**NövényWebShop feladat megoldás**

1. **Alkalmazás előkészítése:**

* Mappába másolás **Git clone URL**
* Note package feltelepítése **npm i**
* Fájlok ellenőrzése **app.component** , **services** , **appmodule.ts (=> ellenőrizni hogy bevan-e a hivatkozva az AngularFireModule, AngularFireDatabaseModule, FormsModule)** , **enviroments.ts** , **part** mappák , **model** mappák , **minták** mappa megnézése.
* **Firebase project** létrehozása (Új web app project létrehozása, majd hozzá Build-elünk egy **REALTIME DATABASE** generálás teszt beállítással.)
* A fájlt importálja a backend adatbázisba (realtime database-be). A növények adatait **a novenyek.json** tartalmazza.

Ezt egy külön csomópontra importáljuk fel => /novenyek néven majd import

* Environments másolása „Project settings-ből” frissités után !
* **ng serve -o**

1. **Alkalmazás fejlesztése Adat MODELLEK**

* Adat model létrehozása novény.ts-ben. A json fájból vadászuk ki a szükséges infokat =>

export class Noveny {

key?: string

ar?: number

kepUrl?: string

leiras?: string

megnevezes?: string

}

* Adat model létrehozása a megrendeles.ts-ben is! A feladatból vadászuk ki az adat model alapját.

export class Megrendeles {

nev?:string

cim?:string

datum?:string

statusz?:string

tetelek?:any[] <=ÜRES TÖMB!

}

1. **Alkalmazás fejlesztése Componensek generálása**

* Új componesek generálás!( A Part mappán belül generálunk )

ng g c part/novenyek

ng g c part/rendeles

1. **Alkalmazás fejlesztése App-routing beállitása**

* Beállítjuk az adott componesek útvonalait

const routes: Routes = [

{path:"novenyek",component:NovenyekComponent},

{path:"rendeles", component:RendelesComponent},

{path:"", component:RolunkComponent}, (EZ JELEN ESETBEN A HOME)

{path:"\*\*", component:RolunkComponent},

];

1. **Alkalmazás fejlesztése nav.component / routing folytatása**

* nav.component.html belül beállitjuk az routerLinkeket, hogy a megfelelő oldalra navigáljon tovább kattintásnál.

<li class="nav-item">

<a class="nav-link"

routerLink**="/novenyek"**

routerLinkActive="active"

[routerLinkActiveOptions]="{exact:true}"

>Növények</a>

</li>

1. **Alkalmazás fejlesztése BASE.SERVICE.TS**

* Referenciát kell csinálni a növényeknek (szerkeszteni kívánt részlegnek)

refNovenyek: AngularFireList<Noveny> (Ne felejtsd el IMPORTÁLNI az AngularFirelist-et.)

* adatbázis szerkesztése => constructor-on belül.

constructor(private db:AngularFireDatabase) {

this.refNovenyek=this.db.list**("/novenyek**") => (Firebase csomópont)

}

* Megírjuk az egyetlen eljárást a base.service-belül a constructor alatt egyből.

getPlants(){

return this.refNovenyek

}

1. **Alkalmazás fejlesztése Novenyek.Component**

* növények component-ts-ben meg pipe-oljuk a növényeket, majd megjelenitjuk őket.
* előszőr felveszünk egy változót pl: növények:any ;
* majd constructoron belül behivatkozzuk a BaseService-t és megirjuk a halál sorokat a pipe-hoz xd

**export class NovenyekComponent {**

**novenyek:any;**

**constructor(private base:BaseService){**

**this.base.getPlants().snapshotChanges()**

**.pipe(**

**map(**

**(ch)=>ch.map(**

**(c)=>({key:c.payload.key, ...c.payload.val()})**

**)))**

**.subscribe((res)=>this.novenyek=res)**

**}**

**}**

* Leellenőrizzük, hogy megérkeznek-e az adatok => átengedjük egy json pipe-on a **novenyek.component.HTML-belül** utána megírjuk a cardot ugyan ott.

{{novenyek|json}}

* Bootstrap-en keresünk egy CARD-ot amiben megjelenítjük a adatokat.

csinálunk egy „ROW”-divet maajd behelyezzük a card-ot.

* A col-os div-et akarjuk majd ismételgetni, szóval a class elé írunk egy \*ngFor függvényt.
* ÁTITJUK A CARDON BELÜL A KIVÚLRŐL ÉRKEZŐ ADATOK BEHIVATKOZÁSAIT

<div class="row">

<div **\*ngFor="let plant of novenyek"** class="col-12 col-md-6 col-lg-4 col-xl-3">

<div class="card h-100">

<**img [src]="plant.kepUrl"** class="card-img-top" **[alt]="plant.megnevezes**">

<div class="card-body">

<h5 class="card-title">{{plant.megnevezes}}</h5>

<p class="card-text">{{plant.leiras}}</p>

</div>

<div class="card-footer">

<small class="text-muted">Last updated 3 mins ago</small>

</div>

</div>

</div>

</div>

* Majd keresünk bootstrapen egy inputot és gombot a card aljára, illetve egy sima <p>-belül berakjuk az árát is a terméknek.
* inputnál megadjuk a type-ot mint number , a value=1 , min=0 és adunk egy egy nevet #db alatt

<div class="input-group mb-3">

<span class="input-group-text" id="basic-addon1">Db:</span>

<input **#db type="number" value="1" min="0"** class="form-control">

<button **(click)="hozzaad(plant, db.value)**" type="button" class="btn btn-secondary">Kosárba</button>

</div>

* a gombnak adunk egy click eseményt hozzaad() függvénnyel => majd azt megirjuk a novenyek.ts-ben a pipe alatt.

hozzaad(plant:any, db:any){}

1. **Alkalmazás fejlesztése Kosar.service**

* Felveszünk egy változót pl: kosar:any
* constructoron belül megadjuk, hogy az kezdésnél egy üres tömb legyen.
* hozzáadunk egy eljárást amivel használni tudjuk majd pl: **addTetel() ezt megkell hívnunk majd a novenyek.comp.ts-ben a hozzaad metóduson belül. => előszőr meghivjuk a kosár service fent a constructornál, hogy lássa a metóduson belül.**

**(private kosar:KosarService)**

**Majd megirjuk a metódust**

**hozzaad(plant:any, db:any){**

**this.kosar.addTetel(plant,db)**

**}**

* Index létrehozása az **addTetel()** metóduson belül, annak érdekébe, hogy korrigáljuk a kosárba kerülő termékeket. (ha 2x kattint valamire, akkor egybe rakja be nem kettő különbbe).
* Ha a kereső index nem találja (nem volt benne előtte) akkor belepusolja, ha ottvan akkor meg csak kezeli, növeli.

addTetel(plant:any, db:any){

let i = this.kosar.findIndex(

(e:any)=>e.key==plant.key

)

if (i==-1) this.kosar.push({key:plant.key, ar:plant.ar, db:db})

this.kosar[i].db=db

console.log(this.kosar)

}

1. **Alkalmazás fejlesztése RENDELÉS COMPONENT**

* **Rendeles.comp.HTML** belül keresünk egy **card-ot** ahol összegezük a rendelést.
* Cardon belül keresünk egy floating-labels inputot az adatok bekérése, illetve egy gombot az aljára a rendelés véglegesítéshez.
* Floating-label belül az input mezőben átírjuk a type-ot ,adunk egy ID-t, és egy name-t
* Floating-label belül a label-nél átirjuk for=”name”-t

<div class="card">

<div class="card-body">

<h5 class="card-title">Megrendelés</h5>

<div class="form-floating mb-3">

<input **type="text"** class="form-control**" name="name" id="name**">

<label **for="name**">Név</label>

</div>

<button type="button" class="btn btn-secondary">Megrendelés véglegesítése</button>

</div>

</div>

* **Rendeles.component.TS**

Csinálunk 2 új változót =>

export class RendelesComponent {

nav:any;

cim:any;

}

* ezeket betudjuk kötni az előzetesen megirt sorokba a rendeles.comp.html-nél
* [(ngModel)] segiségével átkérjük az adatokat, plusz átirjuk a Id, name-t, for-t,
* Keresünk egy Alert-inputot, hogy listázzuk a vásárlandó termékeket.
* Kosár.service.ts-be átmegyünk megirjuk hozzá a Subject-et pluszba illetve a deleteTetel() , deleteKosar() eljárásokat

🡺

export class KosarService {

kosar:any

**kosarSub=new BehaviorSubject<any>(null)**

constructor() {

this.kosar=[]

}

**getKosar(){**

**return this.kosarSub**

**}**

addTetel(plant:any, db:any){

let i = this.kosar.findIndex(

(e:any)=>e.key==plant.key

)

if (i==-1) this.kosar.push({key:plant.key, ar:plant.ar, db:db})

else this.kosar[i].db=db

**this.kosarSub.next(this.kosar)**

console.log(this.kosar)

}

**deleteTetel(key:any){**

**this.kosar=this.kosar.filter(**

**(elem:any)=>elem.key!=key**

**)**

**this.kosarSub.next(this.kosar)**

**}**

**deleteKosar(){**

**this.kosar=[]**

}

* Vissza a rendeles.ts-be lekérjük a kosarat
* Csinálunk 2 új változót kosar és novenyek néven
* Megirjuk az eljárást construtoron belül illetve a HALÁL soros Pipe-ot is behivatkozzuk. Ne felesjd el importálni fent a Base-servicet privatba fent.
* Névkeresés metódót megírjuk

🡺

export class RendelesComponent {

nev:any;

cim:any;

**kosar:any;**

**novenyek:any;**

constructor(private kosarService:KosarService, **private base:BaseService**){

**this.kosarService.getKosar().subscribe(**

**(res)=>this.kosar=res**

**)**

**this.base.getPlants().snapshotChanges()**

**.pipe(**

**map(**

**(ch)=>ch.map(**

**(c)=>({key:c.payload.key, ...c.payload.val()})**

**)))**

**.subscribe((res)=>this.novenyek=res)**

**}**

**nevKereses(key:any){**

**let i =this.novenyek.findIndex(**

**(elem:any)=> elem.key==key**

**)**

**return this.novenyek[i].megnevezes**

}

}

* Vissza a rendeles.html-re megcsinálni a Alert-ot

\*ngFor írás majd egy külön col-divben {{nevKereses(tetel.key)}}

utána még hozzáadjuk a tetelek db-számát meg a tételek ár és a db számának a szorzatát a szokásos modón

<div class="alert alert-secondary" role="alert">

<div **\*ngFor="let tetel of kosar"** class="row">

<div class="col**">{{nevKereses(tetel.key)}}**</div>

<div class="col**">{{tetel.db}}db**</div>

<div class="col**">{{tetel.ar\*tetel.db}}Ft**</div>

</div>

</div>

* Törlős gomb megadása
* rendeles.html- belül a tétel.ar\*tetel.db alatt elhelyezünk egy gombot
* gombhoz hozzáadjuk tetelTorol() eljárást majd megírjuk a rendelelés ts-ben

<div class="col">

**<button (click)="tetelTorol(tetel.key)" type="button" class="btn btn-secondary">Tétel törlése</button>**

</div>

* megírjuk a rendelelés ts-ben

**tetelTorol(key:any){**

**this.kosarService.deleteTetel(key)**

**}**

* Végül hozzáadunk rendelés.html-ben a Alart divnél még egy \*ngIf -et a felő div-hez (OPCIONÁLIS nem kötelező)

**<div \*ngIf="kosar && novenyek" class="alert alert-secondary" role="alert">**

* végül az alsó button „Rendelés véglegesítése” megírunk egy eljárást hogy elküldje backend felé majd törölje is.
* Előszőr Átmesáljuk a megrendeles MODEL mappából az adatmodelt a rendeles.com.ts-be

**megrendelesLead(){**

**let body = {**

**nev:this.nev,**

**cim:this.cim,**

**datum:new Date(),**

**statusz:"Leadva",**

**tetelek:this.kosar**

**}**

**this.base.megrendel(body).then(**

**()=>{}**

**)**

**}**

* base.service-ben megirjuk a megrendel(body) eljárást
* ahoz előszőr egy új referenciát kell megadni fent

**refMegrendeles: AngularFireList<any>**

* Új firebase-es csomópont készítése utána

**this.refMegrendeles=this.db.list("/megrendelesek")**

* Új eljárás zárásul az aljára megrendel()

export class BaseService {

refNovenyek: AngularFireList<Noveny>

**refMegrendeles: AngularFireList<any>**

constructor(private db:AngularFireDatabase) {

this.refNovenyek=this.db.list("/novenyek")

**this.refMegrendeles=this.db.list("/megrendelesek")**

}

getPlants(){

return this.refNovenyek

}

**megrendel(body:any){**

**return this.refMegrendeles.push(body)**

**}**

* **Ellenőrizni firebase csomópontot, hogy működik-e.**
* **rendeles.ts-ben irunk egy új constructor routert private router:Router**
* **majd megtudkuk csinálni kosárservice-ben deletekosar-al kitudjuk törölni rögzités után, a routerrel meg átnavigál a növényekhez.**

**constructor(private router:Router,** private kosarService:KosarService, private base:BaseService){

megrendelesLead(){

let body={

nev:this.nev,

cim:this.cim,

datum:new Date(),

statusz:"Leadva",

tetelek:this.kosar

}

**this.base.megrendel(body).then(**

**()=>{**

**this.kosarService.deleteKosar()**

**this.router.navigate(['/novenyek'])**

**}**

**)**

}

1. **Alkalmazás fejlesztése PIPE filter a növény komponesben**
   * Generálunk egy új pipe-ot

**ng g pipe filter**

* + Keresünk egy imput-ot megírjuk a norveny.html-nél fent minden felett.

**<div class="row my-2">**

**<input [(ngModel)]="keresendo" type="text" class="form-control">**

**</div>**

* + ne felejtsd el felvenni a keresendő változót a noveny-ts-ben

**keresendo:any**

* + írunk egy filtert a növényes divre

<div class="row">

<div \*ngFor="let plant of novenyek |**filter:keresendo"** class="col-12 col-md-6 col-lg-4 col-xl-3">

* + Filter Pipe-ts-re átmegyünk és átirjuk

**export class FilterPipe implements PipeTransform {**

**transform(tomb:any[], keresendo:any): any {**

**if (!tomb) return null;**

**if (!keresendo) return tomb;**

**tomb=tomb.filter(**

**(elem)=>elem.megnevezes.toLowerCase().includes(keresendo.toLowerCase())**

**)**

**return tomb**

**}**

**}**

**☹☹☹☹☹☹☹☹☹☹☹**