ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Кафедра Мобільних та відеоінформаційних технолгій

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«ДОСЛІДЖЕННЯ ВПРОВАДЖЕННЯ КОМПЛЕКСІВ АВТОМАТИЗАЦІЇ ЖИТЛОВИХ БУДИНКІВ З ЗАСТОСУВАННЯМ СИСТЕМ SMARTHOME»

Виконав: студент 6 курсу, групи РТДМ-61

спеціальності 172 Телекомунікації та радіотехніка
(шифр і назва спеціальності)

Луговий І.В.
(прізвище та ініціали)

Керівник _____
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтроль _____
(прізвище та ініціали)

Київ - 2019

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Кафедра Телекомунікаційнх систем та мереж
Ступінь вищої освіти Магістр
Спеціальність 172 Телекомунікації та радіотехніка
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Мобільних та відеоінформаційних технологій
_____

“____” _____2019 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Луговому Іллі Володимировичу

1. Тема роботи: «Дослідження впровадження комплексів автоматизації житлових будинків з застосуванням систем SmartHome», керівник роботи Кравченко Владислав Ігорович, к.т.н., завідувач кафедри МВТ затверджені наказом вищого навчального закладу від _____ року № _____.

2. Строк подання студентом роботи 20.12.2019 р.

3. Вихідні дані до роботи:

1. Показники диму, вологості, температури, освітленості з датчиків встановлених у приміщенні;

2. Науково-технічна література.

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

1. Вдосконалення технології передачі та збереження показників;
2. Обґрунтування доцільності обраних програмних засобів;
3. Оптимізація процессу розгортання системи «Розумний дім»;
4. Впровадження мобільного інтерфейсу системи.

5. Перелік графічного матеріалу:

1. Титул;
2. Мета роботи;
3. Моделювання предметної області;
4. Представлення сховища даних у вигляді DDLпідмножин;
5. Загальна схема та правила прийняття рішень;
6. Вибір інструментарію для створення програмного забезпечення;
7. Алгоритмізація та розробка програмного забезпечення;
8. Емуляція відправлення показників від esp8266;
9. Емуляція пристрою отримання команд;
10. Емуляція відправляння команд від серверу;
11. Висновки.

6. Дата видачі завдання 05.09.2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури	09.09.19	
2.	Вдосконалення технології передачі та збереження показників	11.09.19	
3.	Обгрунтування доцільності обраних програмних засобів	16.09.19	
4.	Оптимізація процессу розгортання системи «Розумний дім»	23.09.19	
5.	Впровадження мобільного інтерфейсу системи	07.10.19	
6.	Висновки, вступ, реферат	21.10.19	
7.	Розробка презентації	04.11.19	

Студент Луговий І.В.

(підпис) (прізвище та ініціали)

Керівник роботи

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської роботи: 66 сторінок, 50 рисунків, 4 таблиці, 20 джерела.

Об'єкт дослідження – комплекси автоматизації житлових будинків.

Предмет дослідження – комплекси автоматизації систем SmartHome.

Мета роботи – впровадження комплексів автоматизації житлових будинків з застосуванням систем smarthome.

Методи дослідження – теорії електрозв'язку, емпіричне дослідження, методи теоретичного дослідження, методи практичного дослідження.

В роботі проаналізовано основні характеристики та аспекти впровадження систем автоматизації з метою реалізації повного комплексу функції SmartHome. Було модернізовано інформаційну систему для впровадження нового функціоналу прийняття рішень та оптимізовано процес моніторингу даних та керуванням пристроями, що підключені системи.

SMARTHOME, IOT, ПЕРЕДАЧА ДАНИХ, СИСТЕМНИЙ АНАЛІЗ, КОМПЛЕКС АВТОМАТИЗАЦІЇ, Apache POI , OPENWEATHERAPI, DOMOTICZ, JAVA

ЗМІСТ

ВСТУП

У сучасному світі веб-сайти у більшості виглядають як додатки з великою кількістю можливостей для взаємодії, а не статичними сторінками, актуальність яких зазнала поразки багато років тому. Для надання можливості прийняття рішень було створено систему, ціль якої задовольнити потреби користувача в автоматизації прийнятті рішень, саме:

- - полегшення взаємодії між масивами даних, алгоритмами аналізу й процедурами обробки даних і моделями прийняття рішень, з однієї сторони, і користувача, який відповідальний за прийняття рішень - з іншої;

- - надання інформації використання якої здатне допомогти, особливо при виконанні слабо структурованих або неструктурованих задач, для яких досить трудомістко заздалегідь змоделювати дані та необхідний функціонал допустимих рішень.

В сучасних умовах розвитку технологій та програмного забезпечення, постає питання оптимальної та доцільної реалізації інновацій в повсякденне життя. Цей процес є цікавим як з точки зору виробників та постачальників обладнання так і з точки зору кінцевого користувача. Особливу увагу виробники та оператори IT-послуг приділяють сучасному та затребуваному напрямку як побудова «розумного дому» (Smart House).

Даний напрям включає в себе не лище новітне обладнання, датчики, системи контролю, системи доступу, тощо, а і програмне забезпечення, за допомогою якого кожен має змогу керувати та отримувати доступ до інформаційних систем. Задяки такому технологічному стрибку стають доступні унікальні можливості дистанційної роботи з обладнанням, керуванням системами безпеки, а також, що найголовніше, отримання доступу систем розумного дому з будь-якої точки світу завдяки глобальній мережі.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Завдання полягає в модернізації створеної інформаційної системи для впровадження нового функціоналу та оптимізації моніторингу даних та керуванням пристроями, що підключені до даної системи.

Модифікація ІС має забезпечити керування пристроями на відстані, відмовостійкість, запровадити модернізацію web інтерфейсу, створити альтернативу web інтерфейсів для мобільних пристроїв та оптимізувати процес розгортання системи.

Реалізувати обмін даними між мікроконтролерами та оновленими ресурсами інформаційної системи на основі використання брокеру повідомлень . Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (таких як: XML,CSV, JSON), використовуючи протокол MQTT.

Особливості реалізації функціоналу створеної системи наведено у таблиці 1.

Таблиця 1

Особливості розроблюваної системи

Функції ІС	Реалізація виконання
Безпечнеопрацювання даних	Spring
Підтримка роботи на будь-якому комп'ютері	Java/Docker
Безпечне збереження даних	MySQL,Mongo DB
Відправлення команд від клієнта до пристрою	HTTP/MQTT
Відображення даних в зручному вигляді	React
Аналіз даних	Apache Spark
Інформування клієнта про позаштатні ситуації	JavaMail/Telegram API
Можливість розпізнавання команд від нових пристроїв	Java
Зчитування показників	Arduino
Відправлення команд на виконання від ІС до МК	RabbitMQ

Оскільки підтримка прийняття рішень в системі “Розумний дім” є транзакцій ним процесом, то ефективність його реалізації, а, отже, й кількість витрачених коштів, буде залежати в першу чергу від тих рішень, що будуть прийняті власниками помешкання на етапі використання інформаційної системи.

Таким чином можна сказати, що основною задачею системи, що створюється, прийняття автоматизованих рішень на основі результатів отриманих показників мікроклімату,формування звітів. Одним з основних елементів системи є база даних (рис.1.1). Вона розроблена з метою збереження та використання даних в найбільш оптимальному вигляді. У зв’язку з тим, що дані, які можуть надаватися безпосередньо датчиками моніторингу мікроклімату, команди , що надходять від користувачів можуть істотно відрізнятися в залежності від помешкання, на якому система використовується. Тому під час проектування бази даних необхідно враховувати гнучкість та залежність від можливості до масштабування.

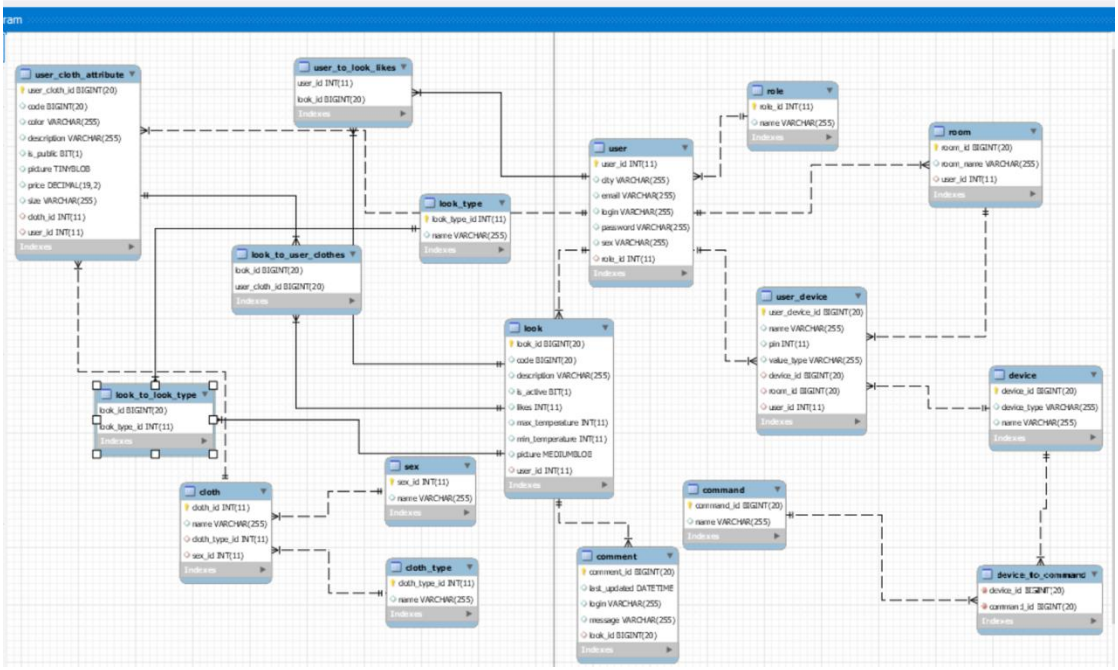


Рис. 1.1 Логічна схема БД

Аналітичну функціональність системи впроваджено за рахунок взаємодії із створеним сховищем даних – який підтримує хронологію показників та

виконання команд користувачів і надає можливість бути використаним як джерело для оперативного аналізу та прийняття рішень (рис. 1.2).

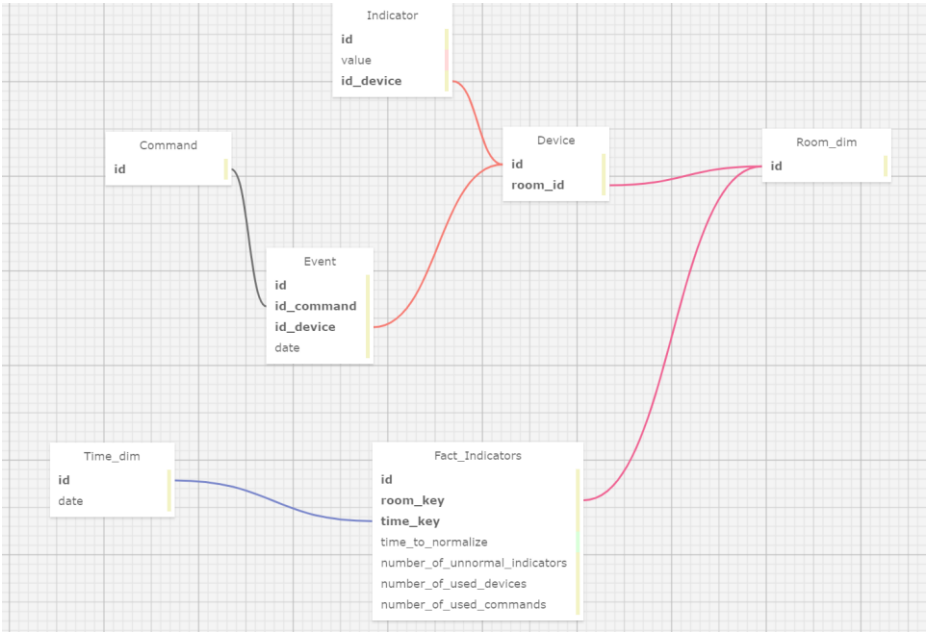


Рис 1.2 Сховище даних

Сховище даних повинно будуватися з метою надання інформацію щодо питань, наведених нижче:

- Кількість часу , що було використано для нормалізації показників
- Кількість пристроїв , що було використано для нормалізація показників
- Кількість ненормованих показників

1.2 Огляд інформаційних джерел та існуючих рішень

Станом на сьогоднішній день, представлені аналоги розробленогододаткуне можуть задовольнити у повній мірівирішення поставлених задач. Одним з прикладів є програма Domoticz, дана програма надає можливість керування пристроями, але для її впровадження кожен користувач має мати локальну робочу станцію , на якій буде розгорнуто цей додаток.Також недоліком є можливість використання пристроїв лише від певних вендорів (наприклад xiaomi) у той час ,

як розроблена система забезпечує зв’язок із пристроями які підтримують використання MQTTпротоколу

1.3 Моделювання предметної області

Діаграма прецедентів — в UML, діаграма, за допомогою якої відображається взаємодія в системі між *акторами* та *прецедентами*.

Діаграма прецедентів являється графом, який можна представити у вигляді:

- списку акторів,
- взаємодії між прецедентами,
- набору прецедентів які не виходятьза границю системи,
- зв’язку прецедентів та акторів,

Ціллю даної діаграми є можливість графічного відображення спроектованої системи у вигляді множини прецедентів та акторів, які взаємодіють між собою на основі стандартних видів взаємозв’язків. Кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому завдяки абстрактній побудові діаграми, фактична реалізація взаємодії акторів із системо не зазначається у даному виді UML діаграм.

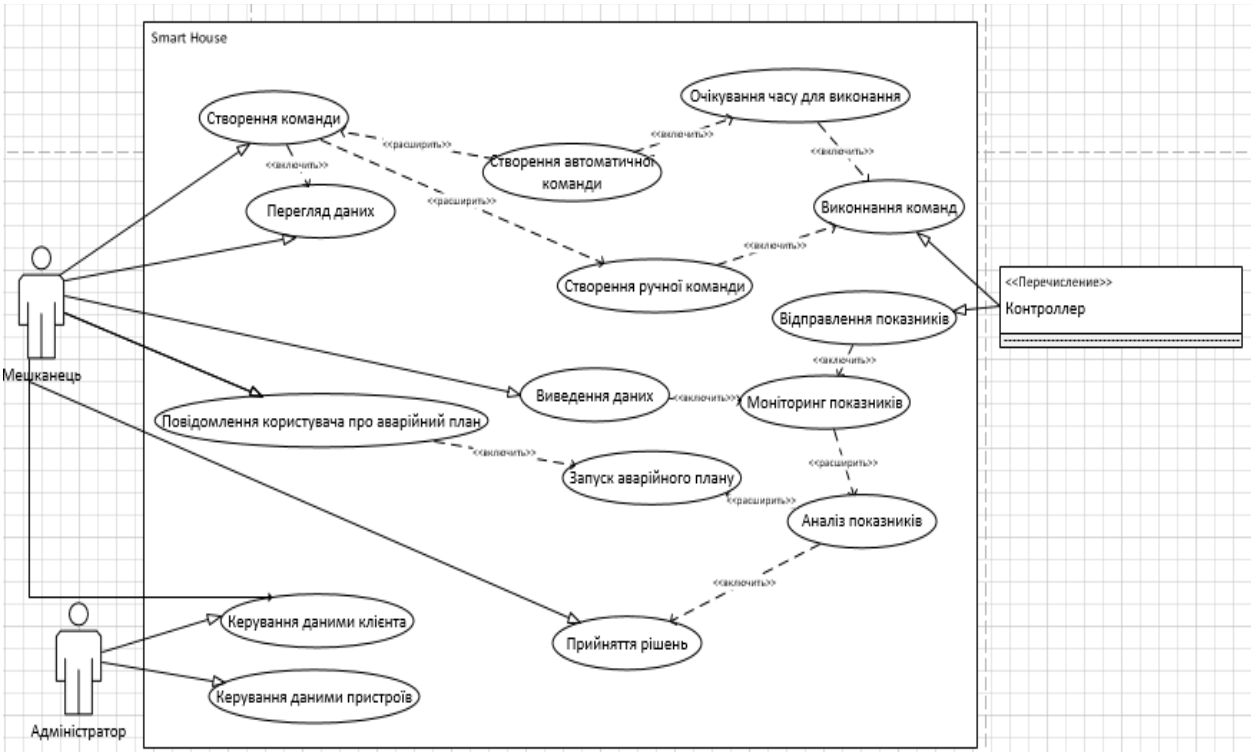


Рис. 1.3 Діаграма прецедентів

На діаграмі прецедентів описані наступні три діючі актори:

- Адміністратор;
- Мешканець;
- Контроллер.

Актор «Адміністратор» відповідальний за виконання дій:

- Управління даними клієнта;
- Управління даними пристроїв.

Актор «Мешканець» може виконувати наступні дії:

- Створення команди;
- Перегляд даних.

Актор «Контроллер» забезпечує автоматичне виконання наступних функцій системи:

- Виконання команд;
- Відправлення показників.

1.3.3 Діаграма розгортання

Діаграму розгортання було застосовано під час моделювання взаємодії вузлів та компонентів.

Діаграма розгортання системи зображена на рис. 1.4

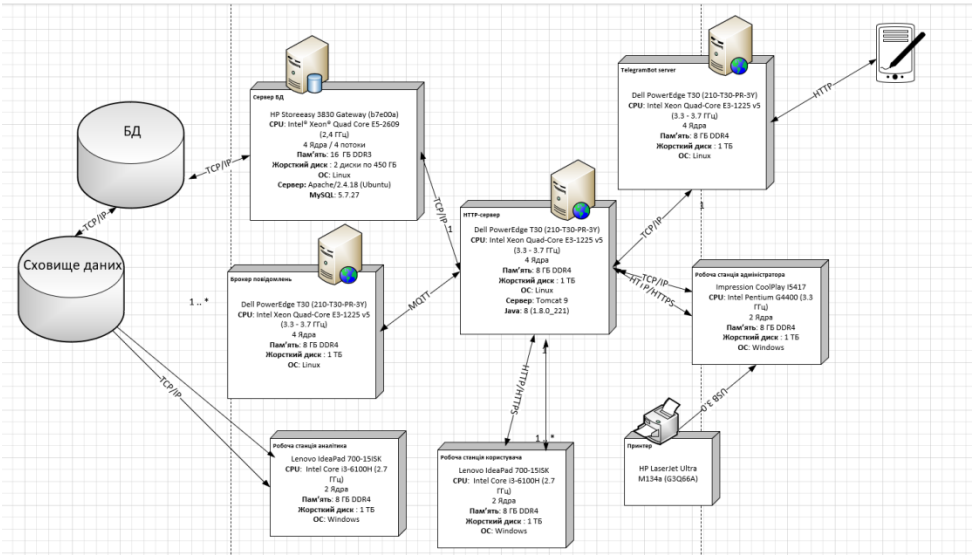


Рис 1.4 Діаграма розгортання системи

1.3.4 Діаграма FDD

Для представлення загальної моделі функціональної системи та представлення її властивостей було використано FDD діаграму

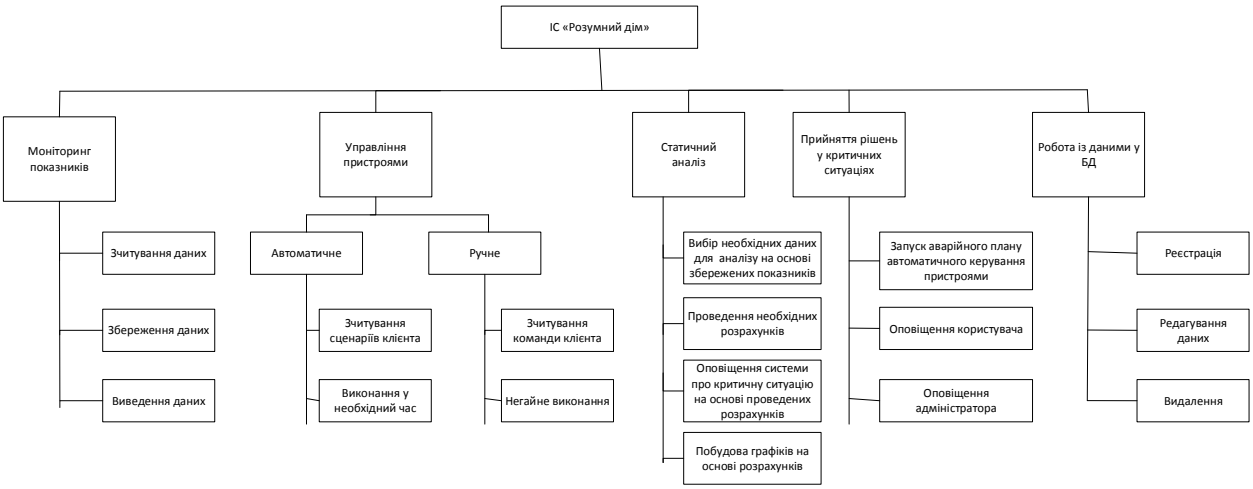


Рис 1.5 FDD Діаграма

2 РОЗРОБКА МОДУЛЮ ПРИЙНЯТТЯ РІШЕНЬ

2.1 Опис задачі алгоритму

Даний функціонал забезпечує постійний та комфортний мікроклімат в будівлі. Температурний комфорт створюється за допомогою постійного моніторингу показників температури, вологості та якості повітря. З подальшим втручанням за допомогою пристроїв керування у зв’язку з небхідністю оптимізації мікроклімату. Задачею даного модулю є контроль температури та якості повітря.

Основні функції:

- Автоматичний контроль температурного режиму та вологості;
- Дистанційнекерування.

За основу для оптимальних показників мікроклімату було взято міждержавний стандарт ГОСТ 30494-2011. В якому описані вимоги до мікроклімату громадських і житлових будівель. Цей ГОСТ надає визначення параметрам мікроклімату убудівлі як «статус внутрішньої екосистеми помешкання, що завдає вплив її мешканців». Внутрішнє середовище – це, переважно, мікроклімат всередині будівлі. Наступним пунктомє уточнення, що мікроклімат приміщення описується як сукупність показників температури та вологості.

Мікроклімат житлових і громадських помешкань нараховує велику кількість параметрів, але основними є:

- Температура повітря;
- Вологість повітря;
- Чистота повітря;

Вимоги до температури повітря:

ГОСТ для мікроклімату регламентує стан показників температури повітря у помешканнях. У літній період вважається оптимальним діапазон 22-25С. Під час зимньої пори року діапазон змінюється до: 20-23С, також є градація в залежності від типу кімнати для житлових кімнат, 24-26° С.

Вимоги до вологості повітря:

Рекомендованою для людини вологістю вважається відмітка в – 40-60%. Якщо показники перевищують вказані– це вже визначається як вогкість, яка може призвести до псуванням майна у приміщенні. ГОСТ і СанПіНзазначають для показників мікроклімату поміщеньнаступні значення оптимальної вологості: 30-45% в зимній період часу і 30-60% в літній.

Вимоги до чистоти повітря:

На якість чистоти повітря в помешканні перед усім вказує вміст вуглекислого газу, який прийнято вимірювати в одиницях ppm. ГОСТ «Параметри мікроклімату» вважає оптимальними показники за рівнем 800 – 1 400.

2.2. Представлення сховища даних у вигляді DDL підмножини

```
-- Table 'Command'
--
-- ---

DROP TABLE IF EXISTS `Command`;

CREATE TABLE `Command` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

```
DROP TABLE IF EXISTS `Device`;

CREATE TABLE `Device` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  `room_id` INTEGER NULL DEFAULT NULL,
  `i` INTEGER NULL DEFAULT NULL,
  `u` INTEGER NULL DEFAULT NULL,
  PRIMARY KEY (`id`, `room_id`)
);

DROP TABLE IF EXISTS `Event`;

CREATE TABLE `Event` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  `id_command` INTEGER NULL DEFAULT NULL,
  `id_device` INTEGER NULL DEFAULT NULL,
  `date` INTEGER NULL DEFAULT NULL,
  PRIMARY KEY (`id`, `id_command`, `id_device`)
);


DROP TABLE IF EXISTS `Room_dim`;

CREATE TABLE `Room_dim` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

```
DROP TABLE IF EXISTS `Fact_Indicators`;

CREATE TABLE `Fact_Indicators` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  `room_key` INTEGER NULL DEFAULT NULL,
  `time_key` INTEGER NULL DEFAULT NULL,
  `time_to_normalize` DATETIME NULL DEFAULT NULL,
  `number_of_unnormal_indicators` INTEGER NULL DEFAULT NULL,
  `number_of_used_devices` INTEGER NULL DEFAULT NULL,
  `number_of_used_commands` INTEGER NULL DEFAULT NULL,
`number_of_used_elec` INTEGER NULL DEFAULT NULL,
`price` INTEGER NULL DEFAULT NULL,


  PRIMARY KEY (`id`, `room_key`, `time_key`)
);

-- ---
-- Table 'Time_dim'
--
-- ---

DROP TABLE IF EXISTS `Time_dim`;

CREATE TABLE `Time_dim` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  `date` INTEGER NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
);
```

```
DROP TABLE IF EXISTS `Indicator`;

CREATE TABLE `Indicator` (
  `id` INTEGER NULL AUTO_INCREMENT DEFAULT NULL,
  `value` MEDIUMTEXT NULL DEFAULT NULL,
  `id_device` INTEGER NULL DEFAULT NULL,
  PRIMARY KEY (`id`, `id_device`)
);

-- ---
-- ForeignKeys
-- ---

ALTER TABLE `Command` ADD FOREIGN KEY (id) REFERENCES `Event` (`id_command`);
ALTER TABLE `Device` ADD FOREIGN KEY (id) REFERENCES `Event` (`id_device`);
ALTER TABLE `Device` ADD FOREIGN KEY (id) REFERENCES `Indicator` (`id_device`);
ALTER TABLE `Room_dim` ADD FOREIGN KEY (id) REFERENCES `Device` (`room_id`);
ALTER TABLE `Room_dim` ADD FOREIGN KEY (id) REFERENCES `Fact_Indicators` (`room_key`);
ALTER TABLE `Time_dim` ADD FOREIGN KEY (id) REFERENCES `Fact_Indicators` (`time_key`);

-- ---
-- TableProperties
-- ---
```

```
-- ALTER TABLE `Command` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Device` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Event` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Room_dim` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Fact_Indicators` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Time_dim` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Indicator` ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

2.3 Загальна схема та правила прийняття рішень

Для того, щоб отримати інформацію, про стан помешкання використовуються датчики вологості, температури та якості повітря. Усі вони підключені до мікроконтролерів esp8266 , який взаємодіє із системою «Розумний дім» за допомогою брокеру повідомлень використовуючи MQTT протокол. Далі на основі отриманих показників ІС формує рішень та відправляє рішення також використовуючи брокер повідомлень на вказаний пристрій керування. Загальна структура описаної схеми вказана на рис. 2.1

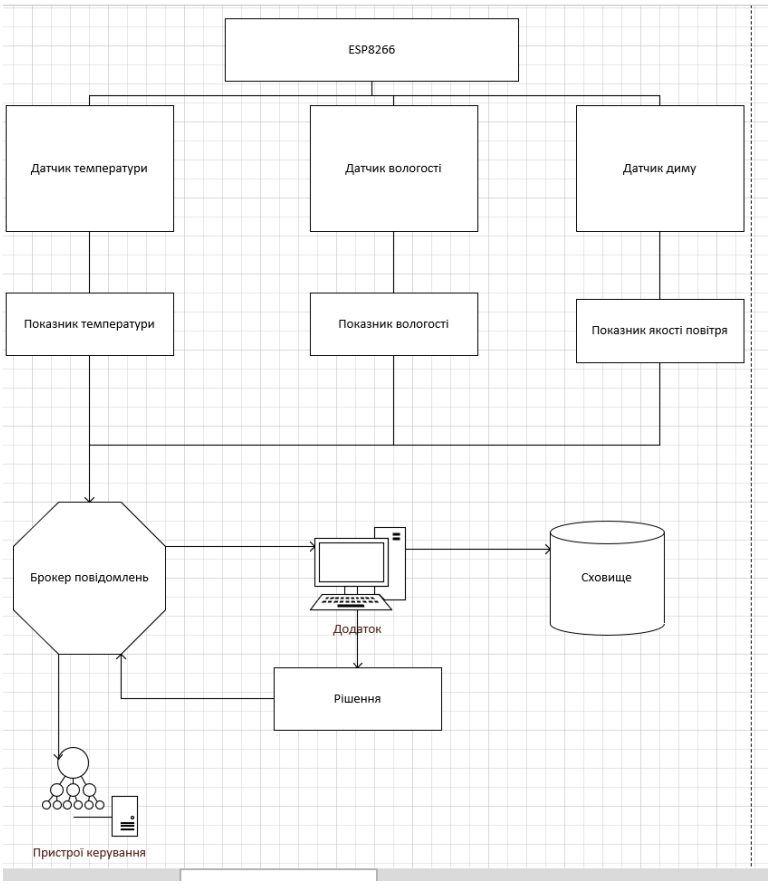


Рис 2.1 Загальна схема взаємодії

Кожне приміщення можна характеризувати як набір показників, що видають датчики, які встановлено в цих приміщеннях в таблиці №2 представлені певні правила.

Основними командами, які будуть керувати при зміні того чи іншого параметру будуть виступати функції:

D0 – виведення інформації про відсутність датчиків

D1-D3- Збільшити температуру, ввімкнути зволожувач, відкрити вікна

D4-D6- Збільшити температуру, ввімкнути зволожувач, закрити вікна

D7-D9- Збільшити температуру, вимкнути зволожувач, відкрити вікна

D10-D12- Збільшити температуру, вимкнути зволожувач, закрити вікна

Таблиця №2

Умови роботи системи по контролю повітря в приміщенні

Чи активована функція	Температура	Вологість	Якість Повітря	Використання ресурсів	Рішення
+	-	-	-	-	D0
	<22	<30	<800	MIN	D1
				NORM	D2
				MAX	D3
			>1400	MIN	D4
				NORM	D5
				MAX	D6
	>22	>45	<800	MIN	D7
				NORM	D8
				MAX	D9
			>1400 -	MIN	D10
				NORM	D11
				MAX	D12
-	-	-	-	-	-

Таким чином, система буде запрограмована на автоматичну зміну характеристик навколишнього середовища. Наприклад, при температурі менше 22С система дистанційно подасть команду «Ввімкнути обігрівач в будинку», а отже, втручання людини в цей процес не є обов’язковим.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ТА АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Обгрунтування технічного забезпечення системи

MQ2 – датчик диму, побудований на основі газоаналізатору, забезпечує пошук у повітрі вуглекислих газів диму та водороду.

Рис. 3.1. Датчик диму

Характеристики

- Напруга: 5 Вольт
- Споживаний струм: 160 міліАмпер

Діапазонвимірів

- Водень: 0,3–5пр.
- Пропан: 0,2 – 5 пр.
- Метан: 5–20 пр.
- Бутан: 0,3–5 пр

DHT22 — датчик температури і вологості, комплекс з каліброваним цифровим виходом сигналу. Даний датчик дозволяє вимірювати вологість резистивного типу і компонент вимірювання NTC температури, завдяки відключенню до 8-розрядного мікроконтролеру, тим самим пропонуючи швидке реагування та задовільну якість.

Рис. 3.2. DHT 22

Характеристики:

- визначення вологості 0-100% з 5% точністю;
- визначення температури -40-125 град. з точністю 2 град;
- вастота опитування не більше 1 Гц (не більше одного разу в 1 сек.);
- Розміри 15.5мм х 12мм х 5.5мм;
- виведення з відстанню між ніжками 0.1 .
- Напругавід 3 до 5В.
- Споживаний струм - 2.5мА

Було вирішено змінити датчик DHT11 на DHT22, порівняльна характеристика приведена у таблиці.

Таблиця 3.1

Характеристика датчиків повітря

Датчик та характеристика	DHT11	DHT22	DHT21
Діапазон виміру температур	0..60°C, ±2%	-40...+125°C, ±0.5°C	
Діапазонвиміру вологості	20..90%, ±2%	0-100%, ±2%	
Вага, гр	1	2.2	11
Округлення результатів	до цілих	до десятих	
Використання току при 5В	0,1	0,14	1,36
Умови для використання	в приміщені		вулиця

Датчик відстані HC-SR04

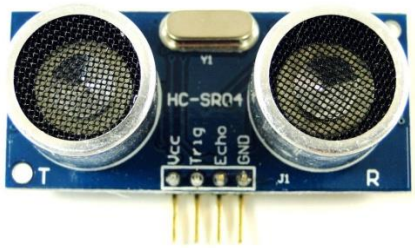


Рис.3.3. Датчик відстані

- Напруга: 5 Вольт
- Струм: 2міліАмпери
- Відстань: 2–400 см

RGB LED Трьохколірний світлодіод RGB LED з діаметром лінзи 5мм. Налічує чотири виводи із загальним катодом. Лінза прозора.

Рис. 3.4. LED Світлодіод

RobotDyn – мікроконтролер ATmega328P який використовується в ArduinoUNO і мікроконтролер ESP8266 зприсутнім wifi модулем і флєш-пам'яттю - 8 Мб. За допомогою пристроїв встановлених на даній платі є можливість

використовувати плату як зазначену вище Arduino UNO, або як окремий модуль– ESP8266, який теж може працювати самостійно. Також є можливість використання даної плати у форматі UNO+ESP8266.Їх взаємодія виконується за допомогою COMпорту. Відповідно, у цьому режимі не буде можливості використовувати моніторинг порту у ArduinoIDE.

Зміна режиму роботи плати виконується за допомогою 8 перемикачів встановлених на платі (табл №).

Взаємодія з ПК виконується з використанням microUSB порту. Під час взаємодії створюється віртуальний COMпорт.Оскільки виконана плата у форм-факторі ArduinoUno, є можливість підключення модулів , що були створені для Arduino.

Рис.3.5. RobotDyn

Для відлагодження пристроїв використовувалася Raspberry Pi 3 Model B з 4-ядерним A53Cortex з ARMтипом, що працює на частоті 1.2GH.

Рис. 3.6. RaspberryPi 3 Model B

Таблиця 3.2.

Режими роботи RobotDyn

Режим роботи /№ перемикачу	1	2	3	4	5	6	7
CH340 <-> ESP8266 (upload sketch)	-	-	-	-	+	+	+
CH340 <-> ESP8266 (connect)	-	-	-	-	+	+	-
CH340 <-> ATmega328 (upload sketch)	-	-	+	+	-	-	-
ATmega328+ESP8266	+	+	-	-	-	-	-
Незалежна робота модулю	-	-	-	-	-	-	-

Використання даного апаратного забезпечення дозволяє забезпечити функціонування, тестування та відлагодження інформаційної системи, набір пристроїв моніторингу покриває собою перелік показників, які слід контролювати у жилих приміщенях за ГОСТ. Встановлення системи на raspberrypi надає можливість тестування системи із використанням ARM архітектури та оптимізації швидкодії системи.

3.2 Вибір інструментарію для створення програмного забезпечення

В наслідок перегляду інфраструктури інформаційної системи «розумний дім» було вирішено спростити можливість впровадження системи шляхом використання dockerконтейнерів.Перелік модулів , що знаходяться в контейнерах зображено на рисунку 3.7.

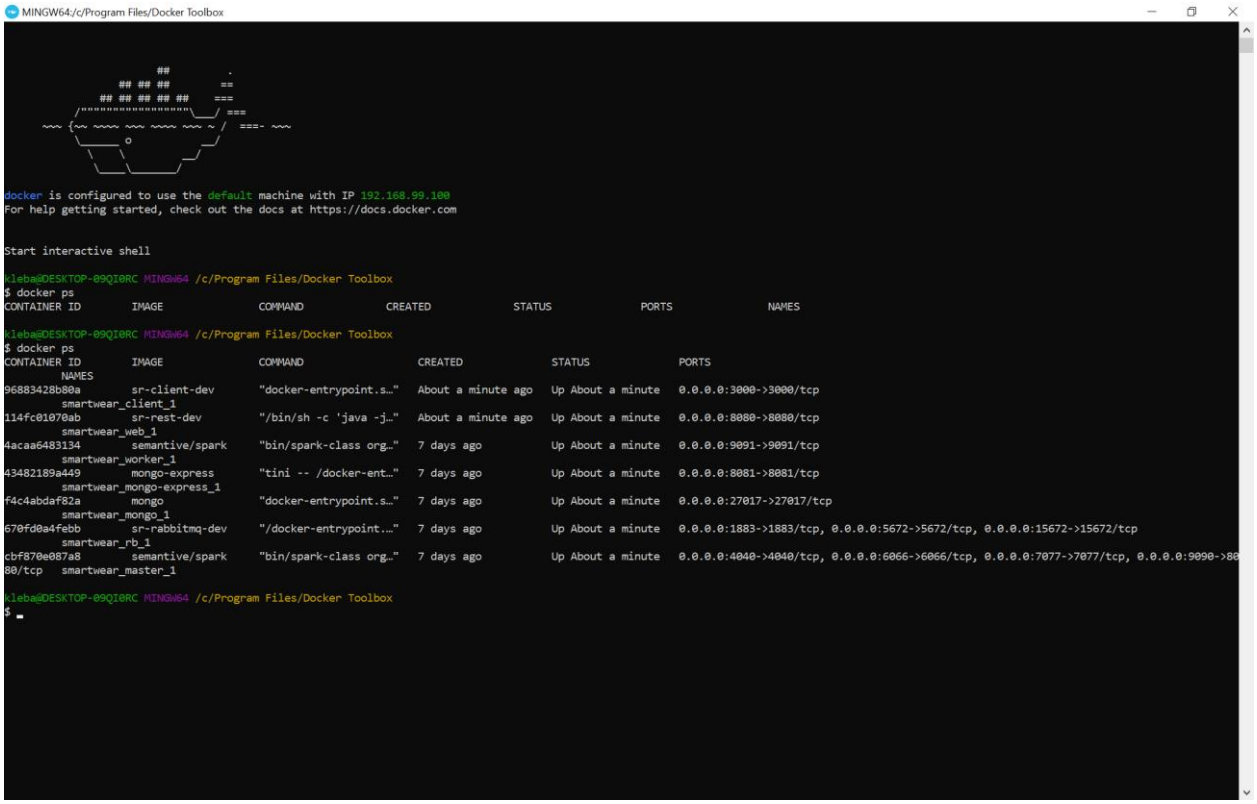


Рис 3.7 Докер контейнери

Архітектура системи була побудована у форматі клієнт-серверної з веб-клієнтом. Датчики підключаються до мікроконтролеру esp8266 який взаємодіє з чергою повідомлень шляхом відправлення повідомлення у певний топік, що в свою чергу прослуховує spring додаток який завантажено в контейнер сервлетів (Tomcat). Після отримання повідомлення springдодаток зберігає показники в нереляційній СУБД MongoDb

Docker — застосунок для керування Linux контейнерами в ізольованому стані. Docker розширює технології LXC більш високорівневим інтерфейсом взаємодії, що забезпечує керування контейнерами використовуючи рівень ізоляції окремих процесів. Наприклад, використання Docker надає можливість не звертаючи увагу на вміст контейнера запускати будь-які процеси використовуючи режим ізоляції апотім переносити і клонувати створені для даних процесів контейнери на інші робочі станції, забезпечуючи всю роботу, що забезпечує, обслуговування, створення і модифікції контейнерів.

Docker налічує два основних процеса:

- Демон Docker, який виконується на машині користувача (у випадку коли ОС є Linux), або із використанням VirtualBox середовища boot2docker (у випадку коли ОС є Windows або macOS).
- Клієнт, який забезпечує взаємодію з демоном.

Docker образ (*Docker image* англ.) – складається з операційної системи, до датку та усіх необхідних його залежностей. Docker образи збираються із шарів. Якщо є необхідність в образі із веб-сервером, то доцільно обрати образ за основу який містить дистрибутив необхідної операційної системи, наступним кроком додати залежність - веб-сервер, та зберегти створене як новий образ, який буде налічувати два шари - перший з операційною системою, другий із веб-сервером. Надається можливість обмінювати створені образи через DockerHub

Docker контейнер - це образ, що було запущено. У docker контейнерів є можливість запуску, зупинки, видалення та переміщення. Є можливість створити коміт docker контейнера, для використання нового образу із станом контейнеру на момент коміту.

Основні можливості Docker:

- Забезпечення використання різних додатків в ізольованому середовищі, що дозволяє використовувати різні комбінаціїбібліотек, конфігураційних файлів, скриптів, файлів jar, war та інших.
- Можливість використання на будь-якому комп'ютері архітектура якого була створена на основі x86/x64, ARM з системою на базі ядра Linux, починаючи з одноплатних комп’ютерів та, закінчуючи кластерами серверів.
- Можливість роботи в стандартних оточеннях більшості популярних дистрибутивів Linux, включаючи Mint, Ubuntu, Debian, Armbian, Raspbian і Arch;
- За рахунок використання ос в мінімальній конфігурації досягається легковаговістьконтейнерів

- Оскільки в контейнерах використовується кас томна та незалежна файлова система, не має різниці, на якому пристрої вони запускаються;
- Завдяки забезпеченню на рівні файлової системи - ізоляції. Процеси виконується у окремій кореневій файловій системі;
- Ізоляція ресурсів: Використання ресурсів системи, таких як навантаження на процесор та втрати оперативної пам'яті, є можливість налаштування окремо для кожного із використовуємих контейнерівза допомогою утіліти cgroups;
- Файлова система контейнера може бути перевикористана як основа для створення нових образів тавиготовлення інших контейнерів, не маючи необхідності оформлення шаблонів або кастомного налаштування складу образів;
- Можливість використання інтерактивної командної оболонки: до стандартного вводу будь-якого контейнера
- Можливість створення контейнерів, що використовують великі стеки технологій, за допомогою налаштування зв'язку між створеними контейнерами, що містять модулі, які цілком формують стек.

Зв'язування здійснюється не за допомогою об’єднання, а завдяки створенню мережевої взаємодії між використовуємими контейнерами (створюється мережевий тунель).

Docker Compose - це інструментальний засіб, що входить до складу Docker. Він призначений для вирішення завдань, пов'язаних з розгортанням проектів.

Docker Compose використовується для одночасного управління декількома контейнерами, що входять до складу програми. Цей інструмент пропонує ті ж можливості, що і Docker, але дозволяє працювати з більш складними додатками.

Під час встановлення Dockerна Windows 10 Home використовується OracleVM VirtualBox, для повноцінної роботи системи слід перевизначити ip адресу хосту (Рис 3.8).

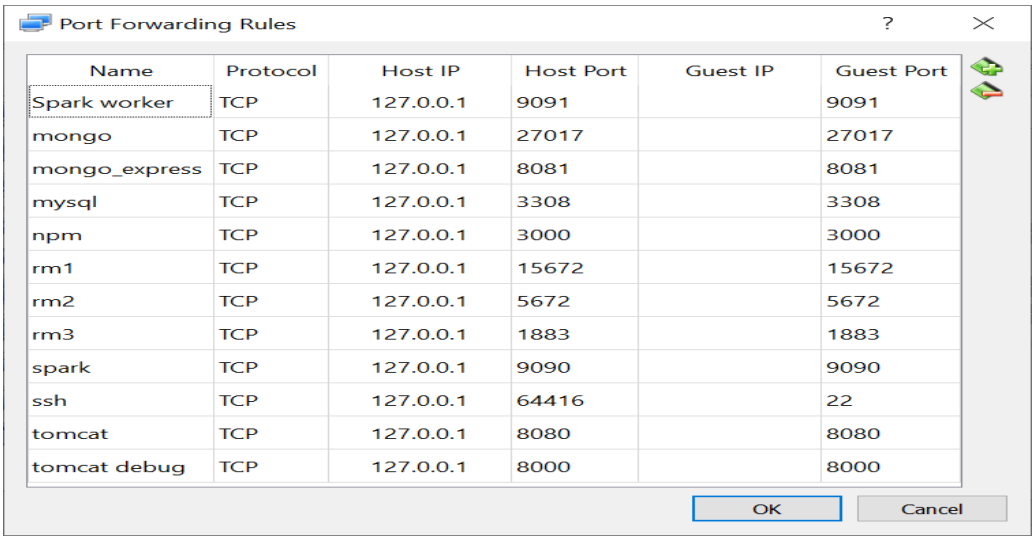


Рис 3.8 Перевизначення HostIP

SpringFramework — фреймворк з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java.

Розроблений Spring веб додаток дозволяє зчитувати показники в реальному часі, керувати пристроями, отримувати звіт у форматі excel. При необхідності, функціонал системи може бути збільшено.

Java – одна з популярних мов програмування, яка була створена в 1995 році й на даний момент має одинадцять виданих версій. Java популярна оскільки, це пов’язано з її унікальною можливістю портативності: програми Java працюють на більшості пристроїв або на всіх операційних системах. З 2009 року Java була не першою для надання можливості написання кросплатформних програм, але вона зазнала успіху: Концепція «Написав код один раз – май можливість запуску скрізь» дозволила зацікавити велику кількість розробників до написання нових версій мови. Java-розробники можуть створювати додаток локально, а потім завантажувати його на необхідний платформі, платформі – сервері, комп’ютері, телефоні та інших. Якщо для компілятора доступні необхідні бібліотеки, код розробника буде перетворено у байт код, який вже необхідна віртуальна машина зможе інтерпретувати.

Бібліотеки які поширюються із відкритим кодом значно оптимізують використання та написання Java додатків Google, Amazon, Apache, та інші

організації створили бібліотеки, що допомагають у розробціта оптимізації швидкості написання програм.

Доцільно шукати створені бібліотеки, перш ніж планувати власну імплементацію. Оскільки є можливість того, що потрібний функціонал вже створено, протестовано та підготовлено до використання.

Spring – це програмний модуль (фреймворк) який поширюється з відкритим кодом та маєконтейнери, що надають реалізацію парадигми інверсії управління для Java додатків.

Інверсія управління (IoC) це парадигма, за якою необхідно створювати програми, при якій частини даної програми використовується із загального модулю для спільного використання – контексту.

Ін’єкція залежностей (DI) – Імплементація IoC парадигми у Spring додатку

Spring Framework налічує модулі, які надають можливість скористатися готовими рішеннями, для взаємодії із БД, брокером повідомлень, тощо: під час написання даної роботи використовувалися:

-Sping Core

-Spring MVC

-Spring Boot

-Spring Data

-Spring Security

Spring core – це ядро, основа платформи, надає інструменти для створення застосунків – управління життєвим циклом компонентів (біни, beans), впровадження залежностей, ініціалізації контексту додатку, транзакції, базовий доступ до БД. В більшості це компоненти та абстракції, що знаходяться на низькорівневому прошарку додатку. Каркас, що не явно використовується іншими модулями для взаємодії.

Spring MVC – створений на основі GRASP паттерну Модель - Вид – Контроллер для застосування компонентів, що слабо зв’язані між собою, для більш легкої підтримки та модифікації. Патерн проектування MVC поділяє основні прошарки додатку (точку взаємодії користувача та системою, бізнес-модель системи і тип відображення результату запитів), надаючи при цьому зв'язок між ними.

- Model (Модель) - дані, у більшості своїй зберігаються у БД, використовуються для обрахунків або відображення стану системи.
- View (Відображення, Вид) – керує відображенням даних Моделі, - як правило, в SAP додатках генеруючи JSON;
- Controller (Контролер) - точка взаємодії користувача та додатка, на запит від користувача (Наприклад, HTTP) створює необхідну Model і надсилає її для відображення в View;

Алгоритм роботи Spring MVC побудовано за допомогою взаємодії з DispatcherServlet, який є точкою входу для всіх HTTP-запитів і відповіді на них. Цей принцип роботи найкраще описано в шаблоні проектування FrontController Фронт-контролер найчастіше використовується у веб-застосуваннях, де є багато подібних речей, які потрібно виконати при обробці запиту. Це може бути безпека, інтернаціоналізація, забезпечення певного вигляду для певних користувачів. Якщо обробка вхідних запитів розподілена між кількома контролерами це може призвести до дублювання поведінки. Крім того виникають складності зі зміною поведінки під час виконання. Фронт-контролер об'єднує обробку запитів шляхом їх направлення через єдиний об'єкт-обробник. Цей об'єкт реалізовує загальну поведінку, яка може бути змінена під час виконання за допомогою декораторів. Після цього Frontcontroller створює потрібні об'єкти відповідно до запиту та викликає методи для реалізації конкретної задачі.

SpringSecurity - це модуль Springframework, що надає функціонал створення систем аутентифікації та авторизації, а також інший необхідний інтерфейс забезпечення безпеки для застосунків, створенихза допомогою Springframework.

Ключові об'єкти контексту SpringSecurity:

- SecurityContextHolder, в ньому зберігаються дані про сучасний контекст та стан безпеки додатку, який містить в собіповну інформацію про клієнта системи (Principal), який в даний час працює над додатком. За налаштуваннями за замовченням SecurityContextHolder використовує ThreadLocal для збереження інформації, що вказує на можливість доступу до контексту безпеки для методів що викликаються та виконуються в межах одного потоку;
- SecurityContext, опрацьовує бін Authentication і коли необхідно містить дані системи безпеки, що зв'язані із запитом, що надійшов від користувача; Authentication відображає клієнта системи (Principal) з точки зору SpringSecurity;
- GrantedAuthority відображає права доступу які базуються на певній ролі користувача у даній системі, що були видані, під час внесення/модифікації даних клієнта. Наприклад, в розробленій системі використовується такі ролі користувачів як: ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN;
- UserDetails надає необхіднідані для створення біну Authentication з об'єктів додатка або інших джерел інформації про користувача для системи безпеки. Бін UserDetails налічує наступні атрибути: логін, пароль, логічні атрибути: чи дія акаунта користувача закінчилася, чмакаунт не було заблоковано, чи креди для авторизації не змінилися, чи акаунт було активовано і колекція - ролей (прав) клієнта.

Односторінковий застоусунок (SPA) — це додаток, що працює в браузері з метою забезпечити користувачу досвід близький до користування настільною програмою. Його архітектура побудована таким чином, що при переході на нову

сторінку оновлюється лише частина вмісту. Таким чином, немає необхідності повторно завантажувати ті самі елементи. Це дуже зручно для розробників та користувачів. Прикладом технології SPA є реалізований сервіс Gmail компанією Google. Для розробки SPA використовується одна з найпопулярніших мов програмування — JavaScript.

Багатосторінковий додаток (MPA) нараховує декілька веб сторінок із інформацією, що не змінюється (текстом, зображеннями тощо) та посиланнями на інші сторінки однаковим вмістом. Під час запиту переходу на іншу сторінку браузер робить новий запит досервера і знову отримає у відповідь всі ресурси, не зважаючи на те, що певні компоненти, можуть повторюватись навсіх сторінках (такі як заголовок, нижній та верхній колонтитули). У цьому випадку, продуктивність використовується назавантаження та відображення елементів, що вже були завантажені. Таким чином, це не може не впливати продуктивність на швидкодію системи. Основними технологіями для такого виду веб-додатку є HTML, JSP та CSS. Описані технології також були використані для створення перших веб-сайтів, і не зважаючи на час продовжують застосовуватися сьогодні, для створення сучасних веб-додатків.

HTML — мова розмітки гіпертексту. Було створено для використання під час розробки веб-сторінок. HTML файл використовується інтерпретатором браузера , після чого виводиться у зручному для користувача представленні. Мова, яку застосовуються для створення розмітки гіпертекстових документів HTML забезпечує визначення різні видів елементів, що надає можливість забезпечити функціональність документа: наприклад частики тексту із заданими параметрами для можливості форматування сторінок, зображення, таблиці, списки,посилання та інші. HTML елементи оголошуються за допомогою використання команд розмітки, або по іншому - тегами. Всі HTML теги, що були знайдені інтерпретатором в тексті документа оброблюються браузером при відображенні документа.

CSS — мова яку було створено, для використовується підперегляду сторінок, написаних на мовах для розмітки даних. Найбільш часто CSS можна зустріти підчас візуального представлення сторінок, написаних HTML, але незважаючи на це формат CSS може бути використаним до XML-документів.

CSS використовують під час визначення кольорів, шрифтів, положення елементів та інших аспектів вигляду сторінки. Одна з головних переваг — це можливість розділення вигляду сторінки від наповнення документу Іплементація даної концепції надає можливість покращення сприйняття та доступності сторінки, тим самим забезпечивши більшу гнучкість і можливість контролю за наповненням контенту в різних умовах.

JavaScript — мова, яка було створена для розробки інтерактивних веб-документів, об'єктноорієнтована прототипна мова програмування. Найчастіше використовується для виконання сценаріїв для веб-сторінок, що надає можливість на стороні користувача, без повного перезавантаження веб сторінки відповідно спростивши навантаження на сервер побудувати взаємодію із користувачем, маніпулювати браузером, створти асинхронний обмін даними із сервером застосунку, а також динамічно змінювати структуру та наповнення вигляд веб додатку.

ApacheMaven – фреймворк що, використовується для можливості автоматизації збирання проектів, на основі їх конфігурації описаної у файлі pom.xml, використовуючи мову POM, яка є підмножиною XML.

Життєвий цикл Maven проекту - це перелік фаз, які описують послідовність дій під час побудови проекту життєвий цикл Maven має наступні основні фази виконання:

clean - цикл використовується для очищення тимчасових або конфігураційних файлів, створених maven. Містить наступні фази:

- pre-clean – дії, що виконуються перед видаленням;
- clean = видалення;
- post-clean дії, що виконуються після видалення;

default - цикл що виконує основні фази збору проекту,

- validate - виконується перевірка, чи є структура проекту повну й правильну;
- generate-sources – генерація source файлів;
- process-sources - опрацювання source файлів;
- generate-resources - генерація файлів з ресурсами;
- process-resources – опрацювання файлів з ресурсами;
- compile– компіляція додатку;
- process-test-sources - опрацювання тестових source файлів;
- process-test-resources - опрацювання текстових ресурсфайлів;
- test-compile – компіляція тестів;
- test - код проходить тестування набором створених тестів. Для подальшого збору проекту усі тести мають завершитися успішно.
- package - пакування відкомпільованогопроекту у вказаний в pom.xmlформат;
- integration-test–тестування програмного забезпечення за допомогою інтеграційних тестів
- install–створення проекту в локальному репозиторію користувача з наданням можливості імпорту в іншіпроекти
- deploy – Після проходження усіх тестів та створення у локальному репозиторії проект може бути відправленим на віддалений maven репозиторій

Site — життєвий цикл генерації звітної документації. Нараховує такі фази:

1. pre-site; - дії, що виконуються перед створенням документації
2. site – створення документації;
3. post-sitедії, що виконуються після створення документації;
4. site-deploy – відправлення документації.

Maven було створено на основі на плагін-архітектурі, яка забезпечує застосування плагінів для різних цілей, під час збирання проекту (компіляція, тестування, побудова, відправка, перевірка стилю та інші) для даного проекту, без прямої необхідності їх інсталяції в ручному виді.

Apache POI - це набір Java інтерфейсів, що надають функціонал для CRUD операцій із файлами у форматах офісних застосунків Microsoft Office,

На основі даних інтерфейсів було створено модуль генерації звітів , що формує звіти на основі показників користувача (рис. 3.8)

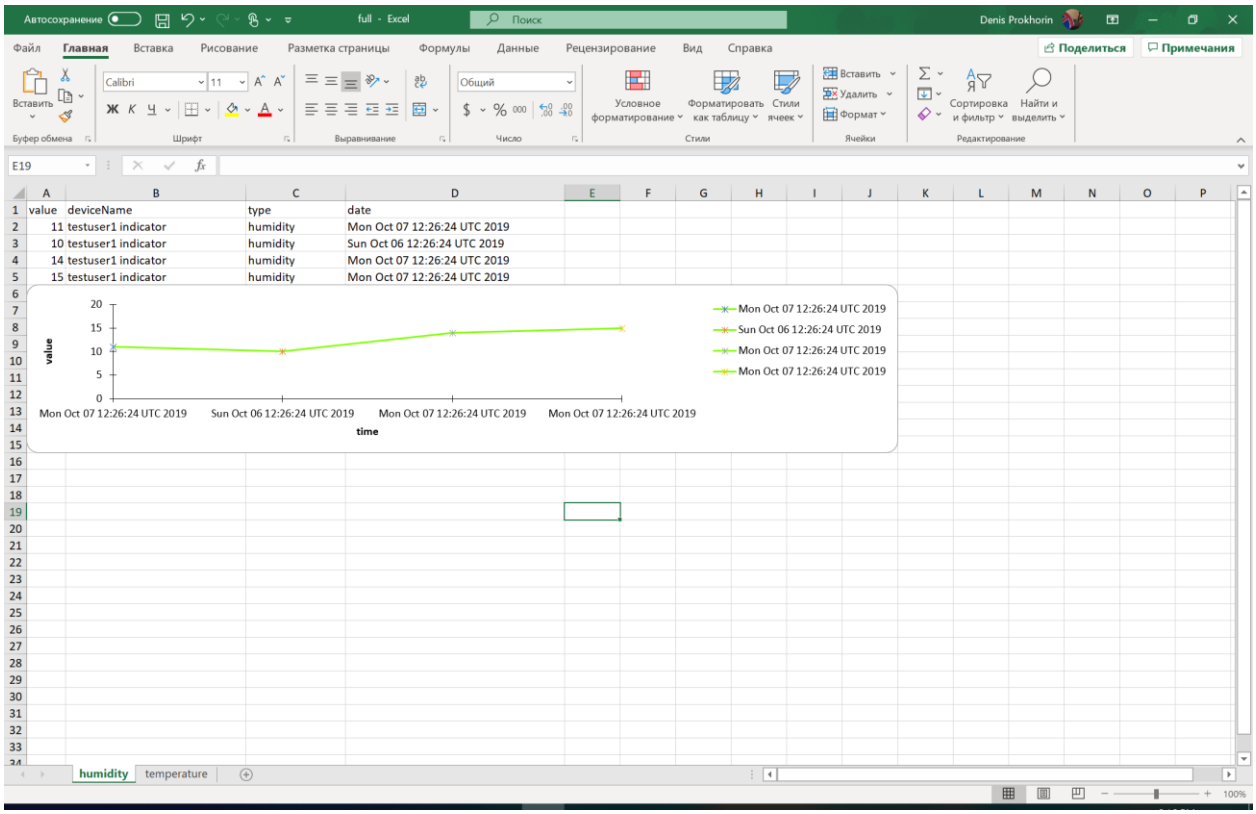


Рис 3.8 Звіт користувача

ApachePoi надає можливість взаємодії з файлами OLE2 і OOXML, має API низького рівня для надання підтримки OPC (із використанням openxml 4j), API високого рівня для електронних таблиць Excel.

ApacheTomcat — контейнер сервлетів, було створено компанією ApacheSoftwareFoundation. Повністю побудований та реалізований на мові Java із впровадженням специфікації сервлетів та JSP від Oracle, що являє собою стандарт, якого слід дотримуватися при проектуванні та розробці веб-додатків із

використанням мови програмування Java. Контейнер сер влетів надає інструментарій для запуску застосунків що були розроблені та спроектовані у контексті впровадження веб платформи, має певний перелік утиліт для можливості самоконфігурації.

Компоненти:

Tomcat версії 4.x був випущений з Jasper (перепроектований механізм JSP), Catalina (перепроектований контейнер сервлетів) і Coyote (стек HTTP).

- Catalina - контейнер сер влетів Tomcat'a впроваджує специфікацію сервлетів та JSP від Oracle, що являє собою стандарт, якого слід дотримуватися при проектуванні та розробці веб-додатків та, що використовується у Web-додатках та відповідає за можливість динамічного serverside рендерингу довільного web-контенту та забезпечує використання java бібліотек.
- Coyote - частина стеку HTTP Tomcat'a, який підтримує протокол HTTP 1.1, що використовує веб-сервери або контейнер додатків. Coyote прослуховує вхідні з'єднання, що надходять на певний TCP порту сервера, пересилає запити в механізм Tomcat для обробки запитів та надсилає відповідь назад клієнту.
- Jasper - технологія для використання JSP у Tomcat. Jasper конвертує JSP для подальшого їх використання як сервлетів наступним кроком копілюючиїхв Java код, в реалізації даної системи – не використовується.

RabbitMQ — додаток, який реалізує концепцію обміну повідомленнями між модулями програмного середовища на основі стандартів обміну повідомлень AMQP (AdvancedMessageQueuingProtocol). У створеній системі був використаний протокол повідомлень стандарту MQTT.

Основні можливості

- Забезпечує побудову кластерної архітектруи шляхом горизонтального маштабування
- Можливість зберігання інформації на диск

- Можливість взаємодії через HTTP протокол
- Представлено APIдля багатьох мов програмування
- Існують плагіни для розширення функціоналу (наприклад плагін підтримки MQTTпротоколу, який було використано від час розробки даної системи).

3.3 Алгоритмізація та розробка програмного забезпечення

3.3.1 Модуль виконання команд

Для моделювання даного модуля використовується мікроконтролер RobotDyn, який отримує команду від серверу завдяки брокеру повідомлень.

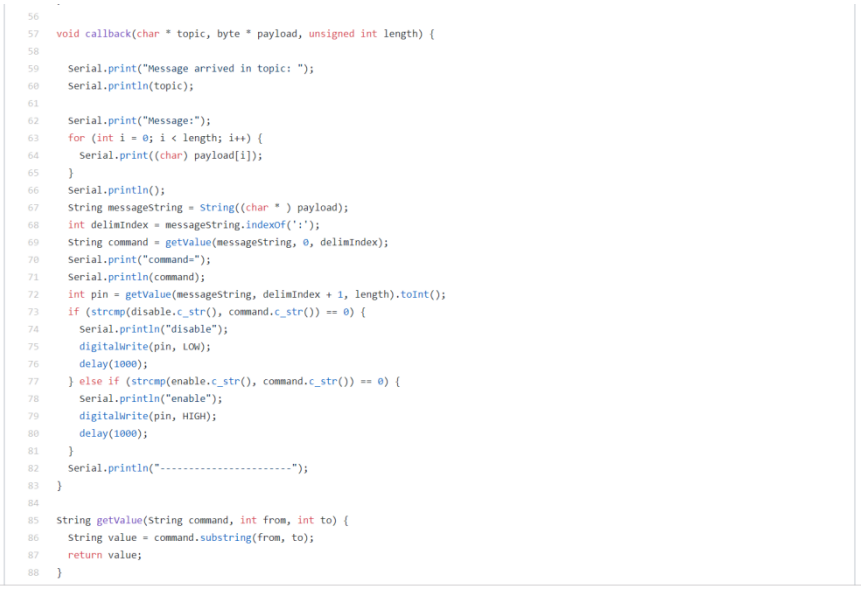
Відповідно до алгоритму. Назва команди та ІН пристрою надходять від клієнта у вигляді HTTP запиту на сервер React. Далі формується команда до Tomcat серверу із spring додатком (рис 3.9). Відбувається перевірка, чи може користувач, що відправив команду керувати даним пристроєм та чи належить дана команда пристрою. Якщо перевірка умов пройшла успішно - наступним кроком йде формування MQTT повідомлення, в топіку передається команда пристрою, якому відповідає ІН пристрою через брокер повідомлень (Рис.3.10). Мікроконтролер ESP8266, зчитує отриману команду та виконує її.

```
@PostMapping("/{command}/{userDeviceName}")
public ResponseEntity doCommand(@PathVariable String command, @PathVariable String userDeviceName, Principal principal) {
    UserDevice userDeviceData = defaultUserDeviceFacade.findModel(principal.getName(),userDeviceName);
    if (Objects.isNull(userDeviceData)) {
        return processingErrors(NO_RIGHTS, PERMISSION_TYPE_ERROR);
    }
    rabbitMqPublisher.send(format("%s/%s",ESP_COMMAND_TOPIC,userDeviceData.getUserDeviceId()),command,userDeviceData.getPir
    return ok().build();
}
```

Рис 3.9 Контролер що отримує команди клієнта.

```
public void send(String topic, String command, int pin) {
    try {
        String commandWithPin = String.format("%s:%d",command,pin);
        MqttConnectOptions options = mqttClientFactory.getConnectionOptions();
        MqttClient sampleClient = new MqttClient(url, login);
        options.setCleanSession(true);
        sampleClient.connect(options);
        MqttTopic mqtttopic = sampleClient.getTopic(topic);
        MqttMessage message = new MqttMessage(commandWithPin.getBytes());
        message.setQos(1);
        mqtttopic.publish(message);
        sampleClient.disconnect();
    }catch (Exception e){
        logger.error(e.getMessage());
    }
}
}
```

Рис 3.10. Генерація повідомлення



```
56
57  void callback(char * topic, byte * payload, unsigned int length) {
58
59    Serial.print("Message arrived in topic: ");
60    Serial.println(topic);
61
62    Serial.print("Message:");
63    for (int i = 0; i < length; i++) {
64      Serial.print((char) payload[i]);
65    }
66    Serial.println();
67    String messageString = String((char * ) payload);
68    int delimIndex = messageString.indexOf(':');
69    String command = getValue(messageString, 0, delimIndex);
70    Serial.print("command=");
71    Serial.println(command);
72    int pin = getValue(messageString, delimIndex + 1, length).toInt();
73    if (strcmp(disable.c_str(), command.c_str()) == 0) {
74      Serial.println("disable");
75      digitalWrite(pin, LOW);
76      delay(1000);
77    } else if (strcmp(enable.c_str(), command.c_str()) == 0) {
78      Serial.println("enable");
79      digitalWrite(pin, HIGH);
80      delay(1000);
81    }
82    Serial.println("----------------------");
83  }
84
85  String getValue(String command, int from, int to) {
86    String value = command.substring(from, to);
87    return value;
88  }
```

Рис 3.11. Метод зчитування повідомлення з брокеру повідомлень.

Для перевірки та моніторингу роботи RabbitMQ, адміністратору необхідно використовувати адміністративну панельRabbitMQ (рис. 3.12)

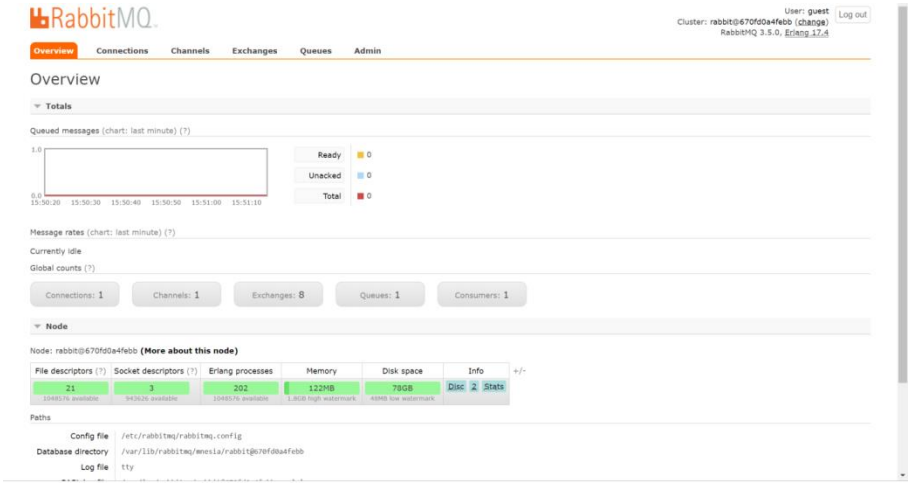


Рис. 3.12. Адміністративна панель

За допомогою адміністративної панелі, адміністратор має можливість моніторингу існуючих топіків (рис 3.13), а також відправлення повідомлень.

Рис 3.13. Моніторингтопіків брокера повідомлень.

3.3.2 Модуль відображення одягу в залежності від погоди

Відповідно до алгоритму (рис. 3.14) показники погоди у місті користувача надходить від OpenWeatherApi, далі йде пошук Look користувача на основі отриманих показників, якщо вони не були знайдені, йде пошук серед Look інших користувачів, які є публічними, та обираються самі популярні (рис 3.15).

Далі йде запис показника відповідного пристрою у БД. Наступним кроком йде перевірка отриманого значення (Рис. 3.15). У випадку, якщо значення показника вказують на небезпечну ситуацію, власнику надходить email оповіщення про критичну ситуацію у помешканні.

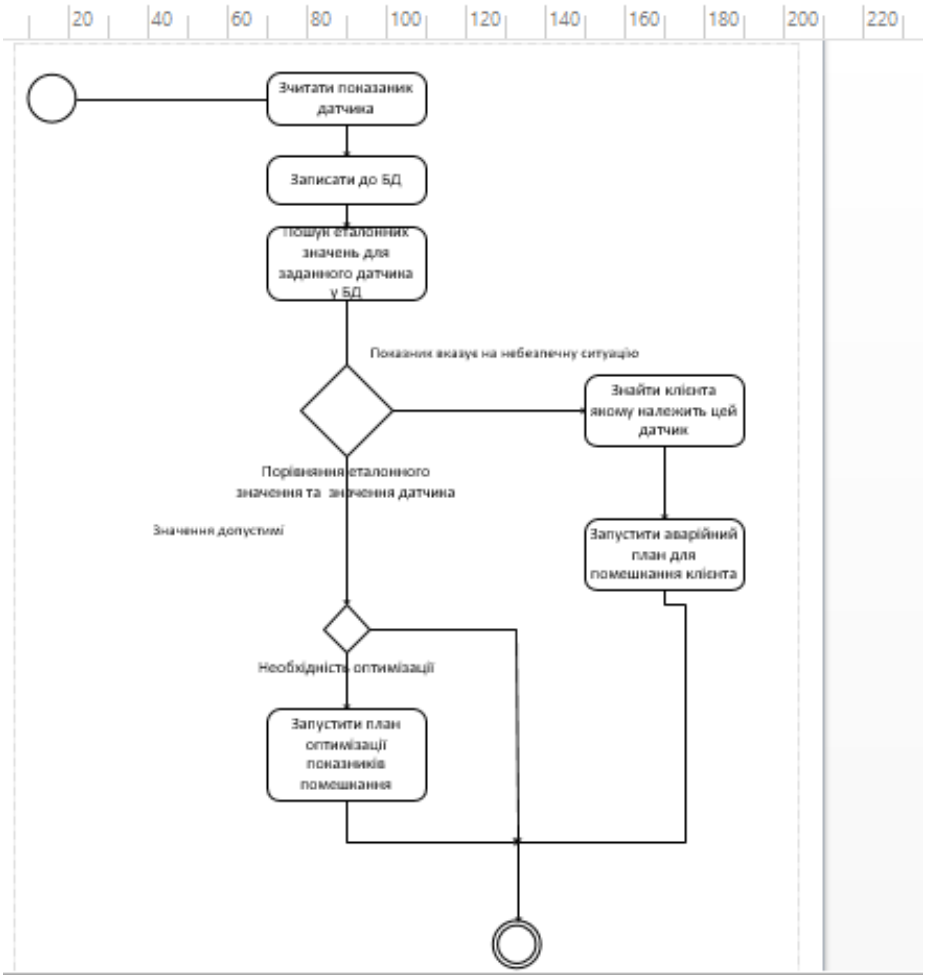


Рис. 3.14. Діаграма діяльності модулю перевірки датчиків

Рис 3.15. Отримання популярнихLooks.

3.3.3 Модуль формування звітів

Відповідно до опрацювання бізнес логіки значення необхідного періоду часу та ІН пристроїв клієнта надходять HTTP запитомдо системи. Наступними пунктами є пошук значень пристроїв за введений проміжок часок та формування excel звіту відповідно у форматі *.xls (рис 3.16 та 3.17).

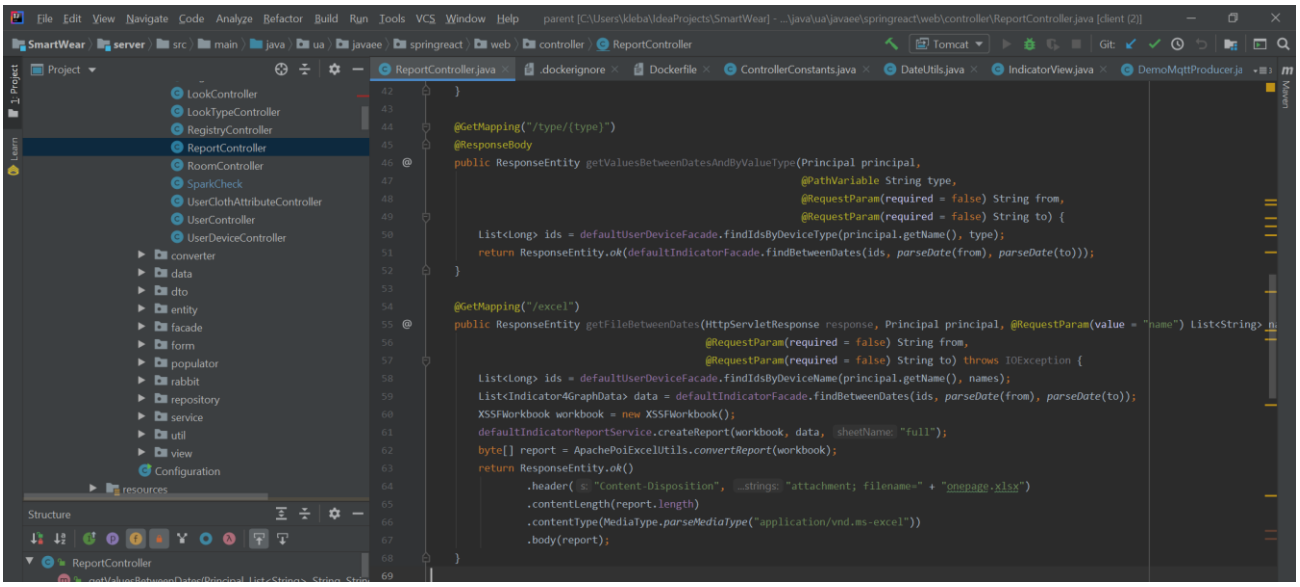


Рис 3.16. Формування звітів

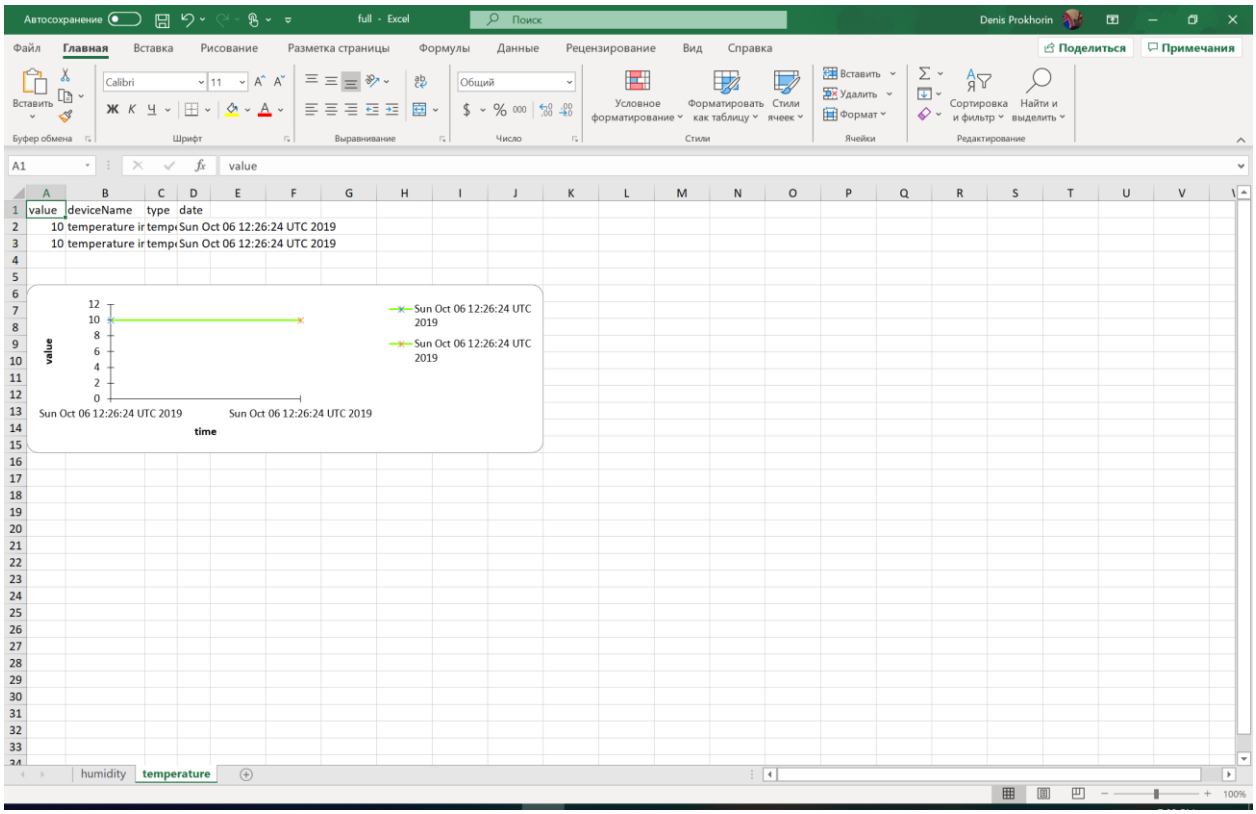


Рис 3.17. Excel звіт

3.3.4 Створення одягу користувача

У зв'язку із використанням у системі OpenWeatherapi, було вирішено розширити його функціонал від моніторингу погоди у місті до запровадження модулю прийняття рішень, що відповідає за поради який одяг надягнути користувачу у зв’язку із станом погоди у місті.надягнути під час певному стану клімату, у його місті.

Після того, як система була повністю запущена (рис 3.18), користувач має можливість авторизації (рис 3.19) Після проходження аутентифікації, користувач бачить головне меню, на якому він бачить температуру в його місті проживання (рис 3.20) та перелік вже створених Look (рис 3.21).

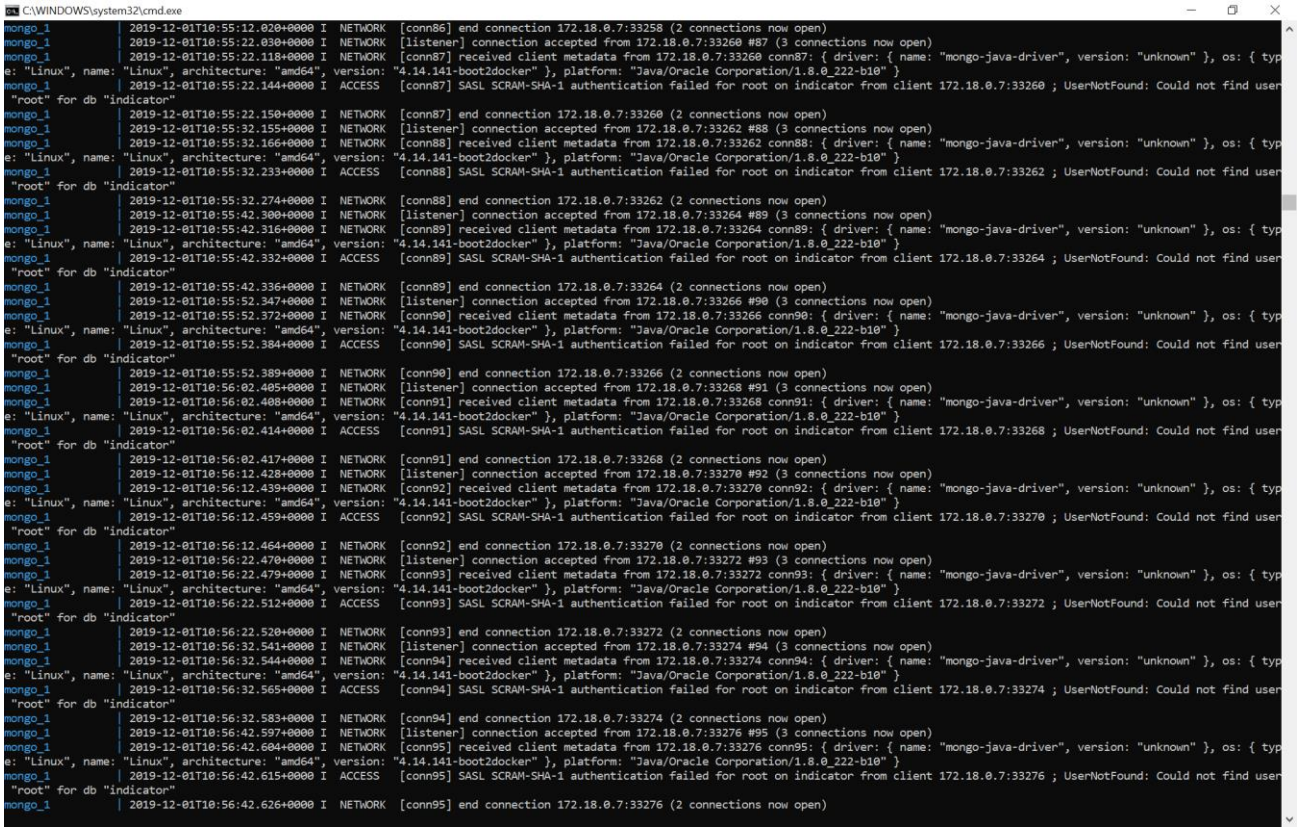


Рис 3.18. Логи запущеної системи

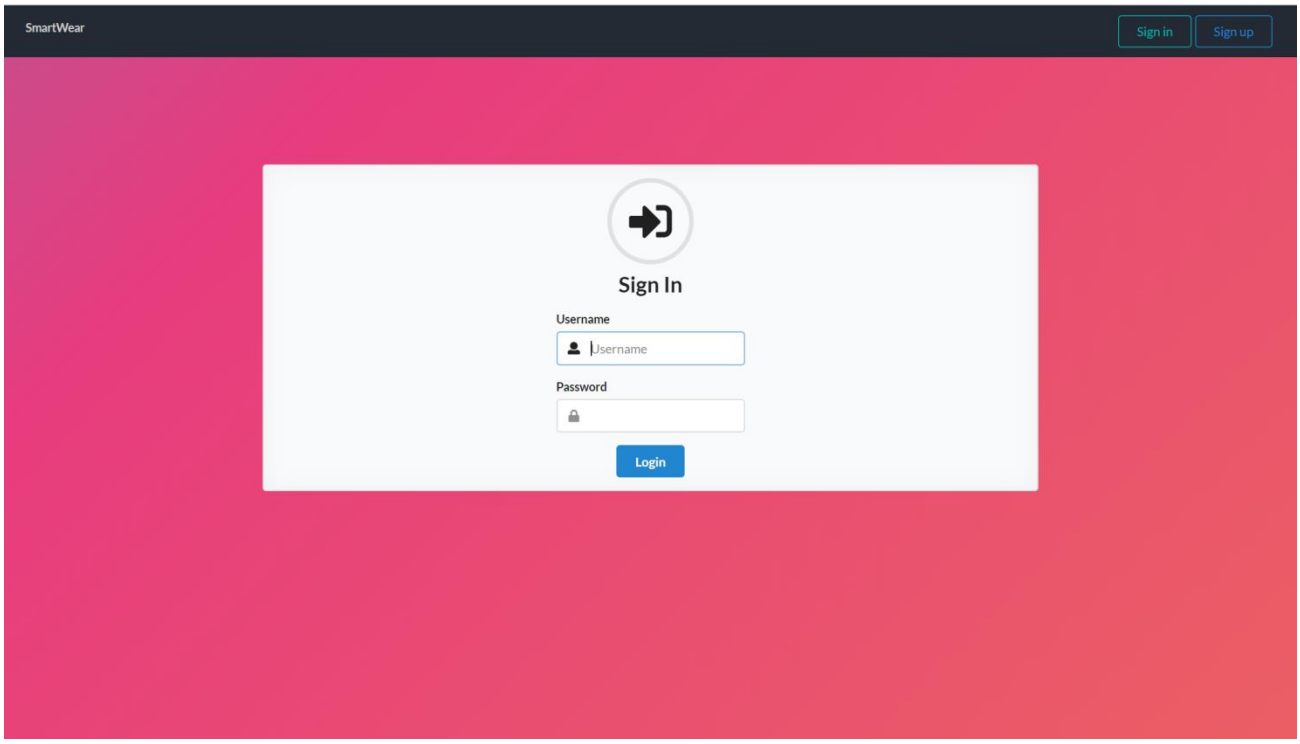


Рис 3.19. Вікно авторизації

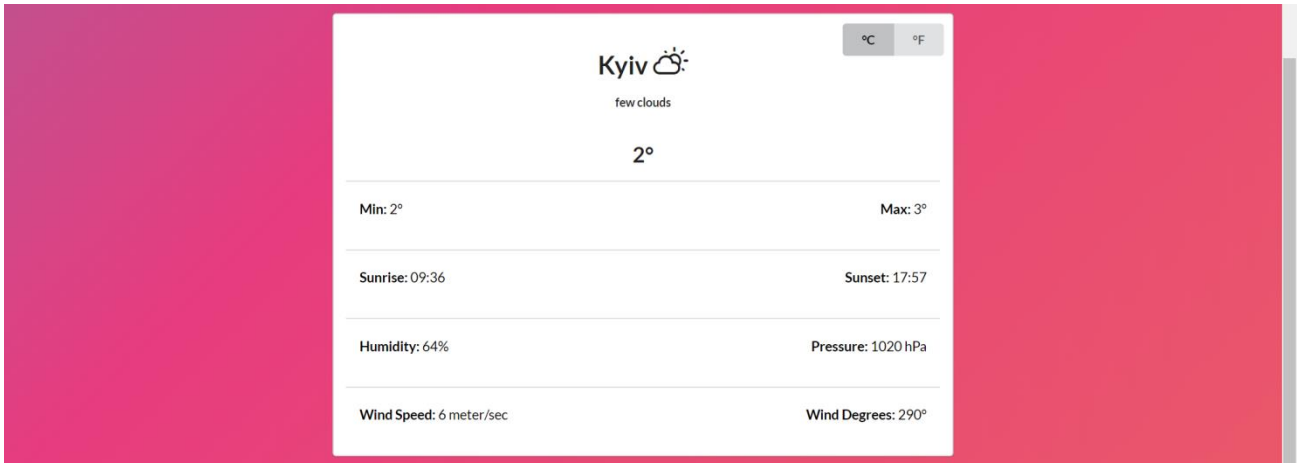


Рис 3.20. Вікно авторизації

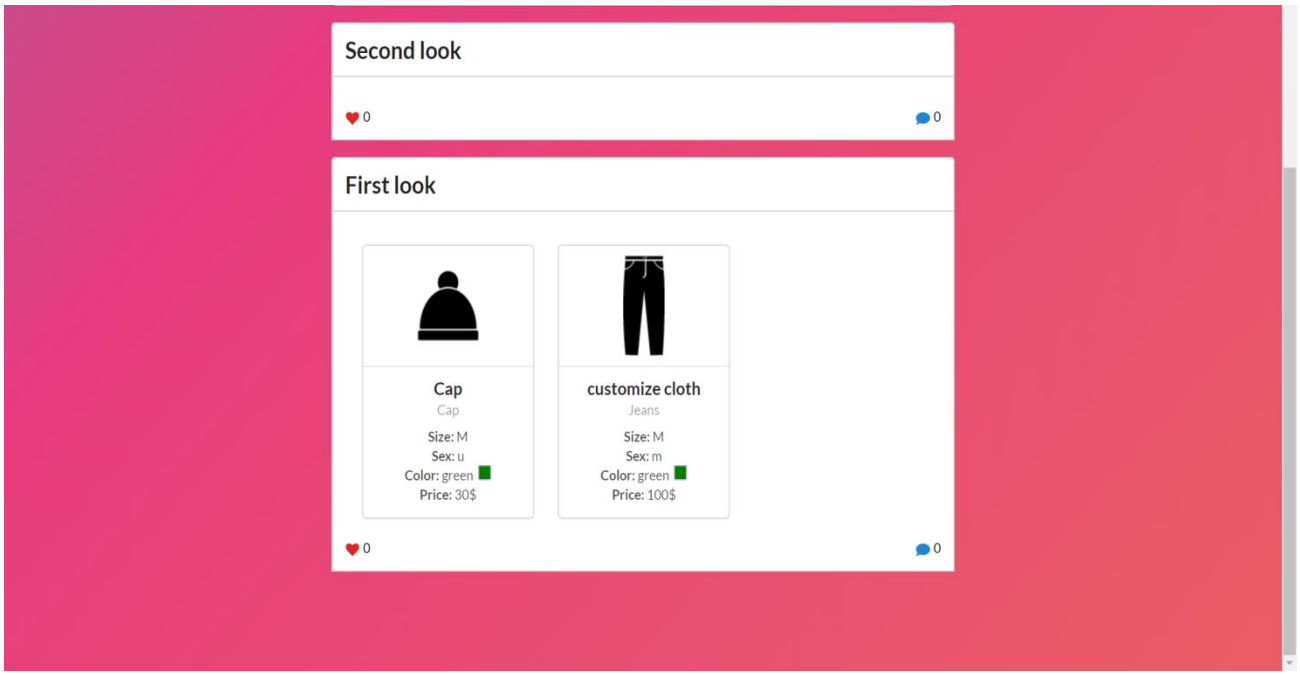


Рис 3.21. Створені луки користувача

Якщо користувач хоче додати новий одяг, йому необхідно обрати вкладку Cloth та створити його (рис 3.22). Одяг можна створити лише за тими типами, що вказав адміністратор. Після того як одяг було створено, його буде відображено на сторінці із речами користувача (рис 3.19). У випадку, коли користувач допустив помилку під час створення одягу ,він може його редагувати (рис 3.20) або видалити (рис. 3.21)

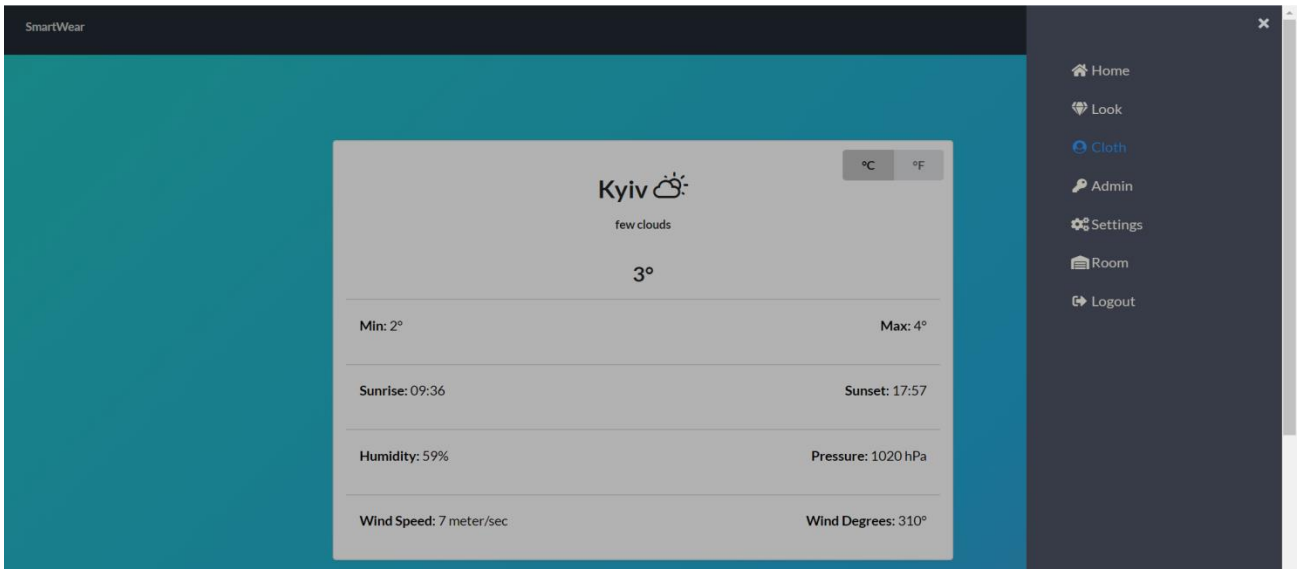


Рис 3.22. Меню системи

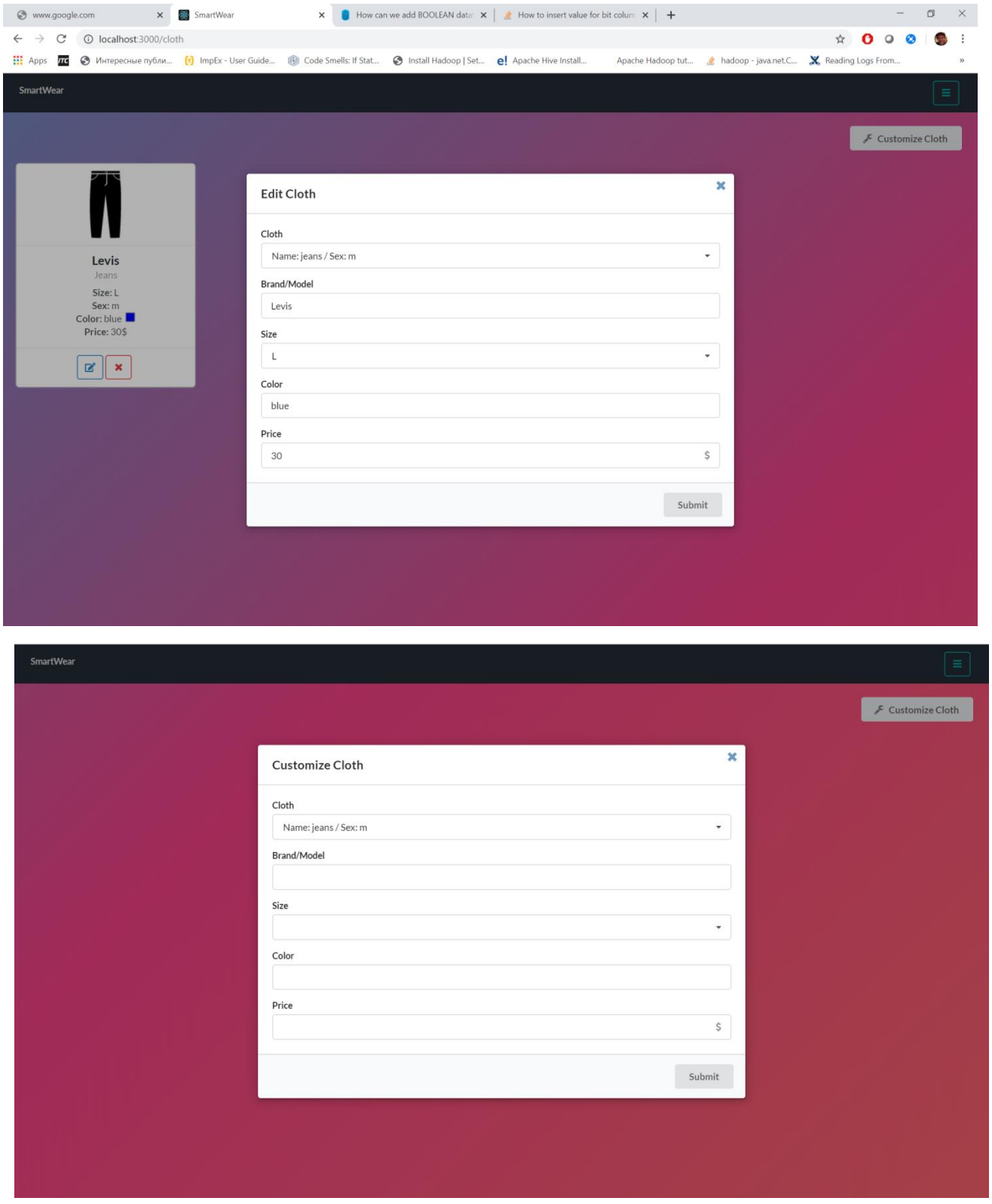


Рис 3.23 Створення нового одягу користувача

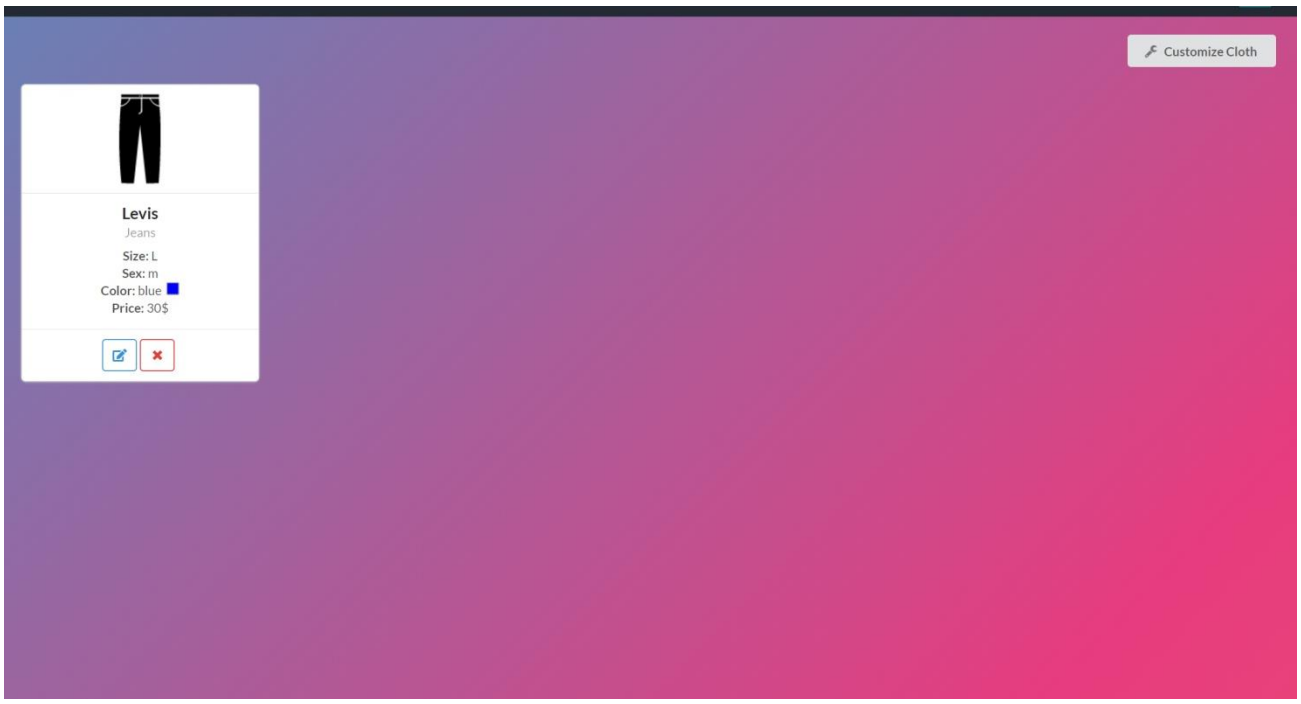


Рис 3.24. Перелік речей користувача

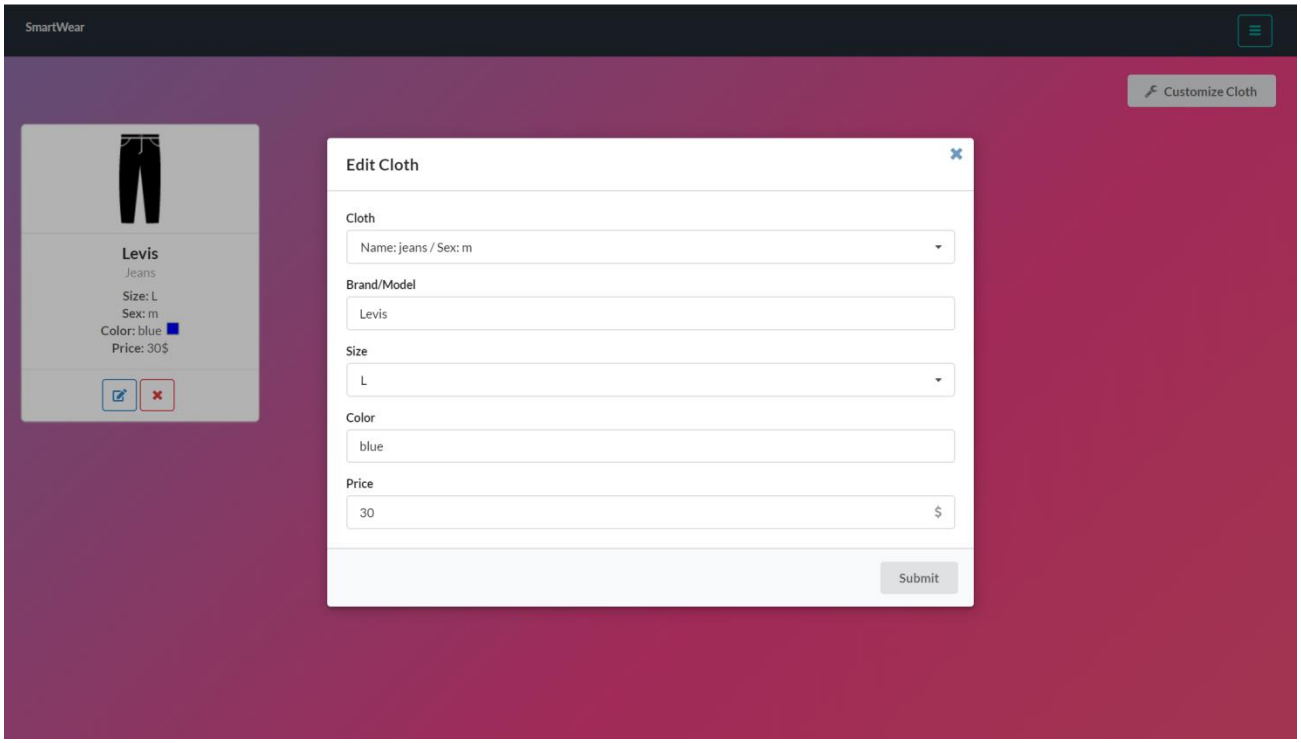


Рис 3.25. Редагування одягу

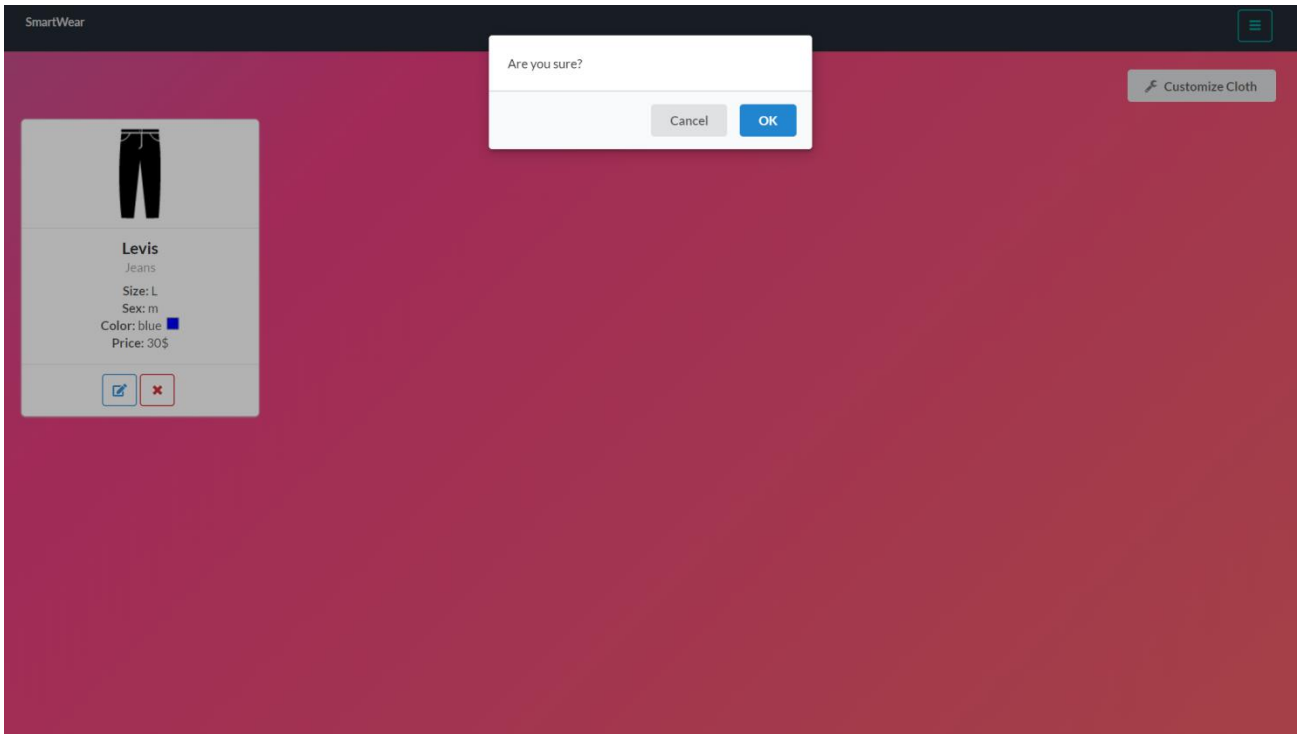


Рис 3.26. Видалення одягу

3.3.5 Створення Look користувача

Look – це перелік одягу користувача, із певними типами місць , куди цей look можна одягнути, якщо користувач хоче його створити ,йому необхідно заповнити форму (рис.3.27)

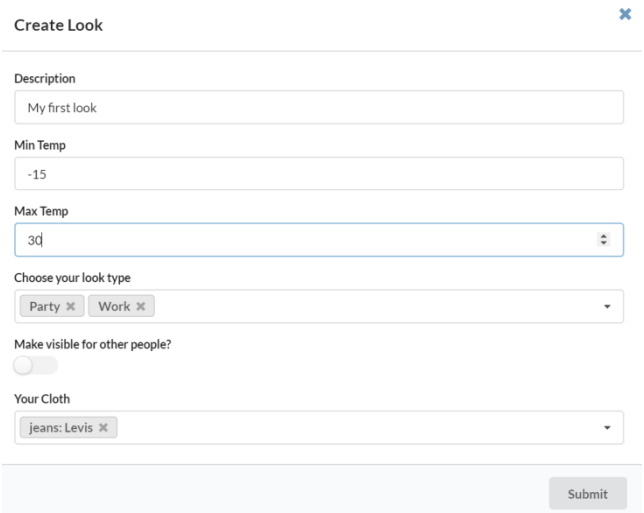


Рис 3.27. Форма створення Look

Після того як look було створено , користувач зможе побачити його у своєму списку створених looks (рис 3.28)

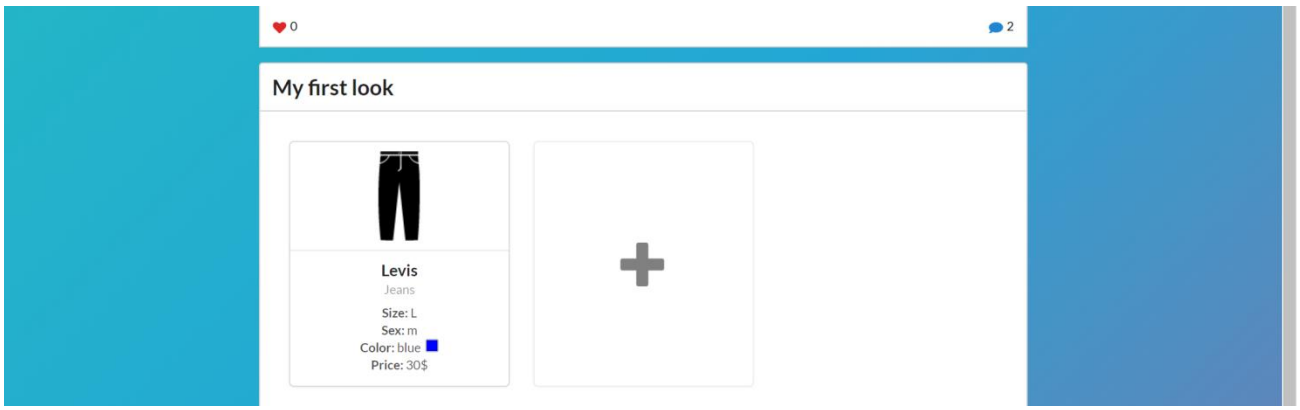


Рис 3.28 Створений Look

Якщо look було створено з можливістю перегляду іншими користувачами , вони можуть його побачити.

3.3.5 Можливості адміністратора системи

Меню адміністратора представлено на рис 3.29.

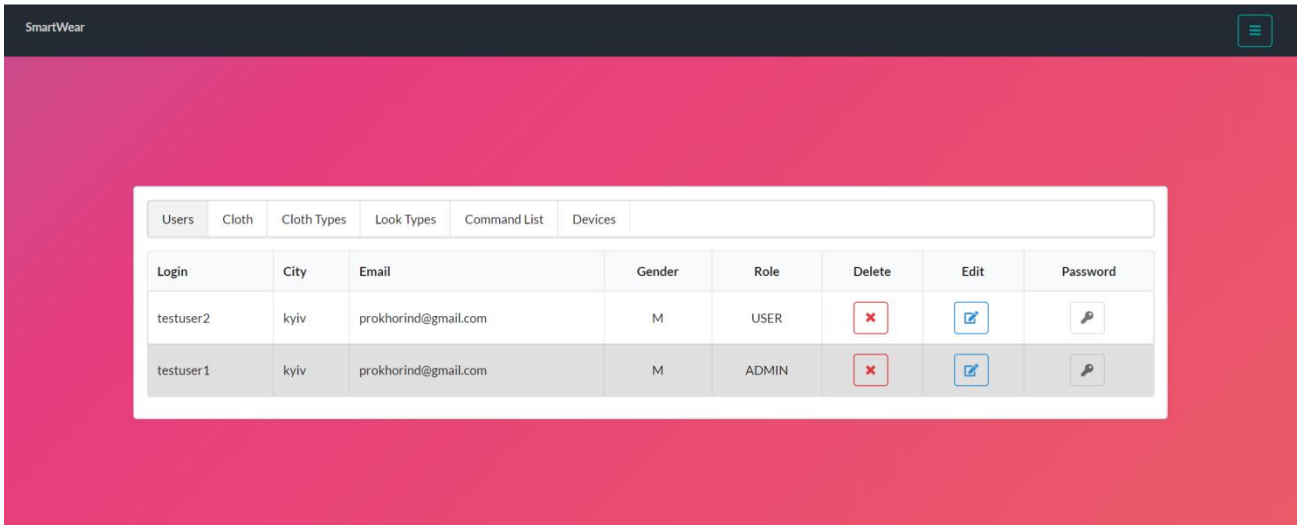


Рис. 3.29 Меню адміністратора

Адміністратор має можливість редагувати дані користувача (рис 3.30)

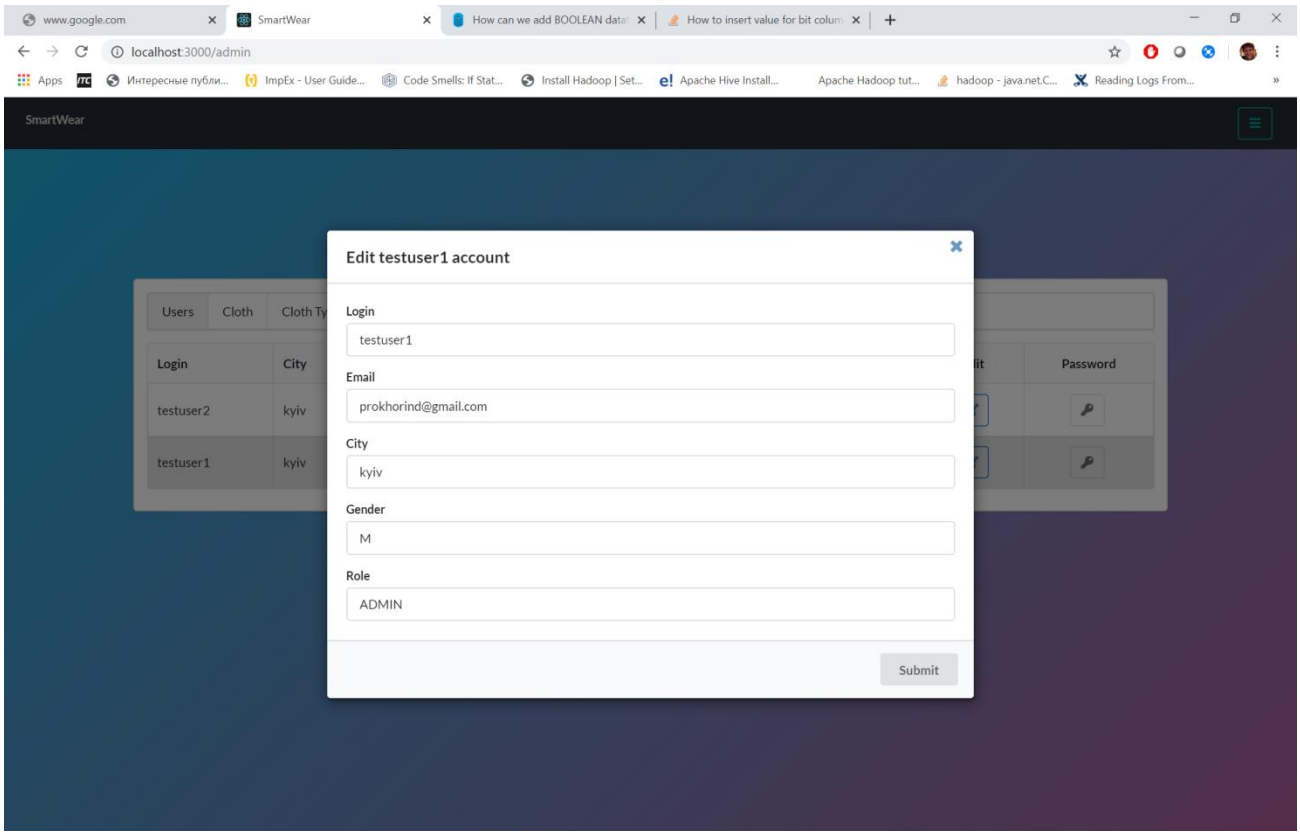


Рис. 3.30 Меню адміністратора

Створення та видалення видів одягу (Рис. 3.30) і опис основних типів одягу (рис 3.31)

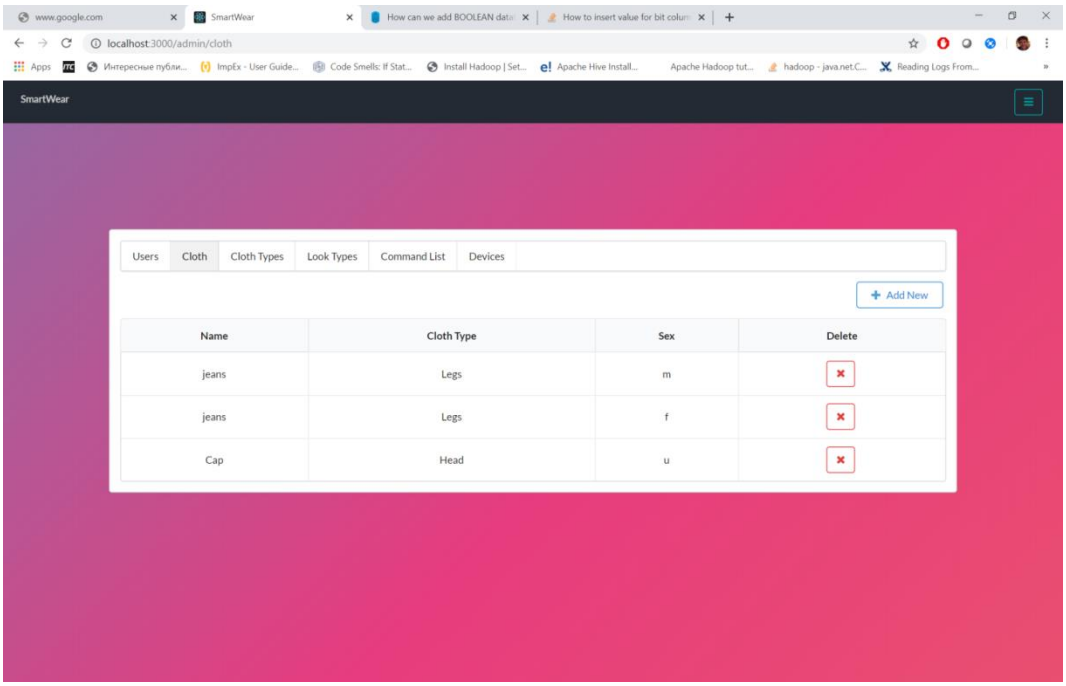


Рис 3.31 Створення та видалення видів одягу

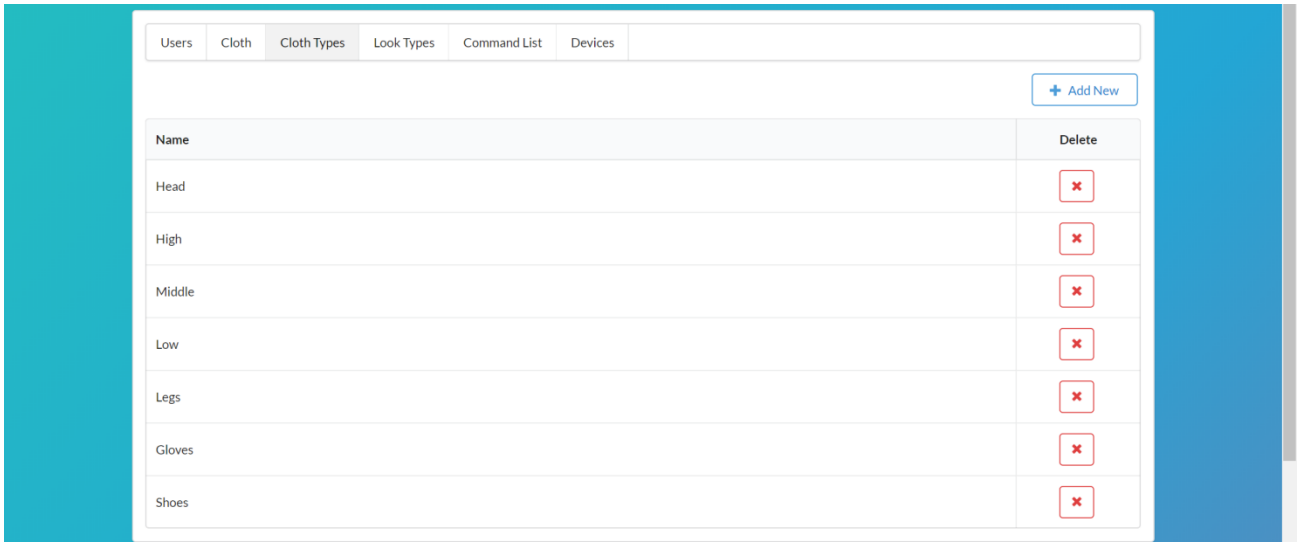


Рис 3.32 Типи одягу

Створення та видалення видів look (рис. 3.33)

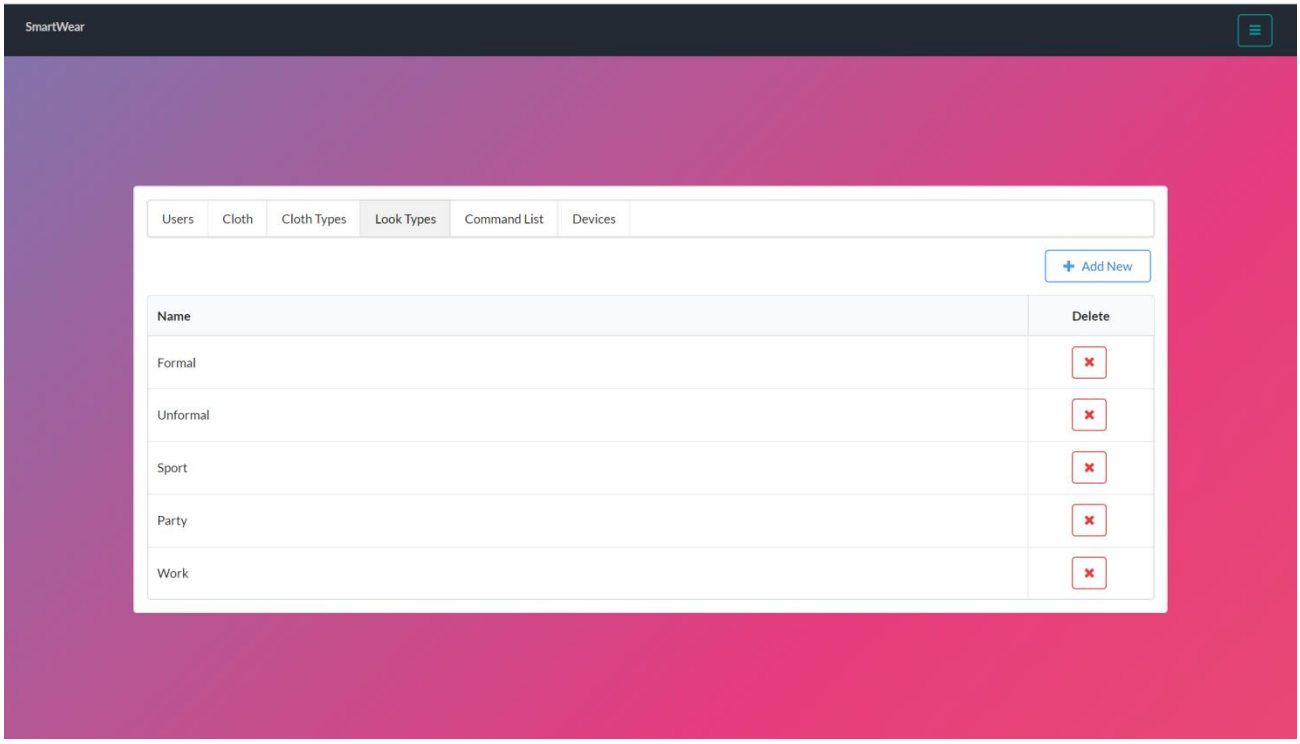


Рис. 3.33. CRD операції із типами Look

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Для тестування API продукту було використано Postman утіліту:

Отримання значень показників в існуючій кімнаті(рис 4.1).

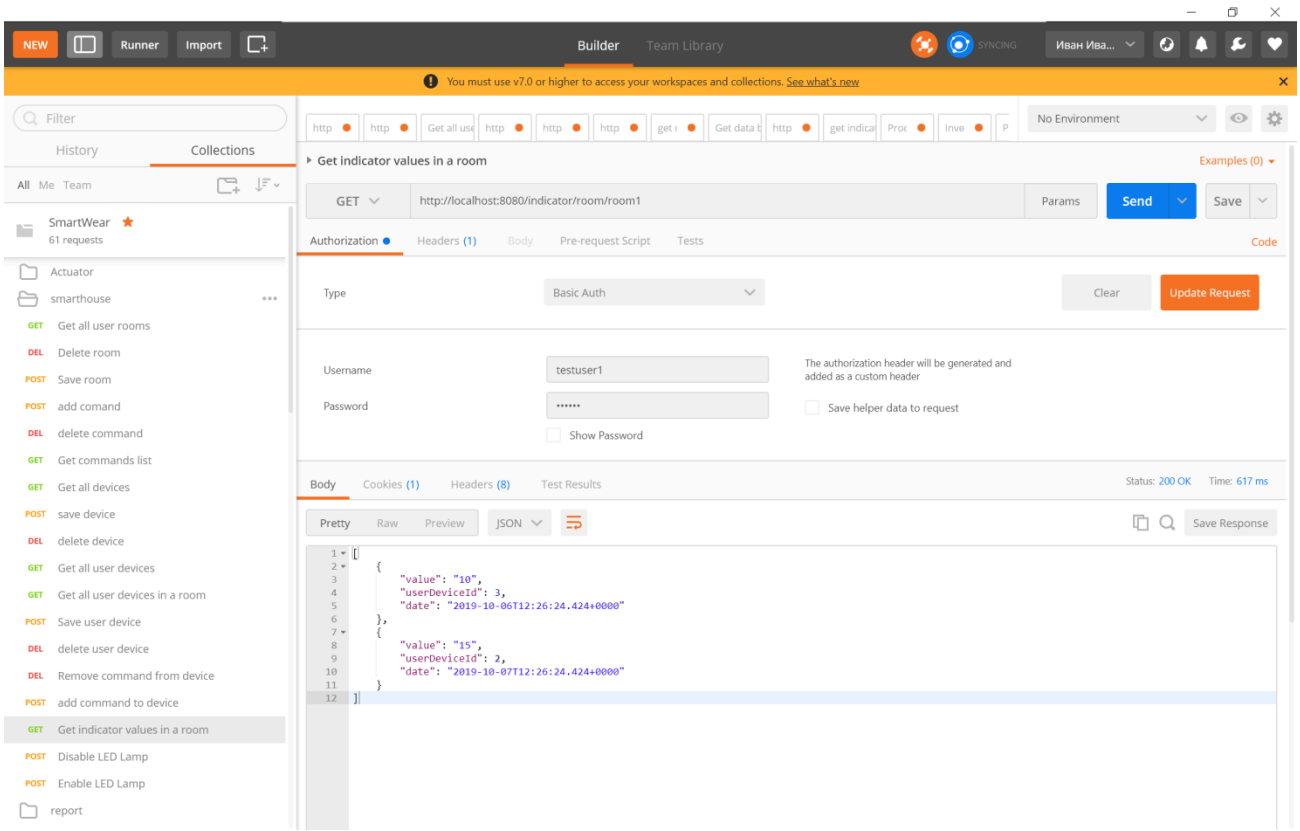


Рис 4.1 Отримання показників

Отримання погоди у місті користувача з допомогою Open Weather Api (рис.4.2):

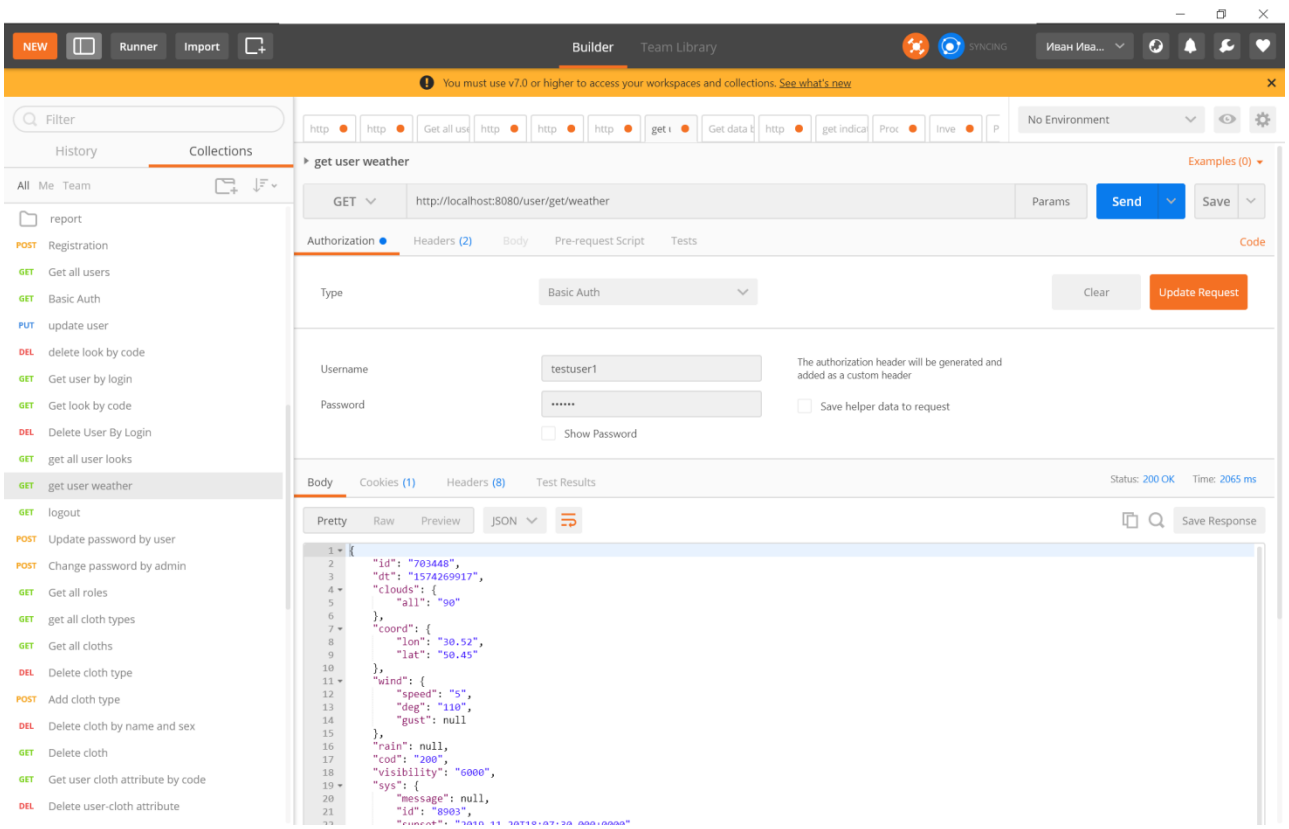


Рис 4.2. Отримання погоди

Спроба отримати показники в неіснуючій кімнаті (рис. 4.3)

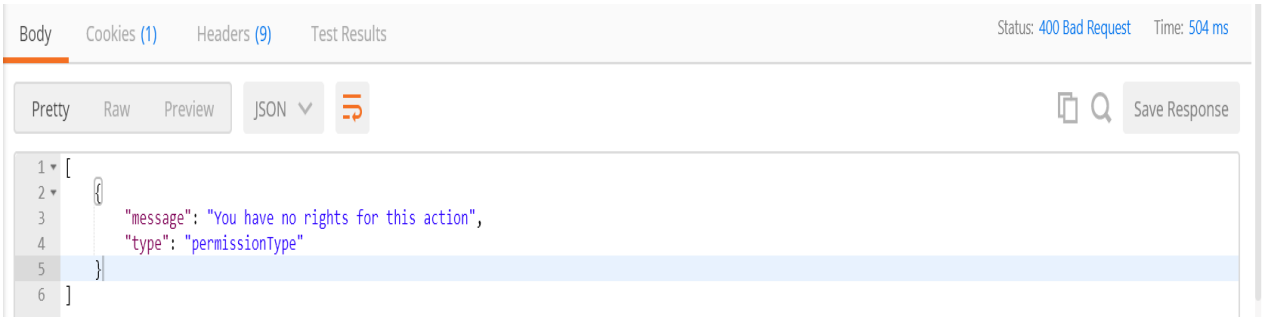


Рис 4.3. Відповідь сервера на спробу отримати показники в неіснуючій кімнаті

Спроба видалити неіснуючу кімнату (рис 4.4)

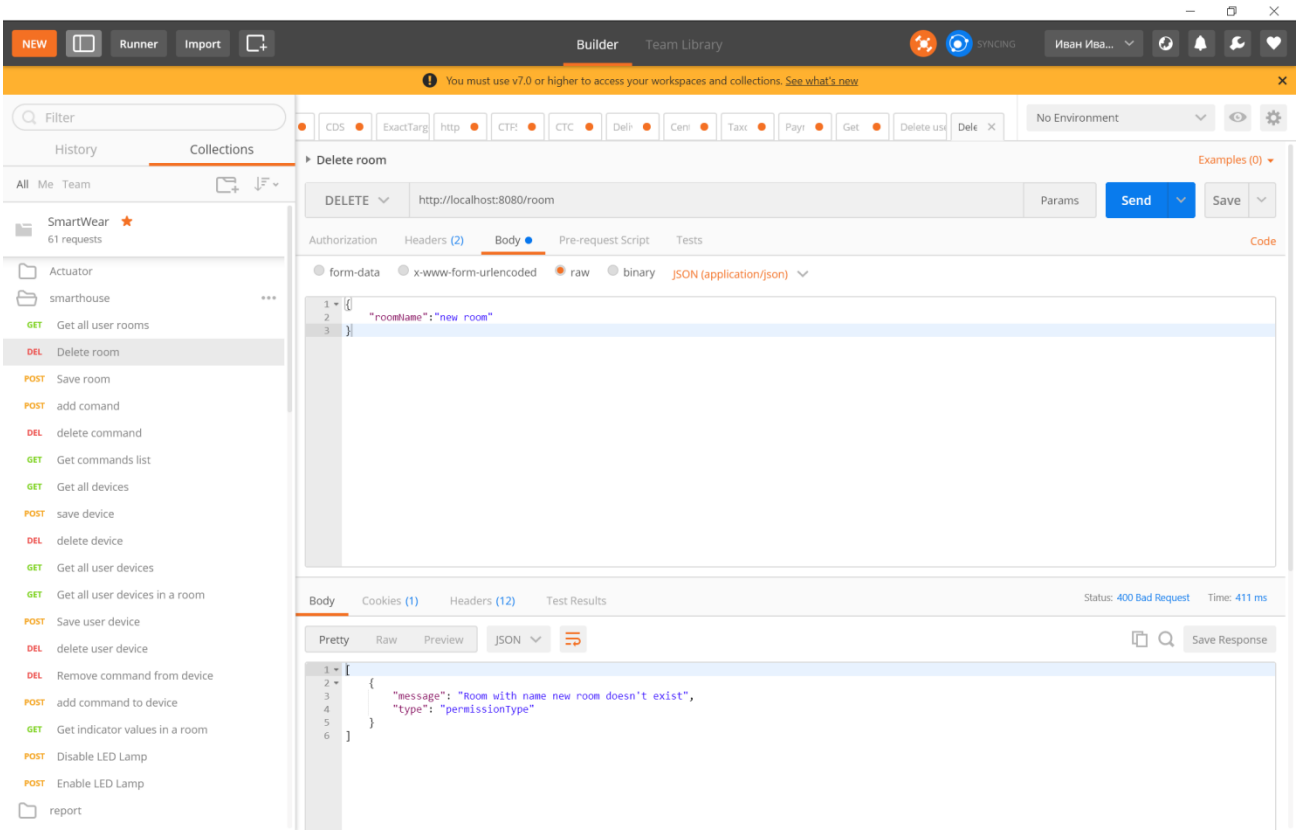


Рис 4.4 Видалення кімнати

Отримання топ 3 наборів одягу для користувача рис 4.5

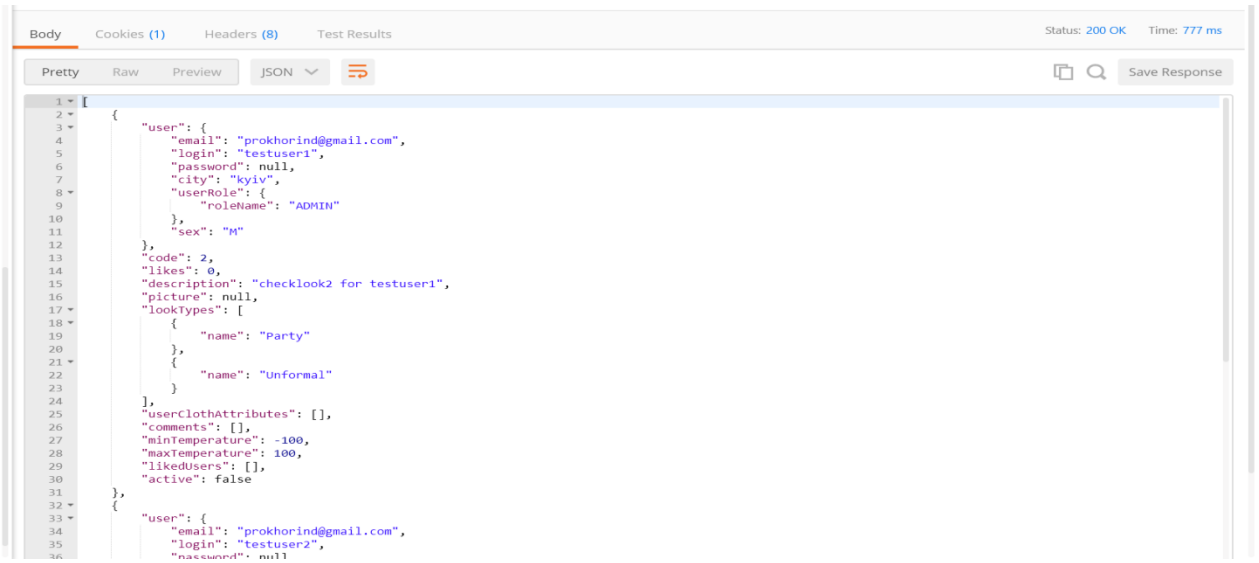


Рис 4.5 Отримання топ 3 наборів одягу

Спроба додати вже існуючий одяг в створеному наборі (рис. 4.6)

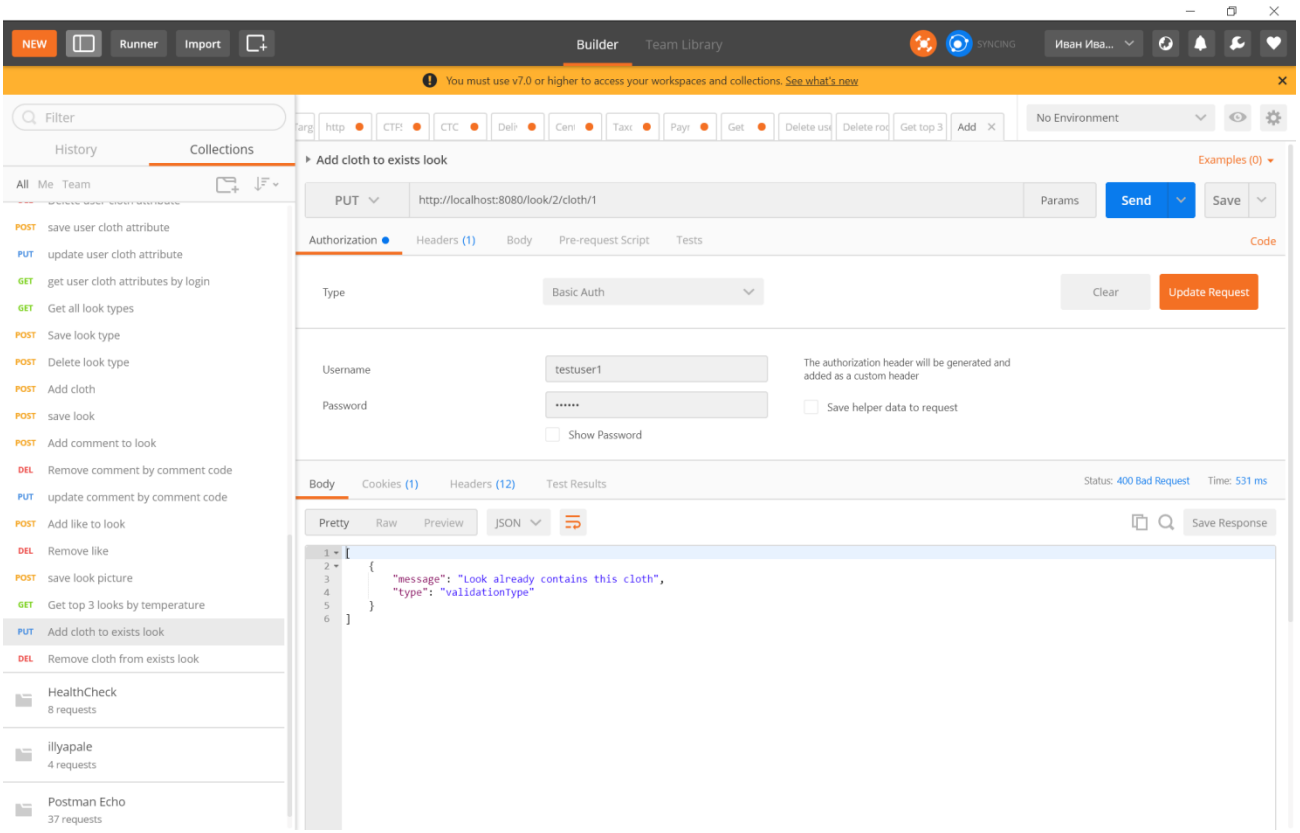


Рис 4.6. Помилка під час додавання вже існуючого одягу

Валідація нового паролю під час зміни (рис 4.7)

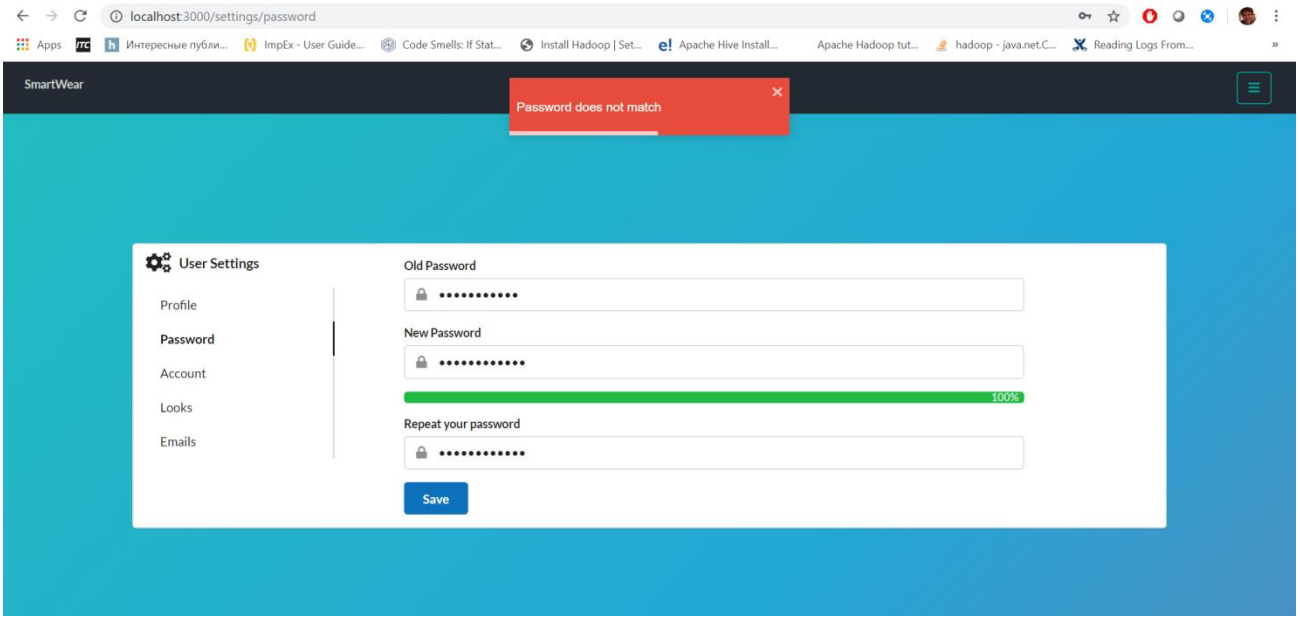


Рис 4.7 Перевірка паролю

Спроба ввести неправильно команду під час використання телеграм боту (рис. 4.8)

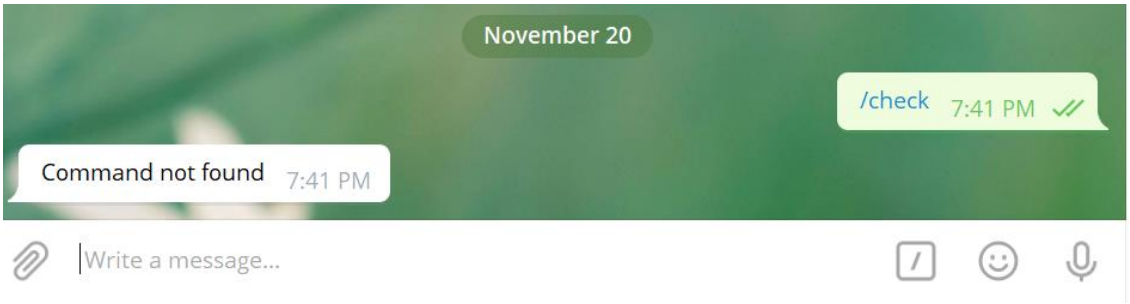


Рис. 4.8 Валідація команд під час використання телеграм боту

4.2Вимоги до апаратного забезпечення

4.2.1 Вимоги до апаратного забезпечення сервера

- Для того, щоб забезпечити діяльность впровадженої системи зі сторони серверної частини слід застосовувати сервери моделі “IBM CICS/390” або “HP ProLiant 10” (4Гб ОЗУ та більше, 240Гб постійної пам’яті з швидкістю читання від 100Мбіт/с). За умови мінімальної кількості користувачів.

4.2.2 Вимоги до апаратного забезпечення клієнту користувача

Для пристрою користувача мінімальними є наступні вимоги до системи:

- Пристрій з можливістю використання браузеру з підтримкою HTML5
- Пристрій з можливістю встановлення телеграм додатку

4.2.3 Вимоги до апаратного забезпечення користувача

- Можливість підключення до брокеру повідомлень
- Використання MQTTпротоколу з передачею команд , що підтримує розроблена система

4.3 Вимоги до програмного забезпечення

4.3.1 Вимоги до системного програмного забезпечення

Для запуску сервера потрібно встановити наступне програмне забезпечення:

- Операційна система яка підтримує можливість встановлення docker

Для використання системи зісторони клієнта необхідно та достатньо будь-яку з останніх версій ОС наприклад: Windows 10;Ubuntu 18.04, Mac OS10.13. Та налаштувати взаємодію за допомогою оного із web-браузерів: GoogleChrome,Chromium, MozillaFirefox,Safarita інші.

4.4 Склад інсталяційного пакету для встановлення розробленої системи

1. client
2. server
3. bot
4. rabbitmq
5. Sketches
6. .gitignore
7. docker-compose.yml
8. pom.xml
9. readmemd

- Файли для відображення інтерфейсу користувача:
  - Client/public/*
- Файл докер образу
  - Client/Dockerfile
- Файли що відповідають за стилізацію сторінок користувача:
  - /client/src/styleForComponents/*
- Файли для клієнтської обробки сторінок та динамічного відображення:
  - /client/src/actions/*

    - /client/src/components/*
    - /client/src/img/*
    - /client/src/service/*

- Файли що забезпечують конфігурацію серверних копонентів:
    - server/src/main/java/ua/javaee/springreact/config/*
    - server/src/main/resources/application.properties
    - server/src/main/resources/data.sql
    - server/src/main/resources/log4j2.xml
    - server/src/main/resources/weather.properties
- Файли що, забезпечують взаємодію користувача із системою:
    - server/src/main/java/ua/javaee/springreact/controller/*
- Файли що відповідають за прошарок фасадів
    - server/src/main/java/ua/javaee/springreact/facade/*
- Файли , що відповідають за прошарок сервісів
    - server/src/main/java/ua/javaee/springreact/service/*
- Файли взаємодії системи із базою даних:
    - server/src/main/java/ua/javaee/springreact/repository/*
- Файли конвертації об’єктів
    - server/src/main/java/ua/javaee/springreact/converter/*
    - server/src/main/java/ua/javaee/springreact/populator/*
- Класи представлення таблиць баз даних
    - server/src/main/java/ua/javaee/springreact/entity/*
- Допоміжнікласи
    - server/src/main/java/ua/javaee/springreact/utils/*

4.5 Емуляція відправлення показників від esp8266

У зв’язку з тим , що для відправлення показників датчиків досистеми, було використано брокер повідомлень, що відправляє дані

використовуючиMQTTпротокол , для біль ефективноговідлагодження програмного забезпечення було створеноJavaзастосунок , завданням якого було запроводаження емуляції взаємодії системи та esp8266

Повідомлення що відправляється налічує у собі ІН пристрою та значення показника. та відправляються у спеціальний топік брокеру повідомлень (рис. 4.9).

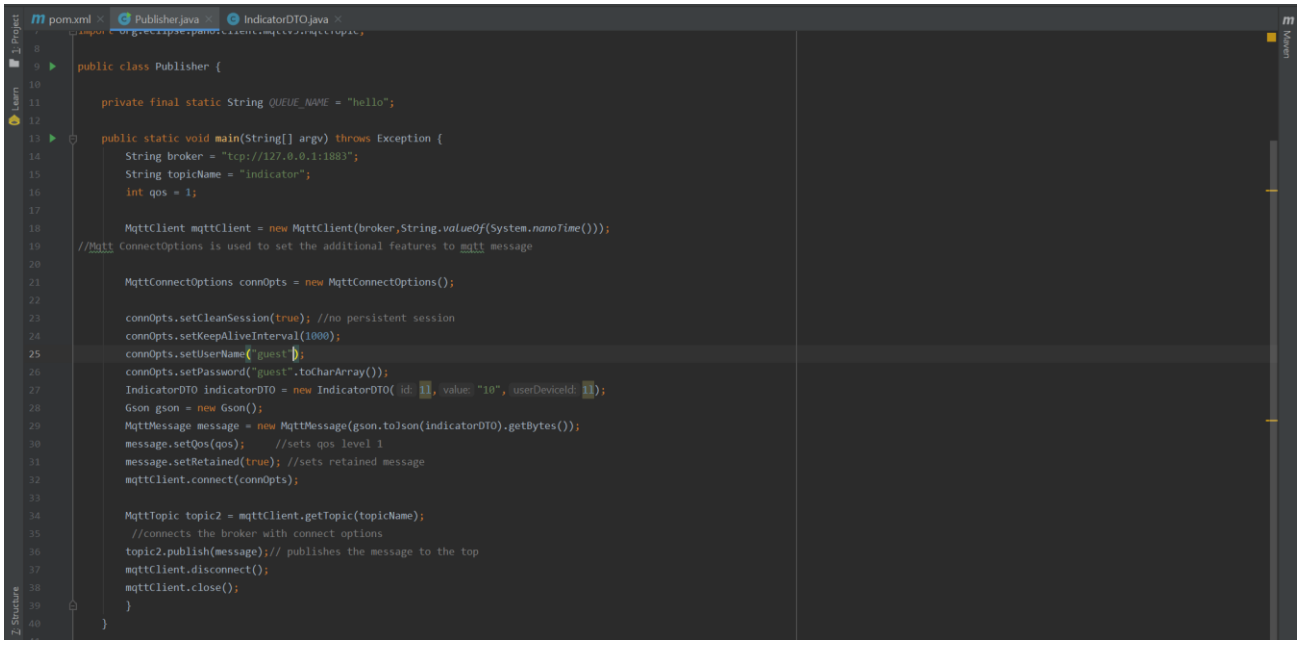


Рис. 4.9 Post запит до серверу

Після того , як повідомлення, опинилося у брокері який в свою чергу зчитує система,наступним кроком йде збереження даних до БД. (рис. 4.10)

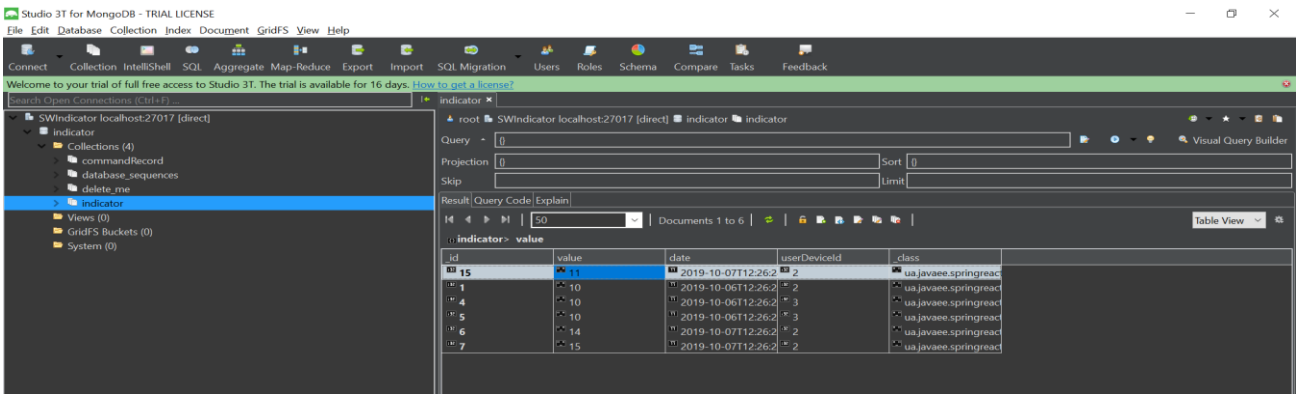


Рис . 4.10 Збереження даних до БД.

4.6 Емуляція пристрою отримання команд

Під час надсилання команд керування пристроями до пристроїв виконання також було використано брокер повідомлення, який відповідає за зберігання та надсилання повідомлень використовуючи MQTT протокол, для відлагодження програмного забезпечення було створено Java додаток, завданням якого було запроводаження емуляції взаємодії системи та esp8266

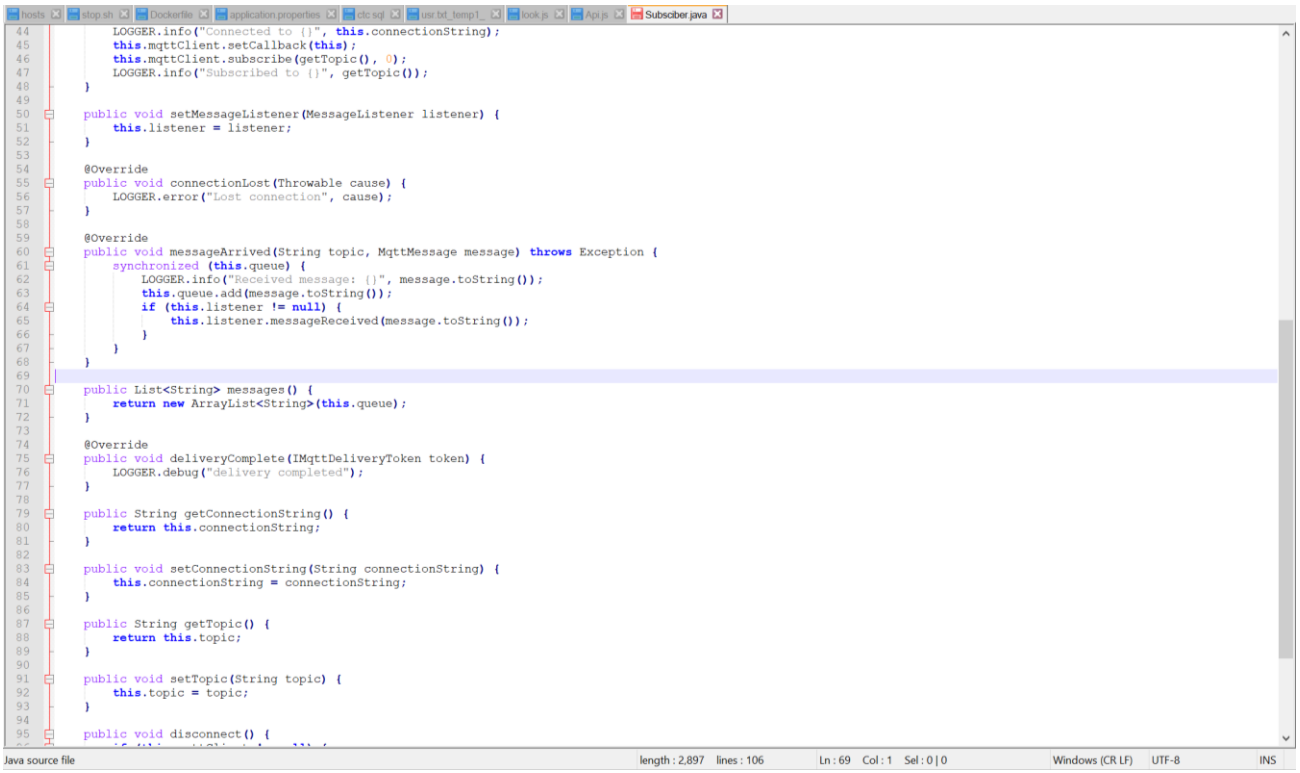


Рис. 4.11. Емуляція отримання команди пристроєм

4.7 Емуляція відправлення команди від серверу

У зв’язку з тим, що для впровадження функціоналу надсилання команд від системи до пристроїв виконання було використано брокер повідомлень, який використовує MQTT протокол під час надсилання повідомлень для налагодження взаємодії апаратного забезпечення із системою було використано

веб інтерфейс брокеру повідомлень rabbitmq, який надає функціонал для ємуляції відправлення команди від серверу (рис. 4.12).

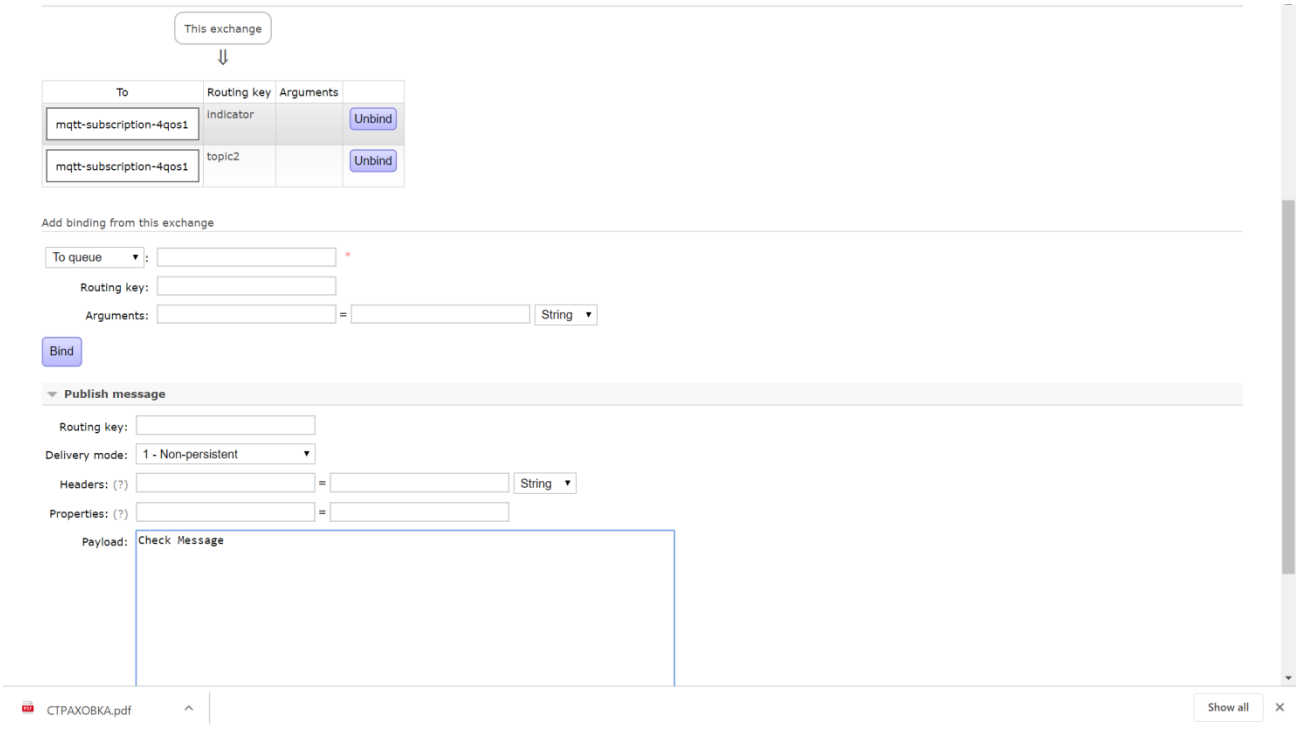


Рис. 4.12. Емуляція відправлення командивід пристрою

ВИСНОВКИ

Проаналізувавши існуючі технології систем автоматизації, було обрано необхідний стек для реалізації модифікації розробленої системи яка забезпечить покращення виконання спектра послуг, що необхідно для впровадження нових рішень та оптимізації роботи додатку.

Було модернізовано інформаційну систему для впровадження нового функціоналу прийняття рішень та оптимізовано процес моніторингу даних та керуванням пристроями, що підключені системи.

Модифікація ІС забезпечила керування пристроями на відстані із використанням нового протоколу, створеного для взаємодії між інтернет речами, було запроваджено модернізацію web інтерфейсу у вигляді переходу від jsp технології до повноцінного web додатку із використанням React, створено альтернативу web інтерфейсу для мобільних пристроїв на основі телеграм боту, було оптимізовано процес розгортання системи за рахунок використання докер контейнерів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ГрингардСэмюэлИнтернетвещей. Будущее уже здесь / СэмюэлГрингард , 2017
2. Портал Хабрахабр [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/259243/
3. Kevin Ashton. That ‘Internet of Things’ Thing. In the real world, things matter more than ideas. (англ.). RFID Journal (22 June 2009).
4. Портал Cisco [Электронный ресурс]. – Режим доступа: https://www.cisco.com/c/ru_ru/solutions/internet-of-things/overview.html
5. MySQLdocumentation [Електронний ресурс]. “MySQL.com” (Oracle 2018). Режим доступу: https://dev.mysql.com/doc/
6. Романовский Владимир Алексеевич Самоучитель по созданиюWeb-страниц: HTML, Dynamic HTML.- К.: А.С.К., 2002.- 472с.
7. Ноутон Патрик, ШилдтГербертJava 2.- СПб.: БХВ-Петербург, 2001.- 1072с.
8. Кит В. Росс. Компьютерные сети. Многоуровневая система интернета. 2-е издание. 764 с
9. Хорстманн Кей С., КорнеллГари JAVA 2. Библиотекапрофессионала.- М., Спб., К.: Вильямс, 2006.-
10. Портал Хабрахабр [Електронний ресурс]. https://habr.com/post/259243/
11. Портал Хабрахабр [Електронний ресурс]. https://habr.com/post/307668/
12. Чмирь, И. А. Объектно-ориентированноемоделирование / И. А. Чмирь, А. Ф. Верлань – Одесса: НАДУ. – 2005. – 243 с.
13. Пирогов, В. Інформаційні системи і бази даних: організація та проектування: Навчальний посібник / В. Пирогов. - СПб.: BHV, 2009. - 528 с.
14. Лаврищева, Е. М. Методыпрограммирования: теория, инженерия, практика /

Е. М. Лаврищева. – К.: Наукова думка. – 2006. – 451 с.

15. Гослинг Д. ЯзыкпрограммированияJava. СПб.: Питер, 1997 – 304 с
16. В теоретичне програмування. А. П. Ершов. Головна редакція фізико-математичної літературивид-ва «Наук», М., 1977Львів
17. М. С., Співаковський О. В. Вступ до об’єктно-орієнтованого програмування: Навчальний посібник. – Херсон: Айлант, 2001. – 210 с.: іл.
18. Шмуллер Д. Освойсамостоятельно UML за 24 часа. Издательскийдом «Вильямс», 2005 – 416 с.
19. Гради Буч. Объектно-ориентированноепрограммирование. 514 с.
20. Чмир, І.О. Моделювання систем у середовищі UML: навч. посібник / І. О. Чмир, М. Ф. Ус ; Черкаськ. акад. менеджменту. - Черкаси : ЧАМ, 2004. - 100 с.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ