# 

**PipeLine**

Töltőállomás-kezelő szoftverrendszer

Készítők:

Gavlik Martin István

Gavrán Tamás

Kónya Milán

## 1. Bevezetés

### 1.1. Az ötlet és probléma

Mivel egyre népszerűbbek az alternatív közlekedési formák, mint például kisebb elektromos közlekedési eszközök, mint például e-rollerek, e-biciklik és e-gördeszkák. Ezen eszközök egy környezetbarátabb alternatívát kívánnak a közlekedés terén, hiszen nincs káros gáz-kibocsájtásuk. Ezen felül tárolásuk és “parkolásuk” nem foglal olyan nagy helyet mint egy autóé, de mégis biztosít egy gyorsabb, lendületesebb közlekedést, a nem elektromos változatukhoz képest.

Mindezen pozitívumok ellenére a lítium-ion akkumulátor nagy hátránya nincs orvosolva, mégpedig az, hogy a hatótávolságuk igazán nagy kellemetlenséget okozhat, ezzel elriasztva potenciális érdeklődőket.

Jelenleg kevés lehetőség van az ilyen eszközök nyilvános töltésére, és nincs egységes rendszer, amely lehetővé tenné ezek kezelését.

### 1.2 A projekt célja

A “PipeLine” projekt célja ezen a problémának a megoldása egy szoftverrendszerrel rendszerrel, amely:

* **feltételezi az ilyen töltőállomások létezését**
* lehetővé teszi nyilvános töltőállomások felvitelét az adatbázisba,
* kezeli a töltőportokat, az eszközöket és a töltési eseményeket,
* valós idejű visszajelzést ad a töltési állapotról,
* lehetőséget biztosít a felhasználóknak saját eszközeik nyilvántartására és töltésére,
* valamint az adminisztrátorok számára karbantartási, statisztikai és hibabejelentési funkciókat is kínál.

**A rendszer három különböző platformon elérhető:**

* **webes alkalmazásként**, amely a felhasználók számára biztosítja a rendszer alapfunkcióit,
* **mobilalkalmazásként**, Android eszközökre,
* valamint egy **desktop adminisztrációs felületen**, amely az adminisztrátoroknak ad jogosultságot a rendszer kezeléséhez.

### 1.3 Társadalmi hasznosság

A PipeLine hozzájárul az elektromos közlekedés népszerűsítéséhez és a fenntartható közlekedés terjedéséhez. A rendszer lehetőséget ad ezen eszközök töltését kezelni és nyomon követni. Az így csökkenő bizonytalansági érzet motiválhatja a lakosságot az elektromos közlekedésre való áttérésre, ezzel pedig hozzájárulhatunk a közlekedési kibocsátások csökkentéséhez és a városi életminőség javításához. Hiszen nő az eszközök használhatósági ideje és nem kell annyit aggódni a töltés miatt.

## 2. Architektúra

A PipeLine rendszer egy többrétegű architektúrára épül, így a rétegek elkülönülése miatt a kód könnyen átlátható, karbantartható és skálázható.

### 2.1. A rendszer rétegei

#### Backend

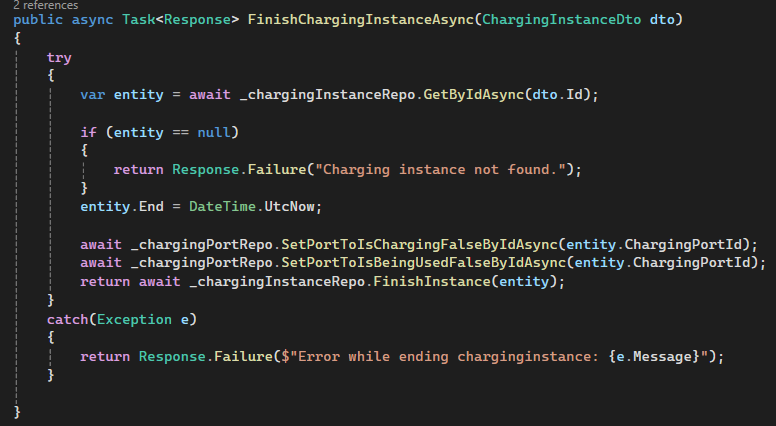
A szerveroldali alkalmazás (backend) ASP.NET Core keretrendszerre épül, REST API és WebSocket kapcsolatokon keresztül kommunikál a kliensoldali alkalmazásokkal (web,mobil és asztali alkalmazás). Az adatokat egy MySQL adatbázis tárolja és a felhasználók kezeléséhez AspNetCore.Identity csomagot használtunk.

**Az adatbázis típusa: relációs adatbázis.**

A relációs adatbázis olyan adatbázis-típus, amelyben az adatokat táblákban tároljuk, és ezek a táblák kapcsolatokkal (relációkkal) kapcsolódnak egymáshoz, ezekhez a kapcsolatokhoz idegen kulcsokat használunk.

**A backend az alábbi rétegekből áll:**

* **Controller**: HTTP végpontokat biztosít, fogadja a kéréseket, továbbítja a megfelelő szolgáltatások felé
* **Service**: üzleti logikát tartalmaz (pl.: ellenőrzések, több repository használata)
* **Repository**: az adatbázishoz való hozzáférést valósítja meg EF Core segítségével
* **Assembler**: DTO-kat és entitásokat konvertál
* **WebSocket Handler**: a valós idejű kommunikációt bonyolítja (töltési állapot frissítése)

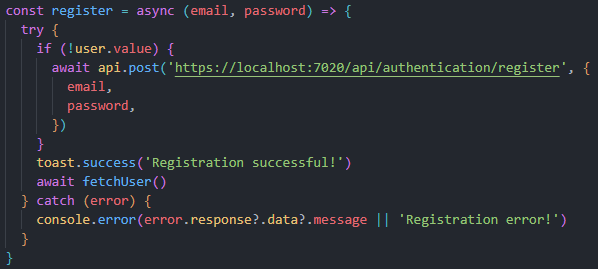


**Kép a töltési esemény service-rétegéből**

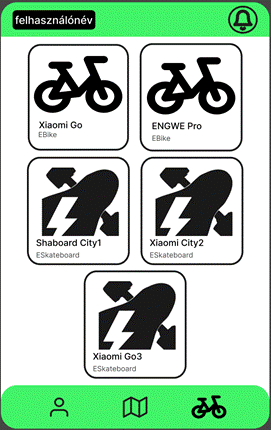
#### **Web- és Mobil alkalmazás**

A frontend Vue 3-ra épül, Composition API-t használva. Az állapotkezelést Pinia segítségével oldottuk meg, a megjelenéshez saját stílust-t és Bootstrapet alkalmaztunk.

* **Web**: Reszponzív, modern felület ami Vue 3-ra alapszik.
* **Mobil**: Az Ionic és Capacitor segítségével Vue-alapú, natívan fut Androidon.



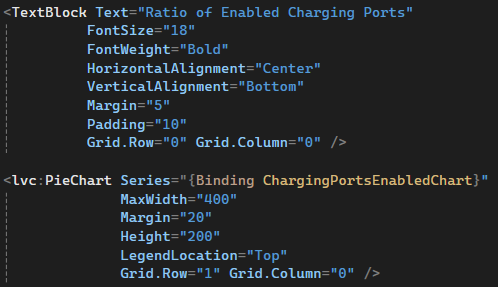
**A felhasználó store-jából a regisztráció a web-ből**



**A mobil applikáció dizájn terve**

#### **Asztali alkalmazás**

A desktop admin alkalmazás WPF-ben készült, .NET és MVVM architektúra szerint. Az adatok lekérése és küldése REST API-n keresztül történik, a httpservice projekten keresztül.



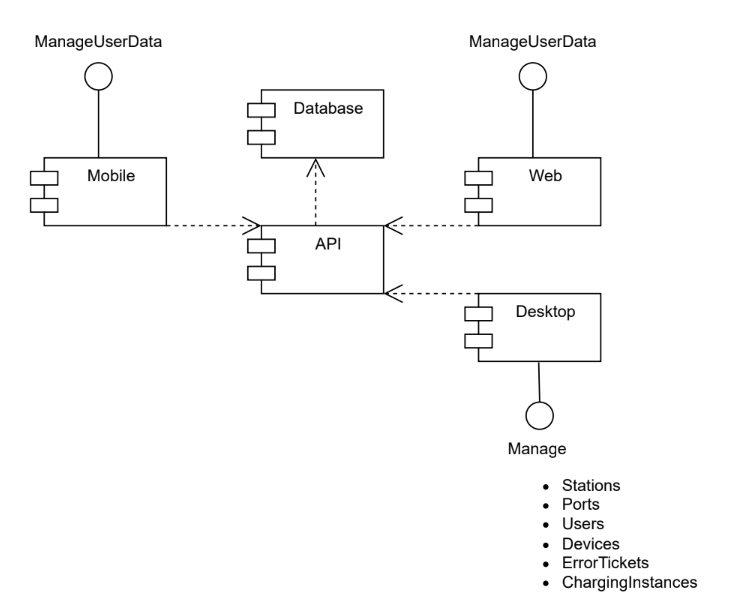
**Egy statisztika View-rétegi része**

#### **Shared és HttpService**

Ezen komponenseket más komponensek vagy komponens használja, önmagukban nem futtathatóak.

* **shared**: tartalmazza a DTO-kat, konvertereket, modelleket.
* **httpservice**: REST kapcsolatokat kezelő C# osztályok, amelyet a desktop használ.

### 2.2. A komponensek kapcsolódása

****

**UML komponens diagram**

### 2.3. Az architektúra előnyei

* Skálázhatóság: külön fejleszthető backend és frontend
* Bővíthetőség: egyszerűen hozzáadható új entitás, funkció, vagy kliensalkalmazás
* Tesztelhetőség: az egyes rétegek külön egységként tesztelhetők

## 3. Felhasználói dokumentáció

A PipeLine rendszer többféle felhasználó számára biztosít különböző funkciókat és kezelőfelületeket. Az alkalmazás célja, hogy a végfelhasználók könnyedén tudják használni a töltőállomásokat saját elektromos eszközeikhez, míg az adminisztrátorok hatékonyan tudják kezelni a rendszerben található adatokat és hibabejelentéseket.

### 3.1 Felhasználói szerepkörök

A rendszer két fő szerepkört különböztet meg:

* **Regisztrált felhasználó** – az applikáció felhasználója, aki e-rollerét, e-bike-jét vagy más eszközét regisztrálja, és használja a töltőállomásokat.
* **Adminisztrátor** – karbantartásért és felügyeletért felelős személy, aki a desktop alkalmazás segítségével teljes hozzáféréssel rendelkezik a rendszerhez.

### 3.2 Web és mobilalkalmazás funkciói

A regisztrált felhasználók a PipeLine webes vagy mobilalkalmazásán keresztül az alábbi funkciókat érhetik el:

* Kezdőlap megtekintése (weboldal)
* Információk a projektről (weboldal)
* Regisztráció, bejelentkezés
* Eszköz hozzáadása, módosítása, törlése
* Töltőállomások megtekintése térképen, szűrése, bejelentett problémáik megtekintése
* Töltőállomások csatlakozóinak megtekintése, szűrése
* Töltési esemény indítása, követése és befejezése
* Hibabejelentés
* Felhasználói adatok módosítása

### 3.3 Web és mobilalkalmazás forgatókönyvei

| Munkafolyamat neve | Ki (kik végzik)? | Lépések | Hibalehetőségek,  speciális dolgok |
| --- | --- | --- | --- |
| Új felhasználó regisztrálása | Felhasználó | 1. Bejelentkezés menüről a regisztrációra továbbhaladás 2. Email és jelszó megadása 3. Regisztrálás gomb | * Az alkalmazás nem fogad el nem bizonyos kritériumoknak megfelelő jelszót és emailt |
| Bejelentkezés | Felhasználó | 1. Bejelentkezés menü kiválasztása vagy egy másik pontról átdobja a felhasználót ha az nincs bejelentkezve és használni akar olyan funkciót amihez ez szükséges 2. Email és jelszó megadása 3. Bejelentkezés gomb | * Az alkalmazás nem fogad el nem bizonyos kritériumoknak megfelelő jelszót és emailt |
| Eszköz hozzáadása | Felhasználó | 1. Eszközök menüpont kiválasztása 2. Új eszköz opció 3. Adatok kitöltése 4. Mentés gomb | * Amennyiben nem ad meg minden adatot, a töltési állapot követése nem lesz elérhető |
| Eszköz módosítása | Felhasználó | 1. Eszközök menüpont kiválasztása 2. (Meglévő eszközre keresés és) Eszköz kiválasztása 3. Adatok megváltoztatása 4. Mentés gomb | * Amennyiben nem ad meg minden adatot, a töltési állapot követése nem lesz elérhető |
| Eszköz törlése | Felhasználó | 1. Eszközök menüpont kiválasztása 2. (Meglévő eszközre keresés és) Eszköz kiválasztása 3. Törlés gomb | * Töltési előzmények követése nehézkesebb lesz |
| Töltési esemény indítása | Felhasználó | 1. Töltőállomás kiválasztása 2. Csatlakozó kiválasztása 3. Eszköz kiválasztása és információk megadása 4. Töltés indítása | * A töltés indítása előtt megadható eszköz akkumulátorának csak akkor van relevanciája ha a kiválasztott eszköznek minden adatát megadta a felhasználó |
| Töltési eseményt befejezése | Felhasználó | 1. Töltési események menüpontban megnyomja a befejezés gombot | * - |
| Hiba bejelentése | Felhasználó | 1. Állomás kiválasztása 2. Hibák menüpont kiválasztása 3. Bejelentés gomb 4. Elküldés | * Esetleg nem valós probléma,vagy ugyanaz a probléma többszöri bejelentése |

### 3.4 Asztali alkalmazás funkciói

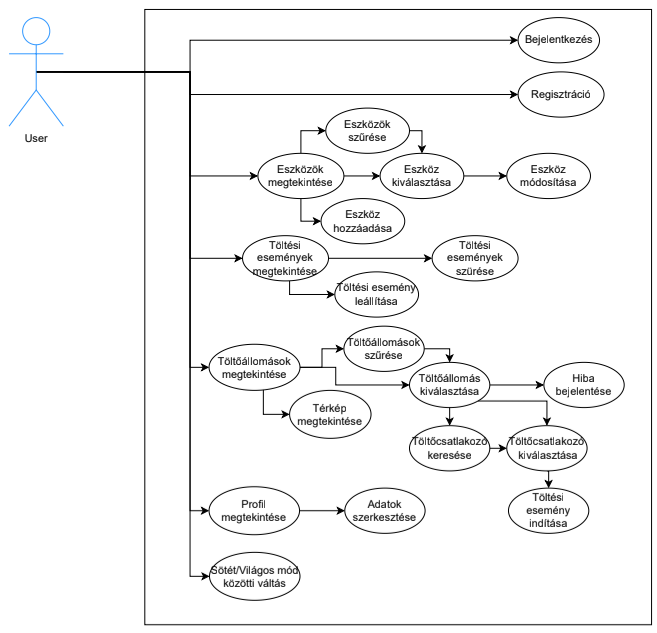
Az adminisztrátor egy különálló desktop alkalmazást használ (WPF alapokon), amely teljes hozzáférést biztosít a rendszer kezeléséhez. Ennek funkciói:

* Bejelentkezés adminisztrátorként
* Felhasználók kezelése
* Eszközök kezelése
* Töltőállomások és töltőcsatlakozók kezelése
* Hibajegyek kezelése
* Töltési előzmények megtekintése
* Statisztikák megtekintése (eszközök eloszlása, aktív portok)

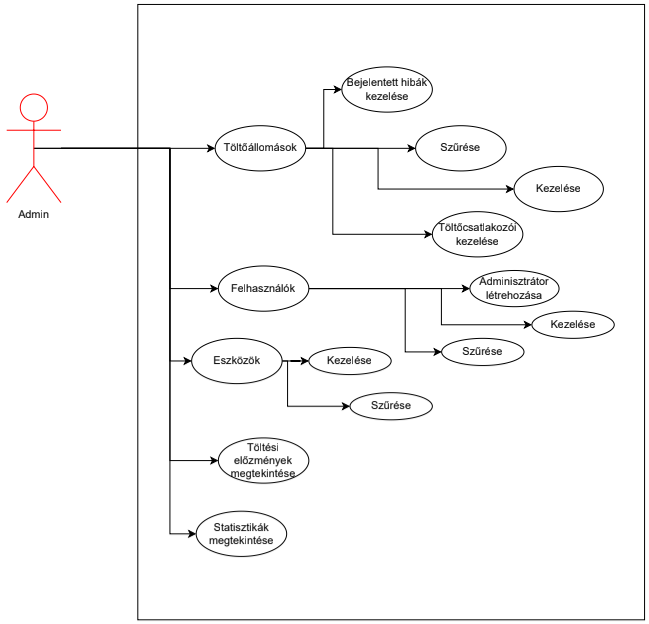
### 3.5 Asztali alkalmazás forgatókönyvei

| Munkafolyamat neve | Ki (kik végzik)? | Lépések | Hibalehetőségek,  speciális dolgok |
| --- | --- | --- | --- |
| Bejelentkezés az alkalmazásba adminként | Adminisztrátor | 1. E-mail cím megadása 2. Jelszó megadása 3. Bejelentkezés gombra kattintás | * Helytelenül megadott e-mail cím * Helytelenül megadott jelszó |
| Adminisztrátor hozzáadása | Adminisztrátor | 1. “Felhasználók” menüpont kiválasztása 2. “Új felhasználó létrehozása” almenüpont kiválasztása 3. Név, e-mail cím, felhasználónév, telefonszám megadása 4. Mentés | * Sikertelen belépés * Helytelenül megadott adatok |
| Felhasználó  /  Adminisztrátor törlése | Adminisztrátor | 1. “Felhasználók” menüpont kiválasztása 2. Felhasználó kiválasztása 3. Törlés | * Sikertelen belépés |
| Felhasználó adatainak változtatása | Adminisztrátor | 1. “Felhasználók” menüpont kiválasztása 2. Felhasználó kiválasztása 3. Módosítás 4. Mentés | * Sikertelen belépés * Módosítás során hibásan került megadásra adat |
| Eszköz keresése | Adminisztrátor | 1. “Eszközök” menüpont kiválasztása 2. Eszköz keresése tulajdonság alapján a keresősávban 3. Keresés gombra kattintás | * Sikertelen belépés * Nem létező eszközre való keresés |
| Eszköz adatainak módosítása | Adminisztrátor | 1. “Eszközök” menüpont kiválasztása 2. Eszköz kiválasztása 3. Módosítás 4. Mentés | * Sikertelen belépés * Módosítás során hibásan került megadásra adat |
| Eszköz törlése | Adminisztrátor | 1. “Eszközök” menüpont kiválasztása 2. Eszköz kiválasztása 3. Törlés | * Sikertelen belépés |
| Töltőállomás keresése | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. Töltőállomás keresése tulajdonság alapján a keresősávban 3. Keresés gombra kattintás | * Sikertelen belépés * Nem létező töltőállomásra való keresés |
| Töltőállomás hozzáadása | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. “Új töltőállomás hozzáadása” almenüpont kiválasztása 3. Név, koordináta megadása 4. Mentés | * Sikertelen belépés * Helytelenül megadott adatok |
| Töltőállomás törlése | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. Töltőpont kiválasztása 3. Törlés | * Sikertelen belépés |
| Töltőállomás adatainak változtatása | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. Töltőállomás kiválasztása 3. Módosítás 4. Mentés | * Sikertelen belépés * Módosítás során hibásan került megadásra adat |
| Hibabejelentések keresése | Adminisztrátor | 1. “Hibák” menüpont kiválasztása 2. Töltőállomás keresése a keresősávban vagy annak kiválasztása a legördülő menüből 3. Keresés gombra kattintás | * Sikertelen belépés * Nem létező töltőállomásra való keresés |
| Töltőállomás hibabejelentések kezelése | Adminisztrátor | 1. “Hibák” menüpont kiválasztása | * A hibajelentés nem biztos, hogy valódi |
| Töltőállomás csatlakozóinak megjelenítése | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. Töltőállomás kiválasztása 3. “Csatlakozók megjelenítése” gombra kattintás | * Sikertelen belépés * Csatlakozó nélküli állomás esetén a gomb nem jelenít meg csatlakozót |
| Töltő csatlakozó hozzáadása | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. “Új töltő csatlakozó hozzáadása” almenüpont kiválasztása 3. csatlakozóhoz tartozó állomás kiválasztása, Id, csatlakozó szám, maximális töltési idő megadása 4. Mentés | * Sikertelen belépés * Helytelenül megadott adatok |
| Töltő csatlakozó törlése | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. Töltő csatlakozó kiválasztása 3. Törlés | * Sikertelen belépés |
| Töltő csatlakozó adatainak változtatása | Adminisztrátor | 1. “Töltőállomások” menüpont kiválasztása 2. Töltő csatlakozó kiválasztása 3. Módosítás 4. Mentés | * Sikertelen belépés * Módosítás során hibásan került megadásra adat |

### 3.6 Használati eset diagram



**Felhasználói**



**Adminisztrátori**

## 4. Fejlesztői dokumentáció

### 4.1 Adatbázis

A PipeLine rendszer MySQL relációs adatbázist használ. Az adatbázis töltőállomások, a hozzájuk tartozó csatlakozók és hibajegyek, felhasználók és az ő eszközeik, valamint a töltési események tárolását valósítja meg.

#### **Egyedek**

Eszközök, töltőállomások, töltőcsatlakozók, felhasználók, töltési esetek, hibajegyek, felhasználó szerepei, szerepek, szerepek jogkörei, felhasználó egyedi tulajdonságai, külső szolgáltatóval történő bejelentkezések, bejelentkezéshez kapcsolódó autentikációs tokenek.

**FONTOS: Nem minden egyedet használunk mivel egy c# csomagot használunk a felhasználókhoz ami automatikusan létrehoz több táblát.**

#### Tulajdonságok

* chargingstations (Töltőállomások) (**Id**, *Latitude, Longitude, Name, Address*)
* chargingports (Töltőcsatlakozók) (**Id***, IsBeingUsed, IsDisabled, IsCharging, ChargingStationId, PortNumber, MaxChargingSpeed*)
* charginginstances (Töltési esetek) (**Id***, ChargingPortId, DeviceId, Start, End, StartingPercentage, EndPercentage, DesiredEndPercentage*) - kapcsolótábla a Az Eszköz és a Töltőport között
* devices (Eszközök) (**Id***, DeviceType, Manufacturer, Model, Name, ApplicationUserId, BatteryCapacity, BatteryVoltage, MaxChargingSpeed, DetachableBattery, IsFoldable, CanBeLocked*)
* errortickets (Hibajegyek) (**Id***, Description, IsSolved, ChargingStationId*)
* aspnetusers (Felhasználók) (**Id***, UserName, FirstName, LastName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount*)
* aspnetuserroles (Felhasználók és szerepkörök kapcsolata) (*UserId, RoleId*) - kapcsolótábla a Felhasználó és a Szerepkör tábla között
* aspnetroles (Szerepkörök) (**Id***, Name, NormalizedName, ConcurrencyStamp*)
* aspnetroleclaims (Szerepkör jogosultságok) (**Id***, RoleId, ClaimType, ClaimValue*)
* aspnetuserlogins (Felhasználó bejelentkezések) (*LoginProvider, ProviderKey, ProviderDisplayName, UserId*)
* aspnetusertokens (Felhasználói tokenek) (*UserId, LoginProvider, Name, Value*)
* aspnetuserclaims (Felhasználói adatok) (**Id***, UserId, ClaimType, ClaimValue*)

#### **Egyedek és tulajdonságaik (adattípusok):**

*chargingstations*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

Name szöveg (LONGTEXT) – a töltőállomás megnevezése

Address szöveg (LONGTEXT) – a töltőállomás címe

Longitude valós szám (DOUBLE) –az állomás hosszúsági koordinátája

Latitude valós szám (DOUBLE) – az állomás szélességi koordinátája

*chargingports*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

IsBeingUsed szám (TINYINT(1)) – Használatban van-e a csatlakozó

IsDisabled szám (TINYINT(1)) – Használható-e a csatlakozó

IsCharging szám (TINYINT(1)) – Éppen tölt-e a csatlakozó

ChargingStationId szöveg (VARCHAR(256))- idegen kulcs

PortNumber szám (INT(11)) – Csatlakozó száma

MaxChargingSpeed valós szám (DOUBLE) – Maximális töltési sebesség (kW)

*charginginstance*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

DeviceId szöveg (VARCHAR(256)) – Melyik eszközhöz tartozik (idegen kulcs)

ChargingPortId szöveg (VARCHAR(256)) – Melyik töltőponthoz tartozik (idegen kulcs)

Start dátumidő (DATETIME(6)) – A töltés kezdésének időpontja

End dátumidő (DATETIME(6)) – A töltés befejezésének időpontja

StartingPercentage szám (INT(11)) – Kezdő töltöttségi szint

EndPercentage szám (INT(11)) – Végső töltöttségi szint

DesiredEndPercentage szám (INT(11)) – Elérni kívánt töltöttségi szint

*devices*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

DeviceType szám (INT(11)) – Eszköz típusa (e-bicikli, e-gördeszka, e-roller)

Manufacturer szöveg (LONGTEXT) – Eszköz márkája/gyártója

Model szöveg (LONGTEXT) – Eszköz modellje

Name szöveg (VARCHAR(100)) – Eszköz neve

ApplicationUserId szöveg (CHAR(36)) – Eszköz tulajdonosa (idegen kulcs)

BatteryCapacity valós szám (DOUBLE) – Akkumulátor kapacitása (mAh)

BatteryVoltage valós szám (DOUBLE) – Akkumulátor feszültsége (V)

MaxChargingSpeed valós szám (DOUBLE) – Eszköz maximális töltési sebessége (kW)

DetachableBattery BOOLEAN – e-bicikli specifikus, kiszedhető-e az akkumulátor

IsFoldable BOOLEAN – e-roller specifikus, összehajtható-e

CanBeLocked BOOLEAN – e-gördeszka specifikus, lezárható-e

*errortickets*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

Description szöveg (LONGTEXT) – A hiba leírása

IsSolved szám (TINYINT(1)) – A hiba megoldott-e

ChargingStationId szöveg (CHAR(36)) – melyik töltőállomáshoz tartozik

*aspnetusers*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

UserName szöveg (VARCHAR(256)) – Felhasználó által megadott felhasználónév, egyedi

FirstName szöveg (LONGTEXT) – Felhasználó keresztneve

LastName szöveg(LONGTEXT) – Felhasználó vezetékneve

NormalizedUserName szöveg (VARCHAR(256)) – Felhasználó által megadott felhasználónév, nagybetűkkel

Email szöveg (VARCHAR(256)) – a felhasználó által megadott e-mail cím

NormalizedEmail szöveg (VARCHAR(256)) – a felhasználó által megadott e-mail cím, nagy kezdőbetűkkel

EmailConfirmed szám (TINYINT(1)) – Felhasználónak igazolt-e az e-mail címe

PasswordHash szöveg (LONGTEXT) – Felhasználó által megadott jelszó titkosítva

SecurityStamp szöveg (LONGTEXT) – Érzékeny adata módosításánál változik

ConcurrencyStamp szöveg (LONGTEXT) – Módosítási bélyeg amely minden adatváltoztatás után változik

PhoneNumber szöveg (LONGTEXT) – Felhasználó telefonszáma

PhoneNumberConfirmed szám (TINYINT(1)) – Felhasználónak visszaigazolt-e a telefonszáma

TwoFactorEnabled szám (TINYINT(1)) – Felhasználó rendelkezik-e kétfaktoros hitelesítéssel

LockoutEnd dátumidő (DATETIME) – Kitiltás időpontjának vége

LockoutEnabled szám (TINYINT(1)) – Felhasználótól megtiltható-e a bejelentkezés

AccessFailedCount szám(INT) – Felhasználó sikertelen bejelentkezéseinek száma

*aspnetuserroles*

UserId szöveg (CHAR(36)) – idegen kulcs

RoleId szöveg (CHAR(36)) – idegen kulcs

*aspnetroles*

**Id** szöveg (CHAR(36)) – elsődleges kulcs

Name szöveg (VARCHAR(256)) – Szerepkör megnevezése

NormalizedName szöveg (VARCHAR(20)) – Szerepkör neve, nagybetűs

ConcurrencyStamp szöveg (LONGTEXT) – Módosítási bélyeg amely minden adatváltoztatás után változik

*aspnetroleclaims*

**Id** szám (INT(11)) – elsődleges kulcs

RoleId szöveg (CHAR(36)) – Felhasználó azonosítója

(idegen kulcs)

ClaimType szöveg (LONGTEXT) – Jogkör típusa

ClaimValue szöveg (LONGTEXT) – Jogkör értéke

*aspnetuserlogins*

UserId szöveg (CHAR(36)) – idegen kulcs

LoginProvider szöveg (VARCHAR(255)) – Bejelentkezéshez használt szolgáltató

ProviderKey szöveg (VARCHAR(255)) – Bejelentkezéshez használt szolgáltató azonosítója

ProviderDisplayName szöveg (LONGTEXT) – Bejelentkezéshez használt szolgáltató neve

*aspnetusertokens*

UserId szöveg (CHAR(36)) – idegen kulcs

LoginProvider szöveg (VARCHAR(255)) – Tokenhez tartozó szolgáltató

Name szöveg (VARCHAR(255)) – Token neve

Value szöveg (LONGTEXT) – Token értéke

*aspnetuserclaims*

**Id** szám (INT(11)) – elsődleges kulcs

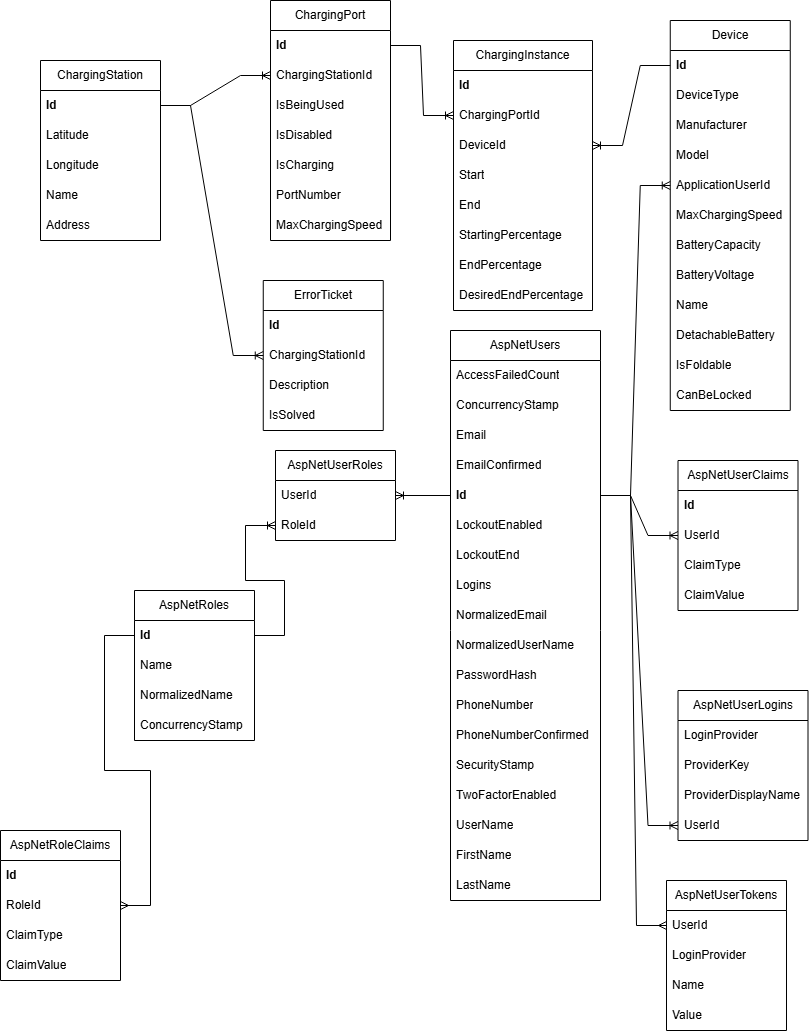
UserId szöveg (CHAR(36)) – Felhasználó azonosítója

(idegen kulcs)

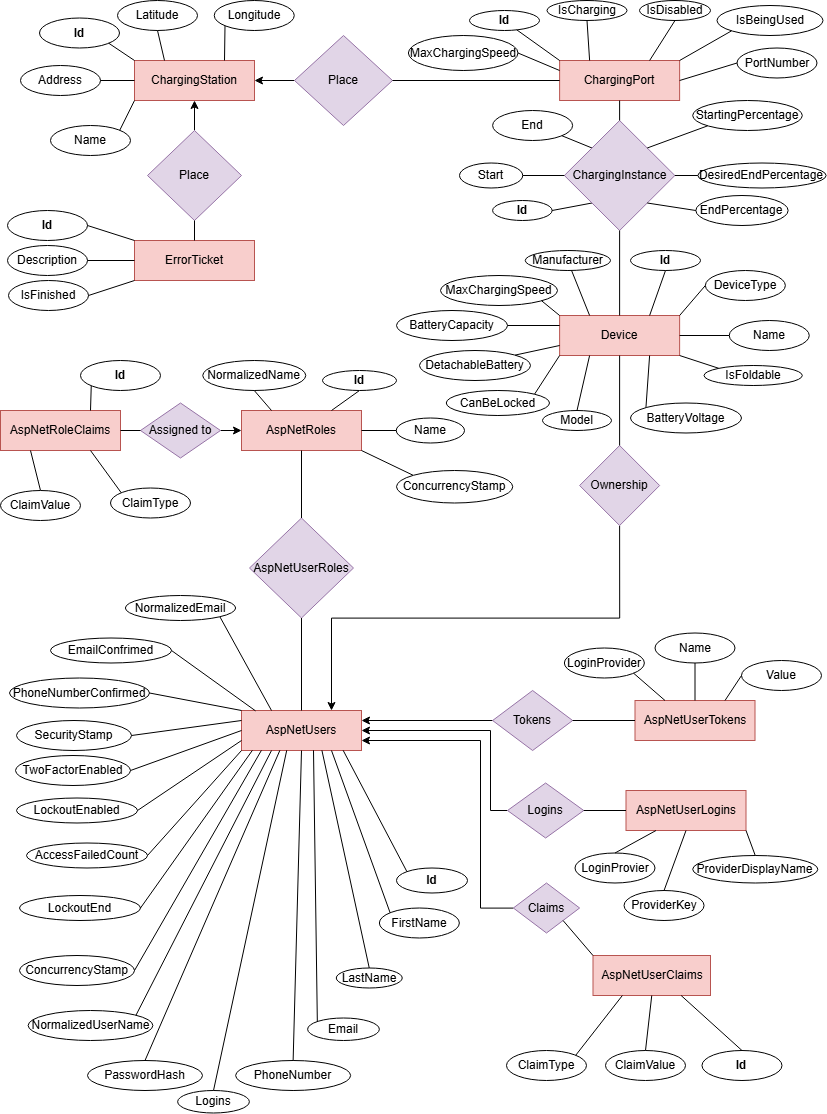
ClaimType szöveg (LONGTEXT) – Adat típusa

ClaimValue szöveg(LONGTEXT) – Adat értéke

#### **Adatbázis ábrái**



**Adatbázis Bachmann ábrája**



**Adatbázis E-K Diagramja**

### 4.2 Backend

A kódot tiszta rétegekre bontottuk.

#### Rétegek:

* **Controller** – HTTP végpontok, amelyek fogadják a kéréseket
* **Service** – üzleti logika: validálás, vezérlés, összetett folyamatok
* **Repository** – EF Core alapú adatbázis-hozzáférés

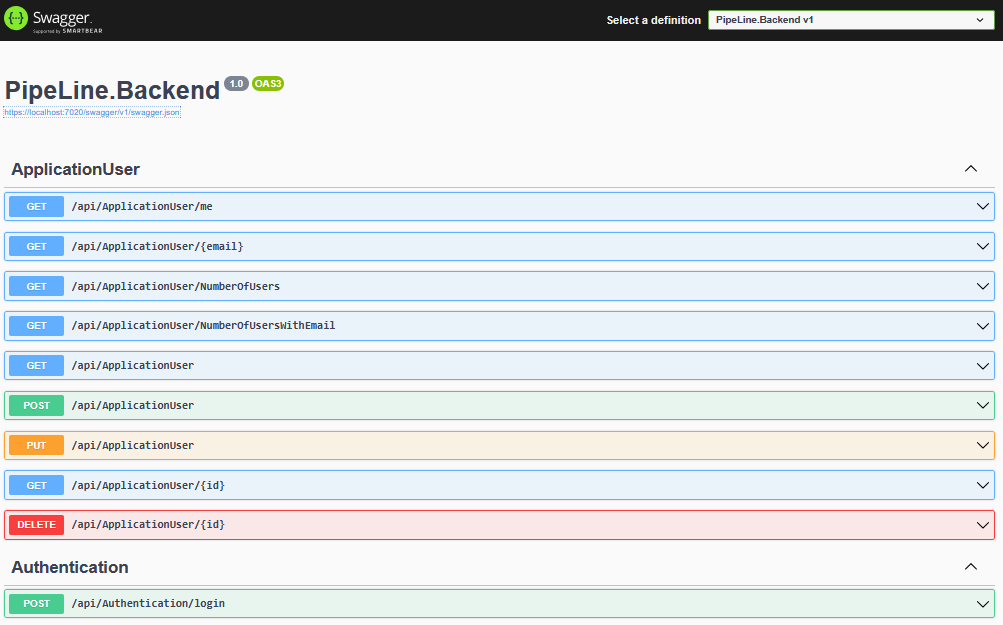
A felhasználók autentikációját JWT tokennel végezzük.

### 4.3 Rest API

Az API végpontjai JSON formátumban szolgáltatnak adatokat. Swagger dokumentáció is elérhető (/swagger/index.html), ahol kipróbálhatók az egyes műveletek.

**Néhány végpont:**

| **Művelet** | **HTTP metódus** | **Végpont** | **Leírás** |
| --- | --- | --- | --- |
| Regisztráció | POST | /api/authentication/register | Új felhasználó regisztrálása |
| Bejelentkezés | POST | /api/authentication/login | Bejelentkezés |
| Eszköz lista | GET | /api/device/myfiltereddevices | A felhasználó eszközeinek (opcionálisan szűrt) listája |
| Hiba bejelentése | POST | /api/errorticket | Töltőállomás hiba bejelentése |



**Kép a swaggerről**

### 4.4 WebSocket

A WebSocket-et a “ChargingProgressWebSocketHandler” osztály kezeli. A kapcsolat létrejöttekor a felhasználó azonosítója alapján kerül a memóriában tárolásra, és azonnal fogad élő frissítéseket a töltésről.

#### **Működés:**

* A felhasználó elindít egy töltést
* A backend periodikusan kiszámítja a töltöttséget
* A metódus kiküldi az adatokat
* A frontend onmessage eseményében frissül az UI

A WebSocket kapcsolat automatikusan újracsatlakozik, ha hibával szakad meg a kapcsolat.

### 4.5 Web és mobil

A frontend Vue 3 alapú, Composition API-t, Pinia-t, Bootstrapet és saját stílust használ. Mobilon Ionic és Capacitor.js segítségével fut a kód.

#### Funkcionális egységek:

* stores/ – minden entitáshoz saját állapotkezelő
* components/ – újrahasználható egységek
* views/ – oldalnézetek
* router/ – navigációs útvonalak
* websockets/ – folytonos kapcsolat létrehozása és kommunikáció

### 4.6 Desktop alkalmazás (admin)

A WPF-re épülő desktop alkalmazás MVVM (Model–View–ViewModel) mintát használ. A komponensek:

* Views – XAML alapú felhasználói interfész (UI)
* ViewModels – adatok, események, logika
* Models – osztályok, DTO-k és szerver válaszok
* Services – httpservice REST kommunikáció

A kapcsolat a szerverrel REST API-kon keresztül történik, a UserHttpService, DeviceHttpService, stb. osztályokon keresztül.

## 5. Tesztelés és minőségbiztosítás

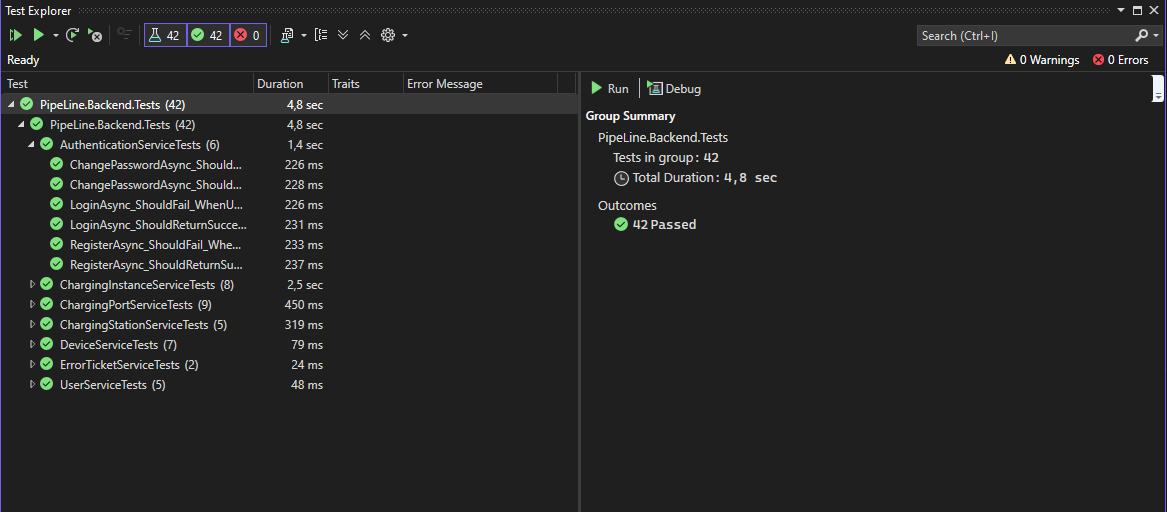
A PipeLine rendszer fejlesztése során figyelmet fordítottunk a szoftver megbízhatóságára, stabilitására és hibamentes működésére. Ennek érdekében a rendszer automatizált tesztelési mechanizmusokat implementáltunk a rendszer különböző rétegeibe.

### 5.1 Backend tesztelése

A rendszer backend automatikus teszteléséhez a MSTest (Microsoft.VisualStudio.TestTools.UnitTesting) keretrendszert használtuk. A mock objektumok kezelésére, mivel async metódusaink vannak, MockQueryable.Mock nevű csomagot használtunk.

**Tesztelt szolgáltatások:**

* AuthenticationService
* UserService
* DeviceService
* ChargingInstanceService
* ChargingPortService
* ChargingStationService
* ErrorTicketService

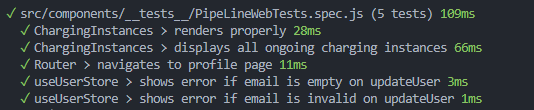
**Kép a backend tesztekről**

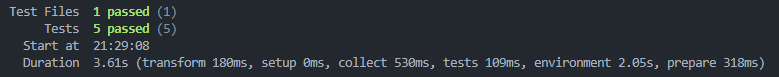
### 5.2 Webalkalmazás tesztelése

A Vue 3 alapú frontend komponenseinek automatikus teszteléséhez a Vitest tesztkeretrendszert használtuk. Csináltunk egy aktív Piniát globálisan, majd a speciális tesztesetekben erre készített Piniát használtunk. Ez nem futtatja ténylegesen a meghívott store action-öket, hanem a meghívásukat figyeli és be lehet állítani kezdőállapotot.

**Tesztelt funkciók:**

* ChargingInstancesView komponens helyes renderelése és töltési események megjelenítése
* Router navigáció (/profile oldalra irányítás)
* Hibás adatok esetén megfelelő toast üzenetek megjelenítése





**Képek a web tesztekről**

### 5.3 Mobilalkalmazás tesztelése

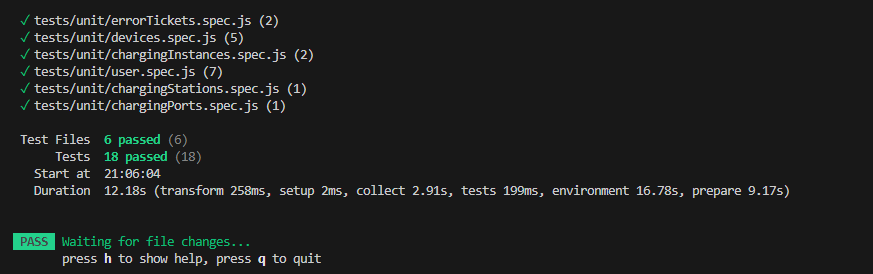
A mobilalkalmazás Pinia store műveleteinek automatikus tesztelésére a Vitest tesztkeretrendszert használtuk.

Az API kommunikációt az axios-mock-adapter könyvtár segítségével szimuláltuk.

**Tesztelt funkciók:**

* Töltési események lekérése, új töltés indítása
* Töltőportok és töltőállomások listázása
* Saját eszközök kezelése: hozzáadás, módosítás, törlés
* Hibajegyek létrehozása és lekérdezése
* Felhasználói hitelesítés: regisztráció, bejelentkezés, kijelentkezés, adatmódosítás

A tesztek ellenőrzik az API hívások sikeres lefutását és a store állapotának helyes frissítését.

**Kép a mobil tesztekről**

### 5.4 Asztali alkalmazás tesztelése

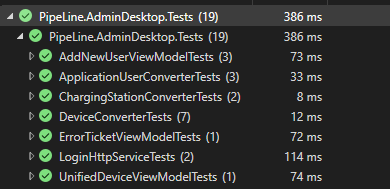
A WPF alapú desktop alkalmazás ViewModel-jeinek és segédszolgáltatásainak tesztelésére az xUnit tesztkeretrendszert használtuk.

A mock objektumok létrehozásához a Moq könyvtárat alkalmaztuk.

**Tesztelt funkciók:**

* Új felhasználó hozzáadása (AddNewUserViewModel)
* Eszközök DTO konverziója (DeviceConverter, ChargingStationConverter, ApplicationUserConverter)
* Bejelentkezés HTTP szolgáltatáson keresztül (LoginHttpService)
* Főmenü navigáció és nézetkezelés (MainViewModel)
* Hibajegyek kezelésének ViewModel-je (ErrorTicketViewModel)
* Eszközök betöltése és frissítése (UnifiedDeviceViewModel)

A tesztek biztosítják, hogy a ViewModel-ek megfelelően inicializálódjanak, frissítsék az állapotot, és kezeljék a felhasználói műveleteket.



**Kép a desktop tesztekről**

## 6. Zárás

### 6.1 Fejlesztés közben felmerült nehézségek és megoldásuk

A PipeLine fejlesztése során számos kihívással szembesültünk, amelyek megoldása során sokat tanultunk és fejlődtünk technikailag és csapatmunkában egyaránt.

* **WebSocket kapcsolat kezelése**: Az egyik legnagyobb kihívás az élő töltési adatok valós idejű frissítése volt. A WebSocketek működését úgy kellett megvalósítani, hogy hibamentesen újracsatlakozzanak mobilon és weben is hibás lekapcsolódás esetén. Nagy probléma volt még, hogy a töltési esemény leállításával ütközött a frissítése hiszen a memóriában a websocketes verziót tárolta a program, így hiába futott le a leállítás jól miközbe élt a valós idejű kapcsolat, felülírta azt újból a websocket.
* **Identity és EF Core integrációja**: A .NET beépített Identity rendszere alapból sok táblát és logikát hoz létre. Ezt össze kellett hangolni a saját entitásainkkal (pl. ApplicationUser), külön figyelmet kellett fordítani az adatkapcsolatokra és jogosultságok kezelésére.
* **Frontend reaktivitás (Pinia + WebSocket)**: A Vue 3 Pinia store-ja és a WebSocket együttműködése során voltak problémák a reaktív frissítéssel. A megoldás az volt, hogy reaktív objektumba helyeztük a chargingProgress-t és figyeltük az eseményeket komponens szinten is.
* **Térképes komponens mobilon**: A MapTiler API integrációja Ionicon belül nem volt zökkenőmentes. Egyedi Vue komponensbe helyeztük a térképet, és csak betöltés után inicializáltuk azt.

### 6.2 Továbbfejlesztési lehetőségek

A PipeLine alapjai stabilak, de rengeteg lehetőség rejlik a rendszer bővítésében. Néhány ötlet a jövőre nézve:

* Google / Facebook belépés (OAuth2 integráció)
* Push értesítések a mobilalkalmazásban (pl. töltés befejezve, hibabejelentés elfogadva)
* Kamera vagy QR-alapú port azonosítás
* Több nyelvre kiterjesztés
* Színtévesztőknek testreszabott színszűrő

### 6.3 Személyes és csapatszintű fejlődés

#### Gavlik Martin

* A backend fejlesztése során elmélyítette ASP.NET Core és EF Core ismereteit, valamint a WebSocket működését.
* Különösen sokat fejlődött az autentikáció (Identity, JWT), REST struktúra és rétegzett architektúrák területén.

#### Gavrán Tamás

* Mobilfejlesztés terén tanult új technológiákat:
  + Ionic használata Vue 3-on belül,
  + Capacitor használata a mobil projekt Android alkalmazássá fejlesztéséhez,
  + Vue 3 mobilnézet optimalizálás és designolás,
  + Frontend tesztelés során mockolt api használata Vitest-el.
* A UX/UI tervezésben szerzett gyakorlati tapasztalatot.

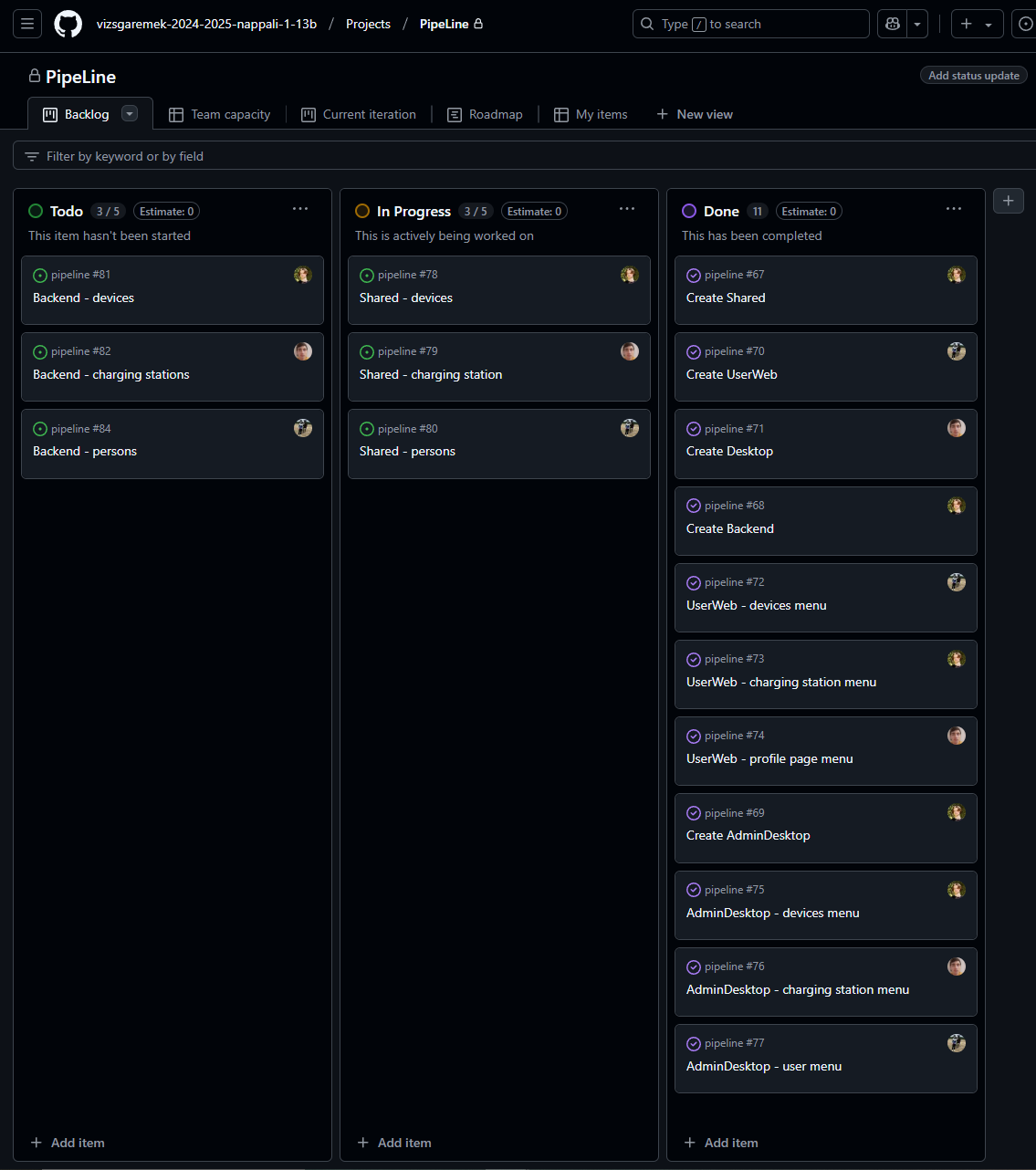
#### Kónya Milán

* A desktop alkalmazás fejlesztése során alaposan megismerte a WPF, MVVM és REST API integráció működését.
* A statisztikai nézetek és felhasználókezelés implementálása során erősödött a C# és XAML tudása.

#### Csapatként

* Megtanultunk verziókezeléssel (Git) hatékonyan dolgozni, feladatokat elosztani, és egymás munkáját támogatni.
* Fejlődtünk a közös tervezésben, tesztelésben és dokumentálásban is.

A fejlesztés során a GitHub projektkezelő funkcióját használtuk a feladatok kiosztására, nyomon követésére és státusz kezelésére.

**Kép a kiosztott feladatokról**

### 6.4 Köszönetnyilvánítás

Ezúton szeretnénk köszönetet mondani a Szegedi SZC Vasvári Pál Gazdasági és Informatikai Technikum tanári karának.

Köszönjük továbbá az iskolának a lehetőséget, hogy részt vehettünk a programon, ahol bemutathattuk cégek képviselőinek a szoftverrendszerünket.