

Fejlesztői dokumentáció

Vidics Márk

T1YAAB

Mérnökinformatikus BSc

Tartalomjegyzék

1. A projekt áttekintése	2
1.1. Funkcionális követelmények	2
1.2. Nem funkcionális követelmények	3
2. Tervezési fázis	3
2.1. Hardveres tervek	3
2.2. Szoftveres tervek	3
3. Hardver specifikáció	4
3.1. Raspberry Pi	4
3.2. RFID olvasó	5
3.3. LCD kijelző	7
3.4. Próbapanel	9
4. Szoftver specifikáció	10
4.1. Raspberry Pi OS	10
4.1.1. Docker	10
5. A fejlesztett kód részletezése	10
5.1. Adatbázis szerkezet	10
5.2. Verziókövető rendszer	12
5.3. Konfiguráció	13
5.4. Indító szkript	13
5.5. Adatbáziskapcsolat	14
5.6. SPI kommunikáció	15
5.7. I ² C kommunikáció	15
5.8. Email küldés	15
5.9. HTTP hívások	16
5.9.1. Az implementált végpontok részletezése:	16

Táblázatok jegyzéke

1. Az RFID olvasó bekötési terve	7
--	---

2.	Az LCD kijelző felhasznált lábai	8
3.	Az LCD kijelző bekötési terve	8
4.	Az felhasználókkal kapcsolatos alapinformációk tárolása	11
5.	Az RFID kulccsal kapcsolatos alapinformációk tárolása	11
6.	Egy felhasználó összekapcsolása több RFID kulccsal (egy-a-többhöz)	12
7.	Egy felhasználóhoz tartozó RFID kulcs belépési idejének tárolása	12
8.	A felhasználók karbantartásához tartozó végpontok	16
9.	A RFID karbantartásához tartozó végpontok	17
10.	A felhasználókhoz rendelt RFID kulcsok végpontjai	18
11.	A belépések karbantartásához tartozó végpontok	19

1. A projekt áttekintése

1.1. Funkcionális követelmények

- A projekt célja egy olyan bemutató berendezés elkészítése, amely segítségével képesek vagyunk belépési eseményeket megjeleníteni, illetve karbantartani RFID kulcsok segítségével.
- Szükségünk van egy kártyaolvasó eszközre, amely képes a felhasználóknak kiosztott kártyák/kulcsok értékét beolvasni.
- Képesnek kell lennünk ezeket a kártyákat felhasználókhoz társítanunk.
- A rendszernek meg kell tudnia jeleníteni egy felhasználó belépése során a belépés állapotát, amik a következők lehetnek: engedélyezett, tiltott és nem ismert kártya.
- A rendszernek rendelkeznie kell egy szolgáltatással, aminek segítségével valamilyen hálózati csatornán keresztül (pl.: HTTP kérések által) lekérdezhetőek és karbantarthatóak az RFID kulcsok, a felhasználók és a belépések.
- Gondoskodnunk kell arról, hogy ha egy felhasználó több alkalommal tiltott kártyával lép be, akkor üzenetet küldjünk a szolgáltatást üzemeltető rendszergazdának.

1.2. Nem funkcionális követelmények

- Egy Raspberry Pi 4 Model B eszköz még a projekt tervezése előtt beszerzésre került, ezért ez nem jelent külön kiadást, így a projektet ezzel kell megvalósítanunk, mert elégséges funkciókkal rendelkezik mind hardveres, mint szoftveres szinten.
- Egy bemutató eszköz elkészüléséhez az elektronikai elemeket tekintve maximum 10.000 forint összegű keretet szabunk meg. Ez egyben azt is meghatározza, hogy milyen hardverek jöhetnek szóba.

2. Tervezési fázis

2.1. Hardveres tervek

- A hardver elemek kiválasztása során figyelembe kell vennem, hogy a nem funkcionális követelményként meghatározott Raspberry Pi-hoz alkalmas eszközöket válasszak. Így ez esetben a GPIO lábkiosztása fogja nekünk megszabni azt, hogy pontosan milyen kommunikációs csatornán tudunk adatokat küldeni és fogadni.
- Szükség esetén az adott integrált áramköri panelek (pl.: RFID olvasó) lábait a csatlakozási pontokhoz kell forrasztani, ezért a forrasztóállomás beszerzéséről és egyéb kellékekről is gondoskodni kell.
- A bemutató eszközt próba panelen célszerű elkészíteni, mert az adott bekötés könnyen módosítható, illetve nem kell forrasztási műveleteket sem végezni ehhez.

2.2. Szoftveres tervek





















- A Raspberry Pi 4 Model B eszköz (lényegében egy ARM architektúrájú kis számítógépről van szó) rendelkezik a gyártó által fejlesztett operációs rendszerrel, ezért ennek használata tűnt a legcélszerűbbnek, mert könnyen kezelhető és széleskörű csomag támogatással rendelkezik.
- Követelményként szükségünk van egy adatokat szolgáltató alkalmazást fejleszteni, és mivel egy bemutató alkalmazásról van szó, ezért az egyszerűség kedvéért Python nyelven készül el, különböző `pip` csomagok használatával.

- Az adattároláshoz szükséges adatbázis kezelő kiválasztásánál is az egyszerűséget és könnyű kezelhetőséget érdemes alapul venni, ami esetén a MySQL jó választásnak bizonyul.

3. Hardver specifikáció

3.1. Raspberry Pi

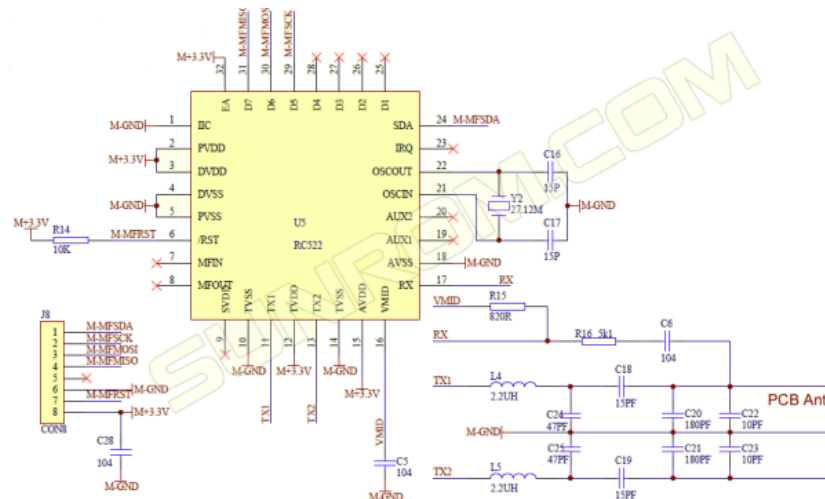
Az eszköz rendelkezik egy 40 csatlakozási pontos GPIO interfésszel. Ennek van egy belső számozása, amit a Raspberry Pi rendszere tud kezelni, illetve egy 1-40-ig tartozó számsorozat, ami a teljes lábkiosztásra vonatkozik. A következő ábrán a *GPIO#* jelölés a belső számozást, a csatlakozási pontok melletti számok pedig a teljes lábkiosztást adják meg. Az LCD kijelző és az RFID olvasó a teljes lábkiosztás szerint van bekötve.

Raspberry Pi 4 Model B (J8 Header)					
GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1		2	5.0 VDC Power
8	GPIO 8 SDA1 (I2C)	3		4	5.0 VDC Power
9	GPIO 9 SCL1 (I2C)	5		6	Ground
7	GPIO 7 GPCLK0	7		8	GPIO 15 TxD (UART) 15
	Ground	9		10	GPIO 16 RxD (UART) 16
0	GPIO 0	11		12	GPIO 1 PCM_CLK/PWM0 1
2	GPIO 2	13		14	Ground
3	GPIO 3	15		16	GPIO 4 4
	3.3 VDC Power	17		18	GPIO 5 5
12	GPIO 12 MOSI (SPI)	19		20	Ground
13	GPIO 13 MISO (SPI)	21		22	GPIO 6 6
14	GPIO 14 SCLK (SPI)	23		24	GPIO 10 CE0 (SPI) 10
	Ground	25		26	GPIO 11 CE1 (SPI) 11
30	SDA0 (I2C ID EEPROM)	27		28	SCL0 (I2C ID EEPROM) 31
21	GPIO 21 GPCLK1	29		30	Ground
22	GPIO 22 GPCLK2	31		32	GPIO 26 PWM0 26
23	GPIO 23 PWM1	33		34	Ground
24	GPIO 24 PCM_FS/PWM1	35		36	GPIO 27 27
25	GPIO 25	37		38	GPIO 28 PCM_DIN 28
	Ground	39		40	GPIO 29 PCM_DOUT 29
<p>Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.</p> <p>http://www.pi4j.com</p>					

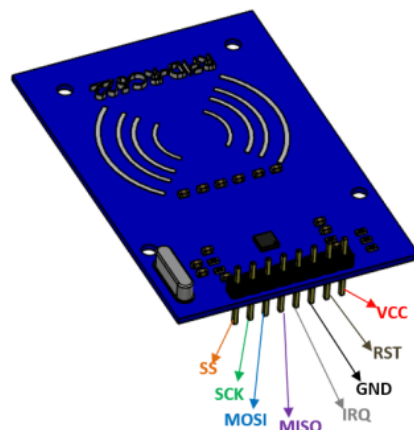
1. ábra. Raspberry Pi 4 Model B lábkiosztása

3.2. RFID olvasó

A felhasznált RFID olvasó modellje: RC522-MFRC. A kommunikáció az olvasóval SPI interfészen keresztül történik, melynek értékét ütemezett intervallumokban lekérdezzük és a táblázatban részletezett módon kerül bekötésre.



2. ábra. Az RFID olvasó kapcsolási rajza



Pin Number	Pin Name	Description
1	VCC	Used to Power the module, typically 3.3V is used
2	RST	Reset pin – used to reset or power down the module
3	Ground	Connected to Ground of system
4	IRQ	Interrupt pin – used to wake up the module when a device comes into range
5	MISO/SCL/Tx	MISO pin when used for SPI communication, acts as SCL for I2c and Tx for UART.
6	MOSI	Master out slave in pin for SPI communication
7	SCK	Serial Clock pin – used to provide clock source
8	MOSI	Acts as Serial input (SS) for SPI communication, SDA for IIC and Rx during UART

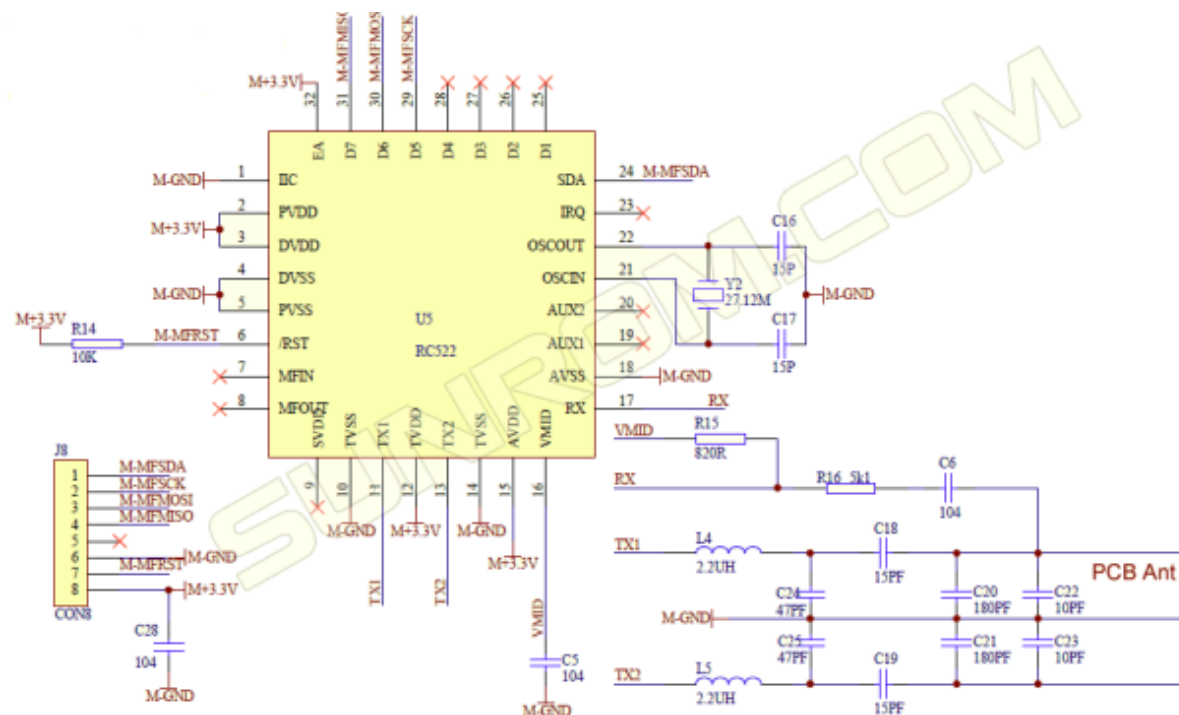
3. ábra. Az RFID olvasó lábkiosztása

RC522-MFRC olvasó bekötési terv	
Raspberry	RFID pin
GPI0 pin	neve
száma	
17	VCC
22	RST
20	GND
21	MISO
19	MOSI
23	SCK
24	SDA (MOSI)

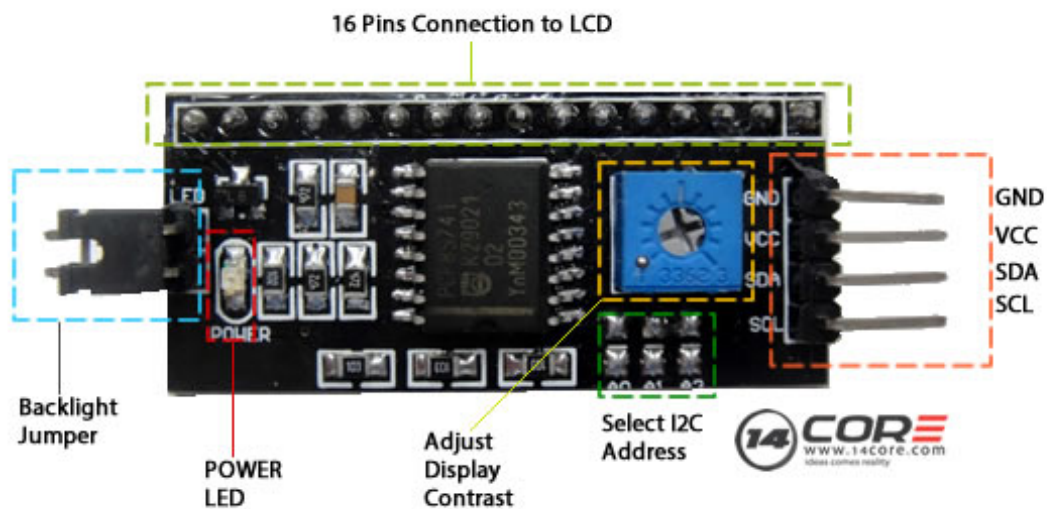
1. táblázat. Az RFID olvasó bekötési terve

3.3. LCD kijelző

A felhasznált LCD kijelző modellje: KC-1602-BB-I2C. A kommunikáció I²C csomagkapcsolt soros buszon keresztül történik és a táblázatban részletezett módon kerül bekötésre.



4. ábra. Az LCD olvasó kapcsolási rajza



5. ábra. Az LCD kijelző kivezetési pontjai

KC-1602-BB-I2C lábkiosztás	
Pin	Funkció
GND	Földelési pont
VCC	5V tápellátás
SDA	Soros adatvonal
SCL	Soros órajel

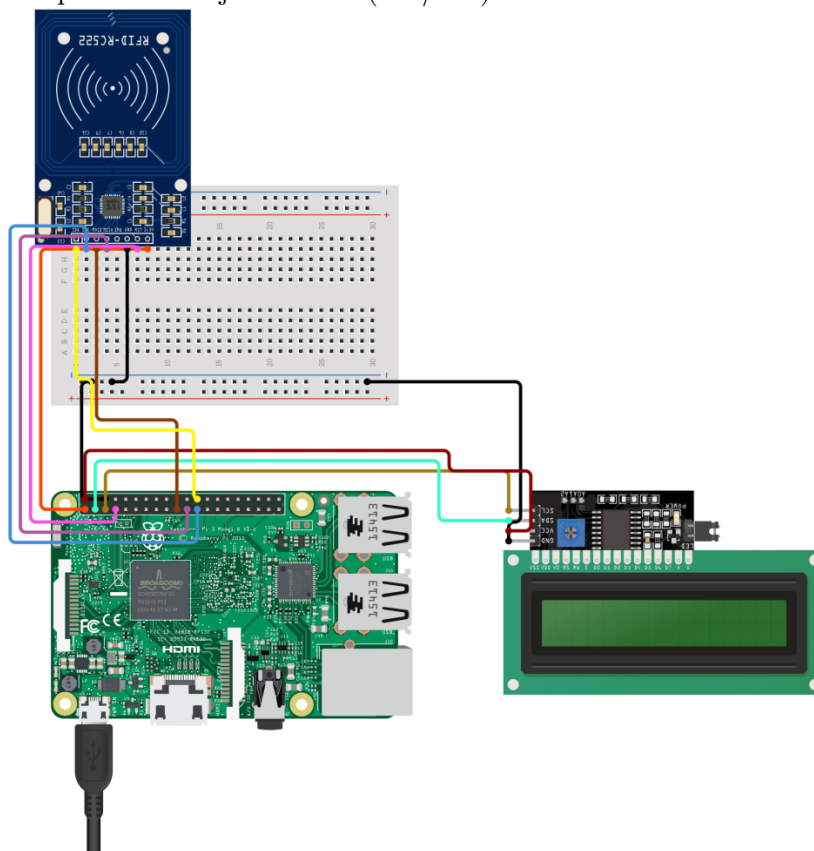
2. táblázat. Az LCD kijelző felhasznált lábai

KC-1602-BB-I2C bekötési terv	
Raspberry	
GPI0 pin száma	RFID pin neve
9	GND
2	VCC
3	SDA
5	SCL

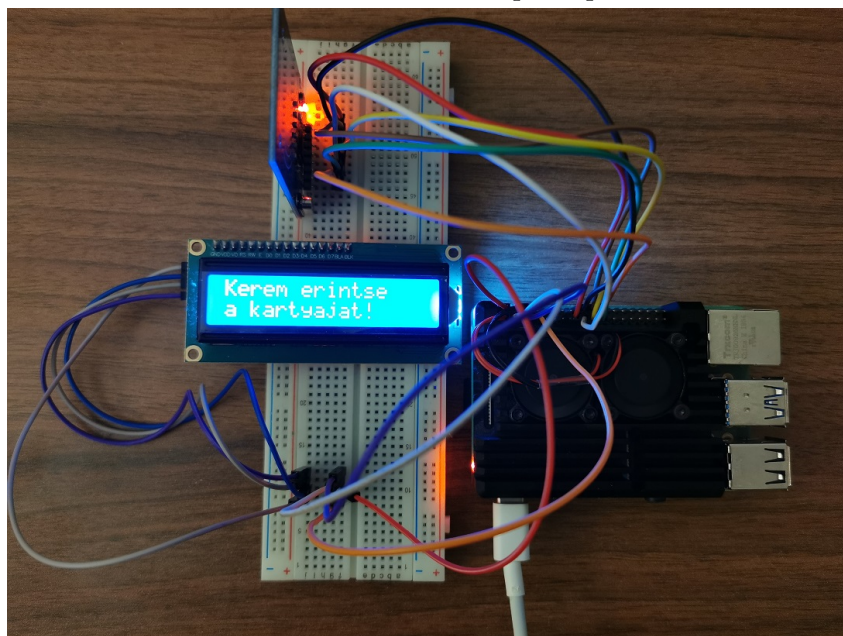
3. táblázat. Az LCD kijelző bekötési terve

3.4. Próbapanel

A felhasznált próbapanel modellje: BB-102 (630/200)



6. ábra. A hardveres terv próbapanelen



7. ábra. A projekt tényleges megvalósítása üzemkész állapotban

4. Szoftver specifikáció

4.1. Raspberry Pi OS

A Raspberry Pi OS egy Debian alapú operációs rendszer, ami kifejezetten a Raspberry Pi hardverekhez lett létrehozva, így ez már tartalmazza a szükséges rendszerszintű függvénykönyvtárakat ahhoz, hogy a hardveres kommunikációt meg tudjuk valósítani. A gyártó által készített függvénykönyvtárakat számos közösségi és egyéni fejlesztő által létrehozott csomag felhasználja, aminek segítségével képesek vagyunk absztraktabb szinten is kommunikálni ezekkel a hardveres eszközökkel, például Python nyelven. A következő operáció rendszer van telepítve:

- Kiadási dátum: 2023 október 10.
- Rendszer: 64-bit
- Kernel verzió: 6.1
- Debian verzió: 12 (bookworm)

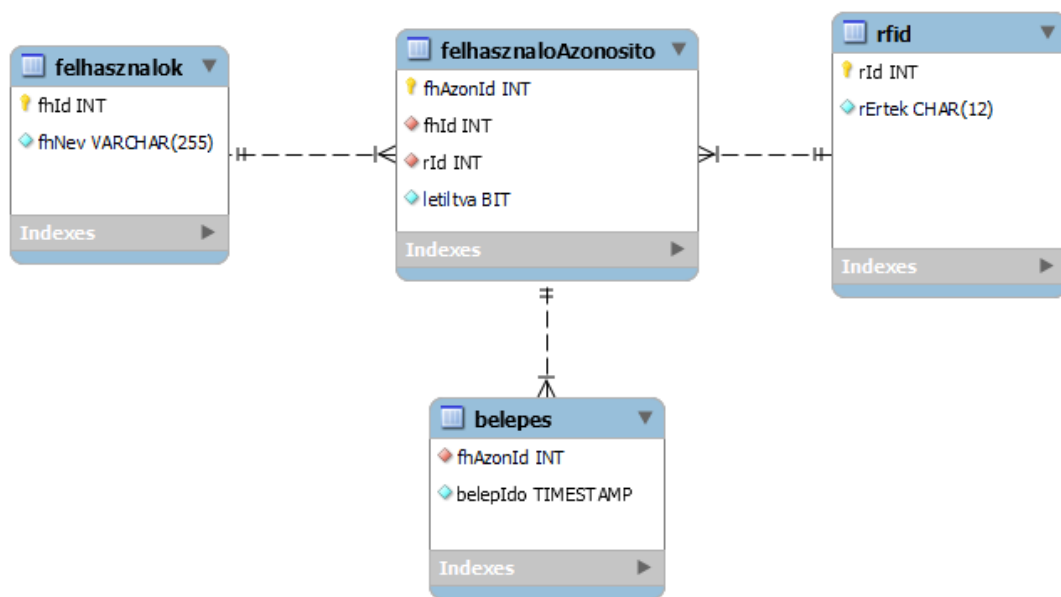
4.1.1. Docker

A Docker egy konténeres virtualizációs platform, lehetővé teszi a felhasználóknak, hogy szoftveralkalmazásokat izolált környezetben futtassák. A konténereket könnyedén és gyorsan el lehet indítani, amik általában egy *Dockerfile* alapján vannak definiálva. A konténerben futott szoftver alapját az úgy nevezett *base image* adja meg. Ebben az esetben ez a megoldás a MySQL adatbázis kezelő szoftverhez van használva, mert így el lehet kerülni a telepítési nehézségeket, valamint a konténert bármikor le tudjuk állítani vagy el tudjuk indítani. A konténer indítását a jelenlegi projekt esetén a szolgáltatáshoz tartozó indító szkript rendszerindításkor megteszi.

5. A fejlesztett kód részletezése

5.1. Adatbázis szerkezet

A szolgáltatás mögött egy MySQL adatbázis fut a korábban említett Docker konténerben, ami a belépési idők rögzítését, illetve az adott felhasználóhoz tartozó belépési idők lekérdezését hívatott megvalósítani. Az adatbázis szerkezete a következő ábrán és táblázatokon van részletezve:



8. ábra. Egyed-kapcsolat modell

Felhasználók			
Oszlop név	Típus	Tulajdonságok	Funkció
fhId	INT	UNIQUE NOT NULL AUTO_INCREMENT	A felhasználó azonosítója
fhNev	VARCHAR(255)	NOT NULL	A felhasználó neve

4. táblázat. Az felhasználókkal kapcsolatos alapinformációk tárolása

Rfid			
Oszlop név	Típus	Tulajdonságok	Funkció
rId	INT	UNIQUE NOT NULL AUTO_INCREMENT	Az RFID azonosítója
rErtek	CHAR(12)	UNIQUE NOT NULL	Az RFID kulcs értéke

5. táblázat. Az RFID kulccsal kapcsolatos alapinformációk tárolása

FelhasznaloAzonosito			
Oszlop név	Típus	Tulajdonságok	Funkció
fhAzonId	INT	UNIQUE NOT NULL AUTO_INCREMENT	Egy rekord azonosítója
fhId	INT	NOT NULL	A felhasznalok tábla fhId oszlopa
rId	CHAR(12)	UNIQUE NOT NULL	Az RFID tábla rId oszlopa
letiltva	BIT	NOT NULL DEFAULT 0	A felhasználóhoz rendelt RFID azonosító tiltásának állapota

6. táblázat. Egy felhasználó összekapcsolása több RFID kulccsal (egy-a-többhöz)

Belepes			
Oszlop név	Típus	Tulajdonságok	Funkció
fhAzonId	INT	NOT NULL	A felhasznaloAzonosito fhAzonId oszlopa
belepIdo	TIMESTAMP	NOT NULL DEFAULT CURRENT_TIMESTAMP()	A belépés ideje
rId	CHAR(12)	UNIQUE NOT NULL	Az RFID tábla rId oszlopa
letiltva	BIT	NOT NULL DEFAULT 0	A felhasználóhoz rendelt RFID azonosító tiltásának állapota

7. táblázat. Egy felhasználóhoz tartozó RFID kulcs belépési idejének tárolása

5.2. Verziókövető rendszer

A projekt követelményeknek megfelelően a kód GitHubon van tárolva, ami [ezen a linken elérhető](#).

5.3. Konfiguráció

Az alkalmazásban használt beállításokat egy [config.ini](#) fájlból olvassuk ki. A Pythonban képesek vagyunk a `configparser` csomag segítségével egy előre megadott struktúrában definiálni kulcs-érték párokat. Ezeket az értékeket kétféle módon használjuk fel: egy az egyben olvassuk ki, vagy érzékeny adatok esetén mutathat egy környezeti változóra is, például ilyen az adatbázis jelszó. Így meggátolhatjuk azt, hogy ezek az adatok a távoli verziókövető rendszerbe legyenek feltöltve, valamint elérjük vele azt is, hogy kizárólag az adott környezetben beállított változókkal tudjanak működni.

A felhasználásuk a következő módon történik:

- `[mysql]` alatt:
 - Host: A MySQL adatbázis IP címe
 - User: A MySQL adatbázishoz tartozó felhasználó
 - Password: A MySQL adatbázishoz tartozó jelszó
 - Database: A MySQL adatbázishoz neve
- `[smtp]` alatt:
 - Host: A SMTP szerver címe
 - Port: A SMTP szerverhez tartozó port
 - User: A SMTP szerverhez szükséges felhasználó
 - Password: A SMTP szerverhez szükséges jelszó
 - FromAddress: Az emailt küldő címe
 - ToAddress: Az emailt fogadó címe
 - DebugLevel: Logikai változó, amely segítségével extra információt kapunk az SMTP szerverrel való kommunikációról

Dokumentáció a `configparser` csomagról [ezen a linken elérhető](#).

5.4. Indító szkript

Az implementáció a következő fájlban található: [start.sh](#)

A kód bázis tartalmát egy Python `venv` virtuális környezetben tároljuk. Ennek segítségével a

futtatási környezetet izolálni tudjuk és az ebben található Python és egyéb binárisok segítségével futtathatjuk a kódot. A letöltött csomagok csak erre a virtuális környezetre lesznek hatással, rendszerszinten nem települnek fel, így meggátolható az is, hogy a különböző verziójú binárisok összeakadjanak a Python ökoszisztémában.

A szkript a következő fő lépésekre bontható:

1. Az indító szkript először leellenőrzi, hogy az említett virtuális környezetben belül használt binárisok léteznek-e.
2. Ez után a [requirements.txt](#) fájlból feltelepíti az abban definiált csomagokat a `pip` csomagkezelő segítségével.
3. Következő lépésben a virtuális környezetben elhelyezett `.env` fájlból beállítjuk az érzékeny adatokhoz tartozó környezeti változókat.
4. Következőnek elindítjuk az adatbázist tartalmazó és működtető Docker konténert, illetve ez létrehozása kerül, ha még korábban nem volt elkészítve.
5. Végző soron elindítjuk a szolgáltatást a `flask` segítségével. Ez a csomag később, az 5.9 részen kerül bemutatásra.

Információ a Python `venv` modulról [ezen a linken elérhető](#).

Információ a Python `pip` csomagkezelőről [ezen a linken elérhető](#).

5.5. Adatbáziskapcsolat

A szolgáltatásban az adatbáziskapcsolatot a `mysql-connector-python` csomag segítségével hajtja végre a `db.py` fájl. A kapcsolatra annak sikeres létrehozása után, az osztályban eltárolt `conn` adattaggal tudunk később hivatkozni. A kapcsolat létrejöttékor az adatbázist elkészítjük, (ha már létezik, akkor töröljük és újra létrehozuk) majd példa adatokkal töltjük fel a [create_db_with_sample_data.sql](#) fájlban, azért, hogy a rendszer alaphól rendelkezzen néhány felhasználóval, RFID-val, amihez a bemutatóban használt létező kulcsokat rendeltünk hozzá. Így a példában mind a három belépési állapotot le tudjuk fedni (letiltott, engedélyezett, a kártya nem létezik).

Az adatbázis lekérdezésekhez szükséges kód a [repo](#) mappában került megvalósításra.

5.6. SPI kommunikáció

Az implementáció a következő fájlban található: [rfid_spi.py](#)

A kommunikáció a `pi-rc522` csomag segítségével történik, amely rendelkezik olyan tagfüggvényekkel, ami az RFID olvasó aktuális állapotát le tudja kérdezni és karakterláncként visszaadni. A Python beépített `threading` moduljával képesek vagyunk időzítőt létrehozni, amely ez esetben másodpercenként kiolvassa az RFID olvasó aktuális értékét. Ha nem üres karakterláncot kapunk vissza az olvasótól, akkor továbbmegyünk az azonosítás részhez.

Információ a Python `threading` modulról [ezen a linken elérhető](#).

Információ a `pi-rc522` csomagról [ezen a linken elérhető](#).

5.7. I²C kommunikáció

Az implementáció a következő fájlban található: [lcd_i2c.py](#)

Az LCD kijelzővel való kommunikáció az `rpi_lcd` csomag segítségével valósul meg. Alapértelmezetten a *Kerem érintse a kártyáját!* feliratot jelenítjük meg, majd kiírás során figyelünk arra, hogy a megjelenítendő üzenetet 5 másodpercig hagyjuk a kijelzőn, majd töröljük a tartalmat és visszaállítjuk az alapméretezett üzenetet.

Információ az `rpi_lcd` csomagról [ezen a linken elérhető](#).

5.8. Email küldés

Az implementáció a következő fájlban található: [smtp_client.py](#)

A kommunikáció az SMTP szerverrel az `smtplib` modul segítségével kerül megvalósításra. A szolgáltatás indításakor autentikáljuk a korábban említett konfigurációban megadott felhasználót. Abban az esetben, hogy ha a letiltott kártyák belépésének száma eléri a beállított értéket, (jelen esetben ez 3) akkor emailt küldünk a konfigurációban beállított címzettnek. A tartalom a `jinja2` csomag felhasználásával, az [unauthorized.html](#) minta segítségével jön létre. Ebben az emailben megjelenítjük a felhasználó azonosítóját, az RFID kártya számát és a tiltott kártyával történt belépési időket.

Információ az `smtplib` modulról [ezen a linken elérhető](#).

Információ a `jinja2` csomagról [ezen a linken elérhető](#).

5.9. HTTP hívások

A HTTP kérések kiszolgáláshoz szükséges kód a [route](#) mappában került megvalósításra.

A különböző HTTP végpontok megvalósítását a `flask` csomag nyújtja, ami egy webalkalmazások fejlesztésére szolgáló keretrendszer, amely támogatja a *Web Server Gateway Interface* (WSGI) szabványt. A válaszban visszaadott adatátviteli objektumok (DTO) a [dto](#) mappában vannak létrehozva. Ezek egyszerű objektumok, amelyek adatok tárolására, lekérdezésére, serializálására és deserializálására lettek létrehozva. Ezek segítségével pontosan annyi információt tudunk visszaadni, amennyire a kliensnek szüksége van. A kérések és válaszok esetén történő adatcserére a szövegalapú JSON szabványt használjuk.

Információ a `flask` csomagról [ezen a linken elérhető](#).

5.9.1. Az implementált végpontok részletezése:

Felhasználók karbantartásához tartozó végpontok			
Végpont (helyi hálózaton)	Minta JSON kérés	Sikeres válasz	Funkció
GET <code>http://localhost:5000/users</code>		<pre>[{ "py/object": "dao.user_dao.User", "_User__fhId": 1, "_User__fhNev": "Gipsz Jakab" }, { "py/object": "dao.user_dao.User", "_User__fhId": 2, "_User__fhNev": "Nagy Paszkál" }, { "py/object": "dao.user_dao.User", "_User__fhId": 3, "_User__fhNev": "Kiss Béla" }]</pre>	Az összes felhasználó lekérdezése
GET <code>http://localhost:5000/user/3</code>		<pre>{ "py/object": "dao.user_dao.User", "_User__fhId": 3, "_User__fhNev": "Kiss Béla" }</pre>	Egy megadott felhasználó lekérdezése azonosító alapján
DELETE <code>http://localhost:5000/user/3</code>		success	Egy megadott felhasználó törlése azonosító alapján
POST <code>http://localhost:5000/user</code>	<pre>{ "nev": "Teszt Béla" }</pre>	<pre>{ "py/object": "dao.user_dao.User", "_User__fhId": 4, "_User__fhNev": "Teszt Béla" }</pre>	Egy felhasználó létrehozása név alapján
PATCH <code>http://localhost:5000/user</code>	<pre>{ "id": "2", "nev": "Teszt Csaba" }</pre>	<pre>{ "py/object": "dao.user_dao.User", "_User__fhId": 2, "_User__fhNev": "Teszt Csaba" }</pre>	Egy felhasználó nevének frissítése azonosító alapján

8. táblázat. A felhasználók karbantartásához tartozó végpontok

RFID karbantartásához tartozó végpontok			
Végpont (helyi hálózaton)	Minta JSON kérés	Sikeres válasz	Funkció
GET http://localhost:5000/rfids		<pre>[{ "py/object": "dao.rfid_dao.Rfid", "_Rfid__rId": 2, "_Rfid__rErtek": "1913502801" }, { "py/object": "dao.rfid_dao.Rfid", "_Rfid__rId": 1, "_Rfid__rErtek": "3100821010" }]</pre>	Az összes RFID kulcs lekérdezése
GET http://localhost:5000/rfid/3100821010		<pre>{ "py/object": "dao.rfid_dao.Rfid", "_Rfid__rId": 1, "_Rfid__rErtek": "3100821010" }</pre>	Egy megadott RFID lekérdezése azonosító alapján
DELETE http://localhost:5000/rfid/3100821010		success	Egy megadott RFID törlése azonosító alapján
POST http://localhost:5000/rfid	<pre>{ "ertekek": "12345678" }</pre>	<pre>{ "py/object": "dao.rfid_dao.Rfid", "_Rfid__rId": 10, "_Rfid__rErtek": "12345678" }</pre>	Egy RFID létrehozása érték alapján
PATCH http://localhost:5000/rfid	<pre>{ "id": "3", "ertekek": "12345678" }</pre>	<pre>{ "py/object": "dao.rfid_dao.Rfid", "_Rfid__rId": 3, "_Rfid__rErtek": "12345678" }</pre>	Egy RFID értékének frissítése azonosító alapján

9. táblázat. A RFID karbantartásához tartozó végpontok

Felhasználó és RFID kapcsolat karbantartásához tartozó végpontok			
Végpont (helyi hálózaton)	Minta JSON kérés	Sikeres válasz	Funkció
GET http://localhost:5000/userId/2		<pre>[{ "py/object": "dao.user_id_dao.UserId", "_UserId__fhAzonId": 2, "_UserId__fhId": 2, "_UserId__rId": 2, "_UserId__rfErtek": "1913502801", "_UserId__letiltva": true }]</pre>	Egy felhasználóhoz tartozó RFID kulcsok lekérdezése felhasználó azonosító alapján
DELETE http://localhost:5000/userId/12345678		success	Egy felhasználóhoz tartozó RFID törlése RFID érték alapján
POST http://localhost:5000/userId	<pre>{ "fhId": "4", "rErtek": "12345678" }</pre>	<pre>{ "py/object": "dao.user_id_dao.UserId", "_UserId__fhAzonId": 8, "_UserId__fhId": 4, "_UserId__rId": 3, "_UserId__rfErtek": "12345678", "_UserId__letiltva": false }</pre>	Egy RFID érték felhasználóhoz rendelése felhasználó azonosító alapján
PATCH http://localhost:5000/userId/lock/1	<pre>{ "fhId": "4", "rId": "3" }</pre>	<pre>{ "py/object": "dao.user_id_dao.UserId", "_UserId__fhAzonId": 8, "_UserId__fhId": 4, "_UserId__rId": 3, "_UserId__rfErtek": "12345678", "_UserId__letiltva": true }</pre>	Egy felhasználóhoz tartozó RFID kulcs tiltása azonosító alapján
PATCH http://localhost:5000/userId/lock/0	<pre>{ "fhId": "4", "rId": "3" }</pre>	<pre>{ "py/object": "dao.user_id_dao.UserId", "_UserId__fhAzonId": 8, "_UserId__fhId": 4, "_UserId__rId": 3, "_UserId__rfErtek": "12345678", "_UserId__letiltva": false }</pre>	Egy felhasználóhoz tartozó RFID kulcs engedélyezése azonosító alapján

10. táblázat. A felhasználókhhoz rendelt RFID kulcsok végpontjai

Belépések karbantartásához tartozó végpontok		
Végpont (helyi hálózaton)	Sikeres válasz	Funkció
GET http://localhost:5000/entry	<pre>[{ "py/object": "dao.user_entry_dao.UserEntry", "_UserEntry__user": { "py/object": "dao.user_dao.User", "_User__fhId": 2, "_User__fhNev": "Nagy Paszkál" }, "_UserEntry__rfId": 2, "_UserEntry__belepIdo": "2023-12-02 21:17:52" }, { "py/object": "dao.user_entry_dao.UserEntry", "_UserEntry__user": { "py/object": "dao.user_dao.User", "_User__fhId": 2, "_User__fhNev": "Nagy Paszkál" }, "_UserEntry__rfId": 2, "_UserEntry__belepIdo": "2023-12-02 21:17:49" }]</pre>	Az összes belépés lekérdezése
GET http://localhost:5000/entry/user/2	<pre>[{ "py/object": "dao.user_entry_dao.UserEntry", "_UserEntry__user": { "py/object": "dao.user_dao.User", "_User__fhId": 2, "_User__fhNev": "Nagy Paszkál" }, "_UserEntry__rfId": 2, "_UserEntry__belepIdo": "2023-12-02 21:21:06" }]</pre>	Belépések lekérdezése felhasználó azonosító alapján
GET http://localhost:5000/entry/rfid/3100821010	<pre>{ "py/object": "dao.user_entry_dao.UserEntry", "_UserEntry__user": { "py/object": "dao.user_dao.User", "_User__fhId": 1, "_User__fhNev": "Gipsz Jakab" }, "_UserEntry__rfId": 1, "_UserEntry__belepIdo": "2023-12-02 21:21:25" }</pre>	Engedélyezett belépés egy megadott RFID kulcs értékével
GET http://localhost:5000/entry/rfid/1913502801	<pre>{ "error": "Unable to enter using rfid", "details": "1913502801 is locked, cannot enter!" }</pre>	Tiltott belépés egy megadott RFID kulcs értékével
GET http://localhost:5000/entry/rfid/4543543543	<pre>{ "error": "Unable to enter using rfid", "details": "4543543543 is not assigned to any user!" }</pre>	Belépés ismeretlen kártyával

11. táblázat. A belépések karbantartásához tartozó végpontok