**Projekt Labor**

**Alkalmazásfejlesztés web alapú mobil platformra**

Baumgartner János Tancsics László

témavezető Németh Ádám

Abafalvi Balázs

csapattagok

Tartalom

[1. Bevezetés 3](#_Toc56516544)

[2. Felhasználói dokumentáció 4](#_Toc56516545)

[2.1 Alkalmazás telepítése 4](#_Toc56516546)

[2.2 Alkalmazás használata 7](#_Toc56516547)

[2.2.1 Szerkezet 7](#_Toc56516548)

[2.2.1.1 User oldali funkciók 7](#_Toc56516549)

[2.2.1.2 Admin oldali funkciók 11](#_Toc56516550)

[3. Fejlesztői dokumentáció 16](#_Toc56516551)

[3.1 Feladat leírása 16](#_Toc56516552)

[3.2 Az alkalmazott fejlesztői eszközök 17](#_Toc56516553)

[3.2.1 React 18](#_Toc56516554)

[3.2.2 Ionic 19](#_Toc56516555)

[3.3 Adatmodell leírása 20](#_Toc56516556)

[3.3.1 Firebase 20](#_Toc56516557)

[3.3.2 Adatbázis felépítése 21](#_Toc56516558)

[3.4 Munkaterv 23](#_Toc56516559)

[3.4.1 Feladat és egyéb tudnivalók átbeszélése 23](#_Toc56516560)

[3.4.2 Ismerkedés a technológiákkal 23](#_Toc56516561)

[3.4.3 Funkcionális követelmények 23](#_Toc56516562)

[3.4.4 Kódolás 28](#_Toc56516563)

[3.4.4.1 Tancsics László kódja (menürendszer*+*Rólunkoldal) 28](#_Toc56516564)

[3.4.4.2 Németh Ádám kódja (Árajánlat kérés, Login, Napirend). 35](#_Toc56516565)

[3.4.4.3 Abafalvi Balázs kódja (Admin felület) 45](#_Toc56516566)

[3.5 Továbbfejlesztési lehetőségek 49](#_Toc56516567)

[3.6 Irodalomjegyzék, forrásmegjelölés 50](#_Toc56516568)

[4. Összegzés 51](#_Toc56516569)

# Bevezetés

A mai világunkban az internet az egyik legfontosabb dolog az ember életében. A távolság, mint korlát többé már nem jelent gondot. El sem tudnánk képzelni a világot internet nélkül. Emberek milliói látogatnak honlapokat nap mint nap információ keresés, kereskedelem, kapcsolattartás és szórakozás céljából. Globalizáció kialakulásának motorja lett. Nem véletlen, hogy a projektfeladatban egy webalkalmazás kifejlesztését tűztük ki célul.

Mind a fiatalok mind a felnőttek számára nélkülözhetetlen lett az okostelefon. Miért is ne lenne, hiszen egy zsebben hordozható számítógépről van szó. Segítségével bármikor és bárhol nehezebb számítás feladatokat tudunk elvégezni, internethez tudunk hozzáférni, munka ügyeket tudunk intézni, videókat nézni, akár még programozhatunk is rajtuk! A felhasználók kényelmi szempontját figyelembe véve, fontos volt számunkra, hogy az applikáció ne csak webes felületen lehessen elérni, hanem offline módon applikációként is. Tehát végső soron egy mobil applikációt fejlesztünk, de webes alapon!

# Felhasználói dokumentáció

## 2.1 Alkalmazás telepítése

Az alkalmazást alapvetőleg két módon használhatjuk.

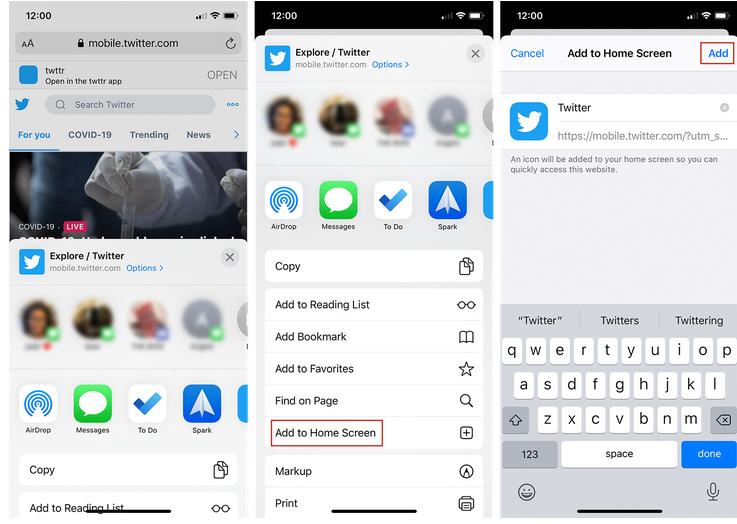
Egyik, hogy nyitunk egy böngészőt (Google Chrome, Mozilla Firefox,Opera), és be írjuk a megfelelő url címet a keresőbe.

Másik, hogy asztali applikációként nyitjuk meg. Ezt a PWA (Pogressive Web Apps) technológia segítségével tudjuk elérni. **Operációs rendszerenként eltérő!**

1. **iOS**

**Telepítése:** A telepítés csak a Safari (11.1) böngészőből működik. Tehát nyissuk meg a Safari böngészőt 🡪 Írjuk be a megfelelő URL címet. 🡪 A telefonon koppintsunk a „Megosztás” gombra 🡪 Ezután koppintsunk a „Hozzáadás a képernyőhöz” elemre🡪Végül írjuk be az alkalmazás nevét és koppintsunk a „Hozzáadás elemre”. Innentől a PWA úgy jelenik meg a képernyőn, mint egy rendes iOS asztali alkalmazás.

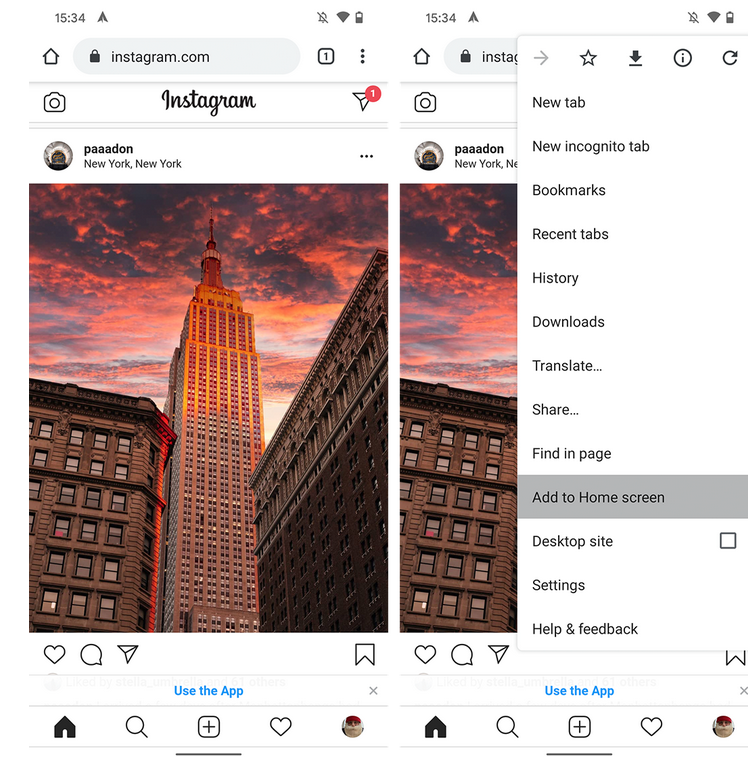
**Törlése:** Az iOS operációs rendszer automatikusan eltávolítja a telepített PWA-t, amennyiben a felhasználó egy ideje már nem használta azt. Vagy manuálisan törölhetjük az alkalmazást úgy mintha egy rendes appot törölnénk.



*PWA telepítése o 1.1 PWA telepítése iOS operációs rendszerre*

1. **Android**

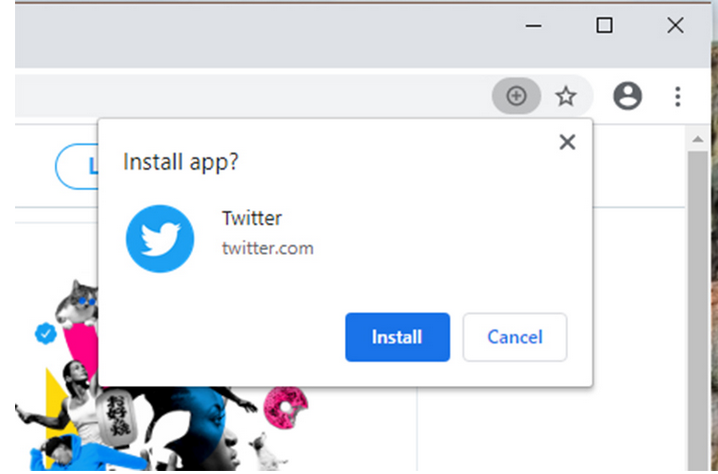
**Telepítése:** A telepítés csak Chrome (40), Firefox (44), Edge (17) böngészőből működik. Tehát nyissuk meg a fentebb említett böngészők közül valamelyiket 🡪 Írjuk be a megfelelő URL címet. 🡪 Koppintsunk a jobb felső sarokban található hárompontos overflow menüt 🡪 Ezután koppintsunk a „Hozzáadás a képernyőhöz” elemre🡪Végül írjuk be az alkalmazás nevét. Innentől a PWA úgy jelenik meg a képernyőn, mint egy rendes Android asztali alkalmazás.

**Törlése:** Manuálisan törölhetjük az alkalmazást úgy mintha egy rendes appot törölnénk.

*1.2 PWA telepítése Android operációs rendszerre*

1. **Windows / macOS**

**Telepítése:** A telepítés a Chrome (40), Edge (17), és a Safari(11.1) böngészőből működik. Tehát nyissuk meg a fentebb említett böngészők közül valamelyiket 🡪 Írjuk be a megfelelő URL címet. 🡪 Chrome esetében egy „+” szimbólum jelenik meg a címsáv jobb oldalán a könyvjelző gomb mellet, amikor olyan webhelyre látogatunk, amely támogatja a PWA-t. 🡪 Kattintsunk a gombra és telepítsük az alkalmazást. Innentől a PWA úgy jelenik meg a képernyőn, mint egy rendes asztali alkalmazás.

**Törlése:** Manuálisan törölhetjük az alkalmazást úgy mintha egy rendes appot törölnénk.

*1.3 PWA telepítése Windows / macOS operációs rendszerre*

## 2.2 Alkalmazás használata

### 2.2.1 Szerkezet

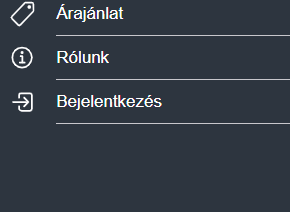
Az alkalmazás 3 user oldali és 4 admin oldali lapból áll. Minden lap más, más feladatot lát el, de szerkezetük ugyan olyan.

A lap felső részén találjuk a menürendszert + egy szöveget, hogy éppen melyik lapon állunk. A menürendszert kattintással(koppintással), valamint jobbra húzással tudjuk előhívni. Ez alatt helyezkedik el a tartalmi rész, ahol egy adott funkció megvalósítása található.

### User oldali funkciók

**Kezdőlap**

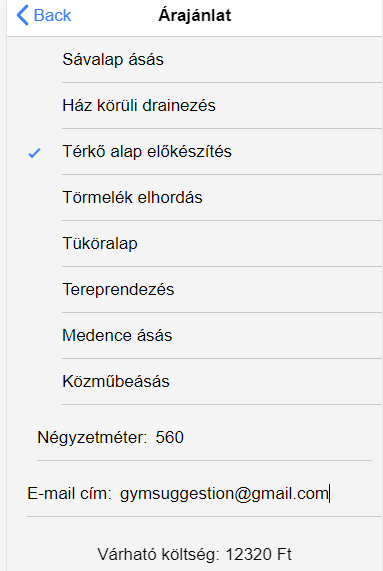
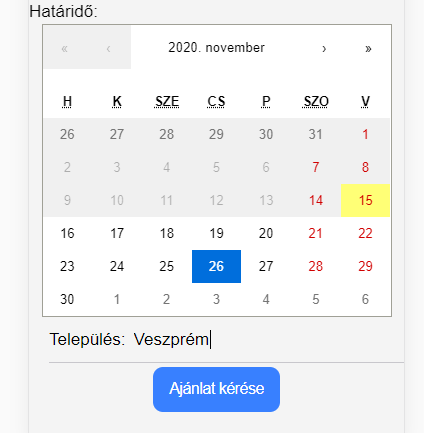
Alkalmazás elindítása után a kezdőlap jelenik meg előttünk. Ez a fő oldala az alkalmazásunknak, ami azt jelenti, hogy miután meglátogattunk egy oldalt utána mindig ide térünk majd vissza. Ezt a menürendszerben elhelyezett „Back” vissza fele nyíl megnyomásával tudjuk elérni. Tehát lényegében mindig innen fogjuk elérni az adott menürendszert. A menürendszer segítségével eltudunk jutni az árajánlatok kéréséhez, a rólunk oldalhoz, valamint az admin számára a bejelentkezés oldalra. Az oldal tartalmaz egy slidert, ami képek és némi szöveg segítségével buzdít arra minket, hogy érdemes az oldalon maradnunk.



*2. Kezdőlap+ felhasználói menü*

**Árajánlat**

Ezen az oldalon kérhetünk adott munkanemekhez árajánlatokat. Az árakat és munkanemeket a Plechinger Bau Kft írja elő. A munkanemek a következőek: Sávalap ásás, ház körüli drainezés, térkő alap előkészítés, törmelék elhordás, tüköralap, tereprendezés, medence ásás, közműbeásás.

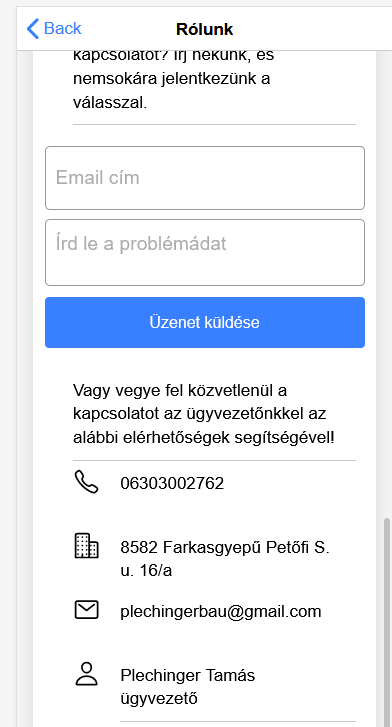
Árajánlat kéréséhez nem kell mást tennünk, mint hogy kiválasztjuk a számunkra megfelelő munkanemet, megadjuk a hozzá tartozó paramétert, és a rendszer automatikusan kiszámolja a munka díját. Abban az esetben, ha ez a díj számunkra megfelelő, akkor adjuk meg az email címünket (elérhetőség szempontjából releváns), egy határidőt (az a nap ameddig maximum várjuk a munka elvégzését Pl:. ha 2020.11.26 választjuk, akkor az azt jelenti, hogy az aznap és 26.-a közötti időpontok valamelyikében kell a munkát elvégeznie a Plechinger Bau Kft-nek), a település helyét (Veszprém, Kecskemét, Kádárta stb…), majd véglegesítsük az „Ajánlat kérése” gombbal a rendelést. Sikeres elküldés esetén „Kérés sikeresen elküldve!” felirat fog megjelenni zölddel. Hiba esetén, a hibának megfelelő hiba üzenet fog megjelenni pirossal.

*3. Árajánlat készítéséhez szükséges információk megadása*

**Rólunk**

Ez az oldal alapvető információkkal szolgál számunkra. Leírja mivel foglalkozik a cég. Kapcsolat tartási információkkal szolgál nekünk.

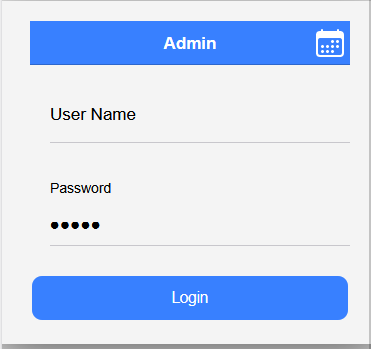
Továbbá egy űrlap segítségével további kérdéseket tehetünk fel a cégnek. Nincs más feladatunk csak meg kell adnunk az email címünket, a megfogalmazott kérdésünket majd elküldeni azokat az „Üzenet küldése” segítségével.



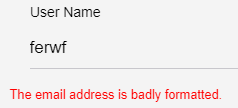
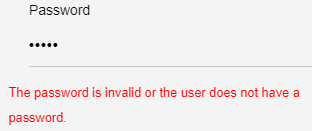
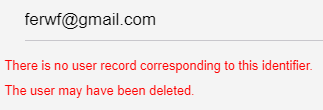
*4 Rólunk oldal*

**Bejelentkezés**

Ez az oldal az admin jogosultággal rendelkező felhasználóknak készült. A „User Name” mezőbe adjuk meg a felhasználó nevünket, a Password mezőbe pedig a jelszavunkat, végezetül nyomjunk rá a „Login” gombra. Ha nem töltjük ki a mezőket, vagy esetleg elgépelés történik, akkor egy piros hiba üzenet tájékoztat a hiba mivoltáról. Sikeres bejelentkezés esetén a ”Naptár” oldal jelenik meg, ahol a már elfogadott árajánlatok jelennek meg.



*5.1 Bejelentkező felület*

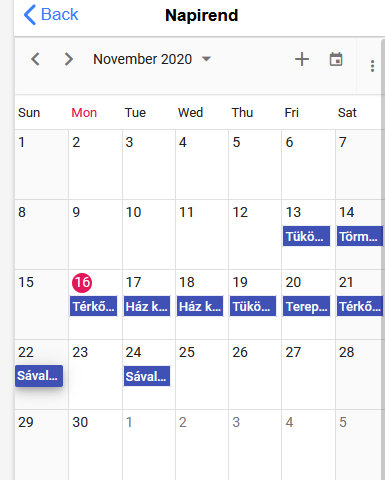
 

*5.2 Lehetséges hibaüzenetek*

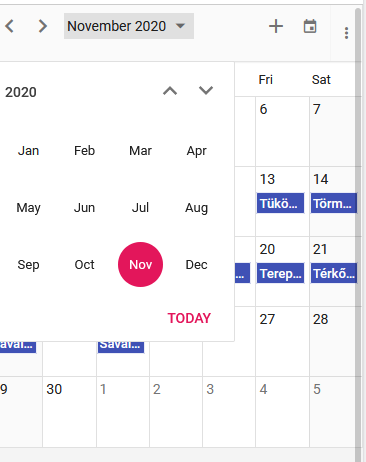
### 2.2.1.2 Admin oldali funkciók

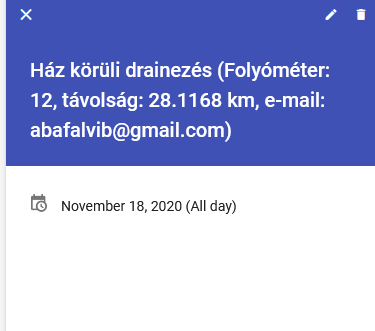
**Naptár**

Bejelentkezés után ez az oldal jelenik meg előttünk. Itt láthatjuk a már elfogadott munka ajánlatokat. Ezek rendszerint kékkel jelennek meg a naptárban az egyes napoknál. Az elfogadott munkák mindig a legelső szabad napra kerülnek (ahol még nincs munka).



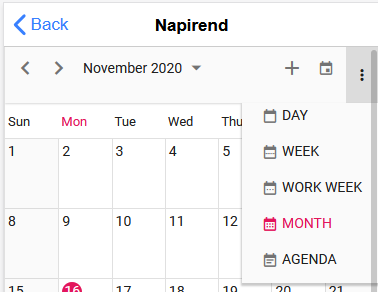
*6.1 Naptár*

Naptárban lévő kék téglalapra kattintva további információkhoz jutunk az elfogadott munkával kapcsolatban, mint pl:. munkanem neve, távolság, e-mail cím. A naptárban lépkedhetünk a napok, hónapok, évek között is egy legördülő lista segítségével.

 *6.2 Munkanem információk*

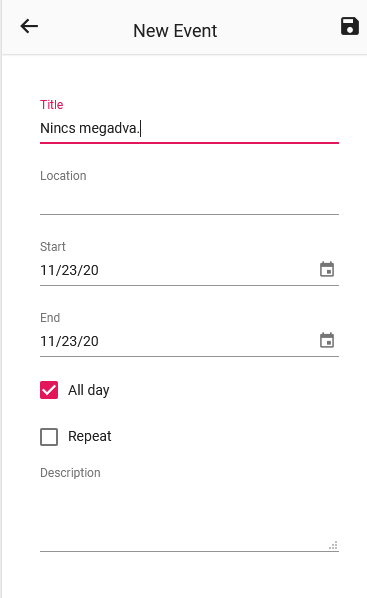
*6.3 Legördülő lista*

A három pontra kattintva kiválaszthatjuk a számunkra éppen optimális naptár nézetet. Ez lehet napi, heti, vagy egész hónapi nézet.



*6.4 Lehetséges naptár nézetek*

Az adott napra vagy fent a 3 pont melleti „+” jelre kattintva/koppintva manuálisan hozzá adhatunk újabb eseményeket. Itt megadhatjuk az esemény nevét, helyét idejét, leírását. A fenti „mentés” ikon segítségével menthetjük az eseményt



*6.5 Új esemény hozzáadása*

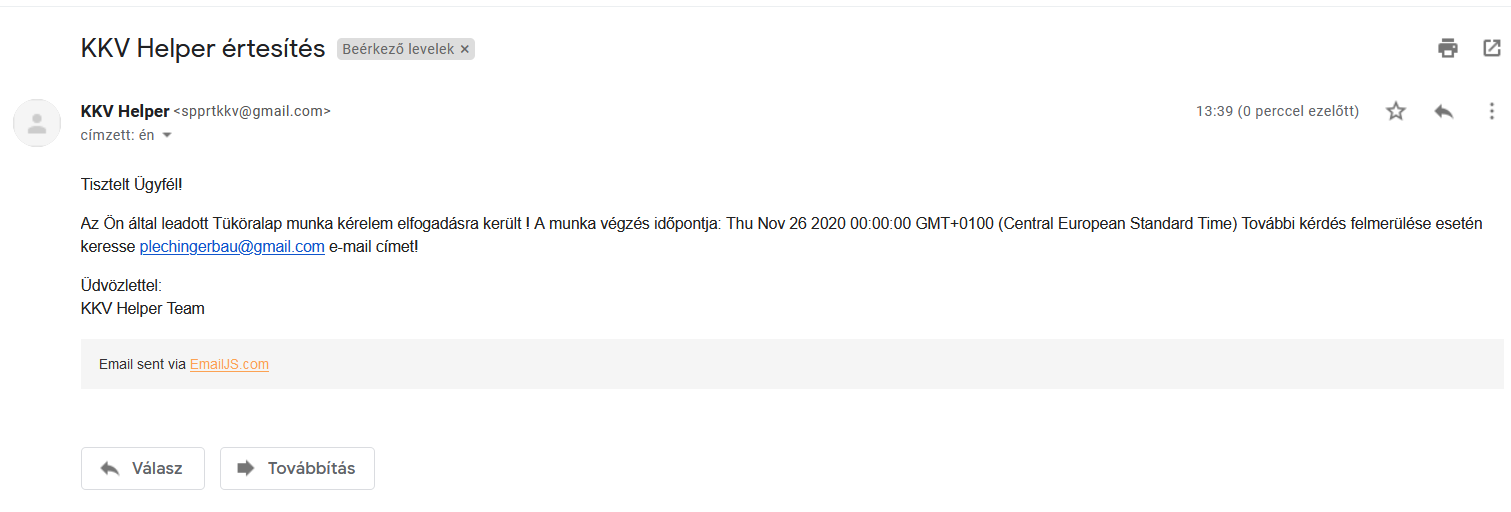
Eseményeket törölhetjük (kuka) módosíthatjuk (ceruza) jel segítségével. Törlésnél kapunk egy felugró üzenetet, hogy biztos szeretnénk-e törölni az adott eseményt. Menjünk a „Delete”-re ha szeretnénk, és a Cancel-re ha nem! Ceruza jelre kattintva, módosíthatjuk az esemény nevét, helyét idejét, leírását.

**Árajánlat kezelése**

Itt tekinthetjük meg az összes bejövő árajánlatot, amiről még nem rendelkeztünk. Továbbá tartalmaz az oldal három gombot. Ezek segítségével rendelkezhetünk a már bejött kérésekről.

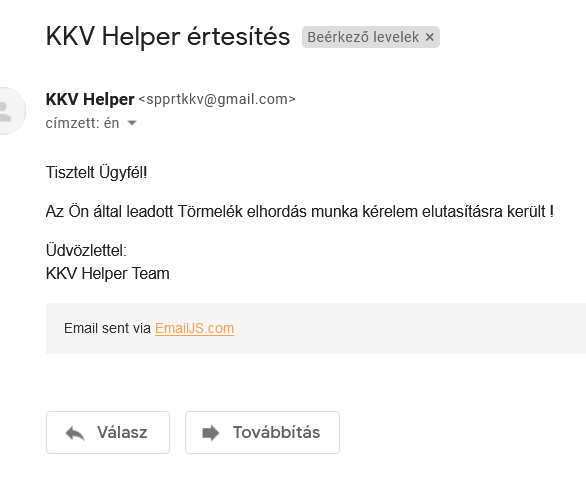
*7.1 Bejövő kérelmek*

Elfogad gomb esetén, elfogadjuk a bejövő kérést. Ekkor a munka bekerül a naptárba a „Naptár” menüpontban leírtaknak megfelelően. Valamint e-mailes tájékoztatást kapunk arról, hogy a kérelem el lett fogadva. Az email egy megadott templatet követve küldődik ki a megadott email cím-re.



*7.2 Email elfogadás esetén*

Elutasít gomb esetén, elutasítjuk a bejövő kérést. Ekkor a munka nem kerül be a Naptárba. Valamint e-mailes tájékoztatást kapunk arról, hogy a kérelmünk el lett utasítva. Az email egy megadott templatet követve küldődik ki a megadott email cím-re.

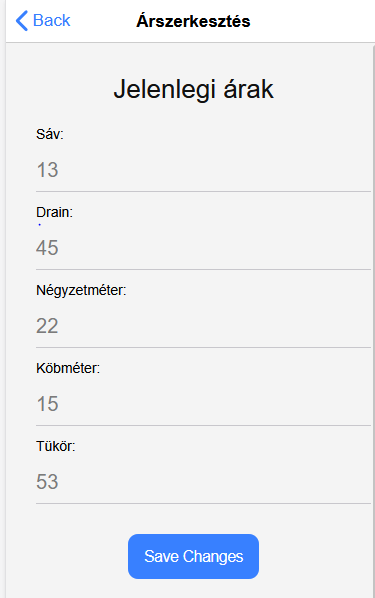


*7.3 Email elutasítás esetén*

Várakoztat gomb esetén, a bejövő kérést, a többi kérés sor végére tesszük. Ekkor a munka nem kerül be a Naptárba, de nem is törlődik. Erről a kérésről a továbbiakban, ugyanúgy rendelkezhetünk, mint az újakról.

**Árszerkesztése**

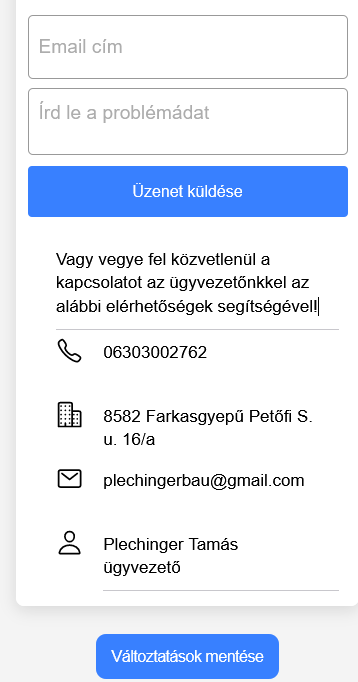
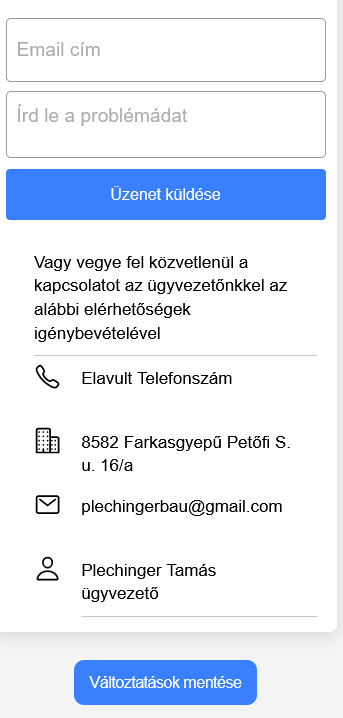
Az árak különböző változók segítségével jönnek létre. Ezeket a változókat admin oldalról is képesek vagyunk manipulálni. Tehát ha csökkenteni vagy növelni szeretnénk az árakat, akkor nincs más teendőnk, mint belekattintunk a változtatni kívánt mezőbe. Beleírjuk az új árat, majd véglegesítjük a döntésünket a „Save Changes” gombbal.



*8. Árszerkesztés*

**Rólunk**

Az oldal megegyezik a user oldalon található „Rólunk” oldallal, annyi különbséggel, hogy itt az egyes szöveges részek szerkeszthetőek. Kattintsunk bele egy szöveges részbe írjuk át úgy ahogy szeretnénk majd menjünk a „Változtatások mentése” gombra.



*9.1 Szerkesztés előtt 9.2 Szerkesztés után*

**Kijelentkezés**

Erre a menüpontra kattintva kijelentkezünk a fiókunkból. Kijelentkezés után a „Bejelentkezés” felület fog megnyitódni számunkra.

# 3. Fejlesztői dokumentáció

## 3.1 Feladat leírása

A projekt során egy webes alkalmazás fejlesztése a cél, amely megkönnyíti, valamint hatékonyabbá teszi a kis- és középvállalkozások (KKV) számára az egyes szolgáltatások használatát

Az alkalmazás segítségével lehetőség nyílik a felhasználók számára az árajánlatok kérése időponttal együtt. Az admin felhasználó pedig ezeket a kéréseket el tudja fogadni, utasítani, esetlegesen „talonba” helyezni. Továbbá lehetőség nyílik admin oldalról a statikus oldalak tartalmának szerkesztése.

Manapság a BYOD (Bring Your Own Device) trend hatalmas népszerűségnek örvend. Mindenki a saját eszközein próbálja intézni az ügyeit. Ezen eszközök közül a legnépszerűbb kétségtelenül az okostelefon, hiszen könnyen hordozható, és szinte mindent meglehet vele csinálni, amit egy asztali számítógéppel is. Így nem is volt kérdés, hogy ennek a webalkalmazásnak mobilapplikációként is működnie kell.

## 3.2 Az alkalmazott fejlesztői eszközök

**Fejlesztői környezet**

* Atom Editor

**Programozási nyelvek**

* HTML5
* CSS
* Javascript

**Adatbázis nyelv**

* JSON (NOSQL)

**Adatbázis kezelő rendszer**

* Firebase

**Javascript könyvtár**

* React

**Keretrendszer**

* Ionic

**Szövegszerkesztő**

* Word

**Képszerkesztő**

* GIMP 2

**További web technológia**

* Progressive Web Apps (PWA)

### 3.2.1 React

A React egy Facebook által kifejlesztett nyílt forráskódú JavaScript könyvtár, amit felhasználói felületek programozására használunk. Az MVC (Model View Controller) programramtervezési mintából a React a „V-t” képezi. Tehát csak React-et használva nem lehet web alkalmazást fejleszteni!

Komponensekből tevődik össze. Ezek JavaScript függvények, tetszőleges számú inputtal és egy outputtal. JSX (JavaScript XML) szintaxissal íródnak. A komponensek fejlesztése ES6 szabvány szerint történik, amely a következőket határozza meg:

A komponensek tetszőleges számú bemenetet fogadhatnak és ezeket az úgynevezett „props” nevű objektumba tárolják.

Mindig kell lenni-e egy „render” metódusnak. Ez a metódus határozza meg, hogy mi legyen megjelenítve, amikor a komponens meghívódik.

Minden komponens egy belső állapotot menedzsel. Amikor az állapot változik akkor az adott komponens „render” függvénye meghívódik.

React könyvtár segítségével íródott alkalmazások teljesítménye kivételesen magas. Ennek az oka nagyon egyszerű. A render-elés Virutal DOM (Document Object Model) segítségével történik. Tegyük fel, hogy sok kis „View” változást szeretnénk eszközölni. Ezek a változások egyfolytában frissítik a DOM-ot🡪alkalmazás teljesítménye romlik. React megoldása: Ezek a „view” módosítások tárolódjanak el a VDOM-ban. Ezután jön egy algoritmus, amely először összehasonlítja a a Virtual DOM jelenlegi és előző állapotát, majd végrehajtja azt a minimum számú módosítást az eredeti DOM-on, ami ténylegesen szükséges, a kívánt állapot eléréséhez. Felhasználói interakciónál is ez történik!

### 3.2.2 Ionic

Az Ionic egy nyílt forráskódú keretrendszer, amit az Ionic cég fejlesztett ki. Segítségével hibrid mobilalkalmazásokat fejleszthetünk. Angularra fejlesztették ki először. 2019-ben oldották meg, hogy React és Vue-val is tudjon együtt dolgozni. Használatához npm modul és Node.js szükséges.

Miután Ionic segítségével lefejlesztettünk egy alkalmazást, azt natív alkalmazásboltba terjeszthetjük (App Store/Google Play). Az alkalmazásboltból Cordova (Android) vagy Capacitor (iOS) segítségével telepíthetjük saját mobilunkra.

Tartalmazza a legújabb web technológiát az úgynevezett „PWA” (Progressive Web Apps)-t. A PWA lehetővé teszi a felhasználók számára, hogy a webalkalmazás native applikációként működjön. Tehát az alkalmazás ikonja (mint a rendes appoknál) kint lesz az asztalon. Elindításkor nem egy web-es felület nyílik meg, hanem egy applikációs felület. Továbbá a webalkalmazás hozzáférhet a telefon különböző eszközeihez, mint például a kamerához, szenzorokhoz. Teszi mind ezt úgy, hogy rajta sincs fizikailag a tárhelyünkön, hanem a böngészőben könyvjelzőként van elmentve.

Az Ionic által támogatott platformok: Windows (10), Andorid (4.4-től), iOS (10-től).

## 3.3 Adatmodell leírása

### 3.3.1 Firebase

A Firebase egy szolgáltatáscsomag a Google-től. Segítségével különböző backend feladatok oldhatóak meg könnyedén. Ilyen például a felhasználók beléptetése akár Google vagy Facebook azonosítással, adatok felhős tárolása, vagy éppen lehetőséget nyújt a felhasználók számára, hogy kedvencek közé rakhassanak tartalmakat.

Firebase szolgáltatásai nem csak a fejlesztésre koncentrálnak, számos lehetőséget nyújt arra, hogy mobil applikációnk mérhető növekedést érjen el:

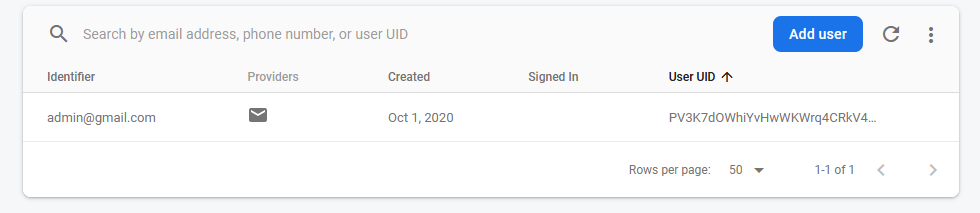
Mérhetők/elemezhetők a felhasználók interakciói

Az appunkat aktívabban használó közönség felé kommunikálhatunk új alkalmazás verziók kiadása nélkül.

Fizetett kampányokkal szerezhetünk további felhasználókat

Ez egy „Realtime Database”, tehát az adatok szinkronizálása valós időben történik. NoSQL felhőalapú adatbázist használ. Itt tárolódnak az adatok. JSON alapú, így könnyen belátható, hogy egyszerű adatszerkezetek tárolására alkalmas ez a rendszer. Használjunk SQL alapú rendszert bonyolultabb adatszerkezetek tárolása esetén!

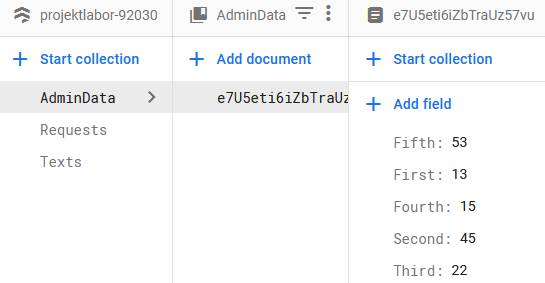
### 3.3.2 Adatbázis felépítése

 A projekt során két alapvető authorizációs adatot tárolunk, amelyek a következők: Username (Email cím), jelszó. Mivel nem kötelezzük a felhasználókat, hogy az alkalmazás funkcióinak használatához regisztrálniuk kelljen ezért, csak egy admin felhasználó van beleégetve a rendszerbe. A beleégetést az Add user gombra rámenve lehet elvégezni. A jelszó automatikusan, titkosítva kerül elmentésre (User UID).

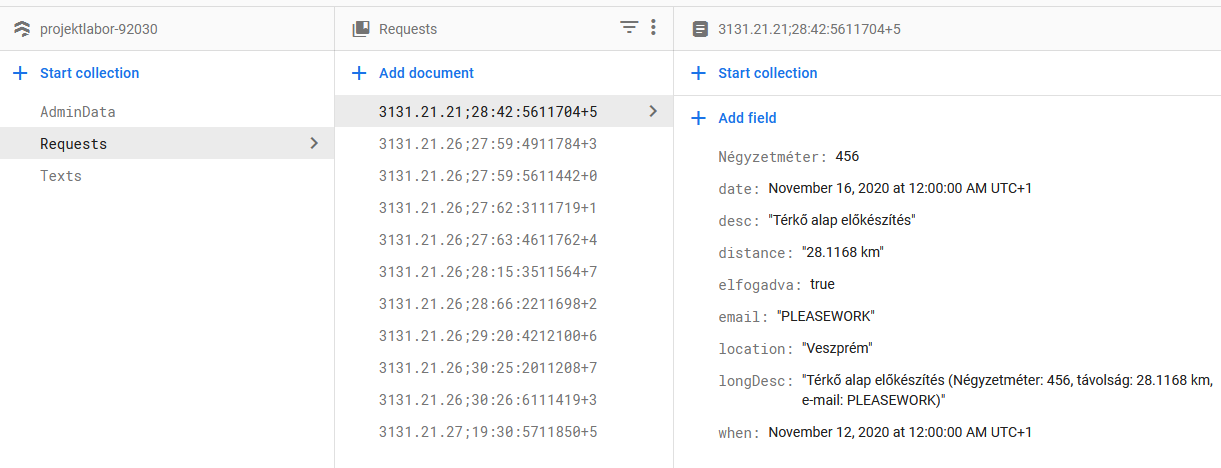
*1.1 Admin felhasználó hozzáadása GUI segítségével*

További három collection:

1. AdminData: Itt tároljuk az árképzéshez tartozó konstans értékeket.
   * First: Sávköltség (Sávalap ásás munkához)
   * Second: Drainköltség (Ház körüli drainezés munkához)
   * Third: Négyzetméter-költség (Térkő alap előkészítés munkához)
   * Fourth: Köbméter-költség (Törmelék elhordás munkához)
   * Fifth: Tükörköltség (Tüköralap munkához)



*1.2 Collection AdminData*

1. Requests: Itt tároljuk a bejövő munkák paramétereit:
   * desc: A Munka megnevezése
   * munkához tartozó mértékegységbeli adatok (pl:. Folyóméter,Kanálméret)
   * date: A Munka határideje
   * distance: A munka helyének és a telephely közötti távolság
   * elfogadva: Küldéskor ez a mező mindig hamis (admin változtatja majd át, ha elfogadja a munkát).
   * email: A Munkakérő email címe (egyedi azonosítást tesz lehetővé).
   * location: Munka helye (ahol a munka fog történni)
   * longDesc: Naptárban megjelenő összesített leírás
   * when: A már elfogadott munkánál a munka végzés mikor fog történni

*1.3 Collection Requests*

1. Texts: Itt tároljuk a bejövő munkák paramétereit:
   * About: A rólunk rész szövege
   * Contact: A kapcsolat rész első szövege
   * Or: Kapcsolat rész második szövege
   * Telefon: Telefonszám szöveg
   * Address: cím szöveg
   * Email: Email cím szöveg
   * Name: Név szöveg

## 3.4 Munkaterv

### 3.4.1 Feladat és egyéb tudnivalók átbeszélése

Mint minden termék előállításánál, itt is az első lépés a feladat értelmezése volt. A témavezető átfogóan elmondta, hogy mit kell megvalósítanunk és hogy azt milyen technológiával kell megtennünk. Ezután tisztázásra kerültek a határidős kérdések. Végezetül a konzultációs időpontok kialakítására helyeződött a hangsúly. Mindenki egyetértett, hogy a lendületes szép munkát csak akkor tudjuk fenntartani, ha legalább heti egy alkalommal „jelentünk” a témavezetőnek.

### 3.4.2 Ismerkedés a technológiákkal

Mielőtt egyből fejest ugrottunk volna a projekt közepébe, elkezdtünk megismerkedni a technológiákkal. Szerencsére számtalan jobbnál-jobb anyag található fent a világhálón.

Segítségünkre voltak:

* A technológiák hivatalos oldalai
* Fejlesztők, fejlesztői csoportok által készített Youtube videók.
* Nyilvánosan publikált anyagok.

Folyamat alatt megtanultuk a technológiák:

* Alapvető ismérveit.
* Előnyeit, hátrányait.
* Kódolási konvenciáit.
* A projekt generálásának lépéseit.

A folyamat végére, mindenki elkészítette a saját kis gyakorló programját.

### 3.4.3 Funkcionális követelmények

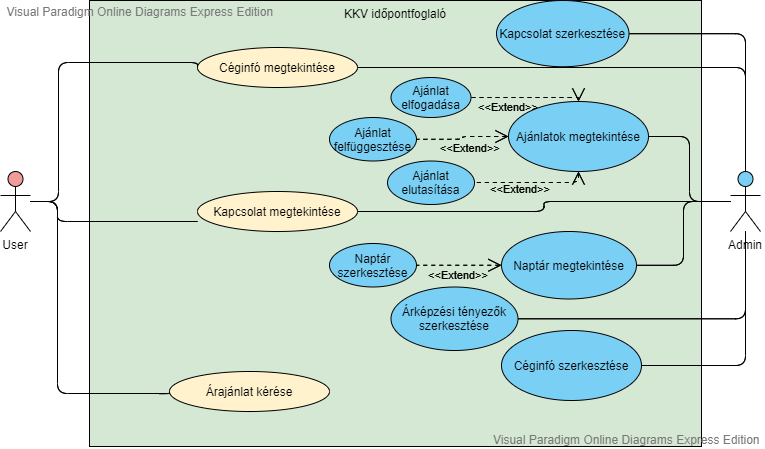
Elérkeztünk az alkalmazás fejlesztés egyik legfontosabb lépéshez. A témavezetőnk összeszedte az összes megvalósítandó funkcionális követelményt, amelyek a következőek:

1. Felhasználói oldalon:

* **Kezdőlap:** Miről szól ez az alkalmazás? Milyen lehetőségeket kínál a felhasználóknak az alkalmazás? Ezekre a kérdésekre kell választ adnunk.
* **Rólunk:** Egy hosszabb leírás, hogy mivel foglalkozunk. Alapvető elérhetőségi információk megadása (email, telefonszám, cím). Továbbá biztosítani a felhasználók számára, ha további kérdésünk van, akkor közvetlenül az oldalon email formájában fel tudják azokat tenni.
* **Árajánlat kérés:** Cél, hogy az ügyfelek alapvető paraméterek megadásával, árajánlatot kapjanak, hogy az adott munkanemet a cégünk mekkora ellenérték fejében fogja elvégezni. Ha az ügyfél elégedett a kapott árral, akkor tudjon leadni rendelést a munkával kapcsolatban.
* **Bejelentkezés:** Admin számára bejelentkezés lehetősége

1. Admin oldalon:

* **Kezdőlap:** Ugyanaz. mint a felhasználói oldalon.
* **Rólunk:** Ugyanaz. mint a felhasználói oldalon, de az oldal tartalmát lehessen szerkeszteni.
* **Árajánlat kezelése:** Itt tekinthetőek meg az ügyfelek által leadott kérések. A kérések mellet legyen mindig megjelenítve a telephely és a munkahely közötti távolság! Az admin tudjon rendelkezni a kérésekről. Három féle rendelkezést különböztetünk meg:
  + - * *Kérés elfogadása:* Ekkor, a kérés paraméterei között megadott határidő alapján, kerüljön be egy naptárba az adott ügyfél által leadott munka kérelem. Kapjon a felhasználó e-mailes tájékoztatást a munka elfogadásáról.
      * *Kérés elutasítása:* Kérés kitörlődik az adatbázisból. Kapjon a felhasználó e-mailes tájékoztatást a munka elutasításáról.
      * *Kérésről később rendelkezés lehetősége:* Nem történik semmi, csak a következő kérésre ugrik.
* **Napirend:** Egy naptár jelenik meg ezen az oldalon. Ez a naptár tartalmazza a már elfogadott munkákat. Lehessen a **naptárat szerkeszteni**, olyan módon, hogy már az elfogadott munkákat lehessen visszamondani (törölni), vagy módosítani.
* **Árak szerkesztése:** Az admin tudja az árképzési változókat az alkalmazásból módosítani.
* **Kijelentkezés:** Admin számára kijelentkezés lehetősége.

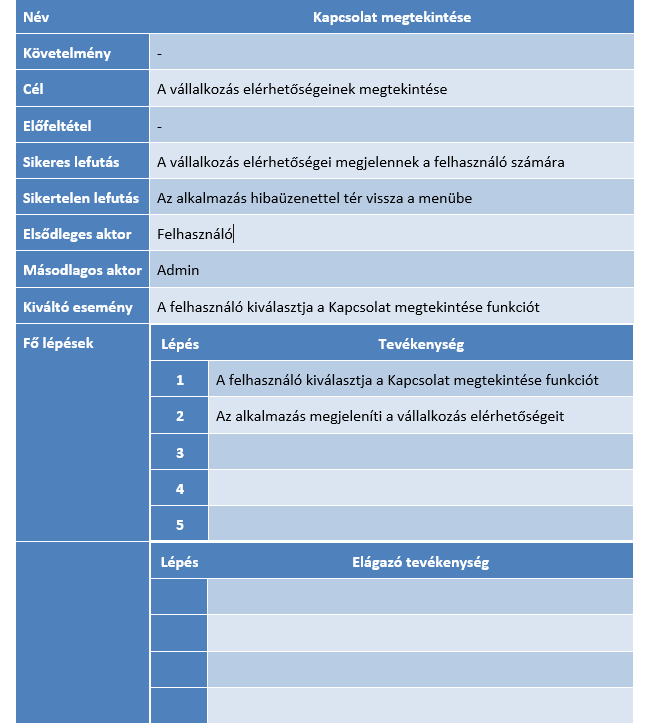
A funkcionális követelmények jobb megértése érdekében Use Case diagramot készítettünk:

*2.1 A rendszer Use Case diagrammja*

A Use Case diagramm lényege, hogy reprezentálja a felhasználói interakciókat a rendszerrel. Ebben az esetben két felhasználót (aktort) különböztetünk meg: admin, és a hagyományos felhasználó. A diagrammon ezek a pálcika emberek. Az aktorokhoz különböző Use Casek (funkciók) tartoznak. A diagrammon ezek az ovális körök. Az aktorok és Use Case-k között futó élek a kapcsolatok.

Diagrammon jól láthatóak az egyes funkciók hovatartozásai. Továbbá látható, hogy nagyon sok funkció az „extended” jelzőt kapta. Ez annyit jelent, hogy azok a funkciók nem mindig futnak le. Érthető hiszen minek terheljük fölöslegesen az alkalmazásunkat azzal, hogy rengeteg funkció fut mikor, könnyen szűrhetőek megfelelő feltételekkel.

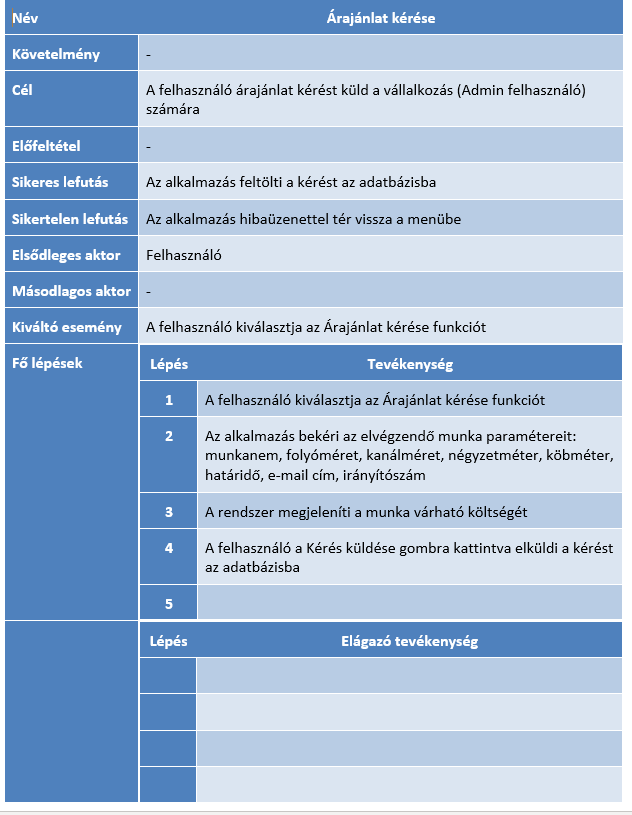
Vizsgáljuk meg néhány funkciót kicsit részletesebben. Ezt az ügynevezett Use Case tábla segítségével tehetjük meg. Ezek a táblák lehetővé teszik számunkra, hogy az egyes funkciókba részletesebb belátást nyerjünk. Kezdjük a legkönnyebbel a „Kapcsolat megtekintése”-vel



*2.2 A „Kapcsolat” Use Case táblája*

Ahogy arra kitértünk még a legelején, ez egy hagyományon statikus oldal lesz. Egy menürendszer segítségével a felhasználó kitudja majd választani ezt a funkciót és be fog töltődni a megfelelő tartalom az oldalra. Ugyan ez a funkció természetesen admin oldalról is elérhető lesz majd egy kis plusszal megfűszerezve.

Nézzünk meg egy kicsit izgalmasabbat: „Árajánlat kérése”.



*2.3 A „Árajánlat kérés” Use Case táblája*

Szituáció: A programnak le kell tudni kezelni-e a kérést, ami a felhasználótól jön. Elöször is a kérést valamilyen formában tárolni kell, különben a kérések elvesznének a semmiben. Erre egy korszerű és egyben egyszerű megoldás az adatbázis. Mi a firebase adatbázis szolgáltatásait vesszük igénybe. Ha a felhasználó nem tölti ki a kötelezően kitöltendő mezőket, akkor az árajánlat kérés meghiúsul, egy hibaüzenet szólítja fel a felhasználót a hiányosságok pótlására. Ha mindent kitöltött a felhasználó, akkor nincs más dolga, mint leadni a kérést. Ekkor a kérés az adatbázis-ba kerül és admin oldalon elérhető lesz.

### 3.4.4 Kódolás

A mai modern alkalmazás fejlesztésénél a programozó felek felosztják egymást között a feladatokat. Ezáltal időt, energiát takarítanak meg. Ráadásul a fejlesztési folyamat is sokkal lendületesebb lesz. De felvetül ilyenkor egy nagyon fontos kérdés: Hogy fejlesszünk úgy, hogy kódjaink mindig szinkronba legyenek? Ne kelljen minden nap egymásnak küldözgetni az egyre növekvő alkalmazást. A válasz egyszerű. Használjunk ilyen fajta szoftvert, ami támogatja ezt a fajta fejlesztést. Az egyik legismertebb ilyen eszköz a GitHub, amit mi is használunk. Használata egyszerű. Létre kell hozni egy csoportot, amibe meg kell hívni a projektben szereplő embereket. Töltsük le a GitHub asztali alkalmazását. Nyissuk meg. Ezután file🡪 local repository. Itt beírjuk a csoport nevét és készen is vagyunk. Az adott alkalmazást, amit ide felrakunk (commit) bárki pillanatok alatt leszedheti (pull) magának. Nem is letöltésről van itt szó, hanem szinkronizálásról. A meglévő projekted kódját összehasonlítja az utoljára commit-elt kóddal, és csak a változásokat eszközöli a saját kódodban.

### 3.4.4.1 Tancsics László kódja (menürendszer*+*Rólunkoldal)

**Menü**

A menürendszer kialakítása egy alkalmazásban mindig kényes ügy. Mielőtt neki kezdünk össze rakni el kell döntenünk, hogy „single page” vagy „multiple page” webalkalmazást szeretnénk fejleszteni. Mi a multi page mellet döntöttünk. Tehát több lapból álló webalkalmazás menürendszerét kellet kialakítani.

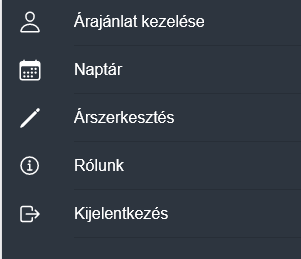
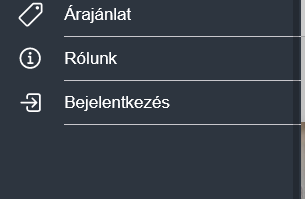
Felmerülő kérdések:

* Hogy csináljuk, hogy platform független legyen?
* Hol legyen az elhelyezkedése?
* Hogy csináljuk meg igényesre?
* Hogyan oldjuk meg az átirányításokat?

Két szó: Ionic + React. Igen egy frameworkel és egy könyvtárral nagyon egyszerűen, és gyorsan el lehet ezt érni. Lássuk hogyan!

A mi menünk egy slide menü lett. Bal oldalon helyezkedik el hamburger ikonként. A menü előhívása kétféleképpen történhet, mind mobilon, mind asztali számítógépen: Kattintással (Koppintással) vagy jobbra húzással.

Alapvetőleg 2 féle menürendszert különböztetünk meg: User és Admin oldali menürendszert.



*3.1 User oldali menü 3.2 Admin oldali menü*

A menü kinézetéért az Ionic Framework a felelős. Ionic számtalan olyan előre megformázott tagekkel (IonMenu, IonMenuToggle, IonItem, IonSplitPane) rendelkezik, amivel gyorsan össze lehet hozni egy jól kinéző menürendszert. De egy jól kinéző menürendszer semmit se ér, ha nem tudja elvégezni a neki szánt feladatokat. Ez nem más, mint az irányítás.

Az irányítás elvégzésért felelős a React könyvtár. 2 lépésre van csak szükségünk:

1. lépés: Vegyük fel a gyökér fájlunk-ba (App.tsx) az összes utat (route), amit használni szeretnék (kép mutatja).



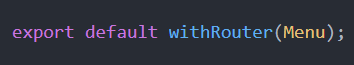
*3.3 Az összes route*

2. lépés: Egy onClick eseményt fűzünk minden egyes menüponthoz (IonItem). Majd megadjuk, hogy melyik oldal „renderelődjön”(history.push) le ha az adott IonItem-re rákattintunk. Ez biztosítja számunkra a lapok közti irányítást. Mivel mindig a megfelelő oldal renderelődik le ezért sosincs oldal frissítés, akkor mikor az egyik menüpontról a másikra ugrunk!



*3.4 Menü szerkezete+onClick esemény*

3.lépés: A react-router-dom npm modul által biztosított withRouter komponensel burkoljuk be az egész menürendszerünket.



*3.5 withRouter segítségével élesítjük a menürendszert*

**Cookie-k**

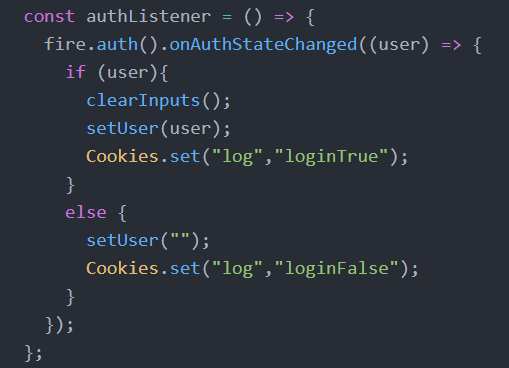
Már csak egy kérdéses rész maradt számunkra. Hogyan különítjük el a user és az admin oldalt? A webes technológia nagyon sok opciót ad arra, hogy hogyan kezeljük a bejelentkezett felhasználókat. Pl:. PHP:Session ID, JSON alapú rendszereknél JSON WebToken. Esetünkben csak egy felhasználót különböztetünk meg a többitől, ezért számunkra elegendő, ha ezt eltároljuk valahol. Nyílván egy olyan tárolót kellet keresnünk, amit dinamikusan, a rendszer futása közben állíthatunk + esetleges oldal frissítéseknél értéke nem veszik el. Erre a problémára egy könnyű megoldás a cookie-k használata. Fontos megjegyezni, hogy csak a bejelentkezett user-nek (esetünkben az egyetlen adminnak) követeljük meg a cookie-k használatát.

Telepítsük a cookie-js npm modul. Majd importáljuk be a projektunkbe.



*3.6 Cookies import*

Ezután a set metódus segítségével beállítunk egy változónak egy értéket. Ezt akkor tesszük meg, amikor a bejelentkezés sikeres volt. Innentől kezdve bárhonnan és bármikor elérhető ez a változó. Elérését a get metódus biztosítja.



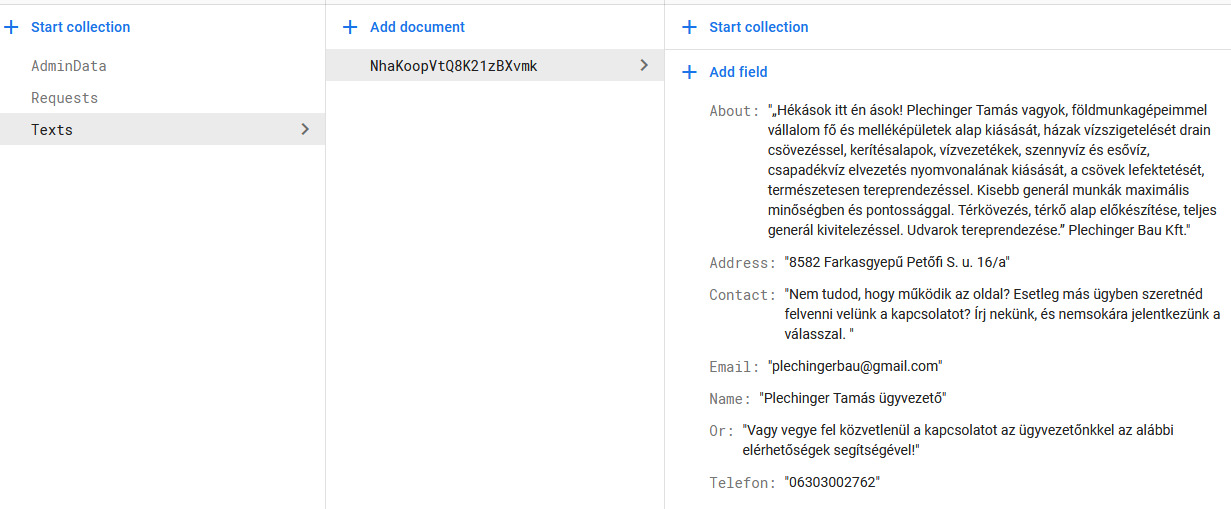
*3.7 A log változó értékét loginTrue értékre változtatjuk*

*a set metódus segítségével*

**Rólunk**

Az oldal tematikája viszonylag egyszerű. Egy egyszerűbb űrlapból és szövegből áll.

Az oldal szövegeit „textarea” elemekbe töltjük be dinamikusan adatbázisból. Erre azért van

szükség, hogy az admin az oldalról is tudja szerkeszteni a tartalmat ne csak adatbázisból.

*4.1 Az adatbázisban tárolt szövegek*

Ahhoz, hogy ezt a szöveget oldalon meg tudjuk jeleníteni és dinamikusan tudjuk változtatni, két dologra van szükségünk.

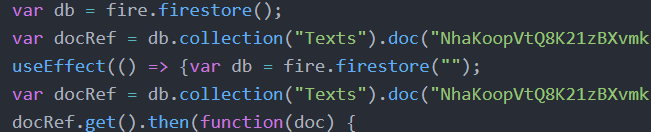
* + - 1. React Hook-okra: Tárolók, amiknek két paraméterük van: érték, és egy függvény, amivel az értéket beállíthatjuk (felfogható egy 2 paraméteres változónak is). Ezek segítségével fogjuk tudni dinamikusan (az alkalmazás futása alatt) szerkeszteni a szövegeinket.
      2. Adatbáisra: Tárolás szempontjából, hogy a változtatások oldal frissítéskor ne tűnjenek el. Alkalmazásunk esetében a firebase-t használjuk. Szövegek feltöltése és letöltése az adatbázisból firebase kódok alapján történik.

Továbbiakban egy konkrét megoldást fogunk illusztrálni képek segítségével (rólunk rész szöveges feltöltése).

1. lépés: React Hooks segítségével deklarálunk egy változót. Esetünkben ez a „text” változó.

*4.2.1 1.lépés*

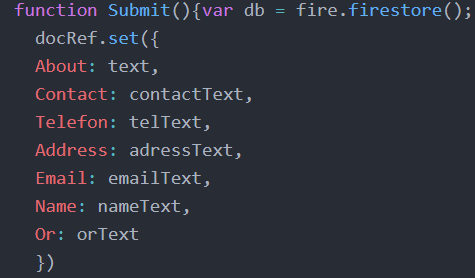
1. lépés: Lekérjük az adatbázisban tárolt megfelelő szövegünket a *doc.get()* metódus segítségével, és a *setText()* metódussal a „text” változó értékét az adatbázisból lekért szöveg értékére állítjuk.

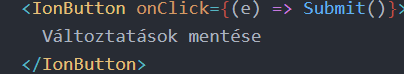


*4.2.2 2.lépés*

1.  lépés: A textarea elem érétékét egyenlővé tesszük a text értékével. Ekkor a textarea tartalma mindig a text változó által aktuálisan tartalmazott értéké lesz. Valamint az onIonChange eseménykezelő (ami az adott objektumon be következő változásokat kezeli), és a React Hook-ban megadott *setText()* metódus segítségével, szerkeszthetjük a text változó aktuális értékét.

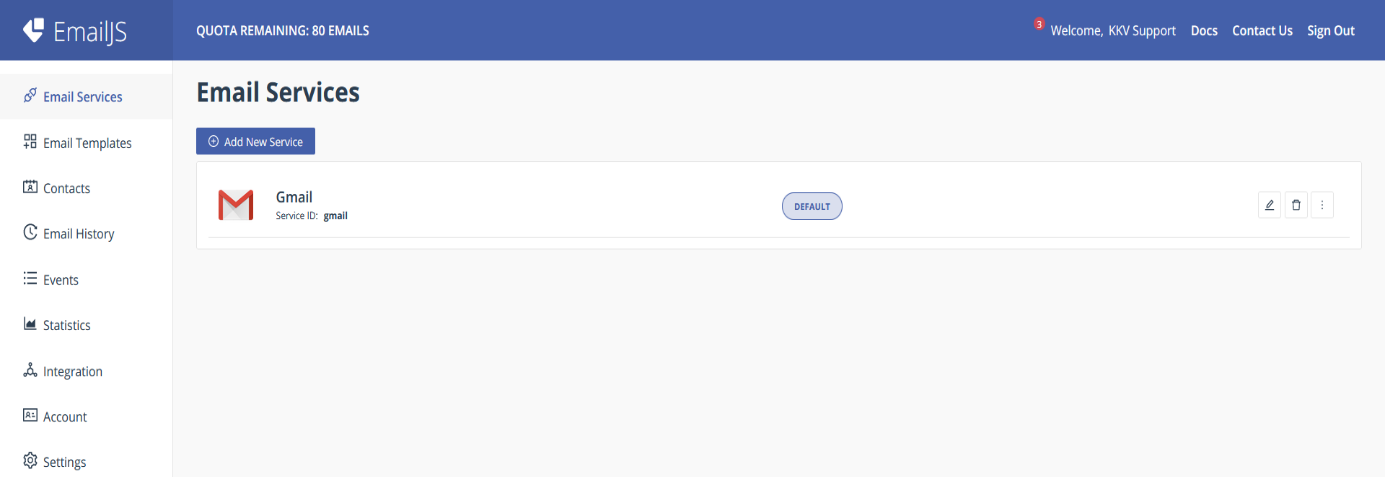
*4.2.3 3.lépés*

1. lépés: Az onClick eseménykezelő (ami az adott objektumon be következő kattintásokat kezeli) segítségével a változtatásokat elküldjük (*Submit()*) az adatbázisnak.



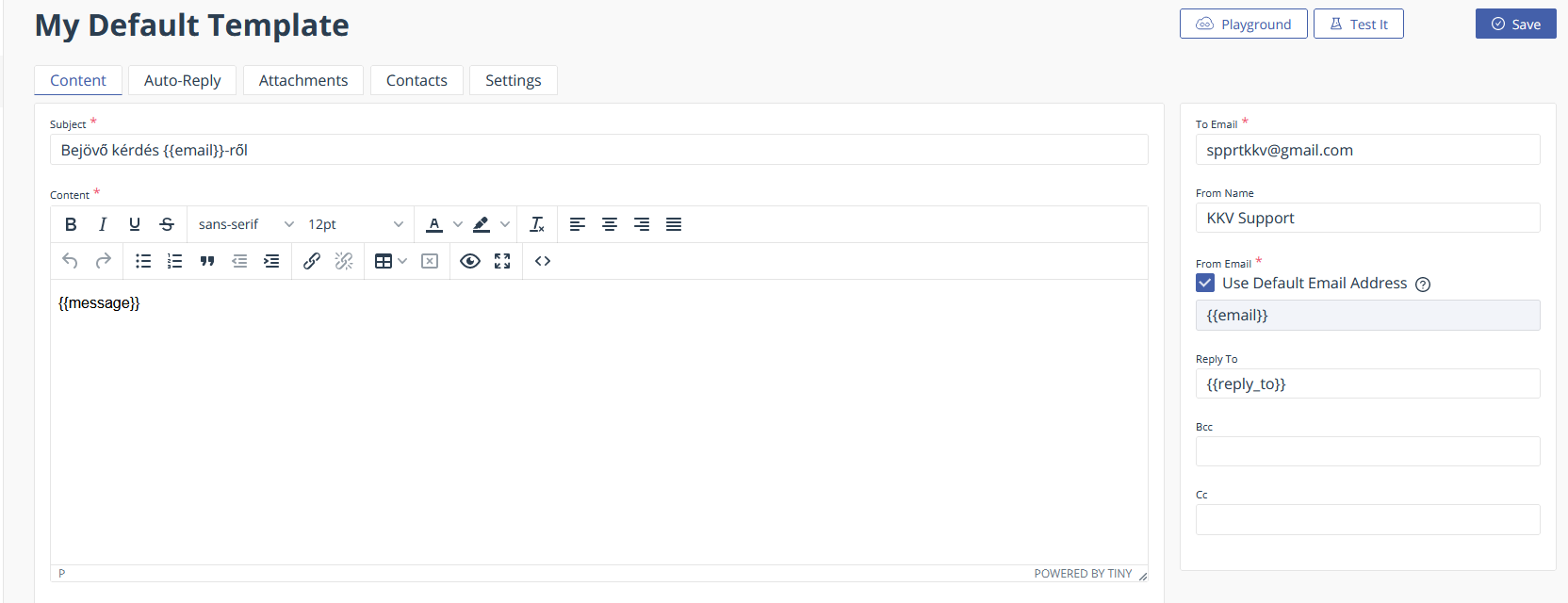
*4.2.4 4.lépés*

**EmailJS**

A levelezések megvalósításához használjuk. Ez egy fizetős szolgáltatás. De van ingyenes verziója is. Az ingyenes verzió ad nekünk egy email kiszolgálót (ami lehet gmail, freemail, hotmail, outlook, yahoo, stb…), kettő darab email templatet (sajátmagunkra szabhatjuk), valamint havonta 200 ingyenesen küldhető emailt.

*4.3 EmailJS oldala bejelentkezés után*

A „Rólunk” oldalon található űrlap kitöltése után ha rákattintunk az „Üzenet küldése” gombra, akkor a képen látható template küldődik ki a [spprtkkv@gmail.com](mailto:spprtkkv@gmail.com) e-mail címre.



*4.4 Email template*

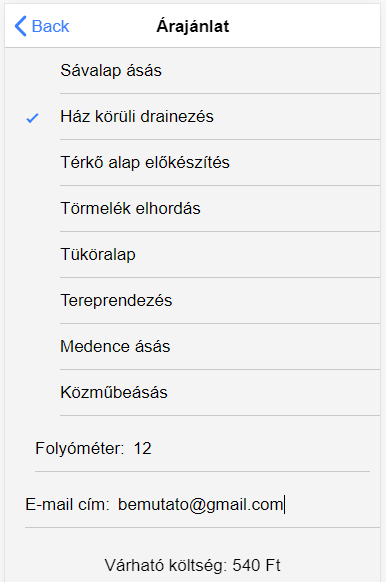
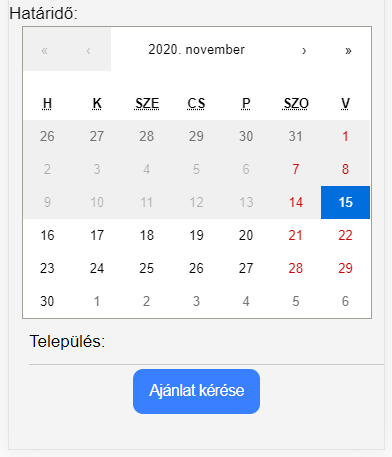
Kódbeli megvalósítása nagyon egyszerű. Szükségünk van egy form-ra és egy metódusra, amit az EmialJS ad számunkra. Ebben a metódusban meg kell adni a kiszolgáló nevét, a template nevét, a template paramétereit (jelen esetben a messaget és az email címet), valamint a userID-t

*4.5 EmailJS metódusa*

### 3.4.4.2 Németh Ádám kódja (Árajánlat kérés, Login, Napirend).

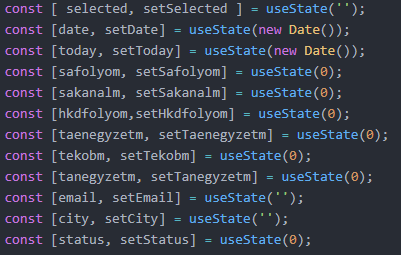
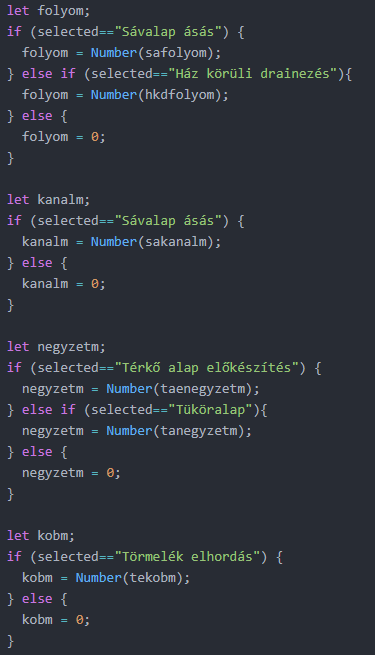
**Árajánlat kérése**

Az árajánlat kérése menü a választható munkanemeket felsoroló rádiógomb-csoportból, az adott munkakörhöz tartozó paramétereket, valamint egy-egy, a felhasználó e-mail címét és az elvégzendő munka települését bekérő beviteli mezőből, egy dátumválasztó naptárból és egy gombból áll, mellyel a felhasználó elküldi az árajánlat kérést az adatbázisba. Ezen kívül látható egy szöveges mező is, amely a felhasználó által megadott paraméterek és az adminisztrátor által megadott konstans értékek alapján megjelenít egy számolt várható költséget is az adott munkához.



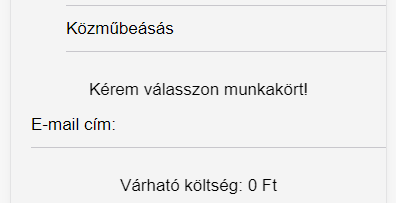
*4.1 Árajánlat kérése menü*

A bekért paramétereket az alkalmazás állapotokban tárolja el. Mivel bizonyos paraméterek több munkakörnél is szerepelnek, ezért elkülönítésük érdekében minden egyes munkakör minden paraméterét külön állapotban tároljuk el. Ezeket később az adatbázisba töltés és a várható költség számolásának könnyítése érdekében egyszerűbb elnevezésű változókba töltjük a kiválasztott munkanem alapján.



*4.2 A paramétereket tároló állapotok és változók*

Az oldalon mindig csak a kiválaszott munkaághoz tartozó paraméterek beviteli mezői jelennek meg dinamikusan, amíg egyik sincs kiválasztva, addig a „Kérem válasszon munkakört!” felirat látható a felhasználó számára.

**

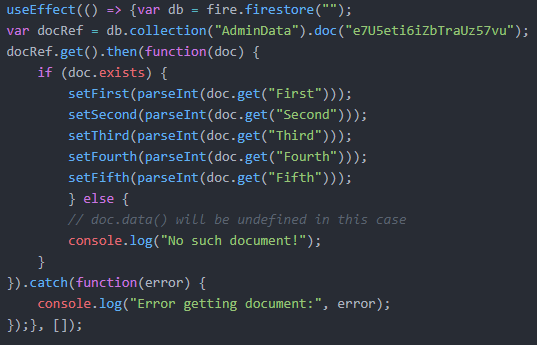
*4.3 A "Kérem válasszon munkakört!" felirat*

Ez egy selected nevű állapottal (a 2. ábrán látható) és egy switch case vezérlési szerkezettel lett megvalósítva. A switch case egyes ágaiban az adott munkanemhez tartozó paramétereket bekérő beviteli mezők az Ionic keretrendszer által nyújtott komponensekkel való megvalósításának renderelése található meg. Az általunk itt felhasznált komponensek pl. az IonGrid, IonItem, IonInput, IonSelect, stb.

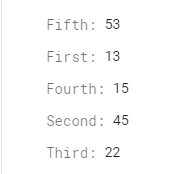
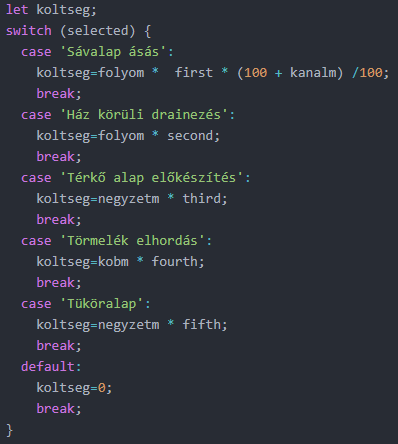


*4.4 Az oldalon megjelenő paraméterlistát megvalósító switch case első ága (Sávalap ásás)*

A várható költséghez tartozó konstans értékeket az adatbázisunk AdminData collectionjében tároljuk el. Ezeket kérjük le az oldal betöltődésekor állapotokba, majd ha a felhasználó beírt egy paraméter értéket, előre beépített képletek alapján számoljuk ki az adott munkához tartozó várható költséget, amit a „koltseg” nevű változóban tárolunk.



*4.5 Konstans értékek lekérése az adatbázisból*



*4.6 Az adatbázisban lévő konstans értékek és a várható költséget számoló képletek*

A határidő bekéréséhez az ingyenesen elérhető React-Calendar package-et használtuk, amely a kiválasztott dátumot egy Date típusú állapotban tárolja el, az adatbázisba pedig timestampként kerül be. A dátumválasztó naptár npm modul telepítéséhez a következő parancsot kell kiadnunk az npm-mel integrált parancssorban: npm install react-calendar. Ezután az alábbi importok használatával tudjuk kódunkban alkalmazni.

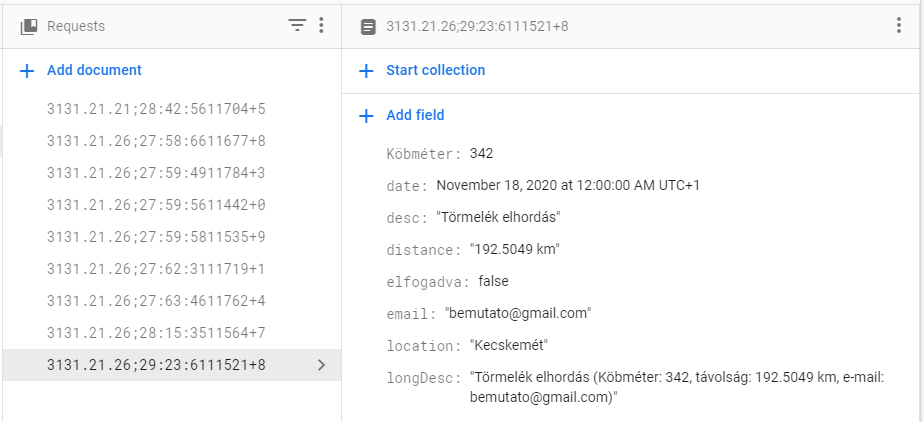


*4.7A dátumválasztó naptár használatához szükséges importok*

Az adatbázisba beküldött kérésekhez szintén csak a releváns paraméterek kerülnek be mezőkként, az előzőhöz hasonló módon megoldva (switch case a selected állapot értékére). A mezőkbe az adott változó- és állapotértékek, egy alapvetően hamis értékű „elfogadva” nevű boolean típusú mező, amely azt jelzi, hogy a kérés még nem került elfogadásra az adminisztrátor által, egy a munkavégzés helye és a telephely városa alapján számolt kilométerben értendő távolság, valamint egy a munkáról hosszabb leírást szolgáltató „longDesc” nevű mező kerül be. A távolságszámítás folyamatáról és az ehhez felhasznált API-ról Balázs kódjában lesz később szó. A dokumentumok ID-ja a beküldés dátumát veszi alapul és ehhez adunk hozzá 1-eseket annak érdekében, hogy pl. egy 11. hóban küldött kérés hátrébb kerüljön a sorrendbe, mint egy 2. havi.

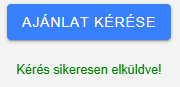
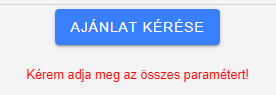
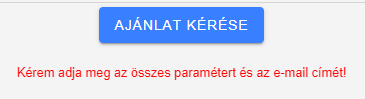


*4.8 Tüköralap munkához tartozó kérés feltöltése az adatbázisba*



*4.9 A Requests collection és egy konkrét kérés szerkezete*

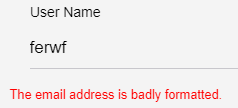
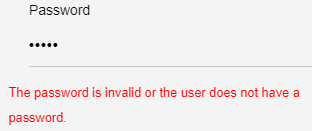
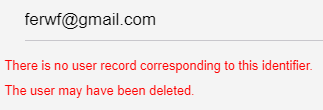
Az árajánlat kérés elküldése az „Ajánlat kérése” gombra kattintva történik meg az oldal alján. Ezalatt a megadott paraméterek és e-mail cím vagy azok hiányának függvényében visszajelzést is kap a felhasználó. Az ajánlat elküldése csak akkor történik meg, ha az adott munkához tartozó összes paramétert, e-mail címet és egy létező várost is megadott a felhasználó.



*4.10 Lehetséges visszajelzések az "Ajánlat kérése" gombra kattintva*

**Bejelentkezés**

A felhasználói autentikáció megvalósítását a Firebase rengeteg előre elkészített, könnyen használható funkcióval segíti. A bejelentekeztetéshez is többféle módszert nyújt, mint pl. az e-mail/jelszó, telefonszám, Facebook-fiók, stb., mi azonban egyelőre csak az e-mail-jelszó párossal való autentikációt választottuk ki. A kódban ezek bekéréséhez két input mezőre, valamint egy gombra volt szükség. Erre rákattintva az alkalmazás elküldi az e-mail címet és a jelszót az adatbázis számára, ami pedig már létező felhasználó és helyes jelszó esetén elvégzi a bejelentkeztetést és átkerülünk az admin oldali napirend menübe. Hiba esetén, mint pl. helytelen formátumú e-mail cím, nem létező felhasználó, letiltott felhasználó, helytelen jelszó, a bejelentkeztetést kezelő függvény a passwordError vagy az emailError állapotnak átadja a hibaüzenetet, ez pedig egy-egy, a beviteli mezők alatt lévő bekezdésben jelenik meg a felhasználó számára.

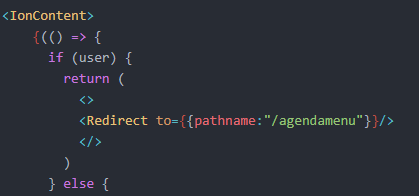
*4.11 Lehetséges hibaüzenetek*

A hibakezelő függvények szintén a Firebase beépített részei, használatukhoz csak meg kell hívnunk az adatbázishoz tartozó auth, illetve signInWithEmailAndPassword függvényeket, utóbbinak paraméterként átadva az e-mail címet és a jelszót tároló állapotokat, majd catch-elnünk és kezelnünk az általuk dobott hibát.



*4.12 A bejelentkezést kezelő függvény*

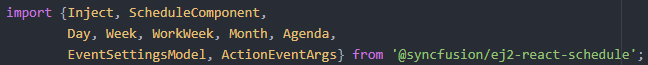
Sikeres bejelentkezés esetén a felhasználó e-mail címe a user nevű állapotban tárolódik el és megjelenik előtte az üdvözlő oldal. Ezt egy, a user értékét vizsgáló elágazással valósítottuk meg. A user állapot alapvetően egy üres string. Amíg ez az állapot áll fent, addig a bejelentkező menü renderelődik az oldalra. Sikeres autentikáció után a Napirend menüpontra irányítjuk át a felhasználót.



*4.13 Átirányítás a Napirend menübe sikeres autentikáció esetén*

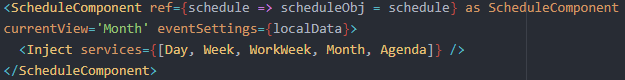
**Napirend**

Ez az adminisztrátori felület elsőként megjelenő oldala sikeres bejelentkezés után. Itt a már elfogadott munkákat láthatjuk egy naptárban több lehetséges nézetben vizualizálva. A mi igényeinknek megfelelő (React keretrendszert használó és helyileg megjelenő, nem a készülék beépített naptárát felhasználó) naptárak választéka viszonylag szűkös, azonban a Syncfusion által kifeljesztett React Scheduler egy nekünk pont megfelelő modulnak bizonyult. Ennek telepítéséhez először is a már jól ismert npm parancssorunkat megnyitva a következő utasítást kell kiadni: npm install @syncfusion/ej2-react-schedule –save. Ezután a következő importokat kell kódunkban alkalmazni.



*4.14 A React Scheduler használatához szükséges importok*

Magát a naptárat az általunk kiválasztott nézeteket Injectelve az alábbi módon tudjuk megjeleníteni.



*4.15 A naptár megjelenítése a kiválasztott nézetekkel*

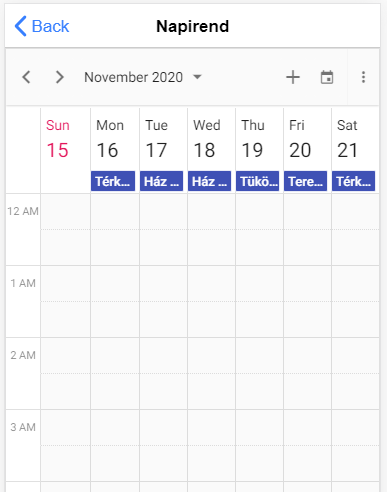
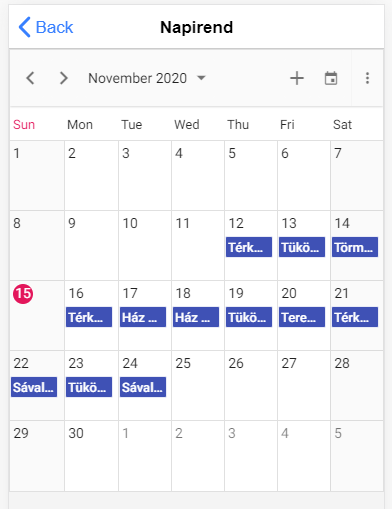
Ennek dinamikus változtathatóságához szükséges példányosítanunk egy ScheduleComponent típusú változót. Az események beállításait egy EventSettingsModel típusú változóban adhatjuk meg.



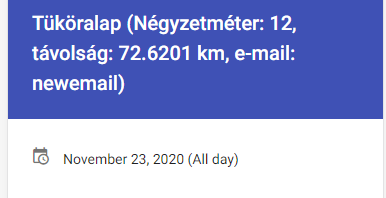
*4.16 A naptárhoz tartozó ScheduleComponent és EventSettingsModel változók*

A naptárban lévő események az oldal betöltődésekor történnek feltöltésre (ezt egy useEffect segítségével tesszük meg), ezeket többféle nézetben is láthatjuk (nap, hét, hónap, stb.). Az itt megjelenő események az adatbázisban lévő, már elfogadásra került munkák. Ezek az elfogadás során kapott dátumra kerülnek be egész napos eseményként, a longDesc mezőben adott leírással.

*4.17 A naptárat eseményekkel feltöltő useEffect*



*4.18 A naptár hónap és hét nézete*

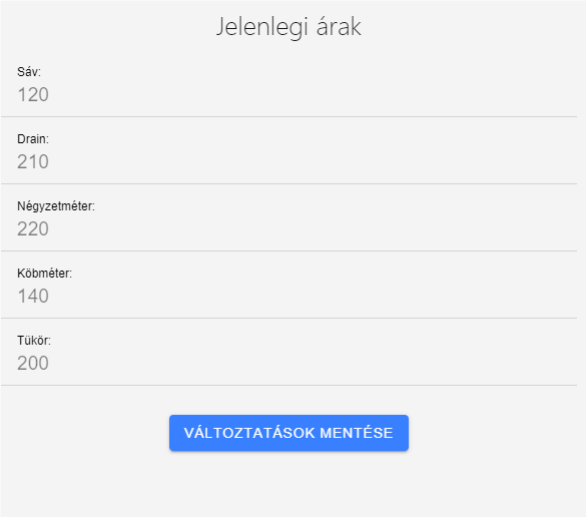


*4.19 Példa egy esemény leírására*

### 3.4.4.3 Abafalvi Balázs kódja (Admin felület)

Admin oldal 2 szegmensből épül fel. Egyrészt itt érheti el a munkavállaló az árbeállításait. Másrészt itt vállalhat, utasíthat el és tehet várakozóra felhasználói kéréseket. Az árbeállítások egy listában tartalmazzák a külömböző munkák költségeit melyeket az admin saját igényei szerint állíthat. Ezt a firebase firestore segítségével tároljuk szerveroldalon. Ezek az adatok egy egyszerű json fileban vannak sorrendben eltárolva így könnyen és gyorsan elérhetjük őket. Árak változtatása esetén az oldal frissül és a megfelelő adatok mind az adatbázisba mind a képernyőn frissülnek. A második funkciója az admin oldalnak az árajánlatok elbírálása. Ez szintén a firestoreban eltárolt requesteket olvassa ki és helyezi el a képernyőre melyeket gestureok (jobb, bal swipe vagy double click/tap) segítségével tudunk elbírálni. Elbírálás frissíti az adatbázist: elfogadás esetén elfogadottá teszi, várakoztatás esetén változatlanul hagyja, elutasítás esetén törli a bejegyzést. A változatlanul hagyott requestek visszakerülnek a lista végére így ha végigért a legfrissebb kéréseken újra elbírálhatja az adott kérést vagy újra halaszthatja a döntést.

**Árváltoztatás**

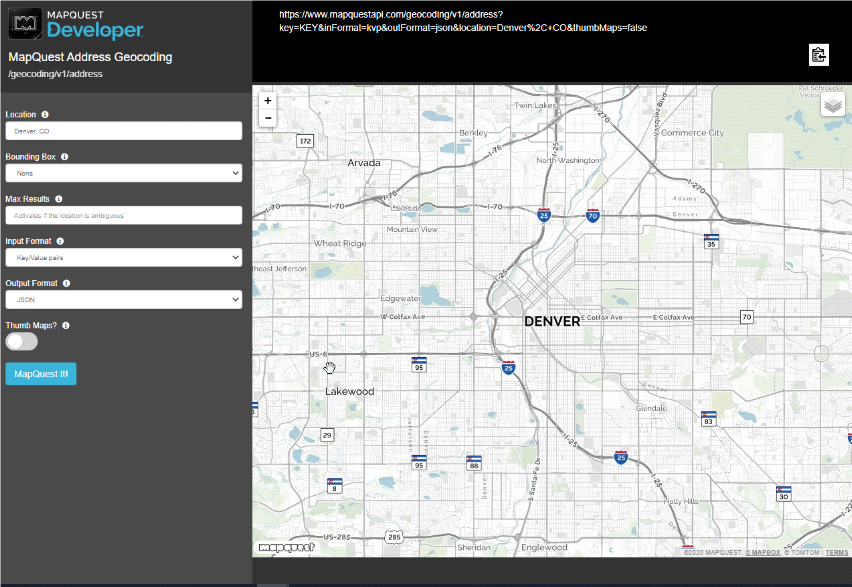


*5.1 Árkezelés oldal*

Lehetővé teszi a szolgáltatások árának módosítását. Üresen hagyott mezők megtartják addigi értéküket. Ezekkel az adatokkal dolgozik az árszámláló.

**Geolocation:**

A helymeghatározást a MapQuest Directiosn API segítségével működtetjük. Egy HTML POST metódussal tudjuk lekérdezni a két helység közti távolságot. Első sorban a szöveges adatot magasság és szélesség koordinátává alakítja (geocoding) majd ezt felhasználva a két pont között egy útvonalat képez. Ennek az útvonalnak az adatait kérdezzük le és használjuk fel. Egy JSON fájlban találhatók az útvonal részletei és ennek a fájlnak használjuk fel a távolság paraméterét mely autóval megtehető távolságot ad meg mely pontosabb számunkra mint légvonalbeli adat.

Mapquest API példa (<https://developer.mapquest.com/>)

*5.2 Geolocation példa*

**Hosting:**

Firebase másik előnye az általuk szolgált hostolási lehetőség. Könnyen használható és meglehetősen sok funkcióval rendelkező szolgáltatás. Lehetővé teszi a komplexebb projektek gyors elérését és könnyű menedzselését pár egyszerű paranccsal. A PWA hostolását a firebase hosting teszi lehetővé. Első sorban buildelnünk kell az applikációt, majd ezt követően firebase CLI: "firebase init" segítségével tudjuk inicializálni majd "firebase deploy" használatával hostolás véglegesedik és megkapjuk a hostolt oldal linkjét. A firebase hosting legnagyobb előnye az ingyenesség mellett az automatikus SSL tanusítvány amit autómatikusan kap az oldal. Kérelmezhető 2 firebase alapú URL de társítható az oldalhoz általunk megvásárolt speciális URL is.

**Elfogadás Elutasítás Várakoztatás:**

A beérkezett kérelmeket a munkavégző 3 féle képpen tudja elbírálni. Elfogadás, Elutasítás vagy várakoztatás. Elfogadás esetén a munkavégző legközelebbi szabad napjára berakja a munkát, feltéve ha az a megrendelő befejezési dátuma előtt van. Ha ez nem teljesül elutasításra kerül a munka. Sikeres munka felvétel esetén a bejegyzés elfogadva mezője igazra és dátuma a munka elvégzésének napjára módosul határidő helyett. Minden munka rendelkezik egy a leadási dátum és idő segtségével kialakított index segítségével így a munkákat érkezési sorrendben tudjuk elbíráltatni. Várakoztatás esetén az adatbázisban új indexet kap jelenlegi dátum és idő segítségével mely így a lista végére kerül tehát az eddig beérkezett munkakérelmek mögé de a jövőben bejövők elé kerül. A bejegyzés többi tagja marad. Elutasítás elején az adatbázisból eltávolítjuk a bejegyzést. A várakoztatáson kívül minden műveletről kap levelet a feladó értesítve őt arról, hogy kérelme elfogadva illetve elutasítva lett valamit annak részleteiről is tájékoztatja. Az elbírált munka után a következőt jeleníti meg, illetve ha kifogyunk értesít erről a rendszer. Az email küldés Email.js alapon működik egy templatebe tudunk változókat beilleszteni (ezek között az egyik a címzett) és így személyre szabott leveleket küldeni.

*5.3 Email küldés kódja*

**Indexelés és queryk**

A naptárba való behelyezés során szükségünk van egy komplikáltabb szűrésre. Szükségünk van az elfogadott munkák közül arra az egyre amelyiknek a legkésőbbi a dátumja. Lekérdezések gyorsítására használjuk a beépített szűrési eljárásokat és ezeknek megfelelő működéséhez egy kompozit index létrehozása volt szükséges. Ennek segítségével jelentősen meggyorsul ennek a lekérdezésnek a sebessége. Ezt az indexet a firebase alkalom adtán felkínálja számunkra és legenerálja. Firebase Queryz meglehetősen egyszerű módon elvégezhető egymás után fűzött functionökkel. Az alábbi kódszelet a requestek közül leszűri a legnagyobb dátumú elfogadott munkát.

*5.3 Indexelés*

## 3.5 Továbbfejlesztési lehetőségek

**GUI**

A gazdaságban fellelhető legtöbb termékre igaz, hogy a megjelenés létfontosságú a fogyasztók számára. Egy kevesebb funkciójú, de jobban kinéző terméknek sokkal nagyobb a piaci kereslete, mint a sok funkciójú de silány módon kinéző terméknek. Erre egy hétköznapi példa az iPhone és bármelyik másik telefon.

A web/mobil alkalmazásokra is ez jellemző. Szívesebben rendelünk egy igényesen kinéző webshopról (még ha nincs is annyi termékük), mint egy igénytelenről. A mi applikációnk számos hasznos funkciót tartalmaz. De sajnos idő és grafikai tudás hiányában igényesség terén hiányt szenved. Mindenképpen célszerű lenne egy grafikus segítségével ezt a problémát orvosolni.

**Komment szekció**

Érdekes lehet elgondolkodni azon, hogy hogyan lehetne közelebb kerülni a saját ügyfeleinkhez. Egy jó ötlet lehet egy komment szekció kialakítása a kezdőlapon.

A komment szekció lényege az lenne, hogy a felhasználók olyan kérdéseket vagy észrevételeket osszanak meg egymással és a céggel, amik segítik az ügyfeleket a megrendeléseinkben, valamint a céget abban, hogy minél jobban kitudja szolgálni az ügyfeleit. Pl:. Egy elégedett ügyfél azt írja, hogy „precíz munka végzés kitűnő áron”, akkor annak az lesz az eredménye, hogy az újabb ügyfeleink nagyobb bizalommal fognak hozzánk fordulni.

A kezdőlapon elhelyezés lényege lenne, hogy a felhasználónak ne kelljen a többi oldalt hiába meglátogatnia. a kommentekből úgy is rá fog jönni, hogy jó helyen jár-e vagy nem!

Későbbiek folyamán ezeket a kommenteket érdemes lehet rendszerezni valamilyen szempont alapján. Pl:. Az oldal használatával, a munkanemekkel, a munkásokkal, a céggel kapcsolatos kommentek különüljenek el egymástól.

**Regisztráció**

Ha tovább szeretnénk vinni a komment szekció ötletét, akkor érdemes lehet azon is elgondolkodni, hogy az egyes kommenteket, hogyan különböztetjük meg a másétól. Erre egy egyszerű megoldás lehet, hogy csak az kommentelhet, aki regisztrált az oldalon, így biztosan eltudjuk majd különíteni egymástól az egyes kommenteket.

## 3.6 Irodalomjegyzék, forrásmegjelölés

**I. Képek**

<https://www.google.hu/?hl=hu/kepek>

**II. Videók**

<https://www.youtube.com/>

**III. Kódok**

<https://stackoverflow.com>/

<https://ionicframework.com/>

<https://reactjs.org/>

<https://www.w3schools.com/>

# 4. Összegzés

A projekt feladat elkészítése előtt semelyikünk se rendelkezett semmilyen tudással azokról a fejlesztői eszközökről, amiket fejlesztés során használnunk kellet. A forrásgyűjtés során azonban egyre jobban beletanultunk a dolgokba. Ugyanis manapság annyi oktató anyag és videó van már fent az interneten, amiből az ember könnyedén tud tanulni. Továbbá, ha program írása közben akad egy problémád, nyugodtan feltételezheted, hogy mással is történt már hasonló eset. Tehát nincs más dolgod, mint interneten keresztül utána keresni a problémának.

A téma kiválasztása nem okozott túlságosan nagy fejtörést, ugyanis mindannyiunknak tetszett az alap feladat koncepció, valamint jártasak voltunk már a weblapszerkesztésben. Ezt az egészet megkoronázta az a tény, hogy egy olyan téma vezető volt kiírva, aki mindhármunkat tanította, és aki mindhármunkra jó benyomást tett.

A projekt feladat készítése közben mindvégig motiváltak voltunk. A rendelkezésre álló rövid idő ellenére megpróbáltuk a lehető legjobbat kihozni az egész fejlesztésből. Mára már tudjuk, hogy a befektetett idő és energia megtérült. Ugyanis amellett, hogy rengeteget tanultunk és rengeteg tapasztalatot szereztünk, egy működőképes terméket állítottunk elő. A mai modern szoftvereket elnézve még nagyon sokat kell fejlődnünk ahhoz, hogy a szakmában jók legyünk. Azonban ez a projekt feladat egy komoly előre lépés volt a célunk elérésében.

Mindent összevetve: a projekt feladatba és a dokumentációba befektetett energiánk és időnk nem volt hiábavaló, ugyanis a tudásunkhoz mérten sikerült elkészíteni őket.