Home

Filip Šturlić edited this page on Oct 22, 2025 · 4 revisions

Edit New page

Programsko inženjerstvo ak.god 2025./2026.

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Računko

Tim: LA11.4

Ime tima: TimFiLAn

Nastavnik: Vlado Sruk

TimFiLAn

+ Add a custom sidebar

Clone this wiki locally

https://github.com/lukskoki/Racunko.

1. Opis projektnog zadatka

Edit New page

Filip Šturlić edited this page on Nov 14, 2025 · 3 revisions

Cilj ovog projekta je razviti mobilnu aplikaciju za evidenciju i praćenje mjesečnih troškova koja će korisnicima omogućiti jednostavniji i učinkovitiji način upravljanja osobnim financijama. Aplikacija će koristiti umjetnu inteligenciju za automatsko prepoznavanje i kategorizaciju troškova putem digitalnog skeniranja računa, čime se uklanja potreba za ručnim unosom. Uz pregled potrošnje po kategorijama i usporedbu troškova po mjesecima, aplikacija će omogućiti i zajedničko praćenje financija unutar kućanstva te davati personalizirane savjete za bolje upravljanje budžetom. Time projekt Računko objedinjuje funkcionalnosti praćenja, analize i planiranja potrošnje u jedinstvenu, intuitivnu platformu prilagođenu svakodnevnom korištenju.

Potencijalni korisnici i njihovi problemi

1. **Pojedinačni korisnici (glavni korisnici):**
   Problem: Većina korisnika nema jednostavan i automatiziran način praćenja svojih mjesečnih troškova. Računi se često gube ili ostaju neorganizirani, a ručni unos troškova u tablice zahtjeva puno vremena i nije pregledan. Nedostatak jasnog uvida u strukturu potrošnje otežava planiranje budžeta i štednju.
   Potrebe: Korisnici žele digitalno rješenje koje automatski prepoznaje i kategorizira troškove, omogućuje analizu potrošnje po kategorijama te daje personalizirane savjete za bolju kontrolu financija.
2. **Članovi kućanstva (sporedni korisnici):**
   Problem: U kućanstvima s više članova teško je zajednički pratiti troškove, jer su računi razdvojeni, a financijske informacije ne objedinjene na jednom mjestu.
   Potrebe: Potreban im je sustav koji omogućava zajedničko praćenje troškova i budžeta, s različitim razinama pristupa, ovisno o ulozi člana (npr. partner, dijete).
3. **Administratori sustava:**
   Problem: Administratori često nemaju pregled nad korištenjem aplikacija i ponašanjem korisnika, što otežava održavanje sustava i provjeru sigurnosti podataka.
   Potrebe: Potreban im je alat koji omogućava nadzor nad korisnicima, transakcijama i funkcionalnostima sustava, uz mogućnost generiranja izvještaja i brisanja korisničkih računa prema potrebi.

Aplikacija omogućava korisnicima niz pogodnosti

- **Glavnim korisnicima:**
  - Digitalno skeniranje računa koristeći kameru mobilnog uređaja
  - Automatsko prepoznavanje i kategorizacija troškova korištenjem umjetne inteligencije
  - Definiranje mjesečnog budžeta i fiksnih troškova koji se automatski uračunavaju na početku svakog mjeseca
  - Prikaz potrošnje po kategorijama kroz tablične i grafičke prikaze
  - Mogućnost dodavanja sporednih korisnika i određivanja njihove razine pristupa
  - Personalizirani savjeti o upravljanju budžetom putem integriranog ChatBota
- **Sporednim korisnicima:**
  - Jednostavno evidentiranje vlastitih troškova putem skeniranja računa
  - Uvid u dodijeljeni mjesečni budžet i trenutačnu potrošnju
  - Automatsko povezivanje svih troškova s glavnim korisnikom radi zajedničkog praćenja kućanstva
- **Administratorima:**
  - Pregled svih korisnika i njihovih aktivnosti
  - Upravljanje korisničkim profilima (kreiranje, izmjena i brisanje)
  - Nadzor nad funkcioniranjem aplikacije i generiranje statističkih izvještaja

Potencijalna korist projekta

Razvoj aplikacije **Računko** donosi brojne koristi za krajnje korisnike i širu zajednicu:

1. **Povećana financijska pismenost:** Korisnici će imati jasan pregled svojih financijskih navika i mogućnost učinkovitijeg planiranja troškova.
2. **Ušteda vremena:** Automatizirano prepoznavanje računa i kategorizacija troškova uklanjaju potrebu za ručnim unosom i analizom.
3. **Bolja organizacija kućanstva:** Mogućnost povezivanja više korisnika u zajednički sustav olakšava upravljanje zajedničkim budžetom i praćenje ukupnih izdataka.
4. **Sigurnost i kontrola:** Administratori mogu nadgledati sustav, osigurati ispravan rad aplikacije i zaštitu korisničkih podataka.

Primjenom umjetne inteligencije i automatizacije unosa podataka, Računko značajno pojednostavljuje svakodnevno financijsko praćenje i potiče odgovornije upravljanje osobnim i kućanskim budžetom.

Uloge korisnika u sustavu

- **Glavni korisnik:**
  - Registrira se putem vanjskih servisa za autentifikaciju (npr. OAuth 2.0)
  - Definira mjesečni budžet i fiksne troškove
  - Pregledava sve evidentirane troškove i financijske statistike
  - Dodaje sporedne korisnike te im dodjeljuje razinu pristupa
  - Prima personalizirane preporuke putem ChatBota

Clone this wiki locally

https://github.com/lukskoki/Racunko.

- **Sporedni korisnik:**
  - Koristi aplikaciju u okviru dodijeljenih ovlasti
  - Može skenirati račune, unositi troškove i pregledavati svoj dio budžeta
  - Njegovi se troškovi automatski povezuju s glavnim korisnikom
- **Administrator:**
  - Ima pristup svim korisnicima i transakcijama
  - Može uređivati, dodavati i brisati korisničke profile
  - Nadgleda funkcioniranje sustava i generira izvještaje o korištenju aplikacije

Ovakva raspodjela uloga omogućuje jasan i siguran pristup sustavu, te jednostavno upravljanje funkcionalnostima prema potrebama svake skupine korisnika.

Tehničke značajke i plan razvoja

Projekt **Računko** bit će razvijen kao **mobilna aplikacija**, optimizirana za Android i iOS platforme. Fokus će biti na intuitivnom korisničkom sučelju i pouzdanom sustavu prepoznavanja podataka s računa. U prvoj fazi razvoja implementirat će se ključne funkcionalnosti: registracija, skeniranje računa, kategorizacija troškova i pregled statistike.

Napredne značajke poput analize financijskih navika, prediktivne procjene potrošnje i naprednog sustava preporuka mogle bi se uvesti ako se odlučimo za nastavak razvoja nakon završetka semestra i potencijalnu komercijalizaciju aplikacije.

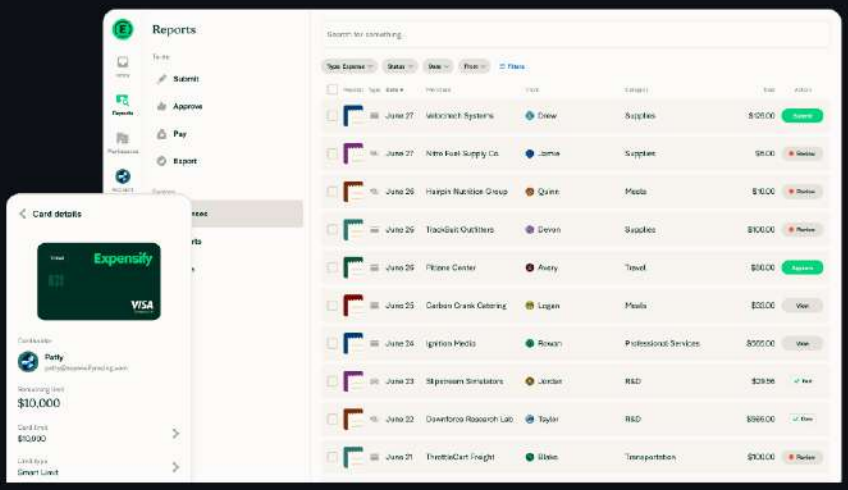
Projekt će uključivati fazu testiranja, s naglaskom na ispravnost prepoznavanja podataka, sigurnost pohranjenih informacija i stabilnost aplikacije. Nakon testiranja bit će izrađene upute za korištenje i tehnička dokumentacija koja će služiti kao temelj za buduće nadogradnje i održavanje sustava.

Primjeri sličnih rješenja

- Spendee Spendee je aplikacija za upravljanje osobnim financijama i praćenje potrošnje kroz intuitivno sučelje i bogat vizualni prikaz. Omogućuje stvaranje kategorija troškova, postavljanje budžeta i povezivanje bankovnih računa radi automatskog uvoza transakcija. Aplikacija nudi i mogućnost stvaranja zajedničkih „walleta" za obiteljske ili grupne budžete, što je korisno za zajedničko praćenje potrošnje. Dostupna je kao mobilna i web verzija, a naglasak stavlja na vizualnu analitiku i jednostavno upravljanje svakodnevnim troškovima.

Slika 1.1: Aplikacija Spendee

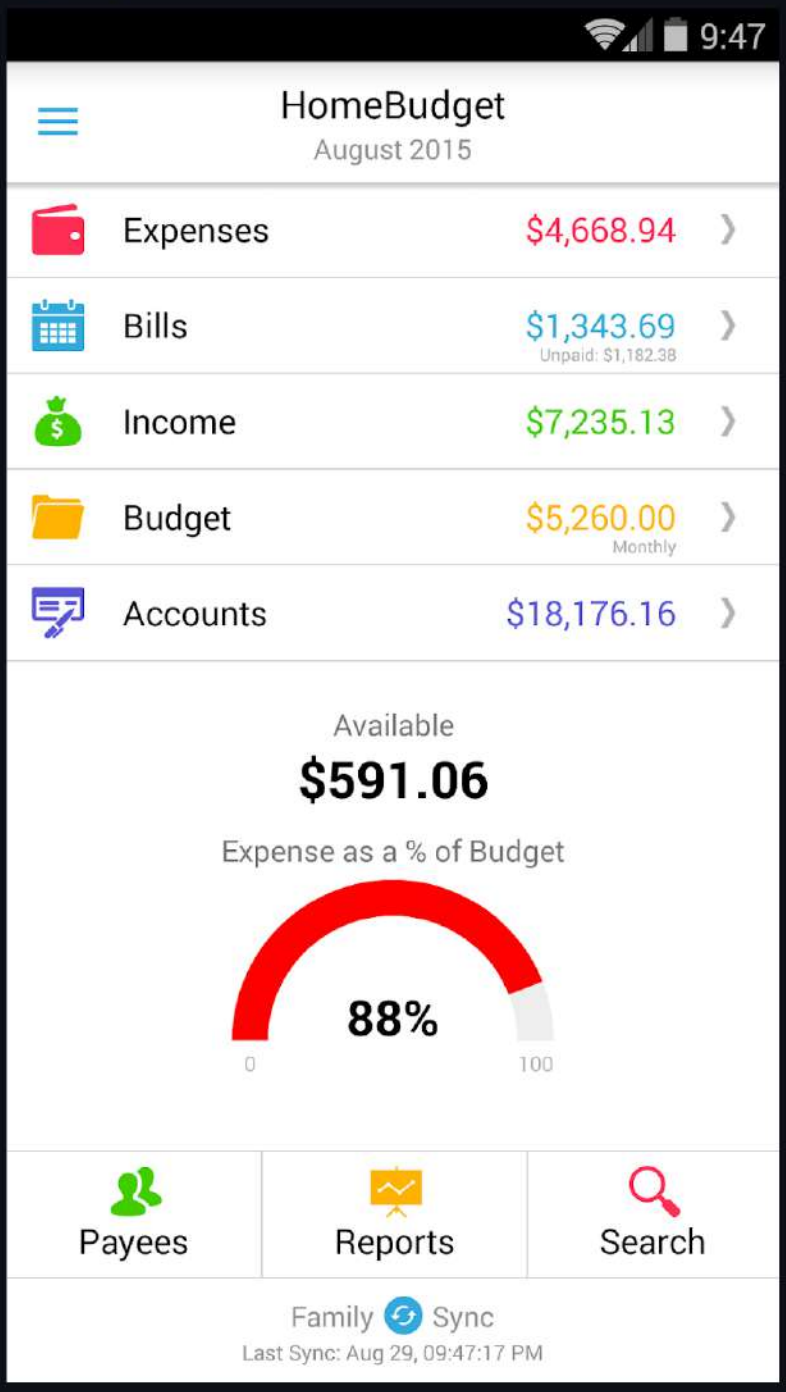
- Expensify Expensify je napredna aplikacija za skeniranje i upravljanje računima pomoću SmartScan tehnologije koja automatski prepoznaje iznos, datum i kategoriju troška. Iako se često koristi u poslovnom okruženju, aplikacija može poslužiti i za osobne financije. Podržava izradu detaljnih izvještaja, praćenje putnih troškova, povezivanje s karticama i automatizirano knjiženje troškova. Dostupna je kao mobilna i web aplikacija, a ističe se snažnim sustavom analitike i automatizacije.



Slika 1.2: Aplikacija Expensify

- Home Budget with Sync Home Budget with Sync omogućuje detaljno praćenje prihoda i rashoda uz mogućnost obiteljskog dijeljenja budžeta. Aplikacija podržava stvaranje više korisnika unutar jednog kućanstva te sinkronizaciju

podataka između uređaja u realnom vremenu. Sadrži pregledne grafikone, praćenje kategorija troškova, postavljanje mjesečnih budžeta i organiziranje digitalnih kopija računa. Dostupna je samo kao mobilna aplikacija i posebno je pogodna za obiteljski budžet.



Slika 1.3: Aplikacija Home Budget with Sync

Dodatne funkcionalnosti koje bi se mogle dodati

Za glavne korisnike

- Napredno planiranje budžeta: automatska raspodjela sredstava prema navikama trošenja
- Praćenje štednih ciljeva: grafički prikaz napretka prema zadanim ciljevima
- Personalizirane preporuke: savjeti temeljeni na prethodnoj potrošnji i AI analizama
- Upozorenja o prekoračenju budžeta: pravovremene notifikacije o povećanom trošenju

Za sporedne korisnike

- Ograničeni budžeti: dodjela potrošnih limita i praćenje njihova korištenja
- Gamifikacija štednje: značke i nagrade za racionalno trošenje
- Obiteljske liste troškova: zajedničko evidentiranje za određene kategorije (npr. prehrana)

Za administratore

- Analitika korištenja aplikacije: aktivni korisnici, najčešće korištene funkcije
- Automatsko generiranje izvještaja: mjesečni, kvartalni ili godišnji pregledi aktivnosti
- Praćenje trendova potrošnje: statistike o najpopularnijim kategorijama troškova

- Upravljanje korisničkim ulogama: detaljnije postavke pristupa i ograničenja

Dodatne mogućnosti za budući razvoj

- Podrška za više valuta i jezika
- Napredno pretraživanje i filtriranje računa i troškova
- Integracija bankovnih API-ja za automatski uvoz transakcija
- Povezivanje s drugim aplikacijama (npr. Google Calendar za označavanje dospijeća računa)
- Automatsko generiranje PDF izvještaja ili analiza troškova

Ove funkcionalnosti mogu dodatno unaprijediti korisničko iskustvo, poboljšati organizaciju kućnog budžeta te proširiti mogućnosti upravljanja financijama za sve vrste korisnika.

TimFiLAn

2. Analiza zahtjeva

Filip Šturlić edited this page on Nov 14, 2025 · 7 revisions

Edit New page

Funkcionalni zahtjevi

ID zahtjeva	Naziv	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-1	Registracija i prijava korisnika	Sustav mora omogućiti registraciju i prijavu glavnog, sporednog korisnika ili administratora putem e-mail adrese ili vanjskih servisa (OAuth 2.0).	Visok	Zahtjev dionika	Korisnik se može registrirati i prijaviti putem e-maila ili Google računa te pristupiti aplikaciji kroz siguran sustav autentifikacije.
F-2	Postavljanje mjesečnog budžeta	Sustav mora omogućiti glavnom korisniku definiranje ukupnog mjesečnog budžeta.	Visok	Dokument zahtjeva	Glavni korisnik unosi iznos budžeta i sustav ga pohranjuje te prikazuje u sučelju.
F-3	Upravljanje kategorijama troškova	Sustav mora omogućiti glavnom korisniku pregled, dodavanje, uređivanje i brisanje kategorija troškova.	Visok	Povratne informacije korisnika	Glavni korisnik može stvarati nove kategorije i uspješno ih spremiti.
F-4	Unos računa	Sustav mora omogućiti svim korisnicima (glavnom i sporednom) ručni unos računa ili skeniranje računa radi automatskog prepoznavanja podataka.	Visok	Dokument zahtjeva	Sustav prepoznaje iznos, datum i artikle s računa pomoću AI modela i automatski ih kategorizira.
F-5	Automatska kategorizacija troškova	Sustav mora automatski kategorizirati troškove pomoću umjetne inteligencije.	Visok	Zahtjev dionika	Sustav točno prepoznaje i kategorizira više od 90% računa bez ručne intervencije.
F-6	Evidencija troškova	Sustav mora omogućiti roditelju i svakom korisniku pregled, uređivanje i brisanje vlastitih unesenih troškova. Roditelj može uređivati i troškove djeteta.	Visok	Dokument zahtjeva	Korisnik vidi svoje troškove po datumima i kategorijama; roditelj vidi i može uređivati troškove djeteta.
F-7	Dodavanje fiksnih troškova	Sustav mora omogućiti glavnom korisniku definiranje fiksnih troškova koji se automatski uračunavaju svaki mjesec.	Srednji	Zahtjev dionika	Fiksni troškovi se prikazuju na početku mjeseca.
F-8	Usporedba potrošnje po mjesecima	Sustav mora omogućiti usporedbu troškova između različitih mjeseci.	Srednji	Povratne informacije korisnika	Korisnik može pregledati grafički prikaz trendova potrošnje po mjesecima.
F-9	Kreiranje grupe (obitelji)	Glavni korisnik može kreirati grupu i automatski postaje njen administrator.	Visok	Dokument zahtjeva	Grupa se sprema u sustav i korisnik dobiva administratorska prava.
F-10	Slanje zahtjeva za ulazak u grupu	Sporedni korisnik šalje zahtjev za pridruživanje postojećoj grupi.	Visok	Zahtjev dionika	Zahtjev se pohranjuje i čeka odobrenje glavnog korisnika.
F-11	Odobravanje zahtjeva za ulazak u grupu	Glavni korisnik pregledava i odobrava zahtjeve za ulazak u grupu te dodjeljuje uloge.	Visok	Dokument zahtjeva	Korisnik dobiva ulogu roditelj/partner/dijete.
F-12	Upravljanje sporednim korisnicima	Glavni korisnik može uređivati i brisati sporedne korisnike te upravljati razinama pristupa.	Visok	Dokument zahtjeva	Sporedni korisnik se uspješno mijenja ili uklanja.
F-13	Evidencija zajedničkih troškova	Prikaz objedinjene potrošnje svih korisnika unutar grupe.	Srednji	Povratne informacije korisnika	Korisnik vidi zajedničke troškove kućanstva.
		Integrirani AI chatbot koji daje financijske		Dokument	Chatbot daje relevantne

F-14	ChatBot savjetnik	koji daje financijske savjete temeljem potrošnje.	Visok	Dokument zahtjeva	odgovore povezane s potrošnjom.
F-15	Grafički prikaz troškova	Vizualizacija potrošnje po kategorijama.	Nizak	Povratne informacije korisnika	Prikazani grafikoni potrošnje.
F-16	Administracija korisnika	Administrator uređuje i briše korisničke račune.	Visok	Zahtjev dionika	Admin može pristupiti i mijenjati korisničke informacije.
F-17	Generiranje izvještaja	Admin može generirati izvještaje o korištenju aplikacije i statistici potrošnje.	Srednji	Dokument zahtjeva	Izvještaji su dostupni u PDF/CSV formatu.
F-18	Pregled pohranjenih računa	Korisnici mogu pregledati slike svih pohranjenih računa.	Srednji	Povratne informacije korisnika	Sve slike računa su dostupne u arhivi.
F-19	Odabir trgovine/lokacije	Sustav omogućuje korisniku da odabere trgovinu ili lokaciju pri unosu računa.	Visok	Dokument zahtjeva	Trgovina se sprema uz zapis o trošku.
F-20	Upravljanje budžetnim ciljevima	Glavni korisnik može definirati ciljeve štednje ili potrošnje.	Srednji	Zahtjev dionika	Ciljevi se pohranjuju i prate unutar aplikacije.
F-21	Postavljanje limita za dijete	Roditelj može postaviti mjesečni limit potrošnje za dijete.	Visok	Dokument zahtjeva	Sustav prati potrošnju i ograničenja djeteta.
F-22	Upravljanje ulogama	Sustav omogućuje glavnom korisniku dodjelu, promjenu i uklanjanje korisničkih uloga u grupi.	Visok	Dokument zahtjeva	Uloga se uspješno mijenja i sprema.
F-23	Obavijesti	Sustav šalje obavijesti o: (1) pristiglim zahtjevima za ulazak u grupu, (2) premašivanju limita potrošnje djeteta.	Nizak	Zahtjev dionika	Korisnik prima odgovarajuće push/inline obavijesti.
F-24	Povijest aktivnosti	Admin vidi povijest događaja u sustavu (registracije, brisanja, promjene uloga itd.).	Srednji	Dokument zahtjeva	Povijest aktivnosti je prikazana u administracijskom sučelju.

Ostali zahtjevi

ID zahtjeva	Naziv	Opis	Prioritet
NF-1.1	Autentifikacija i autorizacija	Sustav mora koristiti sigurne metode autentifikacije (OAuth 2.0, šifrirane lozinke) radi zaštite korisničkih računa.	Visok
NF-1.2	Sigurnost podataka	Svi korisnički i financijski podaci moraju biti zaštićeni u skladu s GDPR regulativom i pohranjeni šifrirano.	Visok
NF-1.3	Komunikacijski protokoli	Cjelokupna komunikacija između klijenta i poslužitelja mora koristiti HTTPS protokol.	Visok
NF-2.1	Performanse	Sustav mora obraditi i prikazati podatke o troškovima unutar 2 s po zahtjevu korisnika.	Srednji
NF-2.2	Prepoznavanje računa	AI modul mora prepoznati tekst s računa s minimalno 90 % točnosti.	Visok
NF-3.1	Skalabilnost	Sustav mora podržati rast do 1000 aktivnih korisnika bez pada performansi većeg od 10 % u vremenu odziva.	Srednji
NF-3.2	Održavanje	Sustav mora biti razvijen prema MVC arhitekturi radi lakšeg održavanja i nadogradnje.	Visok
NF-3.3	Dokumentacija	Sustav mora imati tehničku dokumentaciju, upute za instalaciju i korisnički priručnik.	Nizak
NF-4.1	Dostupnost	Sustav mora biti dostupan minimalno 99 % vremena tijekom godine.	Srednji
NF-5.1	Podržane platforme	Aplikacija mora biti prilagođena mobilnim uređajima i preglednicima (responsivni dizajn).	Nizak
NF-6.1	Testiranje	Sustav mora biti testiran automatiziranim alatima (npr. Selenium, Jest) radi provjere funkcionalnosti i sigurnosti.	Visok
NF-6.2	Integracija s vanjskim servisima	Sustav mora podržavati integraciju s vanjskim autentifikacijskim i AI servisima.	Srednji

Dionici

1. **Glavni korisnik** – krajnji korisnik aplikacije koji upravlja budžetom, troškovima i sporednim korisnicima.
2. **Sporedni korisnik** – korisnik kojem glavni korisnik dodjeljuje pristup i ovlasti (npr. partner, dijete).
3. **Administrator** – osoba zadužena za nadzor sustava, upravljanje korisnicima i izvještavanje.

4. **Baza podataka** – pohranjuje sve korisničke i financijske informacije te slike računa.
5. **Razvojni tim** – odgovoran za implementaciju, održavanje i testiranje aplikacije.

Aktori i njihovi funkcionalni zahtjevi

A-1 Glavni korisnik (inicijator) može:

- Registrirati se i prijaviti (F-1)
- Postaviti i uređivati mjesečni budžet (F-2)
- Upravljati kategorijama troškova (F-3)
- Kreirati grupu i upravljati članovima (F-9, F-11, F-12)
- Upravljati korisničkim ulogama (F-23)
- Odobravati zahtjeve za ulazak u grupu (F-11)
- Dodavati i pregledavati troškove (F-4, F-6)
- Skenirati račune i odabrati trgovinu/lokaciju (F-4, F-19)
- Postaviti fiksne troškove (F-7)
- Postaviti budžetne ciljeve (F-20)
- Postaviti limit potrošnje za dijete (F-21)
- Pregledavati zajedničke troškove kućanstva (F-13)
- Pregledavati grafikone i usporedbu potrošnje (F-15, F-8)
- Koristiti AI chatBota za savjete o potrošnji (F-14)
- Pregledavati svoje računa (F-18)
- Primati obavijesti o zahtjevima za grupu i limitu djeteta (F-23)
- Pregledavati pohranjene slike računa (F-18)

A-2 Sporedni korisnik (inicijator) može:

- Registrirati se i prijaviti (F-1)
- Poslati zahtjev za ulazak u grupu (F-10)
- Skenirati račune i dodavati troškove (F-4, F-5, F-6)
- Odabrati trgovinu/lokaciju pri unosu računa (F-19)
- Pregledavati vlastite troškove (F-6)
- Pregledavati dodijeljeni limit i budžet (ako je dijete) (F-21)
- Pregledavati zajedničku potrošnju kućanstva (F-13)
- Pregledavati svoje računa (F-18)
- Koristiti chatBota (F-14)
- Primati obavijesti (npr. status zahtjeva za ulazak) (F-23)
- Pregledati povijest aktivnosti (F-24)

A-3 Administrator (inicijator) može:

- Upravljati korisnicima – pregled, uređivanje i brisanje profila (F-16)
- Kreirati, mijenjati ili uklanjati korisničke uloge (dodatno proširenje F-22)
- Pregledavati sve transakcije svih korisnika (iz opisa sustava; pokriva više F-zahtjeva)
- Generirati izvještaje o korištenju aplikacije i statistici troškova (F-17)
- Pregledavati povijest aktivnosti sustava (F-24)

A-4 Baza podataka (sudionik) može:

- Pohranjivati podatke o registriranim korisnicima (glavni i sporedni korisnici).
- Pohranjivati podatke o grupama i njihovim članovima (uključujući uloge i razine pristupa).
- Pohranjivati podatke o mjesečnim i fiksnim troškovima, prihodima i računima.
- Pohranjivati podatke potrebne za funkcioniranje ostalih funkcionalnosti sustava (npr. statistika, AI kategorizacija računa).
- Osigurati integritet podataka i sigurnosno kopiranje prema propisanim standardima.

TimFiLAn

3. Specifikacija zahtjeva sustava

Filip Šturlić edited this page 2 hours ago - 9 revisions

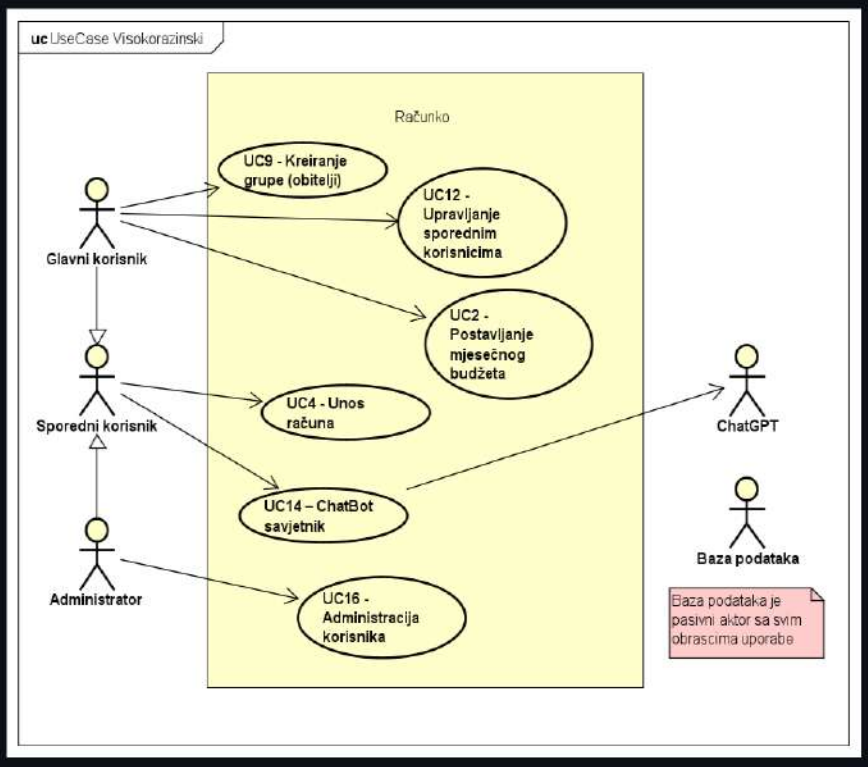
Edit | New page

Obrasci uporabe

Dijagrami obrazaca uporabe

Napomena: Aktori Glavni korisnik, sporedni korisnik i administrator nasljeđuju registriranog korisnika.

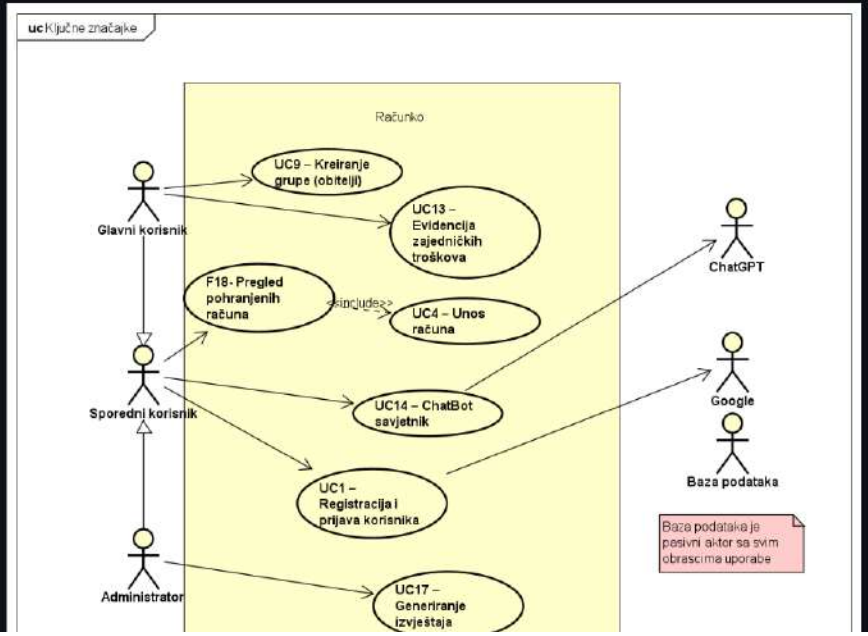
1. Visokorazinski dijagram obrazaca uporabe cijelog sustava



Slika 3.1. Prikazuje najvažnije funkcionalnosti sustava iz perspektive krajnjih korisnika.

2. dijagram obrazaca uporabe za ključne značajke

| detaljnije dijagrame koji se odnose na specifične značajke, poput procesa prijave korisnika, upravljanja podacima, ... obrade transakcija.



Clone this wiki locally

https://github.com/lukskok1/Racunko.

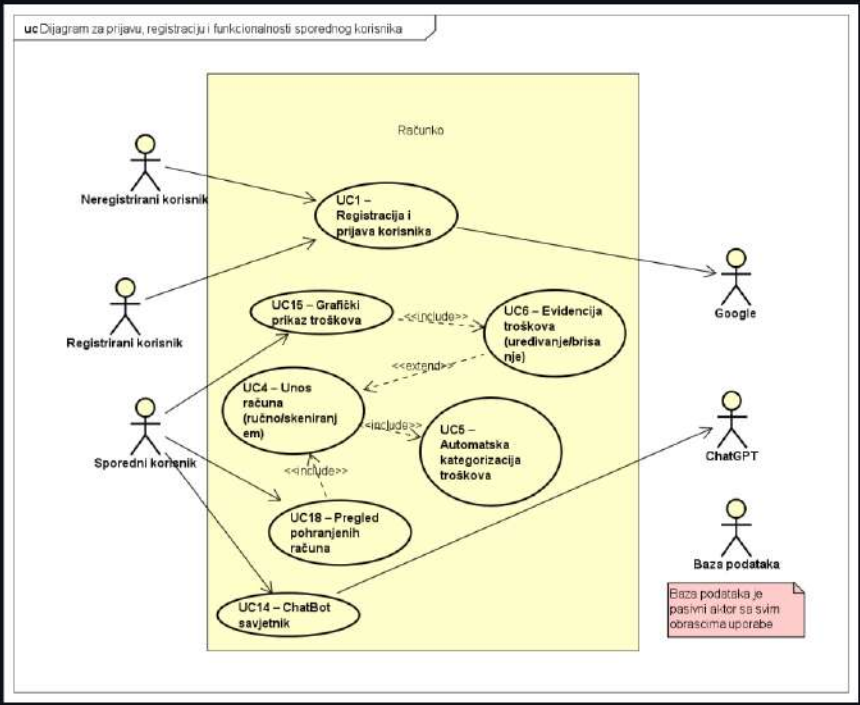
Slika 3.2. Dijagram za ključne značajke sustava

3. dijagram obrazaca uporabe za korisničke uloge

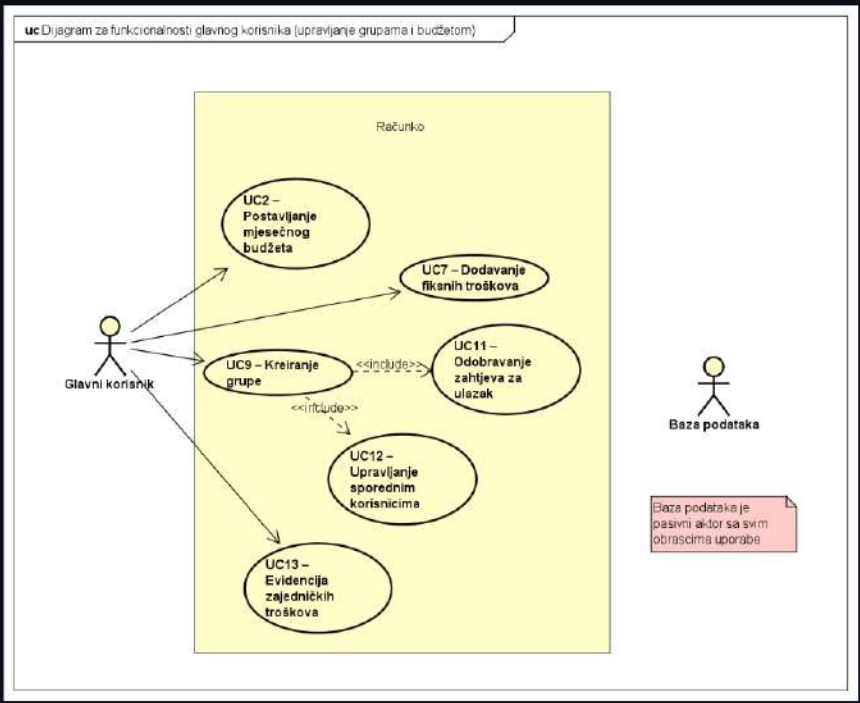
| Identificira glavne korisničke aktore i njihove interakcije sa sustavom.

| definira različitih vrsta korisnika i funkcionalnosti koje su im potrebne

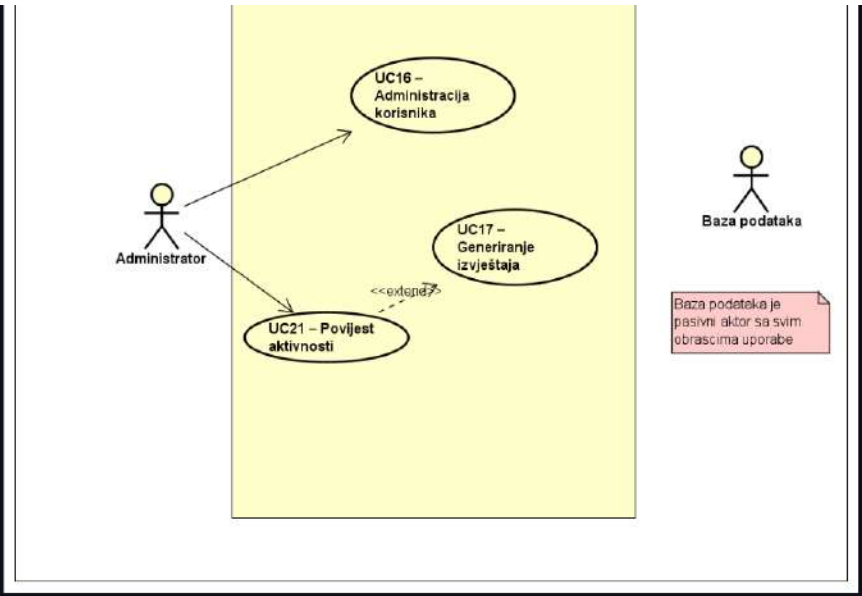
| Osigurava bolje razumjevanje prioritete i specifične potrebe različitih grupa korisnika.



Slika 3.2. Dijagram za prijavu, registraciju i osnovne funkcionalnosti korisnika

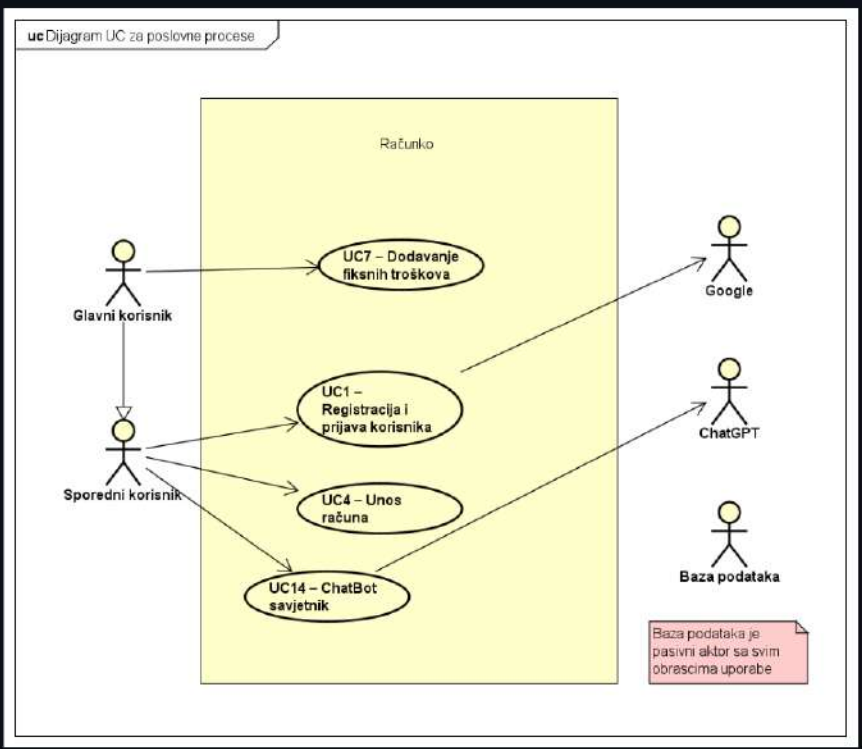


Slika 3.3. Prikazuje funkcionalnosti glavnog korisnika i funckionalnosti vezane za grupe (obitelji)



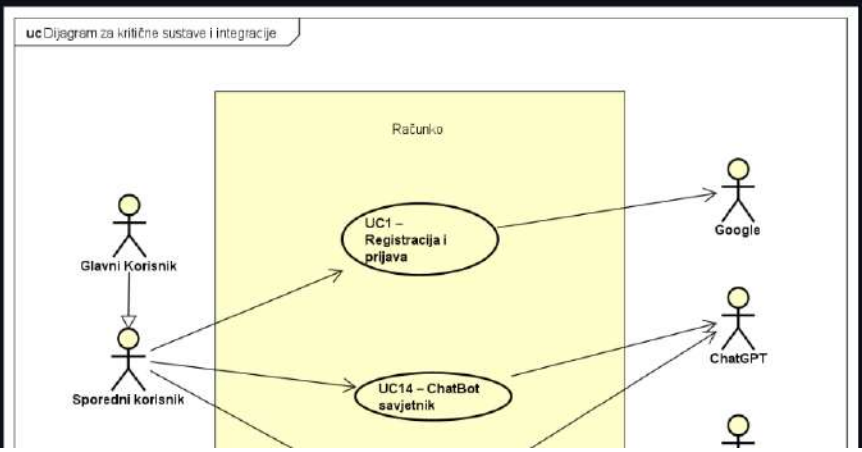
Slika 3.4. Prikazuje najvažnije funkcionalnosti za administraciju sustava i nadzor.

4. dijagram obrazaca uporabe za osnovne poslovne procese



Slika 3.5. Prikazuje kako sustav podržava osnovne poslovne procese organizacije

5. dijagram obrazaca uporabe za kritične sustave i integracije



Slika 3.6. Prikazuje interakcije sustava s vanjskim sustavima ili integracijama

Opis obrazaca uporabe

UC1 – Registracija i prijava korisnika

- **Glavni sudionik:** Glavni korisnik, Sporedni korisnik, Administrator
- **Cilj:** Registrirati novi korisnički račun ili se prijaviti u sustav
- **Sudionici:** Baza podataka, Google (OAuth 2.0)
- **Preduvjet:** Korisnik nije prijavljen
- **Opis osnovnog tijeka:**

> 1. Korisnik odabire "Registracija" ili "Prijava".
> 2. Unosi podatke ili koristi Google prijavu.
> 3. Sustav provjerava podatke.
> 4. Sustav prijavljuje korisnika.

- **Opis mogućih odstupanja:**

> 2.a Neispravni ili nepotpuni podaci 1. Sustav prikazuje poruku o pogrešci.
>
> 3.a Google prijava neuspješna 1. Sustav javlja grešku u autentifikaciji.

UC2 – Postavljanje mjesečnog budžeta

- **Glavni sudionik:** Glavni korisnik
- **Cilj:** Definirati mjesečni budžet
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Osnovni tijek:**

> 1. Korisnik odabire "Postavi budžet".
> 2. Unosi iznos.
> 3. Sustav sprema budžet.

- **Odstupanja:**

> 2.a Neispravan iznos 1. Sustav traži ponovni unos.

UC3 – Upravljanje kategorijama troškova

- **Glavni sudionik:** Glavni korisnik
- **Cilj:** Dodavanje/uređivanje/brisanje kategorija
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Osnovni tijek:**

> 1. Korisnik otvara kategorije.
> 2. Odabire dodavanje/uređivanje/brisanje.
> 3. Sustav sprema promjene.

- **Odstupanja:**

> 2.a Pokušaj brisanja kategorije koja ima povezane troškove 1. Sustav zabranjuje brisanje.
>
> 2.b Duplikat naziva kategorije 1. Sustav javlja da kategorija već postoji.

UC4 – Unos računa

- **Glavni sudionik:** Glavni ili sporedni korisnik
- **Cilj:** Unos računa ručno ili skeniranjem
- **Sudionici:** AI modul, Baza podataka
- **Preduvjet:** Korisnik je prijavljen
- **Osnovni tijek:**

> 1. Korisnik odabire unos računa.
> 2. Ručno unosi podatke ili skenira račun.
> 3. Odabire trgovinu.
> 4. Sustav sprema račun.

- **Odstupanja:**

> 2.a AI ne može pročitati račun 1. Sustav traži ručni unos.
>
> 3.a Trgovina nije odabrana 1. Sustav upozorava korisnika.

UC5 – Automatska kategorizacija troškova

- **Glavni sudionik:** AI modul
- **Cilj:** Automatski kategorizirati troškove
- **Sudionici:** Baza podataka
- **Osnovni tijek:**

> 1. AI analizira račun.
> 2. Predlaže kategorije.
> 3. Sustav sprema rezultat.

- **Odstupanja:**

> 1.a AI ne može prepoznati stavke 1. Sustav prebacuje unos na ručni način.

UC6 – Evidencija troškova

- **Glavni sudionik:** Glavni ili sporedni korisnik
- **Cilj:** Pregled, uređivanje i brisanje troškova
- **Osnovni tijek:**

> 1. Korisnik otvara troškove.
> 2. Pregledava ili uređuje.
> 3. Sustav sprema izmjene.

- **Odstupanja:**

> 2.a Pokušaj uređivanja tuđeg troška bez ovlasti 1. Sustav odbija radnju.

UC7 – Dodavanje fiksnih troškova

- **Glavni sudionik:** Glavni korisnik
- **Osnovni tijek:**

> 1. Korisnik unosi iznos i opis.
> 2. Sustav sprema podatke.

- **Odstupanja:**

> 1.a Nedostaje iznos 1. Sustav traži unos iznosa.

UC8 – Usporedba potrošnje po mjesecima

- **Glavni sudionik:** Glavni ili sporedni korisnik
- **Osnovni tijek:**

> 1. Korisnik odabire mjesece.
> 2. Sustav prikazuje usporedbu.

- **Odstupanja:**

> 1.a Nedostaju podaci za usporedbu 1. Sustav prikazuje obavijest.

UC9 – Kreiranje grupe (obitelji)

- **Osnovni tijek:**

> 1. Korisnik unosi naziv grupe.
> 2. Sustav kreira grupu.

- **Odstupanja:**

> 1.a Naziv grupe već postoji 1. Sustav traži unos novog naziva.

UC10 – Slanje zahtjeva za ulazak u grupu

- **Osnovni tijek:**

> 1. Korisnik odabire grupu.
> 2. Šalje zahtjev.
> 3. Sustav sprema zahtjev.

- **Odstupanja:**

> 1.a Korisnik je već član grupe 1. Sustav blokira radnju.
>
> 1.b Zahtjev već poslan 1. Sustav obavještava korisnika.

UC11 – Odobravanje zahtjeva za ulazak u grupu

- **Osnovni tijek:**

> 1. Glavni korisnik otvara listu zahtjeva.
> 2. Odobrava ili odbija zahtjev.
> 3. Sustav šalje notifikaciju.

- **Odstupanja:**

> 1.a Zahtjev više ne postoji 1. Sustav prikazuje poruku.

UC12 – Upravljanje sporednim korisnicima

- **Osnovni tijek:**

> 1. Korisnik otvara upravljanje.
> 2. Dodaje/uređuje/uklanja dijete.
> 3. Postavlja limit.
> 4. Sustav sprema promjene.

- **Odstupanja:**

> 2.a Pokušaj uklanjanja jedinog admina 1. Sustav ne dopušta radnju.

UC13 – Evidencija zajedničkih troškova

- **Osnovni tijek:**

> 1. Korisnik unosi zajednički trošak.
> 2. Sustav ga prikazuje grupi.

- **Odstupanja:**

> 1.a Korisnik nije član grupe 1. Sustav onemogućava unos.

UC14 – ChatBot savjetnik

- **Osnovni tijek:**

> 1. Korisnik postavlja upit.
> 2. Sustav generira odgovor.

- **Odstupanja:**

> 1.a Nejasan upit 1. Sustav traži pojašnjenje.

UC15 – Grafički prikaz troškova

- **Osnovni tijek:**

> 1. Korisnik odabire graf.
> 2. Sustav prikazuje vizualni prikaz.

- **Odstupanja:**

> 1.a Nema dovoljno podataka 1. Sustav prikazuje upozorenje.

UC16 – Administracija korisnika

- **Osnovni tijek:**

> 1. Administrator otvara panel.
> 2. Pregledava korisnike i aktivnosti.

- **Odstupanja:**

> 1.a Pokušaj brisanja aktivnog admina 1. Sustav blokira radnju.

UC17 – Generiranje izvještaja

- **Osnovni tijek:**

> 1. Korisnik odabire tip izvještaja.
> 2. Sustav generira dokument.

- **Odstupanja:**

> 1.a Nedovoljni podaci 1. Sustav prikazuje obavijest.

UC18 – Pregled pohranjenih računa

- **Osnovni tijek:**

> 1. Korisnik otvara arhivu računa.
> 2. Pregledava račune.

- **Odstupanja:**

> 1.a Arhiva je prazna 1. Sustav prikazuje poruku.

UC19 – Upravljanje budžetnim ciljevima

- **Osnovni tijek:**

> 1. Korisnik definira cilj štednje.
> 2. Sustav sprema cilj.

- **Odstupanja:**

> 1.a Cilj manji od 0 1. Sustav odbija unos.

UC20 – Notifikacije

- **Osnovni tijek:**

> 1. Nastupi uvjet za slanje obavijesti.
> 2. Sustav šalje notifikaciju korisniku.

- **Odstupanja:**

> 1.a Notifikacija se ne može dostaviti 1. Sustav pohranjuje obavijest za kasniju isporuku.

UC21 – Povijest aktivnosti

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati povijest aktivnosti korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen
- **Osnovni tijek:**

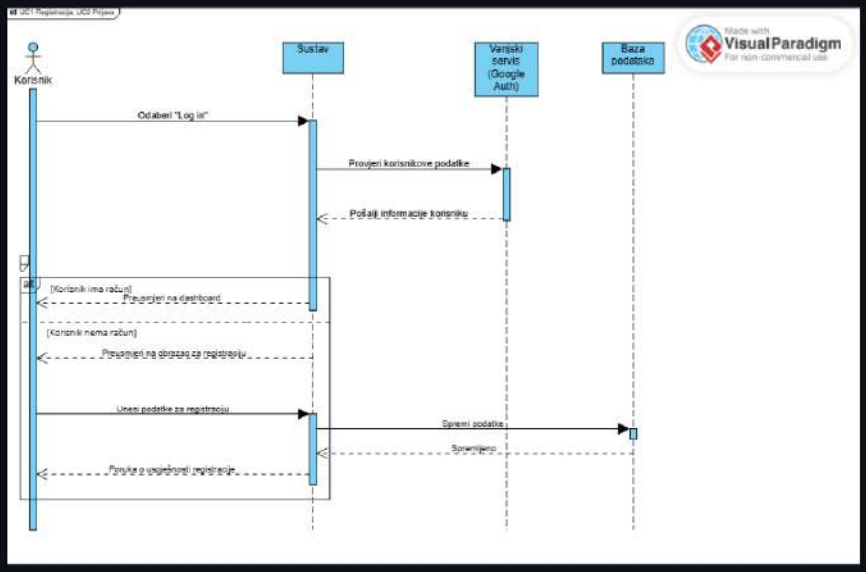
> 1. Administrator otvara "Povijest aktivnosti".
> 2. Sustav dohvaća sve aktivnosti korisnika iz baze podataka.
> 3. Administrator pregledava aktivnosti filtrirane po korisniku, datumu ili vrsti radnje.

- **Odstupanja:**

> 2.a Nema dostupnih aktivnosti 1. Sustav prikazuje poruku da povijest aktivnosti nije pronađena.
>
> 2.b Neuspjeh u dohvaćanju podataka 1. Sustav prikazuje poruku o tehničkoj pogrešci.

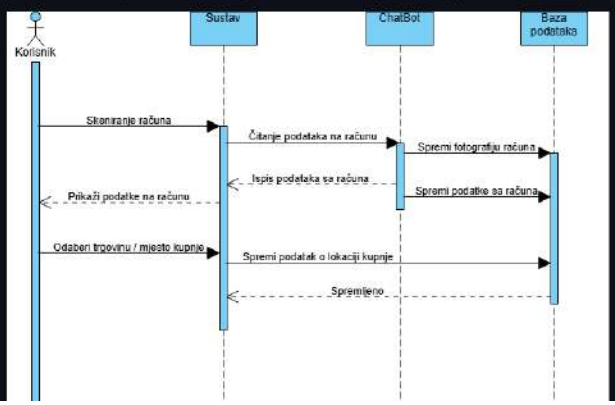
Sekvencijski dijagrami

- Opis sekvencijskog dijagrama za UC1 - Registracija i prijava
  Korisnik odabire "Log in". Sustav komunicira sa vanjskim servisom za autentifikaciju koji provjerava korisnikove podatke. Kao odgovor šalje informaciju o tome ima li taj korisnik račun ili je novi. Ako se radi o postojećem korisniku, prijava je završena i korisnika se preusmjerava na nadzornu ploču aplikacije. U slučaju da se radi o novom korisniku, preusmjerava ga se na obrazac za registraciju. On unosi tražene podatke. Ti podaci se spremaju u bazu podataka. Na kraju se šalje obavijest o uspješnosti registracije.



Slika 3.7: Sekvencijski dijagram za registraciju i prijavu

- Opis sekvencijskog dijagrama za UC4 - Unos računa
  Sekvencijski dijagram na priloženoj slici prikazuje proces kojim korisnik unosi račun u aplikaciju i odabire trgovinu ili mjesto kupnje. Nakon što korisnik pokrene skeniranje računa, sustav šalje zahtjev AI modulu (ChatBotu) da pročita podatke s računa i spremi fotografiju računa u bazu. Kada su podaci s računa očitani i fotografija spremljena, korisniku se prikazuju ti podaci putem aplikacije. U sljedećem koraku korisnik odabire trgovinu, a podatak o lokaciji kupnje sustav ponovno šalje na spremanje u bazu podataka. Sve radnje prati potvrda o uspješnom spremanju informacija. Ovaj dijagram jasno prikazuje razmjenu poruka i koraka između korisnika, sustava, AI modula i baze, te omogućuje transparentan i automatiziran unos, obradbu i pohranu računa s povezanim informacijama o kupnji.



Slika 3.8: Sekvencijski dijagram za unos računa

Provjera uključenosti funkcionalnosti

Obrazac uporabe	Funkcionalni zahtjev
UC1 – Registracija i prijava	F1 – Registracija i prijava (OAuth 2.0)
UC2 – Postavljanje mjesečnog budžeta	F2 – Postavljanje budžeta
UC3 – Upravljanje kategorijama	F3 – Upravljanje kategorijama
UC4 – Unos računa	F4 – Unos računa (ručni/skenirani) + F19 Odabir trgovine
UC5 – Automatska kategorizacija	F5 – AI kategorizacija troškova
UC6 – Evidencija troškova	F6 – Evidencija i uređivanje troškova
UC7 – Dodavanje fiksnih troškova	F7 – Fiksni troškovi
UC8 – Usporedba potrošnje	F8 – Usporedba potrošnje (srednja razina)
UC9 – Kreiranje grupe	F9 – Kreiranje grupe
UC10 – Slanje zahtjeva	F10 – Slanje zahtjeva
UC11 – Odobravanje zahtjeva	F11 – Odobravanje + upravljanje ulogama
UC12 – Upravljanje korisnicima u grupi	F12 – Uređivanje, brisanje, postavljanje limita
UC13 – Zajednička potrošnja	F13 – Evidencija zajedničkih troškova
UC14 – ChatBot	F14 – ChatBot savjetnik (visoka razina)
UC15 – Grafički prikaz	F15 – Grafički prikaz (niska razina)
UC16 – Administracija korisnika	F16 – Administracija
UC17 – Generiranje izvještaja	F17 – Izvještaji
UC18 – Pregled računa	F18 – Pohranjeni računi
UC19 – Upravljanje budžetnim ciljevima	F20 – Budžetni ciljevi glavnog korisnika
UC20 – Notifikacije	F21 – Obavijesti (zahtjevi + prekoračenje limita)
UC21 – Povijest aktivnosti	F24 – Admin vidi povijest aktivnosti aplikacije

4. Arhitektura i dizajn sustava

Filip Šturlić edited this page 2 hours ago · 6 revisions

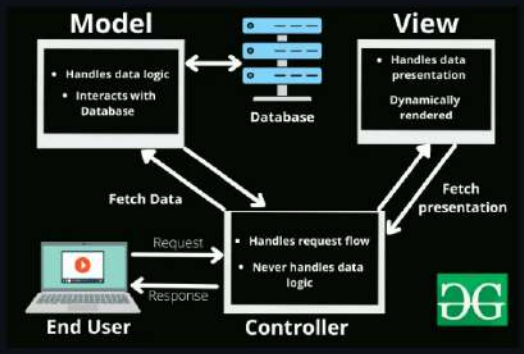
Edit New page

Arhitektura sustava

Opis arhitekture

Stil arhitekture

Za razvoj aplikacije **Računko**, koja omogućuje digitalno skeniranje računa, praćenje troškova i poticanje štednje, odabrali smo **MVC (Model-View-Controller)** arhitekturu. Ovaj arhitekturni stil omogućuje jasnu separaciju između poslovne logike, korisničkog sučelja i interakcije korisnika, što znatno olakšava održavanje, proširivost i testiranje sustava. Zbog kompleksnosti funkcionalnosti poput prepoznavanja računa s pomoću umjetne inteligencije, kategorizacije troškova i upravljanja korisničkim ulogama, MVC pruža optimalnu strukturu i preglednost koda, što je ključno za buduće nadogradnje sustava [7.].



Slika 4.1: MVC model arhitekture

Objašnjenje komponenata i njihove povezanosti

- **Model** Model predstavlja poslovnu logiku aplikacije i upravlja svim podacima koji dolaze iz baze podataka. Ovdje se nalaze objekti poput korisnika, troškova, računa i kategorija. Povezanost: Model komunicira s nadglednikom (Controller) koji mu prosljeđuje zahtjeve za ažuriranje, dok ažurirane podatke prosljeđuje pogledu (View) za prikaz. U Django okviru, Model je implementiran korištenjem Django ORM-a, koji omogućava objektno-relacijsko mapiranje i jednostavnu komunikaciju s PostgreSQL bazom.
- **View (Pogled)** Pogled prikazuje korisniku podatke u obliku tablica, grafikona i vizualizacija troškova. To je dio sustava s kojim korisnik izravno komunicira putem mobilnog sučelja. Povezanost: Pogled prima ažurirane podatke od nadglednika i dinamički ih prikazuje korisniku. U Django REST Frameworku, "pogledi" su implementirani kroz klase ViewSet ili APIView koje vraćaju JSON odgovore prema klijentu.
- **Controller (Nadglednik)** Nadglednik upravlja korisničkim zahtjevima i interakcijama. Obrađuje podatke koje korisnik unosi, šalje ih modelu na obradu i osvježava pogled novim informacijama. Povezanost: Djeluje kao posrednik između modela i pogleda, koordinira komunikaciju između njih i osigurava pravilno funkcioniranje aplikacije. U Django/DRF arhitekturi, kontroleri su implementirani unutar ViewSet-ova i serijalizatora (Serializers) koji obrađuju zahtjeve i validiraju podatke.

Podsustavi

- **Korisničko sučelje (UI):**

> Mobilno sučelje razvijeno koristeći **React Native** tehnologije, omogućava unos računa, pregled troškova i interakciju s ChatBotom.

- **API sloj:**

> Centralna točka za sve zahtjeve između klijentske aplikacije i poslužitelja. Implementiran s pomoću Django REST Frameworka, koji omogućuje jednostavnu izradu REST API-ja i integraciju s mikroservisima. Omogućava sigurnu komunikaciju i pristup podacima uz OAuth2 autentifikaciju.

- **Mikroservisi:**

> Svaka ključna funkcionalnost (npr. autentifikacija, analiza računa, kategorizacija troškova, upravljanje korisnicima) razvijena je kao zaseban servis radi modularnosti i skalabilnosti. Django aplikacije mogu biti organizirane kao odvojeni moduli koji međusobno komuniciraju putem REST API poziva.

- **Baza podataka:**

> Centralizirana relacijska baza (**PostgreSQL**) koja pohranjuje korisničke podatke, račune, troškove i financijske statistike. Django ORM koristi PostgreSQL kao primarni izvor podataka.

Preslikavanje na radnu platformu

- **Klijent:** Korisnici pristupaju aplikaciji putem mobilnih uređaja (Android/iOS). Sučelje omogućava unos i skeniranje računa, pregled mjesečnih troškova te interakciju s ChatBotom.
- **Server:** Na poslužitelju se nalazi Django backend koji obrađuje korisničke zahtjeve, upravlja autentifikacijom (putem **social-auth-app-django** i OAuth 2.0 protokola), AI modelima za prepoznavanje računa te komunikacijom s bazom podataka i API-jem.

Spremišta podataka

Clone this wiki locally

https://github.com/lukskoki/Racunko.

Centralizirana baza podataka pohranjuje:

> Podatke o korisnicima (glavni i sporedni korisnici) Evidenciju računa i pripadajućih troškova Definirane kategorije potrošnje Statistike i izvještaje o štednji i potrošnji

Mrežni protokoli

> **REST API**: za komunikaciju između klijenta i servera **HTTP/HTTPS**: za prijenos podataka i autentifikaciju korisnika **OAuth2 tokovi** koriste se za autorizaciju pristupa API resursima

Globalni upravljački tok

> Korisnik putem mobilne aplikacije unosi račun ili zahtjev za pregled troškova API prosljeđuje zahtjev odgovarajućem Django viewu (npr. ViewSet) Backend obrađuje podatke (npr. AI analiza, kategorizacija) Baza podataka se ažurira putem Django ORM-a, a rezultat se vraća korisniku u JSON formatu putem REST API-ja

Sklopovsko-programski zahtjevi

- **Hardverski:**

> Server minimalno 1 GB diska, 512 MB RAM-a, 100 millicores CPU

- **Softverski:**

> Python, Django, Django REST Framework za backend, React Native za mobilni frontend, PostgreSQL baza podataka

- **Sigurnosni:**

> OAuth 2.0 autentifikacija implementirana primjenom **social-auth-app-django** paketa i tokenizacija korisnika

- **Performanse:**

> Obrada računa (uključujući AI prepoznavanje) do 10 sekundi po zahtjevu

Razlozi za odabir MVC arhitekture

- Jasna podjela odgovornosti i lakše održavanje
- Mogućnost paralelnog rada na frontend i backend komponentama
- Jednostavno testiranje i ponovna upotreba koda
- Laka nadogradnja i dodavanje novih AI funkcionalnosti
- Bolje upravljanje kompleksnošću sustava i brže otklanjanje grešaka

Organizacija sustava s najviše razine apstrakcije

Klijent-poslužitelj

- **Klijent** Mobilna aplikacija putem koje korisnici skeniraju račune, prate potrošnju i upravljaju budžetom. Omogućuje komunikaciju s backendom s pomoću REST API poziva.
- **Poslužitelj** Django REST Framework backend sadrži poslovnu logiku, sustav autentifikacije (**social-auth-app-django + OAuth2**) i AI modele za obradu računa. Komunicira s bazom podataka i vraća rezultate klijentu u JSON formatu.

Baza podataka

Baza podataka pohranjuje sve ključne informacije:

- Korisnici i njihove uloge (glavni, sporedni, admin)
- Računi (slika, datum, iznos, kategorija)
- Fiksni i varijabilni troškovi
- Financijska statistika i izvještaji

Django ORM automatski upravlja migracijama i integritetom podataka unutar PostgreSQL baze.

Datotečni sustav

> Služi za pohranu digitalnih slika računa koje korisnici učitavaju putem aplikacije. Također čuva log datoteke o aktivnosti korisnika i eventualnim pogreškama sustava. U Django aplikaciji, slike se pohranjuju uz pomoć FileField/ImageField modela, a logovi se vode kroz ugrađeni logging modul.

Grafičko sučelje

Grafičko sučelje aplikacije izrađeno je koristeći **React Native** tehnologije. Prikazuje tablične i grafičke prikaze potrošnje, omogućuje jednostavnu navigaciju, skeniranje računa te komunikaciju s ChatBotom za financijske savjete.

Organizacija aplikacije

- **Frontend i Backend slojevi:** Frontend (React Native) prikazuje korisničko sučelje i šalje zahtjeve backendu putem REST API-ja. Backend (Django + Django REST Framework) obrađuje zahtjeve, poziva AI module za prepoznavanje teksta s računa (OCR) i komunicira s bazom podataka (PostgreSQL). Rezultati se vraćaju klijentu u JSON formatu i dinamički prikazuju.
- **MVC implementacija:** Model sadrži Django ORM definicije za korisnike, račune i troškove. Pogled prikazuje grafove i tablice troškova te omogućuje interakciju s ChatBotom. Nadglednik (ViewSet/APIView) upravlja zahtjevima korisnika, obrađuje ih kroz poslovnu logiku i ažurira model i pogled u skladu s rezultatima.

Baza podataka

Odabrana je **PostgreSQL** baza podataka zbog robusnosti, stabilnosti i poznatosti članovima razvojnog tima. Baza je relacijska i koristi tablice za pohranu podataka o korisnicima, troškovima i računima. Korišteni elementi baze:

- **Tablice** – za pohranu podataka
- **Indeksi** – za brže pretraživanje

- **Indeksi** – za ubrzani pristup
- **Pogledi i procedure** – za složenije operacije nad podacima
- **Ograničenja (constraints)** – za očuvanje integriteta podataka
- **Migracije (migrations)** – za kontrolu verzija strukture baze putem Django ORM-a

Baza podataka

Pri izboru sustava za upravljanje bazom podataka nismo naišli na potrebe koje bi zahtijevale specifične funkcionalnosti pojedinih tehnologija. Stoga smo odabrali **PostgreSQL** budući da su svi članovi tima već upoznati s njegovim radom. PostgreSQL je relacijski sustav za upravljanje bazama podataka koji se temelji na relacijskoj algebri. U takvom modelu podaci su organizirani u relacije (tablice) koje imaju nazive i unaprijed definirane atribute, dok se dohvat i manipulacija podacima provode s pomoću operacija relacijske algebre. U kontekstu našeg sustava, primarna uloga baze podataka je praćenje stanja aplikacije, uz osiguravanje pouzdanog dohvaćanja, spremanja, ažuriranja i brisanja podataka. PostgreSQL omogućuje korištenje ključnih elemenata poput tablica, indeksa, pogleda i pohranjenih procedura. Tablice služe za pohranu podataka, indeksi poboljšavaju performanse pristupa, dok pogledi i procedure omogućuju izvođenje složenijih operacija nad podacima. Osim toga, moguće je definirati ograničenja nad tablicama kako bi se očuvala dosljednost i integritet baze podataka.

Opis tablica

profile

Sadrži osnovne podatke o korisniku unutar aplikacije Računko.

Atribut	Tip podatka	Opis varijable
profile_id (PK)	INT	Jedinstveni identifikator profila
first_name	VARCHAR	Ime korisnika
last_name	VARCHAR	Prezime korisnika
email	VARCHAR	Jedinstvena e-mail adresa korisnika
pin	INT	PIN kod za prijavu
income	DECIMAL	Mjesečni prihod korisnika
budget	DECIMAL	Definirani budžet korisnika
profile_pic	TEXT	Slika profila u Base64 formatu

budget

Sadrži podatke o budžetu korisnika.

Atribut	Tip podatka	Opis varijable
budget_id (PK)	INT	Jedinstveni identifikator budžeta
amount	DECIMAL	Iznos budžeta
profile_id (FK)	INT	Referenca na korisnički profil

transaction

Sadrži podatke o svim pojedinačnim transakcijama.

Atribut	Tip podatka	Opis varijable
transaction_id (PK)	INT	Jedinstveni identifikator transakcije
name	VARCHAR	Naziv transakcije
description	TEXT	Opis transakcije
amount	DECIMAL	Iznos transakcije
date	DATE	Datum izvršavanja
category_id (FK)	INT	Kategorija transakcije
profile_id (FK)	INT	Korisnik kojem transakcija pripada

expense

Sadrži ciljane rashode koje korisnik želi pratiti (npr. "Ishrana", "Putovanja"...).

Atribut	Tip podatka	Opis varijable
expense_id (PK)	INT	Jedinstveni identifikator rashoda
allocated_amount	DECIMAL	Planirani iznos za rashod
name	VARCHAR	Naziv rashoda
description	TEXT	Opis rashoda
profile_id (FK)	INT	Korisnik kojem rashod pripada

category

Predstavlja skup prethodno definiranih kategorija troškova (npr. hrana, prijevoz itd.).

Atribut	Tip podatka	Opis varijable
category_id (PK)	INT	Jedinstveni identifikator kategorije
name	VARCHAR	Naziv kategorije

profile_id (FK)	INT	Korisnik kojem kategorija pripada

expense_category

Povezuje rashode s njihovim kategorijama (M:N relacija).

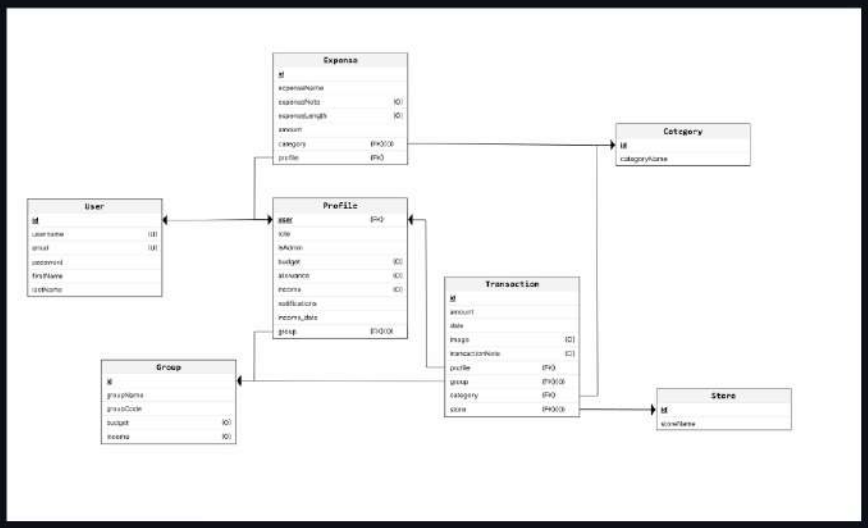
Atribut	Tip podatka	Opis varijable
expense_id (PK, FK)	INT	Referenca na rashod
category_id (PK, FK)	INT	Referenca na kategoriju

expense_transaction

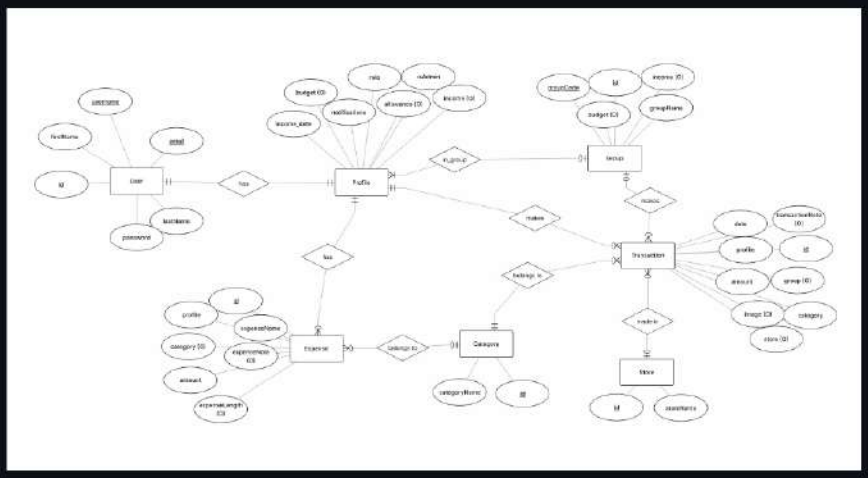
Povezuje rashode s transakcijama (M:N relacija).

Atribut	Tip podatka	Opis varijable
expense_id (PK, FK)	INT	Referenca na rashod
transaction_id (PK, FK)	INT	Referenca na transakciju

Dijagram baze podataka



Slika 4.2: REL dijagram baze podataka

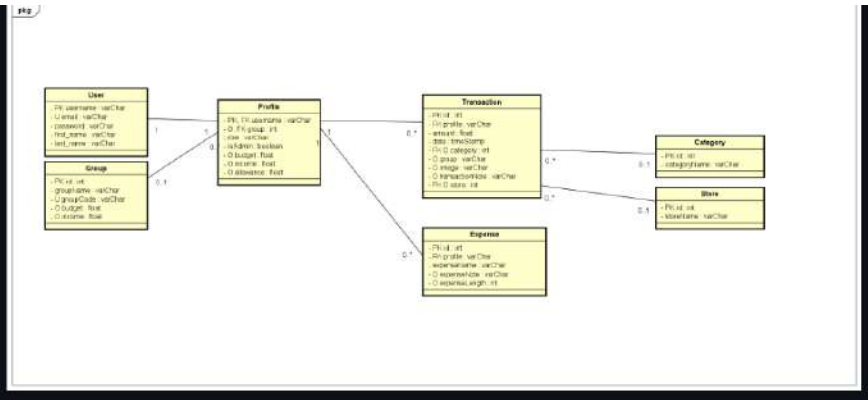


Slika 4.3: ER dijagram baze podataka

Dijagram razreda

dio 1. revizije

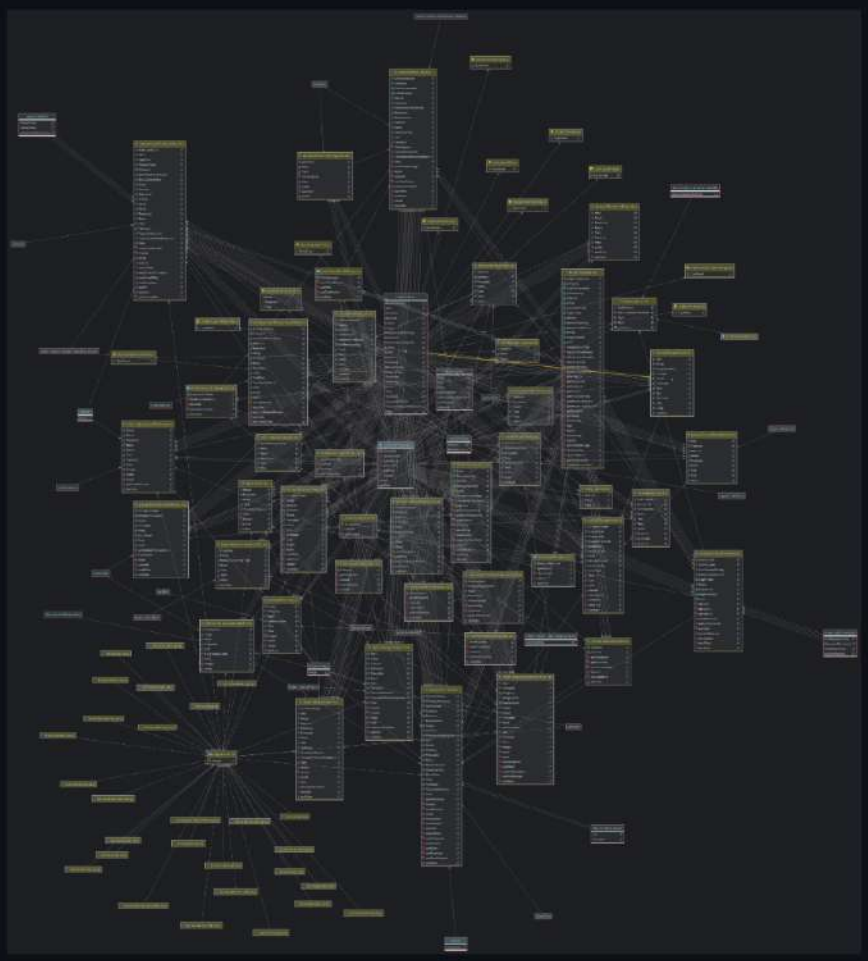
Idejni dijagram razreda sustava Računko prikazuje strukturu ključnih korisničkih entiteta i njihove međusobne odnose. Centralni entitet je klasa User, koja sadrži osnovne informacije o korisniku, uključujući korisničko ime, lozinku i osobne podatke. Klasa Profile nadograđuje klasu User dodavanjem atributa vezanih uz ulogu korisnika, pripadnost grupi te financijske parametre kao što su budžet, prihodi i džeparac. Na taj se način jasno odvaja osnovni korisnički identitet od kontekstualnih podataka sustava. Klasa Group omogućuje organizaciju korisnika u skupine radi zajedničkog upravljanja budžetom i praćenja financijskih aktivnosti. Veze među klasama definirane su asocijacijama jedan-prema-jedan ili jedan-prema-više, ovisno o prirodi podataka i poslovnim pravilima. U području financijskog poslovanja, klasa Transaction predstavlja pojedinačne novčane transakcije s atributima za iznos, datum i poveznicu na korisnički profil. Klasa Expense detaljnije opisuje troškove kroz kategorizaciju, opise i iznose, omogućujući precizno evidentiranje financijskih podataka. Klasa Profile funkcionira kao središnja veza unutar sustava, povezujući korisnike s grupama te prateći sve relevantne transakcije i troškove. Ova integracija osigurava jedinstvenu i preglednu strukturu za upravljanje osobnim i grupnim budžetima u aplikaciji Računko.



Slika 4.4: Idejni dijagram razreda

dio 2. revizije

Implementacijski dijagram razreda (Slika 4.5) prikazuje arhitekturu sustava mobilne aplikacije Računko, temeljene na klijent–poslužiteljskom modelu. U središtu sustava nalazi se klasa User, koja predstavlja korisnika aplikacije, dok klase UserRepository i AuthController omogućuju upravljanje korisničkim podacima te autentifikaciju putem vanjskih servisa (OAuth 2.0). Glavni korisnik (MainUser) upravlja mjesečnim budžetom, fiksnim troškovima i sporednim korisnicima, uz podršku klasa Budget, FixedCost i SubUser, koje omogućuju kontrolu pristupa i zajednički pregled potrošnje. Evidencija troškova ostvarena je kroz klase Receipt i Expense, dok klasa Category omogućuje njihovu kategorizaciju. Integracija umjetne inteligencije realizirana je kroz klasu AIRecognitionService, koja automatski prepoznaje stavke s računa i razvrstava ih u kategorije troškova. Analiza i prikaz podataka ostvaruju se putem klasa StatisticsService i ReportService, koje omogućuju tablični i grafički prikaz potrošnje te usporedbu po mjesecima. Administrativne funkcionalnosti implementirane su kroz klasu Admin, koja omogućuje nadzor korisnika, transakcija i rada sustava. Sve komponente povezane su jasno definiranim odnosima, čime se osigurava modularnost, preglednost i skalabilnost sustava.



Slika 4.5: Implementacijski dijagram razreda

Dinamičko ponašanje aplikacije

Dinamičko ponašanje aplikacije odnosi se na način na koji objekti u sustavu evoluiraju kroz vrijeme, uključujući prijelaze između različitih stanja. To uključuje aktivnosti, događaje, odluke i interakcije unutar aplikacije. UML dijagrami stanja omogućuju vizualizaciju tih promjena i olakšavaju razumijevanje dinamike sustava.

Razumijevanje promjena stanja neophodno je za pravilno funkcioniranje aplikacije jer pruža uvid u interakcije među objektima, komponentama i korisnicima tijekom rada sustava. Korištenjem UML dijagrama stanja i aktivnosti moguće je vizualizirati prijelaze i stanja objekata, identificirati potencijalne probleme, osigurati točnu implementaciju te poboljšati komunikaciju među članovima tima.

Zadatak:

- Dokumentirajte dva dinamička dijagrama koji prikazuju značajne ili neke složene procese u aplikaciji. Registracija i prijava korisnika nisu ključni procesi u ovom kontekstu, stoga se usmjerite na specifične, značajne procese.

UML dijagrami stanja

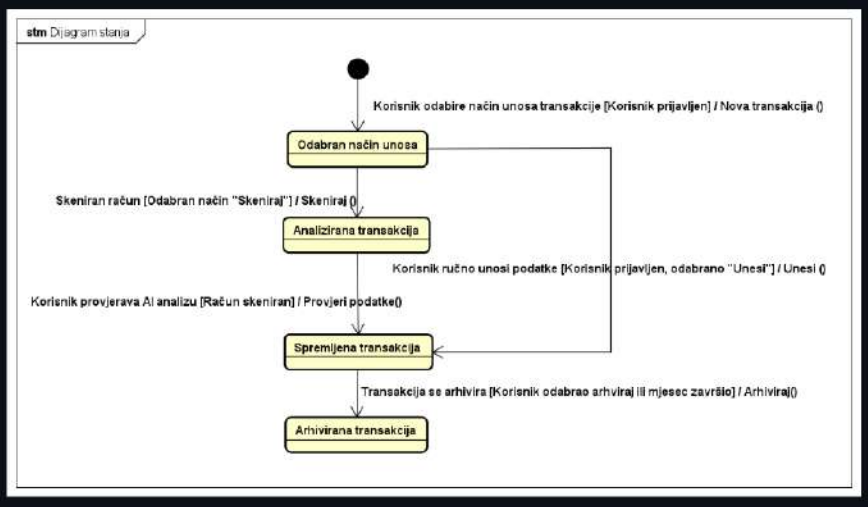
UML dijagrami stanja nužni su za razumijevanje dinamičkog ponašanja sustava. Oni jasno prikazuju promjene stanja objekata tijekom vremena ovisno o događajima i uvjetima.

Ovaj dijagram opisuje proces upravljanja transakcijom (računom ili troškom) u aplikaciji Računko, počevši od početnog stanja "Nepostojeća transakcija". Kada korisnik unese podatke ručno ili skenira račun, proces prelazi u stanje "Kreirana transakcija".

Iz stanja "Kreirana transakcija" moguće su dvije glavne akcije. Prva je analiza fotografije računa pomoću AI-ja, što vodi u stanje "Analizirana transakcija" gdje se automatski odredi iznos i kategorija (npr. kafić ili fakti). Druga je direktno spremanje bez AI-ja, ali samo ako su podaci ispravni. Ako AI ne može pronaći postojeću kategoriju, vraća se u "Kreirana transakcija" za ispravak.

U stanju "Analizirana transakcija" provjeravaju se podaci: ako su valjani (kategorija postoji i iznos točan), transakcija se sprema u bazu i povezuje s profilom ili grupom, prelazeći u stanje "Spremljena transakcija". Ovo stanje označava da je trošak dostupan za pregled u statistikama i budžetu. Ako podaci nisu valjani, vraća se na unos.

Nakon spremanja, transakcija ostaje "Spremljena" i može se koristiti za izvještaje ili chatbota savjete. Na kraju mjeseca ili po potrebi, može se arhivirati. Na taj način dijagram prikazuje cijeli životni ciklus transakcije, od unosa i AI analize, preko provjere i spremanja, do konačnog pregleda u financijama.

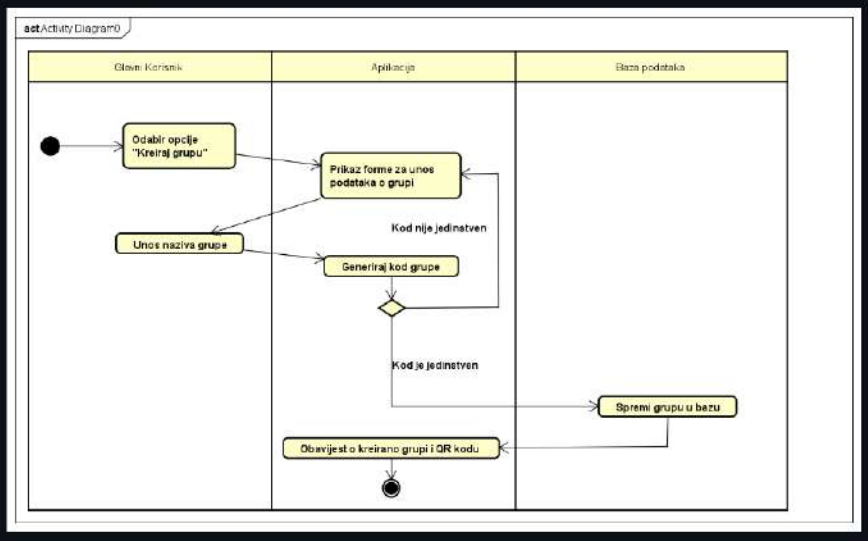


Slika 4.6: Dijagram stanja transakcije/troška

UML dijagrami aktivnosti

Dijagram aktivnosti prikazuje tijek izvršavanja određenog procesa. Osim za razumijevanje toka podataka unutar aplikacije, koristi se za analizu poslovnih procesa.

Dijagram aktivnosti na Slici 3.8 prikazuje proces kreiranja grupe obitelji u aplikaciji Računko, podijeljen na tri particije: Korisnik, Aplikacija i Baza podataka. Proces započinje u particiji Korisnik s odabirom opcije "Kreiraj grupu", gdje se u particiji Aplikacija prikazuje forma za unos naziva grupe. Nakon unosa naziva grupe u particiji Korisnik, Aplikacija generira zahtjev za kreiranje grupe.
Upravljački čvor [groupCode jedinstven?] u particiji Aplikacija provjerava jedinstvenost automatski generiranog groupCode-a u bazi podataka. Ako je groupCode jedinstven, Baza podataka sprema novu grupu s nazivom i groupCode-om, vraća podatke Aplikaciji koja generira QR kod (react-native-qrcode-svg) i prikazuje ga korisniku u particiji Korisnik. U slučaju duplikata groupCode-a, Aplikacija vraća grešku korisniku za novi pokušaj.
Proces završava prikazom QR koda s jedinstvenim groupCode-om u particiji Aplikacija, spreman za skeniranje i pridruživanje sporednih korisnika grupi obitelji za zajedničko praćenje troškova.



Slika 4.7: Dijagram aktivnosti kreiranja grupe

TimFiLAn

5. Arhitektura komponenata i razmještaja

Edit New page

Filip Šturlić edited this page 3 hours ago · 3 revisions

Arhitektura sustava predstavlja temeljni okvir za razumijevanje i implementaciju svih njegovih funkcionalnosti. U kontekstu razvojne dokumentacije aplikacija, dijagrami komponenata i razmještaja odlučujući su za prikaz povezanosti i rasporeda različitih komponenata sustava. Ovi dijagrami omogućuju sudionicima projekta razumijevanje i vizualizaciju fizičkog i logičkog dizajna sustava, uključujući interakcije između dijelova aplikacije, što je odlučujuće za efikasnu implementaciju i dugoročnu održivost sustava.

Arhitektura sustava, u kontekstu dijagrama komponenata i razmještaja, pruža uvid u strukturu i raspored ključnih dijelova aplikacije. Ovi dijagrami nisu korisni samo tijekom faza oblikovanja i implementacije, već služe i kao alati za održavanje i optimizaciju sustava u budućnosti.

Kao dio razvoja aplikacije, važno je osmisliti i dokumentirati arhitekturu sustava s naglaskom na dijagrame komponenata i razmještaja. Vaš zadatak je izraditi **dijagram komponenata** koji će jasno prikazivati ključne funkcionalne komponente aplikacije, njihovu međusobnu povezanost te sučelja za komunikaciju. Također, trebate izraditi **dijagram razmještaja** komponenata, koji treba detaljno prikazivati kako su te komponente raspoređene u infrastrukturi sustava, uključujući fizičke i virtualne resurse poput poslužitelja ili uređaja krajnjih korisnika.

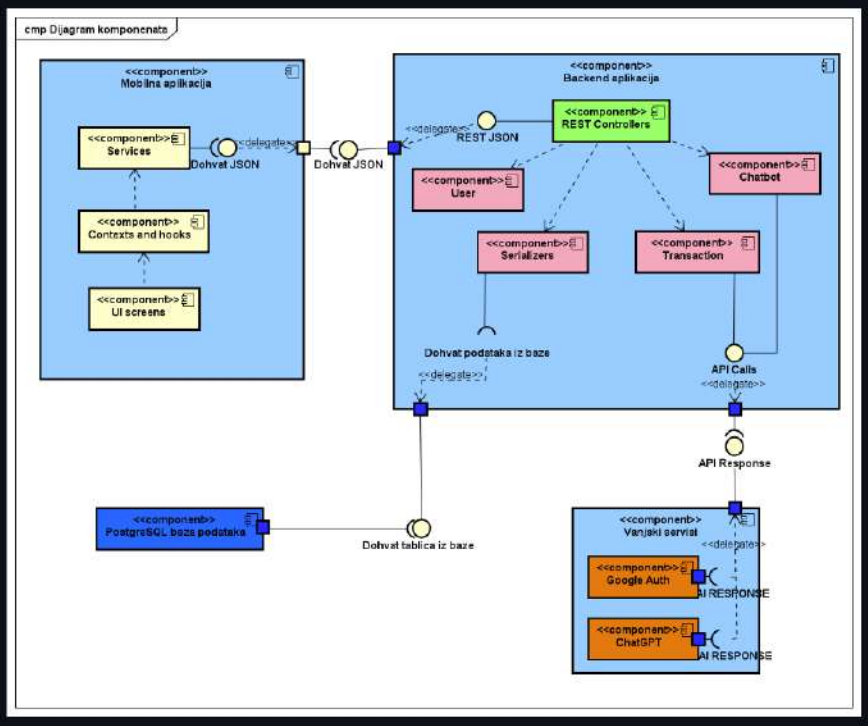
Dijagram komponenata

Komponente sustava predstavljaju bitne dijelove aplikacije koji obavljaju specifične funkcije. Svaka komponenta je autonomna jedinica s vlastitim odgovornostima, ali je povezana s drugim komponentama kako bi sustav u cjelini funkcionirao. Komponente mogu biti elementi poput modula, servisa, razreda ili paketa, te komuniciraju putem jasno definiranih sučelja.

Naš dijagram komponenata prikazuje četiri ključne komponente sustava Računko: mobilnu aplikaciju, backend aplikaciju, PostgreSQL bazu podataka i vanjske servise. Mobilna aplikacija, razvijena u React Native tehnologiji uz Expo okvir, omogućuje korisnicima upravljanje troškovima, skeniranje računa i komunikaciju s chatbotom te s backend sustavom komunicira putem REST API-ja koristeći JSON format.

Backend aplikacija razvijena je u Django okviru te obrađuje zahtjeve mobilne aplikacije, provodi poslovnu logiku i upravlja autentifikacijom korisnika. Sastoji se od modula za korisnike, transakcije i chatbot te komunicira s PostgreSQL bazom podataka radi trajne pohrane podataka.

Sustav se dodatno integrira s vanjskim servisima, kao što su Google Auth za autentifikaciju korisnika i ChatGPT API za generiranje savjeta, pri čemu backend posreduje u komunikaciji između mobilne aplikacije i tih servisa.



Slika 5.1: Dijagram komponenata

Dijagram razmještaja

UML dijagram razmještaja prikazuje fizičku ili virtualnu raspodjelu komponenata sustava unutar infrastrukture. Cilj je prikazati kako su komponente raspoređene (npr. na poslužiteljima, u okruženjima oblaka ili na uređajima krajnjih korisnika) te način komuni čije, API-ja ili drugih komunikacijskih protokola.

U našem sustavu koristi se arhitektura „klijent – poslužitelj", gdje klijentsku stranu predstavlja mobilni uređaj s mobilnom aplikacijom, a poslužiteljsku stranu centralni aplikacijski poslužitelj. Mobilna aplikacija komunicira s backend sustavom putem HTTPS protokola, osiguravajući sigurnu razmjenu podataka.

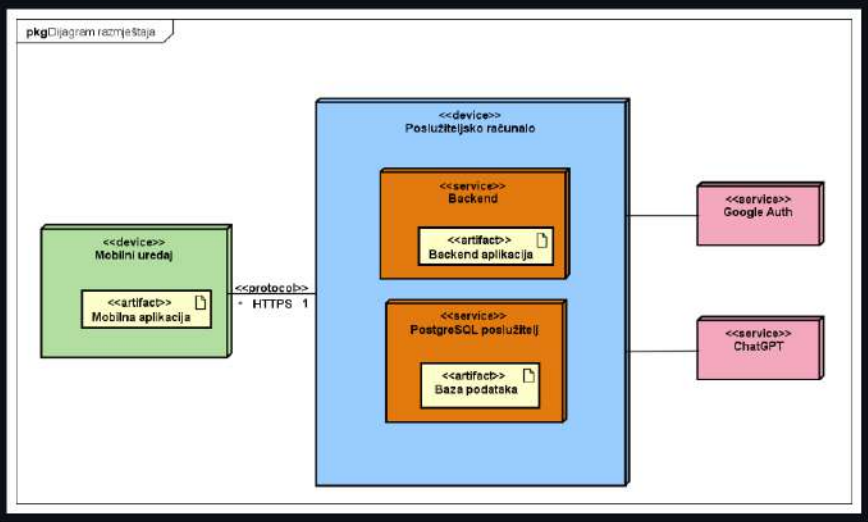
Na poslužiteljskom računalu nalazi se backend aplikacija, koja obrađuje zahtjeve mobilne aplikacije i izvršava poslovnu logiku, te PostgreSQL poslužitelj baze podataka, zadužen za trajnu pohranu podataka. Backend i baza podataka zajedno osiguravaju

Clone this wiki locally

https://github.com/lukskoki/Racunko.

ispravno funkcioniranje sustava.

Sustav se dodatno integrira s vanjskim servisima, kao što su Google Auth za autentifikaciju korisnika te ChatGPT za inteligentnu podršku i generiranje odgovora. Komunikacija s tim servisima ostvaruje se putem API-ja, pri čemu backend posreduje između mobilne aplikacije i vanjskih servisa.



Slika 5.2: Dijagram razmještaja

TimFiLAn

6. Ispitivanje programskog rješenja

Edit New page

Filip Šturlić edited this page 2 hours ago · 3 revisions

Ispitivanje komponenti

Ovo poglavlje opisuje provedena ispitivanja implementiranih funkcionalnosti na razini komponenti i sustava. Fokus je stavljen na odabir i izvedbu ispitnih slučajeva koji obuhvaćaju redovne scenarije korištenja, rubne uvjete te testiranje grešaka, uz primjenu odgovarajućih alata za provedbu testiranja.

Testiranje sustava provedeno je na dvije razine:

- **unit testiranje backend dijela sustava**, s ciljem provjere ispravnosti modela podataka, relacija i poslovne logike
- **end-to-end (E2E) testiranje mobilne aplikacije**, s ciljem provjere ispravnosti rada aplikacije iz perspektive krajnjeg korisnika

Backend je testiran pomoću unit testova unutar Django frameworka, korištenjem `TestCase` klase za provjeru modela, njihovih odnosa, ponašanja prilikom brisanja podataka i automatskog postavljanja atributa. Za backend testiranje nije korišten Selenium jer se testiraju isključivo serverske komponente (modeli i relacije), za što su unit testovi brži, izolirani i prikladniji.

Frontend dio aplikacije testiran je pomoću automatiziranih **E2E UI testova korištenjem alata Maestro**, koji omogućuje simulaciju stvarnih korisničkih interakcija nad mobilnom aplikacijom te provjeru ispravne komunikacije s backend sustavom.

Unit testiranje backend komponenti

1. Ispitni slučaj: Provjera Group modela

Funkcionalnost: Ispitujemo kreiranje i atribute Group modela u user appu.

- **Ime i klasa testa u kodu: Ime** – `test_model_group` **Klasa** – `TestModels`
- **Ulazni podaci:** Group s atributima:
  - `groupName` = "test group"
  - `groupCode` = "test code"
  - `budget` = 200.00
  - `income` = 1000.99
- **Očekivani rezultati:** `__str__` vraća `"test group"`, tip objekta je Group, atributi se podudaraju.
- **Dobiveni rezultati:** Svi assertovi prolaze – **prolaz testa.**
- **Tijek ispitivanja:**
  i. U `setUp` se kreira User i Group.
  ii. Provjerava se `__str__`, tip i atributi pomoću `assertEqual` i `assertTrue`.
  iii. Ako se podudaraju, test prolazi.

---

2. Ispitni slučaj: Provjera Profile modela

Funkcionalnost: Ispitujemo kreiranje i atribute Profile modela s relacijama.

- **Ime i klasa testa u kodu: Ime** – `test_model_profile` **Klasa** – `TestModels`
- **Ulazni podaci:** Profile s atributima:
  - `user` = testuser
  - `group` = test group
  - `role` = "parent"
  - `isAdmin` = True
  - `budget` = 300.11
  - `income` = 1000.99
  - `allowance` = 0
  - `notifications` = False
  - `income_date` = 1
- **Očekivani rezultati:** `__str__` vraća `"Profile(testuser)"`, tip Profile, atributi i relacije se podudaraju.
- **Dobiveni rezultati:** Svi assertovi prolaze – **prolaz testa.**
- **Tijek ispitivanja:**
  i. Kreira se Profile objekt.
  ii. Provjerava se `__str__`, tip i atributi (`assertEqual`, `assertTrue`, `assertFalse`).
  iii. Ako se podudaraju, test prolazi.

---

3. Ispitni slučaj: Provjera Category i Store

Funkcionalnost: Ispitujemo `__str__` metode Category i Store modela.

- **Ime i klasa testa u kodu: Ime** – `test_category_str`, `test_store_str` **Klasa** – `TestTransactionModels`

Pages 13

Find a page or section...

+ Add a custom sidebar

- **Ulazni podaci:**
  - Category: `categoryName = "Food"`
  - Store: `storeName = "Local Store"`
- **Očekivani rezultati:** `__str__` vraća nazive objekata.
- **Dobiveni rezultati:** Assertovi prolaze – **prolaz testa**.
- **Tijek ispitivanja:**
  i. U `setUp` se kreira Category i Store.
  ii. Provjerava se `__str__` pomoću `assertEqual`.
  iii. Ako se podudaraju, testovi prolaze.

---

4. Ispitni slučaj: Kreiranje Transaction

Funkcionalnost: Ispitujemo kreiranje Transaction modela, `__str__` i relacije.

- **Ime i klasa testa u kodu: Ime** – `test_transaction_create_str_and_relations` **Klasa** – `TestTransactionModels`
- **Ulazni podaci:** Transaction s atributima:
  - `profile`
  - `amount = 123.45`
  - `category = Food`
  - `group = Fam`
  - `image = receipt.jpg`
  - `transactionNote = "Lunch with kids"`
  - `store = Local Store`
- **Očekivani rezultati:** `__str__ = "Lunch with kids"`, atributi i reverse relacije (`count = 1`) su ispravni.
- **Dobiveni rezultati:** Svi assertovi prolaze – **prolaz testa.**
- **Tijek ispitivanja:**
  i. Kreira se Transaction.
  ii. Provjeravaju se `__str__`, atributi i `count()` nad relacijama.
  iii. Ako se podudaraju, test prolazi.

---

5. Ispitni slučaj: Auto-postavljanje datuma Transaction

Funkcionalnost: Ispitujemo automatsko postavljanje `date` polja (`auto_now_add`).

- **Ime i klasa testa u kodu: Ime** – `test_transaction_date_autoset` **Klasa** – `TestTransactionModels`
- **Ulazni podaci:** Transaction bez `date`, `transactionNote = "Auto date test"`.
- **Očekivani rezultati:** `date` nije `None` i nalazi se unutar 5 sekundi od `now()`.
- **Dobiveni rezultati:** Assertovi prolaze – **prolaz testa**.
- **Tijek ispitivanja:**
  i. Sprema se `before = timezone.now()`.
  ii. Kreira se Transaction.
  iii. Provjerava se `assertIsNotNone` i razlika u sekundama.
  iv. Ako je unutar granice, test prolazi.

---

6. Ispitni slučaj: On-delete ponašanje Transaction

Funkcionalnost: Ispitujemo ponašanja `SET_NULL` i `PROTECTED`.

- **Ime i klasa testa u kodu: Ime** – `test_transaction_on_delete_behaviors` **Klasa** – `TestTransactionModels`
- **Ulazni podaci:** Transaction s definiranim svim relacijama.
- **Očekivani rezultati:**
  - Brisanje Category/Store → polje postaje `NULL`
  - Brisanje Profile/Group → baca se `ProtectedError`
- **Dobiveni rezultati:** Provjere prolaze – **prolaz testa**.
- **Tijek ispitivanja:**
  i. Kreira se Transaction.
  ii. Briše se Category/Store i provjerava `assertIsNone` nakon `refresh_from_db`.
  iii. Provjerava se `assertRaises(ProtectedError)` za Profile/Group.
  iv. Ako se ponašanja podudaraju, test prolazi.

---

7. Ispitni slučaj: Kreiranje Expense

Funkcionalnost: Ispitujemo kreiranje Expense modela i reverse relacije.

- **Ime i klasa testa u kodu: Ime** – `test_expense_create_str_and_reverse` **Klasa** – `TestTransactionModels`
- **Ulazni podaci:** Expense s atributima:
  - `profile`

- `category = Food`
- `expenseName = "Food"`
- `expenseLength = 12`
- `amount = 250`

- **Očekivani rezultati:** `__str__ = "Food"` `profile.expenses.count() = 1`
- **Dobiveni rezultati:** Assertovi prolaze – **prolaz testa.**
- **Tijek ispitivanja:**
  i. Kreira se Expense.
  ii. Provjerava se `__str__` i `count()`.
  iii. Ako se podudaraju, test prolazi.

Slike rezultata

```
test_change_group_budget_success (user.tests.endpointTests.test_change_group_budget_success) ... ok
test_change_user_allowance_success (user.tests.endpointTests.test_change_user_allowance_success) ... ok
test_create_group_success (user.tests.endpointTests.test_create_group_success) ... ok
test_get_all_groups_success (user.tests.endpointTests.test_get_all_groups_success) ... ok
test_get_group_success (user.tests.endpointTests.test_get_group_success) ... ok
test_get_member_spending_success (user.tests.endpointTests.test_get_member_spending_success) ... ok
test_get_member_transactions_success (user.tests.endpointTests.test_get_member_transactions_success) ... ok
test_get_members_success (user.tests.endpointTests.test_get_members_success) ... ok
test_google_auth_success (user.tests.endpointTests.test_google_auth_success) ... ok
test_google_callback_success (user.tests.endpointTests.test_google_callback_success) ... ok
test_join_group_success (user.tests.endpointTests.test_join_group_success) ... ok
test_leave_group_success (user.tests.endpointTests.test_leave_group_success) ... ok
test_login_success (user.tests.endpointTests.test_login_success) ... ok
test_logout_success (user.tests.endpointTests.test_logout_success) ... ok
test_profile_setup_success (user.tests.endpointTests.test_profile_setup_success) ... ok
test_register_success (user.tests.endpointTests.test_register_success) ... ok
```

Slika 6.1: Rezultati ispitivanja User

```
test_chatbot_success (chatbot.tests.ChatbotEndpointTests.test_chatbot_success) ... ok
test_get_conversations_success (chatbot.tests.ChatbotEndpointTests.test_get_conversations_success) ... ok
test_get_messages_success (chatbot.tests.ChatbotEndpointTests.test_get_messages_success) ... ok
test_category_str (transaction.tests.TestTransactionModels.test_category_str) ... ok
test_expense_create_str_and_reverse (transaction.tests.TestTransactionModels.test_expense_create_str_and_reverse) ... ok
test_store_str (transaction.tests.TestTransactionModels.test_store_str) ... ok
test_transaction_create_str_and_relations (transaction.tests.TestTransactionModels.test_transaction_create_str_and_relations) ... ok
test_transaction_date_autoset (transaction.tests.TestTransactionModels.test_transaction_date_autoset) ... ok
test_transaction_on_delete_behaviors (transaction.tests.TestTransactionModels.test_transaction_on_delete_behaviors) ... ok
test_categorize_receipt_success (transaction.tests.TransactionEndpointTests.test_categorize_receipt_success) ... ok
test_create_expense_success (transaction.tests.TransactionEndpointTests.test_create_expense_success) ... ok
test_delete_expense_success (transaction.tests.TransactionEndpointTests.test_delete_expense_success) ... ok
test_get_categories_success (transaction.tests.TransactionEndpointTests.test_get_categories_success) ... ok
test_get_expenses_success (transaction.tests.TransactionEndpointTests.test_get_expenses_success) ... ok
test_manual_create_transaction_success (transaction.tests.TransactionEndpointTests.test_manual_create_transaction_success) ... ok
test_model_group (user.tests.TestModels.test_model_group) ... ok
test_model_profile (user.tests.TestModels.test_model_profile) ... ok
test_analytics_success (user.tests.endpointTests.test_analytics_success) ... ok
test_authorize_success (user.tests.endpointTests.test_authorize_success) ... http://example.com
ok
```

Slika 6.2: Rezultati ispitivanja Transaction

Maestro E2E Test Dokumentacija (Testiranje frontend sustava)

Test 1: Registracija

Ulazi:

- **Korisničko ime:** testuser123
- **Email:** test@example.com
- **Lozinka:** @TestPassword123!

Koraci ispitivanja:

1. Pokrenuti aplikaciju "com.racunko.progi"
2. Provjeriti da je vidljiv tekst "Prijava" na landing pageu
3. Kliknuti na gumb "Registracija"
4. Kliknuti na input polje za korisničko ime (testID: `signup-username-input`)
5. Unijeti tekst: `testuser123`
6. Kliknuti na input polje za email (testID: `signup-email-input`)
7. Unijeti tekst: `test@example.com`
8. Kliknuti na input polje za lozinku (testID: `signup-password-input`)
9. Unijeti tekst: `@TestPassword123!`
10. Kliknuti na gumb za registraciju (testID: `signup-button`)
11. Provjeriti da je korisnik preusmjeren na stranicu s fiksnim troškovima

Očekivani izlaz:

- Korisnik uspješno registriran
- Preusmjeravanje na profil setup stranicu gdje se prikazuje tekst "Fiksni"
- Sustav automatski prijavi korisnika nakon registracije

Dobiveni izlaz:

```
> Flow: registracija
|
|   +   Launch app "com.racunko.progi"
|   +   Assert that "Prijava" is visible
|   +   Tap on "Registracija"
|   +   Tap on id: signup-username-input
|   +   Input text testuser123
|   +   Tap on id: signup-email-input
|   +   Input text test@example.com
```

```
+ Tap on id: signup-password-input
+ Input text @TestPassword123!
+ Tap on id: signup-button
+ Assert that "Fiksni" is visible
```

Test 2: Prijava Uspješno

Ulazi:

- **Korisničko ime:** testuser123
- **Lozinka:** @TestPassword123!

Koraci ispitivanja:

1. Pokrenuti aplikaciju "com.racunko.progi"
2. Provjeriti da je vidljiv tekst "Prijava" na landing pageu
3. Kliknuti na gumb "Prijava"
4. Kliknuti na input polje za korisničko ime (testID: `username-input`)
5. Unijeti tekst: `testuser123`
6. Kliknuti na input polje za lozinku (testID: `password-input`)
7. Unijeti tekst: `@TestPassword123!`
8. Kliknuti na gumb za prijavu (testID: `login-button`)
9. Provjeriti da je korisnik preusmjeren na početnu stranicu aplikacije

Očekivani izlaz:

- Korisnik uspješno prijavljen
- Preusmjeravanje na home stranicu gdje se prikazuje tekst "Fiksni"
- Token autentifikacije spremljen u sustav

Dobiveni izlaz:

```
> Flow: prijava_uspjesno

+ Launch app "com.racunko.progi"
+ Assert that "Prijava" is visible
+ Tap on "Prijava"
+ Tap on id: username-input
+ Input text testuser123
+ Tap on id: password-input
+ Input text @TestPassword123!
+ Tap on id: login-button
+ Assert that "Fiksni" is visible
```

Test 3: Prijava Krivo

Ulazi:

- **Korisničko ime:** lukak2
- **Lozinka:** wrongpassword (namjerno kriva lozinka)

Koraci ispitivanja:

1. Pokrenuti aplikaciju "com.racunko.progi"
2. Provjeriti da je vidljiv tekst "Prijava" na landing pageu
3. Kliknuti na gumb "Prijava"
4. Kliknuti na input polje za korisničko ime (testID: `username-input`)
5. Unijeti tekst: `lukak2`
6. Kliknuti na input polje za lozinku (testID: `password-input`)
7. Unijeti tekst: `wrongpassword`
8. Kliknuti na gumb za prijavu (testID: `login-button`)
9. Provjeriti da se prikazuje poruka o grešci

Očekivani izlaz:

- Prijava neuspješna
- Prikazuje se error poruka: "Login failed"
- Korisnik ostaje na login stranici
- Sustav ne sprema token autentifikacije

Dobiveni izlaz:

```
> Flow: prijava_krivo

+ Launch app "com.racunko.progi"
+ Assert that "Prijava" is visible
+ Tap on "Prijava"
+ Tap on id: username-input
+ Input text lukak2
+ Tap on id: password-input
+ Input text wrongpassword
+ Tap on id: login-button
+ Assert that "Login failed" is visible
```

Test 4: Logout

Ulazi:

- **Korisničko ime:** testuser123
- **Lozinka:** @TestPassword123!

Koraci ispitivanja:

1. Pokrenuti aplikaciju "com.racunko.progi"
2. Provjeriti da je vidljiv tekst "Prijava" na landing pageu
3. Kliknuti na gumb "Prijava"
4. Kliknuti na input polje za korisničko ime (testID: `username-input`)
5. Unijeti tekst: `testuser123`
6. Kliknuti na input polje za lozinku (testID: `password-input`)
7. Unijeti tekst: `@TestPassword123!`
8. Kliknuti na gumb za prijavu (testID: `login-button`)
9. Provjeriti da je korisnik preusmjeren na home stranicu (prikazuje "Fiksni")
10. Kliknuti na profile tab ikonu (testID: `profile-tab-icon`)
11. Kliknuti na gumb za odjavu (testID: `logout-button`)
12. Provjeriti da je korisnik vraćen na landing stranicu

Očekivani izlaz:

- Korisnik uspješno odjavljen
- Token autentifikacije obrisan iz sustava
- Preusmjeravanje na landing page gdje se prikazuje tekst "Prijava"
- Korisnik više nema pristup zaštićenim stranicama

Dobiveni izlaz:

```
> Flow: logout
|
|   +   Launch app "com.racunko.progi"
|   +   Assert that "Prijava" is visible
|   +   Tap on "Prijava"
|   +   Tap on id: username-input
|   +   Input text testuser123
|   +   Tap on id: password-input
|   +   Input text @TestPassword123!
|   +   Tap on id: login-button
|   +   Assert that "Fiksni" is visible
|   +   Tap on id: profile-tab-icon
|   +   Tap on id: logout-button
|   +   Assert that "Prijava" is visible
```

Test 5: Navigacija Tabovi

Ulazi:

- **Korisničko ime:** testuser123
- **Lozinka:** @TestPassword123!

Koraci ispitivanja:

1. Pokrenuti aplikaciju "com.racunko.progi"
2. Provjeriti da je vidljiv tekst "Prijava" na landing pageu
3. Kliknuti na gumb "Prijava"
4. Kliknuti na input polje za korisničko ime (testID: `username-input`)
5. Unijeti tekst: `testuser123`
6. Kliknuti na input polje za lozinku (testID: `password-input`)
7. Unijeti tekst: `@TestPassword123!`
8. Kliknuti na gumb za prijavu (testID: `login-button`)
9. Provjeriti da je korisnik preusmjeren na home stranicu (prikazuje "Fiksni")
10. Kliknuti na group tab ikonu (testID: `group-tab-icon`)
11. Provjeriti da se prikazuje stranica "Grupe"
12. Kliknuti na camera tab ikonu (testID: `camera-tab-icon`)
13. Kliknuti na chatbot tab ikonu (testID: `chatbot-tab-icon`)
14. Provjeriti da se prikazuje stranica "Financijski asistent"
15. Kliknuti na profile tab ikonu (testID: `profile-tab-icon`)
16. Provjeriti da se prikazuje stranica "Moj profil"
17. Kliknuti na home tab ikonu (testID: `home-tab-icon`)
18. Provjeriti da se prikazuje home stranica (prikazuje "Fiksni")

Očekivani izlaz:

- Svi tabovi su dostupni i funkcionalni
- Svaki tab prikazuje odgovarajući sadržaj
- Navigacija između tabova radi glatko bez greški
- Korisnik može pristupiti svim sekcijama aplikacije
- Tab bar ostaje prisutan i pristupačan tijekom cijele navigacije

Dobiveni izlaz:

```
› Flow: navigacija_tabovi
  |
  |   +   Launch app "com.racunko.progi"
  |   +   Assert that "Prijava" is visible
  |   +   Tap on "Prijava"
  |   +   Tap on id: username-input
  |   +   Input text testuser123
  |   +   Tap on id: password-input
  |   +   Input text @TestPassword123!
  |   +   Tap on id: login-button
  |   +   Assert that "Fiksni" is visible
  |   +   Tap on id: group-tab-icon
  |   +   Assert that "Grupe" is visible
  |   +   Tap on id: camera-tab-icon
  |   +   Tap on id: chatbot-tab-icon
  |   +   Assert that "Financijski asistent" is visible
  |   +   Tap on id: profile-tab-icon
  |   +   Assert that "Moj profil" is visible
  |   +   Tap on id: home-tab-icon
  |   +   Assert that "Fiksni" is visible
```

Test Environment:

- **Test Framework:** Maestro
- **Platform:** Android/iOS
- **App ID:** com.racunko.progi
- **Test Type:** E2E (End-to-End)
- **Automation Level:** UI Testing

Pokretanje testova:

```
# Pojedinačni test
maestro test src/frontend/maestro/registracija.yaml
```

TimFiLAn

7. Tehnologije za implementaciju aplikacije

Filip Šturlić edited this page 2 hours ago · 3 revisions

Korištene tehnologije i alati

Komunikacija članova tima odvijala se putem aplikacija WhatsApp, Discord i Microsoft Teams, koje su korištene za svakodnevnu koordinaciju, online sastanke i razmjenu informacija tijekom razvoja projekta.

UML dijagrami izrađeni su korištenjem alata Astah (https://astah.net) te Visual Paradigm Online (https://online.visual-paradigm.com), koji omogućuju jednostavno i pregledno modeliranje sustava pomoću različitih vrsta dijagrama. Dijagram baze podataka izrađen je u online alatu ERDPlus (https://erdplus.com/). Kao razvojno okruženje korišten je **Visual Studio Code** (https://code.visualstudio.com/), u stabilnoj verziji dostupnoj u vrijeme razvoja projekta, koji uz pomoć ekstenzija omogućuje podršku za razvoj u Python, JavaScript i TypeScript jezicima, debugiranje aplikacije te rad s Git sustavom za kontrolu verzija.

Sustav za kontrolu verzija bio je Git (verzija 2.52.0), a udaljeni repozitorij projekta nalazi se na platformi GitHub, gdje je također korišten GitHub Wiki za izradu i održavanje projektne dokumentacije.

Baza podataka izrađena je u sustavu PostgreSQL (https://www.postgresql.org/), poznatom po svojoj pouzdanosti, stabilnosti i podršci za složene relacijske upite. Za povezivanje Django aplikacije s PostgreSQL bazom korišten je adapter psycopg2 (verzija 2.9.11).

Backend aplikacije razvijen je u programskom jeziku Python uz korištenje web frameworka Django (verzija 5.2.7) (https://www.djangoproject.com/), koji omogućuje strukturirani razvoj web aplikacija, upravljanje korisnicima, autentifikaciju i rad s bazom podataka putem ORM-a. Za izradu REST API-ja korišten je Django REST Framework (verzija 3.16.1), koji omogućuje komunikaciju između backend i frontend dijela aplikacije. Autentifikacija korisnika realizirana je korištenjem biblioteka **django-allauth** (verzija 65.13.0) i **django-oauth-toolkit** (verzija 3.1.0), čime je implementiran standard **OAuth 2.0**.

Frontend aplikacije razvijen je kao mobilna aplikacija korištenjem React Native-a (verzija 0.81.5) i React-a (verzija 19.1.0), uz razvojni okvir Expo (verzija 54.0.29) (https://expo.dev/), koji omogućuje jednostavan razvoj i testiranje mobilnih aplikacija za Android i iOS platforme. Za navigaciju unutar aplikacije korištene su biblioteke React Navigation, dok je stiliziranje korisničkog sučelja ostvareno pomoću*Tailwind CSS-a (verzija 3.4.17) i biblioteke NativeWind (verzija 4.2.1). Funkcionalnosti mobilnih uređaja, poput skeniranja računa i autentifikacije, implementirane su korištenjem Expo biblioteka, uključujući expo-camera i expo-auth-session.

Aplikacija je deployana na cloud platformi Heroku (https://www.heroku.com/), koja omogućuje jednostavno postavljanje i održavanje web aplikacija u oblaku.

Ispitivanje aplikacije provedeno je primjenom više razina testiranja, koje uključuju unit testiranje backend dijela sustava te end-to-end (E2E) testiranje mobilne aplikacije. Time se osigurava ispravnost poslovne logike, stabilnost sustava i korektno ponašanje aplikacije iz perspektive krajnjeg korisnika. Na ovaj način potvrđeno je da aplikacija funkcionira ispravno kako na razini pojedinih komponenti, tako i kao cjeloviti sustav.

Također su korišteni vanjski servisi:

- OpenAI API (Python SDK verzija 2.6.1, model GPT-4o) za implementaciju chatbota koji korisnicima pruža savjete o upravljanju budžetom i poticanju štednje.
- Google OAuth 2.0 za autentifikaciju korisnika putem njihovih Google računa, čime se pojednostavljuje proces registracije i prijave te osigurava dodatna razina sigurnosti.

TimFiLAn

8. Upute za puštanje u pogon

Edit | New page

Filip Šturlić edited this page 2 hours ago · 2 revisions

Upute za puštanje aplikacije u pogon

Ovaj odjeljak dokumentacije daje detaljne smjernice za instalaciju, konfiguraciju, pokretanje i administraciju aplikacije. Cilj je olakšati postavljanje aplikacije u razvojnom, ispitnom i produkcijskom okruženju.

---

1. Instalacija

U ovom poglavlju navedeni su svi preduvjeti i koraci potrebni za instalaciju aplikacije.

Preduvjeti

Za ispravno pokretanje aplikacije potrebno je imati instaliran sljedeći softver:

- Python 3.10 ili noviji
- Node.js 18 ili noviji
- npm 9 ili noviji
- PostgreSQL 14 ili noviji
- Git
- Expo CLI (`npm install -g expo-cli`)
- (Opcionalno) Docker 20.10+ za produkcijsko okruženje

Preuzimanje izvornog koda

Izvorni kod aplikacije preuzima se kloniranjem Git repozitorija:

```
git clone https://github.com/Projekt/racunko.git
cd racunko
```

Instalacija ovisnosti

Backend

```
cd backend
python -m venv .venv
```

Aktivacija virtualnog okruženja:

Windows:

```
.venv\Scripts\activate
```

Linux/Mac:

```
source .venv/bin/activate
```

Instalacija Python paketa:

```
pip install -r requirements.txt
```

Frontend

```
cd frontend
npm install
```

---

2. Postavke (Konfiguracija)

Konfiguracijske datoteke

Aplikacija koristi `.env` datoteke za pohranu osjetljivih konfiguracijskih podataka.

Backend – `.env` datoteka

U direktoriju `/backend` kreirati `.env` datoteku sa sljedećim sadržajem:

```
DEBUG=True
SECRET_KEY=your_secret_key
DATABASE_URL=postgres://user:password@localhost:5432/racunko
ALLOWED_HOSTS=127.0.0.1,localhost
```

U polje `ALLOWED_HOSTS` potrebno je dodati IP adresu ili domenu preko koje se backend poslužitelj pokreće, npr.:

Clone this wiki locally

https://github.com/lukskoki/Racunko.

```
ALLOWED_HOSTS=192.168.1.50,localhost
```

Postavke baze podataka

Potrebno je imati pokrenut PostgreSQL poslužitelj i kreiranu bazu podataka.

Primjena migracija:

```
python manage.py migrate
```

(Prema potrebi) učitavanje početnih podataka:

```
python manage.py loaddata initial_data.json
```

3. Pokretanje aplikacije

Razvojno okruženje

Pokretanje backend poslužitelja

U direktoriju `/backend` :

```
python manage.py runserver 192.168.1.50:8000
```

(umjesto IP adrese koristiti vlastitu adresu računala)

Pokretanje frontend aplikacije

U direktoriju `/frontend` :

```
npx expo start
```

Aplikaciju je moguće pokrenuti u:

- Android emulatoru
- iOS simulatoru
- Expo Go aplikaciji
- web pregledniku

Produkcijsko okruženje

Build frontend aplikacije

```
npm run build
```

Pokretanje backend poslužitelja (primjer s Gunicornom)

```
gunicorn racunko.wsgi:application
```

Provjera rada aplikacije

Nakon pokretanja, aplikacija je dostupna na:

- Backend API:

  ```
  http://<IP_ADRESA>:8000/api
  ```

- Frontend aplikacija: putem Expo aplikacije ili buildanog paketa

4. Upute za administratore

Pristup administratorskom sučelju

Backend Django admin panel dostupan je na:

```
http://<IP_ADRESA>:8000/admin
```

Pristup je omogućen korisnicima s administratorskim ovlastima.

Redovito održavanje

Sigurnosne kopije baze podataka

```
pg_dump racunko > backup.sql
```

Pregled logova

Primjer za Django:

```
tail -f logs/error.log
```

ili kod Docker okruženja:

```
docker logs racunko-backend
```

Ažuriranje aplikacije

```
git pull origin main
pip install -r requirements.txt
npm install
python manage.py migrate
gunicorn racunko.wsgi:application
```

Rješavanje problema

- Provjeriti `.env` postavke
- Pregledati backend i frontend logove
- Provjeriti dostupnost baze podataka
- Provjeriti dozvole i mrežne postavke

---

5. Deploy aplikacije na Render (Cloud platforma)

Priprema repozitorija

U repozitoriju je potrebno imati `render.yaml` datoteku ili Dockerfile.

Primjer `render.yaml` :

```yaml
services:
  - type: web
    name: racunko-backend
    env: python
    buildCommand: pip install -r requirements.txt
    startCommand: gunicorn racunko.wsgi:application
    plan: free
```

Postavljanje na Render

1. Prijaviti se na https://render.com/
2. Kreirati novi **Web Service**
3. Povezati GitHub repozitorij
4. Postaviti build i start komande
5. Dodati environment varijable:
   - `DATABASE_URL`
   - `SECRET_KEY`
   - `DEBUG=False`
6. Pokrenuti deploy

Pokretanje aplikacije

Nakon deploya aplikacija postaje dostupna putem generiranog URL-a, npr.:

```
https://racunko.onrender.com
```

Opis pristupa aplikaciji na javnom poslužitelju

Pristup aplikaciji

Korisnici aplikaciji pristupaju putem mobilne aplikacije (Android/iOS) ili web verzije putem generiranog URL-a produkcijskog poslužitelja.

Za pristup je potrebna registracija korisnika ili prijava postojećim računom.

Ograničenja

- Samo registrirani korisnici imaju pristup funkcionalnostima aplikacije
- Pristup administratorskom sučelju omogućen je isključivo administratorima
- U besplatnom cloud planu moguća su ograničenja brzine i dostupnosti

- Maksimalan broj zahtjeva može biti ograničen hosting platformom

---

Pristup administratorskom sučelju

Administratori pristupaju sučelju putem:

```
https://racunko.onrender.com/admin
```

Uz obaveznu autentifikaciju i dodijeljene administratorske ovlasti.

TimFiLAn

9. Zaključak i budući rad

Filip Šturlić edited this page 2 hours ago · 2 revisions

Edit New page

Zaključak

Projektni zadatak izrade mobilne aplikacije **Računko** uspješno je realiziran. Razvijena je funkcionalna aplikacija za evidenciju mjesečnih troškova i digitalno skeniranje računa, koja korisnicima omogućuje jednostavan, pregledan i moderan pristup upravljanju osobnim financijama. Posebna vrijednost projekta očituje se u primjeni umjetne inteligencije za automatsko prepoznavanje i kategorizaciju troškova te u integraciji chatbota za savjetovanje o upravljanju budžetom.

Projekt je realiziran kroz više faza, pri čemu je u početnoj fazi provedena analiza zahtjeva i izrađena tehnička dokumentacija koja je uključivala specifikacije funkcionalnih i nefunkcionalnih zahtjeva, UML dijagrame i arhitekturu sustava. U razvojnoj fazi implementirane su ključne funkcionalnosti, uključujući registraciju i autentifikaciju korisnika putem vanjskih servisa, upravljanje budžetom, rad s fiksnim troškovima, dodavanje sporednih korisnika, prikaz statistika te skeniranje i pohranu računa.

Posebna pažnja posvećena je sigurnosti, stabilnosti i korisničkom iskustvu, kao i testiranju sustava na razini pojedinih komponenti i cijelog sustava. Time je osigurano pouzdano funkcioniranje aplikacije i spremnost za korištenje u stvarnom okruženju. Kroz rad na projektu stekli smo vrijedna iskustva u timskom radu, upravljanju projektom te primjeni suvremenih tehnologija u razvoju mobilnih aplikacija.

Zaključno, projekt Računko značajno je doprinio našem stručnom razvoju te nas pripremio za rad na složenijim softverskim sustavima. Stečeno znanje i iskustvo omogućuju nam da s većom sigurnošću i kvalitetom pristupimo budućim projektima u području razvoja softverskih rješenja.

TimFiLAn

+ Add a custom sidebar

Clone this wiki locally

https://github.com/lukskoki/Racunko.

A. Dnevnik promjena dokumentacije

Filip Šturlić edited this page 1 hour ago · 7 revisions

Edit New page

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Luka Kovačević	10.10.2025.
0.2	Dodani funkcijski i nefunkcijski zahtjevi	Filip Šturlić	22.10.2025.
0.2.1	Ažurirani funkcijski i nefunkcijski zahtjevi	Filip Šturlić	28.10.2025.
0.3	Dodan opis arhitekture	Filip Šturlić	28.10.2025.
0.4	Dodani obrasci Uporabe	Filip Šturlić	28.10.2025.
0.5	Dodani dijagrami obrasca Uporabe	Filip Šturlić	28.10.2025.
0.6	Dodan opis projekta	Filip Šturlić	28.10.2025.
0.6.1	Ažurirani obrasci uporabe	Filip Šturlić	12.11.2025.
0.6.2	Ažuriran opis arhitekture	Filip Šturlić	12.11.2025.
0.7	Dodani sekvencijski dijagrami	Filip Šturlić	14.11.2025.
0.8	Dodan dijagram razreda, dijagram baze te opis tablica	Filip Šturlić	14.11.2025.
0.9.1	Ažuriran Readme.md, dodana licenca	Filip Šturlić	14.11.2025.
0.9.2	Ažurirani funkcijski i nefunkcijski zahtjevi, obrasci uporabe, dijagrami obrasca uporabe, opis zadatka, arhitektura sustava, literatura	Filip Šturlić	14.11.2025.
1.0	Ažuriran prikaz aktivnosti grupe za kraj 1. revizije	Luka Kovačević, Filip Šturlić	14.11.2025.
1.1	Dodan dijagram razmještaja	Filip Šturlić	15.1.2026.
1.2	Dodane korištene tehnologije	Filip Šturlić	15.1.2026.
1.3.1	Ažurirani dijagrami obrasca uporabe	Filip Šturlić	23.1.2026.
1.3.2	Ažurirani sekvencijski dijagrami	ilip Šturlić	23.1.2026.
1.4.1	Dodan dijagram komponenti	Filip Šturlić	23.1.2026.
1.4.2	Dodan dijagram implementacijski dijagram razreda	Filip Šturlić	23.1.2026.
1.5.1	Dodan dijagram stanja	Filip Šturlić	23.1.2026.
1.5.2	Dodan dijagram aktivnosti	Filip Šturlić	23.1.2026.
1.6	Dodan opis provedenih ispitivanja	Filip Šturlić	23.1.2026.
1.7.1	Dodane upute za puštanje u pogon	Luka Kovačević, Filip Šturlić	23.1.2026.
1.7.2	Ažuriran opis korištenih tehnologija	Filip Šturlić	23.1.2026.
1.8	Dodan dijagram komponenti	Filip Šturlić	23.1.2026.
1.9.1	Dodan Zaključak	Filip Šturlić	23.1.2026.
1.9.2	Ažuriran dnevnik sastajanja i tablica aktinosti grupe	Filip Šturlić	23.1.2026.
2.0	Dodan dijagram pregleda promjena	Filip Šturlić	23.1.2026.

TimFiLAn

+ Add a custom sidebar

Clone this wiki locally

https://github.com/lukskoki/Racunko.

B. Popis literature

Filip Šturlić edited this page on Nov 14, 2025 · 1 revision

Edit New page

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/ books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Astah Community, http://astah.net/editions/uml-new
7. Brainvire. (2020, October 19). Six Benefits of Using MVC Model for Effective Web Application Development. Brainvire. https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/
8. ERDPlus, (https://erdplus.com/)
9. Izvor slike 4.1, https://medium.com/@nwonahr/understanding-the-model-view-controller-mvc-pattern-97c6e057d96a
10. Primjeri sličnih aplikacija:
    10.1. Spendee, https://www.spendee.com/
    10.2. Expensify, https://www.expensify.com/
    10.3. Home Budget with Sync, https://play.google.com/store/apps/details?id=com.anishu.homebudget.full&pli=1
11. Visual Paradigm Online, https://online.visual-paradigm.com/

TimFiLAn

+ Add a custom sidebar

Clone this wiki locally

https://github.com/lukskoki/Racunko.

C. Prikaz aktivnosti grupe

Filip Šturlić edited this page 1 hour ago · 2 revisions

Edit New page

#Dnevnik sastajanja

1. sastanak

- **Datum:** 9. 10. 2025.
- **Prisustvovali:** Luka Kovacević, Luka Majcen, Luka Uršić, Timon Menalo, Ante Galić, Filip Šturlić
- **Teme sastanka:**
  - Definirali ideju i izgled cijelog sustava
  - Napravili inicijalnu podjelu posla, uz dogovor da će se po potrebi mijenjati nakon što vidimo kako se tko snašao

2. sastanak

- **Datum:** 15. 10. 2025.
- **Prisustvovali:** Luka Kovacević, Luka Majcen, Luka Uršić, Timon Menalo, Ante Galić, Filip Šturlić
- **Teme sastanka:**
  - Daljnja razrada organizacije rada: postavljanje GitHub Projects i objašnjenje rada s issues
  - Dogovor da se prvo napravi dizajn u Figmi te da se prema njemu vodi kod programiranja
  - U backendu odlučeno prvo napraviti inicijalni model baze

3. sastanak

- **Datum:** 23. 10. 2025.
- **Prisustvovali:** Luka Kovacević, Luka Majcen, Luka Uršić, Timon Menalo, Ante Galić, Filip Šturlić
- **Teme sastanka:**
  - Napravljen veliki dio dokumentacije
  - Daljnja podjela koraka za frontend i backend: login i register funkcionalnosti

4. sastanak

- **Datum:** 29. 10. 2025.
- **Prisustvovali:** Luka Kovacević, Luka Majcen, Luka Uršić, Timon Menalo, Ante Galić, Filip Šturlić
- **Teme sastanka:**
  - Backend planovi: profile setup, dodavanje transakcija, Google OAuth, povezivanje OpenAI API-ja
  - Frontend planovi: Camera view (odabir tipa unosa – slika ili ručno) te dizajn za profile setup

5. sastanak

- **Datum:** 9. 11. 2025.
- **Prisustvovali:** Luka Kovacević, Luka Majcen, Luka Uršić, Timon Menalo, Ante Galić, Filip Šturlić
- **Teme sastanka:**
  - Sve je bilo spremno za prvu predaju, pa je dogovoreno poboljšati dizajn i user experience na frontendu
  - Dodano još 3 stavke u dokumentaciji

6. sastanak

- **Datum:** 10. 12. 2025.
- **Teme sastanka:**
  - Definirani planovi:
    - Imati funkcionalnu aplikaciju do Božića
    - Nakon Božića prilagoditi dizajn i popraviti sitne bugove
  - Plan za sljedeća dva tjedna:
    - Chatbot page
    - Home page
    - Group page
    - Profile page
  - Raspodjela zadataka:

Frontend:

- Chatbot page – Luka Kovačević
- Group page – Timon Menalo
- Home page – Luka Majcen

Backend:

- Sve potrebno za chatbot – Ante Galić i Luka Kovačević
- Sve za Group i Home page – Luka Uršić i Luka Kovačević

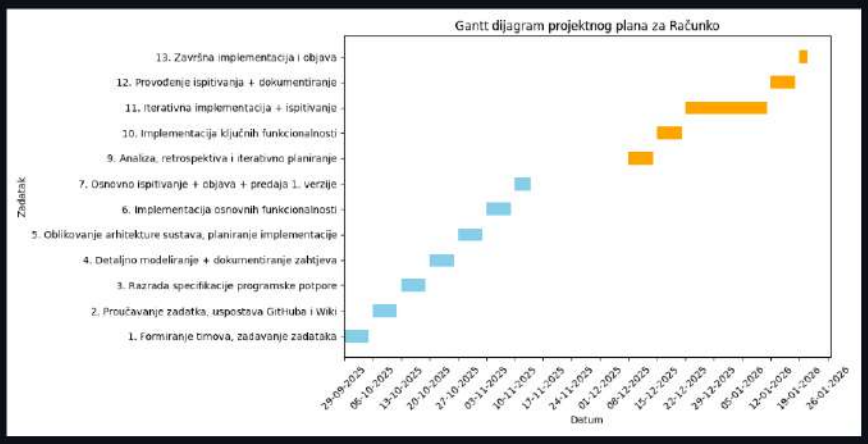
7. sastanak

- **Datum:** 5. 1. 2026.
- **Teme sastanka:**
  - Dogovorena daljnja implementacija:
    - Home tab (analytics endpoint)
    - Group page

8. sastanak

- **Datum:** 14. 1. 2026.
- **Teme sastanka:**
  - Napravljena daljnja podjela zadataka:
    - **Timon Menalo** – prikaz ekrana za člana grupe pri kliku na člana (slično kao u Home pageu)
    - **Luka Uršić** – vraćanje expenses u analytics endpoint te endpoint za dodavanje i brisanje expenses
    - **Luka Majcen** – dio Home pagea ispod glavnog prikaza za prikaz i upravljanje expensesima
    - **Ante Galić** – logout endpoint usklađen s Google autentifikacijom (ako bude potrebno)
    - **Luka Kovačević** – ispravak bugova vezanih uz chatbot i izrada testova

Plan rada



Slika C.1: Gantt Dijagram

Tablica aktivnosti

- Inicijalno postavljanje repozitorija (README, CODE_OF_CONDUCT, inicijalni backend projekt) - L. Kovačević (2h)
- Upravljanje projektom - L. Kovačević (6h)

Dokumentacija:

- Opis projektnog zadatka - F. Šturlić (2h)
- Funkcionalni i Nefunkcionalni zahtjevi - F. Šturlić (3h)
- Opis pojedinih obrazaca - F. Šturlić (2h)
- Dijagram obrazaca - F. Šturlić (3h)
- Sekvencijski dijagrami - F. Šturlić (2h)
- Arhitektura i dizajn sustava - F. Šturlić (2h)
- Baza podataka - L. Uršić (3.5h), F. Šturlić (1h)
- Dijagram razreda - L. Uršić (2h), F. Šturlić (0.5h)
- Implementacijski dijagram razreda - F. Šturlić (0.5h)
- Dijagram stanja - F. Šturlić (2h)
- Dijagram aktivnosti - F. Šturlić (2h)
- Dijagram komponenti - F. Šturlić (3h)
- Korištene tehnologije i alati - F. Šturlić (1h)
- Ispitivanje programskog rješenja - A. Galić (Backend - 3h), L. Kovačević (Frontend - 2h), F. Šturlić (1h)
- Dijagram razmještaja - F. Šturlić (1h)
- Upute za puštanje u pogon - L. Kovačević (2h)
- Dnevnik sastajanja - L. Kovačević (0.5h)
- Zaključak i budući rad - F. Šturlić (1h)
- Popis literature - F. Šturlić (0.5h)
- Prezentacija - F.Šturlić (2h)

Izrada aplikacije:

Backend:

- Modeli i serijalizeri, autentikacija (register/login), AI obrada računa, kategorije u adminu, priprema za deploy na Heroku - L. Kovačević (15h)
- Modeli baze podataka - A. Galić (3h)

- Unit testovi modela baze podataka- A. Galić (4h)
- Pisanje modela – L. Uršić (0.5h)
- Pisanje Serializer-a za modele – L. Uršić(4h)
- Pisanje API endpoint-a – L. Uršić (6h)
- Testiranje endpoint-a – L. Uršić (2h)
- Chatbot funkcionalnost - L. Kovačević (1h)
- Analitika za home page - L. Kovačević (1h)
- Različita nadogradnja - L. Kovačević (2h)
- Endpoint za logout i poziv na endpoint u frontendu - A. Galić (1h)
- Backend dio chatbota - A. Galić (10h)
- Endpointi za group page - T. Menalo (2h), L. Uršić (5h)
- Endpointi za expense - L. Uršić(3h)
- Endpointi za home page i expensove - L. Majcen (2h)

Frontend:

- Landing, login i signup ekrani, dizajn (boje, sjene, tab bar), state za inpute, validacija i osnovni API pozivi - L. Kovačević (5h)
- Dizajn na figmi - T. Menalo (10h)
- Izrada prijave i registracije - T. Menalo (15h)
- Izrada ručnog unosa - T. Menalo (25h)
- Spajanje s bazom podataka - T. Menalo (5h)
- Dizajn na figmi - L. Majcen (7h)
- Postavljene rute - L. Majcen (3h)
- Kamera tab - L. Majcen (5h)
- Profile tab- L. Majcen (2h)
- Početna stranica - L. Majcen (5h)
- Postavljanje profila nakon registracije - L. Majcen (20h)
- Povezivanje podataka za postavljanje profila sa bazom i backendom - L. Majcen (10h)
- Chatbot page - L. Kovačević (2h)
- Home page - L. Kovačević (2h)
- Razliciti bug fixed i nadogradnja - L. Kovačević (2h)
- Profile page na frontendu sa osnovnim informacijama korisnika - A. Galić (2h)
- Endpoint testovi za user, transaction i chatbot app- A. Galić (3h)
- Mali popravci u frontendu - A. Galić(~1h)
- Group page - T. Menalo (6h)
- Bug fix na frontendu - T. Menalo (2h)
- Home page frontend - L. Majcen (6h)

Frontend transakcije:

- Hook za slanje transakcija, analiza računa, available categories, popup za kategorije, poboljšanja manual unosa i bottom bara - L. Kovačević (5h)

Google prijava:

- Google login u frontend i backendu, ispravci redirecta i errora, dodavanje profile_completed polja i redirect na profile setup - L. Kovačević (6h)
- Uspostava Oauth 2.0 tokena za google authentikaciju na backendu - A. Galić (15h)

Dijagram pregleda promjena

Slika C2: Dijagram pregleda promjena

Kjučni izazovi i rješenja

Tijekom razvoja aplikacije najveći izazovi bili su usklađivanje rokova, implementacija složenih funkcionalnosti poput AI prepoznavanja računa i upravljanja grupama te sinkronizacija backend i frontend dijela. Problemi su riješeni kroz redovite timske sastanke, podjelu zadataka prema ekspertizi članova tima i stalno testiranje implementiranih rješenja. Naučili smo važnost komunikacije, planiranja i prilagodbe prioriteta, što je značajno doprinijelo uspješnom završetku projekta.

TimFiLAn