

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 4](#__RefHeading___Toc2342_2345644338)

[2 Megvalósítási tervek 5](#__RefHeading___Toc2344_2345644338)

[2.1 Kezdetleges tervek 5](#__RefHeading___Toc2346_2345644338)

[2.2 Végleges tervek 5](#__RefHeading___Toc2346_23456443383)

[3 Szerver 6](#__RefHeading___Toc2344_23456443381)

[3.1 Technológiák 6](#__RefHeading___Toc2346_23456443381)

[3.2 Adatbázis 6](#__RefHeading___Toc2346_234564433811)

[3.3 Api 7](#__RefHeading___Toc2346_234564433812)

[3.4 Felépítés 8](#__RefHeading___Toc2346_234564433813)

[3.4.1 Prezentációs réteg 8](#__RefHeading___Toc2346_23456443381342)

[3.4.2 Üzleti logikai réteg 8](#__RefHeading___Toc2346_234564433813422)

[3.4.3 Adatelérési réteg 9](#__RefHeading___Toc2346_234564433813423)

[3.5 Biztonság 9](#__RefHeading___Toc2346_2345644338134)

[3.5.1 Felhasználók 9](#__RefHeading___Toc2346_23456443381343)

[3.5.2 Kapcsolat a station-ökkel 9](#__RefHeading___Toc2346_23456443381344)

[3.6 Valós idejű kommunikáció 10](#__RefHeading___Toc2346_23456443381341)

[3.7 Felhasználókezelés 10](#__RefHeading___Toc2346_2345644338135)

[3.8 Hosztolás 11](#__RefHeading___Toc2346_2345644338136)

[4 Station 12](#__RefHeading___Toc2344_23456443382)

[4.1 Hardware 12](#__RefHeading___Toc2346_234564433814)

[4.1.1 Microcontroller 13](#__RefHeading___Toc2346_2345644338142)

[4.1.2 NFC kommunikáció 13](#__RefHeading___Toc2346_23456443381421)

[4.1.3 Adattárolás 13](#__RefHeading___Toc2346_23456443381422)

[4.1.4 Egyéb 13](#__RefHeading___Toc2346_23456443381423)

[4.2 Firmware 13](#__RefHeading___Toc2346_2345644338111)

[4.3 Nyák 14](#__RefHeading___Toc2346_2345644338121)

[4.4 Működés 14](#__RefHeading___Toc2346_234564433815)

[4.4.1 Indulás 14](#__RefHeading___Toc2346_2345644338151)

[4.4.2 Egyéb 15](#__RefHeading___Toc2346_23456443381512)

[4.5 Adatformák 15](#__RefHeading___Toc2346_2345644338152)

[5 Irodalomjegyzék 16](#__RefHeading___Toc2444_2345644338)

# 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 Communication1 (NFC) író és olvasó eszköz. Az elképzelés ami alapján a projektet kialakítottuk, hogy a diákok kihelyezett mikrokontrollerrel2 vezérelt NFC-vel felszrelt 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.

# Megvalósítási tervek

A projektel 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.

## Kezdetleges tervek

Az első elképzelésünk szerint a telefonok emulálnak3 egy passzív NFC taget4, 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, IOS5 esetén teljesen lehetetlen, Android6 közepesen limitált, de mivel az összes diák számára elérhetővé kell tenni a rendszert, más megoldást kellett találnunk.

## 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.

# Szerver

## Technológiák

A szerver megvalósítására az ASP.NET Core7 keretrendzert használtam, illetve az ehez ajánlott Entity Framework Core8-t és az Identity9-t. Adatbázisnak a PostgreSQL10-t választottam.

## Adatbázis

Mivel az alkalmazásnak szüksége van adattárolásra az egyértemű megoldás valamiféle adatbázis használata volt. Több adatbázison is gondolkodtam (pl.: MSSQL, MySQL11), 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.

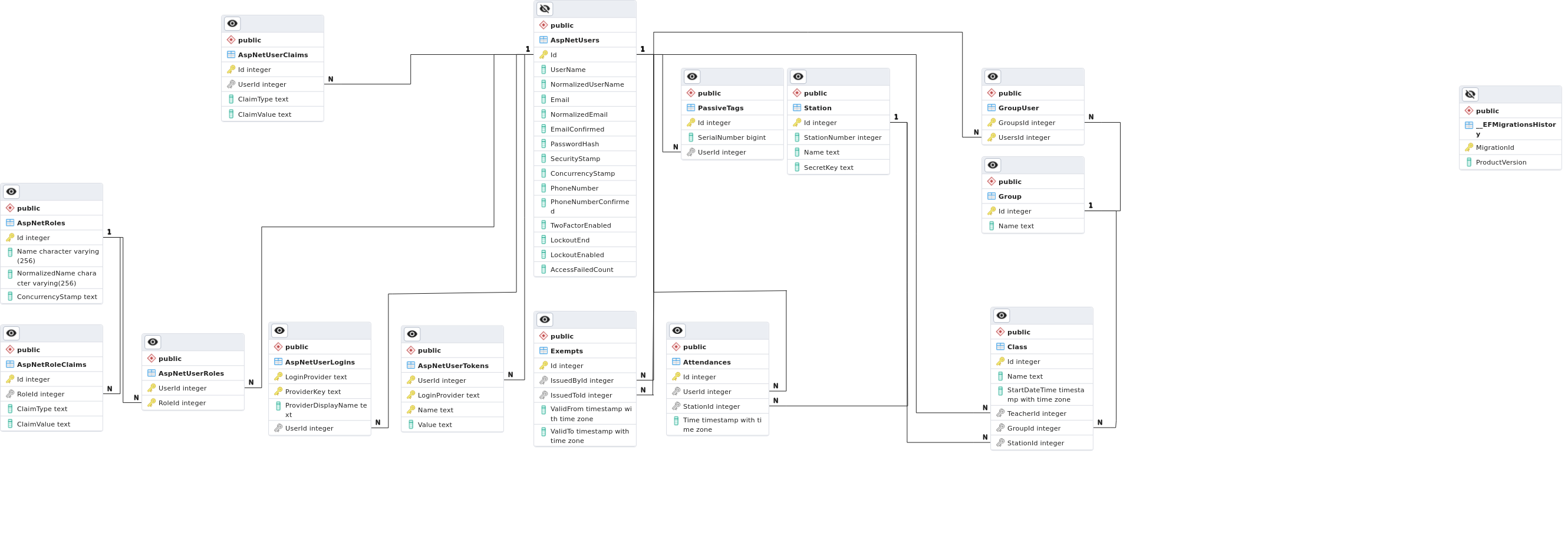
Adatbázis eléréshez Entity Framework Core-t használtam, így az adatbázis kialakításához több lehetőségem is volt, de a code first12 megközelítést találtam legkényelmesebbnek, így azt választottam.

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: Kihelyzezz eszközök
* User: Felhasználók

Ezek mellett az adatbázisban találhatóak egyéb táblák, amiket a felhasználókezeléshez az IdentityServer13 hoztt létre.

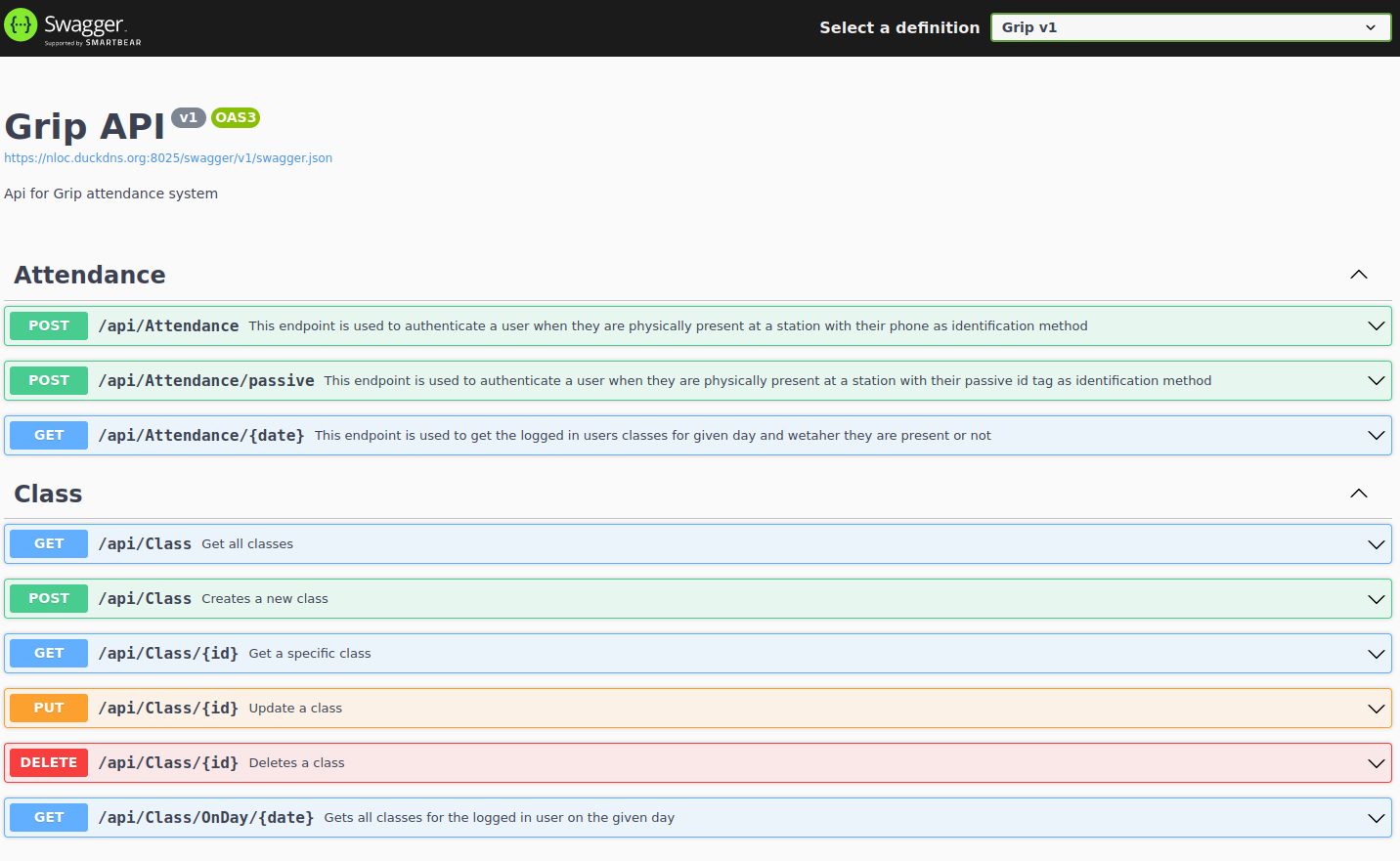
## Api

1. ábra: Adatbázis Entity Relationship14 (ER) diagram

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 Transfer15 (REST) elveit fogom követi.

A REST elveinek megfelelően a végpontok a Hypertext Transfer Protocol16 (HTTP) különböző igéit használja különböző akciók végrehajtásához.

Dokumentálás, illetve az integrálást segítésére az Swagger17 eszközt használtam, hogy gyorsan és egyszerűen OpenAPI18 specifikációnak megfelelő dokumentációt kapjak a végpontokhoz.

 2. ábra: Swagger UI

## Felépítés

A szerver felépítésénél a három rétegű arhitektúrát19 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.

### Prezentációs réteg

A prezentációs réteg tartalmazza az Api végpontokat, a valós idejű kommunikációhoz használt végpontokat, illetve a middleware20-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.

### 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.

#### 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.

#### Providerek

Ezek az osztályok felelősek a service osztályokat segíteni valamiféle funkció megvalósításával. Itt érdemes megemlíteni az HMACTokenProvider osztályt, amely a Hash-Based Message Authentication21 (HMAC) funkciót valósítja meg, amit az azonosítási üzenetek aláírásához használtam.

### Adatelérési réteg

Itt található az adatbázis leképező Object Relational Mapper22 (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.

## Biztonság

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

### Felhasználók

A felhasználók azonosítása sütikkben tárolt JWT23-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.

### 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.

#### 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.

## 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. Ehez a SignalR24 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.

## 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.

## Hosztolás

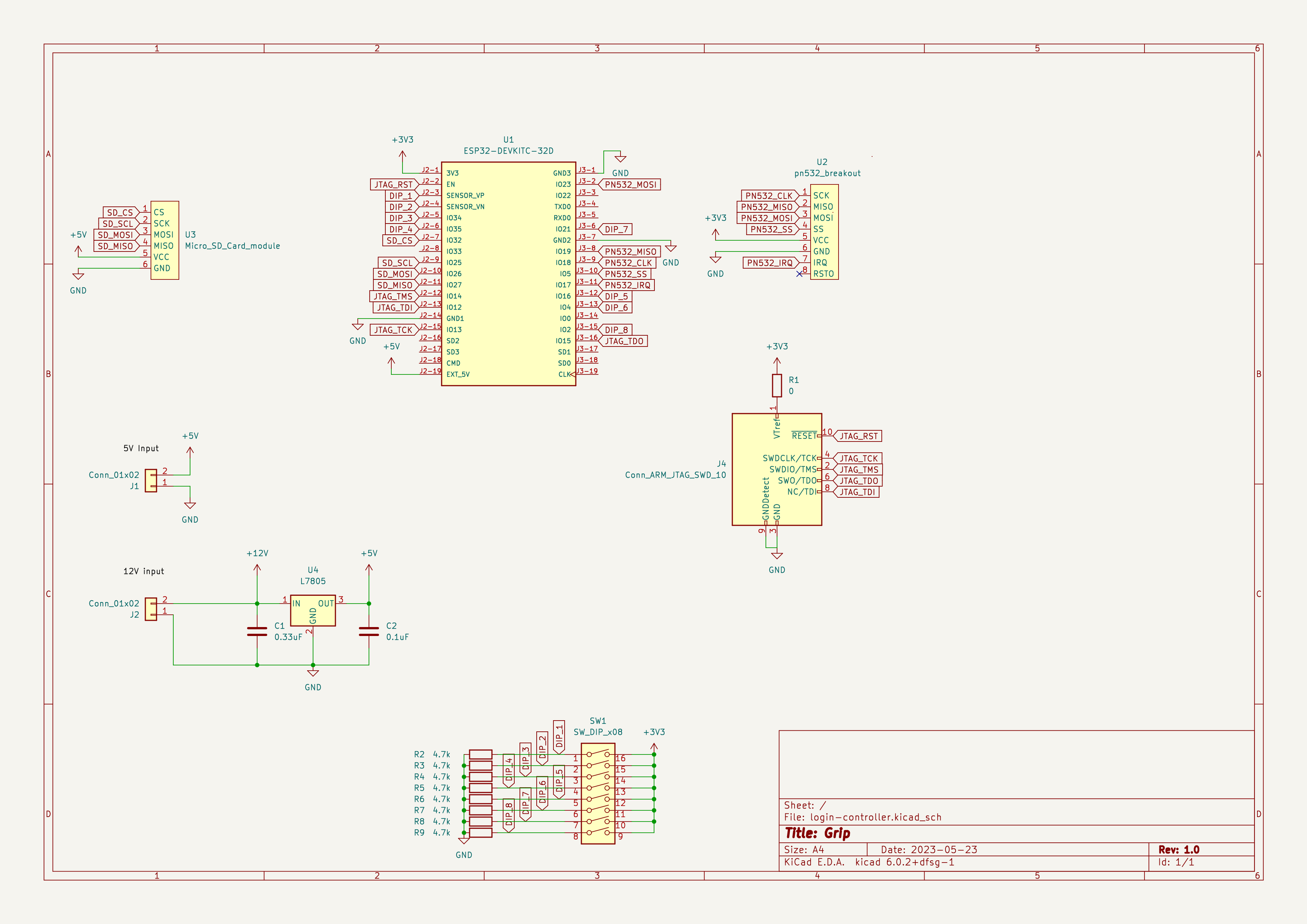
A projekt alatt két helyen publikáltam a szerver. Először az Azure25 által biztosított virtuális gépen lett elérhető, de mivel az egyetem által biztosított kreditem pár hónap alatt elfogyott, ezért a konzulensel egyeztetve a szerver kapott tárhelyet a Nokia szerverein. Az első hosztolási megoldásnál felmerül problémaként, hogy a szerver nem tud email-t küldeni, mivel a nem Enterprise szintű előfizetéseknél az emaileknél használt 25-ös port26 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.

# 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.

## 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

### Microcontroller

A fő vezérlőegység egy ESP32 Wroom32U27 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).

### NFC kommunikáció

A projekt elején a kiválasztott eszköz az RC52228 lett volna, azonban mivel ez nem képes emulációra, ezért inkább váltottottam a PN53229-es modulra.

### 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.

### Egyéb

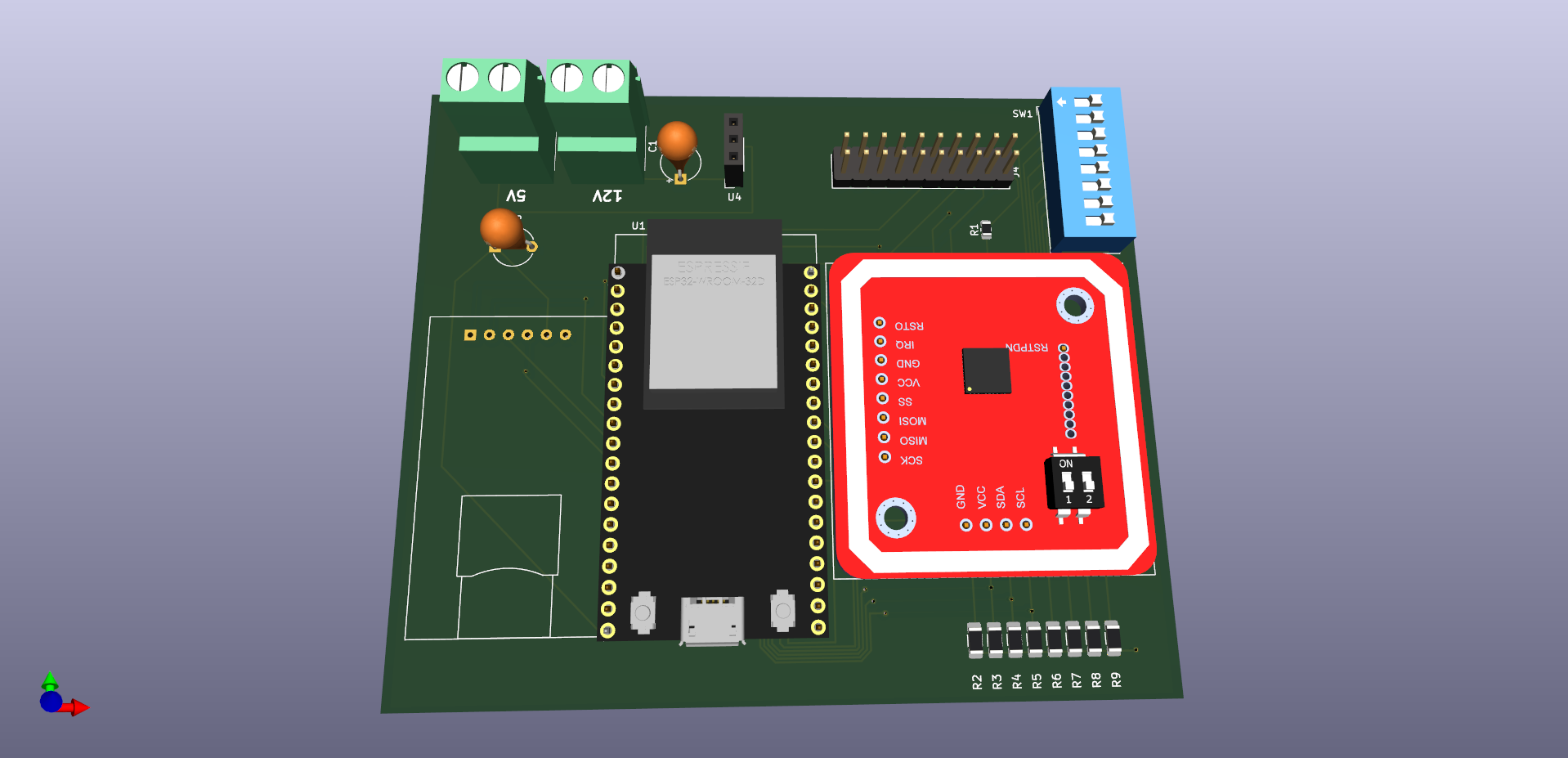
Annak érdekében, hogy a station-ök azonosítója könnyen állítható legyen, és kívülről egyértelműen látszódjon, valamint, hogy az sd kártya tartalmát csak simán másolni kelljen minden középiskolához, és ne kelljen minding átírni azon az azonosítót, ezért az eszközre elhelyeztem egy DIP30 kapcsolót, amin 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.

## 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. FreeRTOS31-t használtam, így egyszerre több szálon futhat a feldolgozás.

## Nyák

Terveztem egy nyákot, hogy az eszközök biztonságban kihelyezhetőek legyenek.



## Működés

### Indulás

Indítás után a kontroller a következő task32-okat indítja el a következő sorrendben:

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

### Egyéb

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

* Periodikusan a kontroller szinkronizálja a belső órát egy külső időszerver33 segítségével.
* Amikor a station éppen NFC emuláló módban van, és egy mobil eszköz kommunikációt kezdeményez, akkor a station a korábban összeállított NDEF34 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.

## 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 timestamp35 formában, <salt> pedig egy véletlenszerűen választott szám, biztonsági okokból.

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

# Irodalomjegyzék

Bibliography

www 1: ,<http://nearfieldcommunication.org/about-nfc.html>

art 2: Microcontrollers and its types, <https://www.geeksforgeeks.org/microcontroller-and-its-types/>

art 3: Host-based card emulation overview, <https://developer.android.com/guide/topics/connectivity/nfc/hce>

art 4: Learn All About NFC Tags, <https://www.nomtek.com/blog/what-are-nfc-tags>

art 5: iOS 16, <https://www.apple.com/ios/ios-16/>

art 6: Android, <https://www.android.com/>

art 7: ASP.NET, <https://dotnet.microsoft.com/en-us/apps/aspnet>

art 8: Overview of Entity Framework Core, <https://learn.microsoft.com/en-us/ef/core/>

art 9: Introduction to Identity on ASP.NET Core, <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio>

art 10: PostgreSQL: The world's most advanced open source database, <https://www.postgresql.org/>

art 11: MySQL vs MSSQL: Comparing Similarities and Differences, <https://www.plesk.com/blog/various/mysql-vs-mssql/>

art 12: Using Entity Framework Core Code First Approach, <https://www.c-sharpcorner.com/article/using-entity-framework-core/>

art 13: Duende Software, <https://duendesoftware.com/products/identityserver>

art 14: What is an Entity Relationship Diagram (ERD)?, <https://www.lucidchart.com/pages/er-diagrams>

art 15: What is REST - REST API Tutorial, <https://restfulapi.net/>

art 16: HTTP, <https://developer.mozilla.org/en-US/docs/Web/HTTP>

art 17: API Documentation & Design Tools for Teams | Swagger, <https://swagger.io/>

art 18: OpenAPI Specification v3.1.0 | Introduction, Definitions, & More, <https://spec.openapis.org/oas/latest.html>

art 19: What is Three-Tier Architecture | IBM, <https://www.ibm.com/topics/three-tier-architecture>

art 20: ASP.NET Core Middleware, <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-7.0>

art 21: HMAC (Hash-Based Message Authentication Codes) Definition | Okta, <https://www.okta.com/identity-101/hmac/>

art 22: What is an ORM – The Meaning of Object Relational Mapping Database Tools, <https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/>

art 23: JSON Web Tokens, <https://jwt.io/>

art 24: Overview of ASP.NET Core SignalROverview of ASP.NET Core SignalR, <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0>

www 25: Cloud Computing Services | Microsoft Azure,<https://azure.microsoft.com/en-us>

www 26: What is a computer port? | Ports in networking | Cloudflare,<https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/>

www 27: ESP32WROOM32D & ESP32WROOM32U Datasheet,<https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf>

www 28: MFRC522 Standard performance MIFARE and NTAG frontend,<https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>

www 29: PN532 User Manual,<https://www.nxp.com/docs/en/user-guide/141520.pdf>

www 30: The Complete Guide to DIP Switches | RS,<https://uk.rs-online.com/web/content/discovery/ideas-and-advice/dip-switches-guide>

art 31: FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions, <https://www.freertos.org/>

www 32: Tasks and Co-routines [Getting Started] - FreeRTOS,<https://www.freertos.org/taskandcr.html>

www 33: Public NTP  |  Google for Developers,<https://developers.google.com/time>

www 34: 4. Introducing NDEF - Beginning NFC [Book],<https://www.oreilly.com/library/view/beginning-nfc/9781449324094/ch04.html>

www 35: Unix Time Stamp - Epoch Converter,<https://www.unixtimestamp.com/>

1: , <http://nearfieldcommunication.org/about-nfc.html>, []

2: Microcontrollers and its types, <https://www.geeksforgeeks.org/microcontroller-and-its-types/>, [ ]

3: Host-based card emulation overview, <https://developer.android.com/guide/topics/connectivity/nfc/hce>, [ ]

4: Learn All About NFC Tags, <https://www.nomtek.com/blog/what-are-nfc-tags>, [ ]

5: iOS 16, <https://www.apple.com/ios/ios-16/>, [ ]

6: Android, <https://www.android.com/>, [ ]

7: ASP.NET, <https://dotnet.microsoft.com/en-us/apps/aspnet>, [ ]

8: Overview of Entity Framework Core, <https://learn.microsoft.com/en-us/ef/core/>, [ ]

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

10: PostgreSQL: The world's most advanced open source database, <https://www.postgresql.org/>, [ ]

11: MySQL vs MSSQL: Comparing Similarities and Differences, <https://www.plesk.com/blog/various/mysql-vs-mssql/>, [ ]

12: Using Entity Framework Core Code First Approach, <https://www.c-sharpcorner.com/article/using-entity-framework-core/>, [ ]

13: Duende Software, <https://duendesoftware.com/products/identityserver>, [ ]

14: What is an Entity Relationship Diagram (ERD)?, <https://www.lucidchart.com/pages/er-diagrams>, [ ]

15: What is REST - REST API Tutorial, <https://restfulapi.net/>, [ ]

16: HTTP, <https://developer.mozilla.org/en-US/docs/Web/HTTP>, [ ]

17: API Documentation & Design Tools for Teams | Swagger, <https://swagger.io/>, [ ]

18: OpenAPI Specification v3.1.0 | Introduction, Definitions, & More, <https://spec.openapis.org/oas/latest.html>, [ ]

19: What is Three-Tier Architecture | IBM, <https://www.ibm.com/topics/three-tier-architecture>, [ ]

20: ASP.NET Core Middleware, <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-7.0>, [ ]

21: HMAC (Hash-Based Message Authentication Codes) Definition | Okta, <https://www.okta.com/identity-101/hmac/>, [ ]

22: What is an ORM – The Meaning of Object Relational Mapping Database Tools, <https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/>, [ ]

23: JSON Web Tokens, <https://jwt.io/>, [ ]

24: Overview of ASP.NET Core SignalROverview of ASP.NET Core SignalR, <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0>, [ ]

25: Cloud Computing Services | Microsoft Azure, <https://azure.microsoft.com/en-us>, []

26: What is a computer port? | Ports in networking | Cloudflare, <https://www.cloudflare.com/learning/network-layer/what-is-a-computer-port/>, []

27: ESP32WROOM32D & ESP32WROOM32U Datasheet, <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf>, []

28: MFRC522 Standard performance MIFARE and NTAG frontend, <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>, []

29: PN532 User Manual, <https://www.nxp.com/docs/en/user-guide/141520.pdf>, []

30: The Complete Guide to DIP Switches | RS, <https://uk.rs-online.com/web/content/discovery/ideas-and-advice/dip-switches-guide>, []

31: FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions, <https://www.freertos.org/>, [ ]

32: Tasks and Co-routines [Getting Started] - FreeRTOS, <https://www.freertos.org/taskandcr.html>, []

33: Public NTP  |  Google for Developers, <https://developers.google.com/time>, []

34: 4. Introducing NDEF - Beginning NFC [Book], <https://www.oreilly.com/library/view/beginning-nfc/9781449324094/ch04.html>, []

35: Unix Time Stamp - Epoch Converter, <https://www.unixtimestamp.com/>, []