Pannon Egyetem

Műszaki Informatikai Kar

Rendszer- és Számítástudományi Tanszék

Programtervező informatikus BSc

SZAKDOLGOZAT

Időpont foglaló és ütemező rendszer erőmű tüzelőanyag ellátáshoz

Smura Nándor

Témavezető: Dr. Heckl István

2018

**Nyilatkozat**

Alulírott Smura Nándor hallgató, kijelentem, hogy a dolgozatot a Pannon Egyetem Rendszer- és Számítástudományi Tanszékén készítettem a programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Veszprém, 2018.

aláírás

Alulírott Dr. Heckl István témavezető kijelentem, hogy a szakdolgozatot Smura Nándor a Pannon Egyetem Rendszer- és Számítástudományi Tanszékén készítette a programtervező informatikus végzettség megszerzése érdekében.

Kijelentem, hogy a szakdolgozat védésre bocsátását engedélyezem.

Veszprém, 2018.

aláírás

Köszönetnyilvánítás

Tartalom

[1. Bevezetés 7](#_Toc530045709)

[2. A probléma 9](#_Toc530045710)

[2.1. A probléma leírása 9](#_Toc530045711)

[2.2. A probléma megoldása 9](#_Toc530045712)

[3. Hasonló programok 12](#_Toc530045713)

[3.1. ASP.NET Doctor Appointment Scheduling 12](#_Toc530045714)

[3.2. AJAX Event Calendar 13](#_Toc530045715)

[4. Green hospital 14](#_Toc530045716)

[4.1. Összehasonlítás 15](#_Toc530045717)

[5. Használt technológiák 16](#_Toc530045718)

[5.1. Microsoft Visual Studio 2017 16](#_Toc530045719)

[5.1.1. Microsoft Entity Framework 16](#_Toc530045720)

[5.2. SQL Server Management Studio 16](#_Toc530045721)

[5.3. ASP.NET 18](#_Toc530045722)

[5.3.1. Mi is pontosan az ASP.NET? 18](#_Toc530045723)

[5.3.2. Mi az az MVC? 19](#_Toc530045724)

[5.4. GIT 21](#_Toc530045725)

[5.5. C# 22](#_Toc530045726)

[5.6. JavaScript 24](#_Toc530045727)

[5.7. HTML 25](#_Toc530045728)

[6. Grafikai tervek és a rendszer működésének részletezése 27](#_Toc530045729)

[6.1. Bejelentkezés/Regisztráció 27](#_Toc530045730)

[6.2. Adatlap szerkesztés 28](#_Toc530045731)

[6.3. Időpont foglalás 29](#_Toc530045732)

[6.4. Időpont foglalás megerősítése 30](#_Toc530045733)

[6.5. Foglalások kezelése 31](#_Toc530045734)

[6.6. A rendszergazda feladatai 32](#_Toc530045735)

[6.6.1. Felhasználói beállítások 32](#_Toc530045736)

[6.6.2. Telephelyi beállítások 33](#_Toc530045737)

[6.6.3. Anyagbeállítások 34](#_Toc530045738)

[6.6.4. Szerepkör beállítások 35](#_Toc530045739)

[6.7. Limitek beállítása 36](#_Toc530045740)

[6.8. Beszállítás információk 37](#_Toc530045741)

[6.9. Kiértesítés 38](#_Toc530045742)

[6.10. Riport készítése 39](#_Toc530045743)

[6.11. Elutasított beszállítások 40](#_Toc530045744)

[7. Adatbázis terv 41](#_Toc530045745)

[7.1. Entitások és relációk 41](#_Toc530045746)

[7.2. Adatbázis specifikus modell 41](#_Toc530045747)

[7.3. Adatbázis modell (részlet) 44](#_Toc530045748)

[8. Aktuális program 45](#_Toc530045749)

[9. Összefoglalás 49](#_Toc530045750)

[10. Irodalomjegyzék 50](#_Toc530045751)

# Bevezetés

A mai világban már szinte elképelhetetlen, hogy internet nélkül éljünk, így a weboldalaknak is nagyon fontos szerepük van a közéletben. Ahhoz, hogy megtaláljuk, a számunkra megfelelő tartalmakat, megfelelő felhasználói felületekre van szükségünk.

Az átlagember számára az interneten megtalálható fontos dolgok közé tartoznak például a hírek nyomon követése is. Ezen felül sok ember már jobban szereti a hivatalos dolgokat is inkább a neten keresztül elvégezni, sokkal egyszerűbb, mint kimozdulni, és járni a várost a megoldásokat kutatva. Az időpont foglaló rendszer egy nagyon gyakori dolog, amit sok helyen használnak már, például a mozikban, orvosi rendelőkben, de például éttermekhez is tartozhat ilyen oldal. Ezekből az okokból kifolyólag egy ilyen keretrendszer kialakítása nem lehet hátrány, főleg, ha az ember úgy készíti el, hogy az könnyen implementálható legyen akármelyik feladat elvégzésére. Az én projektem elvégzésével is azt remélem, hogy sikerül olyat alkotnom, amit későbbiekben felhasználhatok majd, ha hasonló munkára kérnek fel.

Az első fejezetben lesz szó magának a projektnek a leírásáról, és annak részletes feladatköreiről, amiket el kell tudnia végezni.

A második fejezet felsorolja az általam használt technológiákat, milyen programban dolgozom, milyen keretrendszert használok, és ezeknek melyik részeit használom ki, a feladatom megkönnyítésének érdekében.

A harmadik fejezetben megismerkedünk, az általam használt programnyelvekkel, amiknek segítségével megfogom írni a háttérprogramot, ami később az elkészült rendszer alapjául szolgál majd.

A negyedik fejezetben leírom, hogy hogyan terveztem megvalósítani a projektet, milyen lépésekből áll, és ezeket, hogyan szerettem volna ütemezni.

Az ötödik fejezet tartalmazza a grafikai terveket, így képzelem el a programom megjelenését a jövőben, és minden képhez tartozik egy magyarázat is, hogy az aktuális képen pontosan mi és hogyan zajlik, ezáltal kapunk egy leírást a rendszer általános működéséről is.

A hatodik fejezet leírja az adatbázissal kapcsolatos terveimet, milyen táblákat szeretnék létrehozni, ezeknek mi a szerepe, és milyen kapcsolatban állnak egymással.

A hetedik fejezet tartalmazza az általam felhasznált forrásokat, amiknek segítségével elkészítem a feladatomat, és amik használatával pontosabb információt tudok megadni ebben a dokumentációban, az adott nyelvekről és programokról.

KULCSSZAVAK ------------------------------------------------------

# Erőmű beszállítási, ütemezési probléma

## A probléma leírása

A megújuló energiahordozók (fa, széna, ...) szállításának az ütemezése komplex logisztikai problémát jelent az erőművek számára. A szállítók jellemzően egyszerre, jó idő esetén akarnak szállítani, ami torlódáshoz vezet. A lepakolásnak, ellenőrzésnek jelentős ideje van, ráadásul tűzvédelmi szabályozás miatt az erőmű területén várakozni sem szabad. Egy olyan ütemező rendszert kell létrehozni ASP.NET -ben, ahol az erőmű üzemeltetői be tudják állítani a fogadási időpontokat, a szállítók pedig ezekre az időpontokra tudnak feliratkozni.

## A probléma megoldása

A fent említett problémák kiküszöböléséhez a rendszerben szükség van arra, hogy a sofőrök be tudjanak regisztrálni az oldalra, hogy a későbbiekben használhassák annak minden lényeges funkcióját. Ehhez szükség lesz egy adatbázisra, amely tárolja a már regisztrált tagok cégnevét, e-mail címét és jelszavát. A megfelelő védelem érdekében, a jelszónak legalább 8 karakternek kell lennie, amiben az alábbiakból legalább 3 dolognak szerepelnie kell:

* kisbetű
* nagybetű
* szám
* speciális karakterek (@, &, $ stb.)

Miután ez megtörtént, a rendszer automatikusan átirányítja a felhasználót a „Saját adatok módosítása” menüpontba, ahol felhasználónevet és telefonszámot kell megadnia. Ezen felül alkalma nyílik módosítani a már korábban megadott adatait is.

A regisztrációs felület után nélkülözhetetlen, hogy legyen egy bejelentkezés fül, ahol a tagok belépés céljából megadhatják a cégnevüket és az általuk használt jelszót, amivel beregisztráltak, ezek mellett szükséges megadni a felhasználó által betöltött szerepkört, ami lehet átvevő, beszállító, beszerző, elszámoló vagy rendszergazda. Az esetleges problémák elkerülése végett, lehetőség nyílik új jelszó kérvényezésére, arra az esetre, ha valaki elfelejtené, hogy mit adott meg a regisztráció során.

Az időpontfoglalás menüpont teszi ki a rendszer egyik legfontosabb részét. Itt van lehetőségük a cégeknek egy felhasználóbarát kezelőfelület segítségével időpontokat foglalni. Ennek a folyamatnak az első lépése a telephely kiválasztása, ahová a tüzelőanyagot szállítani szeretnénk. Ezt követően a naptárban megadjuk a kívánt dátumot és az anyag típusát. A szabad időpontok, amikor a szállítás lebonyolítható, kilistázásra kerülnek, ezek közül választhat a felhasználó. Minden telephelyhez tartozik egy napi, heti és egy havi foglalható limit, ami meghatározza, hogy hány tonna szállítható oda a kiválasztott anyagból, az adott idő intervallumokban. Miután a szállítani kívánt mennyiség is megadásra került, amit tonnában mérünk, lehetőség lesz a foglalás elküldésére és rögzítésére a rendszerben, miután megerősítettük, hogy minden adat megfelelő.

Az előző menüponthoz szorosan kapcsolódik a „Foglalásaid” menüpont, mely a korábbi foglalásokat tartalmazza, részletes leírással és adatokkal, ezek természetesen módosíthatók, vagy akár törölhetők is. A menüponton belül lehetőség nyílik egy e-mail elküldésére a telephelynek, ahova a foglalást megtettük, hogy ebben értesítsük őket a részletekről. Ehhez a levélhez automatikusan csatolódnak a foglalási adatok.

A beszerző szerepkörbe tartozó felhasználók nyomon követhetik, hogy a saját hatáskörükbe tartozó telephelyre, melyik cég szállított az adott napon/héten/hónapban (ezek kiválasztására egy naptár segítségével van lehetőség), és ezen kívül azt is, hogy ezek a cégek milyen anyagot szállítottak, és abból mennyit. Lehetséges anyagok és mértékegységeik:

* Keményfa (t)
* Szalma (t) / (db)
* Fa apríték (t)
* Apríték (t)
* Lágyfa (t)
* Átmeneti fa (t)
* Fűrészpor (t)

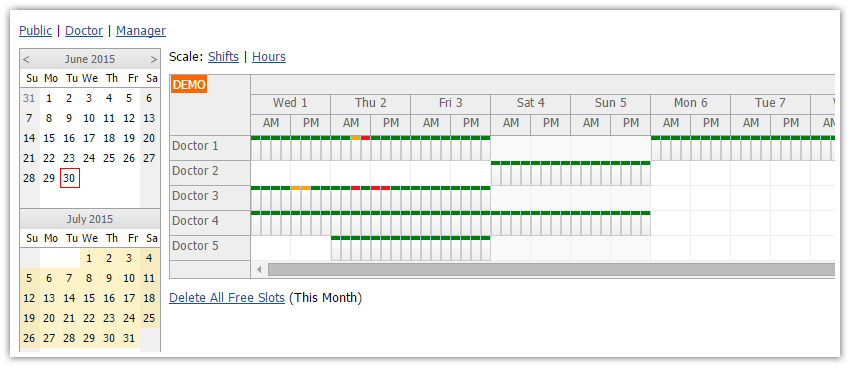
A cégek felsorolása alatt, összesítetten is megjelenik, hogy miből és mennyit szállítottak az adott helyre ezekben az időkben.

Az átvevő feladata, hogy a programban nyilvántarthassuk, hogy melyik nap, melyik beszállítók jöttek az ő telephelyére és hogy az itt megjelenő kamionok, olyan mennyiségben hozták-e a tüzelőanyagot, mint ahogy azt előre lefoglalták. A megjelenő járművek több kategóriába sorolhatók. Elsősorban a jármű miután megkérdezik, átesik egy mérlegelésen, majd távozik, ha minden rendben van. Az is előfordulhat, hogy a jármű megérkezett, de még a telephelyen van. Nem túl szerencsés esetben, a jármű megérkezését követően, az átvevő elutasítja az átvételt, így a kamion a nélkül távozik, hogy bármit is hagyott volna maga után, vagy az is lehet, hogy a jármű még nem érkezett meg a telephelyre. Mindezt az átvevő rögzíti a programban, az egyszerűbb nyomon követés érdekében.

Mint minden rendszer esetében, itt is szükség van egy rendszergazdára, az ő szerepkörébe tartozik az összes felhasználó adatainak a módosítása, ha erre szükség van, illetve jogosultságok kiosztása, rosszabb esetben egy adott felhasználó tiltása. Minden egyes regisztrációt a rendszergazdának kell megerősítenie, hogy biztos legyen a rendszer működése. Továbbá ide tartozik az új telephelyek felvétele és azok adatainak a kitöltése. Itt adható meg, a telephely neve, címe, kapcsolattartó személy neve, e-mail címe, telefonszáma, beszállítható anyagok típusai, nyitva tartás és a maximális kamionok szám/ 2 óra, ami meghatározza, hogy egyszerre hány jármű lehet a telephelyen. Lehetősége van az anyagok kezelésére is, új anyag felvétele, anyag típus felvétele, és a létező anyagokhoz mértékegységek rendelhetők, ahol a megadott egységek között váltószámokat is megadunk.

# Hasonló programok

## ASP.NET Doctor Appointment Scheduling



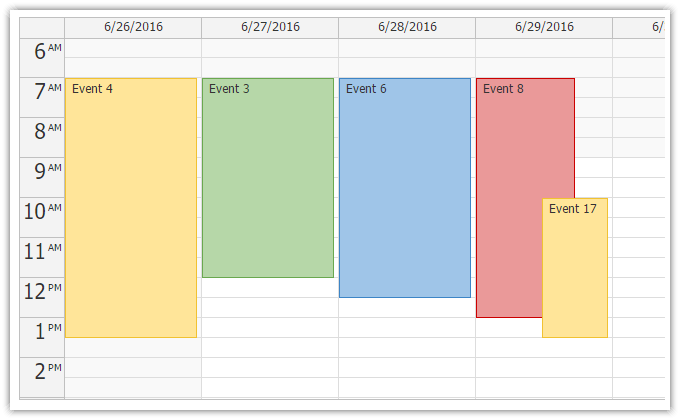
1. ábra: Doctor Appointment Scheduler.

Ez az ASP.NET-ben írt, időpont foglaló rendszer az interneten ingyenesen elérhető forráskóddal együtt. Ez a rendszer orvosi időpont foglalások lebonyolítására alkalmas, és rendelkezik egy grafikus felhasználói felülettel is. A program rendelkezik három szerepkörnek létrehozott kategóriával. A „Manager” létrehozhat időpontokat az orvosok számára, amikor ők tudnak beteget fogadni. A „Public” fülön belül jelentkezhetünk a megadott időpontokra. A „Doctor” jóváhagyhatja a jelentkezést az adott beteg számára. A kezelőfelület rendelkezik naptárral, ahol kiválaszthatjuk a kívánt napot, tud biztosítani heti nézetet, és letudja bontani a napokat úgy, hogy óránként mutassa az elérhető helyeket.

**Pro:** Ingyenes, rendelkezik több szerepkörre szabott kezelőfelülettel, könnyen kezelhető, több időintervallum megjelenítésére is képes.

**Kontra:** Nehezen átlátható, nem rendelkezik adatbázissal, nincs köré weboldal építve, nem lehetséges a regisztráció vagy a rendes felhasználása, kevés rendszerbe implementálható ez a fajta foglaló rendszer megoldás, nem MVC alapú.

## AJAX Event Calendar



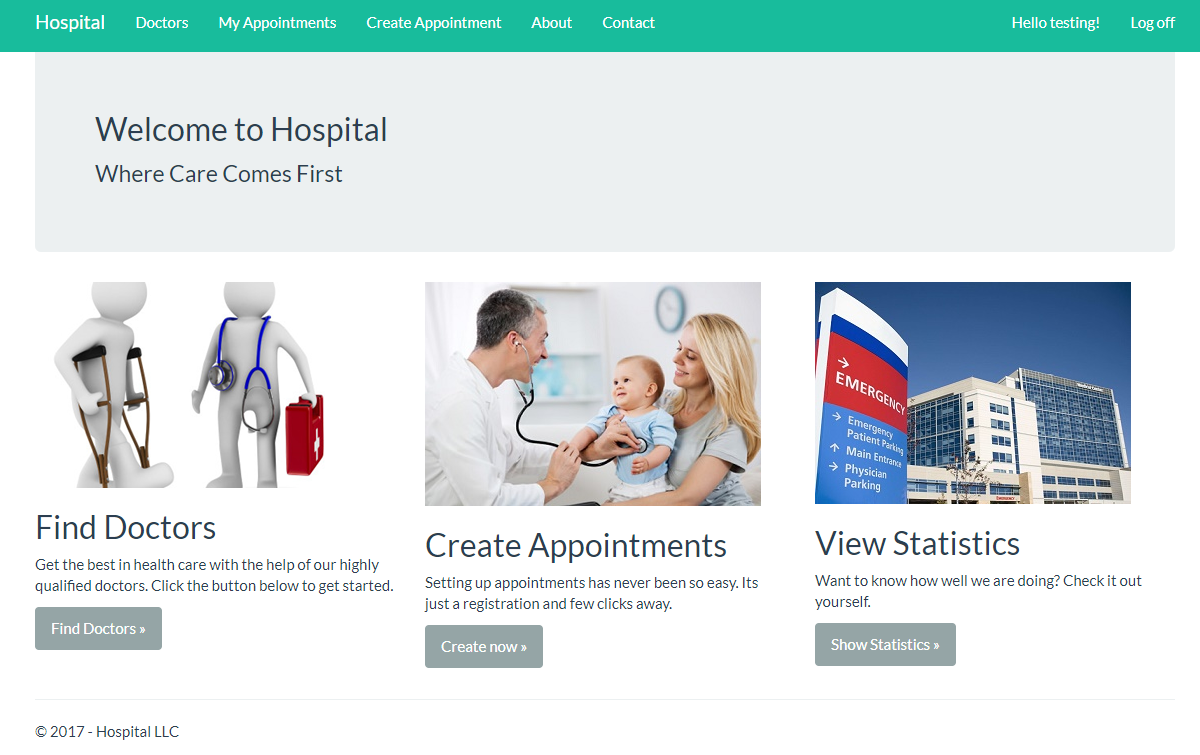
2. ábra: Ajax Event Calendar.

Az AJAX Event Calendar, egy szintén ingyenesen és forráskóddal együtt elérhető program, ez nem egy kifejezetten időpont foglalásra épülő rendszer, de olyan szintan hasonló, hogy itt is eseményeket adhatunk meg, ezekre kéne már csak megírni azt a funkciót, hogy az adott eseményekre a felhasználók jelentkezhessenek is. A program működése annyiban kimerül, hogy rákattintunk a naptárban egy időpontra, és itt egy eseményt hozhatunk létre, tetszőleges néven. Az eseményeket megfoghatjuk egy kattintással és áttehetjük máshova.

**Pro:** Ingyenes, MVC alapú, rendelkezik több nézettel is, nagyon egyszerűen kezelhető, saját eseményeink megjelölésére alkalmas.

**Kontra:** Nem rendelkezik keretrendszerrel, nehezen használható fel más programokban, amit tud azt más cégek már jobban megalkották, nincsenek felhasználók sem adatbázis.

# Green hospital



3. ábra: Green Hospital.

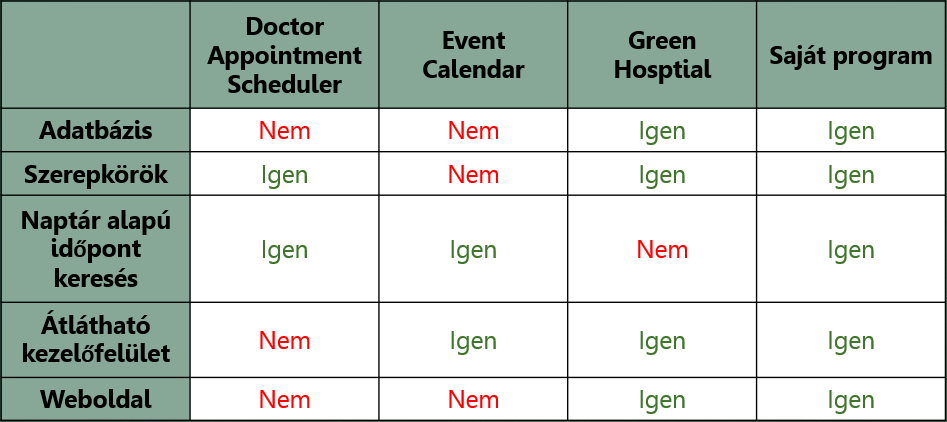
A Green hospital nevű projekt, áll talán a legközelebb ahhoz a rendszerhez, amit nekem is meg kell alkotnom a saját weboldalamon belül az időpont foglalás szekción belül. Rendelkezik rendes kezelőfelülettel, be lehet jelentkezni, regisztrálni, kijelentkezni, van hozzá adatbázis, alkalmazhatjuk orvosként vagy betegként is. Orvosként hozhatunk létre időpontot, amikor szabadok vagyunk, betegként pedig ezeket az időpontokat lehet lefoglalni. A rendszer nagyjából átlátható és rendelkezik az alapvető foglalási funkciókkal és ezeknek a rendes kezelésével. A teszteléshez szükséges regisztrálni is.

**Pro:** Tesztelhető interneten a tényleges működése, rendes adatbázis implementálva, szép megjelenés, orvosi és felhasználói feladatok ellátására is képes, MVC alapú.

**Kontra:** Nem rendelkezünk lehetősséggel, hogy orvosként vagy betegként szeretnénk belépni, nincs rendes naptár nézet a foglalás megkönnyítéséhez, nincs adminisztrátor, saját adataink módosítása kimerül a jelszó megváltoztatásában.

## Összehasonlítás

Az alábbi képen látható táblázat tartalmazza, hogy miben fog különbözni az elkészült programom a hasonló működésű, interneten fellelhető programoktól. Ez segítségül szolgálhat a későbbiekben a tökéletes program megalkotásához.



4. ábra: Összehasonlító táblázat.

# Használt technológiák

## Microsoft Visual Studio 2017

Az egyik általam választott program a Microsoft Visual Studio 2017, ami egy könnyen átlátható és felhasználóbarát fejlesztő környezet, ami rengetek programozási nyelvet támogat. A támogatott nyelvek közé tartoznak például: C++, C#, JavaScript, jQuery, Java és a C programozási nyelv is.

A program számtalan beépített függvénnyel rendelkezik, amik lényegesen megkönnyítik a programozó életét, emellett fejlesztés közben a program felajánlja, hogy a megkezdett függvényt hogyan lehetne kiegészíteni, így, ha valaki tudja, hogy mit szeretne csinálni, annak sokkal egyszerűbbé válik a kódolás. Az automatikus kiegészítésnek hála elég az Enter-t "ütögetni" és máris rendelkezésünkre áll a kívánt függvény.

### Microsoft Entity Framework

Első megjelenése 2008 augusztus 11-én volt, ekkor nagyon sok kritikát kapott a felhasználóktól, de a legújabb 5.0.0 verzió már egy sokak által használt keretrendszerré vált, amit kifejezetten a .NET-hez készítette. Ennek segítségével leegyszerűsíthetjük például az adatbázisaink létrehozását, vagy éppen frissíthetjük az SQL táblánkat miután a kódunkban megváltoztattuk az adattagjainkat. Megtalálható benne például egy adatforrás-specifikus kiszolgáló, leképzelés kiszolgáló de EDM elemző és nézet leképzés is.

## SQL Server Management Studio

Ezek mellett a Visual Studio rendelkezik az SQL Server Management Studio-val is, ennek használatával csatolhatunk egy adatbázist a projektünkhez. Ezt megtehetjük egy már létező, online adatbázis segítségével is, melyet pár kattintás után a VS szinkronizál is a kódunkkal. Megtehetjük, hogy nem egy létező SQL szervert alkalmazunk, hanem lokálisan készítünk egyet, amit csak mi tudunk felhasználni, így offline mode-ban is működik, és tudunk vele tesztelni. A feladatot megközelíthetjük két módon is.

Az egyik, hogy először létrehozzuk a kívánt táblákat, ezt megtehetjük két féle módon. Használhatunk SQL kódokat, vagy grafikus felületen is megadhatjuk, hogy milyen oszlopokat szeretnénk felvenni, és ezek milyen típussal rendelkezzenek, ezek mellett szabhatunk kikötéseket is az adott oszlopra, például megadhatjuk, hogy az vehet-e fel NULL értéket, vagy sem.

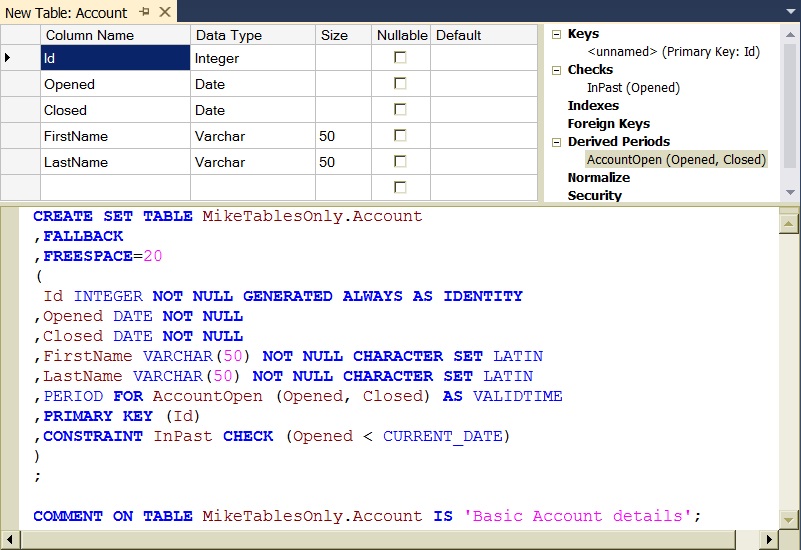
Ha a táblát létrehoztuk, akkor van lehetőség ebből legenerálni a Modellhez tartozó C# fájlokat, ezek tartalmazzák, a különböző adattagokat, és itt adhatunk meg “Validation”-t is. Ez annyit jelent, hogy ha van egy kitöltendő mező, például a felhasználónév, akkor itt megadhatjuk. hogy a felhasználónév legalább 5 karakterből kell, hogy álljon, vagy azt is, hogy ha a jelszónak kell tartalmaznia speciális karaktereket. Innen már csak pár kattintás, és létrehozhatunk egy controllert is, ami tartalmazza a teljes CRUD-ot.

A másik megoldás, hogy először a modell-t készítjük el, benne az adattagokkal, amiket szeretnénk, ha tartalmazna későbbiekben a táblánk. Ezek után a NuGet console-ba beírjuk az alábbi parancsokat és máris létrehozza az SQL tábákat.

Install-Package Microsoft.EntityFrameworkCore.Tools

Add-Migration Initial

Update-Database



5. ábra: Új SQL tábla létrehozása.

Az adatbázisunkat feltölthetjük, a grafikus felületen keresztül, ami rendelkezésünkre áll az SQL ServerManager-en belül, vagy akár a weboldalon is, a létrehozott CRUD segítségével, miután ezt a projekten belül elérhetővé tettük a webes felületen is. Ezen kívül létrehozhatunk egy file-t, ami tartalmazza a “seeded” adatokat. Ide fel vehetjük a kívánt adattagokat, majd ezt az egészet feltölthetjük adatbázisunkba. Így néz ki az interface, ahol a táblát létrehozhatjuk manuálisan, és nem a kódból automatikusan legenerálva.

## ASP.NET Core Web Application 2.0 (MVC)

A legfontosabb szempont, ami miatt a Microsoft Visual Studio-t használom az az, hogy ASP.NET-ben szeretném elkészíteni a projektet. A VS-ban rendelkezésemre áll létrehozni egy ASP.NET Core Web Application 2.0-t, ezen belül is egy MVC alapú webes alkalmazást.

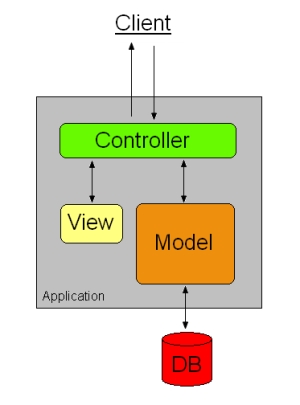
### Mi is pontosan az ASP.NET?

Az ASP.NET[1] egy egységes web fejlesztő modell, ami olyan elemeket tartalmaz, melyek használatával nagyméretű webes alkalmazások készíthetők minimális kódolással. ASP.NET része a .NET keretrendszernek, így amikor ASP.NET-ben kódolunk, elérhetőek a .NET osztályok is. Bármilyen hétköznapi programozási nyelvet használhatunk, például a C#-ot is. Több féle ASP.NET alapú webes alkalmazás készíthető, fejleszthetünk Angular-ban, csinálhatunk sima Web API-t vagy akár MVC alapú projektet is készíthetünk.

A saját rendszeremhez az MVC alapú webes alkalmazást választottam. Ennek a modellnek az alapjai pedig a C# és JavaScript, amik a programozás alapjait adják, és a kinézetet pedig a HTML elemek alkotják, mint minden webes alkalmazásnál is. Ezek mellett fel lehet használni a kódhoz akár jQuery fájlokat is. Például, ha megszeretnénk adni, hogy egy mezőbe milyen értékeket lehessen beírni, és ezt rögtön ellenőrizni is szeretnénk, akkor jQuery-t használunk. Az adatbázis kezelésnél pedig az alapértelmezett, amit a legkönnyebben tudunk implementálni az SQL.

### Mi az az MVC?

Az MVC[3] egy mozaik szó, ami a Modell-View-Controller szavakból áll össze. Ennek magyar jelentése a Modell-Nézet-Vezérlő. Ennek előnye, hogy könnyen elválaszthatók egymástól az adatokat kezelő, tároló és megjelenítő kódrészletek, így ha csak a kezelőfelületet szeretnénk változtatni, vagy fejleszteni, akkor azzal nem ronthatjuk el az üzleti logikát kezelő részleteket, és nem törlünk bele, vagy változtatunk a már meglévő adattagokon, amiket a rendszer készítése és tesztelése során rögzítettünk.



6. ábra: MVC modell felépítése.

**Modell:** Az alkalmazás által kezelt információk tartomány-specifikus ábrázolása. A tartománylogika jelentést ad a puszta adatnak (pl. kiszámolja, hogy a mai nap a felhasználó születésnapja-e, vagy az összeget, adókat és szállítási költségeket egy vásárlói kosár elemeihez).

**Nézet:** Megjeleníti a modellt egy megfelelő alakban, mely alkalmas a felhasználói interakcióra, jellemzően egy felhasználói felületi elem képében. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez.

**Vezérlő:** Az eseményeket, jellemzően felhasználói műveleteket dolgozza fel és válaszol rájuk, illetve a modellben történő változásokat is kiválthat.

Sok alkalmazás használ állandó tároló eljárásokat (mint mondjuk egy [adatbázis](https://hu.wikipedia.org/wiki/Rel%C3%A1ci%C3%B3s_adatb%C3%A1zis)) adatok tárolásához. Az MNV nem említi külön az adatelérési réteget, mert ezt beleérti a modellbe. Az MNV gyakran található meg webes alkalmazásokban, mivel itt a nézet a HTML kód, ami a megjelenésért felelős, a vezérlő pedig az a kód rész, ami összegyűjti a dinamikus adattagokat, amikből létrejön a HTML tartalma. A modellt maga a tartalom, ezt általában adatbázisban vagy XML állományokban tároljuk. Az MNV általános működésére egy példa:

1. Felhasználó megnyom egy gombot.
2. A vezérlő átveszi a bejövő eseményt a felhasználói felülettől egy eseménykezelő útján
3. A vezérlő kapcsolatot teremt a modellel, ha a gomb adatmódosító funkcióként szolgált, akkor ennek megfelelően változtat a modell tartalmán.
4. A nézet ezek után a már frissített adattagokat mutatja. A modellnek nincs közvetlen tudomása a nézetről.
5. Majd visszaérünk az első ponthoz, ahol a felhasználói felület újabb utasításra vár.
6. A modell és a nézet kettéválasztásával az MVC csökkenti a szerkezeti bonyolultságot, és megnöveli a rugalmasságot és a felhasználhatóságot.

## GIT

A projektem elkészítéséhez a Git verziókezelő rendszert választottam. A Git egy olyan rendszer, ami arra szolgál, hogy ide fájlokat (programok, dokumentációk, stb) töltsünk fel, és ezekből különböző verziókat tarthassunk biztonságosan, egy helyen. Így nem kell saját meghajtón elmenteni külön mappákba a különböző próbálkozásainkat, és mappákon belül bogarászni, hogy hol is van épp a legutóbbi használható verzió, ha esetleg a mostaniban már olyan károkat okoztunk, amiknek a visszacsinálása hosszas időt venne igénybe, amit egyébként fejlesztéssel is tölthetnénk.

Amikor commitolunk egy file-t, azt a Git hozzáteszi egy adatbázishoz, ahol nyomon tudjuk követni a fájlon történt változásokat, visszaállíthatjuk akármelyik korábbi commit-ra, vagy a commit parancs segítségével újabb verziót tehetünk fel. Összeköthetjük más gépen fejlesztett dokumentumokkal is, így, ha a két dokumentum között változás történik, azt a Git jelzi nekünk, hogy hol és mi volt pontosan az a változás, ami végbe ment, vagy ha nem sikerült változtatásokat létrehoznia, akkor arról is küld értesítést.

Aki már komolyabb szinten elmélyült a Git világában, az megoldhatja azt is egy csoportos projektmunka keretein belül, hogy amikor commitol az adatbázisba, akkor mindenki másnak, aki abban a könyvtárban dolgozik, küld egy értesítést Git-en keresztül, hogy most már változások történtek. Ha változás történt az adatbázisban meglévő fájlban, akkor mielőtt commitolni tudnánk a saját munkánkat, először le kell “pull”-olni, tehát le kell “húzni” a változásokat a saját számítógépünkre.

A Git könyvtárban mindig van egy fő szál, de ezek mellé létrehozhatunk külön elágazásokat is, ezek arra is jók, ha egy csoport munkában dolgozunk, akkor tudunk saját a könyvtárat létrehozni, ez ugyanazokat a fájlokat tartalmazza, mint a főkönyvtár, de ebben csak mi dolgozunk.

A Git-nek van egy olyan jó tulajdonsága is, hogy a fentiekben említett dolgokat, képes titkosított csatornánk (ssh-n keresztül) is elvégezni. Annyi a különbség a saját gépen való dolgozás, és a Git-en történő verzió kezelés között, hogy Git-en nem elég csak a file-t elmenteni, hanem mindenképp commitolni kell a fájlunkat. Ezzel jár egy komment is, hogy épp az adott verziónál milyen változtatásokat tettünk meg. Majd egy “push” paranccsal az egészet ténylegesen fel is töltjük az adatbázsiba.

Egyre több cég használja a Git-et, és vannak mindenki számára elérhető ingyenes Git szerverek is. Ezeknek annyi lehet a hátránya, hogy publikus könyvtárakat hozhatunk csak létre, de ha hajlandóak vagyunk egy kis pénzt kis kiadni, akkor máris biztonságosabbá tehetjük a projektünk tárolását, és privát repositoryvá alakíthatjuk, azt, ahol tároljuk. Ilyen ingyenes oldalak például a github.com, amit én is használok, vagy a bitbucket.org, de ezeken kívül is akadnak lehetőségek, ha valakinek egy számára kézenfekvőbb felületre van szüksége.

## C#

Maga a név “C sharp” a zenei hangjegyből származik, egyfajta szójáték, mint a C++, ahol a két + azt jelenti, hogy egyel, növeljük az értékét, itt négy darab + található meg a C mellett, ezzel is utalva, hogy a C# jobb nyelvnek lett tervezve, mint a C++.

A C#[4] egy olyan, a Microsoft által a .NET keretrendszer részeként kifejlesztett programozási nyelv, aminek alapjául a C++ és a Java szolgál. Az első bemutatás 2000-ben történt meg, és az első megjelenése 2001-ben volt. Megalkotására azért volt szükség, mert eredetileg a Microsoft féle Javára fejlesztett alkalmazások nem lettek volna futtathatók más rendszereken, ami ellenkezik a Java platform-függetlenségére vonatkozó alapelvével. Az ebből adódó pereskedés után a Microsoft kénytelen volt eltávolítani a Java-t a rendszerükből. Ebből kifolyólag hoztak létre egy saját keretrendszert, ez lett a .NET, ehhez adták ki a C# első verzióját. Később ezt a nyelvet úgy fejlesztették tovább, hogy meglegyen a programozó nyelvi szabadsága és a gyors alkalmazás fejlesztés lehetősége is megmaradjon. A Microsoft rengetek hivatalos bővítménnyel és tutoriallal[2] rendelkezik, mind a C#[6] hoz, mind a .NET-hez. A jelenlegi C# a 7.1 verziószámnál tart, ami az 1.0 óta jelentős fejlesztéseken ment keresztül, és rengetek új lehetőséget tartalmaz a hozzáértő fejlesztők számára. A 7.1-es verzió a Visual Studio 2017 Update 3-mal együtt érkezett meg teljesen elkészülve, és ekkor vált elérhetővé a felhasználók számára is.

1. táblázat: C# verziók fejlődése az évek során.

|  |  |  |
| --- | --- | --- |
| Verziószám | Megjelenés | Főbb tulajdonságok |
| 2.0 | 2005. november | Generikus és parciális típusok, anonim metódusok, iterátorok |
| 3.0 | 2006. november | Implicit módon megadott lokális változók, lambda-, és lekérdezés- kifejezése, objektum inicializálók |
| 4.0 | 2007. november | Language integrated Query, Lambda kifejezések, kiegészítő metódusok |
| 6.0 | 2010. április | Dinamikus kötés, opcionális paraméterek, generikus ko- és kontravariancia, Párhuzamos programozás, PLINQ |
| 5.0 | 2015. július | Null kondicionális operátor, statikus importálás, csak olvasható auto tulajdonságok |
| 7.0 | 2017. március | Lokális funkciók, tuples, szám szeparátorok, Pattern egyezés, Out variables |
| 7.1 | 2017. augusztus | Async main, default literal expressions |

Magának a nyelvnek, a legfontosabb funkciója, hogy objektum orientáltan tudunk benne programozni. Nem csak a weboldalak elkészítésénél használt a .NET rendszeren belül, hanem például az Unity game engine is ezt alkalmazza a játékok fejlesztésénél, és mivel a játékiparnak egyre nagyobb befolyása van a piacon, így a nyelv használhatósága is fontos tényező.

A C# egy erősen típusos nyelv, így nincs implicit típuskonverzió sem. Ez annyit jelent, hogy míg más nyelvén, ha létre szeretnénk hozni egy változót, ami egy számot tartalmaz, akkor először meg kell adnunk a változó típusát (int i = 1), ugyanez, akkor, ha szavakat szeretnénk benne tárolni, és így tovább. Itt elég létrehoznunk egy “var” változót, ami önmagától képes eldönteni az átadott érték segítségével, a változó típusát, Pl.: var i = 1, var mondat = “ez egy mondat”, itt az i változó típusa integer, ami számokat tárol, a mondat változó típusa pedig string, ami karakter tömör tárolására alkalmas. Ugyanígy működik például a C++ nyelvben az auto kulcsszó, aminek segítségével könnyebben járhatunk be listákat, vektorokat vagy mapeket.

Összességében a C# egy könnyen áttekinthető nyelv, amiben egyszerű programozni, és érdemes is beletanulni, mivel a Microsoft még mindig foglalkozik a fejlesztésével, és rendszeresen ad ki hozzá frissítéseket, így nem kell félnünk, hogy elévül a programnyelv, miközben azzal dolgozunk.

## JavaScript

A JavaScript[7], gyakran csak JS-nek emlegetetett, egy gyengén típusos, objektum alapú programozási nyelv. Első megjelenése 1995 decemberében volt.

A HTML és a CSS mellett ez az egyik fő alkotó eleme a weboldal fejlesztésnek. Arra használatos, hogy interaktívvá tegyük vele weboldalunkat, ebbe beleértve az online játékokat is. A weboldalak nagy része támogatja, és ebből kifolyólag nincs is szükség a mai böngészők többségében arra, hogy egy plug-in-t töltsünk le a használatához. Nagy hasonlóság van a JS és a Java között, mind magában a nyelvben, szintaxisban és alapvető könyvtárakban, de a két nyelv nagyon különbözik egymástól a megjelenésükben. Kezdetekben több fajta JS típus volt, amik különböző böngészőkben voltak jelen, ez azzal a hátránnyal járt, hogy ami az egyik böngészőben (Netscape) működött, ugyanaz a másikban (Internet Explorer 3) nem, így, ha egy cég szeretett volna weboldalt létrehozni, akkor azt mindkét, akkoriban legnépszerűbb böngészők közé tartozó programban működésre kellett bírniuk, de erre sok fejlesztőnek nem volt pénze vagy ideje. Későbbiekben ezt a problémát megoldották, és úgy tervezték a nyelvet, hogy a legtöbb platformon ugyanúgy szereplejen.

JavaScript egy igen népszerű nyelvvé vált, annak ellenére, hogy kezdetekben sokan “amatőr” programozási nyelvnek tekintették, mivel webes környezetben volt használatos, ez a nézet akkor változott, meg amikor megjelent az Ajax, ezzel, sokkal nagyobb figyelmet kapott a fejlesztőktől. C# hoz hasonlóan itt is tudunk “var” változó típusokat használni. Mivel a JS egy gyengén típusos nyelv, itt lehetőségünk van összeadni egy szöveget egy számmal, ami azt eredményezi, hogy a szöveghez hozzáfűződik a szám. A JavaScriptben jelen van, az úgy nevezett “hoisting”, ez azt jelenti, hogy a változót nem kell deklarálnunk, mielőtt használnánk, hanem ráérünk utána is. Ettől függetlenül mindig érdemes előtte deklarálni, hogy ne keveredjünk bele, ha ezt szeretnénk kikötni a JS nyelvben is, akkor a “use strict;” sort kell beírnunk a kódunk elejére. ASP.NET is használ JS elemeket, amiknek az oldal megjelenítésében van szerepük.

## HTML

A HTML[5] (HyperText Markup Language = hiperszöveges jelölőnyelv) egy olyan leíró nyelv, amelyet kifejezetten a weboldalak fejlesztéséhez alkottak meg, ez olyannyira jól sikerült, hogy ma már szabvánnyá vált a HTML alkalmazása, és nélküle nem is tudnánk megalkotni egy weboldalt.

Először 1993-ban jelent meg, azóta már az 5.1 2. verziónál jár. A HTML blokkok alkotják meg a weboldalunk kinézetét, ehhez még hozzácsatolhatunk CSS fájlokat, ami kifejezetten arra készült, hogy a HTML formázását tegyék könnyebbé. A HTML tartalmaz beépített eszközöket, mint például táblákat, listákat, naptárt, kép beszúrási lehetőséget, de akár linkeket is megjeleníthetünk, amikre rákattintva rögtön átdob minket az oldalra, amit szeretnénk elérni. Ahhoz, hogy a HTML kódunkat meg tovább fejleszthessük, hozzá csatolhatunk egy megírt JavaScript kódot is. A saját kódomban ez úgy néz ki, hogy létrehoztam egy HTML tag-et, ami egy dátum típusú adattagot vesz át, de megjelenésben egy naptárnak néz ki. Ebben a naptárban kattinthatunk napokra, de az adatot nem tárolja el sehol, html kód segítségével készíthetünk egy külön gombot, ami ezt elküldi a controllerünknek, de sokkal egyszerűbb írni rá egy scriptet, ami akkor aktiválódik, ha a naptárban új adatra kattintottunk, és ilyenkor rögtön elküldi a kiválasztott adatot az üzleti logikát tartalmazó fájlokba. A HTML[8] nem tartalmaz konkrét változókat, mint a többi nyelv, csak képes az oldalon történt változásokat kezelni, és ezt átadni függvényeknek, hogy azt később felhasználhassuk.

“Tag”-eket használva alkotjuk meg az oldalt, például, ha egy címet szeretnénk csinálni, azt úgy tennénk meg, hogy <h1> Cím </h1> így a két tag között megjelenő adat h1 típusú címként jelenik majd meg az oldalon. A legtöbb tag párban van egy nyitó (<h1>) és egy ehhez tartozó záró(</h1>) ként, de vannak olyanok is, amik csak egy darabból állnak, ilyen például a link( <a href=<https://www.youtube.com/>> Ez egy link a youtube-ra</a>), vagy a sortörés(<br>) is.

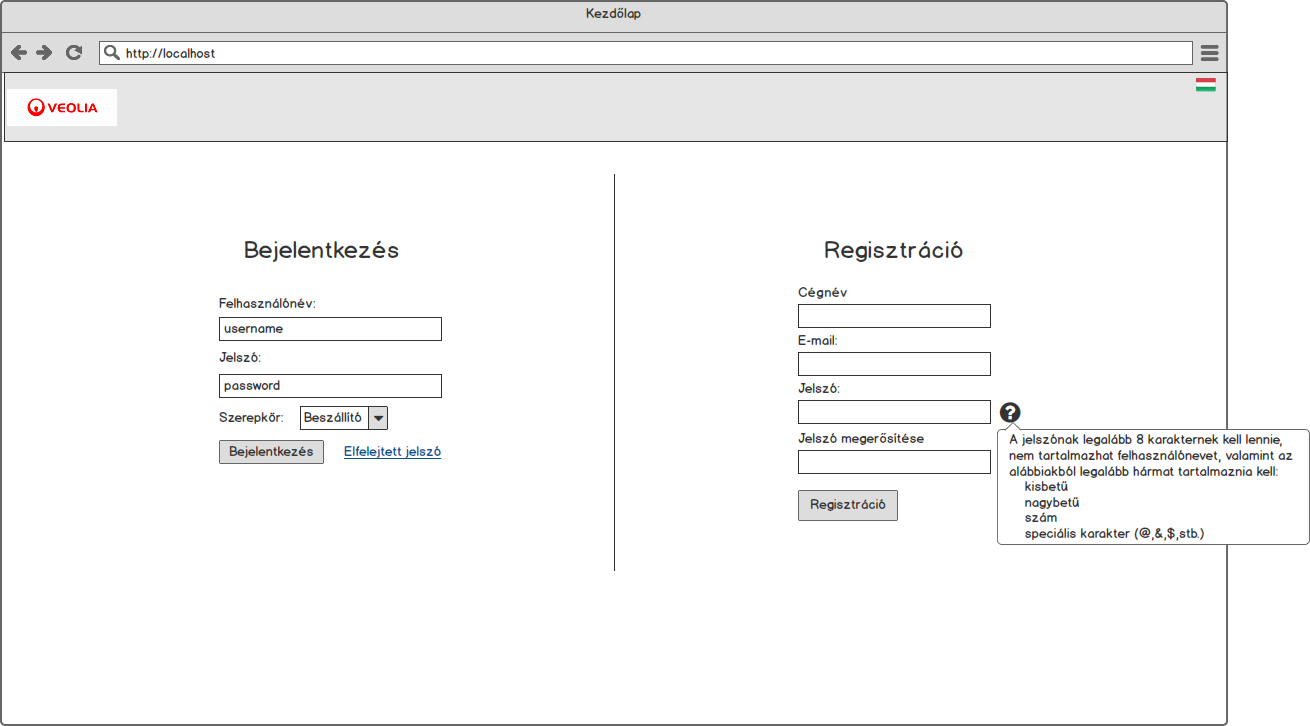
A HTML következő lépése a HTML5, ennek célja, hogy a böngészőknek ne legyen szüksége olyan plug-in-ek használatára, mint például az Adobe Flash, hogy Flash player-t használó oldalakat is tudjunk futtatni.

# Grafikai tervek és a rendszer működésének részletezése

## Bejelentkezés/Regisztráció

Az alábbi képen látható, a kezdőoldal terve, kiválasztható bejelentkezésnél, hogy melyik pozícióba tartozunk, majd megfelelő jelszó és felhasználónév megadása után tovább léphetünk a rendszerbe.

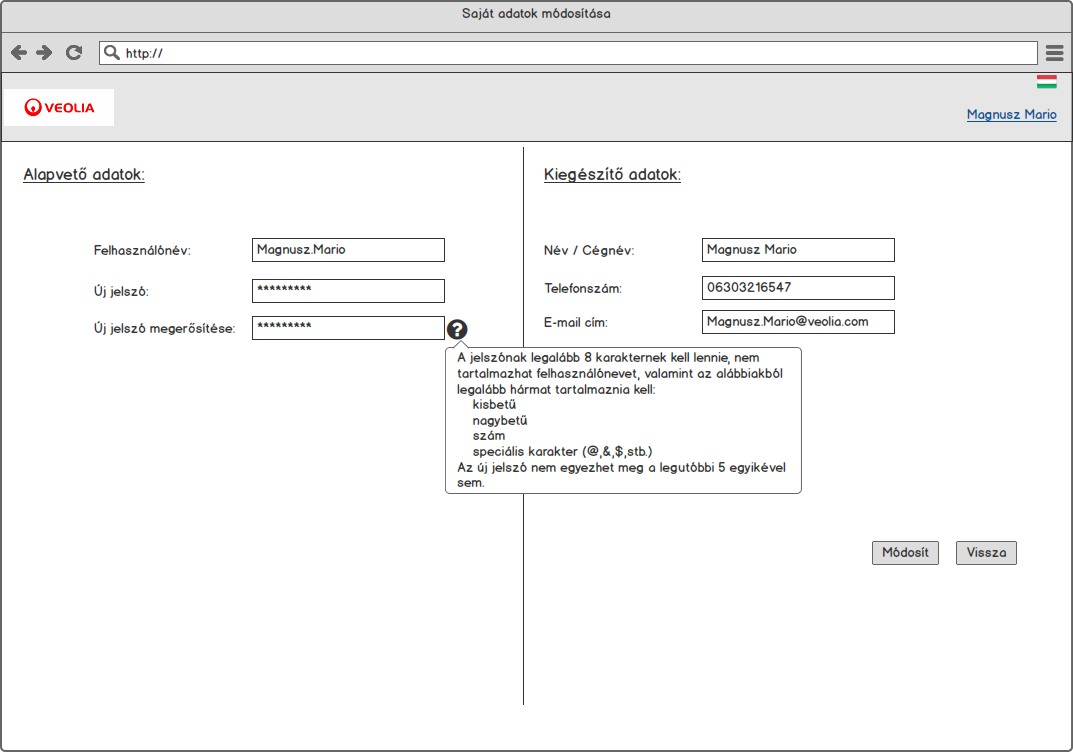
Jobb oldalon látható a regisztráció folyamata, ahol a legfontosabb adatokat kell megadnunk, Cégnév, E-mail és a jelszavunk, de ezekhez jönni fog majd még egy username is, amit ki kell választananunk. Amint azt a rendszer jelzi is a jelszónak legalább 8 karakternek kell lennie, valamint tartalmaznia kell speciális karaktereket is.



7. ábra: Kezdőoldal.

## Adatlap szerkesztés

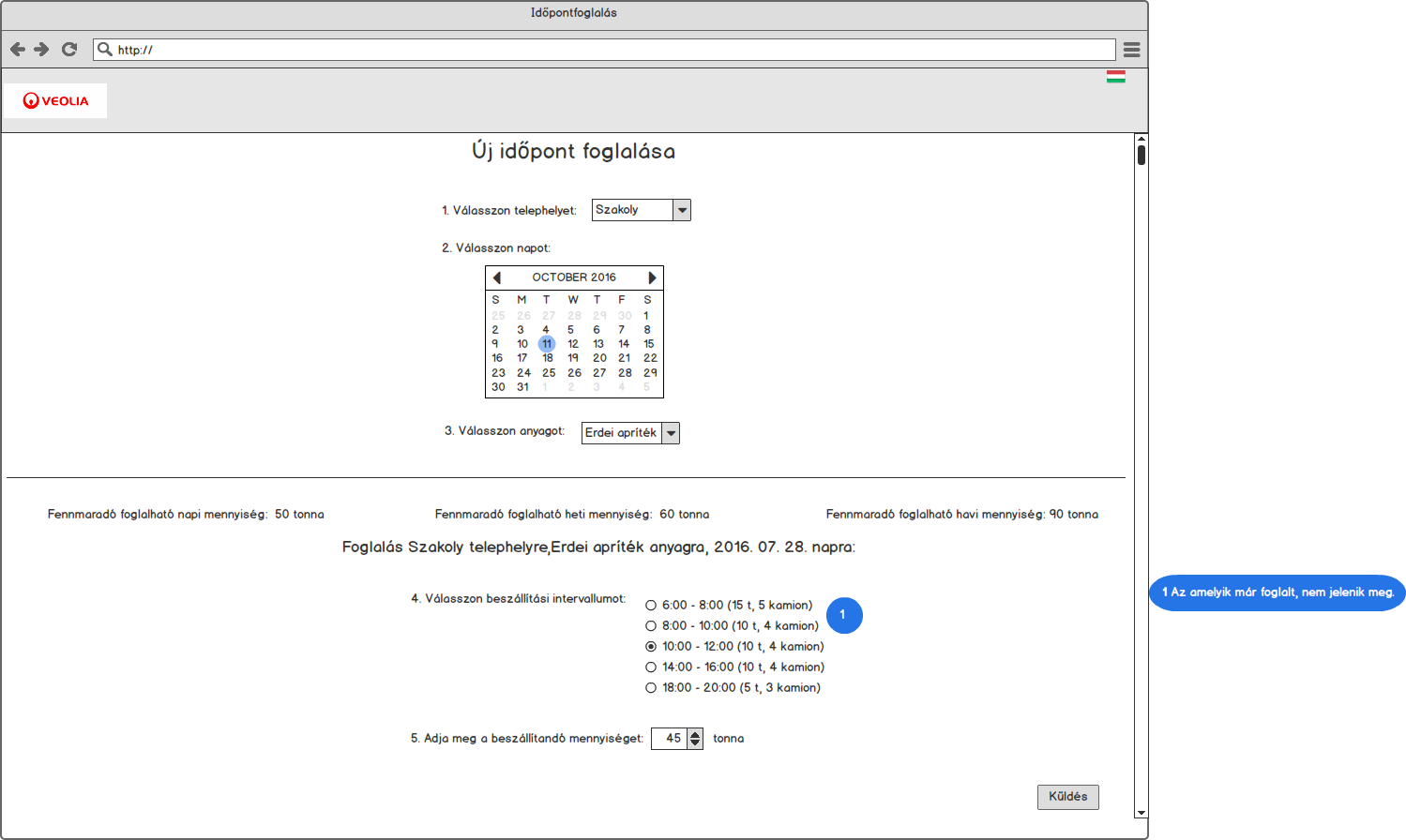
Saját adataink kezelésére a következő felületen van lehetőségünk. Itt tudunk a regisztrációnál megadott adatainkon változtatni, majd elmenteni az újonnan megadott értékeket, amiket szeretnénk, ha a rendszer tárolna.



8. ábra: Saját adatok módosítása.

## Időpont foglalás

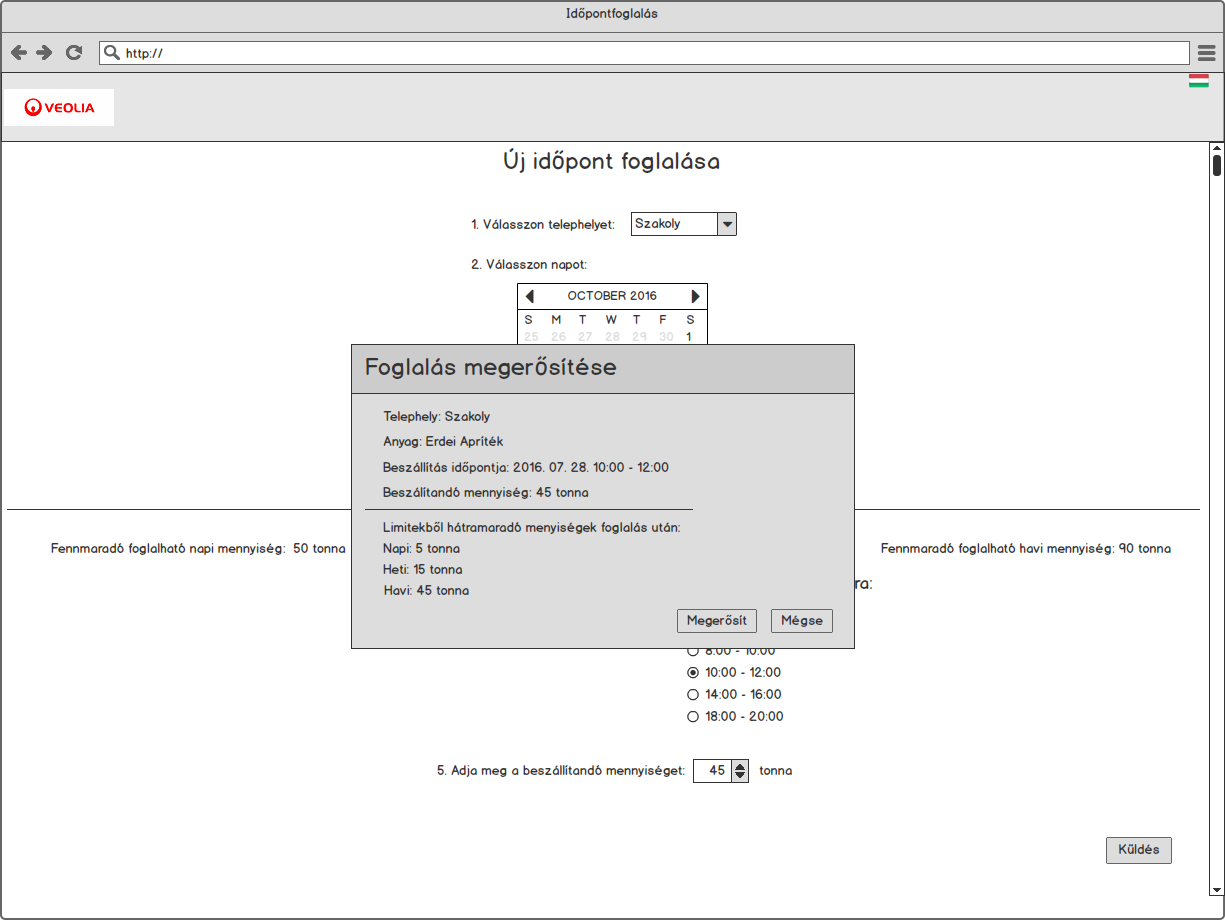
A beszállító számára elérhető foglalási felületen, tudunk új időpontot lefoglalni egy naptár segítségével. Miután kiválasztottuk a telephelyet, a dátumot a naptár segítésével, és a szállítani kívánt anyagot, a rendszer ki jelzi, hogy aznapra mikor vannak elérhető időpontok, kijelzi a napi, heti és a havi korlátjainkat és, hogy az alábbi foglalás ebből, mennyit vesz majd igénybe. Ezek után a szállítani kívánt mennyiség megadását követően, el is küldhetjük a rendelésünket.



9. ábra: Időpontfoglalás.

## Időpont foglalás megerősítése

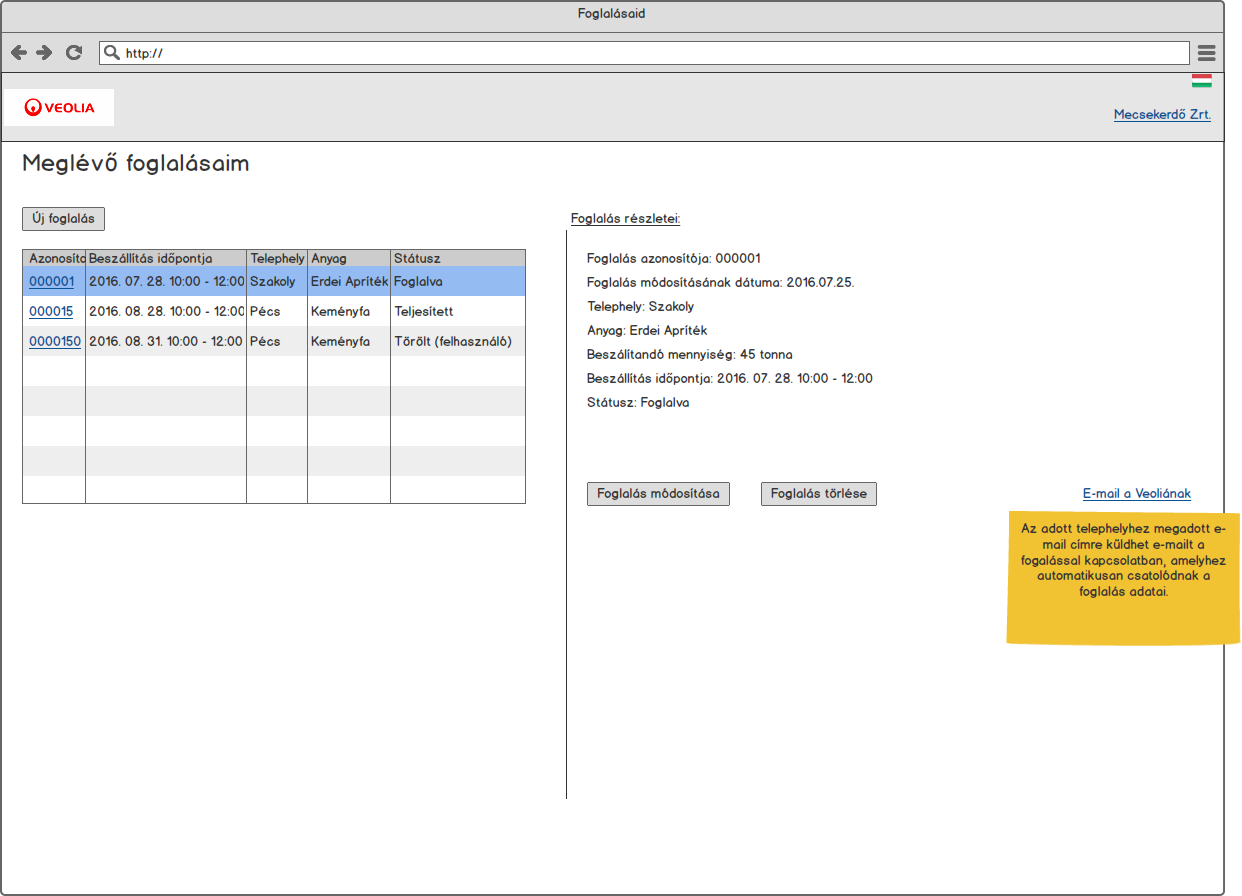
Miután rámentünk a küldés gombra, a rendszer megjeleníti újra az adatokat a foglalásról, és kiszámolja számunka, hogy a foglalás után a személyes limitünkből mennyi marad majd még. Amennyiben megfelelőnek találjuk a megjelenített értékeket, a foglalásunkat megerősíthetjük, vagy vissza is vonhatjuk.



10. ábra: Időpontfoglalás megerősítése.

## Foglalások kezelése

Az aktuális és korábbi foglalásainkat egy külön oldalon tekinthetjük meg, ahol egy táblázatban megjelenítjük az alapvető adatokat, amik a foglaláshoz tartoznak, mint az azonosít, időpont, telephely, anyag és a foglalásunk státusza, hogy épp aktuális-e még vagy már teljesítettük, esetleg törölve lett-e, amennyiben törölve lett, azt is megadjuk, hogy a törlés az adminisztrátor által történt, vagy a felhasználó által, aki a foglalást megtette. A foglalás részleteinél láthatjuk még a szállítani kívánt mennyiséget. Ezen a ponton tudunk módosítani a foglalásunkon is, vagy törölni azt. Rendelkezésre áll egy funkció, hogy E-mailt küldjünk a telephelynek, ahová a foglalást megtettük, és ehhez a foglalási adatokat automatikusan csatolja a rendszer.

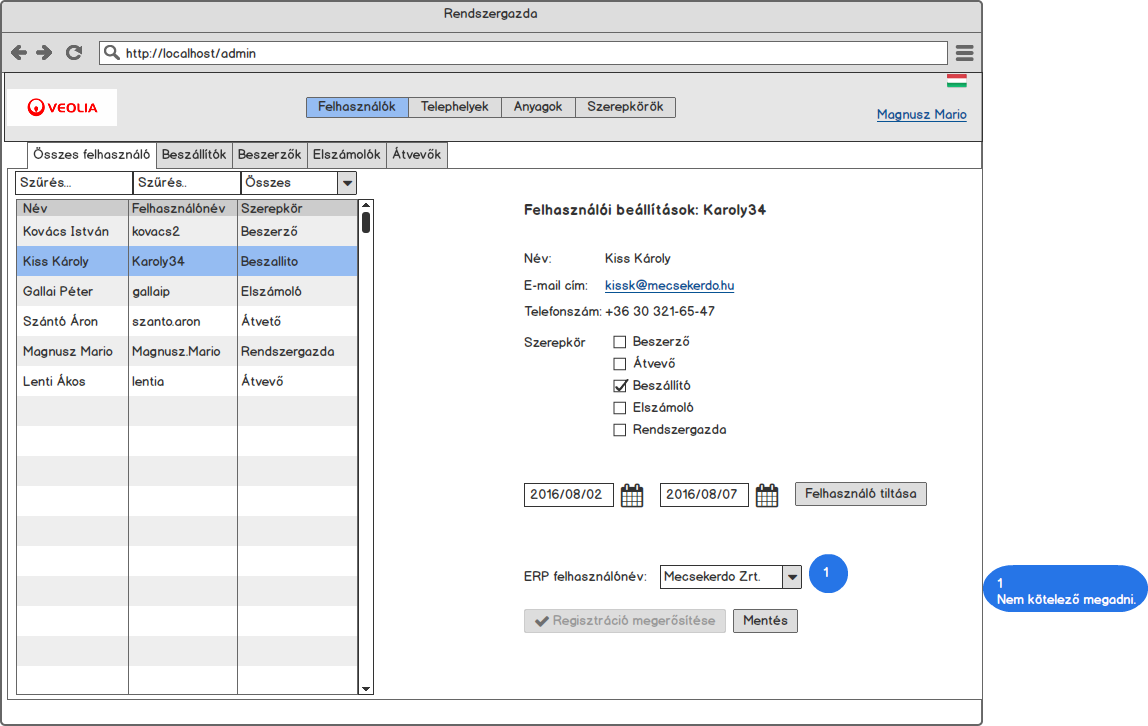


11. ábra: Saját foglalások nyilvántartása.

## A rendszergazda feladatai

### Felhasználói beállítások

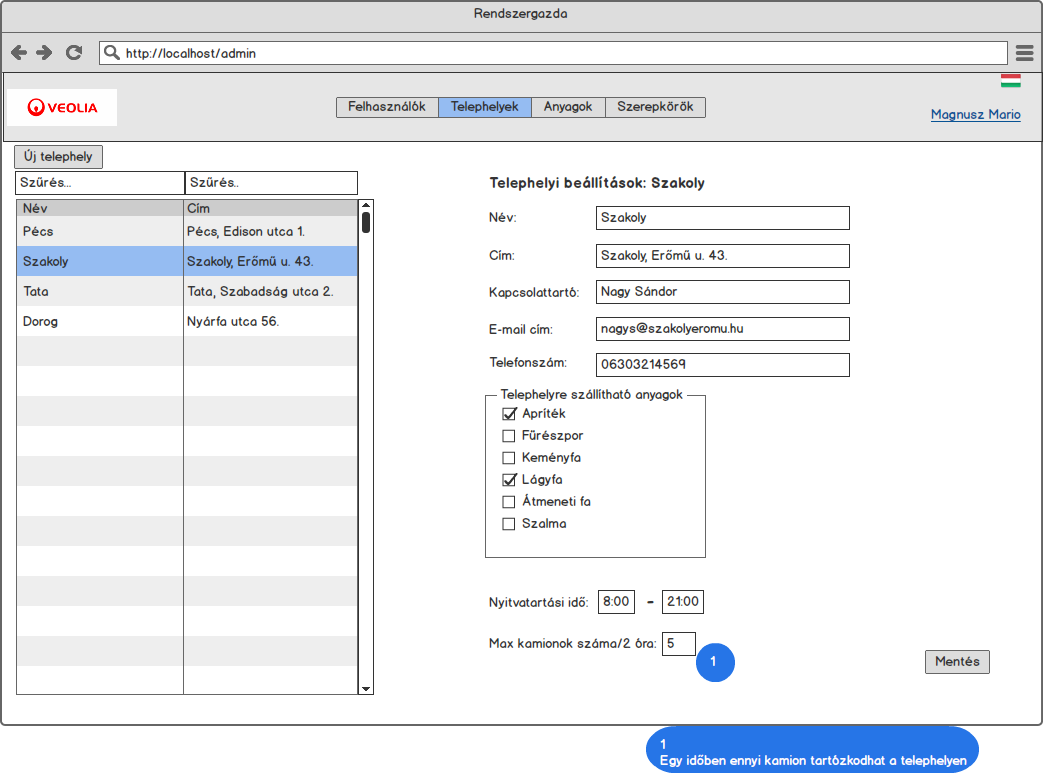
A rendszergazda számára elérhető az összes regisztrált személy adata. Az admin feladata, hogy megadja az adott felhasználó szerepkörét, vagy módosítsa azt. Itt erősítheti meg a felhasználók regisztrációját is.



12. ábra: Felhasználói adatok kezelése.

### Telephelyi beállítások

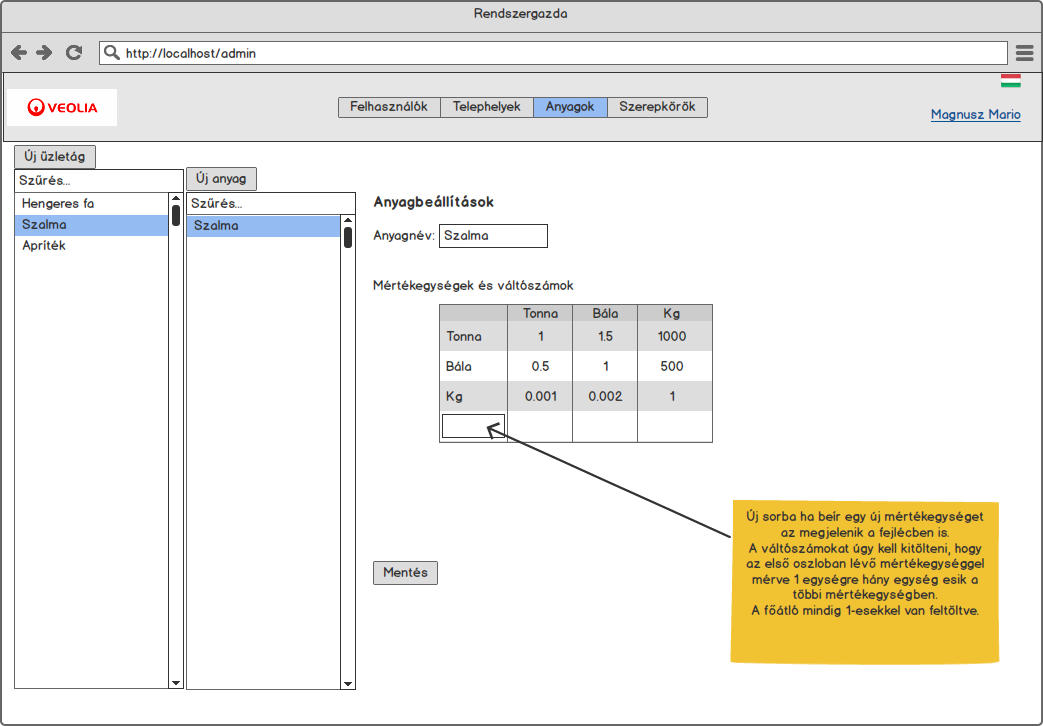
Az admin ugyanúgy hozzáfér a telephelyek adataihoz, mint a felhasználókhoz, itt megtudja adni az adott telephely nevét, címét, kapcsolat tartó nevét, e-mail címét és telefonszámát. Egy listában megadhatjuk a telephelyre szállítható anyagok típusait, nyitvatartási időt és a maximum kamionok számát, amik 2 óránként bemehetnek a telephelyre.



13. ábra: Telephely adatok kezelése.

### Anyagbeállítások

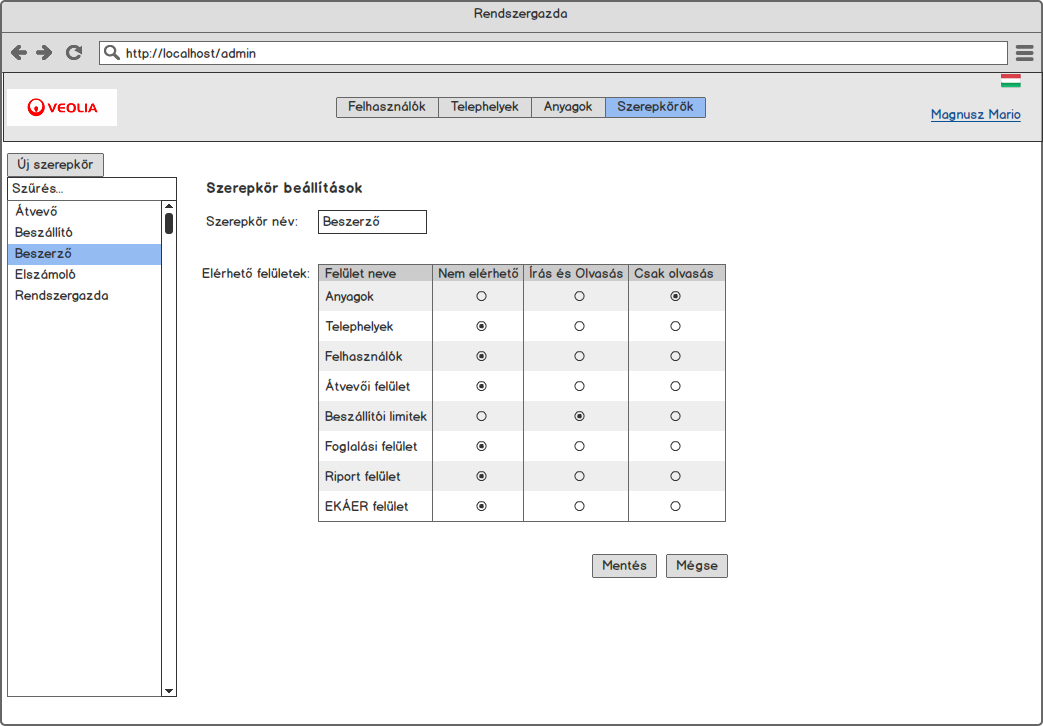
Az anyagok adatainak beállítása is a rendszergazda feladata, ebben a menüpontban tudunk új anyagot felvenni, és ehhez mértékegységeket és váltószámokat megadni. A táblázat automatikusan kitöltődik, és számolja nekünk a mértékegységek közti váltószámot. A meglévő anyagok adatainak módosítására is képesek vagyunk.



14. ábra: Anyagok adatainak kezelése.

### Szerepkör beállítások

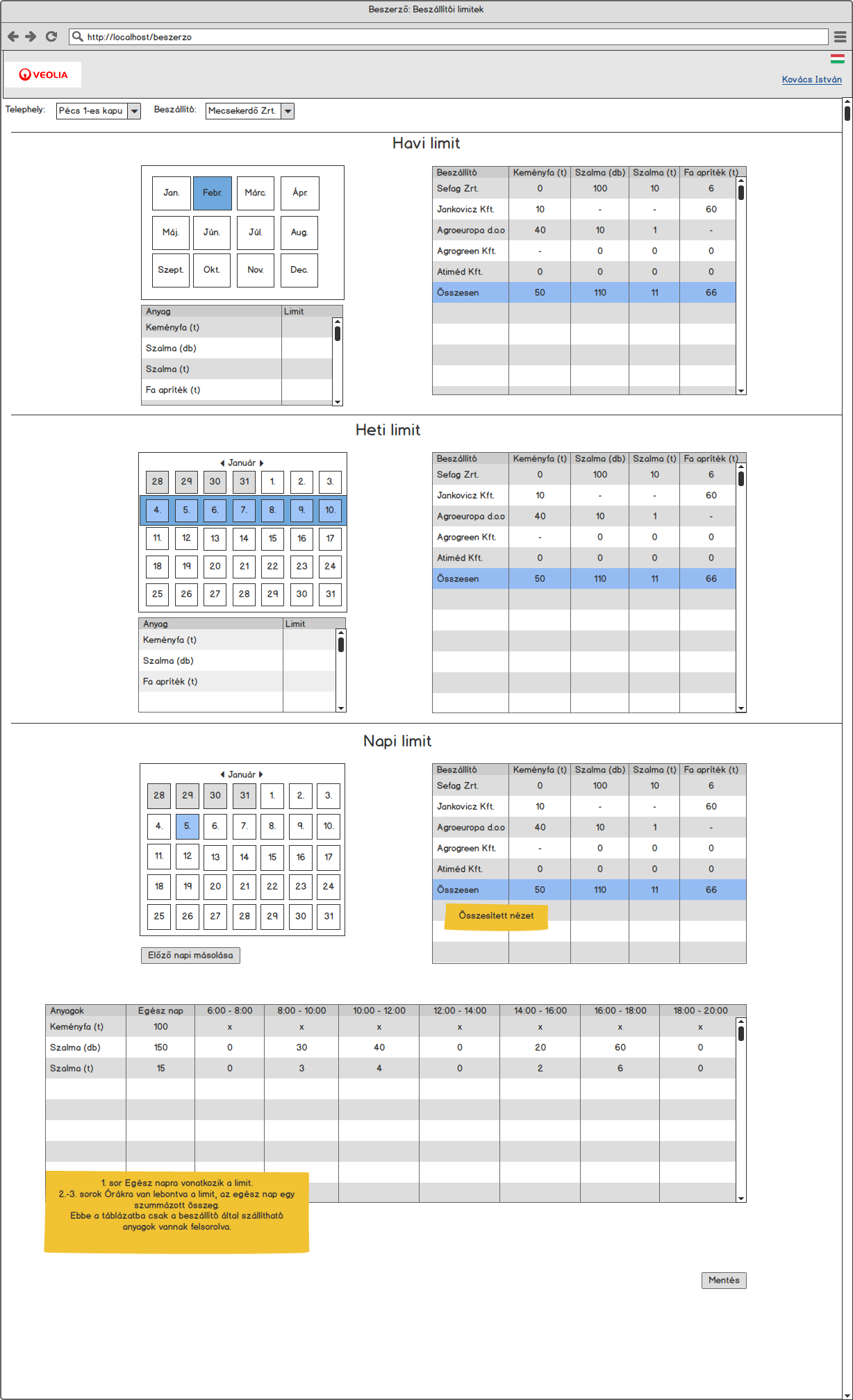
A rendszerben a felhasználók bizonyos szerepkörbe tartoznak, ezek a képen látható kategóriákba tudjuk sorolni. Minden szerepkörnek megvannak a maga jogosultságai, ezeket az adminisztrátor tudja módosítani. Bal oldali oszlopban található a szerepkörök listája, amihez új adatot is felvehetünk, jobb oldali táblában látható, hogy milyen felületek vannak, amikhez jogosultságot kell kiosztanunk.



15. ábra: Szerepkör adatok kezelése.

## Limitek beállítása

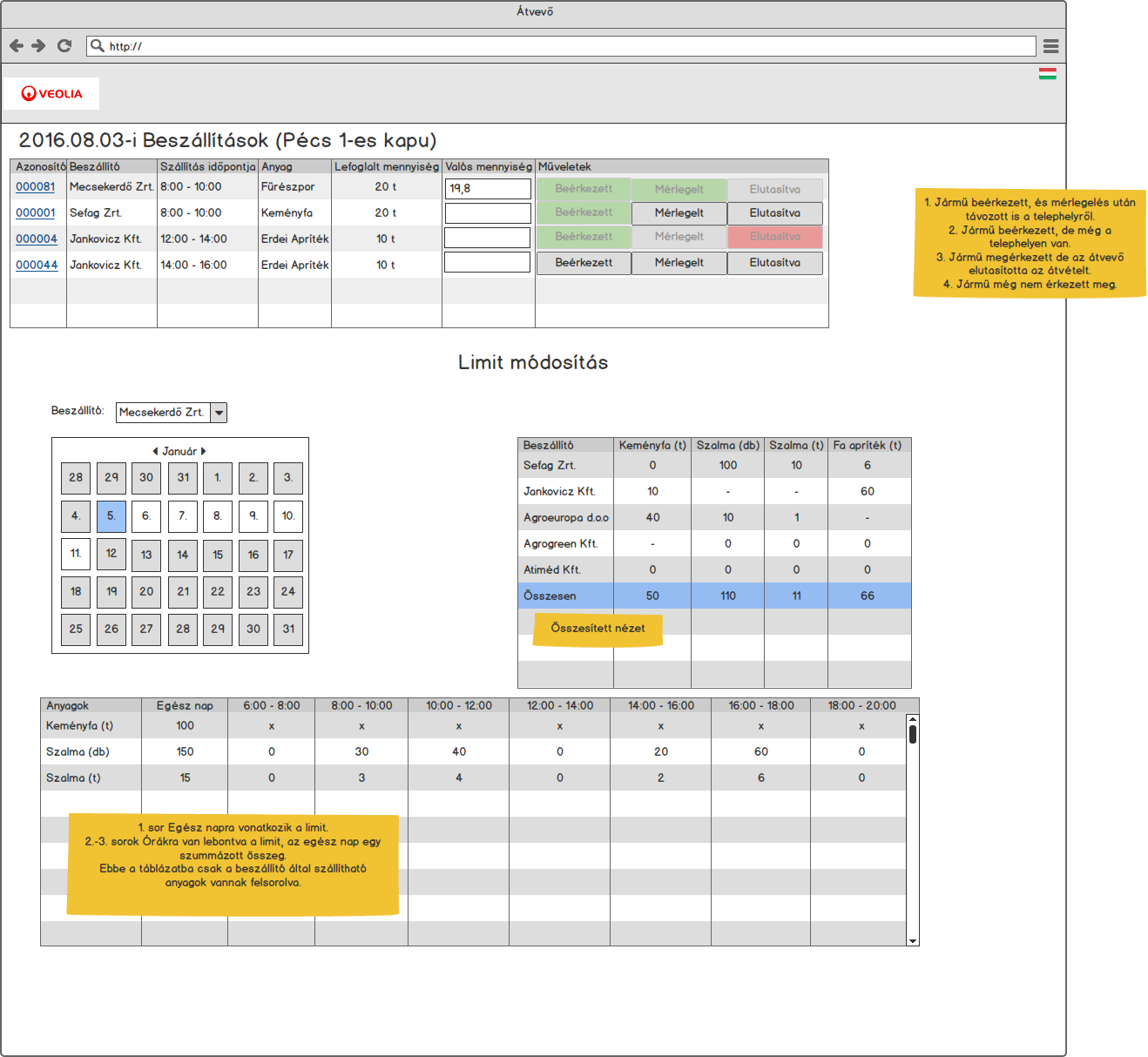
A limiteknél a beszerző állíthatja be, hogy a saját telephelyéhez tartozó területre mik a havi, heti és a napi limitek az adott beszállítóknak. A limitek megadhatók minden egyes anyag típusra. A jobb oldali táblázatban felsorolja a cégeket és a hozzájuk tartozó limitet, majd az alsó sorban kimutat egy összesített értéket is.



16. ábra: Limitek módosítása.

## Beszállítás információk

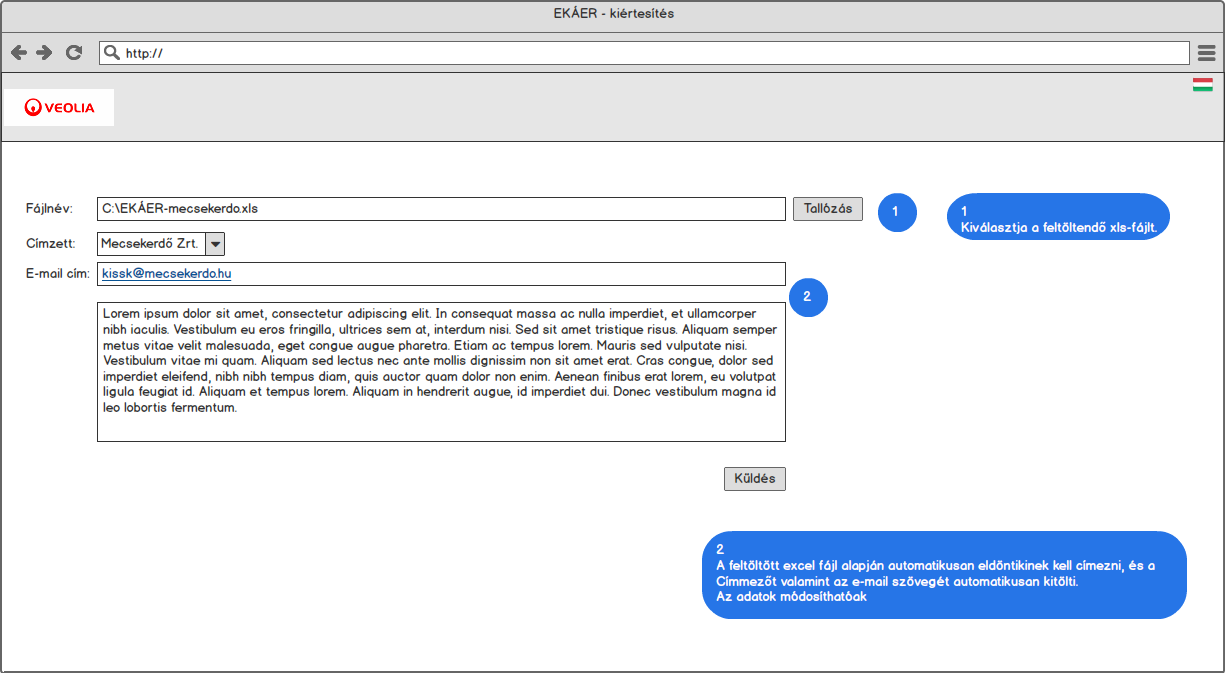
A beszállításokról eltároljuk az adatokat, hogy az adott beszállítás beérkezett-e, beérkezés után megtörtént-e a mérlegelés, és ha esetlegesen a szállítás elutasításra került, akkor azt is itt jelezi a rendszer. Ezt a felületet az átvevő kezeli. Eltárolja a táblázat, hogy mekkora mennyiséget foglalt le az adott beszállító és miután ezt mérlegelték, kiírják, hogy mi volt a tényleges súly a szállítmánynak



17. ábra: Áruátvétel nyilvántartása.

## Kiértesítés

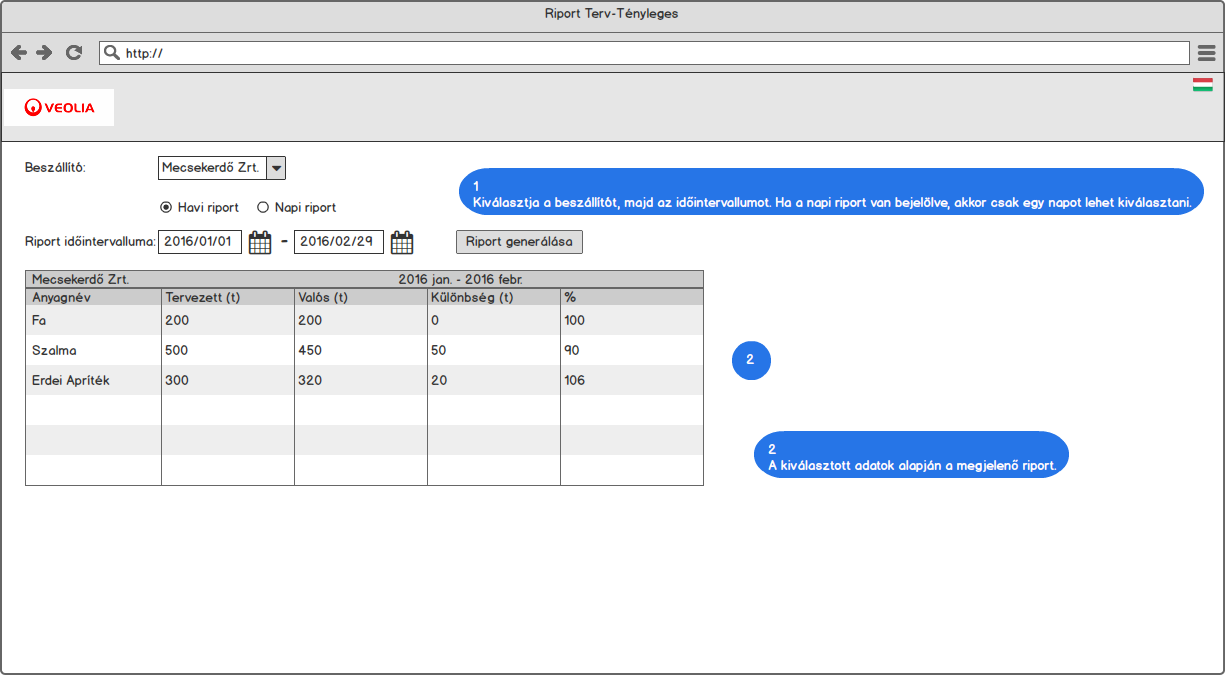
A kiértesítés e-mail küldési lehetőség úgy működik, hogy kiválasztunk egy xls file-t, ami tartalmazza a címzett nevét és e-mail címét, ezekkel az adatokkal a rendszer automatikusan kitölti a mezőket. Ezeket az adatokat a későbbiekben módosíthatjuk, ha szeretnénk, majd írhatunk egy üzenetet, amit el is küldhetünk a megadott címre.



18. ábra: Kiértékelés.

## Riport készítése

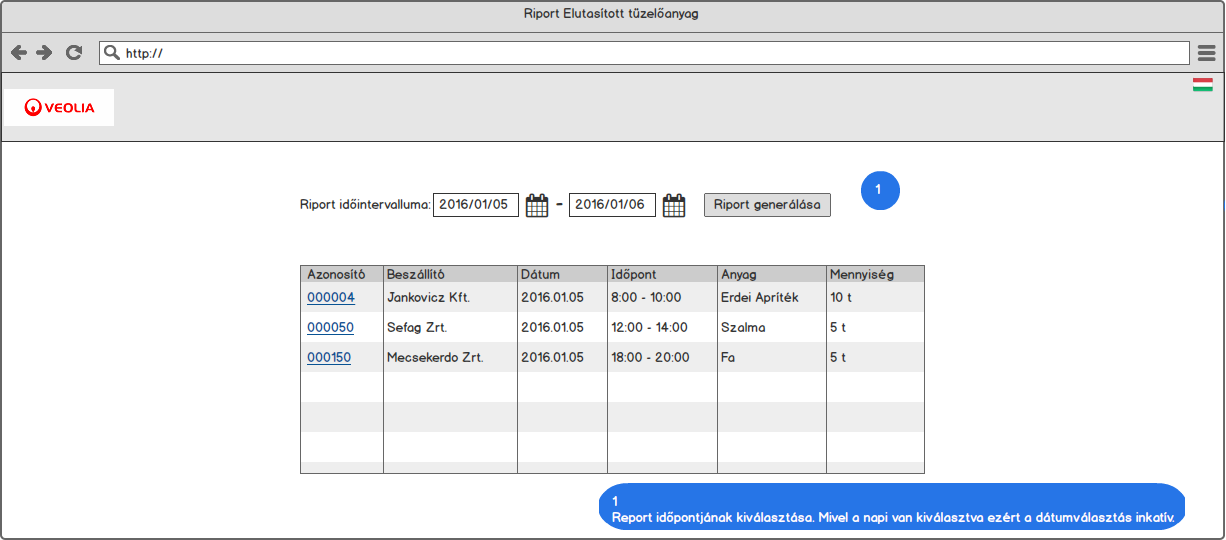
Készíthetünk riportot egy beszállítóról, ez lehet havi vagy napi riport, ha havi riportot készítünk, akkor meg kell adni neki egy idő intervallumot, ha napi riportot, akkor csak egy napot kell kiválasztani. A riport tartalmazza, hogy a beszállító milyen árut szállított be és hogy ehhez mekkora volt a tervezett szállítani kívánt mennyiség és mennyi volt a valós mennyiség, amit meg is kapott a telephely, majd egy kiértékelésben megadjuk a %-ot, amivel kimutatjuk, hogy mennyire megbízható a beszállító.



19. ábra: Riport készítés.

## Elutasított beszállítások

Készíthetünk riportot az elutasított tüzelőanyagról is, amikor a szállító nem jó anyagot szeretett volna hozni, vagy a mennyiség nagyban eltért a lefoglalt mennyiségtől.



20. ábra: Elutasított tüzelőanyagok.

# Adatbázis terv

## Entitások és relációk

Itt adatbázis független módon kerülnek felsorolásra az Entitások (felhasználó, szerepkörök, stb..). Az Entitások közötti Kapcsolat (n:1, 1:n, n:n).

A ‘users’ és a ‘roles’ között N:N kapcsolat áll fent, mivel 1 felhasználó több szerepkörbe is tartozhat, és egy szerepkörhöz több felhasználó is tartozhat.

A ‘users’ és az ‘appointments’ tábla között 1:N kapcsolat van, mivel 1 felhasználóhoz több időpont is tartozhat, de egy időpont maximum egy emberé lehet.

## Adatbázis specifikus modell

Táblák: (PK- elsődleges kulcs, AI – autoincrement, NN – nem nulla)

2. táblázat - felhasználók DB tábla

|  |  |  |
| --- | --- | --- |
| users | | |
| ID | INT | PK, AI, NN |
| name | VARCHAR(45) | NN |
| username | VARCHAR(45) | NN |
| password | VARCHAR(45) | NN |
| phone | VARCHAR(45) | NN |
| e\_mail | VARCHAR(45) | NN |

Ez a tábla tartalmazza a felhasználók adatait. (név/cégnév, felhasználónév, jelszó, telefonszám, e-mail cím).

3. táblázat - szerepkörök DB tábla

|  |
| --- |
| roles |
| ID INT PK, AI, NN |
| role VARCHAR(30) NN |
| material INT NN |
| sites INT NN |
| users INT NN |
| recipient\_ui INT NN |
| supplier\_limit INT NN |
| booking\_ui INT NN |
| report\_ui INT NN |
| EKAER\_ui INT NN |

Ez a tábla tartalmazza minden egyes szerepkör nevét és az oldalon elérhető összes felületet. A felületekhez tartozik egy szám (1-3), ami meghatározza, hogy az adott szerepkör milyen jogosultsággal rendelkezik ehhez a felülethez.

1: Nem elérhető

2: Csak olvasás

3: Írás és olvasás

4. táblázat - felhasználó és szerepkör összekötő DB tábla

users\_roles

ID INT PK, AI,NN

user\_id INT NN

role\_id INT NN

A “users\_roles” tábla, ahogy azt a neve is mutatja, kapcsolja össze a users és a roles táblánkat. Minden elem tartalmaz egy ID-t és egy user\_id role\_id párt, ami megadja, hogy az adott ID-val rendelkező user, melyik ID-val ellátott szerepkörhöz tartozik.

5. táblázat - időpontok DB tábla

|  |  |  |
| --- | --- | --- |
| appointments | | |
| ID | INT | PK, AI, NN |
| date | DATE | NN |
| appointment\_start | TIME | NN |
| appointment\_end | TIME | NN |
| weight | INT | NN |
| number\_of\_trucks | INT | NN |

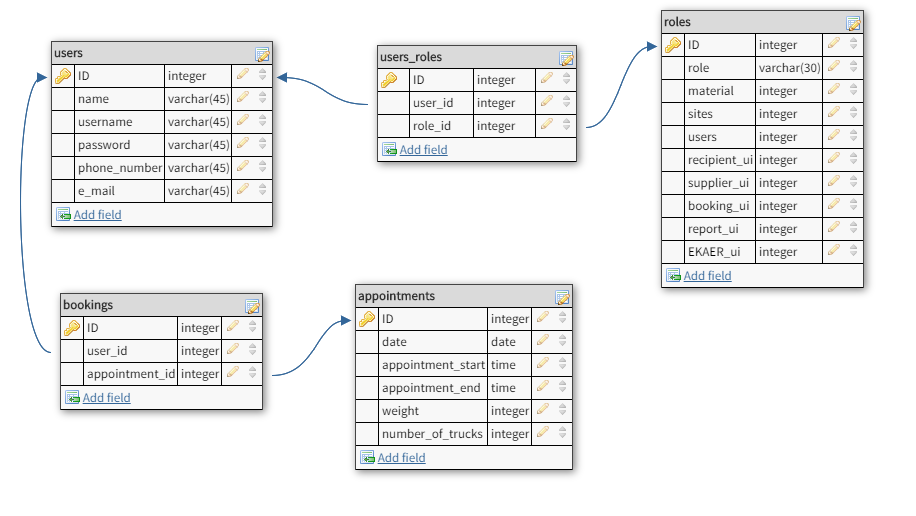
Az “appointments” tábla fogja eltárolni a létrehozott időpontokat, amiket később le lehet foglalni. A date tárolja a napot, amikor az időpont van, az appointment\_start és az appointment\_end oszlopok határozzák meg, hogy az adott időpont mettől meddig tart. A weight adattagok mondják meg, hogy aznap hány tonnát lehet beszállítani az adott termékből, és ezt hány kamionnal (number\_ of\_trucks) lehet megtenni.

6. táblázat - foglalások DB tábla

|  |  |  |
| --- | --- | --- |
| bookings | | |
| ID | INT | PK, AI, NN |
| user\_id | INT | NN |
| appointment\_id | INT | NN |

A bookings táblában tároljuk el, hogy melyik időpont melyik felhasználóhoz tartozik, azok az időpontok, amiknek az azonosítója bekerült ebbe a táblába, többé már nem érhetőek el senki számára, így azokat a rendszer már nem fogja megjeleníteni.

## Adatbázis modell (részlet)

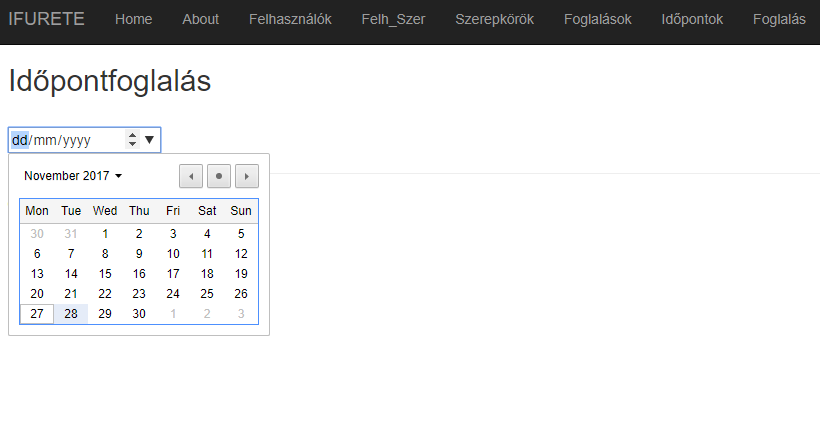


21. ábra - Adatbázis részlet

A fenti kép ábrázolja, a programomban felhasznált adatbázis modelljének egy részletét, itt látható a kapcsolat a táblák között, táblák nevei, és azoknak adattagjai.

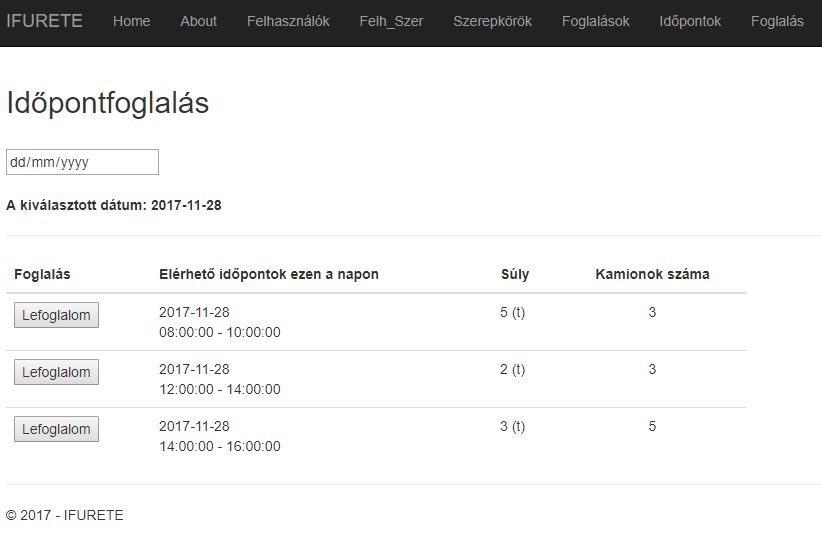
# Aktuális program

A program a mostani formájában, csak egy kezdetleges demo, ami képes adatokat eltárolni az adatbázis tervben látott táblákhoz, és ezeket az adatokat képes módosítani egy grafikus interface segítségével. Ezeket a funkciókat szeretném bemutatni néhány kép segítségével. Az alábbi képen látható az időpont foglalás első lépése, amikor kiválasztjuk a naptárban a számunkra megfelelő napot, hogy itt elérhető foglalásokat nézhessünk meg. A felső sorban látható menüpontok segítségével navigálhatunk a különböző táblákhoz tartozó, adattagokat kezelő felületekhez.



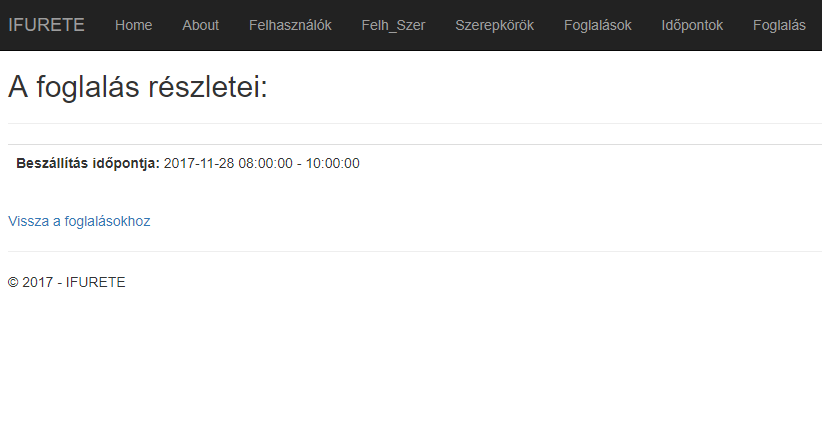
22. ábra - aktuális program: dátum kiválasztás

A következő kép megmutatja, hogy hogyan is néz ki a felület, miután már kiválasztottuk a számunkra megfelelő dátumot. A program kilistázza az elérhető időpontokat, amit már lefoglaltak az nem jelenik meg, ezekhez az időpontokhoz tartozik egy leírás, ami megadja az időintervallumot, a szállítható anyag súlyát és a maximális kamionok számát. A dátumot bármikor szabadon változtathatjuk.



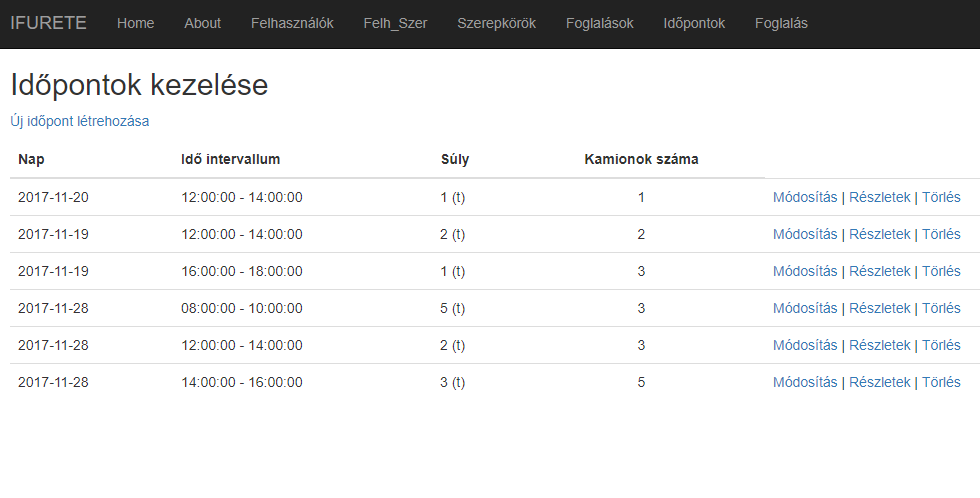
23. ábra - aktuális program: időpont foglalás

A „Lefoglalom” gomb lenyomása után, az oldal átdob minket egy új fülre, ami kiírja a foglalás adatait, későbbiekben, amikor már meglesz a bejelentkezés funkció és a telephelyek is rendelkeznek adatbázissal, akkor részletesebb információt fogunk kapni magáról a foglalásról, egyelőre nincs túl sok megjeleníthető adattag.



24. ábra - aktuális program: foglalás adatainak részletezése

Az időpontok kezelésére szolgáló felületen módosíthatjuk az időpontok tábla adatait, fel vehetünk újat, de akár törölhetjük a meglévőket, vagy módosíthatjuk azokat. A későbbiekben ehhez csak a telephelyeknek és az adminnak lesz hozzáférése.



25. ábra - aktuális program: foglalásaink megjelenítése

# Összefoglalás

A probléma leírásánál megfogalmaztam, hogy mi is a kiindulási alap, amit meg kell oldanom, és ezt milyen rendszer segítségével szeretném megtenni a későbbiekben. Ehhez végeztem el egy kutatást, hogy milyen hasonló programok találhatók az interneten, és ezeket összevetettem azzal, hogy mi az, amit az én programomnak el kell végeznie és mi az, amit ezek a különböző rendszerek képesek nyújtani számomra.

Eldöntöttem, hogy az általam létrehozni kívánt rendszert, milyen technológiák segítségével szeretném megvalósítani, leírom, hogy miért ezeket választottam, és ezeknek a technológiáknak is, mely fajtáját szeretném használni a munkám során.

A létrehozott grafikai tervek segítségével, betekintést nyújtok abba, hogy hogyan is tervezem elrendezni a programon belüli kezelőfelületeket, és ezek hogyan működnének majd a valóságban. A megjelenítés tervei mellett rendelkezem egy adatbázis tervvel is, ami ábrázolja, a még nem teljesen kifejlett, adatbázisrendszerem tervét, ami töredéke annak, amit a későbbiekben szeretnék majd használni.

Képekkel és részletes leírással mutatom be az eddig elkészült program demóját, ennek működését és megjelenését. Később ezt veszem majd alapul a teljes rendszer megalkotásához.

A jövőképben megfogalmazom, hogy eddig pontosan mit is értem el a projektemmel, és ezt milyen módon és pontosan hogyan is szeretném később fejleszteni, mindig van hova fejlődni, így a jövőképben leírt dolgok nem feltétlenül tükrözik a tökéletes végleges változatot, de egy remek kiindulási alapot ad, ami segít eldönteni, hogy milyen irányban is érdemes haladnom a rendszerem fejlesztése során.

A dokumentáció végén feltűntetett irodalomjegyzék tartalmazza, hogy mely oldalak szolgáltak segítségül a felkészüléshez, és melyik könyveket érdemes olvasni, ha szeretnék fejlődni az ASP.NET-hez kapcsolódó nyelvek használatában.

# Irodalomjegyzék

[1]https://en.wikipedia.org/wiki/ASP.NET

[2]https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/

[3]<https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C5%91>

[4]<https://hu.wikipedia.org/wiki/C_Sharp>

[5]<https://en.wikipedia.org/wiki/HTML>

[6] O’Reilly Media – Programing C#

[7] O’Reilly Media – Modern JavaScript

[8] O’Reilly Media - HTML: The Definitive Guide