Implementacija cross-platform web aplikacije korišćenjem Electron framework-a

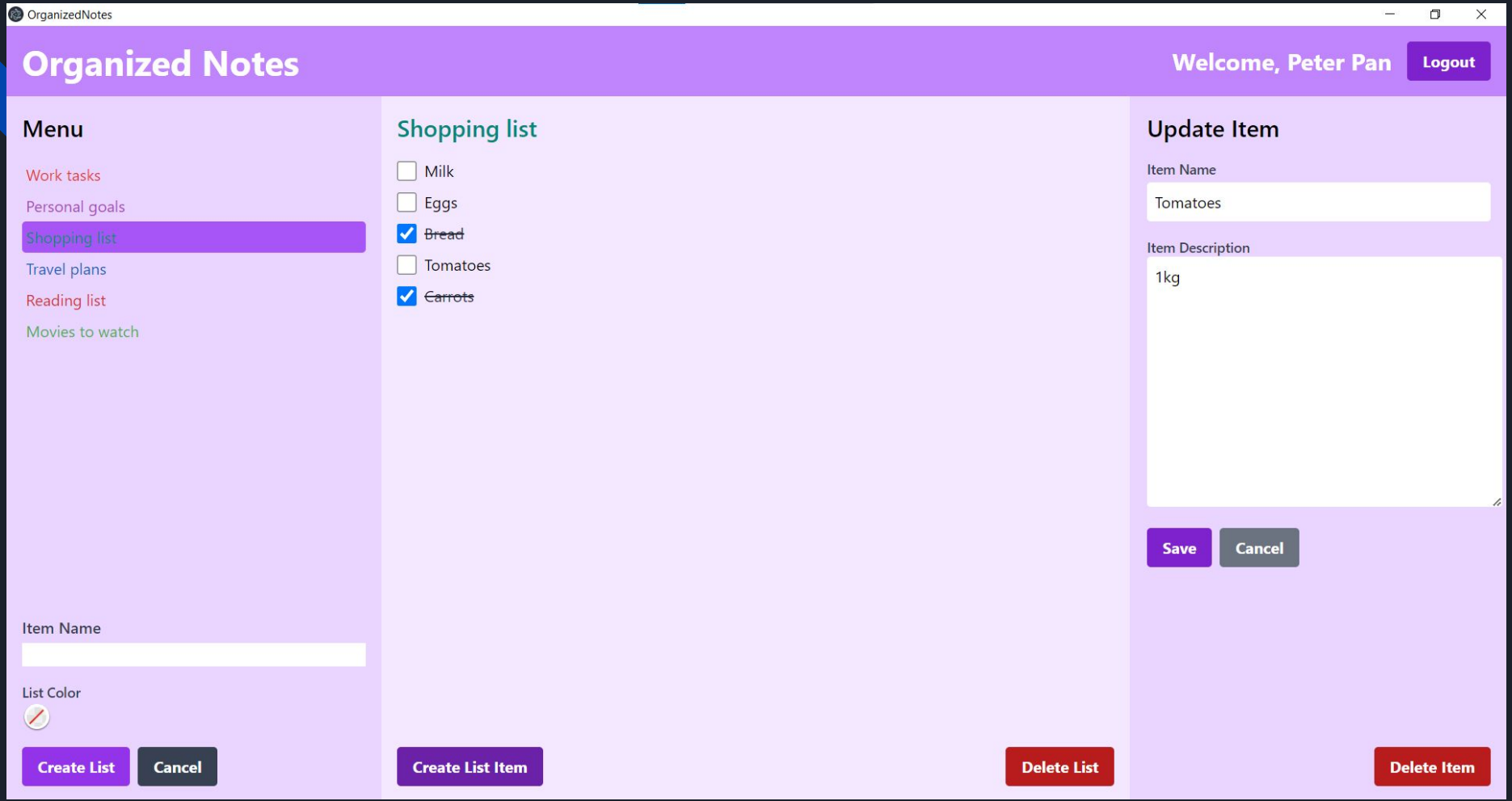
Milja Pejčić 1710

Tijana Stanković 1726

Aplikacija “Organized Notes”

U okviru projekta, razvijena je aplikacija 'Organized Notes', koja kroz implementirane funkcionalnosti replicira karakteristike klasične To-Do aplikacije.

Unutar aplikacije moguće je kreiranje liste, kao i njeno brisanje. Moguće je kreiranje taskova unutar liste, njihova izmena, kao i brisanje. Takođe je moguće označiti određeni task kao završen.


OrganizedNotes
Organized Notes
Welcome, Peter Pan
Logout
Menu
Work tasks
Personal goals
Shopping list
Travel plans
Reading list
Movies to watch
Item Name
List Color
Create List
Cancel
Shopping list
Milk
Eggs
Bread
Tomatoes
Carrots
Create List Item
Delete List
Update Item
Item Name
Tomatoes
Item Description
1kg
Save
Cancel
Delete Item

FRONTEND

- Angular
- Tailwind

BACKEND

- Node.js
- MongoDB

Electron

Šta je Angular?

Angular je moćan open-source frontend web framework za razvoj i održavanje aplikacija, koji je kreiran od strane kompanije Google.

Angular omogućava kreiranje dinamičkih single page web aplikacija (SPA), gde se sadržaj dinamički učitava, što kao rezultat ima glatko i jednostavnije korisničko iskustvo.

Olakšava razvoj dinamičkih web aplikacija putem svojih snažnih funkcija, omogućavajući razvojnim inženjerima da kreiraju skalabilne, održive i aplikacije bogate raznim funkcionalnostima.

Ključni koncepti Angular aplikacije

- ***Komponente***: Angular aplikacije su organizovane u komponente, koje predstavljaju modularne delove korisničkog interfejsa. Svaka komponenta ima svoju funkcionalnost i izgled.
- ***Moduli***: Komponente su zatim organizovane u module. Moduli grupišu slične komponente i funkcionalnosti, čime se poboljšava organizacija i održavanje koda.
- ***Servisi***: Servisi pružaju zajedničku funkcionalnost i deljenje podataka između komponenata. Oni omogućavaju reaktivnost i efikasnu komunikaciju unutar aplikacije.
- ***Direktive***: Angular direktive su posebne oznake u HTML-u koje omogućavaju manipulaciju DOM-om. Primeri uključuju *ngIf za uslovno prikazivanje i *ngFor za iteraciju kroz liste.

Angular CLI

Angular Command Line Interface (CLI) je moćan alat koji olakšava razvoj, testiranje i održavanje Angular aplikacija. Pomaže programerima da brže kreiraju i konfigurišu projekte, generišu komponente, module i druge strukture aplikacije.

Korišćenjem ovog alata moguće je kreirati novi projekat, pokrenuti razvojni server i pratiti promene u projektu, generisati različite komponente u okviru projekta, generisati Angular servise, itd.

Novi projekat se kreira korišćenjem naredbe *ng new project-name*.

Za pokretanje razvojnog servera i praćenje promena u projektu koristi se naredna *ng serve*. Ovom naredbom se omogućava pristup aplikaciji na adresi *http://localhost:4200/*.

Kreiranje projekta

Na samom početku razvoja Angular aplikacije, potrebno je izvršiti nekoliko koraka:

1. Instalacija Node.js i npm: Na početku je potrebno instalirati Node.js, koji automatski uključuje npm (Node Package Manager).
2. Globalna instalacija Angular CLI: Potrebno je otvoriti terminal ili komandnu liniju i izvršiti sledeću komandu kako bi se globalno instalirao Angular CLI:

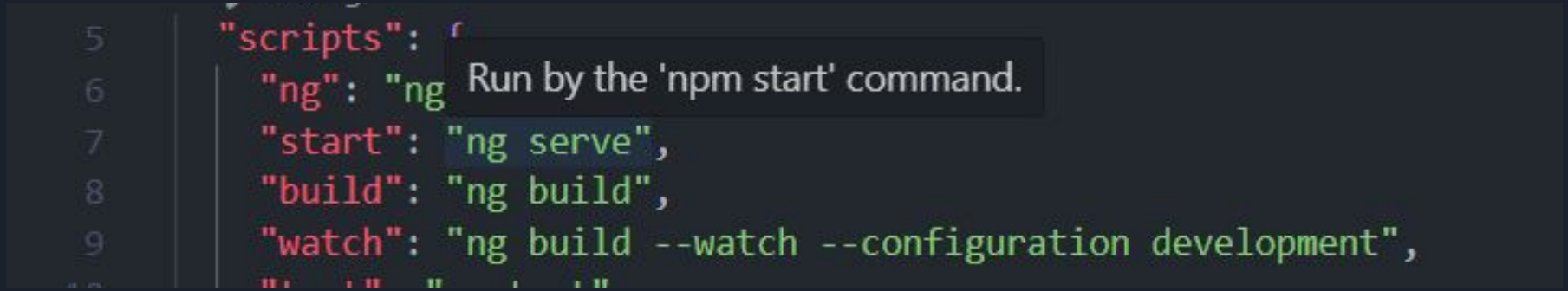
   npm install -g @angular/cli
3. Kreiranje novog Angular projekta: Kada je Angular CLI instaliran, novi Angular projekat se kreira komandom:

   ng new organized-notes
4. Pokretanje aplikacije: Za pokretanje aplikacije, potrebno je otvoriti terminal u direktorijumu novog projekta, u ovom slučaju je to direktorijum “organized-notes”, i izvršiti sledeću komandu:

   npm start

   Ova komanda će pokrenuti Angular aplikaciju koristeći skriptu definisanu u package.json koja okida ng serve.

Skripta definisana u package.json



Kreiranje komponenata i servisa

Komande za generisanje komponenata i servisa u Angular aplikaciji omogućavaju efikasno organizovanje i proširivanje koda.

- Za generisanje nove komponente, koristi se sledeća Angular CLI komanda:

  ng generate component [putanja/ime_komponente]

Npr: *ng generate component components/side-menu*

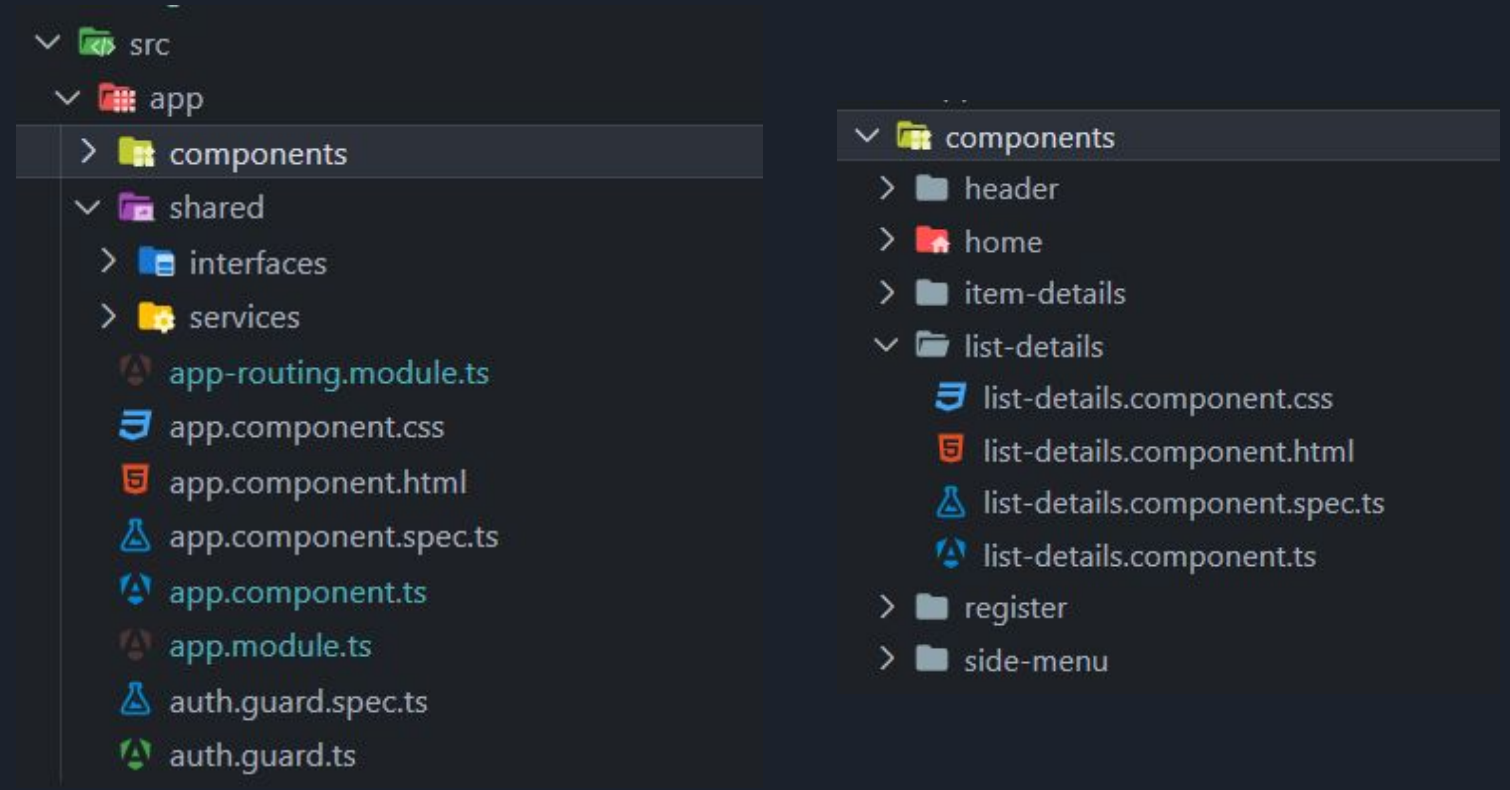
Treba napomenuti da se komponente sastoje od tri ključna fajla: .ts, .css i .html.

- Za generisanje novog servisa, koristi se sledeća Angular CLI komanda:

  ng generate service[putanja/ime_servisa]

Npr: *ng generate service shared/services/apiService*, gde putanja predstavlja relativnu putanju od app foldera.

Struktura projekta



Prednosti Angular framework-a

- ***@Input* i *@Output*** su dekoratori koji se koriste za postavljanje veza između roditeljskih i dečijih komponenti kako bi se omogućio prenos podataka.
- ***Signali*** u Angular aplikaciji često se ostvaruju kroz upotrebu EventEmitter objekta, koji je deo Angular-ove biblioteke za rad sa događajima. Kroz ovakav mehanizam, komponente u Angular aplikaciji mogu komunicirati i deliti informacije putem signala koji se emituju i koje druge komponente prate.
- Angular pruža niz ***ugrađenih modula***, od kojih su neki korišćeni u okviru projekta: *HttpClientModule*, koji omogućava Angular aplikacijama da šalju HTTP zahteve, što je od suštinskog značaja za komunikaciju sa serverom, kao i *FormsModule* koji pruža podršku za rad sa formama, olakšavajući manipulaciju korisničkim unosima. Zahvaljujući ovome ne postoji česta potreba za dodavanjem eksternih biblioteka ili instaliranjem dodatnih paketa.

  *ngx-colors

RxJS

U Angular aplikacijama, RxJS (Reactive Extensions for JavaScript) je često korišćena biblioteka koja omogućava reaktivno programiranje. RxJS pruža mnoštvo operatorskih funkcija koje olakšavaju manipulaciju i obradu asinhronih događaja, kao i upravljanje tokovima podataka.

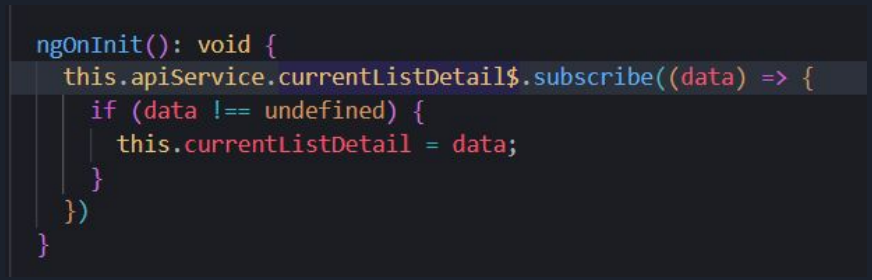
Bitni koncepti korišćeni u projektu:

- *Observable*
- *Subject*
- *BehaviorSubject*

```
currentListDetail = new BehaviorSubject<ListDetails | null>(null);
currentListDetail$ = this.currentListDetail.asObservable();
```

RxJS

U okviru aplikacije se RxJS koristi za praćenje promena između komponenata, jedan primer u projektu bi bio da se detalji o listi koji se prikazuju u komponenti *list-details* reaktivno menjaju.



```
ngOnInit(): void {
  this.apiService.currentListDetail$.subscribe((data) => {
    if (data !== undefined) {
      this.currentListDetail = data;
    }
  })
}
```

U Angular-u, *subscribe* funkcija se koristi za pretplatu (subscription) na Observable objekat. Kada se komponenta ili servis pretplati, ona počinje da osluškuje emitovanje vrednosti iz tog Observable-a. Subscribe funkcija prima jednu ili više callback funkcija koje će se izvršiti kada Observable emituje vrednost, obaveštenje ili grešku.

Šta je Tailwind CSS?

Tailwind je moderni CSS framework koji pruža set korisnih klasa za stilizaciju, eliminišući potrebu za pisanjem klasičnog CSS-a. Dakle, Tailwind omogućava brzo stilizovanje HTML komponenti direktno kroz HTML tagove koristeći princip klasa kao atribute tagova.

```
<h1 class="text-4xl font-bold text-white">Organized Notes</h1>
```

Prednosti su jednostavnost prilagođavanja i fleksibilnost (moguće je jako brzo prilagoditi stilove prema potrebama projekta), responsivnost (jednostavna implementacija responsivnog dizajna), primena light/dark teme...

Integracija Tailwind-a u projekat

npm install tailwindcss
Ova komanda se koristi za instalaciju Tailwind CSS-a.

npx tailwindcss init
Ova komanda inicijalizuje Tailwind CSS konfiguraciju u projektu i dodaje tailwind.config.js fajl u radni direktorijum.

U tailwind.config.js fajlu potrebno je postaviti putanje do svih fajlova u kojima je moguće koristiti tailwind klase.

```
module.exports = {
  content: ["./src/**/*.{html,js,ts}"],
```

Nakon konfiguracije, u styles.css fajlu se importuju paketi neophodni za konfiguraciju Tailwind-a.

styles.css 3

UI > organized-notes > src > styles.css

```
1  @tailwind base;
2  @tailwind components;
3  @tailwind utilities;
```

Šta je NodeJS?

Node.js je open-source, serverski framework zasnovan na JavaScript-u, koji omogućava izvršavanje JavaScript koda na serverskoj strani.

Razvijen je kako bi omogućio efikasno izvođenje real-time aplikacija i API-ja.

Glavne karakteristike Node.js-a

1. JavaScript na serverskoj strani:

   Node.js omogućava korišćenje JavaScript-a i na serverskoj strani, pružajući jedinstveni jezik za programiranje kako na klijentskoj, tako i na serverskoj strani.

2. Event-Driven arhitektura:

   Node.js koristi event-driven arhitekturu, što znači da reaguje na događaje i obrađuje ih asinhrone, čineći ga efikasnim za obradu velikog broja istovremenih zahteva.

3. Neblokirajući I/O:

   Node.js podržava neblokirajuće I/O operacije, što znači da ne čeka na završetak operacija i može efikasno obrađivati više zahteva u isto vreme.

4. Brzo izvršavanje koda:

   Zahvaljujući V8 JavaScript engine-u, Node.js pruža brzo izvršavanje koda, čineći ga pogodnim za performantne aplikacije.

Express.js

Express.js je popularni web framework za Node.js, dizajniran za olakšavanje izrade web aplikacija i API-ja.

Instalacija Express.js-a se vrši pomoću ***npm install express*** komande.

Ključne karakteristike i prednosti Express.js-a su to što:

- pruža osnovne alatke i strukturu, ali ostavlja mnogo slobode programeru u izboru i organizaciji komponenti aplikacije;
- omogućava lako prilagođavanje HTTP odgovora koje šalje klijentu (može se postaviti status kod, dodati HTTP zaglavlje ili poslati određeni tip sadržaja)
- definisanje ruta u Express.js-u je veoma jednostavno (korišćenjem post() i get() metoda, mogu se lako kreirati rute za različite HTTP metode)
- olakšava izvlačenje podataka iz HTTP zahteva (podaci iz tela POST zahteva mogu se lako dohvatiti kroz *req.body*)
- dizajniran je za brzinu i efikasnost, što ga čini pogodnim za izgradnju visoko performantnih web aplikacija i API-ja

Kreiranje backend projekta

- Instalacija Node.js: Node.js je neophodan za izvršavanje JavaScript koda na serverskoj strani.
- Inicijalizacija projekta: Koristi se ***npm init*** komanda da bi se inicijalizovao novi projekat. Na ovaj način se kreira package.json fajl sa osnovnim informacijama o projektu.
- Kreiranje app.js fajla: Kreira se app.js fajl koji će predstavljati početnu tačku Node.js backend projekta. Ovde se postavlja logika za server i rute.
- Instalacija Express-a: Potrebno je instalirati Express kao web framework za Node.js korišćenjem ***npm install express*** komande. Express olakšava rukovanje HTTP zahtevima i rastavljanje aplikacije na modularne delove.

Takođe, dobro je uključiti u program i Nodemon paket koji omogućava automatsko ponovno pokretanje servera svaki put kada se izmene izvrše. Ovo je posebno korisno tokom razvoja. Instaliranje se vrši komandom ***npm install -D nodemon***.

Struktura Express.js projekta

Projekat je podeljen na tri osnovna dela u Angular-u: routes, models i controllers.

Routes definiše putanje (rute) za pristupanje određenim resursima u aplikaciji. Ovde se nalaze definicije ruta koje vode do odgovarajućih kontrolera, omogućavajući navigaciju kroz aplikaciju.

Models definiše strukturu podataka (modeli) koji se koriste unutar aplikacije. Ovde se nalaze deklaracije modela koji opisuju podatke koje aplikacija koristi.

Controllers sadrži logiku koja upravlja podacima i poslovnim operacijama. Ovde se nalazi implementacija funkcija (metoda) koje obrađuju poslovnu logiku aplikacije, povezujući se sa servisima i modelima.

Kratko objašnjenje za flow u app.js:

* app.use('/api/item', item): Definiše da se za sve putanje koje počinju sa /api/item, koriste rute definisane u itemRoutes.

* const item = require('./routes/itemRoutes'): Importuje rute definisane u fajlu itemRoutes i dodeljuje ih promenljivoj item.

* U fajlu routes/itemRoutes: Definiše se ruta router.put('/updateItem/:itemId', UpdateItem) koja omogućava HTTP PUT zahtev na putanji /updateItem/:itemId. Ova ruta je povezana sa UpdateItem metodom u odgovarajućem kontroleru.

MongoDB

MongoDB je popularni, besplatan i open-source sistem za upravljanje bazama podataka (DBMS) koji spada u kategoriju NoSQL (Not Only SQL) baza podataka.

Ključne karakteristike:

- Fleksibilnost schema: MongoDB je NoSQL baza podataka, što znači da nema unapred definisanu šemu podataka. Ovo omogućava veću fleksibilnost, jer različiti dokumenti u istoj kolekciji mogu imati različite atribute.
- Dokument-orijentisana struktura: Podaci se čuvaju u BSON formatu, što je binarna reprezentacija JSON-a. Svaki dokument predstavlja jedan zapis podataka, s atributima i njihovim vrednostima. Dokumenti se grupišu u kolekcije.
- Podrška za replikaciju i sharding: MongoDB pruža mehanizme za replikaciju podataka, čime se obezbeđuje visoka dostupnost i otpornost na otkaze. Takođe podržava horizontalno skaliranje pomoću shardinga, omogućavajući rad s velikim skupovima podataka.
- Mogućnost upita korišćenjem JavaScript-a: Upiti se izražavaju pomoću JavaScript objekata i omogućavaju bogat jezik upita. MongoDB takođe podržava indekse koji poboljšavaju performanse upita.
- Veličina dokumenta: MongoDB ima ograničenje veličine dokumenta, ali pruža podršku za skladištenje velikih dokumenata putem GridFS, posebnog protokola za rad s velikim binarnim datotekama.

Kreiranje MongoDB baze podataka

Mongo baza podataka nije samo obična baza, već MongoDB Cluster, što znači da je distribuirana arhitektura podataka. Prednost ovoga jeste to sto je omogućeno horizontalno skaliranje i bolja otpornost na kvarove.

Umesto tabela, MongoDB koristi kolekcije za organizaciju podataka.

U okviru projekta postoje tri kolekcije:

* Items: Čuva informacije o pojedinačnim taskovima u našem sistemu. Ime taska, opis, i druge karakteristike.

* Lists: Pamti podatke o listama, gde svaka lista može sadržati različite taskove. Veza između Items i Lists se postiže referenciranjem id-a liste u okviru jednog Item-a.

* Users: Sadrži informacije o korisnicima sistema, kao što su korisničko ime i enkriptovana lozinka.

Mongoose

Mongoose je npm paket za razvoj Node.js backend aplikacija koji olakšava interakciju servera sa MongoDB bazom podataka i pruža alate za modelovanje podataka. Omogućava kreiranje Schema i Modela, što olakšava rad sa MongoDB kolekcijama i manipulacijom podataka u bazi.

Modeli

Modeli predstavljaju strukturu podataka i ponašanje entiteta koji će se koristiti u aplikaciji. U kontekstu Node.js, modeli se često koriste za mapiranje podataka iz baze podataka.

Mongoose je ODM (Object Data Modeling) biblioteka za MongoDB i Node.js. Omogućava jednostavnu komunikaciju sa MongoDB bazom i definisanje modela.

Povezivanje sa bazom: Prvo se vrši povezivanje na MongoDB bazu pomoću Mongoose-ove connect metode. U kontrolerima, modeli se koriste za izvršavanje operacija nad bazom.

Modeli odražavaju strukturu i odnose u bazi podataka. Ovaj pristup omogućava efikasno i jasno upravljanje podacima u Node.js aplikacijama, a Mongoose pojednostavljuje interakciju sa MongoDB bazom podataka, čineći modelovanje i rad sa podacima intuitivnim.

Na primeru listModel.js:

* type: Definiše tip podatka. Na primer, String označava da će vrednost biti tipa string.

* required: Postavlja da li je određeno polje obavezno prilikom kreiranja dokumenta. U primeru, name mora biti prisutno.

* unique: Obezbeđuje da vrednost polja bude jedinstvena u okviru cele kolekcije. Na primer, svaka lista mora imati jedinstveno ime.

* ref(userId): Koristi se za definisanje referenciranja drugog modela. U primeru, userId je referenca na model "User".

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const listSchema = new Schema({
    name: {
        type: String,
        required: true,
        unique: true
    },
    color: {
        type: String,
    },
    userId: {
        type: String,
        required: true
    }
})

const List = mongoose.model('list', listSchema);
```

Veza između Angular i Express aplikacije

- Da bi došlo do komunikacije između frontend-a i backend-a, u api servisu na frontendu potrebno je postaviti odgovarajući Api url, kao i inject-ovati HttpClient koji obezbeđuje Angular

```
readonly apiUrl = 'http://localhost:5000/api/';
```

```
constructor(private http: HttpClient, private router: Router, private authService: AuthService) { }
```

Na slici ispod je primer korišćenja http klijenta koji obezbeđuje metode kao što su get, put, delete...

Svakoj od metoda potrebno je postaviti tip odgovora sa servera (u ovom slučaju ListDetails), a prosleđuje joj se kao prvi parametar ruta koja će aktivirati odgovarajuću metodu na backend-u u odgovarajućem kontroleru, a drugi parametar je opcioni, i on podrazumeva body koji se šalje u sklopu zahteva ka bekendu (ukoliko je potrebno slati podatke preko body-ja). Metode koje obezbeđuje http klijent vraćaju Observables, i zbog toga se koristi subscribe za praćenje odgovora sa servera.

```
getList(listId: string){
  this.http.get<ListDetails>(this.apiUrl+`list/getList/${listId}`).subscribe((data)=>{
    this.currentListDetail.next(data);
    this.router.navigate([`list/${data.list._id}`])
  })
}
```

Šta je Electron?

Electron je open-source framework koji omogućava razvoj desktop aplikacija koristeći web tehnologije kao što su HTML, CSS i JavaScript. Ovaj framework omogućava kreiranje istovremenih aplikacija za Windows, macOS i Linux operativne sisteme.

Njegova jednostavnost, modularnost i cross-platform podrška čine ga popularnim izborom među programerima koji žele brzo i efikasno razvijanje desktop aplikacije.

Aplikacije koje koriste Electron

Ključne karakteristike Electron-a

- Cross-platform razvoj:
  Electron omogućava razvoj desktop aplikacija koje rade na više operativnih sistema sa zajedničkim kodom.
- Web tehnologije:
  Koristi poznate web tehnologije poput HTML, CSS i JavaScript-a, a podržava i popularne frontend framework-e, čime se još više olakšava razvoj aplikacija.
- Chromium engine:
  Electron koristi Chromium engine, što omogućava visokokvalitetno korisničko iskustvo.
- Jednostavnije održavanje:
  Ažuriranja i ispravke jednom se primenjuju na sve platforme.

Šta je Chromium?

Chromium u okviru Electron-a predstavlja snažan alat za razvoj desktop aplikacija koje imaju funkcionalnosti i performanse web pregledača, čime se postiže jednostavnost razvoja i konzistentnost korisničkog iskustva na različitim platformama.

Integracija Angular-a sa Electron-om

Najpre je potrebno instalirati electron u sklopu Angular projekta pomoću komande:

npm install electron

Nakon toga, da bi electron imao putanju do ulazne tačke u projekat, u index.html je potrebno dodati ovu liniju koda, konkretno između head tagova:

```
<base href="./">
```

Iako Angular aplikacija koristi typescript, electron je zasnovan na Node.js-u, tako da je potrebno u root folderu projekta kreirati novi javascript fajl (ne typescript!)

main.js

UI > organized-notes > main.js > ...

```
1   const {app, BrowserWindow} = require('electron');
2
3   let appWindow;
4
5   function createWindow () {
6       appWindow = new BrowserWindow();
7       appWindow.removeMenu();
8       appWindow.maximize();
9       appWindow.loadURL('http://localhost:4200');
10
11      appWindow.on('closed', function (){
12          appWindow = null
13      })
14  }
15
16  app.whenReady().then(()=>{
17      createWindow()
18  })
```

Pakovanje Electron aplikacije u izvršni fajl

electron-packager je alat koji se koristi za pakovanje Electron aplikacija u izvršne fajlove (executable files) za različite operativne sisteme kao što su Windows, macOS i Linux. Ovaj alat omogućava programerima da distribuiraju svoje Electron aplikacije kao samostalne, izvršne fajlove koje korisnici mogu pokrenuti bez potrebe za instalacijom Electron-a ili drugih dodatnih paketa.

Korišćenje electron-packager npm paketa

Najpre je u frontend direktorijumu (UI/organized-notes) potrebno instalirati npm paket electron-packager, i to komandom:

npm install electron-packager

Nakon toga, potrebno je izvršiti sledeću komandu:

npx electron-packager ./ OrganizedNotes --platform-win32 --overwrite

‘./’ je putanja koja vodi do package.json fajla frontend projekta kako bi se sve zavisnosti projekta mapirale preko electron-packager-a

‘OrganizedNotes’ je ime koje dajemo izvršnom fajlu

‘--platform-win32’ je flag kojim se označava electron-packager-u da za Windows platformu želimo da kreiramo izvršni fajl

‘--overwrite’ je flag kojim se označava da ukoliko je ova komanda već izvršena i pokušamo ponovo da je izvršimo, doći će do overwrite-a prethodnih fajlova

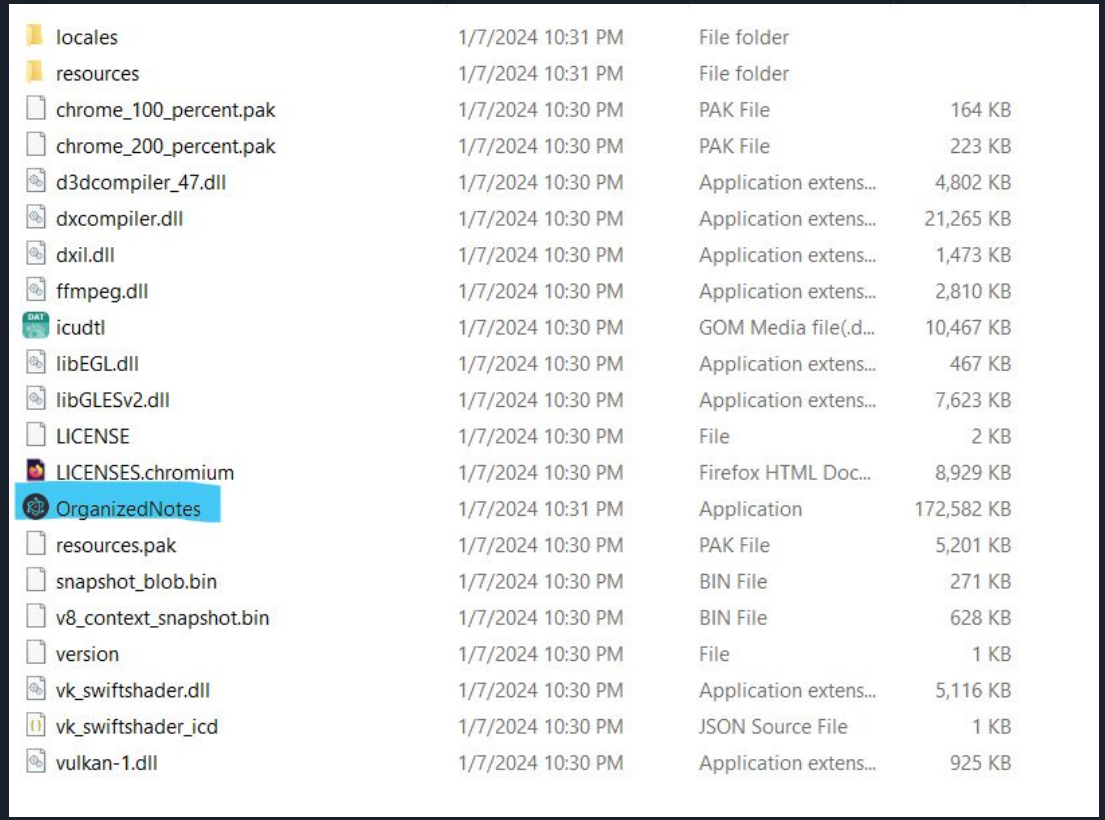
Napomene za korišćenje electron-packager-a

Platform flag može biti:

1. *win32* za Windows
2. *darwin* ili *mas* za macOS
3. *linux* za Linux

Treba napomenuti da postoji ograničenje što se tiče kreiranja mac verzije. Naime, ukoliko radite na Windows operativnom sistemu, nije moguće upakovati aplikaciju za mac platformu.

Izvršenjem prethodno pomenute komande generisan je folder u kome se nalazi naš izvršni fajl, čime smo uspešno kreirali desktop aplikaciju.



locales	1/7/2024 10:31 PM	File folder	
resources	1/7/2024 10:31 PM	File folder	
chrome_100_percent.pak	1/7/2024 10:30 PM	PAK File	164 KB
chrome_200_percent.pak	1/7/2024 10:30 PM	PAK File	223 KB
d3dcompiler_47.dll	1/7/2024 10:30 PM	Application extens...	4,802 KB
dxcompiler.dll	1/7/2024 10:30 PM	Application extens...	21,265 KB
dxil.dll	1/7/2024 10:30 PM	Application extens...	1,473 KB
ffmpeg.dll	1/7/2024 10:30 PM	Application extens...	2,810 KB
icudtl	1/7/2024 10:30 PM	GOM Media file(.d...	10,467 KB
libEGL.dll	1/7/2024 10:30 PM	Application extens...	467 KB
libGLESv2.dll	1/7/2024 10:30 PM	Application extens...	7,623 KB
LICENSE	1/7/2024 10:30 PM	File	2 KB
LICENSES.chromium	1/7/2024 10:30 PM	Firefox HTML Doc...	8,929 KB
OrganizedNotes	1/7/2024 10:31 PM	Application	172,582 KB
resources.pak	1/7/2024 10:30 PM	PAK File	5,201 KB
snapshot_blob.bin	1/7/2024 10:30 PM	BIN File	271 KB
v8_context_snapshot.bin	1/7/2024 10:30 PM	BIN File	628 KB
version	1/7/2024 10:30 PM	File	1 KB
vk_swiftshader.dll	1/7/2024 10:30 PM	Application extens...	5,116 KB
vk_swiftshader_icd	1/7/2024 10:30 PM	JSON Source File	1 KB
vulkan-1.dll	1/7/2024 10:30 PM	Application extens...	925 KB

Hvala na pažnji!