potreban fizički međuljudski kontakt, što uvelike olakšava rad s dokumentima u današnje vrijeme. Također je i dijeljenje i traženje određenih dokumenata jednostavnije i efikasnije nego rad s fizičkim dokumentima.

Naše usluge se dijele na aplikaciju koju će svi korisnici unutar klijentske tvrtke koristiti na svojem mobitelu i pozadinsku strukturu koja će podržavati dio funkcionalnosti aplikacije i skladištiti te čuvati dokumente tvrtke. Cilj nam je učiniti potrebne promjene pri prijelazu na korištenje naših usluga vrlo jednostavnim za korisničku tvrtku, učiniti sam prijelaz brzim i efikasnim te napraviti aplikaciju jednostavnom i intuitivnom za korištenje. Glavne funkcionalnosti aplikacije su skeniranje tekstualnih dokumenata u fizičkom obliku pomoću kamere te kreiranje njegove digitalne kopije koju je moguće dijeliti s drugim korisnicima i skladištiti u sustavu. Također je unutar aplikacije implementiran i korisnički sustav koji podržava rad unutar tvrtke.

Pokazni primjer funkcionalnosti aplikacije, korisnici i tipovi dokumenata

Pri prvom pokretanju aplikacije korisnika, korisniku se na izbor otvara mogućnost kreacije novog računa ili prijavljivanja na vlastiti račun. Pri kreaciji novog računa, korisnik bira i svoju ulogu na temelju koje mu aplikacija kasnije nudi različite funkcionalnosti i dodjeljuje prava. Za kreaciju novog računa potrebni su:

- korisničko ime
- lozinka

- ime i prezime
- broj mobitela
- email adresa
- uloga u tvrtki

Registracijom u sustav korisniku se dodjeljuju prava i funkcionalnosti izabrane uloge te korisnik šalje zahtjev za pristupanje zajednici određene tvrtke. Direktor tvrtke nakon kreacije računa kreira i tvrtku te on pregledava zahtjeve korisnika za pridruživanje tvrtki i odobrava ili odbija te iste zahtjeve. Također direktor može dodijeliti ta ista prava zaposlenicima koje odabere.

Uloge i njihove pripadne funkcionalnosti i prava su:

- Zaposlenik- su najbrojnija pozicija, njihova uloga je skeniranje dokumenata i slanje skeniranih dokumenata revizoru. Zaposlenik također može vidjeti povijest svojih skeniranja.
- Revizor- ima iste funkcije i prava kao i korisnik, uz dodatna prava i funkcije. Dodatna funkcija je da nakon što revizor skenira dokument, aplikacija automatski određuje tip dokumenta i računovođu kojem se skenirani dokument šalje. Također revizor pregledava sve pristigle dokumente i šalje ih odgovarajućim računovođama.
- Računovođa- dobivene dokumente arhivira u bazu podataka. Također, računovođa ima opciju slanja dokumenta direktoru na potpis prije arhiviranja te arhiviranja nakon potpisa.
- Direktor- uz funkcije i prava svih ostalih korisnika, može i vidjeti povijest svih dokumenta te povijest i statistike svih zaposlenika, kreira tvrtku pri kreiranju računa, prihvaća ili odbija zahtjeve korisnika svojoj tvrtki te može mijenjati pozicije svoj zaposlenika i dodjeljivati im određena prava, kao pravo prihvaćanja zahtjeva tvrtki.

Tipovi dokumenata

- računi Računi će unutar dokumenta na kraju teksta imati oznaku računa koja je veliko slovo R te šest znamenaka. Računi osim oznake sadrže ime klijenta, artikla s cijenama i ukupnu cijenu
- ponude ponude će unutar dokumenta na kraju teksta imati veliko slovo P i devet znamenaka. Ponude će osim oznake sadržavati artikla s cijenama i ukupnu cijenu

- interni dokumenti Interni dokumenti će unutar dokumenta na kraju teksta imati „INT“ i četiri znamenke. Ostatak internog dokumenta je nestrukturirani tekst.

Pregled elastičnosti infrastrukture aplikacije i poslovnog modela

Naša aplikacija i sustav pokriva osim predstavljene aplikacije, i pozadinski kod, i bazu podataka unutar koje bi se čuvali svi spremljeni dokumenti. Budući da smo za našu serversku stranu odabrali servise Amazon Web Services-a, to nam daje mogućnost održati našu pozadinsku strukturu skalabilnom koja nam uvelike olakšava dio posla s predviđanjem potrebnog razmjera pozadinske strukture i omogućava lako i okretno prilagođavanje potrebama naših klijenata. Zato na samom početku, naš sustav ima male zahtjeve unutar Amazonovog Web Clouda te naše potrebe ne izlaze izvan dozvoljenih granica bez plaćanja AWS-ovih usluga što nam je sasvim dovoljno za testiranje našeg proizvoda od strane klijenata. Nakon što bi se određeni klijent odlučio za korištenje naših usluga i predstavio nam svoje potrebe, mi bi prikladno odredili potreban razmjer našeg pozadinskog sistema, naknadu koju bi plaćali AWS-u za traženo proširenje usluga, i uzimajući sve u obzir, predstavili klijentu cijenu naših usluga među koje ulazi i podrška korisnicima naših usluga. Predstavljen način izdavanja ponuda klijentima i prikladnog mijenjanja sustava naknadno je mnogo efikasniji i jednostavniji radi brzih i jednostavnih mogućnosti mijenjanja skale pozadinskog sustava koju AWS pruža. Nadalje, nakon prikladnog širenja sustava za jednog klijenta, dolazak i korištenje naših usluga novog klijenta nikako neće utjecati na prethodnog klijenta. Promjene i povećanja sustava imaju i više opcija te, uz prethodno navedena svojstva, mi jednostavno možemo odrediti prikladno rješenje, odrediti zahtjeve novog proširenog sustava unutar AWS-ovog web clouda. Znajući troškove korištenja AWS-ovih usluga, lako možemo predstaviti novom klijentu ponudu korištenja naših usluga koja nikako utječe na ostale klijente. Time korištenje naših usluga od strane klijenata je jednostavno, nikakve integracije sustava nisu potrebne jer se mi brinemo o samoj strukturi koja čuva i skladišti dokumente korisnika te je naš sustav prilagodljiv i okretan za sve potrebne promjene i nove zahtjeve.

Pregled prilagodljivosti korisničkog sustava

Pri ponudi naših usluga novoj korisničkoj stranci, također nudimo i kreaciju posebno prilagođenog sustava korisnika dizajniranom isključivo za tu korisničku stranku. Uzimajući u obzir tipove dokumenata, različite pozicije pojedinih zaposlenika unutar tvrtke i njihovih ovlasti i funkcija, pomoću naših postojećih predložaka

sustava korisnika i dokumenata zajedno s izradom i implementacijom svega novog potrebnog za specifični zahtjev, brzo i jednostavno kreiramo prilagođen sustav korisnika, dokumenata i funkcionalnosti. Cijena ove usluge ovisi o samim specifikacijama određenog zahtjeva.

Usporedba s konkurentnim proizvodima na tržištu

Na tržištu već postoje mnoge aplikacije slične našoj po tome što implementiraju optičko prepoznavanje znakova i omogućuju skeniranje tekstualnih zapisa sa slike te prebacivanje skeniranog teksta u druge formate. Primjeri takvih aplikacija su Text Scanner i Text Fairy dostupne na Google Play-u. Isto tako, mnogi pružatelji usluga skladištenja i rada s digitaliziranim dokumentima su već duže vremena dostupni na tržištu, kao što su ClearDATA i FileCloud, te postoje i mnogi koji uz digitalno skladištenje dokumenta pružaju i usluge skeniranja dokumenata kao što su Apyxx Technologies i GRM Document Management.

Ono po čemu se naša aplikacija razlikuje od navedenih je što osim što sadrži potpunu pozadinsku potporu za skladištenje i rad sa skeniranim tekstualnim dokumentima zajedno s odgovarajućim korisničkim sistemom i održavanjem cijelog sistema dokumenata, također nudi elastičnost i prilagodljivost svakoj zasebnoj korisničkoj stranci u obliku prilagođavanja korisničkog sistema i skale pozadinskog sustava. Korištenje naše aplikacije je jednostavno i intuitivno te dolazi uz kontinuiranu korisničku podršku. Time su naše usluga najbolji izbor za izradu digitalnih kopija dokumenata, održavanje njihovog sistema skladištenja i rad među korisnicima sa digitaliziranim dokumentima za odgovarajuće tvrtke kojima su potrebne navedene usluge.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

dio 1. revizije

Dionici:

1. Zaposlenici
 - (a) Revizor
 - (b) Računovođa
 - (c) Normalan zaposlenik
2. Direktor
3. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može::
 - (a) pristupiti Login aktivnosti
 - (b) pristupiti Sign-up aktivnosti
2. Radnik može:
 - (a) pristupiti Login aktivnosti i prijaviti se
 - (b) skenirati dokument i podvrti točnost skeniranja ili označiti dokument kao krivo skeniran
 - (c) pogledati svoju povijest skeniranja
3. Računovođa može:
 - (a) pristupiti Login aktivnosti i prijaviti se
 - (b) skenirati dokument i podvrti točnost skeniranja ili označiti dokument kao krivo skeniran
 - (c) pogledati svoju povijest skeniranja
 - (d) arhivirati dokument

- (e) poslati dokument direktoru na potpis
- (f) arhivirati potpisani dokument
- (g) dobiti pregled dokumenta prije slanja na potpis / arhiviranja

4. Revizor može:

- (a) pristupiti Login aktivnosti i prijaviti se
- (b) skenirati dokument i podvrti točnost skeniranja ili označiti dokument kao krivo skeniran
- (c) pogledati svoju povijest skeniranja
- (d) provjeriti dokument i preusmjeriti ga računovođi koji je zadužen za taj tip dokumenta
- (e) dobiti pregled dokumenta prije slanja na preusmjerivanja

5. Direktor može:

- (a) pristupiti Login aktivnosti i prijaviti se
- (b) skenirati dokument i podvrti točnost skeniranja ili označiti dokument kao krivo skeniran
- (c) pogledati svoju povijest skeniranja
- (d) vidjeti statistike radnika firme
- (e) potpisati dokument
- (f) dobiti pregled dokumenta prije slanja na preusmjerivanja

6. Baza podataka:

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje skenirane dokumente i ispavnost njihovog skeniranja

3.1.1 Obrasci uporabe

dio 1. revizije

Opis obrazaca uporabe

UC1 - Login

- **Glavni sudionik:** Korisnik aplikacije
- **Cilj:** prijaviti se u aplikaciju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je dobio password i username od direktora
- **Opis osnovnog tijeka:**
 1. Korisnik aplikacije unosi username i password u to predviđena "text input-a"
 2. pritišće Login gumb
 3. biva preusmjeren na određenu aktivnost, ovisno o svojoj ulozi unutar firme
- **Opis mogućih odstupanja:**
 - 1.a Unos krivih podataka
 1. Ponoviti unos, te paziti prilikom unosa podataka
 2. Provjeriti s direktorom ispravnost podataka

UC2 - Skreniranje dokumenta

- **Glavni sudionik:** Korisnik aplikacije
- **Cilj:** Skrenirati dokument
- **Sudionici:** Baza podataka, Revizor
- **Preduvjet:** Korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik aplikacije pritišće gumb za odabir skeniranja dokumenta
 2. Smiruje mobilni uređaj kako skeniranje ne bi bilo mutno
 3. Fokusira kameru
 4. Pritišće gumb za skeniranje, to jest slikanje dokumenta
 5. Korisnik odabire je li dokument ispravno skeniran
 6. Ako je dokument skreniran od strane zaposlenika i obilježen kao ispravno skeniran, taj dokument se šalje revizoru
 7. Ako revizor skenira dokument onda se dokument šalje računovođi koji je zadužen za tu vrstu dokumenta

- **Opis mogućih odstupanja:**

- 2.a Mutna slika

- 1. Pokušati držati mobilni uređaj mirno pri skeniranju

- 5.a Neispravno skeniran dokument

- 1. Skenirati dokument ponovo

UC3 - Preusmjeravanje dokumenta računovođi

- **Glavni sudionik:** Revizor

- **Cilj:** Preusmjeriti podatke

- **Sudionici:** Baza podataka, Računovođa

- **Preduvjet:** Revizor je prijavljen i ima dokument koji može preusmjeriti

- **Opis osnovnog tijeka:**

- 1. Revizor pritišće gumb za pregled "dobro" skeniranih dokumenata
 - 2. Potvrđuje da je dokument dobro skeniran pritiskom na taj dokument
 - 3. Odabirom ispravnog dokumenta može ga poslati računovođi koji je zadužen za taj dokument pritiskom gumba "pošalji računovođi" i odabirom računovođe

UC4 - Arhiviranje dokumenta

- **Glavni sudionik:** Računovođa

- **Cilj:** Arhivirati ispravno skenirani dokument

- **Sudionici:** Baza podataka

- **Preduvjet:** Računovođa je prijavljen i ima dokument koji može arhivirati

- **Opis osnovnog tijeka:**

- 1. Računovođa pritišće gumb za pregled potencijalno potpisanih i preusmjerenih dokumenata
 - 2. Računovođa pritiskom na dokument označuje dokument za arhiviranje ili slanje na potpis
 - 3. Utvrđuje da je dokument ispravno preusmjeren ili potpisan, te pritišće gumb "Arhiviraj"
 - 4. Dokumentu dobiva oznaku "arhiviran" u bazi podataka

UC5 - Slanje dokumenta direktoru na potpis

- **Glavni sudionik:** Računovođa

- **Cilj:** Poslati dokument direktoru na potpis

- **Sudionici:** Baza podataka, direktor

- **Preduvjet:** Računovođa je prijavljen i ima dokument koji može poslati direktoru
- **Opis osnovnog tijeka:**
 1. Računovođa pritišće gumb za pregled potencijalno potpisanih i preusmjerenih dokumenata
 2. Računovođa odabire ne-potpisan dokument i šalje ga direktoru na potpis pritiskom gumba "pošalji na potpis"

UC6 - Potpis dokumenta

- **Glavni sudionik:** Direktor
- **Cilj:** Potpisati dokument
- **Sudionici:** Baza podataka
- **Preduvjet:** Direktor je prijavljen i ima dokument koji može potpisati
- **Opis osnovnog tijeka:**
 1. Direktor pritišće gumb "Dokumenti za potpis" za pregled preusmjerenih dokumenata
 2. Direktor odabire jedan ili više ne-potpisanih dokumenata i potpisuje ih pritiskom gumba "Potpiši"
 3. Dokument se sprema u bazu podataka kao potpisan dokument

UC7 - Pregled povijesti dokumenata

- **Glavni sudionik:** Direktor
- **Cilj:** Pregledati povijest dokumenata
- **Sudionici:** Baza podataka, direktor
- **Preduvjet:** Direktor je prijavljen i baza podataka ima barem jedan dokument
- **Opis osnovnog tijeka:**
 1. Direktor pritišće gumb za pregled povijest i statistiku
 2. Direktor pritišće gumb za pregled skeniranih dokumenata

UC8 - Pregled statistike zaposlenika

- **Glavni sudionik:** Direktor
- **Cilj:** Pregledati statistike zaposlenika
- **Sudionici:** Baza podataka, direktor
- **Preduvjet:** Direktor je prijavljen i postoji barem jedan zaposlenik
- **Opis osnovnog tijeka:**
 1. Direktor pritišće gumb za pregled povijest i statistiku

2. Direktor pritišće gumb za pregled statistike radnika
- **Opis mogućih odstupanja:**
 - 2.a Ne-prikaz nekih statistika
 1. Ako se neke statistike ne prikažu, to znači da korisnik nema ovlasti za te radnje

UC9 - Stvaranje korisnika

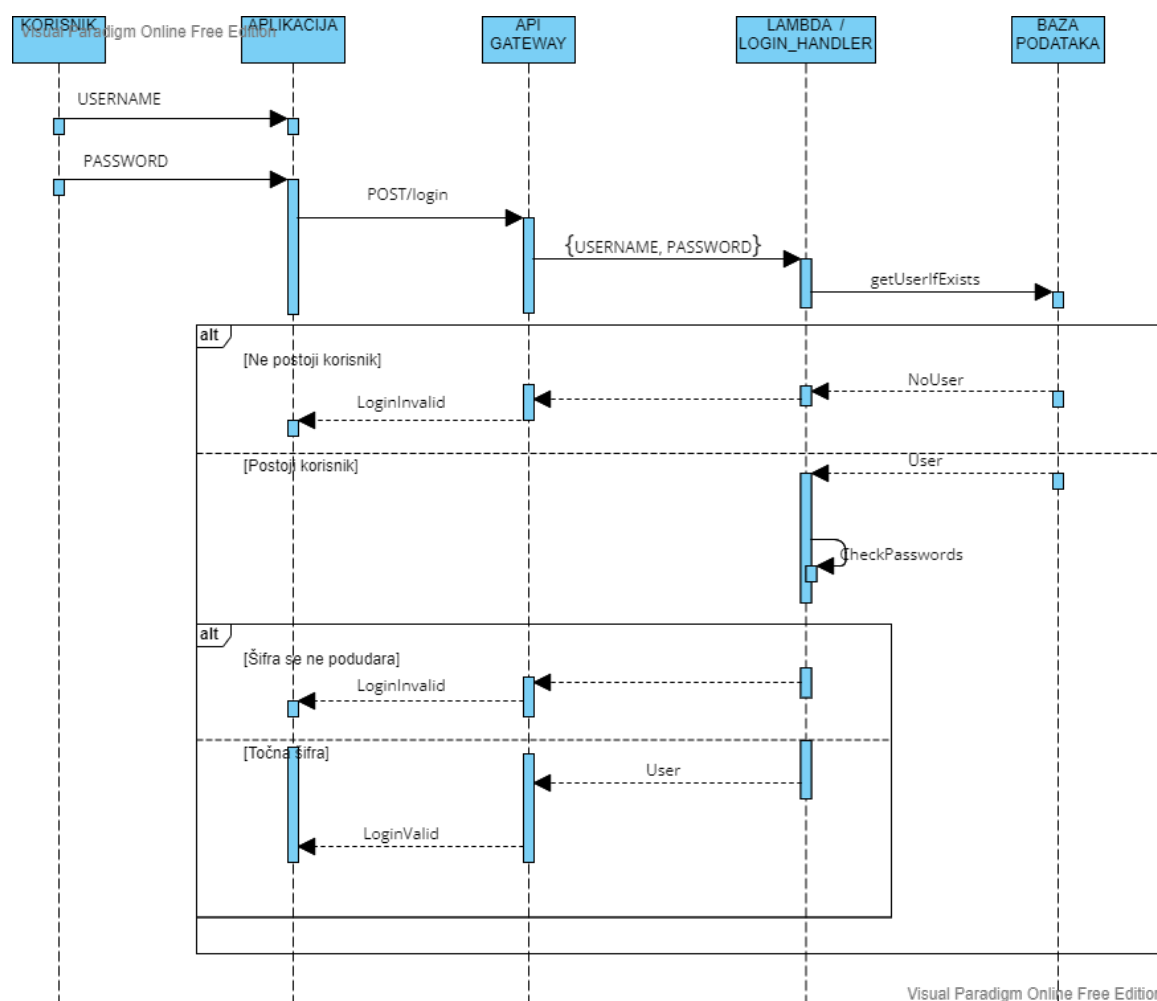
- **Glavni sudionik:** Direktor
- **Cilj:** Stvoriti korisnika
- **Sudionici:** Baza podataka, korisnik
- **Preduvjet:** Direktor je prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik pritišće gumb "sign up" za Signup aktivnost
 2. Korisnik upisuje podatke na za to predviđena mjesta
 3. Korisnik pritišće gumb "Signup"
 4. Korisnik biva pohranjen u bazu podataka

Dijagrami obrazaca uporabe

3.1.2 Sekvencijski dijagrami

Login korisnika

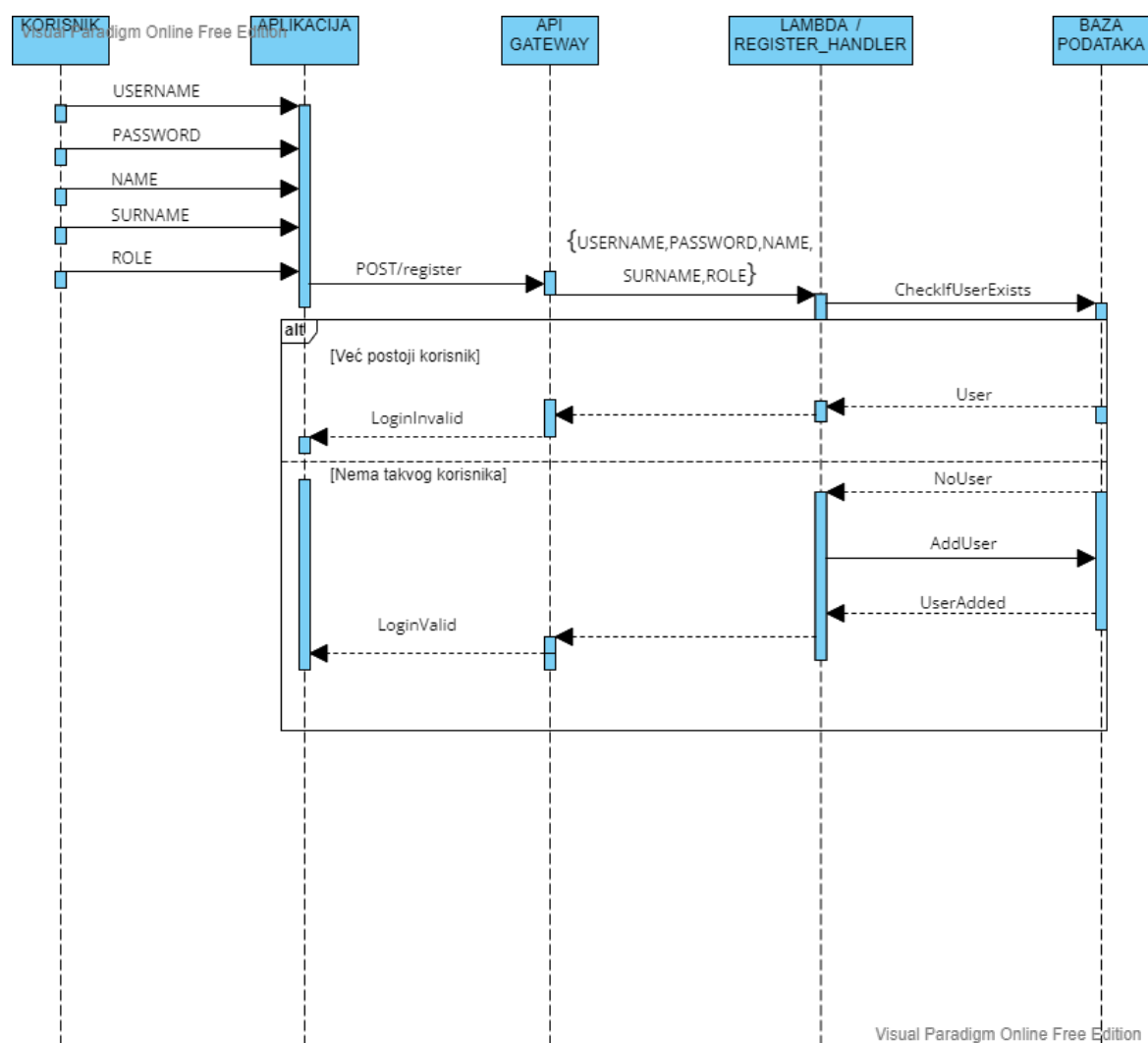
Korisnik unosi korisničko ime i lozinku te se pokušava ulogirati u aplikaciju. Uneseni podaci se preko API Gatewaya i lambda funkcije šalju u bazu podataka. Tamo se provjerava postoji li korisnik s unesenim korisničkim imenom, ako ne postoji korisniku su ispisuje poruka LoginInvalid. Ako je baza pronašla korisnika provjerava njegovu šifru u bazi podataka s upisanom šifrom i ako se one podudaraju ispisuje korisniku LoginValid, u suprotnom ispisuje LoginInvalid.



Slika 3.1: Sekvencijski dijagram za login

Registracija korisnika

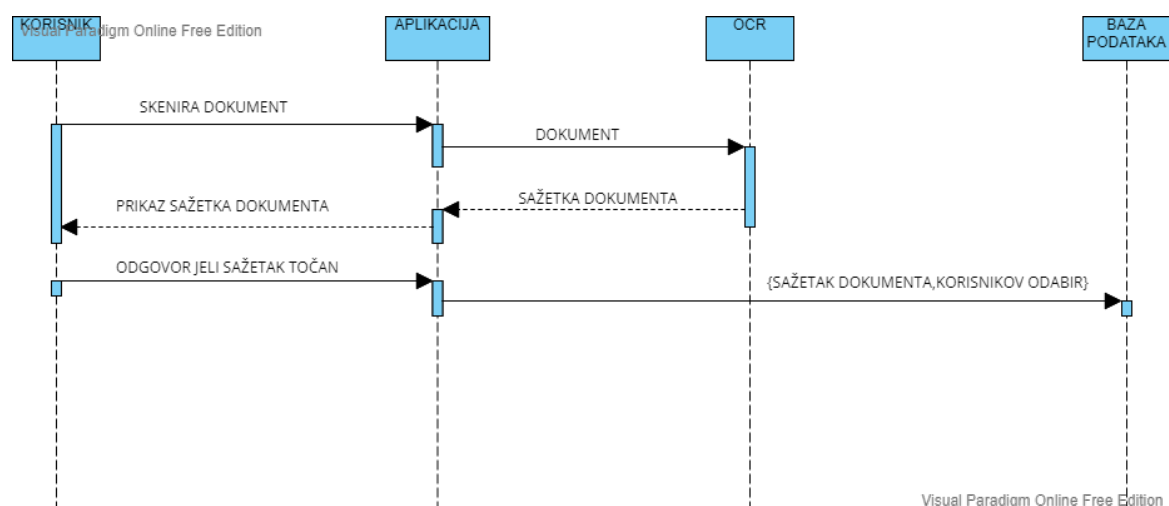
Korisnik unosi korisničko ime, ime, prezime, lozinku i poziciju. Uneseni se podaci preko API Gatewaya i lambda funkcije šalju u bazu podataka. Tamo se provjerava postoji li već korisnik s navedenim korisničkim imenom, ako da ispisuje LoginInvalid. U suprotnom slučaju unosi korisnika u bazu podataka i ispisuje LoginValid.



Slika 3.2: Sekvencijski dijagram za registraciju

OCR funkcija

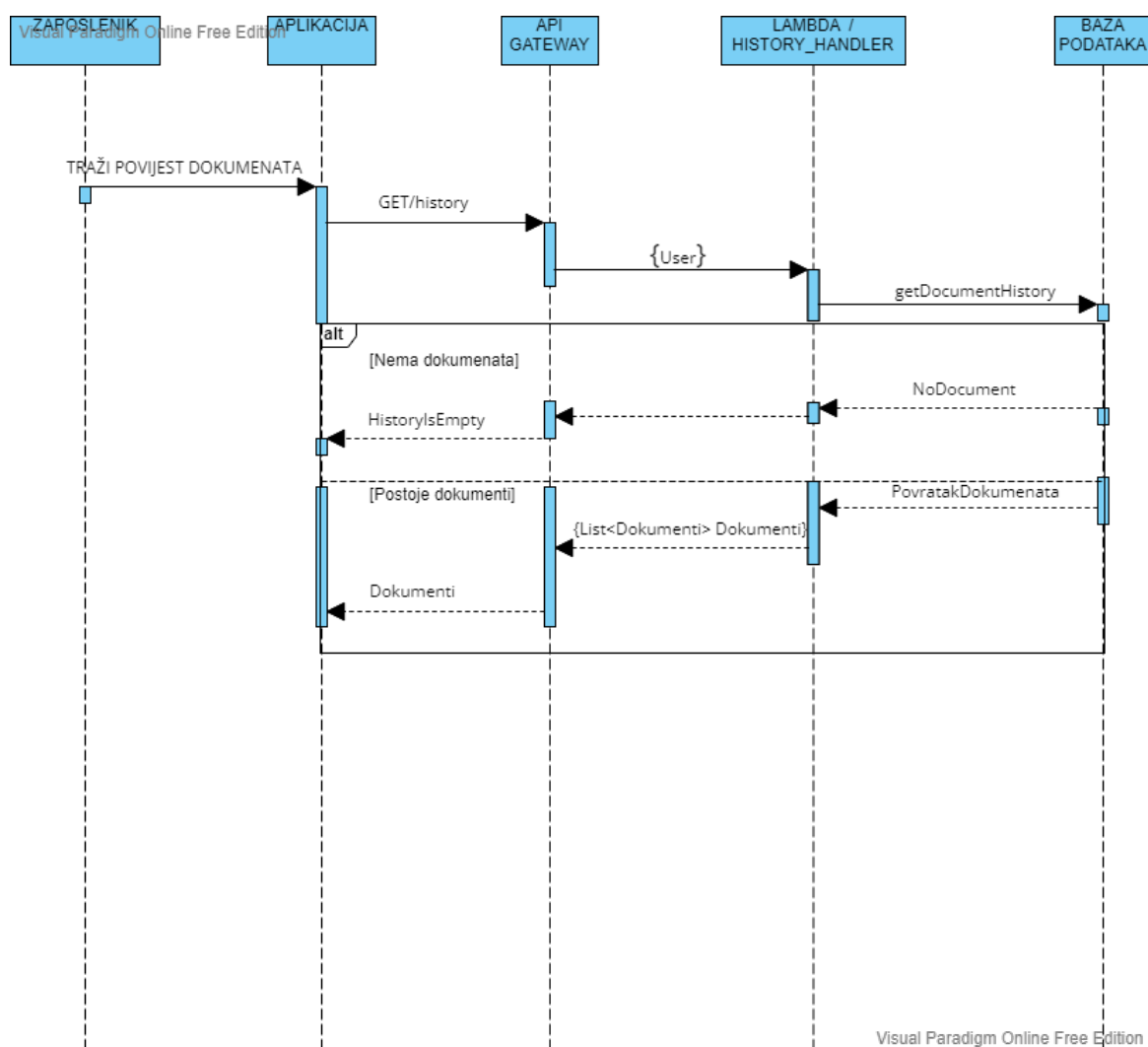
Korisnik skenira željeni dokument te u aplikaciji poziva funkciju OCR koja mu vraća sažetak dokumenta. Zatim korisnik odlučuje točnost dokumenta te to označi u aplikaciji. Sažetak dokumenta i točnost se šalju u bazu podataka neovisno o korisnikovom odabiru.



Slika 3.3: OCR funkcija

Povijest svih skeniranih dokumenata zaposlenika

Zaposlenik u aplikaciji traži povijest svih svojih skeniranih dokumenata. Preko API Gatewaya i lambda funkcije u bazi podatak se provjerava postoji li uopće neki skenirani dokument od zaposlenika, ako ne ispisi zaposleniku da je povijest skeniranja prazna. Ako već postoje skenirani dokumenti u bazi podataka, aplikacija ih daje na pregled zaposleniku.



Slika 3.4: Povijest svih skeniranih dokumenata zaposlenika

3.2 Ostali zahtjevi

dio 1. revizije

Podržani jezik našeg programskog sučelja je Engleski. Vrijeme odziva aplikacije je gotovo trenutačno, možda oko pola sekunde čekanja između klika na login i ulaska u aplikaciju. Podržani broj korisnika nema gornje ograničenje, zato što uvijek možemo dodatno skalirati sustav. Trenutno koristimo AWS free tier usluge te imamo 20GB alociranog prostora i milijun besplatnih zahtjeva na AWS lambda mjesečno. Sa rastom broja korisnika možemo po potrebi skalirati sustav po cijeni ponuđenoj od strane AWS-a. Podržana mobilna platforma je Android OS,

aplikacija je pisana u Javi (Android Studiu) i Pythonu. Razina zaštite na kompletnom sustavu je relativno niska, u bazu podataka se upisuje lozinka u izvornom obliku, bez bilo kakve vrste kriptiranja. Jedina zaštita koju imamo je to što AWS, na kojem nam se sve nalazi, koristi https protokol. Aplikacija je jako pouzdana što se tiče registiranja korisnika i zapisa njihovih podataka u bazu podataka na AWS RDS poslužitelju te dohvaćanju tih podataka iz baze i validaciji prilikom prijave. Registracija je lagana i jednostavna, klikom na "*signup*" gumb se mogu unijeti svi potrebni podaci, a korisničko ime i lozinka su kasnije potrebni za prijavu u sustav. Korištenje aplikacije nakon prijave je isto jako jednostavno, sve opcije za svakog korisnika su ponuđene odmah na početnoj stranici aplikacije.

4. Arhitektura i dizajn sustava

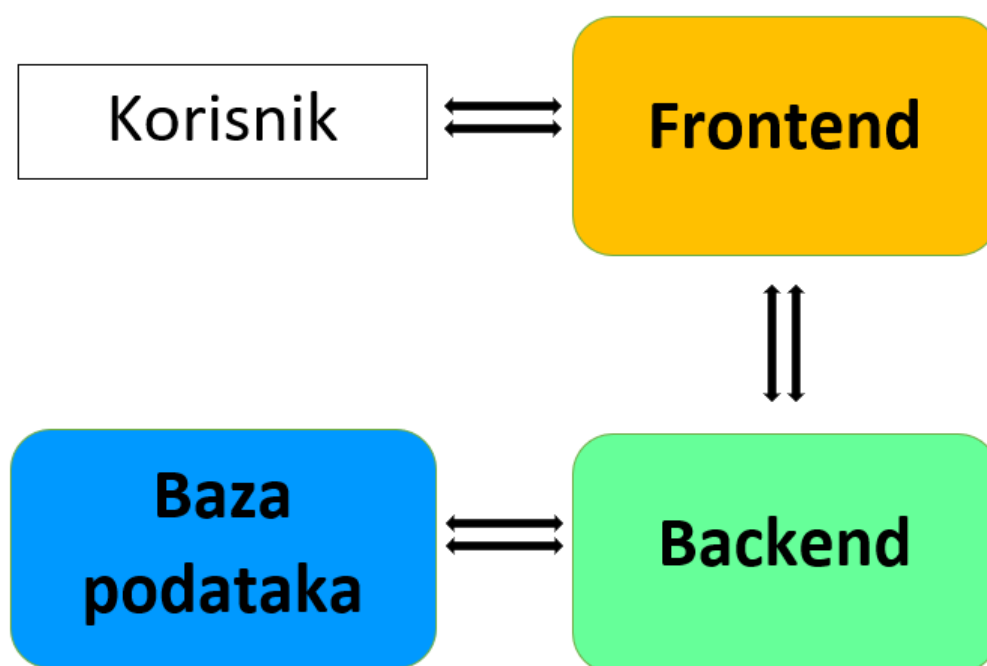
Arhitektura se može podijeliti na tri dijela:

- *Frontend*
- *Backend*
- *Baza podataka*

Frontend predstavlja korisničko sučelje Android aplikacije i njegove funkcionalnosti. Korisničko sučelje prikazuje rezultate korisnikovih zahtjeva te olakšava njihovo slanje. Korisnik putem korisničkog sučelja šalje zahtjeve Backend-u. Korisnik može biti zaposlenik, revizor, računovođa ili direktor. Na temelju svoje uloge korisnici imaju različite ovlasti i samim time mogu slati različite zahtjeve. HTTP zahtjevi se šalju na Amazon AWS API Gateway koji je zadužen za obradu zahtjeva. API Gateway zatim prosljeđuje zahtjev backend-u.

Backend predstavlja dio programa koji se izvodi na web poslužitelju i omogućava izvršavanje zahtjeva koje korisnik šalje. Komunikacija Frontend-a i Backend-a odvija se HTTP protokolom. Nakon što Backend primi zahtjev koji je korisnik uputio, on iz njega izlučuje podatke potrebne za izvršavanje zahtjeva. Ti podaci nalaze se u body-ju HTTP zahtjeva. Backend je ostvaren korištenjem Amazon AWS Lambde. Za svaku vrstu zahtjeva stvorena je zasebna Lambda. Svaka Lambda sadrži funkciju koja kao argument prima body HTTP zahtjeva u obliku rječnika u kojem su ključevi nazivi parametara koji se šalju zahtjevom. Zatim se izvrši kod funkcije i njena povratna vrijednost se ponovo šalje korisniku na frontend-u preko API Gateway-a. Funkcije u Lambdi koriste Pyscopg2 adapter za slanje zahtjeva na bazu podataka.

U bazi podataka nalaze se podaci o korisnicima i dokumenti koje su skenirali. Baza je napravljena koristeći Amazon AWS RDS.



Slika 4.1: Arhitektura sustava

Za izradu Frontend-a naše android aplikacije izabrali smo programski jezik Java, a za Backend smo koristili Python. Za razvoj Frontend-a koristili smo Android Studio, a za razvoj Backend-a Visual Studio Code razvojno okruženje.

Raspodjela arhitekture na Frontend, Backend i bazu podataka omogućila je lakše ispitivanje pojedinih dijelova sustava kao i jednostavnije i nezavisnije razvijanje.

4.1 Baza podataka

Koristili smo relacijsku bazu podataka koju smo napravili u PostgreSQL-u. Trenutna početna verzija baze sadrži relaciju users.

4.1.1 Opis tablica

users		
userId	SERIAL	Jedinstveni identifikator korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

users		
firstName	VARCHAR(50)	Ime korisnika
lastName	VARCHAR(50)	Prezime korisnika
username	VARCHAR(50)	Korisničko ime korisnika - jedinstveno
userPswd	VARCHAR(50)	Lozinka korisnika
userRole	VARCHAR(20)	Uloga korisnika

U tablici users zapisani su svi korisnici aplikacije, svaki korisnik ima jedinstveno korisničko ime (username) i dodjeljuje mu se jedinstveni identifikator (userId) koji je primarni ključ tablice. Ostali atributi koje svaki korisnik ima su njegovo ime i prezime, lozinka u aplikaciji (za prijavu mora unijeti korisničko ime i lozinku) i uloga unutar tvrtke.

scanHistory		
docId	SERIAL	Jedinstveni identifikator dokumenta
userId	INT	Identifikator korisnika
scanDate	DATE	Datum skeniranja

scanHistory je tablica koja u sebi sadrži sve skenirane dokumente, bili oni ispravno ili neispravno skenirani, ispravno skenirani dokumenti će otići u documents dok će neispravni samo ostati ovdje. Njeni atributi su docId (jedinstveni identifikator svakog skeniranog dokumenta), userId (jedinstveni identifikator korisnika koji je skenirao dokument) i datum skeniranja

documents		
docId	INT	Jedinstveni identifikator dokumenta
docLabel	VARCHAR(10)	Oznaka dokumenta
docType	VARCHAR(1)	Tip dokumenta
docText	TEXT	Skenirani tekst

Tablica documents sadrži podatke o svim dokumentima koji su potvrđeni kao ispravno skenirani, podaci koji su zapisani u njoj su docId (jedinostveni identifikator), docLabel (oznaka dokumenta), docType (tip dokumenta) i docText (skenirani tekst u dokumentu)

accountantResponsibility		
userId	INT	Identifikator korisnika
responsibility	VARCHAR(1)	Zaduženje računovođe (Računi, ponude ili interni dokumenti)

accountantResponsibility je tablica koja u sebi sadrži userId nekog računovođe i njegovo zaduženje (računi, ponude ili interni dokumenti)

archive		
archiveId	SERIAL	Jedinostveni identifikator arhive
docId	INT	Jedinostveni identifikator dokumenta
userId	INT	Identifikator korisnika
archivedDate	DATE	Datum arhiviranja
signed	SMALLINT	Dokument potpisan/nepotpisan

archive je tablica u koju se spremaju svi dokumenti koje računovođe arhiviraju, svaka arhiva ima svoj jedinstveni identifikator (archiveId). Atributi koji se spremaju uz archiveId su docId arhiviranog dokumenta, userId računovođe koji ga je arhivirao, datum arhiviranja i signed (dokument potpisan/ne)

reviserPending		
docId	INT	Jedinostveni identifikator dokumenta
userId	INT	Identifikator korisnika

reviserPending je tablica u koju se spremaju dokumenti koji se šalju revizoru, kad će revizor potvrđivati i prosljeđivati dokumente, pregledavati će dokumente iz ove tablice. Ima samo 2 atributa, a to su docId dokumenta kojeg je zaposlenik

poslao i userId revizora kojem se šalje taj dokument.

executivePending		
docId	INT	Jedinstveni identifikator dokumenta
userId	INT	Identifikator korisnika

executivePending je tablica u koju se spremaju dokumenti koje računovođe šalju direktoru na potpis prije arhiviranja, isto ima samo 2 atributa, a to su docId dokumenta poslanog na potpis i userId računovođe kojem se potpisani dokument treba vratiti.

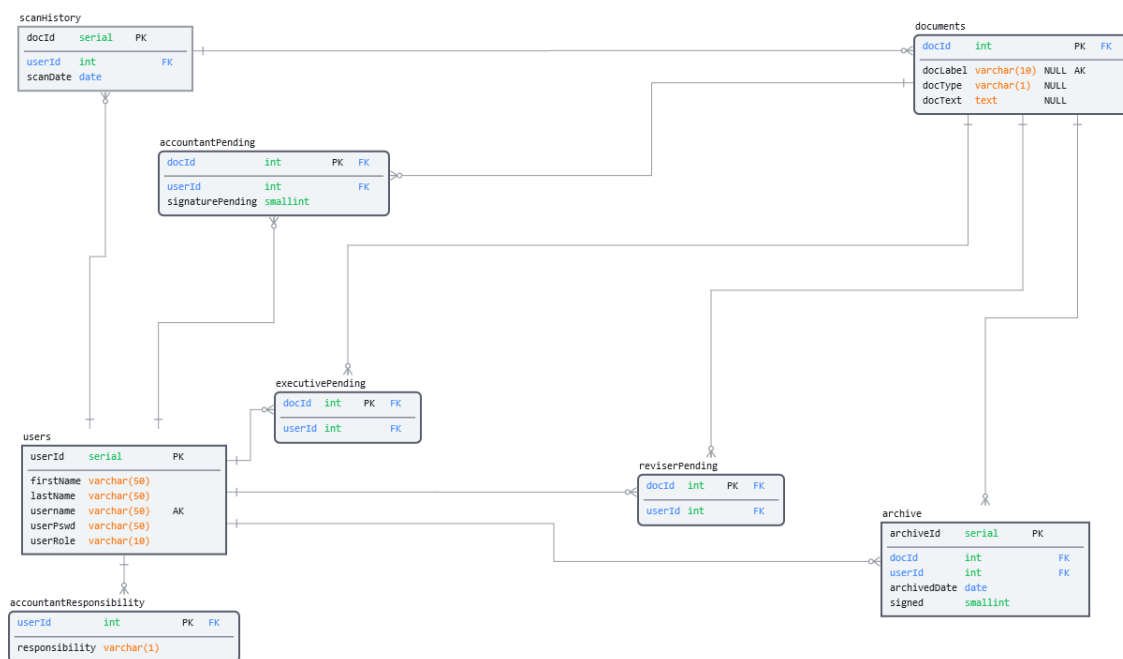
accountantPending		
docId	INT	Jedinstveni identifikator dokumenta
userId	INT	Identifikator korisnika
signaturePending	SMALLINT	Status potpisa

accountantPending je tablica u koju se spremaju dokumenti koji se šalju računovođi na arhiviranje.

Računovođa kojem će dokument biti dostavljen je odabran s obzirom na svoju odgovornost (tablica accountantResponsibility), atributi tablice accountantPending su docId dokumenta koji je poslan računovođi, userId računovođe kojem je poslan dokument i atribut signaturePending koji govori je li potpis tražen.

Ako je vrijednost atributa signaturePending 0, potpis nije tražen, ako je 1, potpis je tražen, ali direktor još nije potpisao dokument, a ako je 2, onda je potpis tražen i direktor je potpisao i vratio dokument.

4.1.2 Dijagram baze podataka



Slika 4.2: ER Diagram

4.2 Dijagram razreda

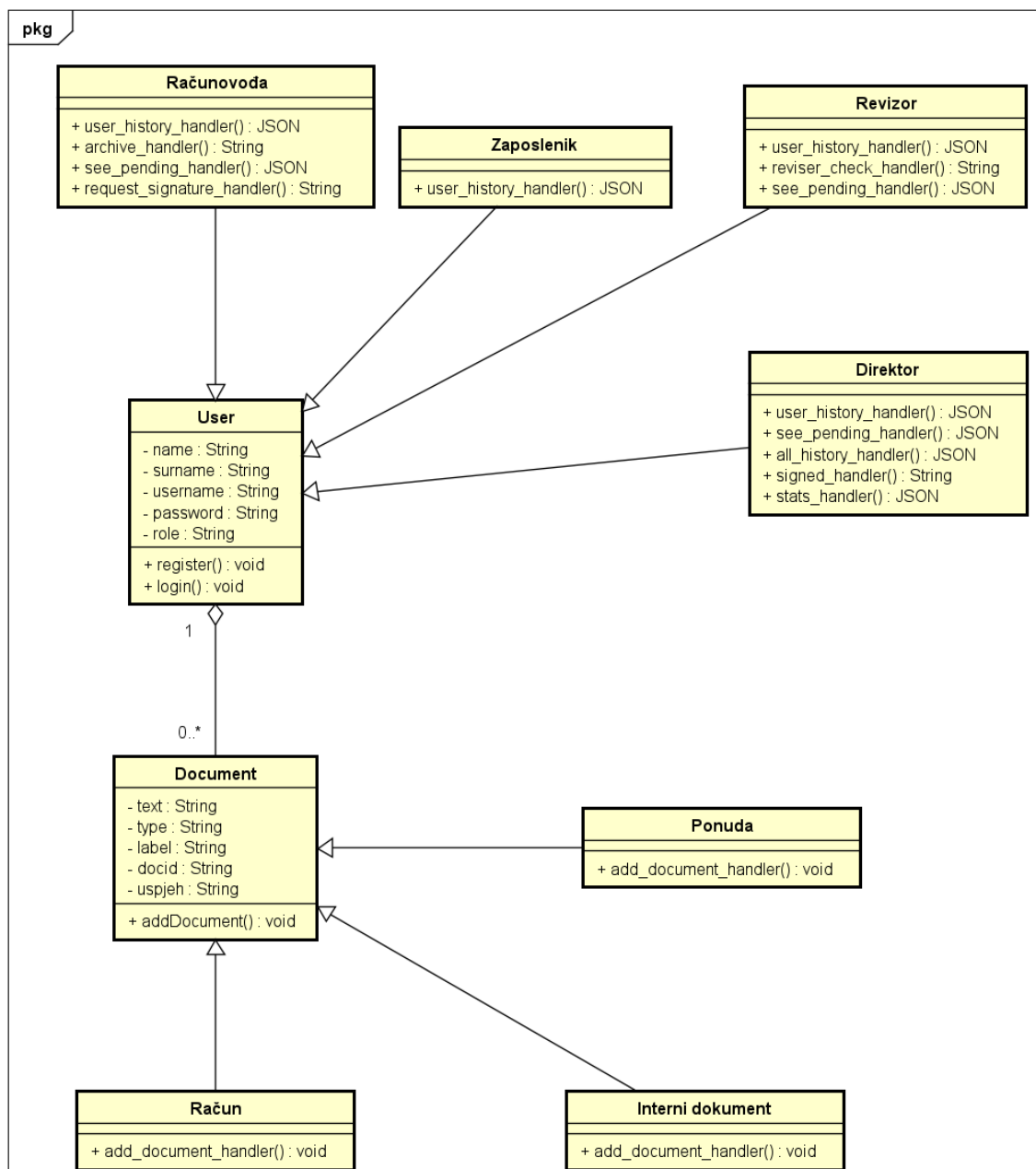
Na slici su prikazani razredi koji pripadaju backend dijelu MVC arhitekture.

U izvornom kodu postoje samo razredi *User* i *Document*.

Razred *User* ima atribut *role* koji određuje kojim funkcijama "handlerima" instanca razreda ima pristup. Postoje tri tipa računovođa koji imaju pristup istim funkcijama. Jedina je razlika u tipu dokumenta kojega arhiviraju.

Razred *Document* ima atribut *type* koji određuje tip dokumenta, te kojem računovođi se šalju. Svi tipovi imaju pristup jednakim funkcijama "handlerima".

Različiti atributi *role* i *type* su u dijagramu prikazani kao razredi izvedeni iz razreda *User* radi preglednosti.

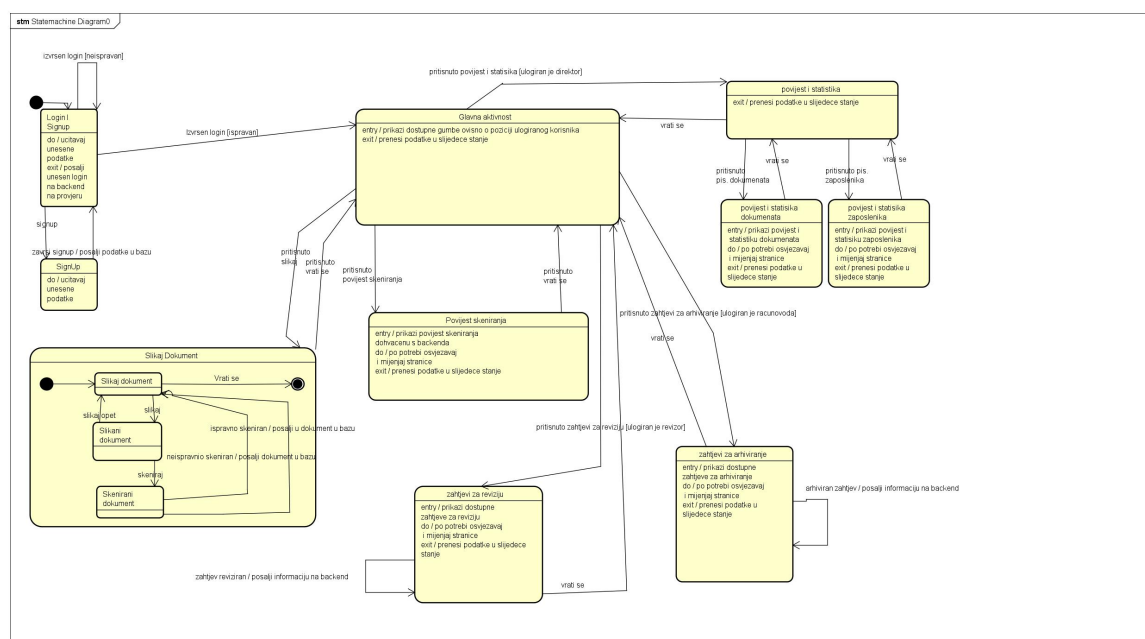


Slika 4.3: Dijagram razreda

4.3 Dijagram stanja

U nastavku je dijagram stanja koji prikazuje stanja, odnosno "stranice" aplikacije. Također su i naznačeni trenutci u kojima aplikacija komunicira s back-endom.

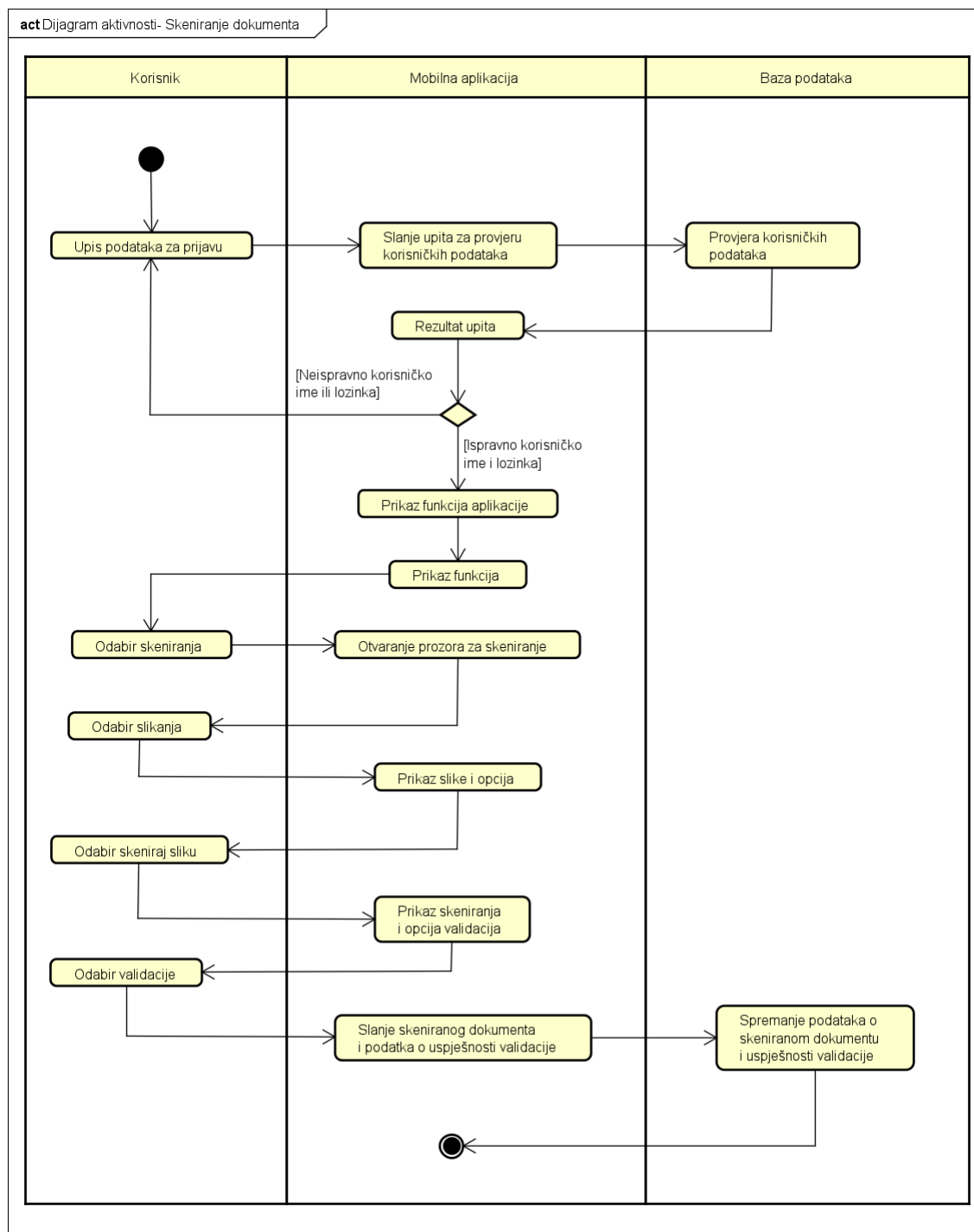
Uvedena je simplifikacija glavne aktivnosti zaposlenika, revizora, direktora i računovođe u glavnu aktivnost radi jednostavnijeg prikaza.



Slika 4.4: Dijagram stanja

4.4 Dijagram aktivnosti

Dijagram aktivnosti pokazuje proces skeniranja jednog dokumenta. Korisnik se prvo prijavljuje u svoj korisnički račun gdje odabire funkciju skeniranja. Zatim korisnik slika željeni dokument i aplikacija mu prikazuje dobiveni sažetak dokumenta. Korisnik završno odlučuje jeli zadovoljan sa sažetkom te se taj njegov odgovor i sažetak šalju u bazu podataka čime je završen proces skeniranja.

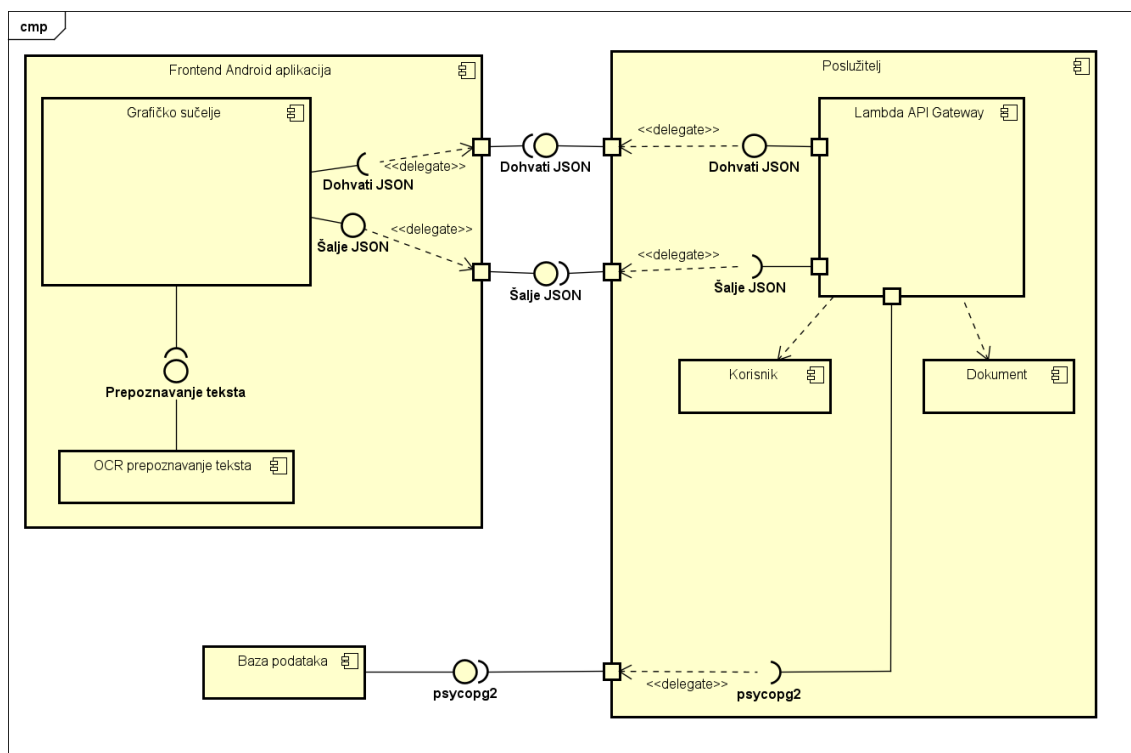


Slika 4.5: Dijagram aktivnosti - Skeniranje dokumenta

4.5 Dijagram komponenti

dio 2. revizije

Dijagram komponenti na slici 4.4 prikazuje komponente sustava te njihove međuovisnosti. Sustavu se pristupa preko komponente grafičkog sučelja koja je zadužena za prikaz sadržaja korisniku. Glavna funkcionalnost aplikacije jest mogućnost skeniranja dokumenata i njihovo spremanje u bazu podataka. Zato je grafičkom sučelju potreban API koji će mu omogućiti prepoznavanje teksta te mu je potrebno sučelje za komunikaciju s poslužiteljem, a sam nudi sučelje za slanje podataka u obliku JSON objekata. Komponenta "OCR prepoznavanje teksta" nudi sučelje za skeniranje dokumenata dok se komunikacija s poslužiteljem odvija "Dohvati JSON" i "Šalje JSON" sučelja. Komponenta "Lambda API Gateway" je zadužena za komunikaciju s bazom podataka i s grafičkim sučeljem, a za rad su joj potrebne komponente "Dokument" i "Korisnik" kako bi mogla komunicirati s bazom podataka. Baza podataka nudi sučelje preko kojeg je omogućena komunikacija s poslužiteljem.



Slika 4.6: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

dio 2. revizije

Komunikacija među članovima tima realizirana je korištenjem aplikacija WhatsApp¹ i Discord². Za izradu sekvencijskih dijagrama korišten je web alat za izradu dijagrama Online Visual Paradigm³. Za izradu svih ostalih dijagrama korišten je program Astah⁴ koji služi za modeliranje i izradu dijagrama u inženjerskoj struci.

Upravljanje izvornim kodom ostvareno je korištenjem sustava Git⁵, a udaljeni repozitorij projekta dostupan je na web sustavu GitLab⁶.

Frontend aplikacije napravljen je u razvojnom okruženju Android Studio⁷ koristeći programski jezik Java⁸. Android Studio je službeno razvojno okruženje za Android platformu i temelji se na Intelij IDEA-u. Prvenstveno se koristi za izradu Android aplikacija i igara. Pri razvoju softvera Android Studio koristi Gradle tehnologiju, predloške koda s GitHub-a i podršku za korištenje Google Cloud platforme.

Backend aplikacije je napisan programskim jezikom Python⁹ u razvojnom okruženju Visual Studio Code¹⁰. Visual Studio Code se najviše koristi za pisanje i debugiranje programskog koda aplikacija. Uz standardne funkcionalnosti Pythona korišten je i adapter za komunikaciju s bazom podataka Psycopg2¹¹.

¹<https://www.whatsapp.com/>

²<https://discord.com/>

³<https://online.visual-paradigm.com/>

⁴<https://astah.net/>

⁵<https://git-scm.com/>

⁶<https://about.gitlab.com/>

⁷Android Studio

⁸<https://www.java.com/en/>

⁹<https://www.python.org/>

¹⁰<https://code.visualstudio.com/>

¹¹<https://pypi.org/project/psycopg2/>

Dodatno je za ostvarenje backenda korištena i cloud platforma Amazon AWS¹². Konkretno su korištene usluge AWS Lambda¹³, API Gateway¹⁴ i RDS Database¹⁵. API Gateway služi za preusmjeravanje HTTP zahtjeva s frontenda na AWS Lambdu u kojoj su implementirane funkcionalnosti backenda. AWS Lambda zatim preko istog API Gatewaya šalje odgovor frontendu.

Baza podataka nalazi se na poslužitelju Amazon RDS.

¹²Amazon AWS

¹³https://aws.amazon.com/lambda/?nc2=h_ql_prod_fs_lbd

¹⁴https://aws.amazon.com/api-gateway/?nc2=type_a

¹⁵https://aws.amazon.com/rds/?nc2=type_a

5.2 Ispitivanje programskog rješenja

dio 2. revizije

5.2.1 Ispitivanje komponenti

Ispitni slučaj 1: Registracija novog korisnika

- funkciji `register_handler` predaju se podaci o korisniku u JSON formatu
- funkcija stvara element klase `User` i poziva metodu `register`
- očekivani rezultat:
 - funkcija vraća poruku "SignupValid"
 - korisnik se upisuje u bazu podataka
- rezultat:

Data Output						
	userid [PK] integer	firstname character varying (50)	lastname character varying (50)	username character varying (50)	userpswd character varying (50)	userrole character varying (10)
1	41	Ime	Prezime	Username	password12345	zaposlenik

Slika 5.1: Rezultat u bazi podataka

```
1 from user import User
2 from document import Document
3 from registerLambda import register_handler
4
5 ##TEST 1
6 ##Uspjesna registracija korisnika
7 response=register_handler({
8     "name": "Ime",
9     "surname": "Prezime",
10    "username": "Username",
11    "password": "password12345",
12    "role": "zaposlenik"
13 }, "")
14
15 if response["message"]=="SignupValid":
16     print("TEST 1: PROLAZ")
17
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
PS D:\py\test> python -u "d:\py\test\test.py"
TEST 1: PROLAZ
PS D:\py\test>
```

Slika 5.2: Rezultat u razvojnom okruženju

Ispitni slučaj 2: Pokušaj registracije s već postojećim korisničkim imenom:

- funkciji `register_handler` predaju se podaci o korisniku u JSON formatu
- funkcija stvara element klase `User` i poziva metodu `register`
- očekivani rezultat:
 - funkcija vraća poruku `"SignupInvalid"`
 - korisnik se ne upisuje u bazu podataka
- rezultat:

Data Output

	userid [PK] integer	firstname character varying (50)	lastname character varying (50)	username character varying (50)	userpswd character varying (50)	userrole character varying (10)
1	41	Ime	Prezime	Username	password12345	zaposlenik

Slika 5.3: Potvrda rezultata u bazi podataka

```

19  ##TEST 2
20  ##Pokušaj registracije s već postojećim korisničkim imenom
21  response=register_handler({
22      "name":"Ime2",
23      "surname":"Prezime2",
24      "username":"Username",
25      "password":"password12345",
26      "role":"zaposlenik"
27  }, "")
28
29  if response["message"]=="SignupInvalid":
30      print("TEST 2: PROLAZ")
31
32

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```

PS D:\py\test> python -u "d:\py\test\test.py"
TEST 2: PROLAZ
PS D:\py\test>

```

Slika 5.4: Rezultat u razvojnom okruženju

Ispitni slučaj 3: Login postojećeg korisnika:

- funkciji `login_handler` predaju se podaci o korisniku u JSON formatu
- funkcija dohvaća korisnika iz baze koristeći korisničko ime
- očekivani rezultat:

- funkcija vraća poruku "LoginValid"
- uz poruku vraća i podatke o korisniku
- rezultat:

```
34  ##TEST 3
35  ##Login postojećeg korisnika
36  print(login_handler({
37      "username": "Username",
38      "password": "password12345"
39  }, ""))
40
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
PS D:\py\test> python -u "d:\py\test\test.py"
{'message': 'LoginValid', 'username': 'Username', 'ime': 'Ime', 'prezime': 'Prezime', 'role': 'zaposlenik', 'id': 41}
PS D:\py\test>
```

Slika 5.5: Rezultat u razvojnem okruženju

Ispitni slučaj 4: Login nepostojećeg korisnika:

- funkciji login_handler predaju se podaci o korisniku u JSON formatu
- funkcija pokušava dohvatiti korisnika iz baze koristeći korisničko ime
- očekivani rezultat:
 - funkcija vraća poruku "LoginInvalid"
 - uz poruku vraća i prazne podatke o korisniku
- rezultat:

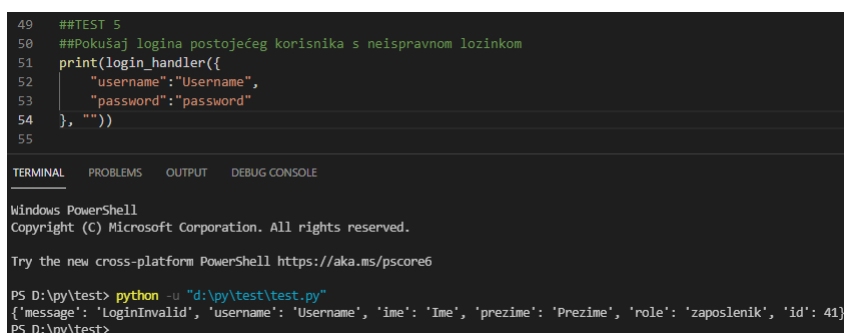
```
41  ##TEST 4
42  ##Login nepostojećeg korisnika
43  print(login_handler({
44      "username": "NepostojeciUsername",
45      "password": "password12345"
46  }, ""))
47
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
PS D:\py\test> python -u "d:\py\test\test.py"
{'message': 'LoginInvalid', 'username': '', 'ime': '', 'prezime': '', 'role': '', 'id': ''}
PS D:\py\test>
```

Slika 5.6: Rezultat u razvojnem okruženju

Ispitni slučaj 5: Pokušaj logina postojećeg korisnika s neispravnom lozinkom:

- funkciji login_handler predaju se podaci o korisniku u JSON formatu

- funkcija dohvaća korisnika iz baze podataka koristeći korisničko ime i uspoređuje lozinke
- očekivani rezultat:
 - funkcija vraća poruku "LoginInvalid"
 - uz poruku vraća i podatke o korisniku jer korisnik postoji, samo je lozinka neispravna
- rezultat:



```
49 ##TEST 5
50 ##Pokušaj logina postojećeg korisnika s neispravnom lozinkom
51 print(login_handler({
52     "username": "Username",
53     "password": "password"
54 }, ""))
55
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS D:\py\test> python -u "d:\py\test\test.py"
{'message': 'LoginInvalid', 'username': 'Username', 'ime': 'Ime', 'prezime': 'Prezime', 'role': 'zaposlenik', 'id': 41}
PS D:\py\test>

Slika 5.7: Rezultat u razvojnom okruženju

Ispitni slučaj 6: Dodavanje ispravno skeniranog dokumenta:

- funkciji `add_document_handler` predaju se podaci o dokumentu, datum skeniranja i ID korisnika koji je skenirao dokument
- funkcija stvara objekt klase `Document` i poziva metodu `addDocument`
- funkcija `addDocument` dodaje zapis o skeniranju u relacije "scanhistory", "documents" i, ako je dokument označen kao ispravan, u relaciju "reviserpending" ili "accountantpending"
- očekivani rezultat:
 - funkcija dodaje zapise o skeniranju dokumenta u relacije "scanhistory", "documents" i, pošto je korisnik koji je skenirao dokument zaposlenik, u relaciju "reviserpending"

- rezultat:

```

58  ##TEST 6
59  ##Dodavanje ispravno skeniranog dokumenta
60  add_document_handler({
61      "doctype": "R",
62      "doclabel": "R676767",
63      "doctext": "Testni dokument",
64      "uspjeh": "da",
65      "userid": "41",
66      "scandate": "10.1.2022."
67  }, "")
68

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

PS D:\py\test> python -u "d:\py\test\test.py"

PS D:\py\test>

Slika 5.8: Rezultat u razvojnom okruženju

Data Output

	docid [PK] integer	userid integer	scandate date
1	140	41	2022-10-01

Slika 5.9: Rezultat u relaciji "scanhistory"

Data Output

	docid [PK] integer	doclabel character varying (10)	doctype character varying (1)	doctext text
1	140	R676767	R	Testni dokument

Slika 5.10: Rezultat u relaciji "documents"

Data Output

	docid [PK] integer	userid integer
1	140	35

Slika 5.11: Rezultat u relaciji "reviserpending"

Ispitni slučaj 7: Dodavanje neispravno skeniranog dokumenta:

- funkciji `add_document_handler` predaju se podaci o dokumentu, datum skeniranja i ID korisnika koji je skenirao dokument
- funkcija stvara objekt klase `Document` i poziva metodu `addDocument`
- funkcija `addDocument` dodaje zapis o skeniranju u relacije "scanhistory", "documents" i, ako je dokument označen kao ispravan, u relaciju "reviserpending" ili "accountantpending"

- očekivani rezultat:
 - funkcija dodaje zapise o skeniranju dokumenta u relacije "scanhistory", "documents" i, pošto je dokument označen kao pogrešno skeniran, ne dodaje ga u relaciju "reviserpending"
- rezultat:
 - funkcija je prošla test

```

70  ##TEST 7
71  ##Dodavanje neispravno skeniranog dokumenta
72  add_document_handler({
73      "doctype": "R",
74      "doclabel": "R676777",
75      "doctext": "Testni dokument 2",
76      "uspjeh": "ne",
77      "userid": "41",
78      "scandate": "10.1.2022."
79  }, "")
80

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```

PS D:\py\test> python -u "d:\py\test\test.py"
PS D:\py\test> python -u "d:\py\test\test.py"
PS D:\py\test>

```

Slika 5.12: Rezultat u razvojnem okruženju

Data Output			
	docid [PK] integer	userid integer	scandate date
1	141	41	2022-10-01

Slika 5.13: Rezultat u relaciji "scanhistory"

Data Output				
	docid [PK] integer	doclabel character varying (10)	doctype character varying (1)	doctext text
1	141	R676777	R	Testni dokument 2

Slika 5.14: Rezultat u relaciji "documents"

Data Output	
docid [PK] integer	userid integer

Slika 5.15: Rezultat u relaciji "reviserpending"

Ispitni slučaj 8: Slučajno dodavanje dokumenta kao ispravnog iako je označeno da je pogrešno skenirano:

- funkciji `add_document_handler` predaju se podaci o dokumentu, datum skeniranja i ID korisnika koji je skenirao dokument
- funkcija stvara objekt klase `Document` i poziva metodu `addDocument`
- funkcija `addDocument` dodaje zapis o skeniranju u relacije "scanhistory", "documents" i, ako je dokument označen kao ispravan, u relaciju "reviserpending" ili "accountantpending"
- pošto je oznaka dokumenta pogrešno skenirana, frontend aplikacije backendu šalje prazan string kao atribut "doclabel"
- očekivani rezultat:
 - funkcija dodaje zapise o skeniranju dokumenta u relacije "scanhistory", "documents" i, unatoč tome što je dokument označen kao ispravno skeniran, ne dodaje ga u relaciju "reviserpending" jer je "doclabel" prazan string
 - vrijednosti "doclabel" i "doctype" su u relaciji document postavljene kao null vrijednosti
- rezultat:
 - funkcija je prošla test

```

82  ##TEST 8
83  ##Dodavanje neispravno skeniranog dokumenta kao ispravnog
84  add_document_handler({
85      "doctype": "R",
86      "doclabel": "",
87      "doctext": "Testni dokument 3",
88      "uspjeh": "da",
89      "userid": "41",
90      "scandate": "10.1.2022."
91  }, "")
92

```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```

PS D:\py\test> python -u "d:\py\test\test.py"
PS D:\py\test>

```

Slika 5.16: Rezultat u razvojnom okruženju

Data Output			
	docid [PK] integer	userid integer	scandate date
1		143	41 2022-10-01

Slika 5.17: Rezultat u relaciji "scanhistory"

Data Output

	docid [PK] integer	doclabel character varying (10)	doctype character varying (1)	doctext text
1	143	[null]	[null]	Testni dokument 3

Slika 5.18: Rezultat u relaciji "documents"

Data Output

	docid [PK] integer	userid integer

Slika 5.19: Rezultat u relaciji "reviserpending"

5.2.2 Ispitivanje sustava

Testiranje logina

Testiramo može li se korisnik prijaviti u aplikaciju.

```
@LargeTest
@RunWith(AndroidJUnit4.class)
public class login_radnik_test {

    @Rule
    public ActivityTestRule<MainActivity> mActivityTestRule = new
        ActivityTestRule<>(MainActivity.class);

    @Test
    public void login_radnik_test() {
        ViewInteraction appCompatEditText = onView(
            allOf(withId(R.id.etUsername),
                childAtPosition(
                    allOf(withId(R.id.LogInText),
                        childAtPosition(
                            withId(android.R.id.content),
                                0)),
                            1),
                isDisplayed()));
        appCompatEditText.perform(replaceText("radnik"),
            closeSoftKeyboard());
```

```
ViewInteraction appCompatEditText2 = onView(
    allOf(withId(R.id.etPassword),
        childAtPosition(
            allOf(withId(R.id.LoginText),
                childAtPosition(
                    withId(android.R.id.content),
                        0)),
                    2),
            isDisplayed())));
appCompatEditText2.perform(replaceText("rad1234"),
    closeSoftKeyboard());

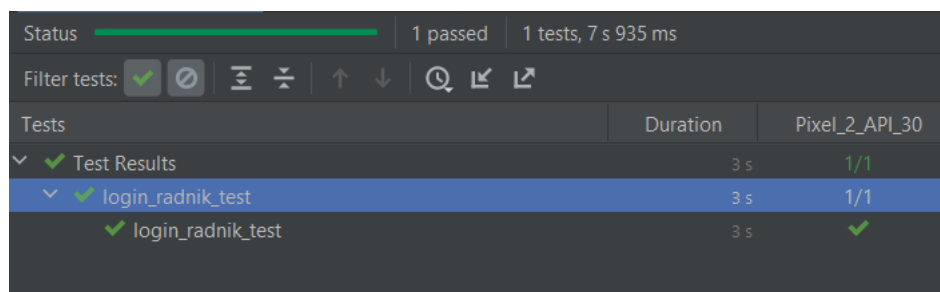
ViewInteraction appCompatEditText3 = onView(
    allOf(withId(R.id.etPassword), withText("rad1234"),
        childAtPosition(
            allOf(withId(R.id.LoginText),
                childAtPosition(
                    withId(android.R.id.content),
                        0)),
                    2),
            isDisplayed())));
appCompatEditText3.perform(pressImeActionButton());

ViewInteraction materialButton = onView(
    allOf(withId(R.id.LoginBtn), withText("Log In"),
        childAtPosition(
            allOf(withId(R.id.LoginText),
                childAtPosition(
                    withId(android.R.id.content),
                        0)),
                    3),
            isDisplayed())));
materialButton.perform(click());
}

private static Matcher<View> childAtPosition(
    final Matcher<View> parentMatcher, final int position) {
```

```
return new TypeSafeMatcher<View>() {  
    @Override  
    public void describeTo(Description description) {  
        description.appendText("Child at position " +  
            position + " in parent ");  
        parentMatcher.describeTo(description);  
    }  
  
    @Override  
    public boolean matchesSafely(View view) {  
        ViewParent parent = view.getParent();  
        return parent instanceof ViewGroup &&  
            parentMatcher.matches(parent)  
            && view.equals(((ViewGroup)  
                parent).getChildAt(position));  
    }  
};  
}
```

Ako je login ispravan, proći ćemo test.



Slika 5.20: "Test logina"

Testiranje prisutnosti slike u ImageView-u

Testiramo je li slika prisutna u ImageView-u.

```
@LargeTest
@RunWith(AndroidJUnit4.class)
public class prikazuje_sliku_test {

    @Rule
    public ActivityTestRule<MainActivity> mActivityTestRule = new
        ActivityTestRule<>(MainActivity.class);

    @Rule
    public GrantPermissionRule mGrantPermissionRule =
        GrantPermissionRule.grant(
            "android.permission.CAMERA");

    @Test
    public void prikazuje_sliku_test() {
        ViewInteraction appCompatEditText = onView(
            allOf(withId(R.id.etUsername),
                childAtPosition(
                    allOf(withId(R.id.LogInText),
                        childAtPosition(
                            withId(android.R.id.content),
                                0)),
                            1),
                    isDisplayed())));
        appCompatEditText.perform(replaceText("radnik"),
            closeSoftKeyboard());

        ViewInteraction appCompatEditText2 = onView(
            allOf(withId(R.id.etPassword),
                childAtPosition(
                    allOf(withId(R.id.LogInText),
                        childAtPosition(
                            withId(android.R.id.content),
                                0)),
                            2),
```

```
isDisplayed());  
appCompatEditText2.perform(replaceText("rad1234"),  
    closeSoftKeyboard());
```

```
ViewInteraction materialButton = onView(  
    allOf(withId(R.id.LoginBtn), withText("Log In"),  
        childAtPosition(  
            allOf(withId(R.id.LogInText),  
                childAtPosition(  
                    withId(android.R.id.content),  
                    0)),  
                3),  
            isDisplayed()));  
materialButton.perform(click());
```

```
ViewInteraction materialButton2 = onView(  
    allOf(withId(R.id.ScanZap), withText("Skeniranje"),  
        childAtPosition(  
            childAtPosition(  
                withId(android.R.id.content),  
                0),  
                0),  
            isDisplayed()));  
materialButton2.perform(click());
```

```
try {  
    TimeUnit.SECONDS.sleep(2);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}
```

```
ViewInteraction materialButton3 =  
    onView(withId(R.id.SlikajButton));  
materialButton3.perform(click());  
try {  
    TimeUnit.SECONDS.sleep(2);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}
```

```

    }
    onView(withId(R.id.SlikaWind)).check(matches(notNullValue()));
}

private static Matcher<View> childAtPosition(
    final Matcher<View> parentMatcher, final int position) {

    return new TypeSafeMatcher<View>() {
        @Override
        public void describeTo(Description description) {
            description.appendText("Child at position " + position + " in
                parent ");
            parentMatcher.describeTo(description);
        }

        @Override
        public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup &&
                parentMatcher.matches(parent)
                && view.equals(((ViewGroup) parent).getChildAt(position));
        }
    };
}
}
}

```

Ako je slika u ImageView-u, proći ćemo test.

Status ████████████████████ 1 passed 1 tests, 13 s 941 ms		
Filter tests: ✓ ⊗ ≡ ⚡ ↑ ↓ 🔍 🔗 🔗		
Tests	Duration	Pixel_2_API_30
✓ Test Results	9 s	1/1
✓ prikazuje_sliku_test	9 s	1/1
✓ prikazuje_sliku_test	9 s	✓

Slika 5.21: "Test prisutnosti slike u ImageView-u"

Testiranje ispravnog skena dokumenta

Testiramo je li skenirani dokument ispravan. Ispravan dokument je:

Internal file in our app

INT9852

```
@LargeTest
```

```
@RunWith(AndroidJUnit4.class)
```

```
public class ispravanSkenTest {
```

```
    @Rule
```

```
    public ActivityTestRule<MainActivity> mActivityTestRule = new  
        ActivityTestRule<>(MainActivity.class);
```

```
    @Rule
```

```
    public GrantPermissionRule mGrantPermissionRule =  
        GrantPermissionRule.grant(  
            "android.permission.CAMERA");
```

```
    @Test
```

```
    public void ispravanSkenTest() {  
        ViewInteraction appCompatEditText = onView(  
            allOf(withId(R.id.etUsername),  
                childAtPosition(  
                    allOf(withId(R.id.LogInText),  
                        childAtPosition(  
                            withId(android.R.id.content),  
                                0)),  
                            1),  
                    isDisplayed())));  
        appCompatEditText.perform(replaceText("user12"),  
                                closeSoftKeyboard());
```

```
        ViewInteraction appCompatEditText2 = onView(  
            allOf(withId(R.id.etPassword),  
                childAtPosition(  
                    allOf(withId(R.id.LogInText),  
                        childAtPosition(  
                            withId(android.R.id.content),
```



```
0)),
2),
isDisplayed()));
appCompatEditText2.perform(replaceText("user12"),
    closeSoftKeyboard());

ViewInteraction materialButton = onView(
    allOf(withId(R.id.LoginBtn), withText("Log In"),
        childAtPosition(
            allOf(withId(R.id.LogInText),
                childAtPosition(
                    withId(android.R.id.content),
                        0)),
                    3),
            isDisplayed()));
materialButton.perform(click());

ViewInteraction materialButton2 = onView(
    allOf(withId(R.id.ScanZap), withText("Skeniranje"),
        childAtPosition(
            childAtPosition(
                withId(android.R.id.content),
                    0),
                    0),
            isDisplayed()));
materialButton2.perform(click());
try {
    TimeUnit.SECONDS.sleep(10);
} catch (InterruptedException e) {
    e.printStackTrace();
}
ViewInteraction materialButton3 = onView(
    allOf(withId(R.id.SlikajButton), withText("SLIKAJ"),
        childAtPosition(
            childAtPosition(
                withId(android.R.id.content),
                    0),
                    1),
```

```
isDisplayed()));  
materialButton3.perform(click());  
try {  
    TimeUnit.SECONDS.sleep(3);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}  
  
ViewInteraction materialButton4 = onView(  
    allOf(withId(R.id.SkenirajSlikuButton), withText("Skeniraj Sliku"),  
    childAtPosition(  
        childAtPosition(  
            withId(android.R.id.content),  
            0),  
            2),  
    isDisplayed()));  
materialButton4.perform(click());  
try {  
    TimeUnit.SECONDS.sleep(3);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}  
  
ViewInteraction textView = onView(  
    allOf(withId(R.id.SkeniraniTekst),  
    withParent(withParent(withId(android.R.id.content))),  
    isDisplayed()));  
textView.check(matches(withText("Internal file in our  
    app\nINT9852\n"))));  
try {  
    TimeUnit.SECONDS.sleep(3);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}  
  
ViewInteraction materialButton5 = onView(  
    allOf(withId(R.id.IspravanScanButton), withText("Ispravan Scan"),  
    childAtPosition(  
        childAtPosition(  
            withId(android.R.id.content),
```

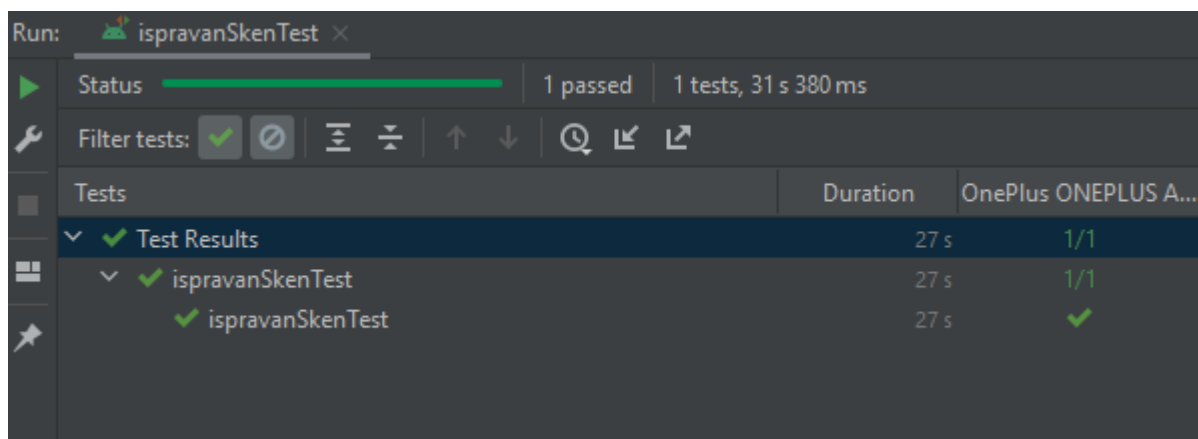
```
0),
1),
isDisplayed()));
materialButton5.perform(click());
try {
    TimeUnit.SECONDS.sleep(3);
} catch (InterruptedException e) {
    e.printStackTrace();
}
}

private static Matcher<View> childAtPosition(
    final Matcher<View> parentMatcher, final int position) {

    return new TypeSafeMatcher<View>() {
        @Override
        public void describeTo(Description description) {
            description.appendText("Child at position " + position + " in
                parent ");
            parentMatcher.describeTo(description);
        }

        @Override
        public boolean matchesSafely(View view) {
            ViewParent parent = view.getParent();
            return parent instanceof ViewGroup &&
                parentMatcher.matches(parent)
                && view.equals(((ViewGroup) parent).getChildAt(position));
        }
    };
}
```

Ako je skenirani dokument ispravan, proći ćemo test.



Slika 5.22: "Test ispravnosti skeniranog dokumenta"

Testiranje neispravnog skena dokumenta

Testiramo je li skenirani dokument neispravan. Ispravan dokument je:

Internal file in our app

INT9852

```
@LargeTest
@RunWith(AndroidJUnit4.class)
public class neispravanSkenTest {

    @Rule
    public ActivityTestRule<MainActivity> mActivityTestRule = new
        ActivityTestRule<>(MainActivity.class);

    @Rule
    public GrantPermissionRule mGrantPermissionRule =
        GrantPermissionRule.grant(
            "android.permission.CAMERA");

    @Test
    public void neispravanSkenTest() {
        ViewInteraction appCompatEditText = onView(
            allOf(withId(R.id.etUsername),
                childAtPosition(
                    allOf(withId(R.id.LogInText),
                        childAtPosition(
                            withId(android.R.id.content),
                                0)),
                            1),
                isDisplayed())));
        appCompatEditText.perform(replaceText("user12"),
            closeSoftKeyboard());

        ViewInteraction appCompatEditText2 = onView(
            allOf(withId(R.id.etPassword),
                childAtPosition(
                    allOf(withId(R.id.LogInText),
                        childAtPosition(
                            withId(android.R.id.content),
```

```
0)),
2),
isDisplayed()));
appCompatEditText2.perform(replaceText("user12"),
    closeSoftKeyboard());

ViewInteraction appCompatEditText3 = onView(
    allOf(withId(R.id.etPassword), withText("user12"),
        childAtPosition(
            allOf(withId(R.id.LogInText),
                childAtPosition(
                    withId(android.R.id.content),
                    0)),
                2),
            isDisplayed()));
appCompatEditText3.perform(pressImeActionButton());

ViewInteraction materialButton = onView(
    allOf(withId(R.id.LoginBtn), withText("Log In"),
        childAtPosition(
            allOf(withId(R.id.LogInText),
                childAtPosition(
                    withId(android.R.id.content),
                    0)),
                3),
            isDisplayed()));
materialButton.perform(click());

ViewInteraction materialButton2 = onView(
    allOf(withId(R.id.ScanZap), withText("Skeniranje"),
        childAtPosition(
            childAtPosition(
                withId(android.R.id.content),
                0),
                0),
            isDisplayed()));
materialButton2.perform(click());
try {
```

```
        TimeUnit.SECONDS.sleep(10);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    ViewInteraction materialButton3 = onView(
        allOf(withId(R.id.SlikajButton), withText("SLIKAJ"),
            childAtPosition(
                childAtPosition(
                    withId(android.R.id.content),
                    0),
                    1),
                isDisplayed())));
    materialButton3.perform(click());
    try {
        TimeUnit.SECONDS.sleep(3);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    ViewInteraction materialButton4 = onView(
        allOf(withId(R.id.SkenirajSlikuButton), withText("Skeniraj Sliku"),
            childAtPosition(
                childAtPosition(
                    withId(android.R.id.content),
                    0),
                    2),
                isDisplayed())));
    materialButton4.perform(click());
    try {
        TimeUnit.SECONDS.sleep(3);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    ViewInteraction textView = onView(
        allOf(withId(R.id.SkeniraniTekst),
            withParent(withParent(withId(android.R.id.content))),
            isDisplayed()));
```

```
textView.check(matches(not(withText("Internal file in our
    app\nINT9852\n"))));
try {
    TimeUnit.SECONDS.sleep(3);
} catch (InterruptedException e) {
    e.printStackTrace();
}

ViewInteraction materialButton5 = onView(
    allOf(withId(R.id.NeIspravanScanButton), withText("NeIspravan
        Scan"),
        childAtPosition(
            childAtPosition(
                withId(android.R.id.content),
                0),
                0),
            isDisplayed())));
materialButton5.perform(click());
try {
    TimeUnit.SECONDS.sleep(3);
} catch (InterruptedException e) {
    e.printStackTrace();
}
}

private static Matcher<View> childAtPosition(
    final Matcher<View> parentMatcher, final int position) {

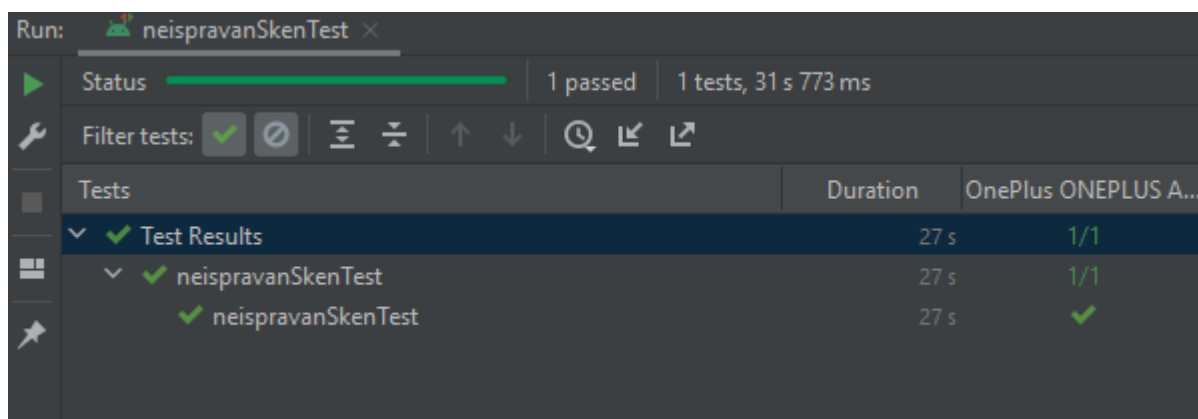
    return new TypeSafeMatcher<View>() {
        @Override
        public void describeTo(Description description) {
            description.appendText("Child at position " + position + " in
                parent ");
            parentMatcher.describeTo(description);
        }

        @Override
        public boolean matchesSafely(View view) {
```



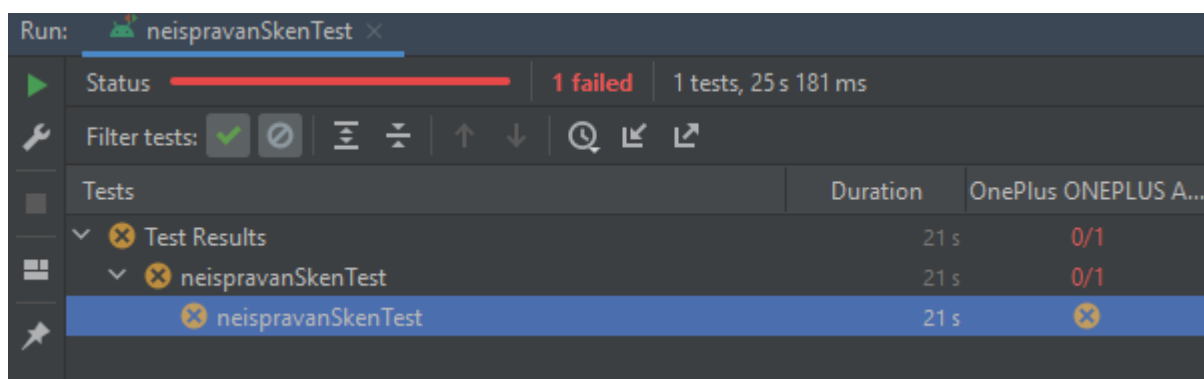
```
ViewParent parent = view.getParent();
return parent instanceof ViewGroup &&
    parentMatcher.matches(parent)
    && view.equals(((ViewGroup) parent).getChildAt(position));
}
};
}
}
```

Ako je skenirani dokument neispravan, npr. neki račun, proći ćemo test.



Slika 5.23: "Test neispravnosti skeniranog dokumenta"

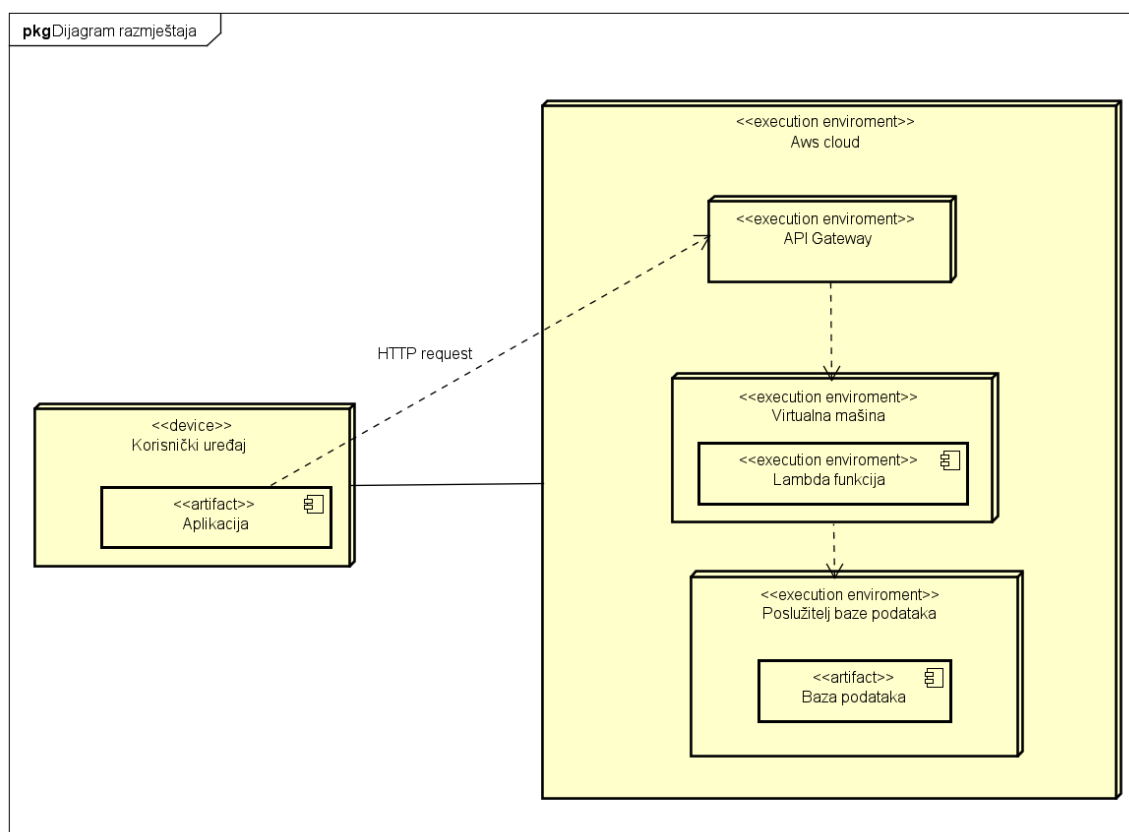
Ako bi kojim slučajem skenirali traženi dokument, rezultat ne bi prošao test.



Slika 5.24: "Test neispravnosti skeniranog dokumenta"

5.3 Dijagram razmještaja

Klijent preko svojeg uređaja pristupa aplikaciji. Aplikacija se spaja na API Gateway koji je u sklopu Awsovog clouda. Unutar clouda se nalazi još virtualna mašina koja pokreće odgovarajuće lambda funkcije koje manipuliraju s bazom podataka koja je također dio clouda.



Slika 5.25: Dijagram razmještaja

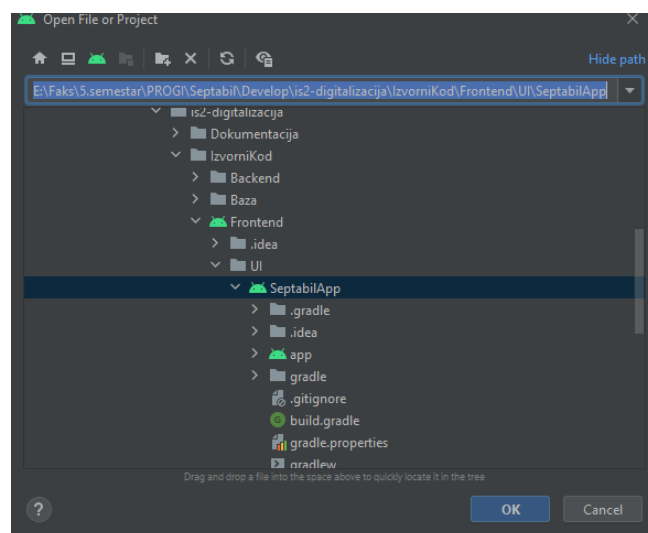
5.4 Upute za puštanje u pogon

dio 2. revizije

5.4.1 Izgradnja aplikacije

Kako bi se aplikacija izgradila, mora se prvo otvoriti u programu Android Studio, njezina putanja je :

- is2-digitalizacija\IzvorniKod\Frontend\UI\SeptabilApp.

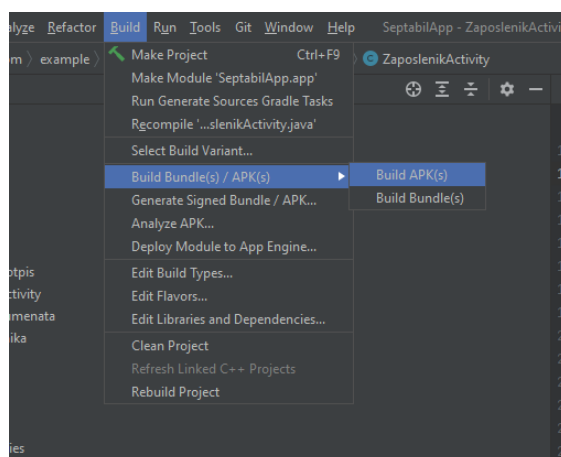


Slika 5.26: Android Studio - Path

Nakon što se aplikacija otvori, treba malo pričekati da Gradle učita sve potrebne pakete za rad i onda kada je sve učitano, korisnik treba kliknuti

- Build → Build Bundle(s) / APK(s) → Build APK

kako bi napravio APK naše aplikacije.



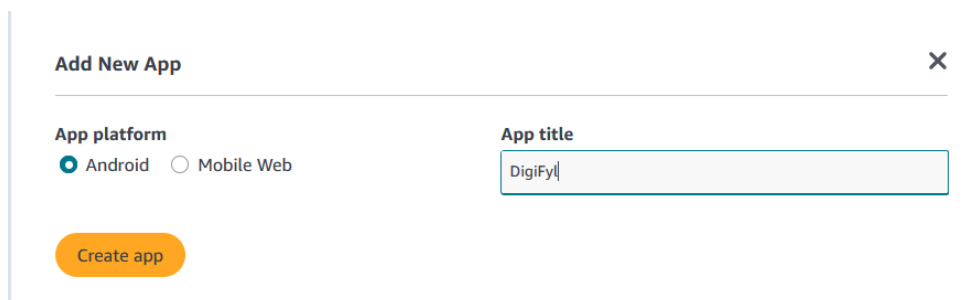
Slika 5.27: Android Studio - Build APK

5.4.2 Postavljanje aplikacije na Amazon App trgovinu

Za postavljanje aplikacije na Amazon App trgovinu, prvo smo se morali registrirati na Amazon Developer. Nakon toga treba odabrati

- Apps and Services → My apps

kliknuti gumb Add New App, odabrati želimo li mobilnu ili web aplikaciju i upisati njezino ime.



The screenshot shows a modal dialog box titled "Add New App" with a close button (X) in the top right corner. Inside the dialog, there are two sections. The first section, "App platform", contains two radio buttons: "Android" (which is selected) and "Mobile Web". The second section, "App title", contains a text input field with the text "DigiFyl" entered. At the bottom left of the dialog is an orange button labeled "Create app".

Slika 5.28: Amazon Developer - Add New App

Kada odaberemo naziv aplikacije, otvorit će nam se prozor koji ima 6 dijelova za ispuniti, prvi dio su osnovne informacije o aplikaciji poput naslova, kategorije, kontakta...

The screenshot shows the 'General Information' tab of the Amazon App Store submission process. At the top, a progress bar indicates 'App version in progress' and '5/6 complete'. A 'Submit App' button is in the top right. The navigation bar includes tabs for General Information, Availability & Pricing, Description, Images & Multimedia, Content Rating, and APK Files. The main form contains the following fields:

- App title ***: Text input with 'DigiFyl' entered.
- App SKU**: Text input, currently empty.
- App Submission API Keys**: A section containing:
 - App ID**: 'amzn1.devportal.mobileapp.251b847af4454c03bac70acfdbe0611c' with a 'Copy' button.
 - Release ID**: 'amzn1.devportal.apprelease.7d72f6bb060548c2bed07f5e37cc42ec' with a 'Copy' button.
- App category ***: A dropdown menu showing 'Business' and a 'Choose Category' button.
- App will be listed in:** A box showing 'Category Business'.
- Customer support contact**: A section with a checked box for 'Use my default support information' and three input fields:
 - Customer support email address ***: 'grupaseptabil@gmail.com'.
 - Customer support phone**: 'Enter phone number'.
 - Customer support website**: 'Enter website'.

At the bottom right, there are 'Cancel' and 'Save' buttons.

Slika 5.29: App - General Information

Drugi dio nas samo pita gdje sve želimo da aplikacija bude dostupna u svijetu i želimo li ju izdati kao besplatnu ili naplaćivati.

Treći dio je opis naše aplikacije, on će biti prikazan kao opis na Amazon App trgovini.

General Information

Availability & Pricing

Description

Images & Multimedia

Content Rating

APK Files

English (U.S.)*

English (U.S.) description will be displayed in all marketplaces if translations are not provided.

English (U.S.)*

Display title* ⓘ
English (U.S.)

DigiFyl

Short description*
English (U.S.)

DigiFyl is our college project app, it is supposed to represent a digital company and how a document gets approved, archived, signed by the manager etc.

Long description*
English (U.S.)

DigiFyl is a app that represents a company, that company has 4 tiers of employees, the regular employee, the reviser, the accountant and the manager. Every employee (including reviser, accountant and manager) can scan a document, that document goes to the reviser who checks it and follows it through to the accountant. There are 3 tiers of accountants because there are 3 tiers of documents, invoices, offers and internal documents, each accountant handles one type of documents. The accountant can archive the file or send it to the manager for his signature and then archive it.

Product feature bullets*
English (U.S.)

- College project
- Simulation of a digital company
- Scan documents with mobile camera and share them between the ranks of employees

Keywords
English (U.S.)

—

Search terms used to increase the discoverability of your app. Use a comma or white space to separate your terms.

Edit

Slika 5.30: App - Description

Četvrti dio od nas traži da postavimo ikonu aplikacije i priložimo 3-10 slika aplikacije, za ikonu smo u ovom slučaju koristili avion koji smo našli na stranici Icon Archive.

Appstore Assets
For detail on each image, refer to the [Image asset guidelines](#).

☐ Preview images on tablet

Icons
English (U.S.)
512 x 512px PNG (with transparency)
114 x 114px PNG (with transparency)



Screenshots
English (U.S.)
Between 3 and 10 PNGs or JPGs
800 x 480px, 1024 x 600px, 1280 x 720px,
1280 x 800px, 1920 x 1080px, 1920 x 1200px,
or 2560 x 1600px (portrait or landscape)



Promotional Image
English (U.S.)
1024 x 500px (landscape only) PNG or JPG

No image uploaded

Video
English (U.S.)
Up to 5 MPEG-2, WMV, MOV, FLV, AVI, or H.264 MPEG-4, 720 -
1080px wide (4:3 or 16:9); 1200 kbps or higher

No video uploaded

Fire TV/Automotive Assets
Required only if you are targeting Fire TV/Automotive devices. See [Image asset guidelines](#).

Slika 5.31: App - Images and multimedia

U petom dijelu moramo napraviti "Content Rating" aplikacije, napisati ima li u njoj nasilja, netolerancije, je li ona u akademske svrhe ili ne...

Pita nas i da odaberemo ciljanu publiku te prikuplja li aplikacija osobne podatke. Zadnje stvari koje nas pita je ima li u aplikaciji reklama, kockanja, prikupljamo li lokacije korisnika i ima li komunikacije među korisnicima.

Kako prikupljamo osobne podatke korisnika i oni mogu slati podatke jedni drugima u obliku skeniranog dokumenta, to smo označili s da.

6. Intolerance*
Any disparagement of race, creed, culture, or religion.



7. Profanity*
Profanity or crude humor



8. Academic*
This application is for educational purposes



No



Yes

Additional Information

1. Target audience for your app*

☐ 0-12 years of age

☐ 13-15 years of age

☒ 16-17 years of age

☒ 18+ years of age

If the target audience for your app includes these age groups, you must comply with our [Child-Directed App Policy](#). If you use Amazon Advertising or Associates programs, by selecting these age groups you certify that you will not deliver advertising to anyone you know is a child, or child-directed areas of your app.

2. Account creation or other personal information collected?*



No



Yes

3. Advertisements*



No



Yes

4. Gambling*



No



Yes

5. Location detection or Location Based Services*



No



Yes

6. User Generated Content or User to User Communication*



No



Yes

Privacy policy URL*
Required if app collects personal information

<https://www.freeprivacypolicy.com/live/a30b05f0-7df1-43e7-8bdf-12020a3489a4>

Characters remaining: 175/250

Slika 5.32: App - Images and multimedia

Na kraju, s obzirom na to da skupljamo podatke od korisnika, morali smo napraviti politiku privatnosti na kojoj piše koje sve podatke prikupljamo, kako ih koristimo i kako nas se može kontaktirati s bilo kojim dodatnim pitanjima. Priložit ćemo ju i ovdje: [link](#)

Zadnji dio u deployanju naše aplikacije je odabrati APK datoteku koju predajemo, podržane jezike aplikacije i dodatno napisati upute za testiranje ako će trebati Amazonovom timu, a moramo i odabrati da smo suglasni da se aplikacija uvozi i izvozi iz SAD-a u druge države u kojima Amazon radi.

Add APK[Public Key](#) | [Appstore Certificate Hashes](#)

APK File*
Upload your APK files one by one.
As part of the ingestion process, Amazon removes your developer signature and applies an Amazon signature. This signature is unique to you, does not change, and is the same for all apps in your account.

+

Drop APK here

Add Other Details

Language Support*

☐ Arabic

☐ Chinese

☐ Cornish

☐ Czech

☐ Dutch

☒ English

☐ French

☐ German

☐ Greek

☐ Hebrew

☐ Hindi

☐ Italian

☐ Japanese

☐ Kazakh

☐ Korean

☐ Polish

☐ Portuguese

☐ Russian

☐ Spanish

☐ Tagalog

☐ Vietnamese

Testing Instructions
If needed, provide special instructions to the Amazon testing team, such as required login credentials to test video streams, or notes that explain corrections to previous issues.

Maximum characters: 4000

Export Compliance*

☒ I certify this app may be imported to and exported from the United States and all other countries and regions in which Amazon operates the program or in which we have authorized sales to end users (without the need for Amazon to obtain any license or clearance or take any other action) and is in full compliance with all applicable laws and regulations governing imports and exports, including those applicable to software that makes use of encryption technology.

Use Amazon Maps Redirection

☐ Amazon devices do not support the Google Maps API. However, the Amazon Maps API provides interface parity with the Google Maps v1 API. By using Amazon Maps Redirection, you can automatically redirect Google Maps v1 API calls from your app to the Amazon Maps API, providing a seamless maps experience in your app with no coding changes. The redirection occurs ONLY on Amazon devices. Portions of the Amazon Maps API are provided by HERE North America, LLC ("HERE"). Your use of Amazon Maps Redirection is subject to the terms of the [Amazon Maps Schedule](#), including the [HERE Materials Terms and Conditions](#).

Cancel

Save

Slika 5.33: App - APK Files

5.4.3 Postavljanje baze podataka na Amazon RDS-u

Kako bi postavili bazu podataka, morali smo otići na Amazon RDS, odabrati "Databases" i kliknuti "Create Database".

Tad nam se prikaže izbornik koji nas pita za vrstu baze podataka, mi smo radili s PostgreSQL bazom.

Choose a database creation method [Info](#)

☒ **Standard create**
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

☐ **Easy create**
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type [Info](#)

☐ Amazon Aurora

☐ MySQL

☐ MariaDB

☒ PostgreSQL

☐ Oracle

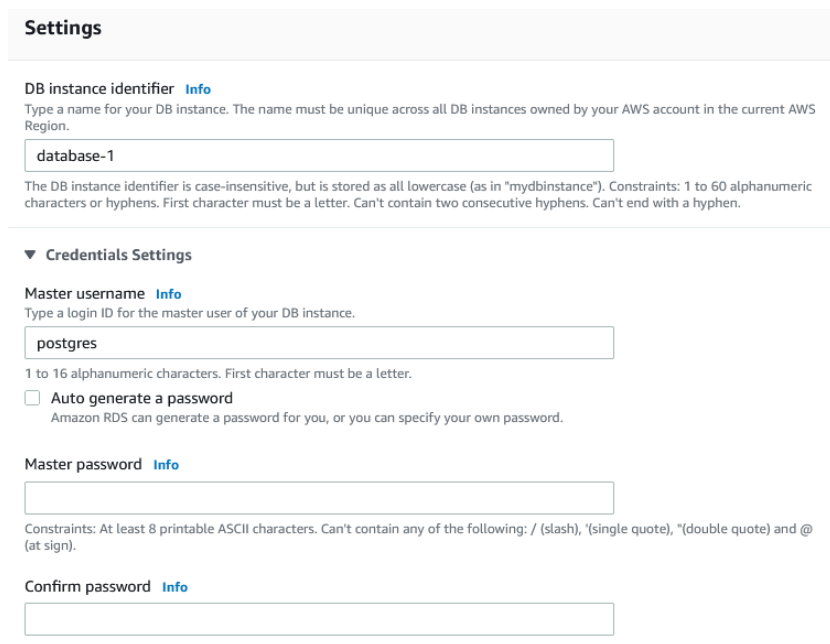
☐ Microsoft SQL Server

Version

PostgreSQL 12.8-R1 ▼

Slika 5.34: Amazon RDS - DB Setup

Nakon odabira baze podataka, RDS traži da upišemo ime instance baze podataka te da joj priložimo korisničko ime i lozinku kojom će joj se pristupati.



The screenshot shows the 'Settings' tab for an Amazon RDS DB instance. It includes fields for the 'DB instance identifier' (set to 'database-1'), 'Master username' (set to 'postgres'), and 'Master password'. There is a checkbox for 'Auto generate a password' which is currently unchecked. The 'Confirm password' field is also visible.

Settings

DB instance identifier [Info](#)
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

Master username [Info](#)
Type a login ID for the master user of your DB instance.

postgres

1 to 16 alphanumeric characters. First character must be a letter.

☐ **Auto generate a password**
Amazon RDS can generate a password for you, or you can specify your own password.

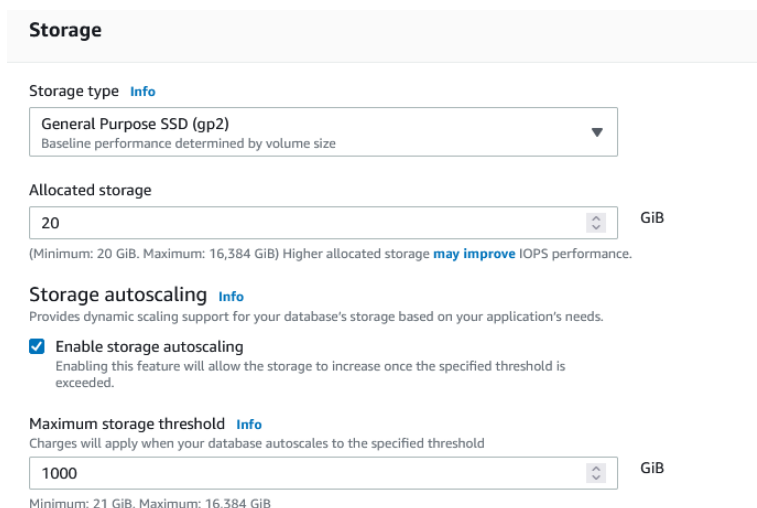
Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), ' (single quote), " (double quote) and @ (at sign).

Confirm password [Info](#)

Slika 5.35: Amazon RDS - DB Instance

Sljedeće što nas RDS pita je koliko nam memorije treba za bazu, minimalan odabir je 20 GB, a nama ni ne treba više pa smo odabrali to, ali smo upalili automatsko skaliranje skladišta, dakle ako nam treba više od 20 GB u nekom trenutku, dobit ćemo tu memoriju.



The screenshot shows the 'Storage' tab for an Amazon RDS DB instance. It includes a dropdown for 'Storage type' (set to 'General Purpose SSD (gp2)'), a field for 'Allocated storage' (set to 20 GiB), and a checkbox for 'Enable storage autoscaling' which is checked. There is also a field for 'Maximum storage threshold' (set to 1000 GiB).

Storage

Storage type [Info](#)

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage

20 GiB

(Minimum: 20 GiB. Maximum: 16,384 GiB) Higher allocated storage [may improve](#) IOPS performance.

Storage autoscaling [Info](#)
Provides dynamic scaling support for your database's storage based on your application's needs.

☒ **Enable storage autoscaling**
Enabling this feature will allow the storage to increase once the specified threshold is exceeded.

Maximum storage threshold [Info](#)
Charges will apply when your database autoscales to the specified threshold

1000 GiB

Minimum: 21 GiB. Maximum: 16,384 GiB

Slika 5.36: Amazon RDS - DB Storage

Sljedeći obrazac se tiče povezivosti, stvaramo virtualni oblak za našu bazu i grupu pod mreža koje baza može koristiti (što smo postavili na sve), također postavljamo i javni pristup našoj bazi kako bi joj mogli pristupiti iz pgAdmin-a.

Connectivity

Virtual private cloud (VPC) [Info](#)
VPC that defines the virtual networking environment for this DB instance.

Default VPC (vpc-0e21b99487f955937)

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

Subnet group [Info](#)
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default-vpc-0e21b99487f955937

Public access [Info](#)

☒ **Yes**
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

☐ **No**
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

☒ **Choose existing**
Choose existing VPC security groups

☐ **Create new**
Create new VPC security group

Existing VPC security groups

Choose VPC security groups

default X

Availability Zone [Info](#)

eu-central-1b

Slika 5.37: Amazon RDS - Connectivity

Zadnji obrazac nas pita za ime baze podataka i vrijeme zadržavanja sigurnosnih kopija prije brisanja.

▼ **Additional configuration**

Database options, backup enabled, backtrack disabled, Performance Insights enabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled.

Database options

Initial database name [Info](#)

IS2_Digitalizacija

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)

default.postgres12 ▼

Option group [Info](#)

default:postgres-12 ▼

Backup

☒ **Enable automated backups**
Creates a point-in-time snapshot of your database.

Backup retention period [Info](#)
Choose the number of days that RDS should retain automatic backups for this instance.

7 days ▼

Backup window [Info](#)
Select the period for which you want automated backups of the database to be created by Amazon RDS.

☐ Select window

☒ No preference

☒ Copy tags to snapshots

Backup replication [Info](#)

☐ **Enable replication in another AWS Region**
Enabling replication automatically creates backups of your DB instance in the selected Region, for disaster recovery, in addition to the current Region.

Slika 5.38: Amazon RDS - DB Name and Backup

Nakon što stvorimo bazu na Amazon RDS-u, dobijemo krajnju točku kako bi se na bazu mogli spojiti iz pgAdmin-a i Python-a. Za spajanje nam trebaju i korisničko ime i lozinka, ali to smo postavili pri stvaranju baze.

Connectivity & security		
Endpoint & port	Networking	Security
Endpoint is2digitalizacijav2.cwck16atirhq.eu-central-1.rds.amazonaws.com	Availability Zone eu-central-1a	VPC security groups default (sg-0be9f6b6d6053a887) ✔ Active
Port 5432	VPC vpc-0e21b99487f955937	Publicly accessible Yes
	Subnet group default-vpc-0e21b99487f955937	Certificate authority rds-ca-2019
	Subnets subnet-05c2f6692776c32d3 subnet-0272b7b571419f4e5 subnet-06c5d6e3767df2049	Certificate authority date August 22, 2024, 07:08 (UTC±7:08)

Slika 5.39: Amazon RDS - DB Endpoint

6. Zaključak i budući rad

Zadatak naše grupe bio je izrada mobilne aplikacije za Android operacijski sustav koja bi ubrzala digitalizaciju računovodstvenim tvrtkama. Izrada aplikacije završena je u zadanom roku, a postupak izrade aplikacije se odvijao u 2 ciklusa.

U prvom ciklusu nam je cilj bio napraviti jednostavnu verziju aplikacije kako bismo se upoznali s novim tehnologijama i alatima. Nakon okupljanja projektnog tima te nakon upoznavanja sa zadanim zadatkom, krenuli smo prikupljati literaturu te potrebne materijale koji su nam pomogli da se upoznamo s korištenim alatima i tehnologijama. Na prvom sastanku smo se podijelili u manje grupe te je svaka grupa dobila zadatak izrade određenog dijela aplikacije, što nam je uvelike olakšalo izradu same aplikacije. Kroz izradu alfa verzije aplikacije smo se upoznali sa novim alatima i problemima koje smo trebali riješiti, te je alfa verzija aplikacije i dokumentacija bila završena na kraju prvog ciklusa.

U drugom ciklusu smo implementirali ostale funkcionalnosti aplikacije te ju uredili, reorganizirali bazu podataka i dodali nove lambda funkcije na back-endu. Nakon toga smo napisali i dodali svu potrebnu dokumentaciju. Kroz proces izrade projekta bili smo suočeni s mnogim novim tehnologijama i problemima koji nam prije nisu bili poznati te smo ih sve uspješno riješili i sad smo bogatiji tim znanjem i kompetentniji. Prvo pitanje koje smo riješili je bilo koju ćemo infrastrukturu koristiti za izradu projekta, za što je bilo potrebno puno istraživanja raspoloživih opcija te bi rješenje puno brže našli da nam se ponovno postavi isto pitanje, jer smo upoznati sa terenom i opcijama. Isto tako je bilo i sa odabirom alatima koje smo koristili za izradu pojedinih dijelova projekta, kao što su Android studio, AWS-ove lambda funkcije i API gateway, no nakon što smo se upoznali samo je bilo potrebno proniknuti dublje u dostupna znanja alata/ tehnologije kojom se služimo i to sve uspješno povezati u jednu cjelinu sa komunikacijom među svojim dijelovima. Područje znanja koje bi nam nakon ovog projekta najviše pomoglo za kvalitetniji i efikasnije riješen slijedeći projekt je poznavanje Android studia, izrada mobilnog dijela aplikacije i front-enda. Nakon ovog projekta smo stekli vrijedno iskustvo izrade potpune aplikacije od početka do kraja sa server-sideom, te bi nam sada izrada nove mobilne aplikacije bila puno jednostavnija i lakša, za što

bi se i odlučili da nam se ukaže prilika ili da pronađemo ideju vrijednu realizacije.

Funkcionalnosti koje bi još ostvarili da usavršavamo aplikaciju bi bilo dodavanje pop-up obavijesti naše aplikacije.

Popis literature

Kontinuirano osvježavanje

Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>

Indeks slika i dijagrama

3.1	Sekvencijski dijagram za login	16
3.2	Sekvencijski dijagram za registraciju	17
3.3	OCR funkcija	18
3.4	Povijest svih skeniranih dokumenata zaposlenika	19
4.1	Arhitektura sustava	22
4.2	ER Diagram	26
4.3	Dijagram razreda	27
4.4	Dijagram stanja	28
4.5	Dijagram aktivnosti - Skeniranje dokumenta	29
4.6	Dijagram komponenti	31
5.1	Rezultat u bazi podataka	34
5.2	Rezultat u razvojnom okruženju	34
5.3	Potvrda rezultata u bazi podataka	35
5.4	Rezultat u razvojnom okruženju	35
5.5	Rezultat u razvojnom okruženju	35
5.6	Rezultat u razvojnom okruženju	36
5.7	Rezultat u razvojnom okruženju	36
5.8	Rezultat u razvojnom okruženju	37
5.9	Rezultat u relaciji "scanhistory"	37
5.10	Rezultat u relaciji "documents"	37
5.11	Rezultat u relaciji "reviserpending"	37
5.12	Rezultat u razvojnom okruženju	38
5.13	Rezultat u relaciji "scanhistory"	38
5.14	Rezultat u relaciji "documents"	38
5.15	Rezultat u relaciji "reviserpending"	38
5.16	Rezultat u razvojnom okruženju	39
5.17	Rezultat u relaciji "scanhistory"	39
5.18	Rezultat u relaciji "documents"	39
5.19	Rezultat u relaciji "reviserpending"	40

5.20 "Test logina"	42
5.21 "Test prisutnosti slike u ImageView-u"	45
5.22 "Test ispravnosti skeniranog dokumenta"	50
5.23 "Test neispravnosti skeniranog dokumenta"	55
5.24 "Test neispravnosti skeniranog dokumenta"	56
5.25 Dijagram razmještaja	58
5.26 Android Studio - Path	59
5.27 Android Studio - Build APK	59
5.28 Amazon Developer - Add New App	59
5.29 App - General Information	60
5.30 App - Description	61
5.31 App - Images and multimedia	62
5.32 App - Images and multimedia	63
5.33 App - APK Files	64
5.34 Amazon RDS - DB Setup	65
5.35 Amazon RDS - DB Instance	66
5.36 Amazon RDS - DB Storage	66
5.37 Amazon RDS - Connectivity	67
5.38 Amazon RDS - DB Name and Backup	68
5.39 Amazon RDS - DB Endpoint	69
6.1 "Dijagram develop"	78
6.2 "Dijagram devdoc"	78

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

Kontinuirano osvježavanje

1. sastanak

- Datum: u ovom formatu: 19.10.2021
- Prisustvovali: Cijeli tim
- Teme sastanka:
 - analiza zadatka
 - prva slika arhitekture sustava
 - generalna podjela posla

2. sastanak

- Datum: u ovom formatu: 7.11.2021
- Prisustvovali: Cijeli tim
- Teme sastanka:
 - detaljnija razrada arhitekture
 - rasprava o detaljima implementacije

3. sastanak

- Datum: u ovom formatu: 9.11.2021
- Prisustvovali: Cijeli tim
- Teme sastanka:
 - konačna razrada arhitekture

4. sastanak

- Datum: u ovom formatu: 16.11.2021
- Prisustvovali: Cijeli tim
- Teme sastanka:
 - raspodjela pisanja dokumentacije

5. sastanak

- Datum: u ovom formatu: 13.12.2021

- Prisustvovali: Cijeli tim
- Teme sastanka:
 - plan alfa verzije i gruba raspodjela ostatka posla pri kreiranju iste

Tablica aktivnosti

Kontinuirano osvježavanje

Napomena: Doprinosi u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.

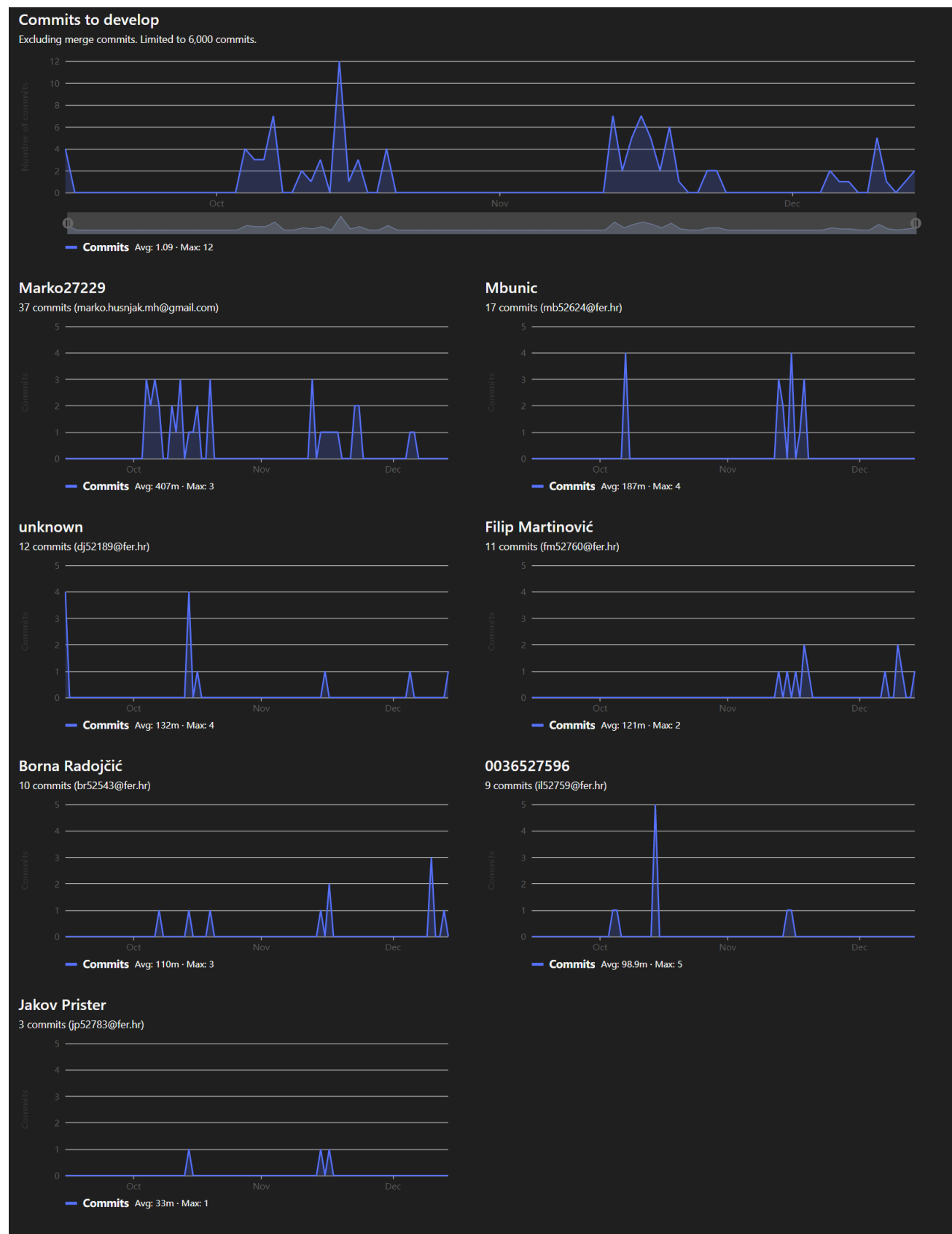
	Dominik Jurinčić	Marko Bunić	Marko Husnjak	Jakov Prister	Filip Martinović	Borna Radojčić	Ivan Lovrić
Upravljanje projektom	2	2	1				
Opis projektnog zadatka		7	1				
Funkcionalni zahtjevi			3				
Opis pojedinih obrazaca					4		
Dijagram obrazaca					7		
Sekvencijski dijagrami				4			
Opis ostalih zahtjeva						1	1
Arhitektura i dizajn sustava	2		3				
Baza podataka						1	1
Dijagram razreda					4		
Dijagram stanja		2					
Dijagram aktivnosti				2			
Dijagram komponenti							4
Korištene tehnologije i alati	1		1				
Ispitivanje programskog rješenja	4		1		6	6	
Dijagram razmještaja				3			
Upute za puštanje u pogon						5	

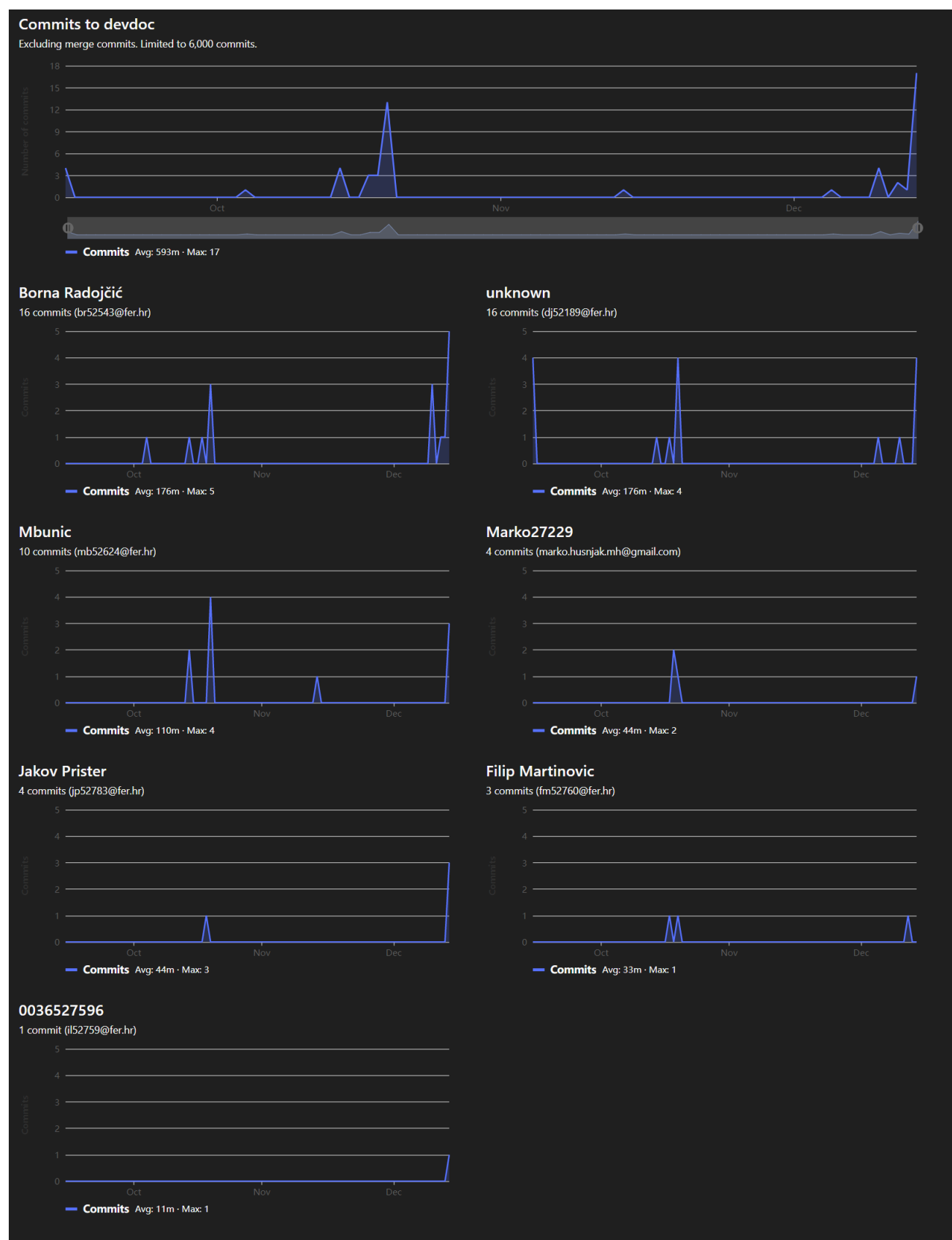
Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Dominik Jurinčić	Marko Bunić	Marko Husnjak	Jakov Prister	Filip Martinović	Borna Radojčić	Ivan Lovrić
Dnevnik sastajanja	1					1	
Zaključak i budući rad		1					1
Popis literature							
<i>Dodatne stavke kako ste podijelili izradu aplikacije – ostavio sam da navedete svoje dijelove -M.B.</i>							
<i>izrada UI-a i front-end aplikacije</i>		5	18	10	3		
<i>izrada baze podataka</i>						17	17
<i>spajanje back-enda s bazom podataka</i>	3			4		4	4
<i>rad na back-end-u i lambda funkcijama</i>	10			2			
<i>spajanje s bazom podataka</i>	8			4		4	4
<i>back end</i>	12			2			
<i>Povezivanje front-enda i back-enda</i>	2	2	2				
<i>Ispitivanje AWS i ostalih usluga za rješenje infrastukture</i>	4	4	1	4		4	4
<i>Istraživanje prikladnog rješenja arhitekture</i>	3	8	1				
<i>implementacija ugrađene kamere, slikanja, OCR-a i akcelerometra</i>					15		

Dijagrami pregleda promjena





Slika 6.2: "Dijagram devdoc"