**Dokumentáció**

Fejlesztők: Móra Barna, Szabó Szebasztián, Kovács Attila

Móra Barna: Backend

Kovács Attila: Frontend

Szabó Szebasztián: Dokumentáció, Teszt

Jegyzet:

Projekt célja

Az ötlet az egy valós problémából indult ki. Ugyanis Pomázon élek viszont semmilyen eseményt nem tudok nyomon követni, valamint haszontalannak és használhatatlannak tartom az önkormányzati oldalt. Megnéztem, hogy hátha más önkormányzati oldalak jobbak vagy használhatóbbak és azt vettem észre sok városnak még ennél is rosszabbul kezelhető oldala van. Ráadásul ezeket elsősorban idősebb korosztály használja főleg számítógépen. A telefonos nézet az végképp használhatatlannak bizonyult. A közösség elsősorban a Facebook csoportban ír és ott lehet olvasni minden városhoz köthető eseményt viszont ezek sajnos nyomon követhetetlenek, egyik értesítés a másik után. Erre jött a megoldás, hogy egy városi fórum/önkormányzati oldalt hozzunk létre. Ez egy sablon amit más önkormányzatok használhatnak és ez sokkal olcsóbb költségvetésből létre tud jönni. Így a városban zajló események a városhoz tartozó oldalon jelennek meg. Ezentúl híreket lokális vagy országos híreket olvashatunk vagy bejelentéseket tehetünk vagy akár tárgyakat hirdethetünk meg amik számunkra már nem kellenek. Így elkerülve a szemetelést is akár.

Felhasznált technológiák

Angular

Bootstrap CSS

Tailwind CSS

Springboot

Frontend

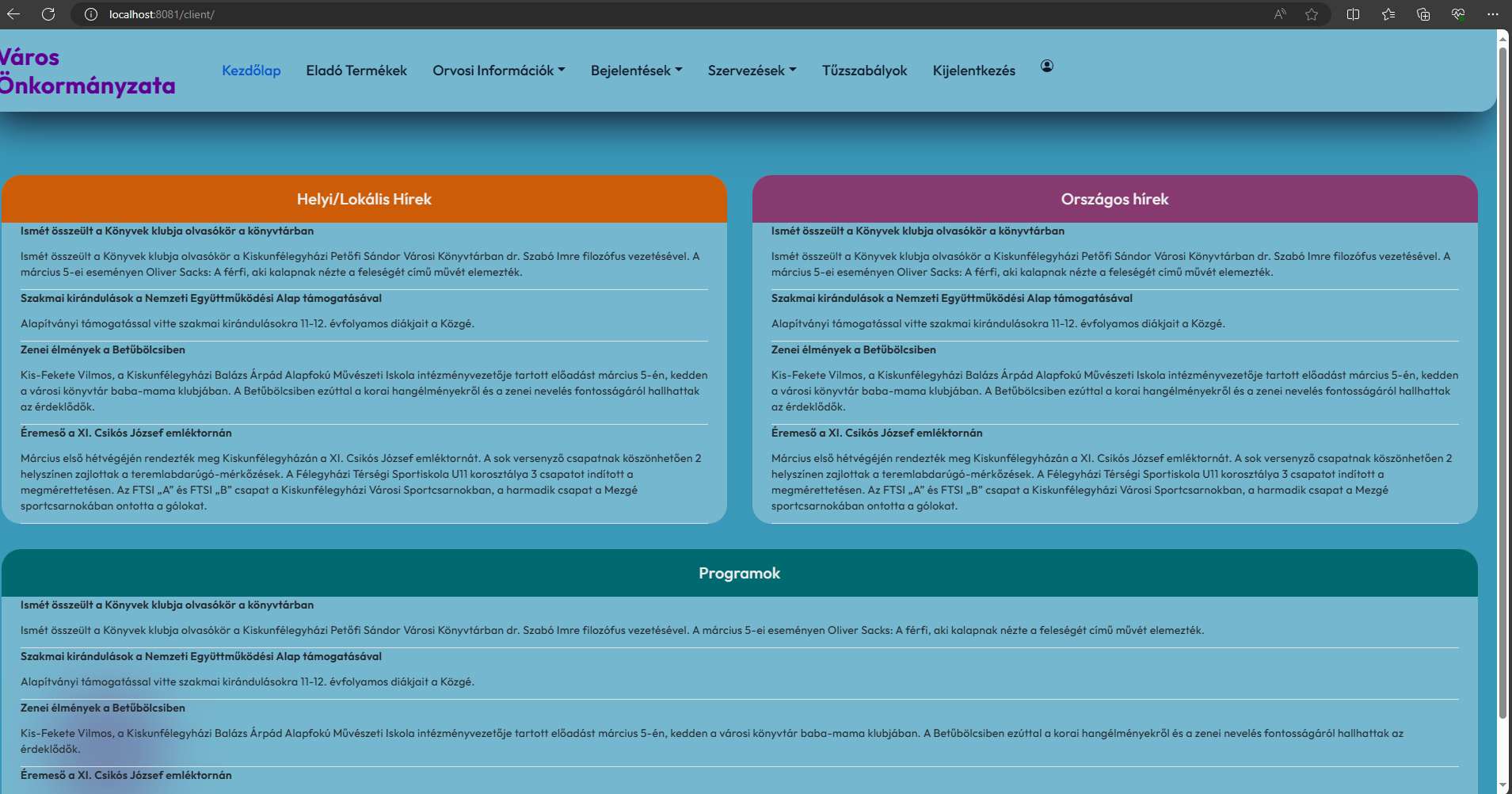
Felhasználói Dokumentáció

Komponensek felhasználói szemszögből:

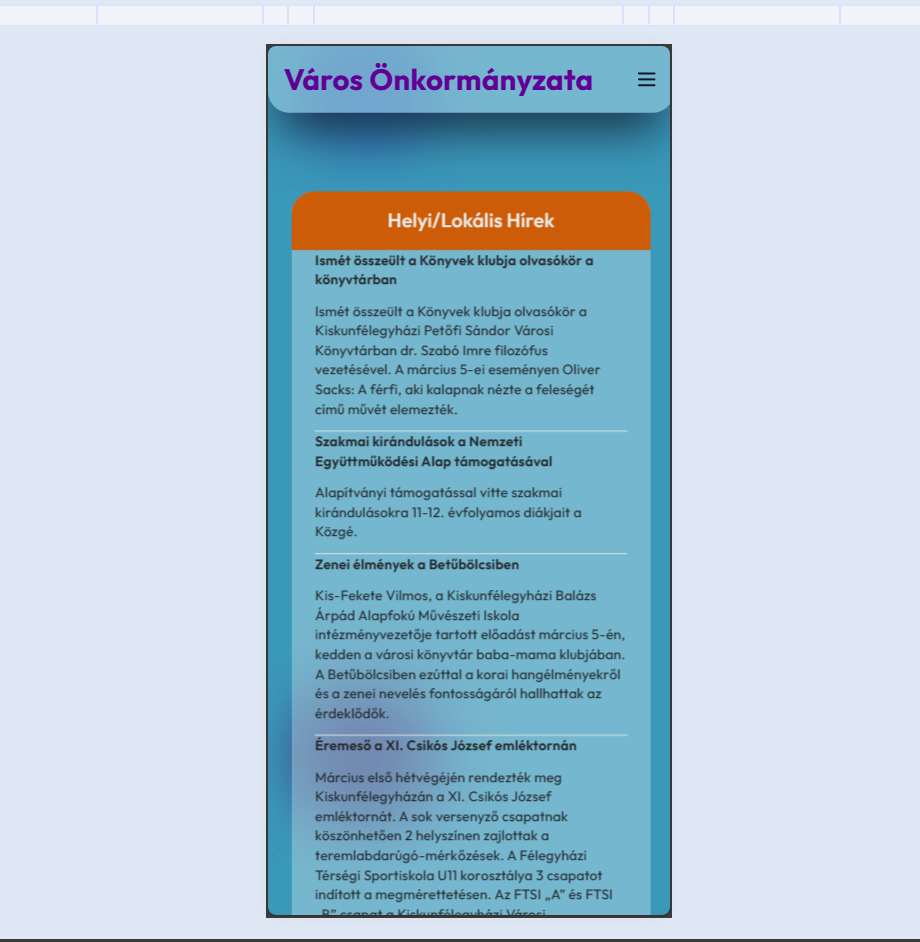
**Body komponens**

Ez a főoldal itt jelennek meg a hírek valamint a programok. Ezentúl az adminnak van lehetősége ezeket szerkeszteni vagy kiegészíteni a híreket. Ez mind asztali, mind telefonos nézetben is elkészült.

*Asztali:*



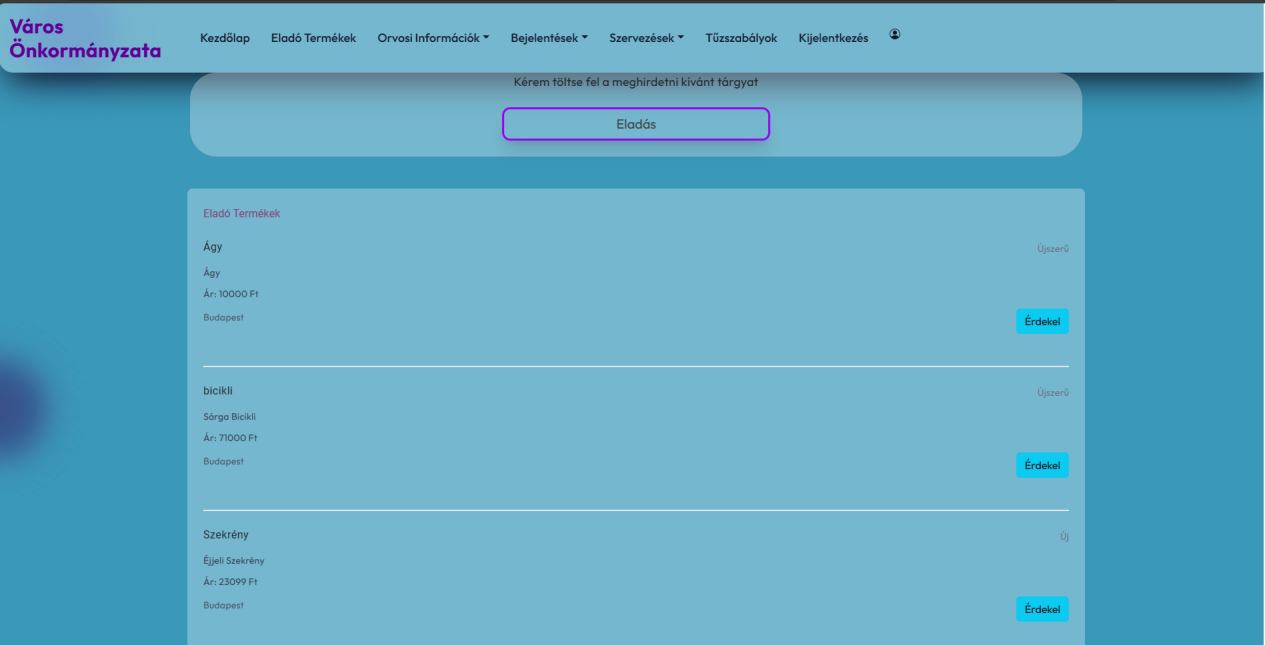
*Telefon:*



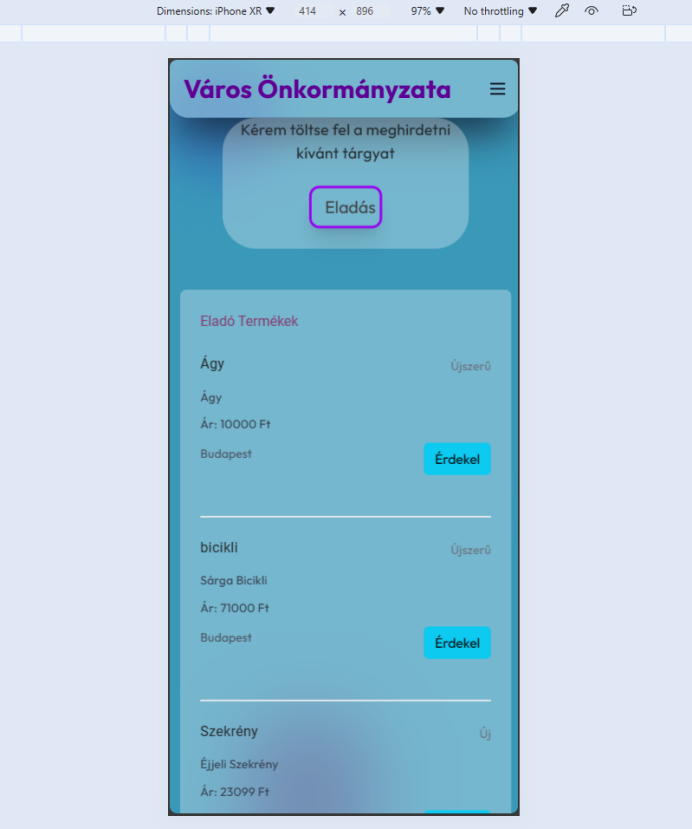
**Elado-termek komponens**

Ide lehet feltölteni a meghirdetni kívánt tárgyakat pénzért vagy akár ingyen ha számodra nincs értéke a tárgynak. Ez természetesen csak bejelentkezés után lehetséges. Valamint a személyes adatokat is csak akkor lehet megnézni ha az illető rendelkezik már fiókkal. Illetve lehet kattintani az előző, illetve a következő oldalakra. Az „Érdekel” gombra kattintva megnézhetjük az eladó személyes adatait.

*Asztali:*



*Telefon:*



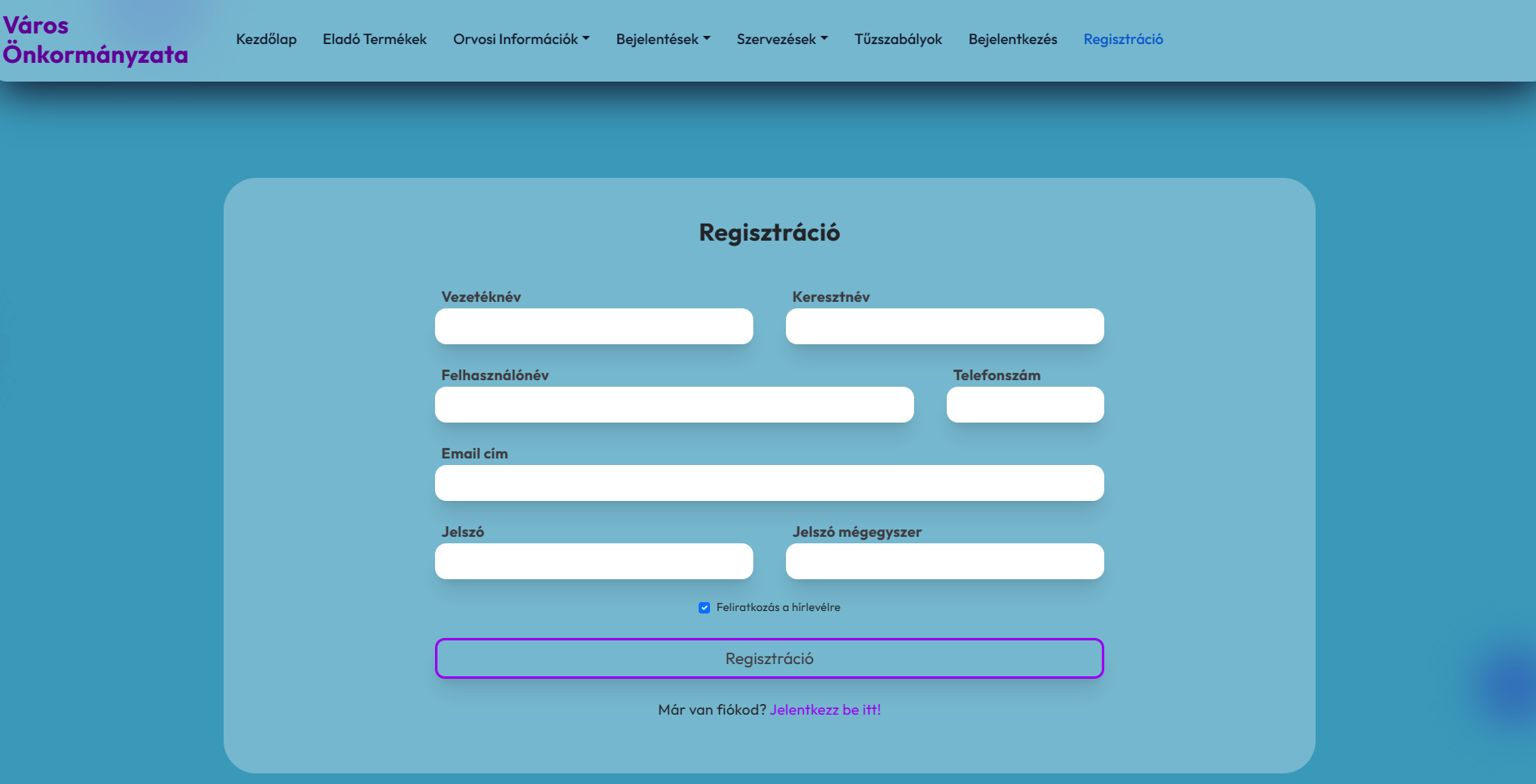
**Login komponens**

A „Bejelentkezés” gombra kattintva továbbvisz minket az ügyfélkapu bejelentkezéshez, itt kell megadni a felhasználó nevet, illetve jelszót.

**Sign-up komponens**

Regisztrácóhoz szükséges komponens ahol a szükséges adatok kitöltése után lehet regisztrálni vagy tovább lehet navigálni a bejelentkezésre ha az illetőnek van már fiókja. Ha nem megfelelőek az adatok, felugrik a popup menü.

*Asztali:*

**

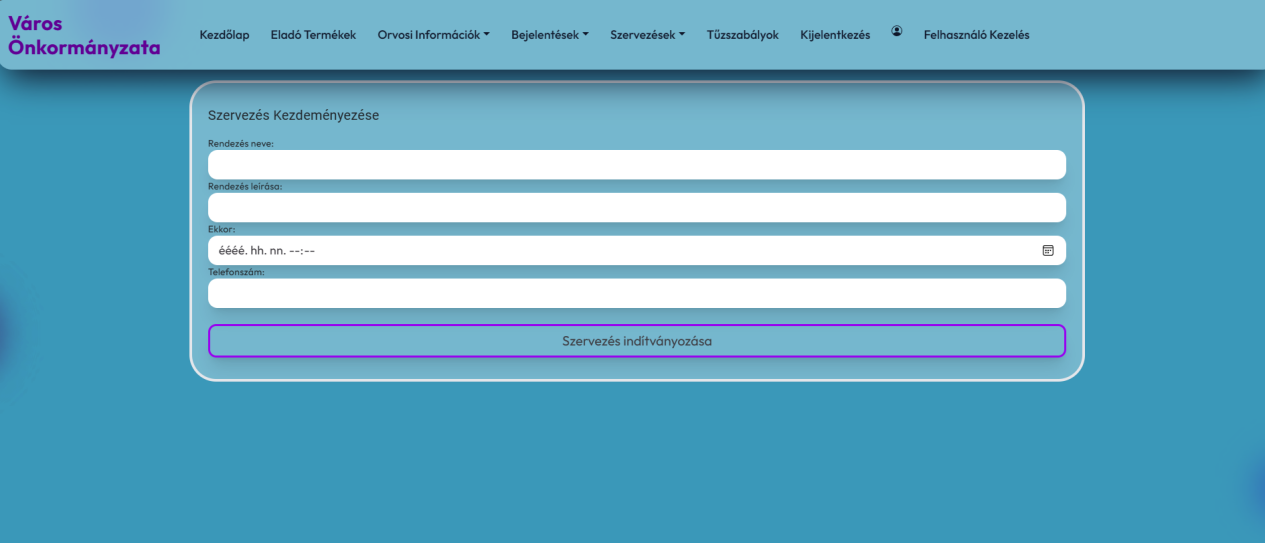
*Telefon:*



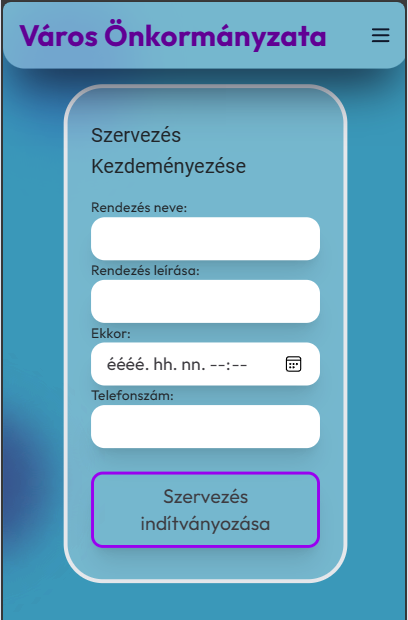
**Szervezeskezdemeny komponens**

Itt kizárólag csak a cégek tudnak rendezvényeket kiírni (orgAdmin). Egy példa; ha egy spar vagy más üzletlánc szeretné magát promotálni akár létrehozhat egy kezdeményezést hogy spar futás ehhez pontos dátumot és rendezvényleírást kell adnia, utána az önkormányzat bírálja el hogy engedélyezi vagy sem és utána a programok résznél a body komponensnél jelenítik meg.

*Asztali:*



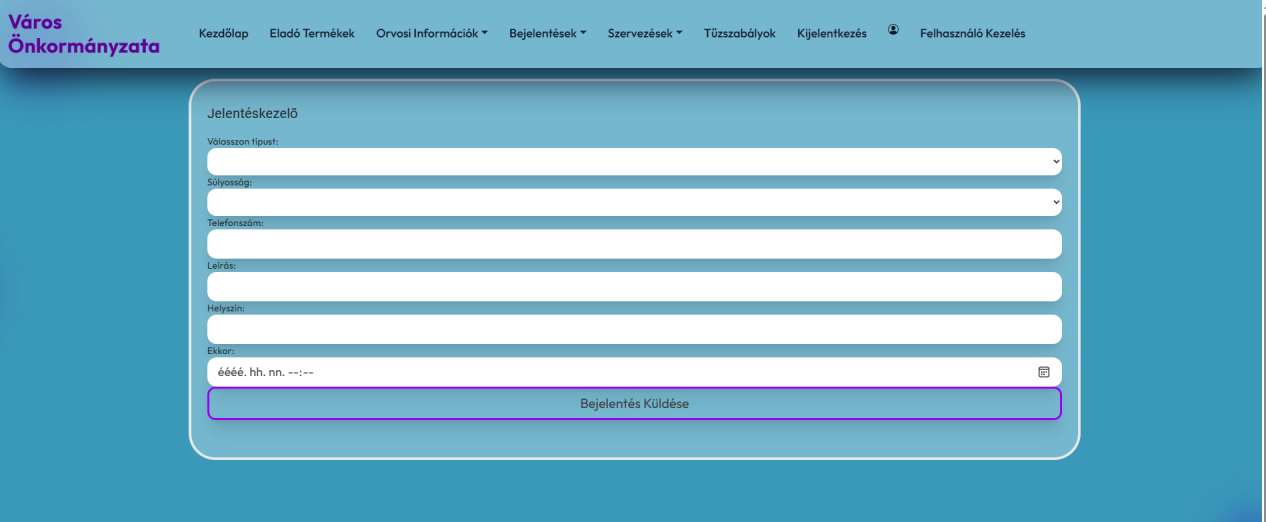
*Telefon:*



**Jelenteskezelo komponens**

A jelentéskezelő egy olyan felület ahol problémát lehet jelenteni. Többféle típus van (Közterületi, Úthiba, Víz-gáz, Lomtalanítás, Szemétszállítás), ezek közül lehet választani és egy súlyosság választó gomb is van benne. Ez annyit takar, hogy ha csőtörés van azonnali „Nagyon Súlyos”-ös súlyosságú elem viszont ha szemetet raktak le valahol akkor az „Súlyos”-as súlyosságú.

*Asztali:*



*Telefon:*



Frontend

Fejlesztői Dokumentáció

**Body** **komponens**

A BodyComponent osztály felelős a hírek kezeléséért és megjelenítéséért a felhasználói felületen. Itt tárolódnak az új cikkek, valamint az országos és helyi hírek listái, valamint azok a feliratkozások, amelyek figyelik a hírszolgáltatást és az autentikációs szolgáltatást.

***Tulajdonságok****:*

*news:* *Hir[]* - Tömb az új hírek tárolására.

*newArticle:* *Hir* - Egy Hir típusú példány az új hírcikk adatainak tárolására.

*subscriptions: Subscription[]* - Tömb az előfizetésekre az emlékeztetők biztosítására.

*user: ProfilAdatok | null* - A felhasználói profil adatai vagy null, ha nem elérhető.

*userRoles: any* - Objektum a felhasználói szerepek tárolására.

*international: Hir[]* - Tömb az nemzetközi hírek tárolására.

*national: Hir[]* - Tömb az országos hírek tárolására.

*local: Hir[]* - Tömb a helyi hírek tárolására.

*orgs: { id: string; name: string }[]* - Tömb a szervezetek adatainak tárolására.

***Metódusok:***

*ngOnInit(): void* - Akkor hívódik meg, amikor az ***Angular*** befejezte az összes adatkötött tulajdonság inicializálását. Hívja a *getUserInfo()* és a *getNews()* metódusokat.

*ngOnDestroy(): void* - Akkor hívódik meg, amikor az ***Angular*** megsemmisíti a komponenst. Leiratkozik az összes előfizetésről a memóriaszivárgások elkerülése érdekében.

*getNews(): void* - Metódus a helyi, országos és nemzetközi hírek lekérdezésére.

*getInterNational(): void* - Metódus a nemzetközi hírek lekérdezésére a *NewsService* segítségével.

*getNational(): void* - Metódus az országos hírek lekérdezésére a *NewsService* segítségével.

*getLocal(): void* - Metódus a helyi hírek lekérdezésére a NewsService segítségével.

*getUserInfo(): void* - Metódus a felhasználói információk és szerepek lekérdezésére az AuthService segítségével.

*changeOrg(event: Event): void* - Eseménykezelő a szervezet kiválasztásának változására.

*submitNewArticle(): void* - Metódus az új hírcikk beküldésére. Ellenőrzi, hogy a felhasználó szervezeti adminisztrátor-e, majd beküldi a hírt.

*deleteLocalNews(news: any): void* - Metódus a helyi hírek törlésére.

*deleteNews(news: Hir): void* - Metódus a hírek törlésére a NewsService segítségével.

*editLocalNews(news: any): void* - Metódus a helyi hírek szerkesztésére.

*editNews(news: Hir): void* - Metódus a hírek szerkesztésére a NewsService segítségével.

*NgFor\**-al végigmegyünk a kiválasztott típusú híreken, ami lehet Lokális, Nemzetközi vagy Program. A címe a „*newArticle.title*” valamint a tartalma „*newArtictle.content*”. A típus a egy **Select** választó gomb, amivel megadhatjuk a *value*-t, ami lehet „*LOCAL*”, „*NATIONAL*” és „*INTERNATIONAL*”. A küld gombra kattintva (submitNewArticle) továbbítjuk a Backendnek.

**Elado-Termekek komponens**

Termékek feltöltéséért és meghirdetéséért felelős komponens; Feltölteni, letörölni, vagy szerkeszteni lehet az adott hirdetéseket, különböző megadható tulajdonsággal, legyen az használtság (*„new”*, *„newish”*, *„used”*) és ár *(„termekek”*, vagy *„termekekFree”*, ha ingyenes), *ngFor*-al végigmegyünk a termékeken, megjelenik a termékek adatai.

***Tulajdonságok:***

*showSellerInfo: boolean* - Logikai érték arra vonatkozóan, hogy megjelenítsük-e az eladó információit.

*eladoTermek: EladoTermekAdatok* - Az új termék adatait tároló objektum.

*buttonClicked: boolean* - Logikai érték arra vonatkozóan, hogy a gombra kattintottak-e.

*pagenum: number* - Oldalszám az oldalak navigálásához.

*userRoles: any -* Felhasználói szerepek tárolása.

*user: ProfilAdatok | null* - Felhasználói profil adatai vagy null, ha nincs elérhető.

*subscriptions: Subscription[]* - Az előfizetések tömbje a memóriaszivárgás elkerülése érdekében.

*conditions: { key: string, text: string }[]* - Tömb az állapotok kulcsa és szövegének tárolására.

*termekek: TermekKepekkel[]* - Tömb az eladó termékek tárolására képekkel együtt.

*termekekFree: TermekKepekkel[]* - Tömb a meghirdetett termékek tárolására képekkel együtt.

***Metódusok:***

*getUserInfo(): void* - Felhasználói információk és szerepek lekérdezése az AuthService segítségével.

*setupDataAndSend(): void* - Az új termék adatainak beállítása és küldése a szerverre.

*ngOnInit(): void* - Akkor hívódik meg, amikor az Angular befejezte az összes adatkötött tulajdonság inicializálását. Inicializálja a felhasználói adatokat és a termékek listáját.

*getTermekek(pageNum: number, price: number): void* - Metódus az eladó termékek lekérdezésére a TermekkezeloService segítségével.

*getTermekekFree(pageNum: number): void* - Metódus a meghirdetett termékek lekérdezésére a *TermekkezeloService* segítségével.

*addBoolErdekel(): void* - Logikai értékek hozzáadása az "erdekel" tulajdonsághoz.

*onSubmit(): void* - Beküldés gomb eseménykezelője.

*toggleForm(): void* - Űrlap megjelenítésének váltása.

*elrejt(): void* - Űrlap elrejtése.

*translateCondition(cond: string): string* - Állapot szövegfordító metódus.

*toggleSellerInfo(item: any): void* - Az eladó információinak megjelenítésének váltása.

*ngOnDestroy(): void* - Akkor hívódik meg, amikor az **Angular** megsemmisíti a komponenst. Leiratkozik az összes előfizetésről a memóriaszivárgások elkerülése érdekében.

***Képfeltöltéshez kapcsolódó metódusok:***

*\_onFilesChanged(files: FileList): void* - Fájlok kiválasztásának eseménykezelője.

*\_onReset(): void* - Kiválasztott fájlok törlésének eseménykezelője.

*\_reset(): void* - Fájlkiválasztó visszaállítása.

***Oldalnavigációval kapcsolatos metódusok:***

*decreasePageNum(): void* - Oldalszám csökkentése.

*increasePageNum(): void* - Oldalszám növelése.

***Adatszolgáltatás (TermekkezeloService)***

A TermekkezeloService felelős az eladó termékek adatkezeléséért.

***Metódusok:***

*getTermekekFree(pageNum: number): Observable<TermekKepekkel>* - Metódus a meghirdetett termékek lekérdezésére.

*getTermekek(pageNum: number, price: number): Observable<TermekKepekkel>* - Metódus az eladó termékek lekérdezésére.

*postTermek(termek: any): Observable<any>* - Metódus az új termék beküldésére.

*updateTermek(body: any): Observable<any>* - Metódus a termék frissítésére.

*deleteTermek(id: any): Observable<any>* - Metódus a termék törlésére.

*getTermekekForUserById(id: any): Observable<TermekKepekkel>* - Metódus az adott felhasználó által meghirdetett termékek lekérdezésére.

**Termek-Eladas komponens**:

Itt tudjuk felvenni a termékeket, *ngModel*-t használtunk, ezzel adjuk meg a termékek leírását, állapotát, elhelyezkedését, az email címet és a telefonszámot. Kitudunk választani képet, majd ezeket az elemeket a *Termek-Eladas* komponens TS-éből továbbítjuk a **Backend** felé.

Tartalmaz pop-up menüt, ha valamelyik rubrika érvénytelenül van kitöltve, sikertelen feltöltést ír ki.

**Felhaszkeres** **komponens**

Felhasználók kilistázására, szerkesztésére (*updateUser()*) és felfüggesztésére (*deleteUser()*) fejlesztett komponens. *AdminAdatok Model* alapján ellenőriz (*„id”*, *„email”*, *„userName”*, *„firstName”*, *„lastName”*, *„phone”*). Ezt az oldalt csak az ADMIN érheti el.

A HTML-ben végigmegyünk a sorokon, és *ngFor* segítségével kiíratjuk az összes keresett felhasználót a *models*-ben írtunk hozzá egy adattáblát ami kezeli a típusait és ezt a *models*-t hívtuk meg tömbként. Itt tárolódnak el az adatok.

A *service*-ben hivatkozunk a végpontokra, itt a **CRUD** műveletek segítségével tudunk törölni és szerkeszteni.

***Tulajdonságai:***

*value* - A keresési érték.

*pageNum* - Az oldalszám.

*category* - A keresési kategória.

*getProfilAdatok(value: string, pageNum: number, category: string): Observable<ProfilAdatok[]>* - Metódus a profil adatok lekérdezésére.

*updateUser(body: any): Observable<any>* - Metódus a felhasználó frissítésére.

*body* - A frissítendő felhasználó adatai.

*deleteUser(id: any): Observable<any>* - Metódus a felhasználó törlésére.

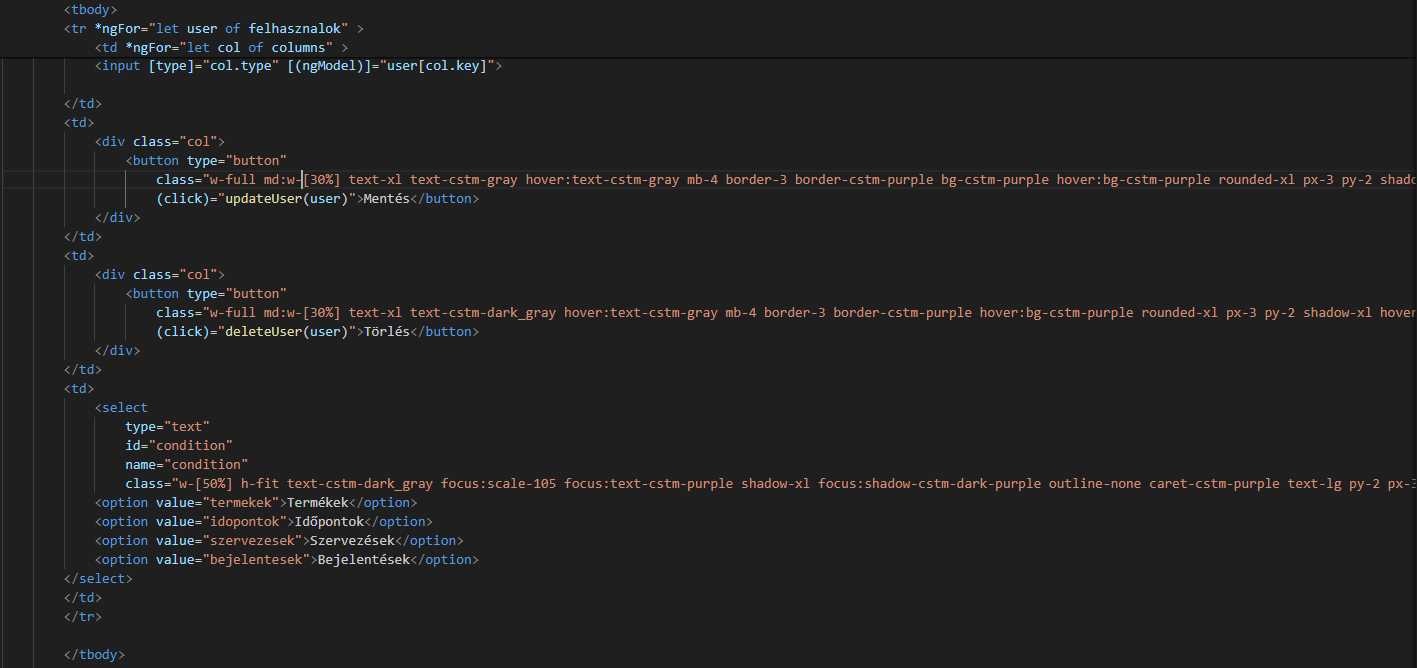
*id* - A törlendő felhasználó azonosítója.

***Metódusok:***

*getAdminAdatok(value: string, pageNum: number, category: string): Observable<AdminAdatok[]>* - Metódus az adminisztrációs adatok lekérdezésére.

***Felhasználók Táblázat Megjelenítése***

Az alábbi HTML kód egy táblázatot jelenít meg a felhasználók adataival:



A táblázat minden sorában lehetőség van a felhasználó adatainak szerkesztésére és törlésére. A "Típus választó" oszlopban egy legördülő lista található a felhasználó típusának kiválasztására.

Az *\*ngFor* direktíva segítségével iterálunk végig a felhasználók listáján, és minden felhasználó adatai egy-egy sorban jelennek meg. A \**ngFor* direktíva segítségével több oszlopot is létrehozhatunk az adatok megjelenítéséhez.

**Hiba** **komponens**

Hibás URL megadására tervezett oldal.

**Jelenteskezelo** **komponens**

A Jelentéskezelő komponens a felhasználók jelentéseinek kezeléséért felelős. Az alábbiakban bemutatjuk a komponens részleteit:

*InputForm*-al adjuk meg hogy csak és kizárólag *user* típusú felhasználó tudjon

jelentést írni. itt a „post” metódust használjuk, amit a *service*ben írtunk meg és a **TS**-ben hívunk meg.

***Űrlap:***

Az űrlap lehetővé teszi a felhasználók számára, hogy új jelentést küldjenek be. Az űrlapon az alábbi mezők találhatók:

*Típus -*A felhasználó kiválaszthatja a jelentés típusát a legördülő menüből. Lehetősége van *közterületi*, úthiba, víz-gáz, lomtalanítás vagy szemétszállítás jelentést küldeni.

*Súlyosság -* A jelentés súlyosságát lehet kiválasztani egy másik legördülő menüből, amely tartalmazza a "Nagyon Súlyos", "Súlyos", "Zavaró", "Meh" és "Megjegyzés" opciókat.

*Telefonszám -* A felhasználó megadhatja a telefonszámát, amelyen keresztül a hatóságok elérhetik.

*Leírás* - A jelentés tartalmazhat egy rövid leírást a problémáról vagy incidensről.

*Helyszín -* A jelentéshez tartozó helyszín megadása.

*Dátum* - A jelentés dátumának és időpontjának megadása.

Az űrlapon található egy "Bejelentés Küldése" gomb, amelyet a felhasználó megnyomhat a jelentés elküldéséhez.

***Felugró Üzenetek***

A komponens két felugró üzenetet is tartalmaz, amelyek a jelentés elküldésének vagy sikertelen elküldésének visszaigazolására szolgálnak:

*Jelentés Elküldve*: Ha a jelentés sikeresen elküldésre került, egy felugró ablak jelenik meg a "Jelentés elküldve!" címmel és a megfelelő üzenettel.

Sikertelen Jelentés: Ha a jelentés elküldése sikertelen volt, egy másik felugró ablak jelenik meg a "Sikertelen jelentés!" címmel és a megfelelő üzenettel.

Mindkét üzenet tartalmaz egy "Bezárás" gombot, amely segítségével a felhasználó bezárhatja az üzenetet.

***Autentikáció és Szolgáltatások***

A komponens használja az *AuthService* és *JelenteskezeloService* szolgáltatásokat az autentikációhoz és a jelentések kezeléséhez.

***JelenteskezeloService***

A *JelenteskezeloService* szolgáltatás felelős a jelentések kezeléséért és a hozzájuk kapcsolódó HTTP kérések végrehajtásáért. A szolgáltatás az alábbi funkciókat kínálja:

*postJelentes(body: BejelentesAdatok)* - Új jelentés beküldése.

*updateJelentes(body: any)* - Jelentés frissítése.

*deleteJelentes(id: any)* - Jelentés törlése.

*getSajatJelentesek(id: any)* - Felhasználó saját jelentéseinek lekérése.

*getAllJelentesek(id: any)* - Minden jelentés lekérése.

*updateJelentesOrg(body: any)* - Szervezeti jelentés frissítése.

*deleteJelentesOrg(id: any)* - Szervezeti jelentés törlése.

*getSajatJelentesekOrg(id: any)* - Szervezeti jelentések lekérése.

*searchName(pageNum:number, name:string)* - Név alapján keresés a szervezetek között.

A szolgáltatás használja a *BejelentesAdatok* modellt a jelentések adatainak reprezentálásához.

***Komponens Életciklusa***

A *JelenteskezeloComponent* életciklusának során figyelemmel kíséri a feliratkozásokat, és azok megszüntetésekor leiratkozik a feliratkozásokról. Ez segít elkerülni a memóriaszivárgást és a nem kívánt viselkedést.

**Navigáció komponens**

A NavComponent felelős a navigációs sáv megjelenítéséért és működéséért az alkalmazásban. Az összecsukható menü és a navigációs elemek kezelése, valamint a felhasználói jogosultságok alapján történő elemek megjelenítése is ebben a komponensben történik.

***Tulajdonságok:***

*isMenuOpen* - Logikai érték, amely azt jelzi, hogy az összecsukható menü éppen nyitva vagy zárva van.

roles: A felhasználó szerepének adatait tároló objektum.

*isDesktopView* - Logikai érték, amely azt jelzi, hogy az alkalmazás éppen asztali nézetben van-e.

*isMobileView* - Logikai érték, amely azt jelzi, hogy az alkalmazás éppen mobil nézetben van-e.

*isNavbarOpen* - Logikai érték, amely azt jelzi, hogy a navigációs sáv éppen nyitva vagy zárva van.

*isMobile* - Logikai érték, amely azt jelzi, hogy az alkalmazás éppen mobil eszközön fut-e.

*subs* - Feliratkozásokat tároló tömb az Observable-k figyeléséhez.

***Metódusok:***

*toggleMenu()* - Az összecsukható menü állapotának váltása (nyitva vagy zárva).

*toggleNavbar()* - A navigációs sáv állapotának váltása (nyitva vagy zárva).

*onResize(event: any)* - A böngészőablak méretváltozásának eseménykezelője, amely frissíti a képernyőméret függvényét.

*checkScreenSize()* - Ellenőrzi az alkalmazás jelenlegi képernyőméretét, és beállítja az *isMobileView* és *isDesktopView* tulajdonságokat ennek megfelelően.

*logout()* - Kijelentkezési funkció, ami a *AuthService* segítségével végzi a kijelentkezést.

*redirectToLogin()* - Átirányítás a bejelentkező oldalra.

***Életciklus metódusok:***

*ngOnInit() -* Az alkalmazás indulásakor lefutó metódus, amely beállítja a kezdeti értékeket és feliratkozik a felhasználói szerepek *Observable*-jára.

Animáció:

*fadeInOut()* - Az összecsukható menü animációját kezelő trigger. Amikor az menü megjelenik vagy eltűnik, a *void* és a nem *void* állapotok közötti átmenetet vezérli.

***Megjegyzés:***

A *NavComponent* alapvetően egy **Angular** komponens, amely felelős a navigációs sáv működtetéséért és a felhasználói interakciók kezeléséért az alkalmazásban. A HTML kódot az **Angular** templating rendszerével kombinálva a *NavComponent* dinamikusan változtathatja a megjelenést és a funkcionalitást a felhasználói interakciók alapján.

**Profil komponens**

Profil adatokat kér be a bejelentkezett felhasználóról.

Lehetőséget biztosít a felhasználó számára a profilinformációk szerkesztésére.

Űrlapot biztosít a profil részleteinek szerkesztéséhez, mint például a keresztnév, vezetéknév, email, felhasználónév és telefonszám.

Funkcióval rendelkezik az átváltás közötti szerkesztési mód és megtekintési mód között.

Funkcióval rendelkezik a profilban történő változtatások mentéséhez.

Tartalmaz egy képfeltöltési funkciót.

***Tulajdonságok:***

*profil -* Objektum a felhasználó profiladatainak reprezentálásához.

*user* - Objektum a bejelentkezett felhasználó adatainak reprezentálásához.

*isEditing* - Logikai érték a szerkesztési és megtekintési mód közötti váltáshoz.

*subscription* - Tömb az előfizetések tárolására, amik a komponens megsemmisítésekor történő takarításra szolgálnak.

*isDisabled* - Logikai érték a mezők letiltásához, amikor nincs szerkesztési mód.

***Metódusok:***

*ngOnInit()* - Életciklus horgonycsomópont, amelyet akkor hívnak meg, amikor az **Angular** inicializálja az összes adatkötött tulajdonságot egy direktívában.

*ngOnDestroy()* - Életciklus horgonycsomópont, amelyet azonnal megelőzően hív meg az **Angular**, mielőtt megsemmisítené a direktívát vagy a komponenst.

*enableEdit()* - Metódus a profilmezők szerkesztésének engedélyezésére.

*getUserInfo()* - Metódus a felhasználói információk lekéréséhez, ideértve a profiladatokat is.

*toggleEditing()* - Metódus az átváltás közötti szerkesztési mód és megtekintési mód között.

*saveChanges()* - Metódus a profilban történt változtatások mentésére.

*onFileSelected(event: any)* - Metódus a fájlkiválasztás kezelésére képfeltöltéshez.

*readImage(file: File)* - Metódus a fájl beolvasásához és átalakításához base64 kóddá.

**Saját szervezések komponenes**

A HTML kód definiálja az oldal felhasználói felületének struktúráját és a komponensek közötti adatkötéseket.

Az első táblázatban egy űrlapot találunk a szervezet nevének kiválasztásához és a szervezetek lekéréséhez.

Az űrlap része egy beviteli mező a szervezet nevének megadásához, valamint egy gomb a lekérés funkció indításához.

A második táblázatban jelennek meg a szervezések adatai, melyeket szerkeszteni és törölni is lehet.

**Angular** komponens:

Az **Angular** komponens felelős az alkalmazás üzleti logikájának és működésének irányításáért.

Az *OnInit* interfész implementálása a komponens életciklusának kezelésére szolgál.

Az *getUserInfo* metódus lekéri a felhasználó adatait és szerepköreit.

Az inputForm metódus létrehozza az új szervezést.

A *getSajatSzervezes* metódus lekéri a felhasználó saját szervezéseit.

Az *updateSzervezes* és *deleteSzervezes* metódusok a szervezések frissítéséért és törléséért felelősek.

Szolgáltatások és modellek:

A szolgáltatások és modellek az alkalmazás adatmodelljét és logikáját definiálják.

A *SzervezesService* és az *AuthService* szolgáltatások az alkalmazás adatkezelését végzik.

A *ProfilAdatok* és *SzervezesAdatok* modellek az alkalmazás adatstruktúráját definiálják.

A dokumentáció segít megérteni az alkalmazás felépítését és működését, valamint megkönnyíti a kódolást és a fejlesztést.