**Dokumentáció**

Fejlesztők: Kovács Attila, Szabó Szebasztián, Móra Barna

Kovács Attila:

Frontend; Admin felület, Mobil és Asztali Dizájn

Szabó Szebasztián:

Frontend; Szervező felület, Hírek, Dokumentáció

Móra Barna:

Backend

**Tartalomjegyzék**

* *Mi a projekt célja?*
* *Felhasznált technológiák*
* *Felhasználói dokumentáció* 
  + *Komponensek*
* *Fejlesztői dokumentáció* 
  + *Komponensek*
* *Jövőbeli tervek*

**A Projekt célja**

Az ötlet az egy valós problémából indult ki. Ugyanis egyik csapattagunk Pomázon él, viszont semmilyen eseményt nem tud nyomon követni, valamint haszontalannak és használhatatlannak tartja az önkormányzati oldalt. Megnéztük, hogy hátha más önkormányzati oldalak jobbak vagy használhatóbbak és azt vettük 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

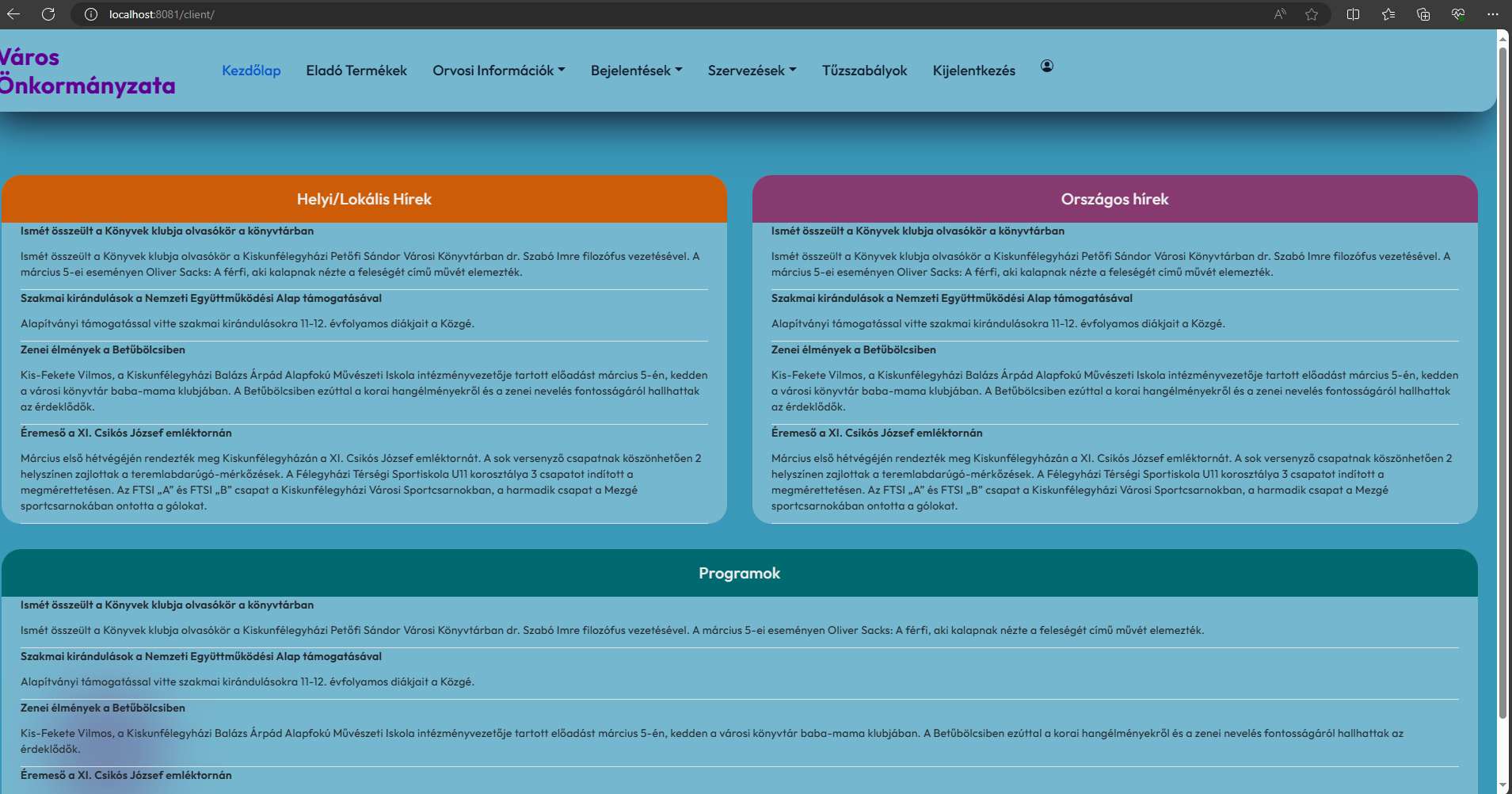
Felhasználói Dokumentáció

Komponensek fejlesztői szemszögből:

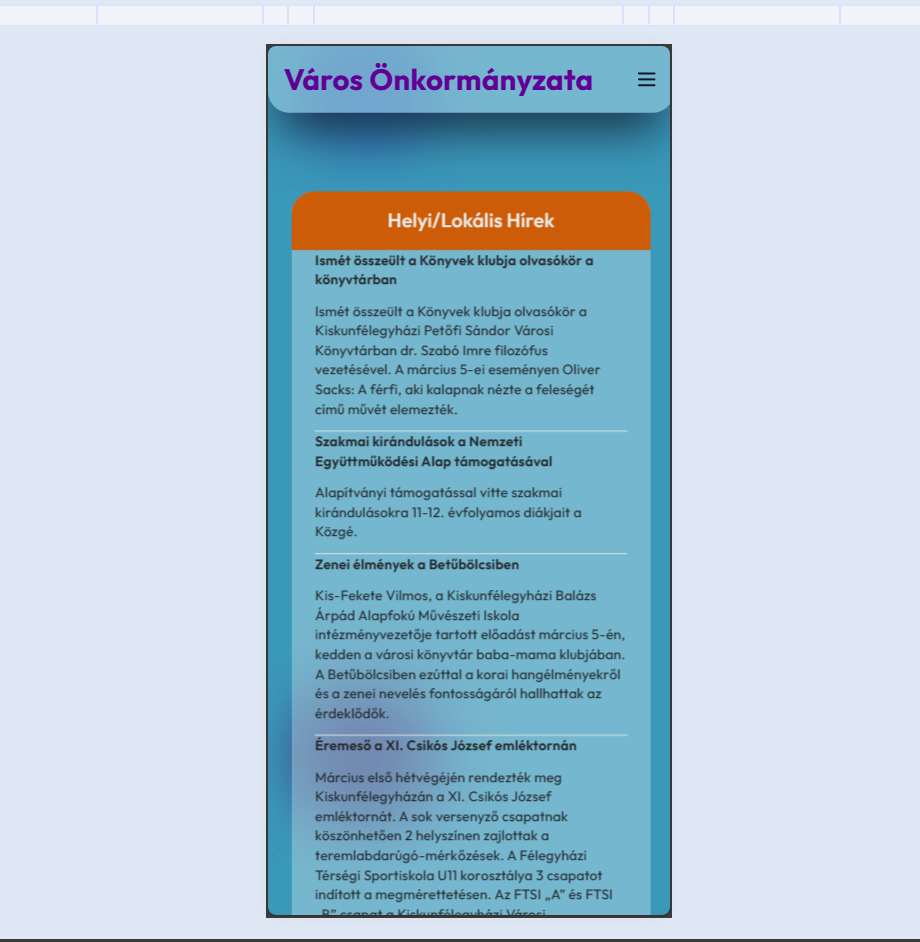
**Body komponens**

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:*



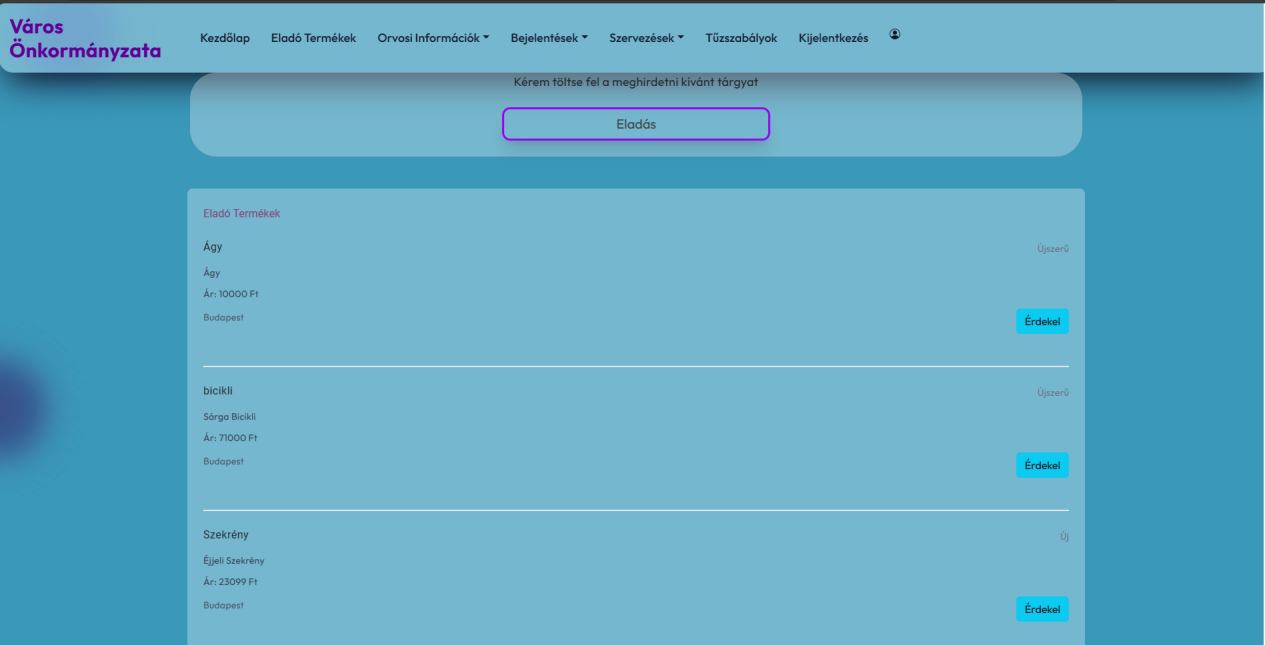
*Telefon:*



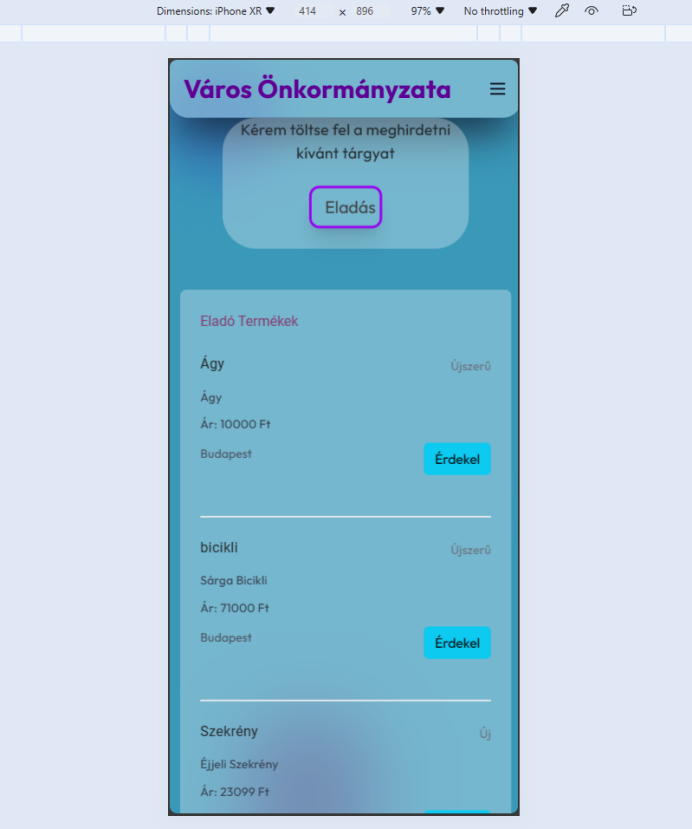
**Elado-termek komponens**

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:*



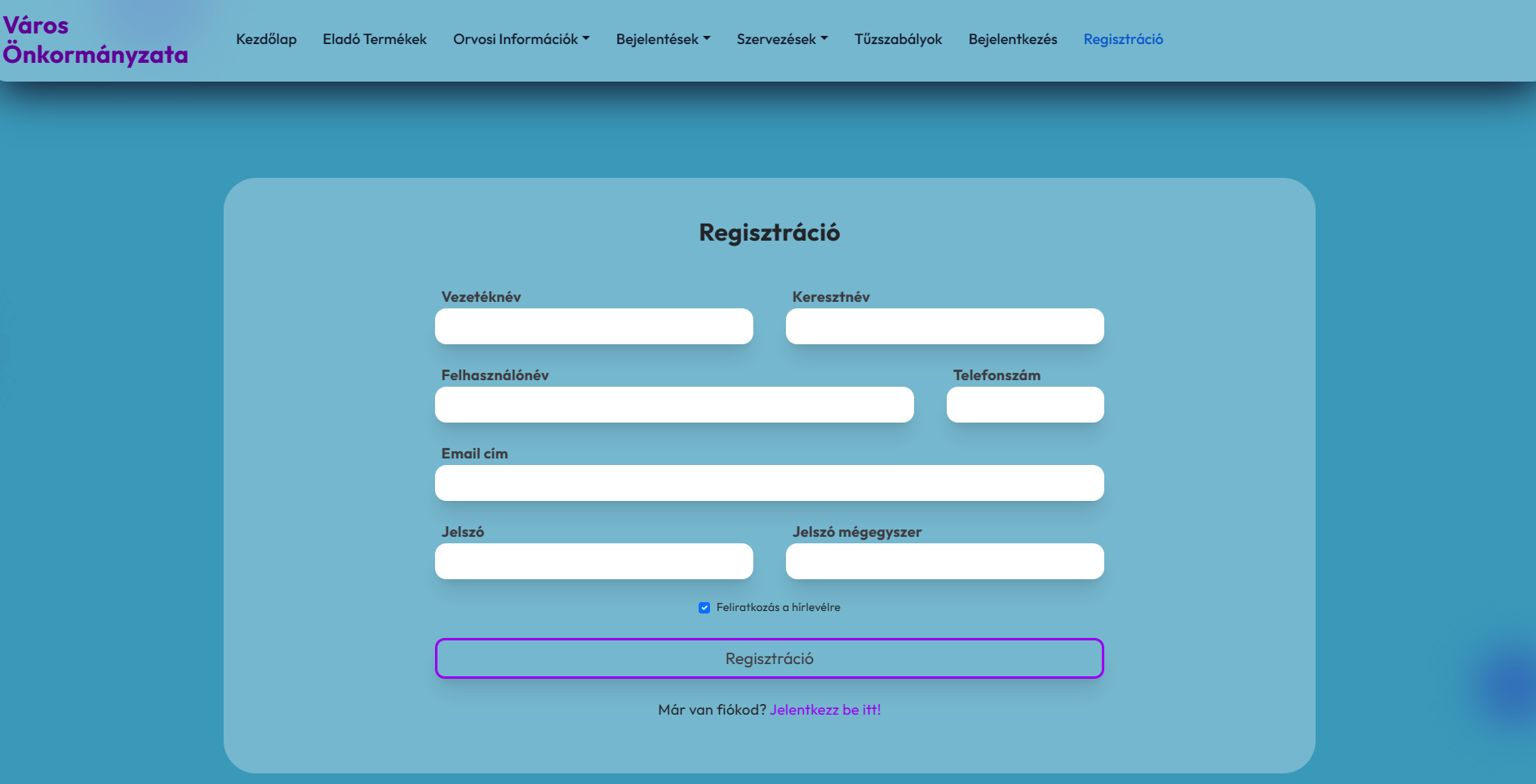
**Login komponens**

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:*

**

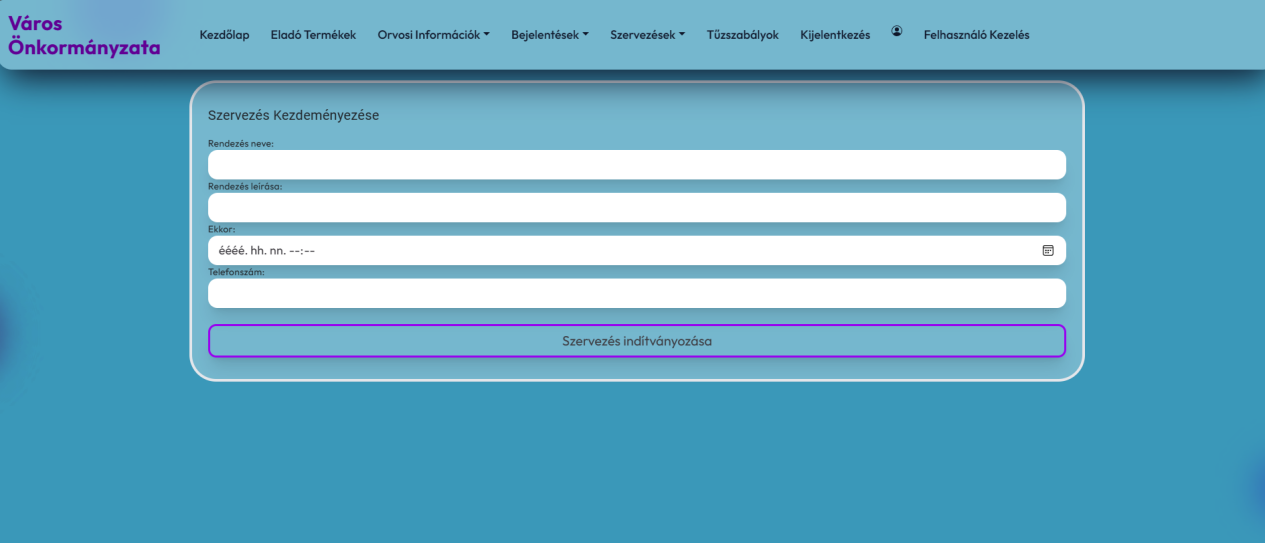
*Telefon:*



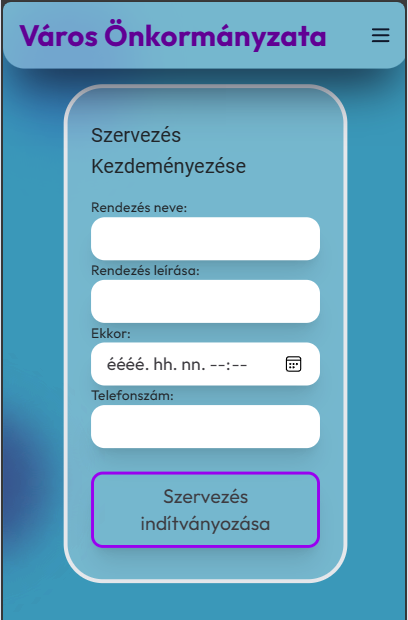
**Szervezeskezdemeny komponens**

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:*



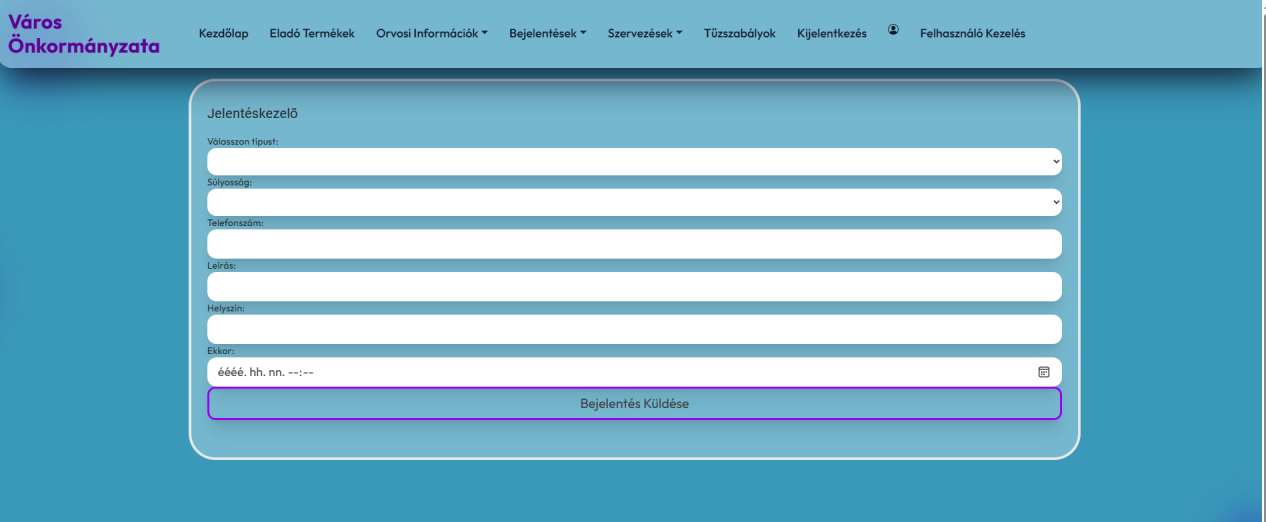
*Telefon:*



**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, admin és szervezői felületen. Itt tárolódnak az új cikkek, valamint a *Nemzeti* és *Helyi* hírek listái.

***Tulajdonságok****:*

* *news:* *Hir[]* - Tömb az új hírek tárolására.
* *newArticle:* *Hir* - Egy *Hir* típusú példány az új hírcikk adatainak tárolására.
* *subscriptions: Subscription[]* – Tömb a feliratkozások tárolásához
* *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 *Nemzeti* hírek tárolására.
* *local: Hir[]* - Tömb a *Helyi* hírek tárolására.
* *orgs: { id: string; name: string }[]* - Tömb a szervezetek adatainak tárolására.

***Metódusok:***

* *ngOnInit(): void* – Inicializálja a *getUserInfo()* és a *getNews()* metódusokat.
* *ngOnDestroy(): void* – Metódus a feliratkozások törlésére memóriaszivárgás 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 a *Nemzeti* 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.

Az *NgFor\** végigmegy a kiválasztott típusú híreken, ami lehet *Lokális*, *Nemzeti* vagy *Nemzetközi*. A címe a „*newArticle.title*” valamint a tartalma „*newArtictle.content*”. A típusa egy **Select** választó gomb, amivel meg lehet adni a *value*-t, ami lehet „*LOCAL*”, „*NATIONAL*” és „*INTERNATIONAL*”. A „*küld*” gombra kattintva (submitNewArticle()) továbbítódik 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[]* – Tömb a feliratkozások tárolásához
* *conditions: { key: string, text: string }[]* - Tömb az állapotok kulcsa és szövegének tárolására.
* *termekek: TermekKepekkel[]* - Tömb a meghírdetett termékek tárolására képekkel együtt.
* *termekekFree: TermekKepekkel[]* - Tömb az ingyenesen meghírdetett 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* - Inicializálja a felhasználói adatokat és a termékek listáját.
* *getTermekek(pageNum: number, price: number): void* - Metódus a meghírdetett termékek lekérdezésére a *TermekkezeloService* segítségével.
* *getTermekekFree(pageNum: number): void* - Metódus az ingyenesen 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* - Metódus a feliratkozások törlésére memóriaszivárgás 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 az ingyenesen meghirdetett termékek lekérdezésére.
* *getTermekek(pageNum: number, price: number): Observable<TermekKepekkel>* - Metódus a meghírdetett termékek lekérdezésére.
* *postTermek(termek: any): Observable<any>* - Metódus az új termék beküldésére.
* *updateTermek(body: any): Observable<any>* - Metódus a termék szerkeztésére.
* *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 lehet felvenni a termékeket, *ngModel*-t használ, ezzel van megadva a termékek leírása, állapota, elhelyezkedése, az email címe és a telefonszáma. Kilehet választani képet, majd ezeket az elemeket a *Termek-Eladas* komponens TS-éből továbbítódik 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 végigmenve a sorokon az *ngFor* segítségével kiíródik az összes keresett felhasználó. a *models*-ben íródott hozzá egy adattábla ami kezeli a típusait és ez a *models* hívódik meg tömbként. Itt tárolódnak el az adatok.

A *service*-ben hivatkozódik a végpontokra, itt a **CRUD** műveletek segítségével lehet 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 szerkeztésére.

*body* - A frissítendő felhasználó adatai.

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

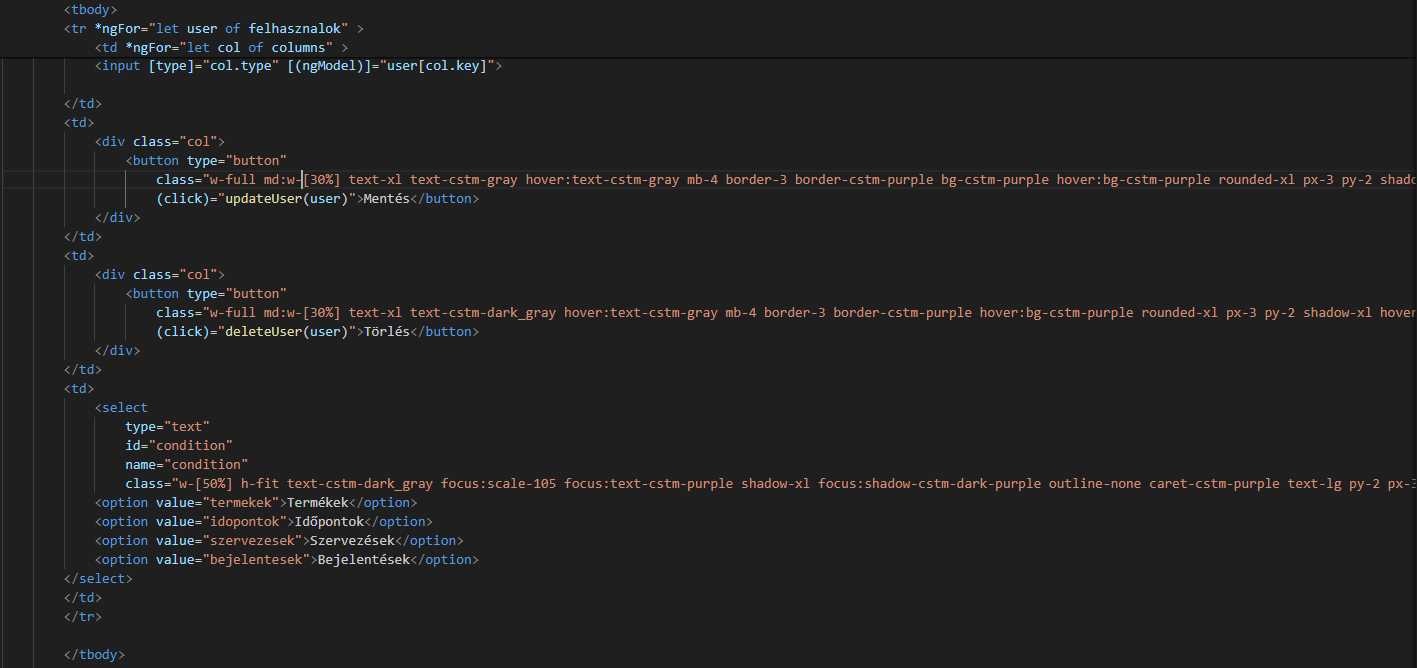
*id* - A törlendő felhasználó azonosítója.

***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 iterálódik végig a felhasználók listáján, és minden felhasználó adatai egy-egy sorban jelennek meg. A \**ngFor* direktíva segítségével több oszlop is létrehozható 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 adódik meg hogy csak és kizárólag *user* típusú felhasználó tudjon jelentést írni. Itt a „*post*” metódust van használva, ami a serviceben lett megírva és a **TS**-ben hívódik 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*" és "*Meh*" 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 szerkeztése.
* *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 a feliratkozások tárolására, amik a komponens megsemmisítésekor történő takarításra szolgálnak.
* *isDisabled* - Logikai érték a mezők letiltásához, amikor nincs szerkesztési mód.

***Metódusok:***

* *ngOnInit()* - Akkor hívódik meg, amikor az **Angular** inicializálja az összes adatkötött tulajdonságot egy direktívában.
* *ngOnDestroy()* – Metódus a feliratkozások törléséhez memóriaszivárgás elkerülése érdekében.
* *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.

**Sajat-szervezesek komponens**

Események lekérésére, kezelésére, szerkesztésére illetve törlésére használt komponens. Ezt a felületet csak az Admin látja.

***Tulajdonságok:***

* *user: ProfilAdatok | null* - A felhasználói profil adatai vagy null, ha nem elérhető.
* *userRoles: any* - Objektum a felhasználói szerepek tárolására.
* *orgs: { id: string; name: string }[]* - Tömb a szervezetek adatainak tárolására.
* *currentOrganization:{[x:string]: any, id: string, name: string}={id:””, name:””}* – Kiválasztott szervezet tárolásához alkalmazott változó
* *szervezesek* – Szervezések tárolására használt tömb
* *subscriptions: Subscription[]* – Tömb a feliratkozások tárolásához

***Metódusok:***

* *getUserInfo(): void* - Metódus a felhasználói információk és szerepek lekérdezésére az *AuthService* segítségével.
* *ngOnInit(): void* – Inicializálja a *getUserInfo()* és a *getNews()* metódusokat.
* *ngOnDestroy(): void* – Metódus a feliratkozások törlésére memóriaszivárgás elkerülése érdekében.
* *getSajatSzervezesekOrg()* – Metódus a szervezések lekéréséhez.
* *updateSzervezes()* – Metódus a szervezések szerkesztéséhez.
* *deleteSzervezes()* – Metódus a szervezések törléséhez.

**Sajat-jelentesek komponens**

Jelentések lekérésére, kezelésére, szerkesztésére illetve törlésére használt komponens. Általános felhasználók csak a sajátjukat, míg az adminok mindenkiéhez hozzá tudnak férni.

***Tulajdonságok:***

* *user: ProfilAdatok | null* - A felhasználói profil adatai vagy null, ha nem elérhető.
* *userRoles: any* - Objektum a felhasználói szerepek tárolására.
* *currentUser: AdminAdatok = new AdminAdatok()* – A kiválasztott felhasználó megjelenítésére alkalmas változó

***Metódusok:***

* *lekeres()* – Metódus jelentés szerkesztéséhez.
* *deleteSajatJelentes()* – Metódus jelentés törléséhez.
* *setCol()* – Beállítja a mezőt.

**Sajat-hirek komponens**

Hírek lekérésére, kezelésére, szerkesztésére illetve törlésére használt komponens. Általános felhasználók csak a sajátjukat, míg az adminok mindenkiéhez hozzá tudnak férni.

***Tulajdonságok:***

* *orgBooleanNews: boolean = false* – logikai érték a hírek megjelenítéséhez
* *currentOrganization: {id: string, name: string, } = {id: „”, name: „”}* – Kiválasztott szervezet megjelenítéséhez változó

***Metódusok:***

* *orgRequest() –* Metódus a hírek lekéréséhez.
* *editNews() –* Metódus a hírek szerkesztéséhez.
* *deleteNews() –* Metódus a hírek törléséhez.

**Jövőbeli tervek**

* *Bejelentések befejezése*
* *Felhasználók általi posztolás, szavazások lehetővé tétele*