Veb aplikacija za bazene

Ovo je projekat za ispit iz predmeta: Praktikum - Internet i veb tehnologije.

Broj indeksa: 2018200770

Ime i prezime: Aleksa Vidaković

Školska godina: 2021/2022

# Projektini zahtev

Aplikacija treba da omogući pristup za dve grupe korisnika.

Prva grupa korisnika su administratori koji imaju uvid u spisak svih zakazanih termina za korišćenje bazena za treninge, da unose u bazu podataka nove slobodne termine, kao i da otkazuju postojeće rezervacije. Administrator može da uređuje i sadržaj stranica na sajtu koji je sastavni deo interfejsa za druge korisnike sajta.

Drugi korisnici sajta moraju da se prijave na portal kako bi mu pristupili, a to rade preko formulara za prijavu na portal. Ukoliko korisnik nema nalog, mora da izvrši regitraciju koju radi preko formulara za registarciju novog naloga. Korisnik unosi svoje podatke, među kojima su ime, prezime, broj telefona i adresa e-pošte.

Kada se korisnik prijavi na poratl sa ispravnim korisničkim imenom i lozinkom, biće mu prikazan spisak svih otvorenih termina u kojima ima slobodnih mesta, a koji još nisu prošli. Korisnik aktivira opciju za registarciju termina i potvrđuje svoju registraciju za konkretan termin (datum i sat). Pošto izvrši odabir i potvrdu registracije za termin za korišćenje bazena, taj i svi ostali termini koji nisu prošli su dostupni u listi "Moji termini". Termini koje je adminsitrator otkazao se prikazuju u listi kao "otkazani" i treba da budu istaknuti (bojom ili sličicom pored termina). Korisnik može i sâm da otkaže svoj termin za korišćenje bazena.

Na veb sajtu se prijazuju stranice koje su sastavni deo veb aplikacije, a kao što su "pravila o korišćenju bazena", "pravila o registarciji termina", "uputstvo za registarciju", "kontakt strana" itd. Ove stranice uređuje administrator kroz panel za upravljanje sadržajem kojem ima pristup. Grafički interfejs veb sajta treba da bude realizovan sa responsive dizajnom.

# Tehnička ograničenja

* Aplikacija mora da bude realizovana na Node.js platformi korišćenjem Express biblioteke. Aplikacija mora da bude podeljena u dve nezavisne celine: back-end veb servis (API) i front-end (GUI aplikacija). Sav kôd aplikacije treba da bude organizovan u jednom Git spremištu u okviru korisničkog naloga za ovaj projekat, sa podelom kao u primeru zadatka sa vežbi.
* Baza podataka mora da bude relaciona i treba koristiti MySQL ili MariaDB sistem za upravljanje bazama podataka (RDBMS) i u spremištu back-end dela aplikacije mora da bude dostupan SQL dump strukture baze podataka, eventualno sa inicijalnim podacima, potrebnim za demonstraciju rada projekta.
* Back-end i front-end delovi projekta moraju da budi pisani na TypeScript jeziku, prevedeni TypeScript prevodiocem na adekvatan JavaScript. Back-end deo aplikacije, preveden na JavaScript iz izvornog TypeScript koda se pokreće kao Node.js aplikacija, a front-end deo se statički servira sa rute statičkih resursa back-end dela aplikacije i izvršava se na strani klijenta. Za postupak provere identiteta korisnika koji upućuje zahteve back-end delu aplikacije može da se koristi mehanizam sesija ili JWT (JSON Web Tokena), po slobodnom izboru.
* Sav generisani HTML kôd koji proizvodi front-end deo aplikacije mora da bude 100% validan, tj. da prođe proveru W3C Validatorom (dopuštena su upozorenja - Warning, ali ne i greške - Error). Grafički korisnički interfejs se generiše na strani klijenta (client side rendering), korišćenjem React biblioteke, dok podatke doprema asinhrono iz back-end dela aplikacije (iz API-ja). Nije neophodno baviti se izradom posebnog dizajna grafičkog interfejsa aplikacije, već je moguće koristiti CSS biblioteke kao što je Bootstrap CSS biblioteka. Front-end deo aplikacije treba da bude realizovan tako da se prilagođava različitim veličinama ekrana (responsive design).
* Potrebno je obezbediti proveru podataka koji se od korisnika iz front-end dela upućuju back-end delu aplikacije. Moguća su tri sloja zaštite i to: (1) JavaScript validacija vrednosti na front-end-u; (2) Provera korišćenjem adekvatnih testova ili regularnih izraza na strani servera u back-end-u (moguće je i korišćenjem izričitih šema - Schema za validaciju ili drugim pristupima) i (3) provera na nivou baze podataka korišćenjem okidača nad samim tabelama baze podataka.
* Neophodno je napisati prateću projektnu dokumentaciju o izradi aplikacije koja sadrži (1) model baze podataka sa detaljnim opisom svih tabela, njihovih polja i relacija; (2) dijagram baze podataka; (3) dijagram organizacije delova sistema, gde se vidi veza između baze, back-end, front-end i korisnika sa opisom smera kretanja informacija; (4) popis svih aktivnosti koje su podržane kroz aplikaciju za sve uloge korisnika aplikacije prikazane u obliku Use-Case dijagrama; kao i (5) sve ostale elemente dokumentacije predviđene uputstvom za izradu dokumentacije po ISO standardu.
* Izrada oba dela aplikacije (projekata) i promene kodova datoteka tih projekata moraju da bude praćene korišćenjem alata za verziranje koda Git, a kompletan kôd aplikacije bude dostupan na javnom Git spremištu, npr. na besplatnim GitHub ili Bitbucket servisima, jedno spremište za back-end projekat i jedno za front-end projekat. Ne može ceo projekat da bude otpremljen u samo nekoliko masovnih Git commit-a, već mora da bude pokazano da je projekat realizovan u kontinuitetu, da su korišćene grane (branching), da je bilo paralelnog rada u više grana koje su spojene (merging) sa ili bez konflikata (conflict resolution).

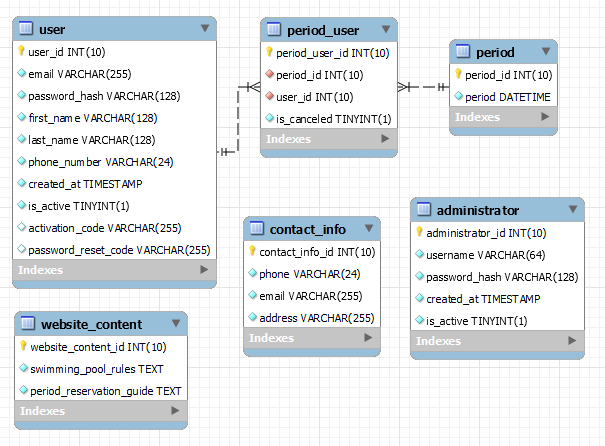
Baza podataka

Tabela “**user**” sadrži sledeće podatke: id, email, password\_hash, first\_name, last\_name, phone\_number, created\_at, is\_active, activation\_code, password\_reset\_code. Atributi email, activation\_code I password\_hash su jedinstveni (unique).

Tabela “**period**”, pored id-a, sadrži samo Datetime podatak termina, koji je jedinstven.

S obzirom da jedan korisnik može rezervisati vise termina, kao I da jedan isti termin može rezervisati vise korisnika, odlučeno je da se primeni veza m:n (vise prema više). Vezna tabela “**period\_user**” pored stranih kljičeva ka “user” I “period” tabelama ima I polje is\_canceled koje sadrži informaciju o tome da li je rezervacija otkazana. Takođe, par stranih ključeva “user\_id” I “period\_id” tretiraju se kao jedinstveni kako bi se jednom korisniku onemogućilo da dva puta rezerviše isti termin.

Tabela “**administrator**” sadrži kolone: id, username, password\_hash, created\_at, is\_active. Username je jedinstven.

S obzirom da je u projektnom zahtevu traženo da administrator treba da ima mogućnost da menja sadržaj stranica koja su sastavni deo aplikacije, napravljene su još dve tabele koje će skladištiti te podatke: “website\_content” tabela I “contact\_info” tabela. Tabela “**contact\_info**” ima 3 atributa: phone, email I address; Tabela “**website\_content**” sadrži podatke: swimming\_pool\_rules, period\_reservation\_guide.

# Uloge korisnika

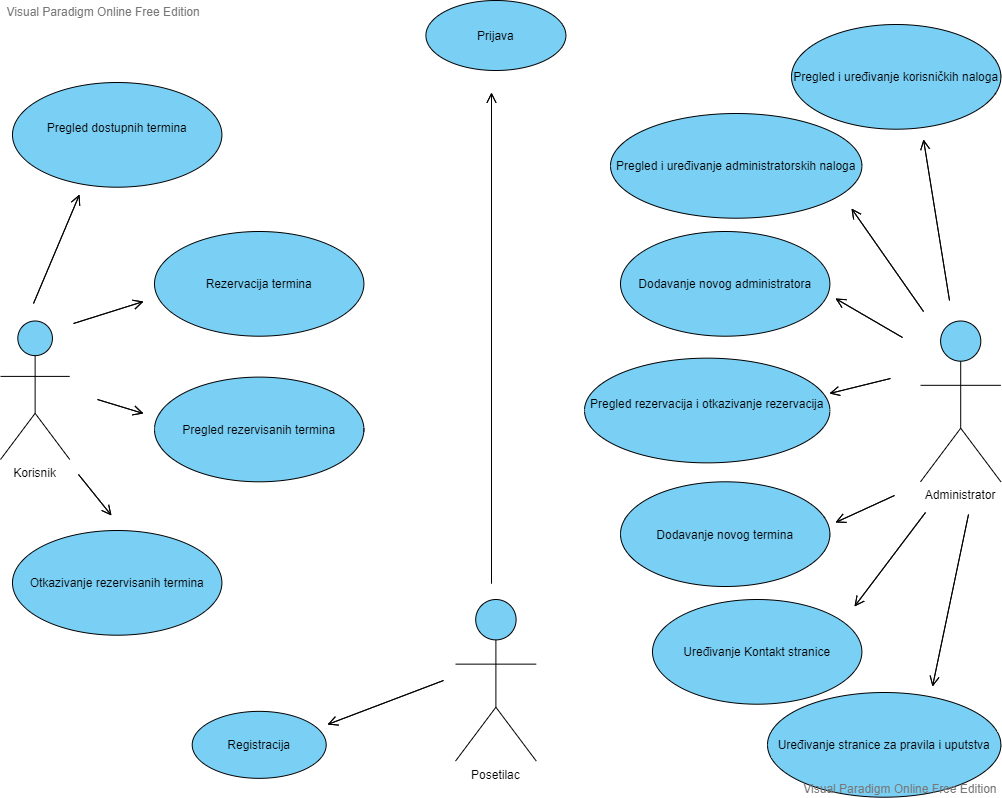
**Posetilac**:

* Prijava
* Registracija

**Korisnik**:

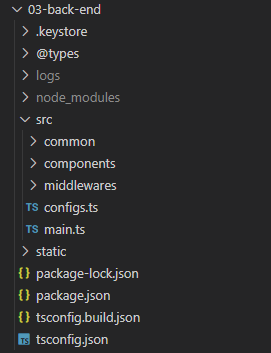
* Pregled dostupnih termina
* Rezervacija termina
* Pregled rezervisanih termina
* Otkazivanje rezervisanih termina

**Administrator**:

* Pregled I uređivanje korisničkih naloga
* Pregled I uređivanje administratorskih naloga
* Dodavanje novog administrator
* Pregled I otkazivanje rezervacija
* Dodavanje novog termina
* Uređivanje kontakt stranice
* Uređivanje stranice za pravila I uputstva
* 

# Back-end

Na slici ispod može se videti struktura back-end dela aplikacije:



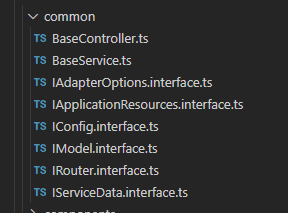
Fajl „**main.ts**“ sadrži main() funkciju u kojoj se pokreće Express server i podešava njegovo funkcionisanje.

Fajl „**configs.ts**“ sadrži DevConfig objekat koji predstavlja implementaciju IConfig interfejsa definisanog u „**common**“ folderu i u njemu se čuvaju parametri za podešavanje aplikacije: informacije za logovanje, parametari za pristup bazi podataka, lista rutera komponenti aplikacije, informacije o mail-u, RSA ključevi za šifrovanje JsonWebTokena.



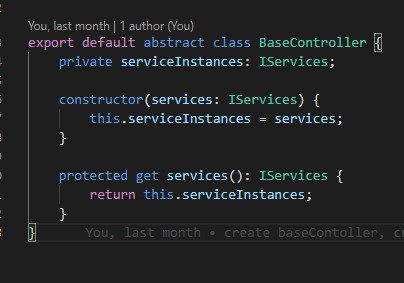
*Deo configs.ts koji se odnosi na bazu podataka I logovanje*

U okviru „**common**“ foldera smešteni su interfejsi i klase koje su zajedničke za više komponenti.



„**BaseService**“ klasa u sebi sadrži funkcije koje pružaju osnovne operacije nad bazom podataka: getAll(), getById(), baseAdd(), baseEdit(), getAllByFieldNameAndValue(), getAllFromTableByFieldNameAndValue(). Svrha ove klase je da se navedene funkcije izmeste u odvojenu klasu kako ne bi morale da se definišu ponaosob u servisu svake komponente (osim ako za to nema izričite potrebe).

Klasa „**BaseController**“ u sebi sadrži polje tipa „**IServices**“, apstraktna klasa koja definiše servise koje će kontroler koristiti.



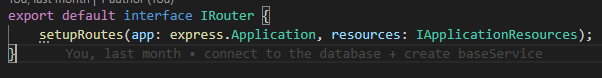
Interfejs „**IAdapterOptions**“ pretstavlja *tagging* interfejs koji će biti ekstendovan od strane pojedinačnih AdapterOptions klasa za svaku komponentu.

Interfejs „**IApplicationsResources**“ sadrži polja databaseConnection, koji je tipa mysql2.Connection, i services polje tipa „**IServices**“. Inicijalizuje se u okviru main() metoda u „**main.ts**“.

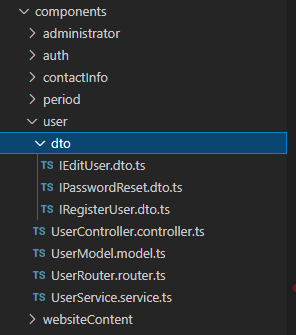
„**IConfig**“ interfejs definiše šta sve konkretna konfiguracija mora da ima. Inicijalizuje se u „**DevConfig**“ objektu u „**configs.ts**“.

Interfejs „**IModel**“ je *tagging* interfejs koji će biti ekstendovan od strane pojedinačnih Model klasa za svaku komponentu.

Interfejs „**IRouter**“ je interfejs će implementirati ruteri pojedinačnih komponenata. U sebi ima jedan metod: setupRoutes() koji kao argumente uzima app, tipa express.Application, i resources tipa „**IApplicationsResources**“.

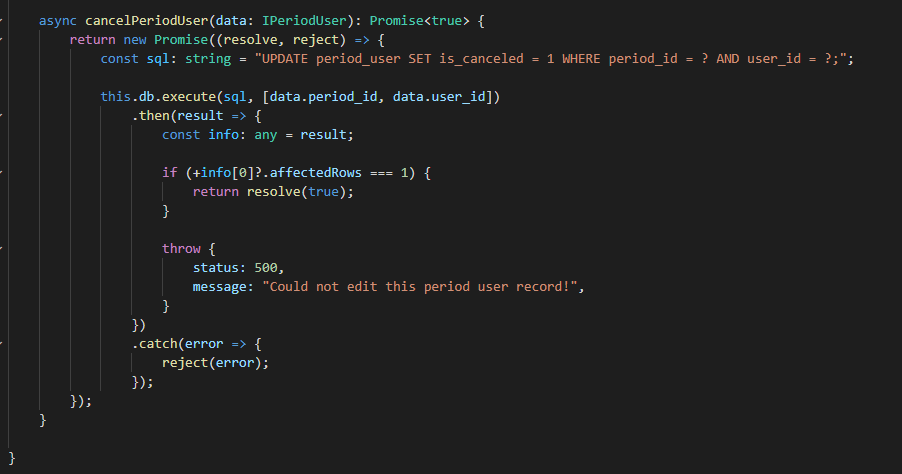


Interfejs „**IServiceData**“ je *tagging* interfejs koji će biti ekstendovan od strane pojedinačnih interfejsa koji će sadržati strukturu podataka za svaki model sa imenima polja koja se poklapaju sa imenima polja u bazi podataka.



Svaka komponenta u sebi ima svoj **servis, kontroler, ruter, model,** kao i folder sa DTO interfejsima (Osim Auth komponente koja ima samo kontroler, ruter i DTO interfejse). **Model** komponente pretstavlja klasu koja preslikava strukturu tabele u bazi podataka.

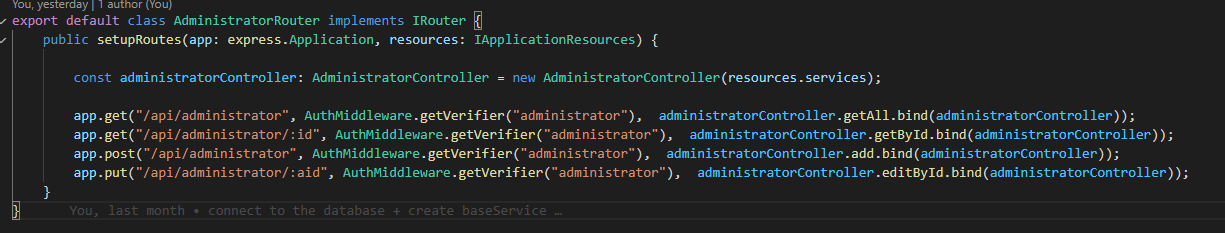
Svaki servis u sebi sadrži implementaciju tableName() i adaptToModel() funkcija „**BaseService**“ klase, moguće dopune ostalih funkcija „**BaseService**“ klase, kao i moguće nove funkcije koje nisu definisane u „**BaseService**“ klasi. „**PeriodService**“, na primer, u sebi sadrži dve dodatne funkcije: funkciju za rezervisanje termina i funkciju za otkazivanje postojećeg termina.



*Funkcija za otkazivanje termina iz “PeriodService”*

**Kontroleri** komponente sadrže funkcije koje pozivaju funkcije **servisa** komponente i *handle*-uju rezultate i moguće greške. Svaka od ovih funkcija kao argumente uzima Request i Response iz Express biblioteke.

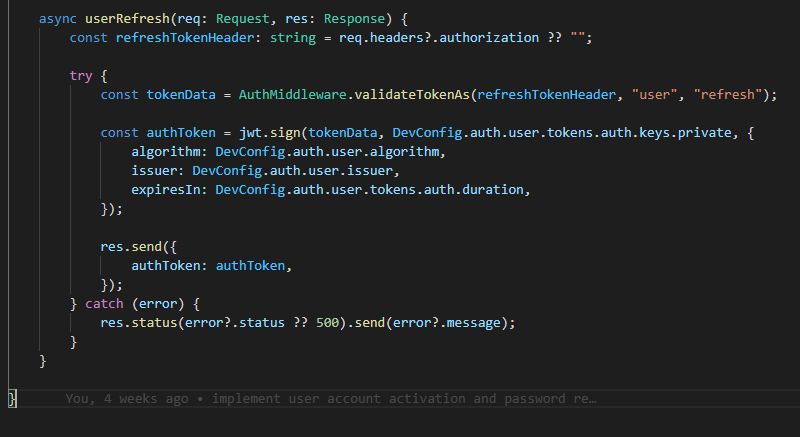
**Ruteri** komponente implementiraju funkciju setupRoutes() „**IRouter**“ interfejsa tako što instanciraju kontroler komponente na osnovu prosleđenih „**IApplicationsResources**“ resursa i express aplikaciji postavljaju pristupne tačke *bind-*ovanjem adekvatne funkcije kontrolera za konkretnu putanju uz preciziranje HTTP metoda koji će se koristiti. Nekim putanjama postavljen je i verifikator iz „**AuthMiddleware**“ koji proverava da li klijent ima pravo pristupa toj putanji.



U okviru DTO interfejsa, osim interfejsa koji pretstavljaju strukturu tabele u bazi, postoje i adekvatni Validatori za podatke koji se prosleđuju. Za validaciju korišćena je AJV Json validator biblioteka.

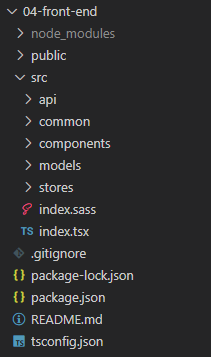
„**AuthController**“ je se razlikuje od kontrolera ostalih komponenti. U sebi sadrži loginAdministrator(), loginUser(), refreshAdministrator() i refreshLogin() funkcije. Svaka od ovih funkcija uzima 2 argumenta: Request i Response iz Express biblioteke. Obe login funkcije treba za početak da provere da li su prosleđeni email (za administratore username) i lozinka ispravni. Ukoliko nisu biće vraćena adekvatna greška, a ukoliko jesu vratiće se objekat sa dva potpisana JSON Web Tokena: authToken i refreshToken (Potpisivanje se vrši na osnovu parametara navedenih u okviru DevConfig objekta u „**configs.ts**“. ). Refresh metode iz autorizacionog dela Request headera uzimaj „**ITokenData**“(Interfejs koji sadrži podatke o ulozi korisnika, njegov id i identity koji može biti username ili email u zavisnosti od uloge) objekat i prosleđuje ga „**AuthMiddleware**“-u na validaciju. Refresh metode kao rezultat vraćaju objekat sa novopotpisanim JSON Web Tokenom.

„**AuthRouter**“ funkcioniše isto kao i ostali ruteri, s tim da nema validacije od strane „**AuthMiddleware**“-a jer je potrebno da obični posetioci web sajta imaju pristup ovim rutama.



# Front-end

Na slici ispod može se videti struktura front-end dela aplikacije:

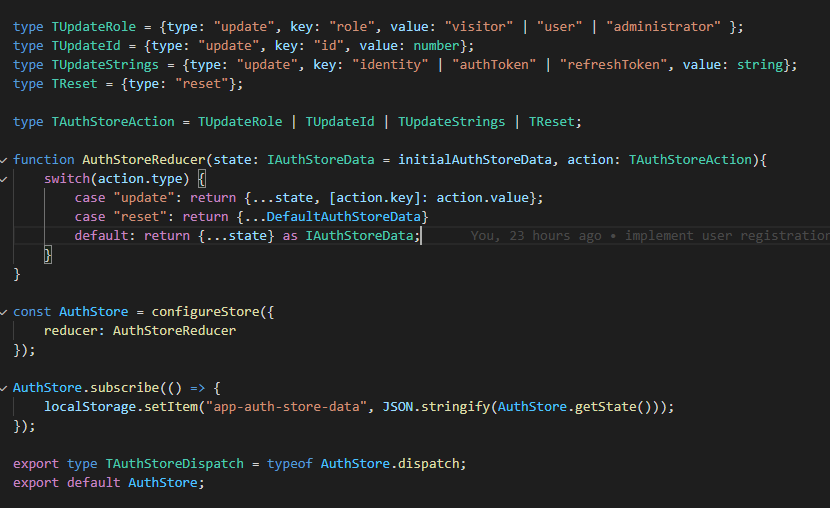


„**index.tsx**“ pretstavlja koren react aplikacije.

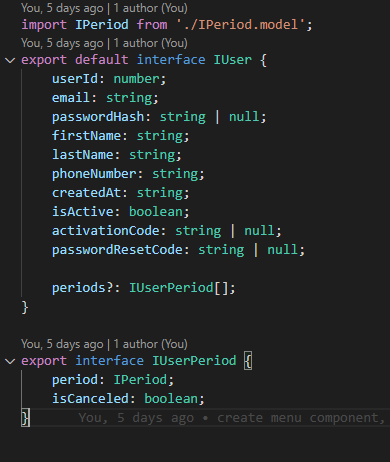
U „**api.ts**“ je izmeštena komunikacija sa back-end delom aplikacije. Eksportuje funkciju api() koju pozivamo u samim komponentama za dohvatanje podataka sa back-end-a. Osim ove funkcije sadrži još i interne funkcije handleApiResponse() i handleApiError() koje se koriste za *handle*-ovanje odgovora sa back-end-a, kao i funkciju refreshToken() koja back-end-u prosleđuje refreshToken iz trenutnog state-a, a kao rezultat vraća authToken dobijen od back-end-a, ukoliko nije došlo do greške u validaciji. Za komunikacju sa back-end delom aplikacije korišćena je Axios biblioteka.



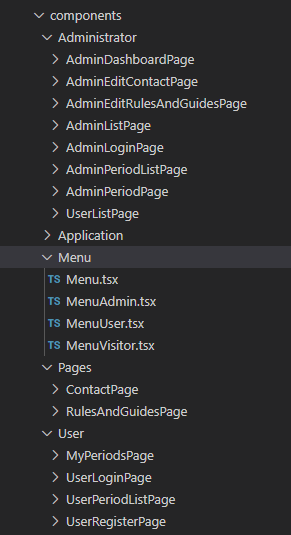
U „**AuthStore.ts**“ nalazi se logika za čuvanje i izmenu state-a aplikacije. Ovde je definisan IAuthStoreData interfejs koji pretstavlja strukturu store objekta. State se čuva u AuthStore objektu, kome se prosleđuje Reducer funkcija, koja će u zavisnosti od tipa akcije koji joj je prosleđen napraviti adekvatne promene u State-u. Zatim se nad AuthStore objektom poziva subscribe() metoda koja će nakon svakog dispatch-a (promene stanja) u localstorage poslati novi IAuthStoreData objekat.



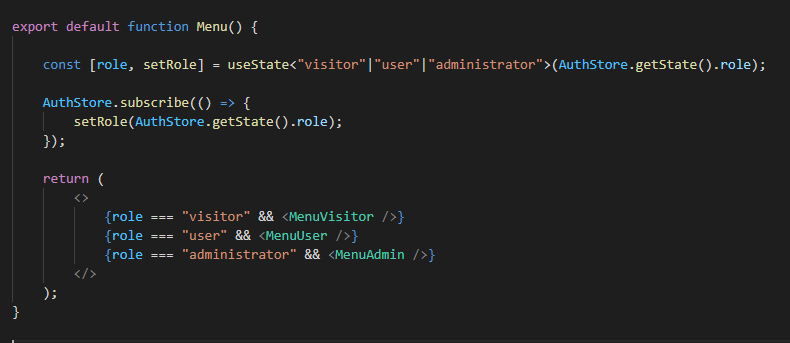
U folderu „**models**“ nalaze se interfejsi u čije objekte će se smeštati podaci dobijeni od strane back-end dela aplikacije.



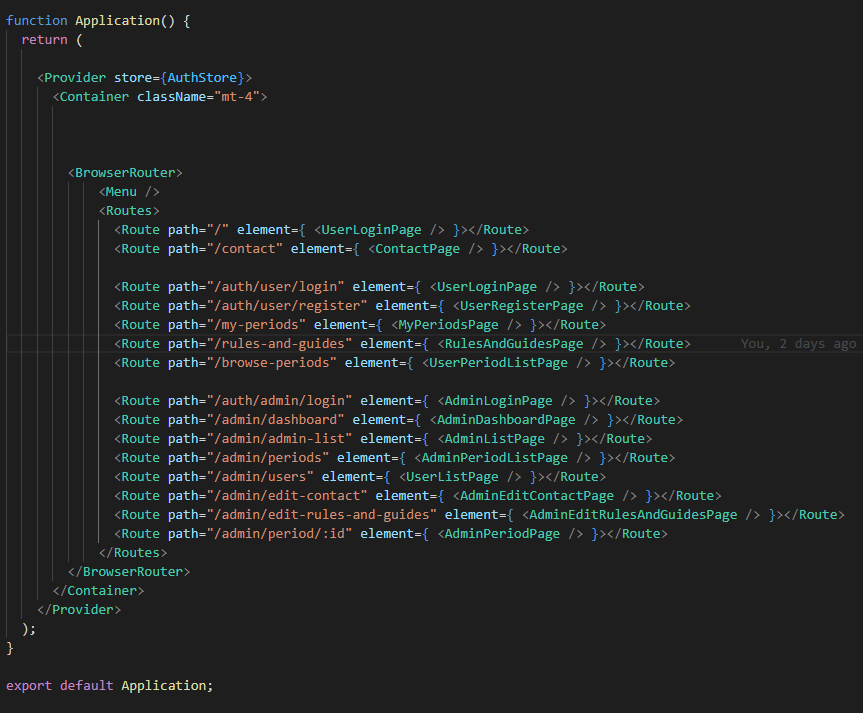
Na slici ispod mogu se videti komponente iz kojih je izgrađena react aplikacija:



Postoje 3 različite komponente za meni za svaku od drugačijih uloga koje korisnik može da ima. „**Menu.tsx**“ će u zavisnosti od trenutne role (koju uzima iz AuthStore state-a) renderovati drugu meni komponentu.



U „**Application.tsx**“ komponenti se *pretplaćujemo* na AuthStore redux store iz „**AuthStore.ts**“ i pravimo rute react aplikacije.



„**AdminDashboardPage.tsx**“ komponenta je administratorska stranica na koju je administrator preusmeren nakon što se uloguje. Prikazuje mu se linkovi koji ga vode na konkretne stranice za upravljanje korisnicima, rezervacijama itd...

„**UserListPage.tsx**“ komponenta je administratorska stranica u kojoj su izlistani svi korisnici aplikacije. Administrator ima mogućnost da promeni status aktivnosti korisnika, da promeni ime ili prezime, da mu promeni lozinku.

„**AdminPeriodListPage.tsx**“ komponenta je administratorska stranica gde se izlistavaju svi termini (izlistavaju se i završeni termini) i administratoru se pruža mogućnost dodavanja novog termina. Sa ove stranice administrator ima mogućnost da pregleda korisnike koji su rezervisali termin klikom na dugme *Pogledaj rezervacije* koje će ga odvesti na „**AdminPeriodPage.tsx**“ komponentu.

U „**AdminPeriodPage.tsx**“ komponenti administratoru su izlistani svi korisnici koji su rezervisali izabrani termin i pruža mu se mogućnost da bilo koju rezervaciju otkaže, ukoliko se termin već nije završio.

„**AdminListPage.tsx**“ komponenta je administratorska stranica u kojoj su izlistani svi administratori. Pruža mogućnost dodavanja novog administratora, promene statusa aktivnosti postojećih administratora, kao i promena lozinke postojećih administratora.

„**AdminLoginPage.tsx**“ komponenta je stranica sa koje administrator može da se prijavi na aplikaciju, ili da odabere da se ipak prijavi kao standardan korisnik.



*AdminLoginPage.tsx*

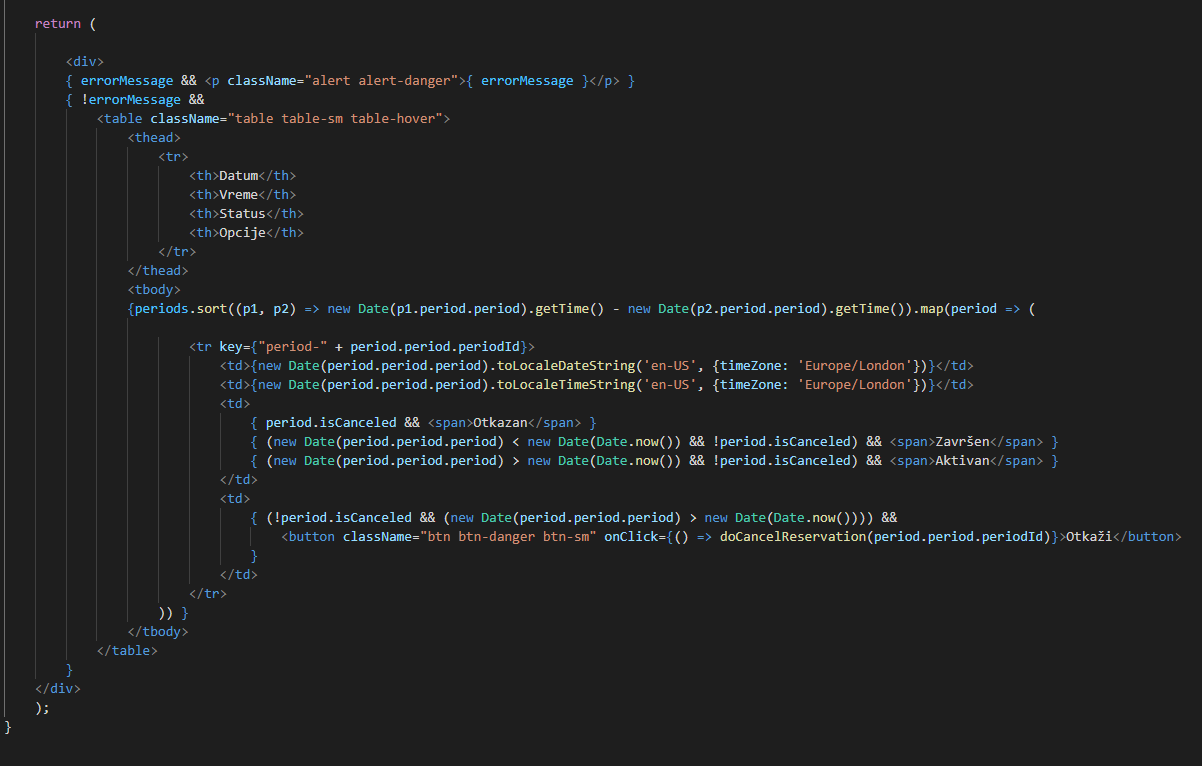
„**AdminEditContactPage.tsx**“ i „**AdminEditRulesAndGuidesPage.tsx**“ su stranice koje administratoru omogućavaju izmenu sadržaja veb sajta, odnosno kontakt stranice i stranice za pravila i uputstva.

„**UserLoginPage.tsx**“ komponenta je stranica sa koje posetilac može da se prijavi na aplikaciju kao običan korisnik, da ode linkom na stranicu za prijavu kao administrator, ili ukoliko nema nalog, da ode na stranicu za registraciju.

„**UserRegisterPage.tsx**“ komponenta je stranica sa koje korisnik može da napravi novi nalog. Nakon što napravi nalog, isti mora aktivirati klikom na link koji je dobio putem mail-a.

„**UserPeriodListPage.tsx**“ komponenta je stranica na kojoj se korisniku izlistavaju svi dostupni termini (koji nisu prošli), prikazuje im se broj slobodnih mesta za svaki termin, i ukoliko je broj slobodnih mesta veći od nule, mogućnost da rezervišu termin. (S obzirom da u projektom zahtevu nije izričito navedeno, odlučeno je da za jedan termin može da se prijavi maksimalno 10 korisnika.)

„**MyPeriodsPage.tsx**“ komponenta je stranica na kojoj se izlistavaju svi termini koje je korisnik rezervisao, i, ukoliko termin nije završen, nudi mu mogućnost otkazivanja termina.



*MyPeriodsPage.tsx*