|  |
| --- |
| cd app  md modules  cd modules  ng g module shared  ng g module public --routing  ng g module admin --routing  cd admin  ng g component cities  cd cities  ng g component cities-edit  └───app  ├───auth-guards  ├───endpoints  │ ├───auth-endpoints  │ ├───city-endpoints  │ └───country-endpoints  ├───helper  ├───modules  │ ├───admin  │ │ ├───admin-error-page  │ │ ├───admin-layout  │ │ ├───cities  │ │ │ └───cities-edit  │ │ ├───dashboard  │ │ ├───destination  │ │ └───reservation  │ ├───auth  │ │ ├───auth-layout  │ │ ├───forget-password  │ │ ├───login  │ │ ├───logout  │ │ ├───register  │ │ └───two-factor  │ ├───public  │ │ ├───about  │ │ ├───blog  │ │ ├───contact-us  │ │ ├───home  │ │ ├───public-layout  │ │ └───travels  │ └───shared  │ └───unauthorized  └───services  └───auth-services  └───dto |

# Angular moduli, komponente, lazy loading

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Endpoint interface

A screenshot of a chat

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

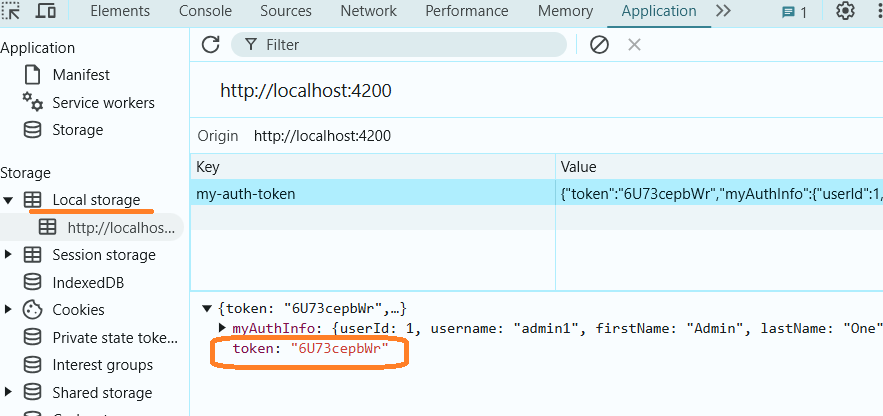
A screenshot of a computer

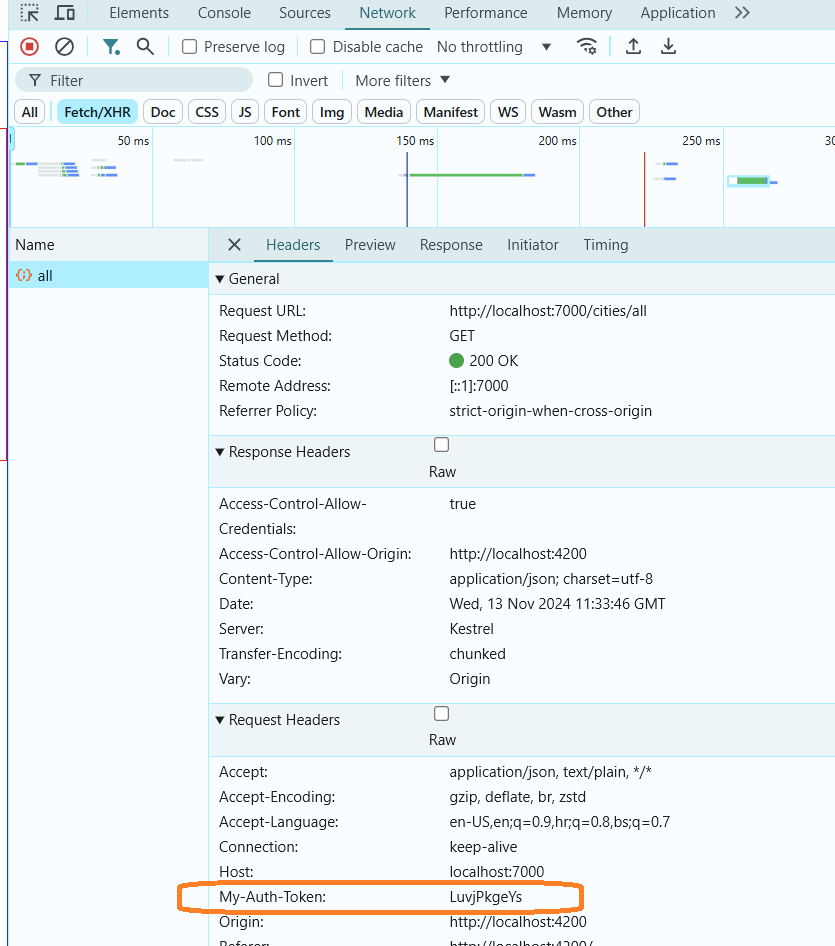
Description automatically generated

# Auth service i http interceptor

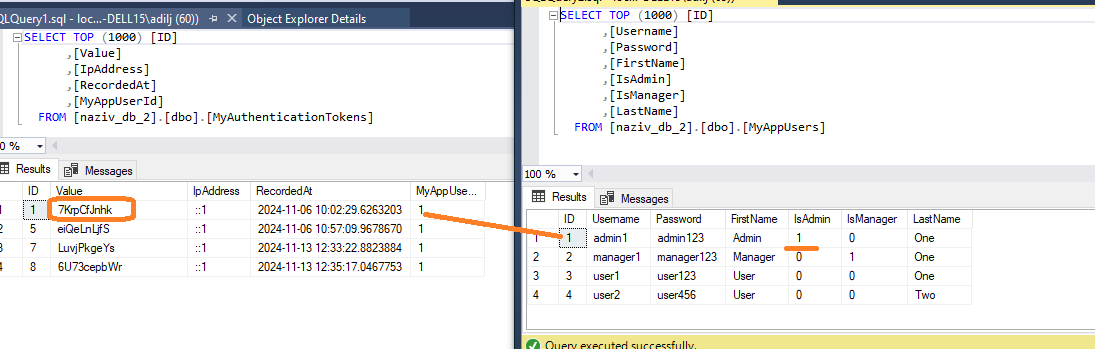
A screenshot of a computer program

Description automatically generated









# AuthGuard

A screenshot of a computer program

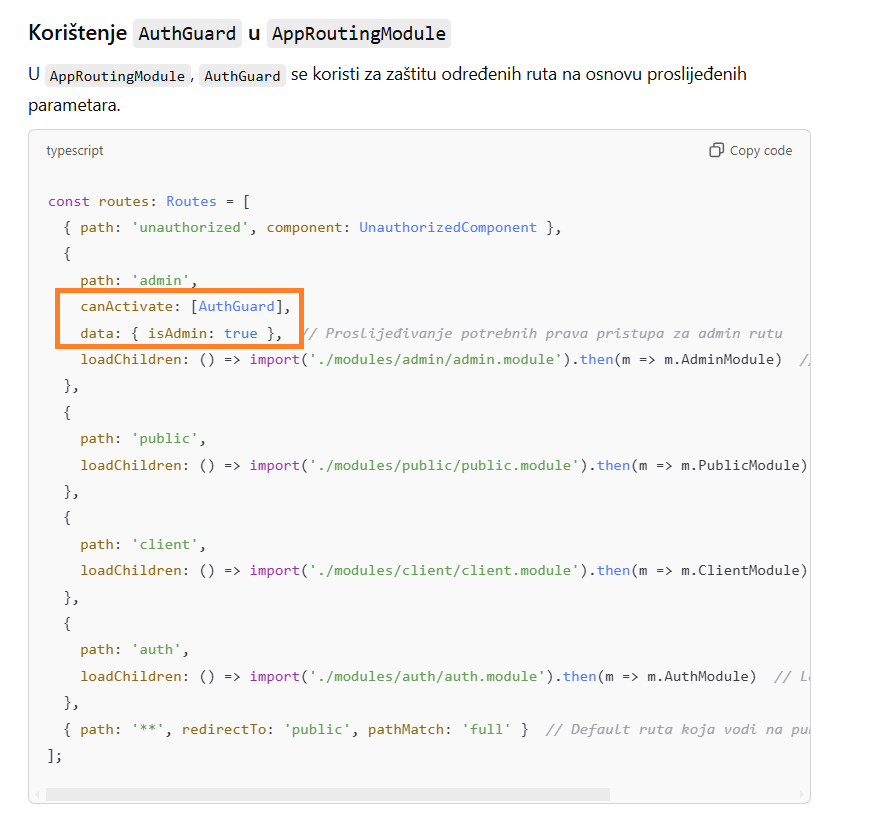
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a chat

Description automatically generated

# Angular Reactive Forms

admin\cities1\cities1-edit – **NgModel**

admin\cities2\cities2-edit – **Reactive Forms (podržava naprednije opcije)**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

# Sedmica 09

## CSS with superpowers - SCSS

<https://sass-lang.com/guide/>

Ispravljeno u projektu

src/styles.css -> src/styles.scss

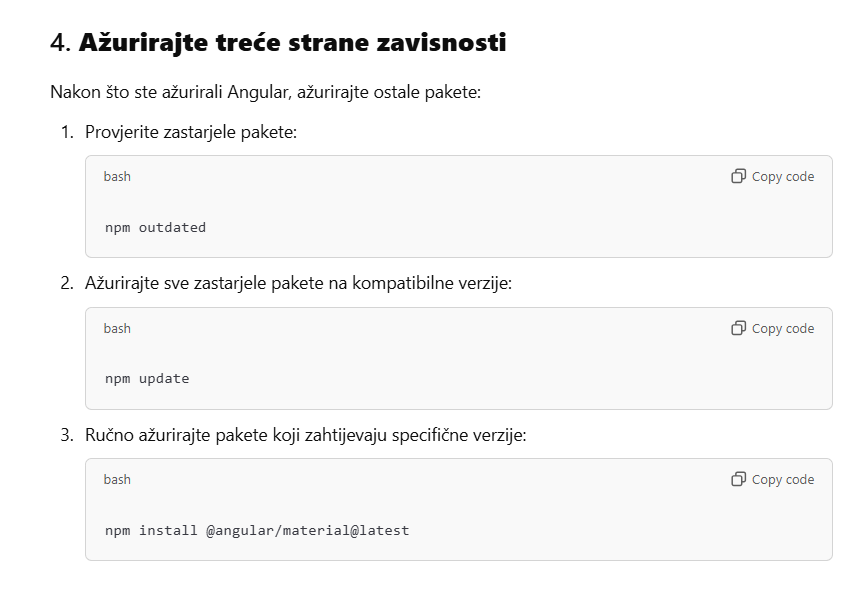
## Angular 19

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



|  |
| --- |
| ng version  npm install -g @angular/cli  ng version  ng update @angular/core @angular/cli --force  npm outdated  npm update  npm install |

U novijim verzijama Angular-a (od verzije **19**), standalone komponente postaju **defaultna konfiguracija** kada je opcija standalone izostavljena. Ovo je dio Angular-ove tranzicije prema modularnijem pristupu bez oslanjanja na NgModule.

**Kako ovo utiče na vaš projekt?**

* Ako niste eksplicitno postavili standalone: false, Angular automatski tretira komponentu kao **standalone**.
* Ovo znači da komponente koje želite koristiti unutar NgModule moraju imati jasno definisano standalone: false ili se moraju koristiti isključivo kroz imports.

|  |  |
| --- | --- |
|  |  |
|  |  |

## Angular material i Angular CDK

**Angular Material** je biblioteka komponenti baziranih na **Google Material Design** standardima. Cilj je omogućiti jednostavno implementiranje dizajnerskih principa i elemenata u Angular aplikacije.

**Ključne karakteristike:**

1. **Predefinisane UI komponente**: Komponente poput dugmadi, tabela, kartica, dijaloga, menija, formi, date-picker elemenata, i mnoge druge.
2. **Material Design pravila**: Sve komponente su usklađene s Material Design standardima za dosledan i moderan izgled.
3. **Responzivnost**: Komponente automatski reaguju na različite veličine ekrana (mobilni, tablet, desktop).
4. **Prilagodljivost**: Možete prilagoditi izgled koristeći **CSS** i **Sass** teme.

**Popularne komponente:**

* **Buttons** (mat-button, mat-raised-button)
* **Inputs** (mat-input)
* **Dialog** (mat-dialog)
* **Table** (mat-table)
* **Snackbar** (mat-snack-bar)
* **Stepper** (mat-stepper)

<https://material.angular.io/guide/getting-started>

**Angular CDK** je biblioteka osnovnih funkcionalnosti za izgradnju prilagođenih komponenti. Fokusira se na **funkcionalnost**, a ne na izgled, i pruža osnovne alate za razvoj kompleksnih UI elemenata.

**Ključne karakteristike:**

* **Prilagođeni UI elementi**: Možete kreirati svoje komponente koristeći ugrađene funkcionalnosti.
* **Pristupačnost (Accessibility)**: Omogućava jednostavno dodavanje pristupačnih elemenata (npr. a11y).
* **Drag & Drop**: Implementacija povuci-i-pusti funkcionalnosti.
* **Virtual Scrolling**: Prikazivanje velikog broja podataka uz optimizaciju performansi.
* **Overlay**: Omogućava prikazivanje elemenata preko drugih (npr. tooltip ili modalni prozor).
* **Responsive utilities**: Alati za responzivni dizajn.

npm install @angular/cdk

**Popularni moduli:**

* @angular/cdk/drag-drop: Povuci-i-pusti.
* @angular/cdk/overlay: Prikazivanje elemenata preko drugih.
* @angular/cdk/scrolling: Virtualni skrol.
* @angular/cdk/a11y: Alati za pristupačnost.

**Razlika između Angular Material i Angular CDK**

* **Angular Material** pruža gotove komponente sa unaprijed definisanim dizajnom (npr. dugmad, tabele, meniji).
* **Angular CDK** omogućava kreiranje prilagođenih komponenti koristeći funkcionalne alate, ali bez unaprijed definisanog dizajna.

<https://material.angular.io/cdk/categories>

**Zadatak 1:**

Kreirati helper dialog

* Simple za ispis obične poruke
* Confirm dialog, npr. delete

<https://material.angular.io/components/dialog/overview>

**Zadatak 2:**

Kreirati helper snackbar

* Simple za ispis obične poruke (npr. Uspješno snimljene izmjene)
* Sa dodatnom opcijom za click

<https://material.angular.io/components/snack-bar/overview>

**Zadatak 3:**

Prepraviti postojeću ngfor tabelu tako da koristi angular material tabelu

* Opcionalnost: koristiti Pagination

<https://material.angular.io/components/table/overview>

**Zadatak 4:**

Kreirati Angular material **Progress bar** tipa Indeterminate te ga povezati sa http interceptorom za fetch podataka sa API-a.

<https://material.angular.io/components/progress-bar/overview>

# Sedmica 10:

## Šta je i18n?

**i18n** (Internationalization) je proces prilagođavanja aplikacije tako da podržava više jezika i kultura bez potrebe za promjenom osnovnog koda. Naziv "i18n" dolazi od riječi "internationalization," gdje su "i" i "n" prva i zadnja slova, a između njih je 18 slova.

**Primjeri za i18n**:

* Prevod korisničkog interfejsa (UI).
* Formatiranje datuma, brojeva i valuta prema lokalnim standardima.
* Obrada desno-lijevog (RTL) jezika kao što su arapski i hebrejski.

**Instalacija i podešavanje i18n u Angular aplikaciji**

**1. Instalacija biblioteke ngx-translate**

Koristi se popularna biblioteka **ngx-translate**:

npm install @ngx-translate/core @ngx-translate/http-loader

**2. Kreiranje loadera za JSON fajlove**

Napravite datoteku src/app/translate-loader.ts:

import {TranslateLoader} from '@ngx-translate/core';

import {HttpClient} from '@angular/common/http';

import {Observable} from 'rxjs';

export class CustomTranslateLoader implements TranslateLoader {

constructor(private http: HttpClient) {}

getTranslation(lang: string): Observable<any> {

return this.http.get(`/i18n/${lang}.json`); // Folder za prevode

}

}

**3. Dodavanje u AppModule**

Importujte potrebne module u src/app/app.module.ts:

|  |
| --- |
| import {NgModule} from '@angular/core';  import {BrowserModule} from '@angular/platform-browser';  import {HttpClientModule, HttpClient} from '@angular/common/http';  import {TranslateLoader, TranslateModule} from '@ngx-translate/core';  import {CustomTranslateLoader} from './translate-loader';    @NgModule({  declarations: [  // Vaši komponenti  ],  imports: [  BrowserModule,  HttpClientModule,  TranslateModule.forRoot({  loader: {  provide: TranslateLoader,  useFactory: (http: HttpClient) => new CustomTranslateLoader(http),  deps: [HttpClient]  },  defaultLanguage: 'bs' // Podrazumijevani jezik  })  ],  providers: [],  bootstrap: [/\* Vaša početna komponenta \*/]  })  export class AppModule {} |

**Podešavanje JSON fajlova za bs i en**

1. Napravite i18n/bs.json:

2. Napravite i18n/en.json:

**Primjer za Login stranicu**

**HTML za Login komponentu:**

<div class="login-container">

<h2>{{ 'LOGIN.TITLE' | translate }}</h2>

<form (ngSubmit)="onLogin()">

<div class="form-group">

<label for="username">{{ 'LOGIN.USERNAME' | translate }}</label>

<input id="username" name="username" [(ngModel)]="username" required placeholder="{{ 'LOGIN.USERNAME' | translate }}">

</div>

<div class="form-group">

<label for="password">{{ 'LOGIN.PASSWORD' | translate }}</label>

<input id="password" name="password" type="password" [(ngModel)]="password" required placeholder="{{ 'LOGIN.PASSWORD' | translate }}">

</div>

<mat-slide-toggle>{{ 'LOGIN.SAVE\_LOGIN' | translate }}</mat-slide-toggle>

<button type="submit">{{ 'LOGIN.BUTTON' | translate }}</button>

</form>

</div>

**Dodavanje padajućeg menija za promjenu jezika**

**HTML za izbor jezika:**

|  |
| --- |
| <div class="language-selector">  <label for="language-select">{{ 'SELECT\_LANGUAGE' | translate }}</label>  <select id="language-select" (change)="changeLanguage($event.target.value)">  <option value="bs">Bosanski</option>  <option value="en">English</option>  </select>  </div> |

**Dodavanje prevoda u TypeScript fajl**

TypeScript za promjenu jezika:

import {Component} from '@angular/core';

import {TranslateService} from '@ngx-translate/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

constructor(private translate: TranslateService) {

this.translate.setDefaultLang('bs'); // Podrazumijevani jezik

}

changeLanguage(lang: string): void {

this.translate.use(lang);

}

}

**1. Pristup prevodu pomoću TranslateService**

U Angular aplikacijama koje koriste **ngx-translate**, prevodi se mogu koristiti ne samo u HTML-u, već i u TypeScript kodu. Evo primjera kako koristiti prevode u TypeScript fajlovima:

TranslateService omogućava pristup dinamičkim prevodima u komponentama, servisima i drugim dijelovima aplikacije.

import { Component } from '@angular/core';

import { TranslateService } from '@ngx-translate/core';

@Component({

selector: 'app-example',

templateUrl: './example.component.html',

styleUrls: ['./example.component.css']

})

export class ExampleComponent {

message: string = '';

constructor(private translate: TranslateService) {}

ngOnInit(): void {

// Preuzimanje prevoda za određeni ključ

this.translate.get('LOGIN.BUTTON').subscribe((translation: string) => {

this.message = translation;

console.log('Translated message:', this.message);

});

// Preuzimanje prevoda sa dinamičkim parametrima

const params = { username: 'Adil' };

this.translate.get('WELCOME\_MESSAGE', params).subscribe((translation: string) => {

console.log('Translated with params:', translation);

});

}

changeLanguage(lang: string): void {

this.translate.use(lang); // Promjena jezika aplikacije

}

}

**2. Dodavanje prevoda sa parametrima**

JSON fajl može sadržavati dinamičke vrijednosti koje se zamjenjuju u prevodima.

Primjer za **bs.json**:

{

"WELCOME\_MESSAGE": "Dobrodošao, {{username}}!",

"LOGIN": {

"BUTTON": "Prijavi se"

}

}

**Korištenje u servisima**

Ako trebate koristiti prevode u Angular servisima, TranslateService se može ubrizgati (inject) direktno:

import { Injectable } from '@angular/core';

import { TranslateService } from '@ngx-translate/core';

@Injectable({

providedIn: 'root'

})

export class NotificationService {

constructor(private translate: TranslateService) {}

showTranslatedMessage(): void {

this.translate.get('LOGIN.BUTTON').subscribe((translation: string) => {

console.log('Translated message:', translation);

// Možete prikazati poruku koristeći snackbar ili alert

alert(translation);

});

}

}

**Pojednostavljeni primjer sa instant**

Ako želite sinhrono preuzeti prevod (bez pretplate na Observable), koristite metodu instant:

const translatedMessage = this.translate.instant('LOGIN.BUTTON');

console.log(translatedMessage); // Direktno prikazuje prevedeni tekst

**Napomena**: instant radi samo za već učitane jezike. Ako prevod još nije učitan, rezultat može biti prazan string.

## MyErrorHandlingInterceptor

Backend app down

Servis vraća grešku. Ispisujemo snackbar poruku.

A screenshot of a computer

Description automatically generated

**Zašto implementirati HttpInterceptor za rukovanje greškama?**

**HttpInterceptor** omogućava centralizovano rukovanje HTTP zahtjevima i odgovorima, uključujući greške. Umjesto da ponavljamo kod za rukovanje greškama u svakom API pozivu, možemo:

1. **Centralizacija rukovanja greškama**: Sva logika za rukovanje greškama nalazi se na jednom mjestu, što smanjuje duplikaciju i olakšava održavanje.
2. **Korisničko iskustvo**: Prikaz korisniku odgovarajuće poruke o grešci u konzistentnom formatu.
3. **Lako proširenje**: Može se proširiti za dodatne scenarije, poput automatskog ponovnog pokušaja zahtjeva ili rukovanja specifičnim kodovima statusa (npr. 401 za autentifikaciju).
4. **Jednostavno skaliranje**: Ako aplikacija raste, nema potrebe za ručnim dodavanjem rukovanja greškama u svakom API pozivu.

**Koraci za implementaciju HttpInterceptor za rukovanje greškama**

**1. Kreiranje HttpInterceptor klase**

Napravite klasu kao u primjeru:

* Koristite intercept metodu za presretanje zahtjeva.
* Dodajte catchError operator iz RxJS za rukovanje greškama.

**2. Centralizovano rukovanje greškama**

Dodajte privatnu metodu handleError u interceptoru:

* Razlikujte greške klijenta (npr. greške u mreži) od server-side grešaka.
* Prikazujte poruke greške korisniku koristeći servis, poput MySnackbarHelperService.

**3. Uključivanje interceptora u aplikaciju**

Dodajte HttpInterceptor u dio providers u Angular modulu:

import { HTTP\_INTERCEPTORS } from '@angular/common/http';

@NgModule({

providers: [

{

provide: HTTP\_INTERCEPTORS,

useClass: MyErrorHandlingInterceptor,

multi: true, // Omogućava više interceptora

},

],

})

export class AppModule {}

**4. Testiranje**

* Testirajte greške klijenta (npr. onemogućavanje mreže) da biste vidjeli odgovarajuću poruku.
* Testirajte greške servera (npr. 500 Internal Server Error).

import {Injectable} from '@angular/core';  
import {HttpErrorResponse, HttpEvent, HttpHandler, HttpInterceptor, HttpRequest} from '@angular/common/http';  
import {catchError} from 'rxjs/operators';  
import {Observable, throwError} from 'rxjs';  
import {MySnackbarHelperService} from '../../modules/shared/snackbars/my-snackbar-helper.service';  
  
@Injectable()  
export class MyErrorHandlingInterceptor implements HttpInterceptor {  
 constructor(private snackBar: MySnackbarHelperService) {  
 }  
  
 intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  
 return next.handle(req).pipe(  
 catchError((error: HttpErrorResponse) => {  
 // Prikaži grešku korisniku  
 this.handleError(error);  
  
 // Propustiti grešku dalje ako je potrebno  
 return throwError(() => error);  
 })  
 );  
 }  
  
 private handleError(error: HttpErrorResponse): void {  
 if (error.error instanceof ErrorEvent) {  
 // Client-side greška  
 this.snackBar.showMessage(`Client error: ${error.error.message}`);  
 } else {  
 // Server-side greška  
 this.snackBar.showMessage(  
 `Server error: ${error.status} - ${error.message}`,  
 5000  
 );  
 }  
 }  
}



A screenshot of a computer error message

Description automatically generated

## Zašto koristiti prilagođene komponente (my-input-text i my-dropdown)?

****

****

1. **Smanjena Redundancija**: Korištenjem prilagođenih komponenti možete izbeći ponavljanje istih HTML struktura i logike za validaciju u različitim dijelovima aplikacije.
2. **Centralizovana Logika**: Validacija, prikaz grešaka i podešavanje podrazumijevanih vrijednosti upravlja se na jednom mjestu, čineći kod preglednijim i lakšim za održavanje.
3. **Standardizacija Dizajna**: Sve kontrole imaju konzistentan dizajn i ponašanje, što povećava kvalitet korisničkog iskustva.
4. **Ponovna Upotreba**: Komponente se mogu lako koristiti u različitim modulima i projektima.
5. **Jednostavnije Promjene**: Kada je potrebno napraviti promjene, dovoljno je izmijeniti kod unutar komponente, a te promjene će se odraziti na svim mjestima gdje je komponenta korištena.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

|  |
| --- |
|  |

A screenshot of a chat

Description automatically generated

|  |
| --- |
| **Šta je Angular dekorator?**  U Angularu, **dekoratori** su specijalne funkcije koje dodaju metapodatke klasama, metodama, svojstvima ili parametrima. Oni omogućavaju Angularu da prepozna određene klase, svojstva ili funkcije kao specifične delove aplikacije, kao što su komponente, servisi, direktive itd.   * **Primer dekoratora u Angularu**:   + **@Component**: Deklaracija komponente.   + **@Injectable**: Označava klasu koja se može koristiti za Dependency Injection.   + **@Input**: Omogućava prenošenje podataka iz parent komponente u child komponentu.   **Šta je @Input dekorator?**  Dekorator @Input označava svojstvo u child komponenti kao **ulazno svojstvo**. To znači da roditeljska komponenta može proslediti vrednosti tom svojstvu kada koristi child komponentu u svom HTML šablonu. |

A screenshot of a computer

Description automatically generated

## Zadaci – Nove custom komponente

1. Prilagoditi postojeće helper funkcije/komponente da koriste **i18n** modul za prikaz prevoda
2. Dodati prolagođenu Angular Reactive forms komponentu za **Date Picker**. Dodati validacije minimalnog i maksimalnog datuma.
3. Dodati prolagođenu Angular Reactive forms komponentu za **Time picker.** Dodati validacije minimalnog i maksimalnog vremena.
4. **Kreiranje prilagođene komponente za višestruki izbor (Multi-select)**

Implementirati Angular komponentu za višestruki izbor opcija pomoću Angular Material mat-select-a. Komponenta treba omogućiti prikaz izabranih opcija i uklanjanje svake opcije.

1. Implementirati **my-file-input**

|  |
| --- |
|  |

1. Implementirati **my-image-input** koja sadrži input kontrolu za odabir fajlova i **preview section**.

Preview section neka sadrži dugme za remove.

1. Premjestiti filter pretragu html tabele u zasebnu componentnu korištenjem **output** decoratora

--

## Output decorator



## Formati za slanje fajlova na server

A screenshot of a computer

Description automatically generated

