# ToDo Web Application

## Osnovno o aplikaciji

ToDo Application predstavlja aplikaciju za upravljanje zadacima. Njena osnovna svrha jeste pruzanje korisnicima mogucnosti da kreiraju svoje taskove unutar aplikacije, da upravljaju njima i da vode efikasno evidenciju o tome koji su task izvrsili, koji task je isteko (ako ga nisu oznacili kao izvrsen) i koliko taskova im je preostalo.

Aplikacija podrzava osnovne **CRUD (Create, Read, Update i Delete) operacije,** a osim toga podrzane su uloge **administratora** i **customera.**

U okviru aplikacije realizovano je i filtriranje taskova, koje korisnicima omogucava da filtriraju svoje taskove na osnovu odredjenih svojstava, kao sto su naziv taska, datum, prioritet, kategorija i status.

## Tehnologije koriscene u razvoju

* **ASP .NET Web API (.NET 9):** koriscen je za implementaicju **backend** dela aplikacije. Web API pruza **RESTful** servis koji omogucava kreiranje, citanje, azuriranje i brisanje podataka iz baze
* **React:** koriscen je za implementaciju **frontend** dela aplikacije. Koristio sam ga jer pruza brzu i dinamicnu izradu UI – a.
* **MSSQL:** relaciona baza podataka za cuvanje informacija o korisnicima i taskovima.
* **Cloudinary:** koriscen je za skladistenje profilnih slika korisnika.

## Arhitektura sistema

Arhitektura backend-a organizovana je kao **slojevita (layered) arhitektura.** Podeljena je na 4 glavna sloja + infrastrukturu.

### EL (Entities Layer /Shared Layer):

* + U okviru ovog sloja nalaze se modeli koji predstavljaju entitete u bazi podataka, kao i DTO klase. DTO koristim da smanjim kolicinu podataka koji se prenose izmedju klijenta I servera.

### DAL (Data Access Layer):

* + Sadrzi **ApplicationDbContext** i njegov factory **ApplicationDbContextFactory:** koristi factory pattern (da klasa koja koristi objekat ne mora da zna kako se tacno objekat kreira), koji je biran za kreiranje migracija.
  + **Repository pattern:** ako zelim npr da dobijem sve taskove, ne pisem SQL direktno po svim tabelama, nego pozovem metodu repozitorijuma koja to radi.
  + **UnitOfWork:** koordinise rad vise repozitorijuma kroz jednu transakciju. Znaci ako nesto ne uspe, baza ostaje cista.
  + Odgovoran je za komunikaciju sa bazom

### BLL (Business Logic Layer:

* + Sadrzi servise: **AuthService** (upravljanje korisnicima); **TaskService** (upravljanje taskovima); **EmailService** (reset lozinke); **CloudinaryService** (upload slike).
  + Svaki servis ima svoj **interfejs** (definise sta servis mora da radi) i **implementaciju** (definise kako servis to radi)– **polimorfizam** I **apstrakcija**
  + BLL sadrzi **poslovnu logiku**

### API sloj:

* + Sadrzi **kontrolere** koji primaju HTTP zahteve, validiraju ih i pozivaju odgovarajuci servis iz BLL – a I vracaju odgovor na kraju.
  + **Kontroleri** su ulazna tacka sistema i njihova logika je u BLL

### Frontend

* + Koristim **RTK Query** koji je deo **Redux Toolkit – a** I predstavlja **data – fetching** I **caching biblioteku.** Resava probleme povlacenja/slanja podatka sa/na server.
  + Svaki **endpoint** moze biti **query:** GET, dohvatanje podataka sa servera; **mutation:** POST, PUT, DELETE,menja podatke
  + **Slices:** koristim ih da cuvam sve podatke koji su vezani za UI ili sesiju. Imaju **reducers:** funkcije koje definisu kako se stanje aplikacije menja kao odgovor na akcije.
  + **Store:** centar Redux arhitekture. Sadrzi i **middleware** funkcije koje omogucavaju modifikaciju akcija pre nego sto one stignu do reducer-a.

## Kako se aplikacija pokrece

1. **Preuzimanje projekta sa Github-a**:
   * Pomocu komande: **git clone** <https://github.com/DusanM998/ToDoApplication.git>
   * Pravi lokalnu kopiju repozitorijuma na racunaru
2. **Backend** deo nalazi se u folderu ToDoApp, potrebno je otvoriti solution koji se nalazi u okviru tog foldera (**ToDoApp.slnx**).
3. **Primena migracija.**
   * Posto migracije vec postoje, potrebno ih je samo primeniti, tako sto se pozicionirate u okviru **DAL** (jer se tu nalazi ApplicationDbContext) i migracije. (cd DAL)
   * Zatim je potrebno pokrenuti komandu unutar Developer Power Shell ili Package Manage Console: **dotnet ef database update** kako bi se kreirala baza i primenile migracije
4. **Pokretanje backend servera:**
   * Izabrati ToDoApp projekat i pokrenuti ga komandom dotnet run
   * Backend je metode mogu se testirati na: [**https://localhost:7070/scalar/v1**](https://localhost:7070/scalar/v1)
5. **Frontend**
   * Frontend deo aplikacije nalazi se u folderu ToDoAppFrontend
   * Potrebno je u Visual Studio Code otvoriti istoimeni workspace koji se nalazi u tom folderu
   * Zatim se otvoriti terminal i pozicionirati se u **to-do-app** (**cd to-do-app**)
   * Potrebno je zatim instalirati sve dependencies: **npm install –save –legacy-peer-deps** (--legacy-peer-deps se koristi da se izbegnu konflikti izmedju pojedinih verzija biblioteka)
   * Frontend se pokrece default na: **http://localhost:5173/**