Razvoj i implementacija AI SAAS platforme

Starčević, Karlo

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI PRIJEDIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Karlo Starčević

ZAVRŠNI RAD

RAZVOJ I IMPLEMENTACIJA AI SAAS PLATFORME

Zagreb, listopada 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni prijediplomski studij informacijskih tehnologija smjer programiranje
Student:	**Karlo Starčević**
Matični broj:	2019020

Zadatak završnog rada

Predmet:	Programiranje na Internetu
Naslov:	**Razvoj i implementacija AI SAAS platforme**
Zadatak:	Razviti sustav AI SAAS platforme korištenjem tehnologija: Typescript, Next.js, Tailwindcss, ZOD, Prisma.
Mentor:	Mariza Maini, pred.
Zadatak uručen kandidatu:	14.4.2024.
Rok za predaju rada:	3.10.2024.
Rad predan:	_____

Povjerenstvo:

Jurica Đurić, v. pred.	član predsjednik	_____
Mariza Maini, pred.	mentor	_____
mr. sc. Julijan Šribar, v. pred.	član	_____

SADRŽAJ

POPIS SLIKA

POPIS KODOVA

1. UVOD

U današnjem digitalnom dobu, napredak u području umjetne inteligencije (engl. *Artificial Intelligence* – AI) sa sobom donosi širok spektar mogućnosti. Jedan od izazova je integracija različitih AI tehnologija unutar softverskih aplikacija radi centralizacije sustava, ubrzanja korištenja te kohezije funkcionalnosti. Tradicionalni pristupi često uključuju korištenje pojedinačnih AI API-a (kratica od engl. *Application Programming Interface*) ili rješenja, što može rezultirati lošijim ili fragmentiranim iskustvom korisnika i kompleksnijom implementacijom. Kako bi se prevladali ovi izazovi, potrebno je razviti rješenje koje će omogućiti učinkovitu integraciju i koherentnost različitih AI sustava.

Ovaj rad analizira problem decentraliziranosti AI-jeva, istražuje zašto bi bilo korisno integrirati ih u jedan centralizirani servis te identificira niz tehnologija koje će olakšati pristup različitim sustavima unutar jednog zajedničkog. Uspješna integracija AI tehnologija omogućava efikasnije iskorištavanje resursa te poboljšava skalabilnost sustava. Fokus je na stvaranju centralizirane platforme koja pruža jednostavno i intuitivno korisničko iskustvo, dok istovremeno omogućuje brzu implementaciju i prilagodbu novih AI funkcionalnosti.

Popis tehnologija koje su korištene za razvoj aplikacije uključuje:

1. Next.js - Framework za izgradnju modernih mrežnih aplikacija, koji omogućava brz razvoj i optimizaciju performansi.
2. MySQL - Relacijska baza podataka koja se koristi za pohranu podataka o korisnicima i njihovim aktivnostima.
3. Prisma - ORM (kratica od engl. *Object-Relational Mapping*) alat koji pojednostavljuje interakciju s bazom podataka putem JavaScript koda.
4. Shadcn/UI - Kolekcija koja pruža modularne komponente i stilove za brzo dizajniranje suvremenih korisničkih sučelja.
5. Axios - JavaScript biblioteka za izvođenje HTTP (kratica od engl. *Hypertext Transfer Protocol*) zahtjeva od klijenta prema poslužitelju i obratno.
6. Replicate AI API - Platforma koja pruža alate za stvaranje i upravljanje AI modelima (u ovom radu korištena za audio i video).
7. ChatGPT API - API za razvoj funkcionalnih chat usluga koji koristi tehnologiju generativnog prevođenja jezika i generiranje slika.

Ovaj interdisciplinarni pristup omogućava korištenje najboljih alata i tehnologija iz različitih područja kako bi se stvorio integrirani sustav koji zadovoljava potrebe korisnika na najbolji mogući način. Sinergija ovih tehnologija omogućava razvoj robusne i fleksibilne aplikacije koja može odgovoriti na različite zahtjeve korisnika, dok istovremeno pruža visoku razinu performansi i sigurnosti, kao i skalabilnost.

U daljnjem tekstu ovog rada detaljno se analizira kako se koriste navedene tehnologije u razvoju aplikacije i kako integracija različitih AI tehnologija doprinosi funkcionalnosti i korisničkom iskustvu. Poseban naglasak stavljen je na evaluaciju performansi i sigurnosti sustava, kao i na mogućnosti daljnjeg unaprjeđenja i skaliranja aplikacije. Analiziraju se i potencijalni izazovi u integraciji te predlažu rješenja za njihovo prevladavanje, s ciljem stvaranja optimalnog korisničkog iskustva i maksimalnog iskorištavanja mogućnosti koje pružaju moderne AI tehnologije.

2. PROBLEMATIKA FRAGMENTIRANOSTI UMJETNE INTELIGENCIJE

Jedan od uobičajenih izazova pri korištenju više AI usluga jest fragmentacija usluga na različitim platformama. To može rezultirati problemima poput decentraliziranih podataka za korisnika, podataka u izoliranim sustavima te povećane složenosti u upravljanju i integraciji tih usluga. Spajanjem svih AI usluga na jednom mjestu, primjerice u SAAS (kratica od engl. *Software As A Service*) aplikaciju, korisnici mogu imati koristi od optimiziranih radnih tokova, centraliziranog upravljanja podacima, jednostavnije integracije te povezanosti između samih AI-jeva. Time se poboljšava učinkovitost korištenja AI funkcionalnosti unutar aplikacije.

2.1. Izazovi fragmentacije usluga kod korištenja više platformi umjetne inteligencije

Korištenje različitih AI usluga raspoređenih na različitim platformama može rezultirati brojnim poteškoćama. Jedan od ključnih izazova je nedosljednost u rezultatima i performansama. Osim toga, podaci su često smješteni u izoliranim sustavima, što otežava njihovo učinkovito korištenje i analizu. Integracija ovih servisa može biti kompleksna i zahtijevati dodatne napore za upravljanje i održavanje.

2.2. Prednosti integracije usluga umjetne inteligencije u aplikaciju

Konsolidacija svih AI usluga unutar jedne SAAS aplikacije donosi mnoge prednosti. Optimizirani radni tokovi olakšavaju krajnjim korisnicima upravljanje svim uslugama na jednom mjestu, što znači i povezivanje funkcionalnosti AI-jeva. Centralizirano upravljanje podacima omogućuje bolju kontrolu i pristup informacijama. Integracija postaje jednostavnija, omogućavajući bolju suradnju između različitih servisa. Konačno, korisnici mogu uživati u koherentnom korisničkom iskustvu bez prekida, što rezultira učinkovitijim i uspješnijim iskorištavanjem AI mogućnosti na jednom mjestu.

2.3. Skalabilnost aplikacije

Povezivanjem različitih AI-jeva omogućuje se integracija njihovih funkcionalnosti, što otvara vrata za laku nadogradnju aplikacije novim AI modelima ili dodavanje novih funkcionalnosti u bilo koje vrijeme. Ova fleksibilnost omogućuje aplikaciji dinamičan i brz rast, prilagodbu potrebama korisnika i novim tehnološkim mogućnostima, bez potrebe za promjenama u arhitekturi aplikacije.

3. ODABIR TEHNOLOGIJA

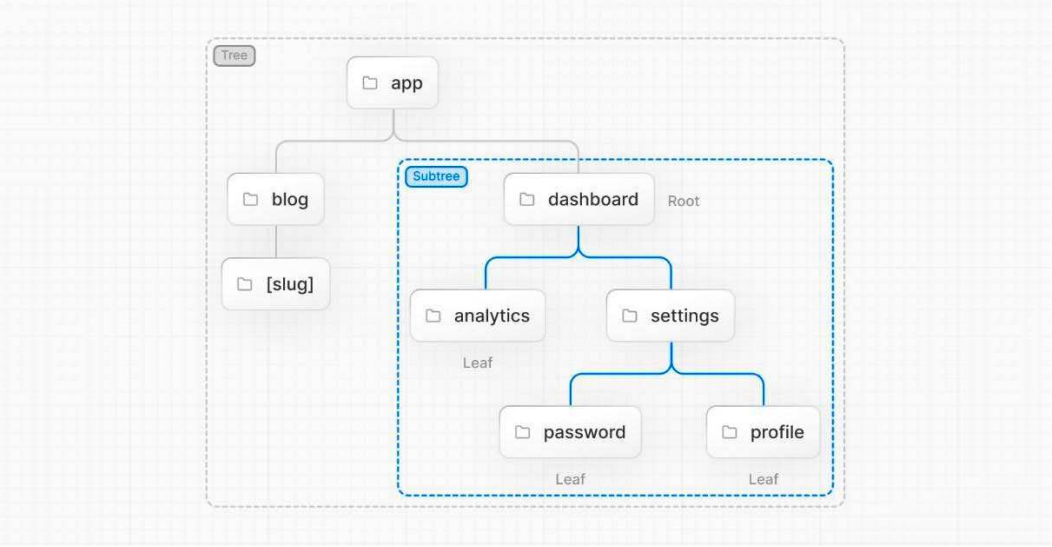
U ovom poglavlju istražuju se razlozi i detaljno objašnjavaju odabiri specifičnih tehnologija kako bi se postigli funkcionalnost, skalabilnost i korisničko iskustvo u projektu. Kombinacija tehnologija kao što su Next.js, MySQL, Prisma, Shadcn/UI (baziran na Tailwind CSS-u), Axios, Stripe, Replicate AI API i ChatGPT API pruža sve što je potrebno za dobre temelje projekta.

3.1. Next js

Next.js je odabran kao glavni framework za izgradnju ove aplikacije zbog optimalne kombinacije performansi, jednostavnosti upotrebe i naprednih funkcionalnosti. Renderiranje na strani poslužitelja (engl. *server-side rendering*) i generiranje statičkih stranica pružaju brzi, SEO-prijateljski pristup (kratica od engl. *Search Engine Optimization),* što je ključno za poboljšanje korisničkog iskustva i optimizaciju mrežnih stranica za pretraživače. Također, mogućnost automatskog optimiziranja aplikacije, uključujući automatsko učitavanje kodeksa i pre-renderiranje ruta, čini Next.js idealnim izborom za izgradnju skalabilnih mrežnih aplikacija. Jednostavna integracija s drugim tehnologijama, poput Reacta i raznih API-ja, pruža fleksibilnost i olakšava razvojni proces. Sve ove karakteristike čine Next.js moćnim alatom za razvoj modernih mrežnih aplikacija s visokim performansama i kvalitetnim korisničkim iskustvom. „Razvoj SaaS aplikacija korištenjem Next.js omogućuje visok stupanj skalabilnosti i efikasnosti zahvaljujući server-side renderiranju.“ (Herron, D. (2020) Node.js Web Development : Packt Publishing.)

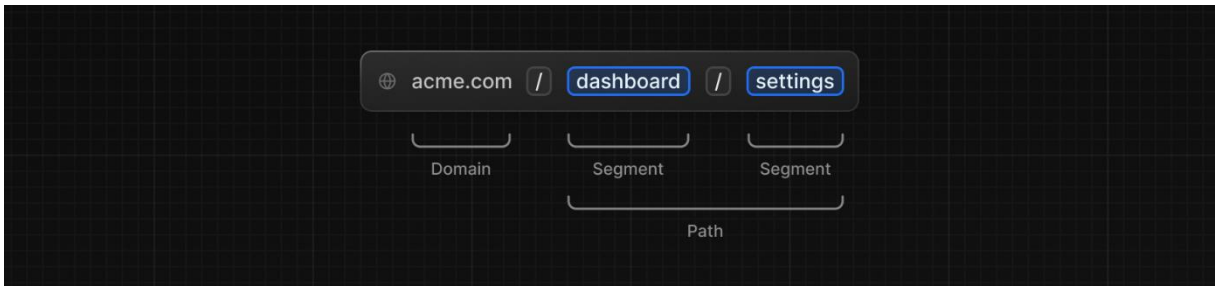
3.1.1. Next js rute

Next.js dolazi s ugrađenim navigacijskim sustavom (engl. *routing*) koji pruža jednostavno definiranje ruta i navigaciju između stranica (Slika 1).



Slika 1. Navigacija u Next.js (Next.js, dokumentacija – *Routing*)

Navigacija se u Next.js temelji na strukturi direktorija. Svaki direktorij predstavlja rutu. Iz primjera na Slika 2, može se zaključiti da ruta „/dashboard/settings“ predstavlja postavke na ruti „/dashboard“ te podruti „/settings“ ().



Slika 2. Segmentiranje URL-a (Next.js, dokumentacija – Navigacija)

3.1.2. Generiranje na strani poslužitelja

SSG (kratica od engl. *Server-Side Generation*) je tehnika koju Next.js koristi za generiranje HTML-a (kratica od engl. *Hypertext Markup Language*) za mrežne stranice tijekom procesa izgradnje (engl. *build*). Ova metoda omogućava da se HTML stranice generiraju dinamički na poslužiteljskoj strani prije nego što se dostave klijentima. SSG kombinira prednosti statičkog i dinamičkog renderiranja mrežne stranice. Jedna od glavnih prednosti SSG-a je poboljšana brzina učitavanja stranica. Budući da se HTML generira tijekom procesa izgradnje, korisnicima se odmah dostavlja potpuno renderirana stranica, bez potrebe za dodatnim generiranjem sadržaja na strani klijenta. Ovo značajno smanjuje vrijeme učitavanja stranica i poboljšava korisničko iskustvo.

SSG također poboljšava optimizaciju za pretraživače (SEO). Unaprijed renderirane HTML stranice omogućavaju tražilicama da lakše indeksiraju sadržaj, što može rezultirati boljom pozicijom u rezultatima pretraživanja i čime se osigurava veća vidljivost i pristupačnost sadržaja na mreži ().

Next.js pruža jednostavnu implementaciju SSG-a korištenjem funkcije *getStaticProps*. Ona nudi dohvaćanje potrebnih podataka tijekom procesa izgradnje, koji se zatim koriste za generiranje statičkih HTML stranica. Podaci mogu biti dohvaćeni iz različitih izvora, uključujući baze podataka, vanjske API-je ili lokalne datoteke, pružajući fleksibilnost i prilagodljivost u generiranju sadržaja. Budući da se većina obrade podataka odvija tijekom procesa izgradnje, smanjuje se rizik od izlaganja osjetljivih podataka klijentskoj strani, što povećava sigurnost aplikacije (stranice). Generiranjem statičkih stranica unaprijed, smanjuje se potreba za dinamičkim pozivima prema poslužitelju, što može smanjiti postotak napada i poboljšati sigurnost aplikacije.

Jedna od ključnih funkcionalnosti SSG-a u Next.js-u je mogućnost inkrementalne statičke regeneracije (ISR). ISR omogućava da se pojedine stranice regeneriraju nakon određenog vremenskog intervala, bez potrebe za ponovnom izgradnjom cijele aplikacije. Ovo osigurava da stranice uvijek prikazuju ažurirane podatke, zadržavajući prednosti statičkog generiranja.

Korištenje SSG-a u aplikaciji omogućava brze, sigurnije i SEO-optimizirane mrežne stranice. Integracija ove tehnike s Next.js pruža moćne alate za generiranje sadržaja unaprijed, osiguravajući optimalno korisničko iskustvo i visoku performansu aplikacije.

3.1.3. Renderiranje na strani poslužitelja

Next.js pruža tri glavne metode renderiranja mrežnih stranica: *Client-Side Rendering* (CSR), *Static-Site Generation* (SSG) i *Server-Side Rendering* (SSR). Svaka od ovih tehnika ima svoje prednosti i primjenu u zavisnosti od potreba aplikacije.

SSR i SSG omogućuju bolje performanse i SEO optimizaciju u odnosu na CSR, gdje klijent mora čekati da JavaScript generira sadržaj na klijentskoj strani. Kod SSR-a, sadržaj se dinamički generira na poslužitelju za svaki zahtjev, dok se kod SSG-a stranice unaprijed generiraju i pohranjuju kao statičke datoteke.

Server-Side Rendering je tehnika koju Next.js koristi za generiranje HTML-a za mrežne stranice na poslužiteljskoj strani u stvarnom vremenu, prije nego što se sadržaj dostavi klijentima. SSR odrađuje dinamičko generiranje HTML stranica na poslužitelju, što može

značajno poboljšati performanse aplikacije i SEO rezultate. Ovo je posebno korisno za korisnike sa sporijim internet vezama ili uređajima sa slabijim performansama, jer smanjuje vrijeme koje korisnik mora čekati da bi vidio sadržaj.

SSR poboljšava optimizaciju za tražilice (SEO). Budući da tražilice mogu direktno indeksirati generirani HTML, sadržaj mrežne stranice postaje vidljiviji i pristupačniji tražilicama, što može rezultirati boljim rangiranjem u rezultatima pretraživanja. Next.js pruža jednostavnu i intuitivnu implementaciju SSR-a.

Korištenjem posebnih metoda, kao što je *getServerSideProps*, Next.js omogućava programerima da definiraju koje podatke treba dohvatiti i kako generirati HTML na poslužitelju za svaku pojedinu stranicu. Ovo omogućava dinamičko dohvaćanje podataka iz različitih izvora, kao što su baze podataka ili vanjski API-ji, prije nego što se sadržaj prikaže korisniku. SSR također omogućava bolju sigurnost aplikacije. Budući da se većina logike obrade podataka odvija na poslužitelju, osjetljivim podacima se može sigurnije rukovati, smanjujući rizik od izlaganja klijentskoj strani. Osim toga, SSR smanjuje potrebu za slanjem velikih količina JavaScript koda klijentu, što može smanjiti površinu napada za potencijalne sigurnosne prijetnje. Jedna od ključnih funkcionalnosti SSR-a u Next.js je mogućnost kombiniranja s drugim tehnikama renderiranja, kao što su SSG i CSR. Ovo omogućava programerima da odaberu najprikladniju strategiju za svaku stranicu ili komponentu unutar njihove aplikacije, osiguravajući optimalnu izvedbu i korisničko iskustvo. Integracija ove tehnike s Next.js pruža fleksibilnost i moćne alate za dinamičko generiranje sadržaja, što rezultira boljim korisničkim iskustvom i povećanom vidljivošću na tražilicama.

3.2. Crisp

Crisp je platforma za neposrednu razmjenu poruka i podršku korisnicima, koja omogućava administratorima aplikacije pružanje brze i učinkovite podrške u stvarnom vremenu, putem različitih kanala komunikacije, uključujući mrežni chat, email, SMS i društvene mreže. Jedna od ključnih funkcionalnosti Crispa je integracija mrežnog chat sučelja na mrežne stranice aplikacija. Kroz ovo sučelje, posjetitelji mrežne stranice mogu postavljati pitanja, tražiti podršku ili ostavljati povratne informacije, dok podrška može odgovarati u stvarnom vremenu i pružiti relevantne informacije ili rješenja. Sama integracija Crisp skočnog prozora nudi bolje korisničko iskustvo u odnosu na standardni pristup preko kontakt forme ili prijave zahtjeva podršci.

3.3. Clerk autentifikacija

Za autentifikaciju i upravljanje korisnicima u aplikaciji korišten je Clerk, napredna platforma za autentifikaciju. Clerk pruža sveobuhvatno rješenje koje uključuje registraciju korisnika, prijavu, upravljanje sesijama, autentifikaciju s više faktora (engl. *Multi-Factor Authentication* - MFA). Ova platforma nudi jednostavnu i sigurnu integraciju korisničkih autentifikacijskih sustava, što poboljšava sigurnost i dobro korisničko iskustvo. Jedna od glavnih prednosti Clerka je jednostavnost integracije. Platforma dolazi s kolekcijom gotovih komponenti i API-ja koji omogućavaju brzu implementaciju autentifikacijskih funkcionalnosti bez potrebe za značajnim resursima. Clerk podržava različite metode autentifikacije, uključujući prijavu putem elektroničke pošte, telefona, društvenih mreža (*Googlea*, *Facebooka* i *GitHuba*) te prilagođene OAuth pružatelje usluga, što omogućava fleksibilnost i prilagodbu specifičnim potrebama korisnika.

Sigurnost je ključna komponenta Clerka. Platforma koristi napredne sigurnosne protokole kako bi osigurala zaštitu korisničkih podataka. Clerk implementira enkripciju podataka tijekom prijenosa, podržava autentifikaciju s više faktora (MFA) kako bi se smanjio rizik od neovlaštenog pristupa, te pruža alate za otkrivanje i prevenciju prijevara. Redovita sigurnosna ažuriranja i usklađenost s najnovijim sigurnosnim standardima dodatno osiguravaju visoku razinu zaštite.

Clerk također nudi odlične alate za upravljanje korisnicima. Administracijska nadzorna ploča omogućuje pregled i upravljanje korisničkim računima, uvid u aktivnosti korisnika te konfiguraciju postavki autentifikacije. Platforma također podržava prilagodljive e-mail i SMS predloške (engl. *template*) za komunikaciju s korisnicima, čime se olakšava održavanje. Jedna od značajnih prednosti Clerka je njegova skalabilnost. Bez obzira na to koliko korisnika aplikacija ima, Clerk je dizajniran da podržava visok promet i veliki broj zahtjeva, bez rizika za performanse. To omogućava rast i razvoj aplikacije bez brige o kapacitetu sustava za autentifikaciju.

Korištenje Clerka u aplikaciji osigurava pouzdan i siguran sustav za autentifikaciju korisnika. Njegove napredne funkcionalnosti, jednostavna integracija i visoka razina sigurnosti čine ga idealnim rješenjem za moderne mrežne stranice i mobilne aplikacije. Integracija Clerka omogućava fokusiranje na razvoj ključnih funkcionalnosti aplikacije, dok Clerk preuzima brigu o složenim aspektima korisničke autentifikacije i upravljanja sesijama.

3.4. MySql

Za bazu podataka korišten je MySQL. MySQL je poznat po svojoj pouzdanosti, performansama i skalabilnosti, što ga čini popularnim izborom za široki spektar aplikacija. Njegova sposobnost upravljanja relacijskim podacima omogućava strukturiranje podataka na siguran i efikasan način. MySQL je baza podataka otvorenog koda, široko korištena u industriji zbog svoje stabilnosti i fleksibilnosti. Nudi podršku za standardne SQL upite (engl. *query*), što omogućava programerima da lako rade s podacima koristeći dobro poznate metode i sintaksu. MySQL podržava ACID transakcije (kratica od engl. *Atomicity, Consistency, Isolation, Durability),* što osigurava integritet i točnost podataka. Optimizirani algoritmi za indeksiranje i pretraživanje podataka omogućavaju brzu obradu velikih količina podataka, čime se poboljšava ukupna učinkovitost aplikacije. MySQL također podržava različite mehanizme za pohranu podataka, poput InnoDB i MyISAM, što omogućava prilagodbu performansi i funkcionalnosti specifičnim potrebama projekta.

MySQL je visoko skalabilan, što znači da može rasti s poslovanjem. Bez obzira na to upravlja li se malom mrežnom aplikacijom ili velikim sustavom s tisućama korisnika, MySQL može učinkovito rukovati povećanom količinom podataka i povećanim brojem transakcija. Podržava replikaciju podataka, što omogućava stvaranje kopija baze podataka radi poboljšanja dostupnosti i raspodjele opterećenja. Sigurnost je još jedan važan aspekt MySQL-a. Nudi robusne sigurnosne funkcionalnosti, uključujući autentifikaciju korisnika, upravljanje pristupom i enkripciju podataka, što osigurava zaštitu osjetljivih podataka i sprječava neovlašteni pristup. Redovita ažuriranja i sigurnosne zakrpe (engl. *patch*) pomažu u održavanju visoke razine zaštite protiv potencijalnih prijetnji.

MySQL također nudi odličnu podršku i dokumentaciju, što olakšava instalaciju, konfiguraciju i održavanje baze podataka. Aktivna zajednica korisnika i programera pruža dodatne resurse, alate i savjete, koji mogu biti izuzetno korisni za rješavanje problema i optimizaciju performansi baze podataka. Integracija ove relacijske baze podataka osigurava stabilno i učinkovito rukovanje podacima, što je ključno za pružanje visokokvalitetnog korisničkog iskustva.

3.5. Prisma

Prisma je odabrana kao ORM alat koji olakšava rad s bazom podataka u aplikaciji. Ovaj alat pruža jednostavno i intuitivno sučelje za povezivanje i upravljanje bazom podataka, čime se pojednostavljuje manipulacija podacima i smanjuje potreba za ručnim SQL upitima. Prisma

omogućava programerima da rade s bazom podataka, koristeći objekte i metode unutar svog koda, što povećava produktivnost i smanjuje mogućnost pogrešaka.

Jedna od glavnih prednosti Prisme je automatsko generiranje tipova i API-ja na temelju sheme baze podataka. Ovo omogućava statičko određivanje tipa podataka i automatsku provjeru sintakse, čime se osigurava veća sigurnost i stabilnost koda. Korištenjem Prisme, programeri mogu definirati modele podataka u datotekama sheme, koje se zatim koriste za generiranje odgovarajućih klasa i metoda za interakciju s bazom podataka.

Prisma također podržava migracije baza podataka, omogućujući jednostavno praćenje i primjenu promjena u strukturi baze podataka tijekom razvoja. Alat automatski generira i izvršava migracijske skripte na temelju promjena u shemi, čime se olakšava upravljanje verzijama baze podataka i osigurava konzistentnost podataka kroz različita okruženja.

Prisma nudi podršku za različite baze podataka, uključujući PostgreSQL, MySQL, SQLite i SQL Server, što omogućava fleksibilnost u izboru baze podataka ovisno o specifičnim potrebama projekta. Alat se također lako integrira s popularnim JavaScript frameworkom i knjižnicama kao što su Express, Apollo i Next.js, omogućavajući besprijekornu integraciju u postojeće aplikacije. Osim toga, Prisma poboljšava performanse aplikacija kroz efikasno generiranje SQL upita. Optimizirani upiti smanjuju vrijeme izvođenja i resurse potrebne za obradu podataka, što rezultira bržim i responzivnijim aplikacijama. Prisma također podržava napredne funkcionalnosti, kao što su relacije između tablica, agregacijske funkcije i transakcije, pružajući sveobuhvatno rješenje za upravljanje podacima.

Integracija Prisme u aplikaciju omogućava učinkovit i intuitivan rad s bazom podataka, poboljšavajući produktivnost i smanjujući mogućnost pogrešaka. Svojom fleksibilnošću, podrškom za migracije i naprednim mogućnostima, Prisma predstavlja idealan izbor za moderni razvoj aplikacija koji zahtijeva robustan i pouzdan ORM alat.

3.6. Shadcn/UI

Za implementaciju korisničkog sučelja u aplikaciji odabran je Shadcn/UI, napredni UI framework dizajniran za izradu responzivnih i estetski privlačnih mrežnih sučelja. Shadcn/UI omogućava razvoj modernih, funkcionalnih i prilagodljivih sučelja, čime se poboljšava korisničko iskustvo i povećava zadovoljstvo korisnika. Shadcn/UI dolazi s bogatom kolekcijom gotovih komponenti koje se lako mogu prilagoditi specifičnim potrebama projekta. Komponente uključuju razne elemente sučelja, kao što su tipke, obrasci, navigacijske trake,

moduli, čime se ubrzava razvoj i smanjuje potreba za pisanjem vlastitog CSS-a ili HTML-a. Svaka komponenta je pažljivo dizajnirana kako bi bila vizualno privlačna i funkcionalna, osiguravajući dosljednost u cijelom sučelju.

Jedna od ključnih prednosti korištenja Shadcn/UI je njegova prilagodljivost i fleksibilnost. Framework omogućava jednostavnu integraciju s drugim JavaScript bibliotekama i alatima, kao što su React, Vue i Angular, čime se olakšava integracija u postojeće projekte. Shadcn/UI podržava prilagođavanje stilova putem CSS-in-JS rješenja, omogućavajući programerima da brzo i efikasno prilagode izgled komponenti specifičnim zahtjevima. Shadcn/UI je optimiziran za performanse, što znači da su sve komponente lagane i brze, čime se osigurava da aplikacija ostane responzivna čak i pri složenim korisničkim interakcijama. Također podržava responzivni dizajn, osiguravajući da sučelje izgleda i funkcionira dobro na različitim uređajima i veličinama ekrana (engl. *media query*), što je ključno za pružanje konzistentnog korisničkog iskustva na mobilnim i desktop platformama. Framework također uključuje ugrađenu podršku za pristupačnost (engl. *accessibility*), osiguravajući da su sve komponente sučelja usklađene s najboljim praksama za pristupačnost. Ovo uključuje podršku za navigaciju tipkovnicom, čitače ekrana (*screen reader*) i druge tehnologije koje pomažu korisnicima s invaliditetom. Pružanjem pristupačnih komponenti, Shadcn/UI omogućava razvoj aplikacija koje su dostupne širokom spektru korisnika.

Korištenje Shadcn/UI u aplikaciji omogućava brzu i efikasnu izradu pristupačnog i prilagodljivog visokokvalitetnog korisničkog sučelja s optimiziranim performansama. Integracijom ovog frameworka, programer se može fokusirati na izradu funkcionalnosti i poboljšanje korisničkog iskustva, jer se oslanja na provjerene i dobro dizajnirane UI komponente Shadcn/UI za vizualni aspekt aplikacije.

Isto tako važno je napomenuti da korištenjem Shadcn/UI nije neophodno implementirati sve komponente već samo one koje su potrebne prilikom izrade aplikacije, što pruža dodatnu optimizaciju aplikacije.

3.7. Axios

Axios je biblioteka za upravljanje HTTP zahtjevima koja se koristi za komunikaciju s vanjskim API-ima. Njena podrška za asinkrono izvršavanje, jednostavnost korištenja i fleksibilnost čine ju pouzdanim alatom za izvršavanje HTTP zahtjeva i obradu odgovora u aplikaciji. Axios je

izgrađen na temelju obećanja (engl. *Promise*), što omogućava lako rukovanje asinkronim operacijama i čini kod čitljivijim i jednostavnijim za održavanje.

Jedna od ključnih prednosti korištenja Axiosa je njegova sposobnost da pojednostavi i optimizira rad s HTTP zahtjevima. Podržava sve glavne HTTP metode kao što su GET, POST, PUT, DELETE, pružajući programerima laku integraciju različitih funkcionalnosti u aplikacije. Axios automatski obrađuje JSON podatke, čime se eliminira potreba za ručnim parsiranjem ili formatiranjem podataka, što dodatno pojednostavljuje rad s API-jem.

Axios također nudi napredne mogućnosti konfiguracije. Omogućava zadavanje globalnih postavki, kao što su osnovni URL, zaglavlja (engl. *headers*) i vrijeme isteka (engl. *timeout*), što omogućava standardizaciju zahtjeva kroz cijelu aplikaciju. Axios podržava presretače (engl. *interceptors*), koji omogućavaju presretanje i modifikaciju zahtjeva ili odgovora prije nego što budu obrađeni, što je korisno za rukovanje autentifikacijom, bilježenjem ili upravljanjem pogreškama. Jedan od primjera bi bio presretač na odgovoru poslužitelja prilikom isteklog JWT-a (kratica od engl. *JSON Web Token*) ili refresh tokena. U tom slučaju bi presretač spremio zahtjev, pokušao dohvatiti novi token te, ovisno o tome je li dobio novi token ili ne, odradio zadanu funkcionalnost.

Jedna od funkcionalnosti koje Axios čini posebno korisnim je njegova podrška za otkazivanje zahtjeva. Ovo je ključno u situacijama gdje je potrebno prekinuti zahtjev prije nego što bude završen, što može poboljšati performanse aplikacije i korisničko iskustvo. Axios može automatski ponovno poslati zahtjeve u slučaju privremenih pogrešaka, čime se povećava pouzdanost komunikacije s vanjskim API-ima. Axios je kompatibilan s Node.js, što ga čini pogodnim za korištenje i na strani poslužitelja i na strani klijenta. Ova svestranost omogućava razvoj jedinstvenih rješenja koja koriste isti alat za komunikaciju s API-ima, bez obzira na okruženje u kojem se nalaze. Osim toga, Axios se lako integrira s modernim JavaScript/TypeScript aplikacijama kao što su React, Vue i Angular, pružajući besprijekorno iskustvo razvoja frontend aplikacija.

Integracija Axiosa u aplikaciju osigurava učinkovit i pouzdan način upravljanja HTTP zahtjevima. Njegova fleksibilnost, jednostavnost korištenja i napredne mogućnosti konfiguracije čine ga idealnim izborom za rukovanje (engl. *handshake*) komunikacijom s vanjskim API-ima, poboljšavajući performanse i funkcionalnost aplikacije.

3.8. Stripe

Za integraciju sustava plaćanja odabran je Stripe, popularna platforma za online plaćanja. Stripe pruža siguran i jednostavan način za prihvaćanje plaćanja putem interneta, pružajući korisnicima praktično iskustvo prilikom obavljanja transakcija. Stripe je poznat po svojoj pouzdanosti i visokoj razini sigurnosti, što ga čini idealnim izborom za poslovanje koje se oslanja na online transakcije.

Stripe omogućava obradu različitih načina plaćanja, uključujući kreditne i debitne kartice, mobilna plaćanja i bankovne transfere, čime se zadovoljavaju potrebe širokog spektra korisnika. Integracija Stripe API-ja u aplikaciju omogućava jednostavno postavljanje i upravljanje plaćanjima, a korisničko sučelje je intuitivno i prilagođeno korisnicima svih razina tehničke vještine. Stripe također podržava različite valute, što je posebno korisno za poslovanja koja posluju na globalnom tržištu.

Jedna od ključnih prednosti Stripea je njegova fleksibilnost i prilagodljivost. Platforma nudi širok raspon alata i funkcionalnosti koji omogućavaju prilagodbu sustava plaćanja specifičnim potrebama poslovanja. Na primjer, Stripe omogućava postavljanje ponavljajućih plaćanja za pretplatničke modele, integraciju s e-commerce platformama, te podršku za napredne analitičke alate koji pomažu u praćenju i optimizaciji poslovnih performansi. Stripe je također poznat po svojoj robusnoj sigurnosnoj infrastrukturi. Sve transakcije obrađene putem Stripea su šifrirane i podložne strogim sigurnosnim protokolima, što osigurava zaštitu korisničkih podataka.

Dodatna prednost Stripea je njegova podrška za programere. Platforma dolazi s opsežnom dokumentacijom, API referencama i primjerima koda, što olakšava integraciju i prilagodbu sustava u bilo koju aplikaciju. Stripe također nudi nadogradnje i poboljšanja svojih usluga, osiguravajući da korisnici uvijek imaju pristup najnovijim tehnologijama i funkcionalnostima.

3.9. Replicate AI API

Integracija Replicate AI API-ja je odabrana radi korištenja naprednih modela za generiranje video i audio sadržaja, koristeći algoritme umjetne inteligencije kako bi se postigla visoka kvaliteta i preciznost. Replicate AI API omogućava stvaranje prilagođenog video i audio sadržaja kroz automatizirane procese, što značajno skraćuje vrijeme potrebno za produkciju i smanjuje troškove. API također podržava različite formate i rezolucije. Jedna od ključnih prednosti Replicate AI API-ja je njegova sposobnost učenja i prilagodbe različitim stilovima i temama.

Kroz napredne algoritme strojnog učenja, API može prepoznati i generirati sadržaj koji je konzistentan s odabranim stilom ili temom te osigurava visoku kvalitetu. Ova funkcionalnost je posebno korisna za marketinške, edukacijske i druge kampanje gdje je personalizacija sadržaja ključna. Replicate AI API također omogućava integraciju s drugim alatima i platformama, čime se postiže veća funkcionalnost. API podržava obradu u stvarnom vremenu, što omogućava brze prilagodbe na promjene. Osim toga, Replicate AI API dolazi s ugrađenim sigurnosnim mjerama za zaštitu korisničkih podataka i privatnosti. Svi podaci obrađeni putem API-ja su hashirani i pohranjeni na siguran način, čime se minimiziraju rizici od zloupotrebe ili neovlaštenog pristupa.

3.10. ChatGPT API

Za glavne interakcije unutar aplikacije korišten je ChatGPT API. Ova integracija omogućuje implementaciju naprednih modela za stvaranje interaktivnih instanci za razgovor. ChatGPT API, kojeg je razvio OpenAI, koristi napredne algoritme strojnog učenja (engl. *Machine Learning*) kako bi razumio i generirao ljudski jezik s visokom razinom točnosti. Ovaj API temelji se na GPT-4 arhitekturi koja je lako promjenjiva na bilo koji drugi model s izmjenom konfiguracije (jedne linije koda). Integracija ChatGPT API-ja omogućava aplikacijama da obavljaju različite zadatke, poput odgovaranja na korisnička pitanja, generiranje koda za razvoj aplikacija te generiranja sadržaja na temelju korisničkih unosa. API se može lako integrirati u različite platforme, uključujući mrežne stranice, mobilne aplikacije i sustave za upravljanje korisnicima, čime se postiže veća fleksibilnost i dostupnost. Pored toga, koristi se za analizu podataka, automatizaciju procesa i poboljšanje interaktivnosti aplikacija.

Jedna od ključnih prednosti ChatGPT modela je njegova sposobnost učenja iz velikih količina podataka, što mu omogućava pružanje preciznih i relevantnih odgovora na razna pitanja. Također, može se prilagoditi specifičnim potrebama korisnika kroz prilagodbu modela, što omogućava optimizaciju performansi za specifične slučajeve uporabe (Kod 1). Jedan od primjera koji se koristi unutar projekta je generiranje koda kroz specifične naredbe modelu. Kroz kontinuirano učenje i prilagodbu, ChatGPT postaje sve učinkovitiji u prepoznavanju konteksta i namjere korisnika, čime se povećava kvaliteta interakcije.

```
"You are a code generator. You must answer in markdown code snippets and must be able to explain how the code works. You may also use code comments for explanations. If possible, provide installation tips for what is needed in the code.“
```

Kod 1. Primjer konfiguracije Chat GPT modela

Korištenje ChatGPT API-ja također donosi značajne prednosti u smislu učinkovitosti i uštede resursa. Automatizacijom brojnih zadataka koji bi inače zahtijevali ljudsku intervenciju, mogu se smanjiti troškovi i povećati produktivnost. Dodatno, API pruža skalabilna rješenja koja se mogu prilagoditi rastućim potrebama poduzeća, bez kompromitiranja kvalitete usluge.

Na kraju, sigurnosne mjere ugrađene u ChatGPT API osiguravaju da su podaci korisnika zaštićeni, a komunikacija sigurna. OpenAI redovito ažurira sigurnosne protokole kako bi zaštitili privatnost korisnika i spriječili zloupotrebu tehnologije.

4. PRAKTIČNI RAD – RAZVOJ PLATFORME S USLUGAMA UMJETNE INTELIGENCIJE

U ovom poglavlju opisuje se proces razvoja platforme koja koristi usluge umjetne inteligencije za unapređenje poslovnih procesa. Fokus će biti na strukturi projekta, implementaciji Chat GPT API-ja i Replicate AI API-ja, kao i drugih QOL (kratica od engl. *Quality of Life*) alata poput Clerka, Crispa i Shadcn/UI-ja. Platforma je osmišljena s ciljem optimizacije radnih tokova, poboljšanja korisničkog iskustva i povećanja učinkovitosti.

4.1. Struktura projekta

Struktura projekta organizirana je tako da omogućuje preglednost, lakoću razvoja i osigurava standardnu strukturu projekta. Projekt koristi modularni pristup, gdje su različiti dijelovi aplikacije smješteni u zasebne direktorije. Ovaj pristup doprinosi preglednosti koda i olakšava rad na pojedinim dijelovima aplikacije.

Projekt se sastoji od sljedećih direktorija:

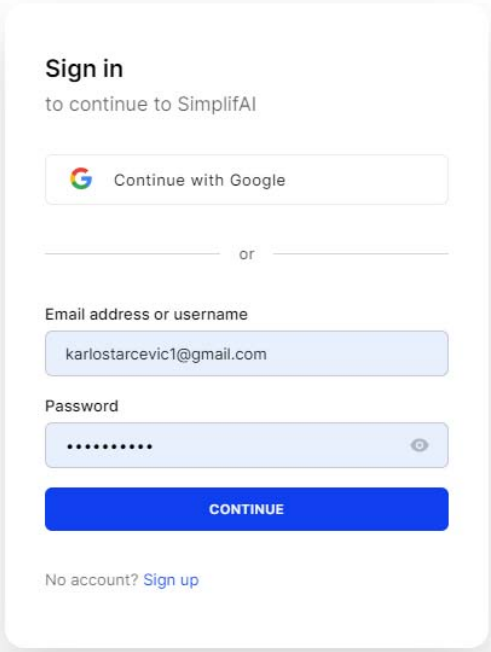
1. app/(auth): Ovaj direktorij sadrži sve datoteke i komponente potrebne za autentifikaciju korisnika. Korištenje Clerka za autentifikaciju omogućuje siguran i jednostavan pristup korisnicima.
2. app/(dashboard): U ovom direktoriju nalaze se komponente i stranice vezane za korisničko sučelje nakon prijave.
3. app/(landing): Direktorij koji uključuje početne stranice aplikacije.
4. app/api: U ovom direktoriju smještene su sve Next.js rute koje koriste Axios za slanje HTTP zahtjeva. Ove rute omogućuju komunikaciju između frontenda i backend sustava te koriste biblioteke kao što su ChatGPT i Replicate.
5. app/components: Ovdje su smještene sve prilagođene komponente bazirane na kolekciji Shadcn/UI.
6. app/prisma: Ovaj direktorij sadrži konfiguracijske datoteke i sheme za Prisma ORM.
7. app/hooks: U ovom direktoriju se nalaze prilagođeni React hookovi koji se koriste za upravljanje stanjem i logikom unutar aplikacije. Hookovi omogućavaju korištenje značajki u funkcionalnim komponentama.
8. app/layout.tsx: U ovoj datoteci se nalazi glavni format rasporeda aplikacije koji služi kao skica za dinamičku promjenu dijelova aplikacije.

9. /.env: Datoteka unutar koje se nalaze API ključevi, tajni stringovi (engl. *secret*), stringovi za povezivanje s bazom. Ova datoteka se uvijek dodaje na .gitignore popis da ključevi i tajne ne bi bili javno dostupni.
10. /constants.ts: Datoteka sa svim konstantama aplikacije. Ovakve datoteke su izvrsne za čuvanje statičkih varijabli.

Ovakva struktura projekta omogućuje jasnu organizaciju i razdvajanje koda. Svaki direktorij i datoteka imaju jasno definiranu ulogu, što doprinosi lakšem razvoju, održavanju i nadogradnji aplikacije.

4.2. Autentifikacija i autorizacija

Autentifikacija i autorizacija su ključni dijelovi svake aplikacije. Za implementiranje sustava autentifikacije korišten je Clerk auth (Slika 3).



Slika 3. Primjer Clerk login stranice

Clerk nudi brojne funkcionalnosti kao što su sigurna prijava i registracija, dvofaktorska autentifikacija te upravljanje sesijama. Integracija Clerka u aplikaciju prilično je jednostavna zahvaljujući dokumentaciji koja se nalazi na stranici Clerka. Clerk implementacija je jedna od najjednostavnijih implementacija autentifikacije i autorizacije, a razlog je taj što sami Clerk odrađuje svu logiku prijavljivanja, odjavljivanja, rola i tako dalje. Koraci implementacije Clerka su:

1. registracija na Clerk stranici
2. kreiranje novog projekta na Clerk stranici
3. instalacija Clerk SDK-a (engl. *Source Development Kit*) dostupnog u npm paketu: npm install @clerk/clerk-sdk-node
4. dodavanje API ključeva u .env datoteku unutar projekta
5. postavljanje Clerk middlewarea (Kod 2)
6. postavljanje Clerk Frontend SDK-a na klijentskoj strani kako bi se omogućila prijava, odjava te uređivanje korisničkih profila.

```
export default function RootLayout({
      children,
}: Readonly<{
      children: React.ReactNode;
}>) {
      return (
            <ClerkProvider> //ClerkProvider se “wrappa” oko tijela
      aplikacije
                  <html lang="en">
                        <CrispProvider />
                        <body className={inter.className}>
                              <ToasterProvider />
                              <ModalProvider />
                              {children}
                        </body>
                  </html>
            </ClerkProvider>
      );
}
```

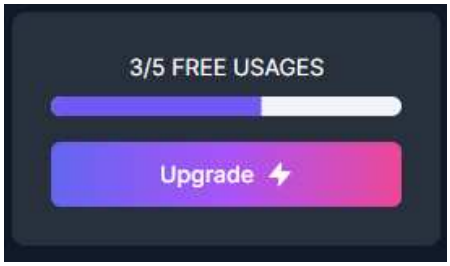
Kod 2. Clerk Provider Wrapper

Ovaj pristup pruža jednostavnu logiku autentifikacije i autorizacije. Clerk se brine o svim aspektima sigurnosti, što ostavlja više vremena za ostale dijelove projekta.

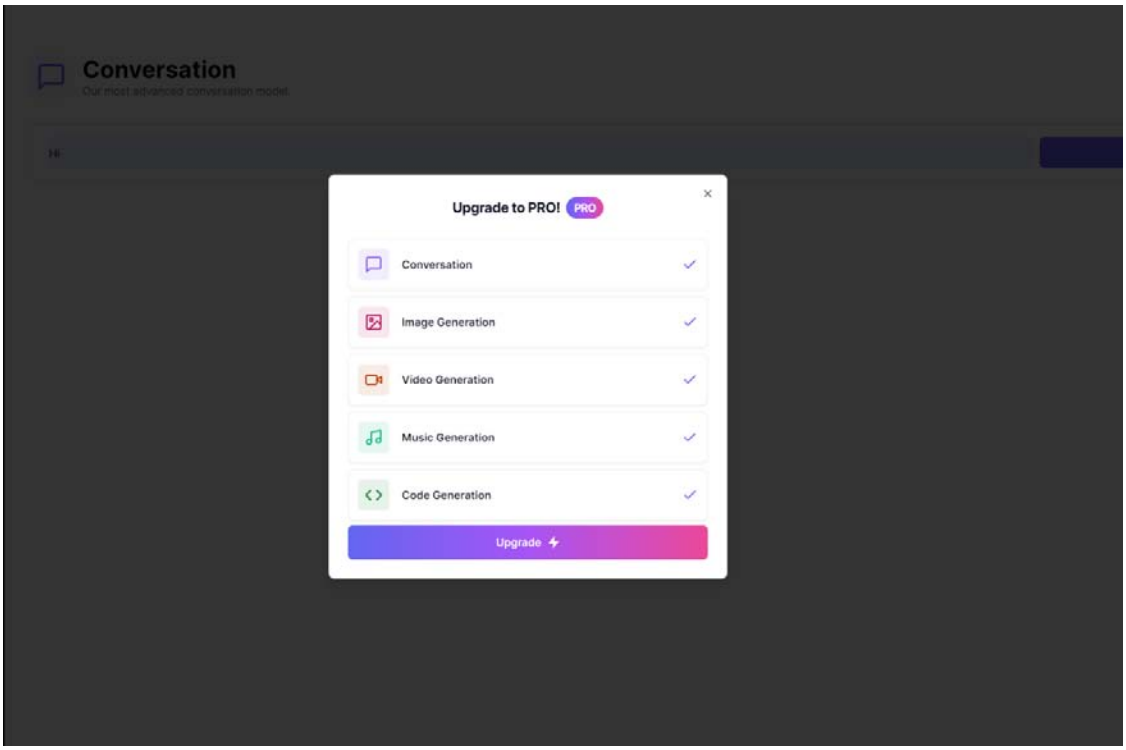
4.3. Postavljanje limita besplatnog korištenja

Unutar aplikacije je moguće postaviti ograničenje na broj kredita za korisnike koji prvi put koriste aplikaciju (Slika 4). Nakon što korisnik potroši kredite (Slika 5) preusmjerava se na Stripe proces plaćanja za neograničenu uporabu aplikacije (Slika 6).

Slika 4. Prisma Model za vođenje evidencije korištenja kredita



Slika 5. Primjer prikaza kredita s klijentske strane



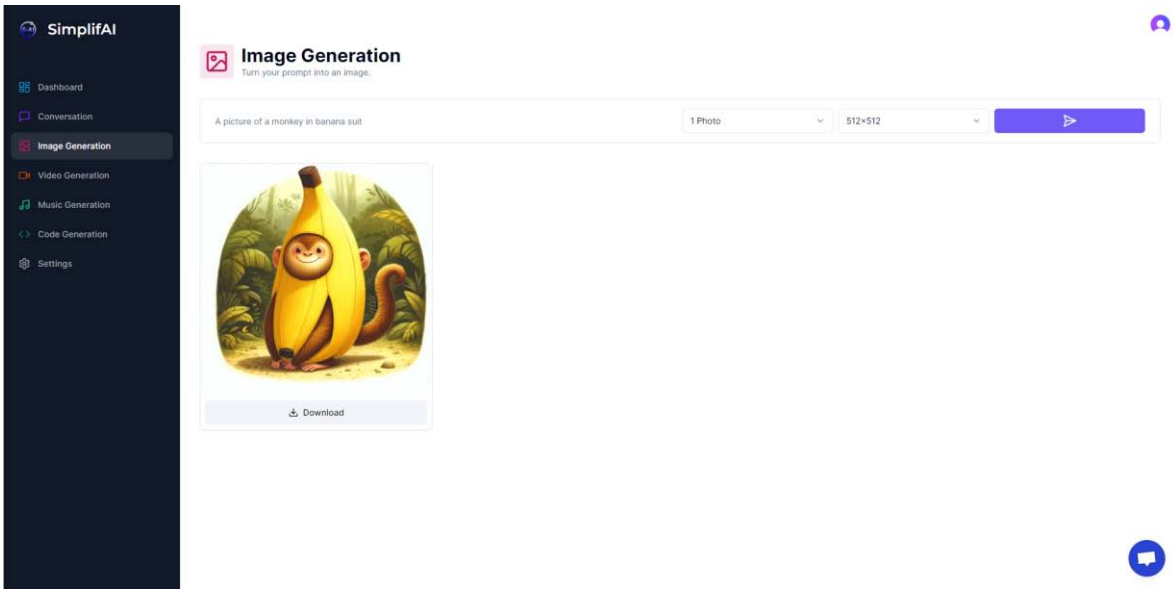
Slika 6. Primjer potrošenih kredita te ponuda za nadogradnju

4.4. Integracija vanjskih API-ja

Integracija vanjskih API-ja ključna je za proširenje funkcionalnosti aplikacije. U ovom slučaju, koriste se dva AI API-ja: ChatGPT i Replicate AI. ChatGPT se u aplikaciji koristi za generiranje odgovora i koda na upite korisnika.

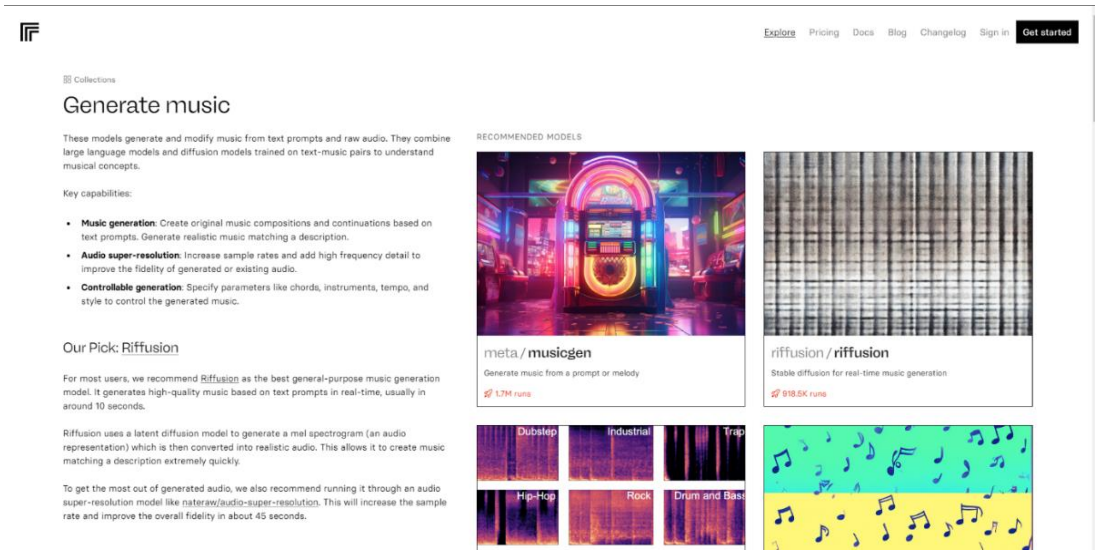
Koraci integracije ChatGPT-a su:

1. registracija na OpenAI i kreiranje API ključa
2. instalacija OpenAI SDK-a pomoću npm paketa: npm install openai
3. postavljanje API ključa u .env datoteku projekta
4. kreiranje servisnog sloja koji šalje upit na ChatGPT API i vraća generirani odgovor (Slika 7).

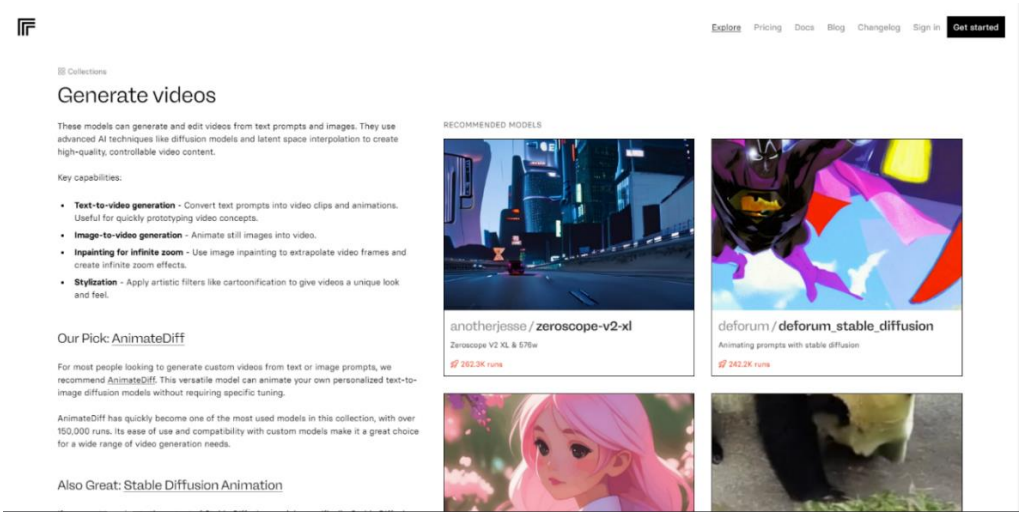


Slika 7. Primjer generiranja slike (Dall-E 3 model)

Replicate AI je platforma koja omogućuje treniranje i implementaciju vlastitih modela zvučnih (Slika 8), video (Slika 9) i ostalih zapisa.

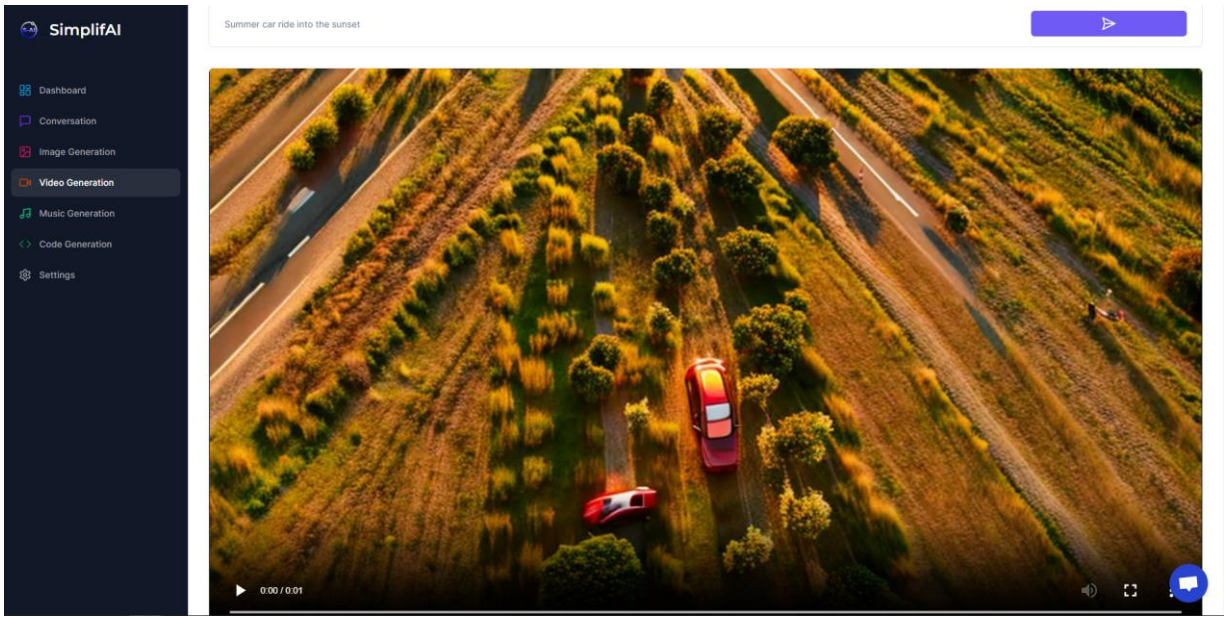


Slika 8. Replicate AI modeli za generiranje zvučnog zapisa



Slika 9. Replicate AI modeli za generiranje video zapisa

Za generiranje zvučnog zapisa (Slika 11) i video zapisa (Slika 10) koristi se Replicate AI model. Integracija Replicate AI-a slična je integraciji Chat GPT-a u kojoj se kreirao API ključ te implementirao u .env datoteku i kreirao servise potrebne za komunikaciju.



Slika 10. Primjer generiranja video zapisa (Replicate AI)

Slika 11. Primjer generiranja zvučnog zapisa (Replicate AI)

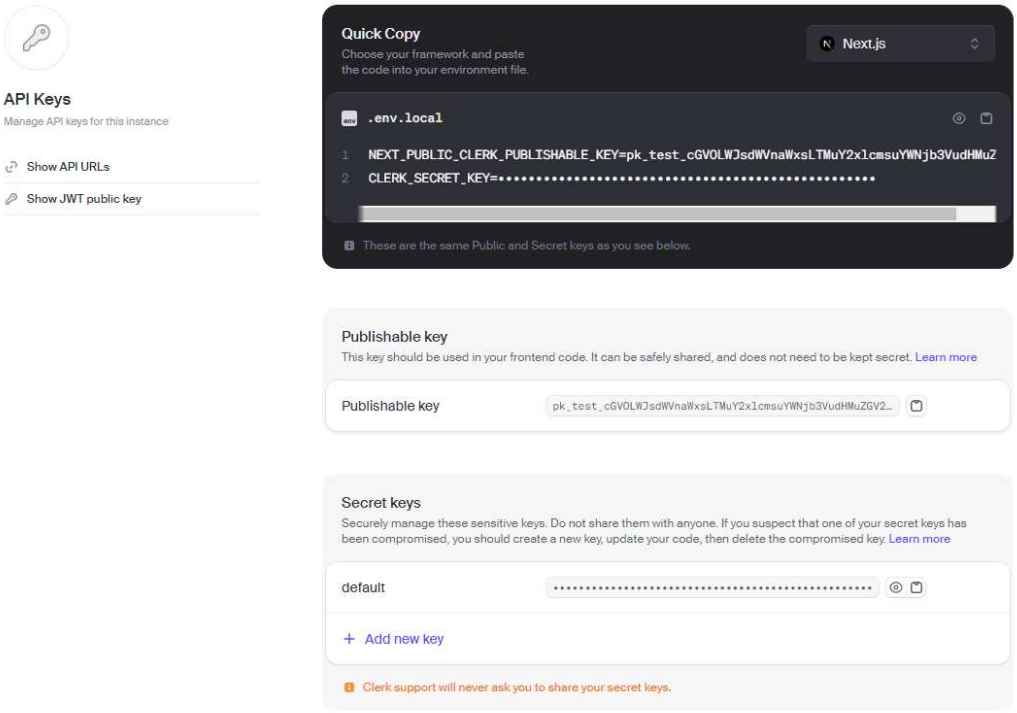
4.5. Uspostava plaćanja putem Stripea

Stripe je vodeća platforma za online plaćanja koja omogućuje brzo i sigurno plaćanje na mrežnim stranicama. Stripe je odabran zbog popularnosti, ažurnosti te sigurnosnih standarda.

Koraci za integraciju Stripea su:

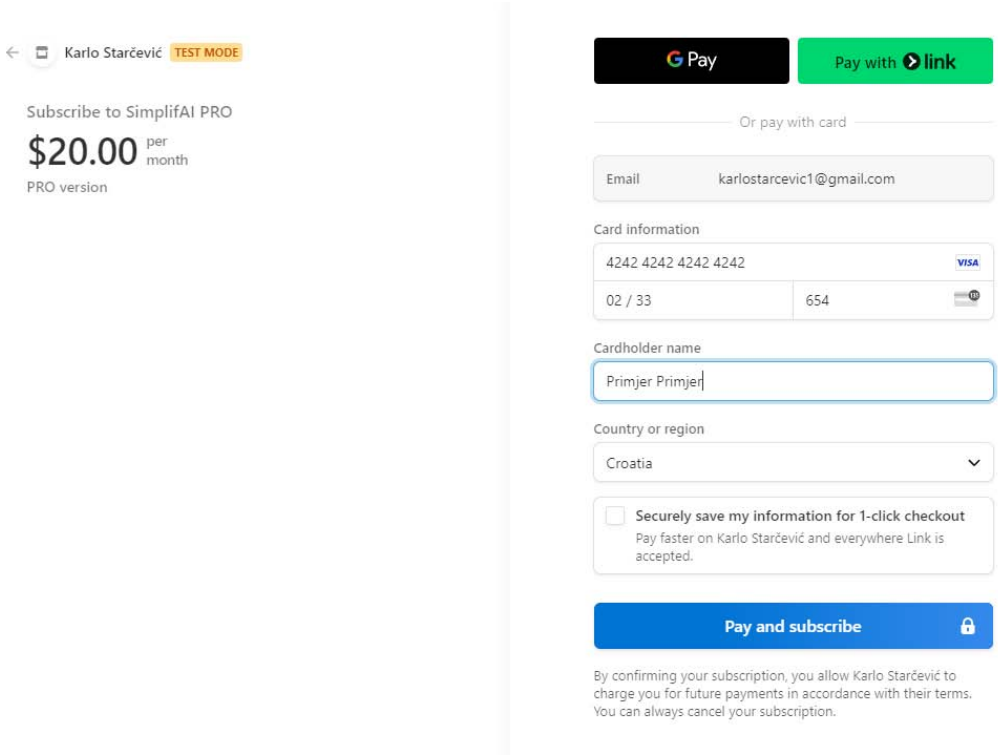
1. registracija i postavljanje stripe računa
2. instalacija Stripe SDK-a koristeći npm: npm install stripe
3. dodavanje Stripe API ključeva u .evn datoteku (Slika 12)
4. kreiranje plana pretplate na Stripe nadzornoj ploči
5. implementacija logike unutar Next.js aplikacije za komunikaciju s MySql bazom podataka za održavanje limita kredita pretplate
6. postavljanje Stripe Checkouta za prikupljanje podataka o plaćanju i vremenu isteka pretplate.

Ova implementacija je omogućila model pretplate za korisnike na temelju mjesečnih kredita.

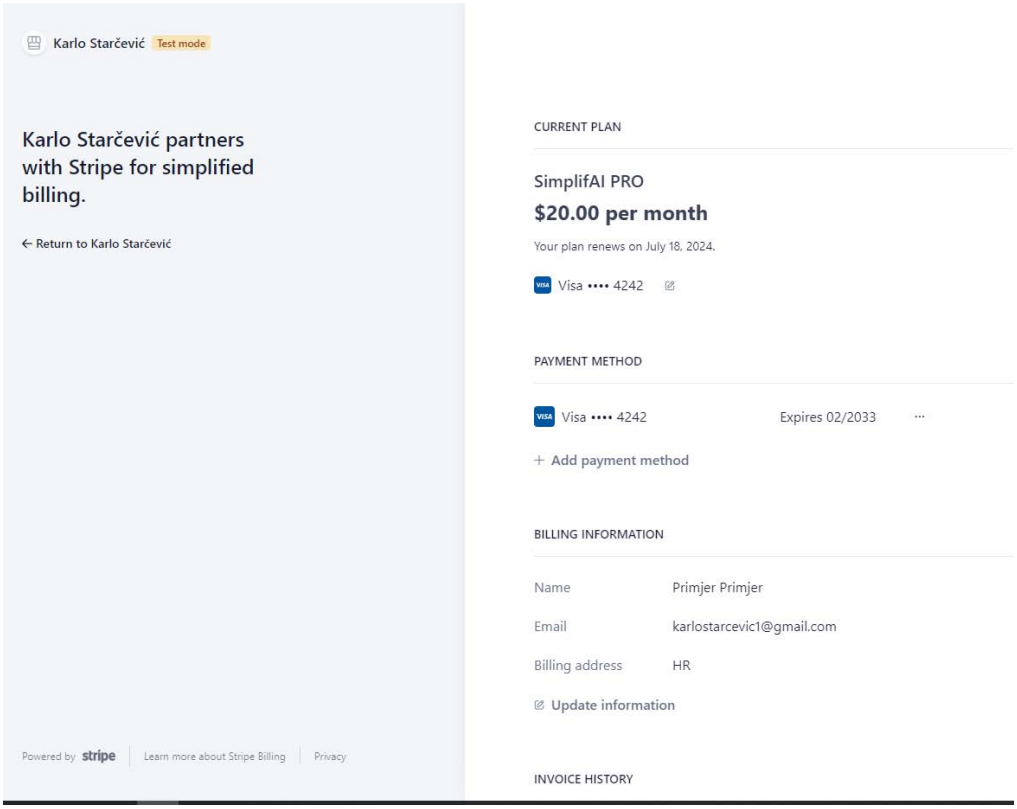


Slika 12. Generiranje Stripe API ključa

Integracijom Stripea unutar aplikacije korisnicima je omogućeno upravljanje računima (Slika 14) te pretplatom na samu aplikaciju (Slika 13**Error! Reference source not found.**).



Slika 13. Primjer kupovine pretplate



Slika 14. Pretplate korisnika te povijest pretplata

Implementacijom plaćanja korisnicima je omogućeno korištenje resursa bez ograničenja.

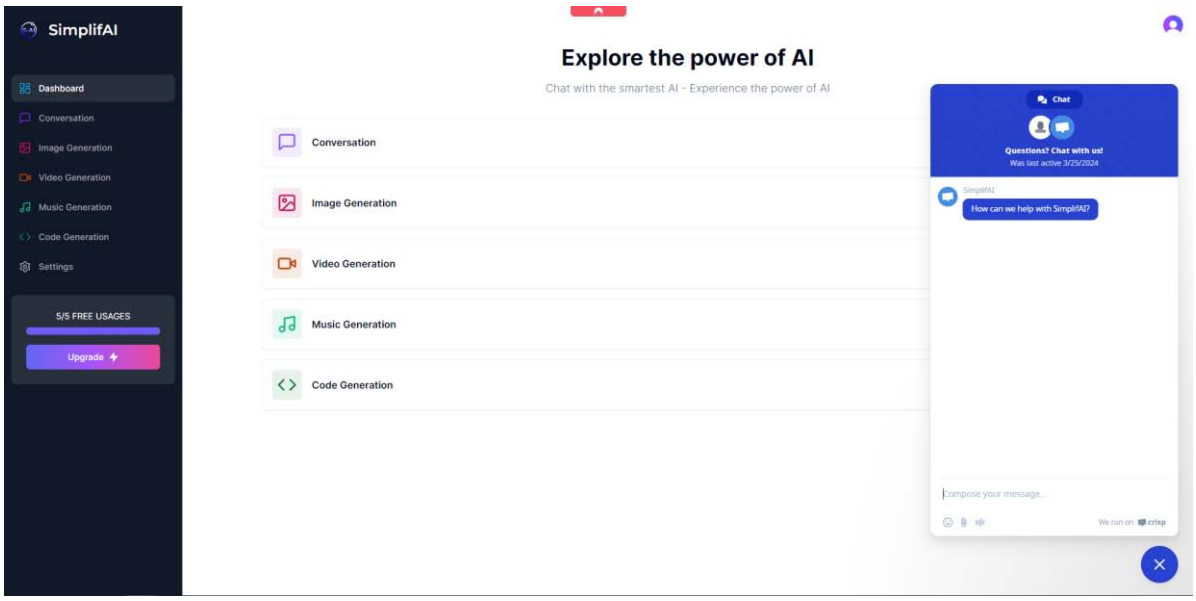
4.6. Implementacija funkcionalnosti razgovora u stvarnom vremenu pomoću Crispa

Crisp je platforma za korisničku podršku koja omogućuje implementaciju chat funkcionalnosti na mrežnim stranicama u obliku skočnog prozora. Unutar ove aplikacije korišten je za pružanje izravne podrške korisnicima u slučaju poteškoća s korištenjem aplikacije.

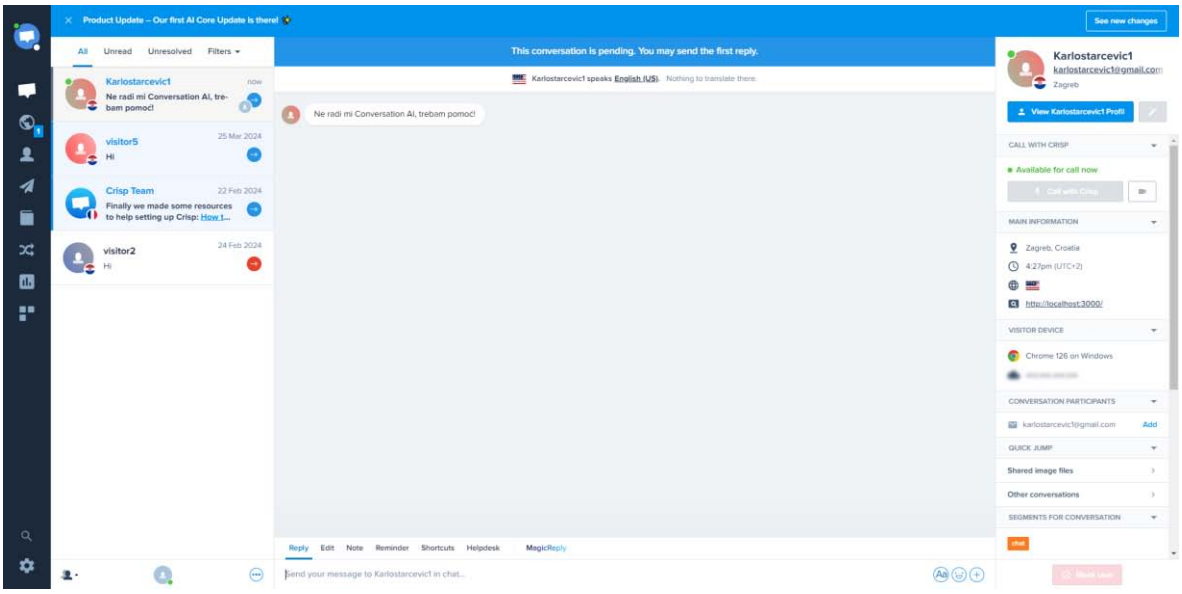
Koraci integracije Crisp su:

1. registracija na Crisp stranicu
2. implementacija Crisp poslužitelja na samom vrhu slojeva aplikacije, kako bi se prozor za podršku prikazivao na svim stranicama
3. postavljanje automatiziranih odgovora i linkova na često postavljana pitanja.

Ova implementacija omogućuje korisnicima dobivanje brzih odgovora ljudske podrške (Slika 15). Crisp nadzorna ploča (engl. *dashboard*) korištena je za praćenje i odgovaranje na poruke u stvarnom vremenu (Slika 16).



Slika 15. Crisp Chat – Primjer Crisp prozora unutar aplikacije



Slika 16. Crisp nadzorna ploča

4.7. Korištenje Prisma objektno-relacijskog mapiranja (ORM) za upravljanje bazom podataka

Prisma je moderni ORM (engl. *Object-Relational Mapping*) alat koji olakšava rad s bazama podataka u Node.js i TypeScript aplikacijama.

Koraci implementacije Prisma ORM-a su:

1. instalacija Prisma CLI koristeći npm: npm install prisma -D
2. inicijalizacija novog Prisma projekta koristeći: npx prisma init

3. konfiguracija Prisme – povezivanje s bazom podataka tj. dodavanje database stringa u .env datoteku
4. kreiranje Prisma sheme (prisma/schema.prisma) koja definira modele baze podataka (Kod 2)
5. generiranje Prisma klijenta koristeći: npx prisma generate. Ovo omogućuje interakciju s bazom podataka koristeći generirane funkcije.

Prisma ORM omogućuje jednostavno upravljanje bazom podataka, uključujući stvaranje, čitanje, ažuriranje i brisanje (engl. *CRUD – Create, Read, Update, Delete*) te migracije baze podataka.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider     = "mysql"
  url          = env("DATABASE_URL")
  relationMode = "prisma"
}

model UserApiLimit {
  id        String   @id @default(cuid())
  userId    String   @unique
  count     Int      @default(0)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}

model UserSubscription {
  id                     String    @id @default(cuid())
  userId                 String    @unique
  stripeCustomerId       String?   @unique @map(name:
"stripe_customer_id")
  stripeSubscriptionId   String?   @unique @map(name:
"stripe_subscription_id")
  stripePriceId          String?   @map(name: "stripe_price_id")
  stripeCurrentPeriodEnd DateTime? @map(name:
"stripe_current_period_end")
}
```

Kod 3. Model korisnika i API limita unutar koda aplikacije

4.8. Demonstracija upotrebe Axiosa u aplikaciji

Axios je biblioteka koja se koristi za slanje HTTP zahtjeva. U aplikaciji, Axios se koristi za komunikaciju s ChatGPT i Replicate AI API-jima.
Primjer koda za Axios GET zahtjev prikazan je Kod 4.

```
axios.get('/user')
.then(function (response) {
      console.log(response);
})
.catch(function (error) {
      console.log(error);
})
```

Kod 4. Primjer Axios zahtjeva (GET request)

Primjer Kod 2 šalje GET zahtjev (engl. *request*) na URL '/user' te vraća odgovor u konzolu. Ako dođe do greške, ispisuje se pripadajuća poruka.

Axios također podržava druge vrste HTTP zahtjeva, uključujući POST, PUT i DELETE.

U Kodu 5 dan je primjer korištenja Axiosa za poziv rute „/api/code“, prilikom koje se poziva API za dohvat odgovora ChatGPT-a za stranicu „/code“. Unutar samog zahtjeva poruka korisnika se šalje kao parametar.

```
const response = await axios.post("/api/code",{
      messages: newMessages,
      });
```

Kod 5. Primjer Axios zahtjeva za dohvat odgovora na ruti „/code“

4.9. Testiranje, nadogradnje i održavanje aplikacije

Testiranje je ključni dio razvoja softvera koji omogućuje provjeru ispravnosti koda i osigurava da aplikacija radi kako je očekivano. Za testiranje aplikacije koriste se različite metode testiranja uključujući integracijsko testiranje i end-to-end testiranje. Nadogradnje i održavanje su također važni za razvoj aplikacija. Kako se tehnologija razvija, tako se i aplikacija mora ažurirati ili prilagoditi. To može uključivati ažuriranje, dodavanje novih funkcionalnosti ili poboljšanje postojećih. Kroz cijeli ovaj proces, ključno je održavati dobru dokumentaciju. Dokumentacija olakšava nasljeđivanje koda, održavanje te dodavanje novih funkcionalnosti.

4.10. Lokalno testiranje i pokretanje aplikacije

Lokalno testiranje i pokretanje aplikacije ključni su koraci u razvoju kako bi se osigurala ispravnost i funkcionalnost prije implementacije na produkcijsko okruženje.
Koraci pokretanja aplikacije su:

1. pokretanje Stripea: stripe listen --forward-to http://localhost:3000/api/webhook
2. pokretanje Prisma studija: npx prisma studio
3. pokretanje aplikacije: npm run dev.

Testiranje uključuje nekoliko faza:

1. jedinično testiranje – provodi se za provjeru ispravnosti pojedinačnih funkcija i modula
2. integracijsko testiranje – provjera ispravnosti interakcije između različitih modula i sustava
3. end-to-end testiranje – provjerava se cjelokupna funkcionalnost aplikacije iz perspektive korisnika
4. testiranje performansi – provodi se kako bi procijenili brzinu i skalabilnost aplikacije.

4.11. Implementacija na Vercel

Vercel je platforma za implementaciju mrežnih aplikacija i statičnih mrežnih stranica. Omogućuje programerima jednostavno postavljanje i upravljanje svojim projektima uz pomoć naprednih značajki kao što su automatsko skaliranje, brza globalna isporuka sadržaja i integracije s različitim razvojnim alatima. Vercel podržava popularne frontend frameworke poput Next.js-a, te pruža besprijekorno iskustvo za kontinuiranu integraciju i isporuku. Korisnici mogu brzo implementirati svoje projekte s minimalnim konfiguracijama i osigurati optimalne performanse i sigurnost svojih aplikacija.

Prvi korak u implementaciji aplikacije na Vercelu je kreiranje računa na Vercel platformi. Nakon registracije, potrebno je povezati svoj GitHub račun s Vercelom. Ovaj korak omogućava automatsko preuzimanje repozitorija s projektom.

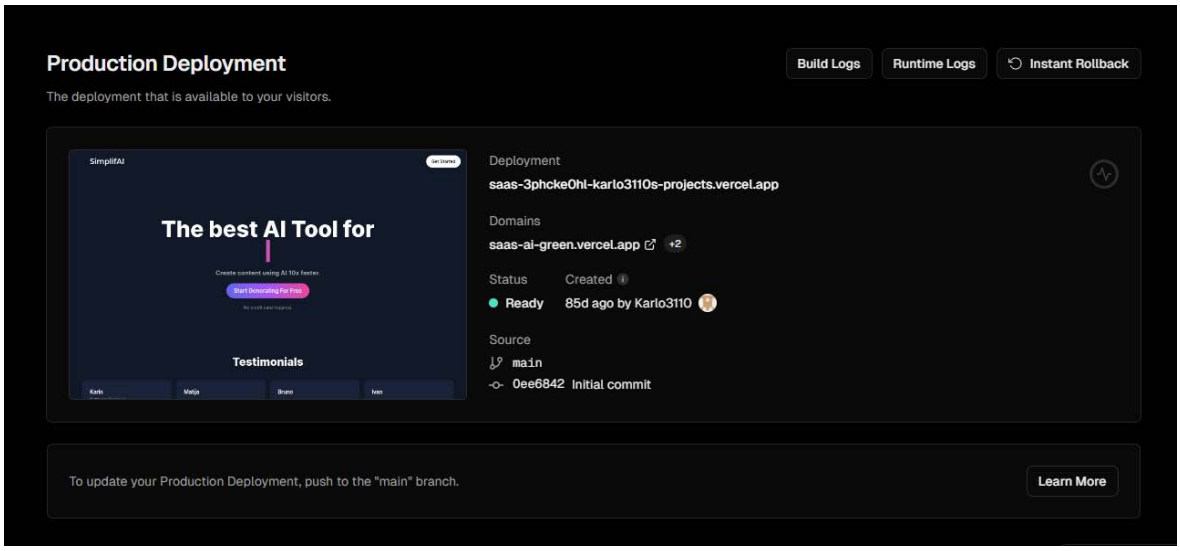
Nakon povezivanja repozitorija, potrebno je odabrati projekt koji se želi implementirati. Vercel automatski prepoznaje da se radi o Next.js aplikaciji i nudi optimalne postavke za implementaciju. Nakon odabira projekta, postavljaju se .env varijable, nakon kojih je potrebno stisnuti tipku „deploy“ čime započinje proces izgradnje i implementacije aplikacije. Nakon

završenog procesa aplikacija je vidljiva na Vercel nadzornoj ploči (Slika 17) te se može vidjeti uživo (Slika 18).

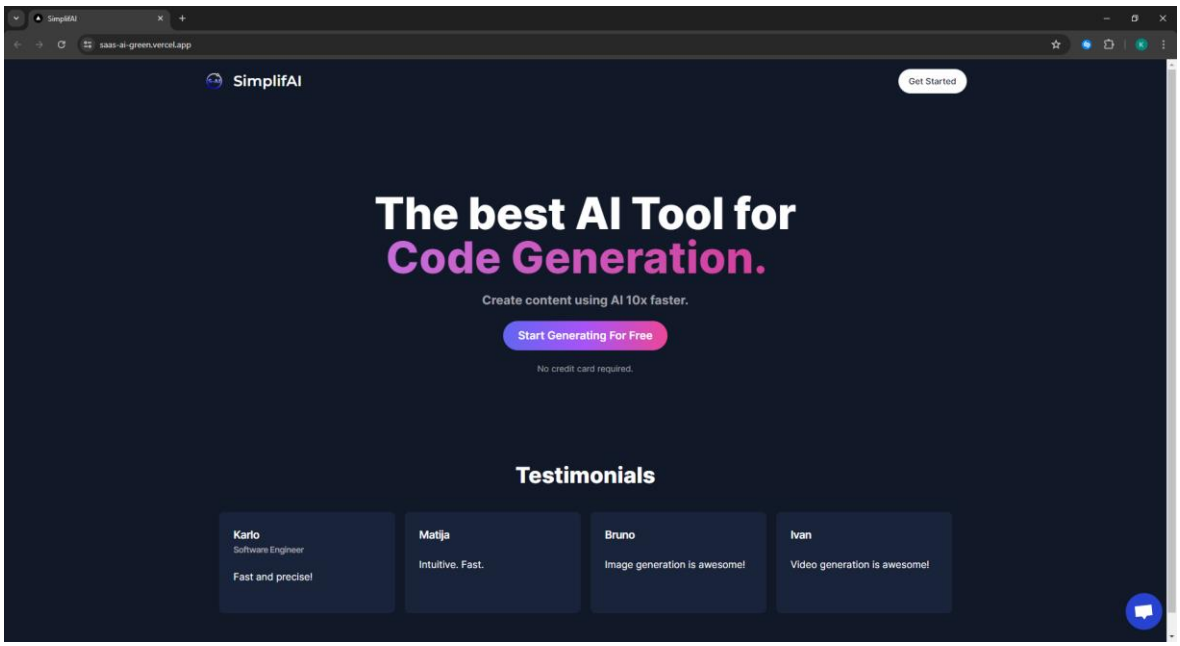
Jedna od ključnih prednosti korištenja Vercela je njegova sposobnost automatskog skaliranja aplikacije na temelju prometa. To znači da aplikacija može podnijeti nagle skokove u broju korisnika, bez potrebe za ručnim skaliranjem infrastrukture. Nakon uspješne implementacije, dobiva se URL na kojem je aplikacija dostupna. Svaka nova promjena u kodu koja se commita u glavni repozitorij, automatski će pokrenuti novu implementaciju, osiguravajući da aplikacija uvijek koristi najnoviju verziju koda.

Vercel također pruža podršku za prilagođene domene, što znači da se može povezati vlastita domena s aplikacijom. Ovo je korisno za promociju i profesionalni izgled aplikacije.

Korištenjem Vercela za implementaciju Next.js aplikacije osigurava se brza i pouzdana distribucija aplikacije, kombinirajući jednostavnost korištenja s moćnim funkcionalnostima optimizacije i skaliranja.



Slika 17. Implementirana aplikacija na Vercelu



Slika 18. Live aplikacija na Vercelu

5. ZAKLJUČAK

Razvojem SAAS aplikacije za korištenje različitih AI sustava predstavljen je postupak integracije AI tehnologija unutar jedne platforme, čime se ujedno predstavila centralizacija sustava, ubrzalo korištenje tehnologije te poboljšala kohezija funkcionalnosti. Kroz ovaj rad, jasno je uočena potreba za objedinjavanjem raznih AI rješenja kako bi se postigla veća učinkovitost i jednostavnost korištenja za krajnje korisnike. Fragmentacija AI tehnologija na različitim platformama često dovodi do problema u kompatibilnosti i usklađivanju različitih sustava, što usporava razvoj i primjenu inovacija. Stoga je centralizirani pristup ključan za postizanje optimalnog korisničkog iskustva i maksimizaciju potencijala AI tehnologija.

Uvođenjem Next.js frameworka za izgradnju mrežnih aplikacija postignuta je visoka razina performansi i optimizacije. Next.js pruža brzu izradu i renderiranje, što je ključno za korisničko iskustvo. Njegova sposobnost da omogući Server-Side Rendering i statičko generiranje stranica doprinosi boljoj optimizaciji za tražilice (SEO) i bržem učitavanju stranica, što direktno utječe na zadovoljstvo korisnika. Osim toga, Next.js olakšava integraciju s drugim alatima i tehnologijama, čineći ga fleksibilnim i moćnim alatom za razvoj modernih mrežnih aplikacija.

Prisma ORM je pojednostavio rad s bazom podataka, omogućujući brže i efikasnije izvršavanje upita i migracija. Korištenje Prisme donijelo je pogodnosti poput automatiziranog generiranja tipova za JavaScript i TypeScript, što smanjuje mogućnost pogrešaka i povećava produktivnost. Integracija Prisme s MySQL bazom podataka omogućila je stabilno i pouzdano pohranjivanje podataka, uz visoku razinu performansi i sigurnosti. Također, Prisma migracije olakšavaju upravljanje promjenama u strukturi baze podataka, što je ključno za održavanje i skaliranje aplikacije.

Korištenjem Shadcn/UI UI biblioteke postignut je moderan dizajn korisničkog sučelja. Tailwind CSS nudi brzu i efikasnu izradu stilova, dok Shadcn/UI pruža gotove komponente koje su lako prilagodljive. Ovaj pristup omogućava konzistentan i atraktivan dizajn, što doprinosi boljem korisničkom iskustvu. Korištenjem Axiosa uz naredbe Prisme omogućena je komunikacija između klijenta i poslužitelja. Axios je robustan alat za slanje HTTP zahtjeva, koji olakšava rad s RESTful API-ima i omogućava jednostavno rukovanje odgovorima i greškama.

Integracija ReplicateAI i ChatGPT API-ja omogućila je implementaciju naprednih AI funkcionalnosti, poput obrade video i audio sadržaja, razgovora s korisnikom te generiranje

slika. Ove funkcionalnosti značajno povećavaju vrijednost aplikacije, pružajući korisnicima sofisticirane alate za različite potrebe.

Integracijom Stripe sustava za plaćanje omogućeni su prihodi s kojima se aplikacija može održavati, skalirati i unaprijediti. Stripe nudi sigurnu i pouzdanu platformu za obradu plaćanja, što je ključno za monetizaciju SAAS aplikacija. Korisnicima je omogućen pristup raznim alatima na jednom mjestu, uz minimalne iznose. Ova fleksibilnost u modelima plaćanja omogućava korisnicima da odaberu plan koji najbolje odgovara njihovim potrebama, dok aplikaciji osigurava stabilan izvor prihoda za daljnji razvoj.

Korištenjem Crispa za chatbot podršku omogućena je brza i efikasna pomoć korisnicima, čime je dodatno poboljšano korisničko iskustvo. Crisp pruža integraciju razgovora uživo, automatiziranih odgovora i analitike, što olakšava komunikaciju s korisnicima i rješavanje njihovih problema u realnom vremenu. Ova podrška je ključna za održavanje visokog stupnja zadovoljstva korisnika i brzo rješavanje njihovih upita i problema. Integracija Crispa također omogućava prikupljanje vrijednih podataka o interakcijama korisnika, što može biti korisno za daljnje poboljšanje aplikacije.

Rezultati rada pokazali su da centralizacija različitih AI sustava unutar jedne aplikacije može imati značajan utjecaj na korisničko iskustvo, omogućiti brži i efikasniji rad korisnika te osigurati skalabilnost. Integracijom AI sustava korisnicima se pruža brz pristup različitim alatima i informacijama što je ključno u današnjem stilu života. Ova centralizacija omogućava korisnicima da efikasnije koriste AI tehnologije, smanjujući potrebu za korištenjem više različitih platformi i alata.

Zaključno, razvoj SAAS aplikacije za integraciju raznih AI sustava predstavio je značajan korak u poboljšanju korištenja AI tehnologija. Postignuti rezultat potvrđuje mogućnost razvijanja centraliziranog i skalabilnog sustava. Budući radovi na SAAS aplikaciji trebali bi se fokusirati na unaprjeđenje performansi, sigurnost i integraciju dodatnih funkcionalnosti kako bi se osigurao dugoročan ciklus života same aplikacije i zadovoljstvo korisnika. Daljnji razvoj može uključivati istraživanje novih AI tehnologija, poboljšanje algoritama za obradu podataka i implementaciju dodatnih sigurnosnih mjera kako bi se osigurala zaštita podataka korisnika.

LITERATURA

1. Herron, D. (2020) Node.js Web Development : Packt Publishing.
2. Next.js, Vercel Inc., Docs | Next.js, https://nextjs.org/docs (pristupljeno 03.04.2024.)
3. Next.js, Vercel Inc., Rendering: Static Site Generation, https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation (pristupljeno 03.04.2024.)
4. Next.js, Vercel Inc., Rendering: Server-side Rendering, https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering (pristupljeno 03.04.2024.)
5. Next.js, Vercel Inc., Rendering:Client-Side Rendering, https://nextjs.org/docs/pages/building-your-application/rendering/client-side-rendering (pristupljeno 03.04.2024.)
6. Crisp, Crisp IM SAS, Guides, https://docs.crisp.chat/guides/ (pristupljeno 03.04.2024.)
7. Prisma, ORM | Prisma Documentation, https://www.prisma.io/docs/orm (pristupljeno 04.04.2024.)
8. Axios, John Jakob "Jake" Sarjeant, Getting Started | Axios Docs, https://axios-http.com/docs/intro (pristupljeno 05.04.2024.)
9. ChatGPT, OpenAI Inc., službena ChatGPT dokumentacija, https://platform.openai.com/docs/api-reference (pristupljeno 06.04.2024.)
10. Replicate, Replicate Inc., službena Replicate dokumentacija, https://replicate.com/docs/reference/http (pristupljeno 06.04.2024.)
11. Replicate, Replicate Inc., model zeroscope-v2-xl, https://replicate.com/anotherjesse/zeroscope-v2-xl (pristupljeno 07.04.2024.)
12. Replicate, Replicate Inc., model riffusion, https://replicate.com/riffusion/riffusion (pristupljeno 07.04.2024.)
13. MySQL, Oracle Corporation, MySQL Workbench documentation, https://dev.mysql.com/doc/workbench/en/ (pristupljeno 07.04.2024.)
14. Stripe, Stripe Inc., Stripe Documentation, https://docs.stripe.com/ (pristupljeno 08.04.2024.)

15. Clerk, Clerk Inc., Welcome to Clerk Docs, https://clerk.com/docs (pristupljeno 09.04.2024.)

SAŽETAK

Ovaj završni rad se fokusira na razvoj SAAS aplikacije koja integrira različite AI tehnologije u jedan centralni sustav. Cilj je bio riješiti problem fragmentacije AI modela različitih primjena koje su rasprostranjene na različitim platformama te omogućiti jednostavnu integraciju i upravljanje tim tehnologijama. Kroz ovaj proces, korištene su različite tehnologije i alati, uključujući Clerk za autentifikaciju, ChatGPT i Replicate AI za AI funkcionalnosti, Stripe za upravljanje plaćanjima, Crisp za chat podršku, Prisma ORM za upravljanje bazom podataka i Axios za upravljanje HTTP zahtjevima. Ove tehnologije su omogućile izgradnju robusne i skalabilne aplikacije koja može podržati širok spektar AI funkcionalnosti.

Kroz ovaj rad, također je shvaćena važnost testiranja, nadogradnje i održavanja aplikacije. Proces testiranja je osigurao ispravnost rada aplikacije i identifikaciju te rješavanje problema. Nadalje, naglašena je važnost stalne nadogradnje i održavanja aplikacije radi osiguranja njene dugoročne održivosti i usklađenosti s najnovijim tehnološkim trendovima.

Ključne riječi: SAAS aplikacija, AI tehnologije, Clerk, ChatGPT, Replicate AI, Stripe, Crisp, Prisma ORM, Axios, Testiranje

SUMMARY

This final thesis focuses on the development of a SAAS application that integrates various AI technologies into a unified central system. The goal was to solve the problem of fragmentation of AI models for different applications spread across various platforms and to enable easy integration and management of these technologies. Throughout this process, various technologies and tools were utilized, including Clerk for authentication, ChatGPT and Replicate AI for AI functionalities, Stripe for payment management, Crisp for chat support, Prisma ORM for database management, and Axios for handling HTTP requests. These technologies facilitated the development of a robust and scalable application capable of supporting a wide range of AI functionalities.

Through this work, the importance of testing, upgrading, and maintaining the application was also understood. The testing process ensured the correctness of the application's functionality and facilitated the identification and resolution of issues. Furthermore, emphasis was placed on the continuous upgrading and maintenance of the application to ensure its long-term sustainability and alignment with the latest technological trends.

Keywords: SAAS application, AI technologies, Clerk, ChatGPT, Replicate AI, Stripe, Crisp, Prisma ORM, Axios, Testing