



UNIVERZITET U NOVOM
SADU
FAKULTET TEHNIČKIH
NAUKA U NOVOM SADU



Danilo Novaković

Aplikacija za rezervaciju apartmana

Projektni zadatak
- WEB Programiranje -

Novi Sad, 2019.

1 UVOD

Tema ovog projekta je realizacija web aplikacije koja podržava rezervaciju apartmana (kao “Airbnb” aplikacija). Potrebno je podržati tri grupe (uloge) **korisnika**: Gost, Domaćin i Administrator, gde je glavna uloga Gosta da rezerviše apartman, Domaćina da izdaje apartman i Administratora da kontroliše rad sajta i korisnike (ima mogućnost da briše, blokira korisnike itd.).

Cilj ovog projekta je da omogući ljudima način da rezervišu apartmane preko interneta i samim tim smanje brigu i stres prilikom putovanja.

U daljem radu će biti opisane tehnike i tehnologije za razvoj sistema, njegova specifikacija i implementacija, kao i prikaz implementiranog rešenja i uputstvo njegovog korišćenja.

2 KORIŠĆENE TEHNIKE I TEHNOLOGIJE

Struktura aplikacije je sačinjena od klijent strane, server strane i baze podataka. Za razvoj klijentske strane je upotrebljen *React*, a za server *ASP.NET Core* u C# programskom jeziku. Kada je u pitanju baza podataka, korišćen je *EF Core* ORM, što omogućava fleksibilnost prilikom odabira baze (potrebno je samo promeniti Connection String u konfiguraciji). U narednim potpoglavljima je svaka od ovih tehnologija detaljno opisana.

2.1 REST

REST - Representational State Transfer, je definisan 2000. godine u doktorskoj disertaciji koju je napisao *Roy Fielding*. On definiše principe softverske arhitekture, ignorišući detalje implementacije i sintakse i daje fokus ulogama komponenti. Najvažniji principi u *REST*-u su:

- Resurs – svaki entitet na vebu
- Adresa resursa – svaki resurs je indetifikovan svojim *URI*-jem
- Operacija – obavlja se nad resursom

U vebu se *REST*, odnosno *RESTful API* oslanja na *HTTP* protokol, pa se samim tim operacije nad resursima svode na *HTTP* metode (*GET*, *POST*, *PUT*, *DELETE* su samo neke od njih).

U *REST*-u klijent i server moraju biti odvojeni i imati mogućnost da funkcionišu nezavisno jedan od drugog. Ovo podrazumeva da izmena jednog od ta dva dela, ne sme kao posledicu da ima izmenu drugog.

Protokol koji se koristi je stateless, odnosno komunikacija je skup nezavisnih pojedinačnih događaja. To znači da se *RESTful API* ne oslanja na podatke koji se nalaze u sesiji, već samo na one podatke koji se nalaze u svakom pojedinačnom zahtevu.

2.2 ASP.NET Core

Za implementaciju *REST API*-a korišćen je *ASP.NET Core*. To je open-source cross platform web development framework napisan za .NET platformu.

Za razliku od njegovog pretka *ASP.NET Frameworka* koji se oslanjao na “convention over configuration”, *ASP.NET Core* nam pruža mnogo veću slobodu prilikom konfiguracije aplikacije, u toj meri da vidimo *Main* funkciju. Ova vrsta fleksibilnosti je ostvarena upotrebom šablona Graditelj (Builder) čiji je glavni cilj olakšana konstrukcija kompleksnih objekata.

```

public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}

```

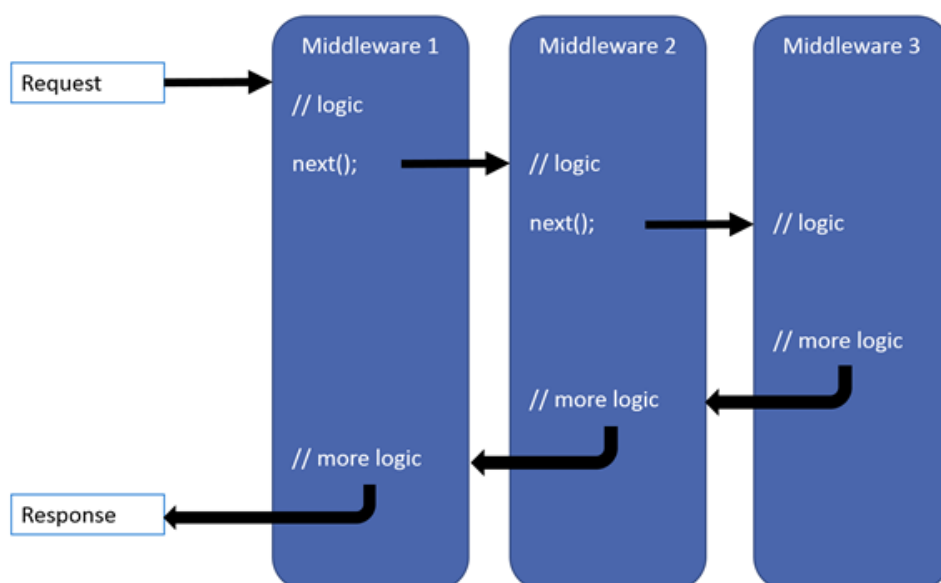
Klasa koja je od najvećeg interesa jeste ona definisana pod `.UseStartup<MOJA_KLASA>` metodom (po defaultu se ta klasa naziva `Startup`). Očekivane dve metode su `ConfigureServices` za registraciju servisa za dependency injection, kao i `Configure` za konfiguraciju HTTP request pipeline (tj. middleware-a)

```

public class Startup
{
    // Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
        ...
    }

    // Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app)
    {
        ...
    }
}

```



SLIKA 2.1 – MIDDLEWARE PIPELINE

Neki od osnovnih podržanih middleware-a su:

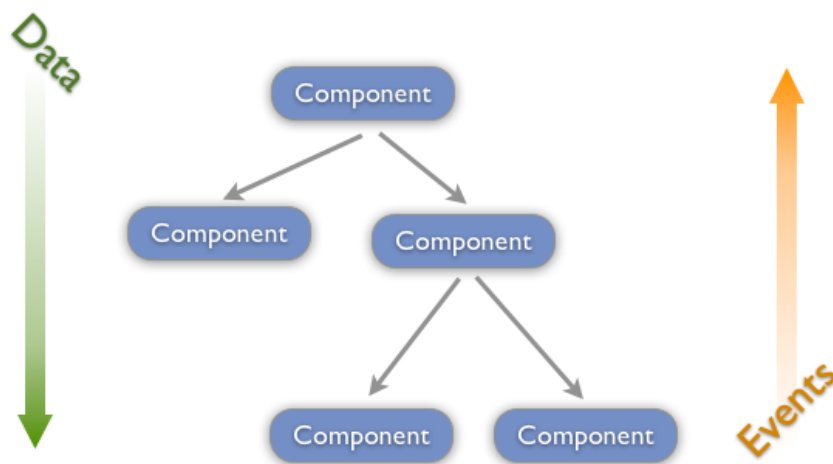
- *UseHttpsRedirection* – Preusmerava HTTP zahteve u HTTPS
- *UseStaticFiles* – Omogućava servis statičkih fajlova (npr. slika)
- *UseAuthentication* – Pokušava da autentifikuje korisnike pre nego što dođu do resursa
- *UseMvc* – Dodaje podršku MVC arhitekture (sa attribute routing-om). Ovo je middleware koji omogućuje implementaciju REST arhitekture putem API kontrolera.
- *UseSpa* - Omogućuje servis Single Page Aplikacija (u ovom slučaju klijentske aplikacije implementirane putem *React*-a (više o njemu u nastavku teksta))

2.3 React

React (poznat i kao **React.js** ili **ReactJS**) je Javaskript okrućenje otvorenog koda koja obezbeđuje pregled podataka zapisanih preko [HTMLa](#). React pregledi su obično obezbeđeni korišćenjem komponenti koje sadrže dodatne komponente definisane

kao prilagođene HTML oznake. React obezbeđuje programeru model u kojem podkomponente ne mogu direktno da utiču na spoljašnje komponente, efikasno ažuriranje HTML dokumenta pri promeni podataka i jasno razdvajanje komponenti na današnjim jednostraničnim aplikacijama.

2.3.1 Jednosmerni tok podataka

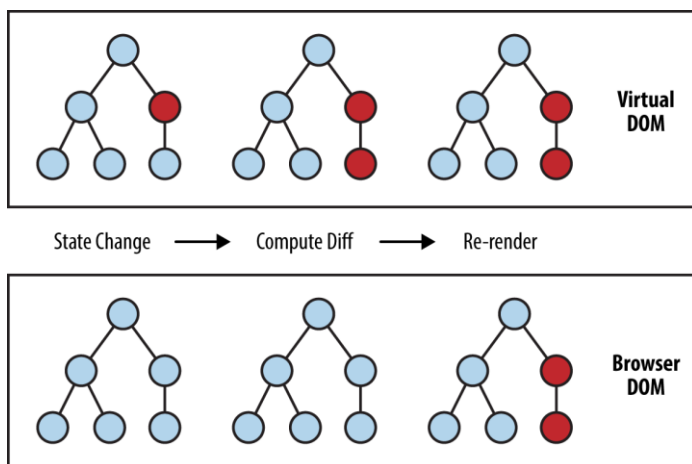


SLIKA 2.2 - TOK PODATAKA U REACT-U

Svojstva, skup nepromenljiva vrednosti, prosleđeni su prikazivaču komponenti kao svojstvo u njegovim HTML oznakama. Komponente ne mogu direktno da promene bilo koje svojstvo koje im je prosleđeno ali može biti prosleđena opozivajuća funkcija da promeni vrednosti. Ovaj mehanizam je izražen kao „svojstva idu dole; akcije gore”.

Na primer, komponenta kolica za kupovinu može da uključuje više komponenti proizvoda. Vizuelizacija proizvoda koristi samo svojstva koja su im prosleđena i to ne može da utiče na ukupnu vrednost kolica za kupovinu. Međutim, proizvod može biti prosleđen opozivajućoj funkciji kao svojstvo koje bi bilo pozvano kada aktiviramo dugme „obriši ovaj proizvod” i tada ta funkcija utiče na ukupan račun.

2.3.2 Virtuelni DOM



SLIKA 2.3 - VIRTUALENI DOM

Još jedna specifičnost je da se koristi virtuelni DOM. React održava u memoriji keš podataka, izračunava aktuelne razlike, a zatim efikasno ažurira pregledačev DOM. Ovo omogućava programerima da pišu kod kao da se cela strana osvežava pri svakoj promeni, dok React biblioteka prikazuje samo podkomponente koje su zaista promenjene.

2.3.3 JSX

Komponente React-a su obično zapisane u JSX-u, a Javaskript sintaksno proširenje omogućava lako navođenje HTML-a i korišćenje sintakse HTML oznaka prikazanim podkomponentama. HTML sintaksa obrađena u Javaskriptu poziva React biblioteku. Programeri takođe mogu da pišu i u čistom Javaskriptu. JSX je sličan fejsbukovom sintaksnom proširenju za PHP, XHP.

JSX:

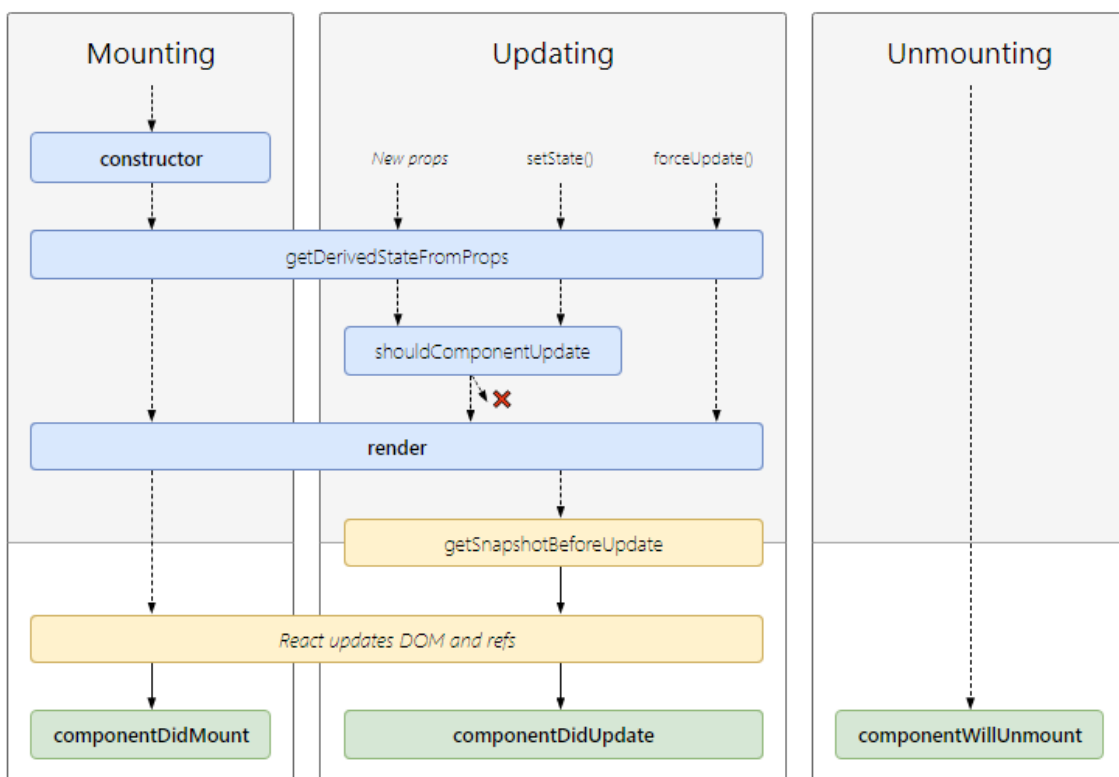
```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

JavaScript (preveden JSX):

```
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

2.3.4 Lifecycle metode

Svaka komponenta sadrži više “lifecycle” metoda koje korisnik može da implementira po potrebi. Neke su korišćene **češće** od ostalih.



SLIKA 2.4 - LIFECYCLE METODE U REACTU

3 SPECIFIKACIJA

Potrebno je realizovati veb aplikaciju za sistem koji podržava rezervacije apartmana (kao Airbnb aplikacija). Aplikaciju koriste tri grupe (uloge) **korisnika**: Gost, Domaćin i Administrator.

3.1 Dijagram slučajeva korišćenja

Aplikacija je osmišljena tako da podržava tri tipa korisnika (isključujući slučaj kada korisnik nije registrovan):

- Korisnik administrator koji upravlja podacima
- Korisnik domaćin koji izdaje apartmane
- Korisnik gost koji rezerviše apartmane

Uloga **administratora** se svodi na to da može da menja postojeće, briše ili dodaje nove entitete (apartmane, sadržaje apartmana, domaćine,...). On je zadužen za to da svi podaci u sistemu budu ispravni i tačni u svakom momentu. Administratori se učitavaju na početku rada programa i ne mogu se naknadno dodavati.

Dostupne aktivnosti za **domaćina** (*Host*) su sledeće:

- Dodavanje, brisanje i modifikacija apartmana
- Mogućnost odbijanja, prihvatanja i markriranje rezervacija kao “završene” nakon završnog noćenja gosta
- Prikaz svih rezervacija nad svojim apartmanima
- Prikaz svojih apartmana
- Odluku koji komentari da se prikazuju na njegovom apartmanu

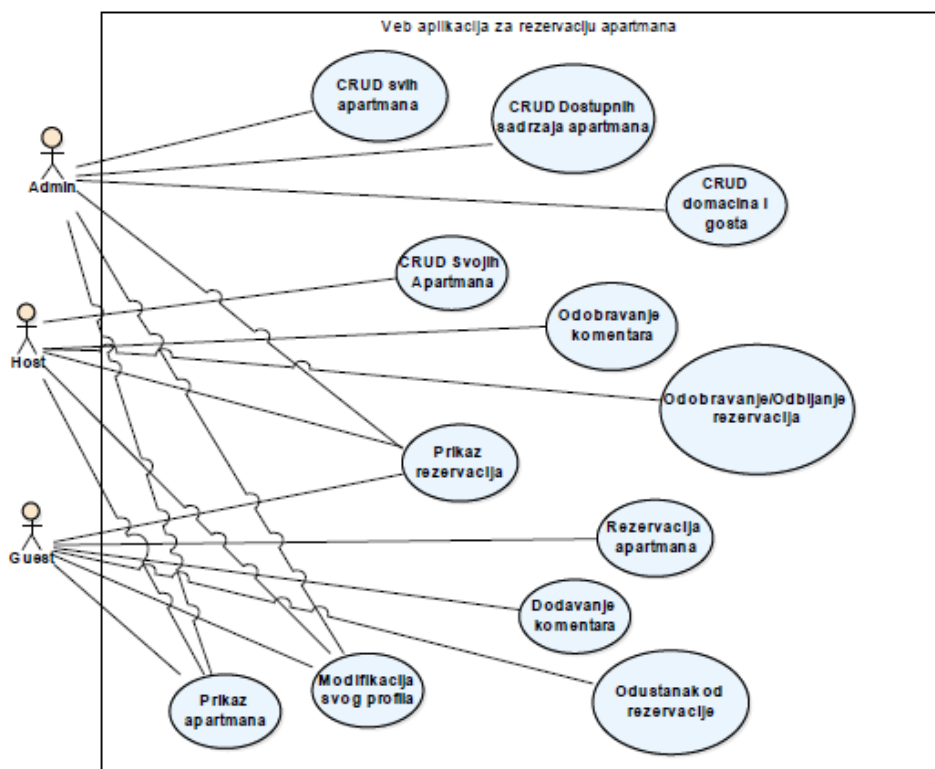
Gostima (*Guest*) su dostupne sledeće mogućnosti:

- Pregled svih apartmana (pretraga, filtriranje, sortiranje)
- Pregled svojih rezervacija i odustanak od istih (ukoliko je to moguće – više o ovome u nastavku teksta)
- Rezervacija apartmana
- Mogućnost ostavljanja komentara za apartman nakon završnog noćenja

Ne registrovani korisnici imaju mogućnost prikaza svih apartmana.

Svi korisnici mogu modifikovati-brisati svoje profile.

Dijagrami slučajeva korišćenja (*use-case* dijagrami) se koriste da predstave niz akcija koje sistem može ili treba da izvrši u toku komunikacije sa korisnikom tog sistema. Oni u sebi sadrže učesnika (aktera) i funkcije koje sistem obezbeđuje učesniku.



SLIKA 3.1 - DIJAGRAM SLUČAJA KORIŠĆENJA

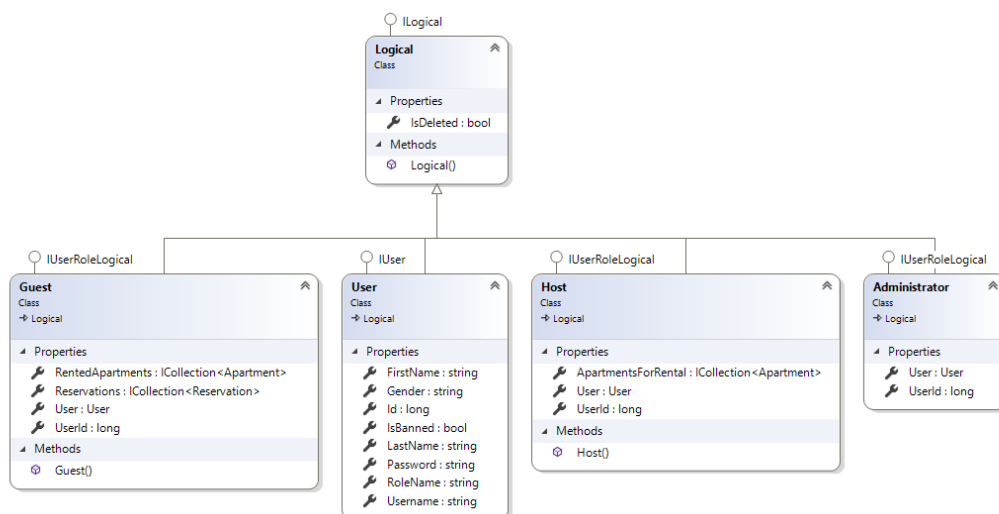
3.2 Dijagram klasa

Aplikacija rukuje sa sledećim entitetima:

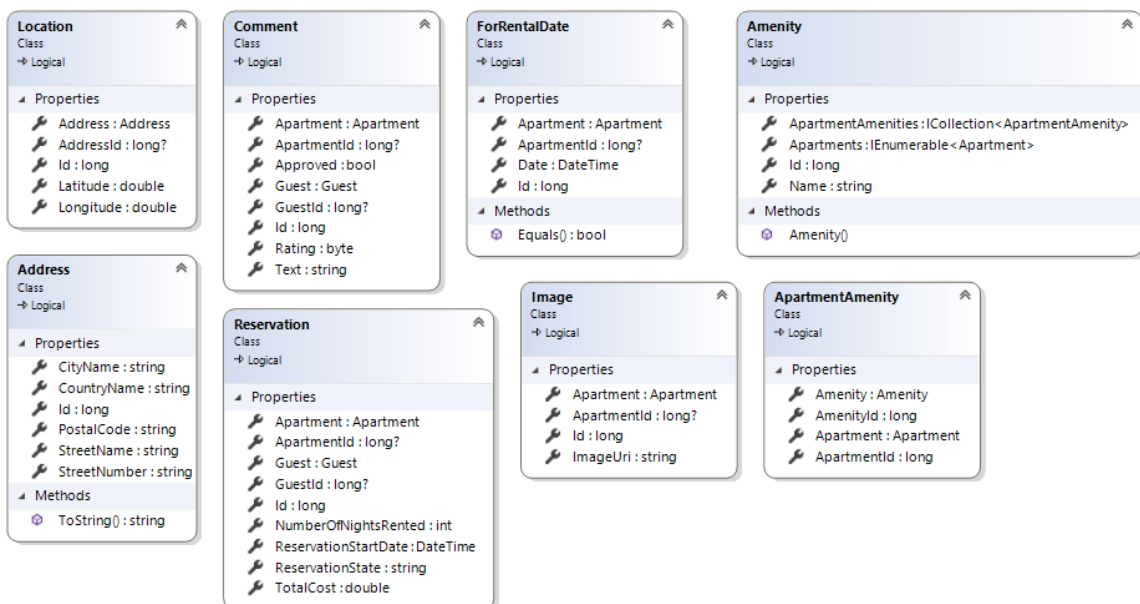
- Address
- Administrator
- Amenity

- Apartment
- ApartmentAmenity
- Comment
- ForRentalDate
- Guest
- Host
- Images
- Location
- Logical
- Reservation
- User

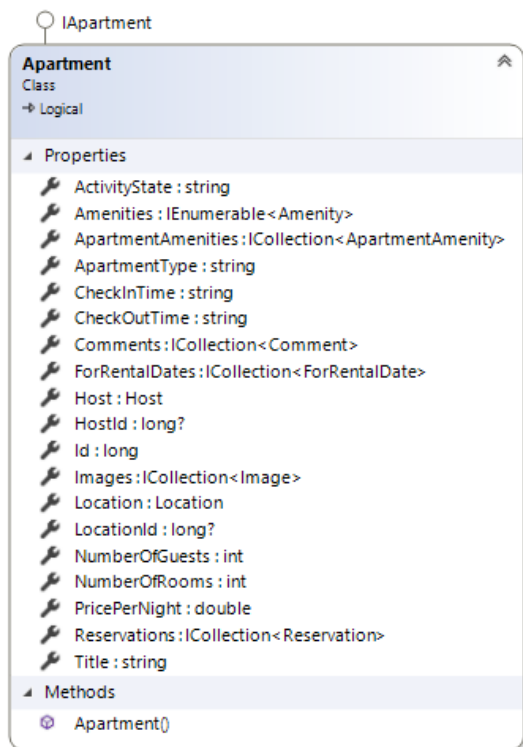
User, Host, Guest i Administrator klase služe za prijavu korisnika i sadrže njegove osnovne podatke kao i njegovu ulogu u sistemu. Logical klasa predstavlja pomoćnu baznu klasu koja sadrži polje *IsDeleted* pomoću kojeg je realizovano logičko brisanje, dok ApartmentAmenity predstavlja tip poveznika. Dijagram klasa aplikacije prikazan je u nastavku teksta.



SLIKA 3.2 - KLASJE ZA REGISTRACIJU



SLIKA 3.3 - DOMENSKJE KLAZE

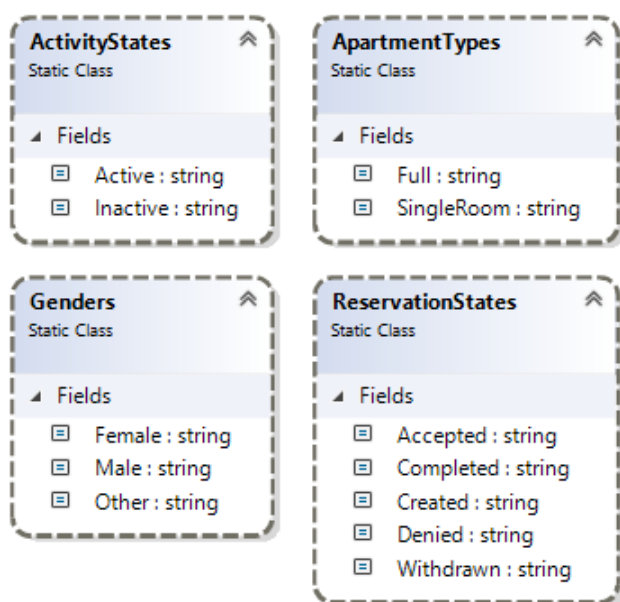


SLIKA 3.4 – KLASA APARTMAN

Na osnovu zahteva u zadatku (tipa stanja i slično) dodate su pomoćne statičke klase (sa svojim konstantama):

- ActivityStates
- ApartmentTypes
- Genders
- ReservationStates

Razlog zbog kojeg je izabran ovaj pristup a ne rešenje preko enumeracija jeste zbog olakšane implementacije na klijentskoj strani (u slučaju enumeracija bi na klijentskoj strani morali pamtit i koji broj znači šta, kao i mapirati taj broj na neki string koji bi korisnik mogao da vidi (npr. dok popunjava formu)).

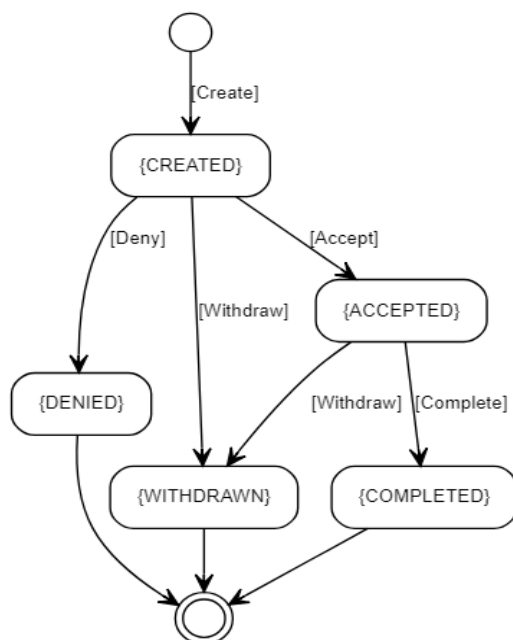


SLIKA 3.5 - KLASA SA KONSTANTAMA

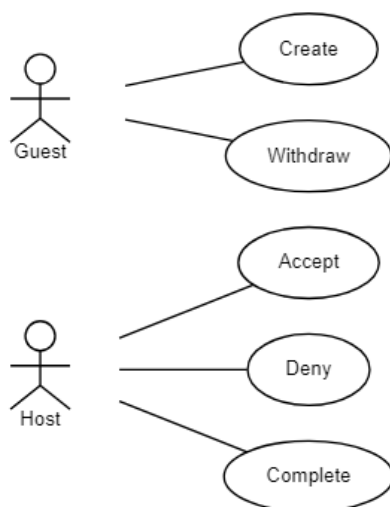
3.3 Dijagram stanja

Dijagrami stanja se koriste za opisivanje ponašanja sistema. Oni mogu da opišu moguća stanja objekta kako se događaji pojavljuju. Svaki dijagram obično predstavlja objekte jedne klase i prati različita stanja tih objekata kroz sistem.

Kao reprezentativni primer biće prikazan opis toka rezervisanja apartmana krećući se od početnog stanja rezervacije do finalnog



SLIKA 3.6 - DIJAGRAM STANJA REZERVACIJE



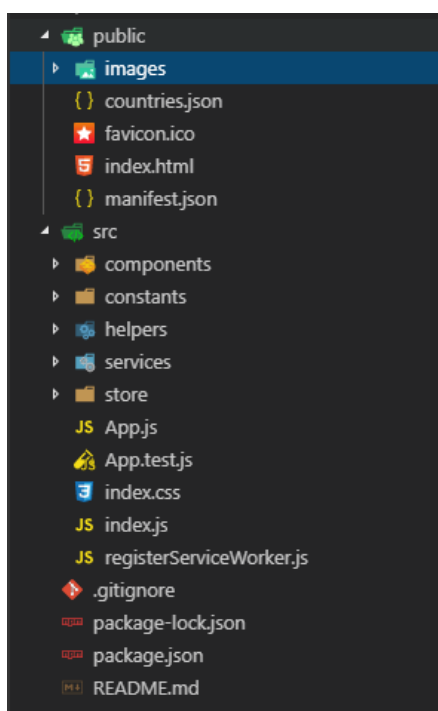
SLIKA 3.7 - USE CASE DIAGRAM ZA OPERACIJE NAD REZERVACIJOM

4 IMPLEMENTACIJA

Dok su u prvom poglavlju detaljno opisane tehnologije koje su upotrebljene za izradu veb aplikacije za rezervaciju apartmana, u ovom poglavlju će biti reč o detaljima implementacije.

4.1 Klijentska aplikacija

Kao što je već rečeno, klijentska aplikacija je razvijena korišćenjem React-a. Unutar korenskog foldera možemo primetiti u suštini dva foldera od interesa, *Public* koji sadrži statičke fajlove (slično kao *wwwroot* na serverskoj strani) i *Src* u kojem se nalaze servisi, komponente itd. (kod sa logikom).



SLIKA 4.1 - STRUKTURA KLIJENTSKOG FOLDERA

Components folder sadrži komponente, tj. React klase i funkcije čija je uloga vizualni prikaz elemenata, formi i slično. One su te koje koriste servise, helpere i reducere (o njima reč u nastavku teksta) radi promene stanja.

Glavna (korenska) komponenta je u suštini *App.js* koja pretstavlja layout i mesto za registrovanje ruta za navigaciju. Dok je *index.js* mesto gde se taj isti App “obmota” sa provajderom iz Redux biblioteke.


```
export const ApartmentCarousel = ({ images, className }) => {
  return (
    <Carousel className={`apartment-carousel ${className || ""}`}>
      {images &&
        images.map((img, index) => {
          return (
            <Carousel.Item key={`apartment-carousel-item-${index}`}>
              <img className="carousel-image" src={img.uri} alt="apartment" />
            </Carousel.Item>
          );
        })}
    </Carousel>
  );
};
```

SLIKA 4.2 - PRIMER REACT KOMPONENTE

Constants – predstavlja lokaciju, tj. skup svih konstanti korišćenih u aplikaciji. Ovo je urađeno da bi se izbegli magični stringovi i brojevi u aplikaciji.

```
export const Guest = "Guest";
export const Host = "Host";
export const Admin = "Administrator";

export const roleNames = {
  Guest: Guest,
  Host: Host,
  Admin: Admin
};
```

SLIKA 4.3 - PRIMER JEDNOG FAJLA SA KONSTANTAMA

Helpers - sadrži skup pomoćnih funkcija kao što su fje. za rad sa vremenom, parsiranjem, javascript objektima i slično.

```
export const makeCancelable = promise => {
  let rejectFn;

  const wrappedPromise = new Promise((resolve, reject) => {
    rejectFn = reject;

    Promise.resolve(promise)
      .then(resolve)
      .catch(reject);
  });

  wrappedPromise.cancel = () => {
    rejectFn({ canceled: true });
  };

  return wrappedPromise;
};
```

SLIKA 4.4 - PRIMER JEDNE POMOĆNE FUNKCIJE

Services – sadrži servise vezane za api pozive gde bi amenitiesService sadržao skup funkcija vezane za sadržaje apartmana, commentService za komentare itd.

```
export const handleResponse = createResponseHandler(logout);

export const userService = {
  login,
  logout,
  register,
  getAll,
  getById,
  update,
  updateCurrentUser,
  ban,
  unban,
  delete: _delete
};

export function login(username, password) {
  const requestOptions = {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username, password })
  };

  return fetch(`api/Account/Login`, requestOptions)
    .then(handleResponse)
    .then(user => {
      // store user details in local storage to keep user logged in between page refreshes
      localStorage.setItem("user", JSON.stringify(user));

      return user;
    });
}
```

SLIKA 4.5 - PRIMER SERVISA ZA RAD SA KORISNIKOM

Poslednji **Store** folder obuhvata fajlove vezane za *Redux* biblioteku te sledi kratak pregled-opis iste.

4.1.1 Redux

Osnovna namena Redux biblioteke jeste da pomogne u radu sa stanjem. Ona uvodi par termina kao što su **store**, **actions**, **reducers**.

Store u suštini predstavlja model tj. objekat bez set-era (ovo je urađeno da bi se izbegli bugovi, tj. problemi koje uvode globalne promenljive). Da bi smo promenili stanje uveden je pojam **akcija**.

```
export const alertActions = {
  success,
  error,
  clear
};

function success(message) {
  return { type: alertConstants.SUCCESS, message };
}

function error(message) {
  return { type: alertConstants.ERROR, message };
}

function clear() {
  return { type: alertConstants.CLEAR };
}
```

SLIKA 4.6 - PRIMER AKCIJA

Akcija predstavlja objekat koji obavezno sadrži polje **type** (opisuje tip akcije kao npr. „ADD_USER”) i opciono dodatne parametre.

Dispatch-eri bi bile funkcije koje ovaj objekat dostavljaju komponentama koje bi ove akcije obradili. Te komponente se nazivaju **Reducers**.

```

const alertReducer = (state = {}, action) => {
  switch (action.type) {
    case alertConstants.SUCCESS:
      return {
        type: "success",
        message: action.message
      };
    case alertConstants.ERROR:
      return {
        type: "danger",
        message: action.message
      };
    case alertConstants.CLEAR:
      return {};
    default:
      return state;
  }
};

export default alertReducer;

```

SLIKA 4.7 - PRIMER REDUCERA

Radi veće preglednosti i skalabilnosti takođe je moguće napraviti više *Reducers* u kojem slučaju bi trebalo napraviti jedan korenski reducer koji bi bio spoj ostalih. Funkcija koja ovo omogućava je ***combineReducers***.

```

import { combineReducers } from "redux";

const rootReducer = combineReducers({
  auth: authReducer,
  project: projectReducer,
  alert: alertReducer,
  apartment: apartmentReducer
});

export default rootReducer;

```

SLIKA 4.8 - PRIMER KOMBINOVANJA REDUCERA

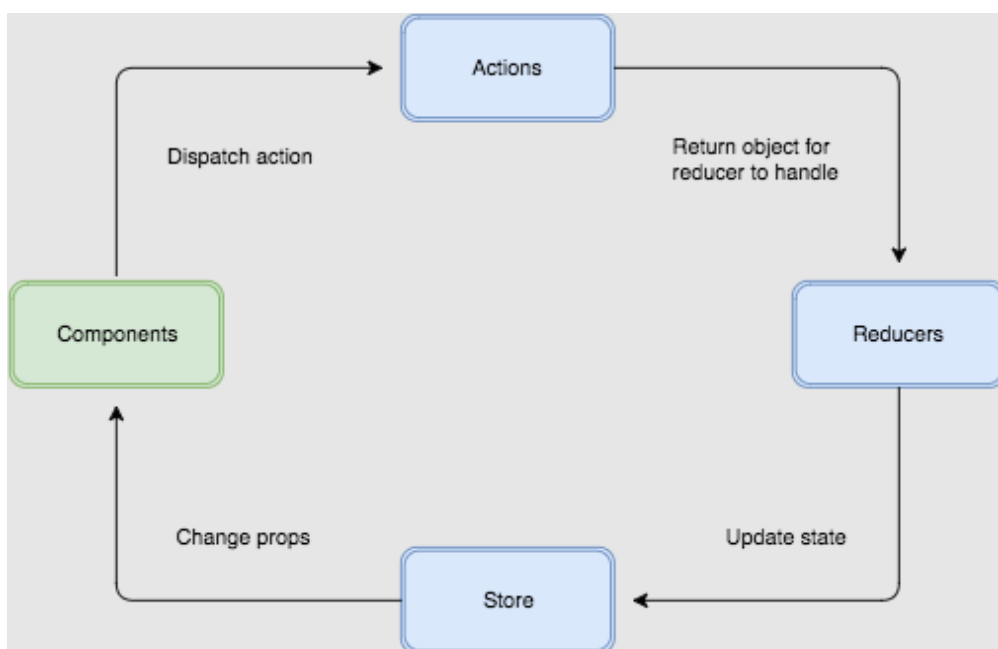
Sa obzirom da Redux nije vezan za React (može se koristiti i u običnom javascriptu bez ikakvog frameworka), uveden je **Provider**, tj. veza između Reduxa ili Reacta. Ova komponenta pruža mogućnost ubacivanja "store" promenljive kao parametre u konstruktoru (*dependency injection*) komponentama konektovanih putem **connect** funkcije iz React-Redux paketa.

```
const logger = createLogger();
const store = createStore(rootReducer, applyMiddleware(thunk, logger));

const app = () => {
  return (
    <Provider store={store}>
      <App />
    </Provider>
  );
};

ReactDOM.render(app(), rootElement);
```

SLIKA 4.9 - POVEZIVANJE REDUX BIBLIOTEKE SA REACT-OM



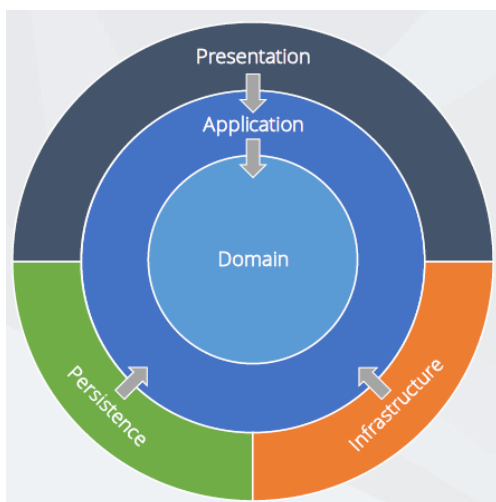
SLIKA 4.10 - TOK REACT-REDUX APLIKACIJE

4.2 Serverska aplikacija

Serverska aplikacija je ona koja komunicira sa svim ostalim i predstavlja centar sistema. Korišćena je višeslojna arhitektura sa sledećim slojevima:

- Domain
- Application
- Persistence
- Infrastructure
- Presentation

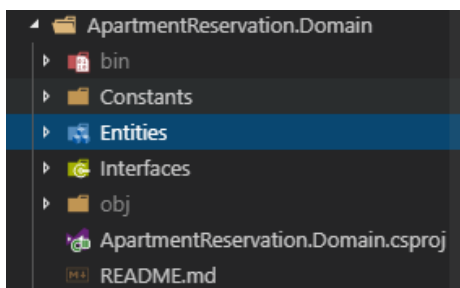
U nastavku teksta će svaki od ovih slojeva biti detaljno objašnjeni.



SLIKA 4.11 - SLOJEVI NA SERVERSKOJ STRANI

4.2.1 Domain layer

Domain sloj predstavlja centralni sloj aplikacije. U njemu se nalaze entiteti, enumeracije, konstante, logika vezano za domenske entitete i slično (npr. domain exceptions). Da bi se dobila veća nezavisnost od tehnologije ovaj sloj nebi trebalo da sadrži "framework dependent features" kao što su data anotacije i slično.

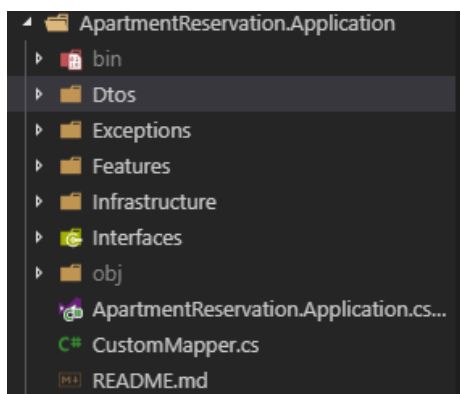


SLIKA 4.12 - IZGLED DOMAIN SLOJA

4.2.2 Application layer

Application, zajedno sa Domain slojem, čini **Core** sloj. Tj. svi ostali zavise od njega dok on ne zavisi od drugih. Kao ovakav Core sloj mora da poštuje D iz SOLID principa, tj. da se oslanja na apstrakciju a ne na konkretizaciju.

Primeri entiteta koji spadaju u ovaj sloj su Interfejsi, DTO, Komande, Custom Exceptions, Validators..

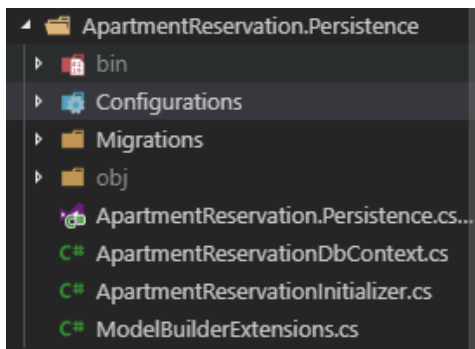


SLIKA 4.13 – IZGLED APPLICATION SLOJA

Glavna, ujedno i najsloženija komponenta ovog sloja predstavlja *Features* folder u kojem se nalaze Commands & Queries (handleri *Mediator* biblioteke) o kojima će biti izdvojeno posebno poglavlje u nastavku teksta.

4.2.3 Persistence layer

Persistence sloj sadrži implementacije interfejsa iz *Core* sloja vezano za čuvanje podataka sa bazom. On u ovom slučaju sadrži klase vezane za Entity Framework, tj. DbContext, Migracije, Fluent API Konfiguracije, Initialize (Seed) metodu, Extenzije (za npr. ModelBuilder) itd.

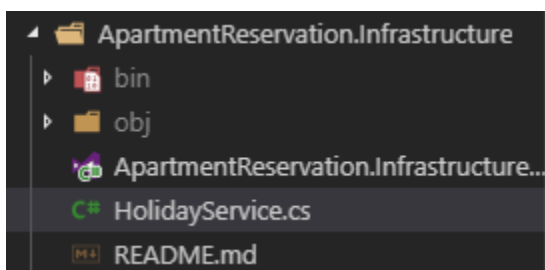


SLIKA 4.14 - PRIMER PERSISTENCE SLOJA

4.2.4 Infrastructure layer

U ovom sloju se nalaze implementacije interfejsa iz *Core* dela koje se odnose na external api pozive (npr. File System, Email, ... anything external)

Zbog niske potrebe za pozivom externih podataka servera ovaj sloj sadži samo jednu klasu a to je *HolidayService* koji u ovom slučaju čita i parsira podatke iz .JSON fajla o datumima praznika u Srbiji.

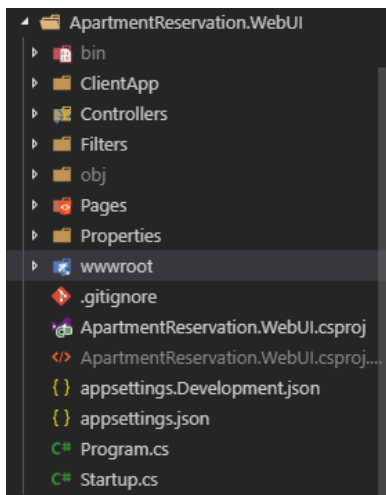


SLIKA 4.15 - PRIMER INFRASTRUCTURE SLOJA

4.2.5 Presentation layer

Presentation sloj predstavlja “najvišeg” člana u slojevitoj arhitekturi. On zavisi od svih ostalih slojeva, ali nijedan drugi ne zavisi od njega. Neke komponente koje se mogu naći u ovom sloju su “SPA” (tj. Klijentski) folder, API kontroleri, Exception filteri, wwwroot folder itd.

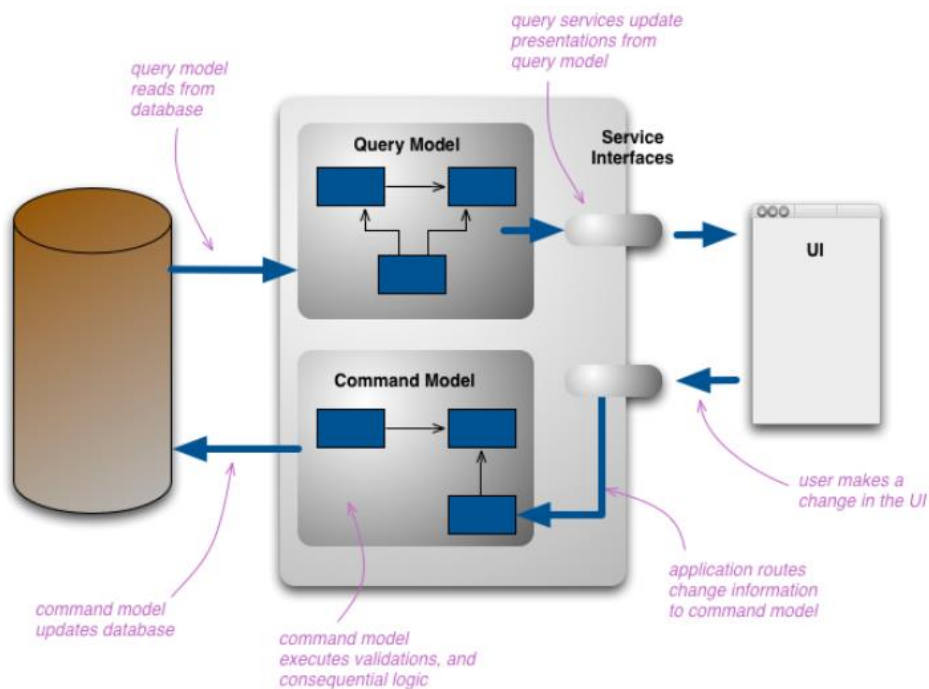
Za razliku od tradicionalnog pristupa, **kontroleri u ovom sloju su “tanki”** što znači da oni samo generišu komande (zahteve) koje obrađuju handleri iz aplikacionog sloja (CQRS pattern, o njemu više u nastavku teksta) sa eventualno dodatnom logikom vezanu za autorizaciju.



SLIKA 4.16 - IZGLLED PRESENTATION SLOJA

4.2.6 CQRS + MediatR

Osnovna ideja **CQRS** (Command Query Responsibility Segregation) šablona jeste razbijanje upita na Commands (modifikaciju) i Queries (čitanje).



SLIKA 4.17 - VIZUELNI PRIKAZ CQRS PATERNA

Na ovaj način naš sistem postaje lakši za ekstenziju, održavanje i postaje fleksibilniji. Separacija nam takođe omogućava potencijalnu optimizaciju query modela za čitanje kao i optimizaciju za command modela za pisanje odvojeno.

MediatR biblioteka predstavlja prostu implementaciju mediator šablona u kojem objekti ne komuniciraju direktno već preko *mediator* objekta. Na ovaj način se zavisnost između objekata smanjuje, samim tim povećava skalabilnost.

Request objekat predstavlja ulazni parametar od metode *RequestHandler*-a. Način na koji MediatR zna da poveže Request sa RequestHandlerom je putem refleksije.

CQRS i MediatR možemo da kombinujemo tako što ćemo commands & queries implementirati kao Request i RequestHandlere. Upravo je to i urađeno.

```
8 references | Danilo Novakovic, 24 days ago | 1 author, 4 changes
public class GetAllReservationsQuery : IRequest<IEnumerable<ReservationDto>>
{
    6 references | Danilo Novakovic, 30 days ago | 1 author, 1 change | 0 exceptions
    public long? HostId { get; set; }
    6 references | Danilo Novakovic, 30 days ago | 1 author, 1 change | 0 exceptions
    public long? GuestId { get; set; }
    3 references | Danilo Novakovic, 27 days ago | 1 author, 1 change | 0 exceptions
    public long? ApartmentId { get; set; }
    2 references | Danilo Novakovic, 24 days ago | 1 author, 1 change | 0 exceptions
    public string GuestUsername { get; set; }
    3 references | Danilo Novakovic, 28 days ago | 1 author, 1 change | 0 exceptions
    public string ReservationState { get; set; }
}
```

SLIKA 4.18 - PRIMER REQUEST OBJEKTA

```
3 references | Danilo Novakovic, 23 days ago | 1 author, 5 changes
public class GetAllReservationsQueryHandler : IRequestHandler<GetAllReservationsQuery, IEnumerable<ReservationDto>>
{
    private readonly IApartmentReservationDbContext context;

    1 reference | Danilo Novakovic, 30 days ago | 1 author, 1 change | 0 exceptions
    public GetAllReservationsQueryHandler(IApartmentReservationDbContext context)
    {
        this.context = context;
    }

    99 references | Danilo Novakovic, 23 days ago | 1 author, 2 changes | 0 exceptions
    public async Task<IEnumerable<ReservationDto>> Handle(GetAllReservationsQuery request, CancellationToken cancellationToken)
    {
        var query = this.context.Reservations
            .Include(r => r.Guest).ThenInclude(g => g.User)
            .Include(r => r.Apartment).ThenInclude(a => a.Host).ThenInclude(h => h.User)
            .Where(r => !r.IsDeleted && !r.Apartment.IsDeleted && !r.Guest.IsDeleted);

        query = this.ApplyFilters(request, query);

        var reservations = await query.ToListAsync(cancellationToken).ConfigureAwait(false);

        return reservations.Select(r => new ReservationDto(r));
    }
}
```

SLIKA 4.19 - PRIMER REQUEST HANDLER OBJEKTA

Ono što nam ovo omogućuje jeste da skoro svu logiku izbacimo iz kontrolera na prezentacijonom sloju. Ono što on postaje jeste samo jako tanka komponenta koja šalje Request objekte mediatoru i vraća rezultat nazad (uz eventualne minimalne modifikacije).

```
[Route("api/[controller]")]
[ApiController]
[Authorize(Policy = Policies.AdministratorOrHostOnly)]
1 reference | Danilo Novakovic, 10 days ago | 1 author, 5 changes
public class AmenitiesController : ControllerBase
{
    private readonly IMediator mediator;

    0 references | Danilo Novakovic, 40 days ago | 1 author, 1 change | 0 exceptions
    public AmenitiesController(IMediator mediator)
    {
        this.mediator = mediator;
    }

    // GET: api/Amenities
    [HttpGet]
    [ProducesResponseType(typeof(IEnumerable<AmenityDto>), StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status401Unauthorized)]
    0 references | Danilo Novakovic, 10 days ago | 1 author, 3 changes | 0 requests | 0 exceptions
    public async Task<IActionResult> Get([FromQuery] GetAllAmenitiesQuery query)
    {
        return this.Ok(await this.mediator.Send(query).ConfigureAwait(false));
    }
}
```

SLIKA 4.20 - PRIMER API KONTROLERA

4.3 Podizanje projekta na cloud

Proces podizanja projekta na cloud Visual Studio čini trivijalnim. Klikom na **Publish** opcije i odabirom **App Service** dobijamo formu u kojoj možemo da napravimo naš Azure SQL Server i bazu podataka. Takođe imamo opciju da odaberemo Resource group, hosting plan i application insights (kao i region za svaku)

Azure SQL Server
Create new

Name: WebApplication12019033

Subscription: Free Trial

Resource Group: WebApplication12019033

Hosting Plan: WebApplication12019033

Application Insights: None

Database Server Name: webapplication120190331054241dbserver

Location: East US

Database administrator username (must have permissions to create)*: [redacted]

Database administrator password: [redacted]

Administrator Password (confirm): [redacted]

⚠ Passwords must match

Buttons: OK, Cancel, Create, Cancel

Klikom na **Create** dugme se potvrđuje forma i Visual Studio pokreće automatski proces podizanja web servisa na Azure.

Azure App Service
Create new

Name: WebApplication120190331054241

Subscription: Free Trial

Resource Group: WebApplication1201903317070850ResourceGroup (centralus) New...

Hosting Plan: WebApplication120190331054241Plan* (Central US, S1) New...

Application Insights: East US

Buttons: Export..., Create, Cancel

Explore additional Azure services

- Create a storage account
- Create a SQL Database

Clicking the Create button will create the following Azure resources

- Azure SQL Database - WebApplication120190331054241 [Settings] [Close]
- SQL Server - webapplication120190331054241dbserver [Settings] [Close]
- Hosting Plan - WebApplication120190331054241Plan [Settings] [Close]
- App Service - WebApplication120190331054241

Treba napomenuti da Azure servis po default-u postavlja *connection string* za Azure DB sa imenom "DefaultConnection" te treba u aplikaciji (tj. dbContext klasi) obezbediti učitavanje istog.

4.3.1 Prednosti

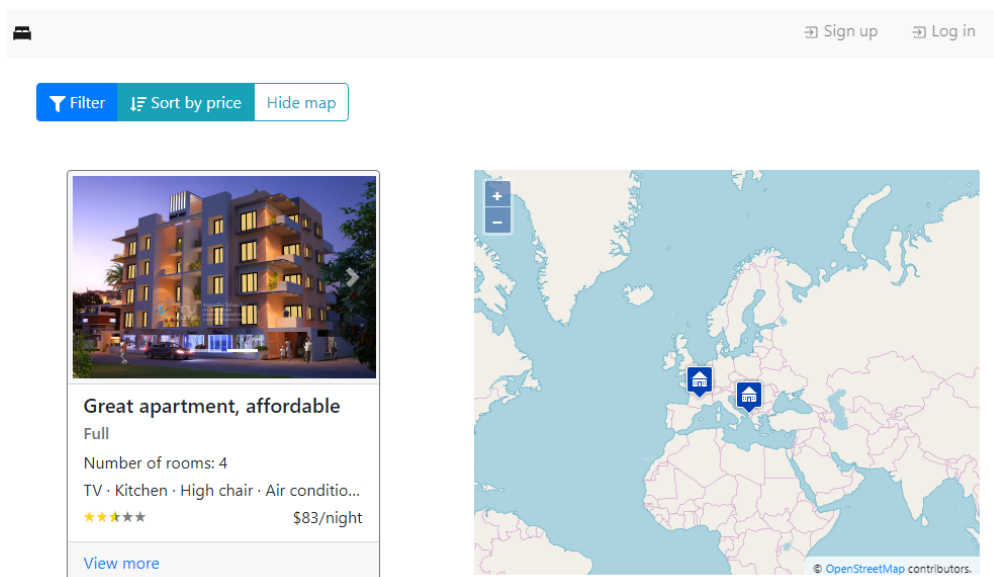
- **Skalabilnost** - Možete ispitati opcije za skaliranje aplikacije, kao i izvan nje. Skaliranje se odnosi na povećanje resursa koji se daju svakom primeru koji hostuje vaša aplikacija. Skaliranje se odnosi na povećanje broja instanci koje hostuju vašu aplikaciju. Automatsku skalu za svoju aplikaciju možete konfigurisati tako da će automatski povećati broj primeraka za hostovanje aplikacije kao odgovor na učitavanje, a zatim ih smanjiti kada se učitavanje smanji.
- **Sigurnost i usklađenost** - Još jedna prednost hostinga naše aplikacije pomoću usluge Azure je sigurnost i usklađenost. Azure App Service pruža ISO, SOC i PCI usklađenost. Možemo odabrati proveru autentičnosti korisnika sa Azure Active Directory ili društvenim prijavama kao što su Twitter, Facebook, Google ili Microsoft. Možemo stvoriti IP ograničenja, upravljati identitetima usluga, dodavati prilagođene domene i podržavati SSL za aplikaciju, kao i konfigurisati sigurnosne kopije sa obnovljivim arhivskim kopijama sadržaja, konfiguracije i baze podataka aplikacije. Ovim se svojstvima pristupa u opcijama menija Authentication / Authorization, Identity, backup i SSL Settings.
- **Deployment slots** - Često kada implementirate aplikaciju, postoji mali period zastoja tokom ponovnog pokretanja aplikacije. Deployment slots izbegavaju ovaj problem tako što vam omogućuju da se postavite na zasebnu instancu ili skupu instanci i da ih zagrejete pre nego što ih zamenite u proizvodnju. Ako se nakon zamene pojavljuju problemi u proizvodnji, uvek se možete vratiti na poslednje poznato stanje proizvodnje.

5 Prikaz implementiranog rešenja

U ovom poglavlju aplikacija će biti opisana iz korisnikovog ugla.

Kao što je već rečeno, postoje tri tipa korisnika, administrator, domaćin i gost, pa će tako biti objašnjeno kako oni koriste aplikaciju, respektivno.

Početni ekran aplikacije sadrži prikaz dostupnih aktivnih apartmana, korisnik koji nije ulogovan može da vrši pregled i sortiranje istih ali ne može da modifikuje apartmane niti da pravi rezervacije i slično.

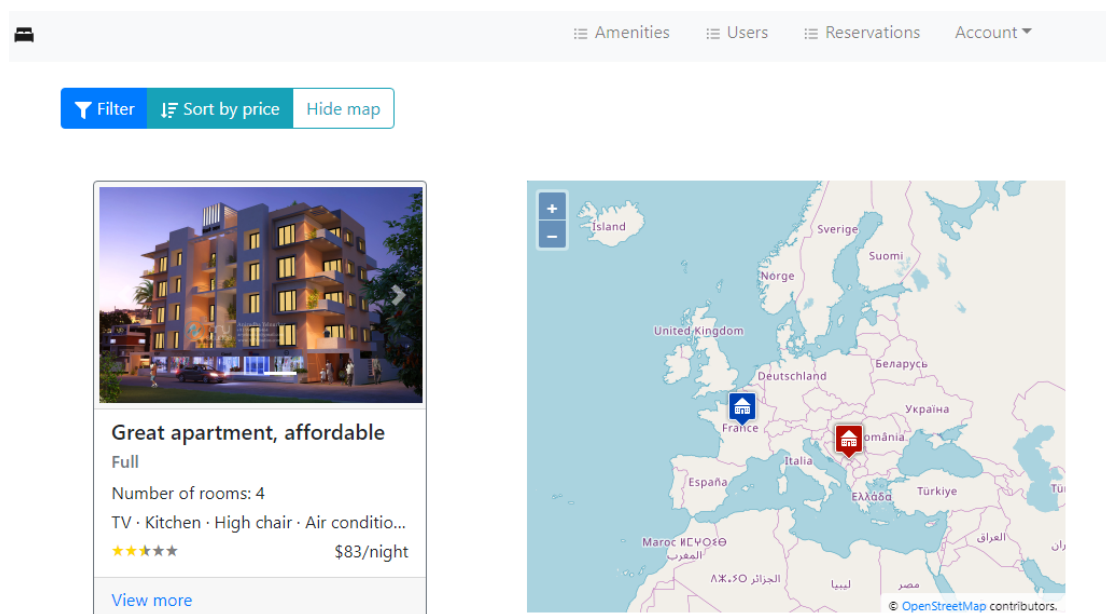


SLIKA 5.1 - POČETNA STRANA

U gornjem desnom uglu se nalaze dve opcije Sign up I Log In za prijavu korisnika. Korisnik putem registrovanja postaje Gost, dok domaćina može jedino administrator da doda, a samog administratora nije moguće programski dodati.

Nakon uspešne prijave se korisniku prikazuje ekran modifikovan u skladu sa njegovom ulogom.

5.1 Administrator



SLIKA 5.2 - POČETNA STRANA ZA ADMINISTRATORA

Za razliku od korisnika koji nije ulogovan, administrator ima takođe prikaz i neaktivnih apartmana (kao mogućnost modifikacije istih klikom na nekih od njih).

Navigacioni meni sa sobom takođe unosi nove opcije (tabove) koji će u nastavku teksta biti prikazani.

5.1.1 Amenities

Amenities

Add Amenity	
name	
TV	<button>Edit</button> <button>Delete</button>
Kitchen	<button>Edit</button> <button>Delete</button>
High chair	<button>Edit</button> <button>Delete</button>
Air conditioning	<button>Edit</button> <button>Delete</button>
Heating	<button>Edit</button> <button>Delete</button>

SLIKA 5.3 - AMENITIES STRANICA

Ovaj prozor omogućava administrator modifikaciju i dodavanje dostupnih sadržaja apartmana. Administratoru je dostupan tabelaran prikaz sa intuitivnim dugmićima Add, Edit i Delete putem kojih vrši modifikaciju.

5.1.2 Users

Users

+ Add Guest		+ Add Host		Filter	
username	firstName	lastName	gender	roleName	
admin	Jotaro	Kujo	Male	Administrator	
host	Enyaba	Geil	Female	Host	<div>Edit</div> <div>Ban</div> <div>Delete</div>
guest	Dio	Brando	Other	Guest	<div>Edit</div> <div>Ban</div> <div>Delete</div>
guest2	Joseph	Joester	Male	Guest	<div>Edit</div> <div>Ban</div> <div>Delete</div>

SLIKA 5.4 - USERS STRANICA

Slično kao i kod Amenitit, i ovde je dat tabelaran prikaz korisnika sa dugmićima koji omogućavaju modifikaciju sadržaja. Međutim ono što ovaj prozor takođe omogućava jeste Filtriranje sadržaja putem *Filter* dugmeta.

Filter

Apply one or more filters to all users on the list.

Role Name

Gender

Username

Accept Clear

SLIKA 5.5 - FILTER USERS MODALNI PROZOR

5.1.3 Reservations

Reservations

Filter
Sort by price

Guest	Host	Apartment	Date	Number of Nights	Total Cost (in \$)	Reservation state	
guest	host	Great apartment, affordable	2019/6/24	3	249	Accepted	
guest2	host	Great apartment, affordable	2019/6/24	3	249	Denied	
guest	host	Great apartment, affordable	2019/6/24	2	166	Withdrawn	
guest2	host	Great apartment, affordable	2019/7/2	2	166	Created	
guest	host	The Pondhouse - A Magical Place	2019/6/24	3	111	Denied	
guest	host	Great apartment, affordable	2019/6/28	1	83	Completed	

SLIKA 5.6 - PRIKAZ REZERVACIJA ZA ADMINISTRATORA

Administraturu je takođe omogućen pregled svih rezervacija u sistemu, međutim nije mu dozvoljena modifikacija istih (samo domaćini I gost imaju mogućnost modifikovanja stanja rezervacije u skladu sa pravilima objašnjenim u prethodnim poglavljima).

5.1.4 Apartman

Klikom nekog apartmana sa početne stranice administrator ima prikaz detalja apartmana u sekcijama (zbog obima podataka koje entitet Apartman nosi sa sobom – bolji *user experience*) gde administrator ima mogućnost modifikaciju svake putem **Edit** dugmeta.

Prva sekcija predstavlja kratak opis apartmana sa nekim ključnim informacijama kao što su Tip, broj soba, broj dozvoljenih gostiju, cena, vreme prijave I odjave, kao I lokacija (grad). Pritiskom na Edit dugme administratoru iskače forma u modalnom prozoru putem koje može da izmeni ove podatke.

Good location, nice view.

Нови Сад, **Inactive**

Type: Full

Number of rooms: 3

Number of guests: 4

Price: \$54 per night

Check-In Time: 14:00:00

Check-Out Time: 10:00:00

[See more apartments from this host](#)

Edit

SLIKA 5.7 - SUMMARY SEKCIJA

Edit Apartment Summary

Title
Good location, nice view. ✓

Apartment Type
Full ✓

Activity State
Inactive ✓

Number of rooms
3 ✓

Number of guests
4 ✓

Price(\$) per night
54 ✓

Check in time
14:00 ✓

Check out time
10:00 ✓

Save Changes Close

SLIKA 5.8 - EDIT APARTMENT SUMMARY FORMA

Namena sledeće sekcije jeste da pruži podršku za rad sa slikama. Administratoru je dat prikaz trenutnih slika putem Carousel komponente. Mogućnost dodavanja novih mu pruža **Add** dugme, dok mogućnost brisanja **Delete**. Klikom na bilo kojih od ova dva iskače forma slično kao i u prošloj sekciji.

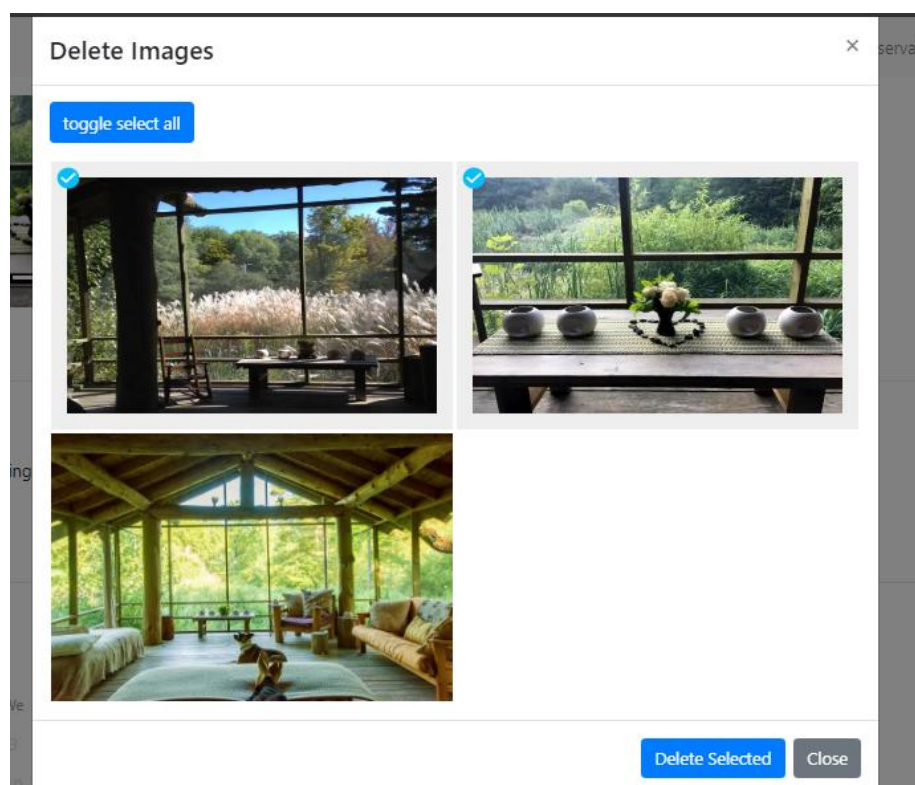
Images



Add

Delete

SLIKA 5.9 - SEKCIJA ZA SLIKE APARTMANA



SLIKA 5.10 - FORMA ZA BRISANJE SLIKA

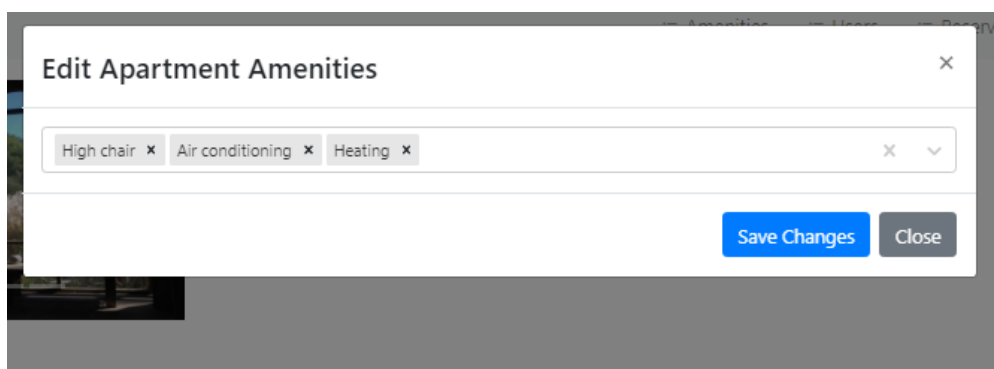
Sledeća sekcija je vezana za sadržaje apartmana. Prikaz sadržaja je izvršen u vidu liste, dok je modifikacija implementirana putem auto-complete multi select liste u kojem administrator može selektovanjem komponente da dobije dropdown svih mogućih amenities ili da kuca ima pa da dobije sugestiju (slično kao u pretraživaču). Brisanje nekog od sadržaja apartmana je moguće obaviti klikom na “x” dugme pored naziva istog.

Amenities

- High chair
- Air conditioning
- Heating

Edit

SLIKA 5.11 – PRIKAZ AMENITIES SEKCIJE



SLIKA 5.12 - FORMA ZA IZMENU SADRŽAJA APARTMANA

Od preostalih sekcija su sekcija za **Komentare** (gde administrator ima samo pregled svih), za **Dostupne datume** noćenja, sekcija za **Lokaciju** koja pruža izmenu putem forme kao i putem klika na mapu (kada klikne na mapu automacki se ažurira forma).

Availability

July 2019

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			


August 2019

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

[Edit For Rental Dates](#)

SLIKA 5.13 - DOSTUPNI DATUMI

Edit Apartment Location ×



Street Name

Street Number

City

State

Zip

Longitude

Latitude

[Cancel](#)
[Save Changes](#)

SLIKA 5.14 - FORMA ZA IZMENU LOKACIJE APARTMANA

Comments

guest

★★★★★

Had fun, enjoyable experience. Neighbours were kinda annoying tho.

guest2

★★★★★

Bad experience! Would not recommend to anybody!

SLIKA 5.15 - SEKCIJA ZA KOMENTARE

6 ZAKLJUČAK

U ovom radu je detaljno objašnjen razvoj veb aplikacije za rezervaciju apartmana. Aplikacija se sastoji od dve manje aplikacije: klijentske strane, i serverske strane. Klijentska strana je rađena u *React*-u, serverska u *AspNetCore*-u.

U radu je data specifikacija i implementacija aplikacije.

U poslednjem poglavlju je prikazano korišćenje aplikacije kako od strane administratora tako i od strane domaćina i gosta.

7 LITERATURA

1. <https://redux.js.org/>
2. <https://docs.microsoft.com/en-us/aspnet/?view=aspnetcore-2.2>
3. <https://reactjs.org>
4. <https://www.martinfowler.com/bliki/CQRS.html>
5. <https://github.com/jbogard/MediatR/wiki>
6. <https://www.smartdraw.com/uml-diagram/>
7. <https://www.codecademy.com/articles/what-is-rest>
8. <https://docs.microsoft.com/en-us/azure/>