МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук,
управління та адміністрування

Кафедра Інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка системи моніторингу та керування екологічними показниками для забезпечення життєвого циклу рослини

Виконав студент 2 курсу групи МІС-20 спеціальності 122 Комп'ютерні науки

Грабіна Віталій Віталійович

Керівник д.т.н., професор
Казакова Надія Феліксівна

Рецензент к.т.н., доцент Гнатовська Г.А.

Одеса 2021

АНОТАЦІЯ

на магістерську кваліфікаційну роботу
«Розробка системи моніторингу та керувння екологічними показниками для забезпечення життєвого циклу рослини»,
студента Грабіни Віталія Віталійовича

Актуальність вирішення проблем моніторингу та керувння екологічними показниками для забезпечення життєвого циклу рослин полягають в тому, що хоча й існує низка відомих систем, але вони немають достатнього функціоналу, та є направленими на виконання інших завдань.

Мета роботи –розробка мережевої системимоніторингу та керування екологічними показниками для забезпечення життєвого циклу рослин.

Об'єкт дослідження – існуючі системи моніторингу життєдіяльності рослин.

Методи дослідження – теоретичне ознайомлення з існуючими системами, призначеними для моніторингу та керуванням показниками життєдіяльності рослини, моделювання архітектури застосунку, методи об'єктно-орієнтованого програмування.

Проведено аналіз систем моніторингу з подібним функціоналом, здійснено порівняльну характеристику, виявлено переваги та недоліки.

Магістерська кваліфікаційна робота містить 70 сторінок, 47 рисунківта 19 джерел.

Ключові слова: GO, МЕРЕЖЕВА СИСТЕМА, VISUALSTUDIOCODE, АРХІТЕКТУРА ONION, POSTMAN, DOCKER.

SUMMARY

for a master'sdegree

«Developmentof a systemofmonitoringandmanagementofenvironmentalindicatorstoensurethelifecycle oftheplant»,

students ofHrabinaVitalii

Theurgencyofsolvingtheproblemsofmonitoringandmanagingenvironmentalindicatorstoensurethelifecycleofplantsisthatalthoughthereare a numberofknownsystems, buttheyarenotsufficientlyfunctional, andareaimedatothertasks.

Thepurposeoftheworkistodevelop a networksystemformonitoringandmanagingenvironmentalindicatorstoensurethelifecycleofplants.

Theobjectofresearchistheexistingsystemsofmonitoringthevitalactivityofplants.

Researchmethods - theoreticalacquaintancewiththeexistingsystemsdesignedformonitoringandcontrolofplantlifeindicators, modelingofapplicationarchitecture, methodsofobject-orientedprogramming.

Theanalysisofmonitoringsystemswithsimilarfunctionalityiscarriedout, thecomparativecharacteristiciscarriedout, advantagesandlacksarerevealed.

Themaster'sthesiscontains 70 pages, 47drawingsand19sources.

Keywords: GO, NETWORK SYSTEM, VISUAL STUDIO CODE, ONION ARCHITECTURE, POSTMAN, DOCKER.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

NDVI (NormalizedDifferenceVegetationIndex) – нормалізований індекс відносної рослинності

NDRE (NormalizedDifferenceRedEdge) − нормалізована різничний індекс рослинності, визначений за допомогою інфрачервоного каналу

MSAVI — ((ModifiedSoilAdjustedVegetationIndex)– модифікований виправлений грунтовий індекс

IDE(IntegratedDriveElectronics) – інтегрована середа розробки

TCP (TransmissionControlProtocol) – протокол керування передачею

UDP (UserDatagramProtocol) – протокол користувальницьких дейтаграм

API (ApplicationProgrammingInterface) – програмний інтерфей додатку

JSON (JavaScriptObjectNotation) - текстовий формат обміну даними, що базується на JavaScript

AJAX (AsynchronousJavaScriptand XML)- асинхронний JavaScript та XML

Npm(NodePackageManager)- менеджер пакетів, що входить до складу Node.js

ОСоперaційна система

API (applicationprogramminginterface)- програмний інтерфейс програми

REST(RepresentationalStateTransfer)- передача репрезентативного стану

URL(UniformResourceLocator) - уніфікований покажчик ресурсу

SQL(structured query language)- мова структурованіх запитів

NoSQL (notonly SQL)- не тільки SQL

БД база даних

PHP (PersonalHomePage)скриптова мова загального призначення, що інтенсивно використовується для розробки веб-додатків

СУБД - Система управління базами даних

IDE (IntegratedDevelopmentEnvironment) – інтегроване середовище розробки

ASP.NET(ActiveServerPages для .NET) платформа розробки вебдодатків

JWT(JSONWebToken) -цевідкритий стандарт для створеннятокенів доступу

BSON(BinaryJavaScriptObjectNotation) - формат електронногообмінуцифровимиданими

RAW формат даних, щоміститьнеоброблені (абообробленімінімально) дані

ВСТУП

Майже в кожної людини вдома є домашні рослини, дехто вирощує вдома, навіть, сільськогосподарські. Зараз стає все популярнішою темою в великих містах оренда невеликих ділянок землі для того, щоб вирощувати на них власні – екологічно чисті, без додавання пестицидів та хімікатів овочі.[1][1)].

Пестициди – одна з головних причин смерті внаслідок отруєння, особливо у країнах із низьким та середнім рівнем доходів.

Споживачі можуть обмежити потрапляння в організм пестицидів, що містяться в залишковій концентрації у продуктах харчування, за допомогою миття фруктів та овочів та видалення шкірки. Це також сприяє скороченню впливу інших джерел небезпеки, пов'язаних з продуктами харчування, таких як хвороботворні бактерії.

Ми живемо в час, коли технології дуже швидко розвиваються і кожна людина не може уявити своє без гаджетів, які спрощуютьжиттяв різних сферах, починаючи з замовлення їжі на дім і закінчуючи управлінням домашньою технікою за допомогою смартфону. Тому постало питання розробки системи, здатної аналізувати та керувати показниками життєдіяльності рослин.

Існуючі аналоги схожих систем недостатньо підходять для вирішення поставленого завдання. У всіх них є низка мінусів, наприклад велика ціна, чи можливість аналізу лише з космосу. Тому було вирішено створити програмний продукт, який би зміг впоратися зі всіма проблемам та не мав тих проблем, які присутні в його аналогах.

Основною метою даної роботи стало створення системи аналізу та керування показниками життєдіяльності рослини, неважливо від того, чи стоїть у користувача горщик з квітками на вікні, або в нього велика теплиця за містом.

[1)][1]СельскоехозяйствоУкраины – чегождать в будущем?. URL: https://www.fao.org/europe/news/detail-news/ru/c/447171/ (дата звернення 09.07.2021).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі будуть розглянуті схожі додатки, а також буде проведено аналіз із визначення переваг та недоліків.

Актуальність і невідкладність розробки програмної системмоніторингу та керування над показниками рослини полягають в тому, що хоча й існує низка відомихпрограмнихрішень, але всі вони не відповідають всім завданням, які ми перед ними ставимо.

Готовий програмний продукт має забезпечувати наступні функціональні можливості:

− моніторингу життєвих показників рослин у реальному часі;

− керування показниками у реальному часі;

− додавання нових боксів рослин, чи редагування над показниками вже існуючих;

− функціонал ручного управління над кожним показником окремо;

− зрозумілий користувачеві інтерфейс;

− швидке додавання боксів з рослинами за допомогою QR-коду.

Моніторинг потрібен для отримання комплексної інформації про стан рослини. З його допомогою визначається рівень схожості та ступінь зараженості хворобами. На основі отриманих даних приймаються управлінські рішення по обробці ґрунту, його добриву, боротьбі зі шкідниками та зміною деяких параметрів оточення.

Загалом моніторинг ділянки с рослинами– це дуже трудомісткий та тривалий процес[2][1)].

[1)][2Мониторинг полей в сельском хозяйстве. URL: https://blog.agrokebety.com/monitoring-poley-v-selskom-khozyaystve/ (дата звернення 12.07.2020)

1.1 Аналіз ринку подібних проектів

Серед існуючого програмного забезпечення, спрямованого моніторинг та керувння екологічними показниками рослини є такі системиAquaspy[3][1)], EOS CropMonitoring[4][2)],EnviroMonitor[5][3)], PhytoVision[6][4)].

1.1.1 Огляд системиAquaspy

Система моніторингу вологи грунтуAquaSpy оснащена сенсорною системою у поєднанні з інтелектуальними інформаційними технологіями AquaSpy допомагає зберегти воду, зменшуючи вилуговування азоту, максимізуючи врожайність, одночасно зменшуючи витрати на добрива.

Дозволяє відстежувати поведінку рослин у відношенні до води та поживних речовин. 12 датчиків у кожному 48-дюймовому зонді вимірюють волгість, електропровідність та температуру. Система надає змогу дистанційного моніторингу даних за допомогою веб-інтерфейсу.

На рис. 1.1 наведене зображення веб-інтерфейсу системи.

[1)][3]МониторингвлажностипочвыAquaSpy. URL: https://www.amitytech.com/ru/crop-management-tools/aqua-spy-soil-moisture-monitoring/ (дата звернення 21.07.2021).

[2)][4]EOS CropMonitoring: Спутниковые технологии в сельском хозяйстве. URL: https://eos.com/ru/products/crop-monitoring/ (дата звернення 21.07.2021).

[3)][5]. Приложения в GooglePlay - EnviroMonitorURL: https://play.google.com/store/apps/details?id=com.davisinstruments.enviromonitor&hl=en_US&gl=US (дата звернення 21.07.2021).

[4)][6] PhytoVisionМониторинг роста растений. URL: https://paskal-group.com/ru/phytovision/ (дата звернення 21.07.2021).

Рисунок 1.1 – Інтегроване зображення рівня вологи

Однією з особливостей даної системи є інтерпретація даних електропровідності, завдяки цьому надається корисна інформація − коли відбулося вилуговування, і наскільки глибоко поживні речовини були переміщені під час вилуговування(див. рис. 1.2).

Рисунок 1.2 – Інтерпретація даних електропровідності

На рис. 1.3 зображено індикатори повноти та поржнечі. Звуковий сигнал надсилається з зонда. Швидкість і ослаблення сигналу при поверненні застосовуються для отримання корисних даних. Тоді алгоритм використовується для розрахунку точок наповнення та поповнення, що відображаються в AquaSpy.

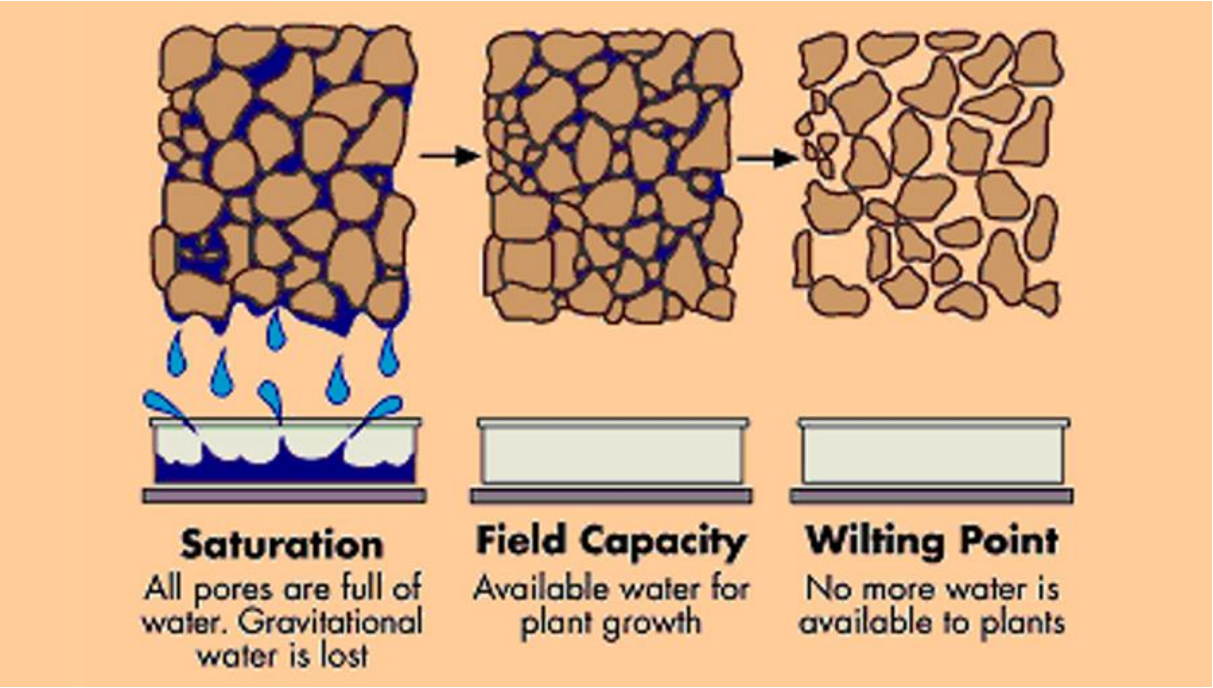

Рисунок 1.3 – Графік речовин

На жаль, система не виконує достатнього функціонала для отримання всіх необхідних данихю

1.1.2 Огляд програмного продуктуEOSCropMonitoring

Система CropMonitoringдозволяєотримати корисну інформацію на основі аналізу супутникових знімків, а саме. традиційний індекс рослинності NDVI та індекси NDRE, MSAVI. Також є можливість отримання сповіщень для зміни стану рослин, керування розвідувальними завданнями, можливість

зонування полів на основі показників рослинності, та можливість створення завдань для диференційованого застосування.

Інтерфейс забезпечує швидкий доступ до будь-якої інформації. Серед мов дисплея є українська. Меню складається з п'яти блоків − полів, погодного аналізу, польової інспекції, менеджера повідомлень, зонування.

На рис. 1.4 зображено блок аналізу погоди − для кожного поля відображаються чотири графіки: щоденна кількість опадів, кількість опадів шляхом зростаючого результату, щоденні температури (мінімум, максимум), сума активних температур.

Рисунок 1.4 – Блок аналізу погоди

Блок польової інспекції − це блок розвідки, на якому аграгрій може встановити завдання для польових досліджень, що вказують на точні координати місця та дату. Виконавець може закрити завдання, завантажувати знімки культур, польових карт і вказати результат розвідника. Усі завдання відображаються на карті точками (у місці проведення вимірів), інформація про завдання визивається після наведення курсору на точку.

Зонування грунту з точки зору зростаючого індексу виконується для подальшої компіляції диференційованих карт. Аграрій обирає один з

чотирьох показників для аналізу, дати зображення, на скільки зон необхідно розбити поле та мінімальну область однієї зони. Блок зонування зображно на рис. 1.5.

Рисунок 1.5 – Карта зонування по рівню рослиності

На рис. 1.6 зображена одна з функцій програмного продукту – алгоритм виявлення сільськогосподарських культур на основі аналізу відражальної здібності рослинного покрову на знімках супутника Sentinel 2. Щоб побачити на якій кількості полівзростає культура не потрібно їх виділяти самостійно. Завдяки цьому інструменту можна визначити потенційний попит на майбутній врожай.

Існує також велика архівна база сільськогосподарських культур на всіх полях в Україні з 2016 року.

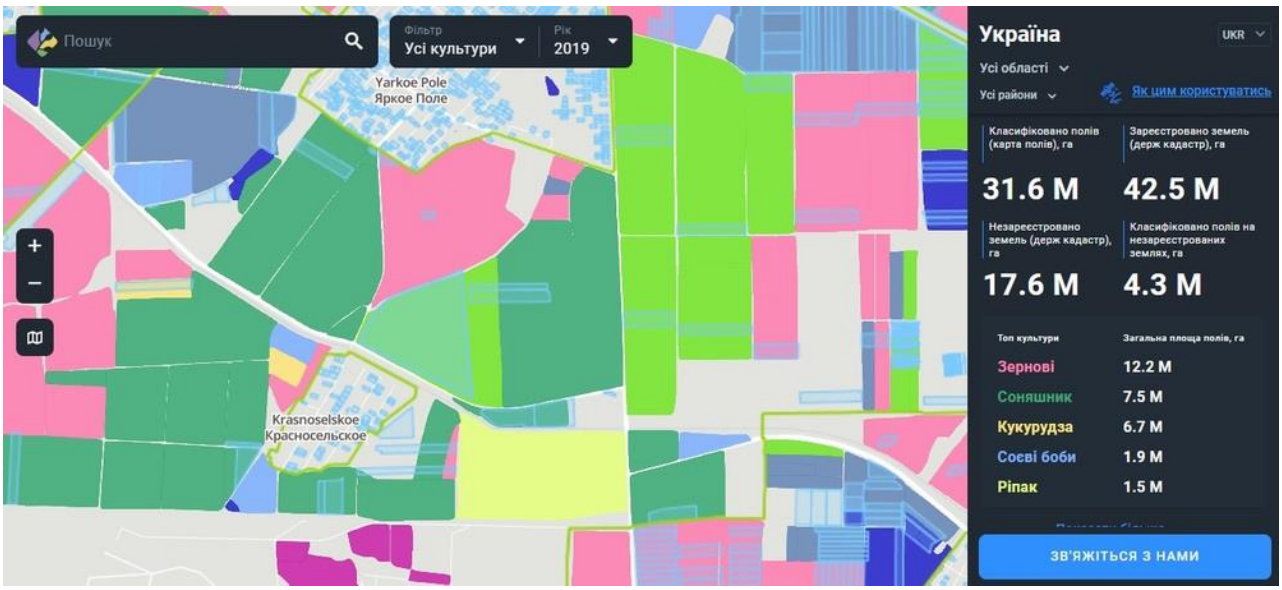

Рисунок 1.6 – Алгоритм виявлення сільськогосподарських культур

Програмний продукт надає велику кількість засобів для аналізу та моніторингу життєдіяльності сільськогосподарських рослин, але на даний час не має готових рішень керування показниками.

1.1.3 Огляд застосування EnviroMonitor

EnviroMonitor використовується для встановлення, моніторингу та взаємодії з системами апаратного та сенсорного обладнання.

Додаток надає можливість налаштувати необмежену кількість систем до своїх шлюзів, встановлюючи мережу з сіткою та встановлення будь-якого з підтримуваних середовищ. Після налаштування системи є можливість поділитися з іншими користувачами, надаючи їм доступ до всіх пристроїв.

Для повноцінної роботи з системою EnviroMonitor необхідно використовувати спеціалізовані інструменти компанії Davis, у сукупності з мобільним додатком (див. рис. 1.7).

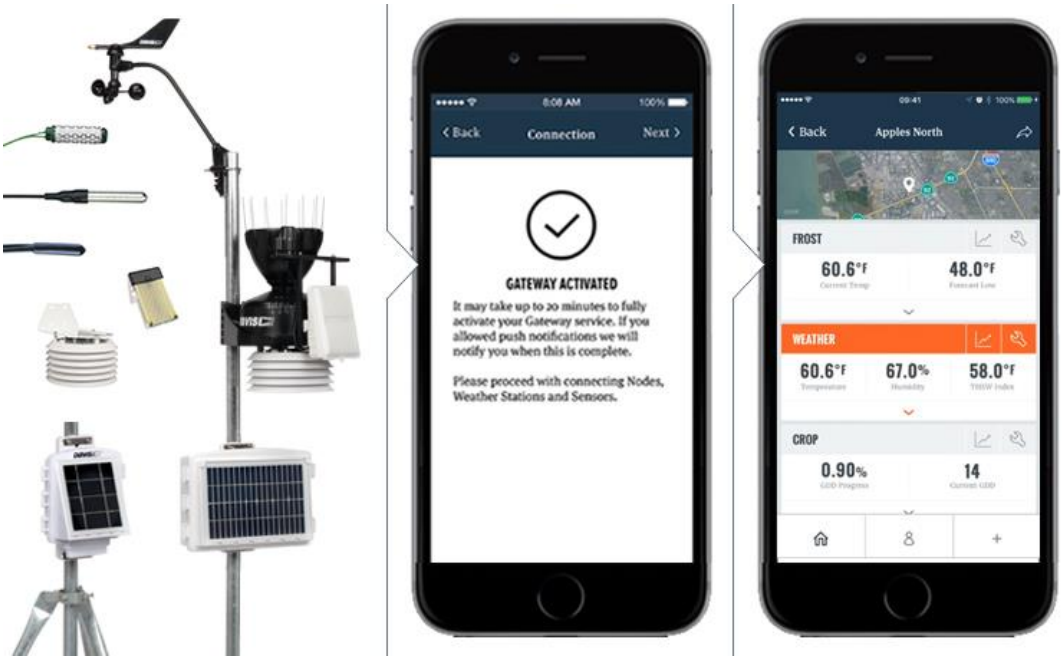

Рисунок 1.7 – Керування системою EnviroMonitor за допомогою мобільного додатку

На рис. 1.8 показано інформаційну панель додатку EnviroMonitor. Дані оновлюються кожні 15 хвилин.

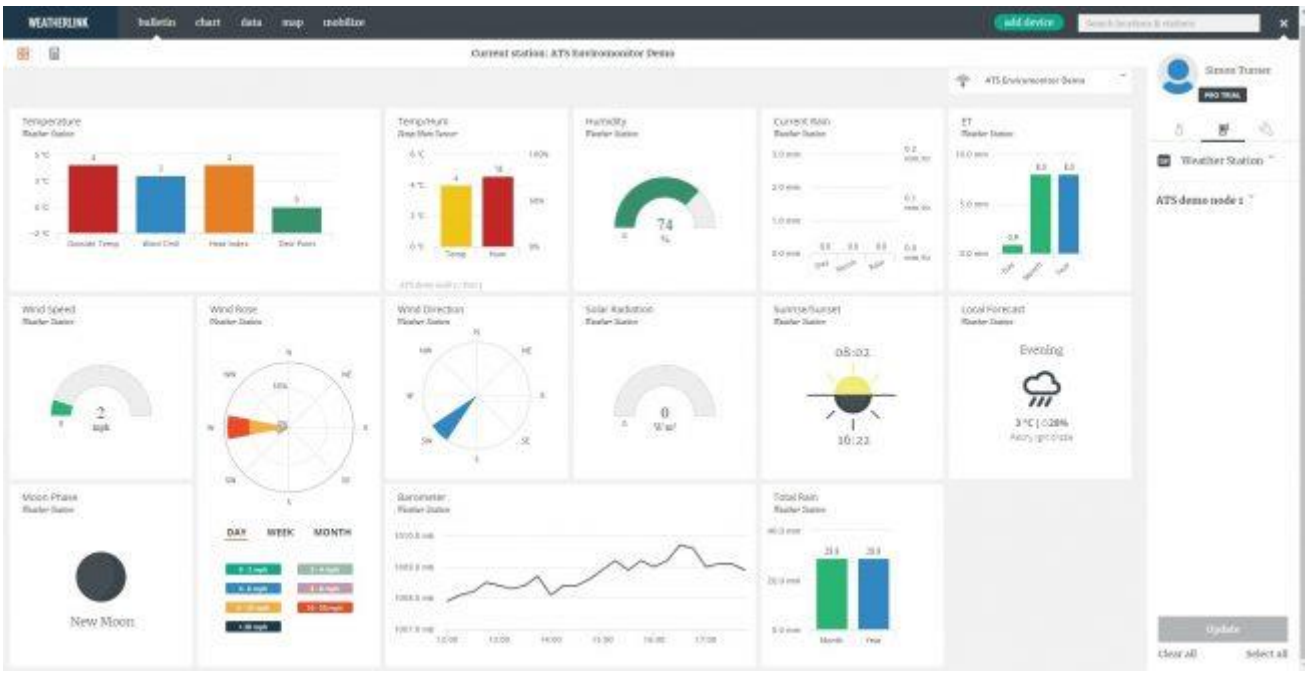

Рисунок 1.8 – Інформаційна панель

Система відправляє звіти користувачеві о стані погодних умов, а також аналізує дані та виводить інформацію у вигляді графіків (див. рис. 1.9).

Рисунок 1.9 – Звіт о стані погодних умов

Вимірюючи пристрої зв’язуються з хмарним сховищем, конвертують дані та передають їх одному, чи групі під’єднаних до системи користувачей, для подальшого аналізу (див. рис. 1.10).

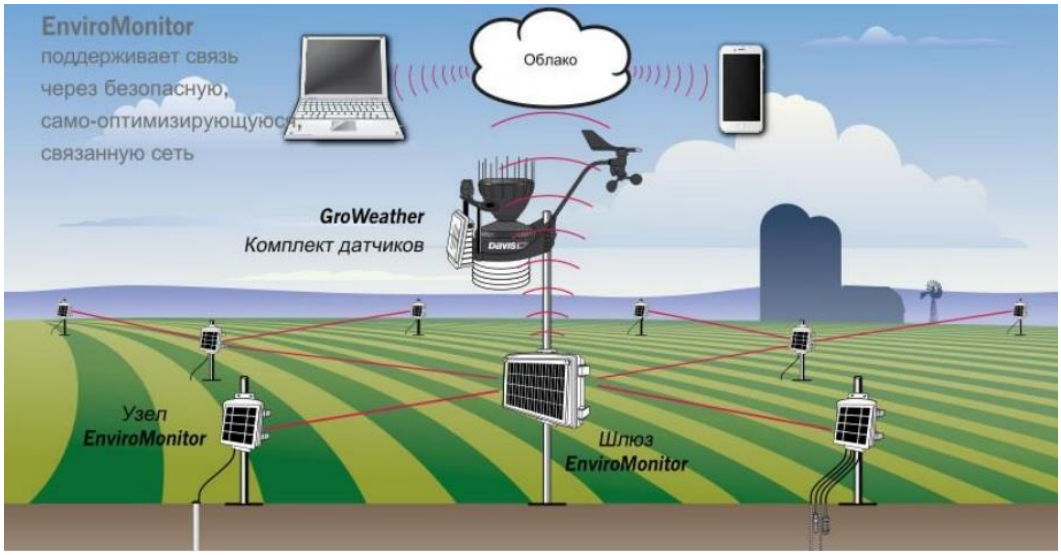

Рисунок 1.10 – Графік AQI

DavisInstruments є провідним виробником рішень для збору погодних даних для сільськогосподарського виробництва, науково-дослідних установ, університетів, шкіл і любителів погоди. Вузол EnviroMonitor – це вузол навколишнього середовища та погоди, який надсилає дані від чотирьох датчиків до шлюзу EnviroMonitor. Можна створити набір датчиків і збирати дані з мікроклімату та різних виробничих полів.

За допомогою шлюзу EnviroMonitorGatewayМожна збирати дані датчиків з до 32 вузлів навколишнього середовища та погоди. Кожен вузол передає дані до шлюзу та/або один одному, створюючи самооптимізуючу сітчасту мережу. Вузол EnviroMonitor підтримує широкий спектр датчиків даних від Davis та інших сторонніх виробників. Можна встановити до чотирьох датчиків у вузол, включаючи датчики рівня, анемометри, реле тиску, датчики температури, датчики солоності, датчики вологості ґрунту, датчики вологості листя.

Недоліком цього застосування є велика вартість пристроїв компанії Davis.

1.1.4 Огляд системи моніторингуPhytoVision

PhytoVision − це система моніторингу зростання рослин для збільшення врожаю та зниження витрат.

Застосунок надає можливістьаналізу, моніторингу та виправлення темпів зростання рослин у теплиці на основі кліматичних та зрошувальних даних. Дає оперативну оцінку показників для подальшої корекції стратегії задовго до збирання врожаю та здатність впливати на його збільшення.

Дані про вирощування рослин поступають в реальному часі на будь-якому пристрої з необмеженою кількістю користувачів (див. рис. 1.11).

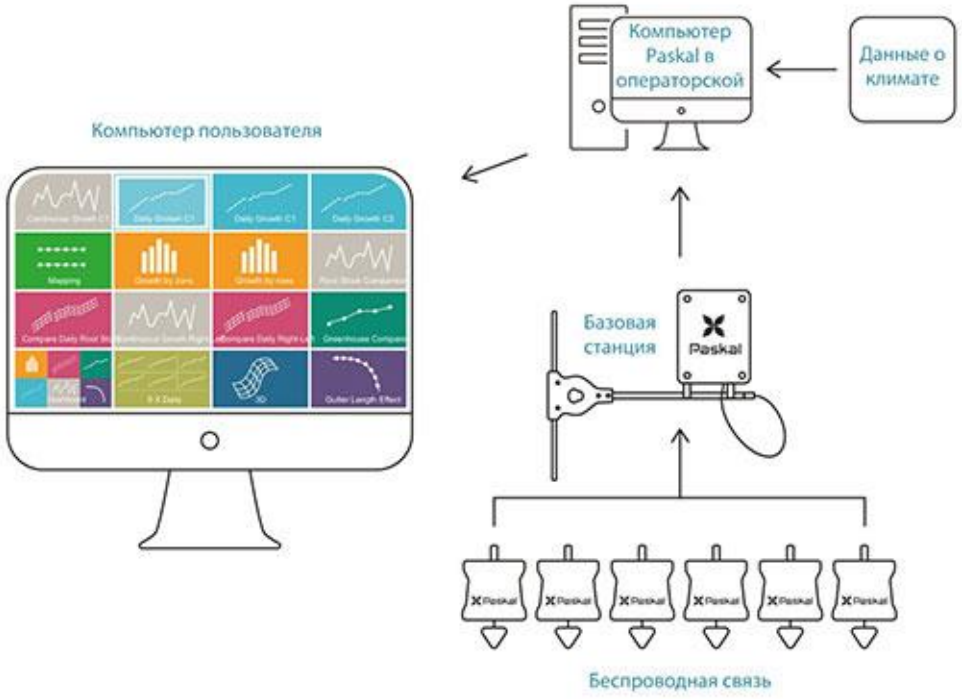

Рисунок 1.11 – Схема передачі даних з приладів користувачу

Користувач може швидко спостерігати за можливими проблемами, аналізуючи зменшення зростання (див. рис. 1.12).Можливість регулювання температури, полива або інших параметрів змінить також і темпи зростання рослин.

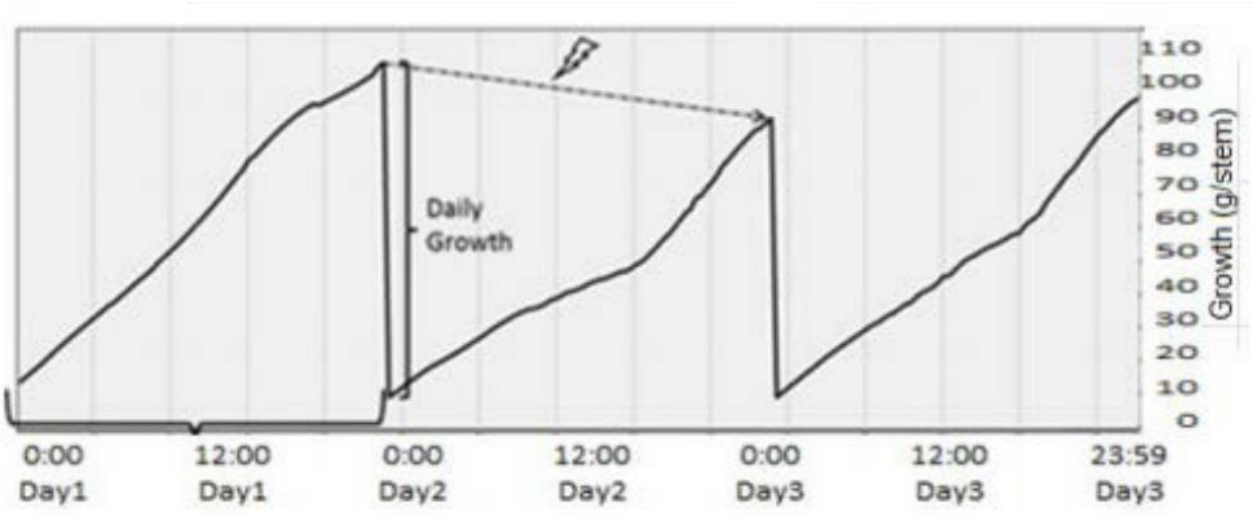

Рисунок 1.12 – Відображення зміни темпу росту рослин

Завдяки датчикам компанії PASKAL, надаються дані для побудови графіків зв’язку між зростанням ваги рослини під час вегетаційного періоду і врожаєм (див. рис. 1.13). Фактична врожайність, представлена на графікуце маса плодів, зібраних у теплиці та вага рослин, виміряна ваговими датчиками. Крива накопичення протягом попередніх трьох тижнів відображає співвідношення між збільшенням ваги рослин та розміром урожаю. Це дозволяє користувачу передбачити обсяг та час збору врожаю.

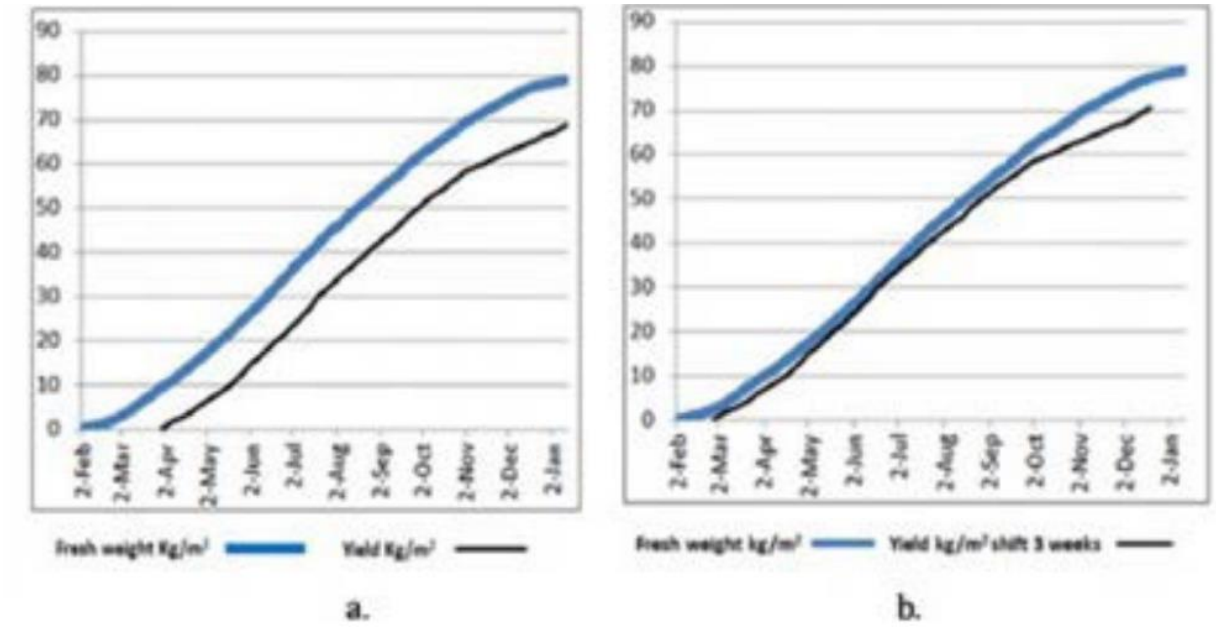

Рисунок 1.13 – Екран ранжування забруднення

Сонячне випромінювання є одним з основних параметрів, що має прямий вплив на зростання рослин.

На рис. 1.14 показано довгостроковий ефект випромінювання, та його вплив на зростання рослин.

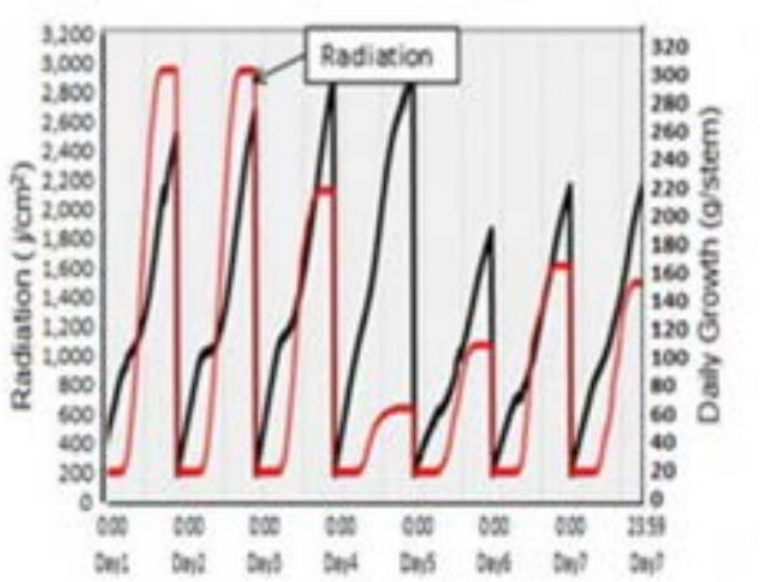

Рисунок 1.14 – Вплив сонячного випромінювання на зростання рослин

Температура також є важливим кліматичним фактором і використовується фахівцями для управління процесами росту або регулювання вегетативно-генеративного розвитку (див. рис. 1.15). В даний час агроном може отримати зворотній зв’язок від рослини про вплив температури на його зростання лише через кілька днів.

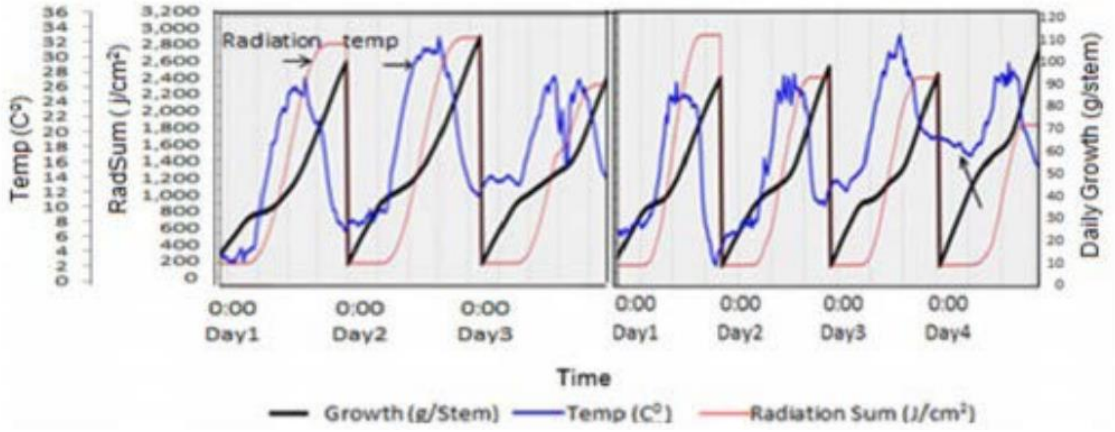

Рисунок 1.15 – Вплив на зростання денних і нічних температур

PhytoVision надає користувачам багато інструментів для моніторингу, аналізу та керуванням життєдіяльністю рослин, але є складним у використанні та більш підходящим для використання на великих аграрних підприємствах.

2 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНІЧНИХ ЗАСОБІВ

Вибір середовища розробки, мови та платформи для програмування – одна з найважливіших частин під час розробки програмного продукту.

2.1 Програмування мережевих завдань

У сфері комп'ютеризації, поняття програмування мережевих завдань або, як його ще називають − мережеве програмування, досить схоже на поняття програмування сокетів та клієнт-серверного програмування, включає в себе написання комп'ютерних програм, які взаємодіють з іншими програмами через комп'ютерну мережу(див. рис. 2.1).

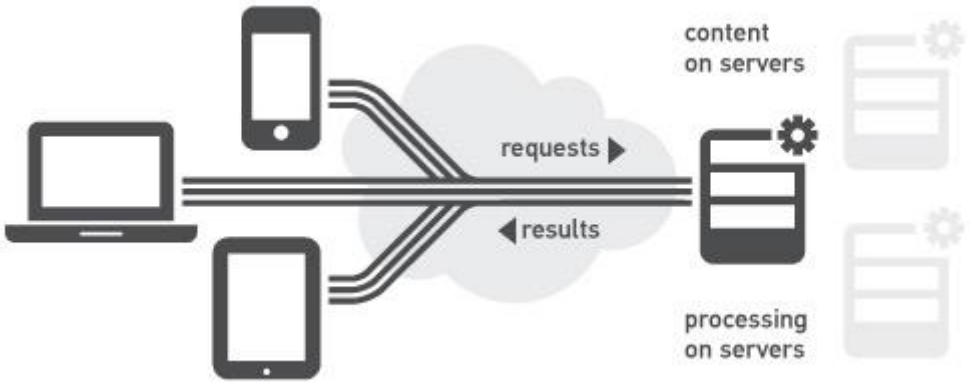

Рисунок 2.1 – Принцип мережевого програмування

Програма або процес, що ініціює встановлення зв’язку, називається клієнтським процесом, а програма, яка очікує ініціювання зв'язку, називається процесом сервера. Клієнтські та серверні процеси разом утворюють розподілену систему. Зв'язок між клієнтськими та серверними процесами можуть бути засновані наосновіз’єднань. Яскравим прикладом є протокол TCP, що встановлює віртуальне з'єднання або сеанс.Також є можливість є можливість зв’язку без з’єднань − на основі UDP-дейтаграм.

Програма, яка може функціонувати, і як клієнт, і як сервер базується на одноранговій комунікації.

Сокети зазвичай реалізуються за допомогою бібліотеки інтерфейсу програмування (API), такими як сокетиБерклі, представленими в 1983 році. Більшість реалізацій засновані на сокетахБерклі, наприклад, Winsock, представлений у 1991 році (див. рис. 2.2).

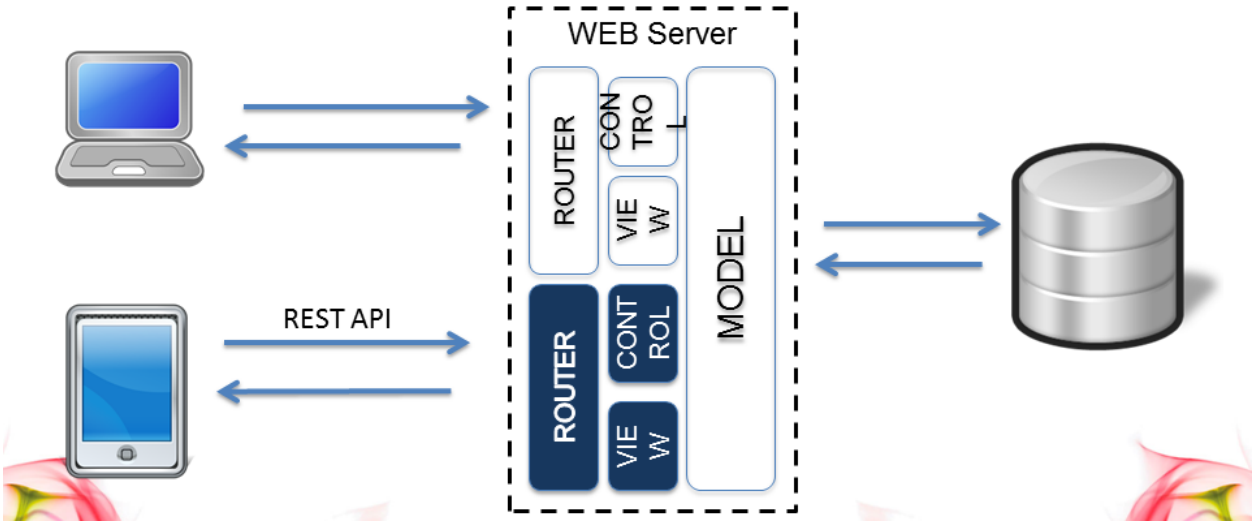

Рисунок 2.2 – Схема роботи API

Наведені нижче приклади функцій або методів, як правило, реалізуються бібліотекою API:

- `socket()` – Створює новий сокет певного типу, ідентифікується за допомогою цілого числа, після чого він виділяє системні ресурси;
- `bind()`– використовується на стороні сервера, асоціюєсокет з адресною структурою сокетів, тобто з певним номером місцевого порту та IP-адресою;
- `listen()` – використовується на стороні сервера, передає TCP-сокет до режиму прослуховування;
- `connect()` – використовується на стороні клієнта, пов'язує номер незайнятого локального порту до сокета;

- `accept()` – використовується на стороні сервера, ця функція приймає отриману спробу створення нового TCP-з’єднання з віддаленого клієнта та створює новий сокет, пов'язаний з парою сокетнихадрес цього з’єднання;
- `send()` `irecv()` чи`write()` `iread()` чи`recvfrom()` `isendto()` – використовується для надсилання та отримання даних до/з видаленого сокету;
- `close()` – викликає визволення системних ресурсів, виділених сокету.

Головною перевагою при виборі мережевого програмування для написанні програмного продукту є його властивість працювати на будь-якій платформі з’єднаній по мережі інтернет[7][1)].

2.2 Вибір мови програмування

При виборі мови програмування були розглянутіGo та JavaScript з підключенням програмної платформи Node.js.

2.2.1 Огляд Node.js

Node.JS − це програмна платформа, що базується на двигуні V8,який перекладаєJavaScript у машиний код та перетворює JavaScript з вузкоспеціалізованоїмови в мову загального назначення. Node.js додає JavaScript можливість взаємодіяти з пристроями введення та виводу через API, написаний у C ++, підключити інші зовнішні бібліотеки, написані на різних мовах, забехпучуючивизови до них з коду JavaScript. Node.js застосовується головним чином на сервері, виконуючи роль веб-сервера, але є можливість і розробки десктопних додатків (за допомогою NW.JS, Appjsабо

[1)][7]Программированиесетевых задач – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Программирование_сетевых_задач (дата звернення 07.08.2021).

Electron для Linux, Windows та MacOS) та навіть програмувати мікроконтролери (наприклад, tessel, low.js та espruino). Node.js базується на подіє-орієнтованому та асинхронному (або реактивному) програмуванні з неблокованимвведеннямтавиведення.

Однією з головних привабливих функцій Node.JS є швидкість. Код JavaScript, який виконується в середовищі Node.js, може бути вдвічі швидше, ніж код, написаний на компілійованих мовах таких, як C або Javata набагато швидшеінтерпритованих мов, як Python або Ruby. Причиною цього є неблокована архітектура платформи (див. рис. 2.3).

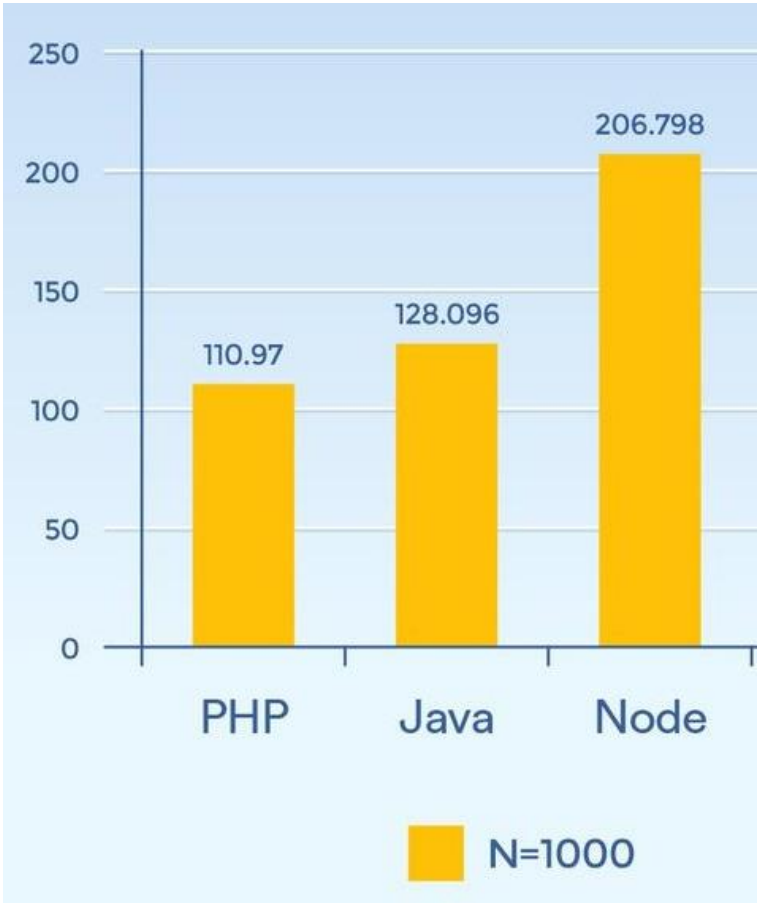

Рисунок 2.3 – Графік продуктивності Node.js у порівнянні з Java та PHP

Браузер та сервер використовує однакові концепції мови JavaScript. Крім того, в Node.JS можнаоперативно перейти до використання нових стандартів ECMASSTRE, оскільки вони реалізуються на платформі. Для

цього не потрібно чекати, поки користувачі виконають оновлення браузера, так як Node.js − це серверне середовище, яке розробник повністю контролює. Як результат нові можливості мови стають доступними при встановленні підтримуваємої версії Node.js.

В традиційних мовах програмування (C, Java, Python, PHP), всі інструкції за замовчуванням є блокованми. В результаті, якщо в такому середовищі зробити мережевий запит,для завантаження JSON-коду, виконання потоку, з якого здійснюється запит, буде призупинено, доки не будуть завершиними отримання та обработка запиту.

JavaScript значно спрощує написання асинхронного та неблокованого коду за допомогою єдиногопотоку, функції зворотного виклику (коллбеків) та підхода до розробки, основаної на подіях. Кожного разу, коли потрібно виконувати важку операцію, потрібнопередати відповідний механізм коллбек, який буде викликаний відразу після завершення цієї операції. Як результат, для того, щоб програма продовжувала працювати, не треба чекати результатів виконання таких операцій (див. рис. 2.4).

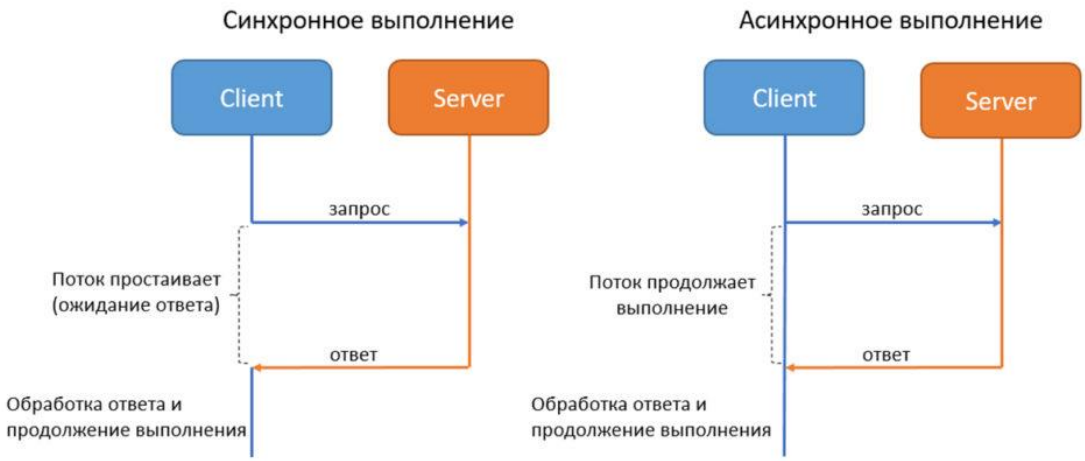

Рисунок 2.4 – Відмінність синхронного та асинхронного виконання операцій

Такий механізм виник у браузерах. Під час роботи з ниминемає можливості чекати закінчення виконання запиту AJAX, не маючи можливості відповісти на дії користувача, наприклад, натискання на кнопки. Для того, щоб користувачу було зручно працювати з веб-сторінками, завантаження даних з мережі та обробка натискання кнопки повинні відбуватися одночасно, в режимі реального часу.

Асинхронні механізми дозволяють єдиному серверу Node.js одночасно обробляти тисячі підключень без завантаження програміста до завдань керування потоками та організацією виконання паралельного коду. Такі речі часто є джерелами помилок.

Node.js забезпечує неблоковані базові механізми введення та виводудля розробника, та всі необхідні бібліотеки, написані за допомогою неблокованих парадигм. Це робить блокувануповедінку коду скоріше винятком, ніж нормою.

Коли Node.JS потрібно виконати операцію вводу/виводу, наприклад, завантаження даних з мережі, доступ до бази даних або до файлової системи, замість того, щобзаблокувати очікуванням результатів такої операції головний поток, Node.js ініціює своє виконання та продовжує займатися іншими завданнями, поки результати виконання цієї операції не будуть отримані.

Через простоту та зручність роботи з менеджером пакетів для Node.JS, який називається npm, екосистемаNode.js постійно розвивається та покращується[8][1)].Станом на травень 2021 року, у реєстрі npm нараховується більше500000 відкритих пакетів, які може вільно використовувати будь-який розробник (див. рис. 2.5).

[1)][8] Руководство по Node.js – Хабр. URL: https://habr.com/ru/company/ruvds/blog/422893/ (дата звернення 09.08.2021).

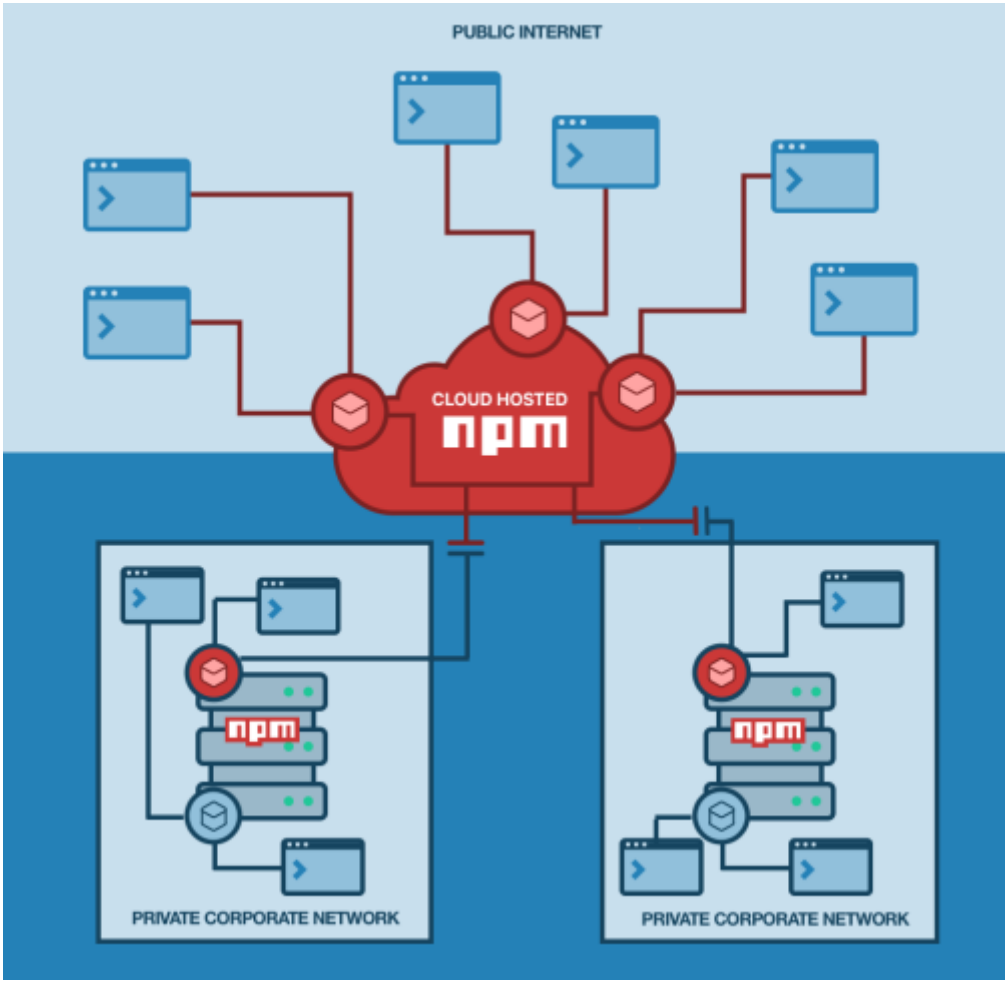

Рисунок 2.5 – Архітектура npm

До переваг Node.js можна віднести наступні особливості програмної платформи:

– наявність асинхронних механізмів;
– масштабування та кросплатформенність;
– наявність великої кількості відкритих бібліотек.

Недоліками в Node.js можна назвати:

– при розробці великих систем використання Node.js ускладнюється процес написання коду через те, що Node.js є інтерпритованою, а не компільвано мовою;
– менша ефективність та швидкість, у порівнянні з мовою Go.

2.2.2 ОглядGo

Go − це компілійована статично набрана мова програмування від Google. Мова Go призначена для створення різних видів програм, але, перш за все, це веб-служби та клієнт-серверні програми. Хоча мова також має можливість працювати з графікою, низькорівнеі можливості тощо.

РоботапостворенніюGo розпочалася у 2007 році у відділах Google. Одним з авторів є КенТомпсон, який є одним з авторів мови C (разом з Денисом Річ). 10 листопада 2009 року був оголошений анонс мови, а в березні 2012 року версія 1.0 була випущена. У той же час мова продовжує розвиватися.

Мова Gorозвивається, як опенсорс, тобто, вона представляє проект з відкритим вихідним кодом і всі його коди та компілятор можна знайти та використовувати безкоштовно.

GO є кросплатформеною мовою, це дозволяє створювати програми для різних операційних систем − Windows, Mac OS, Linux, FreeBSD. Код володіє переносимістю: програми, написані для однієї з цих операційних систем, можуть бути легко передані в іншу ОС.

Мова GO була розроблена в якості мови програмування для створення високоефективних програм, що працюють на сучасних розподілених системах і багатоядерних процесорах. Її можна розглядати як спробу створити заміну мов C і C++ з урахуванням змінення комп'ютерних технологій і накопиченим досвідом розробки великих систем. За словами Роба Пайка, «GO був розроблений для вирішення реальних проблем, пов'язаних з розробкою програмного забезпечення в Google.» В якості основних таких проблем, він називає:

- повільна збірка програм;
- неконтрольовані залежності;
- використання різними програмістами різних підмножин мови;

- труднощі з розумінням програм, викликаними поганочитаємим кодом, поганою документацією тощо;
- висока вартість оновлень;
- несинхронні поновлення при дублюванні коду;
- складність розробки набору інструментарію;
- проблеми взаємодії з іншими мовами.

Основними вимогами до мови стали: ортогональність,простота і регулярна граматика, проста робота з типами,відсутність неявних перетворень, прибирання сміття, вбудованізасобирозпаралелювання, підтримка строк, асоціативних масивів і каналів зв'язку та ефективна система пакетів з чітким зазначенням залежностей, які забезпечують швидку збірку.

Будь-яка програма для Go включає один або декілька пакетів. Пакет, до якого відноситься файл вихідного коду, задається описом `package` на початку файлу. Назви пакетів мають однакові обмеження, що і ідентифікатори, але можуть містити лише літери нижнього регістру. Система пакетів Go-середовища має структуру дерева, аналогічну до дерева каталогу. Будь-які глобальні об'єкти (змінні, типи, інтерфейси, функції, методи, структурні елементи та інтерфейси) доступні без обмежень у пакеті, в якому вони оголошені. Глобальні об'єкти, чиї імена починаються на заглавну літеру, експортуються. [9][1)].

В даний час Go широко використовується в різних галузях. Зокрема, серед відомих проектів, які використовуютьGo, можна знайти наступні: Google, Dropbox, Netflix, куберне, Docker, Twitch, Uber, CloudFlare та ряд інших.

До переваг Go можна віднести:

- швидкість та ефективність;

[1)][9] Go – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Go (дата звернення 18.08.2021).

- успадкування переваг мов низького та високого рівня та компіляція безпосередньо в машиний код;
- можливість виконання декількох завдань паралельно;
- завдяки статичній типізації, за допомогою Go можна писати більш чистий код.

Під час аналізу мов програмування, було прйнято рішення написання програмного додатку на мові програмування Go.

2.3 Програмне забезпечення Docker

Docker це інструмент, який дозволяє розробникам, системним адміністраторам та іншим фахівцям деплоїти їх програми в пісочниці (які називаються контейнерами), для запуску на цільовій операційній системі, наприклад, Linux. Ключова перевага Docker в тому, що він дозволяє користувачам упакувати додаток з усіма його залежностями в стандартизований модуль для розробки. На відміну від віртуальних машин, контейнери не створюють такого додаткового навантаження, тому з ними можна використовувати систему та ресурси більш ефективно.

Стандарт в індустрії на сьогоднішній день − це використання віртуальних машин для запуску додатків. Віртуальні машини запускають програми усередині гостьової операційної системи, яка працює на віртуальному залізі основної операційної системи сервера.

Віртуальні машини відмінно підходять для повної ізоляції процесу для застосування: майже ніякі проблеми основної операційної системи не можуть вплинути на програму гостьової ОС, і навпаки. Але за таку ізоляцію доводиться платити. Існує значне обчислювальне навантаження, необхідне для віртуалізації заліза гостьової ОС.

Контейнери використовують інший підхід: вони надають схожий з віртуальними машинами рівень ізоляції, але завдяки правильному залученню

низькорівневих механізмів основної операційної системи роблять це в рази меншим навантаженням (див. рис. 2.6).

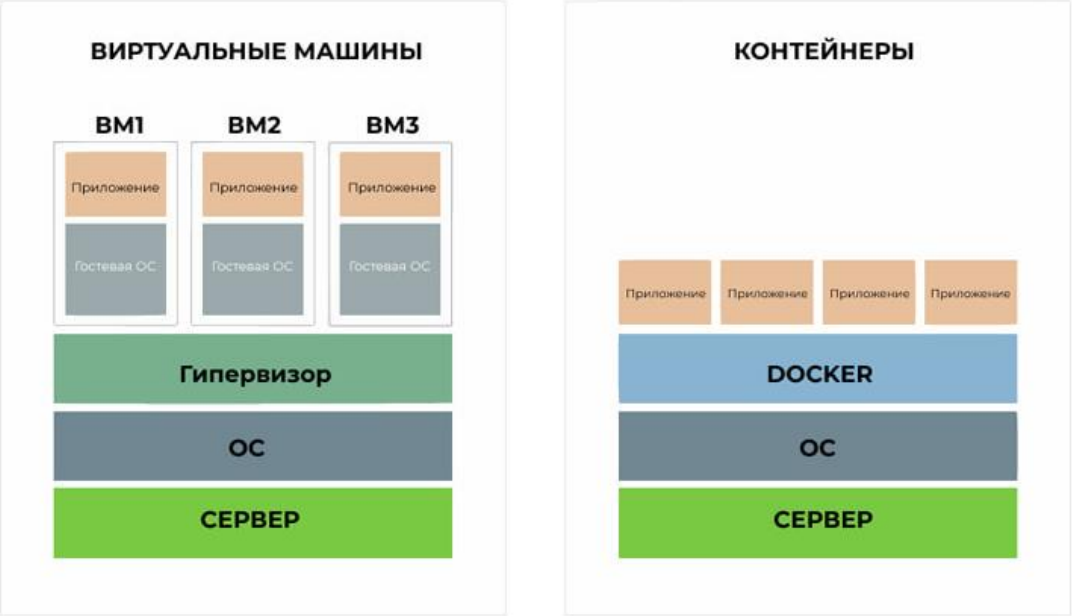

Рисунок 2.6 – Відмінності віртуальних машин та контейнерів

Не дивлячись на те, що контейнери самі по собі — не нова технологія, до появи Docker вони не були настільки поширеними і популярними. Dockerзмінив ситуацію, надавши стандартний API, який спростив створення та використання контейнерів, і дозволив спільноті разом працювати над бібліотеками по роботі з контейнерами.

Робота Docker заснована на принципах клієнт-серверної архітектури, яка базується на взаємодії клієнта з веб-сервером (хостом). Перший надсилає запити отримання даних, а другий їх надає. На рис. 2.7 зображена схема роботи Docker[10][1)].

[1)][10] Полноепрактическоеруководство по Docker: с нуля до кластера на AWSURL: https://habr.com/ru/post/310460/ (дата звернення 23.08.2021).

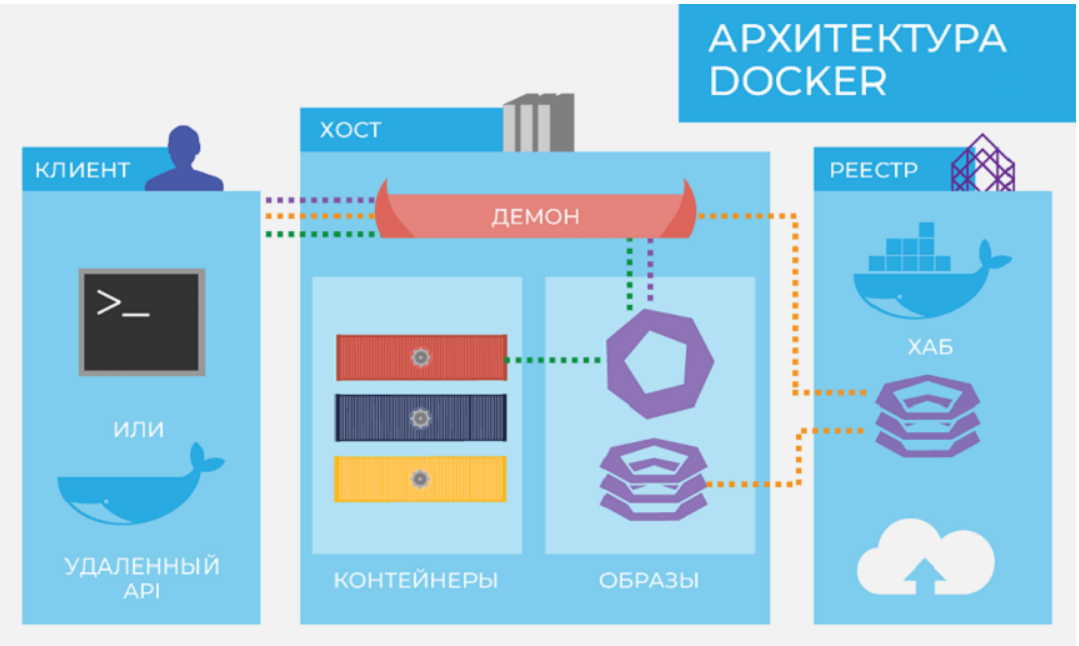

Рисунок 2.7 – Архітектура Docker

Користувач віддає команду за допомогою клієнтського інтерфейсу Docker-демону, розгорнутому на Docker-хості. Наприклад, завантажити готовий образ із реєстру (сховища Docker-образів) за допомогою команди `dockerpull`. Взаємодія між клієнтом та демоном забезпечує REST API. Демон може використовувати публічний (DockerHub) або приватний реєстр.

З команди, заданої клієнтом, демон виконує різні операції з образами з урахуванням інструкцій, прописаних у файлі Dockerfile. Наприклад, здійснює їх автоматичне складання за допомогою команди `dockerbuild`.

Робота образу у контейнері. Наприклад, запуск docker-image за допомогою команди `dockerrun` або видалення контейнера через команду `dockerkill`.

Для економії простору зберігання проект використовуєфайлову систему Aufs за допомогоютехнології каскадно-об'єднаногомонтування: контейнеривикористовують образ базовоїопераційноїсистеми, а змінизаписуються в окрему область. Такожпідтримуєтьсярозміщенняконтейнерів у файловійсистеміBtrfsізвключеним режимом копіюванняпід час запису.

До складупрограмнихзасобів входить демон - сервер контейнерів (запускається командою `docker -d`), клієнтськізасоби, щодозволяють з інтерфейсу командного рядка керувати образами та контейнерами, а такожAPI, щодозволяє в стиліRESTкерувати контейнерами програмно.

Демон забезпечуєповнуізоляціюконтейнерів, щозапускаються на вузлі, на рівніфайловоїсистеми (у кожного контейнера власнакореневафайлова система), на рівніпроцесів (процесимають доступ тільки до власноїфайловоїсистеми контейнера, а ресурсирозділенізасобамиlibcontainer), на рівнімережі (кожен контейнер має доступ тільки до прив'язаного до нього мережного простору імен та відповіднихвіртуальнихмережевихінтерфейсів).

Набірклієнтськихзасобівдозволяєзапускатипроцеси в нових контейнерах (dockerrun), зупиняти та запускатиконтейнери (dockerstop та dockerstart), зупиняти та відновлюватипроцеси в контейнерах (`dockerpause`та `dockerunpause`). Серія команд дозволяєздійснюватимоніторингзапущенихпроцесів (dockerps за аналогією з ps в Unix-системах, dockertop за аналогією з top та інші). Новіобразиможнастворюватизіспеціального сценарного файлу (dockerbuild, файл сценаріюназиваєтьсяDockerfile), можназаписативсізміни, зроблені в контейнері, в новий образ (dockercommit). Усікомандиможутьпрацювати як з docker-демоном локальноїсистеми, так і з будь-яким сервером Docker, доступним через мережу. Крім того, в інтерфейсі командного рядка вбудованіможливості по взаємодії з публічнимрепозиторіємDockerHub, в якомурозміщенізаздалегідьзібраніобразидодатків, наприклад, команда dockersearchдозволяєздійснитипошукобразівсередрозміщених у ньому, образиможназавантажувати в локальну систему (dockerpull), можнатакожнадіслати локально зібраніобрази в DockerHub (`dockerpush`).

2.4 Створення запитів у Postman

Основне призначення програми – створення колекцій із запитами до API. Будь-який розробник або тестувальник, відкривши колекцію, зможе легко розібратися в роботі вашого сервісу. До того ж, Postman дозволяє проектувати дизайн API і створювати на його основі Mock-сервер. Завдяки Postman розробникам не потрібно витрачати час на створення "заглушок". Реалізацію сервера та клієнта можна запустити одночасно. Тестувальники можуть писати тести та проводити автоматизоване тестування прямо з Postman. Для адміністраторів автори передбачили можливість створення колекцій для моніторингу сервісів.

Головні поняття, якими оперує Postman − це Collection (колекція) на верхньому рівні, та Request (запит) на нижньому. Вся робота починається з колекції та зводиться до опису API за допомогою запитів (див. рис. 2.8).

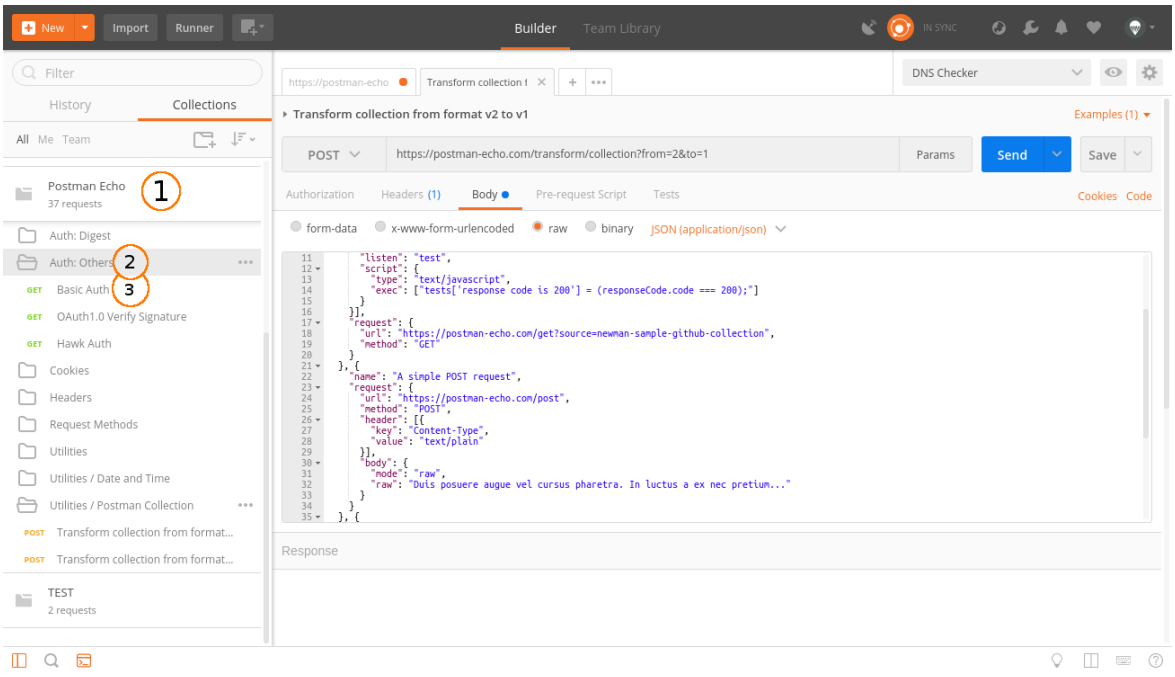

Рисунок 2.8 – Інструменти Postman − колекція (1), папка (2), запит (3)

Колекція – відправна точка для нового API. Можна розглядати колекцію як файл проекту. Колекція поєднує у собі всі пов'язані запити.

Зазвичай API описується в одній колекції, але немає жодних обмежень зробити по-іншому. Колекція може мати свої скрипти та змінні.

Папка — використовується для об'єднання запитів до групи всередині колекції. Наприклад, можна створити папку для першої версії API − "v1", а всередині згрупувати запити за змістом виконуваних дій − "Order&Checkout", "Userprofile" тощо. Папка, як і колекція, може мати свої скрипти, але не змінні.

Запит − основна складова колекції. Запит створюється у конструкторі. Конструктор запитів – це головний простір, з яким доведеться працювати. Postman вміє виконувати запити за допомогою всіх стандартних методів HTTP. Можна змінити або додати необхідні вам заголовки, cookie, і тіло запиту. Запит має свої скрипти. Завдяки вкладці "Pre-requestScript" та "Tests" серед параметрів запиту, можна додати скрипти перед виконанням запиту та після. Саме ці дві можливості роблять Postman потужним інструментом, що допомагає при розробці та тестуванні.

"PostmanSandbox" це середовище виконання JavaScript доступне при написанні "Pre-requestScript" та "Tests" скриптів. "Pre-requestScript" використовується для проведення необхідних операцій перед запитом, наприклад, можна зробити запит до іншої системи та використовувати результат виконання в основному запиті. "Tests" використовується для написання тестів, перевірки результатів і при необхідності їх збереження в змінні (див. рис. 2.9).

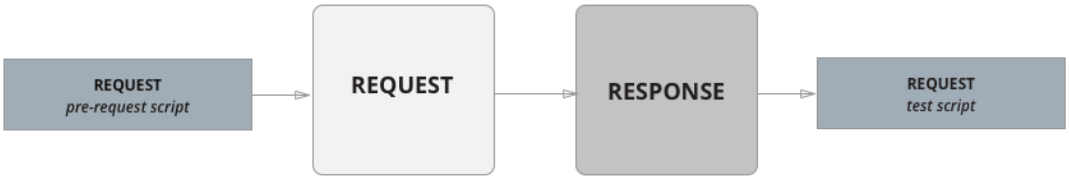

Рисунок 2.9 – Послідовність виконання запитів

Postman має кілька просторів та областей видимості для змінних:

- глобальні змінні;
- зінні колекції;
- змінні оточення;
- локальні змінні;
- змінні рівні даних.

Глобальні змінні та змінні оточення можна створити, якщо натиснути на шестерню у верхньому правому куті програми. Вони існують окремо від колекцій. Змінні рівні колекції створюються безпосередньо при редагуванні параметрів колекції, а локальні змінні з скриптів, що виконуються. Також є змінні рівня даних, але вони доступні тільки з Runner. На рис 2.10 зображено пріорітет простору змінних.

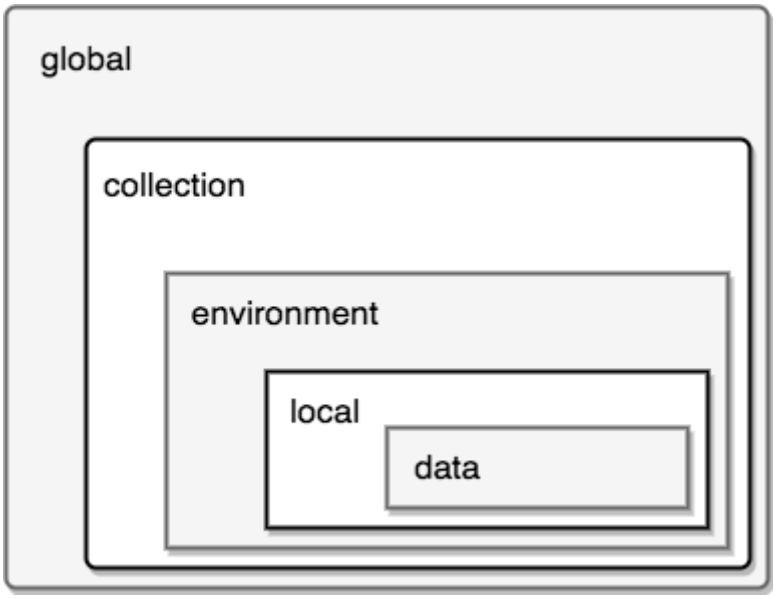

Рисунок 2.10 – Пріорітет простору змінних

Особливістю змінних у Postman є те, що можна вставляти їх у конструкторі запиту, URL, POST параметри, Cookie тощо, використовуючи фігурні дужки як плейсхолдер для підстановки.

Зі скриптів змінні теж доступні, але отримати їх допоможе виклик стандартного методу вбудованої бібліотеки pm

Для тестування та виконання всіх запитів з колекції чи папки на вибір призначений CollectionRunner. При запуску можна вказати кількість ітерацій, скільки разів буде запущено папку або колекцію, оточення, а також додатковий файл зі змінними. Запити виконуються послідовно, згідно з розташуванням у колекції та папках. Порядок виконання можна змінити за допомогою вбудованої команди: `postman.setNextRequest(' ');`.

Після виконання всіх запитів формується звіт, який покаже кількість успішних та провальних перевірок зі скриптівTests (див. рис. 2.11).

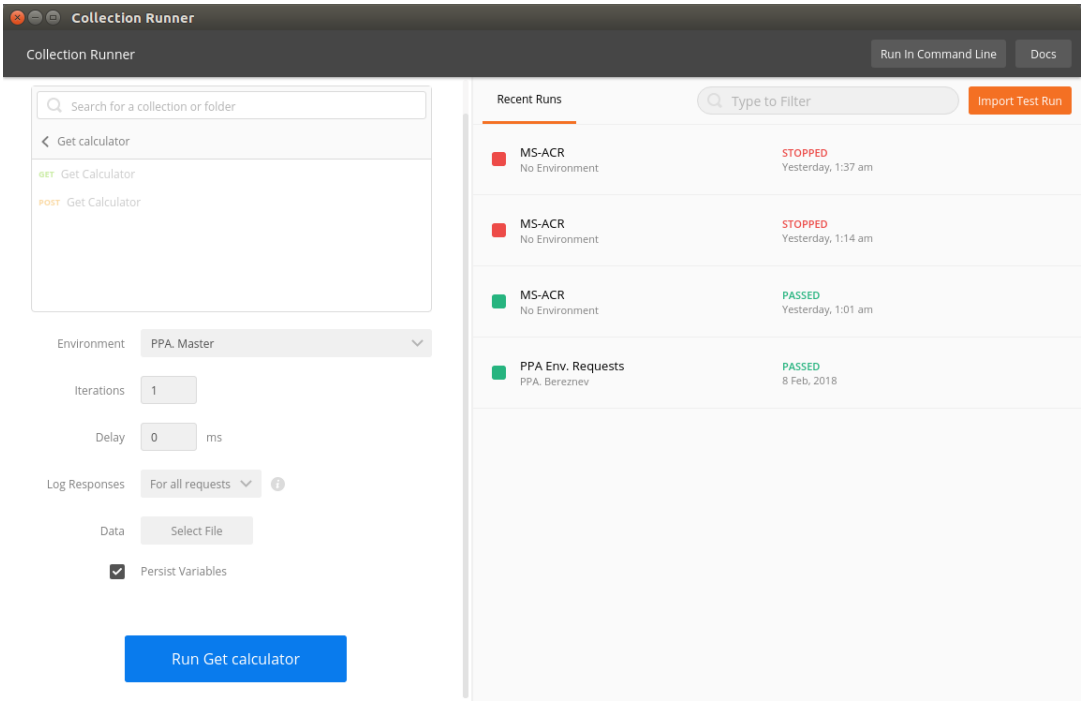

Рисунок 2.11 – Звіт перевірок

Для налагодження скриптів та перегляду додаткової інформації за запитами використовується консоль. Вона працює як під час запуску одного запиту, так і під час запуску пакета запитів через Runner. [11][1)].

[1)][11] Введение в Postman– Хабр. URL: https://habr.com/ru/company/kolesa/blog/351250/ (дата звернення 26.08.2021).

2.5 База даних MongoDB

Для багатьох звичними є бази даних реляційного типу — MySQL, MS SQL, Oracle та інші. У таких базах дані зберігаються у таблицях, а для роботи з ними потрібно писати запити. Але є база даних з іншою архітектурою – база MongoDB[12][1)].

MongoDB – це документоорієнтована база даних типу NoSQL. На відміну від реляційних БД, NoSQL для зберігання даних використовують не таблиці з рядками та стовпцями, а колекції та JSON-подібні документи. Документи складаються з пар «ключ-значення». Пара «ключ-значення» є ім'ям поля (ключ документа) та його значення. Колекції складаються з груп документів (див. рис. 2.12).

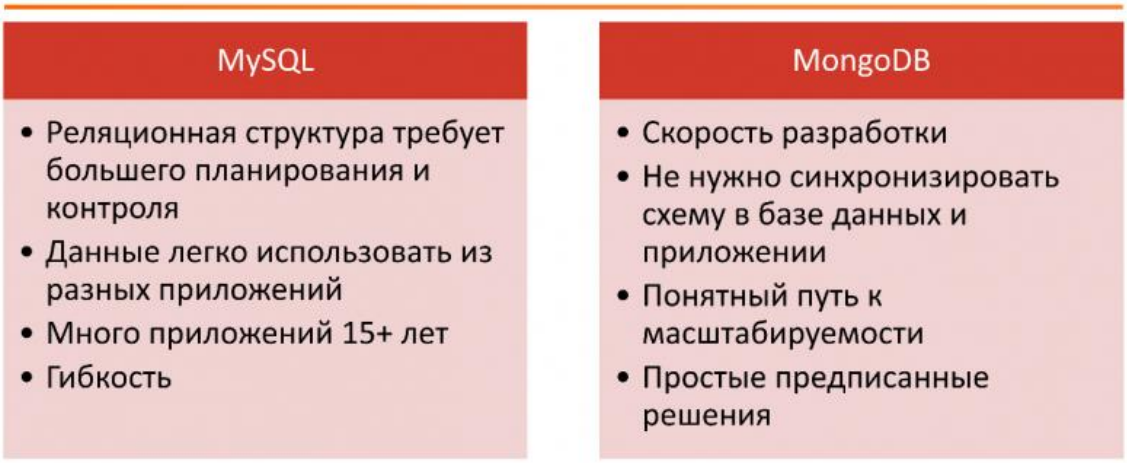

Рисунок 2.12 – ПорывняннняMongoDB з MySQL

У MongoDB можна працювати з різними мовами програмування: PHP, Perl, C/C++, Go, Node.JS. Для MongoDB документація українською мовою відсутня, але на офіційному сайті є методичка баз даних у MongoDB.

[1)][12] Reg.ru: Руководство по MongoDB. URL: https://help.reg.ru/hc/ru/articles/4408054704785 (дата звернення 03.09.2021).

До переваг MongoDB можна віднести:

- гнучкість — MongoDB зберігає дані в документах JSON, а не в таблицях. Це дозволяє зберігати інформацію зі складною структурою. При цьому зміст та розмір документів може бути різним, і не потрібно створювати певну схему бази даних;
- кросплатформенність − MongoDB можна використовувати на операційних системах Windows, Linux (Ubuntu, Debian, CentOS), MacOS;
- динамічні запити до документів;
- реплікація – MongoDB може працювати на кількох серверах;
- проста масштабованість.

На рис. 2.13 зображений приклад документу MongoDB.

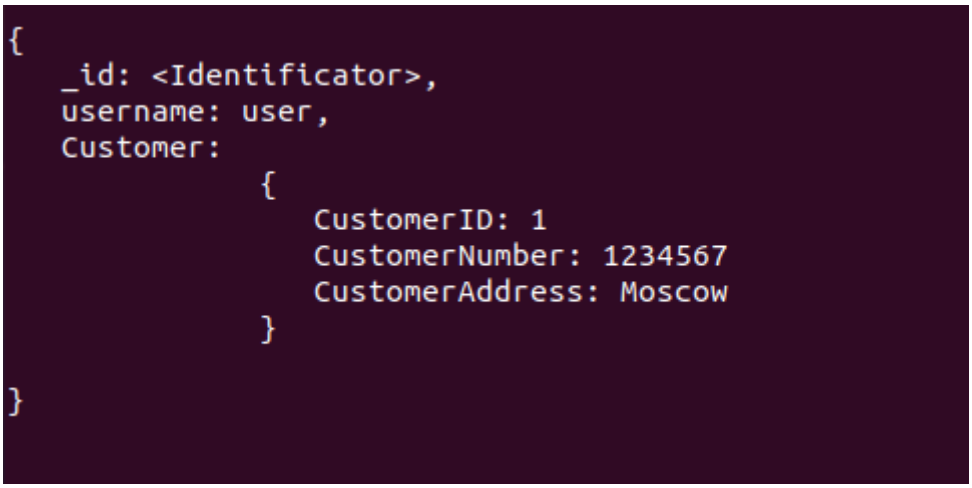

Рисунок 2.13 – Приклад документу MongoDB

Ключові компоненти архітектури MongoDB:

- _id – унікальний ідентифікатор документа MongoDB. Якщо додати новий документ без поля _id, ідентифікатор буде створено автоматично;
- документ — запис, який зберігається у колекції. Це еквівалент рядка в реляційних СУБД. Складається з пар «ключ-значення»;

- колекція − це група документів MongoDB, еквівалент таблиці в реляційних СУБД;
- база даних – це контейнер із колекціями. Кожна база даних має свій власний набір файлів у файловій системі. Сервер MongoDB може зберігати декілька баз даних.

2.6 Вибір інтегрованого середовища розробки

Інтегрована середа розробки (IDE), в якій ви пишете свій код, визначає не тільки його якість, але навіть структуру і стиль написання.

Для створення програмного продукту на мові програмування Gонайбільшj. популярністю користуються такі середовища розробки:

- «GoLand»
- «VisualStudioCode»

2.7.1 ОглядGoLand

GoLand — це кросплатформне інтегроване середовище розробки для розробників Go. GoLand включає в себе такі функції, як контекстно-залежне завершення та рефакторинг коду, налагодження, профілювання, навігація за типами й оголошеннями, а також аналіз помилок. На додаток до інструментів для основної розробки Go, підтримує JavaScript, TypeScript, Node.js, SQL, бази даних, Docker, Kubernetes і Terraform. Також можна розширити поточну функціональність, встановивши додаткові плагіни зі сховища плагінів[13][1)].

GoLand надає низку можливостей для роботи з мовою Go, а саме:

- інтерфейс з вкладками;
- розумне авто-завершення коду;
- інспекції та швидкі виправления;

[1)][13] JetBrains: Introduction. URL: https://www.jetbrains.com/help/go/meet-the-product.html 15.09.2021).

- рефакторинг;
- швидка навігація;
- швидкі спливаючі вікна для документації, визначення, використання, структури тощо;
- генерація коду (наприклад, інтерфейс реалізації);
- виявлення рекурсивних викликів;
- відображення типу будь-якого виразу;
- функція виділення точок виходу;
- підказки параметрів.

ВідладчикGogland побудований поверх Delve. Він використовує знайомі конфігурації відладкиJetBrains для управління всім, що стосується запуску відладки, таким як виконуваний двоїчний файл, змінні засоби, робочий каталог і аргументи командної строки.

Можна встановити контрольні точки, умовні контрольні точки та години. Коли досягається точка зупинки, є можливість спостерігати трассировку стека та локальні змінні в кожному кадрі. На жаль, глобальні змінні не відображаються і часто необхідні в Go.

Тестовий прогін дозволяє запускати, зупиняти та перезапускати тести. Можна використовувати його для запуску та відладки модульних тестів, а також інтеграційних тестів із використанням пакета тестування Go.

Go постачається з різними інструментами, а Goland інтегрує їх прямо в IDE. Ви можете викликати будь-який інструмент з меню «Code | GoTools». Вбудоване форматування коду Goland використовує стандартний інструмент «gofmt».

Goland наслідує систему управління вихідним кодом інших продуктів JetBrains. У тому числі git, яка дозволяє керувати декількома проектами, відображати гілки, множинні набори змін та багато іншого. Інші системи контролю версії, такі як Mercurial і SVN, також підтримуються за допомогою додаткових плагінів.

Goland надає вбудований термінал на випадок, якщо потрібно буде виповнити деякі команди. Це надає можливість переглядати або копіювати з/в панелі редактора, не переключаючи вікна повністю.

Gogland має багато вбудованих можливостей, але він повністю розширюється, багато його вбудованих функцій є попередньо завантаженими плагінами. Можна встановити багато корисних плагінів, а також розробити свої власні плагіни.

Пробна версія доступна для кожного користувач на період 30 днів, після закінчення пробного періоду середа розробки стає платною.

2.7.2 ОглядVisualStudioCode

VisualStudioCode − це редактор коду, що розширюється, з відкритим вихідним кодом, розроблений Microsoft (з вкладом спільноти). Він заснований на Electron, який у свою чергу базується на Chromium. VisualStudioCode підтримує велику кількість мов програмування та орієнтований на веб-розробку. Має сильну підтримку розробки Go, включаючи інтеграцію всіх інструментів Go та Delveвідладчика через виділене розширення. Для початку роботи с Go потрібно буде встановити деякі пакети та інструменти[14][1)].

VisualStudioCode також пропонує інтеграцію з git, ієрархічний оглядач папок/файлів та інтерфейс з вкладками.

Підтримка IntelliSense (автозаповнення, відображення типів параметрів та документація) забезпечує дуже приємний досвід редагування. Вбудований налагоджувач (використовує Delve) добре допомогає при написанні коду.

VisualStudioCode дуже чуйний і швидкий, до його списку можливостей відносять:

- списки завершення (використовуючи gocode);

[1)][14]ВикипедияVisualStudioCodeURL: https://ru.wikipedia.org/wiki/Visual_Studio_Code (дата звернення 12.09.2021).

- довідка за підписом (за допомогою gogetdoc або godef + godoc);
- сніппети;
- коротка інформація (за допомогою gogetdoc або godef+godoc);
- визначення Goto (використовується gogetdoc або godef + godoc);
- пошук посилань (використовуючи guru);
- посилання CodeLens;
- схема файлу (використовуючи go-outline);
- пошук символів робочого простору (з використанням go-символів);
- перейменувати (використовуючи gorename. Примітка. Щоб скасувати перейменування для роботи у Windows, необхідно мати інструмент diff);
- build-on-save (використовуючи gobuild та gotest);
- lint-on-save (з використанням golint або gometalinter);
- формат (використовуючи goreturns або goimports або gofmt);
- генерувати скелет юніт-тестів (використовуючи gotests);
- додати імпорт (використовуючи gopkgs);
- додати/видалити теги у структурних полях (використовуючи gomodifytags);
- семантичні/синтаксичні повідомлення про помилки під час введення (використовуючи gotype-live);
- запускати тести під курсором, у поточному файлі, у поточному пакеті, у всьому робочому просторі (використовуючи gotest).

Також VisualStudioCode підтримує редагування та виконання файлів типу "Блокнот Jupyter" безпосередньо "з коробки" без встановлення зовнішнього модуля в режимі візуального редагування та в режимі редагування вихідного коду.

За допомогою вбудованого в продукт інтерфейсу користувача можна завантажити і встановити кілька тисяч розширень тільки в категорії «programminglanguages».

Також розширення дозволяють отримати більш зручний доступ до програм, таких як Docker, Git та інші. У розширеннях можна знайти лінтери коду, теми для редактора та підтримку синтаксису окремих мов.

Після аналізу, в якості середи розробки була вибрана IDE VisualStudioCode. Причиною вибору є надання більшої кількості можливостей та можливість безплатного користування.

3 ПРОЕКТНА ЧАСТИНА

3.1 Onion-архітектура

Термін "OnionArchitecture" (з англ. «цибульна архітектура») був запропонований Джеффрі Палермо (JeffreyPalermo) ще в 2008 році. Через роки ця концепція стала досить популярною і є однією з найбільш застосовуваних типів архітектури при побудові програми ASP.NET.

Onion-архітектура є поділом програми на рівні. Причому є один незалежний рівень, що знаходиться у центрі архітектури. Від цього рівня залежить другий рівень, від другого – третій тощо. Тобто виходить, що довкола першого незалежного рівня нашаровується другий-залежний. Навколо другого нашаровується третій, який може залежати і від першого. Образно це може бути виражене у вигляді цибулі, в якій також є серцевина, навколо якої нашаровуються всі інші шари, аж до лушпиння (див. рис 3.1).

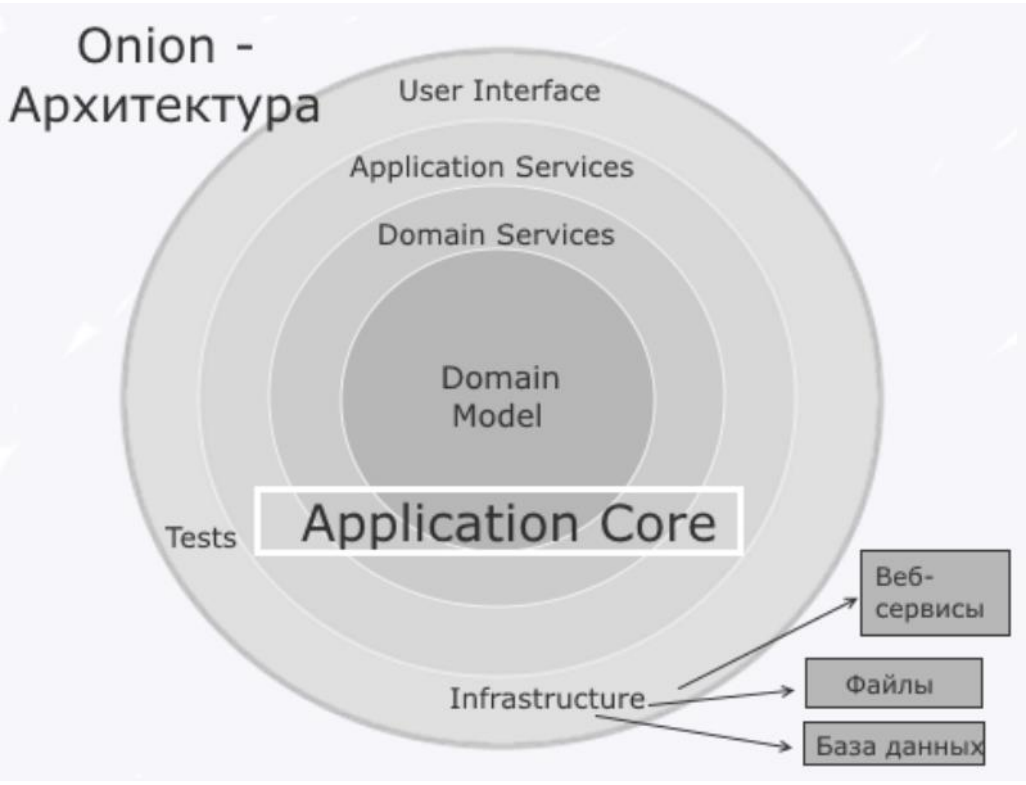

Рисунок 3.1 – Onion-архітектура

Кількість рівнів може відрізнятися, але в центрі завжди знаходиться модель домену (DomainModel), тобто класи моделей, які використовуються в додатку і об'єкти яких зберігаються в базі даних.

Перший рівень навколо моделі домену утворюють інтерфейси, що управляють роботою з моделлю домену. Зазвичай це інтерфейси репозиторіїв, якими ми взаємодіємо з базою даних.

Зовнішній рівень представляє такі компоненти, які дуже часто змінюються. Зазвичай зовнішній рівень утворюють інтерфейс користувача, тести, якісь допоміжні класи інфраструктури програми. До цього рівня також відносяться конкретні реалізації інтерфейсів, оголошених на нижчих рівнях. Наприклад, реалізація інтерфейсу репозиторію, який оголошено лише на рівні DomainServices. Взагалі всі внутрішні рівні, які можна об'єднати в ApplicationCore, визначають тільки інтерфейси, а конкретна реалізація цих інтерфейсів розташовується на зовнішньому рівні.

Також варто відзначити, що всі зовнішні сховища, як бази даних, файли, зовнішні веб-сервіси, від яких можна отримувати дані − все це є зовнішнім по відношенню до архітектури.

При створенні архітектури програми треба розуміти, що реальна кількість рівнів тут є досить умовною. Залежно від масштабу завдань рівнів може бути більше, і менше. Однак важливо розуміти сам принцип, що в центрі у нас моделі домену, а решта залежить від них. Кожен зовнішній рівень може залежати від внутрішнього, але не навпаки[15][1)].

Прикладом внутрішнього рівня є інтерфейс, відповідаючий за створення та отримання даних модулей показу даних.

Нступна функція демонструє створення абстрактної функції завдання параметрів.

[1)][15] Metanit: Архитектура приложений Onion-архитектура. URL: https://metanit.com/sharp/mvc5/23.1.php (дата звернення 30.09.2021).

```
func (r ModulesStateRepository) GDGTSetModulesState(ctxcontext.Context, boxIdstring, ms *models.ModulesState) error {
     objID, _ := primitive.ObjectIDFromHex(boxId)
     varbox = new(Box)
     opts := options.FindOne().SetSort(bson.D{{"model", 1}})
     err := r.boxesDb.FindOne(ctx, bson.D{
          {"_id", objID},
     }, opts).Decode(&box)
     iferr != nil { // ErrNoDocumentsmeansthatthefilterdidnotmatchanydocumentsinthecollection
          iferr == mongo.ErrNoDocuments {
               returnerr
          }
          log.Fatal(err)
     }
     fmt.Printf("founddocument %v", box)

     ms.BoxID = box.ID.Hex()

     jsonModulesState, err := json.Marshal(ms)

     modulesStateRes := r.modulesStateDb.Set(ctx, "modules-state", jsonModulesState, 0)
     println(modulesStateRes)
     returnnil
}
```

В цьому ж класі описана функція отримання даних з модулей.

```
func (r ModulesStateRepository) GetModulesState(ctxcontext.Context, boxIdstring) (*models.ModulesState, error) {
     objID, _ := primitive.ObjectIDFromHex(boxId)
     varbox = new(Box)
     opts := options.FindOne().SetSort(bson.D{{"model", 1}})
     err := r.boxesDb.FindOne(ctx, bson.D{
          {"_id", objID},
     }, opts).Decode(&box)
     iferr != nil { // ErrNoDocumentsmeansthatthefilterdidnotmatchanydocumentsinthecollection
          iferr == mongo.ErrNoDocuments {
               returnnil, err
          }
          log.Fatal(err)
     }
     fmt.Printf("founddocument %v", box)

     varms *models.ModulesState

     modulesStateRes, err := r.modulesStateDb.Get(ctx, "modules-state").Result()
     iferr != nil {
          returnnil, err
     }
```

```
	errRes := json.Unmarshal([]byte(modulesStateRes), &ms)
	iferrRes != nil {
		returnnil, errRes
	}

	returnms, err
}
```

3.2 Реєстрація та авторизація користувачів

Для реалізації авторизації користувачей використаємо JSON WebTokens (JWT).

JSON WebTokens – це відкритий стандарт RFC 7519 для створення токенів доступу. Використовується для передачі даних для автентифікації в клієнт-серверних додатках. У звичайних веб-програмах легко ідентифікувати користувачів за допомогою сесій, однак, коли API веб-програми взаємодіє, з клієнтом Android або IOS, сесії стають малопридатними для використання. За допомогою JWT ми можемо створити унікальний токен для кожного автентифікованого користувача. Цей токен буде включений до заголовка наступного запиту до API. Цей метод дозволяє ідентифікувати всіх користувачів, які виконують дзвінки API[16][1)].

JWT складається з трьох основних частин: заголовка (header), навантаження (payload) та підпису (signature). Заголовок та навантаження формуються окремо, а потім на їх основі обчислюється підпис.

Після першого логіну клієнту повертається згенерований сервером JWT. При кожному наступному запиті клієнт повинен передавати JWT встановленим API способом (наприклад, через заголовок або як параметр запиту). Сервер декодує header та payload та перевіряє зарезервовані поля. Якщо все гаразд, за вказаним у header алгоритмом складається підпис. Якщо отриманий підпис співпадає з переданим, користувача авторизують.

[1)][16]Создание и развёртывание REST API с помощьюGo– Tproger.URL: https://tproger.ru/translations/deploy-a-secure-golang-rest-api/ (дата звернення 26.09.2021).

Наступний фрагмент коду демонструє перевіркутокену отриманого від користувача для видачі йому доступу.

```
func (h *Handler) SignIn(c *gin.Context) {
	inp := new(signInput)
	iferr := c.BindJSON(inp); err != nil {
		c.AbortWithStatus(http.StatusBadRequest)
		return
	}
	token,    err    :=    h.useCase.SignIn(c.Request.Context(),
inp.Username, inp.Password)
	iferr != nil {
		iferr == auth.ErrUserNotFound {
			c.AbortWithStatus(http.StatusUnauthorized)
			return
		}
		c.AbortWithStatus(http.StatusInternalServerError)
		return
	}

	c.JSON(http.StatusOK, signInResponse{Token: token})
}
```

Нижче приведена структура створення боксу для зберігання даних рослин, котрі знаходяться в ньому

```
typeBoxstruct {
	ID     string `json:"id"`
	UserIDstring `json:"userId"`
	QrHashstring `json:"qrHash"`
	Modelstring `json:"model"`
}
typeHandlerstruct {
	useCasebox.UseCase
}

funcNewHandler(useCasebox.UseCase) *Handler {
	return&Handler{
		useCase: useCase,
	}
}

typecreateInputstruct {
	QrHashstring `json:"qrHash"`
	Modelstring `json:"model"`
}

func (h *Handler) CreateBox(c *gin.Context) {
	inp := new(createInput)

	iferr := c.BindJSON(inp); err != nil {
		c.AbortWithStatus(http.StatusBadRequest)
		fmt.Println("Troublewithparsing JSON")
		return
	}
```

```
	user := c.MustGet(auth.CtxUserKey).(*models.User)

	iferr := h.useCase.CreateBox(c.Request.Context(), user,
inp.QrHash, inp.Model); err != nil {
		c.AbortWithStatus(http.StatusInternalServerError)
		fmt.Println("Troublewithcreatingbox...")
		return
	}

	c.JSON(http.StatusOK, map[string]interface{}{
		"input":   inp,
		"message": "Executed!",
	})
}
```

3.3 Підключення до бази даних

Для роботи з MongoDB нам знадобиться драйвер mgo. Для підключення до сервера MongoDB необхідно використовувати функцію `mgo.Dial()`, до якої передається адреса сервера.

Всі дані в базах даних MongoDB структуровані колекціями. І щоб отримати звернення до потрібної колекції, необхідно використати метод C().

JSON документи в mongoDB зберігаються у двійковому форматі, званому BSON. На відміну від інших баз даних, у яких JSON дані зберігаються у вигляді рядків і чисел, кодування BSON додає нові типи, такі як int, long, date, float.

Це значно спрощує обробку, сортування та порівняння даних додатками. Драйвер Go має два типи типів для представлення даних BSON: Типи D і типи RAW[17][1)].

На рис 3.2 фрагмент коду, який відповідає за створення та перегляд існуючих користувачів у базі даних MongoDB.

[1)][17]Создание и тестирование API с помощьюGolang и MongoDB. URL: https://ichi.pro/ru/sozdanie-i-testirovanie-api-s-pomos-u-golang-i-mongodb-autentifikacia-s-pomos-u-jwt-fiksacia-zavisimostej-i-proverka-d-221986626186304 (дата звернення 08.10.2021).

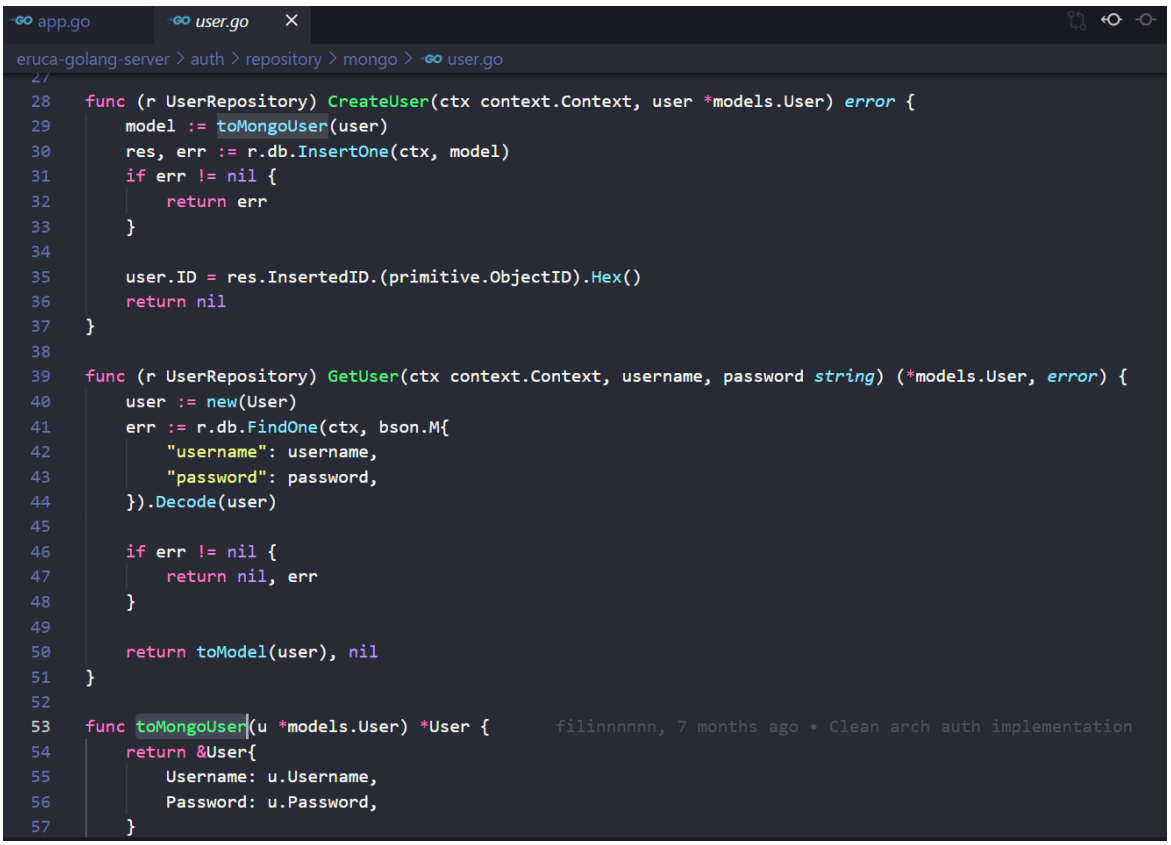

Рисунок 3.2 – Створення та прегляд користувачів в базі даних

3.3 Реалізація створення запитів у Postman

Тестування API проводять, ґрунтуючись на бізнес-логіці програмного продукту. Тестування API відноситься до інтеграційного тестування, а значить під час нього можна відловити помилки взаємодії між модулями системи або між системами. Для тестування використовують спеціальні інструменти, де можна надіслати вхідні дані у запиті та перевірити точність вихідних даних[18][1].. Одним із таких інструментів якраз і є Postman. Ось що він уміє:

– складати та надсилати запити;

– зберігати запити до папок та колекцій;

[1][18]Изучаемосновныевозможности популярного инструментаинтеграционноготестирования. URL: https://gb.ru/posts/kak-testirovat-api-ili-postman-dlya-chajnikov (дата звернення 08.10.2021).

– параметризувати запити;
– подавати до дзвінка API контрольні точки;
– створювати різні оточення для тих самих запитів;
– запускати колекції за допомогою CollectionRunner та використовувати їх як автоматизовані тести.

Запити Postman зберігаються в колекціях, тому потрібно не тільки вигадати назву та опис запиту, але й створити колекцію, де вони зберігатимуться[19][1)].

На рис.3.3 демонструється запит створення боксу за допомогою Postman.

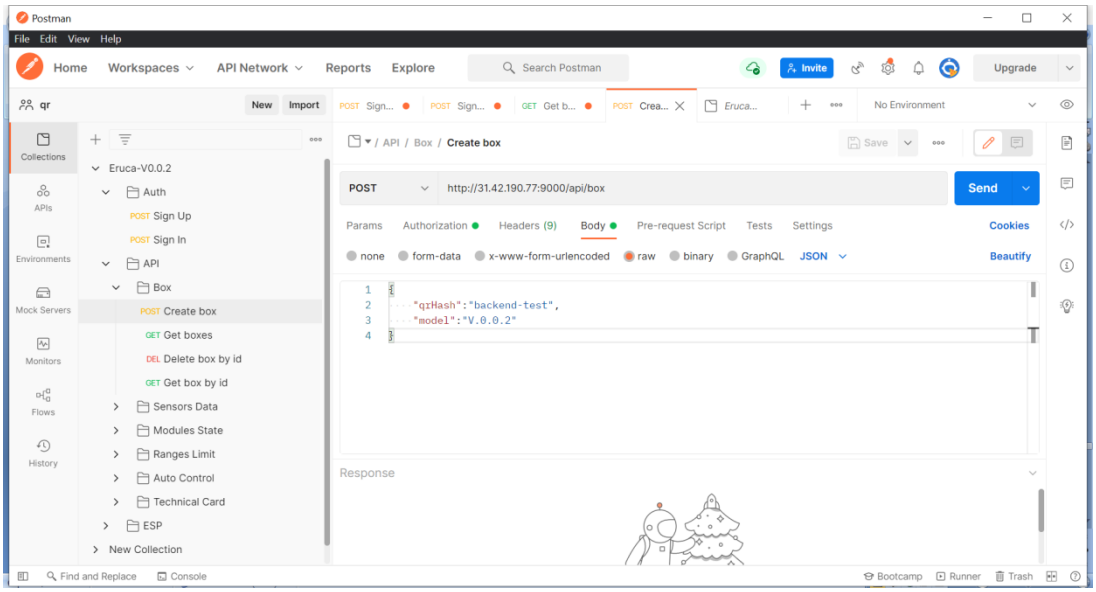

Рисунок 3.3 - створення боксу за допомогою Postman

API може бути внутрішнім, приватним – коли програмні компоненти пов'язані між собою та використовуються всередині системи. А може бути відкритим, публічним — у такому разі він дозволяє зовнішнім користувачам

[1)][18]Коллекция в Postman. URL: https://software-testing.ru/library/testing/testing-automation/2970-creating-a-postman-collection (08.10.2021).

або іншим програмам отримувати інформацію, яку можна інтегрувати у свої програми.

Щоб програмам спілкуватися між собою, їх API потрібно побудувати за єдиним стандартом. Одним із них є REST — стандарт архітектури взаємодії додатків та сайтів, який використовує протокол HTTP. Особливість REST у цьому, що сервер не запам'ятовує стан користувача між запитами. Іншими словами, ідентифікація користувача (авторизаційнийтокен) та всі параметри виконання операції передаються у кожному запиті. Цей підхід настільки простий та зручний, що майже витіснив усі інші.

На рис. 3.4 наведено структуру колекції запитів у Postman.

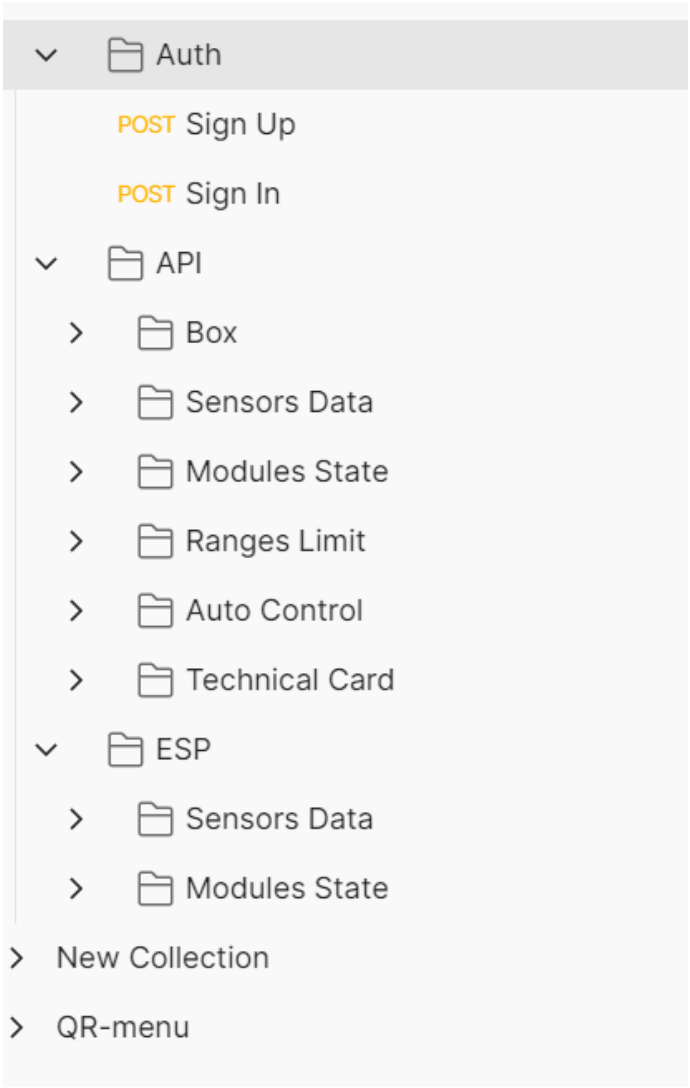

Рисунок 3.4 - Структуруколекціїзапитів у Postman.

Mock сервер – дозволяєрозгорнути заглушки для API на серверах Postmanрозробників. Доступно 1000 запитів на місяць.

ApiMonitoring – дозволяєстворюватиколекції для моніторингу та запуску їх за розкладомізсерверіврозробникаPostman. Система моніторингузробить не більше 1000 запитів на місяць

ApiDocumentation – дозволяєпублікувати автоматично згенерованудокументацію на серверах програмиPostman. Обмеження 1000 переглядів на місяць

4 ОПИС РОБОТИ З ПРОГРАМОЮ

4.1 Інсталяція прогамного додатку

Для інсталяції системи необхідно розпакувати пакет з програмою, за допомогою додатку Docker (див рис 4.1). Після запуску контейнеру потрібно перейти до IDE VisualStudioCode та завантажити всі необхідні пакети, прописані у файлі Makefile. Після завантаження пакетів, достатньо прописати у терміналі команду `makerun`, яка повинна запустити сервер. Після запуску серверу можна використовувати програму перейшовши за адресою, або відправляти запити за допомогою додатку Postman.

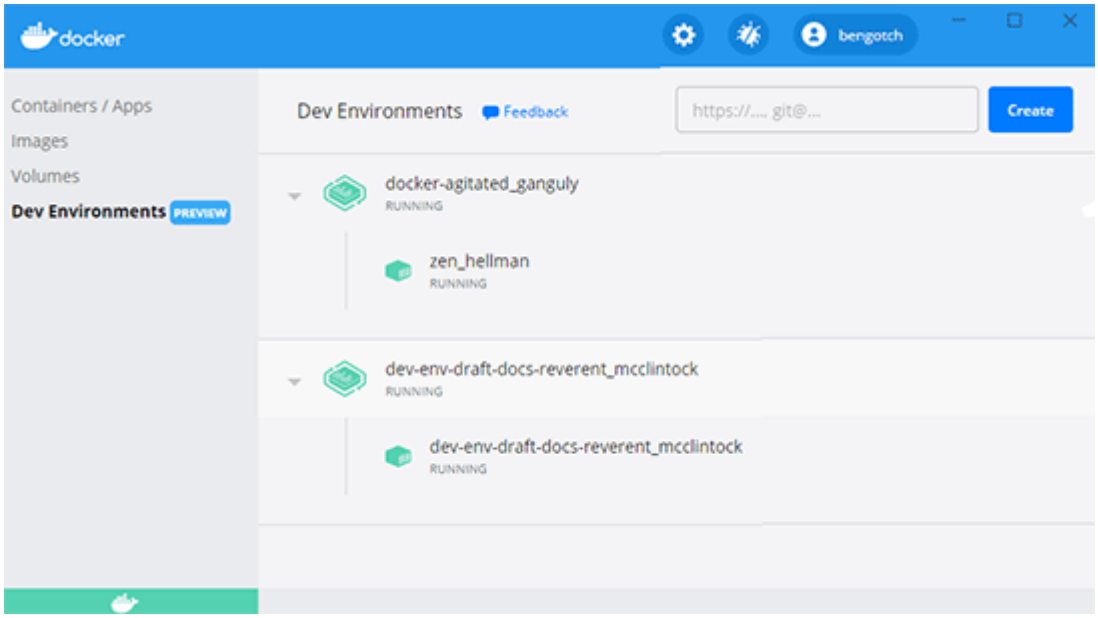

Рисунок 4.1 – Вікно додатку Docker з розпакованим контейнером

4.2Функціонал програми

Зустрічає користувача вікно завантаження з фоном у вигляді зображення зеленого листя (див рис. 4.2).

Рисунок 4.2 – Вікно завантаження

Після завершення завантаженя, відкривається вікно входу до облікового запису. Для незареєстрованих користувачів потрібно пройти процедуру реєстрації, якщо обліковий запис вже є, необхідно ввести E-mail та пароль для авторизації. (див. рис. 4.2).

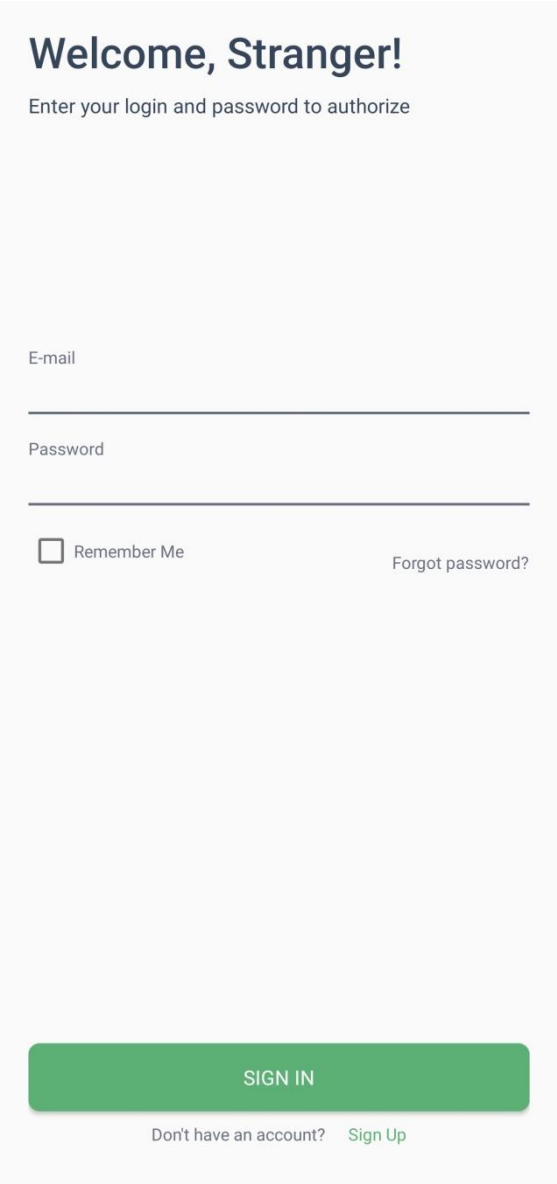

Рисунок 4.3 – Вікно авторизації

Якщо користувач забув пароль, є можливість його відновити. Після заповнення поля E-mail та натискання кнопки Signin, на пошту користувача нуде надіслано лист з інструкцією та посиланням на відновлення паролю (див. рис. 4.3).

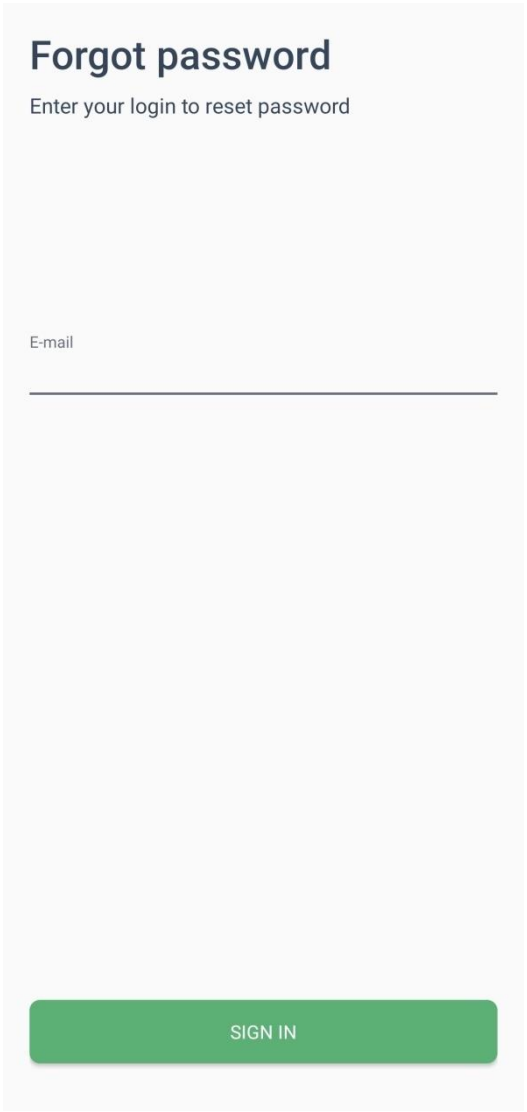

Рисунок 4.4 – Вікно відновлення пароля

На рис. 4.5 зображена головна сторінка додатку після вдалого входу в акаунт користувача. В цьому вікні відображаються дані поточного стану показників обраного боксу з рослинами. Дані обробляються на сервері та відправляються динамічно у додаток. Є можливість керувати поточним станом показників, за рахунок ввімкнення та вимикання систем постачання води, освітлення, повітря тощо.

Є можливість виклику вікна з інформацією про всі доступні бокси з рослинами поточного коричтувача (див. рис 4.6).

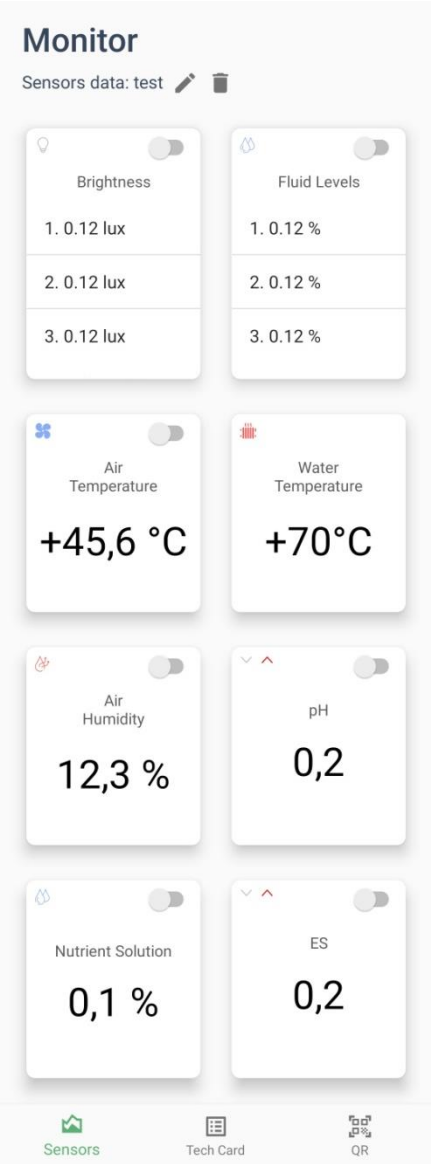

Рисунок 4.5 – Дані поточного стану показників обраного боксу з рослинами

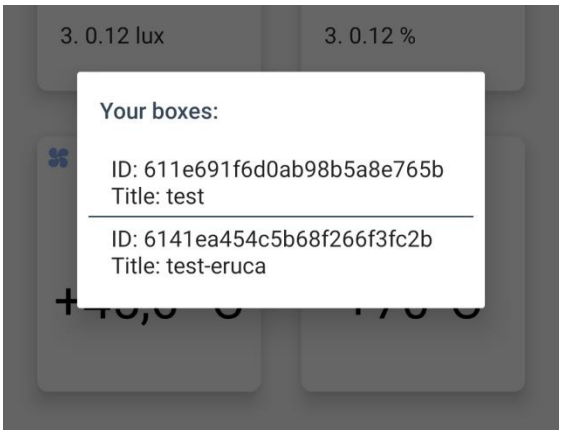

Рисунок 4.6 − Вікно з інформацією про всі доступні боксикоричтувача

На наступному скріншоті показаний функціонал створення новго боксу чи редагування вже існуючого(див. рис. 4.7). Для кожного боксу створюється своя технічна карта, в якій можна визначити показники для життєдіяльності рослин .

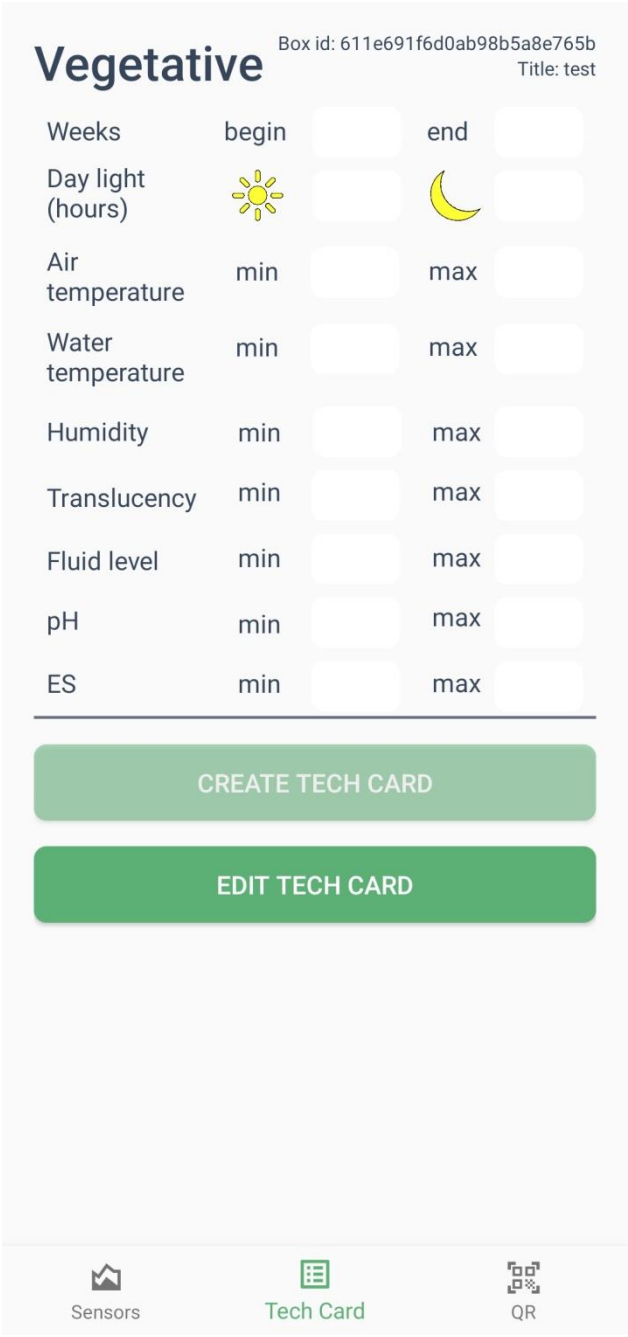

Рисунок 4.7 – Функціонал створення та редагування боксу з рослинами

Для того, щоб не вводити дані вже створених боксів з рослинами вручну, присутній функціонал додавання боксів за допомогою сканування QR-коду (див. рис. 4.8).

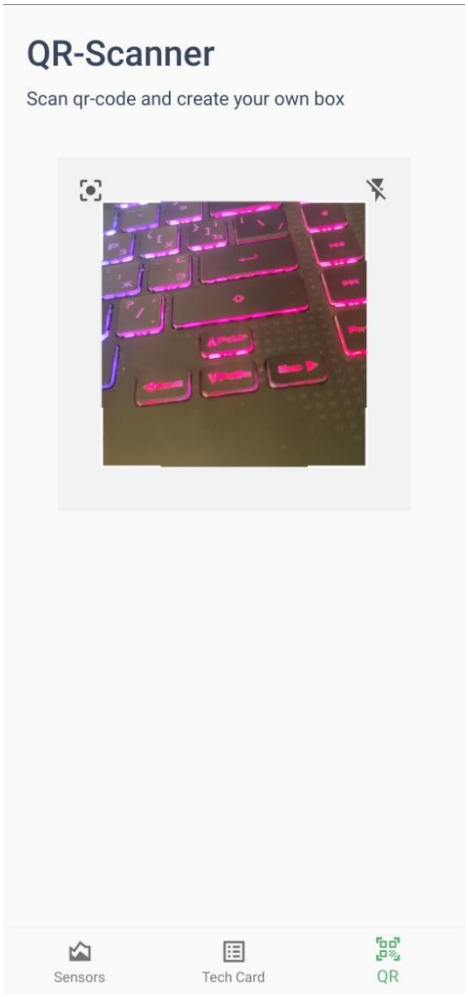

Рисунок 4.8 – Функціонал сканування QR-коду

На рис. 4.9-4.15 демонстрація встановлення ручного контролю над різними показниками життєдіяльності рослин, а саме освітлення, вологість грунту, температура та вологість повітря, вміст кисня в приміщенні тощо.

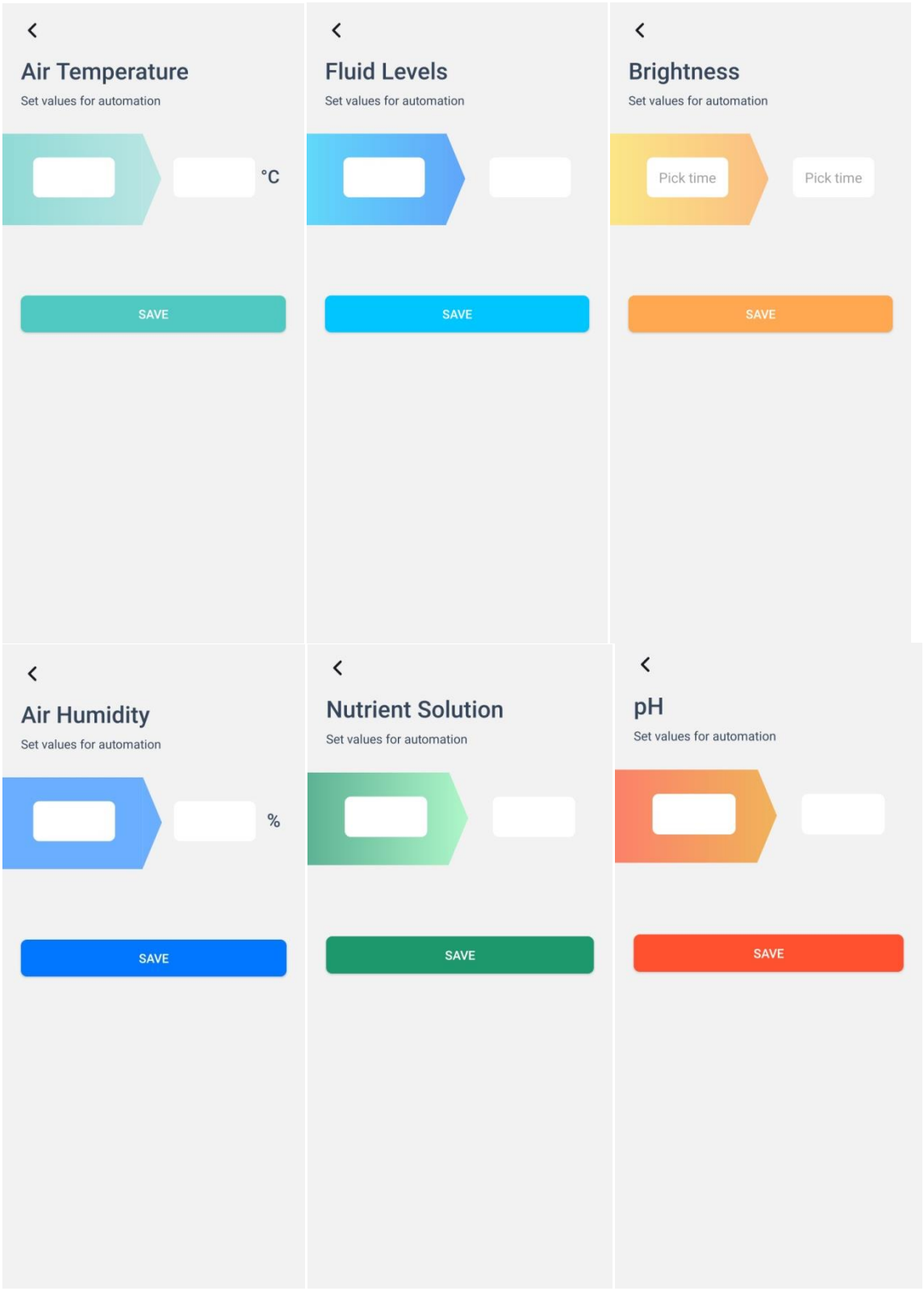

Рисунок 4.9-4.15 – Встановлення ручного контролю над показниками

ВИСНОВКИ

У магістерській роботі був проведений аналіз існуючих рішень для аналізу та керуванням екологічними показниками життєдіяльності рослини, вивчення їх функціональних можливостей та методів реалізації. У результаті аналізу було виявлено, що існуючі рішення здебільшого мають складну структуру та здійснено їх порівняльну характеристику.

Був спроектований та розроблений програмний продукт, що повністю задовольняє вимогам. Система надає змогу користувачу отримати доступ до сервісів, виконуючих процес аналізу та керування.

Було досліджено методи і засоби програмної реалізації продукту. В результаті чого було обрано створення додатку розробленого для мережевого використання. Додаток розроблений на мові “Go” з використанням мережевих технологій розробки. Це дає змогу підвищити гнучкість та зручність системи в процесі розробки а також у процесі використання, а також надає кросплатформеного характеру.

Програмний продукт було випробуванотапротестовано і задовольняє всім програмним вимогам. Додаток призначений для людей, які прагнуть слідкувати за екологічними показниками життєдіяльності рослини та керувати ними.

Отже, було розроблено системумережевого використання, що реалізує відображення даних,їх аналіз та надає можливість керувати ними за допомогою будь-якого пристрою під’єднаного до мережі інтернет. Програма вже на даному етапі є доступною для загального користування.

У подальшому є можливості до розвинення проекту. Можливо додати інші мови до інтерфейсу та покращити оптимізацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. СельскоехозяйствоУкраины – чегождать в будущем?. URL: https://www.fao.org/europe/news/detail-news/ru/c/447171/ (дата звернення 09.07.2021).
2. Мониторингполей в сельскомхозяйстве. URL: https://blog.agrokebety.com/monitoring-poley-v-selskom-khozyaystve/ (дата звернення 12.07.2020)
3. МониторингвлажностипочвыAquaSpy. URL: https://www.amitytech.com/ru/crop-management-tools/aqua-spy-soil-moisture-monitoring/ (дата звернення 21.07.2021).
4. EOS CropMonitoring: Спутниковыетехнологии в сельскомхозяйстве. URL: https://eos.com/ru/products/crop-monitoring/ (дата звернення 21.07.2021).
5. Приложения в GooglePlay - EnviroMonitorURL: https://play.google.com/store/apps/details?id=com.davisinstruments.enviromonitor&hl=en_US&gl=US (дата звернення 21.07.2021).
6. PhytoVisionМониторинг роста растений. URL: https://paskal-group.com/ru/phytovision/ (дата звернення 21.07.2021).
7. Программированиесетевых задач – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Программирование_сетевых_задач (дата звернення 07.08.2021).
8. Руководство по Node.js – Хабр. URL: https://habr.com/ru/company/ruvds/blog/422893/ (дата звернення 09.08.2021).Java – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Java#Основные_особенности_языка (дата звернення 21.08.2021).

9. Go – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Go (дата звернення 18.08.2021).
10. Полноепрактическоеруководство по Docker: с нуля до кластера на AWS URL: https://habr.com/ru/post/310460/ (дата звернення 23.08.2021).
11. Введение в Postman– Хабр. URL: https://habr.com/ru/company/kolesa/blog/351250/ (дата звернення 26.08.2021).
12. Reg.ru: Руководство по MongoDB. URL: https://help.reg.ru/hc/ru/articles/4408054704785 (дата звернення 03.09.2021).
13. JetBrains: Introduction. URL: https://www.jetbrains.com/help/go/meet-the-product.html 15.09.2021)(дата звернення 12.09.2021)..
14. ВикипедияVisualStudioCode URL: https://ru.wikipedia.org/wiki/Visual_Studio_Code (дата звернення 12.09.2021).
15. Metanit: Архитектура приложений Onion-архитектура. URL: https://metanit.com/sharp/mvc5/23.1.php (дата звернення 30.09.2021).
16. Создание и развёртывание REST API с помощьюGo – Tproger.URL: https://tproger.ru/translations/deploy-a-secure-golang-rest-api/ (дата звернення 26.09.2021).
17. Создание и тестирование API с помощьюGolang и MongoDB. URL: https://ichi.pro/ru/sozdanie-i-testirovanie-api-s-pomos-u-golang-i-mongodb-autentifikacia-s-pomos-u-jwt-fiksacia-zavisimostej-i-proverka-d-221986626186304 (дата звернення 08.10.2021).
18. Изучаемосновныевозможности популярного инструментаинтеграционноготестирования. URL: https://gb.ru/posts/kak-testirovat-api-ili-postman-dlya-chajnikov (дата звернення 08.10.2021).

19. Коллекция в Postman. URL: https://software-testing.ru/library/testing/testing-automation/2970-creating-a-postman-collection (08.10.2021).