ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Szabó László**
2022 Hallgató törzskönyvi száma: **T/005816/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Biomatika és Alkalmazott Mesterséges Intelligencia Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Szabó László**
Törzskönyvi száma: T/005816/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Felhőalapú nyomkövető rendszerek
Cloud-based GPS trackers

Intézményi konzulens: Vörösné Dr. Bánáti-Baumann Anna
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Felhőszolgáltatási technológiák és IT biztonság specializáció

A feladat

A mai világban egyre több igény van személygépjárművek használatára. Ebből adódóan egyre fontosabbá válik, hogy minél részletesebb információt kapjunk személygépjárművünkről. Ilyen információ a fogyasztás, a kilométeróra állás, aktuális sebesség, valamint a jármű helyzete. A GPS nyomkövetésnek szerteágazó lehetősége van, mint például a pozíció alapján történő vészhívás, utazási szokások vizsgálata, vagy a lopásgátlóként funkcionáló helyzetjelző készülékek.
A feladat járművek nyomonkövethetőségét célozza meg, melynek fő feltételei, hogy a nyomkövető eszköz kompakt méretű, alacsony fogyasztású és viszonylag pontos helyzetet képes szolgáltatni.
A hallgató feladata egy olyan GPS nyomkövető rendszer specifikálása, megtervezése és megvalósítása, melynek segítségével bármely méretű személygépjárművet nyomon lehet követni, helyzetét bármikor, bármilyen internet kapcsolattal rendelkező eszközről le lehet kérdezni.

A dolgozatnak tartalmaznia kell:

- a létező vagy hasonló megoldások feltérképezését, vizsgálatát és összehasonlítását,
- a feladattal szemben elvárt követelmények meghatározását,
- a követelményeknek megfelelő technológiák kiválasztását,
- a támasztott igényeknek megfelelő hardver beszerzését, a szoftver megtervezését és megvalósítását,
- a kész projekt tesztelési jegyzőkönyvét,
- az eszköz továbbfejlesztési lehetőségeit.

Ph.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

.....
Dr. Eigner György
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

Bácsi Ane
.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: SZABÓ LÁSZLÓ..... Neptun kód: [redacted]..... Tagozat: .....

Telefon: [redacted]..... Levelezési cím (pl: lakcím): .....

Szakdolgozat / Diplomamunka[1] címe magyarul:

FELHŐALAPÚ NYOMKÖVETŐ RENDSZEREK .....

Szakdolgozat / Diplomamunka[2] címe angolul:

CLOUD-BASED GPS TRACKERS.....

Intézményi konzulens: VÖRÖSNÉ DR. BÁNÁTI-BAUMANN ANNA .....

Külső konzulens: .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.03.12	Dolgozat felépítésének megbeszélése	Bánáti Anna
2.	2021.04.12	Irodalomkutatás kiértékelése	Bánáti Anna
3.	2021.04.27	implementációs tervek áttekintése	Bánáti Anna
4.	2021.05.06	prezentáció megbeszélése	Bánáti Anna

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2021.05.14.....

Bánáti Anna

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

9. sz. melléklet

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun kód: Tagozat:

SZABÓ LÁSZLÓ. ..... [redacted] .....

Telefon: Levelezési cím (pl: lakcím):

[redacted] .....

Szakdolgozat / Diplomamunka[1] címe magyarul:

FELHŐALAPÚ NYOMKÖVETŐ RENDSZEREK .....

Szakdolgozat / Diplomamunka[2] címe angolul:

CLOUD-BASED GPS TRACKERS .....

Intézményi konzulens: Külső konzulens:

VÖRÖSNÉ DR. BÁNÁTI-BAUMANN ANNA..... .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	**2022.03.22**	**Dolgozat folytatásának átbeszélése**	Bánáti Anna
2.	**2022.04.20**	**Megvalósított részek, problémák kielemzése**	Bánáti Anna
3.	**2022.04.28**	**Feltárt problémák javításának ellenőrzése**	Bánáti Anna
4.	**2022.05.09**	**Összegzés, prezentáció**	Bánáti Anna

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022.05.12.....

Bánáti Anna

.....

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

Tartalom

1. BEVEZETŐ

A gépjárművek száma jelentősen megnőtt napjainkban, ezáltal az utcán parkoló járművek száma is. Rengeteg gépjármű van, néha riasztó berendezés nélkül, teljesen védtelenül kint hagyva a ház előtt, vagy egy parkolóban. A gépjárművekkel együtt megnövekedett az eltulajdonításoknak a száma is, mely jelentős anyagi kárt okoz rengeteg ember számára.

Az emberek szeretnék biztonságban tudni tulajdonukat, szeretnék, hogy csökkenjen a lopások száma, illetőleg a rendőrség vagy bármilyen hivatalos szerv életét meg tudja könnyíteni egy olyan eszköz, mely nyomon követhetőséget biztosít olyan emberek számára, akiknek érdeke a gépjármű biztonságban tudása.

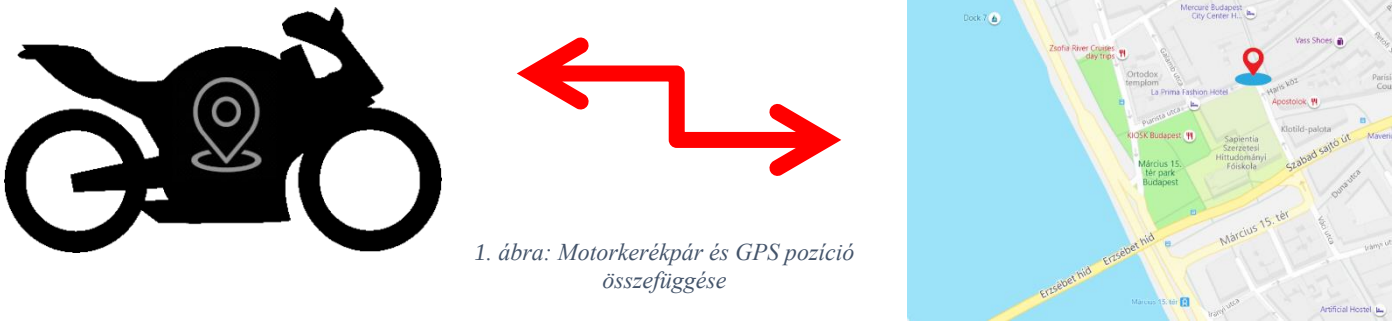
Az autó tolvajok általában hamar kiszúrják az autóban található furcsa eszközöket, így óvakodnak olyan gépjárművek eltulajdonításától, ahol riasztó berendezés található, viszont nem minden autó rendelkezik alapvetően beépített riasztó rendszerrel, így ezen gépjárművek tényleges célpontokká válhatnak.

Motorosok szemszögéből tekintve akár még fontosabbnak vehetjük ezt az eszközt, ugyanis egy két kerekű, bárki által könnyedén megközelíthető gépjárműről beszélünk, amihez még könnyebb hozzáférni, mint egy autóhoz. Ezeknél a járműveknél általában nem található sem riasztó berendezés, se központi zár. Az egyetlen védelmet nyújtó berendezés általában a kormányzár motorkerékpároknál. Kiegészítő védelmet nyújthatnak tárcsafék zárak, ponyvák, minőségi láncok, de ezek sem nyújtanak teljes biztonságot, mert ezek mind látható dolgok és egy tapasztalt tolvaj számára könnyen eltávolíthatóak.

Ezen problémák orvoslása végett, szükség van egy olyan eszközre, mely könnyedén elrejthető a tolvajok elől, kis helyigénnyel rendelkezzen, illetve, ha az illető nem találná ott a gépjárművet, ahol lerakta, akkor megtalálja a pontos helyét, emellett a rendőrség munkáját segítve nyomon követhetőséget biztosítana, tehát jelezné számunkra a gépjármű pontos pozícióját előre deklarált időközönként.

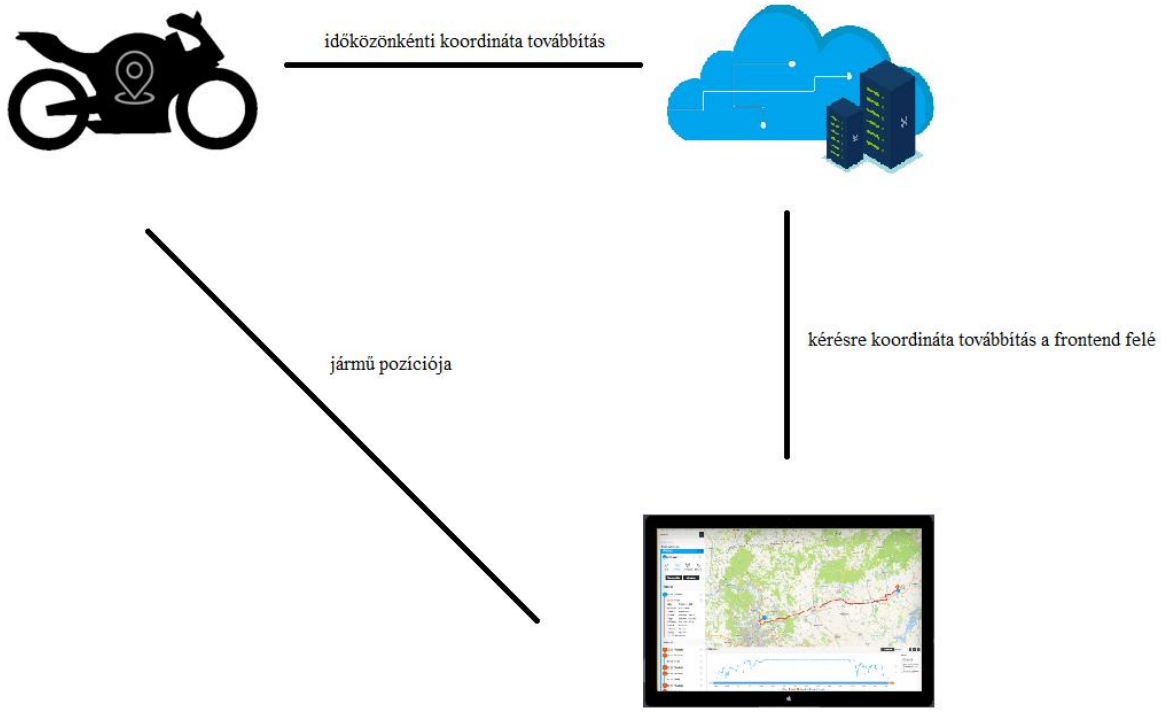
2. PROJEKT ÉS MÁS HASONLÓ ESZKÖZÖK ÖSSZEHASONLÍTÁSA

A projekt célja egy olyan nyomkövető rendszer kiépítése, ahol egy áramforrással ellátott mikrokontroller/ miniatűr számítógép GPS koordinátákat továbbít egy szerver (Backend) felé, ami ezt az adatot egy adatbázisban eltárolja és kérésre a koordinátát továbbítja a Frontend,



1. ábra: Motorkerékpár és GPS pozíció összefüggése

azaz az ügyfél által is látott platform felé.

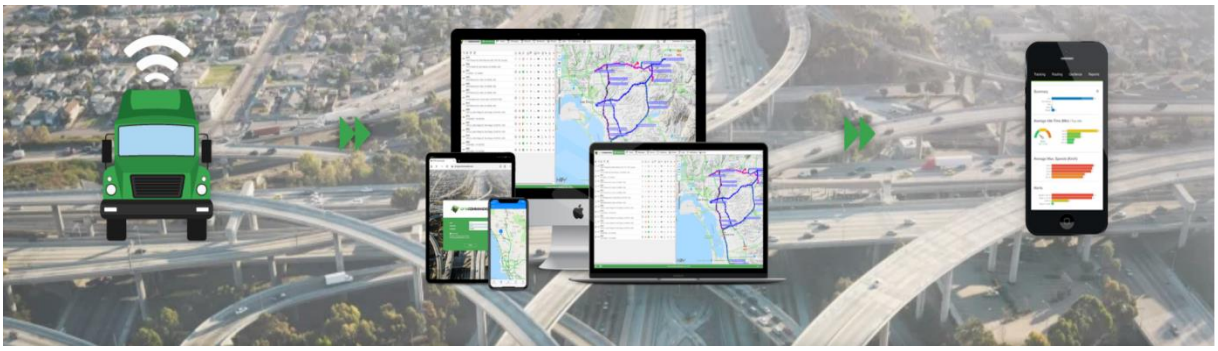


2. ábra Körfolyamat bemutatása

Ez a platform lehet telefonos applikáció, számítógépre letölthető szoftver, illetőleg webes felület is. Fokozott figyelmet kell fordítani arra, hogy projekt „cross-platform” legyen, tehát ha számítógépes vagy telefonos programot készítünk, az operációs rendszertől független módon működjön, illetőleg, ha webes felületről van szó, akkor minden böngészőből reszponzív módon jelenjen meg. Tehát ha telefonról nézzük, akkor telefonos felület jelenik meg, másrészt pedig kijelző mérettől függően jelenjenek meg az adatok. A mikrokontrollert egy internetkapcsolattal rendelkező SIM kártyával látjuk el, ezen keresztül fogja tudni továbbítani a koordinátákat a szervernek.

Utána jártam a létező GPS nyomkövető rendszerek megvalósításának és működési elveinek, példaképp felhoznám a GPSCOMMANDER[1]-t, mely egy olyan GPS nyomkövető rendszert biztosít cégek számára, ami egy telepített eszköz és a jármű antennája segítségével tudja nyomon követni a gépjárművet, ezzel kapcsolatban probléma merül fel motorkerékpárok esetében, ahol többségében nem található antenna, így motorkerékpárok védelme ezen termékkel nem lehetséges. A GPSCOMMANDER alapvetően nem hétköznapi használatra van tervezve, hanem fuvarozó cégek számára, hogy könnyedén nyomon követhessék haszongépjárművük pozícióját.

A GPSCOMMANDER-nek van egy olyan változata, mely beépíthető személygépjárműbe is, de ennek működési elve az, hogy a jeleket akkor továbbítja a rendszer, ha a jármű be van indítva, tehát töltést kap az eszköz. A gépjárművek eltulajdonításának több módja is van, nem mindig a tolvaj vezeti a gépjárművet, előfordulhat az is, hogy egy teherautó rakodó részére rakja fel, illetőleg motorkerékpárok esetében gyakori az, hogy egyszerűen eltolják a célhelyig, indítás nélkül.



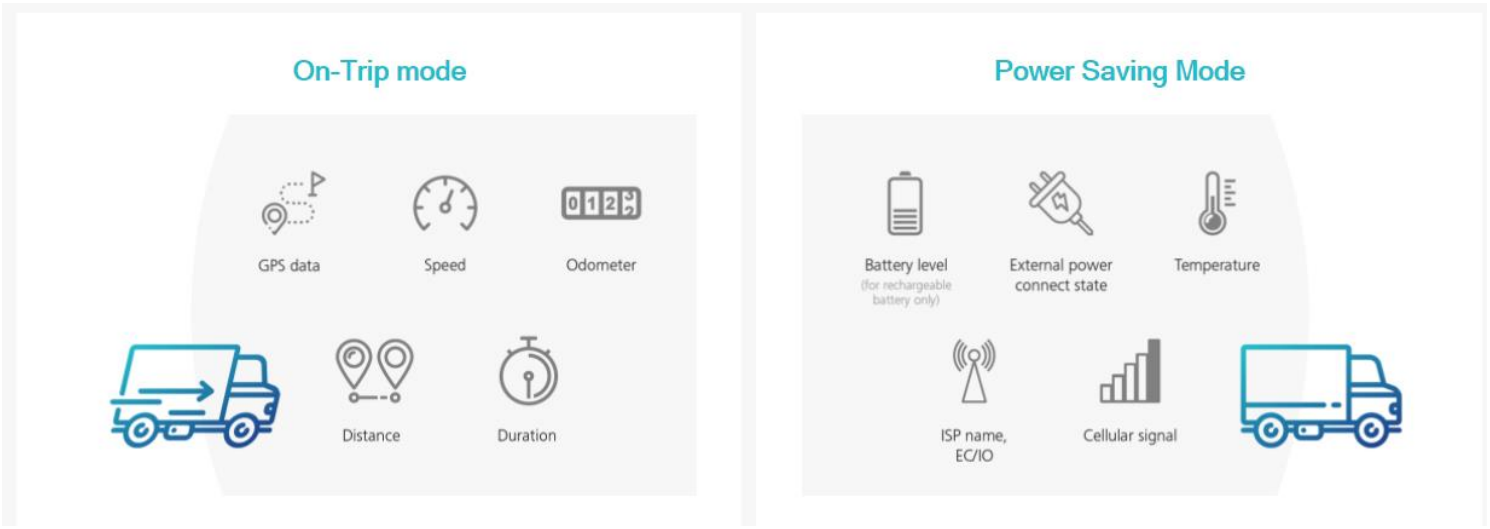
3. ábra: GPSCOMMANDER termék[1]

Rendelkezésünkre bocsájtanak egy ingyenes próbalehetőséget, ellenben termékárat nem tüntetnek fel weboldalukon, kizárólag árajánlat kérése lehetséges.

Említésre méltó a PEPXIM[2] is, amely szintén egy felhő alapú nyomkövető rendszer. Ez egy saját akkumulátorral rendelkező eszköz, két redundáns SIM foglalattal rendelkezik, ezáltal a GPS koordinátákat valós időben tudja továbbítani. Előnyére szolgál a saját akkumulátor, ugyanis nem kell aggódni, hogy az integrált eszköz lemeríti a jármű akkumulátorát, négy C méretű elem vagy újratölthető lítium elem segítségével működik. Ennek hátránya viszont, hogy folyamatosan figyelni kell az akkumulátor töltöttségének szintjét, így a projekt során maradok a jármű akkumulátorához csatlakoztatásánál. Kétféle üzemmód lehetséges, az úgynevezett „On-Trip” mód, illetve az energiatakarékos mód, az On-Trip mód esetén folyamatosan valós időben tölti fel a GPS-adatokat a felhőbe és küldi az értesítést a felhasználónak a jármű pontos pozíciójáról. Az akkumulátor élettartamának megőrzése érdekében az eszköz (ha nincs aktív On-Trip módban) offline tárolja GPS adatait későbbi visszakeresés céljából, ahelyett, hogy folyamatosan valós időben, közvetlenül a felhőbe töltené.

Az imént említett nyomkövető berendezés nagyon fejlett, ám úgy gondolom, a saját akkumulátoros rendszer kevésbé stabil és megbízható energiaforrás, mint egy nagyobb akkumulátorral rendelkező gépjármű, így megmaradok az általam választott módszernél.

Az oldalon pontos információkat nem találhatunk a termék áráról, feltételezhetően itt is csak árajánlat kérése lehetséges.



4. ábra: PEPXIM termék[2]

Számos hasonló cég árulja termékét az interneten, viszont általában ezen termékek célközönsége leginkább fuvarozó cégek, illetve egyéb vállalkozások, nem pedig magánszemélyek. Ezen termékek úgynevezett „flotta követésre" vannak fejlesztve, tehát több jármű egyszerre nyomon követésére is szolgálnak. A projekt célközönsége viszont nem a vállalatok, illetve cégek, hanem az átlagember. Szeretnék egy olyan eszközt, illetve rendszert kiépíteni, amely óvja az emberek tulajdonát a lopástól, illetve nyomon követhetővé teszi a járművet bármilyen mozgás esetében, nem csak járó motor mellett. Emellett fontos megemlíteni azt, hogy költséghatékony megoldást kell találni, hogy a céltermék olcsón megvásárolható legyen átlagemberek számára, emellett figyelve arra, hogy az előállítási költség is kedvező legyen.

Termék	GPS Commander	PEPXIM
Célközönség	Cégek	Cégek
Platform	Minden platformon elérhető	Minden platformon elérhető
Töltés megoldás	Gépjármű akkumulátora	Saját akkumulátor
Saját antenna	Nincs	Van
Jeltovábbítás módja	Járó motornál	Járó motornál/parkoló üzemmódban is
Flotta csomag	Igen	Igen
Ár	Árajánlat	Árajánlat

1. táblázat GPS Commander és PEPXIM termék összehasonlítása

Elsősorban webes felületen szeretném megvalósítani projektet, a későbbiekben akár több platformra is megvalósításra kerülhet.

3. PROJEKT MEGVALÓSÍTÁSÁNAK MENETE

Témámból eredően szükségünk van egy fizikai eszközre, amelyet integrálnunk kell a gépjárműbe. Több opció is felvetésre került, elsősorban az eszköz mérete volt meghatározó, mivel egy laptop nagyságú nyomkövető rendszert senki sem szeretne telepíteni járművébe. Ezen okból kifolyólag olyan méretű eszközöket vettem figyelembe, melyek nagysága nagyjából egy telefonnal egyezik meg. Ilyen opciók például a Raspberry PI bármely típusa, az Arduino [21] mikrokontroller, vagy akár egy régóta nem használt telefon, mely rendelkezik GPS antennával.

A Raspberry PI [24] egy kisméretű egy lapkás számítógép, amit a Raspberry Pi Foundation és a Broadcom együttesen tervezett és talált ki. A Pi-t egy ARM alapú processzor hajtja, a processzor mellé kerül legalább 256 MB RAM, egy nagyon egyszerű videókártya, pár USB port, HDMI port, microSD kártya támogatása, illetve egy 40 pines csatlakozó. Egyes típusok kivételével Ethernet porttal, kamerával, LCD panellal, analóg kimenetekkel, előre telepített WiFi-vel és Bluetooth-al is rendelkezik.

A Raspberry Pi egyik különlegessége, hogy rendelkezik saját operációs rendszerrel, de képes más rendszereket is kezelni, mint például Linux és nem Linux alapú operációs rendszereket. Képes különböző driver API-kat kezelni, mint például a Vulkan[3]-t, melyre példa, hogy a Raspberry Pi 3B+ a Quake 3 nevű játékot 100 FPS-el tudta futtatni. Emellett tud OpenGL ES-t, OpenMAX-ot használni.[4]

Különböző tartozékokkal bővíthető, például többféle kamerával, GPS modullal, antennákkal, kijelzővel, port bővítőkkel, számos interfésszel.

Jelenleg rengeteg típus található a piacon, 8500 Forinttól indul.

Ezzel szemben az Arduino egy mikrokontroller, kevesebb alapértelmezett porttal rendelkezik a Raspberry-hez képest. Az Ardunio a Pi-hez hasonlóan rendelkezik memóriával, bővítő portokkal, lábkiosztással. Nyílt forráskódú a fejlesztőplatformja, ennek köszönhetően a projektekben használt, Arduino-hoz csatlakoztatott eszközök könnyebben használhatóak, kezelhetőek. Processzorok tekintetében Atmel, ARM és Intel CPU-kal került gyártásra. Operációs rendszere nincsen, Optiboot Bootloaderrel [5] működik, árát tekintve körülbelül

Eszköz	Raspberry Pi	Arduino
Típus	Számítógép	Mikrokontroller
Operációs rendszer	Van	Nincs
Bővíthetőség	Bővíthető	Bővíthető
Portok száma	Több alapértelmezett port (bővíthető)	Kevesebb port (bővíthető)
Memória	van	van
Dokumentációk	Van, de kevesebb található róla, mint az Arduino-ról	Régebb óta elterjedt, több dokumentáció található róla
Processzor	Többnyire ARM	Atmel, ARM és Intel

8500 Forinttól kapható.

2. táblázat Raspberry és Arduino összehasonlítása

Az Arduino mikrokontroller mivoltából adódóan nem szeretnék vele foglalkozni, mert a Pi-re létező operációs rendszerek miatt könnyebben találok megoldásokat fejlesztés szempontjából.

3.1 Térkép Alkalmazás

A projektben, a koordináták feldolgozásához szükségünk van egy Térkép API-ra.[6] Három választási lehetőség merült fel, a Google Maps API, a Bing Maps API, továbbá az OpenStreetMap API. A Google Maps API-ja natív iOS és Android vezérlőket nyújt számunkra, ellentétben a Bing-el, ott ehelyett a natív alkalmazásokban használhatjuk a Bing Maps v8 Javascript vezérlőt. A Bing Maps API előnye, hogy rendelkezik WPF és UWP támogatással is, ami nagyon nagy segítség Windows platformra fejlesztők számára. A UWP számos funkcióval rendelkezik, mint például a 3D-s objektummegjelenítéssel és Bing Maps API támogatással.

Geokódolás szempontjából sincs túl sok különbség a három API között, nagyjából hasonló tartományú funkciókkal rendelkeznek. Megkeresik a cím helyét és fordított geokódolással megtalálják a címeket.

Útvonaltervezésben is hasonló funkciókkal rendelkeznek, mindhárom rendelkezik olyan útválasztási opciókkal, melyek segítenek megtalálni a megfelelő útvonalat legyen szó gépjárműről, gyalog, vagy akár tömeg közlekedve. Emellett tájékozódási tippeket szolgáltató API-kal is találkozhatunk, például olyannal, ami figyelmeztet, hogy ha látunk egy leírásnak megfelelő épületet, annál az utcánál kanyarodjunk le jobbra.

A Google Maps rendelkezik olyan API-kal, amik a kerékpárútvonalakkal szolgál, illetve a valós idejű navigációt is támogatja, viszont ezen funkciók nem találhatóak meg a Bing Maps API-ban, sem az OpenStreetMap API-ban. Ezzel szemben a Bing-es API tartalmaz teherautó-specifikus útvonaltervezési opciót, melynek keretében például figyelembe veszi a magasság- és súlykorláttal ellátott helyek, a meredek lejtők, szűk útszakaszok, vagy a robbanásveszélyes, illetve vegyi anyagot szállító járművek kitiltottságát.

Fontos megemlíteni a Bing egyedülálló Isochrone API-ját, amely akár projektem bővítése során számomra is jelentős bónusszal szolgálhat, ugyanis ezen API számításokat végez el a megtett táv és idő függvényében, így akár jóslatot is tud tenni arra, hogy várhatóan mennyi utat vagyunk képesek megtenni „X” idő alatt. Számunkra ez a funkció hasznos tud lenni, ugyanis be tudjuk határolni, hogy az elveszett, vagy ellopott járművünk egy óra múlva körülbelül milyen messze lehet a legutóbbi pozícióktól.

A Google Maps API 2005-ben indult eleinte könnyedén elérhető volt bárki számára, viszont 2016-tól már Google Térkép Kulcs kellett a használatához, ami szintént ingyenes volt 2018 júniusáig. 2018 júniusától[7] a Google Térkép API-ja fizetőssé vált annak érdekében, hogy növelni tudják a rendelkezésre állást és a teljesítményt.

Habár Google Maps API fizetős[8], a Google egy 200 dolláros „ingyen kreditet” ad havonta, ami 5-11 ezer API hívást biztosít. Ez nem megfelelő az alkalmazás fejlesztésére, mert könnyedén átléphető a havi kvóta.[9]

Az OpenStreetMap legnagyobb előnye, mint ahogy a neve is tükrözi, hogy ingyenes, rengeteg hasznos kód található az implementálására és napi 1,000,000 API hívást támogat.

A Bing Maps API-nak az engedélyezett API hívások száma alkalmazás függő, például Session alapon 25 hívást engedélyez, publikus Windows alkalmazások számára 50 ezret naponta, minden más alkalmazás számára 125 ezret évente.

Ezen adatok ismeretében úgy döntöttem, hogy a projekt során a Bing API-t fogom alkalmazni.

Alkalmazás	Google Map+API	Bing Map+API	OpenStreetMap+API
API hívások száma	havonta 5-11 ezer API hívás ingyenes, azután fizetni kell	Windows alkalmazások számára 50 ezret naponta, minden más alkalmazás számára 125 ezret évente	1 millió API hívást támogat NAPONTA!

3. táblázat Google Maps és Bing Maps API hívásainak száma

3.2 Adatszámítás a felhőben

A felhő fontos része az alkalmazásnak, hisz mind az adatok tárolása mind az alkalmazás szerver oldali kommunikációját és a webes felületet is a felhőbe tervezem futtatni. Ezért fontos volt, hogy olyan megoldást alkalmazzak, ami megfelel ennek a környezetnek. Három megoldáson gondolkoztam, az egyik az Amazon AWS rendszere a második a Microsoft Azure rendszere, a harmadik pedig egy self hosting megoldás. Ami mindenképp fontos volt, hogy külön alkalmazásokat lehessen külön rendszereken tárolni megnövekedett erőforrás igény nélkül, például LXC konténerekben[10]. Szerencsére mind három megoldás esetén van erre, vagy ehhez hasonló megoldásra, de pont emiatt megnehezítette a döntést, hogy ennyire sokoldalúak a felhőszolgáltatók, a self hosting megoldás esetében is számos keretrendszer van, ami megkönnyíti az erőforrások megtakarítását, de ezt a megoldást hamar elvetettem, hisz egy ilyen applikációhoz előnyt nem csupán hátrányt okozna főleg, ha több régióba is szeretném később üzemeltetni, és sokkal költségesebb lett volna.

A két szolgáltatás számos funkcióval rendelkezik, ezeknek a szolgáltatásoknak 2 részét vizsgáltam meg: Adattárolás és számítási kapacitás.

A számítási kapacitásnál a gyártók közti különbség a kihasználatlan erőforrásoknál és skálázhatóságnál található. Az Azure VM-eket vagyis virtuális gépeket használ melynek létrehozásánál állítunk be. Hogyha ezek az erőforrások egy része kihasználatlan, van lehetőség a többi virtuális gépünknek ezeket az erőforrásokat kihasználni. Az AWS EC2 (Elastic Computing) megoldása lehetőséget ad, hogy az erőforrások szabadon nőjenek vagy csökkenjenek attól függően, hogy mennyi erőforrásra van szükségünk.[11]

Az adattárolás mindkét szolgáltatónál hasonló és megbízható. Az AWS esetén, ha gyakran használt adat eléréséhez van szükségünk használhatunk S3-at míg, ha olyan adatot szeretnénk tárolni, amit megőrizni szeretnénk és nem gyakran használni lehetőségünk van használni a költséghatékony Glacier megoldást. Nem használt adatunkat S3-ból könnyen törölhetjük, vagy archiválhatjuk Glacier szolgáltatásban. Az Azure esetében van lehetőségünk Hot and Cold storage használatára. A Cold storage havi díja olcsóbb viszont az írás és olvasási műveletek drágábbak, a Hot storaggel szemben.[12]

A harmadik fontos tényező az árazás volt. Mindkét szolgáltatónak vannak kedvezményes/ingyenes szolgáltatásai az első évben, ám az Azure esetében van lehetőség évi 100 USD kreditnyi erőforrás kihasználására a diákoknak tesz lehetővé. Az Azure szolgáltatásai olcsóbbak, ez betudható annak is, hogy a piaci részesedése kisebb. [13-14]

Mivel mindkét szolgáltató erős és megbízható szolgáltatásokat nyújt, de az Azure-t választottam, hiszen az árazása sokkal kedvezőbb, és a diákoknak járó kedvezményét is ki tudom használni.

3.3 Szerver oldali applikáció

Az eszköz folyamatosan küld adatokat és ehhez szükség van egy API-hoz, ami fogadja az eszköztől érkező adatokat.

Az API esetén 3 nyelv között gondolkoztam:

- PHP [15] backend
- C# [28] backend
- JS [29] backend

Mindhárom környezet tökéletesen lehet használni API megírásához, és mindhárom környezet könnyen hozzáférhető mind Windowson és Linuxon. A PHP[15] a 3 közül a legrégebbib programnyelv webalkalmazások írásához, és ez számos pozitívumot, viszont negatívumot is vonz magával. Pozitívuma, hogy a nyelvhez számos példa és sablon található, viszont mivel a PHP sokat fejlődött az évek során számos ilyen sablon elavult lehet. A PHP egy script nyelv, ami azt jelenti, hogy az alkalmazást nem kell buildelni, mivel egy interpreter futtatja.

A második lehetőség a Javascript, amely node.js-n[16] futtatva egy remek környezet webapplikációkhoz. Előnye, olyan applikációknál van, amely intenzív számú kapcsolatot (request) igényel. Egy tradicionális webszerver minden requestnél új szálat nyit, maximalizálva az elérhető memóriát a node.js ezt egyetlen egy szálba optimalizálja. Ez egy példával szemléltetve, ha mindegyik kapcsolat 2mb-ot használ fel, egy 8gb RAM esetében ez 4000 folyamatos kapcsolatot jelent, míg Node.js esetén 8 GB memóriával 1 millió folyamatos kapcsolat érhető el.

A harmadik lehetőség az ASP.Net 5 Core[17-18] ami a .Net 5 keretrendszer webre specializált része. Maga a .NET 5, ami a .Net Core tovább fejlesztett verziója, mely a .Net Framework cross-platform verziója. Az elnevezése picit trükkös, mivel a Microsoft szeretné fókuszba helyezni a .Net 5-öt, de nem akarja, hogy az elnevezés ütközzön már létező technológiákkal, mint például az ASP MVC 5.

Előnye, hogy a keretrendszer Open Source, illetve a moduláris felépítése, mely NuGet packagek segítségével, számos modult használhatunk. Ennek a moduláris felépítésnek köszönhetően van lehetőség Self-Contained (SCD) alkalmazás létrehozására is mely egybecsomagolja a szükséges libraryket, így lehetőség van olyan eszközön is futtatni alkalmazást, amelyen nem telepítettük fel a keretrendszer, ezzel is tovább növelve felhasználási lehetőségeit.

A .Net Core-t választottam, a moduláris felépítése miatt, illetve mert a C# nyelvben vagyok a legtapasztaltabb.

3.4 Webes Felület

Mivel az alkalmazásban szükség van valamilyen megjelenítésre és nem szeretnék platformoktól és eltérő kódbázisoktól függeni, ezért számításba vettem a webes applikáció készítését is. Négy lehetséges keretrendszert vizsgáltam meg, ezek a Vue.js, a React.js, az Angulart[16, 28-29], továbbá az ASP.NET Core MVC 5-öt.

Mindegyik java script alapú front-end keretrendszer, kivéve az ASP.NET Core-t.[20]

A React-ot a Facebook készítette 2013-ban. Fő célja az egyszerűség, a méretezhetőség és a gyorsaság. Híressé válásában rendkívüli szerepet játszott a teljesítménye, a 43 kB-os csomagméretével rendkívül gyorsan vált híressé a sebessége és számos egyéb funkciója miatt, mint például a virtuális DOM használat, erőforrás-terhelések minimalizálása, vagy a szerver oldali megjelenítés (Server-Side rendering). A virtuális DOM használata rengeteg előnnyel jár a teljesítmény és az alkalmazás terhelésének optimalizálása terén. React alkalmazások esetén az adatok csak egy irányba köthetők, ezáltal könnyebben ellenőrizhetjük ez egész projektet, ezt hívjuk egyirányú adatkötésnek. Előnyére szolgál még a könnyű tesztelhetőség, a React segítségével nagyon egyszerű a tesztek írása, ami megkönnyíti a program működésének ellenőrzését.

Ellenben hátránya a fentebb hogy néha extra könyvtárakat kell használnia az állapot és a modell megvalósításához. További hátránya, hogy gyenge dokumentációval rendelkezik, ez leginkább annak köszönhető, hogy a React keretrendszer folyamatos frissítései, valamint az összes támogató könyvtár létrehozása miatt olyan mértékben felgyorsultak, hogy nincs elég idő a megfelelő dokumentációk elkészítésére. Ez a rendkívüli ütemben fejlődés hátránya is lehet, ugyanis a fejlesztőknek rendszeresen fel kell ismerniük a dolgok működésének új módját.

A Vue.js-t a Google egyik alkalmazottja, Evan You fejlesztette ki 2014-ben. Általános meghatározás szerint a Vue.js egy progresszív keretrendszer a felhasználói felületek felépítésére. A nagyméretű keretrendszerekkel ellentétben, mint például az Angular, a Vue úgy lett kialakítva, hogy fokozatosan alkalmazható legyen a felhasználók számára. Tulajdonságainak, viszonylag könnyű tanulási görbéjének, hatékonyságának köszönhetően gyors és kifinomult egyoldalas alkalmazások készíthetőek. Ebből kifolyólag a Github egyik „legcsillagozottabb” JavaScript keretrendszere. Összehasonlítva a React-tal, közös bennük, hogy mindkettő virtuális DOM-ot használ, reaktív és komponens alapú, illetve a lényegi része a könyvtár központi eleme, más fontos műveleteket, mint például a routingot kiegészítő könyvtáraknak hagyja meg. Teljesítményét és méretét tekintve az alap csomag 18 kB így rendkívül gyors keretrendszernek számít. A React.js-el szemben a Vue rendkívül jó dokumentációval rendelkezik, ami hihetetlenül hasznos a fejlesztők számára.

Hátránya azonban, hogy megjelenést tekintve a Vue a három keretrendszer közül a legújabb, így kisebb közösséggel rendelkezik, mint versenytársai.

Az Angular[28] a Google által lett kifejlesztve, először 2010-ben jelent meg, ezzel a „legöregebb” a keretrendszer a felsoroltak közül. Ez egy TypeScript alapú keretrendszer, amit az Angular CLI JavaScript és HTML kóddá fordít. Az Angular a Vue-val, illetve a React-al

ellentétben egyszerű kétirányú adatkötést, valamint beépített modulrendszert, routing csomagot és függőség-injektálást biztosít. Így egy gyorsan bevált technológiává vált, amelyet fejlesztők milliói használtak. Előnyére szól, hogy nagy népszerűségnek örvend, így rengeteg dokumentáció található róla, nem mellesleg nagy közösséggel is rendelkezik, így, ha egy probléma megoldásában segítségre van szükség, valószínűleg hamar megoldást találnak rá.

Hátránya azonban a teljesítményben és a méretben mutatkozik ki, számos szolgáltatása néha megterhelheti a projekteket és lassabb teljesítményt nyújt emiatt a React-hoz vagy a Vue-hoz képest. Ez amiatt mutatkozik ki, hogy az Angular jelentősen több funkcióval rendelkezik.

Tanulási görbéje meredek, ez a rengeteg funkciónak is köszönhető, nehezebb megtanulni ezt a keretrendszert, mint a fent említetteket, viszont egyszerűbb benne programozni, mert rendes programozási szintaktikája van és hasonló a C# szintaktikájához.

Az Angular és az ASP.NET Core MVC 5 lényegi különbsége, hogy míg az Angular a kliens erőforrásait felhasználva JavaScript alapon készíti el a megjelenítendő oldalt, addig az MVC a backend oldalon generálja le. Az Angular teljesen leválasztja a kliensoldali feldolgozást a szerverről, ellenben az MVC 5-el, ahol egyszerre tudjuk felépíteni a backendet és a frontendet.

A projekt során úgy gondolom, hogy kihasználva az MVC felépítést, illetve a tanulmányom során szerzett tapasztalataimat az ASP.NET Core MVC 5 mellett döntök a Webapplikáció esetében.

3.5 Az eszköz programozása

Mint egy korábbi pontban részleteztem a Rasberry PI Zero W[21] eszközt választottam a fizikai eszköznek, viszont ehhez is kell egy szoftver mely kommunikál a felhővel.

Ehhez is három nyelv között gondolkoztam, mely segítségével tudunk ARM architektúrára programozni:

- C++ [22]
- Python [30]
- C#

A C++ az egyik legősibb programozási nyelv, ám a mai napig folyamatosan fejlődik. Legelső megjelenése a C nyelv problémájára adott megoldást, mely az Objektum Orientált Programozás volt. Előnye, a másik két környezettel szemben ez alacsony szintű nyelv, emiatt hardverre optimalizált kódot kapunk, viszont a kód fejlesztése nehézkes lehet hisz egyes funkciók megvalósítása sokkal bonyolultabb a másik két nyelvel szemben

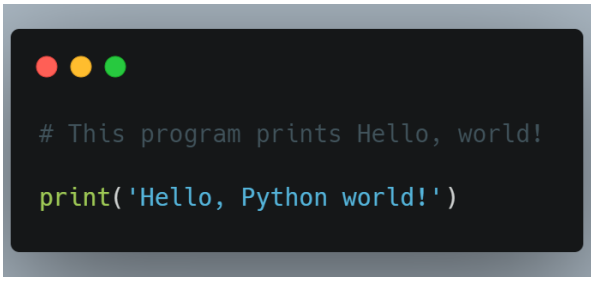
A Python[23] nyelv egy gyors és kezdő barát programozási nyelv. Mivel Script nyelv a kódot nem kell lebuildelni, csupán futtatni a kódot egy távoli kliensen. Szintaktikája az összes kezdő programozónak barátságos lehet, gyakran szokták pseudo programozási nyelvnek hívni. Ami viszont az előnye az könnyen lehet a hátránya is, hisz egy más nyelvekben

tapasztalt programozónak ez a szintaktis fura lehet, illetve könnyen lehet hibázni egy félreütött szóköz és tabulátor karakterrel.

A harmadik lehetőség a C#. Ezen nyelv a .Net és .Net Core keretrendszerek része, ezért utóbbi miatt mára már egyre több platformon elérhető. A nyelvet a Microsoft fejlesztette és előnye, hogy a kód írása egyszerű, és átlátható, viszont ez nem jár jelentős teljesítmény csökkenéssel.

Hello World programok az említett nyelveken:

5. ábra: C# Hello World kód



6. ábra: Python Hello World kód

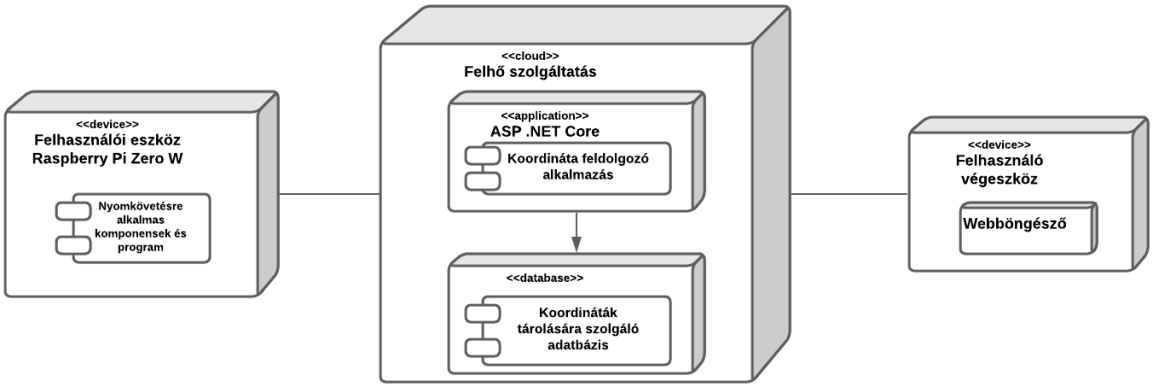
7. ábra: C++ Hello World kód

Az eszköz programozása tekintetében a Python mellett döntöttem, ugyanis a C++-hoz képest egyszerűbb, átláthatóbb, és előnye a C# és a C++-hoz képest, hogy nem kell fordítani, compile-olni sem.

4. A RENDSZER TERVE ÉS FOLYAMATAI

Maga a rendszer áll egy felhasználói eszközből, mely adott esetben egy Raspberry Pi Zero W választására esett. A választás szempontjából fontos tényező volt, hogy az eszköz fogyasztása alacsony legyen, ugyanis közvetlenül a gépjármű akkumulátoráról kapja a töltést. Maga a Raspberry Pi Zero W nem rendelkezik nagy erőforrásokkal, ellenben erre nincs is szükségünk, hisz csak koordináta továbbítása, illetve esetleges offline állapotban koordináta ideiglenes tárolása a feladata.

A koordinátákat egy a felhőben található alkalmazás szervernek továbbítjuk, mely feldolgozza az adatokat, illetve perzisztálja az adatbázisba. Ezen applikáció elkészítéséhez ASP.NET Core keretrendszert választottam alapul.



8. ábra Rendszerterv

4.1 Az adatbázis

Mint korábban említettem, az alkalmazásnak szükség van egy adatbázisra, ahol a futás során létrejövő adatokat tudja tárolni. Alapvetően 4 táblára van szükség, mely minden egyes adat tárolását és kiszolgálását tudja végezni. Elsősorban szükség van egy táblára a felhasználó adatainak tárolása végett, továbbá egy eszköz táblára, ahol a GPS nyomkövetők adatait tároljuk. Ezen nyomkövetők koordinátáját is tárolnunk kell, így szükségünk van egy koordináta táblára is. Ahhoz, hogy egy eszköz hitelesíteni tudja magát és a küldött adatai ténylegesen mentésre kerülhessenek, léteznie kell egy táblának, mely azon eszközök egyedi azonosítóját tartalmazza, melyek ténylegesen kiadásra kerültek felhasználók részére, így legutolsó sorban igényünk van egy kiadott eszközök táblára is.

Az eszközök, koordináták és felhasználók külön-külön osztályok, hiszen mindegyiket külön táblákban tároljuk, különböző módon kell feldolgoznunk.

Az adatbázist a szerverapplikációval MySQL Connection segítségével lehet összekötni. Ahhoz, hogy az adatbázisba adatot tudjunk menteni, felhasználni, módosítani, esetleg törölni, minden osztály esetében külön repository-t kell készíteni.

4.2 Szerver oldali alkalmazás

A szerver oldali alkalmazás fő feladata, hogy kapcsolatot biztosít az adatbázis és a web applikáció között. Minden, a kliens, vagy eszköz felől érkező kérést fogad, feldolgoz és eltárol az adatbázisban, vagy jelenít meg. A szerver oldali applikáció esetében HTTPS protokollt használó alkalmazásra van szükségünk. Ahhoz, hogy a kapcsolat biztonságos legyen a kliens és a szerver között SSL/TLS kapcsolatot kell megvalósítani és azon keresztül kell kommunikáljon az applikáció. Ennek feladata, hogy megakadályozzuk egy illetéktelen, harmadik fél információhoz jusson, esetleg jogosulatlanul hozzáférjen a rendszerhez.

A „self-signed” tanúsítvány előnye, hogy nem kell külső kibocsátóhoz fordulnunk. A tanúsítványok HTTPS protokoll használatát teszik lehetővé, ami biztonságos kapcsolódást biztosít különféle szolgáltatásokhoz, azonban ezen típusú tanúsítvány használata esetében a böngészők figyelmeztetik a felhasználót, hogy az oldal nem hitelesített tanúsítványt használ. A projekt során ebből kifolyólag célszerű nem önaláírt tanúsítványt használni, tekintettel arra, hogy ezen tanúsítvány látványa elriaszthatja a felhasználókat.

A szerver oldali applikáció működéséhez szükségünk van a .NET Core Runtime keretrendszerre, továbbá a .NET Core 5.0 -ra. Az általam választott keretrendszer támogatja a HTTPS protokollt, ezzel is kiváltva egy plusz proxy szerver használatát.

Célom, hogy az alkalmazás csak hitelesített felhasználókat tudjon kezelni, ehhez szükségem van authentikációra. Ahhoz, hogy egy felhasználó hitelesíteni tudja magát, szükséges regisztrálnia, majd csak a bejelentkezést követően tudunk számára, hozzá tartozó szolgáltatás(ok)ról információt biztosítani.

Ezen authentikáció megvalósítására célszerű az ASP.Net Core keretrendszerhez ajánlott „Identity” szolgáltatást igénybe venni, ezzel is megkönnyítve a fejlesztés menetét, hiszen nem kell a már készhez kapott dolgokat kezdetekről megvalósítani. Ezen előre definiált funkciók módosíthatók, így a számunkra szükséges elvárásoknak eleget tud tenni. Az eszközöket a felhasználónak képesnek kell lennie magához rendelni, továbbá a szervernek validálnia kell az eszköz hitelességét, melyet az adatbázissal kapcsolatban tesz meg.

4.3 Kliensoldal

Ahhoz, hogy a felhasználó kezelni tudja a rendszert, megtekinteni az eszköz(ök) lokációját, illetve saját adataihoz hozzáférjen, biztosítani kell számára egy felületet, ahol mindezt megteheti. Ez a felület a kliensoldal, mely HTML5 kódot használ, ebből adódóan az alkalmazás használatához olyan böngészők szükségesek, melyek támogatják a HTML5-öt.

A weboldal ellentétben az Angular, Vue.js és társaival, nem (JavaScript alapon) kliens oldalt generálódik, inkább hasonlítható a PHP-hoz, hiszen szintén szerver oldalon jön létre a kód.

4.4 Eszköz

Az eszköz három fő részből áll, melyek egymásra illesztve csatlakoznak. Első sorban egy egyszerű, kevés erőforrásigényű operációs rendszerre van szükség, ugyanis az eszköz a minimális erőforrásokkal gazdálkodik.

Ezen felül szükségünk van egy GPS Hat-re, amely a Serial porton keresztül adatokat szolgáltat a GSM és a GPS modul felől.

Az operációs rendszer segítségével a GPS Hat-et a harmadik fő résszel, a programkóddal lehet kezelni.

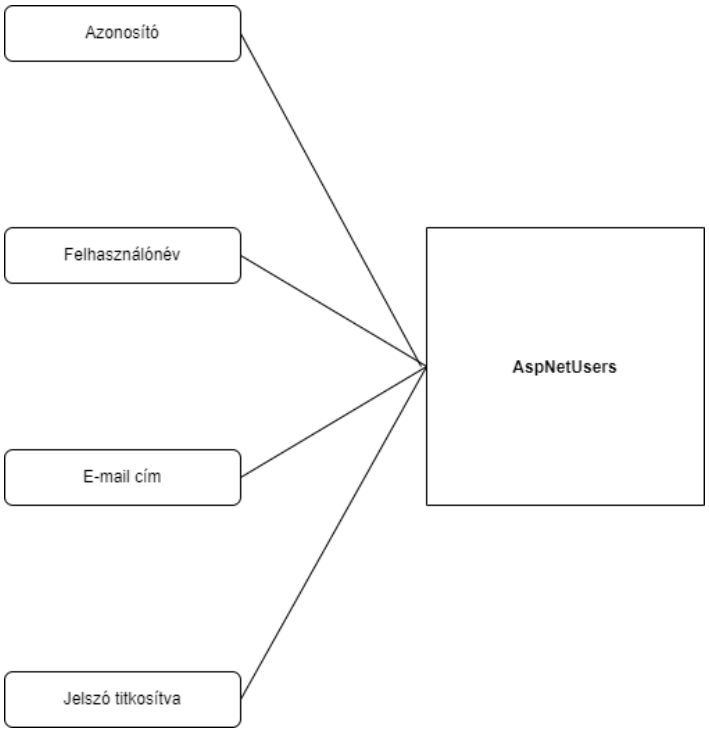
5. AZ ALKALMAZÁS MODELLJEI

Mint korábban említésre került, az adatbázist a klienssel, illetve a nyomkövető eszközzel a szerver oldal köti össze. Azonban ahhoz, hogy ezt a szerver el tudja végezni szüksége van modellekre. Minden egyes modell osztálynak szükséges rendelkeznie valamilyen egyedi azonosítóval, amely során az adatbázis azonosítani tudja ezen osztályokat.

Minden modell számára biztosítani kell legalább egy egyedi azonosítót, melyre az „ID” használható. Az eszközök azonosítására továbbá biztosítanunk kell egy egyedi azonosítót, amit nevezhetünk „Eszköz egyedi azonosítónak”.

5.1 Felhasználó modell osztály

Kötelező eltárolni a felhasználókhoz tartozó adatokat, ezek magában foglalják a felhasználó e-mail címét, felhasználó nevét továbbá jelszavát, illetve egy egyedi azonosítót. Ezen 4 adatból az e-mail cím, az egyedi azonosító és a felhasználó név egyedi kell legyen, ezzel azonosítva a felhasználót.



9. ábra AspNetUsers modell

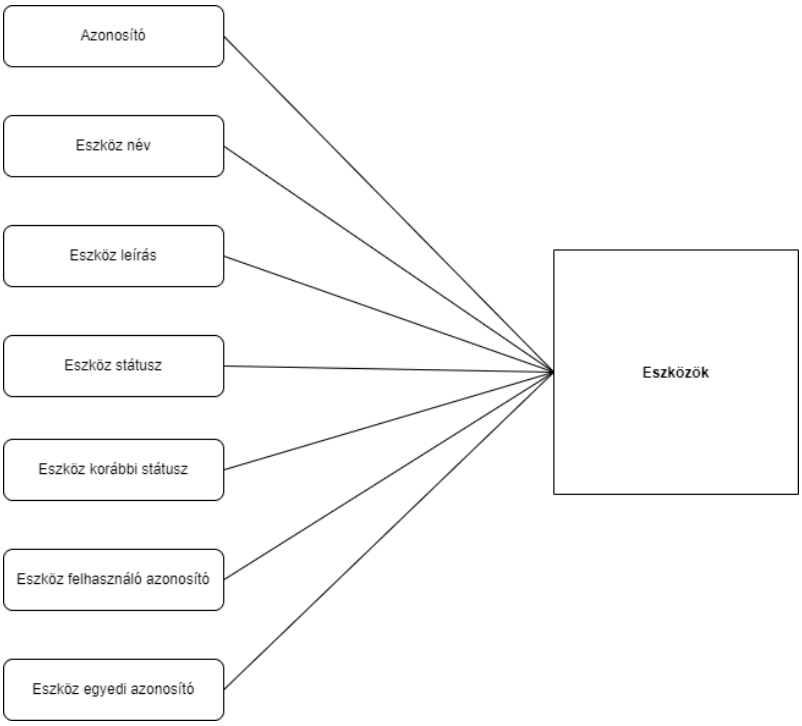
AspNetUsers osztály elemeinek típusa			
Azonosító	Felhasználónév	E-mail cím	Jelszó
string	string	string	(hash) string

4. táblázat AspNetUsers osztály elemeinek típusa

5.2 Eszköz modell osztály

Az eszköz modell számunkra a legfontosabb, hiszen ez a modell teremti meg a kapcsolatot a felhasználó és a koordináta között.

Felépítését tekintve 7 fontos elemre bontható, elsősorban célszerű egy azonosítóval ellátni, mely az alapvető (CRUD) műveletek folyamatát könnyíti meg, ajánlott egy megnevezést biztosítani, mely a felhasználó saját eszközének könnyű beazonosíthatóságára szolgál, emellett egy leírás is biztosítható erre. Szükséges az eszköz esetében egy státusz tárolása, valamint egy korábbi státuszra is, mely arra szolgál, ha esetleg az eszköz elérhetetlenné válna, a szerver oldal tisztában legyen az eszköz státuszával kapcsolatban. Mint korábban említettem, az eszközt kötelező hitelesítenünk, így egy globálisan egyedi azonosítóval is el kell látni, melyet a felhasználó, egy általunk biztosított kód megadásával tud megtenni. Ezt hozzátársítva az eszközhöz tudjuk meg, hova tartozik egy koordináta. Legutolsó sorban hozzá kell rendelnünk az eszközt a felhasználóhoz, aki beregisztrálta, így kell egy eszköz felhasználó kapcsolat is.



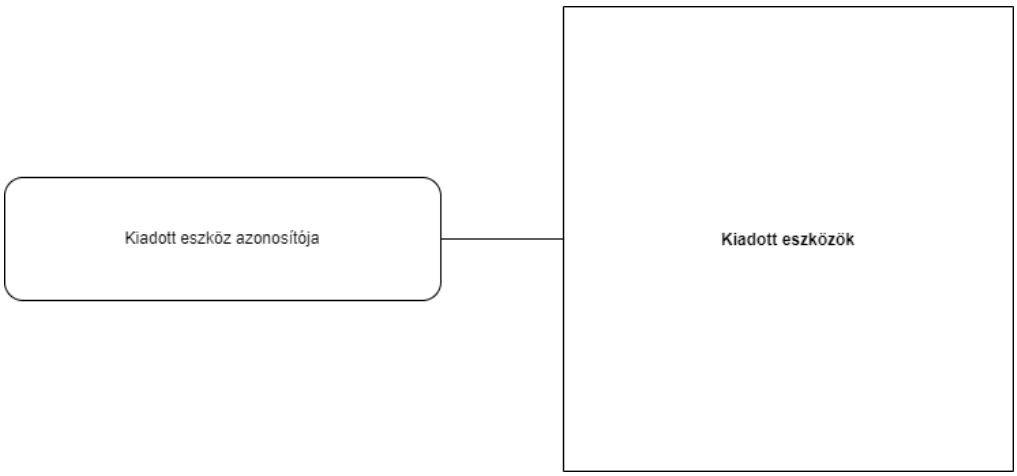
10. ábra Eszközök modell

Eszközök osztály elemeinek típusa						
Azonosító	Eszköz név	Eszköz leírás	Eszköz státusz	Eszköz korábbi státusz	Eszköz felhasználó azonosító	Eszköz egyedi azonosító
int	string	string	enum	enum	string	string

5. táblázat Eszközök osztály elemeinek típusa

5.3 Kiadott eszközök modell

Ezen modell nem tekinthető külön osztálynak, hiszen a csak egy adatból áll, amely arra szolgál, hogy a szerver oldal felé hitelesítsük a kiadott eszközöket. A szerver oldal ez alapján a modell alapján ellenőrzi, hogy az eszköz, amit a felhasználó regisztrálni szeretne, ténylegesen kiadásra került vagy sem.



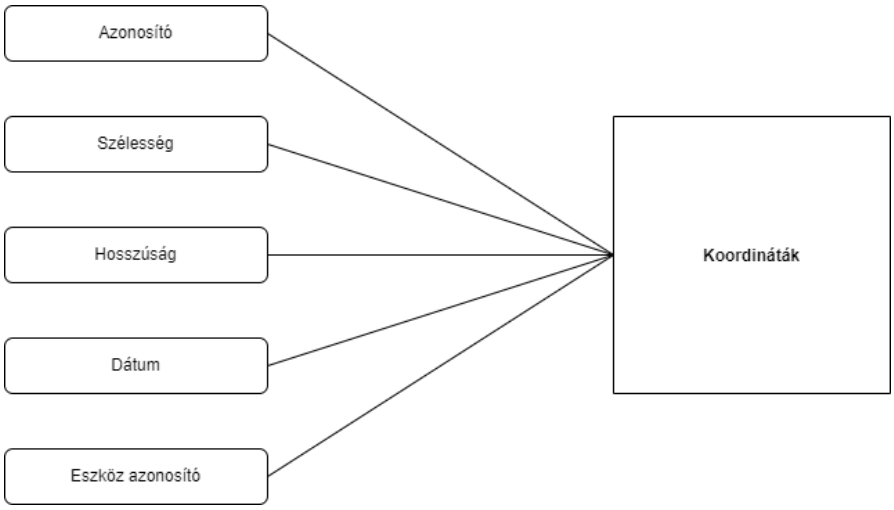
11. ábra Kiadott eszközök modell

Kiadott eszközök elemeinek típusa
Kiadott eszköz azonosítója
string

6. táblázat Kiadott eszközök elemeinek típusa

5.4 Koordináta modell osztály

A koordináta modell közvetlenül az Eszköz modellel áll kapcsolatban, az így a szerver oldal képes elkülöníteni, mely eszközhöz mely koordináták tartoznak. Egy koordinátáról alapvetően 5 adatok kell tároljon, egy azonosítót, egy szélességi fokot, egy hosszúsági fokot, egy dátumot, továbbá egy eszköz azonosítót, mellyel a kapocs létrejön.



12. ábra Koordináták modell

Koordináták osztály elemeinek típusa				
Azonosító	Szélesség	Hosszúság	Dátum	Eszköz azonosító
int	string	string	(hash) string	int

7. táblázat Koordináták osztály elemeinek típusa

6. FELHASZNÁLÓ HITELESÍTÉS

Ahhoz, hogy a felhasználó hitelesíteni tudja magát, szükséges megadnia egy e-mail címet, továbbiakban egy jelszót, melynek a következő szabályoknak meg kell felelnie: tartalmaznia kell kis-, illetve nagybetűt és számot, legalább 8 karakter hosszúnak kell lennie.

Későbbiekben a felhasználónak lehetősége kell legyen a felhasználónév módosítására, alapértelmezetten az e-mail címmel megegyezhet létrehozáskor. Továbbiakban a regisztrált felhasználónak biztosítani kell egy felületet a bejelentkezésre, ahol a regisztrációkor megadott adatokkal tudja hitelesíteni magát. Későbbiek során megadhat telefonszámot, módosíthatja jelszavát, illetve e-mail címét is esetleges változást követően. A műveletek tekintetében minden esetben ellenőrizni kell, hogy az adott felhasználó megfelelő jogokkal rendelkezik-e az egy művelet elvégzéséhez.

7. KONTROLLEREK

Az eszköz és a szerver közötti kommunikációhoz, a felhasználó és a szerver közötti kommunikációhoz szükség van több végpontra is, mely elvégzi a szükséges műveleteket. Emiatt több kontrollert is definiálni kell, hogy átlátható legyen.

Minden egyes kontroller egy bizonyos célosztályhoz van rendelve és ezeket célszerű attribútumokkal ellátni, ilyen például az authentikáció megkövetelése a felhasználók részére, vagy hogy egy API-t milyen útvonalon érjen el az eszköz.

7.1 Felhasználó kontroller

A felhasználó adatainak lekérdezéséhez, módosításához, kezeléséhez szükségünk van egy vezérlőre, amin keresztül a felhasználó kéréseinek eleget tudunk tenni. Ezen a kontrolleren keresztül tudnunk kell új jelszót igényelni, felhasználónevet módosítani, e-mail címet módosítani, jelszót módosítani, telefonszámot megadni stb.

Minden felhasználónak lehetőséget kell biztosítani az bizonyos adatainak módosítására, azonban ezzel párhuzamban a saját adatainak és eszközeinek módosítására, illetve ellenőrzésére szabad csak jogosultságot biztosítani.

Ezért szükséges a felhasználónak authentikálnia magát, amiért ez a kontroller a felelős, továbbá a személyes adatainak módosításáért is kell feleljen.

7.2 Térkép kontroller

A térkép kontroller egyetlen szerepkörrel jár, a Térkép oldalon megjelenő tartalom helyes kiszolgálásáért. Ennek a kontrollernek a feladata, hogy mely eszközöket listázza ki a felhasználónak és jelenítteti meg ezen eszközök koordinátáit. Mindenképp meg kell említenem, hogy ehhez kizárólag authentikált felhasználók férhetnek hozzá, hiszen csak ez alapján képes eldönteni a kontroller mögött található logika, hogy kinek milyen eszközei vannak és melyeket kell megjelenítse.

7.3 Eszköz kontroller

Elérkeztünk egyik legfontosabb kontrollerünkhöz, az eszköz kontrollerhez, melynek feladatai az eszközök kezelése. A felhasználóknak lehetőséget kell biztosítani új eszköz hozzáadására, melyet egy elnevezés, opcionálisan leírás és egy egyedi azonosító megadásával kell megtennie. A megadott adatokat ellenőrizni kell, amennyiben a megadott adatok megfelelnek a követelményeknek (valid egyedi azonosító, kitöltött név mező) azon esetben továbbítja az információt a repository rétegnek és mentésre kerül az adatbázisban. Módosítás esetén az egyedi azonosító módosítására nem szabad lehetőséget biztosítani, kizárólag a név, a leírás és a státusz módosítására szabad lehetőséget biztosítani. Amennyiben az eszköz státusza „Offline” abban az esetben nem tudja módosítani a státuszt sem a felhasználó. Törlés esetén a kiválasztott eszköz azonosítója alapján kerül az adatbázisból törlésre az eszköz.

7.4 Koordináta kontroller

A koordináta kontrollerben nem szükséges a „CRUD” műveletek összességére függvényhívást biztosítania, hiszen törölni és frissíteni koordinátát a felhasználónak nem szabad, így erre funkciót nem biztosít. Figyelnie kell arra, hogy az eszközzel is kommunikálnia kell, hiszen az eszköz erre a kontrollerre hív be koordináta küldése esetén. A koordinátákat különböző módon kell tudnia kezelni, tekintettel arra, hogy amennyiben az eszköz internet hiányában nem képes továbbítani a koordinátákat, azokat lokálisan eltárolja, majd ismételt internetelérés esetén egyben küldi el a szervernek, így ezeket részekre kell

bontani és eltárolni az adatbázisban, azonban képesnek kell lennie az egyesével küldött koordináták feldolgozására is.

Az eszköz az egyedi azonosítóján kívül 3 adatot küld magáról: szélesség, hosszúság, időpont. Ezeket az információkat a kontroller továbbítja a repository-nak, ami pedig feldolgozza azt. Abban az esetben, ha riasztás érkezik, a vezérlő értesíti a felhasználót erről.

7.5 Státusz kontroller

A státusz kontroller feladata, hogy az eszközt kiszolgálja a státuszáról, amennyiben az eszköz érdeklődik erről. Célja, hogy amennyiben a felhasználó módosítja a státuszt a felhasználói felületen (melyet az Eszköz kontroller kezel), erről az eszköz kapjon tájékoztatást, miután lekérdezi és annak megfelelően küldje a koordinátákat és végezze a feladatát.

8. KOMMUNIKÁCIÓ A FELHASZNÁLÓ ÉS A SZERVER KÖZÖTT

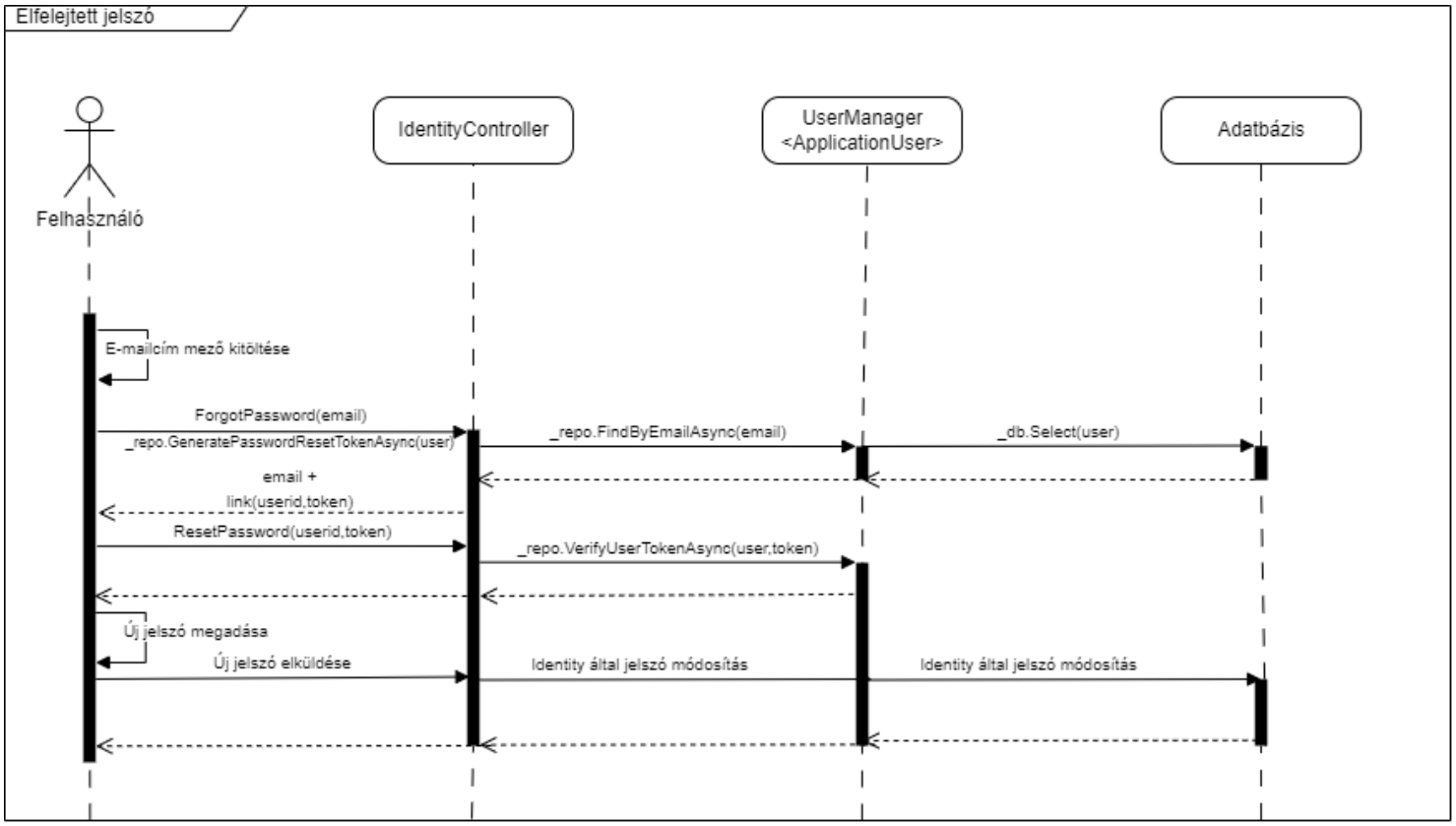
Az alkalmazásnak ki kell szolgálnia a felhasználó igényeit, mind saját profiljának szerkeszthetősége, mind a szolgáltatás által nyújtott funkciók megjelenítése tekintetében.

8.1 Felhasználói műveletek

A felhasználó a webes felületen egy regisztrációs felület kitöltésével kell kezdjen, majd ezt követően a szerver értesíti e-mailben, hogy hitelesítse adatait az e-mailben szereplő linkre kattintva. Fontos megjegyezni, hogy a hitelesítés előtt a felhasználó nem tud bejelentkezni a profiljába. Bejelentkezést követően lehetősége van jelszót módosítani, felhasználó nevet módosítani, továbbá e-mail címet módosítani.

Amennyiben a felhasználó elfelejtené jelszavát, lehetősége van a felületen új jelszót igényelni, melyhez ki kell töltenie az e-mail mezőt és a küldés gombra kattintani. Ezt követően a rendszer kiküld egy e-mailt, amiben a linkre kattintva a felhasználó új jelszót tud beállítani profiljának.

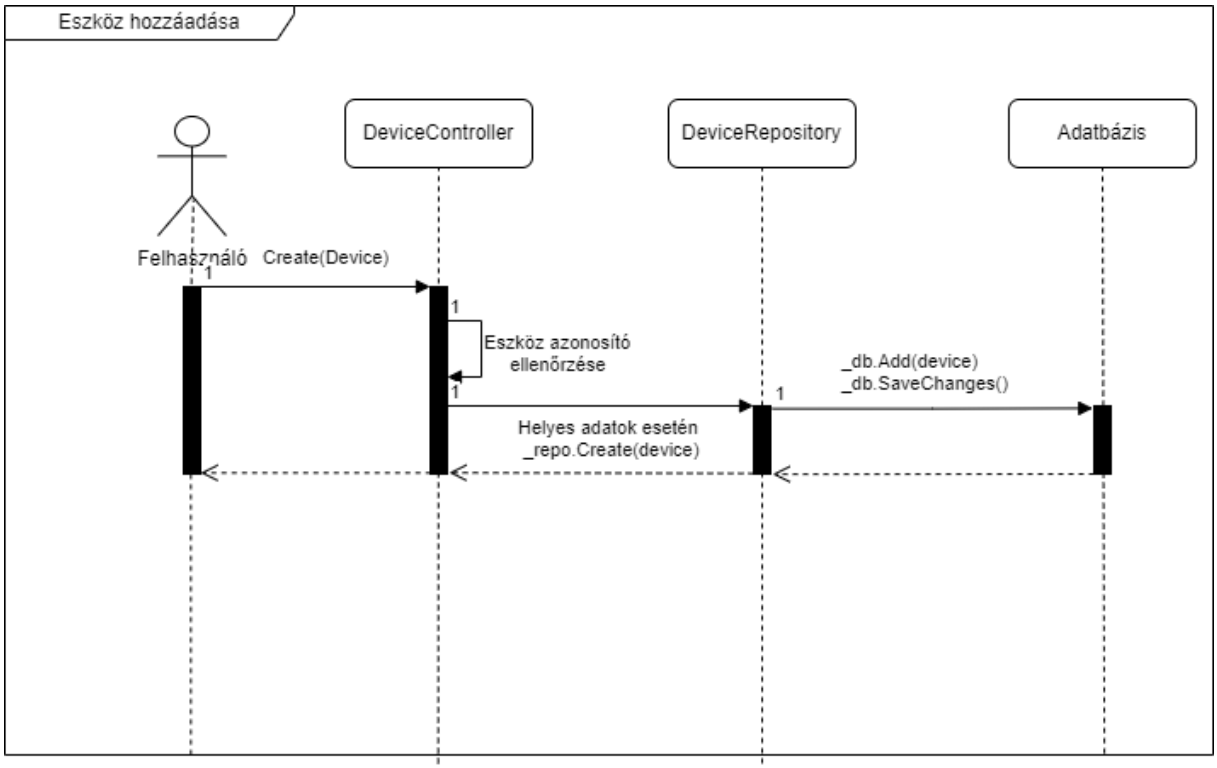
Ennek érdekében készíteni kell egy IdentityController-t, mely segítségével megoldható az elfelejtett jelszó kezelése.



13. ábra Elfelejtett jelszó folyamatábra

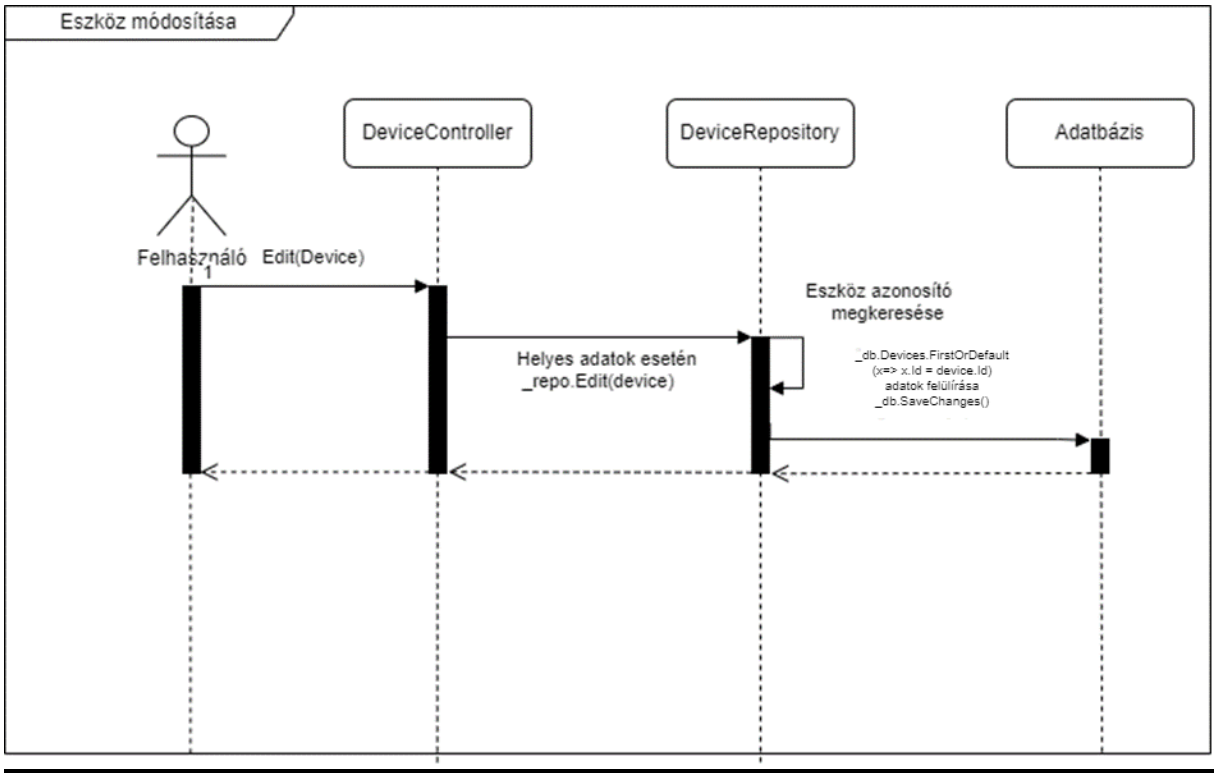
8.2 Eszköz műveletek

A felhasználó hozzá tud rendelni eszközöket a fiókjához, ehhez egy kiadott egyedi azonosító megadása is szükséges. A felhasználónak kötelező megadnia a nevet, illetve az egyedi azonosítót, a leírás opcionális. A szerveroldal ellenőrzi a küldött egyedi azonosítót és egyezés esetén rögzíti az eszközt, egyéb esetben hibát ad vissza a felhasználó számára.



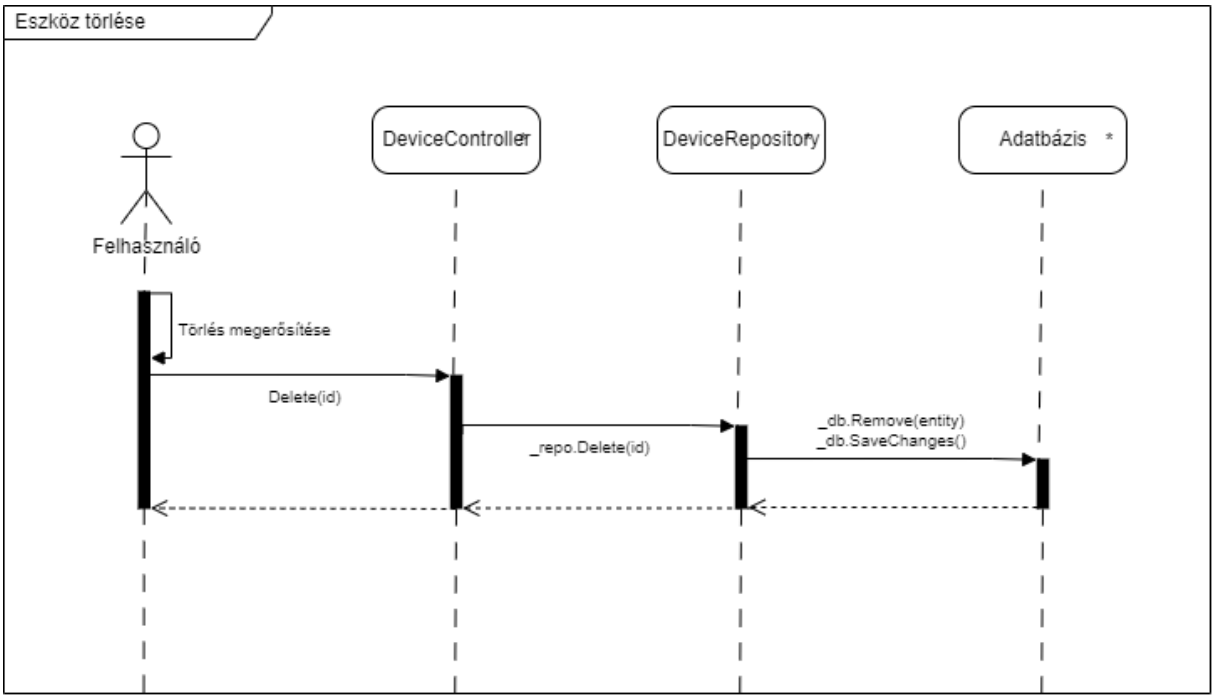
14. ábra Eszköz hozzáadása folyamatábra

Amennyiben módosítani szeretné az eszközeit, az erre megfelelő felületen tudja megtenni, azonban ebben az esetben az eszközhöz rendelt nevet, leírást és státuszt tudja csak módosítani.



15. ábra Eszköz módosítása folyamatábra

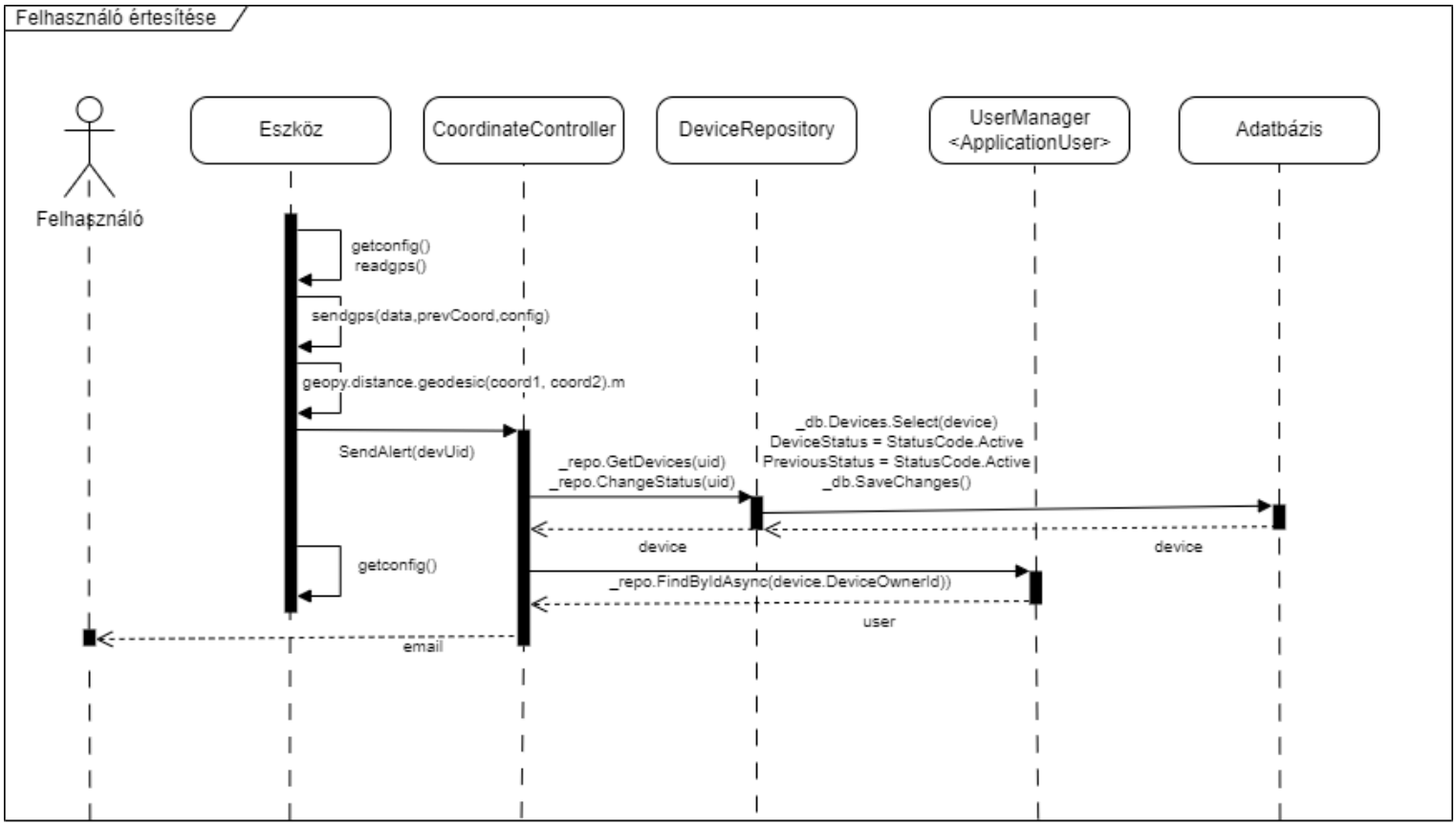
Amennyiben az eszköz már nem releváns számára, tudja törölni ezt az eszközt a felületéről.



16. ábra Eszköz törlése folyamatábra

8.3 Koordináta műveletek

A felhasználó és a szerver oldal között a koordináták esetében 2 közös pont van, az egyik, amikor a felhasználó megtekinti az eszközről eltárolt adatokat a felületen, a másik pedig amikor a szerver értesíti a felhasználót az eszköz megmozdításáról, ugyanis ilyenkor az eszköz a koordinátákat vizsgálja és csak akkor szól a szervernek, mikor nagy a két koordináta közötti különbség.

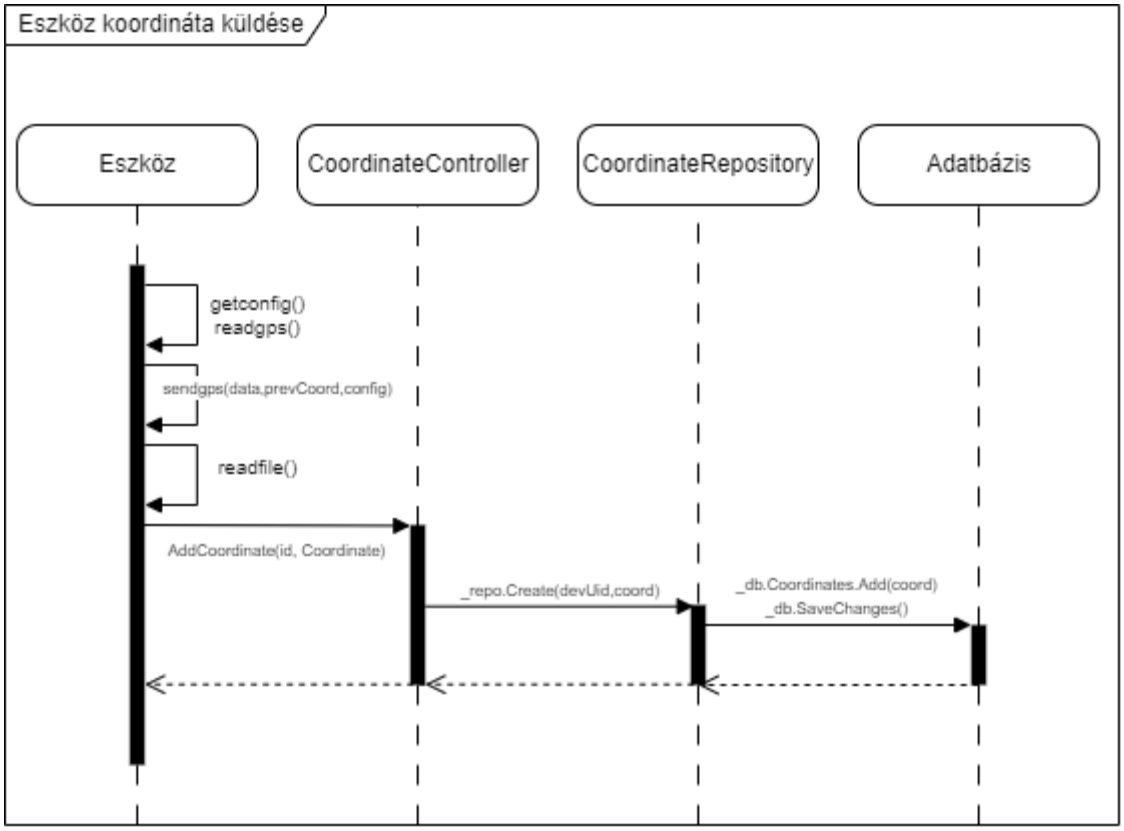


17. ábra Felhasználó értesítése folyamatábra

9. KOMMUNIKÁCIÓ AZ ESZKÖZ ÉS A SZERVER KÖZÖTT

9.1 Koordináta létrehozás

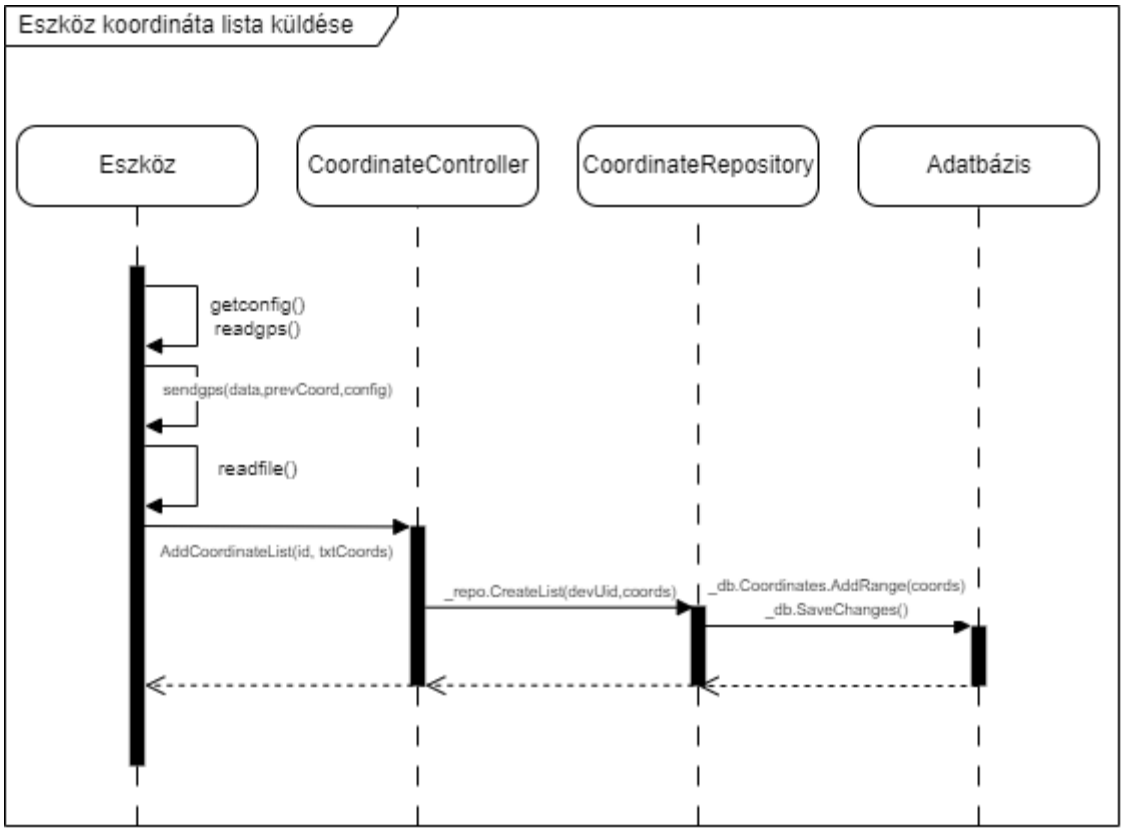
A szerver oldal biztosítja a koordináták fogadását az eszköz számára. Ehhez az eszköz az azonosítóján túl 3 adatok kell továbbítson, egy hosszúságot, egy szélességet és egy dátumot. Ezekből az adatokból készíti el az eszköz a koordináta objektumot, illetve az url-be beleágyazza egyedi azonosítóját is. Ezeket a kapott információkat a repository dolgozza fel. Az egyedi azonosító, melyet az eszköz továbbít a beazonosíthatóságért felel, ugyanis amennyiben a rendszerben nincs benne az azonosító, a koordináta nem kerül bele az adatbázisba, egyéb esetben a megfelelő eszközhöz kerül rögzítésre.



18. ábra Koordináta küldése folyamatábra

9.2 Koordináták listájának feldolgozása, eltárolása

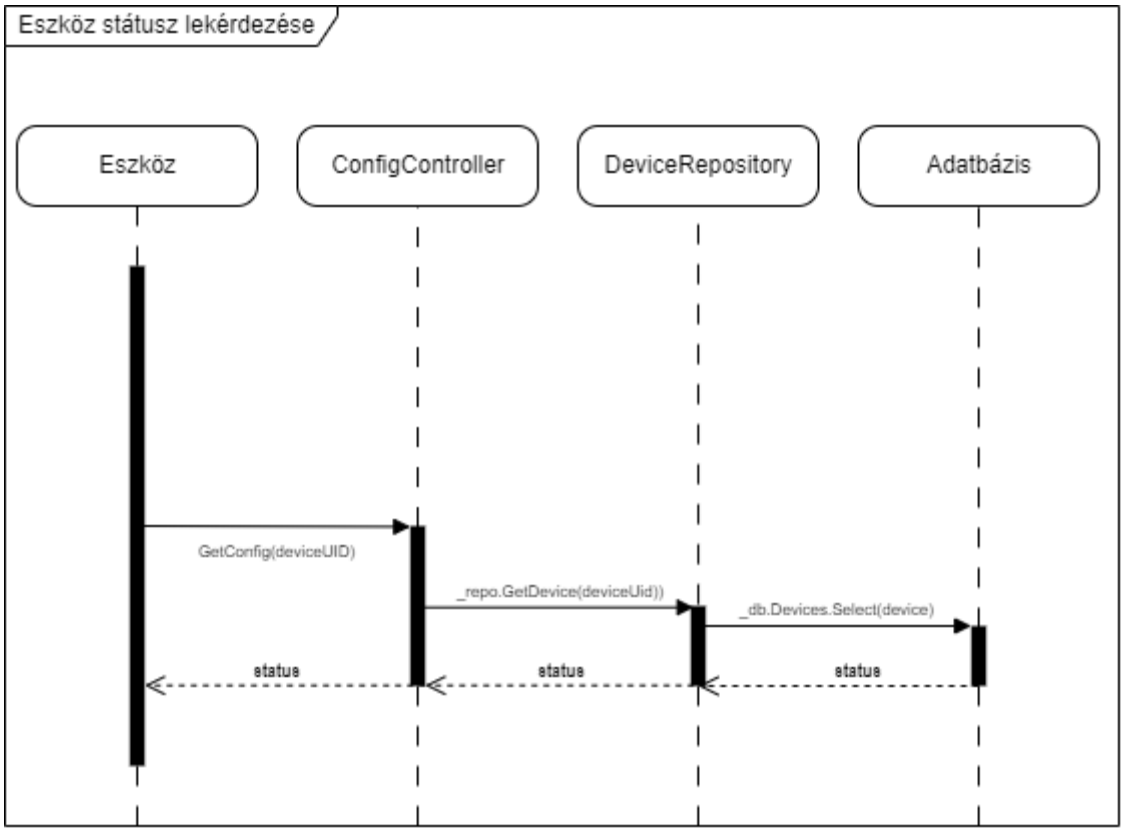
Amennyiben az eszköz elveszti a kapcsolatot a szerverrel, a koordinátákat lokálisan tárolja el, majd egyben továbbítja azt. A szerver képes kell legyen ezeknek a koordinátáknak a tömeges feldolgozására is. A korábbi koordináta létrehozás menetéhez hasonló a folyamat, annyi különbséggel, hogy itt a koordináták egy szöveg formájában érkeznek, melyet részekre kell bontani és a részekre bontott elemeit tárolni.



19. ábra Koordináta lista küldése folyamatábra

9.3 Státusz lekérés

Az eszköz időközönként elkéri a szervertől a státuszát, melyből be tudja azonosítani a koordináta küldés mechanizmusát. Ennek azért van szerepe, mert ha a felhasználó a felületen módosítja a státuszt, az eszközön is módosulnia kell annak megfelelően, ezzel is módosítva a koordináta küldési mechanizmust.



20. ábra Eszköz státusz lekérdezése folyamatábra

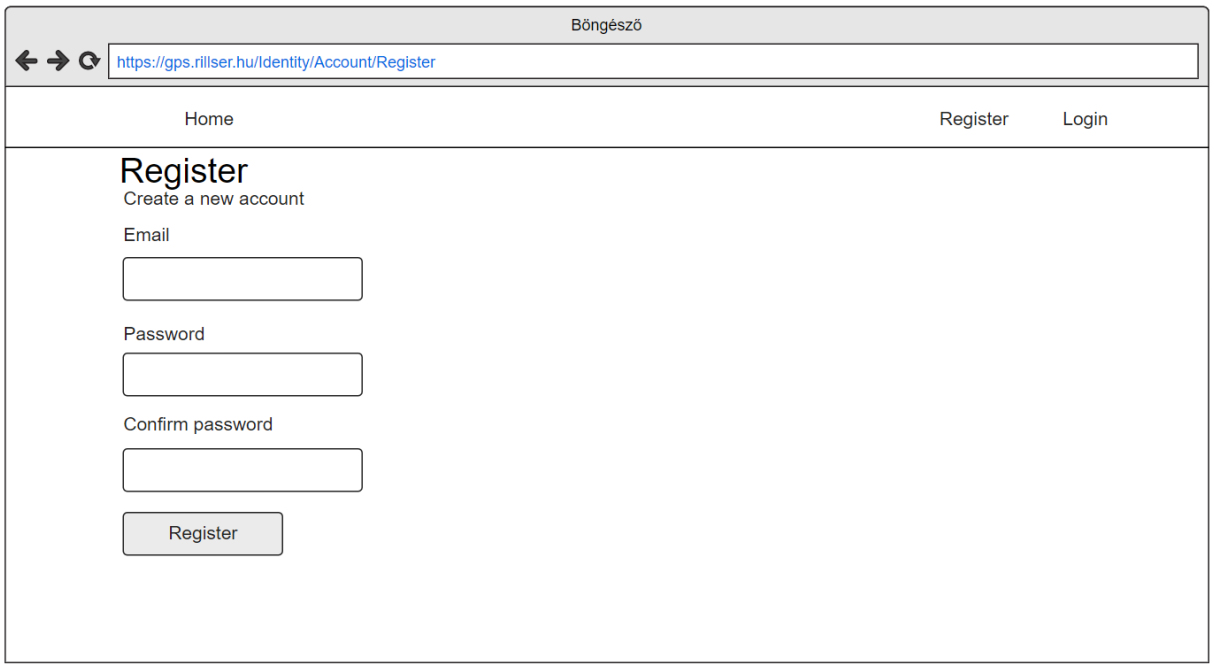
A szervernek a fogadott adatok időpontját előre deklarált időközönként ellenőrzie kell, amennyiben az eszköz a státuszának megfelelő küldési mechanizmus alapján nem küldött koordinátát hosszú ideje, módosítania kell „offline”-ra. Ellenkező esetben, ha a státusz „offline”, de az eszköz ismét küldött adatokat, a státuszát vissza kell állítsa a korábban elmentett állapotra. Erre egy „StatusService job”-ot célszerű megvalósítani.

10.FELHASZNÁLÓI FELÜLETEK

A felületek tervezésénél fontos szemszög számomra, hogy egyes felhasználók szeretnék az alkalmazást a mobil készülékükön használni, ugyanis szeretném, hogy telefonon is használható legyen az alkalmazás. Ehhez egy reszponzív applikációt célszerű készíteni, ami Bootstrap keretrendszert használ a HTML elemek megjelenítése során.

10.1 Regisztráció

A regisztrációs felületen van lehetősége a felhasználónak fiókot létrehozni. A regisztráció során szükséges megadni e-mai címet és jelszót, továbbá egy jelszó megerősítést, ezért a beviteli mezőkből is pontosan ugyanezekre van szükség, továbbá egy gombra, mellyel a kitöltött adatokat továbbítani tudjuk a szerveroldal felé.

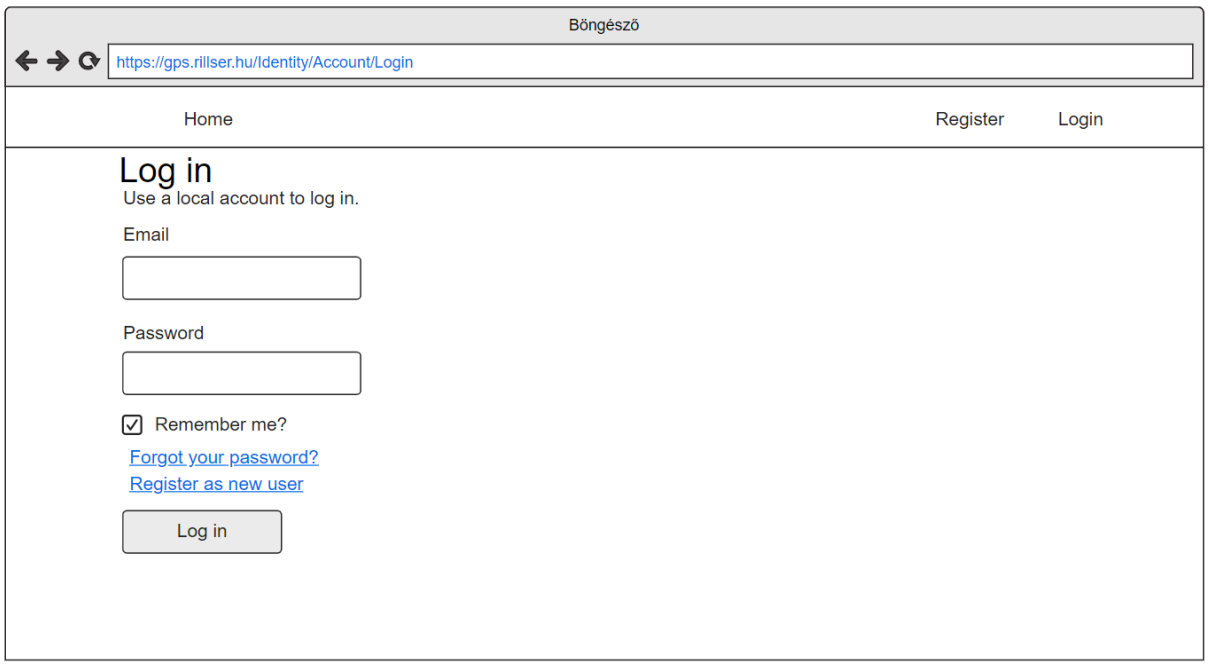


21. ábra Regisztráció wireframe

10.2 Bejelentkezés

Kell egy felület, ahol a felhasználó be tud lépni. A felületen két beviteli mező és egy gomb kell szerepeljen. A beviteli mezők egyike a felhasználónevet várja, a másik pedig a jelszót. A gombra kattintva tud a felhasználó bejelentkezni a felületre

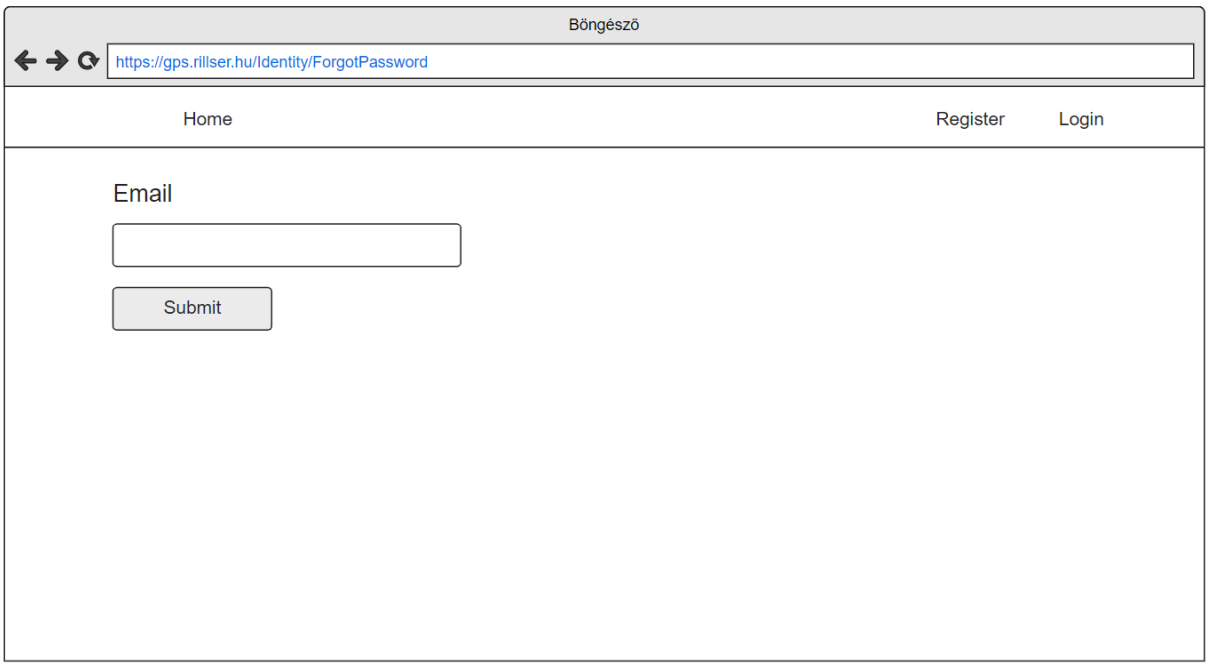
Ezentúl lehetőséget kell biztosítani az elfelejtett jelszó esetében új jelszó igénylésére, így kell egy link erre is. Amennyiben a felhasználónak még nincs fiókja, biztosítani kell számára egy linket, amely a regisztrációs felületre irányítja.



22. ábra Bejelentkezés wireframe

10.3 Elfelejtett jelszó

A felhasználó egy e-mail cím megadásával a küldés gombra kattintva tud magának jelszó emlékeztető linket küldeni, így ezen felületen 2 elemre van szükségünk, egy e-mail címre, továbbá egy gombra.

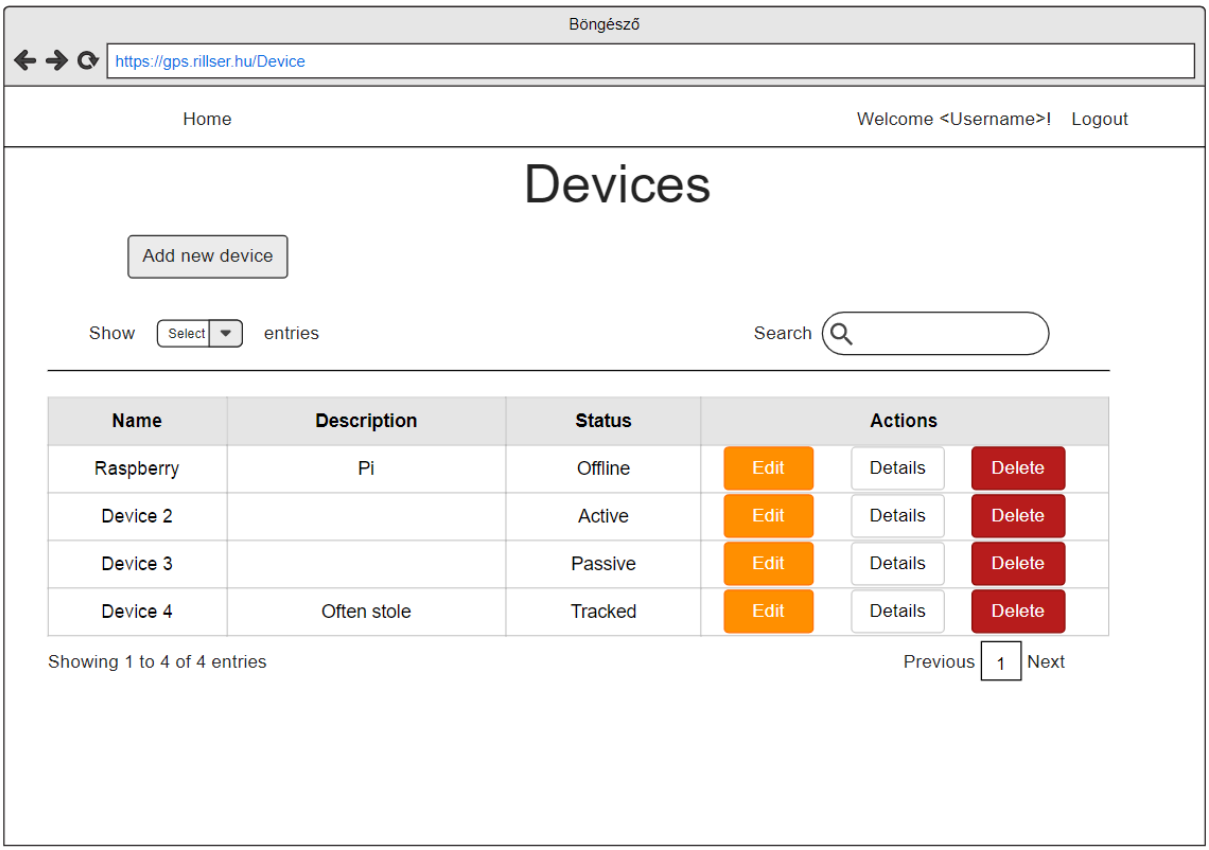


23. ábra Elfelejtett jelszó wireframe

10.4 Eszközök

Bejelentkezést követően a felhasználónak több lehetősége is nyílik, kezdésképp az eszközök, ahol új eszköz felvételére van lehetősége, továbbá a hozzáadott eszközeinek megtekintésére, szerkesztésére, koordinátáinak megtekintésére, illetve az eszköz törlésére.

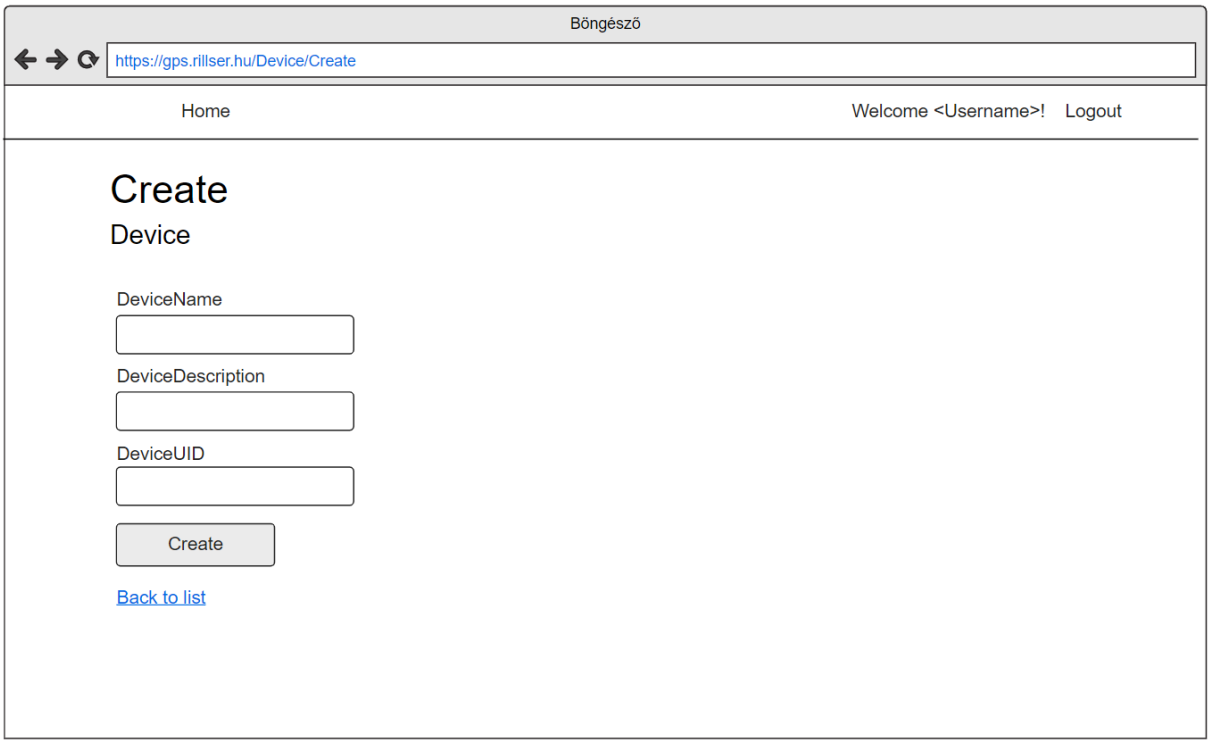
Ügyelni kell arra, hogy egy felhasználónak több eszköze is lehet, így ezt az oldalt célszerű lapozhatóvá tenni, ha esetleg valakinek túl sok eszköze lenne, akkor átláthatóan tudja kezelni őket. Biztosítani kell a felhasználó számára rendezést és keresést is az eszközökre, ezzel is a felhasználói élményt és a könnyű kiigazodást javítva. Valamint kiválaszthatja a felhasználó hány eszköz jelenjen meg egy oldalon.



24. ábra Eszközök wireframe

10.5 Eszköz felvétele

A felhasználónak lehetősége van a birtokában lévő eszköz rögzítésére a profiljában, ezt egy egyedi azonosító és egy név megadásával teheti meg, opcionálisan leírást is adhat meg, így 3 beviteli mezőt kell biztosítani számára, továbbá egy létrehozás gombot. Amennyiben mégsem szeretne eszközt létrehozni, visszaléphet az eszközök oldalra egy linkre kattintva.

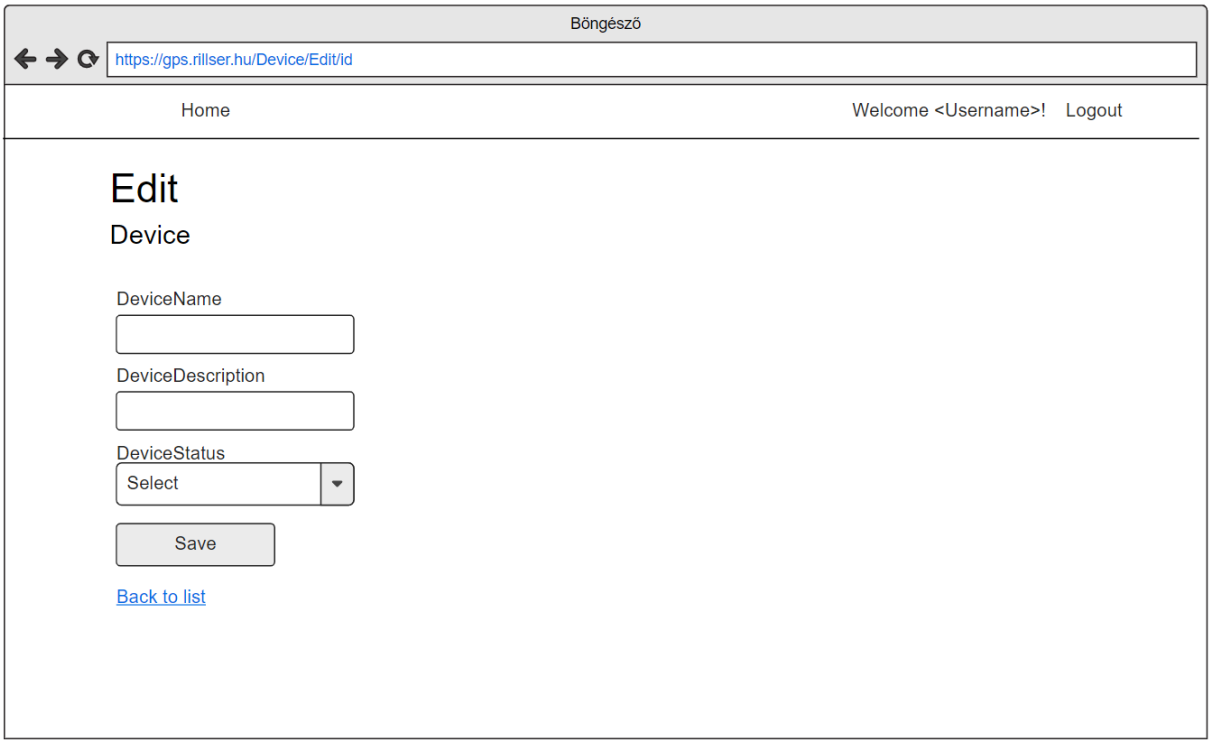


25. ábra Eszköz felvétel wireframe

10.6 Eszköz szerkesztése

Az eszköz szerkesztése esetében 3 elemet szerkeszthet a felhasználó, a nevet, a leírást, illetve a státuszt. A státusz szerkesztésére csak akkor szabad lehetőséget biztosítani a felhasználónak, amennyiben az eszköz elérhető, egyéb esetben azt a mezőt nem módosíthatja.

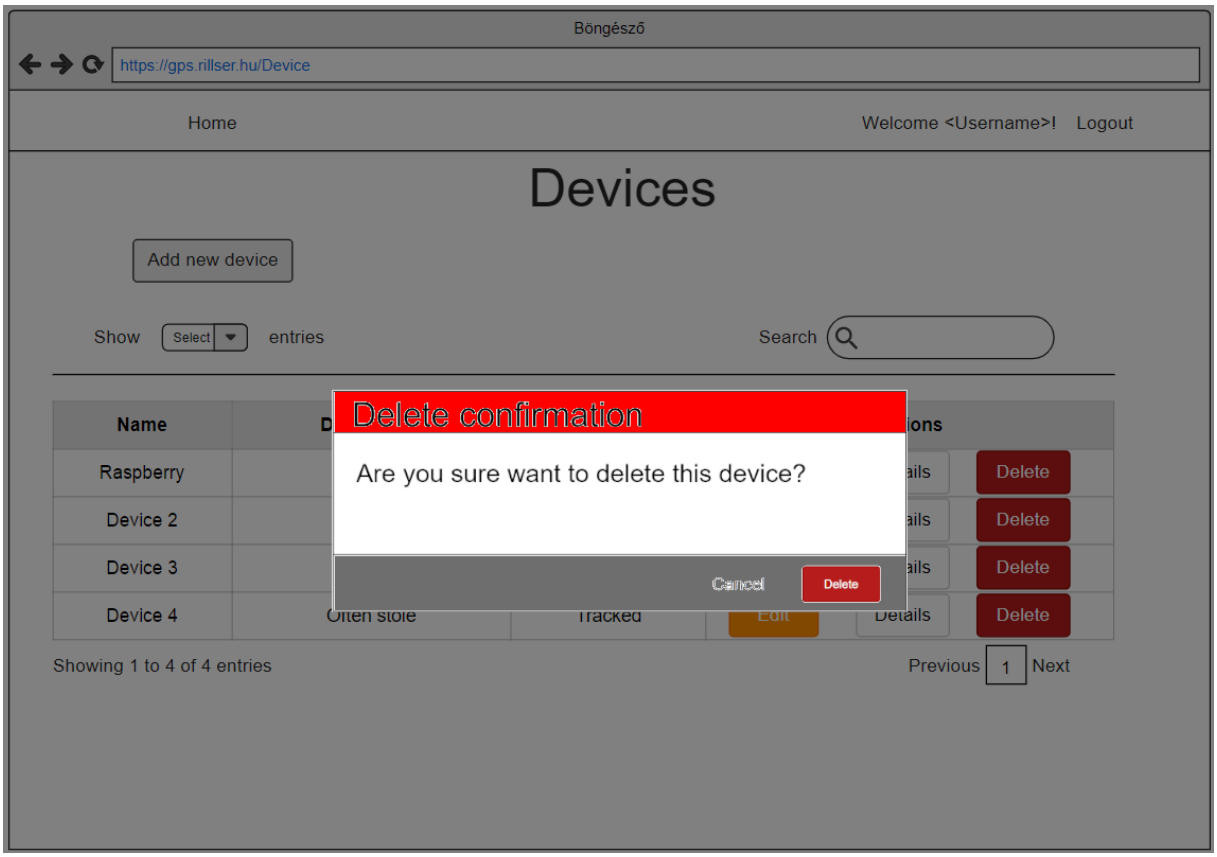
Biztosítani kell a felhasználó számára egy mentés gombot, amivel frissítheti az eszköz adatait, illetve egy linket, mellyel vissza tud lépni az előző oldalra, ha esetleg mégsem akarja módosítani az eszközt.



26. ábra Eszköz szerkesztése wireframe

10.7 Eszköz törlése

Amennyiben a felhasználó törölni szeretne egy eszközt, megerősítését kell kérni ezzel kapcsolatban, hiszen előfordulhat, hogy véletlenül nyom a törlés gombra és így meg lehet előzni a későbbi bonyodalmakat.



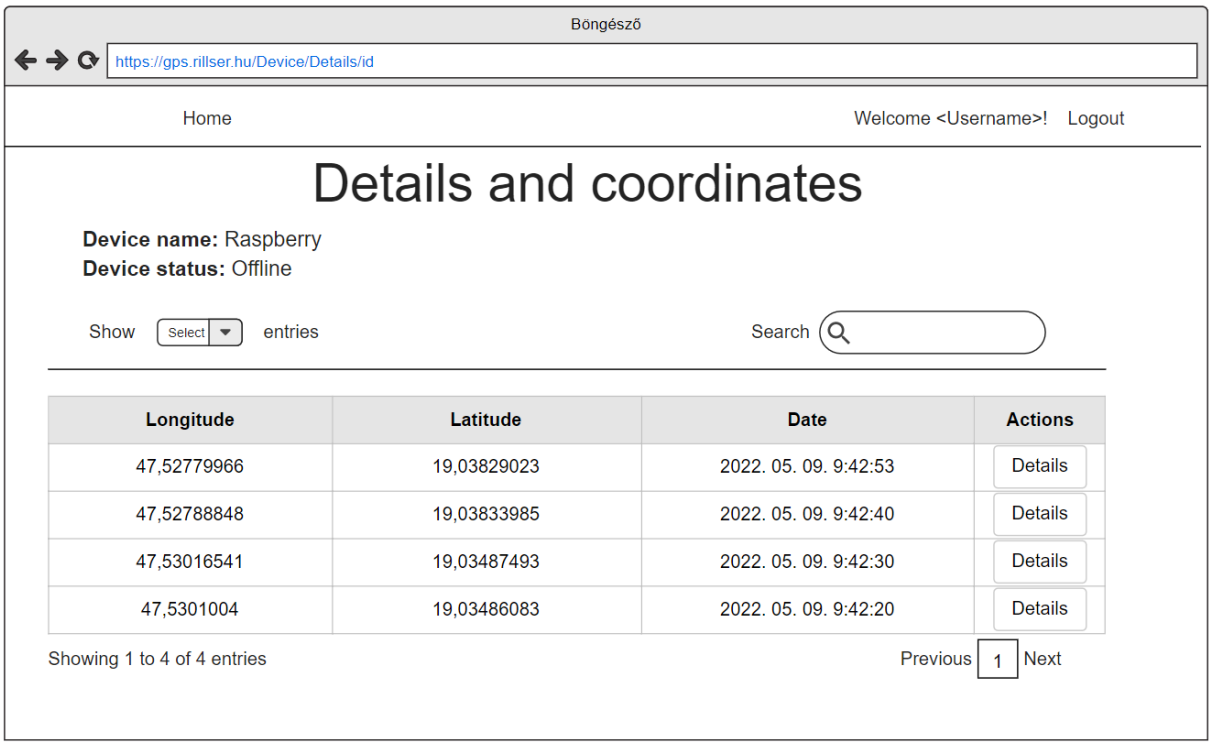
27. ábra Eszköz törlése wireframe

10.8 Eszköz információk és koordináták megtekintése

Az eszközök részletek gombjára kattintva a következő felület amire lépünk a koordináták megtekintése kell, hogy legyen. Ez a felület tartalmazza az eszköz nevét és státuszát is, a koordináták szélességi és hosszúsági fokát, a hozzájuk tartozó dátumot, illetve egy részletek gombot is, amire kattintva egy kis ablak jelenik meg, ahol grafikusan is megtekinthetik a koordináta pozícióját a térképen.

Szintén nagyon fontos, hogy egy eszköznek rengeteg koordinátája van, így ezt az oldalt kötelező lapozhatóvá tenni, hogy a felhasználó könnyebben átlássa a felületet. Biztosítani kell a felhasználó számára rendezést és keresést is a koordinátákra, valamint kiválaszthatja a felhasználó hány koordináta jelenjen meg egy oldalon.

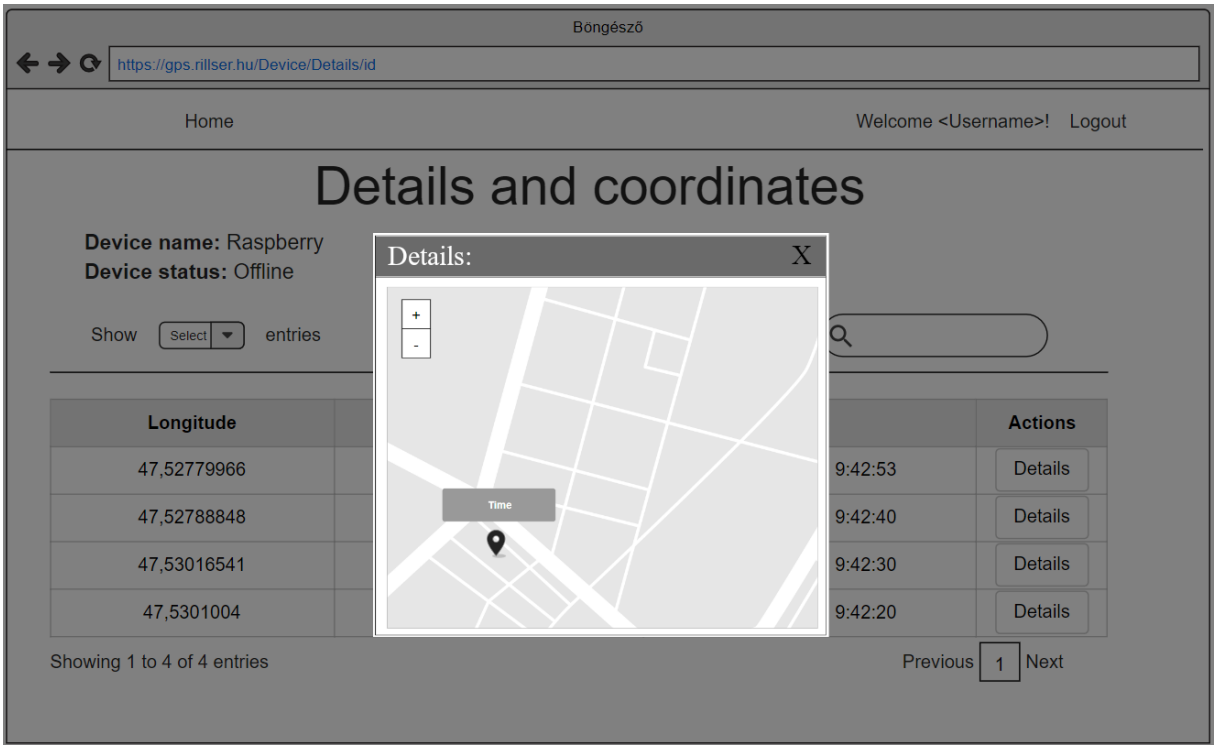
Előfordulhat, hogy a felhasználó vissza szeretne lépni eszközeihez, így szükséges egy gomb is, ami visszavisz az eszközök felületére.



28. ábra Eszköz adatok és koordináták wireframe

10.9 Eszköz információk és koordináták mini térkép megjelenítése

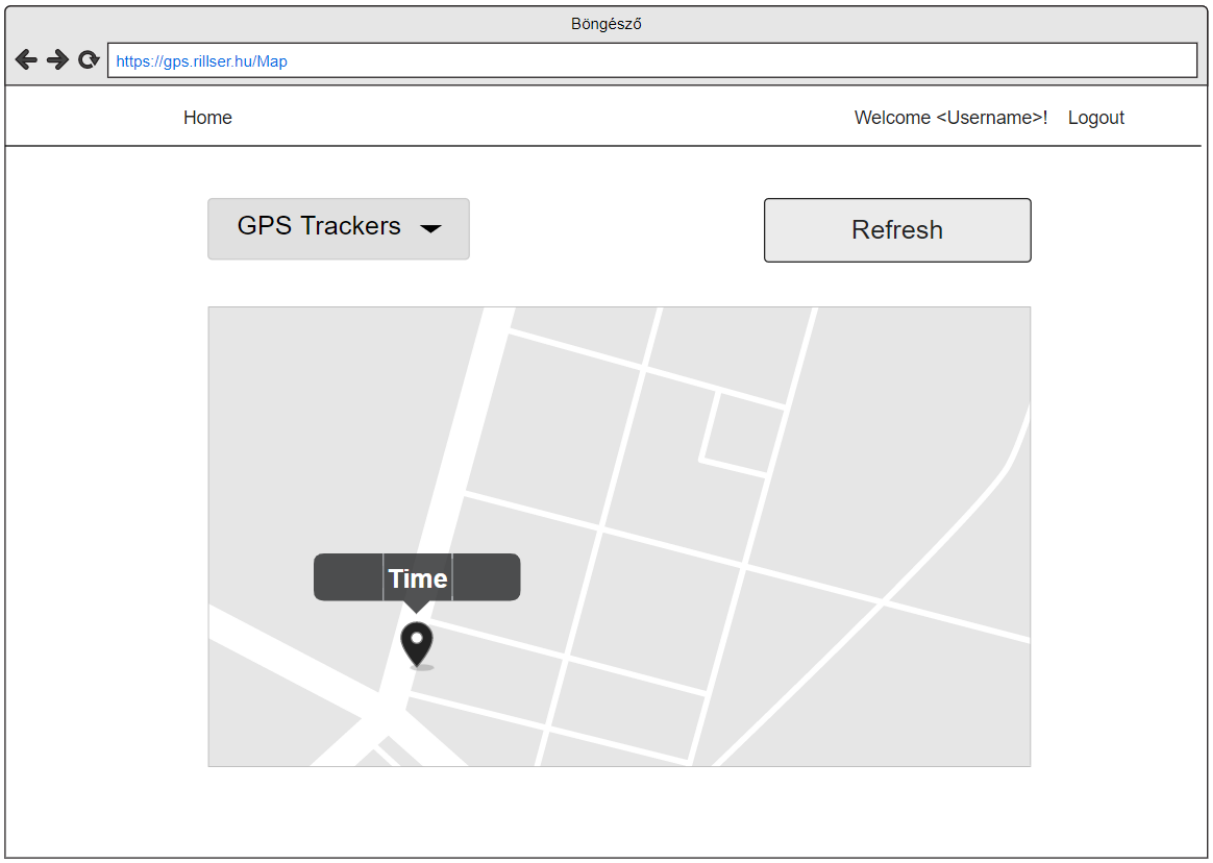
Az eszköz információk és koordináták oldalon található egy részletek gomb, melyre kattintva a felhasználó megtekinthetni az adott koordinátát egy térképen, illetve a koordináták visszaellenőrzését, a jármű haladását.



29. ábra Mini térkép wireframe

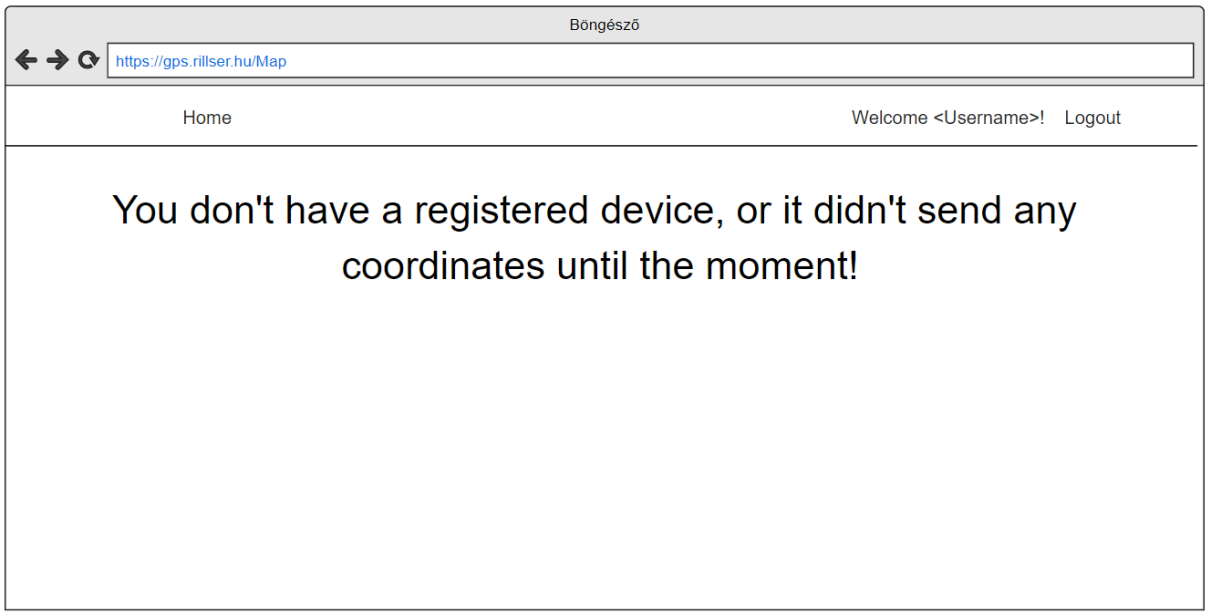
10.10 Fő térkép oldal

Készíteni kell egy olyan felületet, ahol a felhasználók megtekinthetik az összes eszközükhöz tartozó utolsó koordinátát, ezzel is meggyorsítva az eszközök pozíciójának ellenőrzését. Biztosítani kell egy legördülő menüt, ahol kiválaszthatják az adott eszközt, illetve egy gombot, amivel frissíthetik az oldalt, hogy mindig a legfrissebb koordinátát láthassák az eszközről.



30. ábra Fő térkép oldal wireframe

Ezen a felületen csak akkor jelenik meg térkép illetve eszköz, ha ténylegesen rendelkezik eszközzel a felhasználó, illetve az eszköz már küldött koordinátát az applikációnak, egyéb esetben egy szöveget jelenít meg, hogy a két eshetőségből valamelyik nem teljesül.



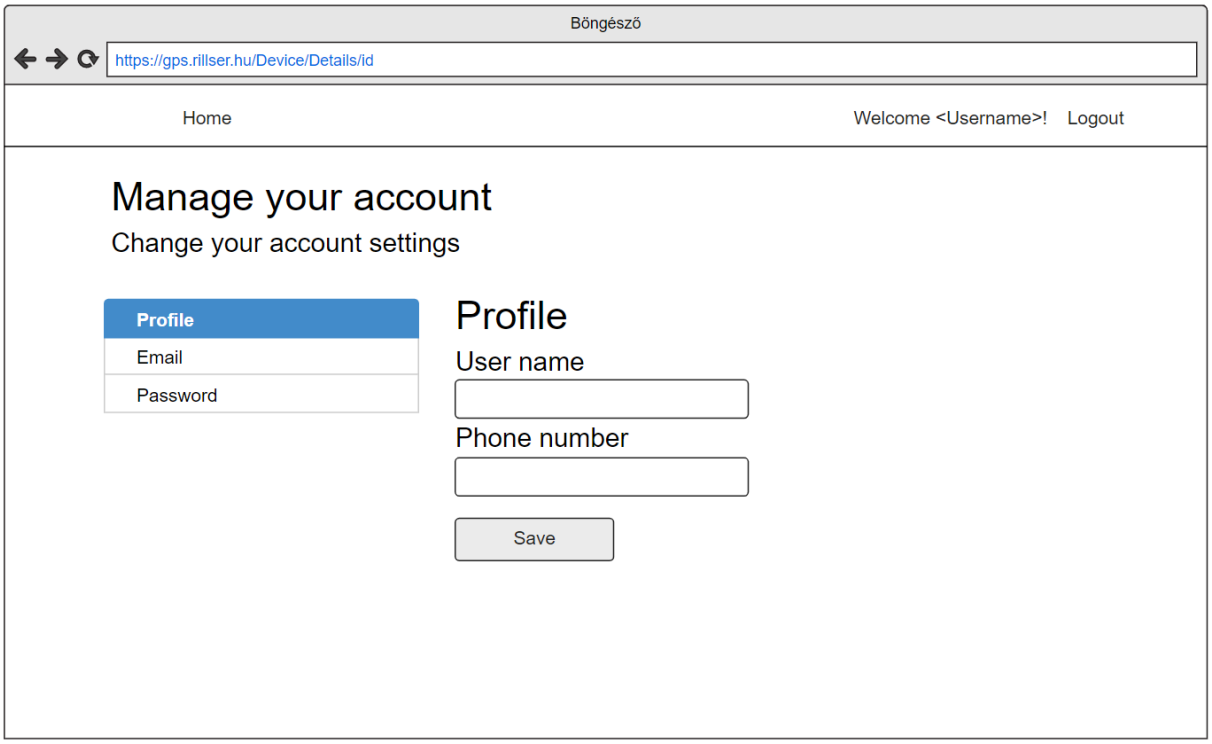
31. ábra Fő térkép eszköz vagy koordináta nélküli wireframe

10.11 Felhasználó menedzsment felület

A felhasználónak lehetőséget kell biztosítani saját adatainak szerkesztésére.

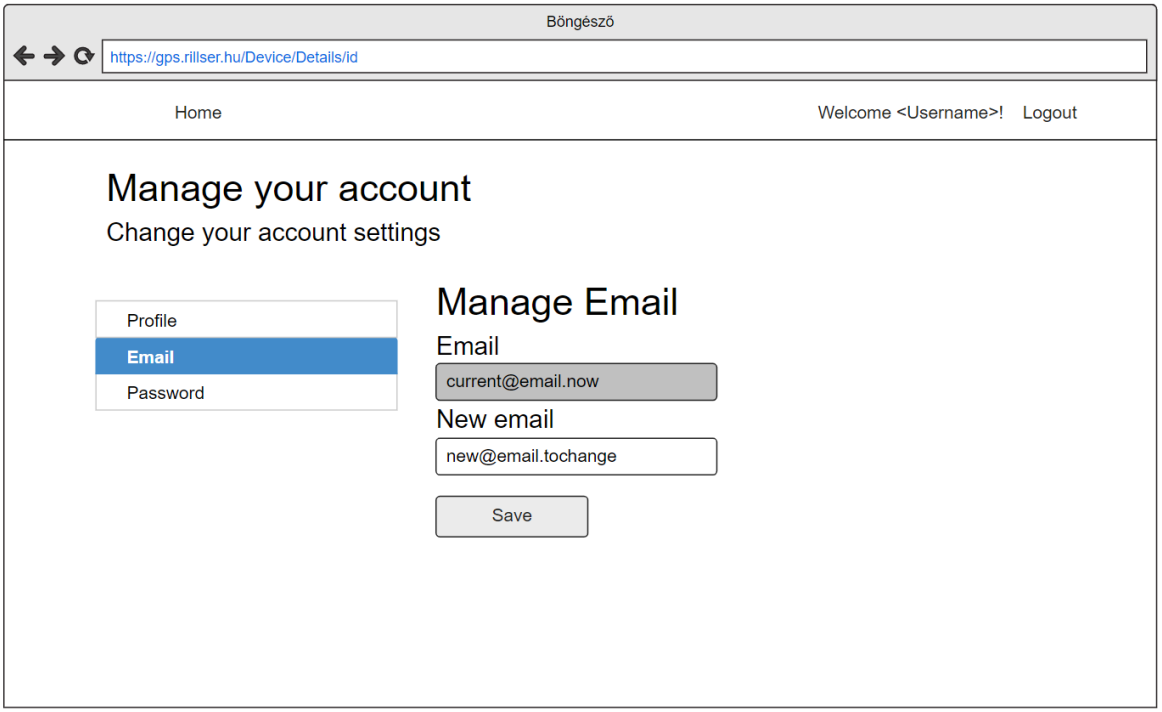
Három részre lehet bontani ezt a felületet: profil adatok (felhasználónév, telefonszám) módosítása, e-mail cím módosítása, jelszó módosítása.

A profil adatok felületen 2 beviteli mezőt és egy mentés gombot kell biztosítani a felhasználó számára.



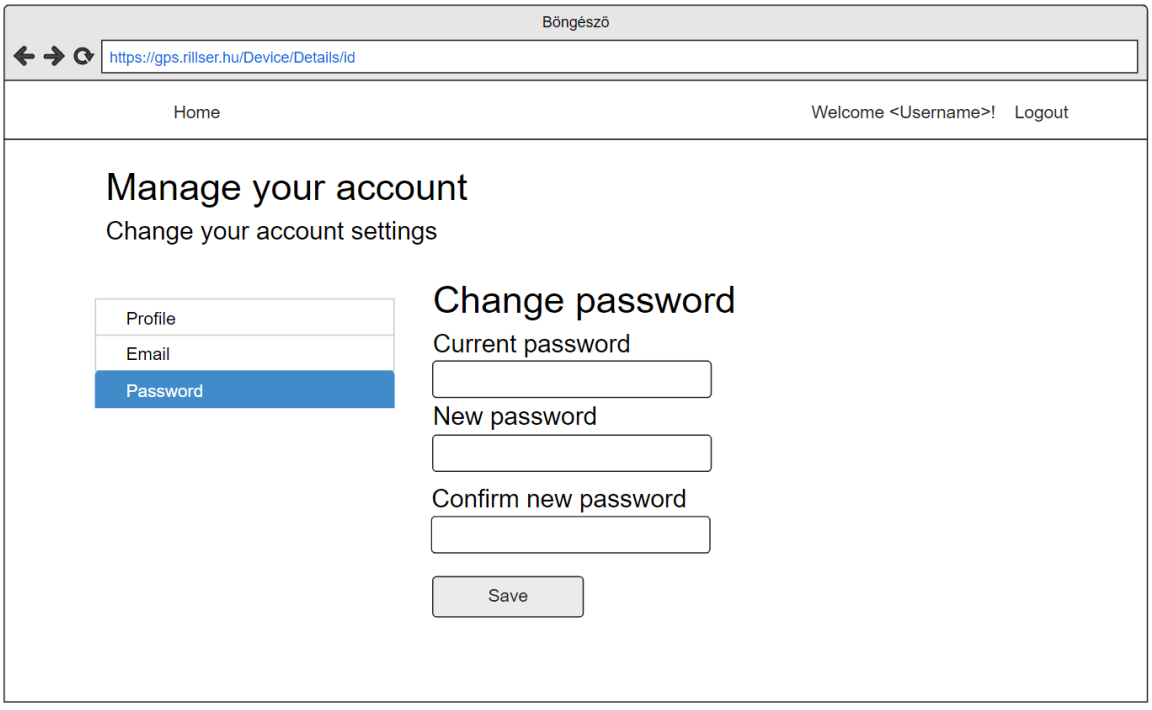
32. ábra Profil adatmódosítás wireframe

Az e-mail cím felületen célszerű megjeleníteni a jelenlegi e-mail címét, valamint biztosítani kell számára egy mezőt, ahova új e-mail címét beírhatja, továbbá egy gombot, amivel mentheti ezeket az adatokat. Ahhoz, hogy az e-mail cím módosítás megtörténhessen, meg kell erősítenie a felhasználónak ezt a kiküldött e-mailben szereplő linkre kattintva.



33. ábra E-mail cím módosítása wireframe

A jelszó módosítás esetében elsősorban el kell kérni a jelenlegi jelszót, továbbá az új jelszót, valamint megerősítésképp az új jelszó ismételt beírását is elvárjuk a felhasználótól. Ebből kifolyólag 3 beviteli mezőre, illetve egy mentés gombra van szükség ezen a felületen



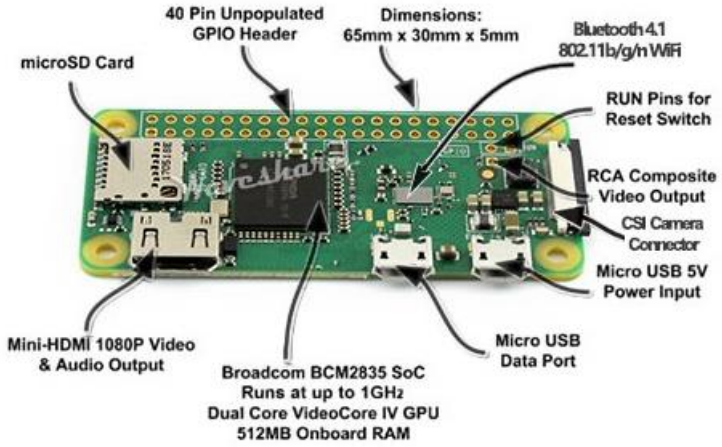
34. ábra Jelszó módosítás wireframe

11.AZ ALKALMAZÁS MEGVALÓSÍTÁSA

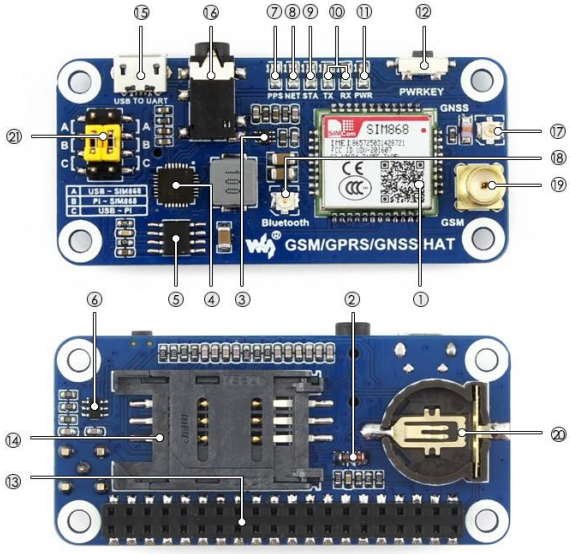
11.1 Eszköz

11.1.1 Eszlözök beszerzése

A GPS nyomkövető projekt egy eszközre épül, mely ki lesz telepítve egy járműre, így elsősorban ennek az eszköznek a kiválasztására és beszerzésére volt szükség. Mint korábban a dokumentáció során említettem, egy Raspberry Pi Zero W-re esett a választás, továbbá szükség volt egy GSM Hat-re is, mely a GPS kapcsolat kiépítésére, illetve a SIM kártyás internet elérés biztosítására szolgál.



36. ábra Raspberry Pi Zero W



1. SIM868 module
2. ZMM5V1: regulator diode
3. SMF05C: TVS diode
4. CP2102: USB TO UART converter
5. MP1482: power chip
6. NDC7002N: voltage level translator
7. GPS status indicator
8. NET indicator:
    1. gyorsan villog, amikor a modul elindult
    2. lassan villog, sikeres GSM regisztráció után
9. STA module working status indicator
10. SIM868 UART Tx/Rx indicator
11. Power indicator
12. SIM868 control button: press the button and hold for 1s, to startup/shutdown the SIM868
13. Raspberry Pi GPIO connector
14. SIM card slot
15. USB TO UART interface
16. 3.5mm earphone/mic jack
17. GNSS antenna connector
18. Bluetooth antenna connector
19. GSM antenna connector
20. RTC backup battery holder
21. UART selection switch
    1. A: control the SIM868 through USB TO UART
    2. B: control the SIM868 through Raspberry Pi
    3. C: access Raspberry Pi through USB TO UART

35. ábra GSM Hat

11.1.2 Eszköz funkcionalitásának megvalósítása

A Hat GPRS kapcsolaton keresztül képes internet kapcsolatot létesíteni, ennek kiépítése érdekében Python scriptet kellet készítenem, melyet az eszköz Serial0 portjára építettem ki. a következő kóddal:

```
import serial
import os, time

# Enable Serial Communication
port = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1)

# Transmitting AT Commands to the Modem
# '\r\n' indicates the Enter key

port.write(b'AT\r\n')
rcv = port.read(10)
print(rcv)

```

37. ábra Python script az eszköz GPRS internet kapcsolatához

Be kellett állítani a szolgáltató által használt APN-t is, hogy a mobilinternet működhessen. Tekintettel arra, hogy a SIM kártyát használok, így csak a szolgáltató által megadott APN („net”) kódot kellett beállítanom:

```
net is the apn for Yettel connection
connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs -T net"

# For Raspberry Pi3 use /dev/ttyS0 as the communication port:
/dev/ttyS0

# Baudrate
9600

# Assumes that your IP address is allocated dynamically by the ISP.
noipdefault

# Try to get the name server addresses from the ISP.
usepeerdns

# Use this connection as the default route to the internet.
defaultroute

# Makes PPPD "dial again" when the connection is lost.
persist

# Do not ask the remote to authenticate.
noauth

# No hardware flow control on the serial link with GSM Modem
nocrtscts

# No modem control lines with GSM Modem
local
```

38. ábra APN kiépítése

Magát a Serial0 porton való internetkapcsolatot a „**sudo pon rnet”** parancs kiadásával kapcsoltam be.

Elsősorban be kellett kapcsolni a GPS modult, beállítani a sávszélességet, majd olvasni a portról a megfelelő „AT” parancsok kiadásával.

Ezután kezdődhetett a megfelelő portról érkező GPS koordináta feldolgozás, illetve a műveletek kiépítése.

```
def readgps():
    fd = port.read(port.inWaiting())        # Read the GPS data from UART
    #print fd
    time.sleep(.5)
    if b'$GNRMC' in fd:         # To Extract Lattitude and
        ps=fd.find(b'$GNRMC')         # Longitude
        dif=len(fd)-ps
        if dif > 50:
            data=fd[ps:(ps+50)]
            if data.split(b',')[2]!= b'A':
                print('Invalid data.')
            else:
                return(data)
    return
```

39. ábra Serial portról RMC adat olvasás

A megvalósítás során derült ki, hogy a port nem konkrét hosszúsági és szélességi fokokat továbbított, hanem RMC koordinátákat, melyeket külön át kell alakítani GPS koordinátákká, méghozzá úgy, hogy az érkező adat egész számait 100-al, míg tört részét 60-al kell osztani, így kapjuk meg a ténylegesen értelmezhető GPS kooridnátát.

A koordinátákat továbbítani kell a szervernek, illetve amennyiben az eszközt megmozdítják (utolsó két koordináta között a különbség nagyobb mint 50 méter) riasztást kell küldjön a szerver felé, melyet szintén egy API hívással tesz. Amennyiben a koordináta küldése sikertelen, azon esetben eltárolja egy „failed_coords” nevű fájlban.

A kimentett koordinátákat minden ciklusban megpróbálja újra kiküldeni, amennyiben a fájl üres azon esetben kiírja számunkra a konzolon, hogy nincs mit elküldenie, ha a kézbesítés sikertelen volt meghagyja a koordinátákat, egyéb esetben kiüríti a tartalmát a „failed_coords”-nak.

Minden ciklus elején a státuszát ellenőrzi, mellyel egyrész érzékeli a szerver, hogy él az eszköz, másrészt friss információkat szerez állapotáról és eleget tud tenni az állapotának megfelelő adatküldésnek. Amennyiben nem kapcsolódik az internethez az eszköz, vagy a szerver válik elérhetetlenné, a lokálisan tárolt státuszát használja.

```
def getconfig():
        context = ssl._create_unverified_context
        headers = {'Content-type':'application/json', 'Accept':'application/json'}
        try:
            response = requests.get(url+'/api/Config/GetConfig?devUid='+str(mac_addr), headers=headers, verify =False, timeout=10)
            if response.status_code == 200:
                config = response.text.replace('"', "")
                with open('data.txt', 'w', encoding='utf-8') as file:
                    file.writelines(response.text.replace('"', ""))
                    file.close()
                    print(config)
                return(config)
            else:
                print('Connection lost. Cannot get config.')
                f = open("data.txt", "r")
                config = f.readline()
                f.close()
                return(config)
        except:
            print('Connection lost. Cannot get config.')
            f = open("data.txt", "r")
            config = f.readline()
            f.close()
            return(config)
```

40. ábra Státusz lekérése API-n keresztül

Időközönként muszáj frissíteni a portról olvasott adatokat, legfőképp azon esetben, amikor passzív, illetve aktív státuszban van az eszköz, hogy annak ellenére, hogy a koordináta küldési ideje redukálódik, mégis a legpontosabb információkkal szolgálhasson

```
def refresh():
    port.write(b'AT+CGNSIPR=115200\r\n') # Set the baud rate of GPS
    rcv = port.read(port.inWaiting())
    #print(rcv)
    time.sleep(.5)

    port.write(b'AT+CGNSTST=1\r\n')     # Send data received to UART
    rcv = port.read(port.inWaiting())
    #print(rcv)
    time.sleep(.5)

    port.write(b'AT+CGNSINF\r\n')        # Print the GPS information
    rcv = port.read(port.inWaiting())
    #print(rcv)
    time.sleep(.5)
```

41. ábra Portról érkező adatok frissítése

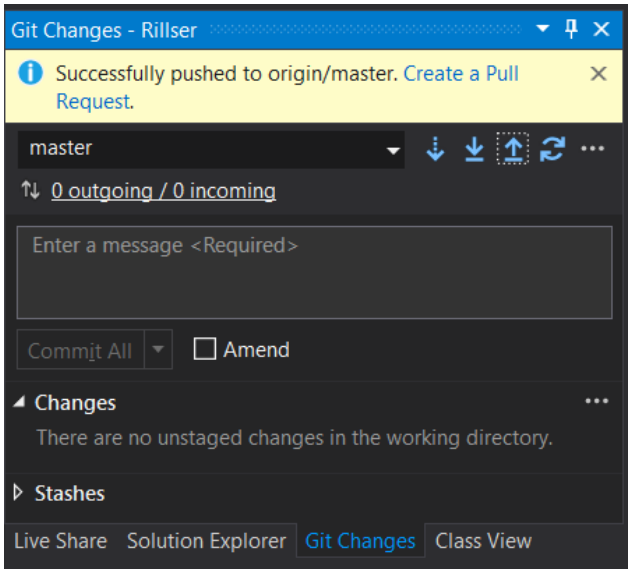
A fentebb deklarált folyamatokat ismételgeti az eszköz előre deklarált funkcióknak megfelelően, aktív státuszban 1 percenként koordinátát küldve, passzív státuszban 10 percenként.

11.2 Alkalmazás megvalósítása

11.2.1 Fejlesztéshez szükséges környezet kialakítása

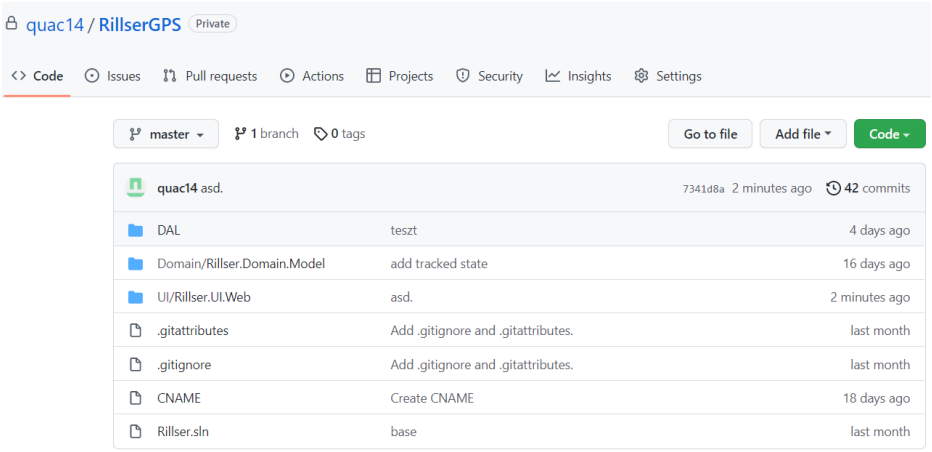
A szerver oldali alkalmazás és a webes felület fejlesztéséhez szükségem volt egy fejlesztői környezet kiépítésére. Mivel a projekt során ASP.Net Core keretrendszert szerettem volna használni, így szükség volt egy szoftver fejlesztői csomag beszerzése.

Szükségem volt egy szoftverre, melyben a kódot meg tudtam valósítani, melyre a Visual Studio 2019-es kiadását használtam, hiszen egyetemi tanulmányaim során is ebben szereztem tapasztalatot.



42. ábra Visual Studio Git

Mivel a kódomat szerettem volna átlátni, illetve verziókövetni, esetleges hiba során visszaállítani egy korábbi verzóra, célszerűnek tartottam egy verziókövető szoftver bevezetését a rendszerbe. A verziókövető szoftver esetében a Visual Studio által biztosított Git kezelőt használtam, illetve létrehoztam egy privát GitHub repository-t a projekt számára „rillserGPS” néven.



43. ábra GitHub repository

11.2.2 Implementáció

A megvalósítás első fázisában kulcsfontosságú volt az alkalmazás és az adatbázis kapcsolatának kiépítése, így első lépésként ezzel foglalkoztam. Ezt az alkalmazás „appsettings.json” nevű fájlában tudtam beállítani a „ConnectionString” szekcióban. A fejlesztés alatt lokális adatbázist használtam, később ezt migráltam az Azure-be.

A következő mérföldkő az volt, hogy kialakítsak egy osztály hierarchiát az alkalmazásban. Létrehoztam a két fő osztályom (Eszközök, Koordináták) kezelésére egy-egy interfészt, majd ezeket örökítettem tovább a megfelelő repository kezelőknek, melyben szintén módosítani kellett a metódusokat.

```
public interface ICoordinateRepository
{

    Coordinate GetCoordinate(int DeviceId);

    void Create(string devUid, DAL.Entity.Coordinate coord);

    List<DAL.Entity.Coordinate> GetCoordinates();

    void CreateList(string devUid, List<DAL.Entity.Coordinate> coords);
}
```

```
public class CoordinateRepository : ICoordinateRepository
```

44. ábra Koordináta interfész és repository

Ezt követően az API végpontok fejlesztésébe kezdtem, hiszen az eszköznek kommunikálnia kellett valamivel, hogy tesztelni lehessen a működését. Elsősorban a koordináták küldéséhez szükséges végpontokat készítettem el, melyből két fajta megoldásra is szükség volt. Elsősorban az átlagos, egyesével küldött koordináták, majd ezt követően a megszűnt kapcsolat miatt tömegesen beérkező koordináták fogadása volt a cél.

Tekintettel arra, hogy az eszköz kontrollerrel csak hitelesített felhasználók kommunikálhatnak, így a kontrollert tetején elláttam egy Authorize attribútummal, amivel azt értem el, hogy az API-kat csak bejelentkezést követően tudják elérni a felhasználók, ezzel is egy biztonsági rést kiaknázva.

```
[Authorize]
1 reference
public class DeviceController : Controller
```

45. ábra Authorize attribútum az eszköz kontrolleren

A koordináta kontroller esetében sajnos nem lehetséges az egész kontrollerre érvényesíteni az Authorize attribútumot, hisz ezzel a korntollerrel az eszköznek is kommunikálnia kell, így csak az egyet metósudokat láttam el különböző attribútumokkal, ellenben mivel a koordinátákat az eszköz kontroller segítségével jelenítjük meg, így a koordináta kontroller teljes mértékben API kontrollernek számít, ezért elláttam egy ApiController attribútummal, illetve beállítottam egy elérési utat az API-k tesztelésének megkönny ítése érdekében.

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class CoordinateController : ControllerBase
```

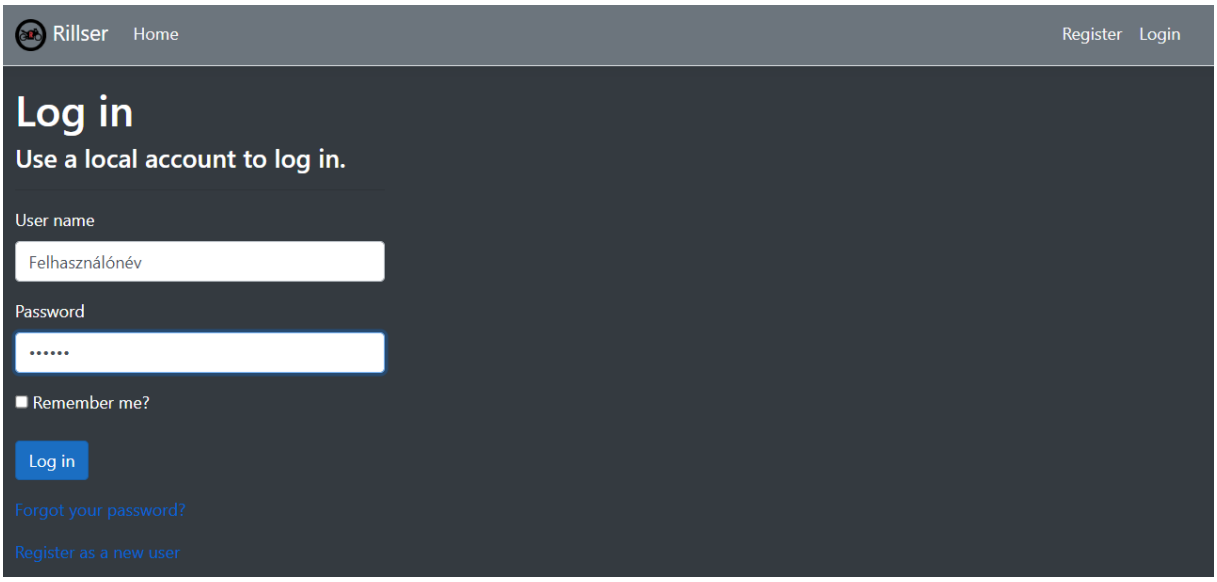
46. ábra Koordináta kontroller elérési útvonala és ApiController attribútuma

Az API-k funkcióinak helyes működésének tesztelése érdekében Postman segítségével küldtem kéréseket és a kapott üzenetek alapján tudtam kivizsgálni, hogy megfelelően funkcionálnak-e vagy sem. Amennyiben problémát tapasztaltam, A kapott válasz alapján be tudtam azonosítani azt, illetve a hibákat elhárítani.

A következő fázis a létrehozott kontrollerek alapján a felhasználói felületek megvalósítása volt.

Az alkalmazásban elsősorban szükség volt egy regisztrációs és bejelentkezés felületre. Ezekez úgynevezett „Scaffolded item” hozzáadásával készítettem el.

Ügyeltem arra, hogy a korábban megtervezett wireframe példájára épüljön a megjelenítés. Azonban a bejelentkezési felületben kisebb módosításokat végeztem, hiszem az ASP.Net Core Identity kezelője alapvetően e-mail címet vár el, én ezt módosítottam, hogy felhasználónévvel lehessen bejelentkezni.

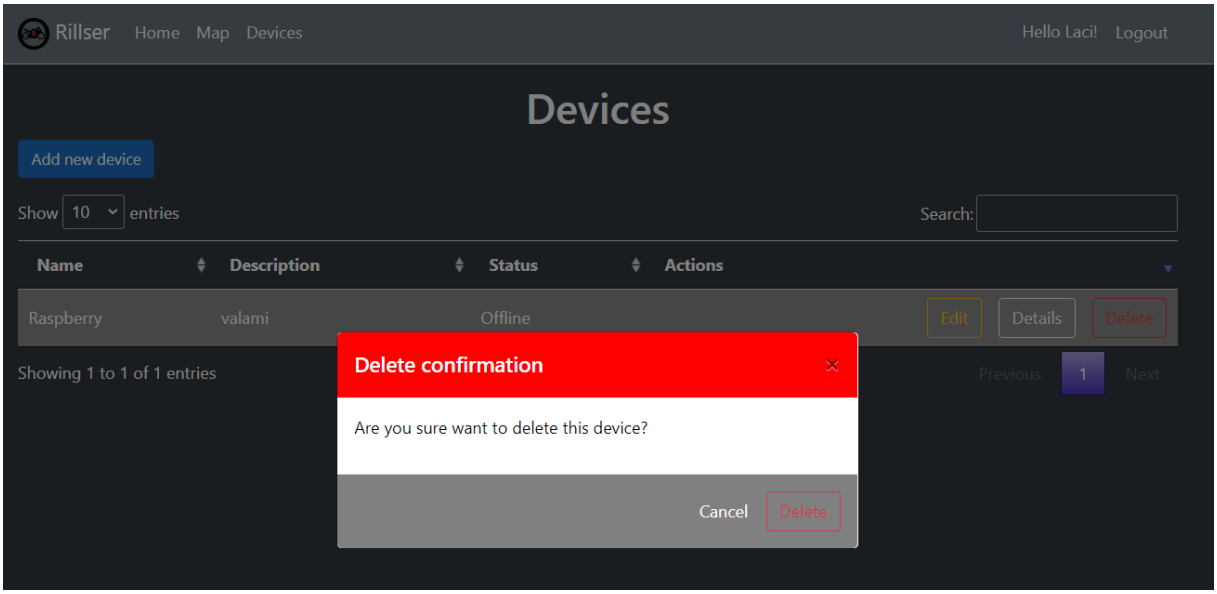


47. ábra Bejelentkezési felület a wireframe-nek megfelelően

A regisztrációs és bejelentkezési felület megléte után a következő cél a funkciók kibővítése volt. Elsősorban létrehoztam az interfészek és repositoryk alapján az eszközöknek és koordinátáknak egy kontrollert, melyek API-ként szolgálják ki a későbbiekben létrehozott webes felületet, illetve az elkészített eszközt.

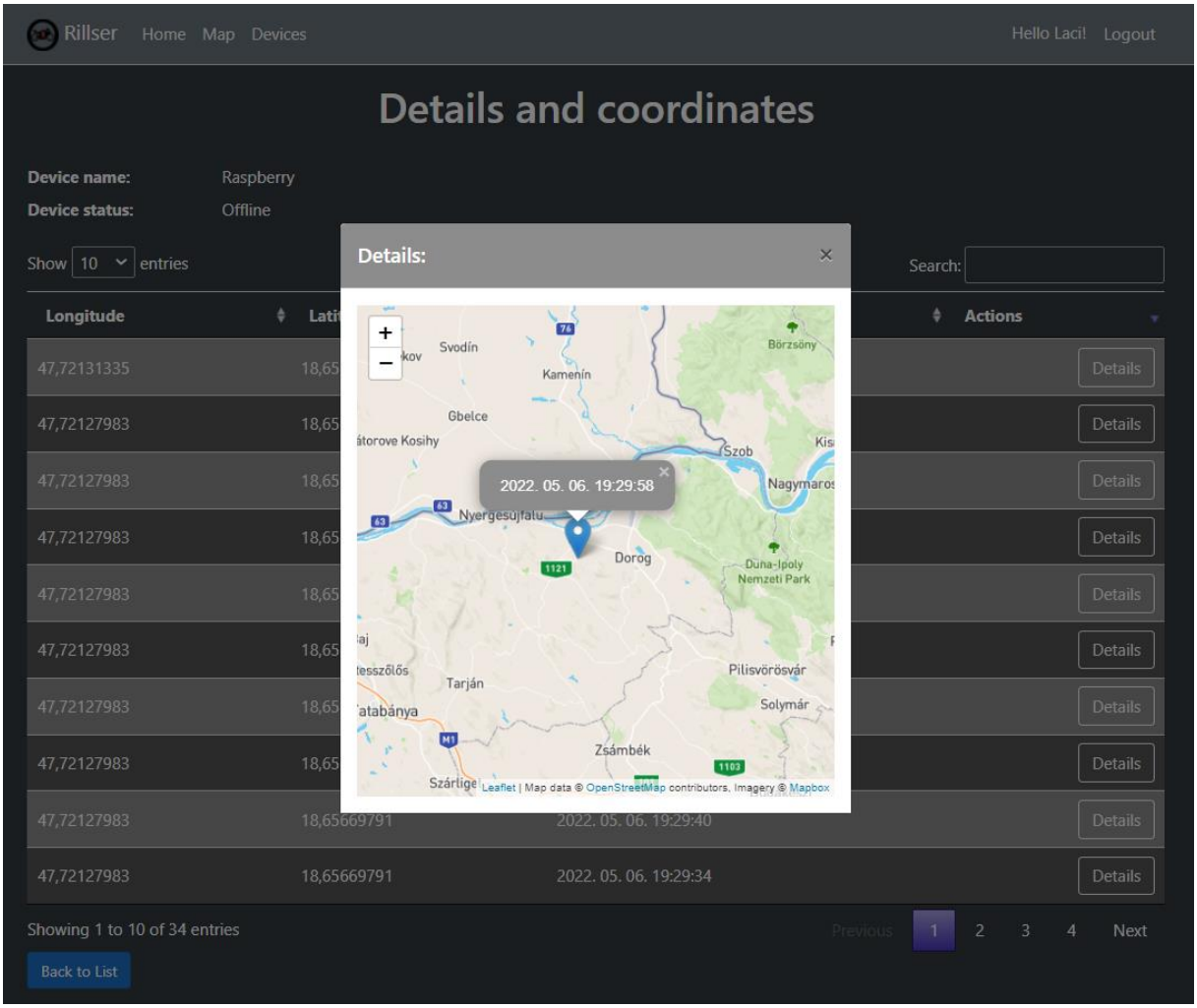
Elsősorban az alapvető CRUD műveletekre, kezdve a kilistázással.A listázás után célszerű volt sorban haladnom így elkészítettem az eszköz létrehozási felületét a korábban definiált wireframe alapján.Ezt követően az eszközök módosítására készitettem el a felhasználói felületet, szintén a korábban specifikált követelmények szerint.

Az eszköz törlésére is készítettem egy gombot, mely a listázási oldalon jelenik meg, azonban extra funkcióként egy Modal-al egészítettem ki, aminek célja a törlés megerősítése, ugyanis előfordulhat, hogy a felhasználó véletlenül félrekattint és törli az eszközt.



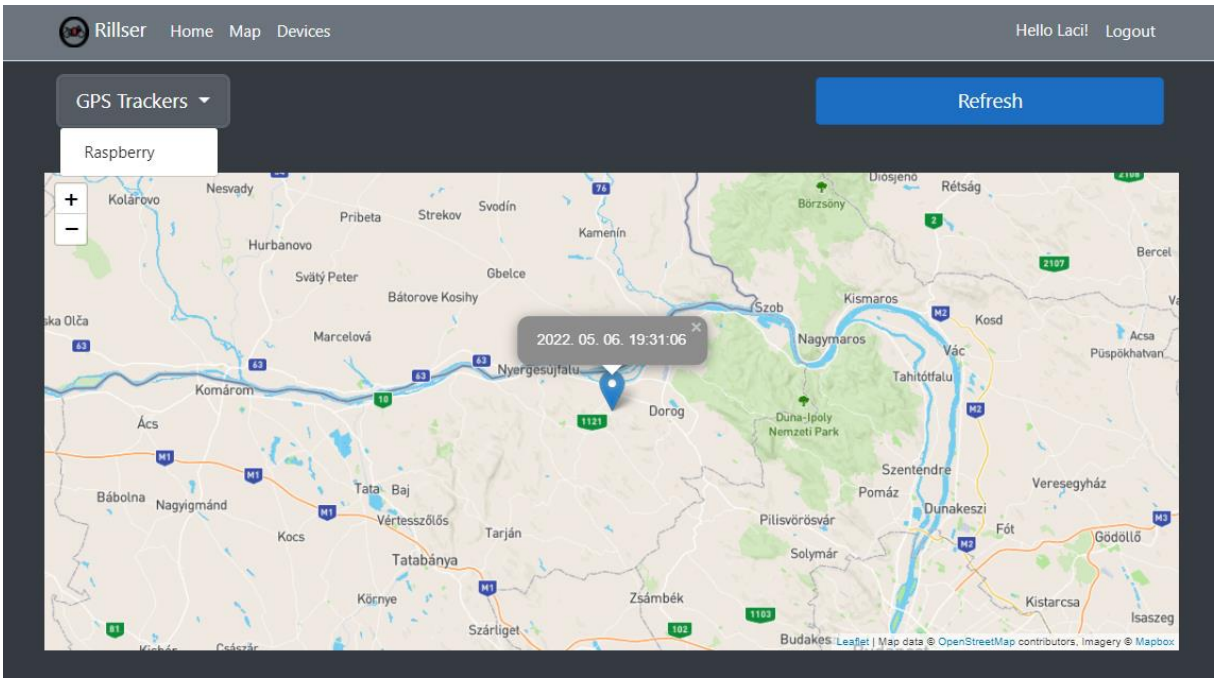
48. ábra Eszköz törlés megerősítése wireframe alapján

Már csak néhány rész maradt hátra a felhasználói felületekből, mint például az eszköz adatok és a hozzájuk tartozó koordináták listázása, illetve a wireframe-ben említett megjelenítése ezeknek a koordinátáknak. A térkép megjelenítéséhez szintén Modal-t használtam az eszköz információk és koordináták oldalon is, akár korábban a törlés gomb esetében.



49. ábra Kis térképes koordináta megjelenítés wireframe alapján

Ezután létrehoztam a fő megjelenítési oldalt, mely mindig minden eszköz legfrissebb (utolsó) koordinátáját jeleníti meg a térképen. Az oldal csak azokat az eszközöket listázza ki, amelyekhez tartozik már koordináta az adatbázisban. A frissítés gombra kattintva mindig az aktuális legfrissebb koordináta jelenik meg a térképen. Megjelenik egy összecsukható ablakban a térképen a jelölő felett egy szövegdoboz is, melyben megtekinthető a küldött koordináta dátuma is.



50. ábra Térkép oldal a wireframe alapján

Ahogy a tervezési fázisban említettem, amennyiben a felhasználónak nincs eszköze, vagy a rögzített eszköz még nem küldött koordinátát az applikáció részére, amely tárolva lenne az adatbázisban, a felhasználó a térkép felületén egy értesítés jelenik meg, hogy valamely feltétel a fentiek közül nem teljesül.

Utolsó sorban létrehoztam a felhasználó számára a profil kezelését, melyre az ASP.Net Core alapértelmezett felhasználó kezelését használtam, így csak néhány dolgot kellett módosítanom ezekben a funkciókban.

Ilyen módosítandó dolog volt például az e-mail küldő rendszer kiépítése. Alapértelmezetten az Identity biztosít e-mail küldő szolgáltatást, azonban konfigurálni kell azt, így saját szolgáltatást készítettem erre, melyet használtam későbbiekben a koordináta kontrollerben is felhasználó értesítésre, valamint a felhasználó kontrollerben is elfelejtett jelszó esetére.

Ennek az e-mail küldő szolgáltatásnak létrehoztam egy interfészt, majd ez alapján valósítottam meg a metódust.

```
public interface IEmailSender
{

    void SendEmailAsync(string email, string subject, string htmlMessage);
}
```

```
public class EmailSender : IEmailSender
```

51. ábra E-mail küldő szolgáltatás

Ahhoz, hogy az oldalon friss adatokat jeleníthessek meg a felhasználónak az eszközéről, illetve az eszközről mindig a legfrissebb adatok tudjam tárolni az adatbázisban, létre kellett hoznom egy extra „job”-ot, mely ellenőrzi, hogy az eszköz elérhető-e. Ezt az IHostedService interfész alapján képeztem le, ugyanis ez egy háttérben futó folyamat kell legyen.

```
public class StatusService : IHostedService
```

52. ábra Státusz ellenőrző szolgáltatás

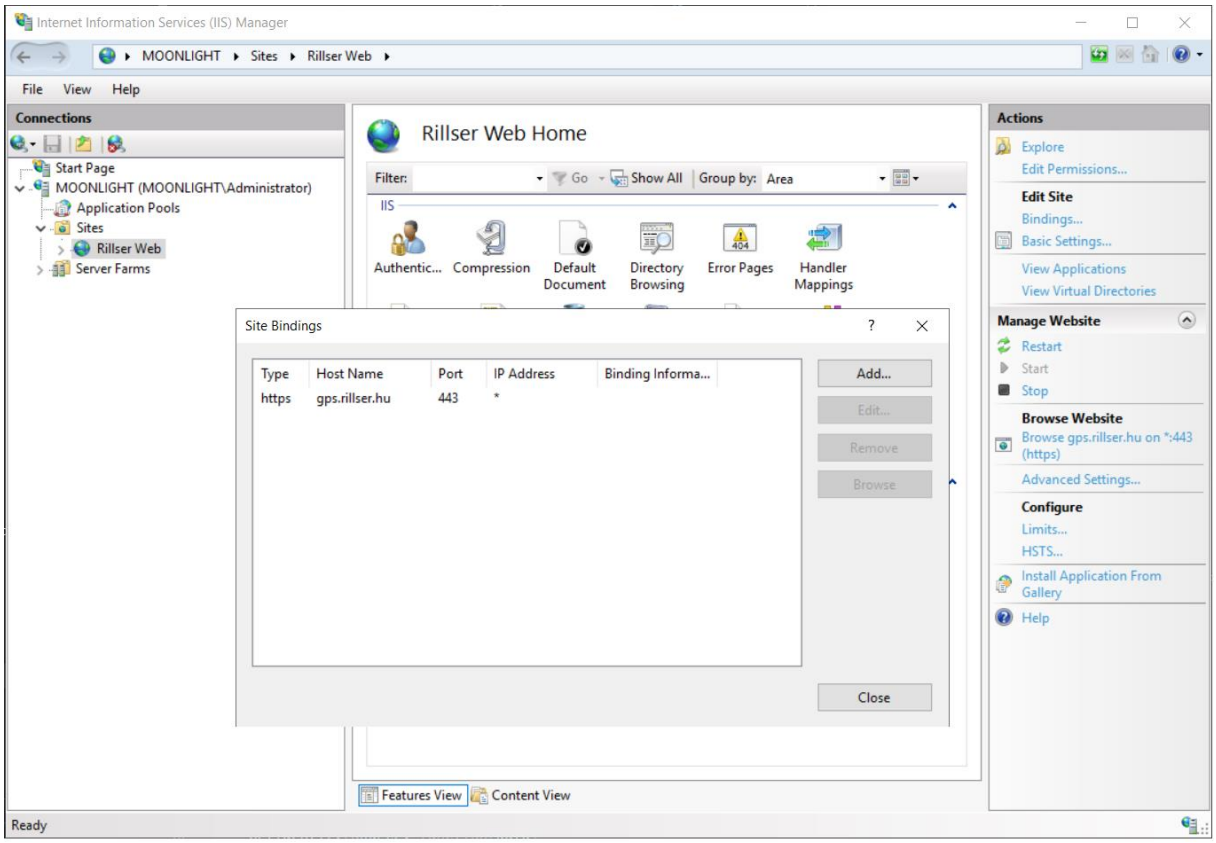
Amint sikerült elérni, hogy minden felhasználói felület késznek tekinthető legyen, továbbá minden elvárt funkciót megvalósítson az applikáció, ellenőriznem kellett, hogy mi történik azon esetben, ha az applikációt publikálom és az eszköz veszi át a kommunikációt a Postman tesztkörnyezet helyett.

A publikálás során használatba vettem a saját tulajdonomban álló domain nevet, melyre létrehoztam egy aldomaint. A domain-t bevezettem a Cloudflare rendszerébe, mely extra védelmet biztosít bizonyos túlterheléses támadások és adathalász programok ellen, méghozzá azzal a módszerrel, hogy az általam beállított IP címet elfedi egy Cloudflare-es IP címmel, így nem közvetlenül az általam beállított IP címre érkeznek a kérések, hanem erre a Proxy IP-re, amiről továbbítódnak a szűrt kérések.

53. ábra Cloudlfare domain beállítás

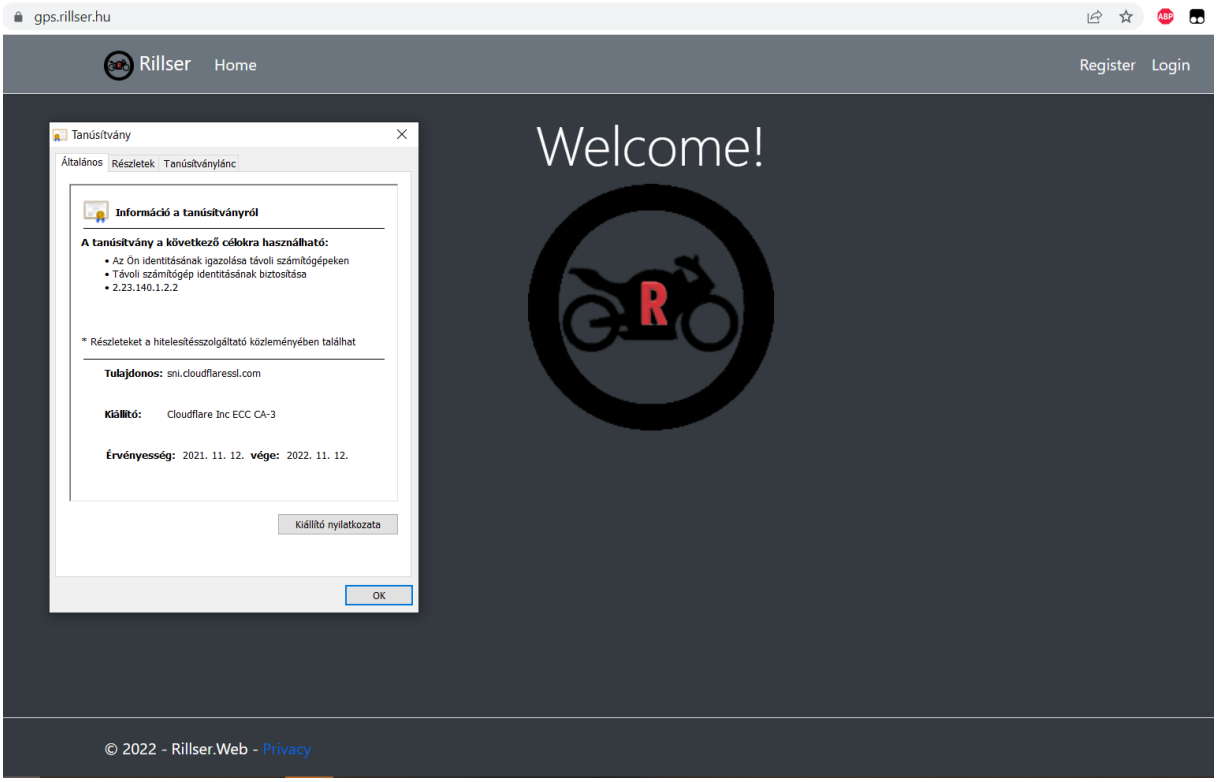
Ezen a felületen állítottam be az Azure VPS-em IP címét, amire a publikációt elvégeztem.

A VPS egy Windows Server 2019-es operációs rendszerrel ellátott virtuális privát szerver, ezen a szerveren konfiguráltam be az IIS-t, hoztam létre a weboldalt és beállítottam a kötéseket megfelelően, hogy az oldal biztonságos HTTPS protokollon keresztül legyen elérhető.



54. ábra IIS weboldal a hozzá beállított bindig-al

Telepítettem egy SSL tanúsítványt, hogy a weboldal ne önaláírt tanúsítvánnyal üzemeljen, hanem biztonságosan jelenjen meg minden böngészőben. A Cloudflare biztosít a domain és a szerver oldalra is tanúsítványt, így csak generálnom kellett egyet és feltelepíteni azt a Windows Server IIS szolgáltatásában. Ezt követően a weboldal biztonsági figyelmeztetés nélkül elérhetővé vált és a tanúsítványa is megtekinthető a böngészőben.



55. ábra Publikált weboldal megjelenése, elérési címe, érvényes tanúsítványa

Ezt követően az ezközön beállítottam, hogy az API hívásokat a megfelelő elérési útvonalon tegye, mivel a publikálás előtt localhost-on teszteltem.

12. TESZTELÉSI JEGYZŐKÖNYV

A tervezési és fejlesztési fázis során rengeteg tesztesetet sikerült összegyűjtenem, bár megemlíteném, hogy a fejlesztés során és után számtalanszor kerültek a funkciók tesztelésre, ettől függetlenül a kész alkalmazás és eszköz tesztelése elengedhetetlen. A tesztelés során minden tesztesetet ellátok egy azonosítóval, egy részletezéssel, egy elvárt és egy produkált eredménnyel, illetve egy kiértékeléssel.

A teszteseteket az alábbi táblázatban fejtem ki:

Azonosító	Részletezés	Elvárt eredmény	Produkált eredmény	Kiértékelés
REGISTER_WEAK_PASSWORD	Gyenge jelszóval történi regisztráció.	Sikertelen	Sikertelen	A rendszer figyelmeztette a felhasználót, hogy a jelszó nem megfelelő a követelményeknek.
REGISTER_EXISTING_EMAIL	Már létező e-mail címmel történő regisztráció	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy ezzel az e-mailcímmel már regisztráltak.
REGISTER_MISMATCH_PASSWORD	A regisztációnál megadott jelszavak nem egyeznek	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy a megadott jelszavak nem egyeznek
LOGIN_WRONG_PASSWORD	A bejelentkezéskor megadott jelszó helytelen	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy a megadott adatok közül valamely helytelen
LOGIN_WRONG_USERNAME	A bejelentkezéskor megadott felhasználónév helytelen	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy a megadott adatok közül valamely helytelen
MAP_WITHOUT_AUTHENTICATION	Térkép oldal megnyitása bejelentkezés nélkül	Sikertelen	Sikertelen	Az url megadása során a felhasználót a bejelentkezési oldalra irányítja a rendszer.
DEVICE_WITHOUT_AUTHENTICATION	Eszköz oldal megnyitása bejelentkezés nélkül	Sikertelen	Sikertelen	Az url megadása során a felhasználót a bejelentkezési oldalra irányítja a rendszer.
DEVINFO_COORDINATE_WITHOUT_AUT	Koordináta és eszközadat oldal megnyitása bejelentkezés nélkül	Sikertelen	Sikertelen	Az url megadása során a felhasználót a bejelentkezési oldalra irányítja a rendszer.

MAP_AUTHENTICATED	Térkép oldal megnyitása bejelentkezést követően	Sikeres	Sikeres	A kiválasztott oldal megjelenik a felhasználónak.
DEVICE_AUTHENTICATED	Térkép oldal megnyitása bejelentkezést követően	Sikeres	Sikeres	A kiválasztott oldal megjelenik a felhasználónak.
DEVINFO_COORDINATE_AUTHENTICATED	Koordináta és eszközadat oldal megnyitása bejelentkezést követően	Sikeres	Sikeres	A kiválasztott oldal megjelenik a felhasználónak.
ADD_DEVICE_WRONG_UID	Eszköz hozzáadása rossz egyedi azonosítóval	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy nem található az eszköz azonosító az adatbázisban.
ADD_DEVICE_WRONG_UID_LENGTH	Nem megfelelő hosszúságú egyedi azonosítóval eszköz hozzáadás	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy nem megfelelő hosszúságú az eszköz
ADD_DEVICE_WTHOUT_NAME	Eszköz hozzáadása név nélkül	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy a név mező kitöltése kötelező
DEVICE_CREATION	Eszköz megfelelő adatokkal történő felvétele	Sikeres	Sikeres	A rendszer eltárolja az adatbázisban az eszközt a megfelelő felhasználóhoz társítva Megjelenik az eszközök oldalon
UPDATE_DEVICE	Eszköz nevének, leírásának, vagy státuszának módosítása.	Sikeres	Sikeres	A módosított adatokat a rendszer elmenti az adatbázisban, illetve befrissíti az eszközök felületen
DELETE_DEVICE	Eszköz törlése	Sikeres	Sikeres	A felugró ablakon megerősítve a törlést az eszköz törlődik az adatbázisból, illetve a felületről is
CANCEL_DELETE_DEVICE	Eszköz törlésének megszakítása	Sikeres	Sikeres	Az eszköz törlése nem megy végbe, benne marad az adatbázisban és a felületen is megjelenik
SHOW_MINIMAP	Koordináták oldalon részletek gombra kattintva kis térképen megjelenik a koordináta	Sikeres	Sikeres	A kis térkép megjelent, a koordináta, mint ahogy a Térkép oldalon, itt is megtekinthető

USED_USERNAME_CHANGE	Már létező felhasználónévre történő felhasználónév módosítás	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy hiba történt a felhasználónév módosítás során
CHANGE_EMAIL_INVALID_FORMAT	E-mailcím módosítása során nem megfelelő formátum megadása	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy az e-mailcím melyet megadott nem megfelelő formátumú
CHANGE_PASSWORD_INCORRECT_PASSWORD	Jelszó változtatás nem megfelelő jelenlegi jelszó megadásával	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy helytelen jelszót adott meg
CHANGE_PASSWORD_MISMATCH	Jelszó változtatás esetén az új jelszavak nem egyeznek	Sikertelen	Sikertelen	A rendszer figyelmezteti a felhasználót, hogy a megadott jelszavak nem egyeznek
USERNAME_CHANGE_VALID	Nem foglalt felhasználónév megadásával felhasználónév módosítás	Sikeres	Sikeres	Az új felhasználónév mentésre kerül az adatbázisban, illetve a weboldalon is megjelenik
CHANGE_EMAIL	Megfelelő formátumú e-mailcím megadása	Sikeres	Sikeres	A rendszer e-mailt küld a megadott címre
VERIFY_EMAIL	A kiküldött e-mailben szereplő hivatkozásra kattint a felhasználó	Sikeres	Sikeres	A módosított e-mail cím mentésre kerül az adatbázisban és a felületen is
CHANGE_PASSWORD_SUCCESS	Megfelelő jelszavakkal jelszó módosítás	Sikeres	Sikeres	A jelszó módosításra kerül, hashelt formában mentésre kerül az adatbázisban, továbbá a bejelentkezést követően már az új jelszó lesz érvényben
DEVICE_SEND_COORDINATE_FAIL	Internetkapcsolat hiányában az eszköz nem tudja elküldeni a koordinátát, azonban eltárolja lokálisan	Sikeres	Sikeres	A koordináta a hozzá tartozó idővel eltárolásra kerül
DEVICE_RESEND_COORDINATES	Az eltárolt koordinátákat ismételt internet kapcsolat mellett továbbítja a szervernek	Sikeres	Sikeres	A szerver fogadja a koordinátákat, eltárolja az adatbázisban a megfelelő eszközhöz rendelve, melyeket megtekinthet a felhasználó a felületen
ALERT_USER	Az eszközt 50 méter távolságon túlra mozdították, értesítést küld a felhasználónak	Sikeres	Sikeres	A rendszer megkapta az értesítési kérést, kiküldte az e-mailt a felhasználónak és módosította az eszköz státuszát aktívra

STATUS_CHANGED_BY_USER	A felhasználó az eszköz státuszát módosítja, hogy gyakrabban kapjon koordinátát	Sikeres	Sikeres	Az adatbázisban az eszköz státusza frissült, az eszköz a következő státusz lekérésnél frissíti a küldési mechanizmust
DEVICE_OFFLINE_STATUS_CHANGE	A felhasználó egy offline eszközt szeretne módosítani	Sikertelen	Sikertelen	Az eszköz legördülő menüje szürke ablakban jelenik meg és nem lehet másik státuszt kiválasztani.
DEVICE_GET_CONFIG_WITHOUT_CONNECTION	Az eszköz érdeklődik státuszra iránt, de nincs internet kapcsolata	Sikertelen	Sikertelen	Az eszköz nem tudja lekérni a státuszát, azonban a lokálisan eltárolt státusz alapján tovább dolgozik
DEVICE_GET_CONFIG_CONNECTED	Az eszköz érdeklődik a státusza iránt, internetkapcsolat rendelkezésre áll	Sikeres	Sikeres	Az eszköz a megadott státusz alapján dolgozik

8. táblázat Tesztforgatókönyv

13. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A koordináták hatékonyabb megjelenítése érdekében a jelenlegi rendszerbe beépítésre kerülhetne a SignalR, mellyel a két irányú kommunikációt lehetne megvalósítani, ezzel is folyamatos és azonnali megjelenítést biztosítva az új koordinátáknak, illetve ezzel együtt egy útvonaljelölést is meg lehetne valósítani.

A jelenlegi elmozdítás érzékelésen javítana egy giroszkópos eszköz, ugyanis akkor nem feltétlenül kellene levizsgálni azt, hogy két koordináta között mekkora különbség van, hiszen mozgatás észlelésekor azonnal tudna jelezni. Létezik már a Raspberry Pi Zero W-vel kompatibilis giroszkóp, ellenben ilyen kiegészítő esetében javasolt lenne a jelenlegi eszközt lecserélni, ugyanis már most is több komponensből áll.

Célszerű lenne egy olyan eszköz használata, mely még kevesebb energiaigénnyel rendelkezik, képes hibernált állapotban is működni, illetve pontosabb adatokat képes a serial portról olvasni. Emellett a korábban említett több modulból összeépítést is el lehetne kerülni, ha olyan eszközt használnék a későbbiekben, ami ezeket a modulokat egy lapkán tartalmazza.

14.ÖSSZEFOGLALÁS, KÖVETKEZTETÉS

Tekintettel a probléma forrására, illetve a létező megoldások hiányosságára, arra a következtetésre jutottam, hogy a termék betöltene egy piaci rést. A vizsgált cégek termékei fuvarozószervekre, illetve egyéb vállalkozásokra összpontosít, ezzel szemben ez a termék a magánszemélyeket célozná meg. Az általam tervezett projekt nem flotta szolgáltatást nyújt, hanem egy viszonylag olcsó, könnyen elérhető terméket átlagemberek számára. A projekt során egy olyan eszközt szeretnék tervezni, amely nem csak a jármű üzemi állapotában küldi számunkra a GPS koordinátákat, hanem parkolás esetén is értesít róla, hogy megmozdítják a gépjárművet. Fontos szempont volt, hogy a platform függetlenségre törekedjek, emellett az összes gyakori böngészőre optimális megjelenést biztosítsak. Az eszköz a lefedetlen területeken is gyűjtené a koordinátákat, amelyeket kapcsolat helyreállásának alkalmával továbbít a szerver felé. A fizikai eszközként a Rasberry Pi-t választottam, mert ez egy számítógép operációs rendszerrel, ezért könnyebb rajta a fejlesztés, mint egy mikrokontrolleren. A térkép API esetén az OpenStreetMap API-ját választottam, mert az API hívások számának tekintetében számomra kedvezőbb, mint a korábban említett opciók. Az adattárolás és az alkalmazás futtatására az Azure-t választottam kedvező árazása miatt. A szerver oldali applikáció keretrendszerének az ASP.NET Core-t választottam, mivel ennek segítségével tudok C#-ban fejleszteni. Mint korábban említettem, a klienshez kerülő fizikai eszközön futtatott kódot pedig Pythonban terveztem fejleszteni, ezzel is optimális, kevés erőforrás igénnyel rendelkező kódot biztosítva a fizikai eszköznek. A felhasználók által használt felületet pedig szintén ASP.NET Core MVC 5 applikációként valósítottam meg.

Az adatbázis, az applikáció, az eszköz és a kliens közötti kommunikáció érzékeny információkat tartalmazhat, így ezek titkosításáról nem feledkeztem meg, az alkalmazás HTTPS protokollt használ a kommunikációra a kliens és a szerver között, továbbá SSL tanúsítvány is telepítésre került a szerverre. A fejlesztés során a Python tudásom hiányosságából adódó nehézségeket sikerült leküzdeni, melynek következtében egy olyan eszközt és alkalmazást tudtam készíteni, ami kielégíti az átlagos felhasználók igényét, akik biztonságban szeretnék tudni gépjárművüket. Az elkészült eszköz és alkalmazás funkciói számtalan teszteseten estek át, mindegyik funkció megfelelt a elvárt eredményeknek, így kijelenthetem, hogy egy fogyasztói kör kielégítésére is képes a projekt.

15.SUMMERY, CONCLUSION

Given the source of the problem and the shortcomings of the existing solutions, I came to the conclusion that the product would fulfill a market gap. The products of the surveyed companies focus on other companies and businesses, instead of my plan. This product would be targeted at private individuals. The project I am planning is not a fleet service, but a relatively inexpensive, easily accessible product for the average people. During the project, I want to design a device that not only sends us the GPS coordinates while the vehicle is in operation, but also notifies us when the vehicle is being parked and someone move it. It was important for me to make a platform independent application, as well as ensure an optimal look for all common browsers. The device would also collect the coordinates in the uncovered areas, which it transmits to the server when the connection is restored. I chose Rasberry Pi as physical device because it has a computer operating system, so it's easier to develop on it than a microcontroller. For the map API, I chose the OpenStreetMap API because the API is more favorable to me than the options mentioned earlier in terms of the number of API calls. I chose Azure to store the data storage and application because of its favorable pricing. I chose ASP.NET Core as the framework for the server-side application because it allows me to develop in C#. As I mentioned earlier, I planned to develop the code running on the physical device in Python, this providing optimal, low-resource code. The interface which is used by the users was also implemented as an ASP.NET Core MVC 5 application.

The communication between the database, the application, the device and the client may contain sensitive information, so I did not forget to encrypt them, the application uses HTTPS to communicate between the client and the server, and an SSL certificate is installed on the server. During the development, I was able to overcome the difficulties caused by my lack of knowledge of Python, which allowed me to create a tool and application that meets the requirements of the average user who wants to know their vehicle is safe. The functions of the completed device and application have undergone numerous test cases, each function met the expected results, so I can say that the project is able to satisfy a consumer group.

16.IRODALOM JEGYZÉK

[1.] A GPS Commander termék hivatalos oldala(Node.js w/1M concurrent connections! – caustik's blog)

[2.] A PEPXIM termék hivatalos oldala(https://www.pepxim.com/)

[3.] A kiadó (NVIDIA) hivatalos leírása a Vulkan-ról (https://developer.nvidia.com/Vulkan)

[4.] Raspberry Pi blog oldalán található dokumentáció(https://www.raspberrypi.org/blog/vulkan-update-now-with-added-source-code/)

[5.] Optiboot GitHub dokumentáció(https://github.com/Optiboot/optiboot)

[6.] Pant, Sangita. "Comparison of Google Map Features with Bing Map."

[7.] A Google hivatalos oldalán található forrás az árváltozásról(https://cloud.google.com/maps-platform/user-guide/pricing-changes)

[8.] A Google hivatalos oldalán található árazás(https://cloud.google.com/maps-platform/pricing)

[9.] Tanács jellegű cikk a Codematters oldaláról(https://codematters.online/how-bing-maps-compares-to-the-new-google-maps-pricing/?fbclid=IwAR3h0ewS5O7Hecb4YCvKMyvkh7v2PpR71xu1NqxUw4LJgXLfMK-TfSFFA3Y)

[10.] A Linux Containers hivatalos oldalán található információ (https://linuxcontainers.org/)

[11.] Madhuri, T., and P. Sowjanya. "Microsoft Azure v/s Amazon AWS cloud services: a comparative study." International Journal of Innovative Research in Science, Engineering and Technology (2016)

[12.] Hivatalos dokumentáció a Microsoft részéről az Azure Blob Storage hozzáférési szintjeiről (https://docs.microsoft.com/hu-hu/azure/storage/blobs/storage-blob-storage-tiers?tabs=azure-portal)

[13.] A CloudNet App összehasonlító cikke az Azure és AWS árazásról(https://cloud.netapp.com/blog/azure-vs-aws-pricing-comparing-apples-to-apples-azure-aws-cvo-blg)

[14.] A Microsoft hivatalos oldalán található árazással kapcsolatos információk diákok részére (https://azure.microsoft.com/en-us/offers/ms-azr-0170p/)

[15.] Prettyman, Steve. "The Basics: PHP 8 Syntax." *Learn PHP 8*. Apress, Berkeley, CA, 2020.

[16.] Koirala, Dhiraj. "Web application for professional union using React and Node. js." (2020).

[17.] A Microsoft hivatalos leírása a .NET-ről (https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet)

[18.] A Microsoft hivatalos dokumentációja a .NET5-ről (https://docs.microsoft.com/en-us/dotnet/core/dotnet-five)

[19.] Bartlett, Jonathan. "Building Projects with Arduino." *Electronics for Beginners*. Apress, Berkeley, CA, 2020.

[20.] Bijoura, Ayush, and K. Murugan. "Comparative Analysis of Web Frameworks (Angular JS, React JS, Vue JS)." *Solid State Technology* (2021)

[21.] Rasberry Pi hivatalos dokumentációja a gyártó oldaláról (https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md)

[22.] Hivatalos Microsoft dokumentációk C/C++ programozáshoz kapcsolódóan (https://docs.microsoft.com/en-us/cpp/?view=msvc-160)

[23.] Lott, Steven. „*Functional python programming*.” Packt Publishing Ltd, 2015.

[24.] Trupin, Joshua. "Sharp new language: C# offers the power of C++ and simplicity of Visual Basic." *MSDN Magazine* (2002). – Microsoft Documentáció (https://docs.microsoft.com/en-us/archive/msdn-magazine/2000/september/sharp-new-language-csharp-offers-the-power-of-c-and-simplicity-of-visual-basic)

[25.] PHP hivatalos oldala (https://www.php.net/manual/en/langref.php)

[26.] Az ASP.NET Core 5 bemutató dokumentációja a Microsoft-tól(https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0)

[27.] A Node.JS bemutató dokumentációja a hivatalos oldalról (https://nodejs.dev/learn)

[28.] Angular szójegyzéke a hivatalos oldalról (https://angular.io/guide/glossary)

[29.] React szójegyzéke a hivatalos oldalról(https://reactjs.org/docs/glossary.html)

[30.] J. Whitington Python from the Very Beginning 2020

17.TÁBLÁZAT JEGYZÉK

18.ÁBRA JEGYZÉK