Univerzitet Crne Gore
ELEKTROTEHNIČKI FAKULTET
Studije primijenjenog računarstva

Baze podataka (napredni kurs)

-dokumentacija projekta-

Tema projekta: Sajt avio agencije

Članovi projekta:

- Stefan Đurišić 7/22
- Marija Perović 10/22
- Đorđe Đuricić 15/22
- Redžep Đerekarac 49/22

Opis projekta:

Cilj našeg zadatka je bla izrada sajta avio-agencije, koji bi omogućio jednostavnu i efektivnu komunikaciju sa backend API-jem. Samim tim funkcije kao upravljanje letova, korisnika, rezervacija, plaćanja, filtriranja se mogu koristiti bez problema, uz jednostavnost i “user friendly” interfejs.

Korisnici mogu obavljati sledeće funkcije: pravljenje naloga, mijenjanje šifre naloga, pretraživanja letova, razne opcije filtriranja letova (po destinacijama, cijeni, klasi), obaviti rezervacije, imati uvid na date rezervacije, kao i njihovo otkazivanje.

Sa Administrativnog pogleda, imamo Admin panel sa različitim funkcionalnostima. Kao što su detaljno unošenje i brisanje letova, uvid u korisnike, uvid u transakcije i plaćanja na samom sajtu. Uz naravno poštovanje pravila, i zabrane pristupa kritičnim podacima korisnika (npr. lozinke i brojevi kreditnih kartica).

Za realizaciju samoga projekta, koristili smo različite alate i programske jezike, zavisno od potrebe i odgovarajuće funkcije. Za backend i kreiranje samog API-a sa pozivima koristili smo Flask kao Python framework. On je služio za definisanje samih ruta i logike poziva. MySQL kao način prikupljanja i čuvanja baze podataka. Enkripciju i hashovanje, za čuvanje kritičnih podataka korisnika i njihovu bezbjednost. httpie smo koristili za testiranje http poziva i komunikaciju sa Flask API-jem. Takođe za uočavanje grešaka pri ranim fazama razvoja, kao i preglednost takozvanih “error codova”. Jinju za formatiranje i kreiranje template-a za rad lakšeg i jednostavnijeg prikaza, pri komunikaciji backenda sa frontendom. Za front end smo odlučili da se držimo jednostavnosti (pošto na kraju krajeva, potencijal

ovog projekta je backend), pa smo koristili HTML sa CSS-om, i JavaScriptom, koji smo koristili kao posrednik logike formi, dugmati i ostalog korisničkog interfejsa.

Imajući jednostavan i intuitivan pristup, sajt omogućava kako korisnicima, tako i administratoru, da na jednostavan način komuniciraju sa bazom podataka. I da kroz intuitivan pristup vrše kompleksne API pozive pomoću pritiska dugmeta na ekranu. Glavni fokus je bio user interface i njegovo uproštavanje krajnjem korisniku, pri očuvanju robusnosti i funkcionalnosti samoga softvera.
Takođe radi unikatnosti, i samog dizajna sajta, odlučili smo da sami kreiramo panele funkcionalnosti za korisnika i administratore.

Opis zadataka članova projektnog tima

U realizaciji ovog projekta učestvovala su četiri člana tima, koji su na osnovu svoje mogućnosti i znanja doprinijeli razvoju aplikacije, izradi baze podataka, funkcionalnosti sistema i pripremi tehničke dokumentacije. Svaki član je bio odgovoran za određeni dio projekta, ali su svi učestvovali u planiranju i evaluaciji cjelokupnog rada.

Đorđe Đuričić je bio zadužen pri pisanju dokumentacije, crtanju odredjenih dijagrama, sugestija za finalni dizajn i izgled sajta. On je dao ideju za rad projekta.

Stefan Đurišić je radio na pisanju i dizajnu frontend-a, sugestije i pomoc oko dokumentacije, realizacije i testiranja projekta. Takodje je pomogao pri povezivanju backend API poziva sa front endom, manja pomoc oko Java Scripta. Bio je od velike pomoci Redzepu pri kritičnim fazama testiranja frontenda i backenda.

Redžep Đerekarac se fokusirao na obradu kompletnog backend API-a. Pisao je rute, testirao ih tokom produkcije, radio zajdeno sa Stefanom na “spajanju” front enda sa backendom, doradjivao dijagrame i pisao dokumentaciju. Implementirao kompletnu logiku iza backenda. Testirao MySQL i kreirao punu bazu podataka. Implementirao enkripciju, osmislio modele i relacije, radio na frontendu, osmislio i dizajnirao admin panele, testirao je program tokom citave produkcije. Sam je odradio UseCase dijagrame i sekvence. Takodje realizovao logiku JavaScripta na front endu.

Marija Perović je radila na dizajnu korisničkog interfejsa, koristeći HTML/CSS kako bi aplikacija bila vizuelno konzistentna i responzivna. Takodje je pomogla pri crtanju dijagrama, i pisanja dokumentacije.

Svi članovi tima su sveobuhvatno zajednički učestvovali u testiranju aplikacije, pisanju dokumentacije, kao i u usaglašavanju sadržaja kako bi finalni rezultat bio funkcionalan, pregledan i dobro dokumentovan.

Sistemski zahtjevi

Funkcionalni zahtjevi

Identifikator	Prioritet	Zahtjev
REQ1	5	Sistem omogućava korisniku da se registruje i prijavi.
REQ2	5	Sistem omogućava korisniku pregled dostupnih letova.
REQ3	4	Sistem omogućava korisniku da filtrira letove po cijeni, datumu i klasi.
REQ4	5	Sistem omogućava korisniku da rezerviše let.
REQ5	4	Sistem omogućava korisniku da vidi svoje rezervacije.
REQ6	5	Sistem omogućava korisniku da izvrši plaćanje za rezervaciju.
REQ7	3	Sistem omogućava korisniku da otkaže rezervaciju.
REQ8	4	Sistem omogućava prikaz broja dostupnih sjedista po letu.
REQ9	5	Sistem omogućava administratoru da dodaje/briše letove.
REQ10	4	Sistem omogućava administratoru da vidi sve korisnike, rezervacije i uplate.

Nefunkcionalni zahtjevi

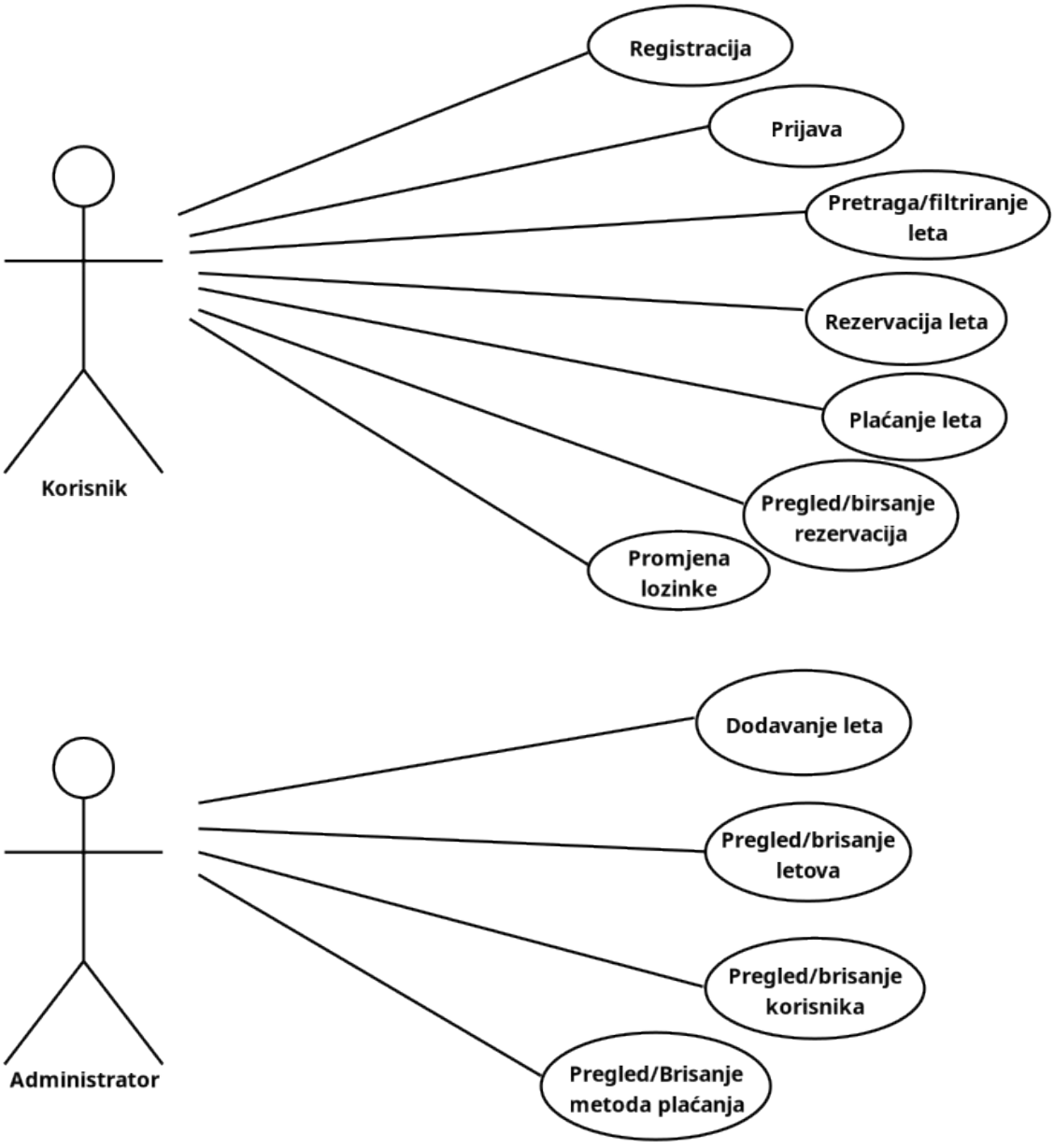
Identifikator	Prioritet	Zahtjev
REQ11	5	Sistem mora biti responzivan i kompatibilan sa modernim pretraživačima.
REQ12	5	Sistem mora obezbijediti sigurnu obradu korisničkih podataka (lozinke, kartice).
REQ13	5	Sistem treba da bude jednostavan za korišćenje i navigaciju.
REQ14	4	Sistem treba da podrži promjenu lozinke
REQ15	3	Sistem treba imati vrijeme odziva manje od 2 sekunde po upitu.
REQ16	4	Sistem mora omogućiti backup i obnovu baze podataka.

Opis odgovornosti

U nastavku je data tabela sa funkcionalnim cjelinama sistema i pripadajućim opisima njihovih odgovornosti.

Funkcionalna cjelina	Odgovornost
Registracija	Omogućava korisnicima da kreiraju nalog, uz validaciju unijetih podataka i provjeru duplikata.
Prijava	Provjerava vjerodostojnost korisničkih kredencijala i otvara pristup korisničkom nalogu.
Pretraga letova	Omogućava korisniku da pronađe letove na osnovu datuma, destinacije i klase. Ili da sam pretraži let.
Rezervacija	Vezuje korisnika za određeni let, bilježi broj sjedišta i ukupnu cijenu rezervacije.
Plaćanje	Prikuplja podatke o kartici, vrši njihovu enkripciju i evidentira plaćanje u bazu.
Pregled rezervacija	Omogućava korisniku pregled prethodnih i aktivnih rezervacija.
Administracija letova	Omogućava administratorima dodavanje, izmjenu i brisanje letova.
Pregled korisnika i rezervacija	Administratorima pruža uvid u sve korisničke naloge i njihove rezervacije.

Početni Use Case dijagram:



Use Case šema 1 – Registracija korisnika

Atribut	Opis
Use Case:	UC-1
Naziv:	Registracija korisnika
Vezani zahtjevi:	REQ1
Akter koji inicira:	Korisnik
Cilj aktera:	Kreirati novi nalog i dobiti pristup sistemu
Akteri koji učestvuju:	Sistem (Web aplikacija avio kompanije)
Preduslovi:	Korisnik nije registrovan i pristupa stranici za registraciju
Post-uslovi:	Novi korisnik je uspješno unesen u bazu podataka
Tok događaja – glavni scenario:	1. Korisnik otvara stranicu za registraciju. 2. Korisnik unosi sve potrebne podatke (ime, prezime, email, lozinka). 3. Sistem provjerava validnost i jedinstvenost email-a. 4. Ako su podaci validni, korisnik se upisuje u bazu i dobija poruku o uspjehu.
Tok događaja – alternativni scenariji:	1.a Ako korisnik ne unese sve obavezne podatke, ili ih pogrešno unese. 2.a Sistem prikazuje poruku o grešci (npr. "Email već postoji", "Lozinka nije dovoljno jaka").

Use Case šema 2 – Prijava korisnika

Atribut	Opis
Use Case:	UC-2
Naziv:	Prijava korisnika
Vezani zahtjevi:	REQ1
Akter koji inicira:	Korisnik
Cilj aktera:	Prijaviti se u sistem, pomocu vec prethodno registrovanog naloga.
Akteri koji učestvuju:	Sistem (Web aplikacija avio kompanije)
Preduslovi:	Korisnik je registrovan u bazi podataka.
Post-uslovi:	Korisnik je uspjesno prijavljen na sajtu. Sajt zadrzava korisnicku sesiju dok se on ne odjavi.
Tok događaja – glavni scenario:	1. Korisnik otvara stranicu za prijavu. 2. Korisnik unosi sve potrebne podatke (email, lozinka). 3. Sistem provjerava validnost i jedinstvenost podataka. Ako su podaci validni, korisnik dobija poruku o uspjehu, i biva prebacen na pocetnu stranu sajta.
Tok događaja – alternativni scenariji:	1.a Ako korisnik ne unese sve obavezne podatke. 2.a Sistem prikazuje poruku o grešci (npr. "Netacan email ili lozinka). 3.a Korisnik je zaboravio lozinku, i pristupa stranici za promjenu lozinke.

Use Case šema 3 – Pretraga/filtriranje leta

Atribut	Opis
Use Case:	UC-3
Naziv:	Pretraga/filtriranje leta
Vezani zahtjevi:	REQ2, REQ3
Akter koji inicira:	Korisnik
Cilj aktera:	Pronaći odgovarajući let po željenim kriterijumima
Akteri koji učestvuju:	Sistem
Preduslovi:	Nema
Post-uslovi:	Prikazuju se letovi koji odgovaraju filterima
Tok događaja – glavni scenario:	1. Korisnik otvara stranicu za pretragu letova. 2. Unosi filtere (destinacija, datum, cijena) ili pretrazuje let u baru za pretragu . 3. Sistem prikazuje rezultate koji odgovaraju unesenim parametrima.
Tok događaja – alternativni scenariji:	1.a Ako nema rezultata pretrage. 2.a Sistem prikazuje poruku: "Nema letova koji odgovaraju pretrazi".

Use Case šema 4 – Rezervacija i plaćanje leta

Atribut	Opis
Use Case:	UC-4
Naziv:	Rezervacija leta
Vezani zahtjevi:	REQ4, REQ6
Akter koji inicira:	Korisnik
Cilj aktera:	Rezervisati željeni let
Akteri koji učestvuju:	Sistem
Preduslovi:	Korisnik je prijavljen i izabrao let iz liste
Post-uslovi:	Let je rezervisan i podaci su sačuvani u bazi
Tok događaja – glavni scenario:	1. Korisnik izabere let. 2. Unosi broj sjedišta i potvrđuje rezervaciju. 3. Sistem provjerava dostupnost. 4. Korisnik unosi podatke o plaćanju. 5. Sistem vrši rezervaciju u bazi.
Tok događaja – alternativni scenariji:	1.a Ako nema dovoljno dostupnih sjedišta. 2.a Sistem prikazuje poruku o grešci i onemogućava rezervaciju. 1.b Ako korisnik ne unese validne podatke za plaćanje. 2.b Sistem izbacuje poruku o grešci.

Use Case šema 5 – Pregled/brisanje rezervacija

Atribut	Opis
Use Case:	UC-5
Naziv:	Pregled/brisanje rezervacija
Vezani zahtjevi:	REQ10
Akter koji inicira:	Korisnik
Cilj aktera:	Izvrsiti pregled i brisanje rezervacije
Akteri koji učestvuju:	Sistem
Preduslovi:	Korisnik je prijavljen, i napravio je najmanje jednu rezervaciju.
Post-uslovi:	Pregled rezervacija uspjesan, korisnik je obrisao rezervaciju.
Tok događaja – glavni scenario:	1.Korisnik odlazi na stranicu sa pregledom rezervacija. 2. Sistem prikazuje korisnicke rezervacije. 3. Korisnik priska dugme za otkazivanje pored zeljene rezervacije, i dobija statusnu poruku.
Tok događaja – alternativni scenariji:	1.a Korisnik nema ni jednu rezervaciju.

Use Case šema 6 – Promjena lozinke

Atribut	Opis
Use Case:	UC-6
Naziv:	Promjena lozinke
Vezani zahtjevi:	REQ14
Akter koji inicira:	Korisnik
Cilj aktera:	Promijeniti lozinku
Akteri koji učestvuju:	Sistem
Preduslovi:	Korisnik je registrovan, ali je zaboravio lozinku.
Post-uslovi:	Korisnik uspješno mijenja lozinku, i od tada se može ulogovati sa novom.
Tok događaja – glavni scenario:	1. Korisnik odlazi na sekciju za promjenu lozinke. 2. Korisnik validira svoj identitet, potvrđujući svoje ime i email. 3. Korisnik dobija polje za unos nove lozinke. 4. Korisnik dobija poruku o uspješnoj promjeni lozinke
Tok događaja – alternativni scenariji:	1.a Korisnik ne može da validira svoj identitet (ne unosi tačno ime ili email).

Use Case šema 7 – Administracija letova

Atribut	Opis
Use Case:	UC-7
Naziv:	Upravljanje letovima (admin)
Vezani zahtjevi:	REQ9
Akter koji inicira:	Administrator
Cilj aktera:	Dodavanje, pregled, brisanje letova.
Akteri koji učestvuju:	Sistem
Preduslovi:	Korisnik mora imati administratorske privilegije.
Post-uslovi:	Letovi ažurirani u bazi podataka.
Tok događaja – glavni scenario:	1. Administrator bira “Admin letove” na panelu. 2. Administrator bira opciju .dodavanja/brisanja. 3. Unosi podatke i potvrđuje akciju. 4. Sistem vrši željenu izmjenu u bazi.
Tok događaja – alternativni scenariji:	1.a Administrator unosi nevalidne podatke. 2.a Sistem prikazuje poruku o grešci.

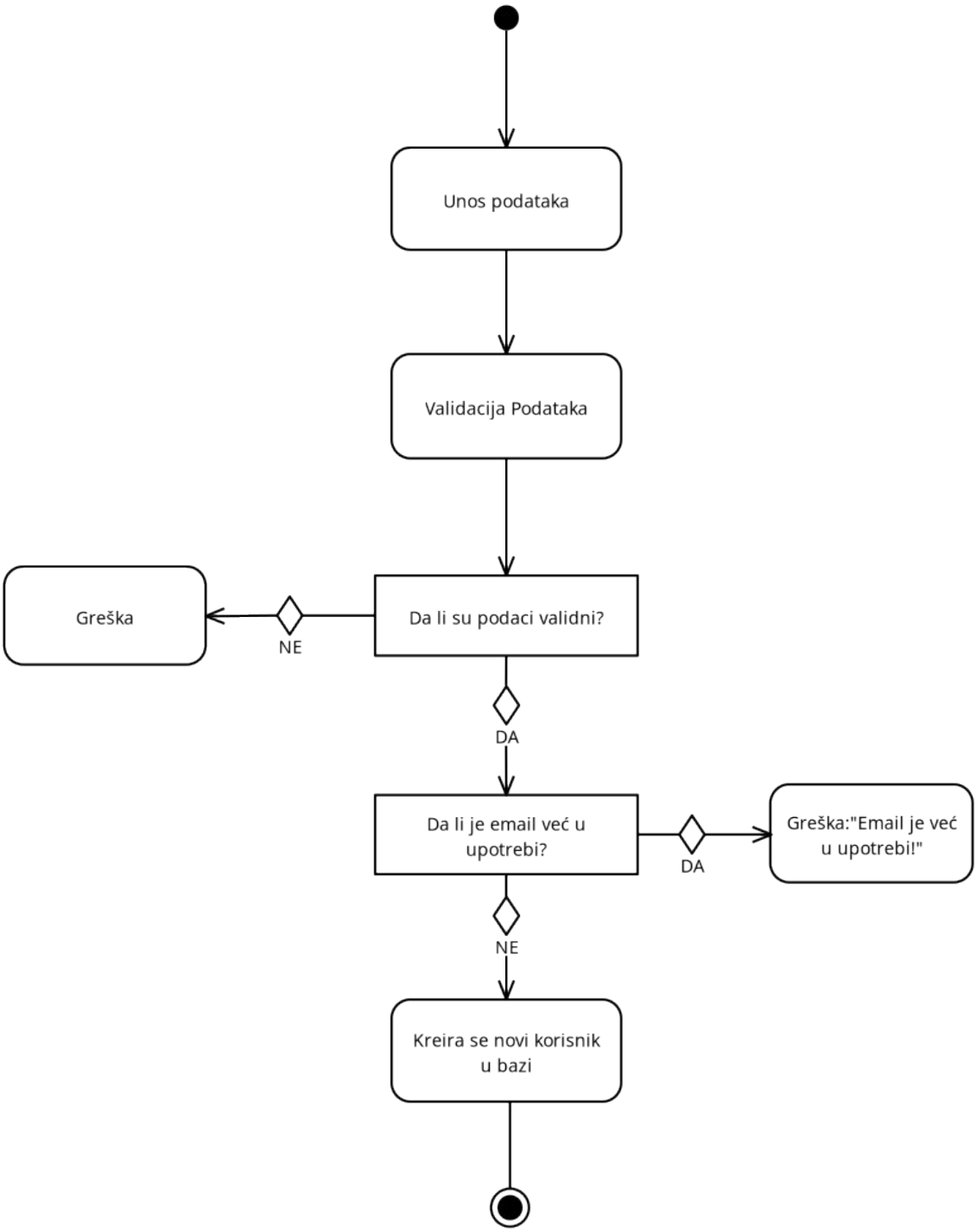
Use Case šema - 8 Administacija korisnika

Atribut	Opis
Use Case:	UC-8
Naziv:	Administracija korisnika
Vezani zahtjevi:	REQ10
Akter koji inicira:	Administrator
Cilj aktera:	Pregled/brisanje nepozeljnih korisnika iz baze podataka.
Akteri koji učestvuju:	Sistem
Preduslovi:	Korisnik mora imati administratorske privilegije.
Post-uslovi:	Korisnici su ažurirani u bazi podataka.
Tok događaja – glavni scenario:	1. Administrator bira “Pregled korisnika” na panelu. 2. Administrator brise nepozeljenog korisnika i potvrdjuje akciju. 3. Sistem vrši željenu izmjenu u bazi.
Tok događaja – alternativni scenariji:	1.a Nema registrovanih korisnika u bazi sem admina. 2.a Sistem prikazuje obavjestenje. 1.b Administrator pokusava da obirse sebe iz baze. 2.b Sistem javlja gresku.

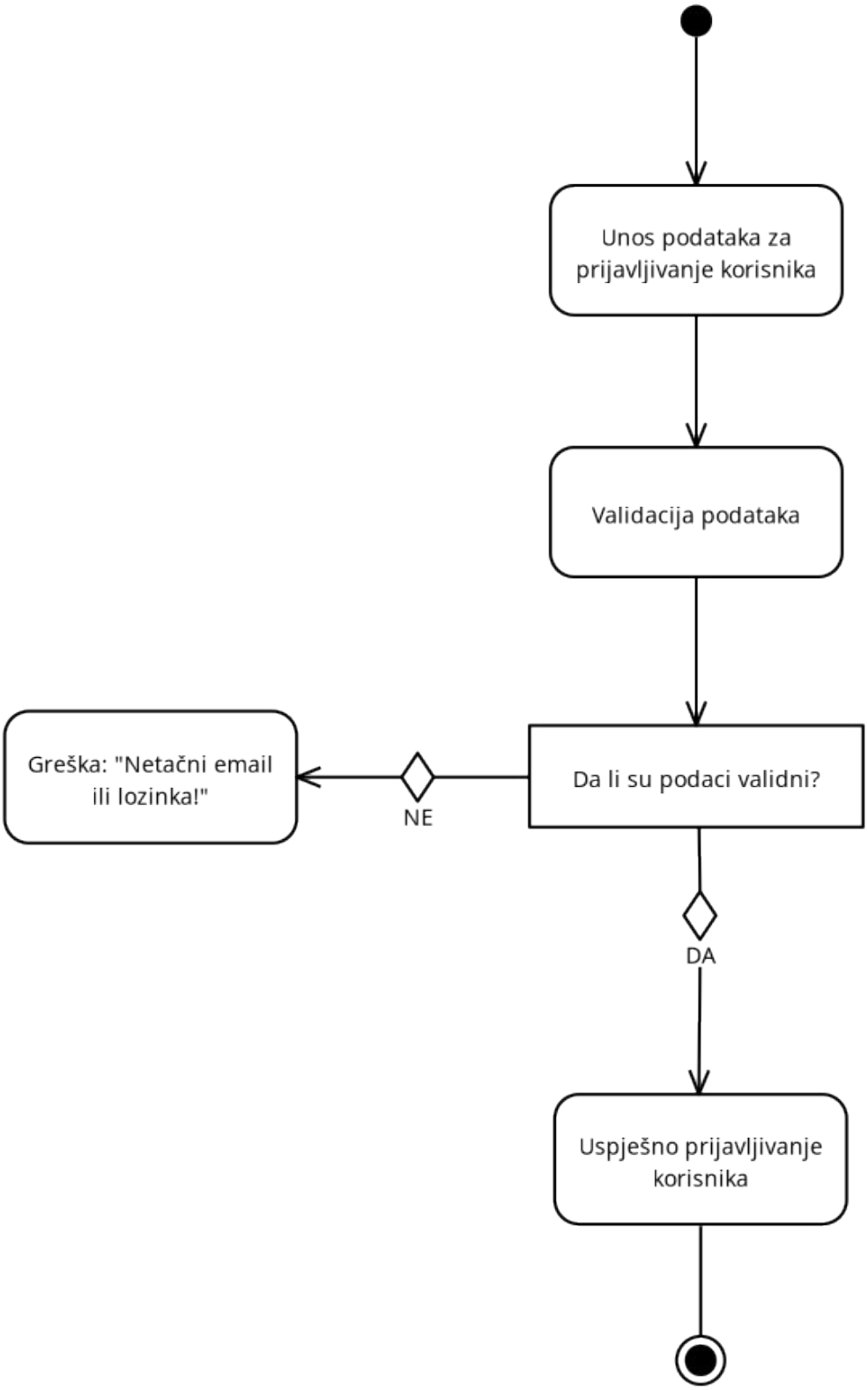
Use Case šema 9 – Administracija uplata

Atribut	Opis
Use Case:	UC-9
Naziv:	Administracija uplata korisnika
Vezani zahtjevi:	REQ10
Akter koji inicira:	Administrator
Cilj aktera:	Pregled/brisanje nepozeljnih uplata iz baze podataka.
Akteri koji učestvuju:	Sistem
Preduslovi:	Korisnik mora imati administratorske privilegije.
Post-uslovi:	Uplate su ažurirane u bazi podataka.
Tok događaja – glavni scenario:	1. Administrator bira “Plaćanja” na panelu. 2. Administrator brise podatke o uplati korisnika i potvrdjuje akciju. 3. Sistem vrši željenu izmjenu u bazi.
Tok događaja – alternativni scenariji:	1.a Nema registrovanih uplata korisnika. 2.a Sistem prikazuje obavjestenje.

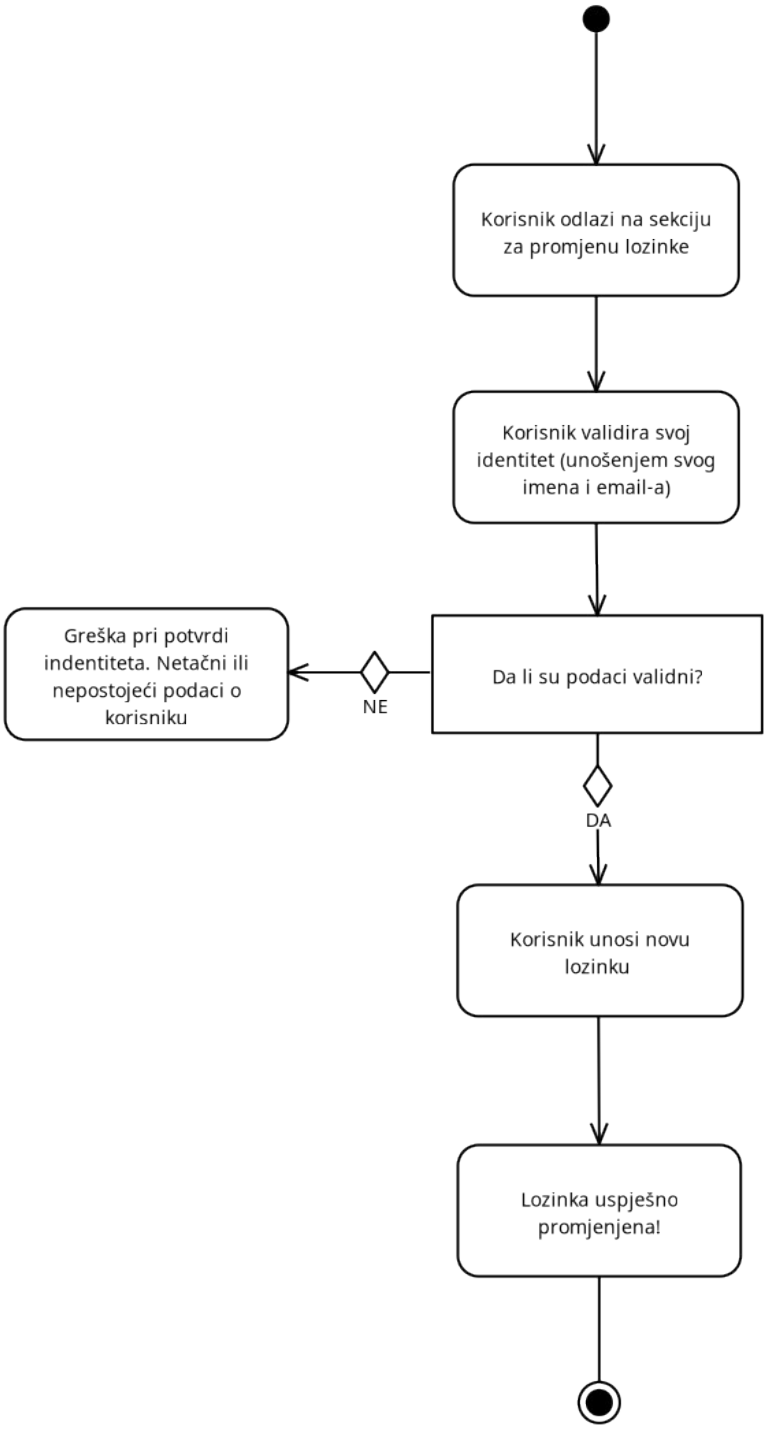
Use Case dijagrami aktivnosti: Registracija



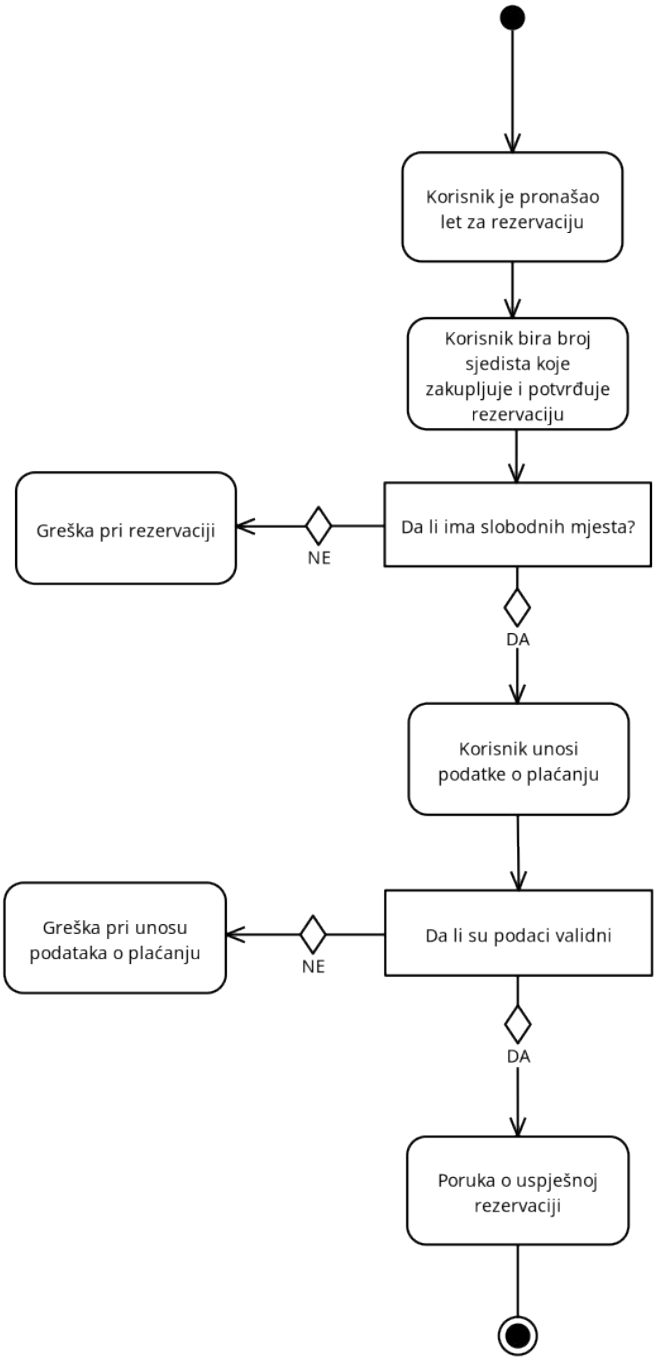
Use Case dijagrami aktivnosti: Prijavljivanje



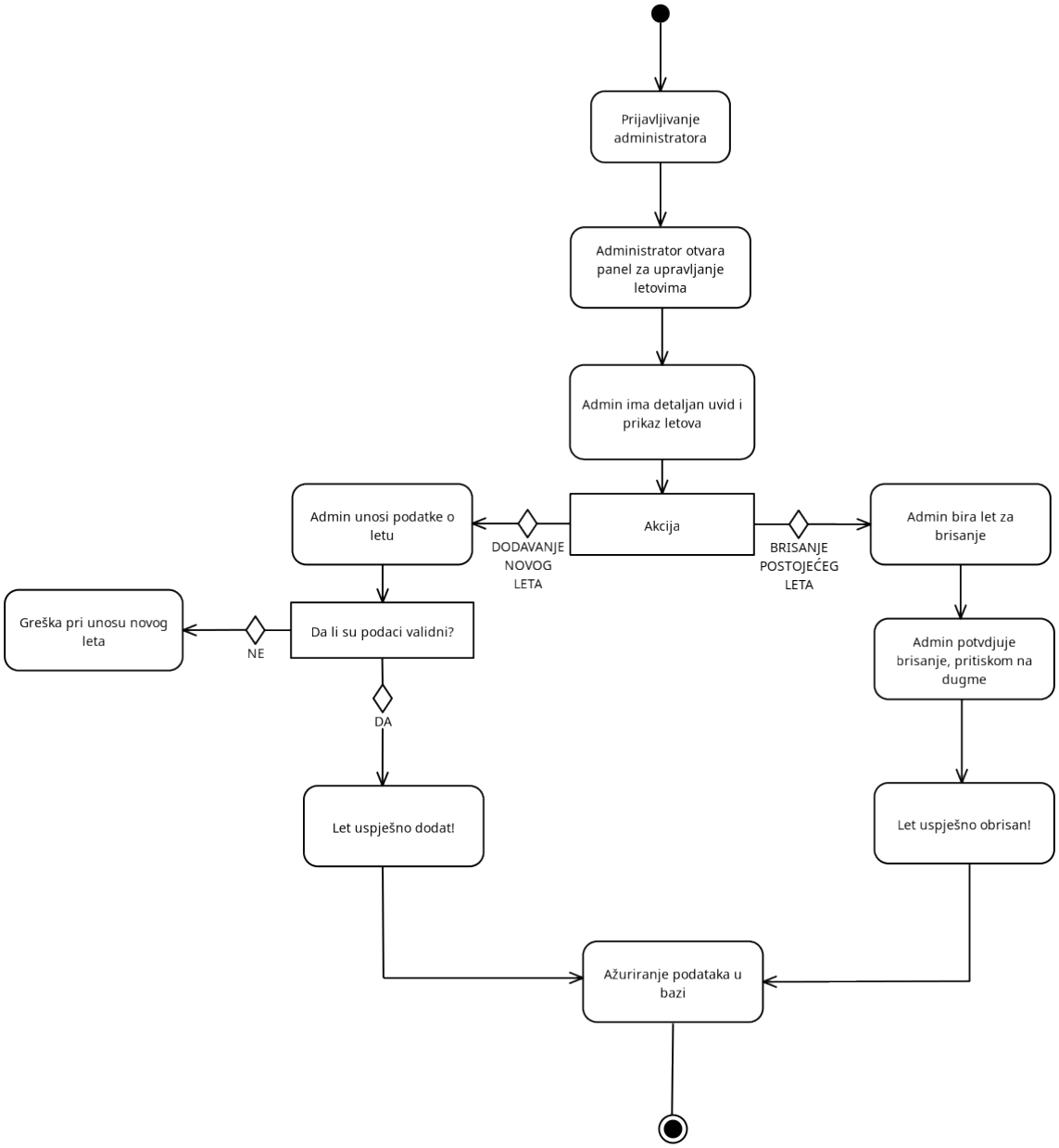
Use Case dijagrami aktivnosti: Promjena lozinke



Use Case dijagrami aktivnosti: Rezervacija i plaćanje

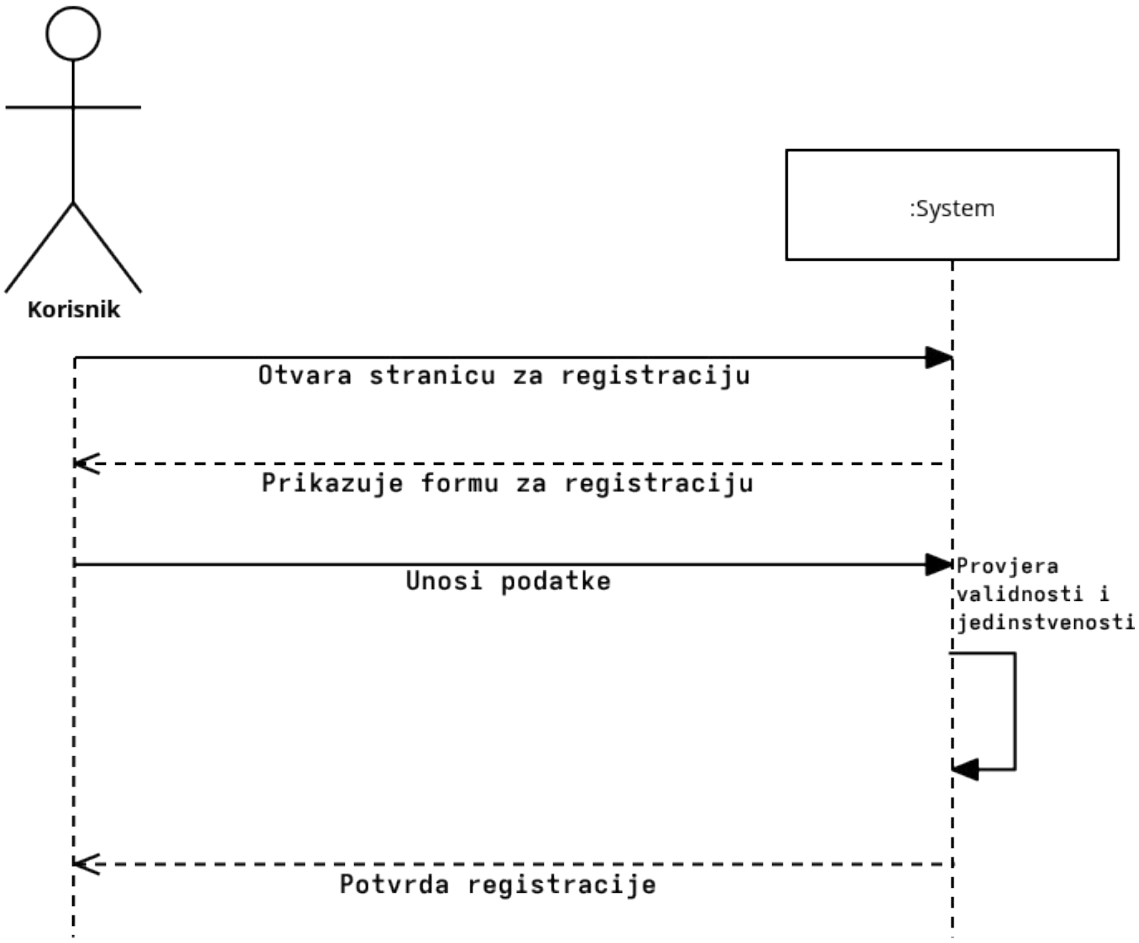


Use Case dijagrami aktivnosti: Upravljanje letovima (admin)

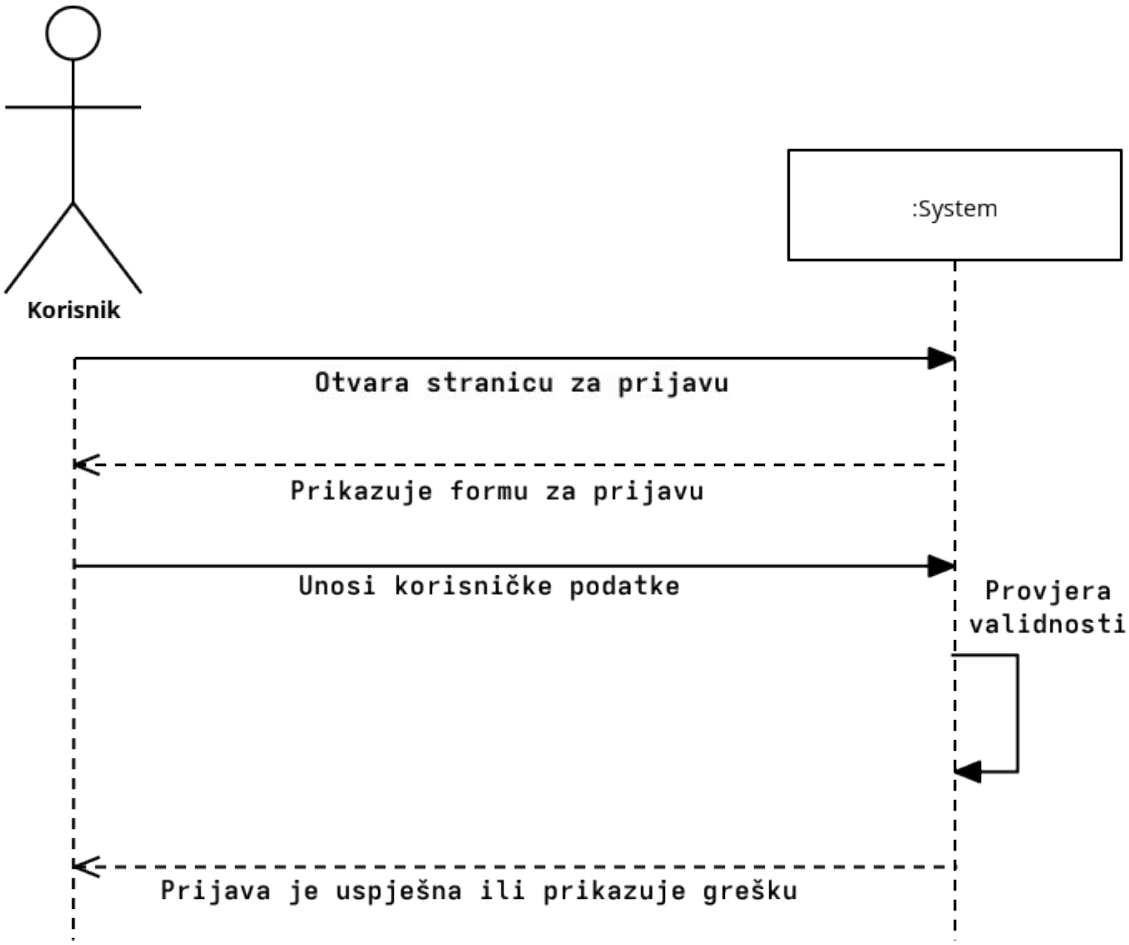


Sistemski dijagrami sekvenci

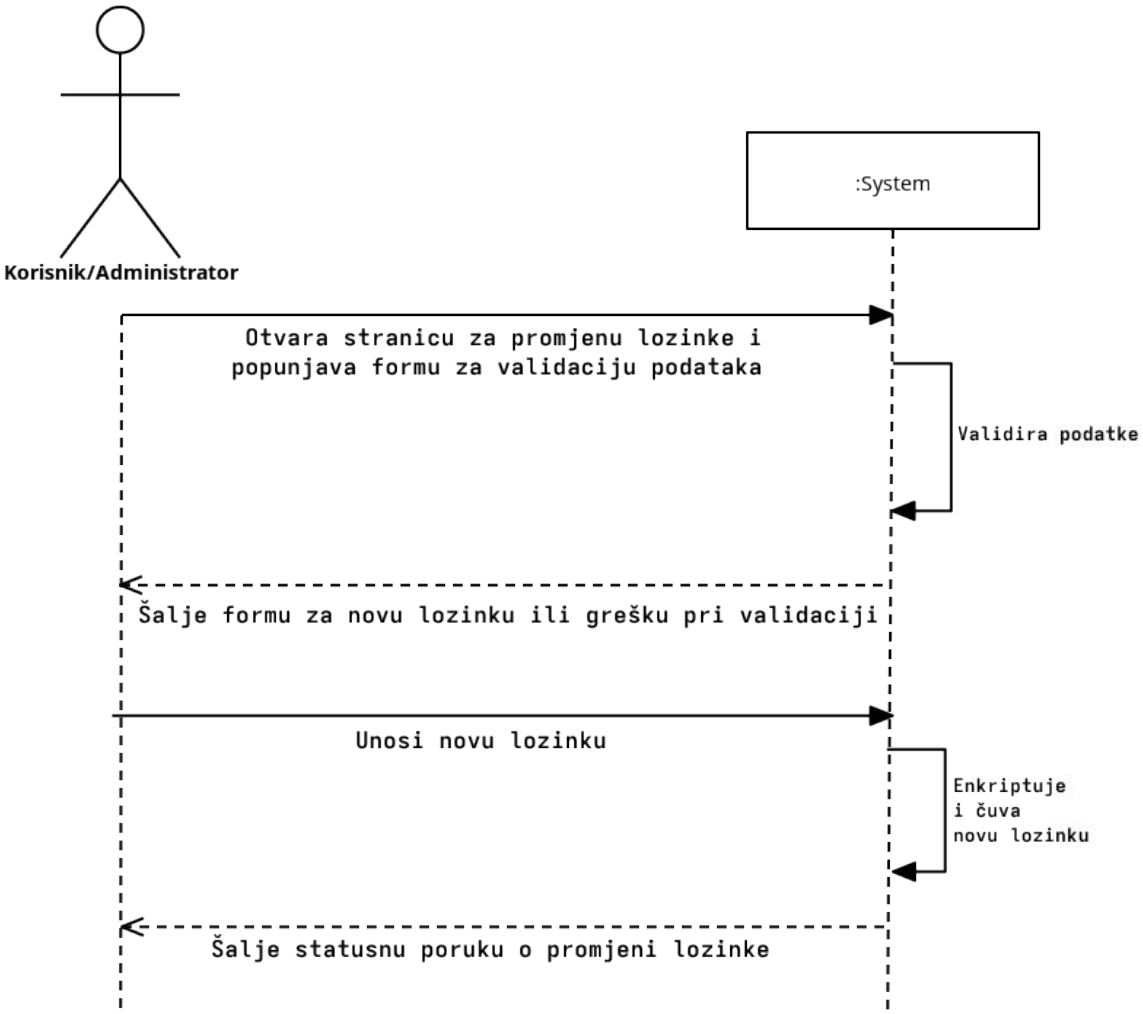
1. Sekvenca: Registracija korisnika



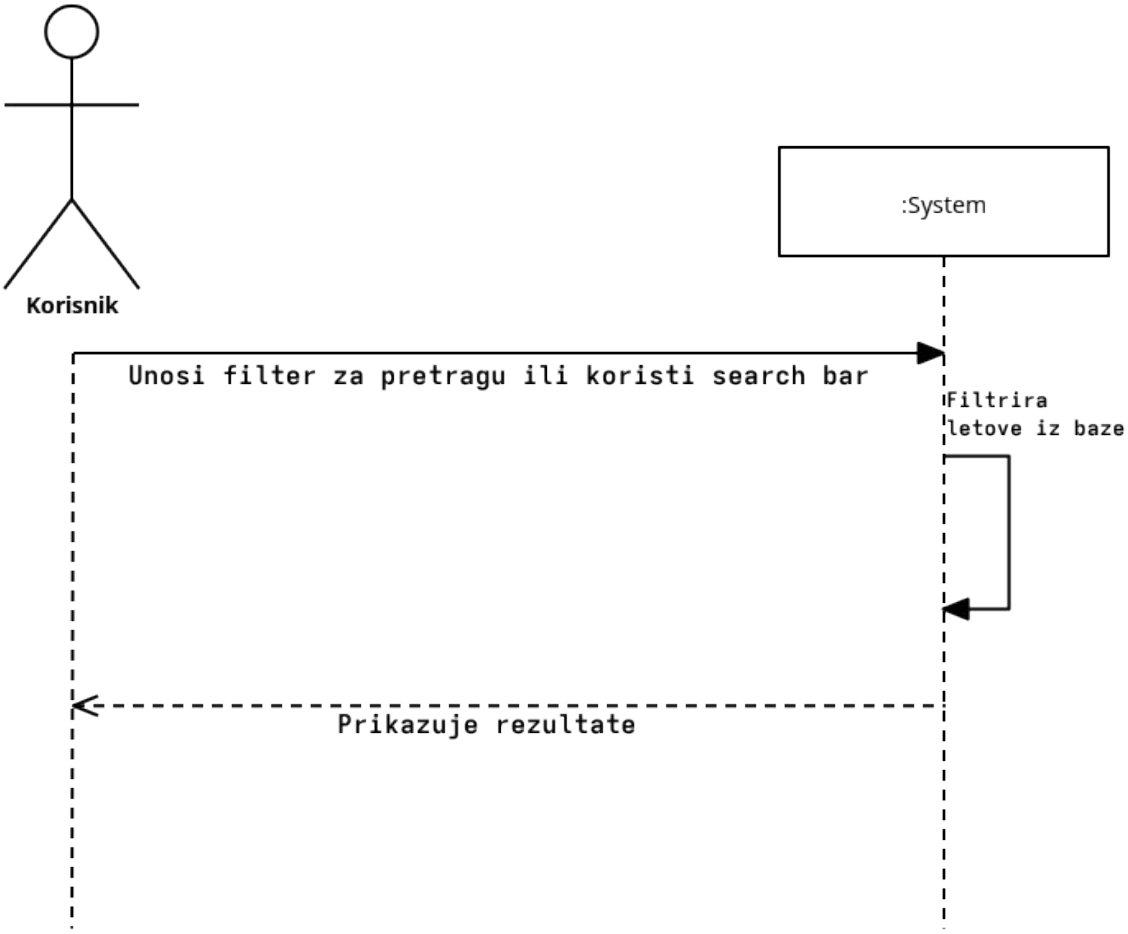
2. Sekvenca: Prijava korisnika



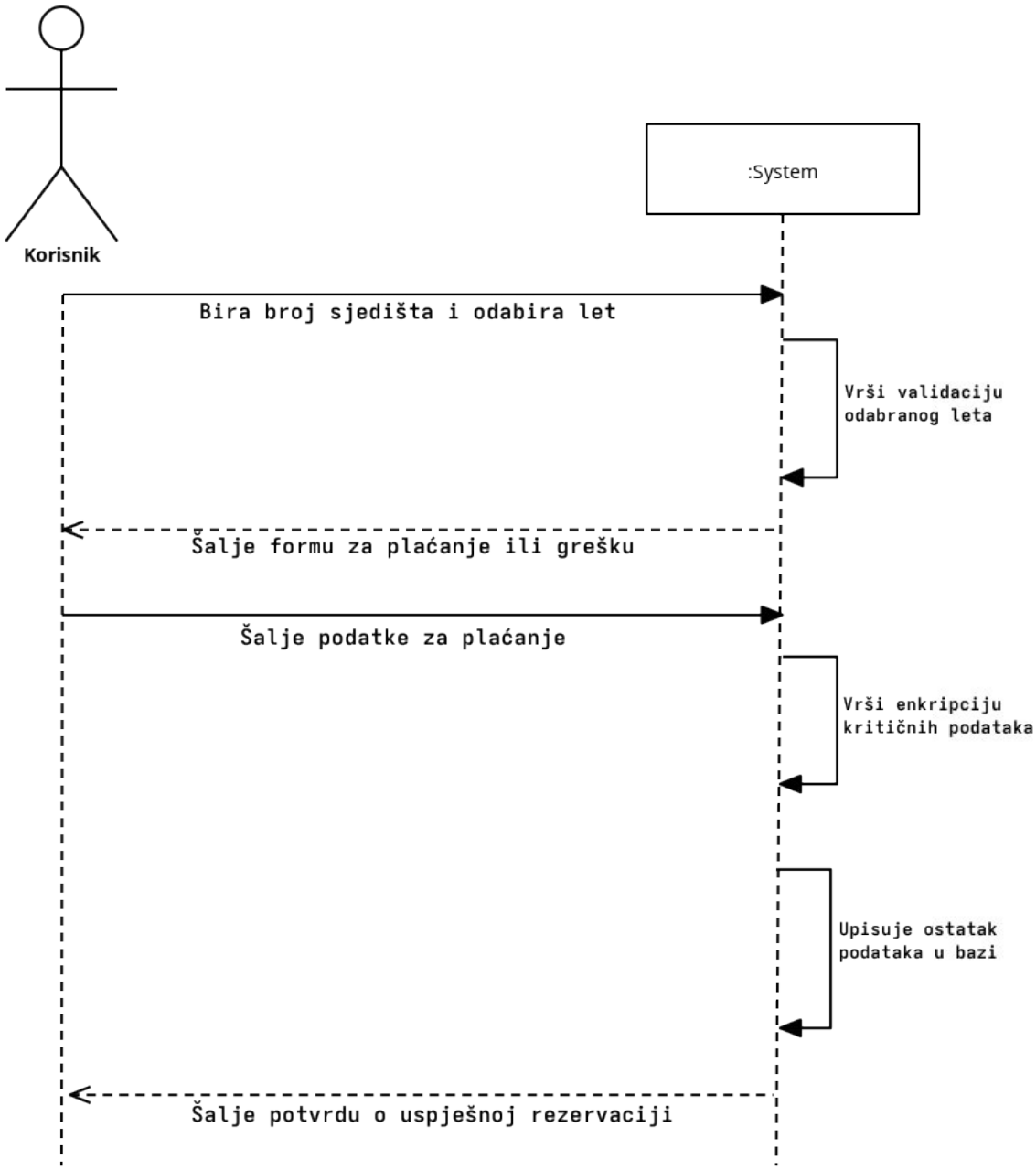
3. Sekvenca: Promjena lozinke



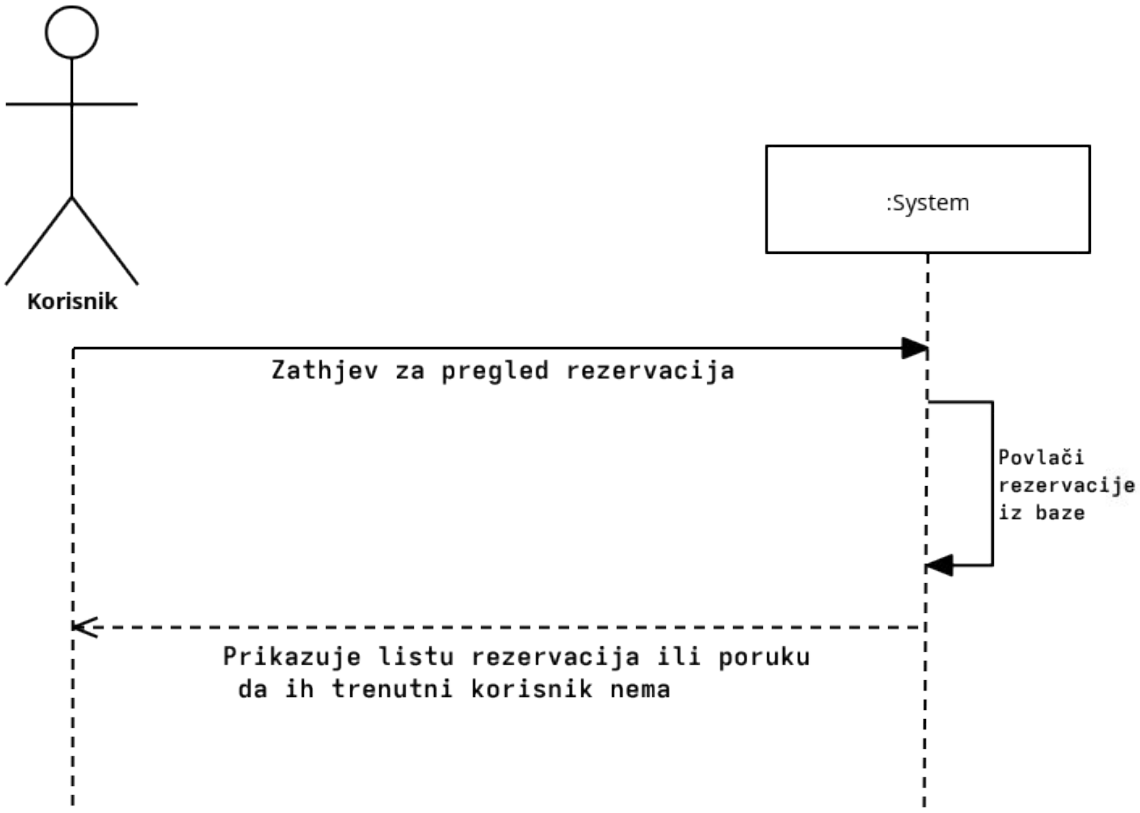
4. Sekvenca: Filtriranje/pretraga letova



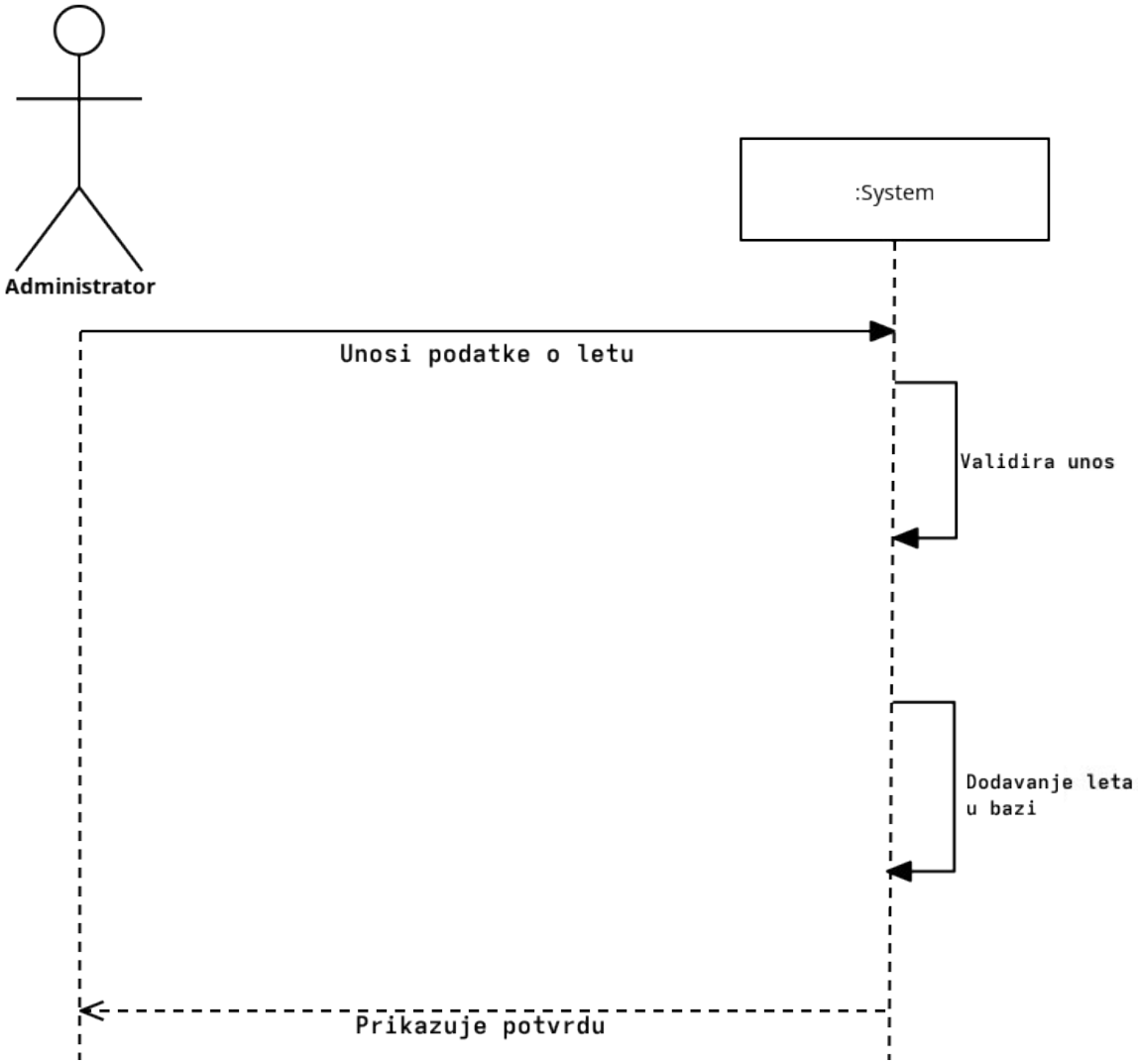
5. Sekvenca: Kreiranje rezervacije i plaćanje



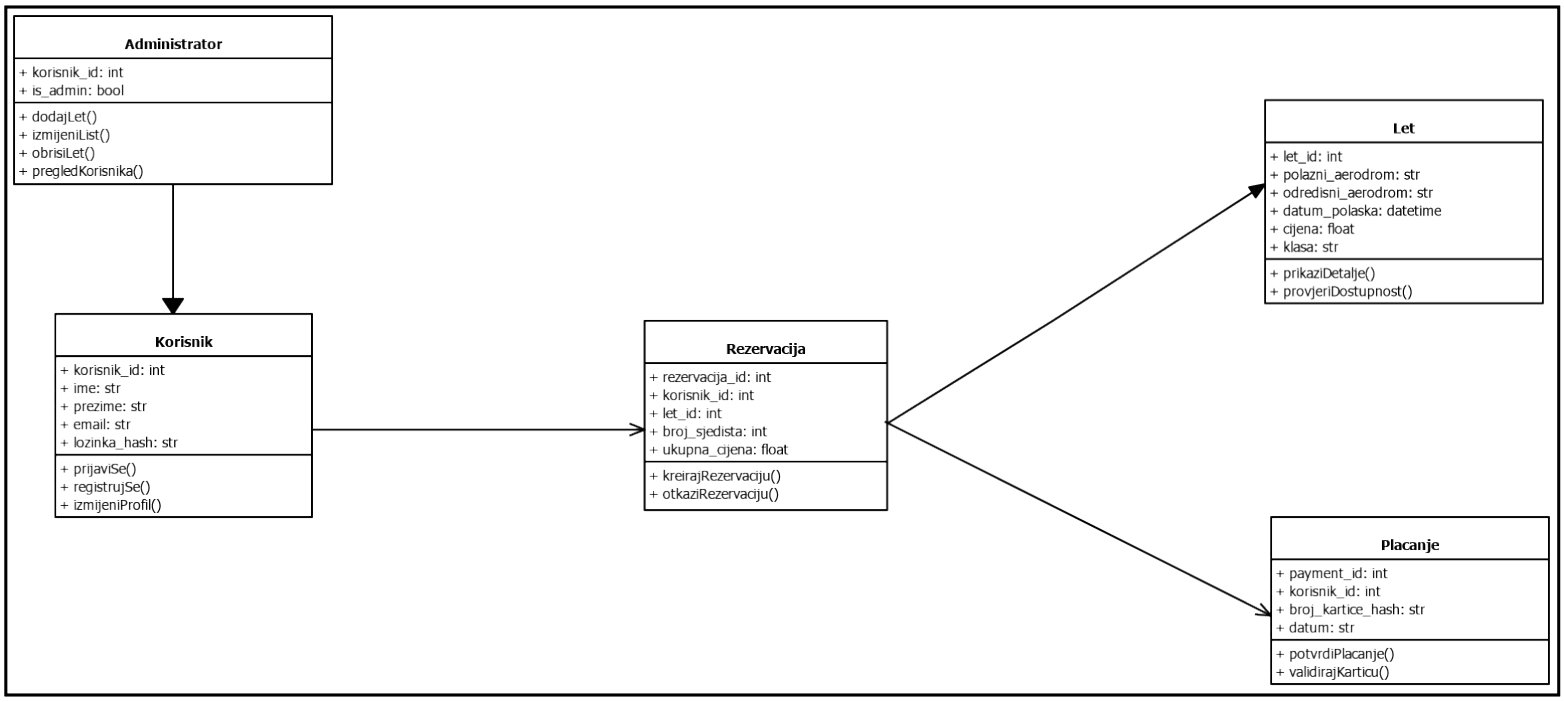
6. Sekvenca: pregled rezervacija



7. Sekvenca: Admin add flight



Dijagram klasa



Relacije između klasa u sistemu odražavaju stvarne logičke veze u okviru poslovnog domena avio kompanije. Klasa **Korisnik** je povezana sa klasom **Rezervacija** odnosom „jedan-prema-više“, jer svaki korisnik može imati više rezervacija. Klasa **Rezervacija** je zauzvrat povezana sa klasom **Let** odnosom „više-prema-jedan“, jer više različitih rezervacija može biti vezano za isti let. Takođe, svaka rezervacija ima tačno jedno plaćanje, što se modeluje relacijom „jedan-prema-jedan“ između klasa **Rezervacija** i **Plaćanje**. Klasa **Administrator** nasljeđuje klasu **Korisnik**, jer predstavlja specijalizovanog korisnika sa dodatnim ovlašćenjima i funkcijama u sistemu.

Opis Baze Podataka

Baza podataka koju koristimo služi za upravljanje podacima o korisnicima, letovima, rezervacijama, plaćanjima i ulogama korisnika. Sadrži više međusobno povezanih tabela koje omogućavaju pravilno funkcionisanje i validaciju podataka u sistemu.

#Tabela Korisnici

Ova tabela sadrži osnovne podatke o korisnicima sistema. Svaki korisnik ima svoj jedinstveni identifikator i dodatne informacije kao što su ime, prezime, e-mail i lozinka.

- korisnik_id – jedinstveni identifikator korisnika
- ime – ime korisnika
- prezime – prezime korisnika
- email – e-mail adresa korisnika (mora biti jedinstvena)
- lozinka_hash – hash vrijednost korisničke lozinke radi sigurnosti
- broj_telefona – broj telefona korisnika (nije obavezan)
- profile_slika – putanja do profilne slike korisnika (nije obavezno)
- is_admin – indikator da li je korisnik administrator, boolean vrijednosti (1 = da, 0 = ne)

#Tabela Rezervacije

Služi za evidentiranje rezervacija korisnika za letove.

- rezervacija_id – jedinstveni ID rezervacije
- korisnik_id – ID korisnika koji pravi rezervaciju
- let_id – ID leta koji je rezervisan

- datum_rezervacije – datum kada je rezervacija napravljena
- status – status rezervacije
- rezervisana_sjedista – broj sjedišta koje je korisnik rezervisao
- ukupna_cijena – ukupna cijena rezervacije

#Tabela Letovi

Ova tabela čuva informacije o svim dostupnim letovima.

- let_id – jedinstveni identifikator leta
- polazni_aerodrom – naziv aerodroma sa kojeg let polazi
- odredisni_aerodrom – naziv aerodroma na koji let stiže
- drzava_input – država iz koje let polazi
- drzava_output – država u koju let ide
- vrijeme_i_datum_polaska – termin polaska
- vrijeme_i_datum_dolaska – termin dolaska
- cijena – cijena karte za taj let
- klasa – klasa leta
- avio_kompanija – naziv aviokompanije
- dostupna_sjedista – broj raspoloživih mjesta
- broj_terminala – oznaka terminala
- broj_izlaza – oznaka izlaza
- trajanje_leta – trajanje leta
- dodatne_informacije – opisne informacije o letu

#Tabela Payment

Čuva informacije o plaćanjima korisnika za rezervacije. Osjetljivi podaci (broj kartice i sigurnosni broj) su kriptovani.

- payment_id – jedinstveni ID transakcije
- korisnik_id – korisnik koji je izvršio plaćanje
- first_name – ime nosioca kartice
- last_name – prezime nosioca kartice
- card_number_hash – hash broja kartice
- date_of_expiration – datum isteka kartice

- security_number_hash – hash sigurnosnog broja kartice (CVV)
- billing_address – adresa naplate
- country – zemlja iz koje je izvršeno plaćanje

#Tabela Roles

Ova tabela identifikuje koje korisnike sistem tretira kao administratore.

- role_user_id – jedinstveni ID zapisa
- korisnik_id – ID korisnika
- is_admin – vrijednost koja označava da li je korisnik admin

Relacija između tabela

Relacije između tabela u sistemu avio kompanije zasnovane su na principima relacionog modela i omogućavaju povezivanje entiteta kroz spoljašnje ključeve (foreign keys), čime se obezbjeđuje integritet podataka i konzistentnost u radu aplikacije.

Tabela rezervacije predstavlja vezu između korisnika i letova. Svaka rezervacija sadrži spoljašnje ključeve korisnik_id i let_id, koji referenciraju odgovarajuće redove u tabelama korisnici i letovi. Ova relacija je **više-prema-jedan** (many-to-one) jer jedan korisnik može imati više rezervacija, ali svaka rezervacija pripada samo jednom korisniku i jednom letu.

Tabela payment je povezana sa tabelom korisnici putem spoljašnjeg ključa korisnik_id. Na taj način se svakom korisniku može pridružiti više plaćanja, što omogućava fleksibilno upravljanje transakcijama – posebno u

slučajevima kada korisnik rezerviše više letova u različitim terminima.

Tabela roles je direktno vezana za korisnici, gdje korisnik_id služi kao jedinstveni ključ koji se koristi za određivanje administratorskih privilegija. Ova relacija je **jedan-prema-jedan** (one-to-one), jer se za svakog korisnika može definisati samo jedna uloga (korisnik ili admin).

Sve relacije su implementirane pomoću **spoljašnjih ključeva (FOREIGN KEY constraints)**, čime se osigurava da ne može doći do kreiranja rezervacije, plaćanja ili dodjele uloge za korisnika koji ne postoji u bazi.

Tehnologije korišćene u projektu

Za izradu full-stack aplikacije, koja treba da simulira avio agenciju, koristili smo različite tehnologije, alate, framework-ove i programske jezike. Upravo iz tog razloga, imali smo mogućnost da napravimo funkcionalnu, sigurnu i interaktivnu platformu za upravljanje letovima, korisnicima, rezervacijama, plaćanjima... Svaki od već navedenih alata i jezika je imao specifičnu ulogu pri kreaciji slojne arhitekture aplikacije, što je dovelo do sigurne funkcionalnosti i robusnosti samog programa.

Python je korisćen kao glavni backend mehanizam samog projekta. On je već dobro poznat kao jednostavan i razumljiv jezik što se tiče sintaksičke perspektive. A pored toga, igra glavnu ulogu pri kreiranju automatizovanih procesa i mehaniziama. Nama je on u projektu bio glavni izvršioc logike samog bakcend API-ja. Upravo zbog svega ovoga bio je korišten za obradu korisničkih podataka i zahtjeva, upravljane sesijama korisnika, validaciju podataka i interakciju sa bazom podataka, uključujuci funkcionalnosti kao što su prijava, registracija, rezervacije i plaćanja...

Flask je framework za Python, koji je bio osnovna logika našeg programa za kreiranje ruta, obradu HTTP zahtjeva, upravljanje sesija, integraciju i komunikaciju sa Jinja templatovima, koji su nam koristili kao glavni komunikatori

između frontenda i backenda aplikacije.

MySQL je korišćen za kreaciju relacionog sisteama za upravljanje bazama podataka. U njemu se skladište svi ključni podaci – korisnici, letovi, rezervacije i transakcije. Omogućava upotrebu stranih ključeva i održavanje

integriteta podataka kroz relacije između tabela. Takođe, podatke sa velikim bezbjednosnim potencijalom smo skladištili pomoću hashova, što je dovelo do dodatno osiguranja bezbjednosti.

Da bi očuvali jednostavnost, robusnost i intuitivnu interakciju samog dizajna, koristili smo ***HTML*** i ***CSS***. Ovo naravno dovodi do minimalnog opterećenja na mrežnom nivou komunikacije sa serverom, pa simim tim sajt ima dobro vrijeme odgovora i visoke brzine.

JavaScript dodaje sloj interaktivnosti korisničkom interfejsu. Korišćen je za validaciju formi, prikaz poruka u realnom

vremenu, dinamičku manipulaciju sadržajem stranice.

Jinja2 predstavlja šablonski način prikazivanja i generisanja HTML stranica na osnovu podataka iz backend-a. Kod nas je služio za organizaciju pri dizajnu stranica, logici i djeljenju zajedničkih

komponenti sajta (navbar, footer, head tagovi).

Werkzeug i Passlib su biblioteke koje služe za bezbjednu obradu korisničkih lozinki i osjetljivih podataka. Omogućavaju hash-ovanje lozinki, čime se osigurava da se podaci ne čuvaju u sirovom obliku, što je od ključne važnosti za sigurnost. Kod nas su korišćeni kako bi očuvali integritet i bezbjednost važnih podataka korisnika (lozinke, brojevi kreditnih kartica). Samim tim, ove biblioteke nam doprinose modernizaciji baze podataka u našoj aplikaciji.

SQL dump* i *backup alati su korišćeni za izvoz strukture i sadržaja baze podataka. Ovi alati olakšavaju migraciju podataka, testiranje i obnavljanje baze u slučaju greške ili gubitka podataka.

HTTPie je moderni, brz, takozvani “command-line” alat za slanje, primanje, testiranje i dijagnostiku HTTP poziva sa serverom. HTTPie je korišćen intenzivno, kako pri ranim tako i kasnim fazama testiranja API-ja. On je doprinjeo jednostavnosti pri nalženju grešaka i dijagnostici u našem projektu.

Sve navedene tehnologije zajedno omogućile su izgradnju pouzdane i funkcionalne aplikacije, sa jasno razdvojenim slojevima: prezentacionim (***HTML/CSS/JavaScript***), aplikacionim (***Python/Flask***) i slojem za upravljanje

podacima (***MySQL***). Ovakva arhitektura doprinosi skalabilnosti, sigurnosti i dugoročnoj održivosti sistema.

U cilju očuvanja jednostavnosti i preglednosti korisničkog interfejsa, odlučili smo da ne koristimo kompleksne frontend framework-e poput Bootstrapa ili drugih vizuelnih biblioteka. Umjesto toga, fokusirali smo se na minimalan i ručno definisan ***HTML/CSS*** pristup kako bismo postigli što lakše učitavanje stranica i intuitivniji vizuelni doživljaj. Ovaj pristup omogućio nam je potpunu kontrolu nad izgledom aplikacije i smanjio složenost korisničkog iskustva, što je naročito važno za krajnje korisnike kojima je prioritet brz pristup osnovnim funkcijama kao što su pretraga, rezervacija i prijava.

Opis djelova aplikacije

Naša aplikacija za avio kompaniju je modularno organizovana i sadrži više ključnih fajlova koji omogućavaju korisnicima interakciju sa sistemom. Svaki fajl ima svoju svrhu i funkcionalnost u okviru sistema. Organizacija aplikacije omogućava lakši razvoj, održavanje i skaliranje sistema. Svaki dio sistema je odgovoran za određeni aspekt funkcionalnosti, a zajednički omogućavaju korisnicima efikasno korišćenje usluga avio agencije.

config_files/

Sadrži konfiguracione fajlove i definicije modela podataka:
config.py – Konfiguracija aplikacije i baze podataka. U njemu je definisan admin account za pristup bazi podataka i tajni kljuc.
models.py – SQLAlchemy ORM modeli za entitete kao što su Korisnik, Let, Rezervacija, Uplata, itd.
__init__.py – Omogućava da folder funkcioniše kao Python

```python
class Korisnik(UserMixin, db.Model):
    __tablename__ = 'korisnici'
    korisnik_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    ime = db.Column(db.String(50))
    prezime = db.Column(db.String(50))
    email = db.Column(db.String(100), unique=True)
    lozinka_hash = db.Column(db.String(500))
    broj_telefona = db.Column(db.String(15), nullable=True)
    profile_slika = db.Column(db.String(200), nullable=True)
    is_admin = db.Column(db.Boolean, default=False)

    rezervacije = db.relationship('Rezervacija', back_populates='korisnik')
    placanja = db.relationship('Placanje', backref='korisnik', lazy=True)
    rola = db.relationship('Role', back_populates='korisnik', uselist=False)
```

paket. U bliskoj vezi sa drugim fajlovima, sluzi kao glavni menadzer između fajlova u našem projektu.

database/

Sadrži SQL skriptu baze podataka:
air_agency.sql – Struktura baze i početni podaci. Uključuje sve tabele i relacije potrebne za rad aplikacije. Glavno skladište našeg projekta, korišćena je za čuvanje svih informacije koje se proslijeđuju preko sajta. Takođe sadrži heširane lozinke, što u slučaju databreach-a moze osigurati korisnike, da će njihovi podaci biti relativno bezbjedni.

```
DROP TABLE IF EXISTS `korisnici`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `korisnici` (
  `korisnik_id` int NOT NULL AUTO_INCREMENT,
  `ime` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `prezime` varchar(50) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `email` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `lozinka_hash` varchar(500) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `broj_telefona` varchar(15) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `profile_slika` varchar(200) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `is_admin` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`korisnik_id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

instance/

Glavna Flask logika ruti:
routes.py – Definiše sve rute aplikacije i vezu između korisničkog interfejsa i funkcionalnosti backend-a. Pored definisanih ruta, fajl takođe sadrži logiku backend-a kao što je na primjer verifikacija admin naloga i određenih privilegija.

templates/

Sadrži HTML fajlove koji se renderuju iz backend-a koristeći Jinju. Zajedno sa JavaScriptom koji se nalazi u njima daju osnovnu logiku za isnpunjavanje formi kod frontend-a. Čine glavni skelet stranice koju korisnik vidi.

Lista html fajlova:

index.html – Početna stranica sa osnovnim informacijama i navigacijom.
login.html / register.html – Forme za prijavu i registraciju korisnika.
letovi.html – Pregled i filtracija letova.
bookings.html – Prikaz korisničkih rezervacija.
placanja.html – Lista uplata, i informacije o korisnicima
admin_letovi.html – Panel za administraciju letova.
change_password.html – Stranica za promjenu lozinke.
navbar.html – Navigacioni meni koji se uključuje u druge stranice.
footer.html – univerzalni footer.
admin_users.html – Admin panel za pregled korisnika.
info.html – informaciona strana o kompaniji.

privacy.html – starana o polici bezbjednosti na sajtu.

static/

Predstavlja folder u kome se nalaze asset-ovi za sajt, poput slika i css-a.

run.py

Glavni pokretački fajl aplikacije. Startuje Flask server i pokreće čitavu aplikaciju.

Dijelovi source coda:

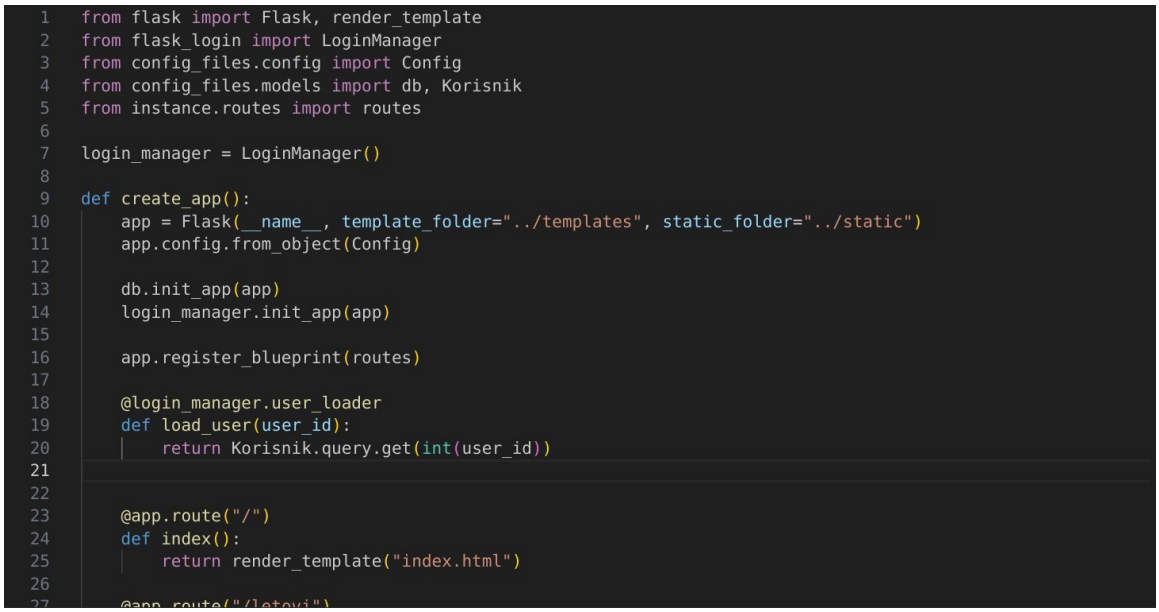
templates > index.html > ...

```
2   <html lang="en">
14  <body>
27      </header>
28
29      <main>
30          <section>
31              <div class="paragaf-1">
32                  <h2>Sigurnost, pouzdanost, fleksibilnost</h2>
33                  <p>
34                      FlyUS je jedinstveno mjesto gdje možete brzo, sigurno i povoljno rezervisati i pregleda
35                  </p>
36                  <p>
37                      Naš tim stručnjaka svakodnevno radi na unapređenju kvaliteta usluge i korisničkog isku
38                      Pratimo moderne trendove, sarađujemo sa najpoznatijim svjetskim avio-kompanijama i pru
39                  </p>
40                  <div class="front_1">
41                      <img src="../static/images/crew.jpg">
42                  </div>
43              </div>
44
45          </section>
46
47          <div class="why_we">
48              <section style="margin-top: 30px;">
49                  <h2>Izbjegnite cekanja, zurbi i odaberite najbolji nacin za rezervacije!</h2>
50                  <p>
51                      Kod nas se mozete osjecati bezbjedno da cete dobiti sledece usluge:</p>
```

Pr.1 index.html

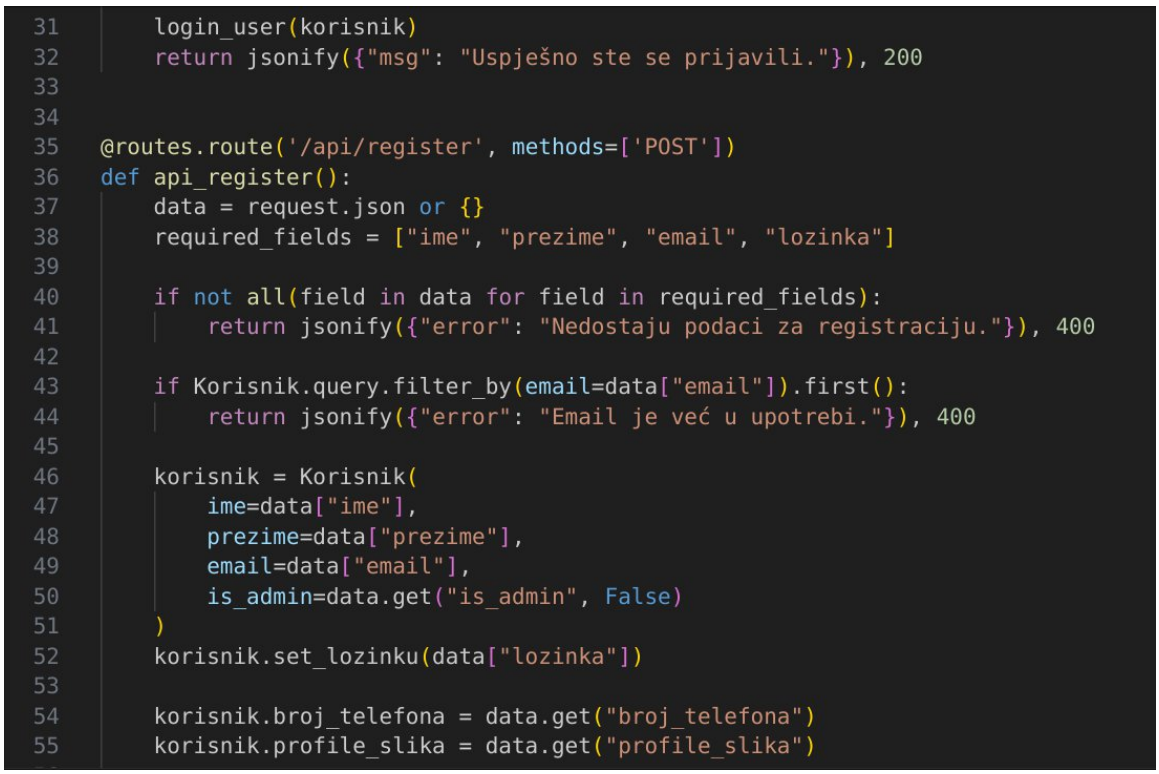
```
105
106 class Placanje(db.Model):
107     __tablename__ = 'payment'
108     payment_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
109     korisnik_id = db.Column(db.Integer, db.ForeignKey('korisnici.korisnik_id'))
110     first_name = db.Column(db.String(50))
111     last_name = db.Column(db.String(50))
112     card_number_hash = db.Column(db.String(200))
113     date_of_expiration = db.Column(db.String(20))
114     security_number_hash = db.Column(db.String(200))
115     billing_address = db.Column(db.String(200))
116     country = db.Column(db.String(50))
117
118     def set_card_data(self, card_number, security_number):
119         self.card_number_hash = generate_password_hash(card_number)
120         self.security_number_hash = generate_password_hash(security_number)
121
122     def verify_card_number(self, card_number):
123         return check_password_hash(self.card_number_hash, card_number)
124
125     def verify_security_number(self, security_number):
126         return check_password_hash(self.security_number_hash, security_number)
127
128
129 @event.listens_for(Korisnik, 'after_insert')
130 def after_insert_korisnik(mapper, connection, target):
```

Pr.2 models.py



```python
from flask import Flask, render_template
from flask_login import LoginManager
from config_files.config import Config
from config_files.models import db, Korisnik
from instance.routes import routes

login_manager = LoginManager()

def create_app():
    app = Flask(__name__, template_folder="../templates", static_folder="../static")
    app.config.from_object(Config)

    db.init_app(app)
    login_manager.init_app(app)

    app.register_blueprint(routes)

    @login_manager.user_loader
    def load_user(user_id):
        return Korisnik.query.get(int(user_id))


    @app.route("/")
    def index():
        return render_template("index.html")

    @app.route("/letovi")
```

Pr.3 __init__.py



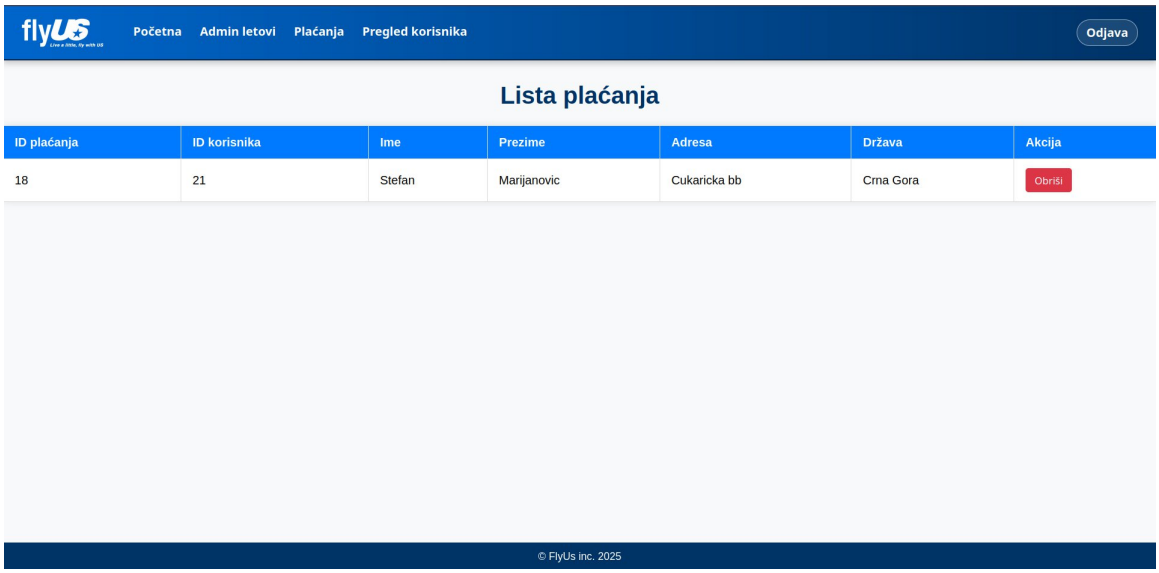
```python
        login_user(korisnik)
        return jsonify({"msg": "Uspješno ste se prijavili."}), 200


@routes.route('/api/register', methods=['POST'])
def api_register():
    data = request.json or {}
    required_fields = ["ime", "prezime", "email", "lozinka"]

    if not all(field in data for field in required_fields):
        return jsonify({"error": "Nedostaju podaci za registraciju."}), 400

    if Korisnik.query.filter_by(email=data["email"]).first():
        return jsonify({"error": "Email je već u upotrebi."}), 400

    korisnik = Korisnik(
        ime=data["ime"],
        prezime=data["prezime"],
        email=data["email"],
        is_admin=data.get("is_admin", False)
    )
    korisnik.set_lozinku(data["lozinka"])

    korisnik.broj_telefona = data.get("broj_telefona")
    korisnik.profile_slika = data.get("profile_slika")
```

Pr.4 Routes.py

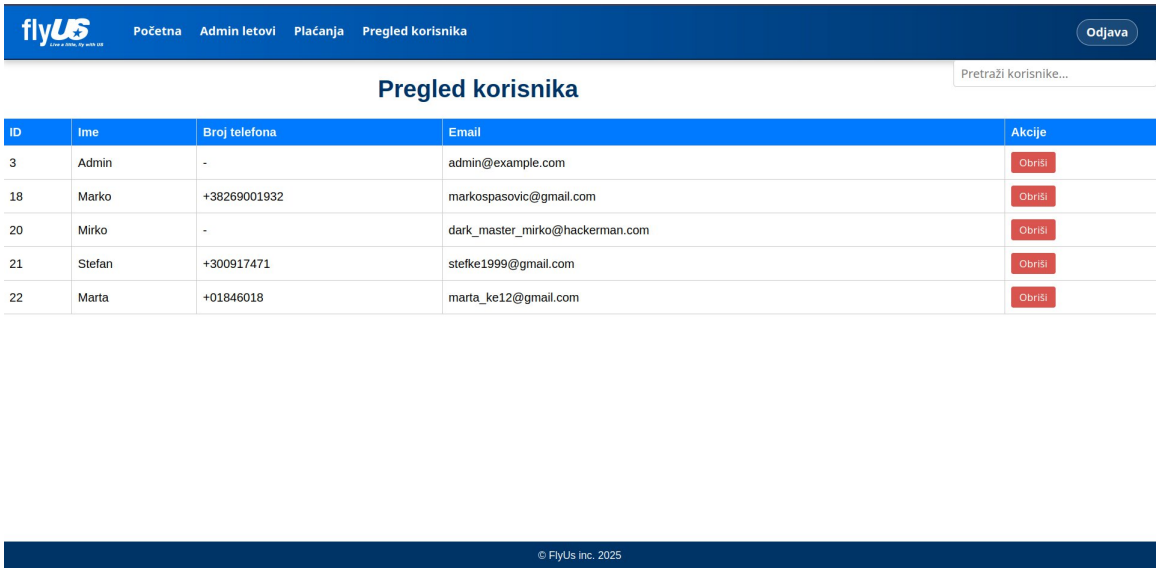
Izgled sajta:

Pr. Pocetna stranica sajta

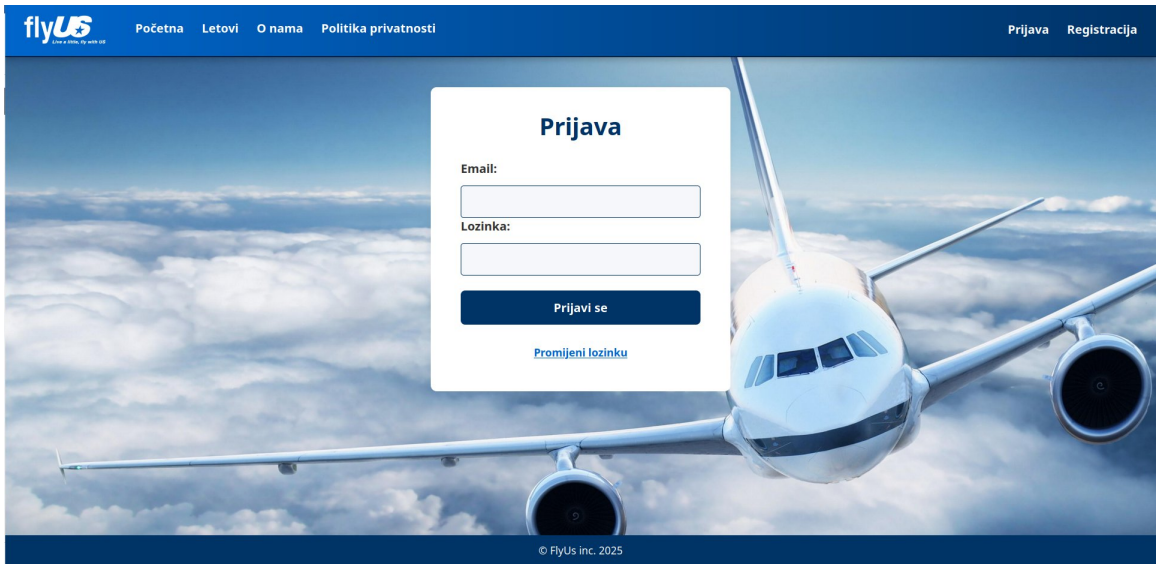
Pr. Admin panel sa uvidom u letove



Pr. Admin panel za uplate

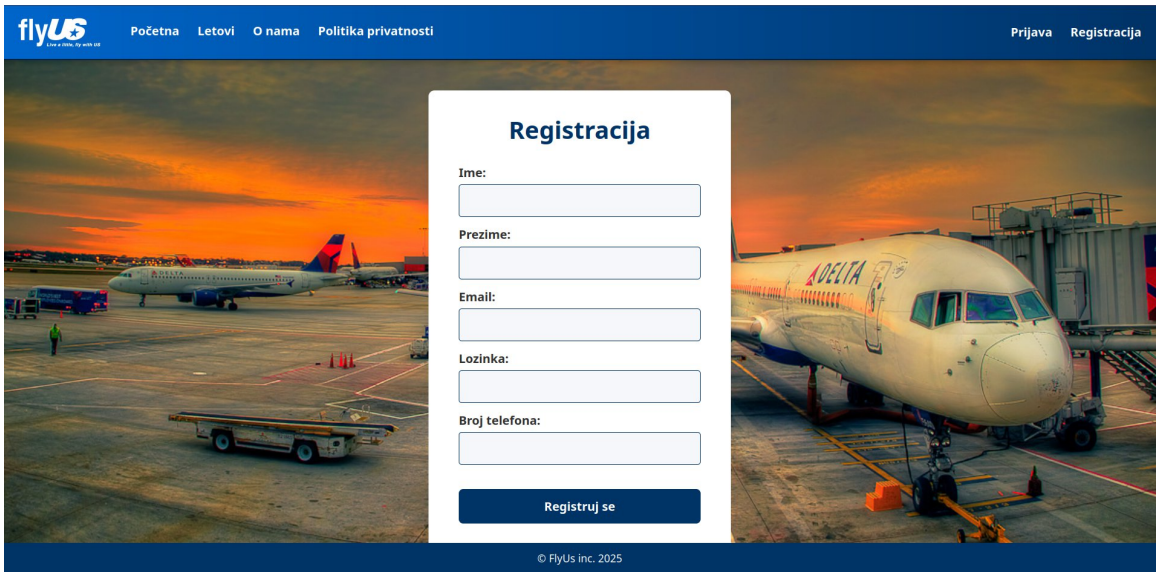


Pr. Admin panel za prikaz registrovanih korisnika

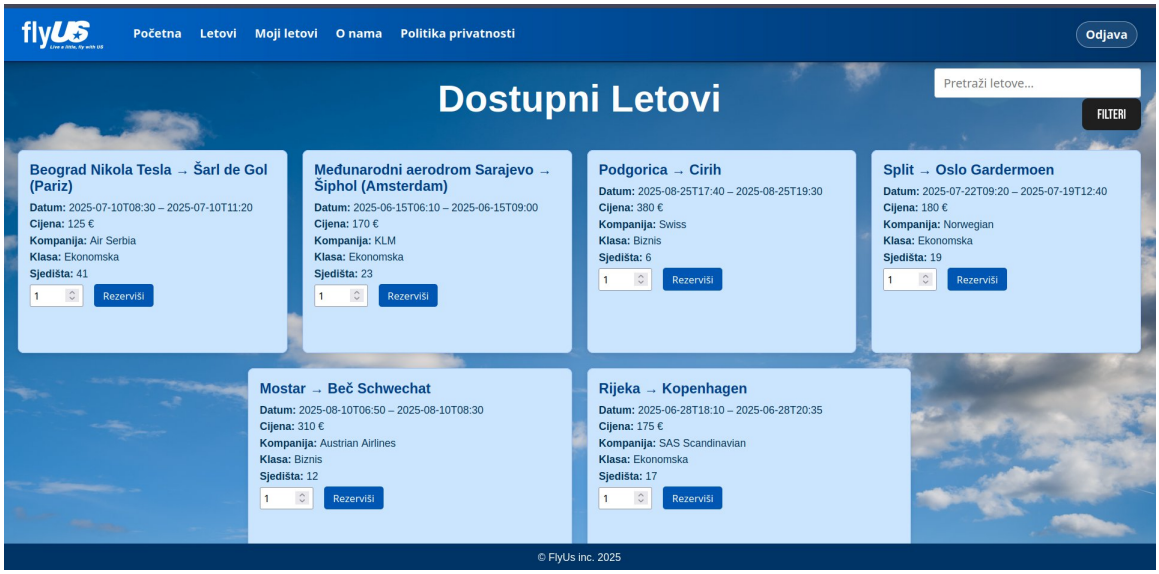


Pr. Stranica za prijavu

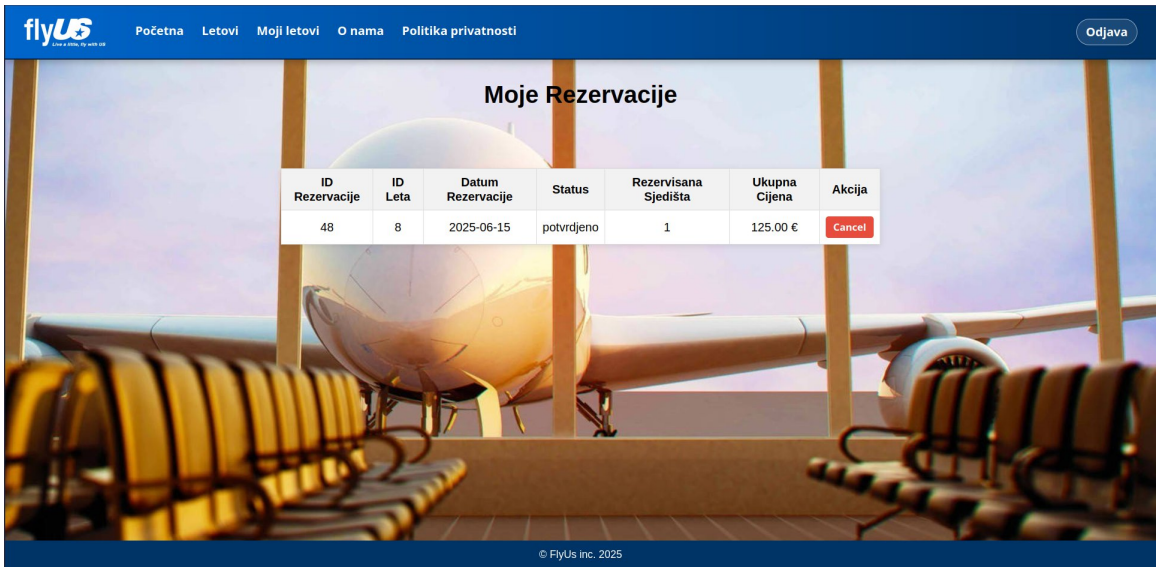
Pr. Stranica za promjenu lozinke



Pr. Stranica za registraciju



Pr. Stranica za Letove



Pr. Stranica za rezervacije

Dokumentacija i izvori:

1. Python: https://docs.python.org/
2. Jinja, Jinja2: https://jinja.palletsprojects.com/ , https://www.devdoc.net/python/jinja-2.10.1-doc/
3. Flask: https://flask.palletsprojects.com/ , https://python-adv-web-apps.readthedocs.io/en/latest/flask.html