**Міністерство освіти і науки України**

**Чернівецький національний університет**

**імені Юрія Федьковича**

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
(повна назва інституту/факультету)

Кафедра інформаційних технологій та комп'ютерної фізики
(повна назва кафедри)

# Інформаційна система для «розумного» будинку «SweeMe». Апаратна частина

**Кваліфікаційна робота**

**Рівень вищої освіти - перший (бакалаврський)**

Виконав:
студент *4* курсу, групи *417ск*
спеціальності
*126 Інформаційні системи та технології*
(назва спеціальності)

*Радомський Євгеній Віталійович*
(прізвище та ініціали)
Керівник *д.т.н., доц., Баловсяк С. В.*
(науковий ступінь, вчене звання, прізвище та ініціали)

**До захисту допущено:**

**Протокол засідання кафедри №** 20

від „15" березня 2023 р.

зав. кафедри _____ доц. Борча М. Д.

Чернівці – 2023

# РЕЦЕНЗІЯ НА БАКАЛАВРСЬКУ РОБОТУ

**Інформаційна система для «розумного» будинку «SweeMe». Серверна частина**

**Студента** Радомського Євгена Віталійовича

Інститут фізико-технічних та комп'ютерних наук

Спеціальність інформаційні системи та технології

Чернівецького національного університету імені Юрія Федьковича

**Актуальність теми.** Розробка інформаційної системи «розумного» будинку передбачає, насамперед, зручне і ефективне використання пристроїв, забезпечення економії ресурсів, автоматизацію рутинної роботи, створення більш безпечного середовища. Тому тема роботи і реалізація серверної частини «розумного» будинку є актуальними.

**Практичне значення.** Впровадження інформаційної системи та API для взаємодії з нею через мобільний додаток в системі розумного будинку забезпечує зручний та простий спосіб керування всіма функціями розумного будинку. Як наслідок – відбувається оптимізація енергоспоживання та забезпечення більш ефективного управління ресурсами.

**Структура та зміст роботи.** Робота складається з 3х розділів. У першому розділі неведені теоретичні відомості про інформаційну систему, описано принцип взаємодії всіх елементів з «розумним» будинком. Наведено обгрунтування вибору системи керування базами даних, мови програмування, серверного середовища. У другому розділі визначено набір технологій, які найкраще відповідають вимогам та потребам проекту. Описано функції, структура системи та її робота з прикладом передачі та відображення даних у базу даних. У третьому розділі описана реалізація інформаційної системи та її взаємодія з API. Продемонстрована робота інформаційної системи та її можливості.

**Зауваження.** Було б добре додати можливість сповіщення клієнта у критичних чи аварійних випадках роботи пристроїв системи.

**Оцінка**. В результаті виконання бакалаврської роботи студент Радомський Є. В. здійснив розробку API для взаємодії з «розумним» будинком в інформаційній системі "SweeMe". Запропонована інформаційна система та API є гнучкою та має потенціал для модифікацій та покращень. Вважаю, що завдання бакалаврської роботи виконано повністю, а Радомський Є. В. заслуговує оцінки „відмінно" та присвоєння кваліфікації бакалавра зі спеціальності 126 – інформаційні системи та технології.

**Рецензент**
Доцент кафедри комп'ютерних систем та мереж
Чернівецького національного університету
Кандидат технічних наук
Яковлева Інна Дмитрівна

"15" 06 2023 р.

UNICHECK
by Turnitin

**Ім'я користувача:**
**Кафедра інформаційних технологій та комп фізики**

**ID перевірки:**
**1015655382**

**Дата перевірки:**
**20.06.2023 11:29:15 EEST**

**Тип перевірки:**
**Doc vs Internet + Library + DB**

**Дата звіту:**
**20.06.2023 11:29:39 EEST**

**ID користувача:**
**36471**

**Назва документа: Радомський**

**Кількість сторінок: 14** **Кількість слів: 7354** **Кількість символів: 61612** **Розмір файлу: 100.99 KB** **ID файлу: 1015300663**

# 9.3%
# Схожість

**Найбільша схожість: 9.3% з джерелом з Бібліотеки (ID файлу: 1015297743)**

**Не знайдено джерел з Інтернету**

**9.3% Джерела з Бібліотеки** | **1** ..... **Сторінка** 16

# 0% Цитат

**Не знайдено жодних цитат**

**Вилучення списку бібліографічних посилань вимкнене**

# 3.29%
# Вилучень

**Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за 12 слів та 2%)**

**0.63% Вилучення з Інтернету** | **43** ..... **Сторінка** 17

**3.29% Вилученого тексту з Бібліотеки** | **236** ..... **Сторінка** 17

**Міністерство освіти і науки України**

**Чернівецький національний університет**

**імені Юрія Федьковича**

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
(повна назва інституту/факультету)

Кафедра інформаційних технологій та комп'ютерної фізики
(повна назва кафедри)

# Інформаційна система для «розумного» будинку «SweeMe». Апаратна частина

**Кваліфікаційна робота**

**Рівень вищої освіти - перший (бакалаврський)**

Виконав:
студент *4* курсу, групи *417ск*
спеціальності
*126 Інформаційні системи та технології*
(назва спеціальності)

*Радомський Євгеній Віталійович*
(прізвище та ініціали)
Керівник *д.т.н., доц., Баловсяк С. В.*
(науковий ступінь, вчене звання, прізвище та ініціали)

**До захисту допущено:**

**Протокол засідання кафедри № ____**

від ,,____" _____ 2023 р.

зав. кафедри _____доц. Борча М. Д.

Чернівці – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
док. фіз.-мат. наук, доц.
_____М. Д. Борча
”___” _____ 2023 р.

# Інформаційна система для «розумного» будинку «SweeMe». Апаратна частина

ЛИСТ ЗАТВЕРДЖЕННЯ

**УЗГОДЖЕНО**
Керівник роботи
докт. Фіз-мат. наук, доцент
_____С.В. Баловсяк
“___” _____ 2023 р.

Виконавець
студент 4-го курсу
_____ Є. В. Радомський
“___” _____ 2023 р.

2023

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

*Радомському Євгенію Віталійовичу*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна система для «розумного» будинку «SweeMe». Апаратна частина*

керівник роботи *Баловсяк Сергій Васильович, докт. техн. наук, доцент*,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від "__"_____2023 року №_____

2. Строк подання студентом проекту (роботи) *2023 р.*

3. Вихідні дані до проекту (роботи) Мета роботи – розробити інформаційну систему «розумного» дому "SweeMe" та специфічної частини цієї системи – розробка апаратної частини. Система повинна мати сенсори DHT на мікроплатах ESP, обмін даними між кінцевим та центральним пристроєм, та обмін даними між сервером та центральним пристроєм. Програмне забезпечення розробити за допомогою мов програмування C++, PHP та програмного середовища Laravel.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

*1) дослідити існуючі інформаційні системи-аналоги*

*2) описати схему роботи між кінцевими та ценральним пристроєм , описати роботу програмного забезпечення для центрального пристрою.*

*3) розробити прототип інтерфейсу*

*4) розробити прототит кінцевого пристрою на базі ESP*

*5) дослідити ефективність роботи розробленої програми*

5. Перелік графічного матеріалу

*1) Інтерфейс користувача*

*2) Загальна схема роботи системи*

*3) Логічна модель бази даних*

**Студент** _____ *Радомський Є.В*

(підпис) (прізвище та ініціали)

**Керівник проекту (роботи)** _____ *Баловсяк С.В,*

*(підпис) (прізвище та ініціали)*

# АНОТАЦІЯ

Кваліфікаційна робота виконана студентом групи 417ск Радомським Євгенієм Віталійовичем. Тема «Інформаційна система для «розумного» будинку «SweeMe». Апаратна частина». Робота направлена на здобуття ступеня бакалавр за спеціальністю 126 «Інформаційні системи та технології»

Метою кваліфікаційної роботи є розробка інформаційної системи «розумного» дому "SweeMe" та специфічної частини цієї системи - розробка апаратної частини. Реалізація системи виконана за допомогою мови програмування PHP , його фреймворку Laravel та C++ .

Бакалаврська робота містить: кількість сторінок – 69, рисунків – 25, додатків – 1, використаних джерел – 12.

**Ключові слова**: інформаційна система, апаратна частина, веб-інтерфейс , PHP, Laravel, C++, програмне забезпечення.

Робота містить результати власних досліджень. Використання чужих ідей, результатів і текстів мають посилання на відповідне джерело.

# ANNOTATION

The qualification work was carried out by the student of group 417sk, Radomskyi Yevhenii Vitaliiovych. The topic is "Information System for the 'SweeMe' Smart Home (Hardware Component)". The work is aimed at obtaining a bachelor's degree in the field of specialization 126 "Information Systems and Technologies".

The goal of the qualification work is to develop an information system for the "SweeMe" smart home and its specific component - the hardware part. The system implementation is done using the PHP programming language, its Laravel framework, and C++.

The bachelor's thesis consists of 69 pages, 25 figures, 1 appendix, and references to 12 sources used.

Keywords: information system, hardware component, web interface, PHP, Laravel, C++, software.

The work includes the results of the author's own research. The use of external ideas, findings, and texts is properly referenced to the respective sources.

**ЗМІСТ**

# ВСТУП

Метою даної кваліфікаційної роботи є розробка інформаційної системи під назвою "SweeMe" для "розумного" дому з метою створення комфортного та ефективного середовища для користувачів. У сучасному світі, де технології швидко розвиваються, інтелектуальні системи в побутовій сфері стають все більш популярними.

Відмінність "розумних" будинків полягає в їх здатності автоматизувати різні аспекти повсякденного життя, забезпечуючи контроль та управління різними пристроями і системами, такими як освітлення, опалення, системи безпеки та багато інших.

Основна мета цього дослідження полягає у розробці ефективної та надійної інформаційної системи для керування розумним будинком, яка відповідає зростаючій популярності "розумних" систем у побутовій сфері.

Одним з головних викликів є збільшення вартості електроенергії та потреба в ефективному використанні ресурсів. Інформаційна система "SweeMe" пропонує вирішення цих проблем шляхом надання повного контролю над електричними пристроями та можливості ефективного енергозбереження.

Основні завдання дослідження включають аналіз предметної області, проектування інформаційної системи та вибір відповідного інструментарію та методів для реалізації цієї системи. Аналіз предметної області передбачає вивчення сучасних технологій та рішень у галузі "розумних" будинків, а також виявлення основних потреб і вимог користувачів.

На основі цього аналізу буде проведене проектування інформаційної системи "SweeMe", яка включатиме розробку інтерфейсу користувача для зручної взаємодії з розумним будинком.

Одна з ключових ідей цього дослідження полягає в можливості розробки і налаштування пристроїв, які використовуються в розумному

будинку, під вимоги кожного клієнта. Користувачі зможуть налаштувати параметри та взаємодію пристроїв з урахуванням своїх потреб і вподобань.

Інформаційна система "SweeMe" надасть індивідуалізований підхід до керування розумним будинком, забезпечуючи комфорт та енергоефективність для кожного користувача. Таким чином, ця кваліфікаційна робота спрямована на розробку інформаційної системи "SweeMe" для "розумного" дому, яка має на меті створення комфортного, енергоефективного та індивідуалізованого середовища для користувачів. Це дослідження розглядає аспекти аналізу галузі, проектування системи та вибір відповідного інструментарію, забезпечуючи розвиток технологій для покращення "розумних" будинків та задоволення потреб сучасного споживача.

Завдання дослідження включають:

- Аналіз предметної області, дослідження потреб користувачів розумного будинку та визначення необхідного функціоналу для апартного забезпечення.
- Проектування загальної системи розумного будинку, загальна схема роботи та підключення апаратного забезпечення до системи.
- Проектування інформаційної системи, включаючи створення концептуальної, логічної та фізичної моделей бази даних для збереження інформації про користувачів та пристроїв.
- Розробка інтерфейсу для локального в межах будинку, управління пристроями.

Реалізація інтерфейсу системи виконана мовою PHP та популярного фреймворку Laravel. Програмне забезпечення для апартної частини розроблена на C++ та спеціальних бібліотек та мікроплат ESP32 та ESP8266

У результаті виконання бакалаврської роботи розроблено інформаційну систему та інтерфейс для моніторингу та управління в межах локальної мережі, яка призначена для системи «розумного» будинку.

Об'єктом дослідження є інформаційна система «розумного» дому "SweeMe".

Предметом дослідження є технології проектування та розробки інформаційних систем, методики аналізу та оцінки інформаційних систем, а також використання популярних технологій для розробки інтерфейсу користувача.

# РОЗДІЛ 1

## 1.1 Аналіз об’єкту дослідження та предметної області

### 1.1.1 Теоретичні відомості про об’єкт дослідження та предметну область

Інформаційна система є складним комплексом взаємопов'язаних компонентів, включаючи апаратне та програмне забезпечення, бази даних, мережеві зв'язки і людські ресурси. Ці компоненти працюють спільно для збору, зберігання, обробки, передачі та використання інформації з метою підтримки процесу прийняття рішень, виконання завдань і досягнення конкретних цілей. Інформаційні системи забезпечують обробку і передачу даних, спілкуються з користувачами і допомагають виконувати різноманітні функції, спрощуючи роботу і підвищуючи ефективність управлінських процесів в різних сферах діяльності.

Основні компоненти інформаційної системи включають:

- Апаратне  забезпечення
- Програмне забезпечення
- Бази даних(БД)
- Мережеві зв'язки
- Людські ресурси

Апаратне забезпечення представляє собою фізичні пристрої, що включають комп'ютери, сервери, мережеві пристрої, сенсори, пристрої зберігання даних та інші обчислювальні пристрої. Ці різноманітні компоненти становлять фундамент апаратної складової інформаційних систем, використовуваних для обробки, зберігання та передачі інформації. Комп'ютери забезпечують виконання розрахунків та обробку даних, сервери забезпечують централізоване зберігання та управління ресурсами, мережеві пристрої забезпечують підключення та комунікацію між компонентами

системи, а сенсори забезпечують збір реального часу та зовнішніх даних. Крім того, пристрої зберігання даних використовуються для збереження інформації на постійних носіях, таких як жорсткі диски або флеш-пам'ять. Усі ці пристрої працюють разом, надаючи необхідну інфраструктуру для оптимального функціонування інформаційних систем.

Програмне забезпечення складається з різноманітних програм і систем, які встановлюються на апаратному забезпеченні з метою виконання конкретних функцій. Цей компонент може включати операційні системи, бази даних, програми для обробки даних, додатки для взаємодії з користувачами та інші спеціалізовані програми. Програмне забезпечення відповідає за керування та координацію роботи апаратних компонентів, забезпечуючи їх взаємодію та ефективне виконання завдань. Воно визначає можливості та функціональність системи, надаючи користувачам інструменти для роботи з інформацією та здійснення потрібних операцій.

Бази даних представляють собою структури для організованого зберігання великого обсягу даних. Вони використовуються для забезпечення доступу, пошуку, модифікації та управління даними, які використовуються в інформаційній системі. Бази даних відіграють важливу роль у забезпеченні ефективного зберігання та організації інформації, дозволяючи ефективно виконувати операції з даними. Вони забезпечують структурованість, цілісність та безпеку даних, а також забезпечують можливість використання різних запитів та аналізу даних для підтримки рішень та досягнення цілей інформаційної системи.

Мережеві зв'язки представляють собою інфраструктуру, яка забезпечує передачу даних і зв'язок між різними компонентами системи. Вони можуть включати локальні мережі (LAN), бездротові мережі (Wi-Fi, Bluetooth), глобальну мережу Інтернет та інші комунікаційні канали. Мережеві зв'язки грають важливу роль у забезпеченні зв'язку і передачі даних між компонентами інформаційної системи. Вони дозволяють обмінюватися

інформацією, спілкуватися, а також забезпечують доступ до ресурсів та послуг, які потрібні для ефективної роботи системи. Мережеві зв'язки можуть бути провідними або бездротовими, локальними або глобальними, і вони відіграють ключову роль у забезпеченні зв'язку та обміну даними в сучасних інформаційних системах.

Людські ресурси в контексті інформаційних систем - це колектив фахівців, які взаємодіють з системою, включаючи адміністраторів, програмістів, користувачів та інших спеціалістів. Людські ресурси виконують різноманітні ролі і завдання, пов'язані з конфігурацією, розробкою, впровадженням, підтримкою та використанням інформаційної системи. Вони відповідають за налагодження та настройку системи, програмування та розробку необхідного програмного забезпечення, надання підтримки та консультацій користувачам, а також за забезпечення безперебійної роботи системи та вирішення поточних проблем. Людські ресурси є важливим складовим елементом успішної функціонування інформаційних систем, оскільки вони забезпечують експертні знання та навички для ефективного управління системою та досягнення поставлених цілей.

Ці елементи взаємодіють між собою для забезпечення збору, зберігання, обробки, передачі та використання інформації у рамках інформаційної системи. Кожен з цих елементів відіграє важливу роль у функціонуванні та ефективності системи в цілому.

Для початку розробки інтерфейсу користувача потрібно ознайомитися з деякими поняттями:

Проектування систем - це процес створення плану або концепції для створення або розвитку інформаційної системи. Цей процес включає в себе ідентифікацію потреб, аналіз вимог, розробку архітектури системи, вибір технологій, планування реалізації та імплементацію системи. Під час проектування системи враховуються потреби та вимоги користувачів, функціональність системи, обсяг та тип даних, безпека, масштабованість та

інші важливі аспекти. Проектування системи передбачає розробку логічної та фізичної структури системи, вибір необхідних компонентів (апаратного та програмного забезпечення), а також визначення способів взаємодії між компонентами та зовнішніми системами. Проектування системи також може включати створення прототипів, моделей або діаграм, що допомагають уявити та уточнити структуру та функціональність системи перед її фактичним розробленням. Метою проектування систем є створення ефективної, надійної та відповідної до потреб користувачів інформаційної системи.

Для моделювання інтерфейсу для центрального пристрою ми будемо використовувати Figma.

Figma є онлайн-інструментом для дизайну та прототипування, призначеним для створення користувацького інтерфейсу (UI) для веб-сайтів, мобільних додатків та інших цифрових продуктів. Він надає можливості спільної роботи в режимі реального часу, дозволяючи дизайнерам, розробникам та іншим зацікавленим сторонам спільно працювати над проектами, коментувати та вносити зміни одночасно.

Figma пропонує візуальний редактор, який дозволяє створювати векторні графічні елементи, інтерфейси та макети. Цей інструмент має широкий набір функцій для малювання, редагування, вирізання, групування та організації різних шарів і компонентів дизайну. Крім того, Figma підтримує стилі, компоненти та бібліотеки, що дозволяє створювати та використовувати повторювані елементи, а також автоматично оновлювати їх у всіх макетах, де вони використовуються. Такий підхід сприяє збереженню консистентності та ефективності у процесі розробки та дизайну.

Завдяки своїм онлайн-функціям, Figma надає зручність спільної роботи, де кожен учасник може бачити зміни в реальному часі, спілкуватися за допомогою коментарів та спільно вирішувати завдання. Це робить його

потужним інструментом для колаборації та творчого процесу у командному середовищі.

Для реалізації інтерфейсу ми будемо використовувати мови програмування PHP, її фреймворк Laravel для реалізації всіх прихованих від користувача процесів та мову програмування JS для візуалізації інтерфейсу користувача.

PHP - це мова програмування загального призначення, яка широко використовується для розробки веб-додатків та динамічних веб-сторінок. Основною особливістю PHP є його вбудована підтримка веб-розробки, що дозволяє вставляти PHP-код безпосередньо в HTML-сторінки.

Переваги PHP включають простоту вивчення завдяки простому синтаксису та великій кількості документації, широку спільноту та підтримку з боку розробників, вбудовану підтримку веб-розробки для швидкого створення функціональних веб-додатків і велику портативність на різних платформах.

Недоліки PHP включають потенційні проблеми безпеки, які можуть виникати через неправильну конфігурацію або програмування, а також можливі проблеми з масштабуванням на великих проектах, що вимагають оптимізації коду. Деякі неоднозначності в синтаксисі PHP також можуть створювати складнощі в розробці та обслуговуванні коду.

В цілому, PHP є потужним і популярним інструментом для веб-розробки, з багатьма перевагами, які роблять його вибором багатьох розробників. Проте, варто враховувати його недоліки та розглядати їх при розробці проектів для забезпечення безпеки та ефективності.

Laravel - це високорівневий фреймворк для розробки веб-додатків на мові програмування PHP. Він надає розробникам зручні інструменти та компоненти, що допомагають прискорити процес розробки і забезпечити

чистий та структурований код. Однією з переваг Laravel є його простота використання.

Він має зрозумілий і елегантний синтаксис, що дозволяє швидко розробляти функціонал інтернет-додатків. Laravel надає широкий спектр готових компонентів, таких як аутентифікація, маршрутизація, сесії та кешування, що допомагають зосередитися на бізнес-логіці додатку, а не на повторному виконанні загальних завдань.

Іншою перевагою Laravel є його розширюваність і модульність. Він підтримує використання сторонніх бібліотек та компонентів, що дозволяє розширювати функціонал фреймворку за потребами проекту. Laravel також надає можливість легко розбити додаток на модулі та компоненти, що спрощує його підтримку та розвиток в майбутньому.

У Laravel також є вбудована підтримка для роботи з базами даних, що дозволяє зручно виконувати операції зберігання, оновлення та вибірки даних. Фреймворк також надає можливість використання міграцій, що дозволяє легко керувати структурою бази даних та версіонувати її зміни.

Загалом, Laravel є потужним фреймворком для розробки веб-додатків з багатьма перевагами. Він допомагає зосередитися на реалізації бізнес-логіки та прискорює розробку завдяки готовим компонентам і зручним інструментам. Проте, варто бути свідомим його особливостей та забезпечити відповідність проекту вимогам фреймворку.

### 1.1.2 Принцип взаємодії інформаційної системи та інтерфейсу користувача з «розумним» будинком

Принцип взаємодії між інформаційною системою та інтерфейсом користувача "розумного" будинку базується на забезпеченні зручного та ефективного способу взаємодії між людиною та системою. Цей принцип охоплює різні аспекти, включаючи комунікаційні протоколи, засоби керування та відображення інформації.

Почнемо з того, що інформаційна система розумного будинку повинна мати здатність взаємодіяти з різними компонентами будинку, такими як освітлення, опалення, побутові пристрої і т.д. Це досягається за допомогою підключення до API.

Другим аспектом є інтерфейс користувача, який є інтерактивною межею між користувачем та системою "розумного" будинку. Інтерфейс користувача реалізується у вигляді мобільного додатку для більшої зручності та мобільності.

Основна мета інтерфейсу користувача полягає в тому, щоб зробити взаємодію з "розумним" будинком максимально зручною та доступною для користувача. Це досягається шляхом простоти використання, інтуїтивно зрозумілих елементів керування та зручного представлення інформації.

Крім того, інформаційна система може надавати користувачу розширені можливості управління та контролю. Наприклад, вона може надавати статистику температури протягом певного періоду часу.

Загалом, принцип взаємодії між інформаційною системою та інтерфейсом користувача "розумного" будинку передбачає створення зручної та інтуїтивно зрозумілої системи, яка дозволяє користувачу легко керувати різними аспектами свого будинку та отримувати необхідну інформацію.

## 1.2. Постановка задачі

### 1.2.1 Вимоги до системи

Розроблена інформаційна система складається з загального моделювання системи, упраління процесом розробки, налаштування та долідження апаратного забезпечення а також створення та візуалізація програмного забезпечення у вигляді інтерфейсу коричтувача для обміну даними між локальним центральним пристроєм та відаленим сервером, щоб налагодити всі процеси у системі.

До розробки інтерфейсу ставляться такі вимоги:

- Простота використання: Інтерфейс повинен бути легким у навігації та зрозумілим для користувача. Користувач має зможу швидко освоїти взаємодію з системою та виконувати необхідні дії без заплутування. Необхідно забезпечити зрозумілість та простоту взаємодії з системою.
- Інтуїтивність: Інтерфейс повинен використовувати відомі символи, іконки та мову, щоб користувачам було зрозуміло, що робити і як керувати системою. При розробці необхідно враховувати загальноприйняті стандарти та конвенції в дизайні інтерфейсу, щоб забезпечити легкість сприйняття та орієнтацію користувачів.
- Швидкість та відгукливість: Інтерфейс повинен бути швидким та реагувати на дії користувача без зайвих затримок. Користувач повинен мати можливість швидко здійснювати дії та одержувати миттєвий зворотний зв'язок від системи. Забезпечення оперативності взаємодії з інтерфейсом є важливим для задоволення потреб користувачів.

Реалізацію інтерфейсу виконано на за допомогою PHP та фрейворку Laravel.

До розробки програмного забезпечення для апаратної частини були поставленні наступні вимоги:

- Швидкодія: програмне забезпечення повинне відгукуватись та не перенавантужувати апаратні ресурси певних пристроїв.
- Мобільність: програмне забезпечення повинне бути якомога ресурсо-зберігаючим.

Реалізацію програмного забезпечення виконано за допомогою мови програмування C++ та середовища програмування Arduino IDE.

### 1.2.2 Очікування від впровадження системи

Впровадження інформаційної системи в системі розумного будинку, яка включає апаратне та програмне забезпечення для взаємодії з пристроями, є

ключовим етапом, що дозволяє забезпечити передачу даних до серверу через центральний пристрій.

Одна з головних переваг апаратних можливостей системи розумного будинку полягає в їх доступності для всіх користувачів. Кінцеві пристрої можуть бути компонентами інших майбутніх систем, що розширює їхні можливості. Розробники мають можливість створювати власні додатки або інтегрувати систему розумного будинку з іншими системами, такими як системи розумного міста або розумної енергетики.

Це відкриває широкі можливості для створення інноваційних рішень та дослідження нових способів оптимізації енергоспоживання і більш ефективного управління ресурсами. Інтеграція з іншими системами дозволяє впроваджувати комплексні рішення та створювати інтелектуальні мережі, що сприяють сталому розвитку та підвищенню якості життя користувачів.

**Висновки до розділу**

У даному розділі було розглянуто теоретичні аспекти інформаційної системи та принципи взаємодії її елементів з "розумним" будинком. Була встановлена постановка задачі, включаючи вимоги до системи та очікування щодо її впровадження.

Після проаналізування аналогічних рішень, які вже присутні на ринку, було прийнято рішення розробити власну інформаційну систему для "розумного" будинку. Це рішення було обґрунтовано з огляду на ціну конкурентних аналогів. Розробка власної інформаційної системи допоможе знизити витрати і зробить такий розумний будинок доступним для більш широкого кола людей, які не можуть дозволити собі дорогі інформаційні системи для "розумного" будинку.

Хоча наша система може не мати такого рівня якості, як деякі з конкурентних аналогів, вона має значний потенціал для подальшого розвитку. Основною перевагою нашої системи є її конкурентоздатність та фінансова доступність. Власна система дозволяє гнучко адаптуватися до потреб користувачів. Крім того, вона надає можливість розширювати функціональність системи та розробляти власні пристрої, що є важливою перевагою в умовах швидкого технологічного розвитку.

# РОЗДІЛ 2 ПОБУДОВА СИСТЕМИ

## 2.1 Моделювання системи

### 2.1.1 Функції та структура системи

Перш за все перед реалізаціє системи було змодельована загальна схема роботи системи(рис. 2.1).

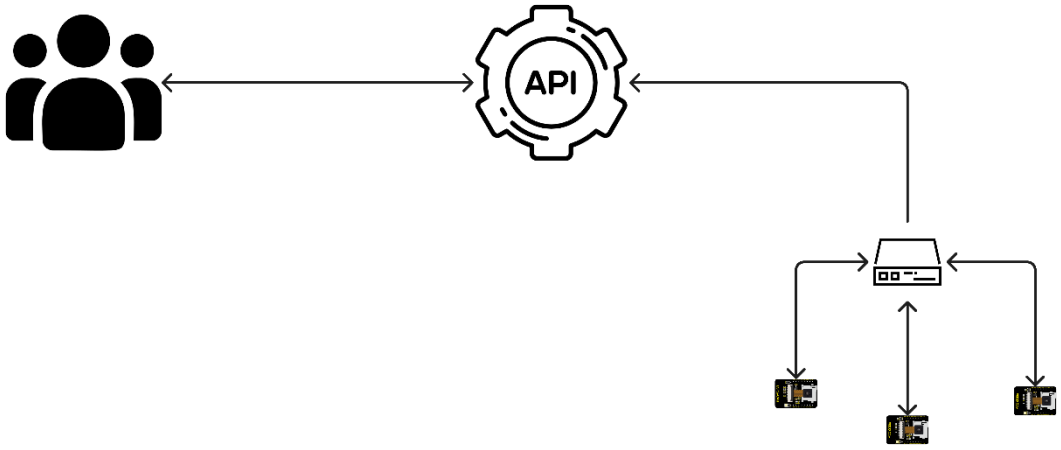



Рисунок 2.1 – Загальна схема роботи для інформаційної системи «розумного» будинку “SweeMe”

На цій схемі зображено участь окремих частин інформаційної системи як однієї цілої , а саме зв’язок між ними.

Центральний пристрій, програмне забезпечення та кінцеві пристрої виконують наступні функції:

- Забезпечення передачі даних з різних сенсорів до віддаленого сервера. Ця функція дозволяє збирати інформацію з різних датчиків або джерел даних та передавати її на віддалений сервер для подальшого аналізу, зберігання або обробки. Це може включати зчитування температури, вологості, руху, давлення та інших параметрів з датчиків і передачу цих даних до сервера для подальшого використання.
- Підключення до мережі Інтернет та обмін даними. Ця функція дозволяє центральному пристрою та кінцевим пристроям підключатися до мережі Інтернет і обмінюватися даними з іншими пристроями або

серверами. Вона забезпечує доступ до онлайн-ресурсів, можливість відправляти та отримувати дані через Інтернет, виконувати різні мережеві операції, такі як синхронізація, оновлення програмного забезпечення та інші. Ці функції відіграють важливу роль у забезпеченні комунікації, збору даних та обміну інформацією між різними компонентами системи. Вони сприяють розширенню можливостей та забезпеченню ефективної роботи всього комплексу пристроїв та програмного забезпечення.

У результаті розгляду різних аспектів інформаційної системи "розумного" будинку, було прийнято рішення щодо використання певних технологій та платформ. Для збереження даних було обрано вбудовану реляційну базу даних SQLite, яка дозволяє зручно та ефективно зберігати інформацію. Додатково, було створено декілька допоміжних таблиць, таких як "Temperatures", "Humidity", "Devices" та "Users".

Таблиця "Temperatures" використовується для збереження даних про температуру, отриманих з кінцевих пристроїв. Ця таблиця містить відповідні поля для збереження значень температури та метаданих.

Таблиця "Humidity" призначена для збереження даних про вологість, також отриманих з кінцевих пристроїв. Вона має поля, які дозволяють зберігати значення вологості та відповідні додаткові дані.

Таблиця "Devices" використовується для збереження інформації про пристрої, що входять до системи "розумного" будинку. Ця таблиця може містити дані, такі як ідентифікатор пристрою, назва, опис, стан тощо.

Таблиця "Users" містить дані про користувачів системи. При встановленні системи, вона включає одного користувача, який має доступ до інтерфейсу. Ця таблиця може містити поля, такі як ідентифікатор користувача, ім'я, електронна пошта, хеш пароля та інші дані, необхідні для аутентифікації та авторизації користувачів.

Завдяки використанню цих таблиць та відповідних полів, інформаційна система "розумного" будинку може зручно зберігати та керувати даними, отриманими з кінцевих пристроїв, а також забезпечувати доступ та керування для користувачів системи. Така комбінація технологій та таблиць дозволяє розробникам створювати ефективні та функціональні системи "розумних" будинків, забезпечуючи зручну взаємодію між людиною та будинком.

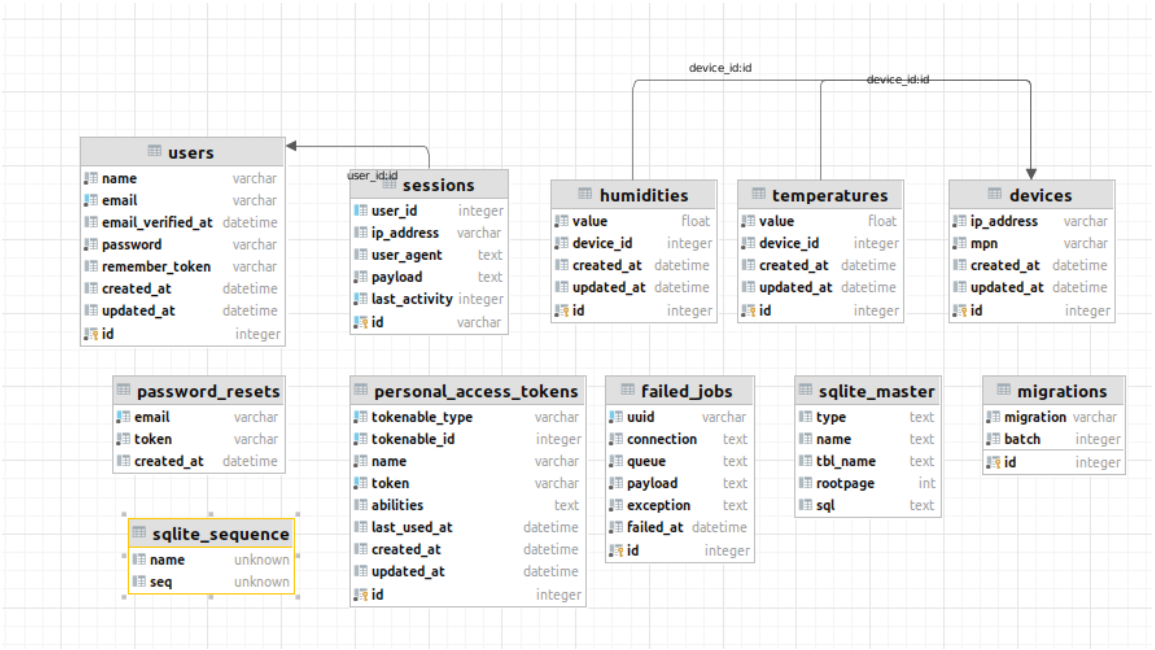



Рисунок 2.2 – Схема бази даних на центральному пристрої

### 2.1.2 Опис роботи системи

Для початку використання "розумного" будинку користувач повинен пройти процес реєстрації в додатку. Цей процес включає створення облікового запису користувача шляхом заповнення відповідної реєстраційної форми. Реєстраційна форма міститиме необхідні поля для введення особистих даних, таких як ім'я, прізвище, електронна адреса та пароль.

Користувач буде зобов'язаний заповнити ці поля з метою ідентифікації та безпеки облікового запису. Після того, як користувач введе всі необхідні дані, він зможе надіслати запит на реєстрацію, натиснувши відповідну кнопку або виконавши певну дію в додатку.

Після цього система перевірить введені дані на відповідність вимогам та наявність унікальної електронної адреси. Якщо введені дані коректні і відповідають усім вимогам, користувач буде успішно зареєстрований в системі. У такому разі він отримає підтвердження реєстрації або повідомлення про успішну реєстрацію на свою електронну адресу.

Цей процес реєстрації є необхідною передумовою для отримання доступу до функціональності "розумного" будинку та взаємодії з ним. Після реєстрації користувач зможе ввійти в свій обліковий запис та налаштувати систему відповідно до своїх потреб та вимог.

Зареєстрований користувач матиме можливість налаштовувати різні параметри системи, такі як температура, освітлення, безпека тощо. Він зможе контролювати та керувати пристроями в будинку через інтерфейс додатку на своєму мобільному пристрої. Крім того, він зможе отримувати статистику та інформацію про стан будинку, таку як споживання електроенергії, рівень вологості, присутність у приміщенні тощо.

Також, система "розумного" будинку може підтримувати кілька облікових записів користувачів, що дозволяє декільком людям мати доступ та керування системою.

Кожен користувач матиме свій власний профіль з індивідуальними налаштуваннями та правами доступу. Це забезпечує гнучкість та зручність використання системи "розумного" будинку для всіх мешканців.

У представленому зображенні (див. Рис. 2.4) ми можемо спостерігати, що нашому користувачеві вдалося успішно пройти реєстрацію. Проте на даному етапі користувач не може увійти до свого облікового запису, оскільки його електронна пошта ще не була підтверджена.

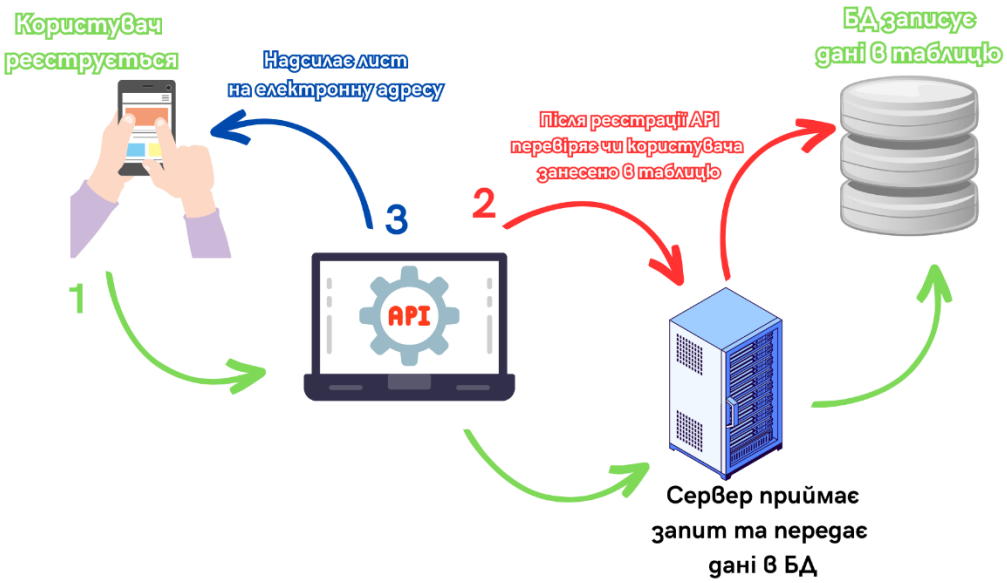



Рисунок 2.3 – Процес реєстрації нового користувача

За замовчуванням, у полі "is_verified" встановлено значення "0". Це означає, що обліковий запис користувача не підтверджений. Допоки це значення не зміниться на "1", користувач буде постійно перенаправлятися на сторінку, на якій йому буде пропоновано підтвердити свою електронну пошту.

Для підтвердження своєї електронної пошти, користувач повинен увійти до своєї поштової скриньки, знайти лист від системи та перейти за посиланням, яке підтвердить його електронну пошту і змінить значення "is_verified" в таблиці на "1". Це підтвердження відіграє важливу роль у забезпеченні безпеки та достовірності облікового запису користувача.

Після підтвердження електронної пошти користувач зможе використовувати свій обліковий запис для входу до системи та отримання повного доступу до функціональності "розумного" будинку. Підтвердження пошти є важливим кроком, який гарантує, що тільки правомірний власник електронної пошти має можливість керувати системою та отримувати повну контроль над своїм "розумним" будинком.

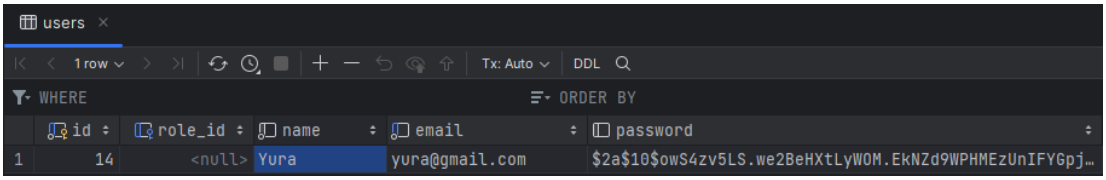



Рисунок 2.4 – Приклад відображення користувача в таблиці

Після успішної реєстрації, користувач потребує авторизації, яку здійснює на сторінці авторизації. Користувач вводить свої дані, і після натискання на кнопку, ці дані відправляються на сервер для перевірки. Сервер перевіряє наявність користувача з вказаною електронною поштою та паролем. У разі відсутності такого користувача, сервер повертає помилку. Далі, перевіряється підтвердження пошти користувача. Якщо електронна пошта не підтверджена, користувач не може увійти в свій обліковий запис, оскільки постійно з'являтиметься вікно з проханням підтвердити пошту. У разі успішної перевірки всіх даних, сервер повертає JWT (токен авторизації) з інформацією про користувача.

Після успішної авторизації, користувач може додавати пристрої на головну сторінку свого облікового запису. Для цього використовується система зі змінним серійним номером. Кожен тип пристрою має свій унікальний номер. Усі можливі типи пристроїв зберігаються в таблиці "devices". Користувачу необхідно сканувати QR-код або вручну ввести серійний номер. Потім відбувається відправка запиту на сервер із серійним номером, який проводить перевірку JWT користувача, що міститься у заголовку запиту. Код перевіряє, чи існує такий пристрій з вказаним ідентифікатором, а також перевіряє, чи пристрій не вже призначений даному користувачу. У разі, якщо пристрій вже призначений даному користувачу, повертається повідомлення "У вас вже є цей пристрій". Якщо всі перевірки пройдено успішно, в таблиці "devices" присвоюється "user_id" користувача, і пристрій додається до його облікового запису.

Які наступні дії, це налаштування центрального пристрою. Перед початком встановлення програмного забезпечення переконайтесь, що у

вас є UNIX-подібна операційна система (наприклад, Linux) на центральному пристрої, на якому ви плануєте розгортати програму. Впевніться, що ви маєте права адміністратора (root) для встановлення пакетів та конфігурації системи.

Завантажте програмне забезпечення, розроблене для центрального пристрою, на ваш комп'ютер або сервер. Ви можете скопіювати його через SSH або використовувати інші доступні методи передачі файлів.

Запустіть файл install.sh для встановлення всіх необхідних компонентів для функціонування програмного забезпечення. Що саме встановиться: PHP та всі необхідні розширення зокрема php-http, для обміну даними мі сервером.

Встановиться веб-сервер Apache, та автоматично додасться програмне забезпечення для як сайт для моніторингу девайсів в локальній мережі за адресою http://sweeme.local. (див. Рис. 2.4-2.5)

Наспуні дії розміщення пристроїв в будинку. Ці пристрої грають роль сервера які отримує команди від центрального пристрою, та повертає дані в залежності від запиту.

В результаті, центральний пристрій знайде кінцеві пристрої по бездротовій мережі WiFi і буде надслати дані через мережу інтернет до головного сервера.

## 2.2 Аналіз та вибір технологій для інформаційної системи

Проводячи аналіз сучасних засобів розробки та технологій для інформаційної системи "розумного" дому, я вибрав наступні компоненти і технології: Laravel, PHP, C++, sqlite, Arduino IDE, ESP32, ESP8266

Laravel - це популярний веб-фреймворк для розробки веб-додатків на мові програмування PHP. Ось переваги та недоліки використання Laravel у порівнянні з іншими веб-фреймворками:

Переваги Laravel:

- Елегантний синтаксис: Laravel пропонує чистий та елегантний синтаксис, що робить розробку зручною і зрозумілою. Він дозволяє швидко та ефективно писати код, що зменшує час розробки.
- Маршрутизація та контролери: Laravel має потужну систему маршрутизації, яка дозволяє легко визначати URL-шляхи та перенаправлення. Крім того, він надає контролери, що спрощує організацію логіки додатку.
- ORM (Eloquent): Laravel має вбудовану ORM-систему під назвою Eloquent, яка дозволяє легко працювати з базою даних. Вона надає зручні методи для виконання запитів до бази даних і спрощує взаємодію з моделями даних.
- Міграції та сівіди: Laravel надає механізми міграцій і сівідів, що дозволяють керувати структурою бази даних і наповнювати її початковими даними. Це допомагає зберігати схему бази даних під контролем версій і спрощує роботу з командою розробників.
- Багата екосистема: Laravel має широку підтримку та активну спільноту розробників, що веде до великої кількості пакетів та розширень, які можна використовувати для розширення функціональності додатків.
- Зручний тестування: Laravel надає інструменти для зручного написання тестів, що спрощує перевірку функціональності вашого додатку та запобігає появі помилок.

Недоліки Laravel:

- Швидкодія: У порівнянні з деякими іншими фреймворками, Laravel може мати трохи меншу швидкодію. Однак, зазвичай ця різниця не є критичною, і вона компенсується зручністю розробки та широкою функціональністю.

- Вимоги до сервера: Laravel вимагає наявності сервера, що підтримує PHP, та веб-сервера, такого як Apache або Nginx. Це може потребувати додаткової конфігурації сервера для належної роботи додатків.
- Крута крива навчання: Для новачків, Laravel може мати круту криву навчання через його розширені можливості і архітектуру. Однак, наявність доброї документації та активної спільноти розробників сприяє подоланню цього недоліку.

Порівняння з іншими фреймворками:

- Laravel vs Symfony: Обидва фреймворки базуються на мові PHP, але Laravel має більш простий синтаксис та більшу кількість вбудованих функцій. Symfony з іншого боку є більш гнучким і масштабованим фреймворком, що підходить для складних проектів.
- Laravel vs CodeIgniter: Laravel має більш розширену функціональність та сучасну архітектуру, порівняно з CodeIgniter. Втім, CodeIgniter є більш легким та швидким фреймворком, який підходить для простіших проектів.
- Laravel vs Django: Laravel і Django (фреймворк Python) мають схожі принципи роботи, але вони використовують різні мови програмування. Обидва фреймворки мають потужну систему маршрутизації, ORM та широкі можливості для розробки веб-додатків.

C++ - це мова програмування загального призначення, яка використовується для розробки різноманітних програм, включаючи вбудовані системи, додатки з графічним інтерфейсом, веб-сервери, наукові обчислення та багато іншого. Ось переваги та недоліки використання C++ у порівнянні з іншими мовами програмування:

Переваги C++:

- Швидкодія: C++ відома своєю високою швидкодією в порівнянні з іншими мовами програмування. Вона наближена до мови машинного коду та дозволяє оптимізувати час виконання програм.
- Висока продуктивність: Завдяки можливості прямого доступу до пам'яті та низькорівневим операціям, C++ дозволяє досягти високої продуктивності та ефективності програм.
- Низькорівневий доступ: C++ надає контроль над апаратними ресурсами комп'ютера та дозволяє безпосередньо взаємодіяти з пам'яттю, пристроями вводу-виводу та іншими системними ресурсами.
- Розширюваність: Можливість роботи з бібліотеками, створеними на C++, дозволяє розширити функціональність програм та використовувати готові рішення для різних завдань.
- Кросплатформеність: C++ є кросплатформеною мовою програмування, що дозволяє писати програми, які працюють на різних операційних системах без змін вихідного коду.

Недоліки C++:

- Складність: C++ є відносно складною мовою програмування, яка вимагає від розробника глибоких знань та розуміння її особливостей. Неправильне використання може призвести до появи помилок та незахищеності програм.
- Велика кількість можливостей: Широкий спектр функцій та можливостей C++ може зробити вибір правильного підходу складним завданням.
- Брак автоматичного управління пам'яттю: C++ не має вбудованої автоматичної системи управління пам'яттю, що може призвести до проблем зі справлянням з пам'яттю та витоками пам'яті.

Порівняння з іншими мовами програмування:

- C++ vs Java: C++ набагато швидший та має більшу контроль над ресурсами, але Java має більш розширену платформу та більш простий синтаксис.
- C++ vs Python: C++ швидший та дозволяє прямий доступ до пам'яті, а Python має простий синтаксис та широкі можливості для розробки веб-додатків.
- C++ vs C#: C++ набагато ближчий до мови машинного коду та дозволяє прямий доступ до пам'яті, а C# має зручнішу роботу з .NET-платформою та більшу підтримку веб-розробки.

SQLite - це вбудовувана реляційна база даних, яка працює без сервера та зберігає дані в одному файлі бази даних. Ось переваги та недоліки використання SQLite у порівнянні з іншими системами управління базами даних (СУБД):

Переваги SQLite:

- Простота використання: SQLite має простий синтаксис та невеликий набір команд, що робить його легким у використанні та навчанні.
- Невеликий розмір: Бібліотека SQLite має невеликий розмір, що дозволяє вбудовувати її в програми з обмеженими ресурсами.
- Відсутність сервера: SQLite працює без окремого сервера, що спрощує налаштування та розгортання бази даних.
- Переносимість: База даних SQLite може бути легко перенесена між різними платформами та операційними системами.
- Швидкість: SQLite відомий своєю високою швидкодією та низькими накладними витратами.

Недоліки SQLite:

- Менша масштабовність: SQLite підходить для невеликих та середніх проектів, але може стати обмеженням для дуже великих баз даних або систем з високим навантаженням.
- Відсутність розподіленості: SQLite не підтримує розподілену архітектуру та не може бути використаний для роботи з розподіленими системами.
- Відсутність деяких функцій: У порівнянні зі великими СУБД, SQLite може мати обмежені можливості, такі як підтримка процедурного мови, розширення мови SQL тощо.

Порівняння з іншими СУБД:

1. SQLite vs MySQL: SQLite підходить для вбудованих систем та простих веб-додатків, тоді як MySQL використовується для веб-розробки та більших систем з багатокористувацьким доступом.

2. SQLite vs PostgreSQL: SQLite надає простоту та швидкість, але PostgreSQL має розширений функціонал та підтримку для високих навантажень.

3. SQLite vs Oracle: SQLite не підтримує деякі розширені функції та масштабовність, що доступні в Oracle, але він легший у використанні та не вимагає додаткових витрат на ліцензію.

Вибір між SQLite та іншими СУБД залежить від конкретних потреб проекту, розміру бази даних, складності запитів та масштабованості системи.

**Висновки до розділу**

В результаті проведеного аналізу та вибору технологій для інформаційної системи "розумного" будинку були визначені оптимальні рішення для реалізації функціональності та взаємодії з пристроями. Програмне забезпечення, написане на Laravel, забезпечує ефективну обробку та зберігання даних, а також зручний доступ до них за допомогою віртуального інтерфейсу та точки доступу. Використання бази даних SQLite

дозволяє забезпечити надійне зберігання інформації та забезпечити швидкий доступ до неї.

За перевагами Laravel варто відзначити його зрозумілий синтаксис, розширені можливості для розробки веб-додатків, високу продуктивність та розширюваність за рахунок великої кількості доступних пакетів та бібліотек. Однак, використання Laravel може вимагати деякого часу та знань для вивчення та оптимального використання фреймворку.

Щодо бази даних SQLite, вона надає простоту використання, легку переносимість та низькі накладні витрати. Однак, у випадку великих проектів або систем з високим навантаженням, можуть виникнути обмеження в масштабовності та розширених функціональних можливостях.

Враховуючи вищезазначені фактори, вибір Laravel та SQLite як основних технологій для програмного забезпечення системи розумного будинку є раціональним, оскільки ці технології забезпечують ефективну реалізацію функціональності, надійне зберігання даних та швидку обробку інформації.

Також варто відзначити, що для реалізації апаратної частини системи розумного будинку були використані плати ESP32 та ESP8266. Ці плати базуються на мікроконтролеріх з архітектурою Xtensa LX6 та мають вбудовані модулі Wi-Fi, що дозволяє їм здійснювати безпроводову комунікацію з центральним пристроєм.

ESP32 та ESP8266 є популярними платформами для Інтернету речей (IoT) та систем розумного будинку. Вони мають низьку споживану потужність, компактний розмір та достатній набір введення-виведення (GPIO) для підключення до різних сенсорів та пристроїв.

Мова програмування C++ була використана для розробки програмного забезпечення на мікроконтролерах ESP32 та ESP8266. C++ є популярною мовою для вбудованих систем, оскільки вона надає низькорівневий доступ до

ресурсів пристрою, ефективну обробку даних та можливість оптимізації роботи програми для обмежених ресурсів мікроконтролера.

Використання плат ESP32 та ESP8266 разом з мовою програмування C++ дозволило створити апаратну частину системи розумного будинку, яка здатна збирати дані про температуру та вологість з кінцевих пристроїв на базі ESP8266 та ESP32 та передавати ці дані до центрального пристрою для подальшої обробки та зберігання.

Всі ці технології разом створюють комплексне рішення для розумного будинку, яке поєднує апаратну та програмну частини, забезпечуючи збір, передачу та зберігання даних про температуру та вологість, а також можливість керування та моніторингу різних пристроїв у будинку.

# РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

## 3.1 Засоби розробки

Розробку інформаційної системи для «розумного» будинку “SweeMe” , а саме програмного забезпечення (інтерфейсу користувача) для центрального пристрою виконано на мові програмування PHP та його фреймворку Laravel

Розробку скетч скрипту для програмування плат ESP32 та ESP8266 використовувалась за допомогою мови програмування C++ та Arduino IDE для компіляції скрипту на плату ESP.

Для функціонування програмного забезпечення для центрального пристрою було обрано наступні бібліотеки :

1. guzzlehttp/guzzle - це PHP-бібліотека, яка надає зручний і простий спосіб виконання HTTP-запитів і взаємодії з веб-серверами. Вона є одним з найпопулярніших клієнтів HTTP для PHP і використовується для створення різних типів веб-додатків, які вимагають взаємодії з зовнішніми API, веб-сервісами або іншими серверами. Основні особливості та можливості бібліотеки guzzlehttp/guzzle включають:

- Виконання HTTP-запитів: guzzlehttp/guzzle дозволяє легко виконувати HTTP-запити, такі як GET, POST, PUT, DELETE та інші. Вона надає простий та зрозумілий API для встановлення різних параметрів запиту, таких як URL, заголовки, параметри, тіло запиту тощо.
- Обробка відповідей сервера: бібліотека guzzlehttp/guzzle автоматично обробляє отримані відповіді від сервера і надає зручний доступ до різних атрибутів відповіді, таких як заголовки, статус, тіло відповіді та інші. Вона також підтримує обробку відповідей у форматі JSON або XML.
- Управління сеансами: guzzlehttp/guzzle дозволяє створювати та керувати сеансами, що дозволяє зберігати стан між кількома HTTP-

запитами. Це може бути корисно, наприклад, для збереження авторизаційних даних або кешування.

- Підтримка асинхронного виконання: бібліотека guzzlehttp/guzzle також підтримує асинхронне виконання HTTP-запитів за допомогою промісів (promises) або обробників подій (event handlers). Це дозволяє виконувати багато запитів паралельно та оптимізувати продуктивність додатка.
- Розширені можливості налаштувань: guzzlehttp/guzzle надає широкі можливості налаштувань, такі як встановлення таймаутів, обмеження кількості перенаправлень, додавання middleware та інші. Це дозволяє забезпечити надійну та ефективну взаємодію з серверами.

1. fruitcake/laravel-cors - це пакет для фреймворка Laravel, який забезпечує підтримку Cross-Origin Resource Sharing (CORS). CORS є механізмом безпеки браузера, який обмежує доступ до ресурсів на веб-сторінках, розміщених на інших доменах. Основні особливості та можливості пакету fruitcake/laravel-cors включають:

    - Налаштування CORS-правил: пакет дозволяє легко налаштовувати правила CORS для вашого Laravel додатку. Ви можете визначити, які домени або джерела можуть зробити запити до вашого додатку, які HTTP-методи дозволені, які заголовки можуть бути включені тощо.
    - Гнучкість управління: ви можете встановлювати правила CORS глобально для всього додатку або налаштовувати їх окремо для конкретних маршрутів або груп маршрутів. Це дозволяє точно керувати поведінкою CORS для різних частин вашого додатку.
    - Обробка префлайт-запитів: пакет fruitcake/laravel-cors автоматично обробляє префлайт-запити, які вимагаються браузером перед виконанням деяких типів запитів, таких як запити з використанням

методів HTTP, відмінних від GET, POST або HEAD, або запити з певними заголовками.

- Захист від CSRF-атак: пакет забезпечує відповідну інтеграцію з CSRF-захистом в Laravel. Він автоматично додає необхідні заголовки CORS до форм, що допомагає уникнути проблем, пов'язаних з взаємодією з формами на інших доменах.
- Підтримка кешування: fruitcake/laravel-cors підтримує кешування CORS-правил, що дозволяє зменшити навантаження на сервер і покращити продуктивність додатку.

Для функціонування програми скетч для плат ESP32 та ESP8266, було обрано наступні бібліотеки :

- WiFiClient.h - це потужна бібліотека, яка надає широкі можливості для взаємодії з TCP-підключеннями через Wi-Fi на мікроконтролерах ESP32 та ESP8266.

  Ця бібліотека розроблена для використання з платформою Arduino або ESP-IDF, що дозволяє зручно і просто працювати з Wi-Fi з'єднаннями на цих мікроконтролерах. За допомогою WiFiClient.h можна налаштовувати та керувати TCP-з'єднаннями з іншими пристроями або серверами через Wi-Fi.

  Вона надає методи для відкриття, закриття, передачі та отримання даних по з'єднанню. Бібліотека також підтримує шифрування з'єднання за допомогою протоколу SSL/TLS, що забезпечує безпеку передачі даних.

  Ця бібліотека є незамінною для розробки проектів "розумного" дому, IoT-пристроїв, мережевих додатків та багатьох інших застосувань, де потрібна безперервна та стабільна комунікація через Wi-Fi з'єднання. Вона спрощує роботу з мережевими протоколами, дозволяє виконувати

з'єднання з віддаленими серверами та передавати дані зручним способом.

- WiFi.h - це потужна бібліотека, яка надає широкий набір функціональностей для зручної роботи з Wi-Fi модулем на мікроконтролерах ESP32 та ESP8266.

  Ця бібліотека спеціально розроблена для використання з платформою Arduino або ESP-IDF, що дозволяє легко та ефективно керувати Wi-Fi з'єднаннями на цих мікроконтролерах. Бібліотека WiFi.h надає різноманітні функції для керування Wi-Fi модулем, такі як підключення до мережі Wi-Fi, розташування доступних мереж, отримання та налаштування IP-адреси, налаштування режиму роботи точки доступу та багато інших. Вона також підтримує різні захищені протоколи, включаючи WPA і WPA2, що забезпечує безпеку підключення до Wi-Fi мережі. Ця бібліотека є незамінною для розробки проектів "розумного" дому, IoT-пристроїв, мережевих додатків та інших застосувань, де потрібне бездротове з'єднання з мережею Wi-Fi. Вона спрощує налаштування та управління Wi-Fi модулем, дозволяючи швидко і зручно забезпечити з'єднання з доступною мережею, передавати та отримувати дані.

- WebServer.h - це потужна бібліотека, яка надає широкий спектр можливостей для створення веб-сервера на мікроконтролерах ESP32 та ESP8266. Ця бібліотека спеціально розроблена для використання з платформою Arduino або ESP-IDF, що дозволяє легко і ефективно реалізувати функціональний веб-сервер на цих мікроконтролерах.

  Бібліотека WebServer.h надає різноманітні можливості для обробки HTTP-запитів, створення маршрутів, передачі даних клієнтам та багато іншого. Вона підтримує роботу з різними типами HTTP-методів, такими як GET, POST, PUT і DELETE, дозволяючи ефективно

обробляти запити від клієнтів. Бібліотека також підтримує використання шаблонів HTML для створення динамічних веб-сторінок.

Ця бібліотека є незамінною для розробки проектів, де потрібно створити веб-інтерфейс для керування пристроями або відображення даних. За допомогою WebServer.h ви можете легко створити власний веб-сервер, який забезпечить взаємодію з клієнтами через протокол HTTP.

- ESPmDNS.h - це потужна бібліотека, яка надає широкі можливості для реалізації Multicast DNS (mDNS) на мікроконтролерах ESP32 та ESP8266. Ця бібліотека спеціально розроблена для використання з платформою Arduino або ESP-IDF, що дозволяє легко і ефективно впровадити механізм mDNS на цих мікроконтролерах.

  Бібліотека ESPmDNS.h надає можливість реалізувати Multicast

  DNS - протокол, який дозволяє пристроям в мережі автоматично виявляти один одного за допомогою доменних імен замість IP-адрес. Завдяки цьому, ви можете легко звертатися до пристроїв в вашій мережі за допомогою зрозумілих доменних імен, що спрощує комунікацію та взаємодію з ними.

  Ця бібліотека є незамінною для розробки проектів, де потрібно забезпечити просту і зручну ідентифікацію та взаємодію з пристроями в мережі. За допомогою ESPmDNS.h ви можете легко налаштувати доменне ім'я для вашого мікроконтролера та зробити його доступним для інших пристроїв в мережі.

- DHT.h - це потужна бібліотека, яка надає розширені можливості для роботи з датчиками вологості та температури, зокрема моделями DHT11, DHT21 (AM2301) і DHT22 (AM2302), на мікроконтролерах ESP32 та ESP8266. Ця бібліотека спеціально розроблена для

використання з платформою Arduino або ESP-IDF, що дозволяє легко і ефективно взаємодіяти з цими датчиками на цих мікроконтролерах.

Бібліотека DHT.h надає зручний і простий спосіб отримання даних про вологість та температуру з датчиків DHT11, DHT21 і DHT22. Вона дозволяє зчитувати значення цих параметрів з датчиків і передавати їх для подальшої обробки. Бібліотека підтримує різні режими роботи, включаючи зчитування значень в реальному часі або відповідно до заданого інтервалу часу.

### 3.2 Опис реалізації системи

За допомогою Figma було створено дизайн інтерфейсу та візуалізації даних інформації що зберігається на центрально пристрої (рис. 3.1).

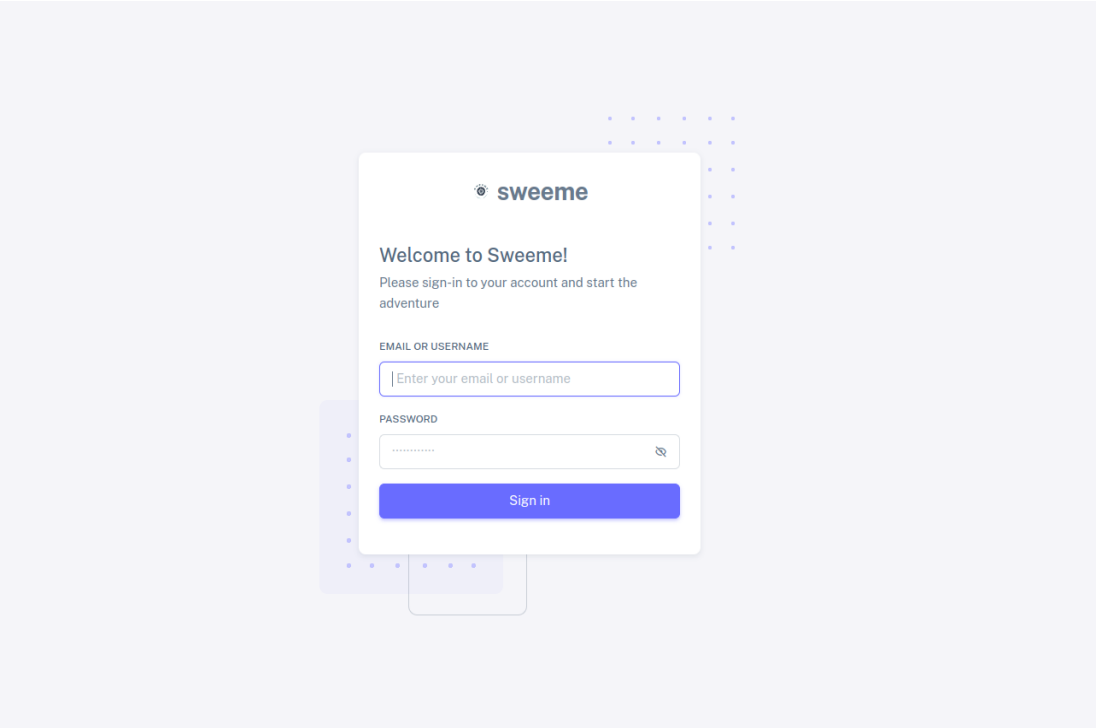



Рисунок 3.1 – Вікно авторизації

На рисунку 3.1 ми бачимо велике, привабливо відображене вікно авторизації, яке служить як додатковий захист для керування пристроями. Це вікно авторизації впроваджено з метою забезпечення безпеки та захисту пристроїв від несанкціонованого доступу. Воно є важливою складовою

частиною системи та дозволяє впевнитися, що лише уповноважені користувачі можуть мати доступ до управління пристроями. На цьому зображенні вікно авторизації відображено в своєму повному розмірі, з усіма необхідними полями та кнопками для введення вірних облікових даних і отримання доступу до системи. Таке вікно є важливим кроком у забезпеченні безпеки і відображає зростаючу увагу до захисту персональних даних та конфіденційності.

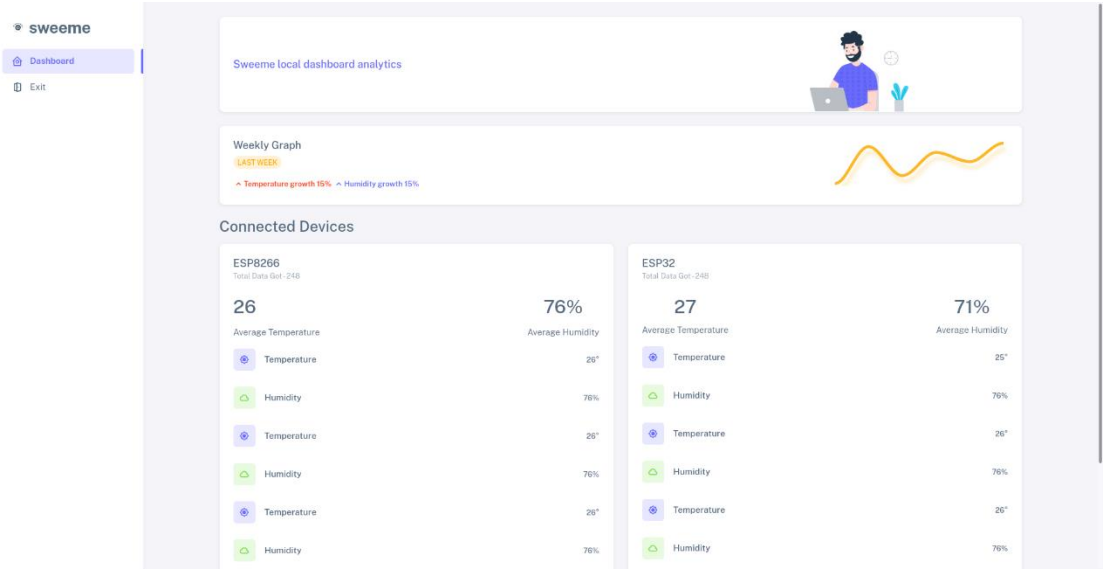



Рисунок 3.2 – Вікно моніторингу

На рисунку 3.2 ми спостерігаємо вражаючу апаратну платформу головного вікна моніторингу та управління пристроями. Це вікно є ключовим інструментом для забезпечення ефективного контролю та керування різними пристроями у системі. Його великий розмір та яскравий дизайн сприяють зручності використання та забезпечують чітке відображення важливої інформації.

У цьому вікні моніторингу та управління пристроями зібрано широкий спектр функціональності. На головному екрані можна спостерігати за статусом різних пристроїв, включаючи їхню доступність, стан підключення та активність. Крім того, вікно надає можливість керувати пристроями, здійснювати різні операції, такі як включення, вимкнення, налаштування параметрів та відстеження ресурсів.

Вікно моніторингу та управління пристроями відображено в повному розмірі, щоб користувачі могли насолодитися відмінною читабельністю та зручністю використання. Його інтуїтивний інтерфейс сприяє швидкому навігуванню та швидкому доступу до необхідної інформації, що дозволяє оперативно реагувати на будь-які зміни або проблеми з пристроями. Загалом, головне вікно моніторингу та управління пристроями відображає передові можливості технології та сприяє впровадженню ефективної та безпечної системи керування.

Перейдемо до розгляду функціональності програмного забезпечення для центрального пристрою. Це не просто красивий інтерфейс, а найголовніший елемент в системі розумного будинку. Можна сказати, що він є мозком будинку, оскільки відповідає за з’єднання з сервером та збір інформації з кінцевих пристроїв.

Програмне забезпечення центрального пристрою виконує важливі функції. По-перше, воно забезпечує зв'язок з сервером, що дозволяє обмінюватися даними між будинком та центральним сервером. Це з'єднання дозволяє здійснювати дистанційний доступ до системи з будь-якого місця, де є Інтернет.

По-друге, центральний пристрій здійснює збір інформації з кінцевих пристроїв, таких як датчики руху, термостати, освітлення і багато інших. Він отримує дані про стан цих пристроїв, їхню активність, температуру, енергоспоживання та багато іншого. Зібрана інформація може використовуватися для аналізу, прийняття рішень та автоматизації різних процесів у будинку.

Центральний пристрій також відповідає за координацію роботи різних пристроїв у системі розумного будинку. Він може надсилати команди для управління освітленням, опаленням, системою безпеки та іншими пристроями згідно з заданими сценаріями або на основі отриманої інформації від датчиків.

Таким чином, програмне забезпечення для центрального пристрою є невід'ємною частиною системи розумного будинку. Воно з

абезпечує з’єднання з сервером, збір та обробку даних з кінцевих пристроїв, а також координацію їхньої роботи. Ця функціональність робить програмне забезпечення центрального пристрою незамінним елементом для забезпечення ефективної та інтелектуальної роботи системи розумного будинку.

### 3.3 Аналіз отриманих результаті

Розглянемо детальніше процес стягування центральним пристроєм інформації з кінцевих пристроїв. При встановленні програмного забезпечення на центральному пристрої створюється точка доступу, яка дозволяє з'єднуватись з іншими пристроями.

Після створення точки доступу програмне забезпечення починає процес аналізу всіх можливих пристроїв, що можуть бути підключені до цієї точки доступу. Воно сканує навколишні мережі і пристрої, шукаючи доступні підключення.

Коли кінцевий пристрій виявлений, його інформація (така як IP-адреса, інформація про пристрій) збирається і додається до локальної бази даних на центральному пристрої. Це дозволяє центральному пристрою відстежувати та управляти підключеними пристроями.

Додавання кінцевих пристроїв до локальної бази даних має кілька корисних переваг. Воно дозволяє збирати статистику про підключені пристрої, контролювати доступ до мережі, встановлювати правила та обмеження для окремих пристроїв, а також полегшує процес ідентифікації та взаємодії з підключеними пристроями.

Загалом, процес стягування центральним пристроєм інформації з кінцевих пристроїв є важливим етапом управління мережею, який дозволяє забезпечити безпеку, контроль та ефективне використання ресурсів.

Ось приклад збереження пристроїв в БД (див. Рис. 3.3).

| id | ip_address | mpn | created_at | updated_at |
|---|---|---|---|---|
| 1 | 192.168.0.117 | ESP321249640124 | 1687099086000 | 2023-06-18 14:38:06.000 |
| 2 | 192.168.0.118 | ESP82661249640124 | 2023-06-18 14:38:06.000 | 2023-06-18 14:38:06.000 |

Рисунок 3.3 — Дані про пристрої

Розглянемо процесс виміру температури. Наступний фрагмент коду вимірює температуру з датчику.

```
void handleTemperature() {
  float temperature = dht.readTemperature();
  server.send(200, "text/plain", String(temperature));
}
```

Рисунок 3.4 — Вимірювання температури

Процесс виміру вологості виглядає схожим чинос. Наступний фрагмент коду вимірює вологість з датчику.

```
void handleHumidity() {
  float humidity = dht.readHumidity();
  server.send(200, "text/plain", String(humidity));
}
```

Рисунок 3.5 — Вимірювання вологості

Всі ці отриманні дані надсилаються у відповідь на запит центрального пристрою.

Розглянемо їх збереження.

Окремо кожен вимір величин зберігаються в БД в табличках temperatures та humitidies. Ось приклад отриманих та збережених даних на центральному пристрої.

| | id | value | device_id | created_at | updated_at |
|---|---|---|---|---|---|
| 1 | 1 | 27 | 1 | 2023-06-19 22:19:06 | 2023-06-19 22:19:06 |
| 2 | 2 | 27 | 1 | 2023-06-19 22:19:07 | 2023-06-19 22:19:07 |
| 3 | 3 | 26 | 1 | 2023-06-19 22:19:08 | 2023-06-19 22:19:08 |
| 4 | 4 | 25 | 1 | 2023-06-19 22:19:09 | 2023-06-19 22:19:09 |
| 5 | 5 | 26 | 1 | 2023-06-19 22:19:10 | 2023-06-19 22:19:10 |
| 6 | 6 | 24 | 1 | 2023-06-19 22:19:11 | 2023-06-19 22:19:11 |
| 7 | 7 | 25 | 1 | 2023-06-19 22:19:12 | 2023-06-19 22:19:12 |
| 8 | 8 | 26 | 1 | 2023-06-19 22:19:13 | 2023-06-19 22:19:13 |
| 9 | 9 | 26 | 1 | 2023-06-19 22:19:14 | 2023-06-19 22:19:14 |
| 10 | 10 | 27 | 1 | 2023-06-19 22:19:15 | 2023-06-19 22:19:15 |
| 11 | 11 | 26 | 1 | 2023-06-19 22:19:31 | 2023-06-19 22:19:31 |
| 12 | 12 | 26 | 1 | 2023-06-19 22:19:32 | 2023-06-19 22:19:32 |
| 13 | 13 | 27 | 1 | 2023-06-19 22:19:33 | 2023-06-19 22:19:33 |
| 14 | 14 | 24 | 1 | 2023-06-19 22:19:34 | 2023-06-19 22:19:34 |
| 15 | 15 | 24 | 1 | 2023-06-19 22:19:35 | 2023-06-19 22:19:35 |
| 16 | 16 | 24 | 1 | 2023-06-19 22:19:36 | 2023-06-19 22:19:36 |
| 17 | 17 | 25 | 1 | 2023-06-19 22:19:37 | 2023-06-19 22:19:37 |
| 18 | 18 | 26 | 1 | 2023-06-19 22:19:38 | 2023-06-19 22:19:38 |
| 19 | 19 | 24 | 1 | 2023-06-19 22:19:39 | 2023-06-19 22:19:39 |
| 20 | 20 | 24 | 1 | 2023-06-19 22:19:40 | 2023-06-19 22:19:40 |
| 21 | 21 | 26 | 1 | 2023-06-19 22:19:42 | 2023-06-19 22:19:42 |

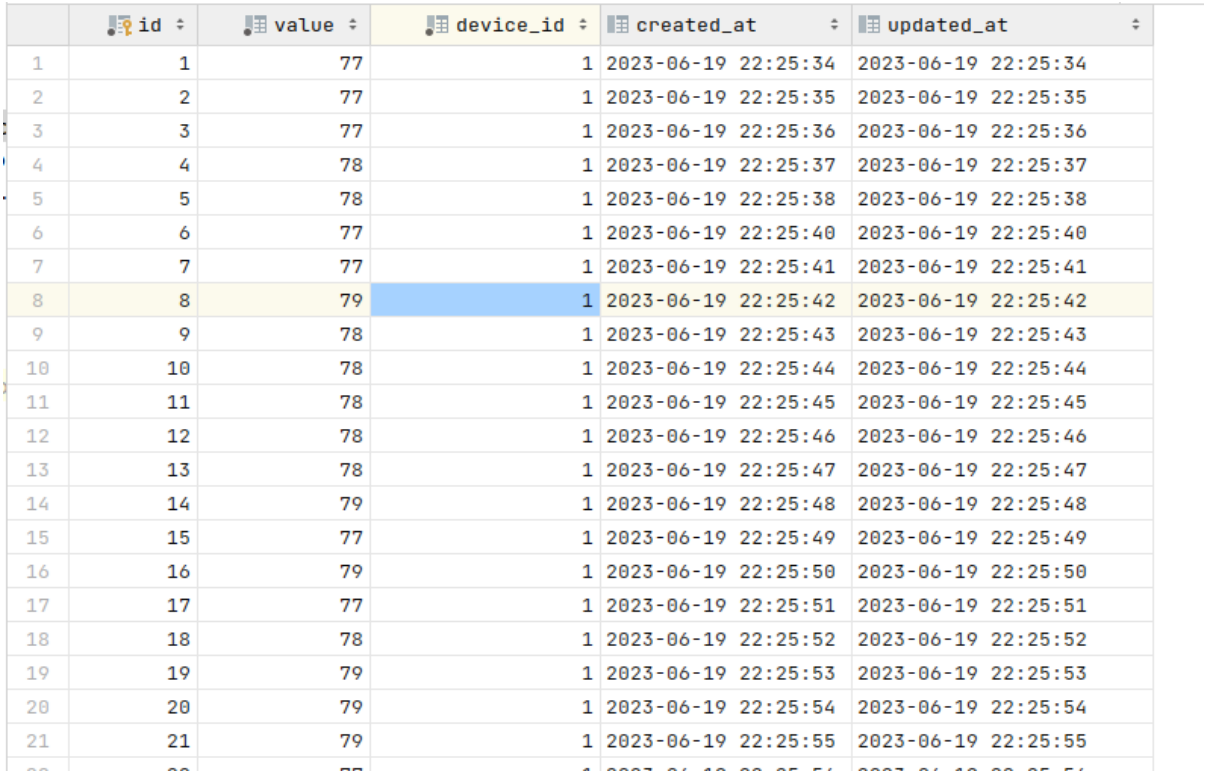
Рисунок 3.6 — Результата виміру температури

Окремо кожен вимір величин зберігаються в базі даних (БД) у спеціальних табличках "temperatures" та "humitidies". Ці таблиці містять докладну інформацію про виміри температури та вологості збережених даних. На центральному пристрої зберігається широкий набір вимірів з різних джерел і датчиків.

Наприклад, табличка "temperatures" містить колонки зі значеннями температури, відповідними часовим міткам та ідентифікаторами джерел даних. Таким чином, ми можемо відслідковувати зміни температури протягом певного періоду часу та встановлювати залежності між різними джерелами даних.

Табличка "humitidies" має аналогічну структуру, але зберігає дані про вологість. Це дозволяє нам аналізувати та порівнювати дані про вологість з різних джерел та досліджувати її вплив на інші фактори.

На наступному рисунку зображено результати виміру вологості.



| | id | value | device_id | created_at | updated_at |
|---|---|---|---|---|---|
| 1 | 1 | 77 | 1 | 2023-06-19 22:25:34 | 2023-06-19 22:25:34 |
| 2 | 2 | 77 | 1 | 2023-06-19 22:25:35 | 2023-06-19 22:25:35 |
| 3 | 3 | 77 | 1 | 2023-06-19 22:25:36 | 2023-06-19 22:25:36 |
| 4 | 4 | 78 | 1 | 2023-06-19 22:25:37 | 2023-06-19 22:25:37 |
| 5 | 5 | 78 | 1 | 2023-06-19 22:25:38 | 2023-06-19 22:25:38 |
| 6 | 6 | 77 | 1 | 2023-06-19 22:25:40 | 2023-06-19 22:25:40 |
| 7 | 7 | 77 | 1 | 2023-06-19 22:25:41 | 2023-06-19 22:25:41 |
| 8 | 8 | 79 | 1 | 2023-06-19 22:25:42 | 2023-06-19 22:25:42 |
| 9 | 9 | 78 | 1 | 2023-06-19 22:25:43 | 2023-06-19 22:25:43 |
| 10 | 10 | 78 | 1 | 2023-06-19 22:25:44 | 2023-06-19 22:25:44 |
| 11 | 11 | 78 | 1 | 2023-06-19 22:25:45 | 2023-06-19 22:25:45 |
| 12 | 12 | 78 | 1 | 2023-06-19 22:25:46 | 2023-06-19 22:25:46 |
| 13 | 13 | 78 | 1 | 2023-06-19 22:25:47 | 2023-06-19 22:25:47 |
| 14 | 14 | 79 | 1 | 2023-06-19 22:25:48 | 2023-06-19 22:25:48 |
| 15 | 15 | 77 | 1 | 2023-06-19 22:25:49 | 2023-06-19 22:25:49 |
| 16 | 16 | 79 | 1 | 2023-06-19 22:25:50 | 2023-06-19 22:25:50 |
| 17 | 17 | 77 | 1 | 2023-06-19 22:25:51 | 2023-06-19 22:25:51 |
| 18 | 18 | 78 | 1 | 2023-06-19 22:25:52 | 2023-06-19 22:25:52 |
| 19 | 19 | 79 | 1 | 2023-06-19 22:25:53 | 2023-06-19 22:25:53 |
| 20 | 20 | 79 | 1 | 2023-06-19 22:25:54 | 2023-06-19 22:25:54 |
| 21 | 21 | 79 | 1 | 2023-06-19 22:25:55 | 2023-06-19 22:25:55 |

Рисунок 3.7 — Результата виміру вологості

Ці результати можна побачити на веб інтерфейсі в локальній мережі за одресою http://sweeme.local. На наступному рисунку зображено отримані результати на веб-інтерфейсі.

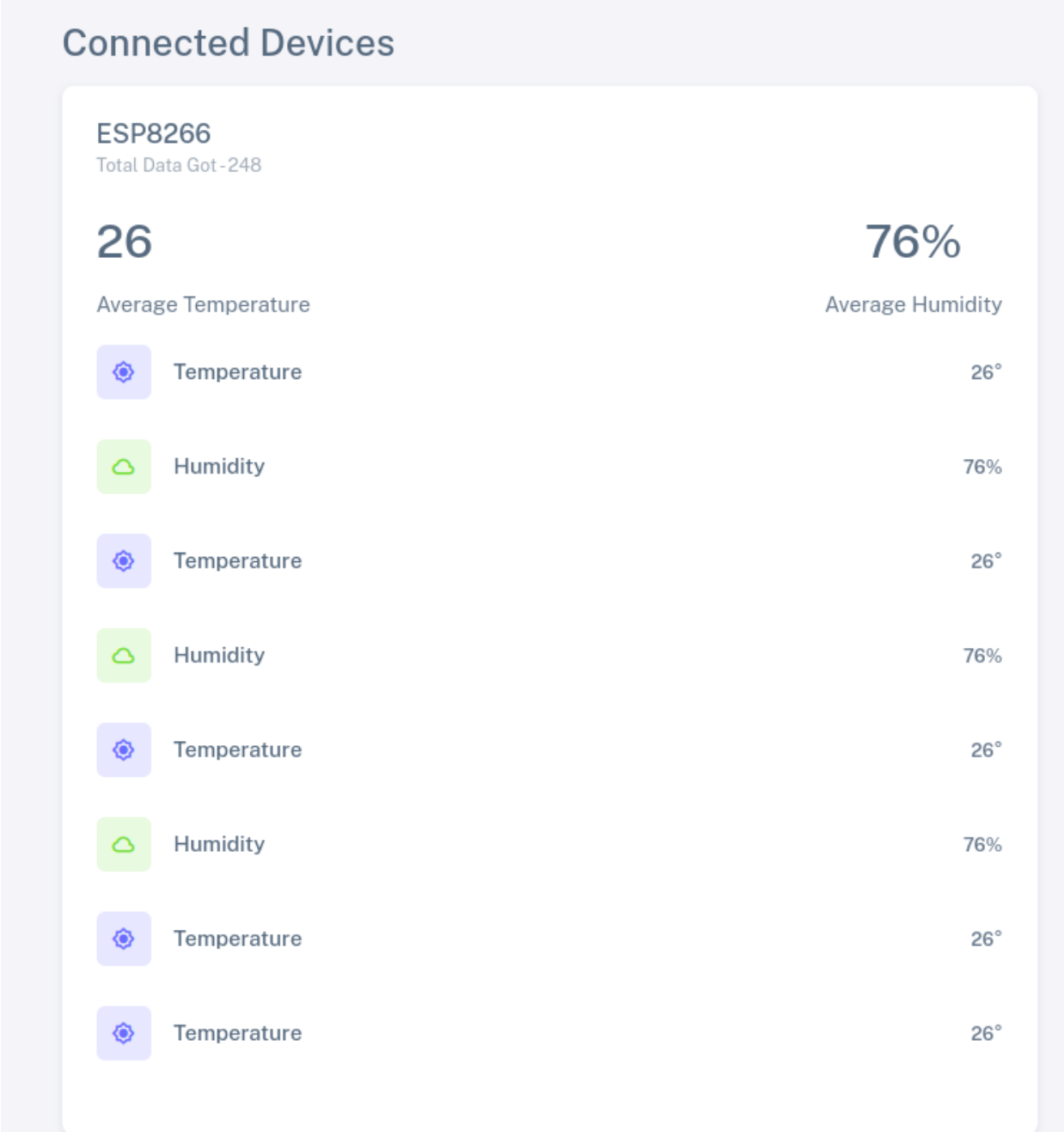



Рисунок 3.8 — Результати на веб-інтерфейсі

Розглянемо код програми центрального пристрою який стягує дані з кіневих пристроїв (див. Рис. 3.9).

```php
/**
 * Execute the console command.
 *
 * @param DeviceRepository $deviceRepository
 * @param DeviceService $service
 * @param NetworkService $networkService
 * @return void
 */
public function handle(
    DeviceRepository $deviceRepository,
    DeviceService $service,
    NetworkService $networkService
): void
{
    if (!$networkService->hasConnection()) {
        sleep( seconds: 10);
    }
    $devices = $deviceRepository->all();

    $devices->map(function (Device $device) use ($service) {
        try {
            $service->trigger($device);
        } catch (Throwable $throwable){

        }
    });

    sleep( seconds: 1);
}
```

Рисунок 3.9 — Команда отримання даних з пристроїв

Команда отримує всі збережені пристрої та звертається до них як до звичайного серверу, та запитом отримує необхідні данні. Після отримання даних він додає в чергу відправку даних на головний сервер для додавання до загальної бази даних.

```php
class HumidityService
{

    const URL = '/humidity/add';

    no usages
    public function __construct(
        private Client $client
    )
    {
    }

    /**
     * @throws GuzzleException
     */
    1 usage
    public function sendHumidity(Humidity $humidity): void
    {
        $data = json_encode([
            'mpn' => $humidity->getDevice()->getMpn(),
            'value' => $humidity->getValue()
        ]);

        $request = new Request(
            method: RequestAlias::METHOD_POST,
            uri: config( key: 'api.url') . self::URL,
            body: $data,
        );

        $this->client->send($request);
    }
}
```

Рисунок 3.10 — Сервіс для віправки вологості на сервер

Для відправки вологості додається MPN (Manufacture Part Number) пристрою, та значення показників.

Наступний рисунок описує відправку температури на сервер.

```php
class TemperatureService
{

    const URL = '/temperature/add';

    no usages
    public function __construct(
        private Client $client
    )
    {
    }

    /**
     * @throws GuzzleException
     */
    1 usage
    public function sendTemperature(Temperature $temperature): void
    {
        $data = json_encode([
            'mpn' => $temperature->getDevice()->getMpn(),
            'value' => $temperature->getValue()
        ]);

        $request = new Request(
            method: RequestAlias::METHOD_POST,
            uri: config( key: 'api.url') . self::URL,
            body: $data,
        );

        $this->client->send($request);
    }
}
```

Рисунок 3.11 — Сервіс для віправки температури на сервер

Аналогічно сервісу для відправки волгості для відправки  додається MPN (Manufacture Part Number) пристрою, та значення показників.

**Висновки до розділу 3**

У цьому розділі було розглянуто і обрано засоби розробки, описані їх переваги і аргументи, щодо доцільності їх використання. Була надана

інформація про реалізацію інформаційної системи та її взаємодію з інтерфейсом користувача.

Останньою частиною розділу була продемонстрована робота інформаційної системи, де були показані основні можливості та способи взаємодії з користувачем. Робота інформаційної системи та її інтерфейсу здійснюється без помилок і здатна взаємодіяти з API.

Система виявляється досить гнучкою та має великий потенціал для внесення змін і покращень.

# ВИСНОВКИ

У процесі проектування та розробки інформаційної системи та інтерфейсу користувача для «розумного» будинку "SweeMe" була досягнута мета проекту, а також успішно виконані всі поставлені завдання.

В ході аналізу предметної області були вивчені сучасні технології та рішення у сфері "розумних" будинків, а також визначені основні потреби і вимоги користувачів. На основі цього аналізу було здійснено проектування інформаційної системи "SweeMe" та розробку інтерфейсу користувача для взаємодії з "розумним" будинком.

Процес розробки надав змогу отримати досвід і навички, які будуть корисні для подальшої роботи з подібними інформаційними системами, а також спростять розвиток вже існуючої системи.

Розробка даної системи виявилася актуальною, оскільки аналіз ринку та подібних систем показав, що створення власної системи дозволить зекономити кошти, а також має потенціал для подальшого розвитку і вдосконалення. Дана система може конкурувати з іншими, оскільки вона передбачає розробку та впровадження пристроїв, які можуть бути налаштовані під потреби конкретного клієнта. Крім того, ця система може бути використана не тільки в приватних будинках та квартирах, але й у підприємствах та різних бізнес-середовищах.

Застосування даної інформаційної системи дозволить зекономити електроенергію, зручно керувати пристроями, розробляти власні системи та легко їх інтегрувати.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Laravel Documentation - https://laravel.com/docs/10.x/readme
2. ESP-IDF Programming Guide - https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
3. ESP8266 Arduino Core’s documentation - https://arduino-esp8266.readthedocs.io/en/latest/
4. Arduino Documentation - https://docs.arduino.cc/
5. PHP Documentation - https://www.php.net/docs.php

# ДОДАТКИ

## Додаток А. Лістинги програми

### A.1 Migrations

2014_10_12_000000_create_users_table.php

```
 return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
};
```

2014_10_12_100000_create_password_resets_table.php

```
return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
```

```
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
};
```

2019_08_19_000000_create_failed_jobs_table.php

```
return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
};
```

2019_12_14_000001_create_personal_access_tokens_table.php

```
return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamps();
        });
```

```
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('personal_access_tokens');
    }
};
```

2023_06_15_210747_create_sessions_table.php

```
return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('sessions');
    }
};
```

2023_06_16_134909_create_devices_table.php

```
return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('devices', function (Blueprint $table) {
            $table->id();
            $table->string('ip_address');
            $table->string('mpn');
            $table->timestamps();
```

```
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('devices');
    }
};
```

2023_06_17_130234_create_temperatures_table.php

```
return new class extends Migration {
  /**
   * Run the migrations.
   *
   * @return void
   */
  public function up(): void
  {
    Schema::create('temperatures', function (Blueprint $table) {
      $table->id();
      $table->float('value');
      $table->foreignIdFor(Device::class);
      $table->timestamps();
    });
  }

  /**
   * Reverse the migrations.
   *
   * @return void
   */
  public function down(): void
  {
    Schema::dropIfExists('temperatures');
  }
};
```

2023_06_17_131037_create_humidities_table.php

```
return new class extends Migration {
  /**
   * Run the migrations.
   *
   * @return void
   */
  public function up()
  {
    Schema::create('humidities', function (Blueprint $table) {
      $table->id();
      $table->float('value');
      $table->foreignIdFor(Device::class);
      $table->timestamps();
    });
  }
```

```
 /**
  * Reverse the migrations.
  *
  * @return void
  */
 public function down()
 {
  Schema::dropIfExists('humidities');
 }
};
```

## A.2 Services

NetworkService.php

```
class NetworkService
{
 public function getLocalIP(): string
 {
  exec(Commands::IFCONFIG, $output);
  return $this->parseLocalIP($output);
 }
 private function parseLocalIP($output): string
 {
  $arr = [];
  foreach ($output as $line) {
   if (preg_match('/inet\s+(\d+\.\d+\.\d+\.\d+)/', $line, $matches)) {
    $arr[] = $matches[1];
   }
  }
  return $arr[sizeof($arr) - 1] ?? 'Unknown';
 }

 public function hasConnection(): bool
 {
  $connected = @fsockopen("www.google.com", 80);
  if ($connected) {
   $is_conn = true;
   fclose($connected);
  } else {
   $is_conn = false;
  }
  return $is_conn;
 }
 public function getWifiPassword(string $ssid): string
 {
  $output = '';
  exec('nmcli connection show ' . $ssid . ' -s | grep psk', $output);
  $matches = [];
  foreach ($output as $item) {
   if (str_contains($item, '802-11-wireless-security.psk:')) {
    $pattern = '/802-11-wireless-security.psk:\s+(.*)/';
    preg_match($pattern, $item, $matches);
    break;
   }
  }
  return trim($matches[1]) ?? '';
 }
 public function scan(): Collection
 {
  exec(Commands::SCAN, $output);
  $ssids = [];
```

```
    foreach ($output as $line) {
      $fields = explode(':', $line);
      if (count($fields) === 2) {
        $ssids[] = [
          'connected' => $fields[0],
          'ssid' => $fields[1]
        ];
      }
    }
    $collection = new Collection();
    foreach ($ssids as $ssid) {
      $collection->add([
        'ssid' => $ssid['ssid'],
        'connected' => $ssid['connected'],
        'password' => $this->getWifiPassword($ssid['ssid'])
      ]);
    }
    return $collection;
  }
  public function connectWifi(string $ssid, string $password): bool
  {
    $command = "nmcli dev wifi connect '$ssid' password '$password'";
    exec($command, $output, $returnCode);
    return $returnCode === 0;
  }
  public function makeHostSpot(): bool
  {
    exec(Commands::HOTSPOT, $output, $returnCode);
    return $returnCode === 0;
  }
}
```

DeviceService.php

```
class DeviceService
{

  public function __construct(
    private Client $client,
    private NetworkService $networkService
  )
  {
  }

  /**
   * @param Device $device
   * @throws GuzzleException
   * @throws Exception
   */
  public function trigger(Device $device): void
  {
    if (!$this->networkService->hasConnection())
      throw new Exception('No connection');

    $data = json_encode([
      'ip' => $this->networkService->getLocalIP(),
    ]);

    $request = new Request(
      method: RequestAlias::METHOD_POST,
      uri: 'http://' . $device->getIpAddress(),
```

```
      body: $data
    );

    $this->client->send($request);
  }
}
```

HumidityService.php

```
class HumidityService
{

  const URL = '/humidity/add';

  public function __construct(
    private Client $client
  )
  {
  }

  /**
   * @throws GuzzleException
   */
  public function sendHumidity(Humidity $humidity): void
  {
    $data = json_encode([
      'mpn' => $humidity->getDevice()->getMpn(),
      'value' => $humidity->getValue()
    ]);

    $request = new Request(
      method: RequestAlias::METHOD_POST,
      uri: config('api.url') . self::URL,
      body: $data,
    );

    $this->client->send($request);
  }
}
```

TemperatureService.php

```
class TemperatureService
{

  const URL = '/temperature/add';

  public function __construct(
    private Client $client
  )
  {
  }

  /**
   * @throws GuzzleException
   */
  public function sendTemperature(Temperature $temperature): void
  {
    $data = json_encode([
      'mpn' => $temperature->getDevice()->getMpn(),
```

```
      'value' => $temperature->getValue()
    ]);

    $request = new Request(
      method: RequestAlias::METHOD_POST,
      uri: config('api.url') . self::URL,
      body: $data,
    );

    $this->client->send($request);
  }
}
```

## A.3 Repositories

DeviceRepository.php

```
class DeviceRepository
{

  public function existsByMpn(string $mpn): bool
  {
    return Device::query()
      ->where('mpn', '=', $mpn)
      ->exists();
  }

  public function getByMpn(string $mpn): Device
  {
    return Device::query()
      ->where('mpn','=', $mpn)
      ->first();
  }

  public function create(string $mpn, string $ip): Device
  {
    return Device::query()
      ->create([
        'mpn' => $mpn,
        'ip_address' => $ip
      ]);
  }

  public function all(): Collection
  {
    return Device::all();
  }
}
```

UserRepository.php

```
class UserRepository
{

  public function exists(): bool
  {
    return User::query()->exists();
```

```
  }

  public function getOne(string $email): User
  {
    return User::query()
      ->where('email', '=', $email)
      ->first();
  }
}
```

## A.4 Controllers

### AuthController.php

```
class AuthController extends ApiController
{
  /**
   * @throws Exception
   */
  public function login(AuthRequest $authRequest)
  {
    if (!$this->userRepository->exists())
      Artisan::call(Commands::DB_SEED);

    $user = $this->userRepository->getOne($authRequest->get('email'));

    if (!Hash::check($authRequest->get('password'), $user->getPassword()))
      throw new Exception('Wrong credentials');

    if (Auth::attempt($authRequest->validated()))
      return new JsonResponse(['message' => 'Successfully logged in!']);

    return new JsonResponse(['message' => 'Unauthorized!', 422]);
  }
}
```

### DeviceController.php

```
class DeviceController extends ApiController
{

  public function __construct(
    private DeviceRepository $deviceRepository
  )
  {
  }

  public function devices()
  {
    return new DevicesResources($this->deviceRepository->all());
  }
}
```

### SensorsController.php

```
class SensorsController
{

  public function __construct(
    private DeviceRepository $deviceRepository
  )
  {
  }

  public function temperature(TemperatureLogRequest $request)
  {
    $ip = $request->get('ip');
    $mpn = $request->get('mpn');
    $value = $request->get('value');

    if ($this->deviceRepository->existsByMpn($mpn))
      $device = $this->deviceRepository->getByMpn($mpn);
    else
      $device = $this->deviceRepository->create($mpn, $ip);

    $temperature = Temperature::query()->create([
      'device_id' => $device->getId(),
      'value' => $value,
    ]);

    dispatch(new SendTemperatureToApi($temperature))->onQueue(Queues::DEFAULT);
    return new JsonResponse(['message' => 'Accepted.'], 200);
  }

  public function humidity(HumidityLogRequest $request)
  {
    $ip = $request->get('ip');
    $mpn = $request->get('mpn');
    $value = $request->get('value');

    if ($this->deviceRepository->existsByMpn($mpn))
      $device = $this->deviceRepository->getByMpn($mpn);
    else
      $device = $this->deviceRepository->create($mpn, $ip);

    $humidity = Humidity::query()->create([
      'device_id' => $device->getId(),
      'value' => $value,
    ]);

    dispatch(new SendHumidityToApi($humidity))->onQueue(Queues::DEFAULT);
    return new JsonResponse(['message' => 'Accepted.'], 200);
  }
}
```

## A.5 Commands

TemperatureHumidityHandle.php

```
class TemperatureHumidityHandle extends Command
{
    /**
     * The name and signature of the console command.
     *
```

```
     * @var string
     */
    protected $signature = 'app:handle';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Command description';

   /**
    * Execute the console command.
    *
    * @param DeviceRepository $deviceRepository
    * @param DeviceService $service
    * @param NetworkService $networkService
    * @return void
    */
    public function handle(
      DeviceRepository $deviceRepository,
      DeviceService $service,
      NetworkService $networkService
    ): void
    {
      if (!$networkService->hasConnection()) {
        sleep(10);
      }
      $devices = $deviceRepository->all();

      $devices->map(function (Device $device) use ($service) {
        try {
          $service->trigger($device);
        } catch (Throwable $throwable){

        }
      });

      sleep(1);
    }
}
```

## A.6 Jobs

SendHumidityToApi.php

```
class SendHumidityToApi implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    /**
     * Create a new job instance.
     *
     * @return void
     */
    public function __construct(
      private Humidity $humidity
    )
    {
    }
```

```
  /**
   * Execute the job.
   *
   * @param HumidityService $service
   * @return void
   * @throws GuzzleException
   */
  public function handle(
    HumidityService $service
  ): void
  {
    $service->sendHumidity($this->humidity);
  }
}
```

SendTemperatureToApi.php

```
class SendTemperatureToApi implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    /**
     * Create a new job instance.
     *
     * @return void
     */
    public function __construct(
      private Temperature $temperature
    )
    {
    }

  /**
   * Execute the job.
   *
   * @param TemperatureService $service
   * @return void
   * @throws GuzzleException
   */
    public function handle(
      TemperatureService $service
    ): void
    {
      $service->sendTemperature($this->temperature);
    }
}
```

## A.7 Models

Device.php

```
class Device extends Model
{

  use WithId;

  protected $fillable = [
    'ip_address',
    'mpn'
  ];
```

```
  public function getMpn(): string
  {
    return $this->getAttribute('mpn');
  }

  public function getIpAddress(): string
  {
    return $this->getAttribute('ip_address');
  }

  public function temperatures(): HasMany
  {
    return $this->hasMany(Temperature::class);
  }

  public function getTemperature(int $limit): Collection
  {
    return $this->temperatures()->limit($limit)->get();
  }

  public function getHumidity(int $limit): Collection
  {
    return $this->humidities()->limit($limit)->get();
  }

  public function humidities(): HasMany
  {
    return $this->hasMany(Humidity::class);
  }
}
```

Humidity.php

```
class Humidity extends Model
{
  protected $fillable = [
    'value',
    'device_id'
  ];

  public function device(): HasOne
  {
    return $this->hasOne(Device::class);
  }

  public function getDevice(): Device
  {
    return $this->device()->first();
  }
}
```

Temperature.php

```
class Temperature extends Model
{
  protected $fillable = [
    'value',
    'device_id'
  ];
```

```
  public function getValue(): ?float
  {
   return $this->getAttribute('value');
  }

  public function device(): HasOne
  {
   return $this->hasOne(Device::class);
  }

  public function getDevice(): Device
  {
   return $this->device()->first();
  }
}
```

User.php

```
class User extends Authenticatable
{
    use HasApiTokens, Notifiable;
    use WithId;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function getPassword(): string
    {
     return $this->getAttribute('password');
    }
}
```

## A.8 Resources

DeviceResource.php

```
class DeviceResource extends JsonResource
{
 /**
   * Transform the resource into an array.
   *
   * @param Request $request
   * @return array
   */
    public function toArray($request):array
    {
      /** @var Device $device */
      $device = $this->resource;

      $result = $device->toArray();
      $result['temperatures'] = $device->getTemperature(10);
      $result['humidities'] = $device->getHumidity(10);

      return $result;
    }
}
```

DevicessResource.php

```
class DevicesResources extends JsonResource
{
 /**
   * Transform the resource into an array.
   *
   * @param Request $request
   * @return array
   */
    public function toArray($request): array
    {
      /** @var Collection $devices */
      $devices = $this->resource;

      return $devices->map(function (Device $device) {
        return new DeviceResource($device);
      })->toArray();
    }
}
```