Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Sánta Tamás

2023

Szegedi Tudományegyetem

Informatikai Intézet

Angular webshop alkalmazás

Szakdolgozat

Készítette:
Sánta Tamás
programtervező
informatikus szakos
hallgató

Témavezető:
Dr. Bilicki Vilmos
adjunktus,
konzulens

Szeged

2023

Feladatkiírás

A szakdolgozat keretein belül egy webshop alkalmazást készítetek, amiben a felhasználók tudnak böngészni a férfi ruházati termékek között.

A webshop lehetőséget biztosít a vásárlók számára, hogy böngésszenek az adott ruhák között, de ezt csak bejelentkezés, illetve regisztráció után tehetik meg. A felhasználói adatok firebase felhő alapú szolgáltatásban kerülnek elmentésre.

A felhasználók tudnak a weboldalon ruhákat nézegetni, amik közül a számukra tetszőeket kosárba tehetik, és később meg is vásárolhatják. Fizetés elött meg kell adnia az adatait, hogy milyen névre szól a rendelése, illetve, hogy milyen címre szeretné, hogy azt majd kiszállítsák, emellett meg kell még adnia egy e-mail címet és egy telefonszámot, amit értesíteni lehet majd a vásárlót. Fizetésnél választhat különböző fizetési módokat, hogy kártyával fizetne vagy utánvéttellel, aminek lesz valami minimális extra költsége is.

Illetve tudnak még szűrni is, hogy ár szerint milyen sorrendben jelenjenek meg a termékek számukra, ezáltal könnyedén megtalálhatóak lesznek a legolcsóbb és legdrágbb darabok is.

A különböző ruhadarabokról láthatnak pár információt és egy képet, hogy hogyan is néz ki, emellett tudnak a termékről kommenteket írni, hogy kinek milyen véleménye van róla, illetve értékelhetik az adott terméket 1-5ig. És ha egy oda nem illő komment jelenik meg, akkor azt egy admin törölheti. De a saját maga kommentjét mindenki kedve szerint tudja törölni.

Tartalmi összefoglaló

- ***A téma megnevezése:***

Egy férfi ruházati webshop

- ***A megadott feladat megfogalmazása:***

A feladat egy webshop alkalmazás elkészítése amiben könnyeden lehet böngészni, szűrni az adott termékekre, véleményt megfogalmazni az adott termékekről, illetve a termékek megvásárlása.

- ***Alkalmazott eszközök, módszerek:***

Az alkalmazás a Visual Studio Code nevű fejlesztői környezetben, TypeScript nyelven készült. A frontend HTML illetve SCSS segítségével készült. A program az Angular keretrendszerben lett elkészítve.

- ***Kulcsszavak:***

Angular, webshop

Tartalomjegyzék:

Használt technológiák, eszközök bemutatása:

Programozási nyelv:

Egy mai számítógép az emberek számára követhetetlen számolási sebességgel bír, ami miatt jó ötletnek tűnhet, ha az egyes emberi feladatokat rábíznánk a számítógépekre. Ezek „agya” egyelőre még nem ért emberi nyelven, de akkor mégis hogyan tudnánk megmondani egy számítógépnek, hogy mit csináljon?

Ennek a problémának a megoldására születtek meg a programozási nyelvek. A programozási nyelvek mindegyike, csak úgy, mint egy emberi nyelv, rendelkezik saját szabályrendszerrel. A szabályrendszer lehetőséget ad egységes módon megfogalmazni egy utasítást, vagy annak sorozatát, egy programot. Egy program a programozási nyelv szabályait követve az más, a szabályokat szintén ismerő ember számára is olvasható és érthető.

TypeScript:

A TypeScript egy ingyenes és nyílt forráskódú, magas szintű programozási nyelv, amelyet a Microsoft fejlesztett ki, és amely statikus gépelést és opcionális típus jegyzéket ad a JavaScript-hez. Nagy alkalmazások fejlesztésére és a JavaScriptre való átültetésre tervezték. Az összes JavaScript program érvényes TypeScript, de biztonsági okokból előfordulhat, hogy nem hajtják végre a típus ellenőrzést.

Egyéb nyelvek:

A HTML (Hipertext Markup Language) egy leíró nyelv, ami előre definiált tag-ek egymásba ágyazásával teszi lehetővé a weboldal vázának felépítését. Frontend-nek a legtöbb webfejlesztési keretrendszer.

A JavaScript egy interpretált programozási nyelv. A böngészővel és a weboldallal való kommunikációs képessége miatt webprogramozásban ez a leggyakrabban használt nyelv. Használatával a HTML DOM-ja is dinamikusan megváltoztatható.

A SCSS (Syntactically Cascading Style Sheet) egy stílusleíró nyelv, ami előre definiált kulcs-érték párok megadásával tudja változtatni a strukturált dokumentumok, például a HTML megjelenését. További funkciókat biztosít, amelyek a normál CSS-ben nem elérhetők. Az SCSS fájlkiterjesztése .scss.

Keretrendszer:

Egy nagy projekt megírása sok profi programozónak is rengeteg időbe telik, azonban léteznek módszerek, amikkel a fejlesztési idő lerövidíthető. Ebben tudnak segítséget nyújtani a keretrendszerek. Ezek általános eszköz tarát biztosítanak, amik segítségével a fejlesztőnek nincs szüksége mindent az elejéről felépítenie. Használatukkal a már működő rendszer ki van terjesztve a munkánkkal, igy fókuszálhatunk a lényegi részre. Ezek elsajátítása nagyban javítja, de rossz használatuk csak ront egy program hatékonyságán.

Angular:

Az Angular egy nyílt forráskódú, TypeScript alapú keretrendszer, amit a Google fejlesztett ki és jelenleg is a Google Team vezeti. Elsősorban frontend oldali fejlesztéshez, összetett webes alkalmazásokhoz találták ki. 2010-ben jelent meg először, ekkor még Angular JS néven, de ma már a legfrissebb verziója az Angular 16.1. De 2023.11.06-án már meg fog jelenni a 17.0-ás verzió is.

Fejlesztői környezetek:

A fejlesztői környezetek olyan programozók számára készült szoftverek, amik fordítót, futtatókörnyezetet és szövegszerkesztőt is tartalmaznak, hogy a programozó számára minden adott legyen egy helyen. Ezek nagyon sokat tudnak segíteni a fejlesztésben. Vannak olyan fejlesztői környezetek is, amik ismerik a különböző programozási nyelvek szabályait. Például a program írása közben az addig begépelt szövegből képesek kitalálni esetleg mit szeretnénk írni. Illetve a hibákat is jelzik a számunkra. Képesek befejezni a nyelv kulcs szavait. Ezeket a fejlesztői környezeteket integrált fejlesztői környezetnek nevezzük, manapság már szinte minden programozási nyelvhez léteznek ilyenek.

Visual Studio Code:

Nem szabad összetéveszteni a Visual Studio-val, ez két különböző fejlesztői környezet. A Visual Studio Code, más nevén VS Code egy forráskód-szerkesztő, amelyet a Microsoft készített 2015-ben Windowsra, Linuxra és macOS-re. A funkciók közé tartozik a hibakeresés támogatása, a szintaxis kiemelése, az intelligens kódkiegészítés, a kódrészletek, a kód újrafeldolgozása és a beágyazott Git. Érdekesség,

hogy a Stack Overflow 2022 fejlesztői felmérésében a VS Code volt a legnépszerűbb fejlesztői környezet a válaszadók körében, kb a 75%-uk számolt be arról, hogy ezt használja.

Adatbázis:

A számítástechnikában az adatbázis elektronikusan tárolt és elérhető adatok szervezett gyűjteménye. A kis adatbázisok fájlrendszerekben tárolódnak, a nagyok pedig clusterekben vagy felhőalapú tárolókban vannak eltárolva. A szakdolgozat elkészítése közben én is felhőalapú adatbázist a használtam, ez a Firebase.

Firebase:

A Firebase a Google által biztosított háttéralapú számítási felhőszolgáltatás, amit 2011-ben alapítottak. Amely segít az alkalmazásfejlesztőknek alkalmazásaik létrehozásában, üzembe helyezésében és méretezésében. Egy távoli felhőtechnológiára támaszkodik, az üzemeltetési kérdésekkel így a fejlesztőnek nem kell foglalkozni, a felhőszolgáltató által biztosított kész megoldásokat tudnak használni.

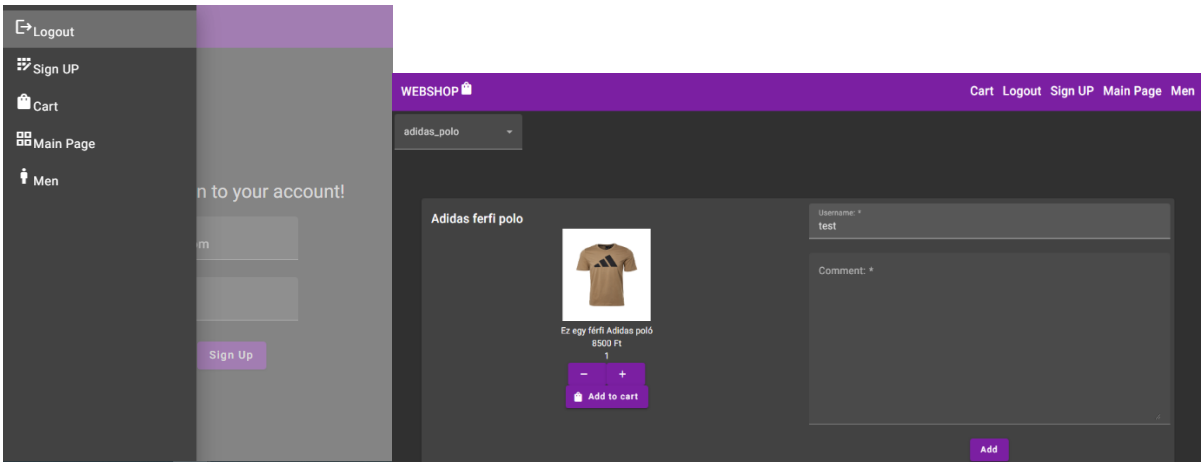
Eddig megvalósított funkciók:

Az eddig elkészített projektben egy bejelentkezési oldal vár minket amikor megnyitjuk a weboldalt, ahol betudunk jelentkezni a már létező profilunkba. A profilunk adatai a már korábban bemutatott Firebase rendszer vannak elmentve. Ha még nem rendelkezünk saját felhasználóval akkor van lehetőség arra is, hogy regisztráljunk egyet, ezt követően mentődnek el az adatok a Firebase-be, ahonnan bejelentkezésnél le ellenőrzi a rendszer, hogy az egy valid, létező email cím – jelszó páros, és benne van-e a rendszerben, és ha igen akkor belépteti őt. Belépés után érhető látható lesz a kosár fül, illetve a kijelentkezés gomb. Emellett csak a bejelentkezett felhasználók mehetnek a főoldalra, illetve ők böngészhetik a webshopban található ruhákat, ezek AuthGuard segítségével lett megvalósítva.

```
const routes: Routes = [
  { path: 'main',
  loadChildren: () => import('./pages/main/main.module').then(m => m.MainModule),
  canActivate: [authGuard]
  },

  { path: 'men',
  loadChildren: () => import('./pages/men/men.module').then(m => m.MenModule),
  canActivate: [authGuard]
```

A ruhákat egy legördülő listában lehet válogatni, és amelyik ruhadarabot válasszuk ki arról megjelenik egy kép a termékről, egy rövid leírás róla, illetve az ára. Be tudjuk állítani hány darabot szeretnénk venni belőle, és kosárba tehetjük. Emellett tudunk kommentet írni a termékről, amik szinten elmentődnek a Firebase-ben, és minden felhasználó a saját kommentjét tudja törölni a rendszerből. Az alkalmazás Firebase Hosting URL-el lett deploy-olva, és minden egyes végpont megfelelő módon betöltődik. Az alkalmazás reszponzív, mind teljes ablaknézetben és mobile-first nézetben is teljesen jól működik. Mobile-first nézetben nem a menü listában lesznek láthatóak az oldalak, hanem oldalt, egy mobil nézetés kinyitható menüből érhetőek el.



A megjelenést Angular Material-okkal oldottam meg. Az Angular Material egy felhasználói felület komponenskönyvtár, amit a fejlesztők használnak, hogy elegáns és konzisztens felületeket fejleszthessenek. Az Angular Material egy újra felhasználható felületösszetevőket kínál, mint a FormField, Card, Button, Icon, sok egyéb, és ezeket importálni kell module.ts-ben. És ha ez megtörtént akkor a megfelelő component html-ben használhatjuk is őket.

```
imports: [
  CommonModule,
  MenRoutingModule,
  FormsModule,
  ReactiveFormsModule,
  MatSelectModule,
  MatOptionModule,
  MatFormFieldModule,
  MatCardModule,
  MatInputModule,
  MatButtonModule,
  FlexLayoutModule,
  MatIconModule
]
```

```
<div class="div-card" *ngIf="imageInput">
    <mat-card fxLayout fxLayoutGap="20px">
        <span fxFlex="1 1 50%">
            <mat-card-header>
                <mat-card-title>{{imageInput.name}}</mat-card-title>
            </mat-card-header>
            <mat-card-content>
                <img class="menImage" [src]="'assets/' + imageInput.id + '.jpg'" />
                <div class="break"></div>
                <div>{{imageInput.description}}</div>
                <div class="break"></div>
                <div>{{imageInput.price}}</div>
                <div class="break"></div>
                <div>{{imageInput.count}}</div>
                <button mat-raised-button color="primary" type="button" (click)="decrease()"><mat-icon>remove</mat-icon></button>
                <button mat-raised-button color="primary" type="button" (click)="increase()"><mat-icon>add</mat-icon></button>
                <div class="break"></div>
                <button mat-raised-button color="primary" type="button" class="cart" >Add to cart<mat-icon>shopping_bag</mat-icon></button>
            </mat-card-content>
```

Az adatbevitel bejelentkezésnél/regisztrációnál Angular Form-ok segítségével lett megvalósítva, ahhoz, hogy ezt használni lehessen importálni kell a module.ts-ben.

```
<div class="main_component">
    <h1>Here you can login to your account!</h1>
    <mat-form-field><mat-label for="email">Email: </mat-label><input matInput type="text" [formControl]="email"
    <mat-form-field><mat-label for="password">Password: </mat-label><input matInput type="password" [formControl
    <span>
        <span><button mat-raised-button color="primary" (click)="login()">Login</button></span>
        <span><button mat-raised-button color="primary" [routerLink]="'/signup'">Sign Up</button></span>
    </span>
</div>
```

```
<form [formGroup]="signUpForm" (ngSubmit)="onSubmit()">
    <mat-form-field>
        <mat-label for="email">Email: </mat-label>
        <input matInput type="text" formControlName ="email" />
    </mat-form-field>
    <mat-form-field>
        <mat-label for="password">Password: </mat-label>
        <input matInput type="password" formControlName ="password" />
    </mat-form-field>
    <mat-form-field>
        <mat-label for="rePassword">Confirm password: </mat-label>
        <input matInput type="password" formControlName ="rePassword" />
    </mat-form-field>
    <div formGroupName="name" class="subForm">
        <mat-form-field>
            <mat-label for="firstname">First name: </mat-label>
            <input matInput type="text" formControlName ="firstname" />
        </mat-form-field>
        <mat-form-field>
            <mat-label for="lastname">Last name: </mat-label>
            <input matInput type="text" formControlName ="lastname" />
        </mat-form-field>
    </div>
```

A projektben meg lett valósítva egy saját Pipe osztály megírása is, ami a dátum-ok kiírtatásáért felelnek, ezeket a kommenteknél használom. A Pipe-ok egyszerű függvények, amelyekben sablonkifejezésekben használhatók bemeneti értékek elfogadására és az átalakított érték visszaadására. A Pipe-ok azért is hasznosak egy alkalmazás során, mert elég őket egyszer deklarálni, de használni meg többször is lehet őket.

```
export class DateFormatPipe implements PipeTransform {

  transform(value: number, ...args: unknown[]): string {
    let tzoffset = (new Date(value)).getTimezoneOffset() * 60000;
    let minOffSet = new Date(value).getTime() - tzoffset;
    let localISOTime = (new Date(minOffSet)).toISOString().replace('Z', '').replace('T', ' ');
    return localISOTime
  }

}
```

A projekt során alkalmaztam Angular Service-ket is, amelyek olyan objektumok, amik az alkalmas élettartama alatt csak egyszer példányosodnak, és olyan függvényeket tartalmaznak, amelyeket az alkalmazás teljes élettartama alatt megőrzik az adatokat, szóval azok így folyamatosan elérhetőek. Service-t használtam az Auth Guard-hoz, a különböző felhasználói fiók műveletekhez – bejelentkezés, regisztráció, kilepés.

Angular promise és observable használata. Az observable egy lustán kiértékelt számítás, amely szinkron vagy aszinkron módon adja vissza az értéket a meghívástól kezdve. Azaz az observable egy olyan függvény, amely a közönséges adatfolyamot megfigyelhető adatfolyammá alakítja. Az observable aszinkron módon adja ki az értéket az adatfolyamból. Teljes elemeket bocsát ki amikor az adatfolyam befejeződik, vagy hibajelzést ad, ha az adatfolyam meghibásodott volna. Fel kell iratkoznunk egy observable-re, hogy megkapjuk az értékét. A feliratkozás során opcionálisan 3 értéket adhat vissza: next(), error(), complete(),. A next(), callback-et hívja amikor az érték megérkezik az adatfolyamba, ezt az értéket argumentumként adja át a következő hívásnak. Ha az error() callback történik, amikor valami hiba fordul elő. És a complete() callback-et kapja amikor az adatfolyam befejeződik.
A promise egy JavaScript objektum, amely értéket állít elő aszinkron művelet sikeres végrehajtása során, ha ez a művelet nem hajtódik végre sikeresen akkor hibát generál.
Az Angular-ban a promise úgy definiálható, hogy egy ismert callback függvényt adunk át argumentumként, a függvény általában 2 argumentumot használ: resolve és reject.
A promise objektumnak van egy .then és egy .catch metódusa, amelyek kezelik a teljesült eredményeket és hibákat, ha előfordultak. Az aszinkron műveltek eredményét adatként vagy hibaként kapják meg.

Szakdolgozat II-ben elkészítendő funkciók:

A következő félévben szeretném, még a projektet kiegészíteni/tovább fejleszteni. Szeretném fejleszteni a webshopot, hogy a különböző termékek egymás mellett jelenjenek meg, és ha rajuk kattint a vásárló akkor ott tudja megtekinteni a termék leírását, kommentelni, értékelni a terméket, illetve itt tudja majd kosárba helyezni a terméket. Illetve szeretnék egy szűrő rendszer beletenni. És meg szeretném valósítani, hogy ami termékek kosárba lettek helyezve azokat meg is lehessen rendelni, azaz egy fizetési rendszerrel egészíteném ki a programot.