SVEUČILIŠTE/UNIVERZITET ''VITEZ'' TRAVNIK

FAKULTET **INFORMACIONIH TEHNOLOGIJA**

GODINA I CIKLUS STUDIRANJA : **III GODINA, I CIKLUS**

SMJER : **INFORMACIJSKE TEHNOLOGIJE**

**TravnikTour turistička aplikacija**

**Seminarski rad**

Travnik,2023.

1. **UVOD**

1.1. Kontekst projekta

1.2. Ciljevi i svrha

1.3. Obim projekta

1.4. Metodologija rada

2. **ANALIZA ZAHTJEVA**

2.1. Funkcionalni zahtjevi

2.2. Nefunkcionalni zahtjevi

2.3. Korisnički interfejs

3. **DIZAJN BAZE PODATAKA**

3.1. Opis baze podataka

3.2. UML dijagram objekata

4. **ARHITEKTURA SISTEMA**

4.1. Opis arhitekture

5. **TEHNIČKI ASPEKT APLIKACIJE**

5.1 Tehnologije korištene za aplikaciju

5.2. Prototip interfejsa

5.3 Održavanje aplikacije

5.4 Baza podataka

6**. ZAKLJUČAK**

**1.UVOD**

Turizam je jedan od najbrže rastućih sektora u suvremenom društvu, a tehnologija igra ključnu ulogu u oblikovanju i unaprjeđenju turističkog iskustva. U skladu s tim, ova istraživačka studija usmjerena je na analizu turističke aplikacije s ciljem istraživanja njezinih postignuća, izazova i mogućnosti za budući razvoj. Turistička aplikacija koju proučavamo je web-aplikacija koja pruža korisnicima mogućnost pristupa detaljnim informacijama o turističkim atrakcijama, smještaju, restoranima i drugim uslugama, kao i omogućuje planiranje putovanja i rezervaciju smještaja.

Ova web-aplikacija ima potencijal značajno olakšati organizaciju putovanja korisnicima diljem svijeta, poboljšati njihovo iskustvo tijekom putovanja te pružiti vrijedne preporuke temeljene na povratnim informacijama drugih putnika. Međutim, kao i svaka aplikacija, suočava se s različitim izazovima koji mogu utjecati na njezinu uspješnost i širu prihvaćenost.

U nastavku ovog rada, detaljno ćemo analizirati postignuća ove web-aplikacije, identificirati izazove s kojima se susreće te predložiti moguće pristupe za njezin budući razvoj i unaprjeđenje. Pritom ćemo posebnu pažnju posvetiti aspektima pouzdanosti informacija, korisničkom sučelju i dostupnosti na različitim geografskim područjima. Kroz ove analize, nadamo se doprinijeti razumijevanju uloge tehnologije u razvoju turizma i istaknuti ključne smjernice za daljnje inovacije u ovom području.

**1.1. Kontekst projekta**

U uvodu našeg seminarskog rada, želimo vam pružiti pregled teme istraživanja ili projekta. U ovom dijelu, objasnit ćemo kontekst teme istraživanja, istaknuti važnost i aktualnost problema koji će biti obrađen. Također, postavit ćemo svrhu rada, odnosno objasniti zašto provodimo istraživanje i navesti ciljeve koje želimo postići.

**1.2. Ciljevi i svrha**

U segmentu "Ciljevi i svrha", navest ćemo jasno definirane ciljeve i svrhu rada. Svrha rada je objašnjenje zašto provodimo istraživanje, dok su ciljevi mjerljivi i specifični rezultati koje želimo postići. Navođenje ciljeva i svrhe pomaže vama da razumijete važnost istraživanja i očekivane rezultate.

**1.3. Obim projekta**

U segmentu "Obim projekta", postavit ćemo granice i opseg istraživanja ili projekta. To uključuje definiranje specifičnih područja koja će biti obuhvaćena radom, ali i onih koja neće biti obrađena.

**1.4. Metodologija rada**

U segmentu "Metodologija rada", objasnit ćemo detaljno koju metodologiju koristimo u istraživanju ili projektu. To uključuje opisivanje metoda prikupljanja podataka ili izvora informacija koje koristimo. Također, navest ćemo analitičke ili statističke metode koji će biti primijenjeni.

**2. Analiza zahtjeva**

U današnjem digitalnom dobu, razvoj softvera je postao ključni faktor za uspjeh brojnih organizacija i industrija. Da bi softver bio koristan i ispunjavao očekivanja korisnika, važno je definirati funkcionalne zahtjeve. Funkcionalni zahtjevi predstavljaju opis sposobnosti, ponašanja i interakcija koje softverski sustav mora ispunjavati kako bi zadovoljio korisničke potrebe. Ovaj seminarski rad pruža pregled funkcionalnih zahtjeva, njihovu ulogu u dizajnu i razvoju softvera te primjere iz stvarnog svijeta.

**2.1 Funkcionalni zahtjevi**

Funkcionalni zahtjevi predstavljaju specifikaciju o tome što softverski sistem treba raditi. Oni opisuju funkcionalnosti, operacije, procese ili akcije koje sustav mora podržavati kako bi se postigao željeni rezultat. Ovi zahtjevi odražavaju očekivanja korisnika i omogućavaju programerima da razviju softver koji zadovoljava njihove potrebe. Funkcionalni zahtjevi mogu se podijeliti na nekoliko kategorija kako bi se olakšalo njihovo razumijevanje i organizacija. Evo nekoliko primjera:

Korisnički zahtjevi: Definiraju funkcionalnosti koje su bitne za korisnike softvera. To mogu biti operacije poput stvaranja korisničkog računa, prijave na sustav, pretraživanja, unos podataka i slično.

Sustavski zahtjevi: Ovi zahtjevi opisuju funkcionalnosti i interakcije između različitih dijelova softverskog sustava. Na primjer, integracija s drugim sustavima, razmjena podataka, sigurnosne mjere i performanse.

Poslovni zahtjevi: Ova kategorija zahtjeva odražava poslovne procese i ciljeve organizacije. Na primjer, automatsko generiranje izvještaja, obrada narudžbi, praćenje zaliha ili upravljanje korisničkim računima.

**2.2 Nefunkcionalni zahtjevi**

Uz funkcionalne zahtjeve, ključan je i razvoj softvera koji zadovoljava nefunkcionalne zahtjeve. Nefunkcionalni zahtjevi definiraju karakteristike, performanse i ograničenja koja softverski sustav mora ispuniti kako bi bio prihvatljiv za korisnike. Ovaj seminarski rad pruža pregled nefunkcionalnih zahtjeva, njihovu ulogu u dizajnu i razvoju softvera te primjere iz stvarnog svijeta. Nefunkcionalni zahtjevi odnose se na kvalitetu, performanse i ograničenja koja softverski sustav mora zadovoljiti. Oni se ne fokusiraju na specifične funkcionalnosti, već na karakteristike koje omogućuju dobru upotrebljivost, pouzdanost, sigurnost i performanse softvera. Nefunkcionalni zahtjevi mogu se klasificirati u nekoliko kategorija kako bi se olakšalo njihovo razumijevanje i organizacija. Evo nekoliko primjera:

Performanse: Nefunkcionalni zahtjevi performansi odnose se na brzinu, vremenski odziv, opterećenje i efikasnost softvera. To može uključivati zahtjeve poput minimalnog vremena odziva, brzine obrade podataka ili minimalnog utroška resursa.

Pouzdanost: Ovi zahtjevi odnose se na stabilnost i pouzdanost softvera. To može uključivati zahtjeve za minimalnim brojem grešaka, oporavkom od kvarova, sigurnosnim kopijama podataka i sposobnošću oporavka sustava nakon pada.

Sigurnost: Nefunkcionalni zahtjevi sigurnosti odnose se na zaštitu podataka, autentifikaciju korisnika, zaštitu od zlonamjernog softvera i sprečavanje neovlaštenog pristupa sustavu.

Upotrebljivost: Ovi zahtjevi odnose se na jednostavnost korištenja, intuitivno sučelje, pristupačnost i korisničko iskustvo. To može uključivati zahtjeve za jasnim navigacijskim sustavom, konzistentnim dizajnom korisničkog sučelja ili prilagodljivost za različite korisničke potrebe.

**2.3 Korisnički interfejs**

Korisnički interfejs (UI) igra ključnu ulogu u korisničkom iskustvu i uspješnosti softvera. Kako bi se osigurala optimalna upotrebljivost, efikasnost i zadovoljstvo korisnika, važno je definirati nefunkcionalne zahtjeve koji se odnose na dizajn korisničkog interfejsa. Ovaj seminarski rad pruža pregled nefunkcionalnih zahtjeva korisničkog interfejsa, njihovu ulogu i primjere iz stvarnog svijeta. Nefunkcionalni zahtjevi korisničkog interfejsa odnose se na aspekte koji utječu na korisničko iskustvo, uspješnost i zadovoljstvo prilikom korištenja softvera. Oni obuhvaćaju karakteristike koje doprinose upotrebljivosti, intuitivnosti, prilagodljivosti i estetici korisničkog interfejsa.

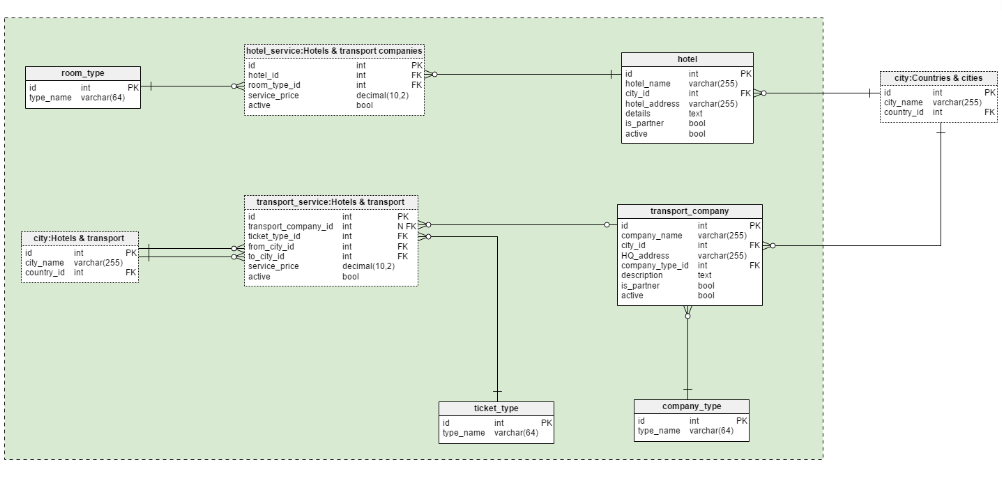
Primjeri nefunkcionalnih zahtjeva korisničkog interfejsa:

Upotrebljivost: Korisnički interfejs treba biti intuitivan i jednostavan za korištenje, bez potrebe za dugotrajnim obukama ili uputama. Nefunkcionalni zahtjevi upotrebljivosti mogu uključivati jasnu navigaciju, konzistentnost ikona i simbola, dobro organizirane korisničke elemente i kratke vremenske odzive.

Prilagodljivost: Korisnički interfejs treba omogućiti prilagodbu korisničkim preferencijama i potrebama. To može uključivati mogućnost personalizacije izgleda i rasporeda elemenata, veličine fontova, boja i jezika.

Responsivnost: Korisnički interfejs treba biti brz i odzivati se na korisnikove akcije bez zastoja ili kašnjenja. Nefunkcionalni zahtjevi responsivnosti uključuju brze promjene stanja sučelja, trenutačno prikazivanje rezultata pretrage i interaktivne animacije.

**3. Dizajn baze podataka**

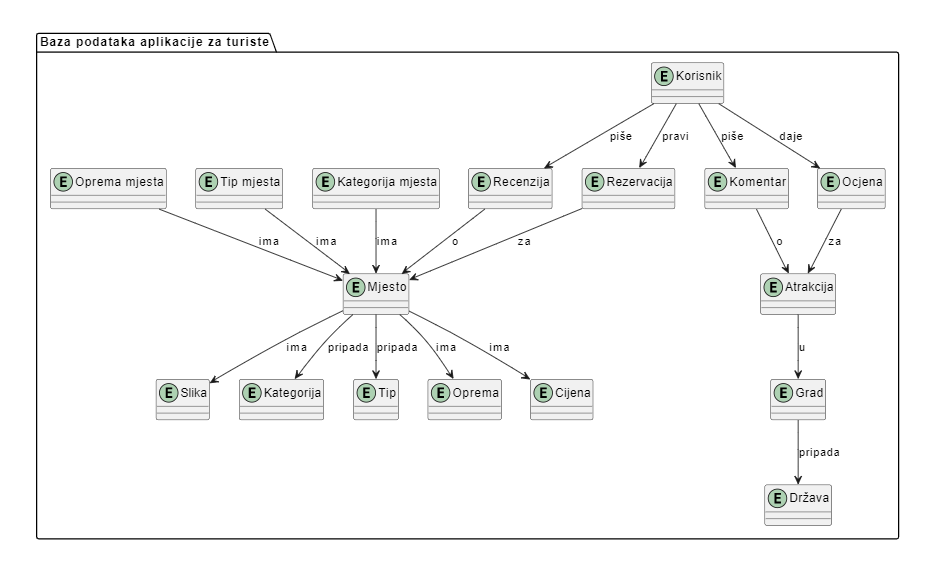
Dizajn baze podataka ima ključnu ulogu u organizaciji, pohrani i upravljanju podacima u softverskim sustavima. Kako bi baza podataka bila efikasna, pouzdana i skalabilna, važno je definirati nefunkcionalne zahtjeve koji se odnose na njezin dizajn. Ovaj seminarski rad pruža pregled nefunkcionalnih zahtjeva dizajna baze podataka, njihovu ulogu i primjere iz stvarnog svijeta. Nefunkcionalni zahtjevi dizajna baze podataka obuhvataju aspekte koji osiguravaju efikasnost, integritet, sigurnost i skalabilnost baze podataka. Oni se fokusiraju na karakteristike koje utječu na performanse, pouzdanost i upravljivost baze podataka.

Slika 1. Dizajn baze podataka za rezervisanje hotela i soba u hotelu

**3.1. Opis baze podataka**

Baza podataka za turističku aplikaciju je struktuirano skladište podataka koje omogućuje pohranjivanje, upravljanje, ažuriranje i dohvat turističkih informacija. Ova baza podataka obično koristi relacijski model podataka koji se sastoji od tablica, odnosa i atributa. Evo pregleda nekoliko ključnih entiteta i njihovih atributa koji se obično koriste u bazi podataka za turističku aplikaciju:

1. Destinacije:
   * ID destinacije
   * Naziv destinacije
   * Opis destinacije
   * Slika destinacije
   * Koordinate destinacije (geografska širina i dužina)
   * Popularne atrakcije u destinaciji
   * Smještajne jedinice u destinaciji
2. Smještajne jedinice:
   * ID smještajne jedinice
   * Naziv smještajne jedinice
   * Vrsta smještaja (hotel, apartman, kuća za odmor, itd.)
   * Adresa smještajne jedinice
   * Broj soba
   * Cijena smještaja
   * Ocjena gostiju
   * Dodatne usluge (bazen, teretana, besplatan WiFi, parking, itd.
3. Atrakcije:
   * ID atrakcije
   * Naziv atrakcije
   * Opis atrakcije
   * Vrsta atrakcije (muzej, park, spomenik, plaža, itd.)
   * Lokacija atrakcije
   * Radno vrijeme
   * Cijena ulaznice
4. Recenzije:
   * ID recenzije
   * ID korisnika
   * ID smještajne jedinice ili atrakcije
   * Ocjena
   * Komentar
5. Korisnici:
   * ID korisnika
   * Ime
   * Prezime
   * E-mail adresa
   * Korisničko ime
   * Lozinka
   * Datum registracije

3.2. UML dijagram baze podataka

Slika 2. Prototip dizajna baze podataka turističke aplikacije

 Ovaj UML dijagram objekata prikazuje strukturu baze podataka za turističku aplikaciju. Entiteti su predstavljeni kao različite klase, a veze između njih prikazuju se odgovarajućim asocijacijama. Primjerice, jedna destinacija može imati više smještajnih jedinica (1 do \*), a jedna smještajna jedinica može imati više recenzija (0 do \*). Također, jedna atrakcija može imati više recenzija (0 do \*) i tako dalje.

**4. Arhitektura sistema**

Arhitektura sustava za turističku aplikaciju obuhvaća organizaciju komponenti i njihovu međusobnu interakciju radi pružanja željenih funkcionalnosti.

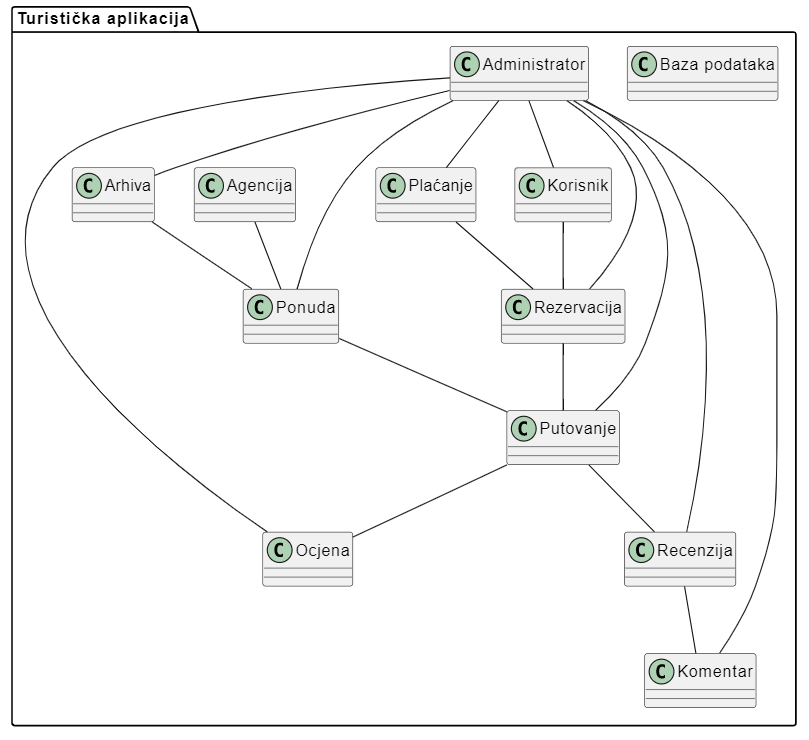
**4.1. Opis arhitekture**

Arhitektura sustava turističke aplikacije osmišljena je s ciljem pružanja optimalnog korisničkog iskustva, efikasnog upravljanja podacima i skalabilnosti kako bi se zadovoljile rastuće potrebe korisnika. Sustav je organiziran u slojeve koji zajednički djeluju kako bi osigurali glatku funkcionalnost i visoku performansu aplikacije.

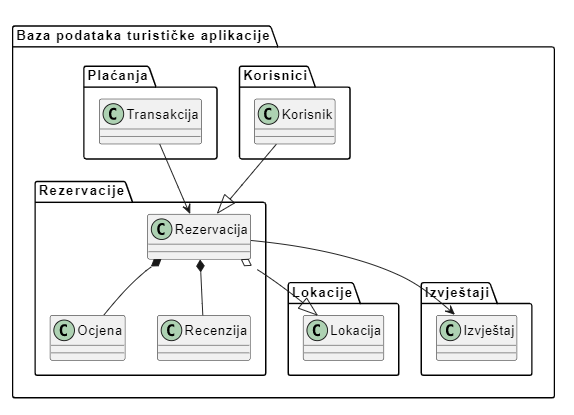
Korisničko sučelje predstavlja prvi sloj aplikacije koji korisnicima omogućuje jednostavan pristup i interakciju s različitim funkcionalnostima. Bez obzira je li to web-aplikacija ili mobilna aplikacija, korisničko sučelje mora biti intuitivno, atraktivno i prilagođeno različitim uređajima i platformama kako bi korisnici imali ugodno iskustvo prilikom korištenja aplikacije.

Poslovna logika, kao srce aplikacije, obuhvaća različite procese, pravila i algoritme koji upravljaju ključnim funkcionalnostima. Ovaj sloj osigurava obradu pretraživanja destinacija, filtriranje rezultata, generiranje preporuka za smještaj i atrakcije te upravljanje korisničkim računima i rezervacijama. Poslovna logika osigurava dosljednost i pouzdanost aplikacije, čime se korisnicima pruža optimalno iskustvo.

Baza podataka je ključna komponenta za pohranu i upravljanje podacima u sustavu turističke aplikacije. Ovisno o potrebama aplikacije, može se koristiti relacijska baza podataka koja osigurava strukturiranu pohranu podataka s podrškom za kompleksne upite i transakcije. Alternativno, ne-relacijske baze podataka mogu pružiti fleksibilnost u rukovanju nestrukturiranim podacima, kao što su podaci o ocjenama i recenzijama. Baza podataka omogućuje učinkovito pohranjivanje i dohvat podataka, osiguravajući da aplikacija ima pristup relevantnim informacijama u stvarnom vremenu.

Integracija s vanjskim uslugama i API-jima ključna je za proširenje funkcionalnosti aplikacije. Kroz integraciju s uslugama plaćanja, geolokacijom, prikazom karata i drugim vanjskim API-jima, aplikacija može pružiti dodatne informacije i usluge korisnicima. Na primjer, integracija s plaćanjem omogućuje korisnicima da sigurno obavljaju rezervacije, dok integracija s geolokacijom omogućuje prikazivanje blizine atrakcija ili smještaja korisnicima.

Slika 3. UML dijagram komponenti



Slika 4. UML dijagram paketa

**5. Tehnologije korištene za aplikaciju**

Naša Turistička aplikacija "TravnikTour" sastoji se od front-end i back-end dijelova koji su izrađeni korištenjem različitih tehnologija. U nastavku se mogu detaljno pročitati tehnologije korištene za rad sa aplikacijom

**5.1 Front-end tehnologije**

1. HTML/CSS: Za izradu osnovne strukture i stilizaciju koristili smo HTML (HyperText Markup Language) i CSS (Cascading Style Sheets) kako bismo definirali izgled aplikacije.
2. JavaScript: Koristili smo JavaScript za interaktivnost i dinamičke elemente na web stranici. Ovo uključuje obradu događaja, manipulaciju DOM (Document Object Model), animacije i komunikaciju s back-end dijelom putem AJAX-a.
3. React: Odabrali smo React kao glavni JavaScript okvir za izgradnju korisničkog sučelja. React nam omogućuje izradu komponentnog arhitekture, što olakšava upravljanje stanjem aplikacije i brže renderiranje UI-a.
4. Redux: Da bismo bolje upravljali globalnim stanjem aplikacije, koristili smo Redux. Redux omogućuje jednostavno upravljanje stanjem i olakšava praćenje podataka i njihovu sinkronizaciju s back-end dijelom.
5. Axios: Kako bismo obavljali HTTP zahtjeve prema back-endu, koristili smo Axios, popularnu JavaScript biblioteku za rad s HTTP zahtjevima. Ona nam omogućuje lakše rukovanje zahtjevima i odgovorima.

**5.2 Back-end tehnologije**

1. Node.js: Izabrali smo Node.js kao platformu za izradu back-end dijela aplikacije. Node.js je vrlo skalabilna i performantna platforma za izgradnju serverskih aplikacija koristeći JavaScript.

2. Express: Kako bismo brže izgradili web server i definirali rutiranje, koristili smo Express.js, popularni okvir za aplikacije temeljene na Node.js-u.

3. MongoDB: Kao bazu podataka, koristili smo MongoDB, NoSQL bazu podataka, koja je skalabilna i omogućuje fleksibilno spremanje podataka povezanih s turističkim informacijama o Travniku.

4. Mongoose: Za lakši rad s MongoDB bazom podataka, koristili smo Mongoose, ODM (Object Data Modeling) biblioteku. Mongoose omogućuje definiranje shema i olakšava rad s podacima.

5. JWT (JSON Web Tokens): Kako bismo omogućili autentikaciju korisnika, koristili smo JWT. Nakon što se korisnik uspješno prijavi, dobiva JWT koji se koristi za provjeru autentičnosti u kasnijim zahtjevima.

6. RESTful API: Razvili smo RESTful API za komunikaciju između front-end i back-end dijela aplikacije. To omogućuje jasno definirane endpointe i HTTP metode za pristup podacima.

Ovom kombinacijom front-end i back-end tehnologija, stvorili smo "TravnikTour" turističku aplikaciju koja omogućuje korisnicima pregledavanje i istraživanje različitih turističkih atrakcija i informacija o Travniku, a istovremeno pruža intuitivno i interaktivno korisničko iskustvo.

**5.3. Prototip korisničkog interfejsa**

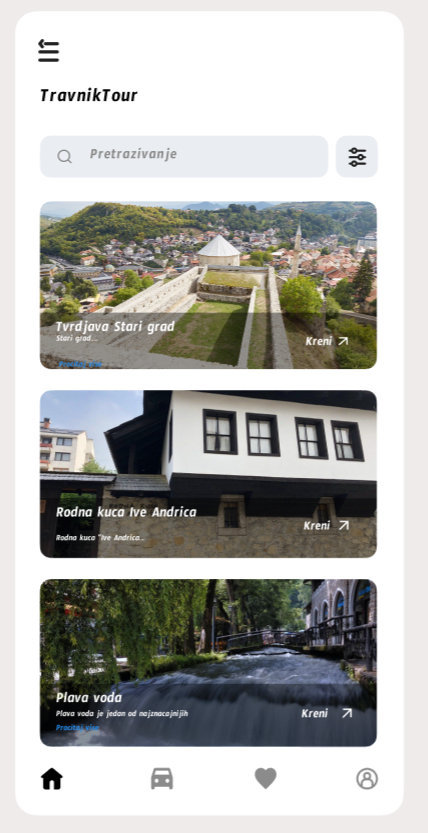
Uvodni dio arhitekture sustava turističke aplikacije ima ključnu ulogu u osmišljavanju optimalne strukture koja omogućuje uspješno ostvarenje poslovnih ciljeva i pružanje visokokvalitetnih korisničkih iskustava. Dizajniranje efikasne arhitekture za turističku aplikaciju zahtijeva pažljivo razmatranje kako bi se zadovoljile specifične potrebe industrije turizma, korisnika i poslovnih zahtjeva. Turistička aplikacija treba biti dizajnirana s fokusom na korisnike i njihove potrebe prilikom planiranja i rezerviranja putovanja. Ključni aspekti dizajna uključuju intuitivno korisničko sučelje, brz odziv aplikacije, jasnu i relevantnu prezentaciju informacija te personalizaciju kako bi se korisnicima pružilo jedinstveno iskustvo. Osim toga, arhitektura turističke aplikacije treba biti skalabilna i fleksibilna kako bi mogla podržati rastući broj korisnika i promjenjive zahtjeve industrije. To zahtijeva pravilno upravljanje resursima, optimalnu organizaciju podataka i sposobnost integracije s drugim sustavima i uslugama. Sigurnost je također ključni aspekt dizajna arhitekture turističke aplikacije. Osiguravanje sigurnosti korisničkih podataka, transakcija i komunikacije ključno je za izgradnju povjerenja korisnika. Implementacija sigurnosnih protokola, šifriranje podataka i mehanizama autentifikacije i autorizacije trebaju biti sastavni dio arhitekture kako bi se osigurala zaštita privatnosti i spriječile zloupotrebe.

**5.4. Održavanje aplikacije**

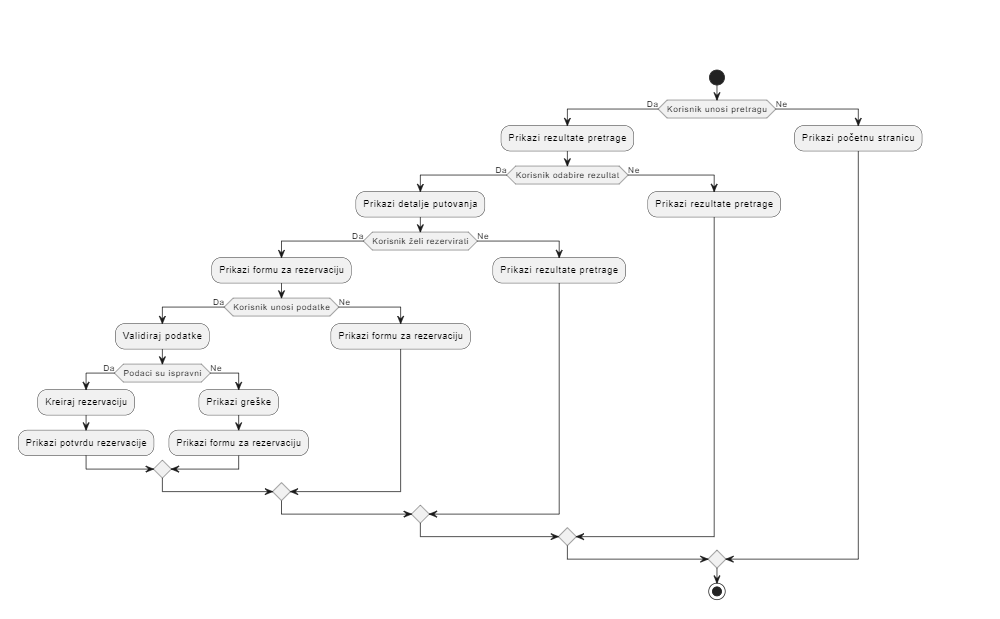
Turistička aplikacija redovito se podvrgava testiranju kako bi se osigurala njena funkcionalnost, performanse i sigurnost. Testiranje uključuje jedinice testiranja, integracijsko testiranje, sustavno testiranje i testiranje korisničkog sučelja. Redovita održavanja, uključujući zakrpe, poboljšanja sigurnosti i nadogradnje, provode se kako bi se osigurala stabilnost i pouzdanost aplikacije.

**5.5 Baza podataka**

Turistička aplikacija koristi relacijsku bazu podataka za pohranu i upravljanje informacijama o turističkim atrakcijama, smještaju, restoranima i drugim uslugama. Relacijska baza omogućuje učinkovito organiziranje podataka i omogućuje složene upite koji pružaju detaljne informacije korisnicima. Također, korištenje baze podataka omogućuje siguran i pouzdan pristup podacima korisnicima diljem svijeta.



Slika 5. Prototip dizajna turističke aplikacije



Slika 6. UML dijagram aktivnosti aplikacije

**6. Zaključak**

U ovom seminarskom radu detaljno smo analizirali turističku aplikaciju s ciljem istraživanja njezinih postignuća, izazova i mogućnosti za budući razvoj. Aplikacija bi se istaknula pružajući korisnicima pristup detaljnim informacijama o turističkim atrakcijama, smještaju, restoranima i drugim uslugama, olakšavajući planiranje putovanja i rezervaciju smještaja. Također, omogućila bi korisnicima praćenje preporuka drugih korisnika, što je poboljšalo njihovo iskustvo putovanja.

Unatoč postignućima, identificirali smo nekoliko izazova s kojima se aplikacija suočila. Nedostatak pouzdanih podataka predstavlja potencijalni izazov koji zahtijeva rješavanje kako bi se osigurala tačnost informacija koje korisnici dobivaju. Poteškoće u korisničkom sučelju mogle bi negativno utjecati na korisničko iskustvo i zahtijevaju poboljšanja kako bi aplikacija bila intuitivnija i lakša za korištenje. Također, ograničena dostupnost aplikacije na određenim geografskim područjima predstavlja prepreku širem prihvatu i uspjehu aplikacije.U biti turistička aplikacija ima potencijal za uspjeh i doprinos razvoju turizma, ali za to će biti potrebno kontinuirano unapređenje i prilagodba novim izazovima tržišta. Programeri koji budu radili trebaju pažljivo pratiti povratne informacije korisnika te ulagati u inovacije kako bi osigurali da aplikacija ostane relevantna i konkurentna na tržištu.