Univerzitet u Beogradu–

Fakultet organizacionih nauka

Katedra za elektronsko poslovanje

Veb aplikacija za e-bankarstvo

|  |  |  |  |
| --- | --- | --- | --- |
| Rbr | Ime | Prezime | Broj indeksa |
| 1. | Miloš | Damjanović | 2021/0126 |
| **Mentor** | | Tamara Naumović | |
| **GitHub link** | | <https://github.com/elab-development/internet-tehnologije-2024-projekat-prvidomaci_2021_0126> | |

Sadržaj

[1. UVOD 7](#_Toc192456288)

[2. METODE I TEHNOLOGIJE ZA RAZVOJ VEB APLIKACIJE 8](#_Toc192456289)

[2.1. Uprošćena Larmanova metoda 8](#_Toc192456290)

[2.2. Tehnologije za razvoj *frontend* dela aplikacije 8](#_Toc192456291)

[2.2.1. JavaScript (JS) 8](#_Toc192456292)

[2.2.2. React 9](#_Toc192456293)

[2.3. Tehnologije za razvoj *backend* dela aplikacije 9](#_Toc192456294)

[2.3.1. PHP 9](#_Toc192456295)

[2.3.2. Laravel 10](#_Toc192456296)

[3. IMPLEMENTACIJA VEB APLIKACIJE 11](#_Toc192456297)

[3.1. Faza prikupljanja korisničkih zahteva 11](#_Toc192456298)

[3.1.1. Verbalni opis 11](#_Toc192456299)

[3.1.2. Slučajevi korišćenja 11](#_Toc192456300)

[SK1: Slučaj korišćenja – Kreiraj transakciju 13](#_Toc192456301)

[SK2: Slučaj korišćenja – Pretraži transakciju 14](#_Toc192456302)

[SK3: Slučaj korišćenja – Pretraži korisnika 15](#_Toc192456303)

[SK4: Slučaj korišćenja – Izmeni korisnika 16](#_Toc192456304)

[SK6: Slučaj korišćenja – Kreiraj račun 18](#_Toc192456305)

[SK7: Slučaj korišćenja – Kreiraj karticu 19](#_Toc192456306)

[SK8: Slučaj korišćenja –Obriši karticu 20](#_Toc192456307)

[SK9: Slučaj korišćenja – Kreiraj admina 21](#_Toc192456308)

[SK10: Slučaj korišćenja – Pretraži admina 22](#_Toc192456309)

[SK11: Slučaj korišćenja –Obriši admina 23](#_Toc192456310)

[3.2. Faza analize 24](#_Toc192456311)

[3.2.1. Sistemski dijagrami sekvence 24](#_Toc192456312)

[DS1: Dijagram sekvence slučaja korišćenja - Kreiranje transakcije 24](#_Toc192456313)

[DS2 : Dijagram sekvence slučaja korišćenja - Pretraga transakcije 25](#_Toc192456314)

[DS3 : Dijagram sekvence slučaja korišćenja - Pretraga korisnika 26](#_Toc192456315)

[DS4 : Dijagram sekvence slučaja korišćenja - Izmena korisnika 27](#_Toc192456316)

[DS5 : Dijagram sekvence slučaja korišćenja - Brisanje korisnika 29](#_Toc192456317)

[DS6 : Dijagram sekvence slučaja korišćenja - Kreiranje računa 30](#_Toc192456318)

[DS7 : Dijagram sekvence slučaja korišćenja - Kreiranje kartice 32](#_Toc192456319)

[DS8 : Dijagram sekvence slučaja korišćenja - Brisanje kartice 33](#_Toc192456320)

[DS9 : Dijagram sekvence slučaja korišćenja - Kreiranje admina: 37](#_Toc192456321)

[DS10 : Dijagram sekvence slučaja korišćenja - Pretraga admina: 38](#_Toc192456322)

[DS11 : Dijagram sekvence slučaja korišćenja - Brisanje admina: 39](#_Toc192456323)

[3.2.2. Struktura softverskog sistema 40](#_Toc192456324)

[3.3. Faza projektovanja 41](#_Toc192456325)

[3.3.1. Projektovanje korisničkog interfejsa (ekstranskih formi) 41](#_Toc192456326)

[3.3.2. Projektovanje aplikacione logike 52](#_Toc192456327)

[Laravel Kontroleri (Controller) 52](#_Toc192456328)

[Laravel Autentifikacija (Authentication) 57](#_Toc192456329)

[Laravel Resursi (Resources) 58](#_Toc192456330)

[Laravel Rutiranje (Api routes) 60](#_Toc192456332)

[Slično izgleda ruta za fetchovanje naloga. 74](#_Toc192456333)

[Laravel Database Seeder 76](#_Toc192456334)

[React Hooks 76](#_Toc192456335)

[React Komponente (Components) 77](#_Toc192456336)

[React Rutiranje (BrowserRouter) 78](#_Toc192456337)

[Laravel i React 81](#_Toc192456338)

[3.3.3. Projektovanje strukture softverskog sistema 84](#_Toc192456339)

[Migracije 84](#_Toc192456340)

[Modeli 86](#_Toc192456341)

[3.3.4. Projektovanje skladišta podataka 89](#_Toc192456342)

[3.4. Faza implementacije 90](#_Toc192456343)

[3.5. Faza testiranja 93](#_Toc192456344)

[4. ZAKLJUČAK 94](#_Toc192456345)

[5. LITERATURA 94](#_Toc192456346)

# UVOD

Aplikacije za bankarstvo su postale ključne komponente svakog modernog bankarskog sistema, omogućivši korisnicima da putem interneta upravljaju svojim računima i transakcijama.

Implementirana web-aplikacija predstavlja sistem za e-bankarstvo koja korisnicima, administratorima i menadžerima omogućava jednostavno i interaktivno upravljanje nalozima, računima, platnim karticama, finansijskim transakcijama i drugim bankarskim funkcionalnostima. Rad se fokusira na projektovanje, implentaciju i dokumentaciju gorepomenutog softverskog rešenja. Obuhvata sve faze razvoja softvera, od prikupljanja zahteva korisnika, sve do testiranja, primenjujući principe uprošćene Larmanove metodologije.

Što se tiče sadržaja narednih poglavlja, prvo sledi kratak osvrt na tehnologije korišćene za razvoj ove aplikacije i pokoja reč o njima. Zatim, detaljno opisan životni ciklus razvoja aplikacije, počevši sa detaljno razrađenim korisničkim zahtevima, pa sve preko definisanja strukture sistema, njegovog projektovanja i implementacije, pa do konačnog testiranja.

Aplikacija za e-bankarstvo koristi B2C (Business-to-Consumer) biznis model, odnosno, direktno je usmerena krajnjim korisnicima tj. klijentima banke. S obzirom da je ovo bankarska aplikacija, neophodan je visok nivo sigurnosti, stoga je osnovni cilj aplikacije da pruži korisnicima bezbedan pristup njihovim podacima, kao i prost način upravljanja svojim finansijama.

Iako su ciljna grupa regularni korisnici, tj. klijenti banke, omogućena je i uloga administratora u sistemu, koja služi kao podrška korisniku i omogućava bezbedan način otvaranja novih računa, platnih kartica i još mnoge funkcionalnosti za koje bi korisnik, u suprotnom, morao fizički biti prisutan u filijali banke.

# METODE I TEHNOLOGIJE ZA RAZVOJ VEB APLIKACIJE

## Uprošćena Larmanova metoda

Uprošćena Larmanova metoda razvoja softvera je jedna od metoda kojom se organizuju aktivnosti za kreiranje, isporuku i održavanje softverskih sistema i zasniva se na iterativnom pristupu.

1. Prikupljanje korisničkih zahteva
2. Analiza
3. Projektovanje
4. Implementacija
5. Testiranje

Prikupljanje korisničkih zahteva se fokusira na identifikaciju potreba korisnika i razumevanje koja su njihova očekivanja od sistema i koje funkcionalnosti oni očekuju od aplikacije. Nakon identifikacije zahteva sledi njihova analiza zarad razumevanja poslovnih procesa. U ovoj fazi se definišu osnovi strukture i ponašanja sistema. Prilikom projektovanja se dizajn sistema detaljno kreira na osnovu poslovne logike dobijene iz faze analize. Projektuje se korisnički interfejs, aplikaciona logika i skladište podataka (tzv. tronivojska arhitektura). Nakon ove faze slede implementacija i testiranje, koje predstavljaju realizaciju samog softvera i verifikaciju i validaciju njegovog sistema.

## Tehnologije za razvoj *frontend* dela aplikacije

Frontend deo aplikacije obrađen je u JavaScript-u, koristeći njegovu popularnu biblioteku – React.

### JavaScript (JS)

Netscape je 1995. godine JavaScript najavio kao jednostavan za korišćenje, objektno-skriptni jezik, dizajniran za pravljenje onlajn aplikacija, koji povezuje objekte i resurse i na klijentu, i na serveru. Ubrzo je postao standard – glavni skriptni jezik korišćen na internet pretraživačima, prosto neophodan za izradu modernih web aplikacija. Iako je prvobitno namenjen da bude jednostavan skriptni jezik, naterao je ljude da ga koriste za pravljenje i daleko većih projekata i programa.

Omogućio je dinamičko ažuriranje sadržaja na stranici, korišćenjem asinhrone komunikacije sa serverom (AJAX). Pored toga, JavaScript kao programski jezik povezan je sa web stranicama pomoću Document Object Model-a (DOM). Omogućio je dodavanje interaktivnosti web stranicama i višu skalabilnost projektima.

Po svojoj prirodi, JS je dinamički-tipiziran jezik i pripada pretežno funkcionalnoj paradigmi. Podržava različite tipove podataka kao što su: number, string, boolean, object, null i undefined, bigint, symbol i koristi C-like sintaksu.

Glavne prednosti su svakako velika zajednica programera i široka rasprostranjenost, obogaćena različitim bibliotekama i alatima koji olakšavaju upotrebu jezika. Pored toga, omogućava izvršavanje koda na klijentskoj strani, i nije težak za korišćenje. Međutim, sam po sebi uglavnom ne skalira najbolje sa visoko-skalabilnim projektima, i dinamička tipizacija može dovesti do neplaniranih grešaka teških za otkrivanje.

### React

React je, trenutno, najpopularnija JavaScript biblioteka zasnovana na konceptu komponenata za izgradnju modularnih i interaktivnih korisničkih interfejsa. Ona predstavlja view (V) sloj M-V-C (Model View Controller) paterna. Omogućava zavoj velikih i kompleksnih web aplikacija gde stanja podataka stranice mogu da se menjaju bez naknadnog osvežavanja stranice. Razvijena je od strane Fejzbuka 2013. godine

Prilikom korišćenja React-a upotrebljava se zgodna XML-zasnovana ekstenzija JS-a, poznata kao JSX sintaksa, kojom se generiše stablo komponenata kao kombinacija više XML čvorova. Zajedno s ovim, performanse softvera razvijenog u React-u su izuzetne, s obzirom da React koristi koncept Virtual DOM (Document Object Model), koji se razlikuje od DOM-a generisanog od browsera. Svaki put kad se promena na stranici dogodi, promene se prvo dešavaju na Virtual DOM-u, gde nakon toga algoritam za upoređivanje ova dva stabla vrši izmene nad DOM-om web stranice. Ovo izuzetno povećava performansu aplikacije, naročito kada dolazi do izmena velikog broja podataka.

Doduše, postoji par ograničenja za korisnika koja treba uzeti u obzir pre rada sa ovom bibliotekom. Samo View sloj MVC patterna je obuhvaćen, tako da je neophodan rad u nekim drugim okruženjima kako bi se završio razvoj određenog projekta. Pored toga, u Reactu se greške javljaju za vreme kompajliranja, ne samo tokom izvršavanja koda, tako da to predstavlja frustrirajuću okolnost za određene programere. Sa druge strane, činjenica je da je React moćan alat visokih performansi sa ne preterano napornom krivom učenja.

## Tehnologije za razvoj *backend* dela aplikacije

Backend deo aplikacije obrađen je u PHP-u, koristeći popularan framework Laravel.

### PHP

PHP(PHP: Hypertext Preprocessor – rekurzivni akronim ) je razvijen 1994. godine od strane Razmusa Lerdorfa. Pri nastanku, nije bio planiran da bude programski jezik širokih razmera, međutim, danas je jezik koji je korišćen za razvoj više od 40% svih web aplikacija. Tada je nastao sa ciljem kreiranja dinamičkog web sadržaja, i dan danas je i dalje jedan od najmoćnijih alata namenjenih za to. Kako bi se generisao HTML sadržaj, neophodan je PHP parser i Web server za slanje dokumenata. Tri primarna načina korišćenja PHP-a su:

* Scripting sa serverske strane
* Command-line scriptin
* GUI aplikacije sa klijentske strane

PHP takođe ima ugrađenu podršku za generisanju PDF, GIF, JPG, PNG fajlova, omogućava upravljanje bazama podataka, autentifikaciju korisnika, i mnoge druge backend funkcionalnosti. Danas pruža širok spektar mogućnosti, i korišćen je kao osnova za razvoj brojnih moćnih frejmvorka (Laravel, Symfony, Phalcon...), koji su propraćeni visokom podrškom i zajednicom, naročito u poslednjih par godina.

Dinamički je tipiziran jezik (što može dovesti do sličnih problema kao i u slučaju JS-a), ali podržava različite tipove podataka, kao na primer: string, integer, float, boolean, array, object, resource, null. Koristi C-like sintaksu. Međutim, dobio je lošu reputaciju zbog velikog broja loše-dizajniranih web-sajtova i nekonzistentnosti vezanih za sintaksu jezika.

### Laravel

Iako je PHP sporijih performansi i nezgodniji za korišćenje u odnosu na neke druge moderne programske jezike, mnogi novonastali backend framework-ovi pisani u jezicima poput Ruby-ja ili Python-a nisu imali ni približno dovoljno mogućnosti i potencijala za Web programiranje. Tako se stvorio prostor za nastanak Laravel-a, koji predstavlja vodeći PHP framework, koji je uzgred otvorenog koda. Stvoren je 2011. godine u nameri da olakša razvoj web aplikacija kroz bogatu funkcionalnost i brojne mogućnosti. Ovaj framework je dosta olakšao glomaznu i neintuitivnu sintaksu PHP-a i povećao ekspresivnost i konzistentnost koda.

Laravel omogućava i znatno olakšava rad sa bazom podataka koristeći Eloquent ORM (Object-Relational Mapper), i na taj način koristeći MVC pattern razvoja softvera. Pregledi (Views) su zaduženi za prikaz odgovora vraćenog od strane kontrolera u zgodnom formatu, uglavnom HTML stranici. Ti formati mogu jednostavno da se dizajniraju korišćenjem Blade template jezika u Laravelu. Laravel Middleware i Laravel Sanctum značajno doprinose zaštiti softvera i sigurnosti od neovlašćenih pristupa.

U svakom slučaju, ovo je visokofunkcionalan framework sa integrisanim mnogobrojnim alatima. Rad u njemu dodatno olakšava velika zajednica programera, i fenomenalna dokumentacija. Međutim, framework je i dalje baziran na PHP-u, tako da bi kriva učenja bila strma i zbunjujuća.

# IMPLEMENTACIJA VEB APLIKACIJE

## Faza prikupljanja korisničkih zahteva

### Verbalni opis

Akcenat je stavljen na ispunjavanje osnovnih atributa kvaliteta softverskog sistema: upotrebljivosti, pouzdanosti, performansi, podrživosti, zahteva svakog tipa korisnika. Regularni korisnici traže aplikaciju u kojoj imaju jednostavan pregled u svoje račune, kartice i transakcije, koja im omogućava interaktivnu manipulaciju njima. Uzevši sve ovo u obzir, izdvojeni su glavni slučajevi korišćenja.

### Slučajevi korišćenja

SK1: Kreiranje transakcije

SK2: Pretraga transakcije

SK3: Pretraga korisnika

SK4: Izmena korisnika

SK5: Brisanje korisnika

SK6: Kreiranje racuna

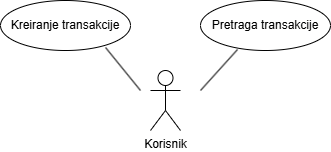
SK7: Kreiranje kartice

SK8: Brisanje kartice

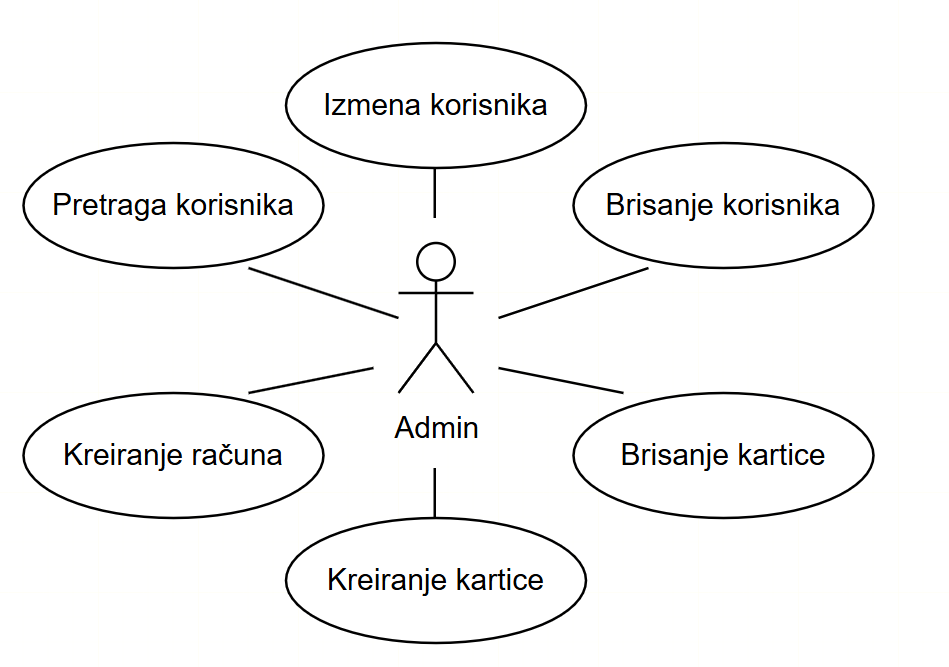
SK9: Kreiranje admina

SK10: Pretraga admina

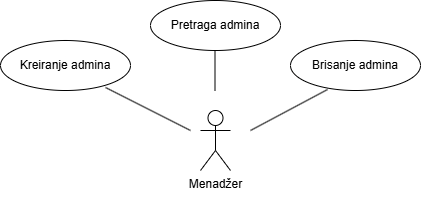
SK11: Brisanje admina



Slika 1 - Slučajevi korišćenja običnog korisnika



Slika 2 - Slučajevi korišćenja admina



Slika 3 - Slučajevi korišćenja menadžera

#### SK1: Slučaj korišćenja – Kreiraj transakciju

**Naziv SK**

Kreiranje transakcije

**Aktori SK**

Korisnik

**Učesnici SK**

Korisnik i sistem

**Preduslov:** Sistem je uključen i korisnik je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa transakcijom.

**Osnovni scenario SK**

1. Korisnik **unosi** podatke za transakciju. (APUSO)
2. Korisnik **kontroliše** da li je korektno uneo podatke za transakciju. (ANSO)
3. Korisnik **poziva** sistem da zapamti podatke o transakciji. (APSO)
4. Sistem **pamti** podatke o transakciji. (SO)
5. Sistem **prikazuje** korisniku poruku: “Uspešno sačuvana transakcija!” (IA)

**Alternativni scenariji**

5.1 Ukoliko sistem ne može da zapamti podatke o transakciji, on prikazuje korisniku poruku: “Popunite sva polja!”. Prekida se izvršenje scenarija. (IA)

#### SK2: Slučaj korišćenja – Pretraži transakciju

**Naziv SK**

Pretraga transakcije

**Aktori SK**

Korisnik

**Učesnici SK**

Korisnik i sistem

**Preduslov:** Sistem je uključen i korisnik je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa transakcijama.

**Osnovni scenario SK**

1. Korisnik **unosi** vrednost po kojoj pretražuje transakcije. (APUSO)
2. Korisnik **poziva** sistem da nađe transakcije po zadatoj vrednosti. (APSO)
3. Sistem **traži** transakcije po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** korisniku transakcije. (IA)
5. Korisnik **bira** jednu transakciju. (APUSO)
6. Korisnik **poziva** sistem da učita podatke o odabranoj transakciji. (APSO)
7. Sistem **učitava** podatke o transakciji. (SO)
8. Sistem **prikazuje** korisniku podatke o transakciji. (IA)

#### SK3: Slučaj korišćenja – Pretraži korisnika

**Naziv SK**

Pretraga korisnika

**Aktori SK**

Admin

**Učesnici SK**

Admin i sistem

**Preduslov:** Sistem je uključen i admin je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa korisnicima.

**Osnovni scenario SK**

1. Admin **unosi** vrednost po kojoj pretražuje korisnike. (APUSO)
2. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
3. Sistem **traži** korisnike po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** adminu korisnike. (IA)
5. Admin **bira** jednog korisnika. (APUSO)
6. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
7. Sistem **učitava** podatke o korisniku. (SO)
8. Sistem **prikazuje** adminu podatke o korisniku. (IA)

#### SK4: Slučaj korišćenja – Izmeni korisnika

**Naziv SK**

Izmena korisnika

**Aktori SK**

Admin

**Učesnici SK**

Admin i sistem

**Preduslov:** Sistem je uključen i admin je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa korisnicima.

**Osnovni scenario SK**

1. Admin **unosi** vrednost po kojoj pretražuje korisnike. (APUSO)
2. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
3. Sistem **traži** korisnike po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** adminu korisnike. (IA)
5. Admin **bira** jednog korisnika. (APUSO)
6. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
7. Sistem **učitava** podatke o korisniku. (SO)
8. Sistem **prikazuje** adminu podatke o korisniku. (IA)
9. Admin **unosi** (**menja**) podatke o korisniku. (APUSO)
10. Admin **kontroliše** da li je korektno uneo (izmenio) podatke o korisniku. (ANSO)
11. Admin **poziva** sistem da zapamti podatke o korisniku. (APSO)
12. Sistem **pamti** podatke o korisniku. (SO)
13. Sistem **prikazuje** adminu poruku: “Korisnik uspešno izmenjen!”. (IA)

**Alternativni scenariji**

13.1 Ukoliko sistem ne može da izmeni korisnika, on prikazuje adminu poruku: “Loš unos podataka!”. Prekida se izvršenje scenarija. (IA)

**SK5: Slučaj korišćenja – Obriši korisnika**

**Naziv SK**

Brisanje korisnika

**Aktori SK**

Admin

**Učesnici SK**

Admin i sistem

**Preduslov:** Sistem je uključen i admin je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa korisnicima.

**Osnovni scenario SK**

1. Admin **unosi** vrednost po kojoj pretražuje korisnike. (APUSO)
2. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
3. Sistem **traži** korisnike po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** adminu korisnike. (IA)
5. Admin **bira** jednog korisnika. (APUSO)
6. Admin **poziva** sistem da obriše korisnika. (APSO)
7. Sistem **briše** korisnika. (SO)
8. Sistem **prikazuje** adminu poruku „Sistem je obrisao korisnika“. (IA)

**Alternativni scenariji**

8.1 Ukoliko sistem ne može obriše korisnika, on prikazuje adminu poruku: “Sistem ne može da obriše korisnika!”. Prekida se izvršenje scenarija. (IA)

#### SK6: Slučaj korišćenja – Kreiraj račun

**Naziv SK**

Kreiranje računa

**Aktori SK**

Admin

**Učesnici SK**

Admin i sistem

**Preduslov:** Sistem je uključen i admin je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa korisnicima.

**Osnovni scenario SK**

1. Admin **unosi** vrednost po kojoj pretražuje korisnike. (APUSO)
2. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
3. Sistem **traži** korisnike po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** adminu korisnike. (IA)
5. Admin **bira** jednog korisnika. (APUSO)
6. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
7. Sistem **učitava** podatke o korisniku. (SO)
8. Sistem **prikazuje** adminu podatke o korisniku. (IA)
9. Admin **unosi** podatke za kreiranje novog računa. (APUSO)
10. Admin **poziva** sistem da kreira novi račun. (APSO)
11. Sistem **kreira** novi račun. (SO)
12. Sistem **prikazuje** adminu poruku o uspešno kreiranom računu. (IA)

**Alternativni scenariji**

12.1 Ukoliko sistem ne može da prikaže adminu poruku o uspešno kreiranom računu, on prikazuje adminu poruku: “Sistem ne može da kreira novi račun!”. Prekida se izvršenje scenarija. (IA)

#### SK7: Slučaj korišćenja – Kreiraj karticu

**Naziv SK**

Kreiranje kartice

**Aktori SK**

Admin

**Učesnici SK**

Admin i sistem

**Preduslov:** Sistem je uključen i admin je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa korisnicima.

**Osnovni scenario SK**

1. Admin **unosi** vrednost po kojoj pretražuje korisnike. (APUSO)
2. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
3. Sistem **traži** korisnike po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** adminu korisnike. (IA)
5. Admin **bira** jednog korisnika. (APUSO)
6. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
7. Sistem **učitava** podatke o korisniku. (SO)
8. Sistem **prikazuje** adminu podatke o korisniku. (IA)
9. Admin **poziva** sistem da učita podatke o karticama za odabrani račun. (APSO)
10. Sistem **učitava** podatke o karticama za odabrani račun. (SO)
11. Sistem **prikazuje** adminu podatke o karticama za odabrani račun. (IA)
12. Admin **unosi** podatke za novu karticu na odabranom računu. (APUSO)
13. Admin **poziva** sistem da zapamti podatke o novoj kartici na odabranom računu. (APSO)
14. Sistem **pamti** podatke o kartici na odabranom računu. (SO)
15. Sistem **prikazuje** adminu poruku: “Uspešno sačuvana kartica!” (IA)

**Alternativni scenariji**

11.1 Ukoliko sistem ne može da prikaže adminu podatke o karticama za odabrani račun, on prikazuje adminu poruku: “Sistem ne može da prikaže podatke za izabranu karticu na računu!”. Prekida se izvršenje scenarija. (IA)

15.1 Ukoliko sistem ne može da zapamti podatke o kartici za odabrani račun, on prikazuje adminu poruku: “Sistem ne može da zapamti podatke o kartici!”. Prekida se izvršenje scenarija. (IA)

#### SK8: Slučaj korišćenja –Obriši karticu

**Naziv SK**

Brisanje kartice

**Aktori SK**

Admin

**Učesnici SK**

Admin i sistem

**Preduslov:** Sistem je uključen i admin je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa korisnicima.

**Osnovni scenario SK**

1. Admin **unosi** vrednost po kojoj pretražuje korisnike. (APUSO)
2. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
3. Sistem **traži** korisnike po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** adminu korisnike. (IA)
5. Admin **bira** jednog korisnika. (APUSO)
6. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
7. Sistem **učitava** podatke o korisniku. (SO)
8. Sistem **prikazuje** adminu podatke o korisniku. (IA)
9. Admin **poziva** sistem da učita podatke o karticama za odabrani račun. (APSO)
10. Sistem **učitava** podatke o karticama za odabrani račun. (SO)
11. Sistem **prikazuje** adminu podatke o karticama za odabrani račun. (IA)
12. Admin **poziva** sistem da obriše karticu sa odabranog računa. (APSO)
13. Sistem **briše** karticu sa odabranog računa. (SO)
14. Sistem **prikazuje** adminu poruku: “Kartica uspešno obrisana!” (IA)

**Alternativni scenariji**

11.1 Ukoliko sistem ne može da prikaže adminu podatke o karticama za odabrani račun, on prikazuje adminu poruku: “Sistem ne može da prikaže podatke za izabranu karticu na računu!”. Prekida se izvršenje scenarija. (IA)

14.1 Ukoliko sistem ne može da obriše karticu sa odabranog računa, on prikazuje adminu poruku: “Sistem ne može da obriše karticu sa odabranog računa!”. Prekida se izvršenje scenarija. (IA)

#### SK9: Slučaj korišćenja – Kreiraj admina

**Naziv SK**

Kreiranje admina

**Aktori SK**

Menadžer

**Učesnici SK**

Menadžer i sistem

**Preduslov:** Sistem je uključen i menadžer je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa adminima.

**Osnovni scenario SK**

1. Menadžer **unosi** podatke za admina. (APUSO)
2. Menadžer **kontroliše** da li je korektno uneo podatke za admina. (ANSO)
3. Menadžer **poziva** sistem da zapamti podatke o adminu. (APSO)
4. Sistem **pamti** podatke o adminu. (SO)
5. Sistem **prikazuje** menadžeru poruku: “Uspešno sačuvan admin!” (IA)

**Alternativni scenariji**

5.1 Ukoliko sistem ne može da zapamti podatke o adminu, on prikazuje korisniku poruku: “Popunite sva polja!”. Prekida se izvršenje scenarija. (IA)

#### SK10: Slučaj korišćenja – Pretraži admina

**Naziv SK**

Pretraga admina

**Aktori SK**

Menadžer

**Učesnici SK**

Menadžer i sistem

**Preduslov:** Sistem je uključen i menadžer je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa adminima.

**Osnovni scenario SK**

1. Menadžer **unosi** vrednost po kojoj pretražuje admine. (APUSO)
2. Menadžer **poziva** sistem da nađe admine po zadatoj vrednosti. (APSO)
3. Sistem **traži admine** po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** menadžeru admine. (IA)
5. Menadžer **bira** jednog admina. (APUSO)
6. Menadžer **poziva** sistem da učita podatke o odabranom adminu. (APSO)
7. Sistem **učitava** podatke o adminu. (SO)
8. Sistem **prikazuje** menadžeru podatke o adminu. (IA)

#### SK11: Slučaj korišćenja –Obriši admina

**Naziv SK**

Brisanje admina

**Aktori SK**

Menadžer

**Učesnici SK**

Menadžer i sistem

**Preduslov:** Sistem je uključen i menadžer je ulogovan pod svojom šifrom. Sistem prikazuje formu za rad sa adminima.

**Osnovni scenario SK**

1. Menadžer **unosi** vrednost po kojoj pretražuje admine. (APUSO)
2. Menadžer **poziva** sistem da nađe admine po zadatoj vrednosti. (APSO)
3. Sistem **traži admine** po zadatoj vrednosti. (SO)
4. Sistem **prikazuje** menadžeru admine. (IA)
5. Menadžer **bira** jednog admina. (APUSO)
6. Menadžer **poziva** sistem da obriše admina. (APSO)
7. Sistem **briše** admina. (SO)
8. Sistem **prikazuje** menadžeru poruku „Sistem je obrisao admina“. (IA)

**Alternativni scenariji**

8.1 Ukoliko sistem ne može da obriše admina, on prikazuje menadžeru poruku: “Sistem ne može da obriše admina!”. Prekida se izvršenje scenarija. (IA)

## Faza analize

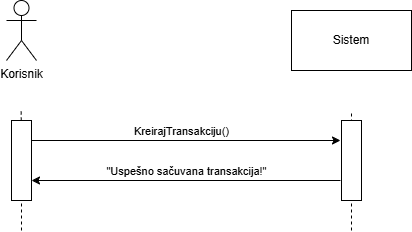
### Sistemski dijagrami sekvence

U ovoj fazi, zahtevi se detaljno analiziraju kako bi se razumeli poslovni procesi i tehnički izazovi. Cilj je da se definišu logička struktura tj. atributi, i ponašanja tj. sistemske operacije sistema. Struktura sistema je opisana konceptualnim modelom, a ponašanja sistema sekvencnim dijagramima.

#### DS1: Dijagram sekvence slučaja korišćenja - Kreiranje transakcije

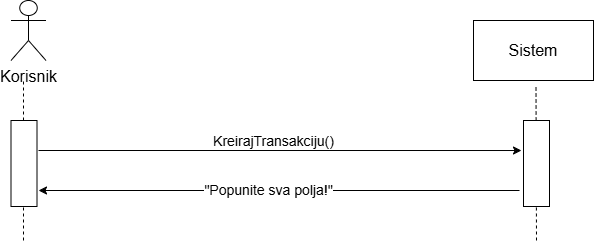
**Osnovni scenario SK**

1. Korisnik **poziva** sistem da zapamti podatke o transakciji. (APSO)
2. Sistem **prikazuje** korisniku poruku: “Uspešno sačuvana transakcija!” (IA)



**Alternativni scenariji**

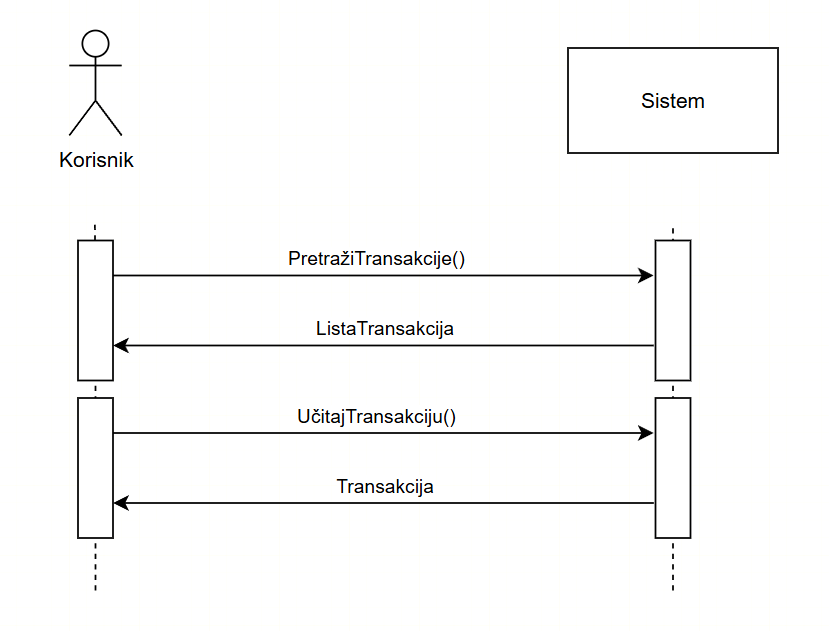
2.1 Ukoliko sistem ne može da zapamti podatke o transakciji, on prikazuje korisniku poruku: “Popunite sva polja!”. Prekida se izvršenje scenarija. (IA)



#### DS2 : Dijagram sekvence slučaja korišćenja - Pretraga transakcije

**Osnovni scenario SK**

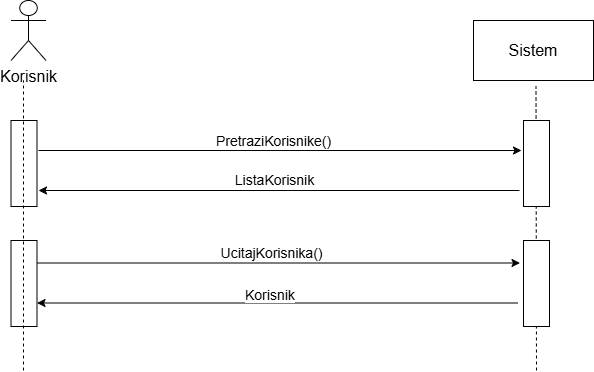
1. Korisnik **poziva** sistem da nađe transakcije po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** korisniku transakcije. (IA)
3. Korisnik **poziva** sistem da učita podatke o odabranoj transakciji. (APSO)
4. Sistem **prikazuje** korisniku podatke o transakciji. (IA)



#### DS3 : Dijagram sekvence slučaja korišćenja - Pretraga korisnika

**Osnovni scenario SK**

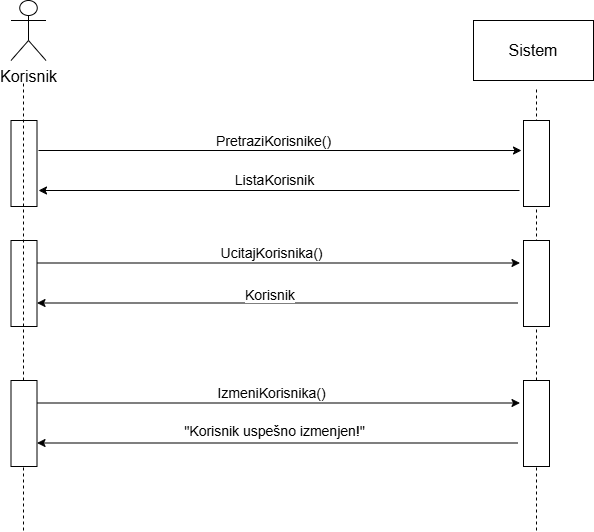
1. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** adminu korisnike. (IA)
3. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
4. Sistem **prikazuje** adminu podatke o korisniku. (IA)



#### DS4 : Dijagram sekvence slučaja korišćenja - Izmena korisnika

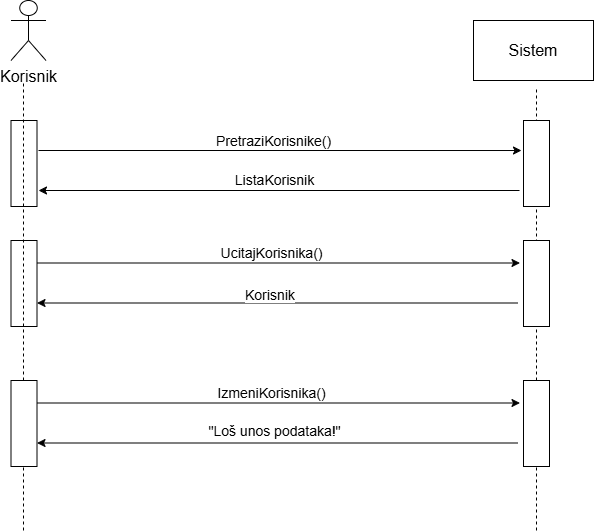
**Osnovni scenario SK**

1. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** adminu korisnike. (IA)
3. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
4. Sistem **prikazuje** adminu podatke o korisniku. (IA)
5. Admin **poziva** sistem da izmeni podatke o korisniku. (APSO)
6. Sistem **prikazuje** adminu poruku: “Korisnik uspešno izmenjen!”. (IA)



**Alternativni scenariji**

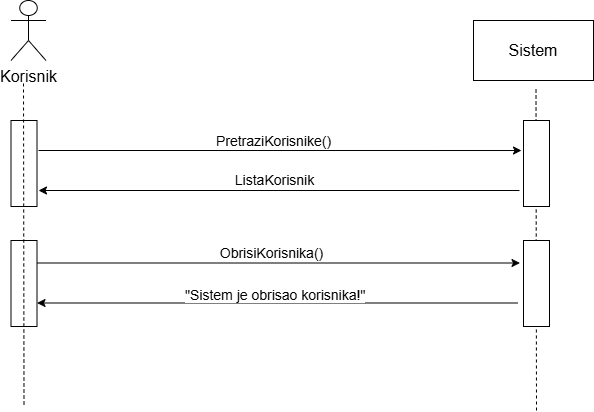
6.1 Ukoliko sistem ne može da izmeni korisnika, on prikazuje adminu poruku: “Loš unos podataka!”. Prekida se izvršenje scenarija. (IA)



#### DS5 : Dijagram sekvence slučaja korišćenja - Brisanje korisnika

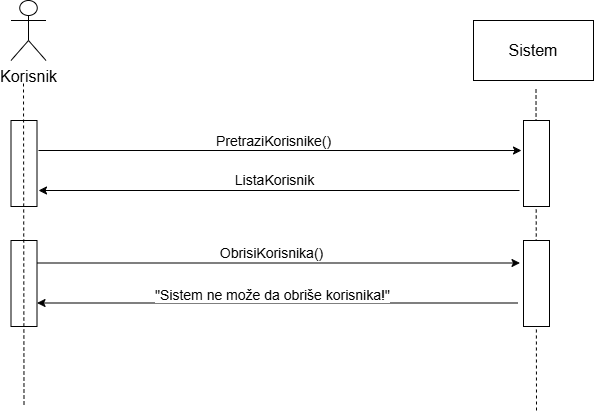
**Osnovni scenario SK**

1. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** adminu korisnike. (IA)
3. Admin **poziva** sistem da obriše korisnika. (APSO)
4. Sistem **prikazuje** poruku “Sistem je obrisao korisnika”. (IA)



**Alternativni scenariji**

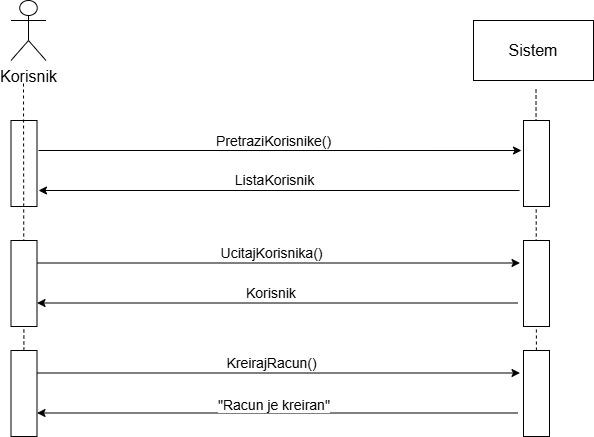
4.1 Ukoliko sistem ne može obriše korisnika, on prikazuje adminu poruku: “Sistem ne može da obriše korisnika!”. Prekida se izvršenje scenarija. (IA)



#### DS6 : Dijagram sekvence slučaja korišćenja - Kreiranje računa

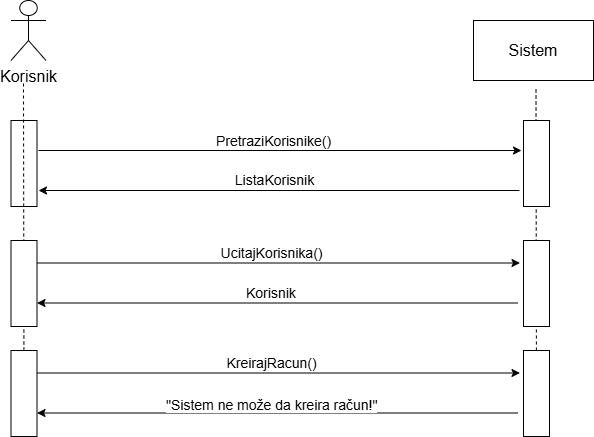
**Osnovni scenario SK**

1. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** adminu korisnike. (IA)
3. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
4. Sistem **prikazuje** adminu podatke o korisniku. (IA)
5. Admin **poziva** sistem da kreira novi račun. (APSO)
6. Sistem **prikazuje** adminu poruku o uspešno kreiranom računu. (IA)



**Alternativni scenariji**

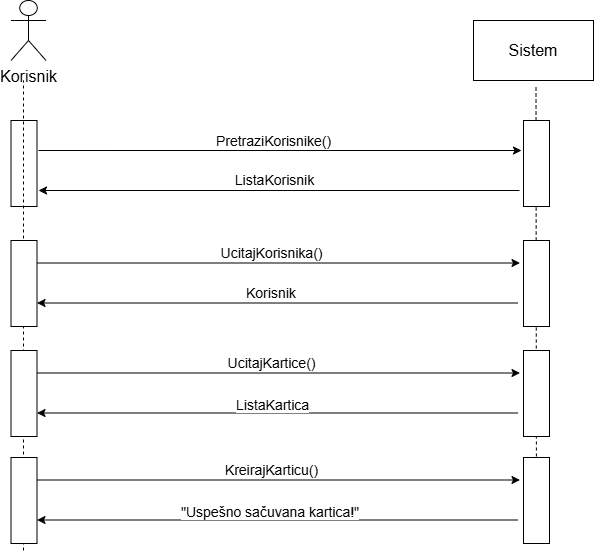
6.1 Ukoliko sistem ne može da prikaže adminu poruku o uspešno kreiranom računu, on prikazuje adminu poruku: “Sistem ne može da kreira novi račun!”. Prekida se izvršenje scenarija. (IA)



#### DS7 : Dijagram sekvence slučaja korišćenja - Kreiranje kartice

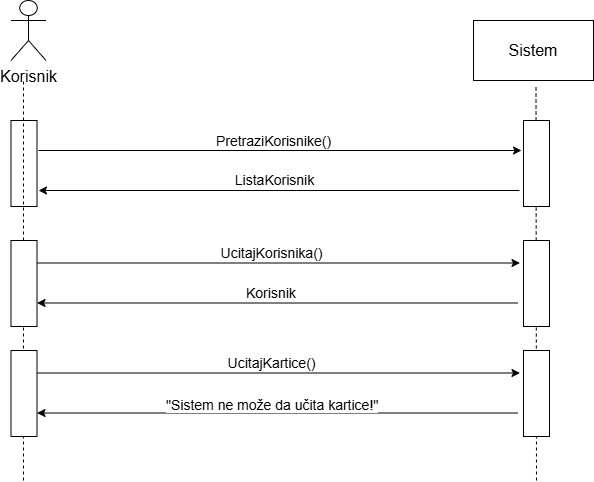
**Osnovni scenario SK**

1. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** adminu korisnike. (IA)
3. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
4. Sistem **prikazuje** adminu podatke o korisniku. (IA)
5. Admin **poziva** sistem da učita podatke o karticama za odabrani račun. (APSO)
6. Sistem **prikazuje** adminu podatke o karticama za odabrani račun. (IA)
7. Admin **poziva** sistem da zapamti podatke o novoj kartici na odabranom računu. (APSO)
8. Sistem **prikazuje** adminu poruku: “Uspešno sačuvana kartica!” (IA)



**Alternativni scenariji**

6.1 Ukoliko sistem ne može da prikaže adminu podatke o karticama za odabrani račun, on prikazuje adminu poruku: “Sistem ne može da učita kartice!”. Prekida se izvršenje scenarija. (IA)

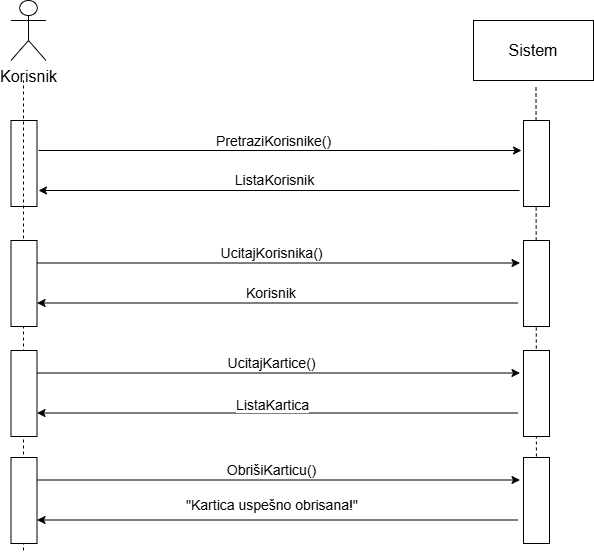


8.1 Ukoliko sistem ne može da zapamti podatke o kartici za odabrani račun, on prikazuje adminu poruku: “Sistem ne može da zapamti podatke o kartici!”. Prekida se izvršenje scenarija. (IA)

#### DS8 : Dijagram sekvence slučaja korišćenja - Brisanje kartice

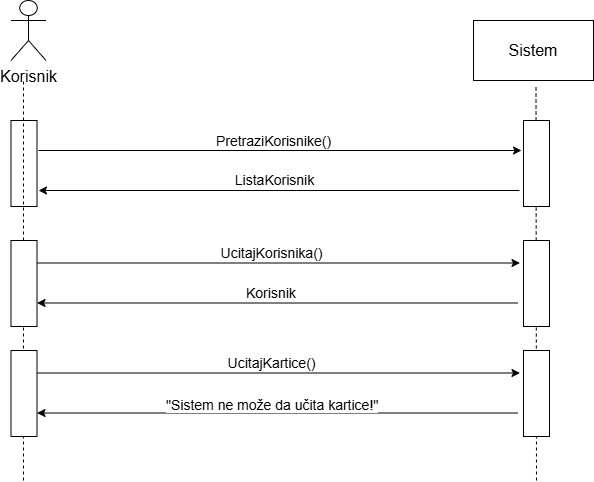
**Osnovni scenario SK**

1. Admin **poziva** sistem da nađe korisnike po zadatoj vrednosti. (APSO)
2. Sistem **prikazuje** adminu korisnike. (IA)
3. Admin **poziva** sistem da učita podatke o odabranom korisniku. (APSO)
4. Sistem **prikazuje** adminu podatke o korisniku. (IA)
5. Admin **poziva** sistem da učita podatke o karticama za odabrani račun. (APSO)
6. Sistem **prikazuje** adminu podatke o karticama za odabrani račun. (IA)
7. Admin **poziva** sistem da obriše karticu sa odabranog računa. (APSO)
8. Sistem **prikazuje** adminu poruku: “Kartica uspešno obrisana!” (IA)

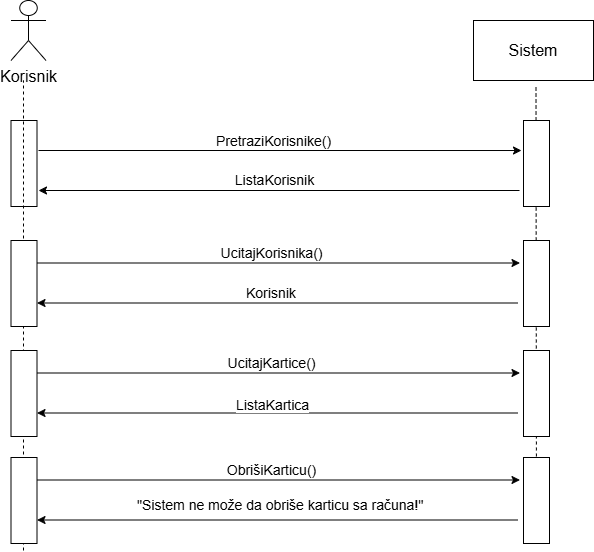


**Alternativni scenariji**

6.1 Ukoliko sistem ne može da prikaže adminu podatke o karticama za odabrani račun, on prikazuje adminu poruku: “Sistem ne može da učita kartice!”. Prekida se izvršenje scenarija. (IA)



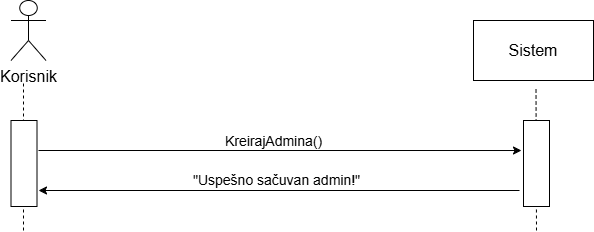
8.1 Ukoliko sistem ne može da obriše karticu sa odabranog računa, on prikazuje adminu poruku: “Sistem ne može da obriše karticu sa odabranog računa!”. Prekida se izvršenje scenarija. (IA)



#### DS9 : Dijagram sekvence slučaja korišćenja - Kreiranje admina:

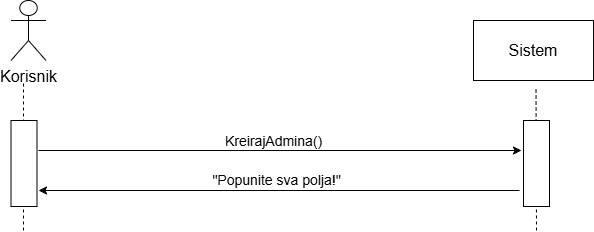
**Osnovni scenario SK**

1. Menadžer **poziva** sistem da zapamti podatke o adminu. (APSO)
2. Sistem prikazuje menadžeru poruku: “Uspešno sačuvan admin!” (IA)



Alternativni scenariji

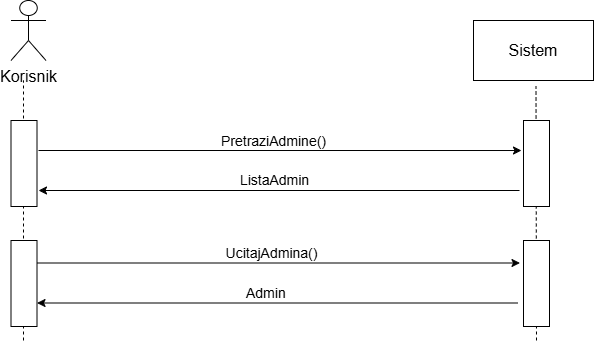
2.1 Ukoliko sistem ne može da zapamti podatke o adminu, on prikazuje korisniku poruku: “Popunite sva polja!”. Prekida se izvršenje scenarija. (IA)



#### DS10 : Dijagram sekvence slučaja korišćenja - Pretraga admina:

**Osnovni scenario SK**

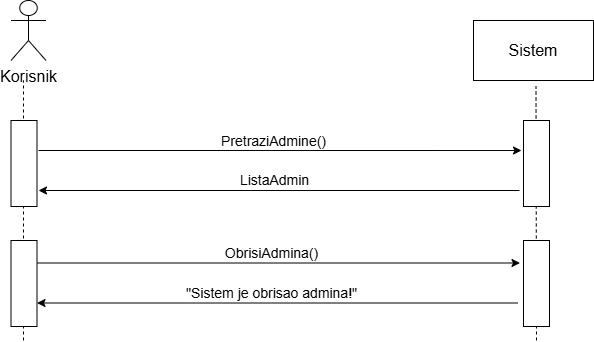
1. Menadžer **poziva** sistem da nađe admine po zadatoj vrednosti. (APSO)
2. Sistem prikazuje menadžeru admine (IA)
3. Menadžer **poziva** sistem da učita podatke o odabranom adminu. (APSO)
4. Sistem **prikazuje** menadžeru podatke o adminu. (IA)



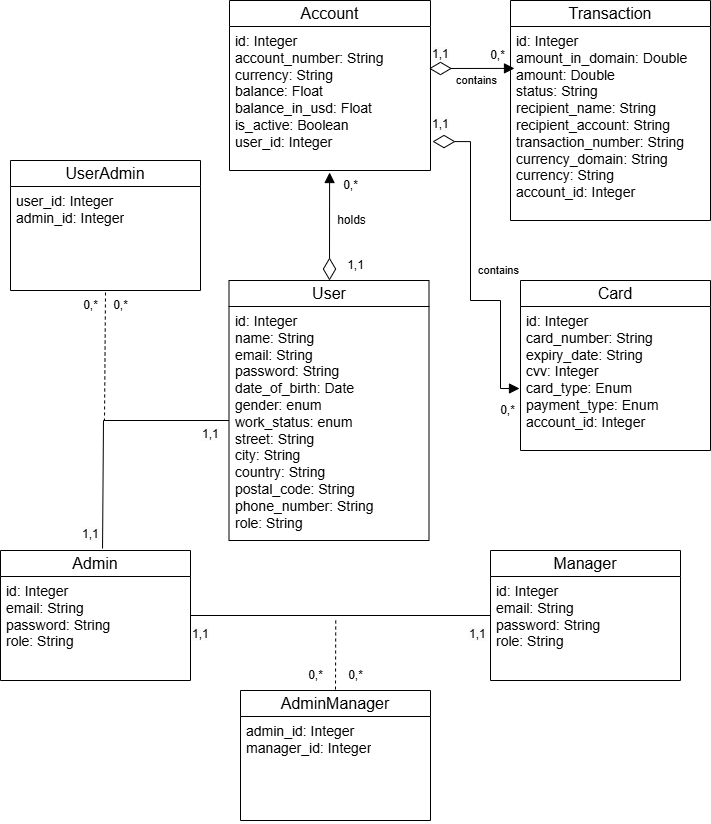
#### DS11 : Dijagram sekvence slučaja korišćenja - Brisanje admina:

**Osnovni scenario SK**

1. Menadžer **poziva** sistem da nađe admine po zadatoj vrednosti. (APSO)
2. Sistem prikazuje menadžeru admine (IA)
3. Menadžer **poziva** sistem da obriše admina. (APSO)
4. Sistem **prikazuje** menadžeru poruku: “Sistem je obrisao admina!”. (IA)



### Struktura softverskog sistema

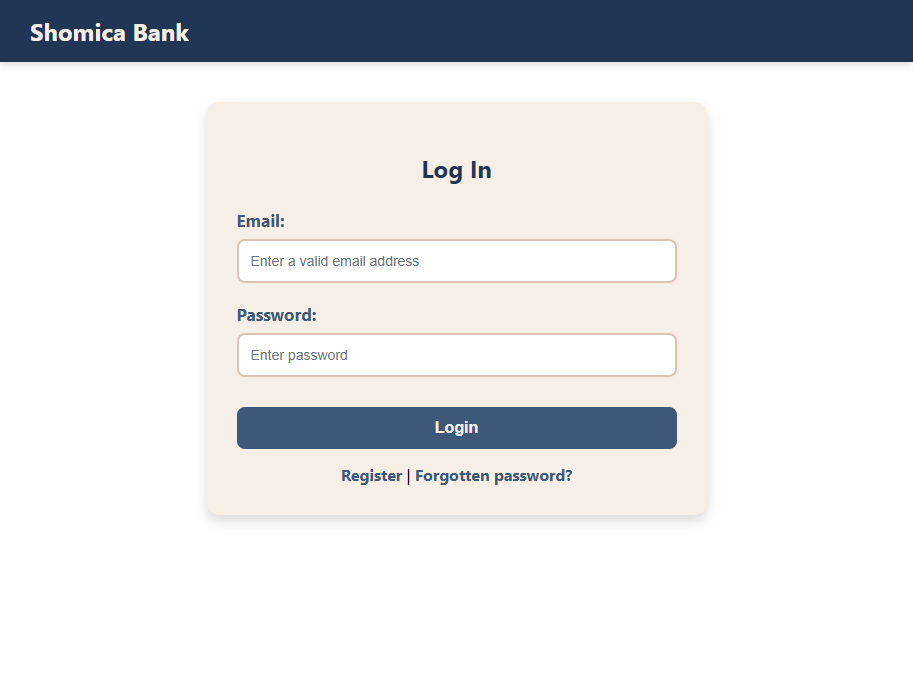


## Faza projektovanja

U ovoj fazi se kreira detaljan dizajn sistema na osnovu rezultata analize. Definiše se korisnički interfejs, aplikaciona logika, koja se sastoji iz kontrolera aplikacione logike, poslovne logike dobijene u procesu analize i brokera baze podataka; i skladište podataka. Pod ovim se podrazumeva da se definišu klase, interfejsi, komponente sistema, baza podataka.

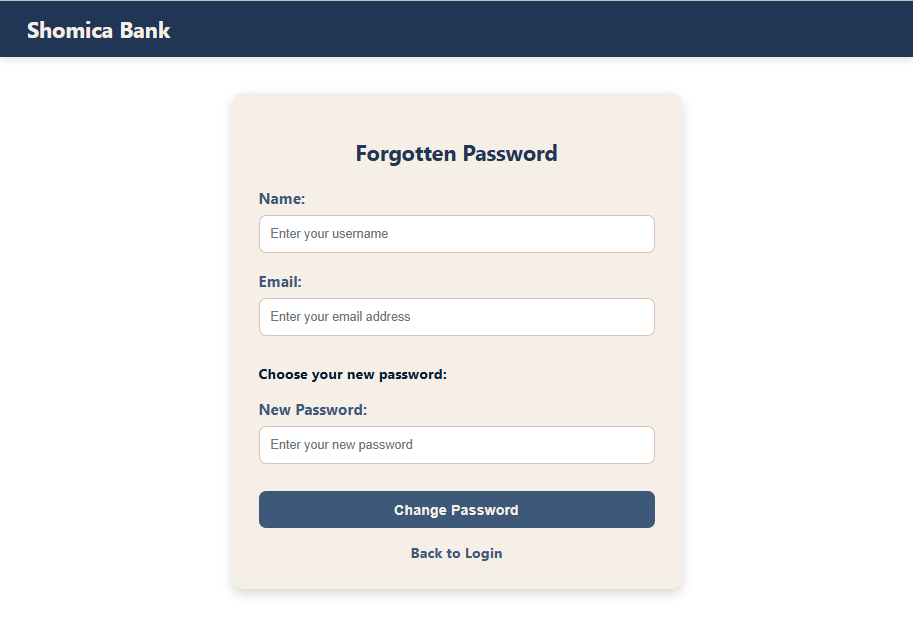
### Projektovanje korisničkog interfejsa (ekstranskih formi)

Prilikom pristupanja aplikaciji, korisnik će uvek biti usmeren ka stranici za prijavljivanje, koja je univerzalna za svaku ulogu u sistemu (ista je za regularnog korisnika, admina i menadžera).

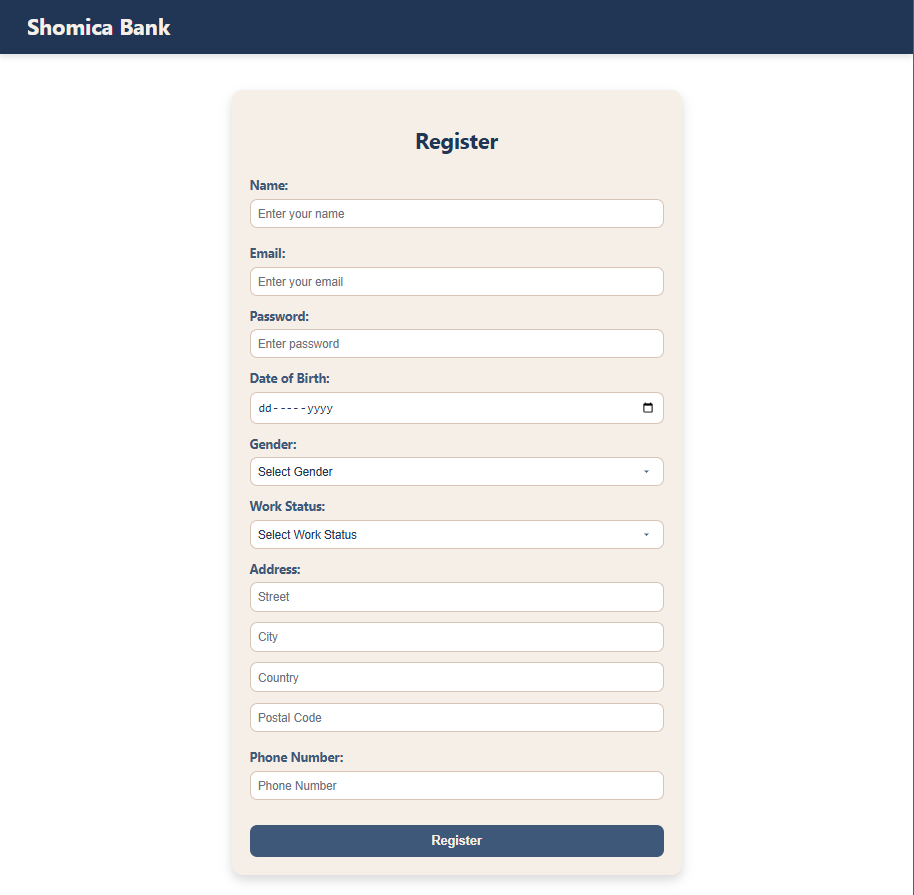


Slika 4 – Login page

Ukoliko korisnik nema prijavljen nalog, pritiskom na dugme “Register” otvara se forma za registrovanje, a ako je, kojim slučajem, zaboravio šifru svog naloga, može promeniti šifru klikom na link “Forgotten password?”.

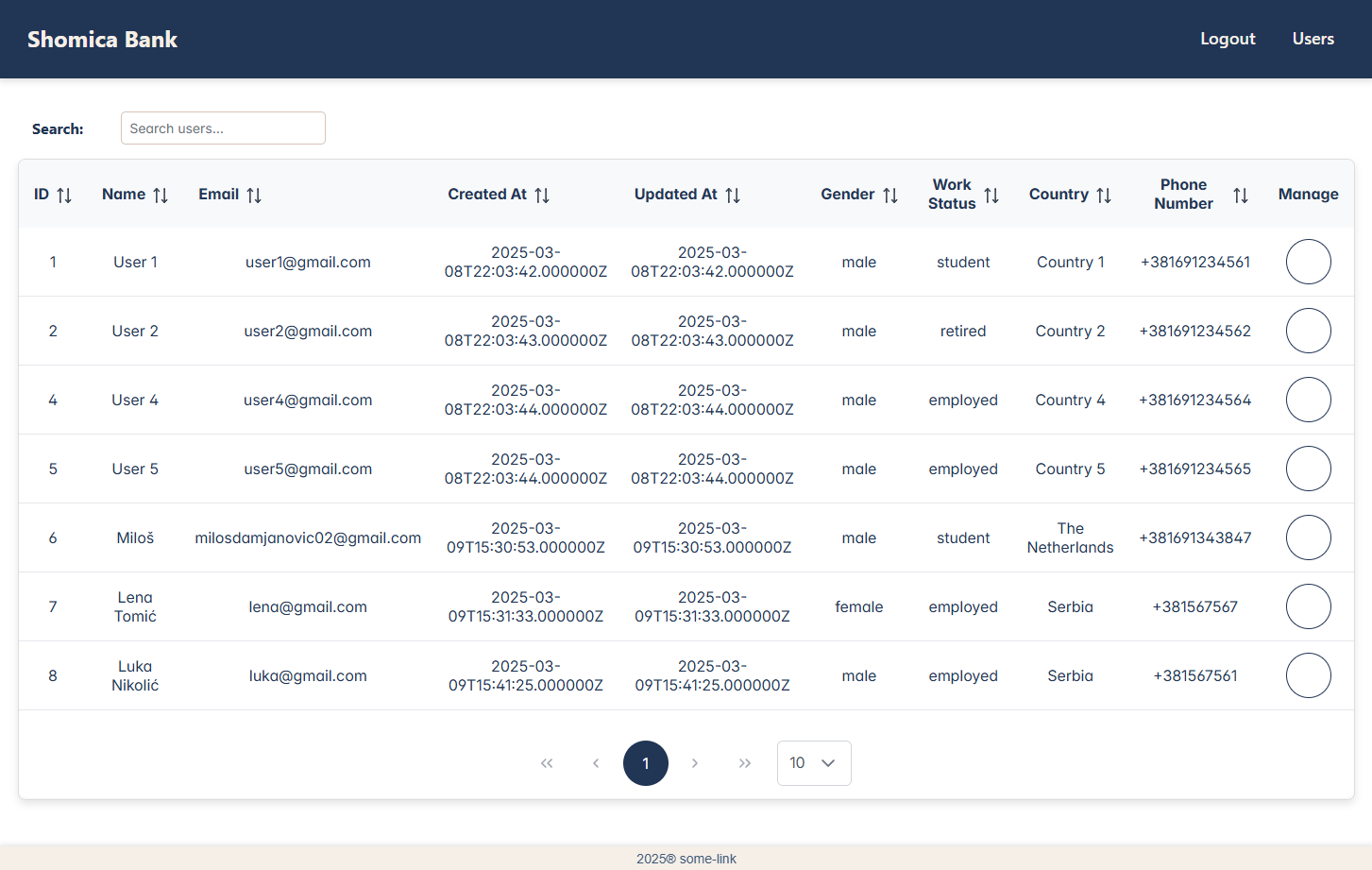


Slika 5 – Forgotten password page



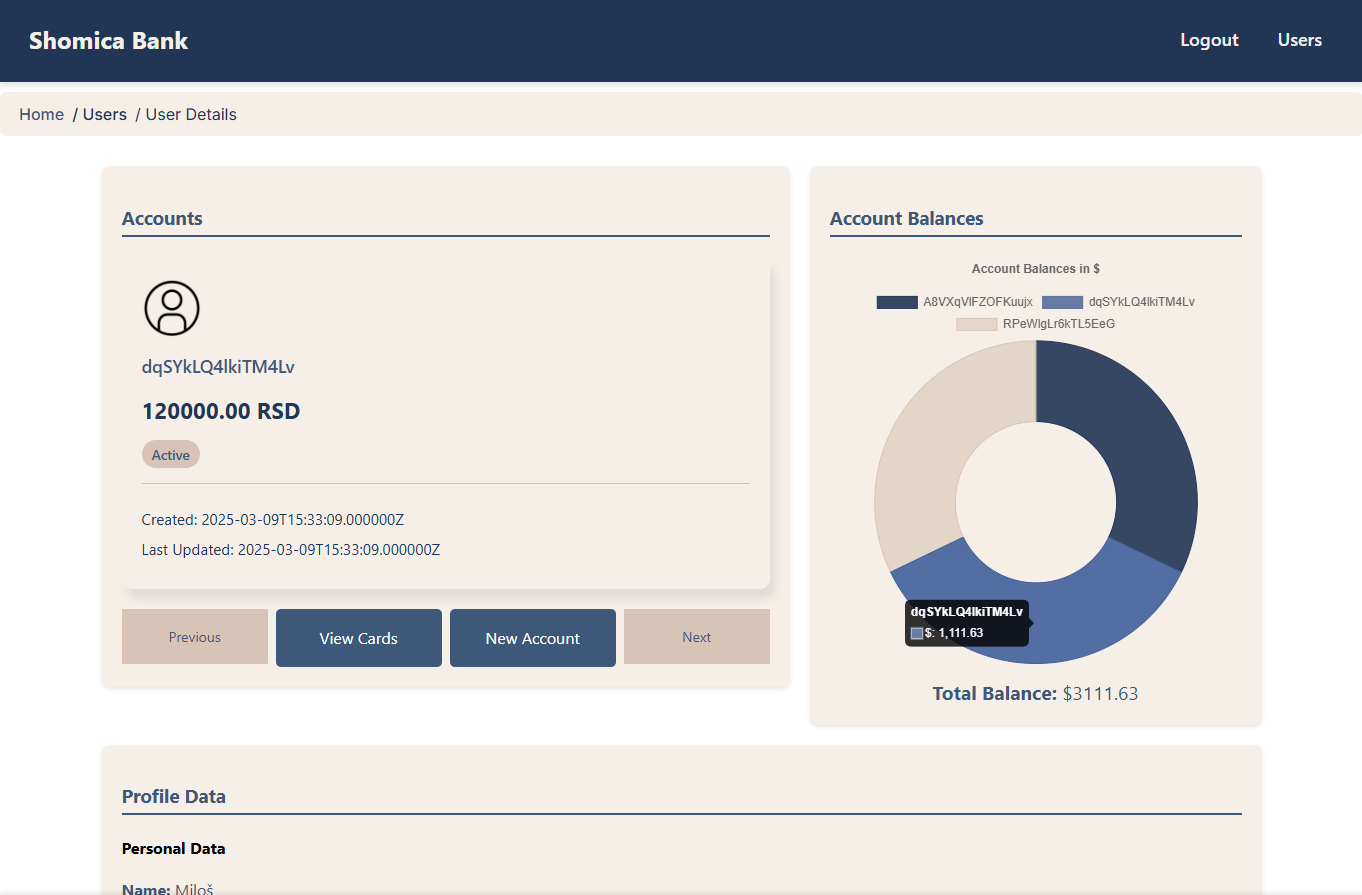
Slika 6 – Register page

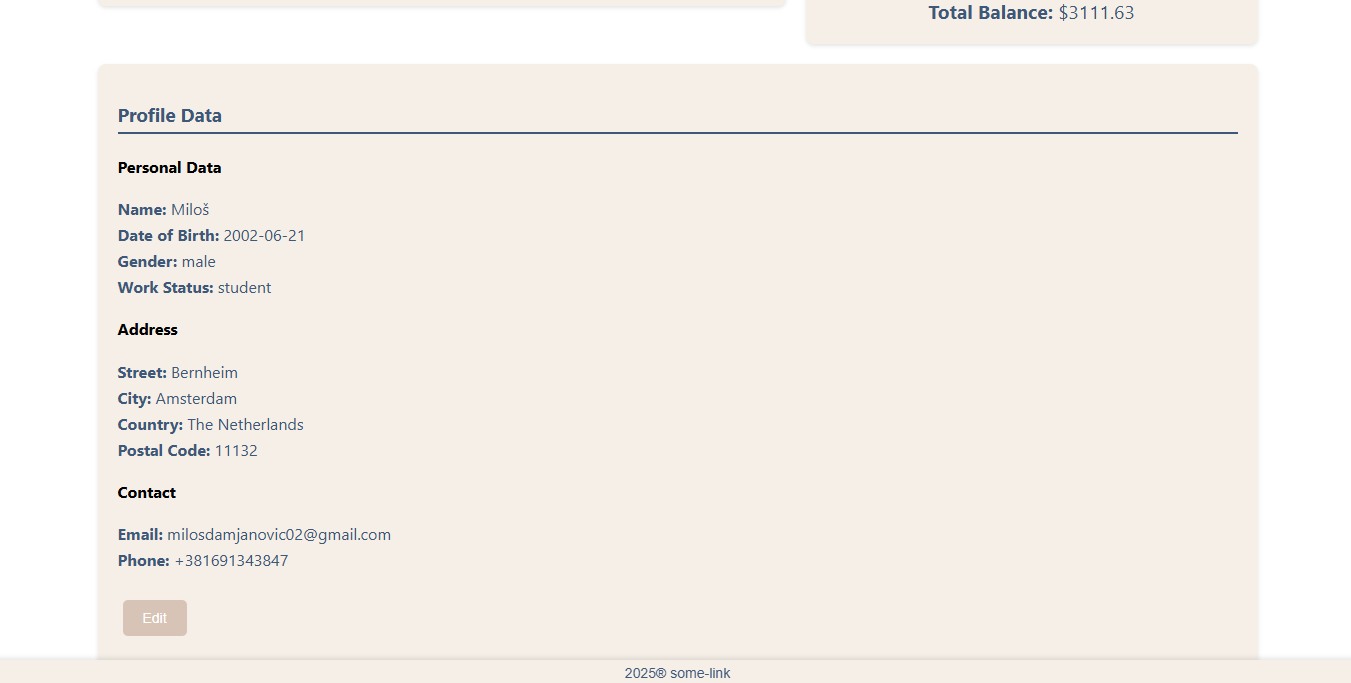
Nakon kreiranja naloga, korisnik na nalogu ne poseduje otvorene račune, a samim tim ni kartice i transakcije. Radi bezbednosti poslovanja, otvaranje računa i kartica može da sprovede samo admin. Korisnik predaje zahtev adminu za otvaranje naloga i platnih kartica, i time počinje postupak otvaranja istih.



Slika 7 – Admin home page

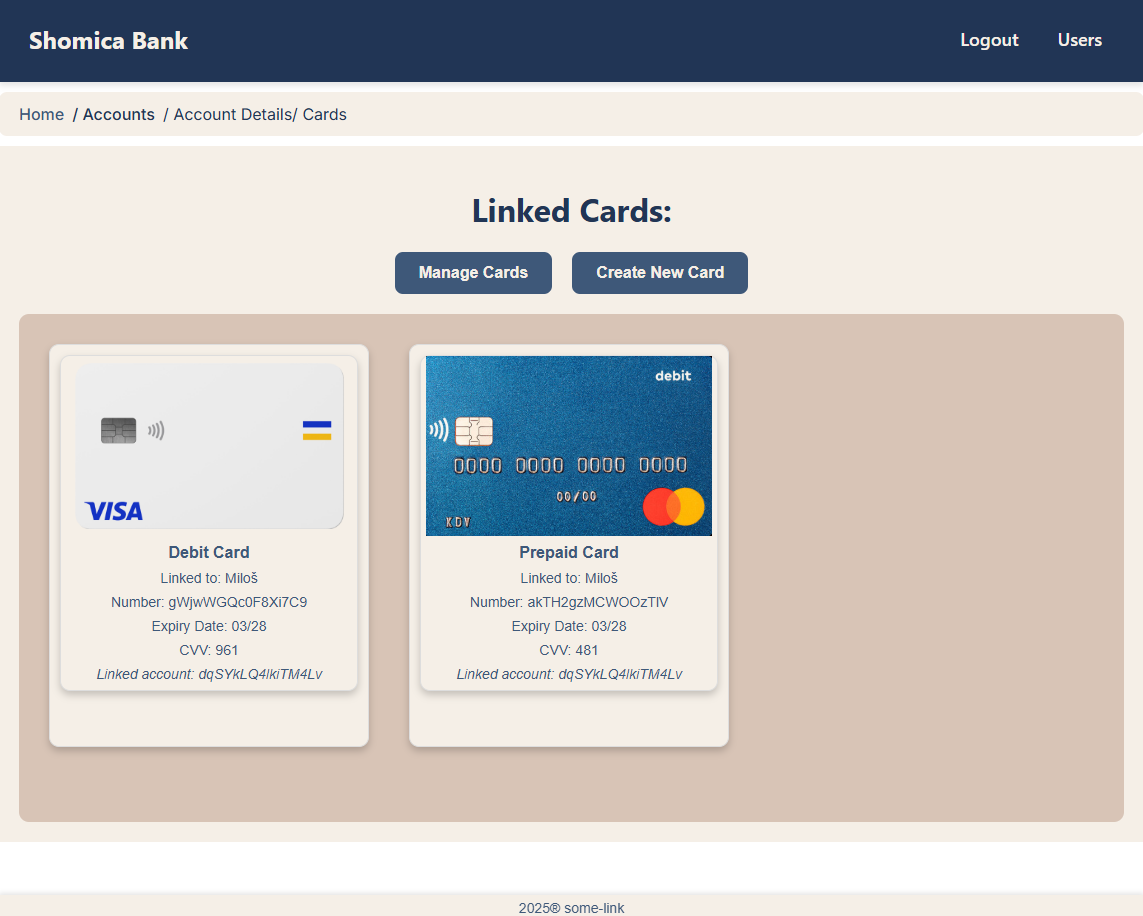
Admin na početnoj stranici dobija uvid u prijavljene korisnike, gde ima pravo da upravlja njihovim podacima. Klikom na određenog korisnika iz tabele, otvara se stranica gde se prikazuju osnovni podaci korisnika.





Slike 8 i 9 – User profile page

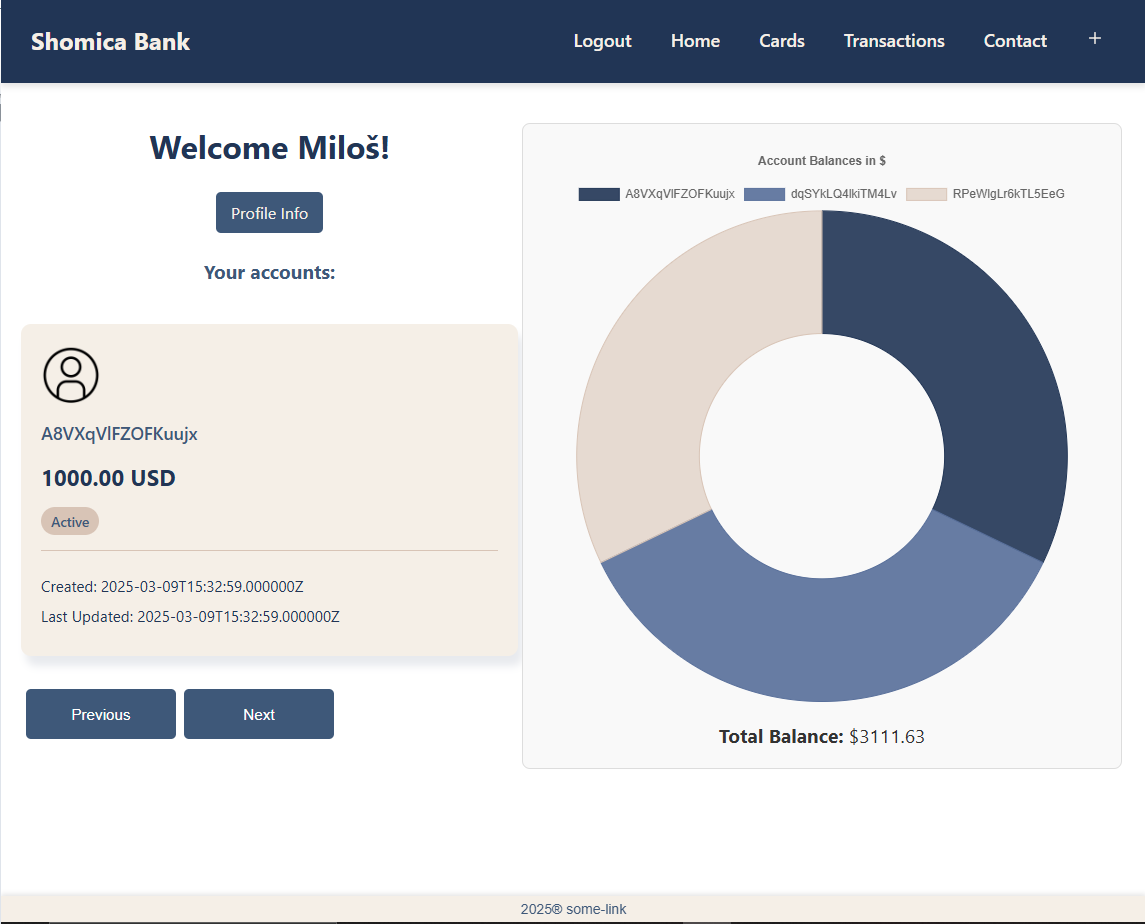
Pritiskom na dugme *New Account* otvara se forma za pravljenje novog naloga, a na dugme *View Cards* se otvara stranica sa prikazom platnih kartica vezanih za trenutno prikazani račun.



Slika 10 – Account Cards page

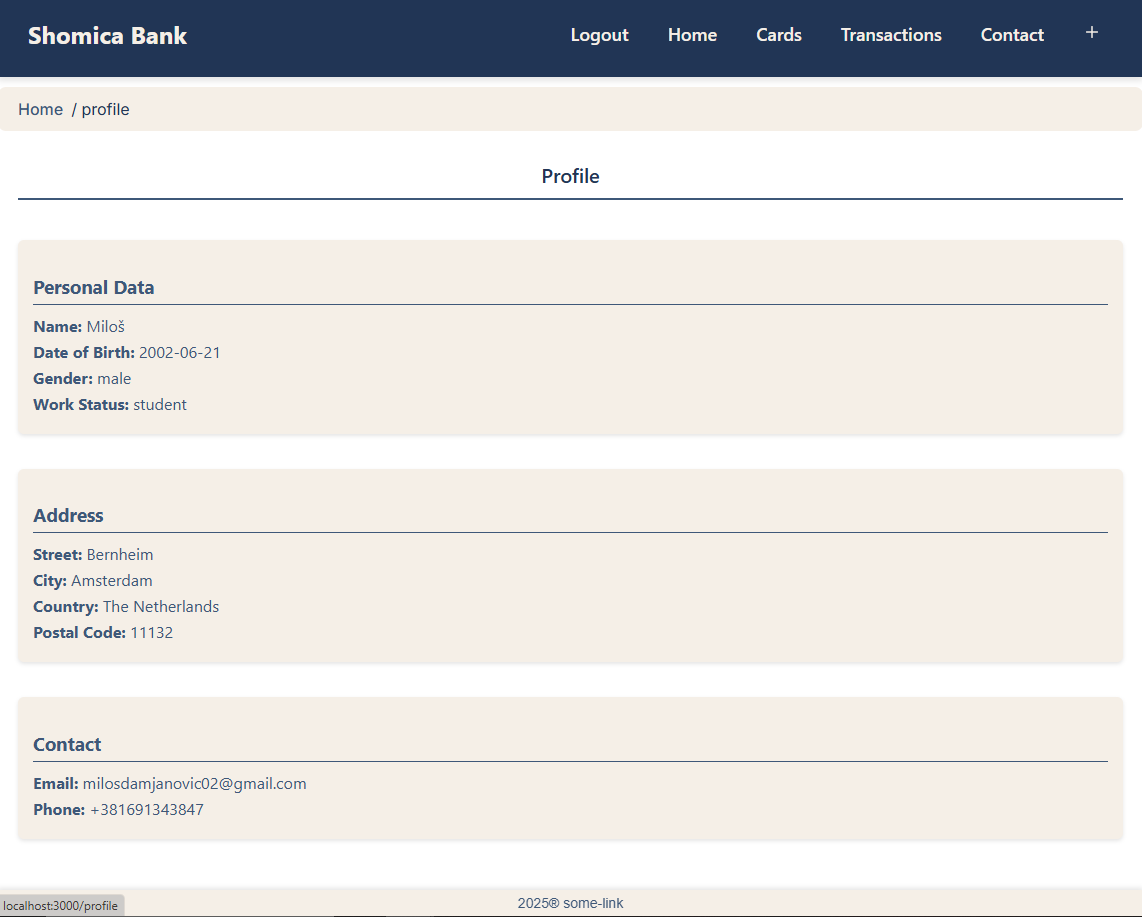
Pritiskom na dugme *Manage Cards* otvara se mogućnost brisanja platnih kartica sa računa, a *Create New Card* otvara formu za pravljenje nove platne kartice.

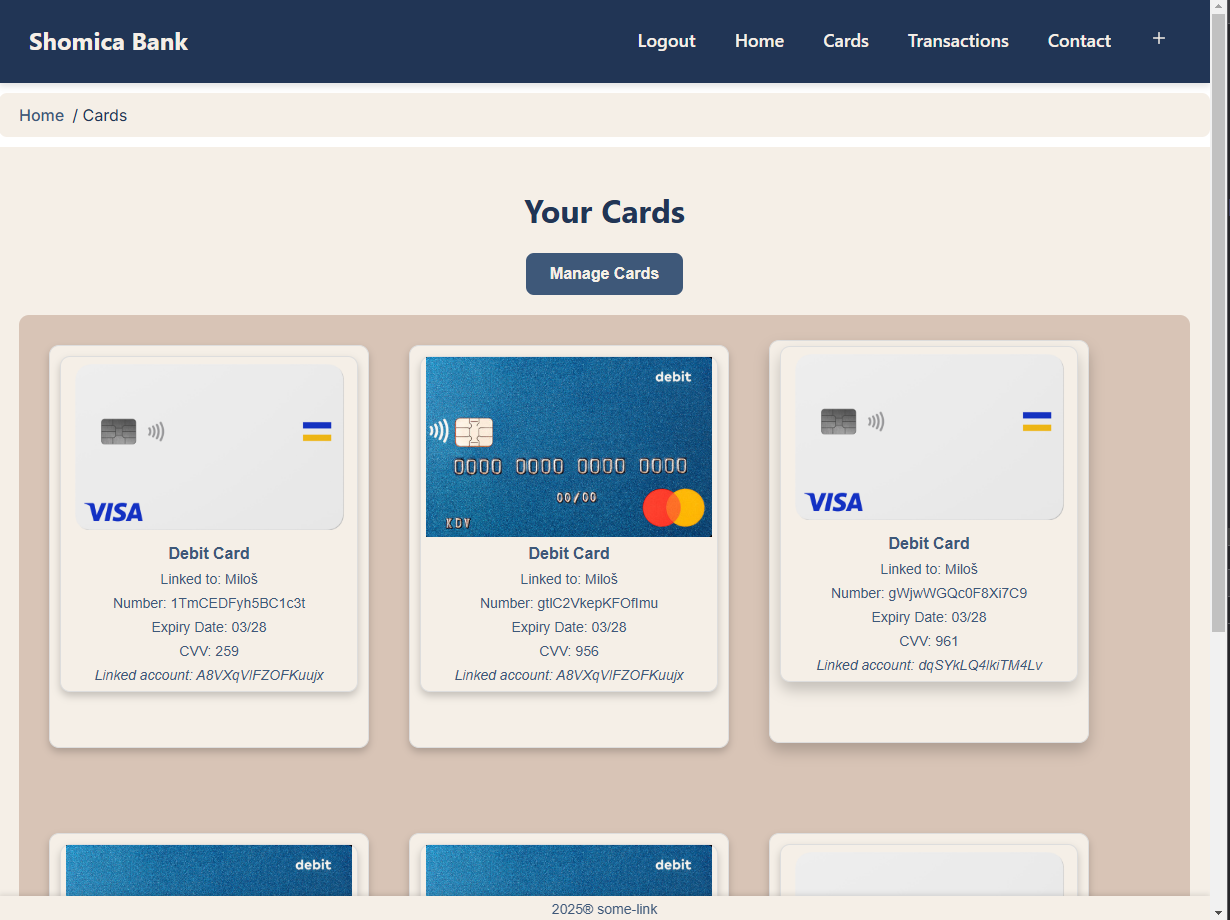
Korisnik prilikom prijavljivanja nailazi Home page sličan adminovom uvidu u profil korisnika.

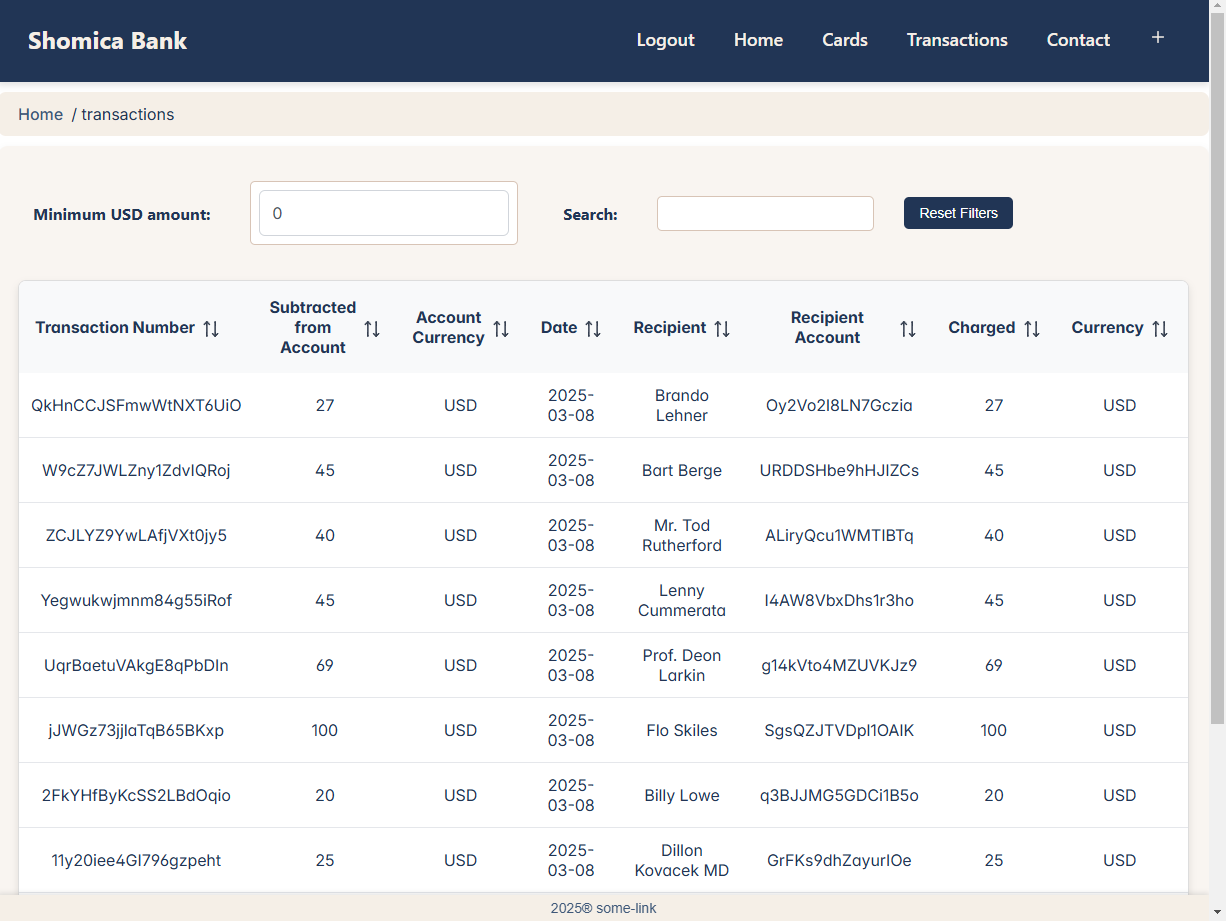


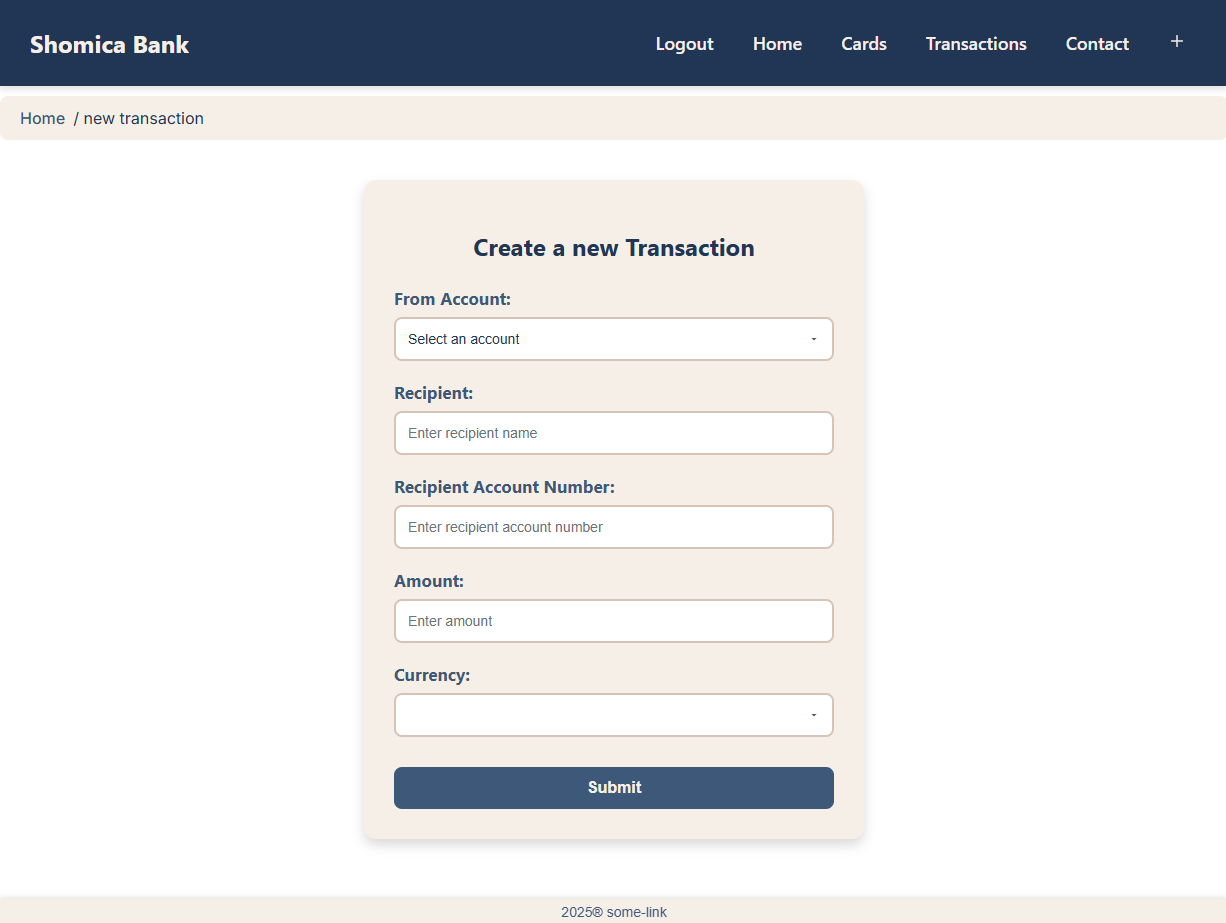
Slika 11 – User Home page

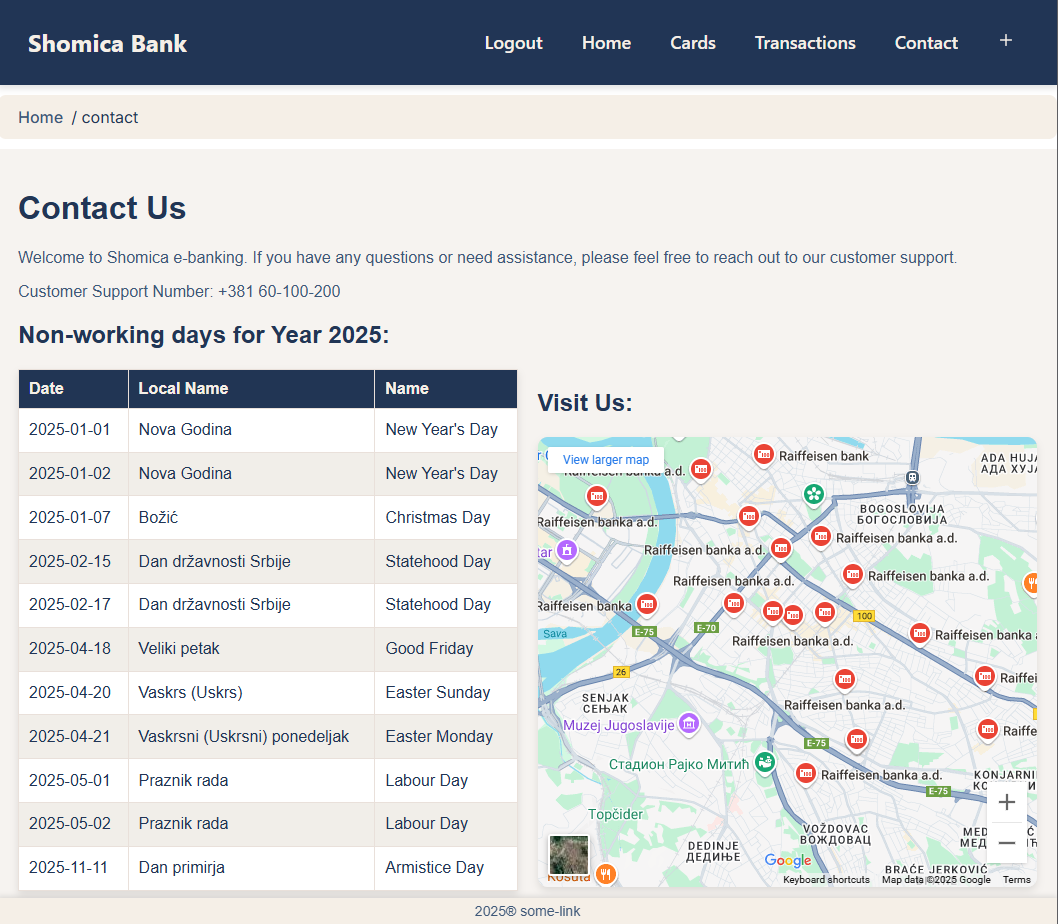
Odavde on ima pristup *Profile Info*, a nadalje svojim platnim karticama, istoriji transakcija, kontakt stranici, kao i formi za kreiranje nove transakcije tj. uplate, čime može da prebacuje materijalna sredstva na jedan od svojih računa, ili na tuđi.





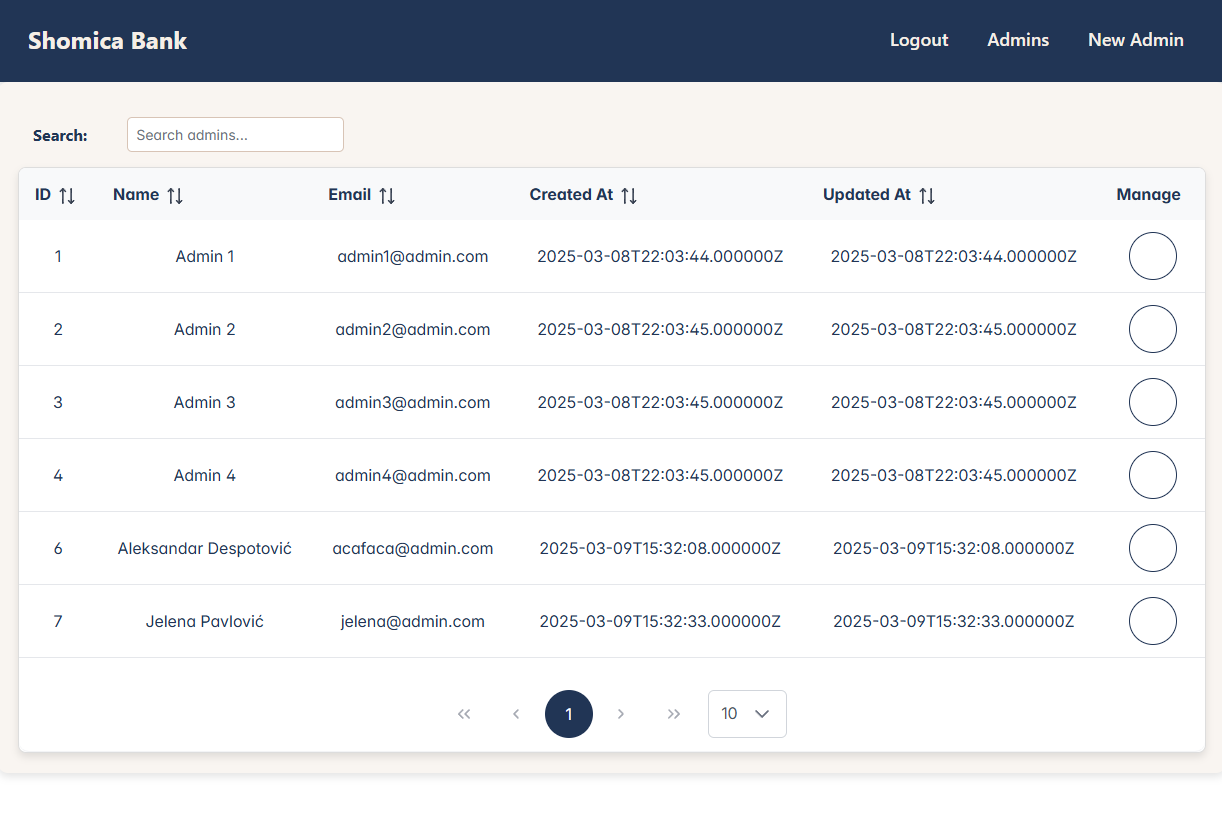




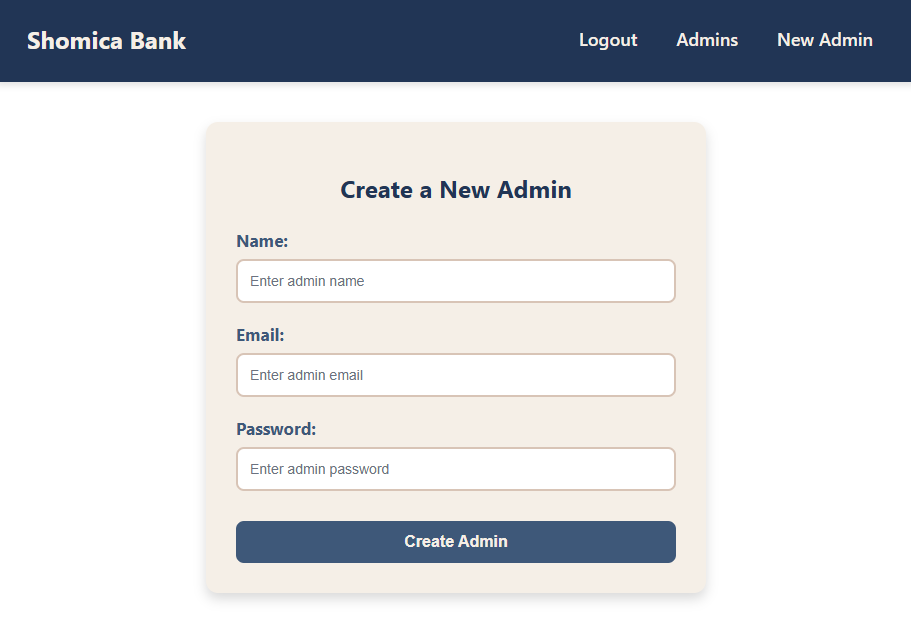


Slike 12, 13, 14, 15, 16 – GUI regularnog korisnika

Što se tiče uloge menadžera, njegova uloga je manipulacija adminima i njihovim podacima. Odmah pri završenoj autentikaciji, na početnoj stranici se pojavljuje lista admina, gde on ima uvid u njihove podatke, i ukoliko zaposleni u ulozi admina da otkaz, ima mogućnost brisanja njegovog naloga.



Slika 17 – Forma za kreiranje novog naloga admina



Slika 17 – Forma za kreiranje novog naloga admina

### Projektovanje aplikacione logike

#### Laravel Kontroleri (Controller)

Kontroleri u Laravelu su implementirani kao klase koje upravljaju logikom aplikacije. Oni primaju HTTP zahteve od korisnika, obrađuju ih i vraćaju odgovarajući odgovor (response – u vidu stranice ili JSON formata). Kontroleri su u ovom projektu organizovani u okviru backend/app/Http/Controllers direktorijuma. Najsloženija logika sadržana je u AccountCardController-u i TransactionController-u:

class AccountCardController extends *Controller*

{

    public function index(*Request* $request)

    {

        $user = $request->user();

        if (!$user) {

            return response()->json(['message' => 'User not authenticated'], 401);

        }

        $accounts = $user->accounts;

*Log*::info('User accounts:', ['accounts' => $accounts]);

        if ($accounts === null) {

            return response()->json(['message' => 'Failed to retrieve accounts for the user'], 500);

        }

        if ($accounts->isEmpty()) {

            return response()->json(['message' => 'No accounts found for the user'], 404);

        }

        // extract account ids from users accounts

        $accountIds = $accounts->pluck('id');

        // query to fetch cards associated with the user's accounts

        $cards = *DB*::table('cards')

            ->join('accounts', 'cards.account\_id', '=', 'accounts.id') // join cards with accounts

            ->join('users', 'accounts.user\_id', '=', 'users.id') // join accounts with users

            ->whereIn('cards.account\_id', $accountIds) // filter by the user's account IDs

            ->select(

                'cards.\*', // select all card fields

                'accounts.account\_number', // include account number

                'users.name as user\_name' // include user name

            )

            ->get();

        if ($cards->isEmpty()) {

            return response()->json(['message' => 'No cards found for the user\'s accounts'], 404);

        }

        return response()->json([

            'message' => 'Cards retrieved successfully',

            'data' => $cards,

        ], 200);

    }

    public function index1(*Request* $request, $accountId)

    {

        $account = *Account*::find($accountId);

        if (!$account) {

            return response()->json(['message' => 'Account not found'], 404);

        }

// double join query

        $cards = *DB*::table('cards')

            ->join('accounts', 'cards.account\_id', '=', 'accounts.id')

            ->join('users', 'accounts.user\_id', '=', 'users.id')

            ->where('cards.account\_id', $accountId)

            ->select(

                'cards.\*',

                'accounts.account\_number',

                'users.name as user\_name'

            )

            ->get();

        if ($cards->isEmpty()) {

            return response()->json(['message' => 'No cards found for this account'], 404);

        }

        return response()->json([

            'message' => 'Cards retrieved successfully',

            'data' => $cards,

        ], 200);

    }

}

TransactionController:  
  
class TransactionController extends *Controller*

{

    public function index(*Request* $request)

    {

        $user = $request->user();

        $transactions = $user->accounts()->with('transactions')->get()->pluck('transactions')->flatten();

        return response()->json([

            'success' => true,

            'data' => $transactions,

        ]);

    }

    public function store(*Request* $request)

    {

*Log*::info('Transaction request received:', $request->all());

        $validator = *Validator*::make($request->all(), [

            'account\_id' => 'required|exists:accounts,id',

            'recipient\_account' => 'required|string',

            'recipient\_name' => 'required|string|max:255',

            'amount' => 'required|numeric|min:0.5',

            'amount\_in\_domain' => 'required|numeric',

            'currency' => 'required|string|size:3',

            'currency\_domain' => 'required|string|size:3',

        ]);

        if ($validator->fails()) {

*Log*::error('Validation failed:', $validator->errors()->toArray());

            return response()->json([

                'success' => false,

                'message' => 'Validation error',

                'errors' => $validator->errors(),

            ], 422);

        }

// beginning of the TRANSACTION

*DB*::beginTransaction();

        try {

            // fetch sender account

*Log*::info('Fetching sender account:', ['account\_id' => $request->account\_id]);

            $fromAccount = *Account*::findOrFail($request->account\_id);

            // sufficient fund check

            if ($fromAccount->balance < $request->amount\_in\_domain) {

*Log*::error('Insufficient funds:', ['account\_id' => $request->account\_id, 'balance' => $fromAccount->balance, 'amount' => $request->amount]);

                return response()->json([

                    'success' => false,

                    'message' => 'Insufficient funds.',

                ], 400);

            }

            // fetch recipient account

*Log*::info('Fetching recipient account:', ['recipient\_account' => $request->recipient\_account]);

            $recipientAccount = *Account*::where('account\_number', $request->recipient\_account)->first();

            if ($recipientAccount == $fromAccount) {

                return response()->json([

                    'success' => false,

                    'message' => 'You cannot send money to yourself!',

                    'error' => 'You cannot send money to yourself!',

                ], 500);

            }

            // convert the transaction amount to the recipient's currency

            $exchangeRates = *$this*->fetchExchangeRates();

            $amountInUSD = $request->amount / $exchangeRates[$request->currency];

            $amountInRecipientCurrency = $amountInUSD \*

                ($recipientAccount ? $exchangeRates[$recipientAccount->currency] : $exchangeRates[$request->currency\_domain]);

            // updating user's balance

            $fromAccount->balance -= $request->amount\_in\_domain;

            $fromAccount->balance\_in\_usd -= $amountInUSD;

            $fromAccount->save();

*Log*::info('Sender account updated:', ['account\_id' => $fromAccount->id, 'new\_balance' => $fromAccount->balance, 'new\_balance\_in\_usd' => $fromAccount->balance\_in\_usd]);

            // if recipient account exists in DB, update its balance

            if ($recipientAccount) {

                $recipientAccount->balance += $amountInRecipientCurrency;

                $recipientAccount->balance\_in\_usd += $amountInUSD;

                $recipientAccount->save();

*Log*::info('Recipient account updated:', ['account\_id' => $recipientAccount->id, 'new\_balance' => $recipientAccount->balance, 'new\_balance\_in\_usd' => $recipientAccount->balance\_in\_usd]);

            } else {

*Log*::info('Recipient account not found in local database. Proceeding with transaction.');

            }

            $transaction = *Transaction*::create([

                'account\_id' => $request->account\_id,

                'recipient\_name' => $request->recipient\_name,

                'recipient\_account' => $request->recipient\_account,

                'amount' => $request->amount,

                'amount\_in\_domain' => $request->amount\_in\_domain,

                'currency' => $request->currency,

                'currency\_domain' => $request->currency\_domain,

                'status' => 'completed',

                'transaction\_number' => *Str*::uuid(),

            ]);

*Log*::info('Transaction created:', $transaction->toArray());

            // removing cached data

            $cacheKey = 'user\_profile\_' . auth()->id();

*Cache*::forget($cacheKey);

*DB*::commit();

            return response()->json([

                'success' => true,

                'message' => 'Transaction successful.',

                'data' => $transaction,

            ], 201);

        } catch (*\Exception* $e) {

*DB*::rollBack();

*Log*::error('Transaction failed:', ['error' => $e->getMessage()]);

            return response()->json([

                'success' => false,

                'message' => 'Transaction failed.',

                'error' => $e->getMessage(),

            ], 500);

        }

    }

    protected function fetchExchangeRates()

    {

        $response = *Http*::get('https://api.exchangerate-api.com/v4/latest/USD');

        if ($response->successful()) {

            return $response->json()['rates'];

        }

        throw new *\Exception*('Failed to fetch exchange rates.');

    }

}

#### Laravel Autentifikacija (Authentication)

Laravel pruža ugrađen sistem za autentifikaciju korisnika, kojim se podrazumeva registracija, prijava, odjava. Laravel koristi JWT (JSON Web Token) ili Session-based autentifikaciju. Mnoge funkcionalnosti autentifikacije omogućene su adekvatnim podešavanjem guard-ova i provider-a. Sledeća podešavanja su uspostavljena unutar backend/config/auth.php datoteke.

'guards' => [

        'web' => [

            'driver' => 'session',

            'provider' => 'users',

        ],

        'api' => [

            'driver' => 'sanctum',

            'provider' => 'users',

        ],

        'admin' => [

            'driver' => 'session',

            'provider' => 'admins',

        ],

        'admin-api' => [

            'driver' => 'sanctum',

            'provider' => 'admins',

        ],

        'manager' => [

            'driver' => 'session',

            'provider' => 'managers',

        ],

        'manager-api' => [

            'driver' => 'sanctum',

            'provider' => 'managers',

        ],

    ],

    'providers' => [

        'users' => [

            'driver' => 'eloquent',

            'model' => *App\Models\User*::class,

        ],

        'admins' => [

            'driver' => 'eloquent',

            'model' => *App\Models\Admin*::class,

        ],

        'managers' => [

            'driver' => 'eloquent',

            'model' => *App\Models\Manager*::class,

        ],

    ],

#### Laravel Resursi (Resources)

#### Resursi u Laravel-u služe za transformaciju modela u JSON odgovor. Oni omogućavaju da se definiše struktura i izgled API odgovora (response). Za ovaj projekat značajni su resursi vezani za korisnike i njihove račune.

class UserResource extends *JsonResource*

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @return array<string, mixed>

     \*/

    public static $wrap = 'user';

    public function toArray(*Request* $request): array

    {

        return [

            'id'=> *$this*->resource->id,

            'name'=>*$this*->resource->name,

            'email'=>*$this*->resource->email

        ];

    }

}

class AccountResource extends *JsonResource*

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @return array<string, mixed>

     \*/

    public static $wrap = 'account';

    public function toArray(*Request* $request): array

    {

        return [

            'id' => *$this*->resource->id,

            'user'=> new *UserResource*(*$this*->resource->user),

            'account\_number' => *$this*->resource->account\_number,

            'currency' => *$this*->resource->currency,

            'balance' =>*$this*->resource->balance,

            'is\_active'=>*$this*->resource->is\_active,

        ];

    }

}

#### Laravel Rutiranje (Api routes)

Rutiranje u Laravel-u definiše kako će se HTTP zahtevi mapirati na odgovarajuće kontrolere. API rute su definisane u backend/routes/api.php fajlu. U narednim tabelama prikazaću response koji axios dobija iz kontrolera fetchovanjem podataka, a zatim ispisuje u konzoli.

|  |  |
| --- | --- |
| Opis funkcije | Prijavljivanje korisnika |
| HTTP metoda | POST |
| URL | /api/auth/login |
| HTTP body parametri | **data**: {id: 6, name: 'Miloš', email: 'milosdamjanovic02@gmail.com', email\_verified\_at: null, created\_at: '2025-03-09T15:30:53.000000Z', …}  **message**: "User Logged In Successfully"  **role**: "user"  **success**: true  **token**: "14|bq73H7cfrq611kwhMZ0pXxF5F0ny4eQn0BaHcFtDc27507bf" |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Registrovanje korisnika |
| HTTP metoda | POST |
| URL | /api/auth/register |
| HTTP body parametri | **data**: {name: 'Anđela Panić', email: 'andjela@gmail.com', date\_of\_birth: '2002-05-02', gender: 'female', work\_status: 'student', …}  **message**: "User Created Successfully"  **success**: true  **token**: "13|ncapZD4ySCQyHJV6BJphSWREruQ3WJojSQBihISV6207db6c" |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Odjavljivanje korisnika |
| HTTP metoda | POST |
| URL | /api/auth/logout |
| HTTP body parametri | **{"success":true,"message":"User logged out successfully."}** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Profil korisnika |
| HTTP metoda | GET |
| URL | /api/profile |
| HTTP body parametri | 1. *{* 2. *"message": "Welcome to user profile",* 3. *"user-data": {* 4. *"id": 6,* 5. *"name": "Miloš",* 6. *"email": "milosdamjanovic02@gmail.com",* 7. *"email\_verified\_at": null,* 8. *"created\_at": "2025-03-09T15:30:53.000000Z",* 9. *"updated\_at": "2025-03-09T15:30:53.000000Z",* 10. *"date\_of\_birth": "2002-06-21",* 11. *"gender": "male",* 12. *"work\_status": "student",* 13. *"street": "Bernheim",* 14. *"city": "Amsterdam",* 15. *"country": "The Netherlands",* 16. *"postal\_code": "11132",* 17. *"phone\_number": "+381691343847",* 18. *"role": "user",* 19. *"accounts": [* 20. *{* 21. *"id": 16,* 22. *"account\_number": "A8VXqVlFZOFKuujx",* 23. *"user\_id": 6,* 24. *"currency": "USD",* 25. *"balance": "1000.00",* 26. *"is\_active": 1,* 27. *"created\_at": "2025-03-09T15:32:59.000000Z",* 28. *"updated\_at": "2025-03-09T15:32:59.000000Z",* 29. *"balance\_in\_usd": "1000.00"* 30. *},* 31. *{* 32. *"id": 17,* 33. *"account\_number": "dqSYkLQ4lkiTM4Lv",* 34. *"user\_id": 6,* 35. *"currency": "RSD",* 36. *"balance": "120000.00",* 37. *"is\_active": 1,* 38. *"created\_at": "2025-03-09T15:33:09.000000Z",* 39. *"updated\_at": "2025-03-09T15:33:09.000000Z",* 40. *"balance\_in\_usd": "1111.63"* 41. *},* 42. *{* 43. *"id": 18,* 44. *"account\_number": "RPeWlgLr6kTL5EeG",* 45. *"user\_id": 6,* 46. *"currency": "GEL",* 47. *"balance": "2780.00",* 48. *"is\_active": 1,* 49. *"created\_at": "2025-03-09T15:33:36.000000Z",* 50. *"updated\_at": "2025-03-09T15:33:36.000000Z",* 51. *"balance\_in\_usd": "1000.00"* 52. *}* 53. *]* 54. *}* 55. *}* |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Kartice korisnika |
| HTTP metoda | GET |
| URL | /api/profile/cards |
| HTTP body parametri | **{**  **"message": "Cards retrieved successfully",**  **"data": [**  **{**  **"id": 46,**  **"card\_number": "1TmCEDFyh5BC1c3t",**  **"expiry\_date": "03/28",**  **"cvv": 259,**  **"account\_id": 16,**  **"created\_at": "2025-03-09 15:33:50",**  **"updated\_at": "2025-03-09 15:33:50",**  **"card\_type": "debit",**  **"payment\_type": "visa",**  **"account\_number": "A8VXqVlFZOFKuujx",**  **"user\_name": "Miloš"**  **},**  **{**  **"id": 47,**  **"card\_number": "gtlC2VkepKFOfImu",**  **"expiry\_date": "03/28",**  **"cvv": 956,**  **"account\_id": 16,**  **"created\_at": "2025-03-09 15:33:54",**  **"updated\_at": "2025-03-09 15:33:54",**  **"card\_type": "debit",**  **"payment\_type": "mastercard",**  **"account\_number": "A8VXqVlFZOFKuujx",**  **"user\_name": "Miloš"**  **},**  **...**  **]**  **}** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje kartice |
| HTTP metoda | DELETE |
| URL | /api/delete-card |
| HTTP body parametri | **"Card is deleted successfully."** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Transakcije korisnika |
| HTTP metoda | GET |
| URL | /api/profile/transactions |
| HTTP body parametri | **{**  **"message": "Transactions retrieved successfully",**  **"data": [**  **{**  **"id": 226,**  **"amount": 52,**  **"created\_at": "2025-03-09",**  **"updated\_at": "2025-03-09T21:29:41.000000Z",**  **"status": "completed",**  **"account\_id": 16,**  **"recipient\_name": "Jahoda Andrej",**  **"recipient\_account": "12O0p3Lsr6B1ebXx",**  **"transaction\_number": "d7e55654-6175-4a30-bf72-a43becf6896c",**  **"currency": "USD",**  **"amount\_in\_domain": "52.25",**  **"currency\_domain": "USD"**  **},**  **{**  **"id": 229,**  **"amount": 24,**  **"created\_at": "2025-03-09",**  **"updated\_at": "2025-03-09T21:33:28.000000Z",**  **"status": "completed",**  **"account\_id": 16,**  **"recipient\_name": "Bill Gates",**  **"recipient\_account": "E3Z8xF7VPSCSGJNc",**  **"transaction\_number": "f4862a37-5e85-4829-ab2a-0ff3f1c8f3b5",**  **"currency": "USD",**  **"amount\_in\_domain": "23.75",**  **"currency\_domain": "USD"**  **},**  **{**  **"id": 227,**  **"amount": 56,**  **"created\_at": "2025-03-09",**  **"updated\_at": "2025-03-09T21:31:16.000000Z",**  **"status": "completed",**  **"account\_id": 17,**  **"recipient\_name": "Lena Tomić",**  **"recipient\_account": "RhBf7n6fvDic3v6i",**  **"transaction\_number": "fc63dd2f-1c3b-417f-8d84-ee7c645dc88c",**  **"currency": "GEL",**  **"amount\_in\_domain": "2154.68",**  **"currency\_domain": "RSD"**  **},**  **]**  **}** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Pravljenje nove transakcije |
| HTTP metoda | POST |
| URL | /api/new-transaction |
| HTTP body parametri | **{**  **"success": true,**  **"message": "Transaction successful.",**  **"data": {**  **"account\_id": 18,**  **"recipient\_name": "Aleksandar Despotovic",**  **"recipient\_account": "E17ZaXhb5rd3azwc",**  **"amount": "9000",**  **"amount\_in\_domain": "231.77",**  **"currency": "RSD",**  **"currency\_domain": "GEL",**  **"status": "completed",**  **"transaction\_number": "eeec11f9-bb22-4a6b-bfae-4c7f8791d495",**  **"updated\_at": "2025-03-09T21:41:38.000000Z",**  **"created\_at": "2025-03-09",**  **"id": 231**  **}**  **}** |
| Format HTTP body parametara | JSON |

Slično izgleda i ruta kreiranja novog naloga u slučaju admina.

|  |  |
| --- | --- |
| Opis funkcije | Dobijanje kursne liste |
| HTTP metoda | GET |
| URL | /api/exchange-rates |
| HTTP body parametri | **{**  **"success": true,**  **"rates": {**  **"USD": 1,**  **"AED": 3.67,**  **"AFN": 73.04,**  **"ALL": 92.34,**  **"AMD": 394.32,**  **"ANG": 1.79,**  **"AOA": 918.64,**  **"ARS": 1066.04,**  **"AUD": 1.59,**  **"AWG": 1.79,**  **"AZN": 1.7,**  **"BAM": 1.8,**  **...**  **}**  **}** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Fetch liste admina |
| HTTP metoda | GET |
| URL | /api/admins |
| HTTP body parametri | **[**  **{**  **"id": 4,**  **"name": "Admin 4",**  **"email": "admin4@admin.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-08T22:03:45.000000Z",**  **"updated\_at": "2025-03-08T22:03:45.000000Z",**  **"role": "admin"**  **},**  **{**  **"id": 6,**  **"name": "Aleksandar Despotović",**  **"email": "acafaca@admin.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T15:32:08.000000Z",**  **"updated\_at": "2025-03-09T15:32:08.000000Z",**  **"role": "admin"**  **},**  **{**  **"id": 7,**  **"name": "Jelena Pavlović",**  **"email": "jelena@admin.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T15:32:33.000000Z",**  **"updated\_at": "2025-03-09T15:32:33.000000Z",**  **"role": "admin"**  **}**  **]** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje admina iz baze |
| HTTP metoda | DELETE |
| URL | /api/admins/{id} |
| HTTP body parametri | **'User deleted successfully.'** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Fetchovanje korisnika iz baze |
| HTTP metoda | GET |
| URL | api/users |
| HTTP body parametri | **[**  **{**  **"id": 6,**  **"name": "Miloš",**  **"email": "milosdamjanovic02@gmail.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T15:30:53.000000Z",**  **"updated\_at": "2025-03-09T15:30:53.000000Z",**  **"date\_of\_birth": "2002-06-21",**  **"gender": "male",**  **"work\_status": "student",**  **"street": "Bernheim",**  **"city": "Amsterdam",**  **"country": "The Netherlands",**  **"postal\_code": "11132",**  **"phone\_number": "+381691343847",**  **"role": "user"**  **},**  **{**  **"id": 7,**  **"name": "Lena Tomić",**  **"email": "lena@gmail.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T15:31:33.000000Z",**  **"updated\_at": "2025-03-09T15:31:33.000000Z",**  **"date\_of\_birth": "1999-05-12",**  **"gender": "female",**  **"work\_status": "employed",**  **"street": "Tomazeova",**  **"city": "Belgrade",**  **"country": "Serbia",**  **"postal\_code": "11175",**  **"phone\_number": "+381567567",**  **"role": "user"**  **},**  **{**  **"id": 8,**  **"name": "Luka Nikolić",**  **"email": "luka@gmail.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T15:41:25.000000Z",**  **"updated\_at": "2025-03-09T15:41:25.000000Z",**  **"date\_of\_birth": "2000-09-22",**  **"gender": "male",**  **"work\_status": "employed",**  **"street": "Kralja Milana 123",**  **"city": "Belgrade",**  **"country": "Serbia",**  **"postal\_code": "11125",**  **"phone\_number": "+381567561",**  **"role": "user"**  **},**  **{**  **"id": 9,**  **"name": "Anđela Panić",**  **"email": "andjela@gmail.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T20:52:31.000000Z",**  **"updated\_at": "2025-03-09T20:52:31.000000Z",**  **"date\_of\_birth": "2002-05-02",**  **"gender": "female",**  **"work\_status": "student",**  **"street": "Novobeogradska 10",**  **"city": "Belgrade",**  **"country": "Serbia",**  **"postal\_code": "11107",**  **"phone\_number": "+381527561",**  **"role": "user"**  **}**  **]** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Brisanje korisnika iz baze |
| HTTP metoda | DELETE |
| URL | /api/users /{id} |
| HTTP body parametri | **'User deleted successfully.'** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Update korisnika |
| HTTP metoda | PUT |
| URL | /api/admins/{id} |
| HTTP body parametri | **[**  **“message” : “User updated successfully”,**  **“user” :**  **{**  **"id": 6,**  **"name": "Miloš",**  **"email": "milosdamjanovic02@gmail.com",**  **"email\_verified\_at": null,**  **"created\_at": "2025-03-09T15:30:53.000000Z",**  **"updated\_at": "2025-03-09T15:30:53.000000Z",**  **"date\_of\_birth": "2002-06-21",**  **"gender": "male",**  **"work\_status": "student",**  **"street": "Bernheim",**  **"city": "Amsterdam",**  **"country": "The Netherlands",**  **"postal\_code": "11132",**  **"phone\_number": "+381691343847",**  **"role": "user"**  **}**  **]** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Pregled kartica za izabrani račun |
| HTTP metoda | GET |
| URL | /api/accounts/{id}/cards |
| HTTP body parametri | **{**  **"message": "Cards retrieved successfully",**  **"data": [**  **{**  **"id": 46,**  **"card\_number": "1TmCEDFyh5BC1c3t",**  **"expiry\_date": "03/28",**  **"cvv": 259,**  **"account\_id": 16,**  **"created\_at": "2025-03-09 15:33:50",**  **"updated\_at": "2025-03-09 15:33:50",**  **"card\_type": "debit",**  **"payment\_type": "visa",**  **"account\_number": "A8VXqVlFZOFKuujx",**  **"user\_name": "Miloš"**  **},**  **{**  **"id": 47,**  **"card\_number": "gtlC2VkepKFOfImu",**  **"expiry\_date": "03/28",**  **"cvv": 956,**  **"account\_id": 16,**  **"created\_at": "2025-03-09 15:33:54",**  **"updated\_at": "2025-03-09 15:33:54",**  **"card\_type": "debit",**  **"payment\_type": "mastercard",**  **"account\_number": "A8VXqVlFZOFKuujx",**  **"user\_name": "Miloš"**  **}**  **]**  **}** |
| Format HTTP body parametara | JSON |

#### Slično izgleda ruta za fetchovanje naloga.

|  |  |
| --- | --- |
| Opis funkcije | Kreiranje naloga |
| HTTP metoda | POST |
| URL | /api/new-account |
| HTTP body parametri | **{**  **"success": true,**  **"message": "Account created successfully.",**  **"data": {**  **"account\_number": "6fwFUGONJ43Rodv2",**  **"currency": "RSD",**  **"balance": "25000",**  **"balance\_in\_usd": "231.59",**  **"is\_active": true,**  **"user\_id": "9",**  **"updated\_at": "2025-03-09T22:13:07.000000Z",**  **"created\_at": "2025-03-09T22:13:07.000000Z",**  **"id": 24**  **}**  **}** |
| Format HTTP body parametara | JSON |

|  |  |
| --- | --- |
| Opis funkcije | Kreiranje kartice za izabrani nalog |
| HTTP metoda | POST |
| URL | /api/accounts/{accountId}/cards |
| HTTP body parametri | **{**  **"message": "Card created successfully",**  **"data": {**  **"account\_id": "22",**  **"card\_type": "debit",**  **"payment\_type": "visa",**  **"card\_number": "h1AdfiNIKZET3vPC",**  **"expiry\_date": "03/28",**  **"cvv": 611,**  **"updated\_at": "2025-03-09T22:14:13.000000Z",**  **"created\_at": "2025-03-09T22:14:13.000000Z",**  **"id": 57**  **}**  **}** |
| Format HTTP body parametara | JSON |

#### Laravel Database Seeder

Seeder-i se koriste za popunjavanje baze podataka testnim podacima. Oni su korisni za razvoj i testiranje aplikacije. Unutar projekta se unutar backend/database/seeders foldera nalazi DatabaseSeeder fajl u koji je ubačena šema za popunjavanje baze podataka. Ova šema kreira 5 korisnika, gde za svakog korisnika kreira 3 korisnička računa. Za svaki račun korisnika se generišu po 3 platne kartice i 15 transakcija. Takođe, kreira se 5 admina i 1 menadžer.

#### React Hooks

Unutar frontend/src/hooks/useLocalStorage.js napravljena je kuka koja dozvoljava korisniku da čuva vrednost stanja određenih podataka unutar lokalnog skladišta stranice. Kod:

import React, {useState, useEffect} from 'react'

function getSavedValue(*key*,*initialValue*){

    const savedValue = JSON.parse(localStorage.getItem(*key*))

    if(savedValue) return savedValue

    if(*initialValue* instanceof *Function*) return initialValue()

    return *initialValue*;

}

export default function useLocalStorage(*key*, *initialValue*) {

    const [value,setValue] = useState(() => {

        return getSavedValue(*key*, *initialValue*);

    })

    useEffect( () => {

        localStorage.setItem(*key*, JSON.stringify(value))

    }, [value])

    return [value,setValue]

}

#### React Komponente (Components)

Komponente su osnovni gradivni blokovi u React-u. Možemo ih implementirati kao funkciju, ili klasu. Komponente ovog projekta su sadržane unutar frontend/src/components direktorijuma. Tu se nalaze komponente računa i platnih kartica, footer dela stranice, breadcrumbs. Kao komponenta izdvaja se NavBar po načinu na koji je implementirana.

NavBar u zavisnosti od uloge koju ulogovani korisnik ima u sistemu (regularni korisnik – klijent, admin ili menadžer) prikazuje linkove ka stranicama

  const userLinks = <>

    <li><*Link* *to*="/logout" *onClick*={handleLogout}>Logout</*Link*></li>

    <li><*Link* *to*="/">Home</*Link*></li>

    <li><*Link* *to*="/cards">Cards</*Link*></li>

    <li><*Link* *to*="/transactions">Transactions</*Link*></li>

    <li><*Link* *to*="/contact">Contact</*Link*></li>

    <li><*Link* *to*="/new-transaction" *className*="new-transaction-link"><*FiPlus* /></*Link*></li>

  </>

  const adminLinks = <>

    <li><*Link* *to*="/logout" *onClick*={handleLogout}>Logout</*Link*></li>

    <li><*Link* *to*="/">Users</*Link*></li>

  </>

  const managerLinks = <>

    <li><*Link* *to*="/logout" *onClick*={handleLogout}>Logout</*Link*></li>

    <li><*Link* *to*="/">Admins</*Link*></li>

    <li><*Link* *to*="/new-admin">New Admin</*Link*></li>

  </>

return (

    <nav *className*="navbar">

      <div *className*="navbar-logo">

        <*Link* *to*="/">Shomica Bank</*Link*>

      </div>

      <ul *className*="navbar-links">

        {/\* login/logout \*/}

        {window.sessionStorage.getItem("auth\_token") == null

        ? <>

        </>

        : *user*?.role === 'user' ? userLinks : *user*?.role === 'admin' ? adminLinks : managerLinks

           }

      </ul>

    </nav>

  );

#### React Rutiranje (BrowserRouter)

React Router Dom biblioteka omogućava rutiranje u React aplikacijama, gde omogućava prikaz različitih React stranica i komponenti na osnovu unešenog URL-a. Konkretno, BrowserRouter je komponenta koja omogućava rutiranje u web pregledaču.

Unutar ovog projekta, logika rutiranja je slična kao u NavBaru, tj. lista ruta kojima prijavljeni korisnik ima pristup zavisi od njegove uloge u sistemu. Spisak konkretnih ruta kojima imaju pristup samo korisnici navedene uloge:

  const userRoutes = <>

    <*Route* *path*="/" *element*={<> <*Home* *user*={user} *accounts*={accounts} /> </>} > </*Route*>

    <*Route* *path*="/profile" *element*={<><*Breadcrumbs* /><*Profile* *user*={user} /></>} />

    <*Route* *path*="/cards" *element*= {<><*Breadcrumbs* />{<*Cards* *cards* ={cards} *setCards*={setCards}/>}</>} />

    <*Route* *path*="/transactions" *element*={<><*Breadcrumbs* />{<*Transactions* *transactions*={transactions}/> }</>} />

    <*Route* *path*="/new-transaction" *element*={<><*Breadcrumbs* />

                      {<*NewTransaction* *accounts* = {accounts } *setAccounts* ={setAccounts}

*transactions*={transactions} *setTransactions* = {setTransactions}/>}

    </>} />

    <*Route* *path*="/contact" *element*={<><*Breadcrumbs* /> <*Contact*/></>}>  </*Route*>

    <*Route* *path*="\*" *element*={<*Navigate* *to*="/" *replace* />} />

  </>

  const adminRoutes = <>

    <*Route* *path*="/" *element*={<*Users* *users*={allUsers} *setUsers*={setAllUsers}/>}/>

    <*Route* *path*="/users/:id" *element*={<><*Breadcrumbs* /><*UserProfile* *accounts*={accounts} *setAccounts*={setAccounts}/> </>} />

    <*Route* *path*="/new-account/:userId" *element*={<><*Breadcrumbs*/><*NewAccount*/></>} />

    <*Route* *path*="/accounts/:accountId/cards" *element*={<><*Breadcrumbs*/><*ViewCards* *cards*={cards} *setCards*={setCards}/></>} />

    <*Route* *path*="/accounts/:accountId/cards/new-card" *element*={<><*Breadcrumbs*/><*NewCard*/></>} />

    <*Route* *path*="\*" *element*={<*Navigate* *to*="/" *replace* />} />

  </>

  const managerRoutes = <>

    <*Route* *path*="/" *element*={<*Admins* *admins*={admins}/>}/>

    <*Route* *path*="/new-admin" *element*={<*NewAdmin*/>}/>

    <*Route* *path*="\*" *element*={<*Navigate* *to*="/" *replace* />} />

  </>

Proces adekvatnog povezivanja ovih ruta sa ostalim je omogućen korišćenjem ProtectedRoute.jsx i AuthenticatedRoute.jsx komponenti:

const ProtectedRoute = () => {

  const isAuthenticated = window.sessionStorage.getItem('auth\_token') !== null;

  // if not authenticated, redirect to login

  if (!isAuthenticated) {

    return <*Navigate* *to*="/login" *replace* />;

  }

  // if authenticated, grants access to child components

  return <*Outlet* />;

};

const ProtectedRoute = () => {

  const isAuthenticated = window.sessionStorage.getItem('auth\_token') !== null;

  if (isAuthenticated) {

    return <*Navigate* *to*="/" *replace* />;

  }

  // if not authenticated, grants access to child components

  return <*Outlet* />;};

Unutar BrowserRouter koda, rute su podeljene na one kojima može pristupiti jedino autentifikovani korisnik, i na one kojima autentifikovani korisnik nema pristup. Unutar AuthenticatedRoute se nalaze login, register i forgotten password stranice, te stoga njima prijavljeni korisnik nema pristup, a unutar ProtectedRoute se nalaze malopre definisane rute, po svojim ulogama.

Na ovaj način postignuto je da prijavljeni korisnici nemaju pristup formama za autentifikaciju, niti stranicama namenjenim za druge uloge, a neprijavljeni korisnici da moraju da se prijave kako bi mogli da pristupaju ostatku stranica ovog projekta.

  return (

    <div *className*="App">

      <*BrowserRouter*>

        <*NavBar* *user*={user} *setUser*={setUser} *setAccounts*={setAccounts} *setCards*={setCards} *setTransactions*={setTransactions}/>

        <*Routes*>

          <*Route* *element*={<*AuthenticatedRoute*/>}>

            <*Route* *path*='/login' *element*= {<*Login* *setUser*={setUser}/>}/>

            <*Route* *path*='/register' *element*= {<*Register*/>}/>

            <*Route* *path*='/forgotten\_password' *element*= {<*ForgottenPassword*/>}/>

          </*Route*>

          <*Route* *element*={<*ProtectedRoute*/>}>

          {/\* enabling routes based on the role of the authenticated user \*/}

          {user?.role === 'user' ? userRoutes : user?.role === 'admin' ? adminRoutes : managerRoutes}

          </*Route*>

          {/\* uz ovo, koju god rutu (cak i pogresnu) korisnik da ukuca, uvek ga vodi ka login page ako nije logged \*/}

          {/\* routes/api.php i ovaj element omogucavaju zastitu od IDOR napada, jer nijedan neautorizovan korisnik ne moze da

          pristupa rutama koje mu nisu dozvoljene \*/}

          <*Route* *path*="\*" *element*={<*Navigate* *to*="/login" *replace* />} />

          </*Routes*>

          <*Footer* />

      </*BrowserRouter*>

    </div>

  );

#### Laravel i React

Za povezivanje backend i frontend dela aplikacije, odnosno, koda napisanog u Laravelu i u Reactu, zaslužni su kontroleri unutar backend/app/Http/Controllers direktorijuma, api.php zbog svoje tabele ruta, i axios, JS biblioteka koja služi pravljenju HTTP zahteva u web aplikacijama. Axios pojednostavlja proces slanja asinhronih zahteva i manipulacije odgovorom API-ja.

***/routes/api.php:***

*Route*::get('/user', function (*Request* $request) {

    return $request->user();

})->middleware('auth:sanctum');

*Route*::get('/exchange-rates', [*CurrencyController*::class, 'getExchangeRates']);

// GUEST middleware

*Route*::middleware('guest')->group(function () {

*Route*::post('/auth/register', [*AuthController*::class, 'createUser']);

*Route*::post('/auth/login', [*AuthController*::class, 'login']);

*Route*::post('/auth/forgot-password', [*AuthController*::class, 'forgotPassword']);

});

// USER middleware

*Route*::group(['middleware' => ['auth:sanctum']], function () {

*Route*::get('/profile', function (*Request* $request) {

        $cacheKey = 'user\_profile\_' . auth()->id();

*Log*::info('Cache key: ' . $cacheKey);

        if (*Cache*::has($cacheKey)) {

*Log*::info('Fetching profile data from cache for user ' . auth()->id());

        } else {

*Log*::info('Fetching profile data from database for user ' . auth()->id());

        }

        /\*cuvanje je u sekundama, zato 60 min \* 60 sek/min\*/

        $data = *Cache*::remember($cacheKey, 60 \* 60, function () use ($request) {

            $user = auth()->user()->load('accounts');

            return [

                'message' => "Welcome to user profile",

                'user-data' => $user,

            ];

        });

        return response()->json($data);

    });

*Route*::post('/delete-card', [*CardController*::class, 'destroy']);

*Route*::post('/new-transaction', [*TransactionController*::class, 'store']);

*Route*::post('/transactions', [*TransactionController*::class, 'index']);

*Route*::get('/profile/cards', [*AccountCardController*::class, 'index'])->name('profile.cards');

*Route*::get('/profile/transactions', [*AccountTransactionController*::class, 'index'])->name('profile.transactions');

*Route*::post('auth/logout', [*AuthController*::class, 'logoutUser']);

});

// ADMIN middleware

*Route*::group(['middleware' => ['auth:admin-api']], function () {

*Route*::resource('users', *UserController*::class)->only(['index', 'show', 'destroy', 'update']);

*Route*::resource('accounts', *AccountController*::class)->only(['index', 'show']);

*Route*::post('/new-account', [*AccountController*::class, 'store']);

*Route*::get('/accounts/{accountId}/cards', [*AccountCardController*::class, 'index1']);

*Route*::post('/accounts/{accountId}/cards', [*CardController*::class, 'store']);

*Route*::post('admin/logout', [*AuthController*::class, 'logoutAdmin']);

});

// MANAGER middleware

*Route*::group(['middleware' => ['auth:manager-api']], function () {

*Route*::get('/admins', [*AdminController*::class, 'index']);

*Route*::post('/new-admin', [*AuthController*::class, 'createAdmin']);

*Route*::delete('/admins/{id}', [*AdminController*::class, 'destroy']);

*Route*::post('manager/logout', [*AuthController*::class, 'logoutManager']);

});

Primer zahteva za skladištenjem transakcije unutar baze podataka:

  const handleSubmit = async (*e*) => {

*e*.preventDefault();

    if (!recipient || !amount || !recipientAccount || !selectedAccount) {

      setError('Please fill out all fields.');

      return;

    }

    if (isNaN(amount) || amount <= 0) {

      setError('Please enter a valid amount.');

      return;

    }

    try {

      const authToken = window.sessionStorage.getItem("auth\_token");

      // convert the amount to USD

      const amountInUSD = (amount / exchangeRates[selectedCurrency]).toFixed(2);

      // convert the amount to the sender's currency (domain currency)

      const domainAmount = (amountInUSD \* exchangeRates[selectedAccount.currency]).toFixed(2);

      const response = await axios.post('/api/new-transaction', {

        account\_id: selectedAccount.id,

        recipient\_account: recipientAccount,

        recipient\_name: recipient,

        amount\_in\_domain: domainAmount,

        amount: amount,

        currency: selectedCurrency,

        currency\_domain: selectedAccount.currency,

      }, {

        headers: {

          Authorization: `Bearer ${authToken}`,

        },

      });

      if (response.data.success) {

        const [accountsResponse, transactionsResponse] = await *Promise*.all([

          axios.get('/api/profile', {

            headers: {

              Authorization: `Bearer ${authToken}`,

            },

          }),

          axios.get('/api/profile/transactions', {

            headers: {

              Authorization: `Bearer ${authToken}`,

            },

          }),

        ]);

        setAccounts(accountsResponse.data['user-data'].accounts);

        setTransactions(transactionsResponse.data.data);

        navigate('/transactions');

      } else {

        setError(response.data.message || 'Transaction failed.');

      }

    } catch (error) {

      if (error.response && error.response.status === 400) {

        setError('Insufficient funds.');

      } else {

        console.error(error);

        setError('An error occurred while processing the transaction.');

      }

    }

  };

### Projektovanje strukture softverskog sistema

#### Migracije

Migracije u Laravelu predstavljaju sistem za upravljanje verzijama baze podataka. Omogućavaju (re)definisanje strukture baze podataka na početku, i tokom razvoja same aplikacije, što značajno olakšava proces ažuriranja baze podataka i omogućava verzionisanje baze podataka i praćenje promena. Dodatna mogućnost je rollback određene migracije, gde možemo vratiti bazu podatak u neko od prethodnih stanja (verzija).

U ovom projektu se nalazi puno migracija baze podataka, različitih tipova. Postoje migracije koje kreiraju tabele, koje dodaju ili menjaju polja u tabeli, ili ih skroz brišu, takođe i one koje dodaju spoljni ključ ka nekoj drugoj tabeli. Biće prikazane migracije vezane za tabelu korisnika:  
  
***create\_users\_table:***

return new class extends *Migration*

{

    /\*\*

     \* Run the migrations.

     \*/

    public function up(): void

    {

*Schema*::create('users', function (*Blueprint* $table) {

            $table->id();

            $table->string('name');

            $table->string('email')->unique();

            $table->timestamp('email\_verified\_at')->nullable();

            $table->string('password');

            $table->rememberToken();

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

*Schema*::dropIfExists('users');

    }

};

***add\_columns\_to\_users\_table:***

return new class extends *Migration*

{

    /\*\*

     \* Run the migrations.

     \*/

    public function up(): void

    {

*Schema*::table('users', function (*Blueprint* $table) {

            $table->date('date\_of\_birth')->nullable();

            $table->enum('gender', ['male', 'female', 'other'])->nullable();

            $table->enum('work\_status', ['unemployed', 'student', 'employed', 'retired'])->nullable();

            $table->string('street')->nullable();

            $table->string('city')->nullable();

            $table->string('country')->nullable();

            $table->string('postal\_code', 20)->nullable();

            $table->string('phone\_number', 20)->nullable();

            $table->string('role')->default('user');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

*Schema*::table('users', function (*Blueprint* $table) {

            $table->dropColumn('date\_of\_birth');

            $table->dropColumn('gender');

            $table->dropColumn('work\_status');

            $table->dropColumn('street');

            $table->dropColumn('city');

            $table->dropColumn('country');

            $table->dropColumn('postal\_code');

            $table->dropColumn('phone\_number');

            $table->dropColumn('role');

        });

    }

};

#### Modeli

Modeli u Laravel-u predstavljaju objekte koji omogućavaju interakciju sa tabelama u bazi podataka. Svaki model je povezan sa jednom tabelom u bazi podataka i omogućava čitanje, upis, ažuriranje i brisanje podataka (CRUD operacije). Predstavljaju jedan od delova Laravelovog Eloquent ORM (Object-Relational-Mapping) sistema, koji omogućava jednostavan i elegantan način interagovanja sa bazom podataka, kao i pravljenje relacija između tabela.

Glavni modeli u bazi su svakako vezani za korisnika, njegove račune, kartice i transakcije:

class User extends *Authenticatable*

{

    use *HasApiTokens*, *HasFactory*, *Notifiable*;

    protected $fillable = [

        'name',

        'email',

        'password',

        'date\_of\_birth',

        'gender',

        'work\_status',

        'street',

        'city',

        'country',

        'postal\_code',

        'phone\_number',

        'role',

    ];

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    protected function casts(): array

    {

        return [

            'email\_verified\_at' => 'datetime',

            'password' => 'hashed',

        ];

    }

    public function accounts()

    {

        return *$this*->hasMany(*Account*::class);

    }

}

Ovaj model pokazuje polja kojima je dozvoljeno dodeljivati vrednost ($fillable) prilikom masovne dodele vrednosti, i one kojima to nije dozvoljeno ($hidden). Možemo primetiti i odnos korisnika prema računu, gde jedan korisnik ima više naloga, isto kao u konceptualnom modelu.

class Account extends *Model*

{

    /\*\* @use HasFactory<\Database\Factories\AccountFactory> \*/

    use *HasFactory*;

    protected $fillable = [

        'account\_number',

        'user\_id',

        'currency',

        'balance',

        'balance\_in\_usd',

        'is\_active',

    ];

    public function user()

    {

        return *$this*->belongsTo(*User*::class);

    }

    public function cards()

    {

        return *$this*->hasMany(*Card*::class);

    }

    public function transactions()

    {

        return *$this*->hasMany(*Transaction*::class);

    }

}

Što se Modela računa tiče, svaki račun pripada tačno jednom korisniku, i za njega je vezano više platnih kartica i transakcija.

class Card extends *Model*

{

    /\*\* @use HasFactory<\Database\Factories\CardFactory> \*/

    use *HasFactory*;

    protected $fillable = [

        'card\_number',

        'expiry\_date',

        'cvv',

        'account\_id',

        'card\_type',

        'payment\_type',

    ];

    public function account()

    {

        return *$this*->belongsTo(*Account*::class);

    }

}

class Transaction extends *Model*

{

    use *HasFactory*;

    protected $fillable = [

        'account\_id',

        'recipient\_name',

        'recipient\_account',

        'amount',

        'amount\_in\_domain',

        'currency',

        'currency\_domain',

        'status',

        'transaction\_number',

    ];

    protected $dates = ['created\_at'];

    public function getCreatedAtAttribute($value)

    {

        return *\Carbon\Carbon*::parse($value)->format('Y-m-d'); // created\_at in new format: yyyy-mm-dd

    }

    public function account()

    {

        return *$this*->belongsTo(*Account*::class);

    }

}

Svaki objekat transakcije ili kartice je vezan za tačno jedan račun korisnika.

### Projektovanje skladišta podataka

Relacioni model softverskog sistema:

User( ***#id***, name, email, password, date\_of\_birth, gender, work\_status, street, city, country, postal\_code, phone\_number, role)

Account( ***#id***, account\_number, currency, balance, balance\_in\_usd, is\_active, ***user\_id)***

Transaction(***#id***, amount\_in\_domain, amount, status, recipient\_name, recipient\_account, transaction\_number, currency\_domain, currency, ***account\_id***)

Card(***#id***, card\_number, expiry\_date, cvv, card\_type, payment\_type, ***account\_id*** )

Admin(***#id***, email, password, role)

Manager (***#id***, email, password, role)

UserAdmin ( ***#idUser, #idAdmin*** )

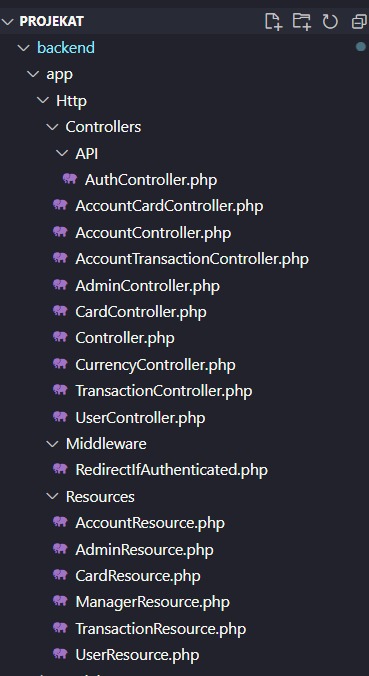
AdminManager ( ***#idAdmin, #idManager***)

## Faza implementacije

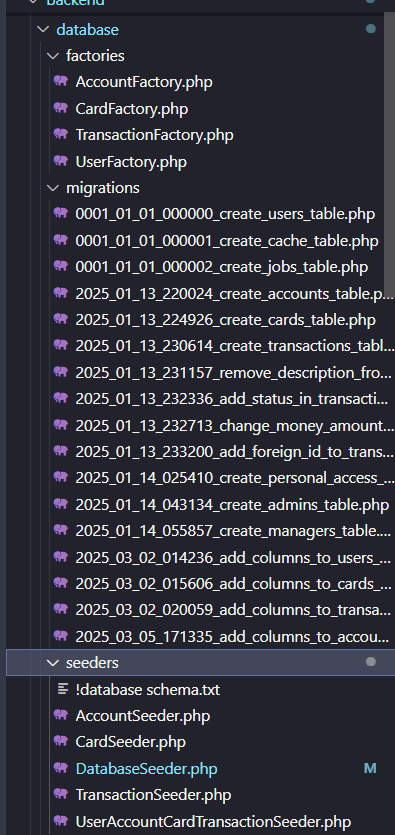
U ovoj fazi se vrši programiranje sistema na osnovu prethodno definisanog dizajna. Cilj je da se realizuju sve funkcionalnosti i da se integrišu sve komponente. Pored toga, bitno je lepo struktuirati direktorijume i fajlove unutar projekta zarad lakšeg snalaženja u kodu i programu.

**Struktura backend dela koda:**

Unutar app foldera nalaze kontroleri, resursi i middleware sadržaj:

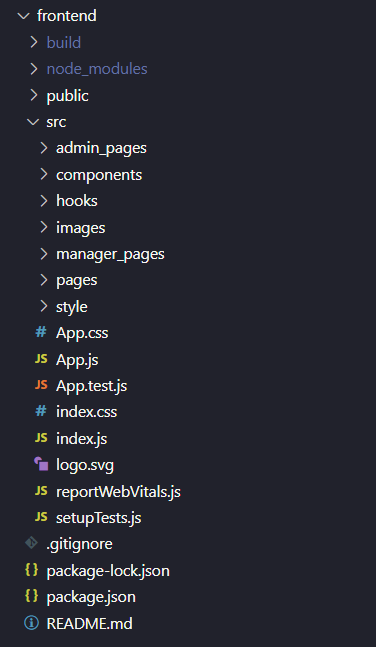


Unutar database direktorijuma, nalaze se migracije za rekonfigurisanje baze podataka, kao i različiti seeder-i i factories koji omogućavaju popunjavanje baze zarad praktičnog testiranja programa tokom njegovog razvoja:



Unutar routes direktorijuma, nalazi se api.php, izuzetno važan fajl vezan za rutiranje i povezivanje backenda sa frontendom.

**Struktura frontend dela koda:**



Unutar components direktorijuma nalaze se reusable komponente koje služe svim ostalim stranicama, nevezano da li one služe u izgradnji stranica regularnog korisnika (folder pages), admina (admin\_pages) ili menadzera (manager\_pages). Tu se nalazi i folder slika koje se koriste unutar projekta, kao i custom hooks. Direktorijum style je vezan za CSS stilizovanje stranica i komponenata.

## Faza testiranja

U ovoj zavržnoj fazi projektovanja softvera vrši se verifikacija i validacija sistema, zarad sigurnosti da sistem ispunjava sve korisničke zahteve, da obezbeđuje sigurnost i radi nesmetano.

Tokom ove faze pronađeni su neki problemi koji su uglavnom bili vezani za frontend izgled i stilove. Međutim pronađen je veći bug unutar logike kontrolera za transakcije. Naime, u pokušaju da sa jednog računa korisnika prebacimo materijalna sredstva na isti taj račun, ne samo da je transakcija prošla, nego se, pored toga, bilans tog računa povećao za iznos te transakcije. Na taj način je unutar store funkcije TransactionControllera ubačen deo koda koji prekida transakciju u slučaju da je objekat korisnika koji pokreće transakciju isti kao objekat korisnika primaoca uplate.

if ($recipientAccount == $fromAccount) {

//ubaceni deo koda

                return response()->json([

                    'success' => false,

                    'message' => 'You cannot send money to yourself!',

                    'error' => 'You cannot send money to yourself!',

                ], 500);

            }

# ZAKLJUČAK

Završetkom izrade ovog seminarskog rada, uspešno je razvijena veb aplikacija za e-bankarstvo, korišćenjem PHP i JavaScript programskih jezika, i njihovih najpopularnijih framework-a i biblioteka, respektivno. Implementiranje aplikacije zahtevalo je pažljivo praćenje zahteva definisanih u početnoj fazi razvoja projekta. Kao što je i tada, u uvodu objašnjeno, akcenat je bio, i ostao, na implementiranju modernih sigurnosnih mehanizama, na ispunjavanju eksplicitnih želja klijenta. Softver je oprezno i detaljno kucan kako bi bio upotrebljiv, pouzdan, podrživ, i dobrih performansi.

Neki od značajnijih izazova bili su svakako integracija frontend-a i backend-a, a uz to i struktuiranje ruta i razvijanje logike kontrolera. Naročito izazovna je bila implementacija funkcionalnosti dodavanja nove transakcije, uzevši u obzir mogućnost plaćanja u raznim valutama, čije se vrednosti fetchuju sa API-ja.

Uprkos ovim izazovima, uživao sam radeći u ovim tehnologijama. Produbio sam svoje znanje iz ove oblasti e-poslovanja, kao i metodologiji razvoja modernog web softvera.

# LITERATURA

1. JavaScript – Jensen, S. H., Møller, A., & Thiemann, P. (2009). Type analysis for JavaScript. In Lecture notes in computer science (pp. 238–255). <https://doi.org/10.1007/978-3-642-03237-0_17>
2. React.js – Sanchit Aggarwal et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 5, Issue 1, March 2018, pp. 133-137 (<https://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf>)
3. PHP – Programming PHP. (n.d.). Google Books. <https://books.google.rs/books?hl=en&lr=&id=h-E1lVko-skC&oi=fnd&pg=PR5&dq=php&ots=ysIWhSVyVh&sig=_3m47liPFYsYPWDJDUEP-NTaGos&redir_esc=y#v=onepage&q=php&f=false>
4. Laravel – Laravel 5 Essentials. (n.d.). Google Books. <https://books.google.rs/books?hl=en&lr=&id=BWO4CAAAQBAJ&oi=fnd&pg=PP1&dq=laravel&ots=8cY2WvvEXM&sig=JAUh9pw43hWD7HJ0ixK9MXNRpB8&redir_esc=y#v=onepage&q=laravel&f=false>
5. Larmanova metoda razvoja softvera – Dr Siniša Vlajić, Beograd 2020. <https://www.researchgate.net/profile/Sinisa-Vlajic/publication/344459754_PROJEKTOVANE_SOFTVERA_SKRIPTA_-_RADNI_MATERIJAL_-_2020/links/5f7efd67299bf1b53e16133e/PROJEKTOVANE-SOFTVERA-SKRIPTA-RADNI-MATERIJAL-2020.pdf>