Debreceni Egyetem

Informatikai Kar

# UniLocker

**Témavezető:** Dr. Kocsis Gergely

Egyetemi docens

**Készítette:** Tarr Imre

Mérnökinformatikus (BSc)

Debrecen, 2023

# Köszönetnyilvánítás

Először is szeretném kifejezni köszönetemet Dr. Kocsis Gergelynek, a témavezetőmnek és tanáromnak, azáltal, hogy támogatott, irányított és rengeteg értékes ötlettel látott el a szakdolgozatom írása során. Az ő tapasztalata, lelkiismeretes munkája, előzékenysége és pozitív hozzáállása nagyban hozzájárult a dolgozatom minőségéhez és sikeréhez.

Továbbá köszönettel tartozom minden olyan személynek, aki segített nekem megvalósítani a projektet és ezáltal hozzájárult a szakdolgozatom sikeréhez.

Köszönettel,
Tarr Imre

# Tartalomjegyzék

# 1. Bevezető

Egy olyan projektet szerettem volna létrehozni, ami bárkinek, illetve bárhonnan, bármikor elérhető és az embereknek/hallgatóknak is segítek vele.

A diplomamunka kitalálásakor elég sokat adtam és vettem át csomagokat, így jött az első ötlet, ami egy számítógépes alkalmazás volt, amely segítségével az egyetem területén belül lehetett volna nyomon követni és kezelni csomagokat. Ezt az alkalmazást "UniPost" néven képzeltem el, amivel megkönnyíthetem a hallgatók életét és ezzel együtt lehetne alkotni egy egyetemi postai szolgáltatást. Második ötletként, magát a szekrényt képzeltem el, viszont akkor még nem tudtam, hogy ezt képes vagyok-e megvalósítani. Dr. Kocsis Gergellyel beszélgetve, említette, hogy az Informatika Karon találhatóak szekrények, amiket sajnos a hallgatók nem tudnak használni, mert nem megfelelő rajta a zár és nem biztonságos. Így jött az ötlet, hogy ötvözzem a két gondolatot és megszületett az okos szekrény - UniLocker.

Az alkalmazás szoftver részének tervezésekor egy mobil alkalmazást szerettem volna. Azért választottam ezt a megközelítést, mert a hallgatók mindig maguknál hordják a telefonjukat, így könnyen elérhetővé tehettem számukra a szolgáltatást. Az eredeti elképzelésem egy Java vagy Kotlin alapú alkalmazás létrehozása volt, viszont mivel nem vagyok annyira tapasztalt ebben a két nyelvben, inkább úgy döntöttem, hogy egy PWA-t (Progresszív Webalkalmazás) készítek, mert ebben már volt némi tapasztalatom. Ennek számos előnye van, például: az, hogy szinte az összes modern telefonon fut, nem igényel külön fejlesztést különböző márkákhoz, és könnyen hozzáférhető a böngészőn keresztül.

Az UniLocker projekt egy olyan progresszív webalkalmazás, amelynek célja, hogy az oktatási intézményekben elérhető szekrények kezelését és vezérlését egyszerűsítse és biztonságossá tegye. A projekt egyesíti a modern webfejlesztési technológiák és az IoT (Internet of Things) eszközök világát.

Központi célkitűzésem volt, hogy egy olyan platformot hozzak létre, amely lehetővé teszi a hallgatók számára, hogy könnyedén és biztonságosan használják az egyetemi szekrényeket. Az alkalmazás lehetővé teszi a felhasználók számára a regisztrációt és az azonosítást, ideértve az UniPass kártyák használatát is, amelyek segítségével egy egyszerű érintéssel nyithatják ki a szekrényeiket. A kártyák beolvasása és az azonosítás szerves részét képezik az alkalmazásnak, biztosítva ezzel a biztonságos hozzáférést a szekrényekhez.

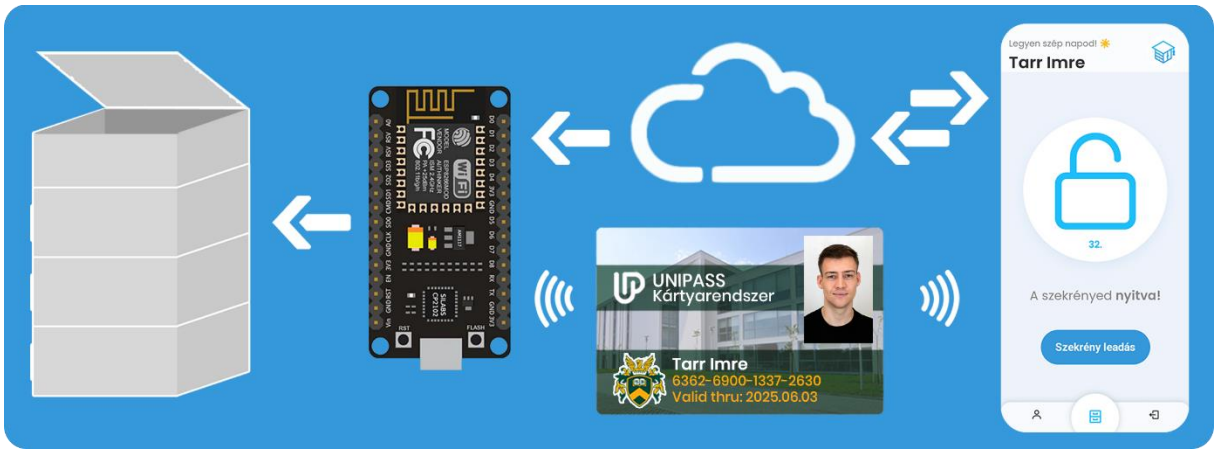
Az alkalmazás két fő része: az első rész a webes felület, mely lehetővé teszi a felhasználók számára a szekrények kiválasztását és kezelését, valamint az azonosítást és regisztrációt.

A második rész pedig az eszközökkel történő kommunikációt szolgálja, amelyek az egyetemi szekrényeket irányítják. Az alkalmazás lényeges eleme a biztonság, amely magába foglalja az adatok, tárgyak - tulajdonok védelmét és az azonosítás megbízhatóságát. Csak azok a felhasználók férnek hozzá az adatokhoz és jogosultságokhoz, akik erre valóban jogosultak.

Az UniLocker előnyöket rejt a hallgatók és az oktatási intézmények számára is. Az alkalmazás lehetővé teszi, hogy a hallgatók autonóm módon használják a szekrényeket, nem szükséges személyzet közreműködése a hozzáféréshez. Ezáltal az oktatási intézmények hatékonyabban használhatják az emberi erőforrásaikat, mivel a dolgozók más fontos feladatokra koncentrálhatnak. Emellett a szekrények hatékonyabb kihasználása növelheti a hallgatók elégedettségét és kényelmét, ami hozzájárul az oktatási intézmény jó hírnevének és vonzerejének növeléséhez.

A projekt ütemterve magába foglalja a tervezést, fejlesztést, adatbázis kialakítást, biztonság biztosítását, tesztelést és dokumentálást. A projekt megvalósításához szükséges eszközök beszerzése és a weboldal létrehozása is része a tervezett munkafolyamatoknak. Az alkalmazás minden fázisában kiemelt figyelmet fordítok mind a felhasználói felület (UI), mind pedig a felhasználói élmény (UX) tervezésére és fejlesztésére, hogy a végeredmény egy kiváló minőségű és maximálisan használható alkalmazás legyen. Az UI és UX tervezési szempontokat mindvégig az elsődleges szempontok között kezelem.

Az elkövetkezendő oldalakon részletesen bemutatom a projekt tervezését és megvalósítását, hogy átfogó képet mutassak az alkalmazás fejlesztéséről és annak hozzáadott értékéről.




*Szoftver és hardver kapcsolata*

# 2. Hardware

A feladat elvégzéséhez olyan hardvert kerestem, amely alkalmas volt a számomra ismert vagy könnyen elsajátítható magasszintű programozási nyelveken való fejlesztésre. Fontos szempont volt továbbá, hogy rendelkezzen elegendő számítási kapacitással és megfelelő számú IO porttal, hogy a későbbiekben bővíthessem az általam vezérelt berendezéseket.

Az első gondolatom az Arduino Uno használata volt a projekthez, egy ESP-01 modullal, hiszen már rendelkeztem némi tapasztalattal ezen a területen. Azonban az internethez való csatlakozás problémát jelentett ezen a platformon, és mivel az internetkapcsolat elengedhetetlen volt a tervezett alkalmazás működéséhez, másik mikroprocesszor után kellett néznem.

Ekkor találtam rá az ESP8266 NodeMCU V2-re (ESP-12E) modulra, amely számos előnnyel rendelkezett. Ez a mikroprocesszor lehetővé tette számomra, hogy könnyedén csatlakozzak az internethez, és a magas számítási kapacitása révén alkalmas volt az általam tervezett alkalmazás futtatására. Emellett a modul megfelelő számú IO porttal rendelkezett, amely lehetővé tette a különböző eszközök vezérlését és bővítését.

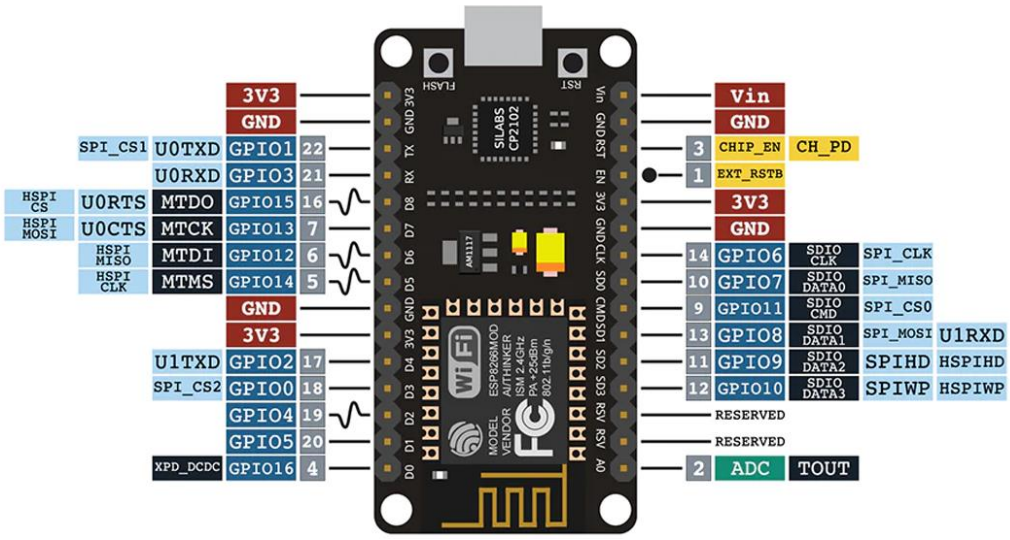
Kezdetben beszereztem pár eszközt, kezdve az ESP-vel, a mini 5 voltos elektromos zárakkal, egy 4 csatornás relé modullal, illetve vezetékekkel.

Az eredeti tervem az volt, hogy egy vezeték nélküli megoldással valósítom meg a projektet. Ehhez először az ESP működéséhez egy powerbankot, a zárakhoz egy 9 voltos elemet használtam, majd egy LM7805 feszültségszabályzóval próbáltam csökkenteni a feszültséget 5 voltos szintre. Ezzel a megoldással azonban akadtak gondok. Először is, a 9 voltos elem kapacitása kevésnek bizonyult ahhoz, hogy hosszú távon működőképes legyen a rendszer. Emellett az LM7805 nem volt képes elegendő áramot biztosítani a megfelelő működéshez, ezért kénytelen voltam egy másik megoldást keresni.

Ebben a fázisban szereztem be egy külső adaptert, amely stabil 5 voltos feszültséget és 6 ampert volt képes szolgáltatni a rendszer számára. Ez megszüntette az elemekkel és a feszültségszabályozóval kapcsolatos problémákat, és biztosította a megfelelő működést.

Továbbá beszereztem az összes létező variációját az úgynevezett jumper vezetékeknek, apa-anya, apa-apa, anya-anya. Majd egy Mifare RFID író/olvasó (MFRC522; 13.56MHz) modult. 2 darab kapcsolót, mely segítségével külön kapcsolom fel az ESP-t, illetve a zárakat.

## 2.1. ESP8266 NodeMCU V2




*1. ábra: ESP8266 NodeMCU V2 (ESP-12E)*[1]

Az ESP8266 NodeMCU V2 (*1. ábra*) egy rendkívül hasznos mikroprocesszor, jelentősen megkönnyíti a WiFi kapcsolatok létesítését és kezelését. Ez az eszköz különösen hasznos azok számára, akik IoT (Az Internet of Things) projektjeiken dolgoznak, mivel lehetővé teszi a vezeték nélküli kapcsolatot a különböző eszközök és számítógépek között. Az ESP8266 NodeMCU V2 egy kifejezetten népszerű platform az elektronika és programozás világában.

A PC oldal felé nem igényel semmiféle plusz kiegészítést, az ESP8266 chip soros kommunikációját a CP2102 soros-USB illesztőchip valósítja meg. A könnyű kezelhetőséget alapesetben az egyszerű AT parancsok biztosítják.

Ráadásul a saját 49 milliméter szélességével, 24 milliméter magasságával és 13 milliméter mélységével rendkívül kompakt méreteinek köszönhetően könnyedén elhelyezhető és elférhet bárhol.

Főbb paraméterei[2]:

- Tápfeszültség: A modul 3.3V tápfeszültséget igényel az ESP8266 chip működéséhez, míg a USB portról (5V) kapja az energiát. Ezt a chip soros kommunikációjára is használja, ami szintén rendkívül kényelmes.
- I/O felület: A modulnak 3.3V LVTTL szintű I/O portjai vannak, amelyek lehetővé teszik a digitális jelek bemenetét és kimenetét. Ezek a portok nagyon fontosak az eszközök közötti kommunikációhoz és az érzékelőrendszerekhez.

[1] Kép forrása: https://meanderingpi.com/connection-details/ *(u. l. 2023.11.08)*
[2] Bővebb információk az Irodalomjegyzékben találhatók. [1]

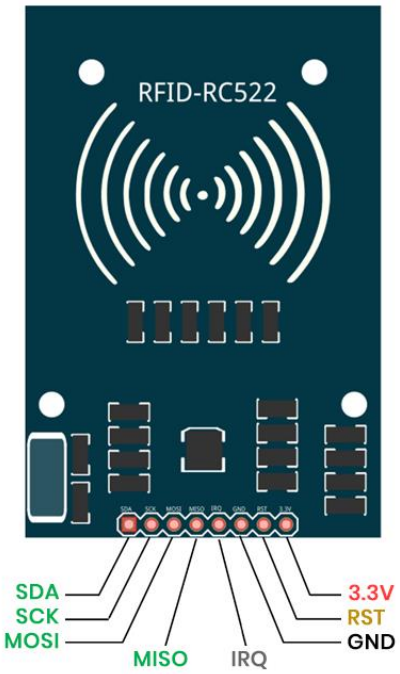
- Áramfelvétel: Maximálisan 240mA áramot fogyaszt WiFi használata közben, amit az USB port szolgáltat.
- Frekvencia: A modul a 2.4 GHz-es WiFi sávot használja, és támogatja az 802.11 B/G/N szabványokat, ami jó teljesítményt és kompatibilitást biztosít.
- Biztonság: A modul támogatja a különböző WiFi biztonsági protokollokat, például: OPEN, WEP, WPA_PSK, WPA2_PSK és WPA_WPA2_PSK. Ez lehetővé teszi az adatok biztonságos átvitelét a hálózaton.
- Működési mód: Képes működni kliensként, hozzáférési pontként, valamint mindkét szerepben egyszerre. Ez nagyfokú rugalmasságot nyújt az alkalmazások tervezéséhez.
- Kommunikáció: A modul támogatja a TCP és UDP kommunikációt, amelyek lehetővé teszik a kétirányú adatcserét más eszközökkel vagy szolgáltatásokkal.
- Mintahardware: nyomógomb, flash nyomógomb, külső LED,
- USB illesztő: CP2102 chip

Az adott számítógépen való felismeréshez elengedhetetlen, hogy letöltsük és telepítsük a megfelelő drivert, amely lehetővé teszi a számítógép és az eszközünk közötti kommunikációt. Ez majd a telepítés alfejezet résznél bővebben kifejtésre kerül.

A GPIO portok, vagyis a General Purpose Input/Output portok, alapvető fontosságúak az elektronikai projektekben. Ezek az univerzális interfészek lehetővé teszik a digitális jelek bemenetét és kimenetét, így a fejlesztők kommunikálhatnak, érzékelhetnek és vezérelhetnek különböző eszközöket. A GPIO portok rugalmasan alkalmazhatók, és lehetővé teszik testreszabott projektek létrehozását, mint például: robotok, okosotthon rendszerek vagy érzékelőrendszerek.

Az alacsony szintű kommunikációs megoldások alkalmazásával teljes mértékben irányíthatóvá válnak a digitális jelek, és ez a lehetőség különösen kiemelkedő fontosságú olyan projektek esetében, ahol kiemelt prioritást élvez a pontosság és az érzékelés. A szóban forgó technológia lehetővé teszi a felhasználók számára, hogy teljes kontrollt gyakoroljanak az adatok felett, optimalizálva azokat és alkalmazkodva az adott feladathoz, legyen szó mérnöki alkalmazásokról, tudományos kutatásokról vagy bármely olyan területről, ahol a digitális jelek alapvető fontosságúak. Ezen kívül, az alacsony szintű kommunikáció hozzájárul a rendkívül precíz működéshez, növelve ezzel a projektek hatékonyságát és megbízhatóságát.

## 2.2. RFID (MFRC522)




*2.ábra: Mifare RFID író/olvasó (MFRC522; 13.56MHz)*[3]

Az RFID MFRC22 egy olyan RFID (Radio Frequency Identification) olvasó, amely képes azonosítani és kommunikálni RFID címkékkel vagy kártyákkal. Az MFRC522 integrált áramkört 13.56MHz-es érintés nélküli kommunikáció során használják kártyamemóriák kezelésekor. A technológiát az NXP hozta létre, kis költségű és kisméretű, kontaktus nélküli okos eszközöket tesz lehetővé, amelyek ma már szinte mindennapjaink részévé váltak.

Az MFRC522 áramkör az ISO14443A szabványon alapul, és a Mifare Crypto-1 titkosítási algoritmus kódolását használja. Ez azt jelenti, hogy ez az áramkör alkalmas a különböző okoskártya technológiák kezelésére, és biztonságos kommunikációt tesz lehetővé a kártya és az olvasó között. Az adatátvitel sebessége akár 424 kbit/sec is lehet, ami lehetővé teszi a gyors és hatékony adatcserét.

Az RFID-RC522 modul az eredeti Philips MFRC522 chipre épül, ami garanciát nyújt a minőségre és a megbízhatóságra. A kártyaolvasó modul 3.3V feszültségen működik, és az SPI (Serial Peripheral Interface) buszon keresztül kommunikál a mikrokontroller rendszerrel. Ennek eredményeként könnyen integrálható különböző elektronikai projektekbe, és lehetővé teszi az érintés nélküli azonosítás és adatkezelés különböző alkalmazásait. Az MFRC522 áramkör egy olyan technológiai eszköz, amely sokféle iparágban és alkalmazási területen forradalmasítja az érintés nélküli kommunikációt és az okoskártya technológiát (*2. ábra*).

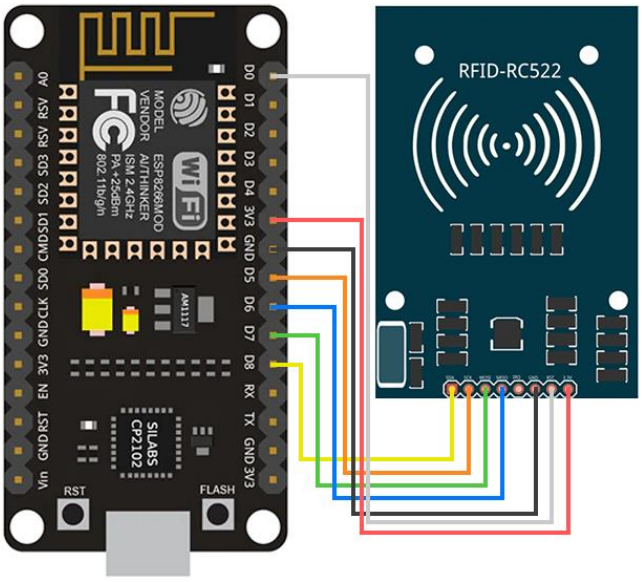
[3] Kép forrása: https://linuxhint.com/interface-rfid-rc522-sensor-arduino-nano/ *(u. l. 2023.11.08)*

Az olvasó paraméterei az alábbiak[4]:

- Áramfelvétel: Az olvasó áramfelvétele 13-26mA között van, amikor 3.3V-os tápfeszültséget használ.
- Nyugalmi áram: Amikor az olvasó inaktív állapotban van, a nyugalmi áram 10-13mA között mozog. Ez azt jelenti, hogy az olvasó készenlétben van, de még nem olvas vagy kommunikál RFID címkékkel.
- Csúcsáram (írás): Amikor az olvasó ír egy RFID címkére vagy kártyára, a csúcsáram kisebb, mint 30mA.
- Működési frekvencia: Az olvasó 13.56MHz-es működési frekvenciát használ. Ez a gyakoriság lehetővé teszi az olvasót és a címkék valós időben történő kommunikációját.
- Kommunikáció: Az olvasó SPI (Serial Peripheral Interface) kommunikációs interfészt használ, és támogatja a maximális 10Mbit/s átviteli sebességet.

Rendeléskor az MFRC522 nem rendelkezett lábakkal, külön mellékelt tüskesor járt hozzá, utólag egy 90°-os sort forrasztottam hozzá, forrasztópáka segítségével.

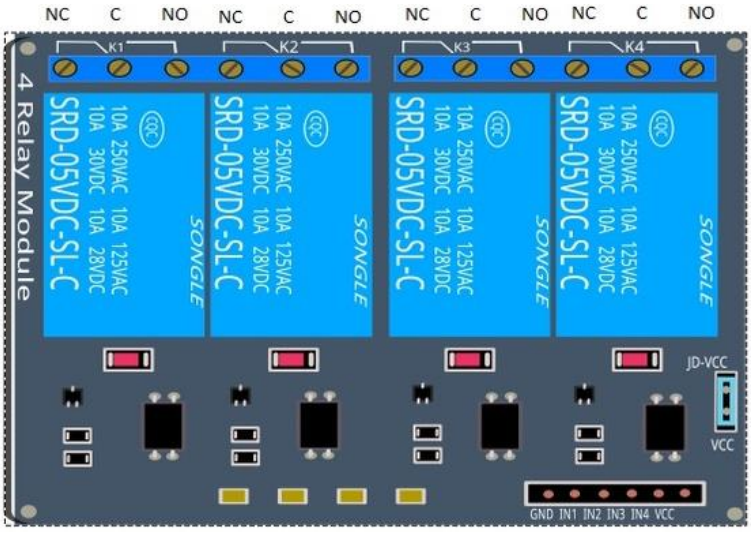
Az ESP8266 és az MFRC522 összekapcsolása során fontos, hogy megfelelően csatlakoztassuk a tüskesorokat, és gondoskodjunk a megfelelő tápellátásról, hogy a modul stabilan működjön. Az ilyen technikai részletek figyelembevétele nélkülözhetetlen az elektronikai projekt tervezése és megvalósítása során. Az ESP8266 és az MFRC522 összeköttetését (*3. ábra*) az alábbi módon valósítottam meg:




*3.ábra: RFID kapcsolási rajz*

[4] Bővebb információk az Irodalomjegyzékben találhatók. [2]

## 2.3. Relé modul




*4.ábra: Négy csatornás relé* [5]

A 4-csatornás (*4. ábra*) relémodul egy olyan eszköz, amely lehetővé teszi a hálózati fogyasztók könnyű és távvezérelhető kezelését mikrokontrolleres környezetben. Ez az eszköz kiválóan alkalmas olyan projektekhez, ahol a különböző berendezéseket, például: lámpákat vagy elektromos készülékeket, távvezérléssel kell kapcsolni.

Ezen relémodulnak a kisáramú vezérlést optocsatolt bemenetek biztosítják, ami lehetővé teszi a biztonságos és megbízható vezérlést a mikrokontroller vagy más vezérlőeszközök segítségével. Emellett a modul stabil működését a reléken keresztüli függetlenítés biztosítja az 5V rendszerfeszültségtől.

A relémodul négy darab 250V/10A relével van felszerelve, amelyek mind meghúzott, mind elengedett állapotban külön kontaktust biztosítanak. Ez lehetővé teszi a különböző fogyasztók egyszerű és hatékony kapcsolását.

Fontos megjegyezni, hogy az áramkörre kapcsolható teljesítmény legfeljebb 35V/8A lehet. Emellett az eszköz nem tartalmaz érintésvédelmet és túlfeszültség/túláram védelmet, így ezekre a szempontokra a projekt tervezésekor különös figyelmet kell fordítani.

A relémodul támogat számos mikrokontroller processzort, például: AVR (Arduino), PIC, MSP430, 8051, és ARM-et. Támogatott programnyelvek közé tartozik a C, Arduino-00xx és Arduino-1.x, Bascom-AVR, MicroPascal.

Emellett számos szolgáltatást nyújt, például: visszajelzést a csatornánkénti aktivitásról, valamint a relé-rendszerfeszültséget 5V-os szinten kezeli.

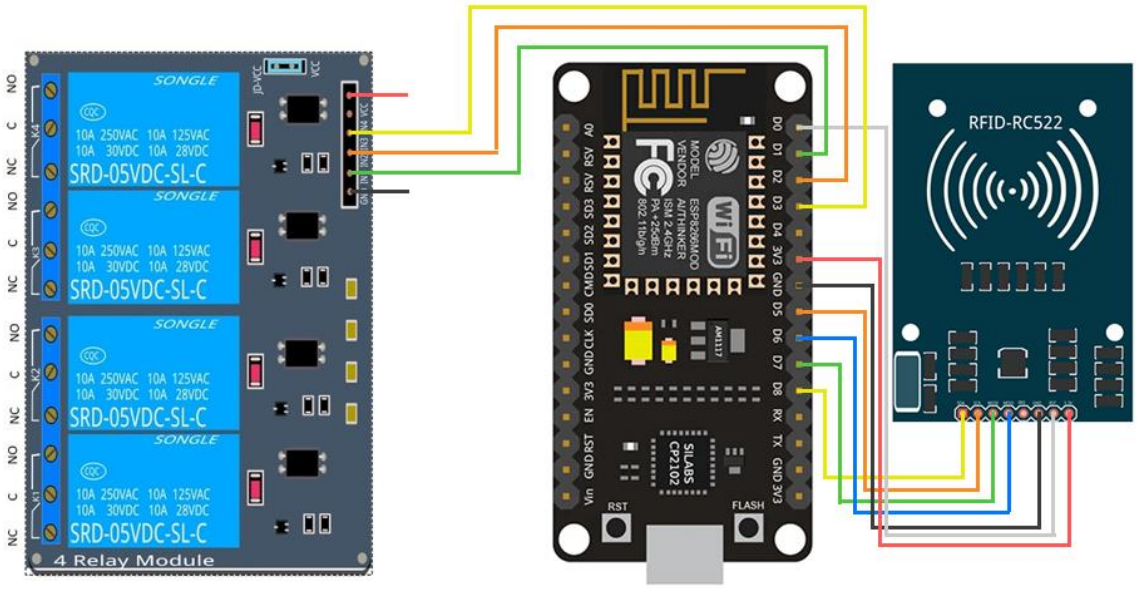
[5] Kép forrása: https://timgolisch.wordpress.com/2015/09/12/fritzing-4-channel-relay-part/ *(u. l. 2023.11.08)*

A maximális áramfelvétel 250mA minden relé meghúzott állapotban, és a vezérlőfeszültség/áram 2.5V (0.1mA) és 5V (0.18mA) között változhat. Ez a relémodul sokféle alkalmazásban használható, ahol a távvezérlés és az elektronikai kapcsolások elengedhetetlenek.[6]

Az "NC," "C," és "NO" rövidítések a relék kapcsolóinak állapotait jelzik. Ezek a rövidítések a következőképpen értelmezhetők:

- NC (Normally Closed - Általában zárva): A NC kapcsolóállapot azt jelenti, hogy a relé normál állapotban zárt (bezárt) állapotban van, amíg nincs aktiválva.
- C (Common - Közös): A "C" a relé közös (common) csatlakozását jelöli. Ez a központi pont az áramkörben, amelytől a kapcsolóállapotok függenek.
- NO (Normally Open - Általában nyitva): Az "NO" kapcsolóállapot azt jelzi, hogy a relé normálállapotban nyitva (kihúzott) állapotban van, amíg nincs aktiválva. Ez azt jelenti, hogy az áramkörben lévő fogyasztók vagy eszközök közötti kapcsolat nyitva van, amíg a relét be nem kapcsolják.

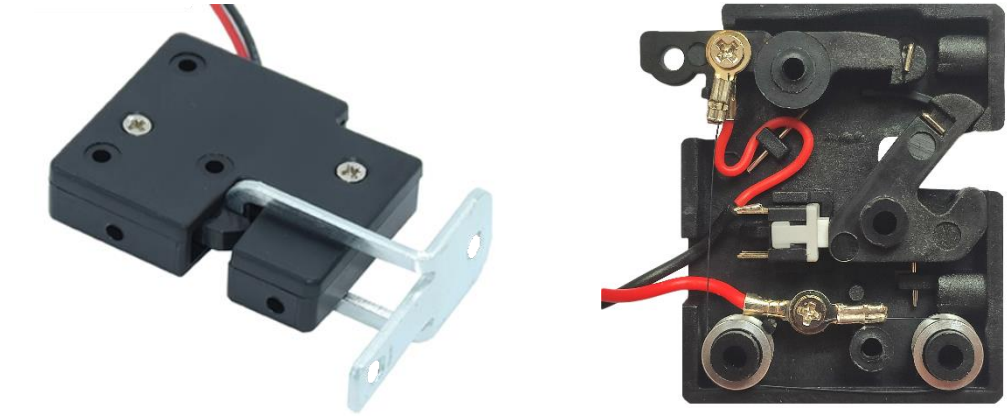
Ezek az állapotok alapvetők a relék működésének megértéséhez. Amikor a relé aktiválódik vagy kapcsol, átkerül egyik vagy másik alapállapotba, azaz az NC vagy NO helyzetbe. Ez pedig lehetővé teszi, hogy a relé által kapcsolt áramkörben található fogyasztókat vagy eszközöket vezéreljük vagy kapcsoljuk. Az alábbi módon (*5. ábra*) kapcsolhatjuk össze az ESP-vel:




*5.ábra: Relé kapcsolási rajz*

[6] Bővebb információk az Irodalomjegyzékben találhatók. [3]

## 2.4. Elektromos zár



*6.ábra: Mini elektromos zár* [7]

Ez a zár egy olyan szerkezet, amely számos alkalmazásban használható és könnyen integrálható különböző rendszerekbe (*6. ábra*). A következőkben részletesen ismertetem ennek jellemzőit és előnyeit:

- Alacsony feszültségigény (5V, 0.6A): rendkívül hatékonyan működik alacsony feszültség mellett, ami azt jelenti, hogy könnyen beilleszthető bármilyen rendszerbe vagy projektbe anélkül, hogy nagy energiaigényt igényelne.
- ABS műanyagból készült: Az ABS műanyagból való gyártásának köszönhetően a zár könnyű, ugyanakkor rendkívül tartós is. Ez lehetővé teszi a hosszú élettartamot és ellenáll a kopásnak és a környezeti hatásoknak.
- Belső biztonsági kézi kapcsoló: A zár hátulján található belső biztonsági kézi kapcsoló lehetővé teszi a kézi vezérlést, amely hasznos lehet olyan helyzetekben, amikor az automatikus zárás vagy nyitás nem kívánatos.
- Erős húzószilárdság (20 kg): A zár kiemelkedő húzószilárdsággal rendelkezik, ami azt jelenti, hogy képes tartani nagy terhelést anélkül, hogy meghibásodna.
- Könnyű bekötés: A fekete kábel a földeléshez (GND) csatlakozik, míg a piros kábel a pozitív feszültségforrás (VCC) bekötéséhez szolgál.
- Kompakt méret: A zár kompakt mérete (42 mm x 34 mm x 10 mm) lehetővé teszi, hogy szűk helyeken is használható legyen anélkül, hogy sok helyet foglalna el.

[7] Kép forrása: https://www.aliexpress.us/item/3256805025131562.html *(u. l. 2023.11.08)*

## 2.5. Tápellátás

Ahhoz, hogy a projekt során elegendő feszültséggel és áramerősséggel elláthassam a rendszert, kénytelen voltam beszerezni egy külső adaptert. Ez az adapter specifikációiban teljes mértékben megfelelt az elvárásaimnak: 5V-os kimeneti feszültséget és 6A-es kimeneti áramerősséget biztosított, mely összesen 30W-os teljesítményt jelentett. Az adapter egy műanyag házban helyezkedik el, amely védelmet nyújt a belsejében található elektronika számára (*7. ábra*).

A méretei is ideálisnak bizonyultak a projektben való használathoz, hiszen mindössze 5,5x2,5x13mm-es volt, és súlya is csak 170g. Ezek a kompakt méretek lehetővé tették a könnyű elhelyezést és mozgatást a projekten belül.

Az adapteren található DC csatlakozónak a végére egy sorkapocs átalakítót (*8. ábra*) tettem, ahol a két kábelt egy többpólusú vezetékösszekötőhöz (*9. ábra*) kapcsoltam.


*7.ábra: Tápegység, YDS 5V 6A* [8]

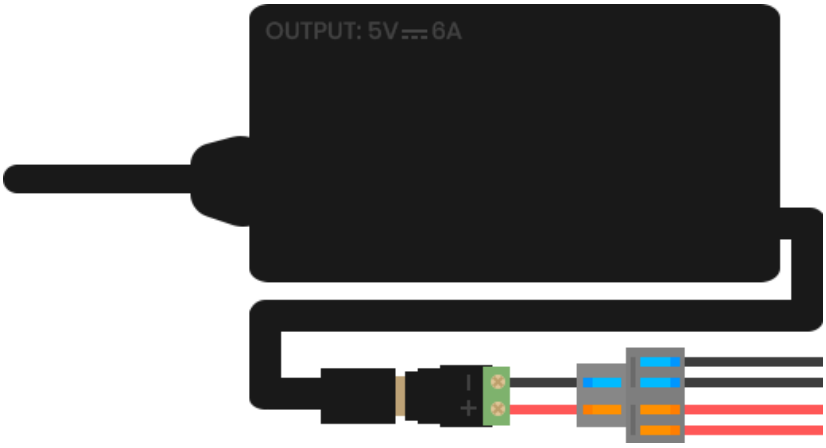
*8.ábra: DC – Sorkapocs átalakító* [9]

*9.ábra: Vezetékösszekötő* [10]

[8] Kép forrása: https://www.emag.hu/tapegyseg-yds-5v-6a-vezetekes-muanyag-hazzal-5-5x2-5x13mm-201801013940/pd/DVXHC5MBM/ *(u. l. 2023.11.08)*
[9] Kép forrása: https://ledlampahaz.hu/spd/AN-4273/DC-Sorkapocs-atalakito-Anya *(u. l. 2023.11.08)*
[10] Kép forrása: https://www.mentavill.hu/termek/vez-osszek-2-4p-4mm2-nyith-5db-187766 *(u. l. 2023.11.08)*

Erre a lépésre szükséges volt azért, mert a rendszerben két különböző elektronikai egységnek volt szüksége különböző pozitív és negatív polaritású feszültségre. Így az egyik vezetékpár a szekrényekhez tartozó pozitív és negatív pólusokat szolgáltatta, míg a másik vezetékpár az ESP pozitív és negatív pólusait szállította.

Ezt a megoldást alkalmazva lehetővé vált, hogy a két különböző rendszer a megfelelő feszültséget és polaritást kapja meg a problémamentes működéshez (*10. ábra*).




*10.ábra: Tápegység kapcsolási rajz*

Szükségem volt egy Micro USB csatlakozóra is, amit az ESP-re fogok tudni csatlakoztatni, így kerestem itthon egy feleslegessé vált töltőkábelt és levágtam az USB-A részét. Egy ilyen kábelben 4 db vezeték található:

- piros (+5V)
- fekete (GND)
- zöld (DATA +)
- fehér (DATA -)

Az ESP-hez történő csatlakoztatáshoz csak a piros és fekete vezetékeket kellett használnom, mivel ezek a vezetékek felelnek meg a pozitív és negatív pólusoknak.

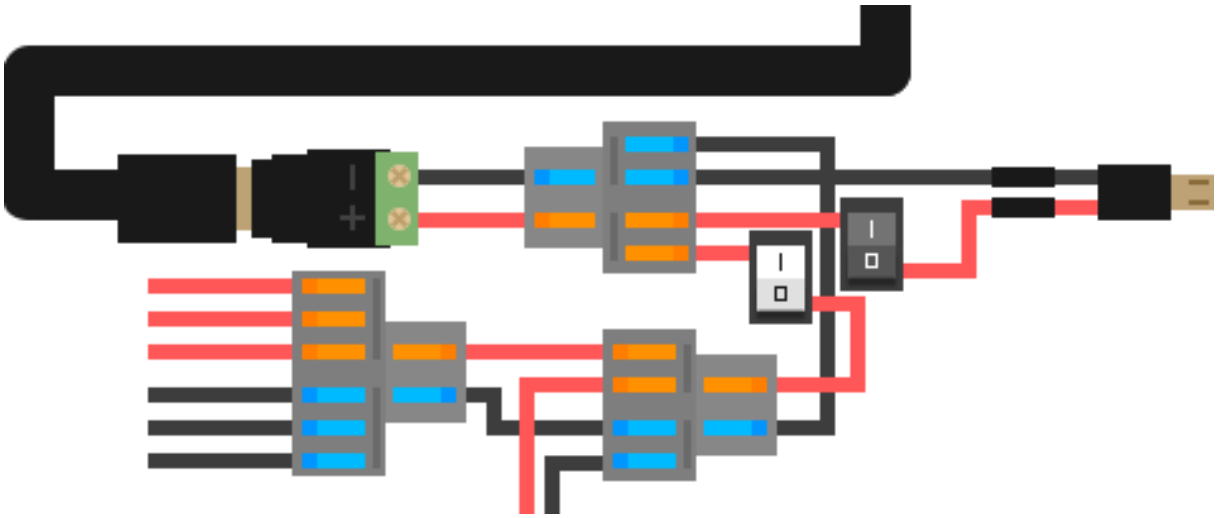
A kábelen található borítást levágtam egy blankoló fogó segítségével, majd szintén elvágtam egy piros és egy fekete apa jumper vezetéket és összeforrasztottam őket a Micro USB vezetékeivel, forrasztás után pedig zsugorcsövet melegítettem rá (*11. ábra*).

*11.ábra: Micro USB kábel*

A projektbe két darab kétállású kapcsolót is beépítettem. Ezek közül az egyik kapcsoló (fekete) az ESP működését irányítja, míg a másik (fehér) a szekrények, relék működését szabályozza. Mindkét kapcsoló rendelkezik két lábbal, amelyekre forrasztottam egy-egy vezetéket (*12. ábra*).

A maximális terhelhetőség 3A 250V AC. Fontos azonban figyelni arra, hogy a terhelés soha ne haladja meg a kapcsolók által megadott maximális értékeket, mivel ez veszélyeztetheti az eszközök és az egész rendszer biztonságos működését.

A szekrényekhez tartozó kapcsoló után kötöttem két vezetékösszekötőt, az egyik a feszültséget biztosítja a relé modul VCC (táplálás) csatlakozójához, míg a másik a relé C (common) csatlakozójába lesz bekötve.



*12.ábra: Kapcsolók*

A projekt végén beszereltem még egy piros színű LED-et is (*13. ábra*), egy 220Ω ellenállással (*14. ábra*), ami a visszajelzésekre szolgál. Például: villog, amíg felkapcsolódik az adott hálózatra, ezt a lépést 15 alkalommal próbálja meg, azaz 15 villogás, majd ha nem talál hálózatot folyamatosan világít és 30 másodperc után újra megpróbál felcsatlakozni, ha sikerült neki, akkor kialszik a LED, viszont ha működés közben hálózati hiba adódik, például: nem elérhető a hálózat vagy a szerver, akkor is elkezd villogni.

*13.ábra: Piros LED* [11]

*14.ábra: 220Ω ellenállás*

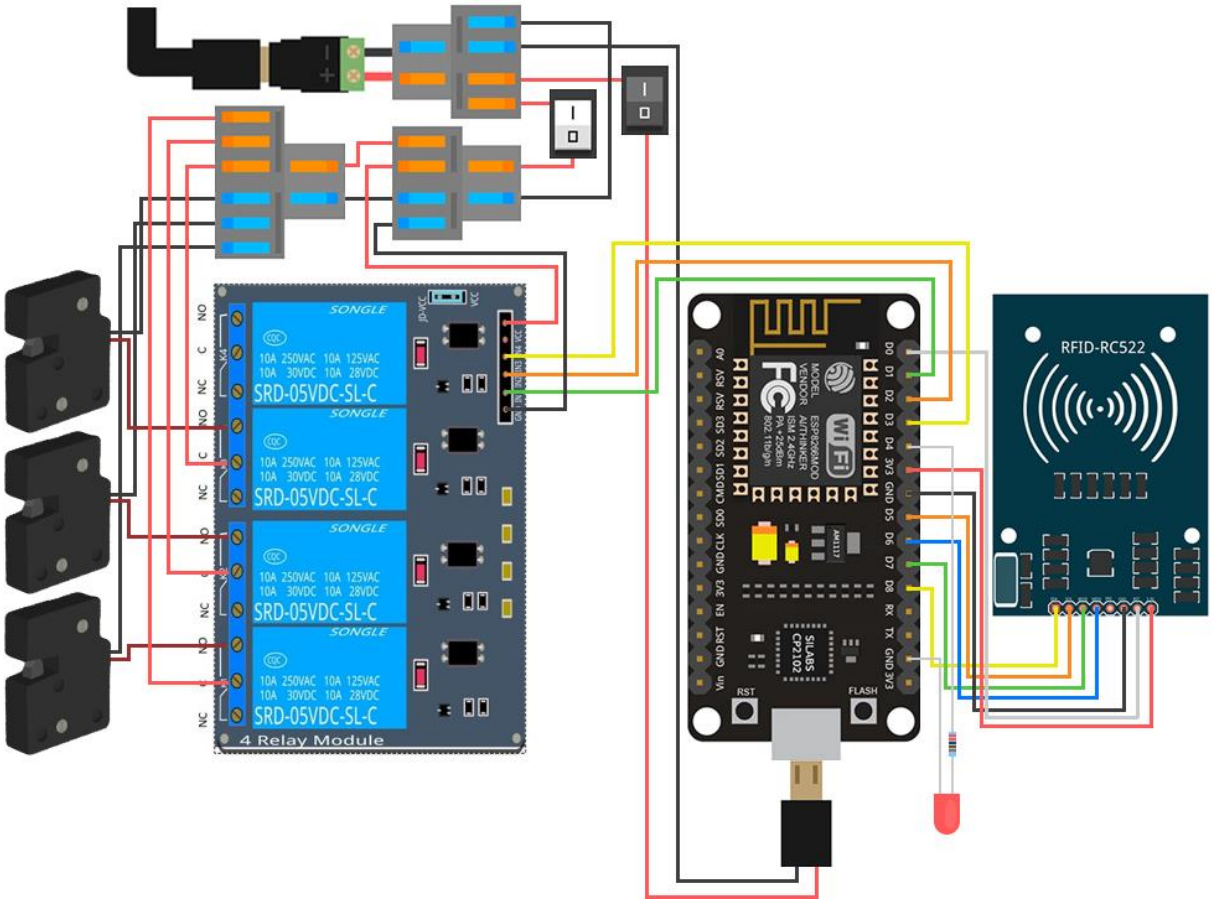
[11] Kép forrása: https://www.pixelsquid.com/png/led-diode-2830790612255511804 *(u. l. 2023.11.08)*

# 3. Hardware összeállítása

Ebben a fejezetben bemutatom, hogy hogyan kell megfelelően összerakni, beállítani és beprogramozni az eddigi komponenseket. Az előző részben megtalálható bekötési rajzok és az addig összegyűjtött információk most alkalmazásra kerülnek, hogy a projekt minden komponense megfelelően működjön együtt.

## 3.1. Modulok összeszerelése

Most már minden egyes komponenshez rendelkezésre áll a bekötési rajz, és a szükséges információk is rendelkezésünkre állnak ahhoz, hogy összeszereljük a projektet. A következő ábra (*15. ábra*) részletesen bemutatja a bekötést és az összekapcsolást:




*15.ábra:Összeszerelés*

## 3.2. Fejlesztői környezet

Miután sikeresen összeraktuk, szükségünk van egy fejlesztő környezetre, amiben megírjuk és elkészítjük a programot, én jelen esetben az Arduino IDE-t használtam.

Mi is azaz IDE? Az angol megfelelője „Integrated Development Environment”, azaz Integrált fejlesztési környezet, röviden: fejlesztői környezet.

Egy pár szót ejtve az Arduino IDE-ről, olyan szoftveralkalmazás, amelyet elektronikai projektfejlesztők és hobbi programozók használnak, hogy egyszerűen és hatékonyan fejlesszék és programozzák a saját mikrovezérlős projektjeiket. Az Arduino mikrovezérlők széles skáláját támogatja. Ingyenesen letölthető és telepíthető. Az Arduino IDE többféle operációs rendszeren működik, beleértve a Windows, a macOS és a Linux rendszereket is, így széles körben elérhető. Egyes főbb jellemzők az Arduino IDE-ben:

- Könnyen használható: A platform egy, egyszerű és felhasználóbarát felülettel rendelkezik, amely lehetővé teszi a programok írását és feltöltését az Arduino mikrovezérlőkre anélkül, hogy bonyolult beállításokkal kellene foglalkoznod.
- Nyelv: Egyedi programozási nyelvet használ, amely nagyon hasonlít a C és a C++ nyelvekre. Egy olyan fordító, amely érti mind a C, mind a C++ nyelvet, gyakorlatilag egy keveréke mindkét nyelvnek. Sok egyedi elemmel és kiterjesztéssel, hogy a kód könnyebben leforduljon a kompatibilis mikrovezérlők által használt speciális bytecode változatává. Sok könyvtár C++ nyelven van írva, de néhány csak C-t használ. Ez a nyelv könnyen tanulható és érthető, így akár kezdők is könnyen használhatják.
- Könyvtárak és példaprogramok: az alkalmazás tartalmaz számos beépített könyvtárat és példaprogramot, amelyek segítenek abban, hogy gyorsan elkezdhess projekteket fejleszteni. Ezek a források segítenek a szenzorok, kijelzők, motorok és más perifériák használatának megértésében és kezelésében.
- Feltöltés: Az Arduino IDE lehetővé teszi a programok feltöltését az Arduino mikrovezérlőkre különböző csatlakozási módokkal, például: USB-kábelen.

Az Arduino IDE a világ egyik legnépszerűbb és legelterjedtebb fejlesztői környezete az Arduino projektekben, sok közösségi támogatás és dokumentáció is található hozzá az interneten.

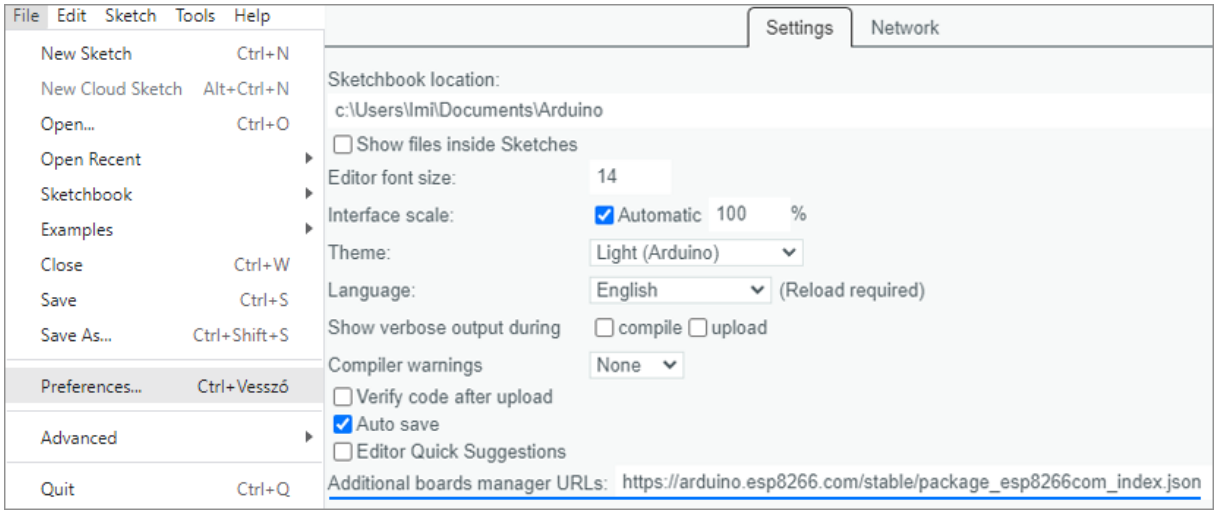
## 3.3. Telepítés

Miután sikeresen feltelepítettük a programot és az ESP CP2102 chiphez tartozó drivert[12], csatlakoztassunk egy Micro-USB kábelt az ESP-hez, a másik végét pedig a számítógépünkhöz, ekkor, ha mindent jól csináltunk a számítógépnek fel kell ismernie. Ezt akár Windows operációsrendszeren belül az Eszközkezelőben is ellenőrizhetjük.

[12] Bővebb információk az Irodalomjegyzékben találhatók. [4]

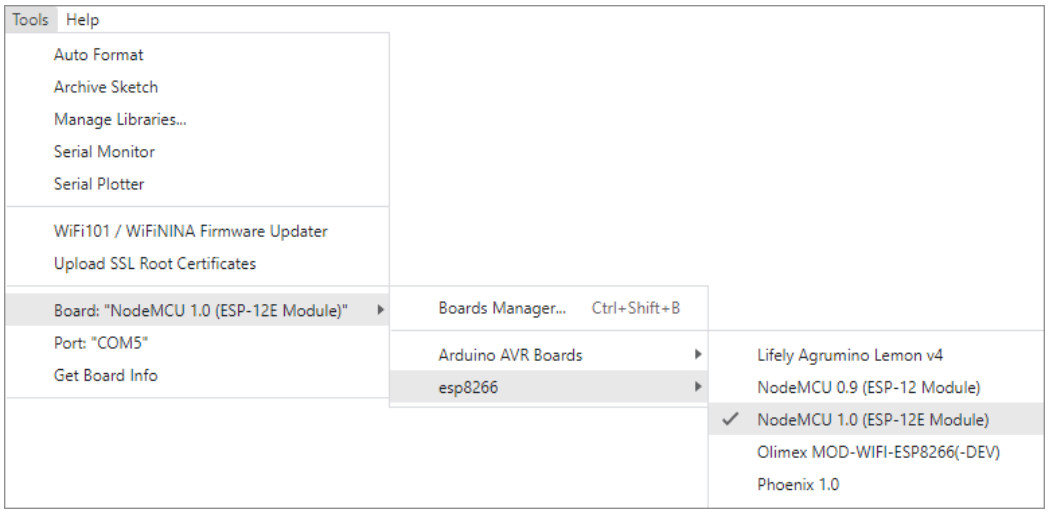
Állítsuk be a megfelelő board packeget, hogy ki tudjuk választani a megfelelő boardot, ehhez a *File*, *Preferences*… menüpontra van szükségünk, keressük olyat, hogy „*Additional boards manager URLs*” és másoljuk be az alábbi URL-t és indítsuk újra a programot (*16. ábra*).

*https://arduino.esp8266.com/stable/package_esp8266com_index.json*



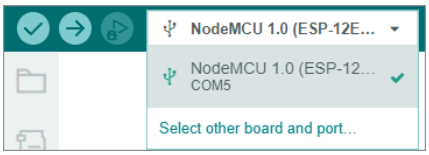

*16.ábra: ESP package beillesztése*

Ezután válasszuk ki a *Tools* menüt, keressük meg a *Board* menüpontot és válasszuk azt, hogy *esp8266*, majd azon belül *NodeMCU 1.0 (ESP-12E Module)*, utána az USB portot amin csatlakozik a számítógéphez, a menüpont a *Board* alatt van, nekem *Port: „COM5”* (*17. ábra*).




*17.ábra: Board és Port beállítása*

Amennyiben jól csináltunk mindent, a program a felső sávjában látnunk kell az általunk kiválasztott *Boardot* és *Portot* (*18. ábra*).




*18.ábra: Sikeres beállítás*

## 3.4. ESP programozás

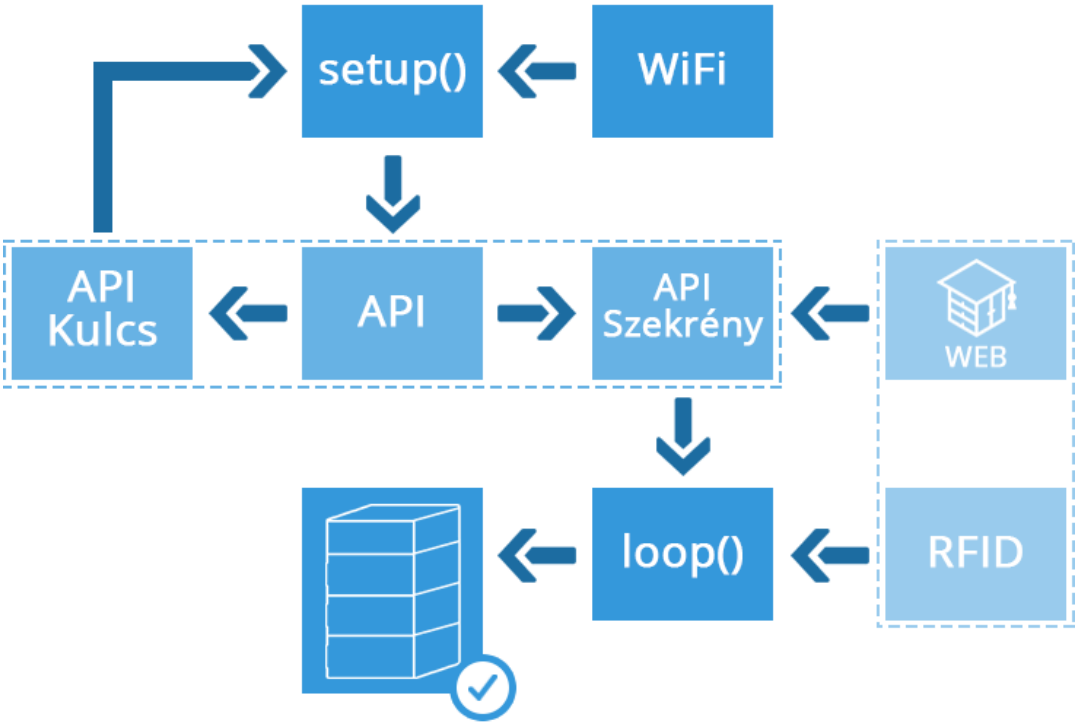
Most már, hogy sikerült beállítanunk az Arduino IDE-t, elkezdhetjük a program megírását. A program a következő fő részekből áll:

- Include-ok importálása: A program először a szükséges könyvtárakat és header fájlokat importálja, mint például: a WiFi és HTTP kliens könyvtárak, a JSON feldolgozásra szolgáló ArduinoJson könyvtár és a MFRC522 RFID könyvtár.
- Konfigurációs adatok: Az ESP8266 WiFi hálózati nevét (SSID) és jelszavát egy külső header fájlban tároltam el, meghívjuk, majd betöltjük ezeket a változókba.
- RFID modul inicializálása: Az RFID modult a megfelelő pinekhez (`SS_PIN` és `RST_PIN`) kapcsoljuk, majd inicializáljuk azt.
- Zárak inicializálása: Definiáljuk a zárak állapotát (`LOCKED` vagy `UNLOCKED`) és az utolsó kinyitás idejét. Az állapotokat egy tömbben tároljuk.
- Relék és LED inicializálása: A programban használt relé pineket és az egy LED pint inicializáljuk.
- API kulcs és API végpontok beállítása: Az API kulcsot egy külső API-tól lekérjük, majd az API kulcsot és a zárakhoz tartozó API végpontokat létrehozzuk.
- WiFi-hez való csatlakozás: A program megpróbál csatlakozni a WiFi hálózathoz, és ha nem sikerül, 30 másodperces várakozás után újra próbálkozik.
- API kulcs ellenőrzése: Az API kulcsot rendszeresen ellenőrizzük. Ha a kulcs megváltozik, újra indítjuk az eszközt.
- RFID olvasás: Az RFID modul folyamatosan olvassa, és tárolja az RFID kártyák azonosítóit.
- Zárak kezelése: Az API végpontokon keresztül lekérjük a zárak állapotát és az azokhoz tartozó kártyák azonosítóit. A program ennek megfelelően vezérli a zárakat.
- Időzítő: A program periodikusan végrehajtja az RFID olvasást és az API hívásokat egy meghatározott időközönként.

A program összekapcsolja az RFID kártyák azonosítóit a zárak állapotával, és lehetővé teszi a távoli vezérlést az API kulcs és végpontok segítségével. Ha a program új API kulcsot észlel, az eszköz újraindul biztonsági okokból. Az időzítő segít az adatok rendszeres frissítésében és a zárak állapotának ellenőrzésében.

### 3.4.1. System Design

Készítettem egy System Design-t (*19. ábra*), ami egy képen megmutatja, hogy hogyan is működik a program, úgy gondolom, ez segít könnyebben megérteni a program működését.




*19.ábra: ESP System Design*

## 3.5. 3D nyomtatás

A 3D nyomtatás egy rendkívül izgalmas és innovatív technológia, amely lehetővé teszi, hogy saját elképzeléseinket valósággá formáljuk. Először ki kellett választanom, hogy milyen anyagból szeretném, az alap 3 alapanyag az a PLA, PETG, ABS.[13] A PLA előnyei, hogy környezetbarát, biológiailag lebomlik, alacsony nyomtatási hőmérsékletet igényel. A hátrányai, hogy kisebb a hőállósága, emiatt hajlamos deformálódni vagy megolvadni nagy hőhatás esetén. Kevésbé strapabíró, mint az ABS vagy PETG.

Az ABS előnyei, hogy erős, tartós, jó hő és ütésállósággal rendelkezik. Hátrányai viszont, hogy a túlzott UV-fénynek való kitettség az ABS-t törékennyé teszi.
Tehát jobb, ha nem ABS-t választunk kültéri alkalmazásokhoz.

A PETG általában vizes palackokban és élelmiszer-tartályokban található meg. Előnyei: nagyon tartós és ütésálló, ezért alkalmas a funkcionális alkalmazásokhoz és prototípusokhoz. Hátránya az ABS-hez hasonlóan a PETG is érzékeny az UV-fényre, ami azt jelenti, hogy leginkább belső alkalmazásokhoz használható.

[13] Bővebb információk az Irodalomjegyzékben találhatók. [5]

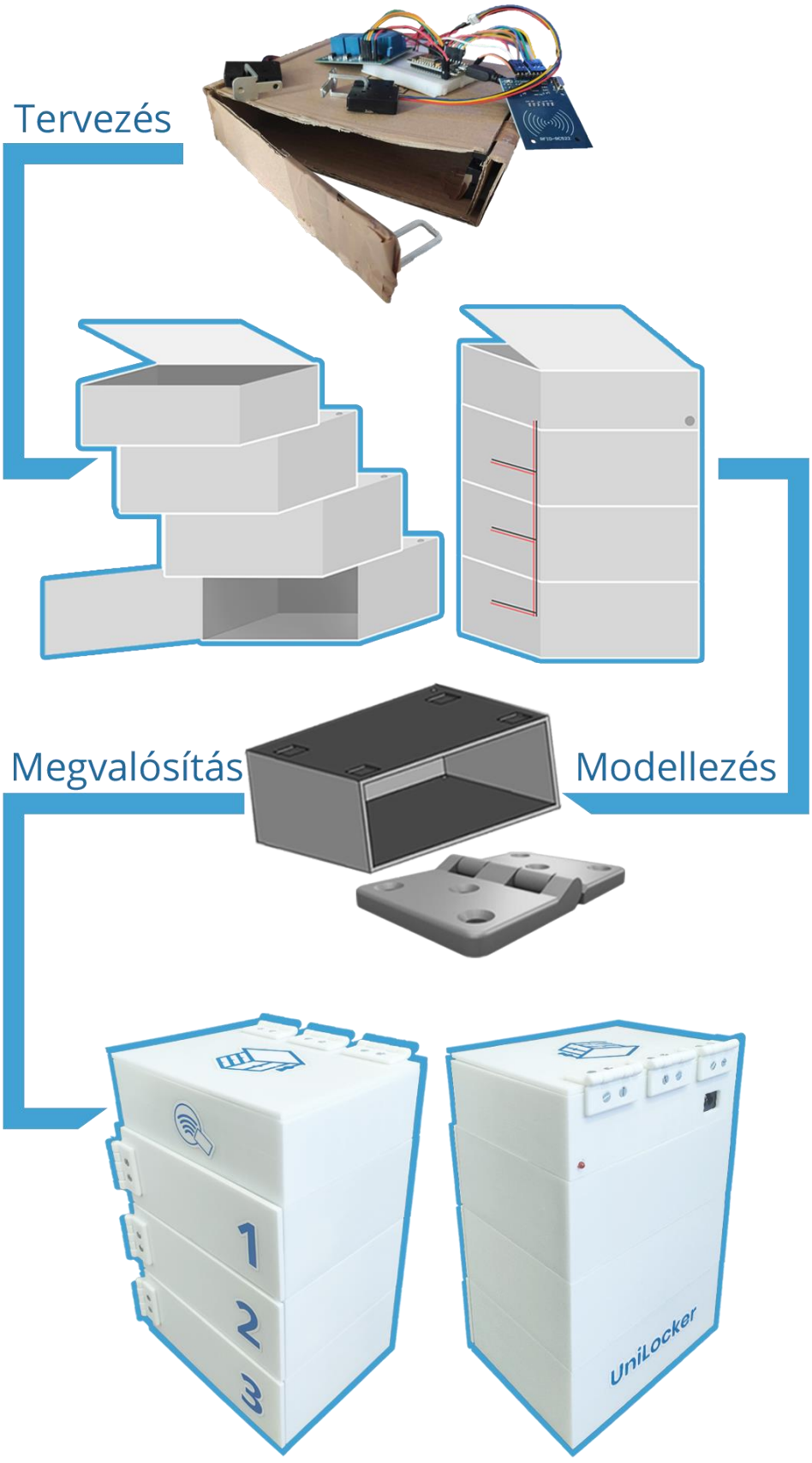
Az általam választott alapanyag fehér PETG volt, bár számos más 3D nyomtatási anyag is rendelkezésre áll. Azért választottam, mert rendkívül tartós. Ez ideális választásnak bizonyult a szekrény elkészítéséhez, mivel a tervezés célja a funkcionális és strapabíró készülék létrehozása volt.

A tervezési folyamat során először a Photoshop képszerkesztő programban készítettem el a terveket, amelyek később a valóságban is sikeresen megvalósultak. Az alsó rész méretei meghatározottak voltak: 20 cm hosszú, 15 cm széles és 8 cm magas, előre nyíló ajtókkal, amelyekben a zárak találhatóak. A legfelső rész mérete 20x15x6 cm, és ebben a részben helyezkednek el a szekrény elektronikai komponensei. A teteje felfelé nyitható, ami lehetővé teszi a könnyű hozzáférést és karbantartást. Az ajtók, zsanérok segítségével működnek, amelyek külön nyomtatással készültek el, majd csavarokkal lettek rögzítve. Az ilyen zsanérok alkalmazása azért volt fontos, hogy esetleges meghibásodás esetén könnyen cserélhetőek legyenek. A falak vastagsága 4 mm, amely a megfelelő stabilitást és tartósságot biztosítja a szekrényeknek. A dobozok alján található, 4 db négyzet alakú 2 mm-es kiemelkedés, amelyek pontosan illeszkednek az alsó dobozok tetején található mélyedésekbe, így a dobozok egymásra helyezhetők és nem csúsznak el.

Az elkészült szekrény bérnyomtatással készült. Az alap nyomtató egy Creality Ender 3 Pro volt, de alapos fejlesztéseken esett át, mind mechanikai, mind szoftveres szempontból. A nyomtatón a legfrissebb Marlin firmware futott, és a modellek szeleteléséhez a Super Slicer szoftvert volt használva. A rétegmagasság 0.2 mm-re volt beállítva az utolsó rétegig, és ez egy 0.4-es fúvókával lett nyomtatva, 40 mm/s sebességgel, hogy a falak minél simábbak és esztétikusabbak legyenek.

Az elkészült projekt a Freecad programban lett megtervezve, és a szekrény részleteinek gondos tervezése és megvalósítása szemléletesen mutatja be a 3D nyomtatás sokoldalúságát és az alkotói szabadságot, amelyeket ez a technológia kínál. A végeredmény egy tartós, funkcionalitásban és esztétikában is kiváló szekrény, amely megmutatja a 3D nyomtatásban rejlő lehetőségeket és a tervezési kreativitást.

## 3.6. Elkészülés folyamata




*20.ábra: Elkészülés folyamata*

# 4. Software összeállítása

A fejezet, amely a tárhely vásárlásáról, az adatbázis létrehozásáról, az API felépítéséről és a weboldal létrehozásáról szól, az egyik legfontosabb és kritikusabb szakasz egy projekt vagy alkalmazás fejlesztése során.

## 4.1. Tárhely

Ahhoz, hogy elkezdjünk egy weboldalt vagy webalkalmazást létrehozni, először is szükségünk van egy tárhelyre. Számos tárhelyszolgáltató található az interneten, különböző ajánlatokkal. Az ajánlatokban eltérések lehetnek például: a lemezhasználatban, az aldomainek számában, az adatbázisokban és a sávszélességekben. Az első lépésünk tehát az, hogy megtaláljuk a megfelelő tárhelyszolgáltatót, és megvásároljuk a kiválasztott csomagot.

Ezen kívül szükségünk van egy domain névre is, amely a weboldalunk vagy alkalmazásunk címét fogja meghatározni az interneten. A domain nevet kiválasztva ellenőrizhetjük, hogy foglalt-e vagy sem. Ez az opció a legtöbb szolgáltatónál elérhető. A domain név kiterjesztése is fontos szempont, például: .hu, .com, .xyz. A domain név kiterjesztése befolyásolja az árat.

A webtárhely-szolgáltatók általában rendelkeznek egy olyan felhasználói felülettel, amelyet a cPanelnek (Control Panel) neveznek. Ezen a felületen számos funkcióhoz férhetünk hozzá, amelyek segítségével könnyedén kezelhetjük weboldalunkat és a hozzá kapcsolódó szolgáltatásokat. A cPanel lehetőséget biztosít az adatbázisok létrehozására, a fájlok feltöltésére és kezelésére, valamint a domainek és aldomainek beállítására is.

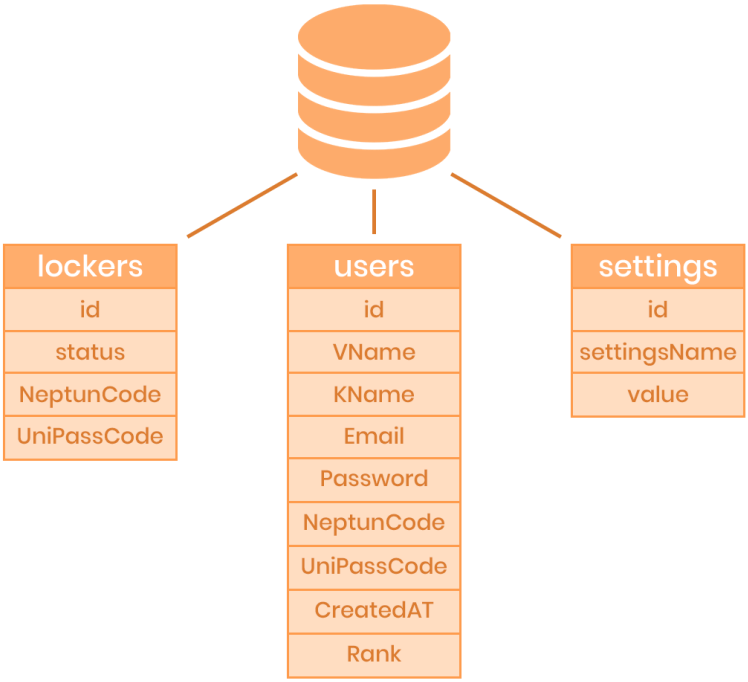
Az aldomainek lehetővé teszik, hogy különböző részekre vagy szolgáltatásokra mutató címeket hozzunk létre a fődomain alatt, így könnyen elérhetőek lesznek a weboldalunk különböző részei vagy szolgáltatásai.

Én már rendelkeztem egy meglévő tárhellyel és domain névvel, amely a toxy.hu, ezen a domain néven hoztam létre további két aldomaint:

- unideb.toxy.hu: az alkalmazásomért felelős, ezt a domaint érik el a felhasználók akik használni szeretnék az UniLockert.
- api.toxy.hu: ez az aldomain pedig az API (alkalmazásprogramozási felület) számára van fenntartva. Az API egy kulcsfontosságú része az alkalmazásnak, itt történik a kommunikáció a web és az ESP között.

A cPanelen keresztül feltölthetjük a fájljainkat, illetve megnyithatjuk a phpMyAdmin-t ahol az adatbázisunk fog megjelenni. Én Total Commanderben FTP kapcsolaton keresztül szoktam feltölteni a szükséges fájlokat.

## 4.2. Adatbázis




*21.ábra: Adatbázis felépítése*

Az adatbázist eleinte úgy terveztem, hogy két táblát hozok létre, viszont a későbbiekben szükség volt egy harmadikra is, a settings táblára. Így az adatbázis három fő táblát tartalmaz: szekrények (`lockers`), felhasználók (`users`) és beállítások (`settings`) (*21. ábra*). Hadd bontsam részletesen le, hogy milyen információk tárolódnak ezekben a táblákban.

- Szekrények (`lockers`) tábla:
    - id: Ez egy egyedi azonosító, amely minden szekrényhez hozzá van rendelve. Automatikusan növekszik, és egyedi azonosítót biztosít a szekrények számára.
    - status: Ez a mező lehetőséget ad arra, hogy nyomon kövessük a szekrények állapotát. Két értéke lehet: "`off`" vagy "`on`". Ezek a státuszok azt jelzik, hogy a szekrény éppen nyitva vagy zárva.
    - NeptunCode: Ez a mező tartalmazza a hallgató Neptun kódját, aki éppen használja a szekrényt. Ez lehetővé teszi a szekrényhasználat nyomon követését és a felhasználók azonosítását.
    - UniPassCode: Ez a mező tárolja a hallgató Unipass kártya kódját, ami szintén segít azonosítani a felhasználót a kártyával történő szekrény nyitás közben.

- Felhasználók (`users`) tábla:
    - id: Ez a mező hasonlóan az előzőhöz egy egyedi azonosító, amely minden felhasználóhoz tartozik.
    - VName és KName: Ezek a mezők tárolják a felhasználók vezeték és keresztnevét.
    - Email: Az email cím segíti az azonosítást és a kapcsolattartást a felhasználókkal.
    - Password: A jelszó lehetővé teszi a felhasználók bejelentkezését. Ügyelve a biztonságos tárolásra.
    - NeptunCode: Ez a mező tárolja a felhasználó Neptun kódját, ami egy másik azonosítási mód, illetve a belépéshez szükséges.
    - UniPassCode: a regisztrációkor beolvasott Unipass kártya azonosítóját tároljuk el.
    - CreatedAT: Ez a mező rögzíti azt az időpontot, amikor a felhasználó létrehozta a fiókját.
    - Rank: A felhasználók rangja lehet "`Student`" vagy "`Admin`", és ezzel a jogosultságok szabályozhatók. Az "`Admin`" rangú felhasználók például: további funkciókhoz vagy jogosultságokhoz férnek hozzá.
- Beállítások (`settings`) tábla:
    - id: szintén egy egyedi azonosító.
    - settingsName: Ez a mező tárolja a beállítás nevét, amelyet később az alkalmazásban használhatsz.
    - value: A beállítás értéke ebben a mezőben van tárolva, és lehetővé teszi a konfigurációs beállítások könnyű módosítását és kezelését.

Ez az adatbázis struktúra úgy gondolom, hogy jól szervezett és jól áttekinthető. Lehetővé teszi a szekrények és a felhasználók közötti kapcsolat nyomon követését. Ezen kívül a rangrendszer segít a felhasználók jogosultságainak hatékony kezelésében.

Az adatbázisokat a MySQL adatbáziskezelő rendszerben hoztam létre, valamint a phpMyAdmin nevű webes felületen hajtottam végre a szükséges műveleteket. A MySQL egy erőteljes és elterjedt adatbáziskezelő rendszer, melynek segítségével definiálhatjuk az adatbázisok struktúráját és tárolhatjuk az adatokat. A phpMyAdmin egy webes alkalmazás, mely lehetővé teszi az adatbázisok kezelését és adminisztrációját a böngészőből, így könnyen hozhatunk létre, módosíthatunk vagy törölhetünk adatbázisokat, táblákat, illetve adatokat a MySQL rendszerben.

## 4.3. API

Ahhoz, hogy a webalkalmazás és az ESP tudjon kommunikálni egy API-t hoztam létre, ami lényegében egy megbízott, akin keresztül történik a feladat elvégzés. A JSON formátumú adatok segítségével könnyen és strukturált módon továbbítom az információkat, amennyiben hiányzó vagy hibás paramétereket adunk meg, azokat közlöm a felhasználóval. Három fő php fájl található, az alábbiakban részletesebben bemutatom ezeket a fájlokat és azt, hogy milyen szerepük van a rendszerben:

`read_api.php`:

https://api.toxy.hu/read_api.php?esp

- Funkció: Ez az a fájl amit legelőször lekérdez az ESP, a kulcs miatt.
- Működés:
    - Ez a fájl a rendszer adatbázisából kinyeri azokat az információkat, amelyeket az ESP vagy más eszköz későbbi használatra szolgáltat.
    - Tartalmazza az API kulcsot: A hozzáférés biztosítása érdekében az API kulcsot tárolja, ami szükséges az adatokhoz való hozzáféréshez a további php fájloknál.
    - Tartalmazza a weboldalon megjelenő szekrények számát: Az adatbázisból kinyeri, hogy hány szekrény jelenik meg a weboldalon.
    - Tartalmazza a fizikailag elérhető szekrények számát: Szintén az adatbázisból olvassa a fizikailag elérhető szekrények számát.
    - Hozzáférés: Az "`esp`" lekérdezési karakterlánc hozzáadásával lehet hozzáférni ehhez a fájlhoz.

`read_all.php`:

https://api.toxy.hu/read_all.php?id={ID}&apikey={APIKULCS}

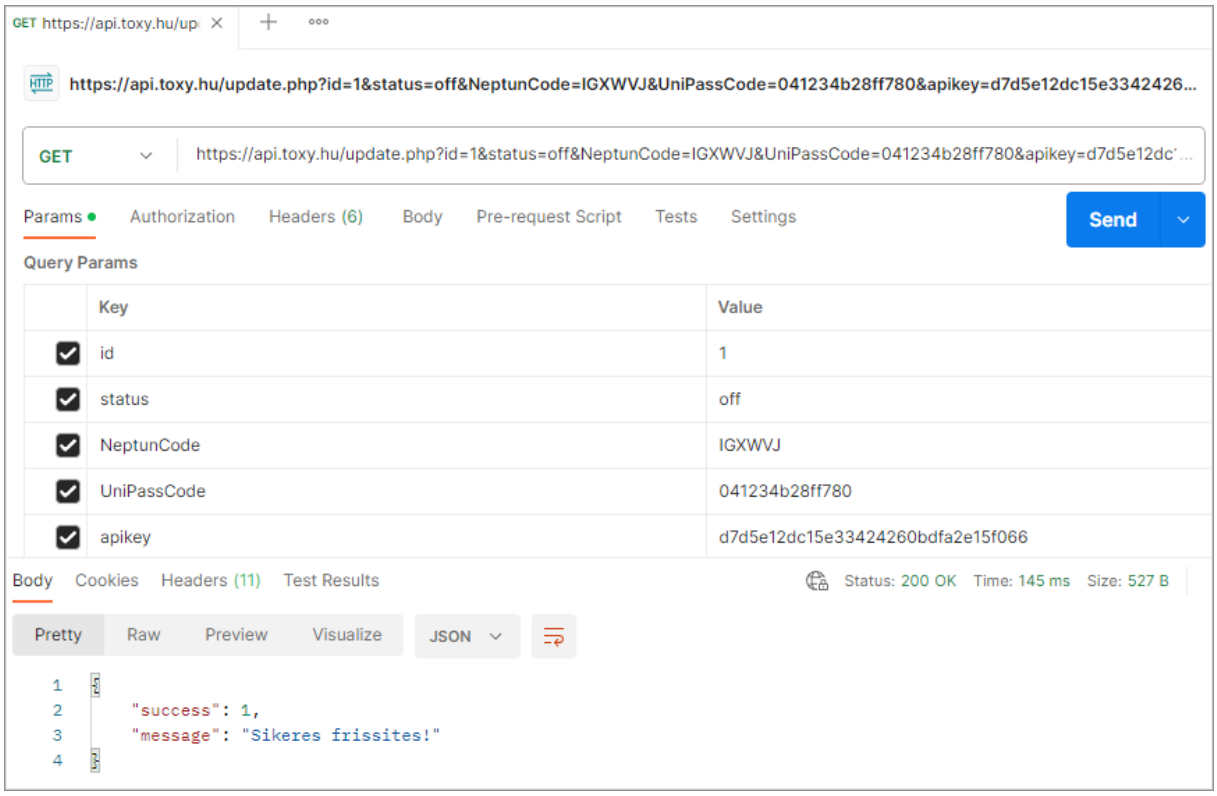
- Funkció: Ez a fájl a webalkalmazás által megjelenített szekrényeket jeleníti meg.
- Működés:
    - A weboldalon megjelenő szekrények: Az `{ID}` paraméterrel lehet megadni, hogy melyik oldalt szeretnénk nézni. Egy oldalon 16 szekrény található, és a hozzájuk tartozó, id, status, neptun kód, unipass kód. Az `{ID}` értékének növelésével lehet a további oldalakon lévő szekrényekhez eljutni.
    - API kulcs: Az `{APIKULCS}` paraméter a szekrényekhez való hozzáférést biztosítja, ami az adatbázisban található.

`update.php:`

https://api.toxy.hu/update.php?id={ID}&status={ON/OFF}&NeptunCode={NKÓD}&UniPassCode={UKÓD}&apikey={APIKULCS}

- Funkció: Ez a fájl lehetővé teszi a szekrények állapotának frissítését.
- Működés:
    - Id: Az `{ID}` paraméterrel azonosítja a szekrényt, amelynek az állapotát frissíteni szeretnéd.
    - Állapot (status): Az `{ON/OFF}` paraméterrel lehet beállítani a szekrény állapotát, ami vagy `"ON"` (nyitott) vagy `"OFF"` (zárt) lehet.
    - Neptun kód (NeptunCode), UniPass kód (UniPassCode): `{NKÓD}` és `{UKÓD}` paraméterek a felhasználó azonosítására szolgál.
    - API kulcs: Az `{APIKULCS}` paraméterrel azonosítom az API-hoz való hozzáférést.

Az API-t a Postman programmal ellenőriztem, teszteltem (*22. ábra*). A Postman egy speciális fejlesztői eszköz, amely lehetővé teszi HTTP kérések létrehozását, küldését és tesztelését. A Postman segítségével könnyedén létrehozhatók különböző típusú kérések, mint például: GET, POST, PUT vagy DELETE, és ellenőrizhetők a válaszok. Ezzel az eszközzel validálhatók az API visszakapott adatstruktúrái, és ellenőrizhetők a visszatérési értékek helyessége, ami segíthet a hibák gyors felderítésében és javításában.




*22.ábra: Postman*

## 4.4. Weboldal


*23.ábra: UniLocker logó*

A weboldal készítése során ügyeltem arra, hogy az megfeleljen a mai UI és UX elvárásoknak. A kinézetet SCSS-ben készítettem el. Az SCSS[14] (Sassy CSS) egy CSS kiterjesztés, amely lehetővé teszi a stíluslapok hatékonyabb és olvashatóbb módon történő létrehozását. A weboldalt reszponzív módon alakítottam ki, hogy mindenféle eszközön optimálisan jelenjen meg. Emellett különös figyelmet fordítottam az adatbiztonságra.

### 4.4.1. Design

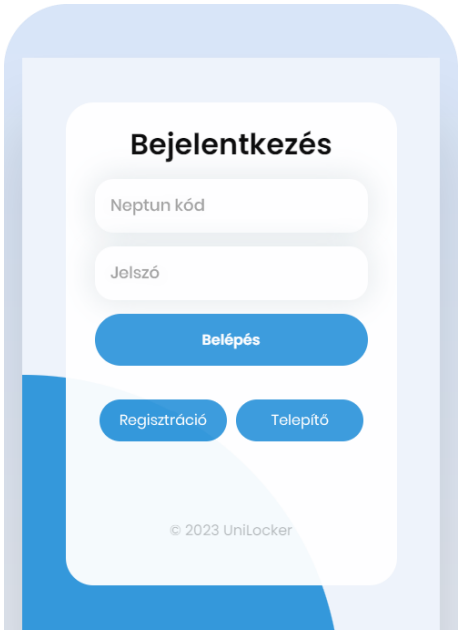
A logó tervezésénél azt tűztem ki célul, hogy az egyetem szellemiségét, a szekrényeket és a biztonságot harmonikusan ötvözzem. Az ikonikus kalapot, mely az egyetemek hagyományos szimbóluma, összekapcsoltam a hallgatók számára elérhető szekrényekkel, a kalapon lógva egy kulcslyuk látható, ezzel a biztonság érzetét közvetítve (*23. ábra*).

Egyik alapszínnek a kéket választottam, mely a modernitást sugallja. A fehér és szürke árnyalatai nyugalmat és az eleganciát. Betűtípusként a Poppins-t alkalmaztam, mely a letisztultságot erősíti meg. A weboldalon használt ikonokhoz a Boxicons[15] nyílt forráskódú ikonok ingyenes gyűjteményét használtam (*24. ábra*).


*24.ábra: Mockup képek*

[14] Bővebb információk az Irodalomjegyzékben találhatók. [6]
[15] Bővebb információk az Irodalomjegyzékben találhatók. [7]

### 4.4.2. Bejelentkezés




*25.ábra: Bejelentkezés*

Amikor a felhasználó megnyitja a weboldalt, ez a kezdőképernyő jelenik meg (*25. ábra*). Ezen a kezdőképernyőn az alábbi lehetőségek állnak rendelkezésre: a felhasználó dönthet, hogy belép a meglévő fiókjába, regisztrál egy új fiókot, vagy akár az alkalmazást is telepítheti.

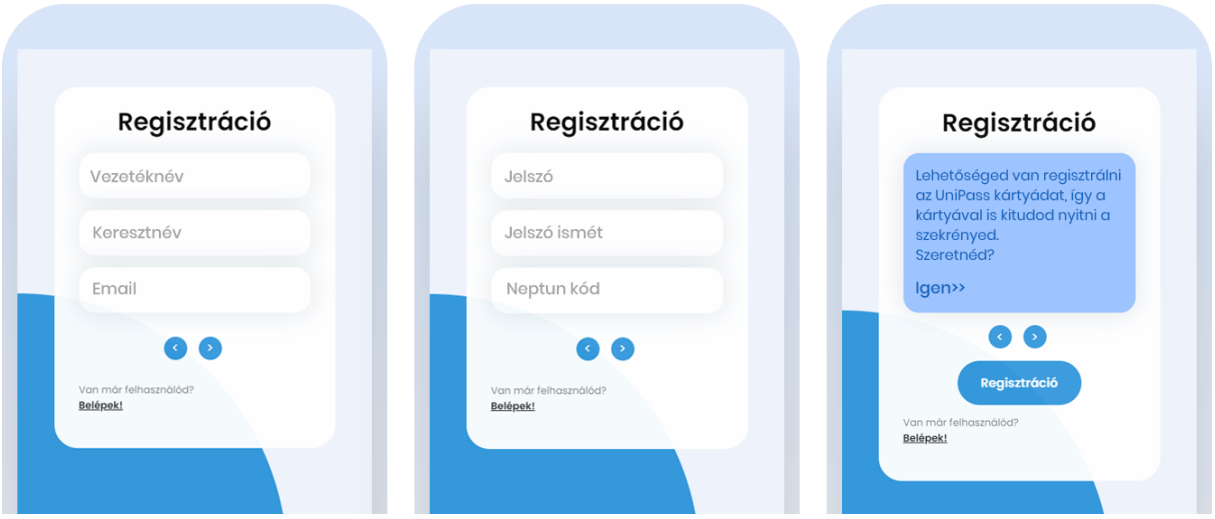
A bejelentkezési folyamat során a felhasználónak meg kell adnia a Neptun kódját és jelszavát. Az alkalmazás elvégzi a szükséges ellenőrzéseket, például: hogy minden kötelező mező ki van-e töltve, és hogy a Neptun kód érvényes hosszúságú-e (nem lehet több mint 6 karakter, és kevesebb sem). A felesleges szóközöket automatikusan eltávolítom, az alkalmazás nem tesz különbséget a Neptun kódban, kis- és nagybetűk között, mivel automatikusan nagybetűsre alakítja. A jelszavakat a belépési folyamata során a `password_verify` függvénnyel hasonlítom össze, a felhasználó által megadott jelszót, a tárolt, hash-elt jelszóval.

Ha bármelyik mező hibásan van kitöltve, vagy a felhasználó nem létezik az adatbázisban, ezt egy felugró üzenet (popup) közli a felhasználóval. Ezt a felugró üzenetet a Sweetalert2 nevű JavaScript könyvtár[16] segítségével hoztam létre, ami ingyenesen elérhető bárki számára.

Ha a bejelentkezés sikeres, a felhasználó adatait `Session`-ként tárolom el, így könnyedén továbbíthatók a későbbi lépésekhez, és az alkalmazás teljes körű funkcionalitásához. A `Session` segítségével a felhasználói állapot és az azonosítás biztonságosan kezelhető az alkalmazás futása során.

[16] Bővebb információk az Irodalomjegyzékben találhatók. [8]

### 4.4.3. Regisztráció



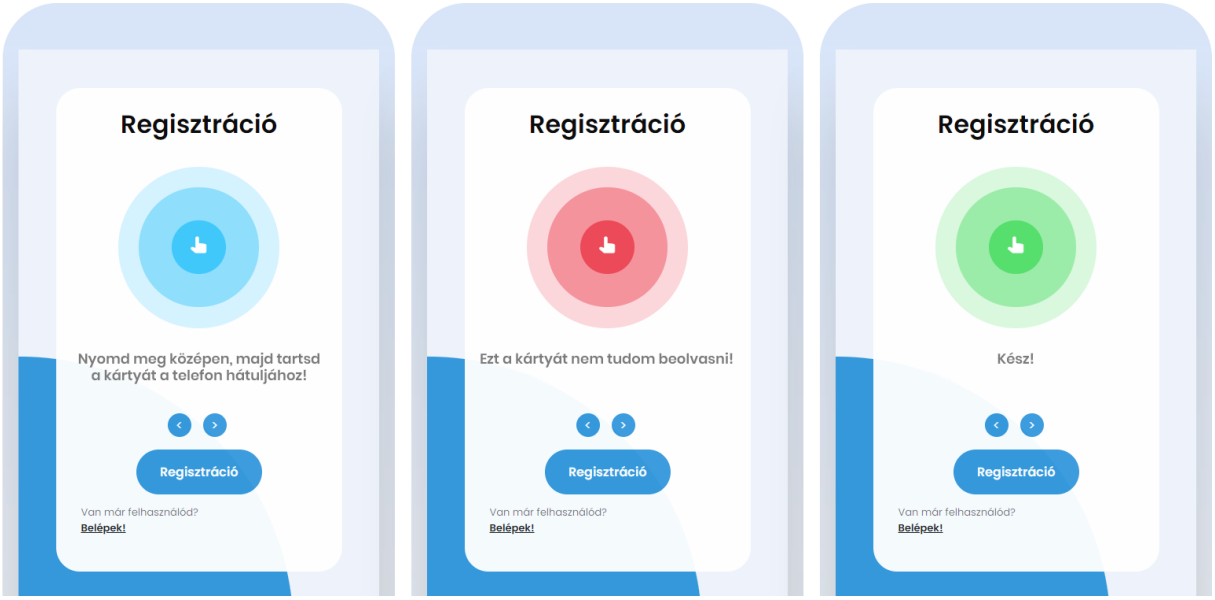

*26.ábra: Regisztráció*

A regisztrációs folyamatot három oldalra osztottam fel a felhasználók számára, hogy könnyebbé és átláthatóbbá tegyem (*26. ábra*). Az első oldalon a felhasználó nevét és e-mail címét kell megadni. A második oldalon a jelszót kell kétszer megadni, illetve a Neptun kódot. A harmadik oldalon az UniPass kártya beolvasására van lehetőség. Minden felhasználói adat egy `input_validation.php` fájlban kerül ellenőrzésre különböző függvények segítségével:

- Ellenőrzöm, hogy minden mező ki van-e töltve.
- Ellenőrzöm a vezeték- és keresztnév mezőket, hogy csak betűket tartalmaznak, és nem hosszabbak 30 karakternél.
- Ellenőrzöm az e-mail cím formátumát.
- A jelszónak minimum 8 karakter hosszúnak kell lennie, és mindkét jelszó mezőnek meg kell egyeznie.
- A Neptun kód 6 karakter hosszú lehet, és csak betűket és számokat tartalmazhat.
- Az UniPass kártya esetén is ellenőrzöm, hogy 14 karakter hosszú-e, és csak betűket és számokat tartalmazhat.
- Végül ellenőrzöm, hogy az e-mail címet, Neptun kódot és az UniPass kártyát már regisztrálták-e korábban.

Ezen ellenőrzéseket alkalmazva biztosítom, hogy a regisztrációs folyamat zökkenőmentesen zajlik, és az adatok helyesek és biztonságosak maradnak.

#### 4.4.3.1 NFC beolvasás




*27.ábra: UniPass kártya beolvasás*

Az UniPass kártya olvasását, a készülékekben található NFC (Near Field Communication) technológiával valósítottam meg. Az NFC egy vezeték nélküli kommunikációs technológia, amely lehetővé teszi az adatcserét rövid távolságon belül. Ennek segítségével az UniPass kártyák olvasása és az azokon található információk beolvasása gyors és hatékony módon történhet. A kód vagy technológia, amellyel megvalósítottam, a WEB NFC API[17]. Ez az API lehetőséget biztosít az adatcserére könnyű NFC Data Exchange Format (NDEF) üzeneteken keresztül. Az NDEF üzenetek egy szabványos formátumot követnek az NFC-eszközök közötti adatátvitelre, könnyű és strukturált módon.

Vannak készülékek, melyek viszont nem engedik ezt használni, emellett az NFC API csak adott böngészőverzióban működik, például: a Chrome 89 verzió[18] vagy annál újabb verziók.

A beolvasott UniPass kártya azonosítója egy input mezőben kerül eltárolásra, mint a többi adat. Azonban ez a mező nem látható a felhasználó számára, biztonsági okokból.

Amennyiben nem megfelelő kártyát érint a felhasználó a készülékhez, például: bankkártya, azt jelzem felé egy hibaüzenettel és a kék kör átvált pirosra. Ha az UniPass kártyát érinti a készülékhez, akkor átvált zöldre és kiírja, hogy „Kész!”. Nem kötelező megadni a regisztrációhoz, erre később is lesz lehetősége a profil beállításoknál. A „Regisztráció” gombot megnyomva az azonosító, valamint az egyéb releváns adatok eltárolásra kerülnek (*27. ábra*).

[17] Bővebb információk az Irodalomjegyzékben találhatók. [9]
[18] Bővebb információk az Irodalomjegyzékben találhatók. [10]

### 4.4.4. OWASP

Az adatok megfelelő és biztonságos kezelése ma már létfontosságú. Az Open Web Application Security Project[19] (OWASP) egy olyan online szervezet, amely szabadon hozzáférhető forrásokkal, például: cikkek, módszertanok, dokumentációk, eszközök és technológiákkal segíti azokat, akik a webalkalmazások biztonságának megőrzésére törekednek.

A megfelelő biztonsági lépések nélkül az alkalmazások sérülékenységekkel és támadásokkal szemben védtelenek lehetnek, ha nem alkalmaznak megfelelő biztonsági lépéseket. Sérülékenységek közé tartozik a Cross-Site Scripting (XSS), mely során a támadó próbálja bejuttatni a webalkalmazásba a káros kódot. Ez ellen különböző módszerek állnak rendelkezésre, egy ilyen módszer a „`htmlspecialchars`” függvény használata.

A `htmlspecialchars` függvény alapvetően az adatokat HTML speciális karaktereinek ártalmatlanítására szolgál. Például: nem lehet elküldeni a szervernek olyan kódot, amely potenciálisan veszélyes lehet, például: egy script. Ezáltal a támadó nem tud olyan kódot beilleszteni az alkalmazásba, amely veszélyeztetheti az oldalak integritását vagy a felhasználók biztonságát.

Az OWASP iránymutatásai között kiemelt figyelmet fordítanak az SQL injection támadások megelőzésére is. Ez egy olyan támadás, amelynek során a támadó megpróbálja befecskendezni a káros SQL parancsokat az alkalmazás adatbázisába. Ezt meglehet előzni olyan eszközökkel, mint például: a „`mysqli_real_escape_string`” függvény.

A `mysqli_real_escape_string` függvény segít az adatok ártalmatlanításában, és megakadályozza, hogy a támadó manipulálja vagy károsítsa az adatbázist. Ezáltal a webalkalmazás sokkal biztonságosabb lesz, és az adatbázisban tárolt érzékeny információk védelme is biztosított lesz.

A felhasználók által megadott jelszavak biztonságos tárolása is kritikus fontosságú. A jelszavakat `hash`-eltem, vagyis átalakítottam olyan formába, amelyet nem lehet könnyen visszafejteni. A `PASSWORD_DEFAULT` konstans segítségével a PHP automatikusan a legjobban elérhető `hash` algoritmust használja. A használt algoritmus általában erős és biztonságos, és biztosítja, hogy a jelszavak ne legyenek könnyen visszafejthetők még akkor sem, ha az adatbázisban tároltak feltörésre kerülnek.

Az OWASP és a fent említett gyakorlatok alkalmazása együttműködve biztosítja, hogy a webalkalmazások megfelelően védettek legyenek a potenciális támadásokkal szemben.

[19] Bővebb információk az Irodalomjegyzékben találhatók. [11]

### 4.4.5. PWA

A PWA (Progressive Web App) egy olyan webalkalmazás, amely ötvözi a hagyományos weboldalak és a natív mobilalkalmazások előnyeit. Az alábbi fájlokhoz van szükségük egy PWA létrehozásához:

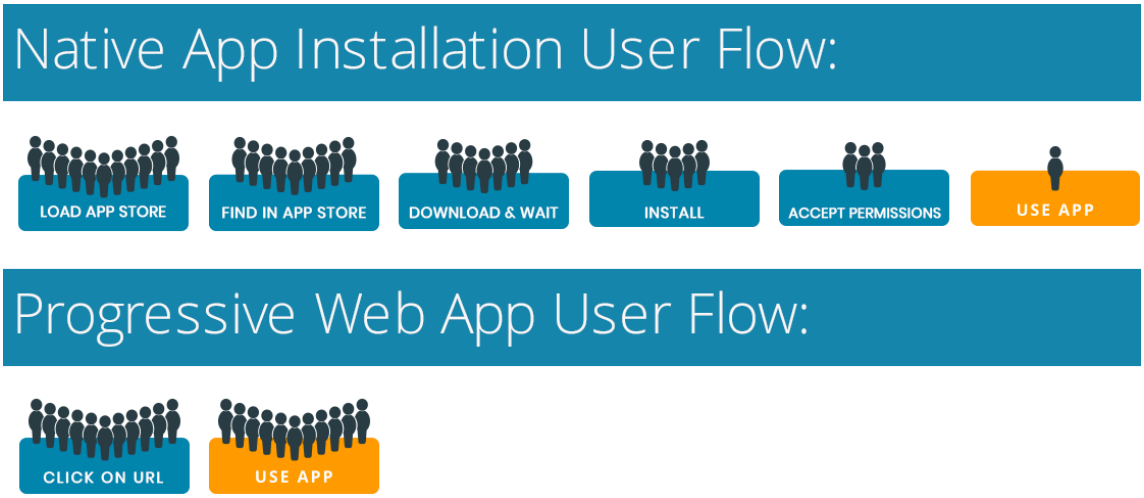
- index.html: Ez az alkalmazás fő HTML-fájlja, amely tartalmazza az alkalmazás felhasználói felületét.
- manifest.json: Ez a JSON-fájl tartalmazza az alkalmazás metaadatait, például: nevet, ikont, színeket, képernyőképeket és az alkalmazás indulási URL-jét. Az "`icons`" objektum segítségével definiálhatunk különböző méretű ikonokat az alkalmazáshoz, amik azért fontosak, mert az alkalmazásnak több platformon és eszközön kell jól kinéznie. A "`screenshots`" objektum lehetővé teszi, hogy bemutassuk, hogyan néz ki az alkalmazás, és megmutathassuk az alkalmazás fő funkcióit vagy vonzó dizájnját. A cél, a felhasználók figyelmének felkeltése. Ezek a képek segítenek a felhasználóknak eldönteni, hogy letöltik-e az alkalmazást vagy sem. (*28. ábra*)
- service-worker.js: A service-worker egy olyan JavaScript-fájl, amely a háttérben fut, és lehetővé teszi az alkalmazásoknak az offline működést, a gyorsítótárazást és az értesítéseket. A service-worker egy kulcsfontosságú elem a PWA működésében, mivel lehetővé teszi az alkalmazásnak, hogy a felhasználók számára gyors és zökkenőmentes élményt nyújtson, még rossz internetkapcsolat esetén is.


*28.ábra:Manifest screenshots*

A Progressive Web App (PWA) számos előnnyel rendelkezik a natív alkalmazásokhoz képest (*29. ábra*), néhány közülük:

- Telepítési folyamat egyszerűsége: Natív alkalmazások telepítésekor a felhasználónak el kell navigálnia az alkalmazás áruházába, megtalálnia az alkalmazást, letöltenie és telepítenie kell, ami időigényes lehet. A PWA-k esetében ezt a folyamatot egyszerűen megspórolhatja, mivel csak egy webhivatkozást kell megnyitnia a böngészőben.
- Nem igényel alkalmazás áruházat: A PWA-k használata nem függ alkalmazás áruházaktól, így nem kell elfogadnia a specifikus áruházak szabályait és előírásait.
- Alacsonyabb terhelés és tárhelyigény: Mivel a PWA-kat a felhasználó böngészőjében futtatják, nem foglalnak nagy területet a készülék tárhelyén.
- Gyors hozzáférés: Azonnal elérhetők, nincs hosszas idejű letöltés és telepítés.
- Frissítések könnyű elérése: A PWA-k frissítései egyszerűen érvénybe lépnek, mivel a legfrissebb változat mindig a web szerveren található. A felhasználónak nincs szüksége manuális frissítésre az alkalmazás áruházból.
- Platformfüggetlenség: Mivel böngészőn keresztül működnek, platformfüggetlenek.
- Kisebb fejlesztési és karbantartási költségek: A PWA-k fejlesztése és karbantartása általában olcsóbb lehet, mivel egyetlen kódbázist lehet használni több platformon, és nem szükséges minden platformhoz külön alkalmazást fejleszteni.

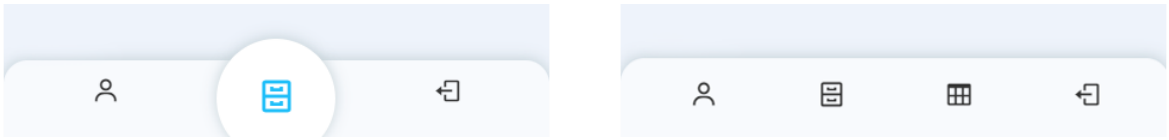
Ezek az előnyök azt mutatják, hogy a PWA-k versenyképes alternatívát nyújtanak a natív alkalmazásokkal szemben, különösen olyan esetekben, ahol a könnyű hozzáférés és a gyors telepítés fontos szempontok.




*29.ábra: PWA előnyei, natívval szemben* [20]

[20] Bővebb információk az Irodalomjegyzékben találhatók. [12]

### 4.4.6. Belépve



*30.ábra:Hallgató és Admin menü*

Amint a felhasználó sikeresen regisztrált és bejelentkezett, három menü közül választhat:

- balra „Profil” (4.4.6.3)
- középen a főmenü „Szekrények” (4.4.6.1 / 4.4.6.2)
- jobbra a „Kijelentkezés”. (4.4.6.4)
- + Admin felület (4.4.6.5)

Ha a felhasználó rendelkezik Admin ranggal, akkor az elérhető menükhöz hozzáadódik még egy opció, ami az „Admin felület”, és további beállításokhoz is képes hozzáférni. A menük segítségével könnyedén navigálhatunk az oldalak között (*30. ábra*). A reszponzív menühöz az alábbi GitHub könyvtárat [21] használtam alapként, ez került átalakításra.

Létrehoztam egy fájlt ami eldönti, hogy a felhasználó rendelkezik-e már szekrénnyel, amennyiben nem, a „Szekrények” oldalt fogja látni, ha igen, akkor a „Saját szekrények” oldalt. A felhasználó nem képes az URL átírásával sem áttérni más oldalra, mert ellenőrzöm, hogy van-e már szekrénye, azzal, hogy a Neptun kódja megtalálható-e a `lockers` nevű táblában.

#### 4.4.6.1 Szekrények

A felhasználók számára elérhető a szabad szekrények megjelenítése és foglalása valós időben. Ahhoz, hogy valós időben lehessen látni a szekrényeket, erre egy `lockers.php` nevű fájlt hoztam létre, amelynek segítségével lekérdezhetem a szabad szekrények számát és azt, hogy melyek foglaltak. AJAX segítségével egy függvényt hoztam létre, amely rendszeresen hívja meg ezt a fájlt, két másodpercenként, és frissíti az adott szekrények állapotát a weboldalon. Így a felhasználók mindig láthatják a szabad szekrényeket. Ha valaki lefoglal egy szekrényt, az azonnal látható lesz, mert az el fog tűnni a listából.

Azonban az előzőleg kiválasztott szekrények `checkbox` formájában jelennek meg, és itt jött elő egy kis probléma. Ha az oldal időközönként frissül, a kiválasztott `checkboxok` állapota is törlődik. Ezt a problémát egy függvénnyel oldottam meg. Ez a függvény kiválasztja azokat az elemeket, amelyeknek az osztálya `"save-cb-state"`, és visszaállítja a `checkboxok` állapotát az előzőleg mentett adatok alapján, vagy alapértelmezetten kikapcsolt állapotba helyezi őket.

[21] Bővebb információk az Irodalomjegyzékben találhatók. [13]

Egy eseménykezelő figyeli az állapotváltozásokat, és az aktuális állapotot elmenti a böngésző helyi tárhelyére (`localStorage`) az objektum neve alapján. Így a felhasználók beállításai hosszú távon megmaradnak, akkor is, ha bezárják a böngészőt. Azonban, erre nekem nem volt szükségem, így létrehoztam egy további függvényt, amely törli ezeket az adatokat, amikor az oldalra lép.

A szekrények száma attól függ, hogy az adatbázisban, a beállítások (`settings`) táblában mennyi a `NumberOfLockers` értéke. A foglalás követően a `file_get_contents` függvény segítségével a PHP kód egy http GET kérést hajt végre a megadott URL-re, ami jelen esetben a fentebb említett `api.toxy.hu/update.php?` A paraméterek a következők lesznek:

- `id = $_POST[$i]` (a szekrény azonosítója)
- `status = off` (kikapcsolt állapot)
- `NeptunCode = $neptunCode` (változóban eltárolt `$_SESSION['neptuncode']`)
- `UniPassCode = $UniPassCode` (változóban eltárolt `$_SESSION['UniPassCode']`)
- `apikey = getApiKey()` (függvény, ami visszaadja az API kulcsot)

Végül, ha a felhasználó lefoglalta a kiválasztott szekrényt, átirányítom őt a "Saját szekrény" oldalra, ahol kezelheti a foglalt szekrényét.

### 4.4.6.2 Saját szekrény

Miután a felhasználó sikeresen lefoglalta a számára jó szekrényt, lát egy felugró popup ablakot, hogy „Sikeres foglalás!”, és egy külön oldalra kerül átirányításra, amit próbáltam minél egyszerűbben és egyértelműbben megcsinálni, hogy mindenki számára érthető legyen.

Egy interaktív lakat ikon látható, amely reagál a felhasználó kattintására. Alapértelmezett szerint a lakat zárt állapotban jelenik meg, ami azt jelenti, hogy a szekrény zárva van, alatta a szekrény számát, illetve egy „Szekrény leadás” gombot látni (*31. ábra*).

Amikor a felhasználó rákattint a lakatra, történik néhány fontos dolog. Először is, az ikon megváltozik, hogy jelezze a szekrény nyitását. Ezen kívül, a felhasználó számára két figyelmeztető üzenet jelenik meg. Az első üzenet rögtön a kattintás után megjelenik, és figyelmezteti a felhasználót arra, hogy „A szekrényed nyitva!”, a második üzenet pedig a lakat zárás animációja után figyelmeztet, hogy „Ne felejtsd el bezárni!”

Az animált lakat ikon CodePenről származik[22], de annak kinézetét és működését át kellett alakítani a projekthez. Alapból nyitott állapotban jelenik meg és kattintásra nyílik – zárul.

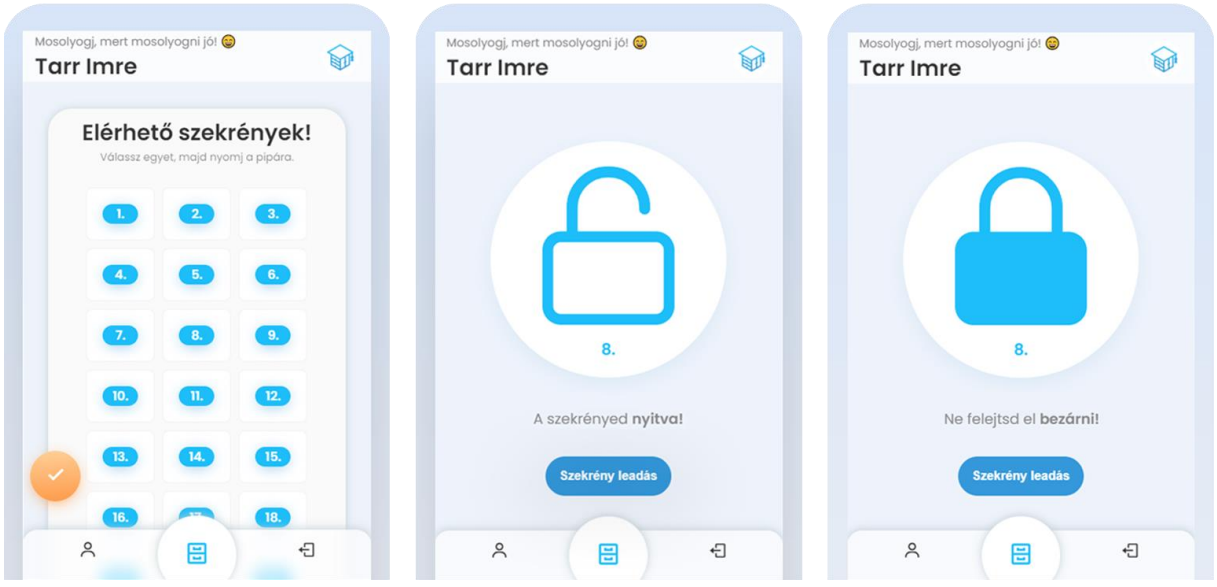
[22] Bővebb információk az Irodalomjegyzékben találhatók. [14]

Azonban nekem ez automatikusan kellett, hogy zárt állapotból kinyíljon, majd visszaváltozzon zártra. Az ikon és üzenetek megjelenítése minden eszközön és környezetben egyaránt biztosított, az eredeti kódban ez sem működött teljesen, így a felhasználók számára kényelmes és áttekinthető módon jelenik meg minden eszközön.

Amikor a felhasználó kinyitja a szekrényt, ez JavaScript segítségével történik, egy háttérben történő AJAX kérést is indítunk. Ehhez a `$.getJSON()` függvényt használjuk, amely az „`api.toxy.hu/update.php?`” címre irányul.

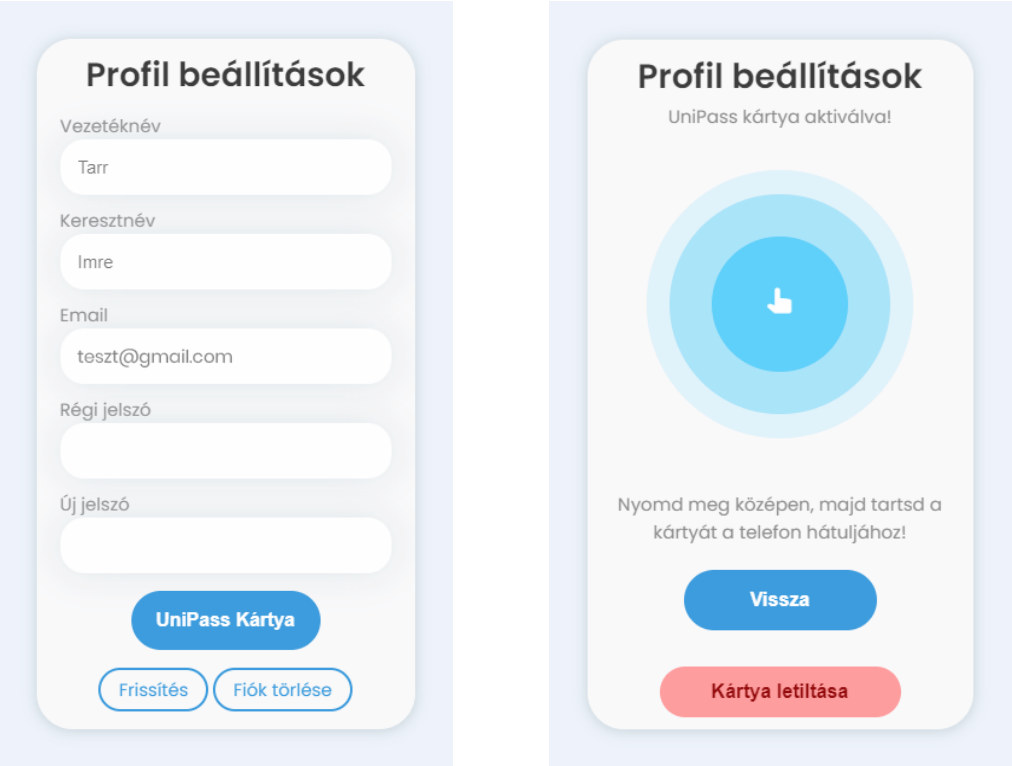
A kérés a szekrény státuszát „`on`”-ra állítja, amely azt jelenti, hogy a szekrény nyitva van. Ez segít a háttérszervernek követni a szekrény állapotát. Az időzített visszaállítás („`off`”-ra) gondoskodik róla, hogy a szekrény a felhasználói interakció után automatikusan ismét bezáródjon.

A „Szekrény leadás” gombbal egyszerűen és könnyedén leadhatjuk a szekrényt, ha már nem kell többé nekünk. A Neptun kód alapján lekérdezem a felhasználó által foglalt szekrény azonosítóját (`id`), majd amit már korábban is használtam, a php `file_get_contents` függvénnyel kérést hajtok végre az „`api.toxy.hu/update?`” URL-re. Az `id`-nek megadom a felhasználó által foglalt szekrény azonosítóját és minden paramétert egy üres értékkel töltök fel, majd az API kulcsot adom meg a végén. A felhasználó két másodpercig lát egy pop ablakot, hogy sikeresen leadta a szekrényt és visszairányítom a kezdőoldalra, és a többi felhasználó számára elérhetővé válik a szekrény.




*31.ábra: Foglalás folyamata*

#### 4.4.6.3 Profil




*32.ábra: Profil beállítások*

A felhasználóknak számos lehetőségük van a személyes fiókjaik kezelésére. Ezen a funkciók közé tartozik a nevük és e-mail címük módosítása, valamint a jelszóváltoztatás lehetősége. Ezek az opciók különösen fontosak lehetnek, amikor a felhasználók szeretnék frissíteni vagy változtatni az adataikat (*32. ábra*).

A név és az e-mail cím módosításához egyszerűen be kell lépni a fiókba, majd a megfelelő mezőbe megadni a frissítendő adatokat. Ezzel a móddal a felhasználók könnyedén aktualizálhatják a személyes adataikat, hogy mindig a legfrissebb információkat tükrözzék.

A jelszóváltoztatás lehetősége is rendelkezésre áll a felhasználók számára, ahol meg kell adni a jelenlegi (régi) jelszót, majd a kívánt új jelszót. Ez a fiók biztonságosságát szolgálja.

Az UniPass kártya is módosítható. Lehetővé téve a felhasználók számára, hogy az aktuális kártyájukat használják. Ha úgy frissítjük, hogy közben van foglalt szekrényünk, akkor sincs semmi probléma, a régi kártya helyett az új kártya azonosítja a felhasználót, így továbbra is hozzáférhet a szekrényhez. Ezen kívül, ha valaki elveszíti az UniPass kártyáját, és attól tart, hogy valaki visszaél vele, a kártyát le is lehet tiltani, hogy megvédje a személyes tárgyait.

A fiók törlése is lehetséges, amennyiben a felhasználók már nem kívánják többé használni az alkalmazást. Ebben az esetben minden fiókhoz kapcsolódó adat véglegesen törlődik, biztosítva a felhasználók személyes adatainak védelmét.

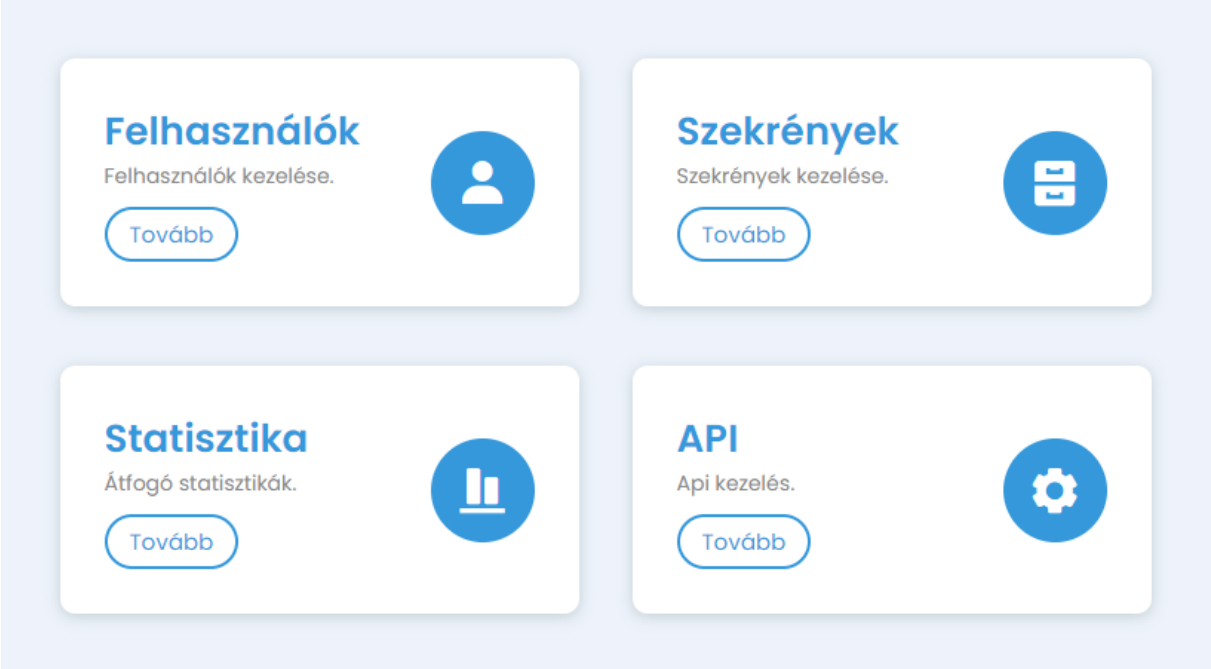
#### 4.4.6.4 Kijelentkezés

Amikor a felhasználó kijelentkezik, automatikusan visszairányítjuk őt a kezdőoldalra, hogy könnyen, új munkamenetet (`Session`-t) indíthasson. Ezen kívül minden felhasználói adatot sikeresen eltávolítunk a munkamenetből, így biztosítva a felhasználók személyes adatainak védelmét és a biztonságos kijelentkezést. Ezáltal a felhasználók teljesen kizárhatók a korábbi munkamenetből, és adataik nem maradnak a rendszerben.

#### 4.4.6.5 Admin felület

Az említett felület (*33. ábra*) a weboldal olyan része, amely csak azok számára érhető el, akik Adminisztrátori (`Admin`) ranggal rendelkeznek. Az Adminisztrátor rang azt jelzi, hogy az adott felhasználó az oldal egyik legmagasabb szintű jogosultságával rendelkezik, és így teljeskörű hozzáférése van az oldal adminisztrációs funkcióihoz és beállításaihoz. Az ilyen felhasználók a webhelyen belül kiváltságos helyzetben vannak, és képesek különféle fontos adminisztrációs feladatokat ellátni.

Az ellenőrzésük az úgy történik, hogy belépéskor lekérdezem az adatbázisból a felhasználó rangját, majd eltárolom egy munkamenetben, ha ez a `Session` megegyezik „`Admin`”-nal, akkor elérhetővé válik ez a menüpont, illetve a további oldalak. Ha valaki, aki nem rendelkezik ilyen jogosultsággal és megpróbál hozzáférni az oldalakhoz, azonnal átirányításra kerül, és a rendszer nem engedi be.




*33.ábra: Admin felület*

Az Adminisztrátorok az oldalon számos kulcsfontosságú tevékenységet végezhetnek el:

- Felhasználói fiókok kezelése
- Szekrények kezelése
- Statisztika
- API

**Felhasználói fiókok kezelése:**

Magába foglalja a meglévő fiókok módosítását és az esetleges problémák kezelését. Emellett az Adminisztrátorok hozzáférést kaphatnak a felhasználók személyes információihoz, amelyekre az oldal rendszerén belül szükség lehet.

A Neptun kód megadásával lekérhetjük az alábbi adatokat:

- Vezetéknév és keresztnév
- Email
- Rang
- Neptun kód
- UniPass kártya azonosító
- A felhasználó által foglalt szekrényszámát
- A fiók létrehozásának dátumát

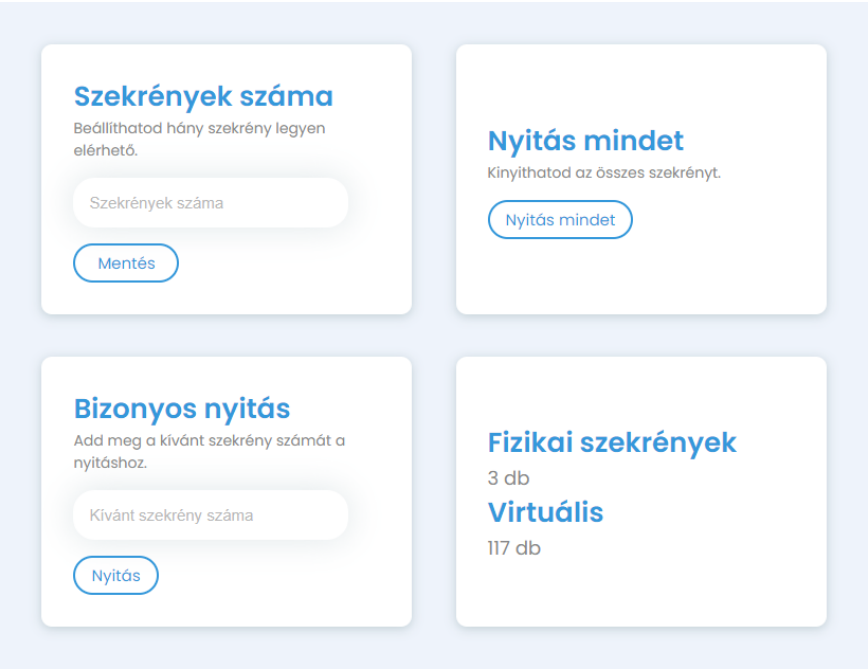
Ezekből az adatokból csak a teljes név, email cím és a rang módosítható. Próbáltam minél egyszerűbbre és letisztultra megcsinálni a felületet, így, amikor kilistázzuk a felhasználót, alul megjelenik egy kis szerkesztő piktogram, amire ha rákattintunk, azokat az adatokat módosíthatjuk, amik félkövér betűvel ki lesznek emelve, továbbá a szerkesztési mód elindításával további három ikon jelenik meg:

- Pipa ikon (Mentés): Ezzel a pipa ikonnal az Admin jóváhagyhatja a módosításokat, amelyeket az adatokban végzett. Ezzel megerősíti, hogy az új adatok érvénybe lépnek, és a rendszer ezeket elmenti.
- X ikon (Mégse): Az X ikon segítségével az Admin megszakíthatja a módosításokat és visszatérhet az eredeti adatokhoz. Ezzel a gombbal elveti az összes változtatást.
- Kuka ikon (Fiók törlése): Ha az Admin úgy dönt, hogy nem csak adatokat szerkeszt, hanem a teljes fiókját törölni szeretné az illetőnek, akkor a kuka ikonra kattintva törölheti a felhasználó fiókját. Ezt csak óvatosan és körültekintően kell használni, mivel ez visszavonhatatlanul eltávolítja a fiókot az rendszerből, erre az oldal rá is kérdez, hogy biztosan szeretné-e törölni.

**Szekrények kezelése:**

Ezen az oldalon található 4 darab panel egy rendkívül hasznos és sokoldalú eszköztár, amely lehetővé teszi az Adminisztrátorok számára a szekrényekkel kapcsolatos beállítások és műveletek elvégzését (*34. ábra*). Az alábbiakban kifejtem ezeket a lehetőségeket:

- Szekrények számának beállítása: Az első panel lehetővé teszi a felhasználók számára, hogy megadják, hány szekrény legyen elérhető az alkalmazásban, ez az érték nem lehet kisebb, mint a fizikai szekrények száma. Azonban fontos tudni, hogy ennek a beállításnak az elvégzésekor minden foglalt szekrény törlődik.
- Szekrények egyszerre történő nyitása: lehetővé teszi, hogy az összes szekrényt egyszerre kinyissuk. Ez a funkció gyors és hatékony lehet például: ha karbantartást szeretnénk végezni, vagy egyszerűen csak az összes szekrényt ki szeretnénk nyitni egy adott időpontban.
- Bizonyos szekrények kinyitása: A harmadik panelen keresztül lehetőségünk van kiválasztani, hogy mely szekrényeket szeretnénk kinyitni. Ehhez egyszerűen meg kell adnunk a szekrény számát és az alkalmazás kinyitja.
- Fizikai és virtuális szekrények száma: Az utolsó panelen találjuk a fizikai és virtuális szekrények számát. Fontos megjegyezni, hogy a fizikai szekrények számát csak az adatbázisban tudjuk szabályozni, mivel ez a valóságban létező szekrények számát jelenti. A virtuális szekrények száma pedig az összes szekrényből kivonva a fizikai szekrények számát adja meg, az alkalmazás virtuális szekrényeket is kezel.



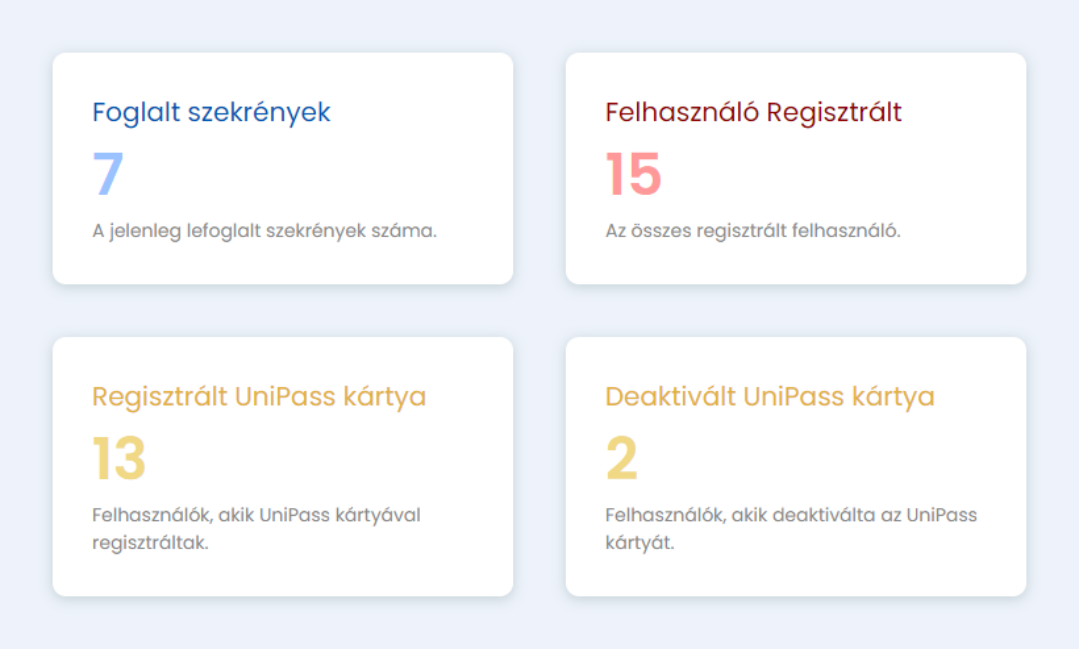

*34.ábra: Admin műveletek*

**Statisztika:**

Az adatbázisban található információk elemzése és statisztikák készítése kulcsfontosságú a rendszer hatékonyságának fejlesztésében és a felhasználói élmény javításában. Számos statisztikát hozhatunk létre az adatok alapján, ami segíthet abban, hogy a rendszerünk még hatékonyabb legyen. Néhány példa azokról, amiket létrehoztam (*35. ábra*):

- Jelenleg foglalt szekrények száma: Ezen statisztika segít nyomon követni, hogy mennyi szekrény van jelenleg használatban.
- Regisztrált felhasználók száma: A regisztrált felhasználók számának nyomon követése alapvető fontosságú az ügyfélbázis növekedésének figyelemmel kíséréséhez. Ez lehetővé teszi, hogy felmérjük a szolgáltatásunk népszerűségét és növekedését.
- UniPass kártyák regisztrálása: Az UniPass kártyák regisztrálásának száma jelzi, hány felhasználó rögzítette ezt az azonosító kártyát a rendszerben.
- Deaktivált UniPass kártyák: A letiltott, deaktivált UniPass kártyák száma jelenik meg.

Ezen kívül rengeteg féle statisztikát lehetne még létrehozni. Például: meghatározhatjuk, milyen gyakran használják a szekrényeket, átlagosan mennyi ideig, vagy mikor a legaktívabbak a felhasználók. Ehhez az adatbázison is kicsit módosítani kell, és további adatokat kell gyűjteni, például: az időbélyegzéseket és a szekrényhasználati adatokat. Az adatokat még grafikusabb formában is megjeleníthetjük, például: diagramok és grafikonok segítségével. Ezek a vizuális eszközök még érthetőbbé teszik az adatokat és segítenek az összefüggések gyorsabb felismerésében.




*35.ábra: Statisztikák*

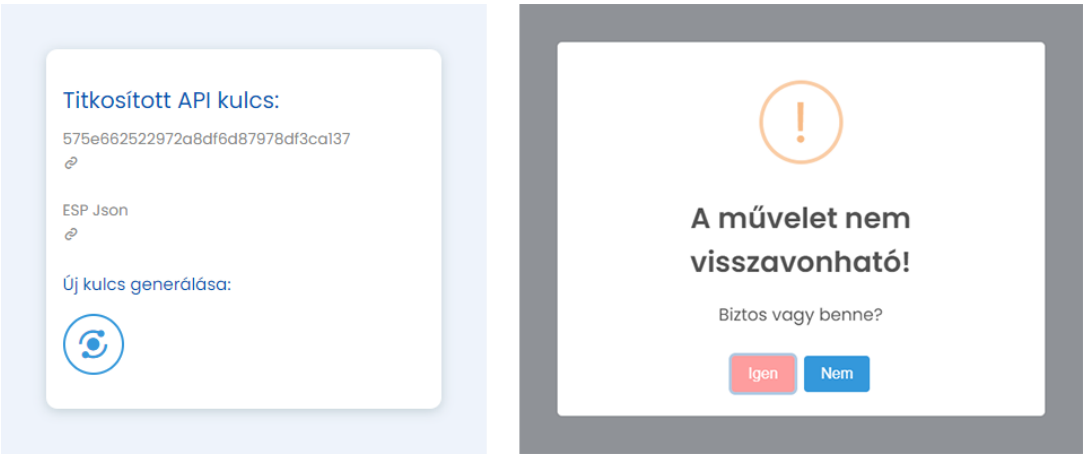
**API:**

Ez az oldal egy kritikus fontosságú felhasználói felületet biztosít az API-val kapcsolatos kulcsfontosságú műveletek végrehajtásához. Az itt található funkcionalitások között szerepel a JSON fájlok megtekintése, valamint az új API kulcs generálása, biztonsági okokból. A következőkben részletesebben kifejtem, hogyan működnek ezek a folyamatok:

- JSON fájlok megtekintése: Az oldal lehetővé teszi, hogy megtekinthessük az API által használt JSON fájlokat. Ehhez csak a kapocs ikonra kell kattintani. Ennek a funkciónak a segítségével az Adminisztrátorok láthatják és ellenőrizhetik az adatok szerkezetét és tartalmát, ami különösen hasznos lehet, ha hibakeresést végeznek vagy az API-val kapcsolatos működést próbálják megérteni.
- Új API kulcs generálása: Az API biztonságának érdekében az oldal lehetőséget kínál egy új API kulcs generálására. Az API kulcs egy 32 karakter hosszú karakterlánc, amely tartalmazhat számokat (0-9), ékezetek nélküli kis- és nagybetűket. Ezen karakterláncra alkalmazok egy MD5 titkosítást a biztonság növelése érdekében.

A kulcs generálásához a felhasználó a "Generálás" gombra kattint. Ekkor egy figyelmeztető üzenet jelenik meg, amely felhívja a figyelmet arra, hogy a művelet nem vonható vissza, és biztosan szeretnénk-e folytatni. Amennyiben a felhasználó úgy dönt, hogy generál egy új API kulcsot, akkor az oldal az alábbi paraméterekkel hívja meg az API generálást végző függvényt: `api.php?generate=true`. Amikor a `generate` értéke `true`, a függvény elvégzi az új kulcs generálását (*36. ábra*).

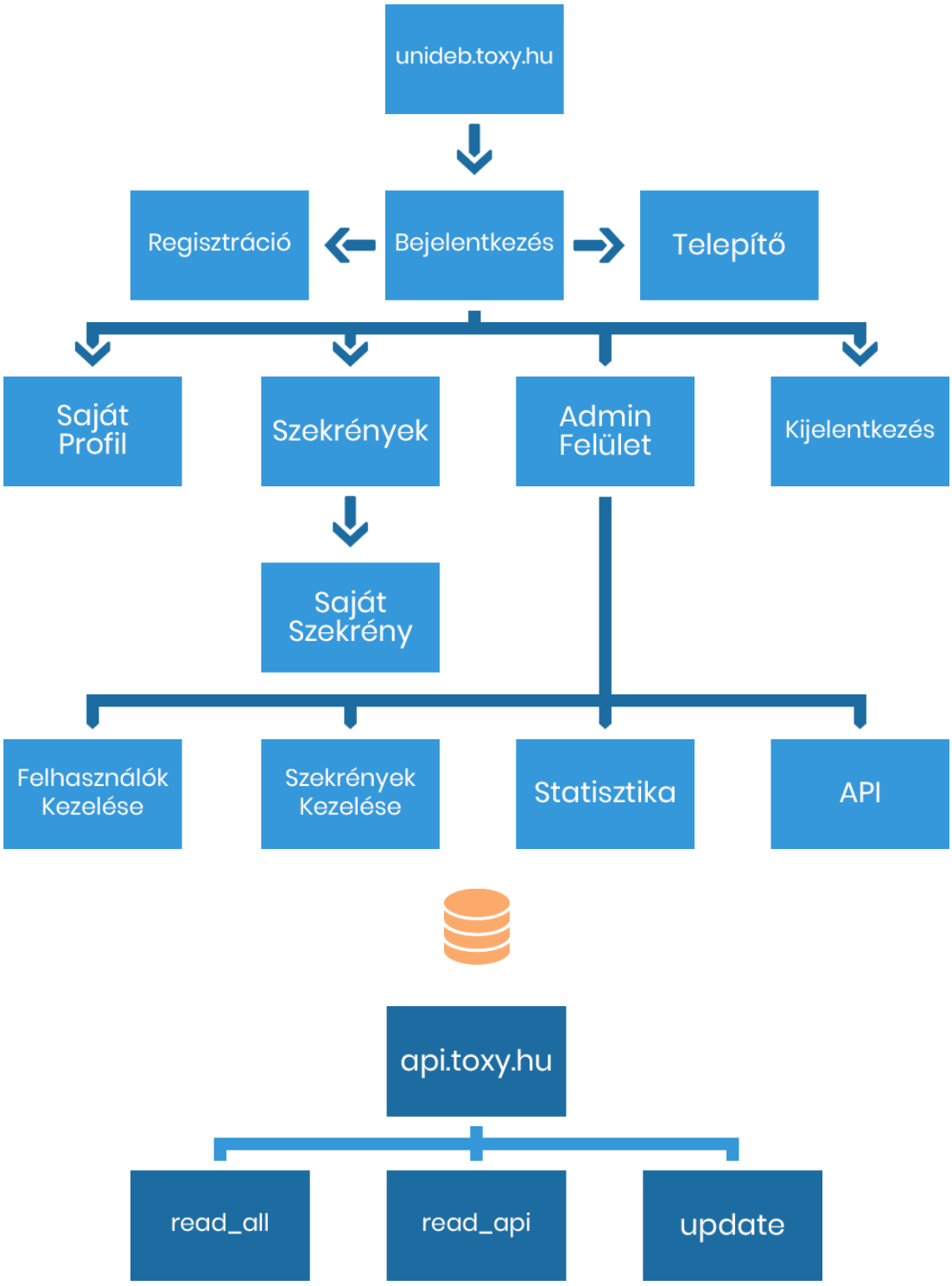
Fontos megjegyezni, hogy az új API kulcs generálása a meglévő szekrények működésére is hatással van. Amikor egy új kulcs generálódik, a szekrények is újra indulnak. Ez a biztonsági intézkedés annak érdekében történik, hogy minden szekrényhez hozzáférés csak az új kulcs segítségével lehessen.




*36.ábra: API*

## 4.5. Szoftver architektúrális ábra

Készítettem egy szoftver architektúrális ábrát (*37. ábra*), amely egy képen bemutatja az alkalmazás szerkezetét. Úgy vélem, hogy ez segít könnyebben megérteni a program felépítését.




*37.ábra: Szoftver architektúrális ábra*

# 5. Összegzés

A projekt fő célja az volt, hogy egy olyan innovatív platformot hozzak létre, mely lehetővé teszi a hallgatók számára, hogy egyszerűen és biztonságosan használják az egyetemi szekrényeket. Az alkalmazás fejlesztésének minden fázisában kiemelt figyelmet fordítottam a felhasználói felület és felhasználói élmény tervezésére és optimalizálására annak érdekében, hogy a végeredmény egy kiváló minőségű és teljes mértékben használható alkalmazás legyen. Az alkalmazás lehetőséget nyújt a felhasználóknak a regisztrációra és az azonosításra, ideértve az UniPass kártyák használatát is, melyekkel könnyedén nyithatják ki a szekrényeket.

Sikeresen megvalósítottam ezeket a célokat. A projekt során izgalmas kihívásokkal találtam szembe magam, rengeteg apró részlettel kellett foglalkoznom. A rendszer megtervezése és a megfelelő komponensek összeállítása számos új ismeretet hozott számomra. Emellett volt szerencsém profi szakemberekkel is megismerkedni és kapcsolatot kialakítani különböző területeken. Betekintést nyerhettem a 3D nyomtatás világába, tanácsokkal láttak el elektronikai oldalról és nagy segítőkészséget, kedvességet tapasztaltam attól a cégtől is, akik a matricák gyártásban voltak segítségemre.

A projekt során számos új területen is körülnéztem. Mélyen beleástam magam a mikroprocesszorok világába, ami egy lenyűgöző tapasztalat volt számomra. Emellett már tervezek egy új, személyes projektet, amelyre az UniLocker projekt során szerzett tapasztalataim nagyban inspiráltak.

Röviden összefoglalva, a projekt célkitűzésem egy könnyen kezelhető, átlátható felülettel rendelkező, rendkívül felhasználóbarát alkalmazás létrehozása volt. Emellett az alkalmazáshoz egy okos zárral felszerelt szekrényt is terveztem hardveres komponensként.

Az UniLocker projekt elősegíti az egyetemi hallgatók életét és az oktatási intézmények hatékonyabb működését. Az okos zár, mint fizikai megvalósítás, a modern technológia és az IoT alkalmazása révén hatékony megoldást nyújt a szekrények kezelésére, míg az alkalmazás egyszerűsíti és biztonságossá teszi a használatukat. A folyamatos fejlesztés és a digitalizáció hatására az oktatási intézmények és a hallgatók számára is ígéretes eredményeket hozhat.

A GitHub könyvtárat és a projektről készült videót a QR kód beolvasásával lehet elérni.

# 6. Irodalomjegyzék és hivatkozások


[1] ESP8266 paraméterei, *(utolsó látogatás: 2023.11.08)*:
https://shop.tavir.hu/termek/shop/arduino/expressif-esp32-esp8266/alaplap-expressif-esp32-esp8266/nodemcu-esp-12e-esp8266-v2-keskeny-modul-cp2102/

[2] MFRC522 paraméterei, *(utolsó látogatás: 2023.11.08)*:
https://shop.tavir.hu/wp-content/uploads/datasheet-mfrc522.pdf

[3] Relé paraméterei, *(utolsó látogatás: 2023.11.08)*:
https://shop.tavir.hu/termek/shop/modulok/rele/rele-modul-4-csatorna-optocsatolt-kek/

[4] CP2102 chip driver, *(utolsó látogatás: 2023.11.08)*:
https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads

[5] 3D nyomtatás alapanyagok, *(utolsó látogatás: 2023.11.08)*:
https://www.zaccord.com/blog?id=20

[6] Syntactically Awesome Style Sheets, *(utolsó látogatás: 2023.11.08)*:
https://sass-lang.com/guide/

[7] Boxicons, *(utolsó látogatás: 2023.11.08)*:
https://boxicons.com

[8] SweetAlert2, *(utolsó látogatás: 2023.11.08)*:
https://sweetalert2.github.io

[9] Web NFC Sample, *(utolsó látogatás: 2023.11.08)*:
https://googlechrome.github.io/samples/web-nfc/

[10] Web NFC API, dokumentáció *(utolsó látogatás: 2023.11.08)*:
https://developer.mozilla.org/en-US/docs/Web/API/Web_NFC_API

[11] OWASP, *(utolsó látogatás: 2023.11.08)*:
https://owasp.org

[12] PWA user flow, *(utolsó látogatás: 2023.11.08)*:
https://www.urphoneguy.com/blog-summary/2017/8/16/progressive-web-apps-the-next-mobile-experience

[13] Reszponzív menü, *(utolsó látogatás: 2023.11.08)*:
https://github.com/bedimcode/responsive-bottom-navigation

[14] Lock animation, *(utolsó látogatás: 2023.11.08)*:
https://codepen.io/HamzaHerbou/pen/NWvPbXq