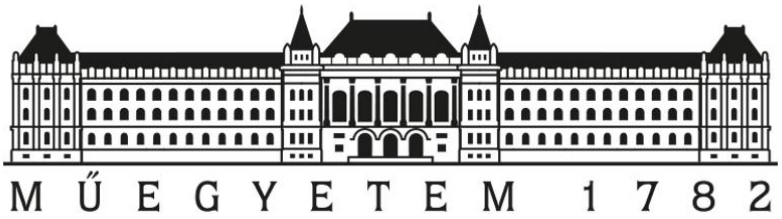


IoT Greenhouse

Önálló laboratórium beszámoló

Készítette: *Dancs Balázs*

Konzulens: *Schulcz Róbert*

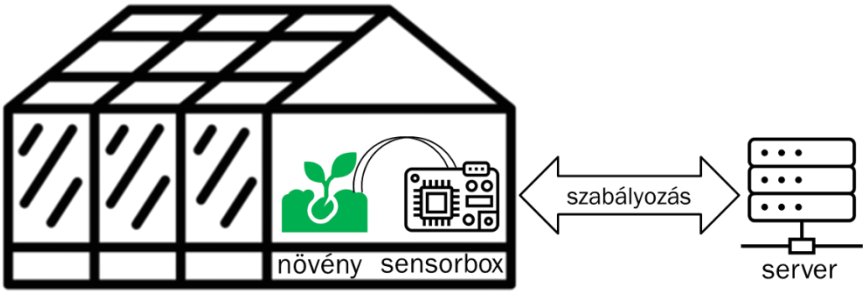
A projekt áttekintése

A projekt célja, hogy a növénytermesztéshez ideális körülmények megteremtését automatizálja. A hőmérséklet, a talajnedvesség, a CO2 szint, a páratartalom és a fényintenzitás mértékét, az üvegházban növényfajtánként elhelyezett szenzorok folyamatosan monitorozzák.

A termesztett növények fajtától függő igényei a szerver adatbázisában vannak tárolva. A sensorbox sorozatszáma alapján történik a sensor növénytípussal való összekapcsolása a szerveren. A sensorbox által küldött, mért értékek befutnak a szerverre és szükség esetén a szerver gondoskodik a beavatkozásról. Vezérlőjeleket küld vissza, melyek működésbe hozzák a szükséges eszközöket. A beavatkozás hatása a sensorboxból érkező folyamatos jelek alapján követhető.

A szabályozás az interneten keresztül történik, a szerver és az üvegház egymástól független helyszínen lehet. A felhasználói felületen grafikusan követhető a szabályozás folyamata.

Ebben a projektben az én feladatom a szabályozó szerver létrehozása, és a kezelő web felület elkészítése.



1. ábra IoT Greenhouse – működési vázlat

Tervezés

A felhasználói nézetek elkészítése előtt a felületekről mockup-okat készítettem. Ehhez felhasználtam ezt az internetes eszközt (https://balsamiq.cloud). A szerveroldali fejlesztéséhez Node.js-t választottam, mert könnyen skálázható, és lehetővé teszi a hatékony JavaScript alkalmazások készítését a szerveroldalon. Az NPM csomagok segítettek a funkcionalitás kiterjesztésében és az alkalmazások gyorsabb fejlesztésében. A permanens adattároláshoz a MongoDB-t választottam a NoSQL léte miatt, ami lehetővé teszi változó struktúrájú adatok

tárolását. A kommunikáció megvalósításához az MQTT protokollt választottam, mivel lehetővé teszi az egy-több kommunikációt és minimális sávszélességet, erőforrást igényel a működéséhez.

Megvalósítás

Felhasználói felület

A felhasználó számára az oldalak HTML tartalma sablonok felhasználásával generálódnak. Ehhez az EJS NPM csomagot használtam, ami minimális JavaScript írásával (for ciklus, if elágazás) készít kész HTML oldalakat, amibe beilleszti az adatbázisból vett és egyéb értékeket a megfelelő helyekre. Az esztétikus megjelenítés érdekében a Bootstrap CSS könyvtárat használtam fel. Ez mobil barát is, bár nem ez volt az elsődleges célom, de így mobil környezetben is jól néz ki. A diagrammok megvalósításához a Chart.js JavaScript könyvtárat használtam. Új adat érkezésekor beszúrom azt a megfelelő diagramba, valamint, ha már túl sok adatpont van egy diagrammon, akkor a régiek lekerülnek róla.

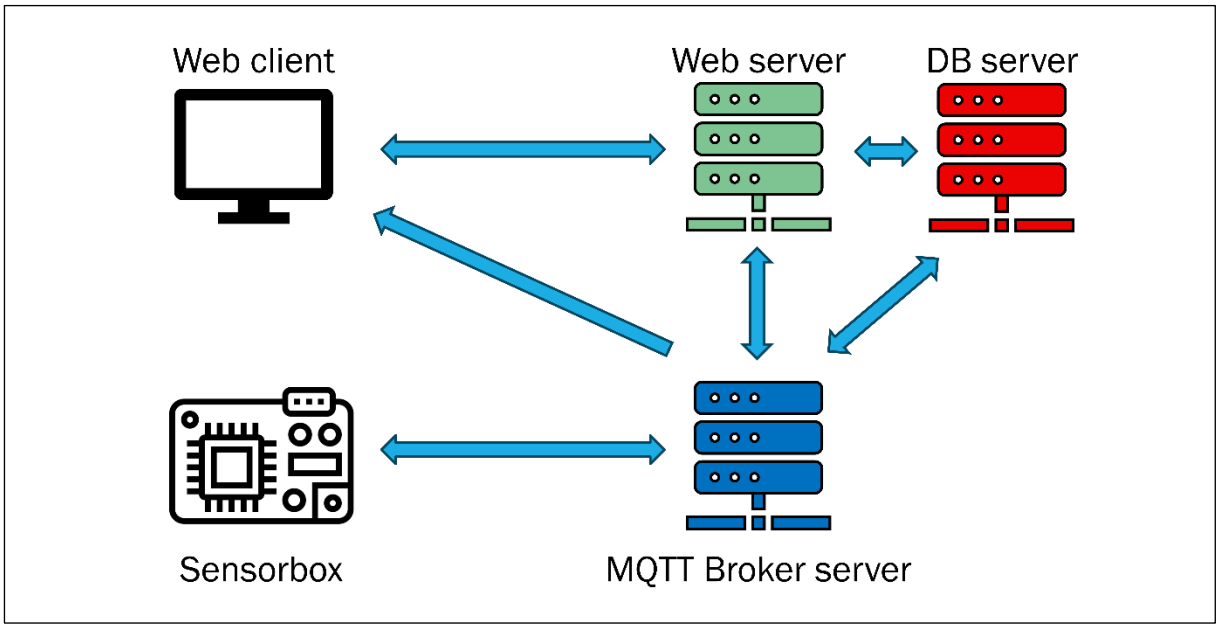
Kivétel kezelés

Ha valamilyen hiba keletkezik pl.: nem létező felhasználó, rossz jelszó vagy olyan zöldségfajta törlése amihez még sensorbox-ok vannak társítva, akkor erről visszajelzést adunk afelhasználónak.

Felhasználó kezelés

A felhasználókról eltároljuk a nevüket, e-mail címüket és jelszavukat SHA-512 kódolva. A bejelentkezéskor az érkező jelszó hash-ét hasonlítjuk az eltárolthoz. Sikeres bejelentkezés esetén egy 2 hétig működő Session ID-t küldünk a felhasználónak, így nem kell minden egyes művelethez újra hitelesítenie magát. A Session ID egy secure cookie-ban tárolódik és a kliens minden egyes kéréshez, ezt hozzá csatolja, hogy beazonosítsa magát a szerver felé.

A kommunikáció modellje



2. ábra Kommunikációs Modell

A sensorbox-ok az MQTT Broker szervernek küldik a mérési adataikat. A Web szerver ezekre fel van iratkozva, megkapja a broker-től az adatokat és ha azok nem megfelelőek az adatbázisban tárolt növényi igényeknek, akkor szabályozási üzenteket küld vissza. A web kliens csatlakozásakor betölti a weboldalt a Web szerverről, ami tartalmazza az adatbázisból kinyert adatokat. A betöltés után viszont nincs folyamatos kapcsolat a kliens és a Web szerver között, hanem a kliens közvetlen kapcsolatot épít ki az MQTT Broker-rel. A kliens is feliratkozik az MQTT topic-okra és megkapja a szenzor adatokat, amiket megjelenít a diagrammokon. Az MQTT Broker-hez való csatlakozáshoz szükséges hitelesítő adatok is az adatbázisban vannak tárolva.

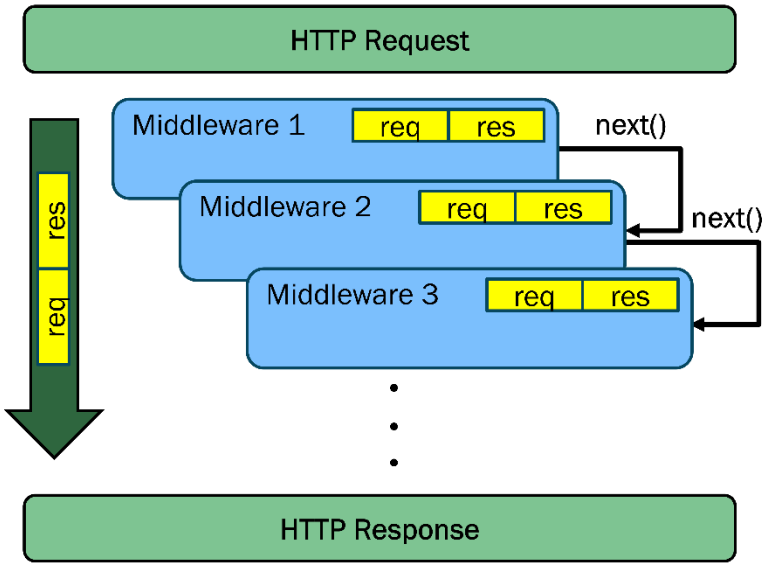
Az MQTT üzenetek JSON-t tartalmaznak a mérési adatokkal:

```
{
   "temp":23.1,
   "hum":45.3,
   "CO2":522.3,
   "moist":51.1
}
```

A megvalósítás során problémát okozott, hogy az MQTT Broker szerverhez használt JavaScript MQTT.js könyvtár, bár jól működött a Web szerveren és a virtuális sensorbox-okon, de nem működött kliens oldalon a böngészőben. Ennek az oka az volt, hogy a böngészők nem

támogatják az mqtt protokollt TLS felett a 8883 porton, csak a WebScoket kapcsolatokat. Az EMQX szoftverben engedélyeznem kellett az ehhez tartozó listener-t és ez megoldotta a problémát.

Middleware architektúra



3. ábra Middleware architektúra

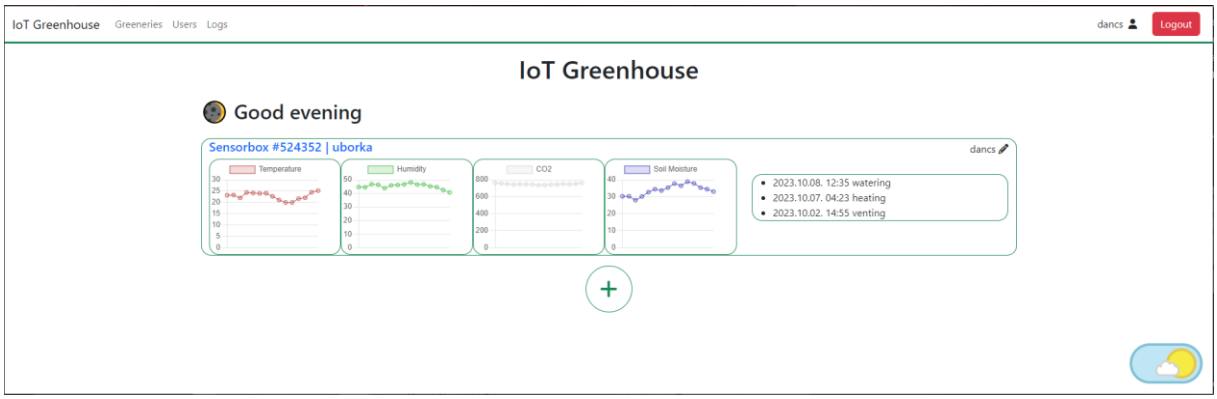
A middleware-ek olyan kódmodulok, amelyek a HTTP kérések és válaszok feldolgozását végzik az alkalmazás futása során. Ezek a modulok az alkalmazás endpoint-jainak hívása alkalmával hajtódnak végre, és lehetővé teszik különböző tevékenységek, mint például adatok előkészítése, adatbázis műveletek, hitelesítés vagy naplózás végrehajtását. A middleware-ek egymásra épülő láncokban dolgoznak, és a next függvény segítségével továbbítják a vezérlést a következő middleware-nek. Ez a tervezési minta javítja az alkalmazás modularitását, karbantarthatóságát és könnyű kiterjeszthetőségét.

A felhasználói web felület

Az IoT Greenhouse szoftver jelenleg a saját szerveremről, böngészőprogramon keresztül érhető el a https://greenhouse.dancs.org/ címen.

A felhasználó bejelentkezése felhasználói névhez és jelszóhoz kötött, így illetéktelenek nem tudnak beavatkozni a folyamatokba.

Bejelentkezés után a kezdő képernyőn megjelennek az üvegházban jelenleg termesztett, a szoftverben rögzített növények. Mindegyik esetében grafikonokon, valós időben nyomon követhetőek a sensorboxról érkező mért jelek, illetve láthatóak a szükséges beavatkozások időrendben.



4. ábra Felhasználói felület

A felhasználó kétféle megjelenítés (nappali/éjszakai mód) közül választhat. A választás localStorage-ban eltárolódik, oldal váltás esetén is megmarad.

A felhasználónak lehetősége van újabb felhasználók, sensorboxok és növények, illetve a nekik megfelelő igények felvételére. Természetesen ezeket később törölni is lehet az adatbázisból.

Tesztelés

A szoftver teszteléséhez írtam virtuális sensorbox-okat amik csupán véletlenszerű adatokat küldtek folyamatosan. Az adatok nem teljesen véletlenszerűek voltak, hanem csak egy bizonyos százalékban térhettek el az előzőtől, valamint volt egy alsó és felső határuk, amit nem léphettek át. Így egészen valósághű diagrammokat sikerült vizualizálni ezekkel az adatokkal.

Továbbfejlesztési lehetőségek

Web framework használata (pl. Angular)

Web keretrendszerek, mint például az Angular, előnyei közé tartozik a komponensalapú struktúra, az adatkötés és a teljesítményoptimalizáció, amelyek egyszerűsítik a fejlesztést, növelik az alkalmazás hatékonyságát és biztosítják a könnyebb karbantartást a manuálisan fejlesztett kliensoldali JavaScript kóddal szemben.

Mobil app

A mobil app jobb felhasználói élményt nyújtana, mint a weboldal megjelenítése egy böngészőben, még akkor is, ha az van mobilra optimalizálva. Push notification-öket lehetne küldeni a felhasználóknak, amik értesítenék őket az üvegház állapotáról.

JavaScript átírása TypeScript-re

A JavaScript kód átírása TypeScript-re több előnnyel járhatna. A TypeScript erős típusellenőrzést biztosít, ami csökkenti a hibák számát a fejlesztési folyamat során. A típusos kód segít jobban megérteni és dokumentálni a kódot, növelve ezzel a karbantarthatóságot és olvashatóságot.