1) Hozza létre az alkalmazást és adja hozzá az alkalmazáshoz a választott UI keretrendszer

(pl.: Bootstrap) legfrissebb verzióját!

A parancssorba:

ng add @ng-bootstrap/ng-bootstrap

2) Készítsen Base néven service-t, amely az adatáramlásért lesz felelős, és a II/2 feladatban

elkészített API-hoz csatlakozik. Ha nem készítette el a backend programozás feladatot

használjon json szervert adatforrásként Kolcsonzok.json és a Kolcsonzesek.json fájlokat

felhasználva

**Ez kész, ezt csináltuk az előzőben. Ugyanazt a projektet használjuk. /Library/**

3) Hozzon létre egy komponenst Home néven, legyen ez a webalkalmazás nyitó oldala.

Állítsa be a routing-ot. Itt használja fel a Reszponzív viselkedésű weboldalt amelyet a

II/1. feladatrészben készített. (Minta1)

Terminálba:

ng g c home

az *app.component.html* tartalmát rakjuk át a [*home.component.html*](http://home.component.html) -be.

4) Hozzon létre két komponenst Kolcsonzok és Kolcsonzesek néven. A Home komponensen

található első könyv képre kattintva nyíljon meg a Kolcsonzok komponens. Állítsa be a

routing-ot.

Terminálba:

ng g c Kolcsonzok

ng g c Kolcsonzesek

Írjuk az első *href*-be a *Kolcsonzok* útvonalat. /*home.component.html*/

<h1>Library</h1>

<div class="navigation">

  <a href="kolcsonzok">

    <img src="assets/img/Book.png" alt="" class="book-btn-left">

  </a>

  <a href="#">

    <img src="assets/img/Book.png" alt="" class="book-btn-right">

  </a>

</div>

Menjünk az *app.routing.module.ts*-be.

Írjuk be:

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

import { RouterModule, Routes } from '@angular/router';

import { KolcsonzokComponent } from './kolcsonzok/kolcsonzok.component';

import { HomeComponent } from './home/home.component';

const routes: Routes = [

  { path: '', redirectTo: '/home', pathMatch: 'full' },

  { path: 'home', component: HomeComponent },

  { path: 'kolcsonzok', component: KolcsonzokComponent }

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

Az *app.component.html*.be:

<router-outlet></router-outlet>

És az *app.module.ts*-be:

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

és lent az *imports* tömbbe is írjuk be:

imports: [

    BrowserModule,

    AppRoutingModule,

    NgbModule,

    HttpClientModule

  ],

A *styles.css*-ből a *body* formázásán kívül mindent tegyünk át a [*home.component.css*-be](http://home.component.css-be).

Ez azért kell mert az előző feladat szerint még nem kellettek komponensek, de most már igen.

5) A Kolcsonzok komponensen az ügyfelek nevét és születési idejét listázza ki, ízlésesen. Az

adatokat a Base service-től kapja meg. (Minta2)

A *kolcsonzok.component.ts*-be:

importáljuk a BaseService-t és a Kolcsonzok osztályt:

import { BaseService } from '../base.service';

import { Kolcsonzok } from 'src/Kolcsonzok';

hozzunk létre egy tömböt a kölcsönzőknek:

  kolcsonzok: Kolcsonzok[] = [];

írjuk meg a konstruktort, ami lekéri a kölcsönzők adatait az api-tól.

constructor(private baseService: BaseService) {

    this.baseService.getKolcsonzok().subscribe(kolcsonzok => {

      this.kolcsonzok = kolcsonzok;

    })

  }

Az egész komponens most így néz ki:

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

import { BaseService } from '../base.service';

import { Kolcsonzok } from 'src/Kolcsonzok';

@Component({

  selector: 'app-kolcsonzok',

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

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

})

export class KolcsonzokComponent {

  kolcsonzok: Kolcsonzok[] = [];

  constructor(private baseService: BaseService) {

    this.baseService.getKolcsonzok().subscribe(kolcsonzok => {

      this.kolcsonzok = kolcsonzok;

    })

  }

}

A *kolcsonzok.component.html*-be:

<h1>Library</h1>

<div class="container">

    <div class="content">

        <div \*ngFor="let kolcsonz of kolcsonzok" class="row">

             <div class="col-sm-12 col-lg-4">

                <div class="name">{{kolcsonz.nev}}</div>

        </div>

        <div class="col-sm-6 col-lg-4 text-center">

            <div class="birthdate">({{ kolcsonz.szulIdo | date: 'yyyy-MM-dd' }})</div>

        </div>

        <div class="col-sm-6 col-lg-4 text-end">

            <div class="button">

                <p>Kölcsönzések</p>

            </div>

        </div>

    </div>

</div>

és a *kolcsonzok.component.css*-be:

h1 {

    font-family: "Roboto", sans-serif;

    /\* ez nem volt a feladatban de szerintem jobban néz ki ezzel a betűtípussal \*/

    font-weight: 300;

    font-style: normal;

    color: white;

    /\* fehér betűszín \*/

    width: 100%;

    /\* teljes szélesseg \*/

    margin-top: 2rem;

    /\* a margó beállítása \*/

    font-size: 21vh;

    /\* a 20rem helyett ezt adtam meg mert a 20rem -el ronda \*/

    background-color: rgb(161, 211, 54, 60%);

    /\*háttérszín\*/

    text-align: center;

    /\* a szöveg középre igazítása \*/

}

.container {

    background-color: rgba(236, 12, 12, 0.6);

    border-radius: 2rem;

}

.content {

    margin-top: 2rem;

    padding: 2rem;

    font-size: 3rem;

    color: white;

}

.name {

    font-weight: bold;

    font-size: 3rem;

}

.birthdate {

    font-size: 1.5rem;

    margin-top: 1.5rem;

}

.button p {

    display: inline-block;

    padding: 0rem 1rem;

    background-color: #D68910;

    color: #333;

    font-weight: bold;

    font-size: 1.5rem;

    border-radius: 0.5rem;

    cursor: pointer;

}

A formázás tetszőleges, a feladat csak azt írja elő hogy „ízlésesen” 😊

6) A név mellett kapjon helyett egy gomb, Kölcsönzések felirattal, melyre kattintva jelenjen

meg/tűnjön el az adott ügyfélhez tartozó Kolcsonzesek komponens.

A Kölcsönzések gombokhoz adunk egy click eseményt, ami meghív egy függvényt, ami listázza majd a kölcsönzéseket.

<p (click)="kolcsonzesek(kolcsonz.id)">Kölcsönzések</p>

Csinálunk a gomb alá egy div-et ami tartalmazza majd kölcsönzések input mezőit:

 <div class="row d-none" id="toggle{{kolcsonz.id}}">

            <app-kolcsonzesek [id]="kolcsonz.id"></app-kolcsonzesek>

        </div>

Az [id]=”kolcsonz.id” azt jelenti hogy átadja az id értéket a szülő komponensnek.

És a *kolcsonzok.component.ts*-ben elkészítjük a függvényt:

kolcsonzesek(id: number) {

    console.log(id);

    var div = document.querySelector("#toggle" + id);

    if (div) {

      div.classList.toggle("d-none");

    }

  }

A *kolcsonzesek.component.html-be*:

<div class="content">

    <div class="row header">

        <div class="col">

            <p>Író</p>

        </div>

        <div class="col">

            <p>Műfaj</p>

            </div>

        <div class="col">

            <p>Cím</p>

            </div>

        <div class="col">

            <p>Mentés</p>

            </div>

        <div class="col">

            <p>Törlés</p>

            </div>

    </div>

    <div class="row">

        <div class="col">

            <input type="text" name ="ujIro" [(ngModel)]="ujIro">

        </div>

        <div class="col">

            <input type="text" name="ujMufaj" [(ngModel)]="ujMufaj">

        </div>

        <div class="col">

            <input type="text" name="ujCim" [(ngModel)]="ujCim">

        </div>

        <div class="col button">

            <p (click)="mentes()">Mentés</p>

        </div>

        <div class="col">

        </div>

    </div>

    <div class="row" \*ngFor="let k of kolcsonzesek; let i = index" >

        <div class="col">

        <input type="text" name="iro" [(ngModel)]="k.iro">

    </div>

        <div class="col">

        <input type="text" name="mufaj" [(ngModel)]="k.mufaj">

    </div>

        <div class="col">

        <input type="text" name="cim" [(ngModel)]="k.cim">

    </div>

    <div class="col button">

        <p (click)="javitas(i)">Mentés</p>

        </div>

    <div class="col button">

        <p (click)="torles(i)">Törlés</p>

        </div>

    </div>

</div>

Létrehoztuk a minta szerinti input mezőket és a vezérlő gombokat. az [(ngModel)] direktívával adjuk át az értékeket a html és a *kolcsonzesek.component.ts* között.

A *kolcsonzesek.component.ts*-be:

export class KolcsonzesekComponent implements OnInit {

  @Input() id: number | undefined;

  kolcsonzesek: any;

  ujIro: string | undefined;

  ujMufaj: string | undefined;

  ujCim: string | undefined;

  iro: string | undefined;

  mufaj: string | undefined;

  cim: string | undefined;

  constructor(private baseService: BaseService) {

  }

ngOnInit() {

    if (this.id !== undefined) {

      this.baseService.getKolcsonzesek(this.id).subscribe(kolcsonzesek => {

        this.kolcsonzesek = kolcsonzesek;

      });

    }

  }

A változók létrehozása, az @input -> fogadja a küldött id-t.

Hozzuk létre a crud függvényeket:

mentes() {

    if (this.id !== undefined) {

      const kolcsonzes = {

        id: 0,

        kolcsonzoId: (this.id).toString() || "0",

        iro: this.ujIro || '',

        mufaj: this.ujMufaj || '',

        cim: this.ujCim || ''

      }

      this.baseService.createKolcsonzes(kolcsonzes).subscribe();

      this.ujratoltes(this.id);

    }

  }

  javitas(index: any) {

    const row = this.kolcsonzesek[index];

    const kId = row.id;

    this.baseService.updateKolcsonzes(kId, row).subscribe();

  }

  torles(index: any) {

    const id = this.kolcsonzesek[index].id;

    this.baseService.deleteKolcsonzes(id).subscribe();

    this.ujratoltes(this.id);

  }

  ujratoltes(id: any) {

    this.baseService.getKolcsonzesek(id).subscribe(kolcsonzesek => {

      this.kolcsonzesek = kolcsonzesek;

    });

    this.ujIro = '';

    this.ujMufaj = '';

    this.ujCim = '';

  }

és a formázás a *kolcsonzesek.component.css*-be:

.content {

    width: 100%;

    margin: 5rem auto;

    padding: 0.5rem;

    border: 4px solid white;

    border-radius: 1rem;

    background-color: rgb(161, 211, 54, 60%);

    font-size: 1rem;

}

.header {

    font-weight: bold;

    text-align: center;

}

.button p {

    display: inline-block;

    padding: 0rem 1rem;

    background-color: #D68910;

    color: #333;

    font-weight: bold;

    font-size: 1.5rem;

    border-radius: 0.5rem;

    cursor: pointer;

}

input {

    font-size: 1.5rem;

    border-radius: 0.5rem;

    border: none;

    margin-bottom: 0.5rem;

}

THE END