Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Hankóczi Gábor

NFC JELENLÉTI RENDSZER

KÜLSŐ KONZULENS

FARKAS LÓRÁNT

TANSZÉKI KONZULENS

Iváncsi Szabolcs Dr.

BUDAPEST, 2023

Tartalomjegyzék

1 Feladatkiírás

A projekt célja egy középiskolai jelenléti rendszer megvalósítása, hogy a tanároknak ne, vagy csak kevesebb időt kelljen a jelenlét adminisztrálásával tölteni. Ezen cél megvalósításához az általunk legcélravezetőbb és legbiztonságosabb megoldás az okostelefonok használata, melyek nagy részében található Near Field Communication [1] (NFC) író és olvasó eszköz. Az elképzelés ami alapján a projektet kialakítottuk, hogy a diákok kihelyezett mikrokontrollerrel [2] vezérelt NFC-vel felszerelt állomásoknál (továbbiakban station) tudják azonosítani magukat, illetve ezzel az azonosítással jelzik jelenlétüket. Az így felvitt adatok alapján a rendszer számon tudja tartani, hogy ki mikor hol volt jelen, így a jelenlét vezetés problémáját automatizáltan megoldja.

2 Megvalósítási tervek

A projekttel Göbhardter Ádámmal ketten foglalkoztunk. A projektet három fő részre osztozzuk, ezek a mobil alkalmazás, amit a tanárok, diákok, portások, és adminok használnak, szerveroldal ami a backendet adja, illetve a station-ök. Ezekből Ádám foglalkozott a mobil alkalmazással, én pedig a szerverrel, illetve a station-ökkel.

2.1 Kezdetleges tervek

Az első elképzelésünk szerint a telefonok emulálnak [3] egy passzív NFC taget [4], ezt olvassa be a station a felhasználó azonosítani kívánja magát, majd ezt az interakciót a station átküldi a szerverre, ahol ez eltárolásra kerül.

Kutatás közben azonban rájöttünk, hogy a mobil eszközök emulálási képességei erőteljesen limitáltak, IOS [5] esetén teljesen lehetetlen, Android [6] közepesen limitált, de mivel az összes diák számára elérhetővé kell tenni a rendszert, más megoldást kellett találnunk.

2.2 Végleges tervek

Legjobb megoldásnak azt találtuk, ha egy kicsit átlakítjuk a rendszer kialakítását, méghozá úgy, hogy a mobilok az NFC funkciók közül csak az olvasást valósítják meg (ezt támogatott minden általunk használni kívánt operációs rendszeren), és a stationök több funkciót valósítanak meg:

- Emulálás: így a mobil eszközök be tudják olvasni a station által kibocsájtott adatot
- Olvasás: így amennyiben egy diáknak nincs okostelefonja, vagy nincs rajta NFC preiféria, akkor ők igényelhetnek az isklolától egy passzív NFC taget, amit a station beolvas, és ezen az úton azonosítja a felhasználót.

Így az azonosításnak két külön módja lesz: A mobil eszköz beolvassa a station által kibocsájtott adatot, és ezt a mobil alkalmazás küldi fel a szervernek tárolásra, vagy a station olvassa be az iskola által szolgáltatott taget, majd a station küldi el az azonosítást a szervernek.

3 Szerver

3.1 Technológiák

A szerver megvalósítására az ASP.NET Core [7] keretrendzert használtam, illetve az ehez ajánlott Entity Framework Core [8]-t és az Identity [9]-t. Adatbázisnak a PostgreSQL [10]-t választottam.

3.2 Adatbázis

Mivel az alkalmazásnak szüksége van adattárolásra az egyértemű megoldás valamiféle adatbázis használata volt. Több adatbázison is gondolkodtam (pl.: MSSQL, MySQL [11]), de végül a választás a PostgreSQL-re esett. Ennek oka nem technikai, sokkal inkább kíváncsiságbeli, mivel míg a korábban felsorolt adatbázisokkal már dolgoztam korábban, PostgreSQL-el még sosem volt dolgom, és szerettem volna kipróbálni.

Adatbázis eléréshez Entity Framework Core-t használtam, így az adatbázis kialakításához több lehetőségem is volt, de a code first [12] megközelítést találtam legkényelmesebbnek, így azt választottam.

A következő entitásokat hoztam létre:

- Attendance: Egy azonosítás, megmondja, hogy melyik felhasználó mikor melyik station-nél volt jelen
- Class: Egy adott tanóra
- Exempt: Kikérő/igazolás, ami egy időintervallumban érvényes
- Group: Diákok azon csoportja, akiknek azonosak az órái (pl.: 10.a, 11. matekfakt)
- PassiveTag: Passzív NFC tag, amivel felhasználók azonosíthatják magukat
- Station: Kihelyezz eszközök
- User: Felhasználók

Ezek mellett az adatbázisban találhatóak egyéb táblák, amiket a felhasználókezeléshez az IdentityServer [13] hoztt létre.

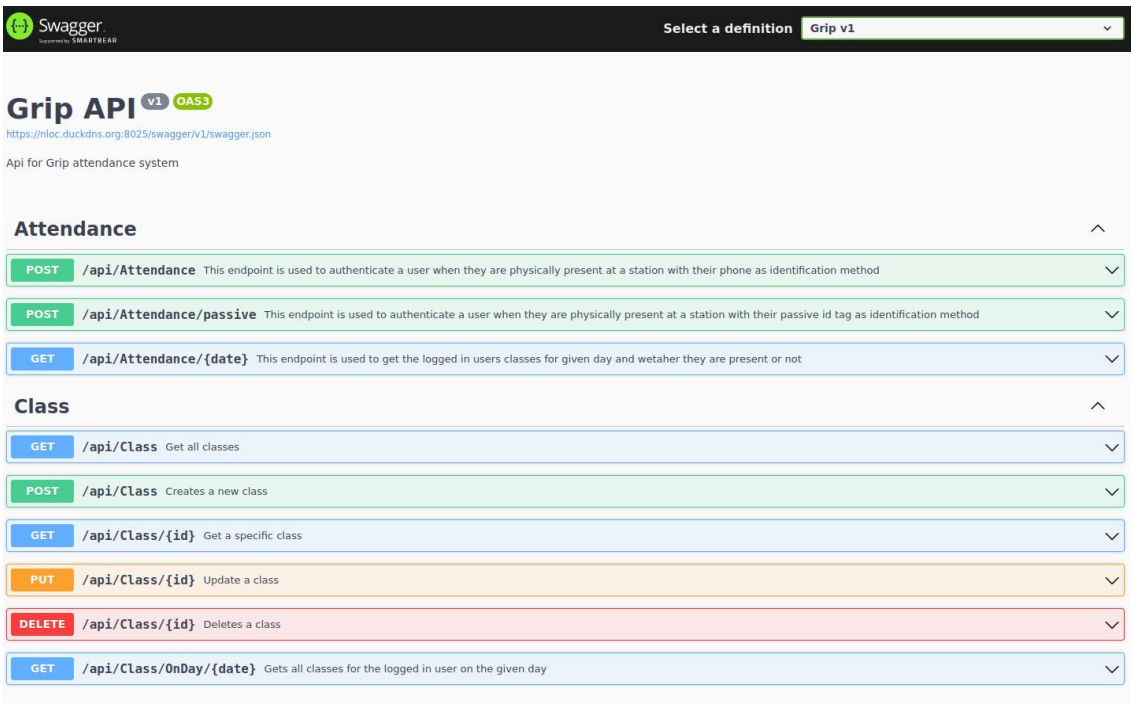
1. ábra: Adatbázis Entity Relationship [14] (ER) diagram

3.3 Api

A szerver oldal főleg backend funkciót lát el, így egy Application Programming Interface-en (API-n) keresztül teszi elérhetővé a belső szolgáltatásait. Mivel ugyan azt az api-t magas szintű oksostelefonokról, és alacsony szintű mikrokontrollerekről is el kell érni, úgy döntöttem, hogy a Representational State Transfer [14] (REST) elveit fogom követi.

A REST elveinek megfelelően a végpontok a Hypertext Transfer Protocol [15] (HTTP) különböző igéit használja különböző akciók végrehajtásához.

Dokumentálás, illetve az integrálást segítésére az Swagger [16] eszközt használtam, hogy gyorsan és egyszerűen OpenAPI [17] specifikációnak megfelelő dokumentációt kapjak a végpontokhoz.



2. ábra: Swagger UI

3.4 Felépítés

A szerver felépítésénél a három rétegű arhitektúrát [18] használtam, így a szoftver felépítése jól átlátható, és az egyes rétegek, majdnem teljesen le lettek választva egymásról, csak interfészeken keresztül érik el egymást. Szigorúan követtem, hogy alsóbb réteg sose hivatkozzon felsőbb rétegre.

3.4.1 Prezentációs réteg

A prezentációs réteg tartalmazza az Api végpontokat, a valós idejű kommunikációhoz használt végpontokat, illetve a middleware [19]-eket, amiket amiket valamilyen módon használ a prezentációs réteg.

Az api végpontok nagyrészt csak továbbhívnak a Business Logic Layer-be (BLL) de emellet itt történik a authentikáció, authorizáció, és a végpontok dokumentálása.

3.4.2 Üzleti logikai réteg

A BLL tartalmazza az alkalmazás logikáját. Itt vannak megfogalmazva a lekérdezések, a feldolgozás logikája, illetve itt találhatóak a prezentációs réteggel

kommunikációra használt Data Transfer Object-ek (DTO-k), valamint a hibák jelzésére használt kivételek.

3.4.2.1 Szolgátatások

Ezek az osztályok felelősek az adatok kezeléséért, lekérdezések megcálaszolásáért, illetve a feliratkozott értesítések kiküldéséért. Mindegyik osztályhoz tartozik egy interfész, így biztosítva a gyenge csatolást, és a könnyű tesztelhetőséget.

3.4.2.2 Providerek

Ezek az osztályok felelősek a service osztályokat segíteni valamiféle funkció megvalósításával. Itt érdemes megemlíteni az HMACTokenProvider osztályt, amely a Hash-Based Message Authentication [20] (HMAC) funkciót valósítja meg, amit az azonosítási üzenetek aláírásához használtam.

3.4.3 Adatelérési réteg

Itt található az adatbázis leképező Object Relational Mapper [21] (ORM), ennek segítségével az adatbázisban található táblák és adatok könnyedén elérhetőek C# kódból. A használt ORM az ASP.NET-el nagykörben elterjedt Entity Framework Core. Itt találhatóak a leképezéshez használt modell osztályok.

3.5 Biztonság

Hangsúlyt fektettem a rendszer biztonságára, hogy ne lehessen hamis adatokat feltölteni, valós adatokat meghamisítani, vagy bármi egyéb módon kijátszani a szervert.

3.5.1 Felhasználók

A felhasználók azonosítása sütikkben tárolt JWT [22]-kkel történik, au authorizáció pedig szerep alapú, tehát a diákok csak azokat a végpontokat hívhatják, amit nekik hívni szabad, de az adminoknak és a tanároknak nagyobb hatáskörük van.

3.5.2 Kapcsolat a station-ökkel

Alapvetően a station-ök egy a szerver és a station-ök konfigurációjában szereplő api kulcsal tudnak egymással kommunikálni közvetlenü, azonban amikor ezt a mobil eszközön keresztül teszik, más megoldás szükséges.

3.5.2.1 Közös titok kiosztása

Annak érdekében, hogy minden station-nek saját aláíró kulcsa legyen a szerver a felelős. A station-ök lekérdezhetik ezt a kulcsot a közös api kulcs használatával, majd használják ezt az üzenetek aláírásához, így biztosítva, hogy az üzenetek módosítatlanul érnek el a szerverhez.

3.6 Valós idejű kommunikáció

A projekt jelenleg egy helyen igényel valós idejű kommunikációt: A portásoknak látniuk kell, hogy az általuk felügyelt kapunk ki azonosította éppen magát. Ennek érdekében a szerver lehetőséget biztosít a feliratkozás, majd szerver oldali push alapú kommunikációra. Ehhez a SignalR [23] technológiát használtam. A végpontra csatlakozva a kliensnek ki kell választania, hogy melyik station-t szeretné figyelni. Ha ezt megtette, akkor utána a szerver minden sikeres azonosításnál üzenetet küld a kliensnek, így az meg tudja jeleníteni azt.

3.7 Felhasználókezelés

A felhasználókezelés az egyéni igények miatt kicsit eltér a megszokott megoldástól. Azért, hogy ne tudjon bárki regisztrálni, csak azok, akik a középiskola hallgatói, a regisztrációt az adminisztrátorok feladatává tettük. Nekik meg kell adni a diák e-mail-címét, illetve nevét, hogy a felhasználó bekerüljön az adatbázisba. Ezt követően a rendszer egy emailt küld a megadott e-mail-címre, így értesítve a diákot a regisztrációról. Ebben az üzenetben található egy email megerősítő kód, amit a diákok felhasználnak az első bejelentkezésnél.

A többi bejelentkezéssel ellentétben az első egy kicsit más, mivel itt meg kell adni az email-ben megkapott kódot, illetve a jelszót, amit a későbbi belépéseknél szeretne használni. Ezek után a rendszer ellenőrzi a megadott kódot, így igazolja, hogy a fiók tényleg a megadott diákhoz tartozik, majd eltárolja a megadott jelszót.

Ezek után a megszokott felhasználói folyammal találkozhatnak a felhasználók a további felhasználókezeléssel kapcsolatos akciók során. Lehetőségül van elfelejtett jelszó esetén új jelszót igényelni, illetve e-mail-cím jelszó párossal belépni.

3.8 Hosztolás

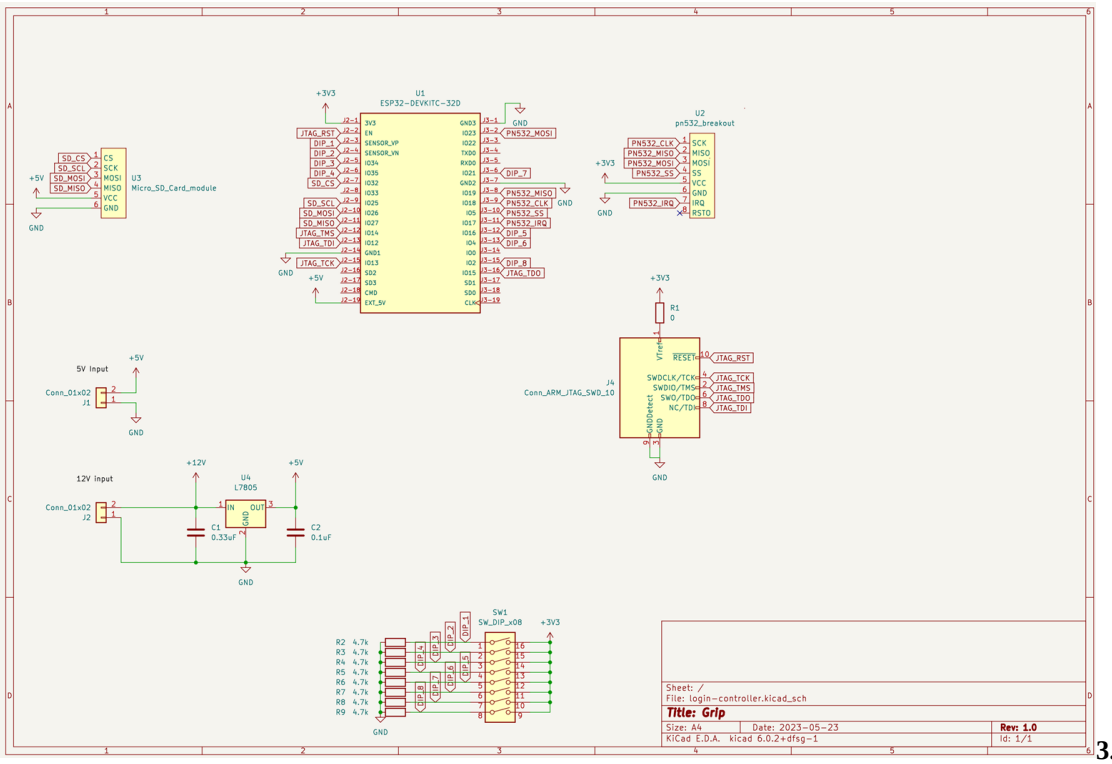
A projekt alatt két helyen publikáltam a szerver. Először az Azure [24] által biztosított virtuális gépen lett elérhető, de mivel az egyetem által biztosított kreditem pár hónap alatt elfogyott, ezért a konzulensel egyeztetve a szerver kapott tárhelyet a Nokia szerverein. Az első hosztolási megoldásnál felmerül problémaként, hogy a szerver nem tud email-t küldeni, mivel a nem Enterprise szintű előfizetéseknél az e-maileknél használt 25-ös port [25] blokkolva van, így más megoldást kellett keresnem. A legegyszerűbb alternatív megoldásnak az tűnt, hogy egy külső email szolgáltatót használunk a levelek kiküldésére. Esetünkben ez a Gmail lett, így ott beregisztráltam egy fiókot az alkalmazás nevében, és az email-ek azóta onnan kerülnek kiküldésre.

4 Station

A station-öknek többféle elvárásnak is meg kell felelnie. Magas rendelkezésreállást kell biztosítani, a szerverrel kommunikálni kell, illetve azonosítani kell a felhazsnálókat, mind mobil eszközzel mind passzív NFC tag-el, illetve az azonosításnak gyorsnak kell lennie, hogy ne keletkezzen nagy sor.

4.1 Hardware

A hardware elemek kiválasztásánál több szempontot is figyelembe vettem: megfeleljenek a funkcionális elvárásoknak, könnyen elérhető alkatrészek legyenek, illetve segítsék az eszköz telepítését és karbantartását, illetve megfelelő dokumentáció legyen elérhető az alkatrészekhez.



3. ábra: Kapcsolási rajz

4.1.1 Microcontroller

A fő vezérlőegység egy ESP32 Wroom32U [26] mikrokontroller. Ez a kétmagos eszköz beépített wifi perifériával kapható, így megoldhatóvá teszi a kommunikációt a

szerverrel. Ez egy kimondottan gyakori és erős mikrokontroller, így meg tudja oldani a gyors NFC kommunikációt, Wifi kapcsolat kezelését, illetve a kriptográfiai feladatokat is (HMAC aláírás).

4.1.2 NFC kommunikáció

A projekt elején a kiválasztott eszköz az RC522 [27] lett volna, azonban mivel ez nem képes emulációra, ezért inkább váltottottam a PN532 [28]-es modulra.

4.1.3 Adattárolás

Mivel a firmwarenek több konfigurációs infromációra is szüksége van (pl.: wifi ssid, wifi jelszó, szerver elérési útja, api kulcs …), illetve szükség lehet logolási lehtőségre, és a logok későbbi visszaolvasására, ezért úgy döntöttem, hogy egy mikro SD kártya lenne erre a célra a legmegfelelőbb.

4.1.4 Egyéb

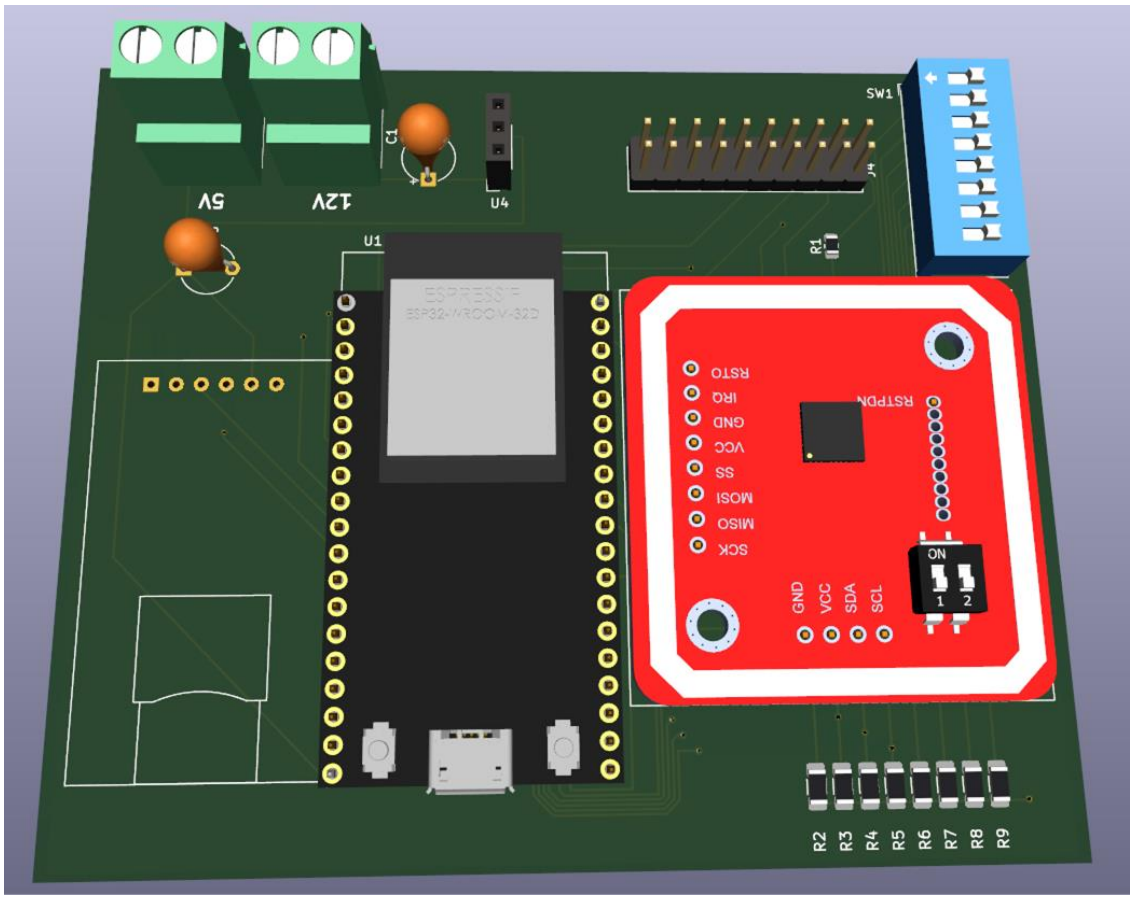
Annak érdekében, hogy a station-ök azonosítója könnyen állítható legyen, és kívülről egyértelműen látszódjon, valamint, hogy az sd kártya tartalmát csak simán másolni kelljen minden középiskolához, és ne kelljen minding átírni azon az azonosítót, ezért az eszközre elhelyeztem egy DIP [29] kapcsolót, amin bináris formában be lehet állítani az eszköz azonosítóját. A tervezett áramkör tartalmaz többféle tápellátási lehetőséget is. Meghajtható akár 5V-ról (közeli táp esetén) nominális 12V-ról, így távolabbi tápegység esetén sem kell aggódni a felszültségveszteség miatt.

4.2 Firmware

A firmwaret komponensekre bontottam, ígyy megőrizve a kód áttekinthetőségét. Külön komponenst hoztam létre az NFC periféria alacsony és magas szintű kezeléséhez, így egy absztrakciós réteget vezettem be. FreeRTOS [30]-t használtam, így egyszerre több szálon futhat a feldolgozás.

4.3 Nyák

Terveztem egy nyákot, hogy az eszközök biztonságban kihelyezhetőek legyenek.



4.4 Működés

4.4.1 Indulás

Indítás után a kontroller a következő task [31]-okat indítja el a következő sorrendben:

1. SD task
2. Config task
3. Wifi task
4. RTC task
5. NFC task

4.4.2 Egyéb

Indítás után a következő esetekben tér el a firmware működése az alapvető ciklustól:

- Periodikusan a kontroller szinkronizálja a belső órát egy külső időszerver [32] segítségével.
- Amikor a station éppen NFC emuláló módban van, és egy mobil eszköz kommunikációt kezdeményez, akkor a station a korábban összeállított NDEF [33] üzenetet átküldi az olvasónak.
- Amikor a station éppen NFC olvasó módban van, és egy passzív NFC tag található a hatókörében, akkor azt beolvassa, majd elküldi a szervernek az azonosítási próbálkozást.

4.5 Adatformák

Mobil eszközzel az NFC kommunikáció NDEF formátummal történik. Egy ilyen üzenetben két rekordot helyez el a station:

1. Azonosítási token
2. Azonosítási token HMAC aláírása

Az eszköz által készített azonosítási token a következő formát veszi fel:

<station id>_<unix time>_<salt>

Ahol <station id> a station azonosítója, ami a nyákon található DIP kapcslolóval korábban be lett állítva, <unix time> a belső óra szerinti idő uinix timestamp[35] formában, <salt> pedig egy véletlenszerűen választott szám, biztonsági okokból.

Az aláírás a station-höz tartozó szervertől lekért kulcsal aláírt változata az azonosítási token-nek.

5 Irodalomjegyzék

[1] "What is NFC? Near Field Communication Explained," [Online]. Available: http://nearfieldcommunication.org/about-nfc.html. [Accessed 06 06 2023].

[2] "Microcontrollers and its types," [Online]. Available: https://www.geeksforgeeks.org/microcontroller-and-its-types/. [Accessed 06 06 2023].

[3] "Host-based card emulation overview," [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/hce. [Accessed 06 06 2023].

[4] "Learn All About NFC Tags," [Online]. Available: https://www.nomtek.com/blog/what-are-nfc-tags. [Accessed 06 06 2023].

[5] "iOS 16," [Online]. Available: https://www.apple.com/ios/ios-16/. [Accessed 06 06 2023].

[6] "Android," [Online]. Available: Android. [Accessed 06 06 2023].

[7] "ASP.NET," [Online]. Available: https://dotnet.microsoft.com/en-us/apps/aspnet. [Accessed 06 06 2023].

[8] "Overview of Entity Framework Core," [Online]. Available: https://learn.microsoft.com/en-us/ef/core/. [Accessed 06 06 2023].

[9] "Introduction to Identity on ASP.NET Core," [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio. [Accessed 06 06 2023].

[10] "PostgreSQL: The world's most advanced open source database," [Online]. Available: https://www.postgresql.org/. [Accessed 06 06 2023].

[11] "MySQL vs MSSQL: Comparing Similarities and Differences," [Online]. Available: https://www.plesk.com/blog/various/mysql-vs-mssql/. [Accessed 06 06 2023].

[12] "Using Entity Framework Core Code First Approach," [Online]. Available: https://www.c-sharpcorner.com/article/using-entity-framework-core/. [Accessed 06 06 2023].

[13] "Duende Software," [Online]. Available: https://duendesoftware.com/products/identityserver. [Accessed 06 06 2023].

[14] "What is REST - REST API Tutorial," [Online]. Available: https://restfulapi.net/. [Accessed 06 06 2023].

[15] "HTTP," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP. [Accessed 06 06 2023].

[16] "API Documentation & Design Tools for Teams | Swagger," [Online]. Available: https://swagger.io/. [Accessed 06 06 2023].

[17] "OpenAPI Specification v3.1.0 | Introduction, Definitions, & More," [Online]. Available: https://spec.openapis.org/oas/latest.html. [Accessed 06 06 2023].

[18] "What is Three-Tier Architecture | IBM," [Online]. Available: https://www.ibm.com/topics/three-tier-architecture. [Accessed 06 06 2023].

[19] "ASP.NET Core Middleware," [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-7.0. [Accessed 06 06 2023].

[20] "HMAC (Hash-Based Message Authentication Codes) Definition | Okta," [Online]. Available: https://www.okta.com/identity-101/hmac/. [Accessed 06 06 2023].

[21] "What is an ORM – The Meaning of Object Relational Mapping Database Tools," [Online]. Available: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/. [Accessed 06 06 2023].

[22] "JSON Web Tokens," [Online]. Available: https://jwt.io/. [Accessed 06 06 2023].

[23] "Overview of ASP.NET Core SignalROverview of ASP.NET Core SignalR," [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0. [Accessed 06 06 2023].

[24] "Cloud Computing Services," [Online]. Available: https://azure.microsoft.com/en-us. [Accessed 06 06 2023].

[25] "What is a computer port? | Ports in networking | Cloudflare," [Online]. Available: https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/. [Accessed 06 06 2023].

[26] "ESP32WROOM32D & ESP32WROOM32U Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf. [Accessed 06 06 2023].

[27] "MFRC522 Standard performance MIFARE and NTAG frontend," [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf. [Accessed 06 06 2023].

[28] "PN532 User Manual," [Online]. Available: https://www.nxp.com/docs/en/user-guide/141520.pdf. [Accessed 06 06 2023].

[29] "The Complete Guide to DIP Switches | RS," [Online]. Available: https://uk.rs-online.com/web/content/discovery/ideas-and-advice/dip-switches-guide. [Accessed 06 06 2023].

[30] "FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions," [Online]. Available: https://www.freertos.org/. [Accessed 06 06 2023].

[31] "Tasks and Co-routines [Getting Started] - FreeRTOS," [Online]. Available: https://www.freertos.org/taskandcr.html. [Accessed 06 06 2023].

[32] "Public NTP | Google for Developers," [Online]. Available: https://developers.google.com/time. [Accessed 06 06 2023].

[33] "Introducing NDEF - Beginning NFC," [Online]. Available: https://www.oreilly.com/library/view/beginning-nfc/9781449324094/ch04.html. [Accessed 06 06 2023].

[34] "What is an Entity Relationship Diagram (ERD)?," [Online]. Available: https://www.lucidchart.com/pages/er-diagrams. [Accessed 06 06 2023].

[35] "Unix Time Stamp - Epoch Converter," [Online]. Available: https://www.unixtimestamp.com/. [Accessed 06 06 2023].