ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Neumann János Informatikai Kar

SZAKDOLGOZAT

OE-NIK
2021.

Hallgató neve: **Kovács Csaba Rihard**
Hallgató törzskönyvi száma: **T/005696/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Kovács Csaba Rihard**
Törzskönyvi száma: T/005696/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Vertikális kert monitorozása és automatizálása
Monitoring and automation of vertical gardens

Intézményi konzulens: Simon-Nagy Gabriella
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Big Data és üzleti intelligencia specializáció

A feladat

Tervezzen és valósítson meg egy olyan szoftvert, amely lehetővé teszi vertikális kertek monitorozását és menedzselését. Kutassa fel és vizsgálja meg a feladat megoldásához használható technológiákat, beleértve az adatforrásként tipikusan előforduló szenzorokat (a hardveres kivitelezés nem része a feladatnak). Ismertesse a hasonló rendszerek működését és képességeit. Valósítson meg egy olyan tesztrendszert, amely lehetővé teszi mérési adatok (pl. hőmérséklet, nedvesség, különböző tápanyagok szintjei) begyűjtését szimulált szenzorok által. Implementáljon egy olyan interfészt is, amely beavatkozó jeleket tud közvetíteni a növények köré telepített berendezések felé (locsolás, fűtés, stb. szabályozása). Készítsen továbbá egy felületet, amely a növények állapotának monitorozása mellett lehetőséget nyújt manuális beavatkozás indítására, valamint automatikus beavatkozás konfigurálására is.

A dolgozatnak tartalmaznia kell:

- a feladat pontos ismertetését,
- a hasonló rendszerek működését és képességeit,
- a feladat megoldásához alkalmazható technológiák és adatforrások tömör bemutatását,
- a választott megoldások indoklását,
- a tervezés és megvalósítás részletes leírását,
- a tesztelés tervét és eredményeit,
- az elkészített rendszer eredményeinek részletes bemutatását és értékelését,
- a továbbfejlesztési lehetőségek számba vételét.

Ph.

FR

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat/diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban/diplomamunkában található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2021. 12. 14.

.....

hallgató aláírás

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun Kód: Tagozat: Nappali

Kovács Csaba Rihard..... [redacted].....

Telefon: Levelezési cím (pl.: lakcím):

[redacted].....[redacted].....

Szakdolgozat címe magyarul:

Vertikális kert monitorozása és automatizálása .....

Szakdolgozat címe angolul:

Monitoring and automation of vertical gardens.....

Intézményi konzulens: Külső konzulens:

SIMON-NAGY GABRIELLA

Alk.	Dátum	Tartalom	Aláírás
1.	**2021. 02. 26.**	**A téma meghatározása**	
2.	**2021. 03. 19.**	**A kutatási területek áttekintése**	
3.	**2021. 04. 16.**	**Az irodalomkutatás összegzése**	
4.	**2021. 05. 10.**	**A rendszerspecifikáció véglegesítése**	

A hallgató a Szakdolgozat I. tantárgy követelményét teljesítette, beszámolóra bocsátható.

.....

Intézményi konzulens

Budapest, 2021. május 14.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Kovács Csaba Rihard..... Neptun Kód: [redacted]..... Tagozat: Nappali .....

Telefon: [redacted]..... Levelezési cím (pl.: lakcím): [redacted].....

Szakdolgozat címe magyarul:

Vertikális kert monitorozása és automatizálása .....

Szakdolgozat címe angolul:

Monitoring and automation of vertical gardens.....

Intézményi konzulens: SIMON-NAGY GABRIELLA

Külső konzulens:

Alk.	Dátum	Tartalom	Aláírás
1.	**2021. 09. 21.**	**Megvalósítási lehetőségek áttekintése**	
2.	**2021. 10. 19.**	**Fejlesztési fázis ellenőrzése**	
3.	**2021. 11. 22.**	**Tesztelés ellenőrzése**	
4.	**2021. 12. 10.**	**Eredmények, összegzés**	

A hallgató a Szakdolgozat II. tantárgy követelményét teljesítette, védésre bocsátható.

.....

Intézményi konzulens

Budapest, 2021. december 10.

Tartalom

1. Bevezetés 7
2. Kertészeti háttér 9
2.1. Meglévő kertészeti megoldások 9
2.1.1. Kertészeti újítások az ókorban 9
2.1.2. Üvegházak 9
2.1.3 Kutatás a Nemzetközi Űrállomáson 10
2.1.4 Hidroponikus rendszerek 10
2.2. Növények igényei, viselkedései 12
2.3. Összegzés 13
3. Informatikai háttér 15
3.1. Szenzorok 15
3.1.1 Levegő hőmérséklete és páratartalma 15
3.1.2 Víz hőmérséklete és nedvességi szint 15
3.1.3 Levegő széndioxid szintje 16
3.1.4 Tápanyag szint és pH érték 16
3.1.5 Fényerősség 16
3.2. Nagy mennyiségű adat tárolása 17
3.2.1 Nagy mennyiség kérdése 17
3.2.2. SQL és NoSQL rendszerek 18
3.3. Monitorozásra alkalmas felület 18
3.4. Összekötő rendszer 19
4. Specifikáció 21
4.1. Automatizmus 21
4.2. Használati esetek 21
4.3. Adatok feldolgozása és tárolása 22
5. Rendszerterv 23
5.1. Rendszer részei 23
5.1.1. Vertikális kert komponens 23
5.1.2. Monitorozó felületet biztosító komponens 24
5.1.3. Szerver komponens 24
5.2. Választott megoldások 27

6. Implementáció 29
6.1. Szimulációs oldal fejlesztése 29
6.1.1. Fájl olvasása és írása 29
6.1.2. Növények változtatása 31
6.1.3. Adatok küldése és kommunikáció a rendszer szerver oldalával 32
6.2. Backend fejlesztése 33
6.2.1. MongoDB adatbázis 33
6.2.2. Socket.IO telepítése, előkészítése, és működtetése 39
6.2.3. Növények gondozását végző automatizmus 39
6.3. Frontend fejlesztése 40
6.3.1. Felhasználói felület létrehozása és külalakja 40
6.3.1. Felhasználói felület funkciói 41
7. Tesztelés 47
7.1. Tesztelési tervek 47
7.2.2. Szimulációs oldalhoz tartozó tesztek 47
7.2.3. Backend oldalhoz tartozó tesztek 47
7.2.4. Frontend oldalhoz tartozó tesztek 48
7.2. Tesztelési eredmények 49
7.2.2. Szimulációs oldalhoz tartozó teszt eredmények 49
7.2.3. Backend oldalhoz tartozó teszt eredmények 50
7.2.4. Frontend oldalhoz tartozó teszt eredmények 50
8. Továbbfejlesztési lehetőségek 52
9. Összefoglalás 53
10. Summary 54
Irodalomjegyzék 55

1. Bevezetés

Szakdolgozatom témája egy olyan rendszer kialakítása, amely képes egy nagy mennyiségű növénnyel rendelkező vertikális kert monitorozására és automatizálására. A témaválasztásomat több tényező is indokolja.

Az egyik, és talán legegyszerűbb indok, hogy kiskorom óta érdekel a kertészet, a növények, állatok, és tulajdonképpen az egész természetes élővilágunk. A kertészet iránti érdeklődésem talán abból is ered, hogy olyan családban nőttem fel, ahol az önellátó gazdálkodás és kertészkedés egy teljesen általános és mindennapi tevékenység volt, emiatt már kiskoromban is sokat tanítottak felmenőim, a különböző növényekről. Bár régen létkérdés volt, mára inkább hobbiként foglalkozom kertészkedéssel. Erkélyemen különböző csípős paprikákat szoktam ültetni, és azok keresztezésével is foglalkozom.

Ugyan akkor a természet mellett, az emberiség fejlődése is egy érdekes témakör számomra. Mindig is érdekeltek a különböző technológiai újítások, az olyan fejlesztések, amelyek pozitív irányba mozdítják előre az emberiséget. Lehet bár naiv gondolat, de személyes véleményem, hogy létezhet egy olyan jövő, amelyben a természet és az emberiség harmóniában él egymással oly módon, hogy az nem hátráltatja az emberi fejlődést, és emellett ápolja környezetünket, hiszen a természet, a földünk egészsége a mi érdekünk is.

Sajnos sokszor láttunk rá példát, hogy az emberi fejlődést csak a természet károsítása mellett tudtuk véghez vinni. Szerencsére úgy tűnik, hogy egyre nagyobb teret hódítanak azok az új technológiai megoldások, amelyek eltalálják azt a szűkkeresztmetszetet, amelyben nem csak az emberi előrehaladás lehetséges, de környezetünk sem károsodik belőle, sőt akár kifejezetten hasznára is válhat. Úgy gondolom, hogy szakdolgozatom is az utóbbira mutathat egy lehetséges példát.

De miről is van szó konkrétan? A világon több helyen példát láthatunk nagy mértékű erdő írtásokra, amelyeknek különböző okai lehetnek, de talán az egyik legnagyobb oka a mezőgazdaság terjeszkedése. Természetesen az emberiséget etetni kell, és mivel egyre több ember van, egyre több helyre lesz szüksége a mezőgazdaságnak is. Ha tudnánk találni egy olyan élelmiszer termelési módszert, amely kevés helyet igényel, azonban mégis nagy mennyiségű zöldséget, gyümölcsöt tudna termelni, akkor azzal nem csak megoldanánk a növekvő populáció élelmezését, de ugyan akkor a természetnek is több helyet hagyhatunk.

Egy ilyen termelési módszer például egy vertikális kert kiépítése. A vertikális kert egy olyan kertészeti megoldás, amely során nem a hagyományos horizontális vetési módszereket alkalmazzuk, amelyek során nagy területen elhelyezkedő szántóföldekre van szükség, hanem egy magas torony szerű épületben növesszük a növényeket. Ez persze több kihívást is felvet. Többek között: Hogyan oldjuk meg a növény fényellátását,

ha fölötte egy emelet van? Hogyan oldjuk meg a locsolást? Mibe ültetjük a növényeket és hogyan? Az elmúlt évek technológiai fejlődései egyre inkább azt bizonyítják be, hogy ezekre a kérdésekre tudunk válaszolni, és a közeljövőben akár ténylegesen megvalósítható lenne egy hasonló projekt. Természetesen kompromisszumok is vannak, amelyekre a későbbiekben részletesebben is kitérek.

Mivel a projekt olyan elemekből is áll, amelyekhez nagy mennyiségű tőkére lenne szükség ahhoz, hogy teljes valójában megvalósítsam, ezért bizonyos részeit szimulációval fogom helyettesíteni. Ezek a tőke igényes elemek, amiket nem fogok tudni megvalósítani a következők: Egy elég nagy épület megépítése, amely nagy mennyiségű növény tárolására képes, illetve a növények gondozásához tartozó infrastruktúra (gondolok itt például az öntözőrendszerre), és nagy mennyiségű szenzor, amelyek a növényekről, és a környezetükről küldenek információkat. Ennek ellenére szeretném elérni, hogy a projekt a lehető legrealisztikusabb legyen, ezért részletes irodalomkutatást végzek a növények igényeiről, és viselkedéseiről, valamint a szenzorok működéséről, hogy az ezeket pótló szimuláció a lehető legpontosabb legyen.

Szakdolgozatomat 7 nagyobb fejezetre osztottam a következők szerint: Írni fogok a projekt kertészeti oldaláról, amiben már meglévő különböző megoldásokat vizsgálok meg, azok előnyeivel és hátrányaival együtt, illetve a növények szimulációhoz szükséges információit itt fogom részletezni. Bővebben írni fogok a projekt informatikai oldaláról, amelyben alterületekre osztom a projektet és mindegyik alterületnél különböző lehetséges informatikai megoldásokat tekintünk meg. Többek között itt lesz szó a szenzorok szimulálásához szükséges fontos információkról. Utána a rendszer részeit, és azok feladatait, funkcióit és lehetőségeit specifikálom jobban. Majd ismertetem, hogy az egyes funkciók elvégzésére milyen konkrét megoldásokat fogok alkalmazni, valamint, hogy a különböző elemek hogyan fognak együttműködni, és egy komplett rendszert alkotni. Bemutatom a rendszer konkrét implementációját, utána tesztelésekről és azok eredményeiről fogok írni. Végül egy összefoglalóval zárom a szakdolgozatot, amiben kitérek arra is, hogy a későbbiekben hogyan szeretném továbbfejleszteni a projektet.

Szeretném megragadni az alkalmat, hogy köszönetet mondjak konzulensemnek Simon-Nagy Gabriellának amiért segített a szakdolgozat témájának kialakításában, hasznosabbnál hasznosabb információkkal segítette munkámat, és amiért végtelenül türelmes, megértő, és segítőkész volt.

2. Kertészeti háttér

2.1. Meglévő kertészeti megoldások

2.1.1. Kertészeti újítások az ókorban

Ahogy haladunk előre az időben, a klasszikus horizontális mezőgazdasági földműveléstől, egyre több újabb megoldás lát napvilágot. Azonban ez nem feltétlenül csak modern időnkre igaz. Már az ókorban is törekedtek arra, hogy a szokványostól eltérő kertészeti formákat alkossanak meg. Ez főleg az észak-afrikai, és közel keleti tájegységekre igaz, ahol nagyobb szárazság, és több kopár sziklás vidék található. A vertikális kertek, valamint a tető kertek koncepcióját már az ókori perzsiai-, és egyiptomi birodalomban is előszeretettel használták. Azonban az is megjegyzendő, hogy nagy mennyiségű élelmiszer termelés helyett, inkább dekoratív szerepeket töltöttek be, illetve a környezetük hőmérsékletének csökkentésére használták, mivel a sok zöld növény, amelyeket ezekbe a kertekbe ültettek tökéletesek voltak árnyékolási célokra, és hatékonyan csökkentették az épületek hőmérsékletét.

2.1.2. Üvegházak

Az üvegházak célja egy olyan fedett elszigetelt tér kialakítása, amelynek belső klímáját irányítani tudjuk. Már az ókori rómaiak is képesek voltak kezdetleges „üvegházakat” alkotni, amelyekben olyan zöldségeket és gyümölcsöket tudtak növeszteni, amelyeknek a helyi klíma nem lett volna megfelelő. Eredeti funkcionalitásában bár csak a hőmérséklet, és a páratartalom irányítása a főbb szempont, azonban magyar elnevezése az „üveg” szóval már magában foglalja a fényforrás jellegét is, ami egy fontos szempont, mivel eredetileg az üvegházak (angolul: Greenhouse) akár természetes napfény megléte nélkül is „üvegháznak” minősülnek. Ez felveti a természetes, illetve mesterséges fényforrás kérdését is. Az egyszerűség kedvéért a továbbiakban az üvegházat magyar értelmezése szerint vesszük, tehát elsődleges fényforrásnak a napfényt tekintjük.

A fentieket figyelembe véve az üvegház előnye, hogy elzárt környezetet alkotva, hatékonyabban tudjuk a növények környezetét vizsgálni. A vizsgálatot végezhetjük analóg hőmérőkkel és páratartalom mérőkkel, azonban, ha automatizálni, illetve digitális eredményeket szeretnénk begyűjteni, célszerűbb szenzorokat alkalmazni. A DHT11 nevű szenzor egyszerre alkalmas hőmérséklet és páratartalom információk begyűjtésére. A szenzorok működését a 3.1-es fejezetben részletezem.

Hátrányuk viszont, hogy ugyan akkora területet foglalnak el, mint a klasszikus szabadtéri kertek, mivel a növények az üvegházban is napfényre vannak hagyatkozva, ezért, ha emeleteket szeretnénk tenni az üvegházba azt csak akkor tehetjük meg ha az alacsonyabb szinteken, ahova kevesebb fény jut, árnyékot kedvelő növényeket tartunk, de még így is sajnos nagyon limitált, hogy hány emeletet tehetünk, hiszen, ha több emelet van, akkor az alsóbb szintekre még kevesebb fény jut. Szintén egy fontos szempont a költsége az üvegháznak. Egy nagy területen elhelyezkedő, akár még emeletekkel is rendelkező üvegháznak gyorsan növekvő költségei vannak, hiszen rengeteg üvegre van szükség, amelynek ráadásul elég erősnek kell lennie, hogy külső tényezőknek, mint például jégesőnek is ellenálljon.

2.1.3 Kutatás a Nemzetközi Űrállomáson

2014 május 8.-án el lett ültetve az első olyan növény, amely nem földünkön lett elültetve, hanem az űrben, pontosabban a Nemzetközi Űrállomás VEGGIE nevű moduljában. A VEGGIE (The Vegetable Production System) egy olyan modul, amelyben növények termesztésével, fejlődésével kapcsolatos kutatásokat végeznek. A kutatások többek között kitérnek arra, hogy milyen hatással van a súlytalanság a növények fejlődésére, valamint, hogy hogyan lehetséges mesterséges fénnyel növényeket termeszteni. A szakdolgozat szempontjából most a mesterséges fénnyel kapcsolatos információk a relevánsabbak, így azokat fogom részletezni a továbbiakban.

A beérkező kutatási eredmények meglepőek. Kiderült, hogy a növények a teljes fényspektrum helyett, annak csak egy kis frekvenciáját hasznosítják igazán. Egyik ilyen tartomány a látható fény spektrum 400 – 500 (nm) közötti hullámhossza. Ezt a frekvenciát az emberi szem a piros színként érzékeli. Bár a fényspektrumnak ezt a tartományát hasznosítják a leginkább, emellett a kék fénynek is van szerepe a növények egészséges fejlődésében. Ez körülbelül 700 nm hullámhossznál található. A zöld fényt hasznosítják a legkevésbé. Ez azt jelenti, hogy az emberi szem számára lila színként ismert fénnyel a legoptimálisabb a növény fejlődése. A mesterséges fényforrásaként talán a legcélszerűbb egy programozható LED (Light Emitting Diode) használata. Mivel ezeknek a diódáknak RGB kódolással lehet a színét állítani, körülbelül RGB (255, 0, 118) színre van szükség.

Egy másik fontos tényező a mesterséges fénnyel kapcsolatban, hogy bolygónk növényei az evolúció során, ahhoz szoktak hozzá, hogy magas fényű, és alacsony fényű periódusok váltogatják egymást. Tehát fontos, hogy a LED megvilágítás körülbelül 16 órán át tartson, és legyen 8 óra szünet, amely során a növény pihenni tud.

2.1.4 Hidroponikus rendszerek

Eddig olyan rendszerekről volt szó, amelyek a növények hőmérséklet, páratartalom, és fény igényeire nyújtottak megoldásokat. Ebben a részben egy olyan rendszerről lesz szó, amely a növények folyadék-, és tápanyag szükségleteit elégíti ki hatékonyan. Földbe ültetett növények esetében a gyökerek nem tudnak igényüknek megfelelően tápanyagot és nedvességet felvenni. mivel egy locsolás alkalmával hirtelen nagy mennyiségű víz érkezik, amelynek nagy része nem is fog a gyökerekkel kapcsolatba lépni, mert elpárolog a talajból, vagy túl mélyre szivárog, ahonnan a gyökerek már nem érik el. Ezen kívül a gyökereiket szélesen szét kell terjesszék a talajban ahhoz, hogy a lehető legnagyobb eséllyel vegyenek fel tápanyagot. Ennek egyik hátránya, hogy sok helyet igényel, másik hátránya, hogy sokkal nehezebb irányítani, hogy pontosan mennyi és milyen tápanyagot vesz fel a növény. Hasonlóan, mint ahogy az üvegház egy jól irányítható zárt rendszert alkot a hőmérséklet és páratartalom szempontjából, úgy ebben az esetben is tudunk zárt rendszert alkotni a nedvesség és tápanyag szempontjából. Erre megoldás az úgynevezett hidroponikus rendszer, amelynek több változata is van, de ami mindegyikben közös, hogy a növény gyökérzete egy zárt rendszerben található, amelyben a növény számára hasznos tápanyagokkal összekevert víz található. A különböző hidroponikus megoldások abban térnek el egymástól, hogy miben van stabilizálva a növény gyökere. Az úgynevezett „Raft Culture" egy olyan megoldás, amelyben a növényt egy víz felszínén úszó tutajhoz erősítik úgy, hogy a gyökerei az alatta lévő vízbe lógjanak. Előnye, hogy olcsó és jól kezelhető megoldás. Hátránya viszont, hogy kevés növénynél lehet ezt az eljárást alkalmazni. Leggyakrabban szükség van valamilyen közegre, ami a növény gyökereit tartja, és a közeggel együtt lógatjuk bele egy olyan cső rendszerbe, amelyben folyamatosan tápos vizet keringetünk. A közegnek a típusa függ attól, hogy milyen növényt ültetünk bele, hiszen növénytől függően különböző mértékű nedvesség felszívási képességre van szükség.

A gyökereket támasztó közegek előnye, hogy szenzorokat tudunk belehelyezni, amelyek pontos információkat adhatnak arról, hogy milyen nedvességi szintekkel rendelkeznek az adott növény gyökerei. A nedvességi szint mérése mellett NPK (Nitrogén, Foszfor, Kálium) szenzorral a növények számára legfontosabb tápanyagok szintjét is mérni tudjuk. A vízben lévő tápanyagok miatt fontos, hogy minimum naponta egyszer mérjük annak ph értékét, amire szintén van alkalmas szenzor.

Összesítve a hidroponikus kertészet előnyei a hagyományossal szemben, hogy sokkal kevesebb helyet igényel, a talaj hiánya miatt sok károkozótól és betegségtől megkíméli a növényt, sokkal kevesebb vízre van szükség, mivel a növény pontosan annyi tápanyagot és nedvességet vesz fel amennyire szüksége van, a maradék víz pedig újrahasznosul a rendszerben, és végül, de nem utoljára, mivel zárt rendszerről van szó, hatékonyan tudunk szenzorokkal méréseket végezni, és hatással lenni a rendszerre.

2.2. Növények igényei, viselkedései

Mivel nagy mennyiségű növény szenzorokkal való megfigyelése kifejezetten pénz igényes projekt, ezért a szakdolgozat ezen részét szimulációval oldom meg. Ebben a részben a növények viselkedésének szimulációjához szükséges háttértudást összegzem.

A növényeknek különböző igényeik vannak, amiket számításba kell venni. Mivel a növények egyik legfontosabb energia szerzési forrása a fény, ezért fontos megállapítani, hogy pontosan mennyi fényre van szükség. A legtöbb zöldség, gyümölcs, és fűszernövénynek minimum 2000 lumenre van szüksége ahhoz, hogy fotoszintetizáló képességének 100%-át kitudja használni. Ahogy azt fentebb a mesterséges fény kapcsán is említettem, azt is fontos számításba venni, hogy a legtöbb növénynek 16 óra aktív fotoszintetizálás után, 8 óra pihenési időre is szüksége van, amikor nem kap semmilyen fényt.

A hőmérséklet szintén egy fontos szempont. A növényeket ebből a szempontból két nagyobb csoportba soroljuk. Vannak a hűvös évszakokat kedvelők, és vannak a melegebb évszakokat kedvelő növények. Általánosságban elmondható, hogy a hűvösebb évszakokat kedvelő növényeknek éjszaka 10 – 15 °C-ra, nappal pedig 16 – 22 °C-ra van szükségük, míg a melegebb évszakokat kedvelő növényeknek éjszaka 18 °C-ra, nappal pedig 24 – 27 °C-ra van szükségük.

A levegő hőmérséklete mellett annak összetételére is figyelnünk kell. A fotoszintézis gördülékeny lejátszódásához széndioxidra (CO_2) és aránylag magas páratartalomra van szükség. A széndioxid szint körülbelül 500 ppm (parts per million) és 1000 ppm között kell legyen ahhoz, hogy a fotoszintézis beindulhasson a növényekben, míg a relatív páratartalom (RH: Relative Humidity) 75% környékén a legoptimálisabb. Kutatások azonban megállapították, hogy 800 ppm és 1200 ppm közötti széndioxid szintek akár 20%-kal is növelni tudják a termés mennyiségét. Összehasonlításképpen az átlagos széndioxid tartalom az atmoszférában 400 ppm, zárt helységben 350 – 450 ppm, a legnagyobb kényelmesen elfogadható széndioxid szint 1000 ppm, és 2000 ppm felett már veszélyesnek minősül. Az egyik legegyszerűbb és legolcsóbb eszköz széndioxid generálására, egy speciális komposztáló edény használata, amely gomba komposztálásával enged széndioxidot a levegőbe körülbelül 1200 ppm és 1500 ppm között. Mivel ez kifejezetten magas érték, ezért célszerű először egy tartályba generálni a széndioxidot, amelyben eltároljuk, és később szükség esetén felhasználjuk.

A legtöbb nagyobb komplexitással rendelkező növénynek egész hasonló tápanyag felvételre van szüksége. Természetesen minimális eltérések lehetnek a különböző fajok között, de általánosságban elmondható, hogy 16 fő alkotóelemből épülnek fel, melyeket két csoportba sorolunk, úgy nevezett makró- és mikró elemekre. Az alábbi táblázat összegzi, hogy melyik elem körülbelül mekkora részét képezi egy adott növénynek.

Alkotóelem	Jelölés	Elérhető formátum	Koncentráció (%)
Hidrogén	H	H_2O	6
Szén	C	CO_2	45
Oxigén	O	O_2, H_2O	45
Makró elemek			
Nitrogén	N	NO_3, NH_4	1,5
Kálium	K	K	1
Kalcium	Ca	Ca	0,5
Magnézium	Mg	Mg	0,2
Foszfor	P	H_2PO_4, HPO_4	0,2
Kén	S	SO_4	0,1
Mikró elemek			
Klór	Cl	CL	0,01
Bór	B	BO_3, B_4O_7	0,002
Vas	Fe	Fe^{+++}	0,01
Mangán	Mn	Mn^{++}	0,005
Cink	Zn	Zn^{++}	0,002
Réz	Cu	Cu^{++}	0,0006
Molibdén	Mo	MoO_4	0,00001

2.1. Táblázat: Növények főbb alkotó elemei. Forrás: [13]

Minket elsősorban jelenleg a nitrogén, kálium és foszfor érdekel hiszen azokat az elemeket tudjuk szenzorokkal mérni. A tápanyagokat a vízbe keverve kell keringetni a rendszerben. Az ilyen tápos víznek kettő fontos tulajdonságát kell még figyelembe vegyük. Az egyik ilyen tulajdonság a víz hőmérséklete. A legtöbb növény 18-21 °C körüli hőmérsékletet kedvel. Másik fontos tulajdonság a víz pH értéke, amit a különböző hozzáadott tápanyagok erősen befolyásolni tudnak. A legtöbb növény 6,0 és 7,0 közötti pH értéket kedvel, azonban ezt kifejezetten fontos növényenként külön kezelni mivel vannak olyan növények, amelyek ettől jelentősen eltérhetnek, mint például a káposzta, ami az 5,5 és 5,8 közötti pH értéket kedveli a legjobban.

Ahogyan a pH érték példájában is láthattuk, vannak növények, amelyek különös figyelmet igényelnek. Fontos minden növénynél részletesen külön specifikálni az egyes igényeket. A különböző növények igényeinek specifikus értékeit a későbbiekben fogom részletezni.

2.3. Összegzés

Az előző alfejezetekben ismertettem a különböző kertészeti technológiákat, melyek összekombinálásából kaphatunk egy elszigetelt, és ezáltal jól felügyelhető és automatizálható környezetet. A hidroponikus technológiával kis helyen sok növényt, kevés víz használatával, jól felügyelhető módon tudunk növeszteni, a lila színű mesterséges fényt felhasználva nem kell a napfényre hagyatkoznunk, és nagy mértékű vertikalitást tudunk bevezetni a rendszerbe, hiszen nem számít, hogy az emeletek mennyire árnyékolják be az alattuk lévő szinteket, és ha üvegházakhoz hasonlóan ezt mind egy jól elszigetelt környezetbe tesszük akkor a hőmérsékletet és páratartalmat is kedvünk szerint tudjuk befolyásolni, akár szintenként különböző mesterséges klímákat kialakítva, hogy az adott szinten lévő növényeknek a lehető legoptimálisabb körülmények legyenek elérhetőek.

3. Informatikai háttér

3.1. Szenzorok

Akárcsak a növények viselkedéseit, az azokat vizsgáló szenzorokat is szimulációval oldom meg. Ebben a fejezetben leírom, hogy milyen szenzorokkal lehetséges a fentebb tárgyalt fontos növényi és környezeti tulajdonságokat vizsgálni, és ezek az eszközök hogyan működnek, milyen adatokat szolgálnak nekünk. Ezeket az adatokat fogjuk a későbbiekben felhasználni a projekt monitorozás és automatizálás részéhez.

3.1.1 Levegő hőmérséklete és páratartalma

Erről a két környezeti tulajdonságról azért írok egyszerre, mert a szenzorok melyek ezen tulajdonságok mérésére alkalmasak, sokszor közösen egy összetett szenzort alkotnak. Ilyen például a DHT11 nevű szenzor is. A hőmérsékletet 0 °C és 50 °C között tudja mérni ±2 °C pontossággal. Az output mértékegysége szintén °C.

A páratartalom (Angolul: Relative Humidity) mérésére alkalmas része százalékban adja meg a mérés eredményét. 20 % és 90% közötti relatív páratartalmat tud érzékelni az eszköz, ±5% pontossággal.

3.1.2 Víz hőmérséklete és nedvességi szint

Bár a DHT11 nevű szenzor tud hőmérsékletet mérni, nem alkalmas víz hőmérsékletének mérésére, mivel nem vízálló. A DS18B20 vízállóságának köszönhetően tökéletes erre a célra. Az output mértékegysége °C, tartománya pedig -55 °C és +125 °C. Ez a tartomány tökéletesen megfelel, mivel a növények által kedvelt víz hőmérséklete általánosan 18 – 21 °C közé esik.

A nedvességi szintet kétfajta szenzorral tudjuk mérni. Ellenállás alapú szenzor, és kapacitív szenzor. Mindkét szenzor kimenete 0 – 3 Volt közötti egyenáramú jel, amit egy Arduino segítségével, egy 0 és 1023 közötti számmá tudunk alakítani. Ha a szenzorral a levegőből próbálunk mintát venni akkor az értéke 620 míg, ha vízből próbálunk mintát venni 310. Ez azt jelenti, hogy ez a két szám a két szélső érték, ami a gyakorlatban előfordulhat. Ennek a két szélsőértéknek a segítségével megadhatjuk a nedvességi szintet százalékban mérve, ami a mi esetünkben hasznosabb, mivel a növények nedvességi igényeit is százalékos formában ismerjük.

3.1.3 Levegő széndioxid szintje

A levegő hőmérséklete és páratartalma mellett a széndioxid szintje is egy fontos tényezője mivel a fotoszintézis egy fontos alapfeltétele. Értéke nem szabad túl alacsony legyen különben nem fog elindulni a fotoszintézis folyamata, a túl magas szintek pedig emberek számára veszélyesek. Az általunk mérni kívánt tartomány 400 ppm és 2000 ppm közé esik. Az MQ135 nevű szenzor egy népszerű megoldás a levegő minőségének ellenőrzésére azonban mérési tartománya csak 10 ppm és 1000 ppm közé esik, ami általános otthoni használatra elég lehet, de számunkra nem megfelelő. A CRIP M1 nevű szenzor 400 ppm és 2000 ppm között képes mérni, ±40 ppm pontossággal pontosan az általunk kívánt tartományt fedi le. Az output mértékegysége ppm.

3.1.4 Tápanyag szint és pH érték

A vízhez hozzáadott tápanyagokkal tudjuk kielégíteni a növények Nitrogén, Foszfor, és Kálium igényeit. Ha ezt szeretnénk automatizálni az előbb említett elemek mérésére is szükségünk van. Szerencsére pont erre a célra fejlesztették ki az NPK szenzort. A Traidacent RS485 NPK szenzor például megfelelő lehet erre a célra. Output mértékegysége mg/kg. Mérési tartománya 0 mg/kg és 1999 mg/kg közé esik. Az NPK szenzor a gyökereket tartó közegbe szúrva méri a tápanyag szintet.

A tápanyagok hozzáadásával, változik a víz pH értéke is. Ezért fontos, hogy gyakran ellenőrizzük a pH szintet is. Erre egy alkalmas eszköz lehet a GAOHOU PH0-14 nevű szenzor, amely 0 és 14 közötti pH szintek mérésére alkalmas.

3.1.5 Fényerősség

A vertikális kertek alapvetően nincsenek napfényre hagyatkozva, mivel azt feltételezzük, hogy a különböző szinteken lévő növények ugyanannyira sötét árnyékban vannak, és ugyan akkora mértékű mesterséges megvilágítást igényelnek. Azonban egy optimalizálási lehetőség lehet, ha a vertikális kert oldala üvegfalból készül, hasonlóan, mint egy üvegház esetében. Tovább növelhetjük a természetes napfény bejutásának mértékét, ha az egyes emeleteken gondosan elhelyezett fényvisszaverő tükröket telepítünk. Bár sokkal több napfényt tudunk így bejuttatni az egyes emeletekre, még így sem elég, hogy önmagában ellássa kellő mennyiségű napfénnyel a növényeket, illetve figyelembe kell venni a váltakozó időjárás okozta fényszint ingadozásokat is.

Ha fényérzékelő szenzorokat alkalmazunk, akkor tovább tudjuk optimalizálni a rendszert. Ahol a szenzor azt érzékeli, hogy nincs kellő mennyiségű fény, ott a mesterséges

megvilágítás rásegít a szükséges mennyiséggel. Az fénymennyiség optimalizálása mellett, azért is hasznos a fényérzékelő szenzorok alkalmazása, mert így kiszűrhetjük az esetlegesen meghibásodott fényforrásokat. Egy erre alkalmas eszköz például az SSR-L16 szenzor, amely kimenetének mértékegysége lux, valamint mérési tartománya 0 és 65535 lux közé esik.

3.2. Nagy mennyiségű adat tárolása

3.2.1 Nagy mennyiség kérdése

Mivel sok növény van a vertikális kertben, amit sok szenzor vizsgál, nagy mennyiségű adatról kell eldönteni, hogy mi történjen velük. Mielőtt a monitorozó rendszernek továbbítódnának az adatok, hogy kiírják a felhasználó számára a jelenlegi mérések eredményét, célszerű lenne először valamilyen kezelhető formában eltárolni azokat. Ennek egyik előnye, hogy így lehetőségünk adódik az adatokat szükség szerint átalakítani, illetve több különböző felhasználásra is alkalmazhatjuk őket. Például a felhasználó számára kiírjuk a jelenlegi mérési eredményeket, ezen kívül a korábbi adatok felhasználásával gráfokat és statisztikákat tudunk alkotni, melyek segítségével a felhasználó könnyebben átláthatja az elmúlt mérések eredményeit és ezekből következtetéseket vonhat le, de az automatizmus számára is szükséges, hogy valamilyen feldolgozható formában adjuk át az adatokat.

Tehát megállapítható, hogy fontos az adatok strukturált tárolása. Először is meg kell értenünk, hogy milyen adatokkal dolgozunk. Tekintsük meg a következő három szempontot: Az adatok feldolgozási, érkezési sebessége, az adatok mérete, mennyisége, végül az adatok strukturáltsága.

A sebesség szempontjából érdemes átgondolni a technikai limitációkat, mint például, hogy milyen közegben végezzük az adatszállítást a szenzorok és a feldolgozó egység között. Mivel ez a része a projektnek szimulációval készül ezért ez most kevésbé releváns, de különben egy fontos kérdés lehet. A jelenlegi esetben a fontosabb kérdés, hogy mekkora adatküldési, illetve fogadási sebességre van szükség? Mivel növények és környezetük tulajdonságait mérjük és küldjük tovább adatok formájában, elmondhatjuk, hogy ezt alacsony sebességgel is elvégezhetjük, mivel nem jellemző, hogy a hőmérséklet, pH érték, tápanyagtartalom vagy a nedvességi szint gyorsan változna.

Vizsgáljuk meg az adatok mennyiségének szempontjából. Mivel nagy mennyiségű növénynek több tulajdonságáról, illetve környezetük tulajdonágairól gyűjtünk információkat, ezért elmondhatjuk, hogy nagy adat mennyiségről van szó. Az adatok külön-külön nem nagyok méretüket tekintve, azonban sok különböző adat érkezik a szenzorok felől. Erre érdemes külön figyelmet fordítani a továbbiakban.

A strukturáltságot tekintve három esetet különböztetünk meg. Strukturálatlan adat, félig strukturált adat, és strukturált adat. A szenzorok felől strukturálatlan adatok érkeznek, amiket össze kell gyűjtenünk és megfelelően kell rendszereznünk. Például érdemes lehet az egy adott növénnyel kapcsolatos információkat egy helyen tárolnunk.

3.2.2. SQL és NoSQL rendszerek

Az adatbázis kezelő alkalmazásoknak két nagy csoportja van. SQL (Structured Query Language) és NoSQL adatbázis rendszerek. Az SQL alapú rendszerek relációs adatbázisokat kezelnek melyekben szigorúan strukturált adatok vannak táblák formájában tárolva, míg a NoSQL adatbázis kezelők nem relációsak, és félig strukturált adatok tárolására is alkalmasak. A relációs tábla alapú modell helyett különböző megoldásokat nyújtanak.

A dokumentum alapú rendszerek, dokumentumokban (jellemzően JSON vagy BSON fájlokban) tárolnak adatokat. A dokumentumokra úgy is tekinthetünk, mint objektumokra, amelyekben az objektum tulajdonságai vannak tárolva. Ezt a szemléletmódot használva könnyen tudunk kapcsolatot teremteni az adatbázis kezelő rendszer és objektum orientált programozási nyelvek között. Ami fontos szempont, ha automatizmust szeretnénk létrehozni.

Az oszlop alapú rendszerek rekordokat tárolnak táblákban hasonlóan a relációs adatbázisokhoz, azonban lehetővé teszik az oszlopok (attribútumok) dinamikus változtatását.

A kulcs-érték alapú rendszerek nagy mennyiségű adat tárolására alkalmasak gyors lekérdezésekkel.

A gráf alapú rendszerek csúcsok és élek formájában tárolnak adatokat. Akkor kifejezetten hasznosak, ha objektumok közötti kapcsolatokat akarunk vizsgálni.

3.3. Monitorozásra alkalmas felület

Adatokat nem csak az automatizmusnak adunk át, hanem a felhasználó számára is szeretnénk információkat nyújtani a vertikális kert állapotáról. Egy olyan felületre van szükség, amely alkalmas statisztikák, gráfok, és a legfrissebb mérési eredmények részletes megjelenítésére. Manapság rengeteg lehetőség közül válogathatunk. Megtekinthetjük adatainkat egy asztali alkalmazás segítségével, amely lehetővé tenné az adatok gyors elérését, ha az alkalmazás az adattároló szervert tartalmazó gépen helyezkedik el, viszont emiatt nem lehet az adatokat bárhonnan bármikor elérni.

Webalkalmazás fejlesztésével a monitorozó rendszert az interneten keresztül egy laptop segítségével bárhonnan eltudnánk érni, azonban ehhez szükséges egy publikus domain bérlése, valamint laptopot se akarunk feltétlen csak ebből a célból magunkkal hordani.

Okostelefon viszont manapság már mindenki zsebében van, amire szintén lehetséges speciális hálózaton keresztül kommunikáló mobil kompatibilis alkalmazásokat írni. Azonban a kisebb képernyő miatt, oda kell figyelni arra, hogy a statisztikákat, és diagrammokat oly módon rajzoljuk ki, hogy az kényelmes, de informatív legyen a felhasználó számára. Ha azt akarjuk, hogy az adatokat bárhonnan bármikor a hálózaton keresztül eltudjuk érni, akkor publikus domain bérlése ebben az esetben is szükséges lehet. A publikus domain igénye a webalkalmazásnál és a mobilalkalmazásnál is ugyanúgy hátrány, viszont virtuális helyi hálózattal megoldható ez a probléma, amelyről bővebben a következő fejezetben írok.

Lehetséges olyan webalkalmazás készítése is, amely nem csak számítógépről, vagy laptopról hanem okostelefonról is elérhető. Azonban ehhez a webalkalmazást különös odafigyeléssel kell elkészíteni, hogy minden eszközön jól használható legyen.

3.4. Összekötő rendszer

Mivel az adatlétrehozó eszközök, mint például a szenzorok, sokszor nem azonos helyen vannak az adatfeldolgozást végző eszközzel, valamint az adatmegjelenítésre szolgáló eszköz is máshol lehet, ezért az adatokat hálózaton keresztül kell mozgassuk. Erre a műveletre is manapság egyre több API (Application Programming Interface) áll rendelkezésre.

Az API-k olyan csatlakozási felületek, melyekre más alkalmazásokkal kapcsolódhatunk, és így, az API mögött álló könyvtár által szolgáltatott funkcionalitásokat elérhetjük anélkül, hogy pontosan tudnánk, hogy azoknak mi a pontos megvalósítása.

Ilyen például a Socket.IO nevű könyvtár, amely két irányú esemény alapú hálózaton keresztüli kommunikációt biztosít. két API-val segíti az adatok hálózaton keresztüli mozgatását. Az egyik a kliens oldali API, amely elsődleges feladata a hálózaton kiváltódó események figyelése, de lehetőséget biztosít esemény kiváltására is. A másik a szerver oldali API, amely segítségével létrehozhatjuk a rendszer központi elemét, a szervert, valamint, ugyanúgy lehetőséget ad események kiváltására és figyelésére. Az események információkat tárolhatnak magukban, így lehetővé téve a hálózaton keresztüli adatküldést.

Ahogy arról az előző fejezetben is szó volt, ahhoz, hogy a szervert elérjük távolról hálózaton keresztül, publikus domain-t kell béreljünk, vagy virtuális helyi hálózatot kell létrehoznunk. A virtuális helyi hálózat (VLAN) egy olyan technológia, amely lehetővé

teszi távoli eszközök számára, hogy úgy kommunikáljanak egymással mintha azonos szórási tartományban lennének. Ez azt eredményezi, hogy publikus domain bérlése nélkül is ellehet érni bárhonnan, bármikor a szervert az interneten keresztül. Hátránya viszont az, hogy a virtuális hálózaton csak az erre előkészített és konfigurált eszközök tudnak kommunikálni egymással.

4. Specifikáció

4.1. Automatizmus

A rendszer három nagyobb részből áll össze. Automatizmus, monitorozás, adattárolás. Az automatizmus feladata, hogy a szenzorok felől érkező adatokat kiértékelje, és ha szükséges akkor műveleteket hajtson végre.

Beérkező információ	Reakció
Alacsony nedvességi szint a gyökereknél	Vízkeringető pumpa beindítása/locsolás
pH érték túl alacsony / túl magas	Pumpa beindítása, amely fokozatosan kivezeti a rendszerből a tápanyaggal rendelkező vizet, miközben egy másik pumpa tiszta vizet enged a rendszerbe
Levegő hőmérséklet túl alacsony	Fűtés bekapcsolása
Levegő hőmérséklet túl magas	Légkondicionáló bekapcsolása
Páratartalom túl alacsony	Párásító készülék bekapcsolása
Páratartalom túl magas	Szellőztető bekapcsolása
Széndioxid szint túl magas	Szellőztető bekapcsolása, széndioxid tartályba bevezetés
Széndioxid szint túl alacsony	Széndioxidos tartály megnyitása
Nitrogén szint túl alacsony	Nitrogénes víz pumpálása a rendszerbe
Foszfor szint túl alacsony	Foszforos víz pumpálása a rendszerbe
Kálium szint túl alacsony	Káliumos víz pumpálása a rendszerbe
Nitrogén, Foszfor, Kálium szintek túl magasak	Rendszerben lévő tápos víz fokozott elvezetése, és tiszta víz bevezetése
Víz hőmérséklete túl alacsony	Vízben lévő fűtőszál bekapcsolása
Víz hőmérséklete túl magas	Vízben lévő hűtőspirál bekapcsolása
Fényerősség szintje	Mesterséges fény erősségének állítása

4.1. Táblázat: Automatizmus által kiértékelt esetek.

4.2. Használati esetek

A felhasználó számára biztosított monitorozó rendszer elsődleges feladata, hogy a felhasználó a lehető legfrissebb információkat megtekintse az egyes növényekről. Ezen kívül, az aznapi óránként elmentésre kerülő adatok is elérhetőek, valamint, hogy egy átfogó képet kaphasson a vertikális kert állapotáról statisztikákat is megnézhet,

amelyekben az elmúlt napok aggregált értékeit láthatja (minimum-, átlag-, maximum értékek), emeletenként lebontva, egy évre visszamenőleg.

A rendelkezésre álló adatok alapján, ha a felhasználó úgy gondolja, hogy az automatizmus által használt küszöb értékek nem megfelelőek, akkor lehetősége van azoknak az átállítására. Ha átállítja, akkor ezek a beállítások elmentésre kerülnek és az automatizmus ezeket a beállításokat fogja használni a későbbiekben.

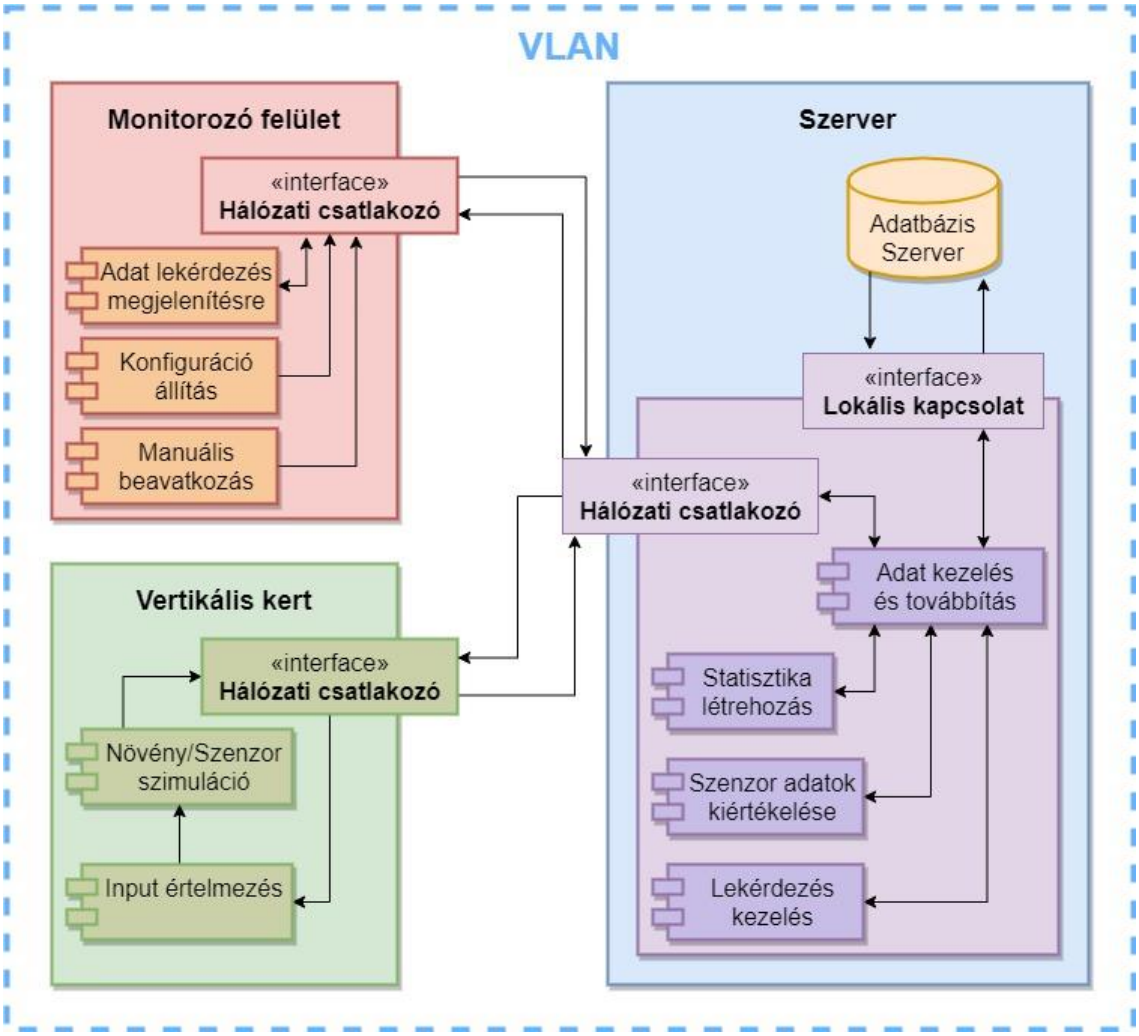
A küszöbérték állítás mellett, ha a felhasználó szükségét látja, akkor lehetősége van manuális beavatkozásra is. Bármilyen funkciót, amit az automatizmus képes ellátni, manuálisan is ellehet indítani. Ilyen funkciók például az öntözés elindítása, vagy a hőmérséklet növelése.

4.3. Adatok feldolgozása és tárolása

Többféle adat tárolására is szükség van. Adatok érkeznek a szenzorok felől, és a monitorozó felület felől is. A monitorozó felület felől a felhasználó által beállított küszöbértékek kerülnek mentésre. A szenzorok felől érkező adatokat háromféleképpen kell tárolni. Finoman- és durván szemcsézett módon, valamint statisztikák formájában. A szenzorok felől öt percenként érkezik adat, melyeket először finoman szemcsézett módon tárolunk. A felhasználó számára ezeket az adatokat küldi a rendszer megjelenítésre, amikor a növények jelenlegi állapotát akarja megtekinteni, valamint ezekből az adatokból készül a durván szemcsézett adatok tárolása is, oly módon, hogy az öt percenként váltakozó finoman szemcsézett adatokból egy óránként kiemelünk, és külön eltároljuk azokat. A durván szemcsézett adatokból pedig statisztikák készülnek, amiket szintén külön kell eltárolni, illetve a finoman szemcsézett adatokhoz hasonlóan megjelenítésre kerülnek a felhasználói felületen. A statisztikák emeletenként lebontva napi átlag-, minimum-, maximum-, értékeket tárolnak egy évig.

5. Rendszerterv

5.1. Rendszer részei



5.1. Ábra: A rendszer elemei és kapcsolatuk

5.1.1. Vertikális kert komponens

A növények és szenzorok szimulációját végző modul tárolja a növények jelenlegi állapotát, a növények fejlődésének szabályait és jelenségeit figyelembe véve módosítja azt, majd a szimulált szenzorokon keresztül, a szenzoroknak megfelelő mértékegységgel kimeneti értékeket állít elő. A következő táblázatban összegzem a kimeneti értékeket és tulajdonságaikat.

Szenzor	Mértékegység	Min/Max érték	Pontosság
Levegő hőmérséklet	°C	0 - 50	±2
Páratartalom	RH %	20 - 90	±5
Víz hőmérséklete	°C	-55 - +125	±0
Széndioxid	ppm	400 - 2000	±40
NPK	mg/kg	0 - 1999	±0
pH érték	pH	0 - 14	±0
Fényerősség	lux	0 - 65535	±0
Nedvességi szint	%	0 - 100	±0

5.1. Táblázat: Szenzorok kimeneti értékei

Az inputot értelmező modul feladata, hogy a beérkező utasításokat beleszámítsa a szimulációba, függetlenül attól, hogy az utasítás a felhasználó által manuális úton, vagy az automatizmust végző modultól származik.

Az itt jelenlévő hálózati interfész feladata, hogy fogadja a szimuláció által felhasznált utasításokat, illetve, hogy a szenzorok kimeneti eredményeit továbbítsa a hálózaton.

5.1.2. Monitorozó felületet biztosító komponens

Az „adat lekérdezés megjelenítésre” nevű modul feladata, hogy a felhasználó által megtekinteni kívánt információk lekérdezését továbbítsa, és a lekérdezés eredményét pedig megjelenítse.

A „konfiguráció állítás” nevű modul feladata, hogy továbbítsa a felhasználó által beállított konfigurációs értékeket a hálózati interfésznek.

A „manuális beavatkozás” nevű modul feladata, hogy a felhasználó által bekacsolt beavatkozásokat összegyűjtse, és továbbítsa a hálózati interfésznek.

5.1.3. Szerver komponens

A szerver komponens egy számítógép, amely az adatbázis szervert tartalmazza, illetve az ehhez kapcsolódó szerver funkcionalitásokat végző háttéralkalmazást. A szerver funkcionalitások közé tartoznak a következők.:

Kapcsolat biztosítása az adatbázis szerverrel, valamint hálózati interfész biztosítása, amire csatlakozhatnak a különböző hálózati eszközök.

Az „Adat kezelés és továbbítás” nevű modul felel azért, hogy a beérkező és kimenő adatok a megfelelő helyre kerüljenek továbbításra.

A „Statisztika létrehozás” nevű modul feladata, hogy naponta egyszer az utolsó óránként mintavételezett mérés után, egy napi statisztikát hozzon létre, amely tartalmazza az egyes emeletek átlag, minimum és maximum értékekeit. A statisztika létrehozása után, az adattovábbító modulon keresztül, elküldi az adatbázis szervernek hosszú távú mentésre.

A „Szenzor adatok kiértékelése” modul végzi a beérkező szenzor adatok értelmezését, és ennek megfelelően egy intézkedés visszaküldését (Lásd: 4.1. Táblázat). Ezt oly módon teszi, hogy a beérkező szenzor információkat a konfigurációs fájlban elmentett küszöbértékek közé helyezi, és ha az adott tartományon kívül esik, akkor intézkedő utasításokat küld a megfelelő eszköz felé, ebben az esetben, a szimulációt végző modul felé. Az 5.2. és 5.3. táblázatban részletezem az egyes növények által elfogadott tartományokat.

Mivel a növények annyi tápanyagot vesznek fel amennyire szükségük van, ezért nem szükséges a pontos mg/kg mennyiséget vizsgálni, inkább az a fontosabb, hogy milyen arányban vannak jelen a tápanyagok. Azokat a növényeket, amelyeknek nagyobb arányban van nitrogénre szüksége N betűvel jelöltem, azokat a növényeket melyeknek nagyobb arányban van foszforra és káliumra szüksége PK betűkkel jelöltem.

Növény	Víz °C	Levegő. °C	Nedv. %	RH %	CO_2 ppm	NPK	pH	Fény lum.
Hűvös évszakos növények								
Saláta	18 - 21	15 – 18	25 - 30	75 - 80	800 - 1200	N	5,5 – 5,8	2000
Káposzta	18 - 21	18 - 24	25 - 30	75 - 80	800 - 1200	N	6,0 - 6,8	2000
Brokkoli	18 - 21	18 - 23	25 - 30	75 - 80	800 - 1200	PK	5,5 – 5,8	2000
Spenót	18 - 21	2 - 21	25 - 30	75 - 80	800 - 1200	N	6,5 – 7,0	2000
Karfiol	18 - 21	15 - 28	35 - 40	75 - 80	800 - 1200	PK	6,5 – 6,8	2000

5.2. Táblázat: Hűvös évszakos növények által elfogadott tartományok

Növény	Víz °C	Levegő. °C	Nedv. %	RH %	CO_2 ppm	NPK	pH	Fény lum.
Meleg évszakos növények								
Paradicsom	18 - 21	20 - 22	25 - 30	75 - 80	800 - 1200	PK	6,0 – 6,4	2000
Bazsalikom	18 - 21	21 - 27	15 - 20	75 - 80	800 - 1200	N	5,5 – 7,5	2000
Paprika	18 - 21	21 - 32	25 - 30	75 - 80	800 - 1200	PK	4,8 – 6,2	2000
Uborka	18 - 21	23 - 28	25 - 30	75 - 80	800 - 1200	PK	6,0 – 7,0	2000
Eper	18 - 21	20 - 29	25 - 30	75 - 80	800 - 1200	PK	5,5 – 6,5	2000
Padlizsán	18 - 21	22 - 30	25 - 30	75 - 80	800 - 1200	PK	5,5 – 7,2	2000

5.3. Táblázat: Meleg évszakos növények által elfogadott tartományok

5.2. Választott megoldások

Adatbázis szervernek a MongoDB nevű dokumentum orientált adatbázis kezelő alkalmazást választottam, mert a dokumentum orientált NoSQL szemléletmód és az objektum orientált kóddal rendelkező háttéralkalmazás között hatékonyan lehet adatokat mozgatni. A MongoDB további előnyei, hogy a szenzorok felől érkező strukturálatlan adatokat tökéletesen lehet kezelni és tárolni, valamint könnyen skálázható a rendszer.

A szerveren lévő háttéralkalmazást, valamint a kapcsolatokat támogató alkalmazásokat Node.js és Socket.IO API-k segítségével hozom létre. A Node.js alapú háttéralkalmazás tökéletes környezetet ad ahhoz, hogy Socket.IO segítségével kapcsolatot teremtsünk a hálózaton, valamint a MongoDB adatbázis szerverrel is kifejezetten kompatibilis. Objektum orientáltsága révén arra is tökéletes, hogy elvégezzük a szükséges háttérszámításokat.

Míg a hálózati kapcsolathoz szükséges interfészeket a Socket.IO biztosítja, maga a kommunikáció egy VLAN-on történik, amit egy Hamachi nevű VPN (Virtual Private

Network) alkalmazás segítségével tudunk létrehozni. Azért választottam a Hamachi nevű VPN megoldást a publikus domain helyett, mert jelen esetben teljesen elegendő, ha csak a hálózaton lévő eszközök tudnak kommunikálni egymással. Publikus domain esetében idegen eszközök is eltudnák érni a rendszert, de ez akár kifejezetten káros is lehet a mi esetünkben.

A szenzorok és növények szimulációját egy külön számítógépen elhelyezkedő Node.js alkalmazással oldom meg, amely képes a szerver felől érkező parancsok fogadására, és azokat felhasználja a szimuláció számítása során. A szimulációs alkalmazás szintén tartalmaz egy Socket.IO API megvalósítást ahhoz, hogy kommunikálni tudjon a szerverrel. Azért van külön számítógépen a szimuláció, hogy ezzel is egy fokkal realisztikusabbá tegyem a projektet, hiszen egy valós vertikális kert is valószínűleg helyileg máshol helyezkedne el, mint a szerver és a kliens.

A felhasználói felületet webalkalmazás formájában valósítom meg, amelyet számítógépről és mobil eszközről egyaránt el lehet érni. A webalkalmazás elkészítéséhez HTML, CSS, Javascript, és egy Node.js-hez készült Express nevű letölthető csomagot használok.

6. Implementáció

6.1. Szimulációs oldal fejlesztése

6.1.1. Fájl olvasása és írása

A szimulációs oldal lényege, hogy pótolja a fizikai vertikális kertet. Valós növények helyett, digitális növényeket tárolok, melyeknek a tulajdonságai változni tudnak akárcsak a valós növényeknek. A növények tulajdonságainak módosítását egy Node.js alkalmazás formájában valósítom meg.

Azonban ahhoz, hogy a növények állapota ne vesszen el a program újraindítása során, a növények adatait valamilyen formában tárolnom kell a memórián kívül. A növények adatainak a tárolásához használhatnék egy adatbázis rendszert, de ezzel elvenném a szakdolgozat lényegét hiszen az egyik cél, hogy a növények adatait eljuttassam egy adatbázisba. Mivel a valós szenzorok is egy fájlba írnák a kimeneti értékeiket, ezért a növények adatait én is egy szimpla txt fájlban fogom tárolni, és onnan fogja a szimuláció beolvasni amikor szükséges.

Készítettem egy növény osztályt, és minden a szakdolgozatban használt növénynek példányosítottam egy-egy objektumot, amelyek mezői tartalmazzák az adott növény vizsgált tulajdonságainak kiinduló alapértékeit. Ezeket az alapértékeket úgy határoztam meg hogy a már korábban megállapított küszöbértékek között félúton legyenek.

```
JS Plants.js > plantObject_Lettuce
1   var plantObject_Lettuce= {
2       PlantName: "Lettuce",
3       WaterTemp: 19,
4       AirTemp: 16,
5       Moisture: 27,
6       RelHumidity: 77,
7       CO2: 1000,
8       NPK: "N",
9       Nitrogen: 310,
10      Phosphorus: 17,
11      Potassium: 200,
12      PH: 5.7,
13      Light: 2000
14  }
15
```

6.1. Ábra: A saláta alapértékei

Ezután egy algoritmus legenerál egymillió példányt, mindegyiket egy egyedi azonosítóval. A növény típusok sorrendje határozott. A száz emeletes vertikális kertben, tíz emeletenként váltakoznak a növény típusok, tehát például a legalsó tíz emeleten

találhatóak a saláták. Ezután ezeket az objektumokat szöveggé alakítva egy txt fájlba kiírja. Minden sorban egy növény található.

A fájl írását és olvasását külön kihívás volt jól megoldani, ugyanis sok különböző technika létezik fájl írására és olvasására, de én szerettem volna ezt a lehető leghatékonyabban megoldani. Bár a sebesség növények vizsgálata során nem nagy prioritás, annak érdekében, hogy jól skálázható legyen a projekt, a beolvasást és kiírást hatékonyan kellet megoldanom.

Mivel javascript backend környezetben írom a kód ezen részét, először evidensnek tűnt, hogy minden objektumom egy JSON (JavaScript Object Notation) dokumentum lesz. Ez azt jelenti, hogy egy millió fájl íródik ki a háttértárra. Ez a művelet 12 percet vett igénybe. A következő számítógépen:

PC	**Leírás**
Processzor	AMD Ryzen 5 3600X 6-Core Processor Alapsebesség: 3,79 GHz Magok: 6 Logikai processzorok: 12
RAM	16 GB DDR4 2133 Mhz
Háttértár	SSD 500 GB Olvasási sebesség: 550 MB/s Írási sebesség: 520 MB/s SATA3
Videókártya	GTX 1050 Ti Grafikus memória: 4 GB

6.1. Táblázat: Számítógép specifikáció, amelyen a fejlesztés történt

Az eljárásnak a lassúsága abból adódik, hogy a Windows operációs rendszer nagyon nehezen tud egy millió fájlt egyszerre kezelni.

Ebből levonva a tanulságot, a következő módszernek, amit kipróbáltam az alapötlete, hogy egy fájlba írom bele az összes adatot. Ezt egy Node.js által adott AppendFile nevű beépített metódus segítségével valósítottam meg, amely felgyorsította a kiírást másfél percre. Ennek az eljárásnak a lassúságát az okozta, hogy minden új sor hozzáfűzésénél újra meg kell nyitni a fájlt, és be kell zárni azt.

A következő, egyben utolsó módszer, amit kipróbáltam pontosan ezt a problémát küszöböli ki. Ennek a beépített metódusnak a neve StreamWrite. Az eljárás lényege, hogy manuálisan nyitunk egy csatornát a fájl felé, ami addig marad nyitva amíg parancsot nem

adunk rá, hogy záruljon be, így folyamatosan tudunk szöveget adogatni a csatornának bármilyen fennakadás nélkül, majd bezárjuk a csatornát. Az eljárás hátránya, hogy a szöveges fájlhoz hozzányúlni addig nem lehet amíg az meg van nyitva a program által. Amikor a csatorna bezárul, a fájl akkor válik elérhetővé, és az eredményt is akkor látjuk benne. Hatalmas előnye viszont, hogy az egymillió növény kiírását így 1 másodperc alatt be lehet fejezni. A fájlból való beolvasást ennek mintájára szintén egy StreamReader metódussal valósítottam meg.

6.1.2. Növények változtatása

A növények tulajdonságainak módosításához kettő listát használok. Az egyik lista a növények listája, a második pedig egy parancslista, ami tartalmazza, hogy melyik növénnyel mi a teendő. A két lista ugyan annyi elemből áll, és azonosító számuk alapján folyamatosan sorrendben vannak. Ez azt eredményezi, hogy amikor megakarjuk nézni, hogy egy bizonyos növényre milyen parancsok vonatkoznak, akkor nem kell végig menni a parancslistán, hogy megkeressük a releváns parancs objektumot, hanem a növény azonosító számával azonnal tudunk hivatkozni a parancs objektum helyére a parancs listában.

A parancs objektum tíz mezővel rendelkezik. Ezek közül az első egy azonosító szám, amely nem csak a parancs objektumot, de egyben a hozzá tartozó növényt is azonosítja. A maradék kilenc mező pedig a növény különböző tulajdonságaira vonatkozó parancsokat tárolja. Ezek a parancsok szövegesen egy kód formájában tárolódnak. Például az egyik mező a „wt_order”, amely a víz hőmérsékletével kapcsolatos utasítás kódját tárolja. Az egyik ilyen utasítás például a „WT_HEAT”, ami azt jelenti, hogy az adott növénynél a víz túl hideg, tehát melegíteni kell. A következő táblázat mutatja a parancs objektum összes mezőjét, és azoknak az összes lehetséges értékét.

Mezők	+	-	**Semleges**
Víz hőmérséklete	WT_HEAT	WT_COOL	WT_0
Levegő hőmérséklete	AT_HEAT	AT_COOL	AT_0
Öntözés	IRRIGATE		IR_0
Páratartalom	HUMIDIFY	DEHUMIDIFY	HU_0
Levegő CO2 tartalma	CO2_MORE	CO2_LESS	CO2_0
Nitrogén	NIT_MORE	NIT_LESS	NIT_0
Kálium	POT_MORE	POT_LESS	POT_0
Foszfor	PHO_MORE	PHO_LESS	PHO_0

Víz PH értéke	PH_MORE	PH_LESS	PH_0

6.2. Táblázat: Parancs kódok

A táblázatban látható, hogy nem feltétlen kell minden mezőben három lehetséges kódra számítani, mert az öntözésnek például csak öntöző és egy semleges parancsra van szüksége. Semleges parancs mindenhol kell legyen, mivel ez jelzi a rendszer számára, hogy az adott növénnyel jelenleg nem kell csinálni semmit.

A szimuláció előrehaladását végző kód végighalad egy ciklussal az összes növényen, és megnézi a hozzá tartozó parancs objektum mezőit. Ha semleges parancsot talál, akkor egy 0 és 1 közötti véletlenszerű számot generál, amit tizedes pontossággal kerekít, (kivéve a pH érték és a Foszfor szint esetében, mert akkor egy 0 és 0,5 közötti számot generál százados pontossággal kerekítve) és ezt a számot az adott növényi tulajdonságtól függően kivonja vagy hozzáadja, vagy a 50% eséllyel valamelyiket a kettő közül. Például a növény nedvességi szintje semleges kód esetén biztosan csökken, de a víz hőmérséklete nem feltétlen csak csökken, vagy csak nő. Ezzel a fajta véletlenszerű érték változtatással a szimuláció lehetőséget arra, hogy az összes parancs funkció tesztelésre kerüljön, ugyanakkor az értékek változása valamilyen szinten mégis realisztikus képet mutasson.

6.1.3. Adatok küldése és kommunikáció a rendszer szerver oldalával

A kommunikációt a rendszer különböző részei között (Szerver oldal, Vertikális kert, Frontend) egy Socket.IO nevű API-val oldom meg, amelyet kifejezetten Node.js környezetet használó rendszerekhez fejlesztettek. Az API használatához szükséges kezdeti inicializálásokat a „6.2. Backend fejlesztése” fejezetben részletezem.

A vertikális kert oldalán mindössze annyi volt a feladatom, hogy a Socket.IO kliens oldali változatát fel kellett telepítenem, majd példányosítottam. Ennek a folyamatnak az eredménye egy socket példány, amely az adott kapcsolatnak a tulajdonságait tartalmazza. A 6.2. ábrán látható az IP cím és port beállítása a példányosítás mellett.

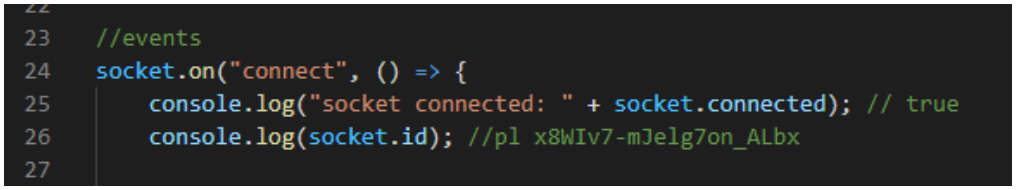
```
4
5   const io = require("socket.io-client");
6   const socket = io("http://25.97.30.133:3000/");
7
```

6.2. Ábra: Csatolási felület példányosítása

Fontos részlet, hogy a példányosítás során határozzuk meg a kapcsolat címét. Ehhez egy IP címre és egy port számra van szükség. A port számot a szerver oldalon határozzuk meg, az IP cím az én esetemben egy hamachi által adott IP cím. A hamachi biztosítja a virtuális helyi hálózatot. Az ebben a hálózatban felvett eszközöknek a hamachi egy új IP címet oszt, amire, ha hivatkozunk bárhonnan elérhetjük azt a bizonyos eszközt, mindaddig amíg rendelkezünk internet hozzáféréssel. A hamachi segítségével létrehozott

hálózatokra felvett eszközök úgy kommunikálhatnak egymással mintha egy helyi hálózaton lennének.

Ezután a socket példányt használva megírtam, hogy milyen eseményekre kell figyeljen. Egyik ilyen nagyon fontos esemény például a „connect” esemény, amely akkor váltódik ki a hálózaton, amikor a kliens csatlakozik a szerverhez. A 6.3. ábra egy példát mutat a csatlakozáskor történő hálózati esemény lekezelésére.



6.3. Ábra: Kapcsolódáskor kiváltódó esemény lekezelése

A képen látható kódrészlethez hasonlóan kell elkapnom a többi általam definiált eseményt is.

Másik fontos esemény, amit a vertikális kert oldalán le kell kezelni, az a beérkező parancslista. A vertikális kert oldalán egy tömb van biztosítva a beérkező parancslista tárolására. Mielőtt megérkezik a parancslista, a tömböt ürítem, hogy biztosan ne okozzon gondot a korábban benne szereplő parancslista. A parancslistát és minden más nagyobb mennyiségű adatlistát, feldarabolt, úgynevezett „Batch” alapú adatküldési eljárással mozgatok a hálózaton. Ennek előnye, hogy nagyon jól skálázható, nem csak a teljes küldendő adatmennyiség terén, hanem a batch méretek terén is. Ezzel az eljárással, nagyobb mennyiségű adatot lehet egyszerre egy hálózati csomagban küldeni, és egyéb hálózati inkonzisztenciák is kiküszöbölhetőek. A kódot úgy írtam meg hogy a batch méreteket rugalmasan lehessen változtatni. A fogadó oldalt az egyes beérkező batch csomagokat hozzáfűzöm a küldési folyamat előtt tisztított tömbhöz, és így épül fel a teljes adatlista. Az egymillió növény hálózaton keresztüli küldését a 6.2.2. fejezetben részletezem.

6.2. Backend fejlesztése

A backend a rendszer azon része, amivel a felhasználó közvetlenül nem lép interakcióba. Itt történnek a háttérszámítások. Elméletileg az előző alfejezetben tárgyalt vertikális kerthez tartozó program is a backend része a felhasználó szempontjából, de azért bontottam külön fejezetre, mert a szerver szemszögéből kliensként van csatlakozva a szerverre.

6.2.1. MongoDB adatbázis

A MongoDB adatbázis rendszernek különböző változatai vannak. Én az ingyenesen elérhető on-premise közösségi verziót töltöttem le és telepítettem fel. Az alap adatbázis szerver mellett egy MongoDB Compass nevű grafikus interfészt is feltelepítettem. Mivel először fejlesztettem node.js alkalmazást, és először próbáltam javascript kódon keresztül elérni és manipulálni MongoDB adatbázist, ezért a tanulási folyamat közben sokszor hasznos volt, hogy volt egy grafikus felület, ahova gyorsan rátudtam nézni és kísérletezéseimnek azonnal láttam hatásait.

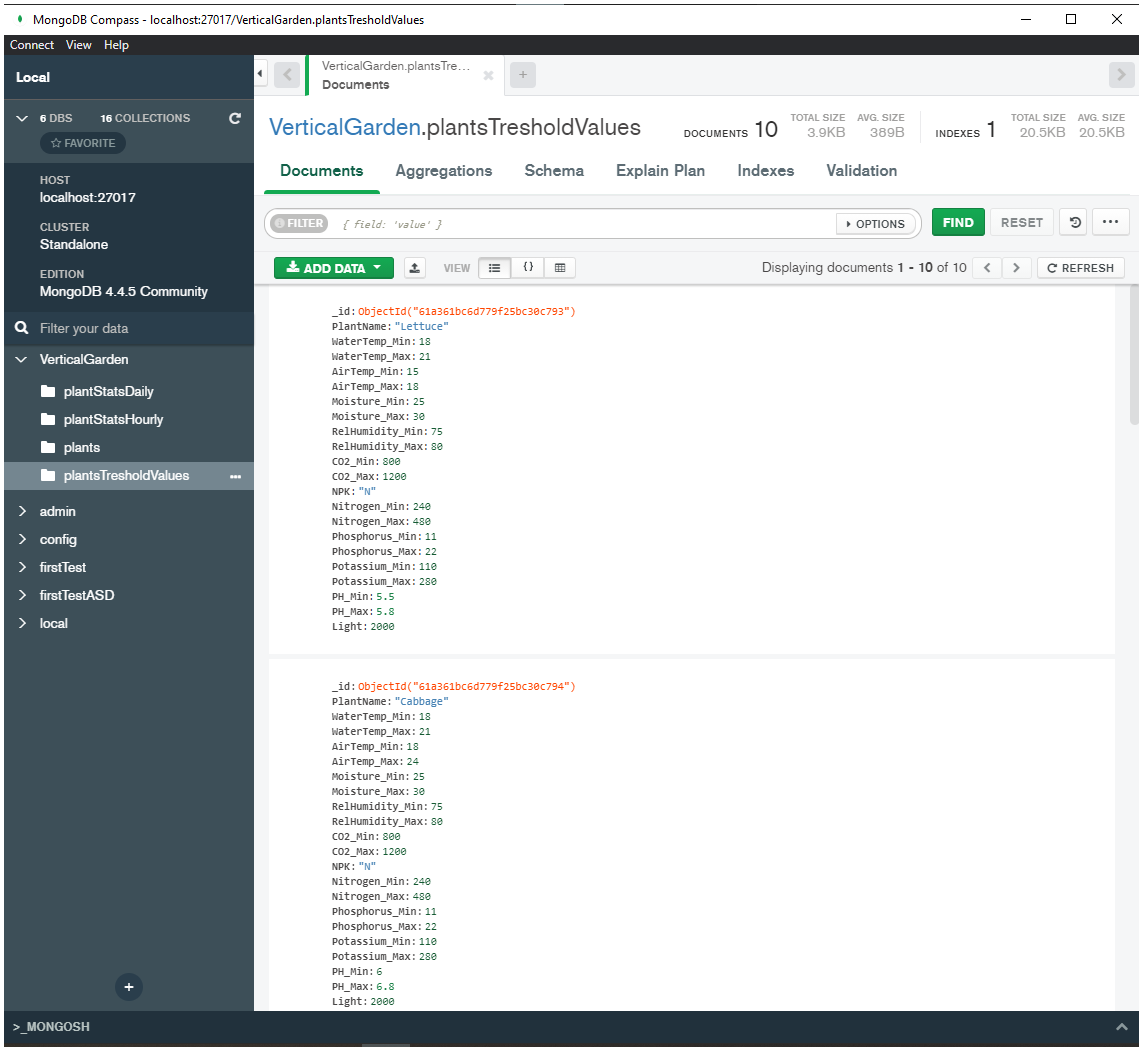
Ahhoz, hogy létrejöjjön a kapcsolat a node.js alkalmazás és a MongoDB szerver között, a node.js alkalmazáshoz fel kell telepíteni a MongoDB csomagot, a node package manager segítségével. A 6.4. ábrán a MongoDB javascript kódon keresztüli elérésének szintaktikája látható.

```
13  //mongodb
14  const MongoClient = require("mongodb").MongoClient;
15  const uri = "mongodb://localhost:27017/";
16
17  const client = new MongoClient(uri, {useNewUrlParser:true, useUnifiedTopology:true});
18
19  client.connect();
20  const db = client.db("VerticalGarden");
21  console.log("Connected to " + db.databaseName);
22
```

6.4. Ábra: MongoDB elérése node.js kódból

A csatlakozás és adatbázis létrehozása után, metódusokat írtam, amelyek segítségével létrehoztam a különböző szükséges kollekciókat, és feltöltöttem őket ideiglenes dokumentumokkal, amik a vázát fogják alkotni az adatbázisnak.

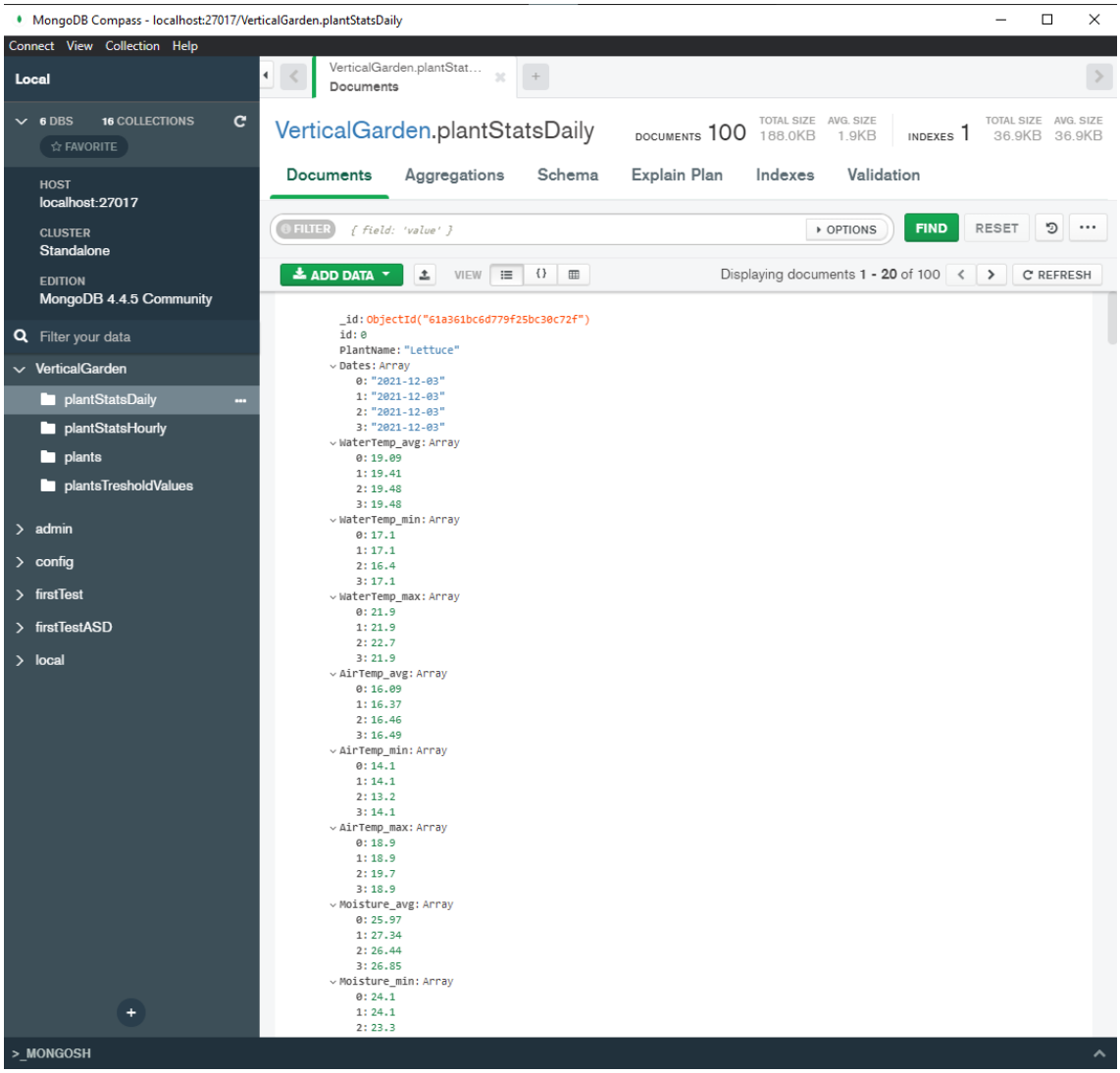
A rendszerhez tartozó adatbázis négy kollekcióból áll. Az első kollekció a küszöbértékeket tartalmazó kollekció. Minden növény típusnak van egy saját dokumentuma, amelyben elvannak tárolva az adott növényre érvényes alsó és felső küszöbértékek. Mivel a szakdolgozat jelenlegi állapotában tíz különböző növénnyel foglalkozom, így 10 különböző dokumentummal rendelkezik a küszöbértékeket tároló kollekció.



6.5. Ábra: Növények küszöbértékei MongoDB adatbázisban tárolva

A 6.5. ábra mutatja hogyan néznek ki a MongoDB Compass vizuális felületén eltárolt dokumentumok. Az ábrán a saláta és a káposzta alapvető küszöbértékei láthatóak. Ha a felhasználó módosítja az alapvető küszöbértékeket akkor az itt íródik felül, és a növények gondozását végző automatizmus az itt eltárolt adatokat használja fel számításaihoz.

A második kollekció, amit a MongoDB adatbázisban tárolok, az az éves statisztikai adatokat tartalmazó kollekció.



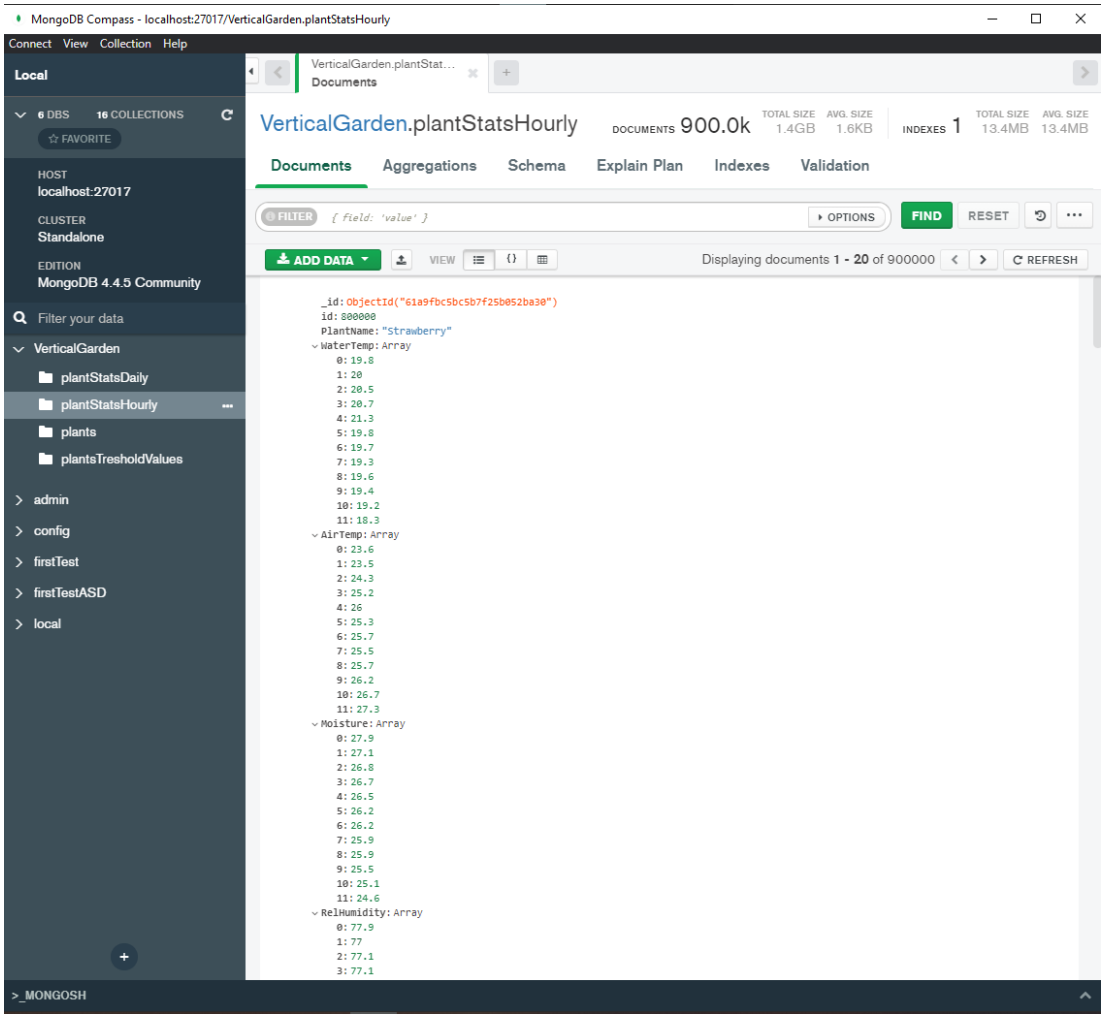
6.6. Ábra: Emeletek napi statisztikái egy évre visszamenőleg

A 6.6. ábrán látható hogyan tárolódik a MongoDB adatbázisban egy emelet napi statisztikái egy évre visszamenőleg. A problémát úgy oldottam meg hogy egy dokumentum tartalmazza egy specifikus emelet statisztikáit. A jelenlegi felállásban 100 emelettel rendelkezik a vertikális kert, és minden emeleten csak egy bizonyos növény található meg. Az emelet azonosítója és az ott található növény típusa mellett, tárolom a statisztikák létrehozásának dátumait, és az összes szenzorok által vizsgált tulajdonság minimum, átlag, és maximum értékeit tömbök formájában.

A statisztikák számítása úgy történik, hogy ha az óránként tárolt statisztika elemszáma elérte a 24-gyet, az azt jelenti, hogy 24 órányi adat összegyűlt, vagyis a napot lezártnak tekinti a rendszer, és egy algoritmus segítségével kiszámolja minden növény minimum, maximum, és átlag értékeit minden tulajdonságára, majd végigmegy az adott emelet összes növényén és azok statisztikai értékeiből is kiszámolja a minimum, maximum és átlag értékeket. Ha ezzel is végzett, a kiszámolt értékek mellé a jelenlegi dátum hozzáadódik és továbbításra kerül az adatbázis felé, ami úgy történik, hogy lekérdezésre

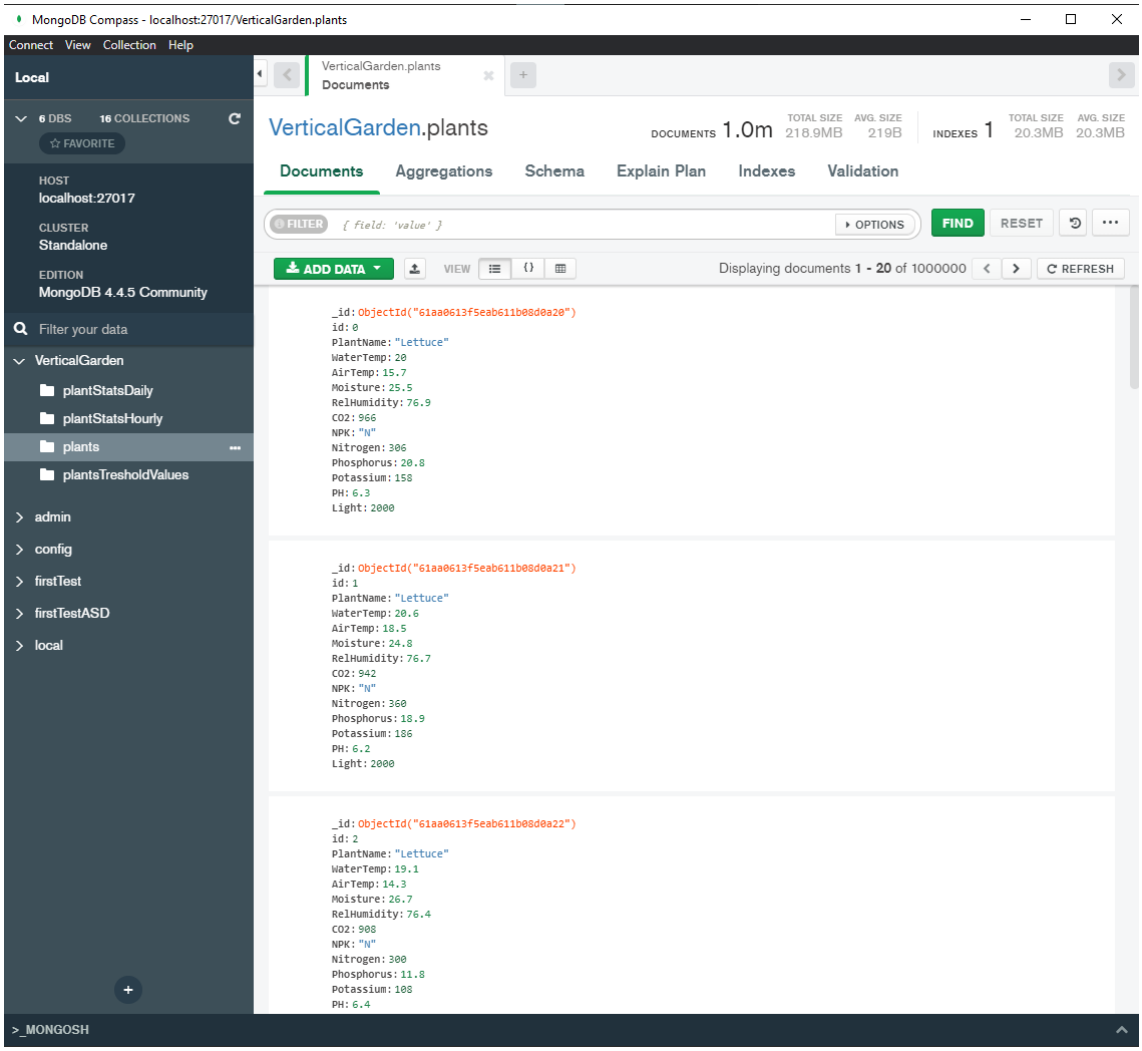
kerül az adott emelet, majd a backend hozzáfűzi az újonnan kiszámolt értékeket, és feltölti az adatbázisba. A tesztelések során gyorsított szimuláció miatt, a 6.6. ábrán látható dátumokat tároló tömbben csak egy nap dátuma található, de egy valós környezetben minden dátum különböző kell, hogy legyen. A dátumokhoz tartozó sorszám mutatja, hogy a többi tömbben melyik adatok tartoznak az adott dátumhoz, mivel a tömbön belüli elhelyezkedésük megegyezik.

A harmadik kollekció, amit a MongoDB adatbázisban tárolok, az a növények óránként mentett aznapi adatai. Adatbeolvasás 5 percenként történik, de új beolvasáskor az adatok felülíródnak, így a régi adatok elvesznek. Azonban óránként egyszer egy beolvasás jelenlegi értéke kimentésre kerül. Az óránként mentett adatok kollekciójában minden dokumentum egy növényt reprezentál, hasonlóan a friss adatokat tároló kollekcióhoz. Az az egy különbség a kettő között, hogy itt minden tulajdonság egy tömb. Új adat beszúrásakor minden tulajdonság értéke a hozzá megfelelő tömbhöz adódik hozzá. Ez egy számítás igényes feladat, mivel egymillió dokumentumot kell lekérdezni, és egy hozzáfűzést végezni. Ez a művelet a jelenlegi megoldásban 40 másodpercet vesz igénybe. A 6.7. ábrán látható milyen formában mentődnek óránként az adatok.



6.7. Ábra: Példa egy dokumentumra az óránként mentett adatok kollekciójában

Adat beolvasás a szenzorok felől öt percenként történik. Az újonnan beolvasott adatokat oly módon frissítem az adatbázisban, hogy beolvasáskor mind az egymillió elmentett dokumentumot kitörlöm és az új adatokat elmentem a helyükre. Ez a módszer azért a leggyorsabb, mert a MongoDB által adott beépített javascript metódusok között, bár megjelenik az update (frissítés) metódus, de ez csak egy dokumentum adatainak frissítését képes lekezelni. Nagy mennyiségű felülírás esetén sokkal hatékonyabb megoldás a deleteMany és insertMany metódusok használata, amelyek szándékosan nagy mennyiségű adat mozgatásához lettek kitalálva. Az updateMany metódus is létezik, azonban ezzel csak egy közös értékkel lehet módosítani a lekérdezett dokumentumokat, nem pedig dokumentumonként különböző értékekkel, amire nekem lenne szükségem. A 6.8. ábrán látható hogy milyen szerkezettel tárolódnak a növények adatai a MongoDB adatbázisban.



6.8. Ábra: Növények dokumentumai finoman szemcsézett adatokkal

6.2.2. Socket.IO telepítése, előkészítése, és működtetése

A node package manager segítségével feltelepítettem a szükséges socket.io csomagot. A telepítés után létrehoztam a szükséges példányokat. Ez egy új szerver példány és egy új IO példány pontosan. A szerver példánynál beállítottam, hogy a 3000 számú porton hallgasson kérések után, az IO példány pedig a továbbiakban a kliensek (socket-ek) és a szerver közötti kérések és fogadások lebonyolítását végzi.

Működése nagyon egyszerű, miután egy socket csatlakozott a szerverhez, az adott socket és a szerver között létrejön egy csatorna, amin kérések továbbítódnak oda-vissza. Ezek a kérések két elemből állnak, egy névből és egy adat tartalomból. Az adat tartalomba bármilyen típusú adatot helyezhetünk. A kommunikáció az eseménykezeléshez hasonló. Az egyik oldalt kiváltódik egy üzenet egy névvel, amely üzenet neve után, ha már hallgatózik a másik oldal, akkor azt elkapja, és elindít valamilyen általunk definiált metódust. Az üzenet adattartalma paraméterként átadódik a metódusnak.

A legnagyobb kihívás az egymillió adat hálózaton keresztüli mozgatása volt. Különböző módszerekkel próbálkoztam míg eljutottam a jelenlegi leghatékonyabb megoldáshoz, ami egy Batch alapú adatküldési eljárás. Jelen konfigurációban a leghatékonyabban úgy tudok egymillió növény adatát átküldeni, hogy ha kliensről küldöm a szerver felé, akkor maximum 10 ezres darabszámú csomagokra bontom az egymillió növényt, ha a szerverről a kliens felé, akkor maximum 100 ezres darabszámú csomagokat hozok létre, függetlenül attól, hogy az adatok méretben ugyan akkorák mindkét irányba. Ezzel a megoldással a rendszer 1 másodperc alatt küldi át az összes adatot az egyik oldalról a másikra.

Egy korábbi próbálkozásban egy ciklussal mentem végig az összes adaton és egyesével küldtem át őket a hálózaton. Ez a folyamat 20 másodpercet vett igénybe. Lassúsága mellett nagyon érzékeny volt, és gyakran megszakadt a kapcsolat a küldési folyamat közben. A batch alapú technika gyorsabban és megbízhatóbban működik.

6.2.3. Növények gondozását végző automatizmus

Amikor új szenzor beolvasás történik és megérkezik az egymillió növény adatainak legfrissebb változata a szerverhez az első dolog, ami történik, hogy elindul az adatok feltöltése a MongoDB szerverre. Amint a feltöltés befejeződött, elindul az adatok kiértékelése.

A küszöbértékek a program futása során a memóriában tárolódnak, azonban mivel a felhasználó bármikor módosíthatja őket, így a közvetlen felhasználásuk előtt frissítjük őket az adatbázisból. A küszöbértékeket azért érdemes az adatbázisban is tárolni, mert a program újraindítása esetén a felhasználó által beállított adatok máskülönben elvesznének.

Miután befejeződött a küszöbértékek frissítése, egy metódus végighalad az összes növényen, és a legfrissebb értékeit behelyezi a releváns növényhez beállított küszöbértékek közé. Az eredmények alapján összeállítja a korábbi fejezetekben már említett parancslistát, a megfelelő kódokkal.

```
try {
    orderObj.id = item.id;

    if (item.WaterTemp < tresholdValues[i].WaterTemp_Min) {
        //melegítés

        orderObj.wt_order = "WT_HEAT";


    } else if(item.WaterTemp > tresholdValues[i].WaterTemp_Max){
        //hűtés

        orderObj.wt_order = "WT_COOL";

    } else {
        //nincs szükség beavatkozásra

        orderObj.wt_order = "WT_0";
    }
```

6.9. Ábra: Kiértékelés kódrészlet

A 6.9. ábrán látható ahogyan egy növényhez tartozó víz hőmérséklete értékelésre kerül, és a megfelelő parancs kódja bekerül a parancslistába. A parancslista összeállítása után megkezdődik annak küldési folyamata. Mivel a parancslista minden növényhez parancsokat társít, így a parancslista mérete is hasonló lesz, mint a növények listájának mérete, emiatt itt is célszerű a batch alapú küldés, amit a vertikális kert kliensként fogad.

6.3. Frontend fejlesztése

A projekt frontend része tartalmazza a felhasználói felületet, amely segítségével a felhasználó interakcióba tud lépni a rendszerrel. Megtekintheti a növények legfrissebb állapotát és statisztikáit, valamint beállíthatja az automatizmus által használt küszöbértékeket.

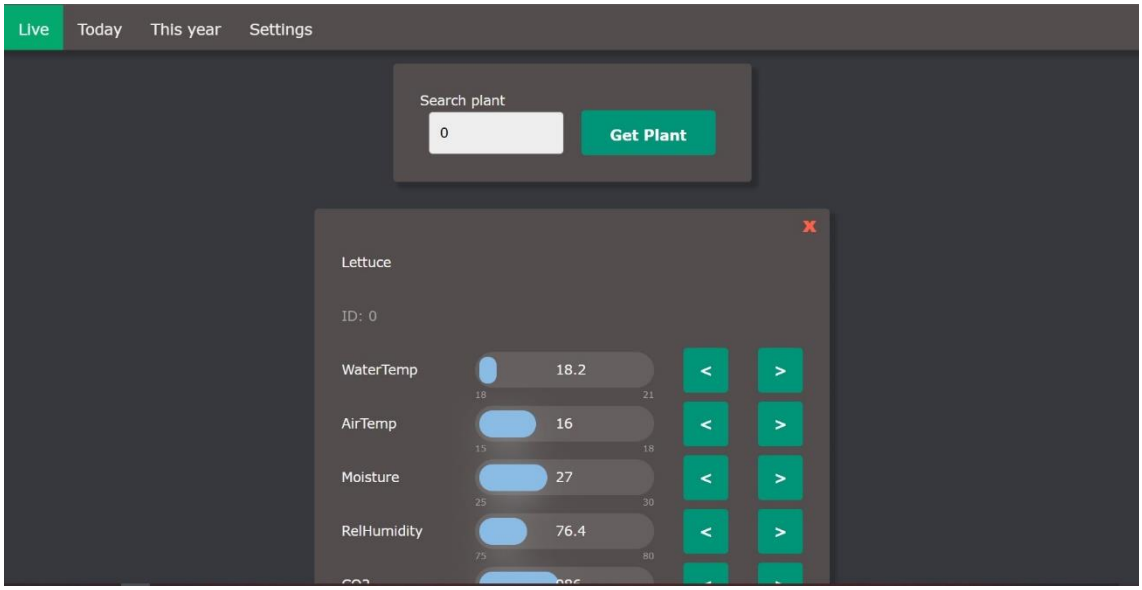
6.3.1. Felhasználói felület létrehozása és külalakja

A felhasználói felülethez webalkalmazást készítettem, melyet egy express nevű csomag segítségével készítettem el. Az express egy széleskörűen elterjedt csomag a node.js alapú rendszerek között, ami jelentősen leegyszerűsíti és felgyorsítja a webalkalmazás

fejlesztést. Telepítés után példányosítottam, és a szerver létrehozásához felhasználtam az elkészült objektumot.

Ezután már csak annyi volt a dolgom, hogy megadjam, hogy milyen URL kérésekre milyen fájlt adjon az alkalmazás válaszul. Az én esetemben ez főleg a menüben történő oldal váltásra vonatkozik. Amikor a felhasználó a menüből megnyit egy másik oldalt, akkor egy új HTML fájl töltődik be. Ezt a HTML fájlt az express alkalmazás továbbítja a kliens oldalára.

A főoldalt és a menüpontokhoz tartozó oldalakat különálló html fájlok formájában készítettem el. Az oldal kinézetének elkészítéséhez egy közös CSS fájlt használtam. A statisztikák kirajzolásához a w3school CSS moduljából használtam fel elemeket, és egy charts.js nevű kiegészítőt használtam még.

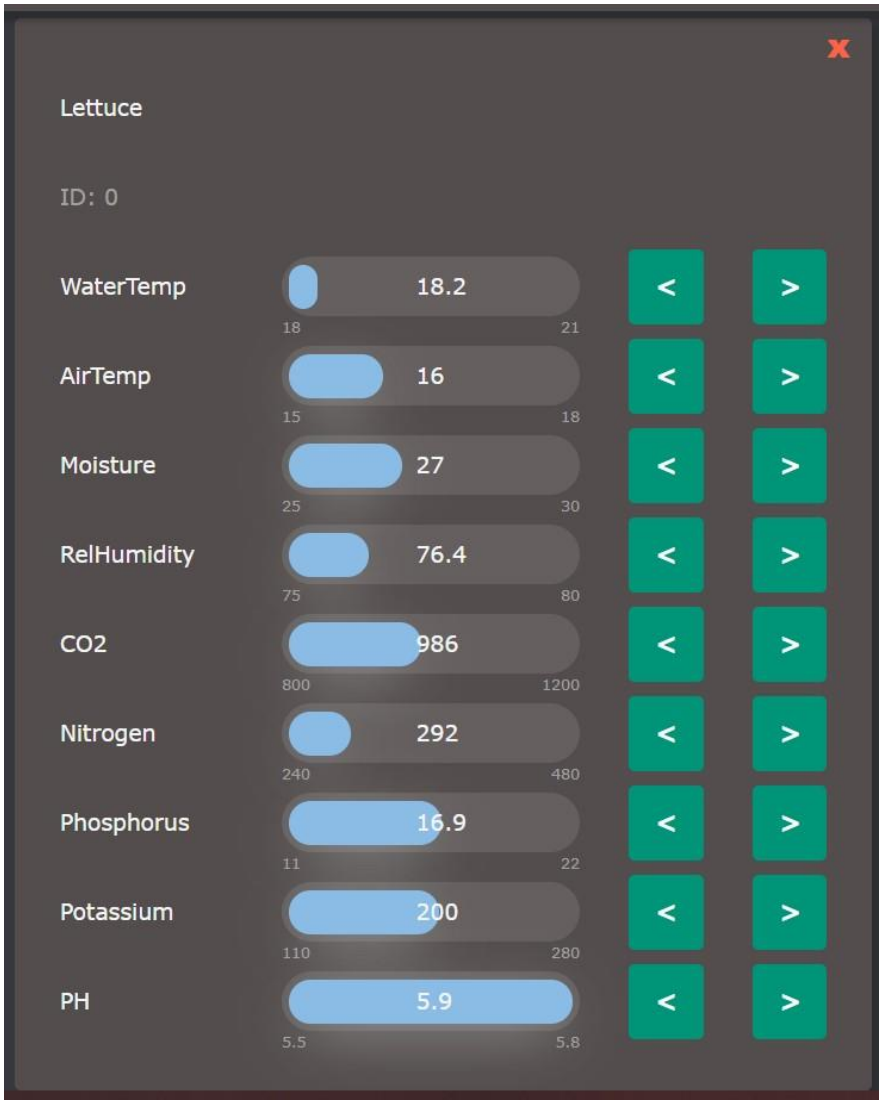


6.10. Ábra: Növények legfrissebb adatait mutató weboldal

A 6.10. ábrán a weboldal külalakja és példaként egy növény élő adatainak megjelenítése látható. A weboldal tetejére készítettem egy navigációs sávot, amely tartalmazza a menüpontokat, amelyek a különböző oldalakra vezetnek. A dizájn szempontjából próbáltam egy korszerű modern letisztult stílust kivitelezni. Az oldal tartalma „kártyák" formájában jelenik meg. Amikor a felhasználó több növényt kérdez le, minden növény a saját kártyáján jelenik meg a listában. A kártyák vertikálisan sorakoznak egymás után. Az oldal görgetése esetén, a navigációs sáv az oldal tetején marad.

6.3.1. Felhasználói felület funkciói

A webalkalmazás megnyitásakor az élő adatokat mutató oldal jelenik meg alapvetően. A navigációs sáv alatt található a kereső mező, amivel az adott oldalnak megfelelően növényeket lehet lekérdezni.



6.11. Ábra: Egy saláta tulajdonságainak kirajzolása

A 6.11. ábrán egy saláta kártyája található. A kártya tulajdonképpen egy konténer, amiben megtalálható a növény típusa, azonosítója, és a többi szenzorokkal mért tulajdonsága.

Ezen az oldalon a tulajdonságoknak kizárólag csak a legfrissebb értékei láthatóak. A megjelenített adatok valós időben módosulnak amikor szenzor beolvasás történik. A szenzorok adatai elküldődnek a szerverhez, ahol feltöltődnek az adatbázis szerverbe. Amikor ez a folyamat megtörténik egy értesítés kerül kiküldésre a frontendnek, hogy az adatok frissítésre kerültek. Ekkor a frontend összegyűjti a weboldalon jelenleg kirajzolt növények azonosító számait, és ezeket visszaküldi a backend részére. A backend lekérdezi a beérkezett azonosító számokkal rendelkező növényeket, és egy csomagban visszaküldi a frontend részére.

A frontenden a növények adatait tartalmazó konténerek HTML elemei objektumként el vannak mentve egy listában. Ez azért fontos mert ha objektumként vannak tárolva, később könnyen hivatkozni lehet rájuk, és a hozzájuk tartozó HTML elemekre javascript kódon keresztül. Amikor megérkezik a friss adatokat tartalmazó csomag a backend felől, akkor az algoritmus végigmegy az oldalon megjelenített növények listáján, és az objektumok tulajdonságait felhasználva, felülírja a meglévő tulajdonságok értékeit, és a kirajzolt folyamatsáv hosszúságát is megváltoztatja az új adatoknak megfelelően. A folyamatsáv hosszúságának változtatásához az adott HTML elemhez tartozó CSS stílus kódját változtatom javascript kódon keresztül.

A növények tulajdonságainak legfrissebb értékei mellett, a folyamatsáv két végén megtalálhatóak a küszöbértékek is, amelyek értékei nem a memóriában tárolódnak, hanem az adatbázisban. Így garantált, hogy egy növény lekérdezésekor, a küszöbértékek is a lehető legfrissebbek lesznek, ha azok esetleg módosításra kerültek.

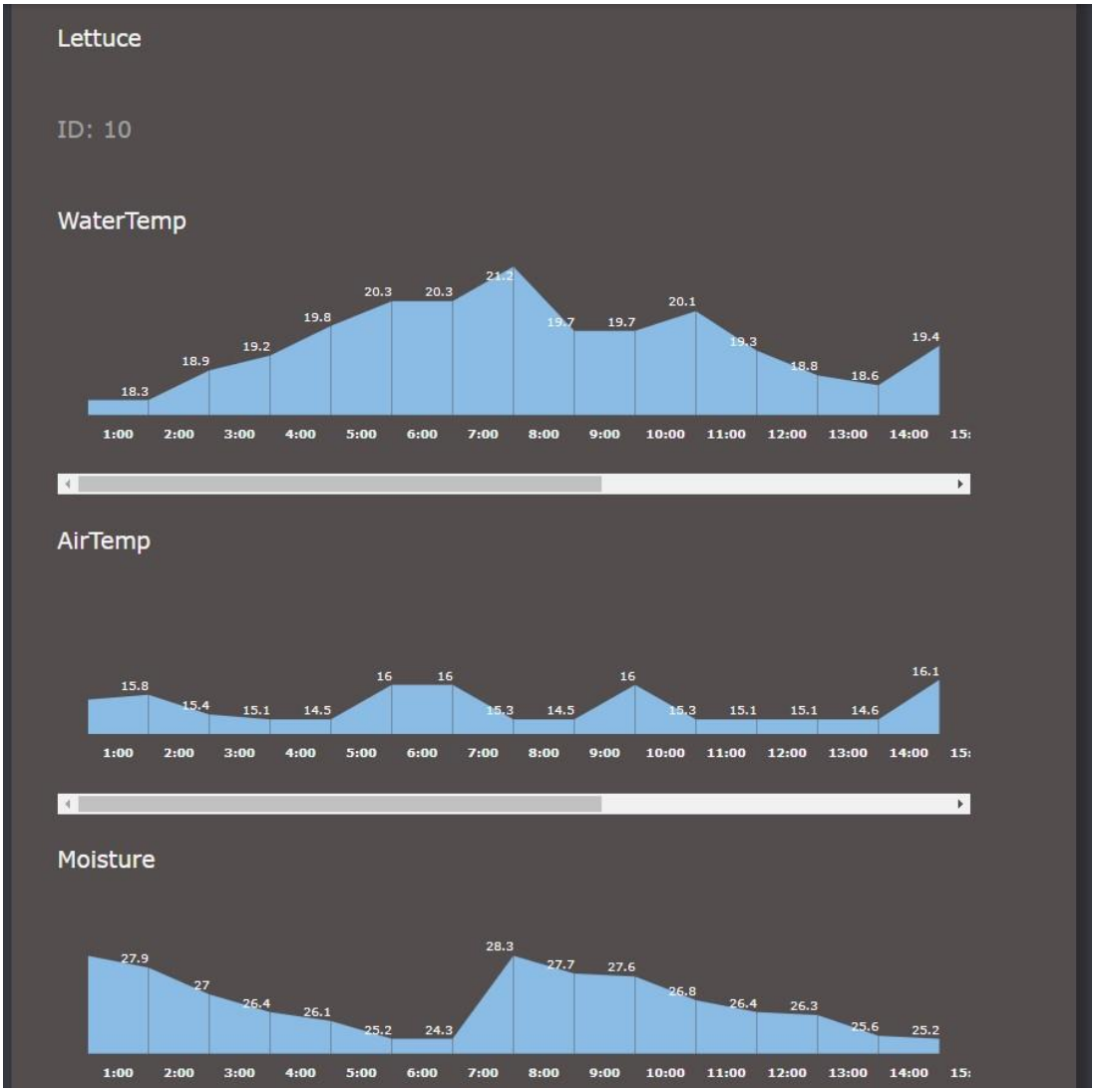
A folyamatsáv pontos hosszúságát úgy állapítom meg, hogy kivonom a küszöbérték felső határából és a beolvasott szenzor értékéből a küszöbérték alsó határát, majd az így kapott számokat elosztom és megszorzom százzal. Így kapok egy számot 0 és 100 között, amely egy százalékértéknek felel meg. Ezt a százalékértéket társítom hozzá az adott folyamatsáv szélességéhez a CSS kódjában.

A konténer jobb szélén található zöld gombok segítségével lehet manuális beavatkozásra parancsot leadni. A balra mutató nyíllal ellátott gombok a hozzájuk tartozó értékek csökkentésére utaló parancsok kiadására szolgálnak, míg a jobbra mutató nyíllal ellátott gombok a hozzájuk tartozó értékek növelésére utaló parancsok kiadására szolgálnak. A szimuláció korlátozásai miatt, a manuálisan kiadott parancs nem tud azonnal végrehajtódni, ezért úgy oldottam meg a manuálisan kiadott parancs kiértékelését, hogy a parancs beépül a kiértékelést végző automatizmus parancslistájába. Az automatizmus elvégzi a kiértékelést, majd előállítja a parancslistát. Amikor ez megtörténik, a manuálisan kiadott parancs felülíródik a hozzátartozó növénynek a parancs objektumában. Ehhez azonban még a felülírás előtt a manuálisan kiadott parancsot a már korábban említett parancs kódok szerint formázni kell. A formázás után megtörténhet a felülírás. A szimuláció következő ciklusában a növények módosítását végző metódus, végighalad a parancslistán, és végrehajtja a parancsokat. Nem tesz különbséget manuálisan kiadott, vagy automatikusan kiadott parancs között.

Mivel az oldalon több növény megjelenítése is lehetséges egyszerre, ezért új növény megjelenítése mellett, meglévő növény konténerének az eltávolítása is szükséges lehet. Ha a felhasználó elakar távolítani egy növényt a kirajzolt növények listájából, azt az adott növény kártyájának a jobb felső sarkában található piros X-szel teheti meg. A piros X-re nyomva egy javascript metódus fut le, ami azonnal hivatkozni tud a saját magát tároló HTML objektumra, így a törlés azonnal, keresés nélkül megtörténhet.

A navigációs sávon a Today menüpontra kattintva, megjelenik a napi adatokat mutató oldal. A napi adatokat mutató oldal szerkezeti felépítése hasonló a korábban említett élő

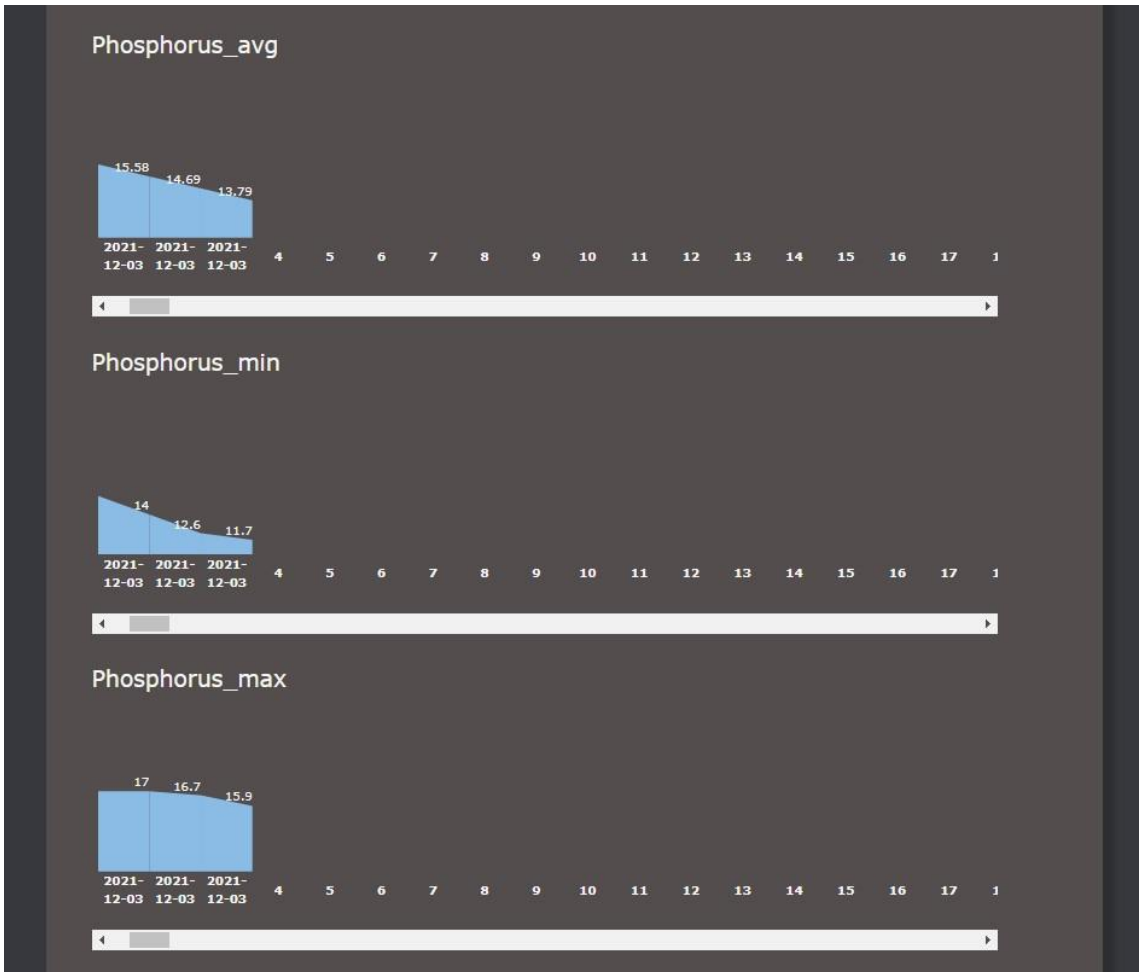
adatokat megjelenítő oldal felépítéséhez. Ez az oldal is ugyanúgy rendelkezik egy keresési mezővel, amely segítségével lekérdezhetünk egy tetszőleges növényt.



6.12. Ábra: Egy saláta napi adatai

A 6.12. ábrán a 10-zes azonosítószámú salátának látható a víz hőmérsékletére, levegő hőmérsékletére, és nedvességi szintjére utaló napi adatok grafikonja. Az itt megjelenő adatoknak nincsenek semmilyen extra számítási igényeik. A napi értékek között nem átlagolt, minimum vagy maximum értékek jelennek meg, hanem óránként egyszer az 5 percenként történő szenzor beolvasás eredményei kerülnek elmentésre a korábban már említett óránként mentett kollekcióba. Ez a kollekció a forrása a jelenleg tárgyalt oldalon történő lekérdezéseknek.

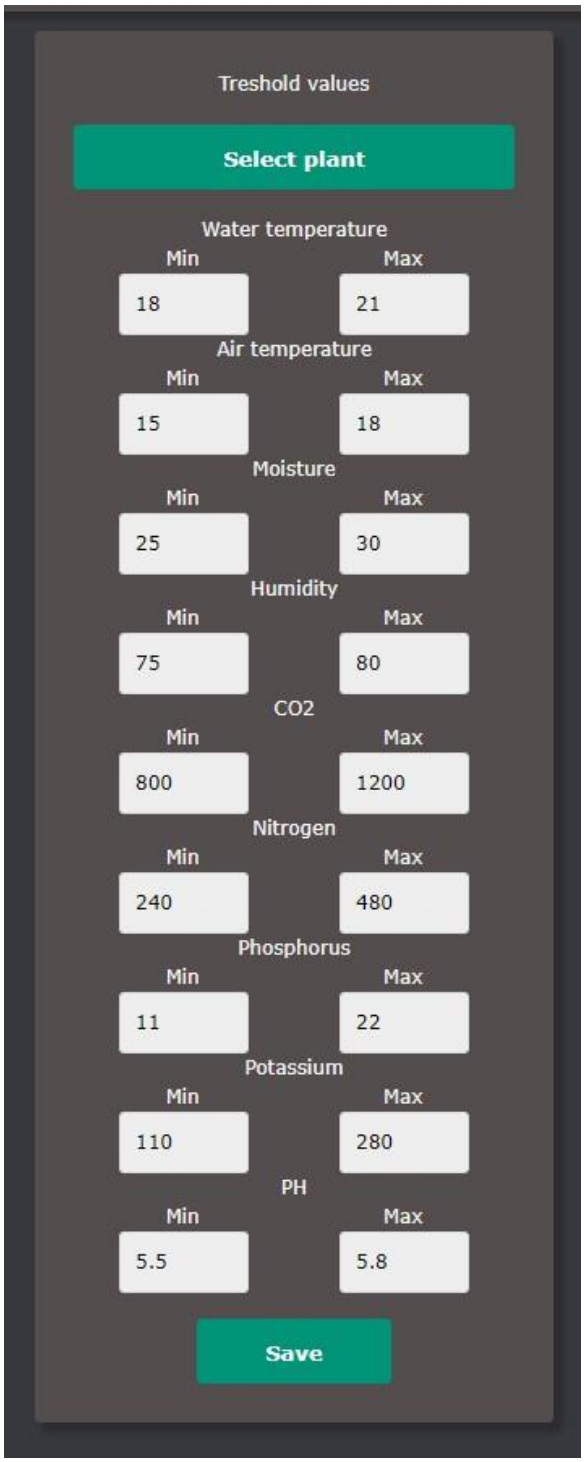
A navigációs sáv harmadik menüpontjára, a „This year” gombra kattintva, betöltődik az az oldal, amelyen megtekinthetjük a napi statisztikákat emeletekre bontva egy évre visszamenőleg.



6.13. Ábra: Egy emelet napi foszfor szintjének statisztikái

A 6.13. ábrán egy részlete látszódik egy emelet több napi átlag, minimum és maximum statisztikáinak. Az ábrán látható módhoz hasonlóan a növények többi tulajdonságainak a statisztikái is megtalálhatóak. A grafikon egyes hasábjai alatt látható a statisztikák létrehozásának a dátuma, amelyek ebben az esetben a gyorsított szimuláció miatt ugyan azzal a dátummal szerepelnek.

A negyedik oldal tartalmazza a küszöbértékek állítására szolgáló felületet. A felhasználó által módosítani kívánt növény kiválasztását egy legördülő menüből teheti meg. A növény kiválasztása utána a beviteli mezők a megfelelő növény jelenlegi küszöbértékeivel töltődnek fel. A küszöbértékeket a felhasználó felülírhatja, majd a mentés gombra kattintva az új értékek frissülnek az adatbázisban. A 6.14. ábrán egy salátának a küszöbértékei látszódnak.



6.14. Ábra: Küszöbérték állítására szolgáló felület

7. Tesztelés

7.1. Tesztelési tervek

7.2.2. Szimulációs oldalhoz tartozó tesztek

A szimulációs oldal tesztelésénél először azt vizsgáltam meg, hogy a növények tulajdonságait tároló szöveges fájlban ténylegesen 1 millió sor van a teljes folyamat ideje alatt és nem lép fel adatvesztés a szimuláció során. Az egyetlen olyan idő periódus amikor a fájlban nem egymillió sor van akkor történik, amikor a rendszer megnyitja a fájlt módosításhoz. Módosítás ideje alatt, a fájl tartalmát nem lehet látni. Ha a fájlt olyankor nem lehet megnyitni, és az operációs rendszer azt írja, hogy 0 MB a mérete, akkor a szimulációt végző program valóban megnyitotta a fájlt.

Másik fontos tesztelendő funkció, hogy a szimuláció ténylegesen a parancslista alapján hajtja végre a növények módosítását. Például amikor egy növényt öntözni kell, akkor a szimuláció lefutása után a fájlban látható növény nedvességi értéke valóban növekedett.

Végül fontos letesztelni, hogy hogyan reagál a rendszer arra a sarkalatos esetre, amikor adatküldés közben szakad meg a hálózati kapcsolat.

Teszt	**Elvárt működés**
Fájl sorainak száma módosításon kívül	Mindig egymillió sor
Fájl elérhetősége módosítás ideje alatt	Nem lehet megnyitni a fájlt és mérete 0 MB
Parancslista értelmezése	Minden paramétert sikeresen számításba vesz
Hálózati kapcsolat megszakadása adatküldés ideje alatt	A szimulációs oldalon nem vesznek el adatok és nem befolyásolja a szimulációs számításokat

7.1. Táblázat: Szimulációs oldal teszt esetei

7.2.3. Backend oldalhoz tartozó tesztek

Az első és talán legfontosabb teszt, hogy a vertikális kert felől érkező növényi adatok, valóban maradéktalanul továbbításra kerülnek az adatbázis szerverhez. Másik fontos tesztelendő funkció, hogy a parancslista előállítása során, valóban mindig a legfrissebb küszöbértékeket használja fel a rendszer, és helyes parancsokat állít elő. Fontos kérdés még hogy mi történik, ha pont a szenzor adatok beolvasásakor szakad meg a hálózati

kapcsolat. Ekkor a backend és az adatbázis oldalán sajnos ideiglenesen adatvesztés történhet, de a program semmiképpen sem állhat le.

Teszt	Elvárt működés
Növények továbbítása az adatbázis szerverhez	Maradéktalanul megtörténik
Küszöbértékek használata	Mindig a legfrissebbet használja a rendszer
Parancslista előállítása	Küszöbértékeknek megfelelő
Hálózati kapcsolat megszakadása	Nem jár a program leállásával
Statisztikák előállítása	Statisztikák helyesen előállítódnak és feltöltésre kerülnek az adatbázisba

7.2. Táblázat: Szerver oldal teszt esetei

7.2.4. Frontend oldalhoz tartozó tesztek

A felhasználói felület tesztelése során fontos megvizsgálni, hogy minden HTML oldal sikeresen megjelenik azok lekérésekor. Ha az oldalak megjelennek akkor azok funkcionalitását is érdemes tesztelni, mint például a kereső működését, hogy valóban annak a növénynek vagy emeletnek az adatait gyűjti be, amelyre a felhasználó rákeresett.

Fontos tesztelendő funkciók még azok az interakciók, amelyekkel a felhasználó manuális beavatkozást tud végezni. Ezek a küszöbértékek módosításai, és a manuális parancsok kiadásai.

A hálózati kapcsolat megszakadásának kérdése itt is felmerül. Ha a felhasználói felület és a szerver között merül fel a kapcsolat megszakadása, akkor a már megjelenített kilistázott adatoknak nem szabad elveszniük, a felhasználó továbbra is megtekintheti őket, azonban új adat lekérése nem lehetséges.

Ha a felhasználó nem létező azonosítószámra próbál hivatkozni, akkor a rendszernek nem szabad összeomlania.

Teszt	Elvárt működés
Különböző oldalak elérése	HTML oldalakat sikeresen ellehet érni a navigációs sáv használatával
Kereső funkció működése	Beviteli értéknek megfelelő növény vagy emelet adatait gyűjti be a program

Új adatokat megjelenítő konténer hozzáadása	Új lekérdezés esetében az előző eredmény nem törlődik, hanem listába rendeződve eltolódik
Megjelenített konténer bezárása	A piros X-re kattintva a megfelelő konténer bezárul
Manuális parancs kiadás	Parancsok sikeresen hozzáadódnak a parancslistához
Küszöbértékek állítása	Küszöbértékek sikeresen felülíródnak az adatbázis szerverben
Hálózati kapcsolat megszakadása	Már megjelenített adatok továbbra is megmaradnak, de új adatlekérés már nem lehetséges
Hivatkozás nem létező adatra	A rendszernek nem szabad összeomlania

7.3. Táblázat: Felhasználói felület teszt esetei

7.2. Tesztelési eredmények

7.2.2. Szimulációs oldalhoz tartozó teszt eredmények

Teszt	Eredmény
Fájl sorainak száma módosításon kívül mindig egymillió sor	Sikeres
Módosítás ideje alatt nem lehet megnyitni a fájlt és mérete 0 MB	Sikeres
Parancslista értelmezésekor minden paramétert sikeresen számításba vesz	Sikeres
A szimulációs oldalon nem vesznek el adatok és nem befolyásolja a szimulációs számításokat hálózati kapcsolat megszakadása	Sikeres

7.4. Táblázat: Szimulációs oldal teszt esetei

7.2.3. Backend oldalhoz tartozó teszt eredmények

Teszt	Elvárt működés
Növények továbbítása az adatbázis szerverhez maradéktalanul megtörténik	Sikeres
Küszöbértékek használatakor mindig a legfrissebbet használja a rendszer	Sikeres
Parancslista előállítása küszöbértékeknek megfelelő	Sikeres
Hálózati kapcsolat megszakadása nem jár a program leállásával	Sikeres
Statisztikák helyesen előállítódnak és feltöltésre kerülnek az adatbázisba	Sikeres

7.5. Táblázat: Szerver oldal teszt esetei

7.2.4. Frontend oldalhoz tartozó teszt eredmények

Teszt	Elvárt működés
HTML oldalakat sikeresen ellehet érni a navigációs sáv használatával	Sikeres
Beviteli értéknek megfelelő növény vagy emelet adatait gyűjti be a program	Sikeres
Új lekérdezés esetében az előző eredmény nem törlődik, hanem listába rendeződve eltolódik	Sikeres
A piros X-re kattintva a megfelelő konténer bezárul	Sikeres
Manuális parancs kiadásakor a parancsok sikeresen hozzáadódnak a parancslistához	Sikeres

Küszöbértékek módosításakor, azok sikeresen felülíródnak az adatbázis szerverben	Sikeres
Hálózati kapcsolat megszakadásakor a már megjelenített adatok továbbra is megmaradnak, de új adatlekérés már nem lehetséges	Sikeres
A rendszer nem omlik össze nem létező adatra való hivatkozáskor	Sikeres

7.6. Táblázat: Felhasználói felület teszt esetei

8. Továbbfejlesztési lehetőségek

Kétségkívül a projekt legfontosabb továbbfejlesztési lehetősége, hogy szimuláció helyett valós növényeket vizsgáló valós szenzorok felől érkezzenek adatok a szerver felé. A backend és frontend oly módon készült el, hogy nagyobb módosítások nélkül készen álljon valós szenzor adatok fogadására.

A projektben egyfajta csoportosítási rendszer kialakítása is hasznos lehet, amellyel több azonos tulajdonságú, illetve azonos igényű növényt egy növényként kezel a rendszer. Ennek az indoka, hogy a szomszédos növényeknek sok esetben hasonló tulajdonságaik vannak. Például az egy emeleten lévő növények, amelyek egy azonos elszigetelt üvegházi környezetben vannak, hasonló levegő hőmérséklet, és páratartalom tulajdonságokkal rendelkeznek.

A projekt jelenlegi állapotában feltételezve van, hogy mivel a fény egy mesterséges automatizáltan irányított forrásból érkezik 16 óra világítással és 8 óra pihentetéssel, így a fény nem egy változó elem, ami azt jelenti hogy szenzorral való vizsgálata sem feltétlen célszerű, azonban ha üvegházhoz hasonló üvegfalakkal van ellátva a vertikális kert, sőt más egyéb természetes fény elosztását segítő technológiákat is alkalmazunk, mint például tükrök amelyek segítségével az emeletek közepére is fényt lehetne juttatni, így a fény szenzorokkal a természetes fényt is számításba vehetjük, ami által a mesterséges fény csak kiegészítő szereppel rendelkezne, ami viszont jelentősen csökkentené az áramfogyasztást.

A felhasználói felületen is vannak továbbfejlesztési lehetőségek. Például több fajta statisztika előállítása és kimutatása. A felület kinézetének tovább optimalizálása, hogy több információt lehessen a felhasználó számára kényelmesebben átadni.

9. Összefoglalás

A szakdolgozat célja egy olyan rendszer kialakítása, amely alkalmas nagymennyiségű növény monitorozására és automatizálására. A témaválasztás indoka, hogy személyes véleményem szerint az olyan eljárások, amelyek segítségével kevesebb helyen tudunk több élelmiszert termelni, a jövőben egyre nagyobb prioritást fognak élvezni. A vertikális kertek olyan sok emeletes toronyépületek, amelyek segítségével kis alapterületen, nagy mennyiségű növény termesztése lehetséges. A szakdolgozat írásának idejében nagy mennyiségű növény termelésére alkalmas vertikális kertek jelenleg még nem léteznek. Azonban különböző technológiák fejlődése és a területet érintő kutatások hirtelen megszaporodása egyre inkább lehetővé teszik, hogy a vertikális kertek a közeljövőben valóságosak legyenek. Technológiák, mint például a hidroponikus kertészet, üvegházak fejlődése, megújuló energia termelési technológiák fejlődése és szenzorok fejlődése, valamint kutatások, mint például a nemzetközi űrállomáson végzett mesterséges fénnyel növesztett növényeket érintő kutatás fokozatosan közelebb hozzák a vertikális kerteket a realitáshoz.

Az irodalomkutatás során megvizsgáltam a történelem során előforduló különböző növénytermelési technológiákat, amelyek részben hozzájárulnak a vertikális kertek potenciális létezéséhez. Utána olvastam, hogy pontosan milyen fajta szenzorok léteznek, valamint ezek pontosan milyen tulajdonságokkal rendelkeznek. Különböző informatikai technológiákat vizsgáltam meg, amelyek segítségével nagy mennyiségű adatot lehet hatékonyan kezelni, és tárolni, valamint olyan megoldásokat, amelyekkel olyan komplex rendszereket lehet készíteni, amelyek segítségével a felhasználók nem csak megtekinteni tudják a tárolt adatokat, hanem interakciókkal befolyásolni tudják a rendszer működését.

Specifikáltam, hogy pontosan milyen funkcionalitásokkal rendelkezik a projekt, milyen használati esetek merülhetnek fel, és hogy a növények gondozását végző automatizmus pontosan milyen reakciókkal válaszol a beérkező információkra. Készítettem egy rendszertervet, amelyben felvázoltam, hogy a rendszer milyen elemekből áll, és hogy ezek az elemek milyen funkcionalitásokat látnak el, valamint, hogy a rendszer különböző komponensei között hogyan történik az adatáramlás.

Az implementáció leírása során részletesen tárgyaltam, hogy pontosan milyen szoftverfejlesztési, és adatbáziskezelési megoldásokkal oldottam meg a felmerülő fejlesztési kihívásokat. Tesztek segítségével ellenőriztem, hogy a rendszer az elvárt működésnek megfelel, és hogy egyes sarkalatos esetekben is képes tovább működni összeomlás nélkül. Végül arról írtam hogyan lehet a projektet jelenlegi állapotához képest továbbfejleszteni.

A szakdolgozat készítése során sok olyan technológiát használtam, amelyekkel előtte még nem találkoztam, vagy csak kevés tapasztalatom volt vele. A projekt által rengeteg hasznos tudásra tettem szert, amelyeket a munkaerő piacon kamatoztatni tudok.

10. Summary

The purpose of this thesis is to create a system which is able to monitor and automate large amount of data from plants. The reason of this topic is that in my personal opinion methods which are able to produce large amount of food in smaller areas, are going to be more and more important in the future. Vertical gardens are buildings that are able to produce large amount of food on a small architectural footprint. In the time of writing this thesis, currently no such vertical garden exists. However various technological developments and the rapid growth in the number of researches in this area suggests that vertical gardens may be a possibility in the near future. Technologies such as hydroponics, development of greenhouses and renewable energy sources, development of sensors, and researches such as the one conducted on the International Space Station which are helping the understanding of the relation between plants and artificial light, are bringing vertical gardens closer to reality.

During my researches I examined various farming technologies used throughout history, that are contributing to the potential existence of vertical gardens. I have read about sensors and their properties. I examined various technologies that help to manage and store large amount of data efficiently, and technologies which help the user not only to observe the data, but to interact with the whole system.

I specify the exact functions of the project, and use-cases that can occur, also the exact responses of the automation that controls the development of the plants to incoming information. I made a system plan which shows the different components of the system, and the means of data flow between them.

I wrote about the exact software development, and database methodologies that I used in response of the occurring developing challenges. With the help of various tests, I made sure that the system works as intended, and does not crash when unfavorable environmental events occur. At last, I wrote about the possibilities of upgrading the current project.

During the creation of this thesis, I used a lot of technologies which I never met before or had little experience with. Because of this project I learned a lot of useful skills, that I can use in the labor market.

Irodalomjegyzék

[1] ALLEN, Julia C.; BARNES, Douglas F. The causes of deforestation in developing countries. *Annals of the association of American Geographers*, 1985, 75.2: 163-184.

[2] SALAS, M. C., et al. Vertical gardening. Adaptation of hydroponic systems and ornamental species. In: *XXVIII International Horticultural Congress on Science and Horticulture for People (IHC2010): International Symposium on 937*. 2010. p. 1153-1160.

[3] JENSEN, Merle H. Hydroponics. *HortScience*, 1997, 32.6: 1018-1021.

[4] RAMANE, Deepa V.; PATIL, Supriya S.; SHALIGRAM, A. D. Detection of NPK nutrients of soil using Fiber Optic Sensor. In: *International Journal of Research in Advent Technology Special Issue National Conference ACGT 2015*. 2015. p. 13-14.

[5] RESH, Howard M. *Hydroponics for the home grower*. CRC Press, 2015.

[6] ANDREW C. SCHUERGER, CHRISTOPHER S. BROWN, ELIZABETH C. STRYJEWSKI, Anatomical Features of Pepper Plants (Capsicum annuum L.) Grown under Red Light-emitting Diodes Supplemented with Blue or Far-red Light , *Annals of Botany*, Volume 79, Issue 3, March 1997

[7] GENT, M. P. N. HortScience: a publication of the American Society for Horticultural Science. *HortScience*, 2007, 42.3: 514-520.

[8] YORIO, Neil C., et al. Improving spinach, radish, and lettuce growth under red light-emitting diodes (LEDs) with blue light supplementation. *HortScience*, 2001, 36.2: 380-383.

[9] BAHARVAND, Mohammad. SINCE ANCIENT PERSIAN GARDENS TO MODERN VERTICAL GARDENS AND ROOF GARDENS. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 2020, 17.4: 3159-3170.

[10] PARIS, H. S.; JANICK, J.; PITRAT, M. What the Roman emperor Tiberius grew in his greenhouses. 2008.

[11] NASA: Veggie, (https://www.nasa.gov/sites/default/files/atoms/files/veggie_fact_sheet_508.pdf), utoljára megtekintve: 2021.05.05

[12] LENNARD, Wilson A.; LEONARD, Brian V. A comparison of three different hydroponic sub-systems (gravel bed, floating and nutrient film technique) in an aquaponic test system. *Aquaculture International*, 2006, 14.6: 539-550.

[13] RESH, Howard M.Hydroponic food production: a definitive guidebook for the advanced home gardener and the commercial hydroponic grower. CRC Press, 2012.

[14] GAY, Warren. DHT11 sensor. In: *Advanced Raspberry Pi*. Apress, Berkeley, CA, 2018. p. 399-418.

[15] ZHAO, Xuefeng, et al. Active thermometry based DS18B20 temperature sensor network for offshore pipeline scour monitoring using K-means clustering algorithm. *International Journal of Distributed Sensor Networks*, 2013, 9.6: 852090.

[17] Nedvesség sensor, (https://how2electronics.com/interface-capacitive-soil-moisture-sensor-arduino/), utoljára megtekintve: 2021.05.11

[18] Flutter, (https://flutter.dev/?gclid=CjwKCAjw1uiEBhBzEiwAO9B_Hd-2bsq5gBXMkyxIBX_5rvhoCzvQJN5GW3tM5-vuOC6kkUcMzi8a5xoCJhoQAvD_BwE&gclsrc=aw.ds), utoljára megtekintve: 2021.05.11

[19] MongoDB, (https://www.mongodb.com/nosql-explained/nosql-vs-sql), utoljára megtekintve: 2021.05.11

[20] MEHRA, Manav, et al. IoT based hydroponics system using Deep Neural Networks. *Computers and electronics in agriculture*, 2018, 155: 473-486.