Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики
факультету інформатики

Текстова частина до курсової роботи
за спеціальністю 113 Прикладна математика

освітня програма «Прикладна математика»

Реалізація веб-застосунку розумних речей з використанням хмарних засобів Amazon

Керівник курсової роботи
доц. Ющенко Ю.О.

“____” _____ 2022 р.

Виконав студент 3-го курсу
Забродський В.Є

“____” _____ 2022 р.

Київ 2022

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
Доцент., к.ф.-м.н.
_____С. С. Гороховський
(підпис)

“____” _____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Забродському Владиславу Євгеновичу факультету інформатики 3-го курсу
ТЕМА: Реалізація веб-застосунку розумних речей з використанням хмарних засобів Amazon

Вихідні дані:
Веб-застосунок розумних речей придатний для використання із Гугл

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
1. Аналіз предметної області
2. Створення розумного предмету побуту за допомогою плати ESP8266
3. Створення веб-застосунку
4. Розгортання веб-застосунку до хмари Amazon
5. Інтеграція з додатком Google Home
Висновки
Список літератури
Додатки (за необхідністю)

Дата видачі “____” _____ 2022 р.

Керівник _____ Завдання отримано _____

Календарний план виконання курсової роботи

Тема: Реалізація веб-застосунку розумних речей з використанням хмарних засобів Amazon

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	20.10.2021	
2.	Огляд документації за темою роботи	10.2021-12.2021	
3.	Аналіз предметної області	12.2021	
3.	Проектування веб-застосунку	01.2022	
4.	Програмування веб-застосунку	05.2022	
5.	Написання текстової частини	29.05.2022-2.06.2022	
6.	Остаточне оформлення роботи	06.06.2022	
7.	Здача роботи на перевірку на плагіат	07.06.2022	

Студент Забродський В.Є _____

Керівник Ющенко Ю.О _____

“_____”_____ 2022 р.

ЗМІСТ

Стор.

Анотація

Курсова робота присвячена створенню веб-застосунка розумних речей. Веб-застосунок створений для управління розумним домом і інтеграцією із ріщенням Google для розумного дому – Google Home. Для розробки веб-застосунка були використані:
плата ESP8266 оснащена мікроконтролером ESP8266 з WI-FI інтерфейсом; мова програмування Java; інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, розроблене компанією JetBrains - IntelliJ IDEA; системної автоматичної збірки - Gradle; NoSQL база даних – mongoDB ; framework – Spring boot; бібліотеки для мови програмування Java

Основні функції веб-застосунку: Надання команд платі ESP8266 за допомогою Google Home за допомогою голосу, тексту; Надання команд платі ESP8266 за допомогою HTTP[7] запитів; Авторизація і аутентифікація користувачів;

Ключові слова: розумний дім, Google Home, Java, Spring, користувач

Вступ

Постійний прогрес людства неодмінно супроводжується використанням нових технологій для полегшення повсякденних справ. Яскравим прикладом такого використання є розумний дім.

Перші електричні побутові прилади почали з’являтися ще у XX сторіччі і продемонстрували готовність суспільства замінити роботу домашнього персоналу дешевими механічними пристроями.

У сьогоденні актуальність розумного дому і розумних речей не поменшала , бо у наших реаліях усе змінюється дуже швидко і розумний дім може допомогти зекономити настільки дорогоцінний час .

Саме тому я, як людина, якій через велику кількість справ часто не вистачає часу мав на меті створити веб застосунок , за допомогою якого можна буде контролювати розумні речі.

Основними завданнями дослідження були: аналіз предметної області та створення функціонального веб-застосунку із інтеграцією із Google.

Об’єктом дослідження є процес розробки веб-застосунку розумного дому з розгортанням його у хмару Amazon та інтеграцією з Google Home.

Предметом дослідження є сам веб-застосунок.

Основна частина роботи складається з 5-ти розділів.

Головним результатом є створений веб-застосунок.

Перший розділ присвячено проведенню аналізу предметної області, визначення та опис основних функціональних вимог, згідно яких буде реалізовано можливості веб-застосунку. Другий розділ присвячено опису

створення розумного предмету побуту на основі плати ESP8266. У третьому розділі розглядається процес створення веб застосунку , створення функцій для спілкування із розумний предметом та архітектурні рішення для веб-застосунку. У четвертому розділі розглядається процес розгортання веб-застосунку у хмару Amazon та підтримка безперебійної роботи веб-застосунку. У п’ятому розділі розглядається процес інтеграції веб-застосунку із Google Home та огляду функціоналу веб-застосунку.

1. Аналіз предметної області

Призначенням даного розділу є аналіз предметної області . Визначення функціональних вимог і відповідальності веб-застосунку.

1.1 Характеристика предметної області

Обраною темою є створення веб-застосунку розумних речей . Застосунок складається з двох частин: сервера авторизації та серверу для виконання команд. Сервер авторизації виконує протокол авторизації OAuth2.0[9] типу Authorization Grant. Сервер для виконання команд отримує тип команди і параметри необхідні для виконання даної команди і , якщо дані інструкції можуть бути виконані існуючими розумними речами , виконує їх . Робота серверів відбувається відповідно до функціональних вимог

1.2 Функціональні вимоги

- Можливість авторизації і аутентифікації юзера за допомогою логіна і пароля відповідно до протоколу авторизації OAuth 2.0 Authorization Grant.[10]
- Можливість виконувати команди надіслані предмету розумного дому.
- Можливість опитування про нагальний стан предметів розумного дому.
- Можливість інтеграції застосунку із Google Home.

За підсумками двох підпунктів отримуємо вимоги до веб-застосунку.

2. Створення розумного предмету побуту за допомогою плати ESP8266

Призначенням даного розділу є опис процесу створення розумного предмету за допомогою плати ESP8266

2.1 Що собою являє ESP8266

ESP8266 – програмуємий мікроконтролер з інтерфейсом WI-FI. Для роботи мікроконтролера необхідна напруга 3.3В. Мікроконтролер дозволяє виконувати записаний в пам’ять цього мікроконтролера код. Записувати код на плату можна за допомогою інтегрованого середовища розробки. У веб-застосунки розумних речей плата використовується в якості демонстрації розумної речі , на яку будуть приходити команди про виконання або опитування стану речі. Для цього на плату потрібно записати код який би дозволяв це виконувати.

2.1 Програмування ESP8266

Один із способів записати код на плату - за допомогою інтегрованого середовища розробки Arduino IDE. Для запису коду на плату треба підключити її до комп’ютера. Програма на ESP8266 складається з двох частин :

void setup() [2]

Завантаживши програму, Arduino дає коду можливість взяти участь у ініціалізації системи. Треба вказати мікроконтролеру команди, які він виконуватиме в момент завантаження . Ці команди виконуються тільки один раз при старті системи. Саме з цією метою в програмі виділяється блок, в якому зберігатимуться такі команди. void setup() і є таким місцем усередині скетчу Arduino. Слово setup – є назвою функції. Слово перед назвою описує

тип повертаємих даних. Так як setup() ніяких даних програма не повертає – пишемо void.

void loop()[1]

Функція loop - функція, куди треба помістити команди, які будуть виконуватися весь час, поки включена плата Arduino. Почавши виконання з першої команди, мікроконтролер дійде до кінця і відразу перестрибне на початок для повторення тієї ж послідовності. Даний цикл буде йти поки на плату подають електроенергію.

Для веб-застосунку розумних речей потрібно аби плата могла слухати і виконувати команди. З цією метою у void setup() ми покладемо код для під’єднання до мережі WI-FI і створення власного серверу задля реагувань на запити до мікроконтролера. У void loop() покладемо код для реагування на запити клієнта . Задля імітації лампи створимо змінну «on» яка буде відповідати за стан «вимкнено-ввімкнено» імітованої лампи. Запит клієнта «1» буде ставити значення змінної “on” на «true» , запит «0» - на “false” , а запит «0» - видаватиме значення змінної “on”.

У кінці розділу маємо запрограмовану плату ESP8266, яка буде слгувати як розумна річ для веб-застосунку розумних речей

3. Створення веб застосунку

Призначенням даного розділу є опис процесу створення веб-застосунку розумних речей

3.1 Спілкування веб-застосунку із розумним предметом побуту на базі

ESP8266

Із розумною річчю веб-застосунок буде спілкуватися шляхом підключення до серверу , який було запрограмовано на створення у мікроконтролері ESP8266. Для підключення до серверу розумної речі треба знати нікальний числовий ідентифікатор пристрою у комп'ютерній мережі, що працює за протоколом IP – IP адресу. Для зручності спілкування треба призначити розумній речі статичну IP-адресу усередині мережі. Призначення статичної адреси потрібно для того, щоб завжди знати до якої адреси підключатися у коді. Підключення відбувається за допомогою протокола Telnet - мережевого протокол для реалізації текстового термінального інтерфейсу через мережу. Веб-застосунок підключатиметься до мікроконтролера за допомогою ір-адреси та номеру порту, на якому було запущенно сервер на мікроконтролері, виводитиме данні у термінальний інтерфейс і зчитуватиме відповіді розумної речі на надані їй команді через термінальний інтерфейс.

Для зручності програмування і збереження принципів чистого коду спільна поведінка можливих розумних речей буде виокремлена у абстрактні типи, які використовуватимуться для визначення спільної поведінки розумних речей даного типу. Так, для створеної умовної лампи має бути реалізованим

абстрактний тип «можна ввімкнути і вимкнути», який матиме наступні методи:

void turnOn(Boolean on) – ввімкнення та вимкнення розумної речі;

Boolean getOn() – опитування розумної речі чи ввімкнена вона;

Де Boolean це тип даних який може приймати значення “true” або “false”.

Після створення логіки спілкування розумної речі і застосунку можна переходити до створення архітектури самого веб-застосунку .

3.2 Архітектурні рішення веб-застосунку

Архітектура веб-застосунку не є вимушеною. Так як веб застосунок складається у тому числі із сервісу авторизації – потрібно обрати тип бази , у якій зберігатиметься інформація про юзерів . Для цілей веб-застосунку ідеально підійде NoSQL база даних mongoDb. Mongodb - документоорієнтована система управління базами даних, яка не вимагає опису таблиць, використовує JSON-подібні документи та схему бази даних. Переваги такої бази даних:

Швидший доступ до даних ніж у SQL базах;

Легше розгорнути ніж SQL бази даних;

Легше втілювати зміни у уже готову базу ніж це було б із SQL базою даних;

Усі ж недоліки такої бази є несуттєвими для даного веб-застосунку , так як веб-застосунку не важливий постійний гарантований доступ до свіжих даних у базі і у базі даних нам не потрібно міняти більш ніж одну сутність за раз .

Також потрібно обрати тип архітектури. Мікросервіси чи Моноліт.

Мікросервіси являють собою варіант сервіс-орієнтованої архітектури програмного забезпечення, спрямований на взаємодію невеликих, слабко пов'язаних між собою і легко змінюваних модулів називаємих мікросервісами.

Монолітна ж модель відноситься до єдиної неподільної одиниці. Концепція монолітного програмного забезпечення полягає в тому, що різні компоненти програми об'єднуються в одну програму на одній платформі.

Для даного веб-застосунку краще підійде мікросервісна архітектура , бо наявні 2 не пов’язані між собою сервіси : сервіс авторизації та сервіс управління девайсами .

У обох сервісах HTTP запит потрапляє на ендпоінт і далі передається сервером на сервіси всередині мікросервісу , які оброблюють запит в залежності від ендпоінта на який він прийшов і параметрів цього запиту .

У кінці розділу маємо сформований веб-застосунок

4. Розгортання веб-застосунку у хмару Amazon

Призначенням даного розділу є опис процесу розгортання веб-застосунку до хмари Amazon

4.1 Опис Amazon Web Services

Amazon Web Services (AWS) - комерційна публічна хмара. Надає послуги як за інфраструктурною моделлю (віртуальні сервери, сховища ресурсів), так і платформного рівня (хмарні дані, хмарне програмне забезпечення, хмарні безсерверні обчислювальні потужності, засоби розробки).

Після створення веб застосунку виникає питання де підтримувати його роботу. На локальній машині застосунок займав би забагато місця і оперативної пам’яті для постійної підтримки його роботи, тому рішення розгорнути у хмару видається гарним . Провівши аналіз функціоналу різних хмар було прийнято рішення використати хмару Amazon через обширний функціонал . Хоча майже увесь цей функціонал і не потрібен веб-застосунку із функціонал зазначеним у першому розділі , проте така кількість сервісів як у Amazon дозволить в майбутньому зекономити багато часу , якщо буде прийняти рішення розробляти застосунок і далі. Наприклад, використавши Amazon SQS. Amazon Simple Queue Service – це повністю керований сервіс черг повідомлень, за допомогою якого можна ізолювати та масштабувати мікросервіси.

Поки для веб-застосунку із сервісів Amazon потрібен тільки Ec2 – служба оренди віртуальних серверів.

4.2 Процес розгортання веб-застосунку

Веб-застосунок буде розгорнуто на службі оренди віртуальних серверів Amazon – Ec2. Ec2 дозволяє користувачеві орендувати віртуальний сервер, називаємий instance. Для запуску віртуальних серверів використовуються попередньо налаштовані образи, що скорочує час завантаження нового сервера.

Орендуємо instance з операційною системою Linux.

Один із способів доступу до даного instance є термінал . Цей спосіб є кращим у випадку розгортання веб-застосунку на віртуальну машину, бо потрібно мати доступ до локальних файлів комп'ютера , зокрема до файлів веб-застосунку. Виконуваними файлами на мові програмування Java є ".jar"[8] файли . ".jar"- це Java-архів. Являє собою ZIP-архів, в якому міститься частина програми мовою Java. Даний файл для цього веб-застосунку збудуємо за допомогою системи атоматичної збірки Gradle.

Для виконування Java-архіву на Ec2 потрібно заванатажити на Ec2 Java Development Kit. Після завантаження файл можна запустити командою у терміналі. Так як звичайна повіденка таких файлів при прериванні зв'язку із instance це вихід программи , то треба зробити спосіб запуску файлу окрім як командою запуску безпосередньо jar-файлу. Для таких випадків у операційній системі Linux існує утіліта systemctl створена для роботи із systemd.

systemd - набір базових компонентів Linux системи. Являє собою менеджер системи та служб. Зокрема надає можливість для запуска демонів і служб.

Демон - комп'ютерна програма в UNIX-подібних системах, що запускається самою системою і працює у фоновому режимі без безпосередньої взаємодії з користувачем.

systemctl – головна команда для роботи із systemd . Вона дозволяє відстежувати стан системи та керувати системою та службами (те що і демон).

Отже потрібно створити службу яка б виконувала jar-файл , щоб при термінації сесії із instance веб-застосунок не завершував свою роботу автоматично .

Після створення демона і розгортання веб-застосунка можна зістнкутися із проблемою, що підключення через telnet до плати ESP8266 стає неможливим. Причина цьому у тому , що у плати є своя ip-адреса лише усередині локальної мережі. Для підключення до плати за межами локальної мережі потрібно використовувати публічно ip-адресу. Публічна IP-адреса – це IP-адреса, до якої можна отримати доступ безпосередньо через Інтернет і призначається вашому мережевому маршрутизатору вашим постачальником послуг Інтернету. Отже, за допомогою публічної адреси можна підключитися до роутера . Для підключення ж до девайсу треба налаштувати на роутері port-forwarding. У комп’ютерних мережах port-forwarding — це програма трансляції мережевих адрес, яка перенаправляє запит на зв’язок з однієї комбінації адреси та номера порту на іншу, поки пакети проходять через мережевий шлюз, такий як маршрутизатор. Налаштувавши port-forwarding на переадресацію з публічного ip з обраним портом на ip-дресу усередині мережі можна буде спілкуватися із розумною річчю за межами локальної мережі.

У підсумку цього розділу маємо веб-застосунок розгорнутий у хмарі Amazon.

5 Інтеграція з додатком Google Home

У цьому розділі буде описано процес інтеграції веб застосунку із сервісом Google Home та пояснено що таке Google Home і навіщо він потрібний.

5.1 Опис Google Home

Google Home – застосунок який дозволяє контролювати елементи розумного дому.

Інтеграція із Google Home складається із двох обов’язкових і 1 опціональної дії.

Обов’язкові:

Account Linking[3] - Аутентифікація, яка дозволяє пов’язувати облікові записи користувачів Google із обліковими записами користувачів веб-застосунку з яким відбувається інтеграція.

Fulfilment[4] - це код, який розгортається як веб-хук, який дозволяє генерувати динамічні відповіді для запитів користувача. Під час розмови користувача з Google Assistant, виконання дозволяє використовувати інформацію, отриману за допомогою обробки природної мови Google, щоб генерувати динамічні відповіді або ініціювати дії на сервері веб-застосунку, наприклад, увімкнути світло [5]. Реалізація буде основана на документації гугл, з відповідними об’єктами транспортування даних (DTO).

Опціональна:

Report state - функція, яка дозволяє Smart Home Action проактивно повідомляти останній статус пристрою користувача до Google Home.

Де веб-хук - метод розширення чи зміни поведінки веб-сторінки або веб-застосунку за допомогою зворотніх звернень;

Google Assistant - хмарний сервіс особистого помічника.

5.2 Сервіс для авторизації і аутентифікації

Сервір для авторизації і аутентифікації виконує Account Linking. За документацією Google сервім має працювати на протоколі авторизації Oauth2.0 типу Authorization Code Grant . Для авторизації юзер потрапляє на сервер веб-застосунку де може ввести данні потрібні для доступу до аккаунту данного юзера (рисунок 5.2.1).

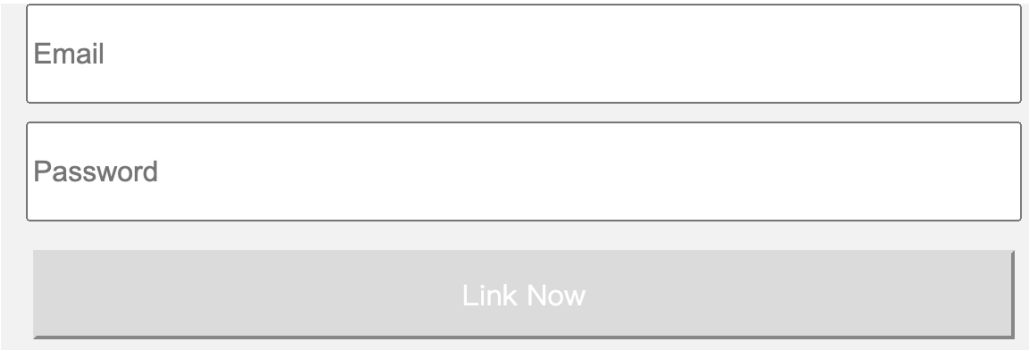

Рисунок 5.2.1

Після натискання на кнопку Link Now на сервер ініціювавшого запит відправляється відповідь із авторизаційним кодом, який показує що користувач підтверджує свою авторизацію. В майбутньому при міжсерверному спілкуванні даний код обмінюється на токен доступу , який активує сесію користувача.

5.3 Прив’язка до Google Home та тестування

Для прив’язки веб-застосунку до Google Home потрібно створити проект типу Smart Home у Google та заповнити необхідні поля у консолі розробників від Google (рисунок 5.3.1 та 5.3.2).

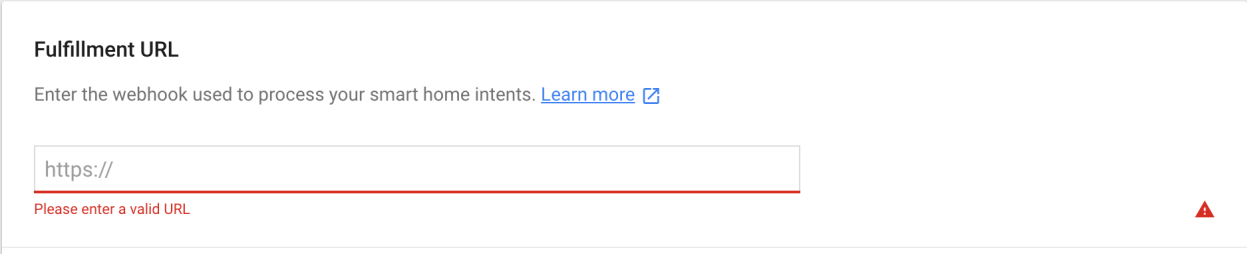

Рисунок 5.3.1

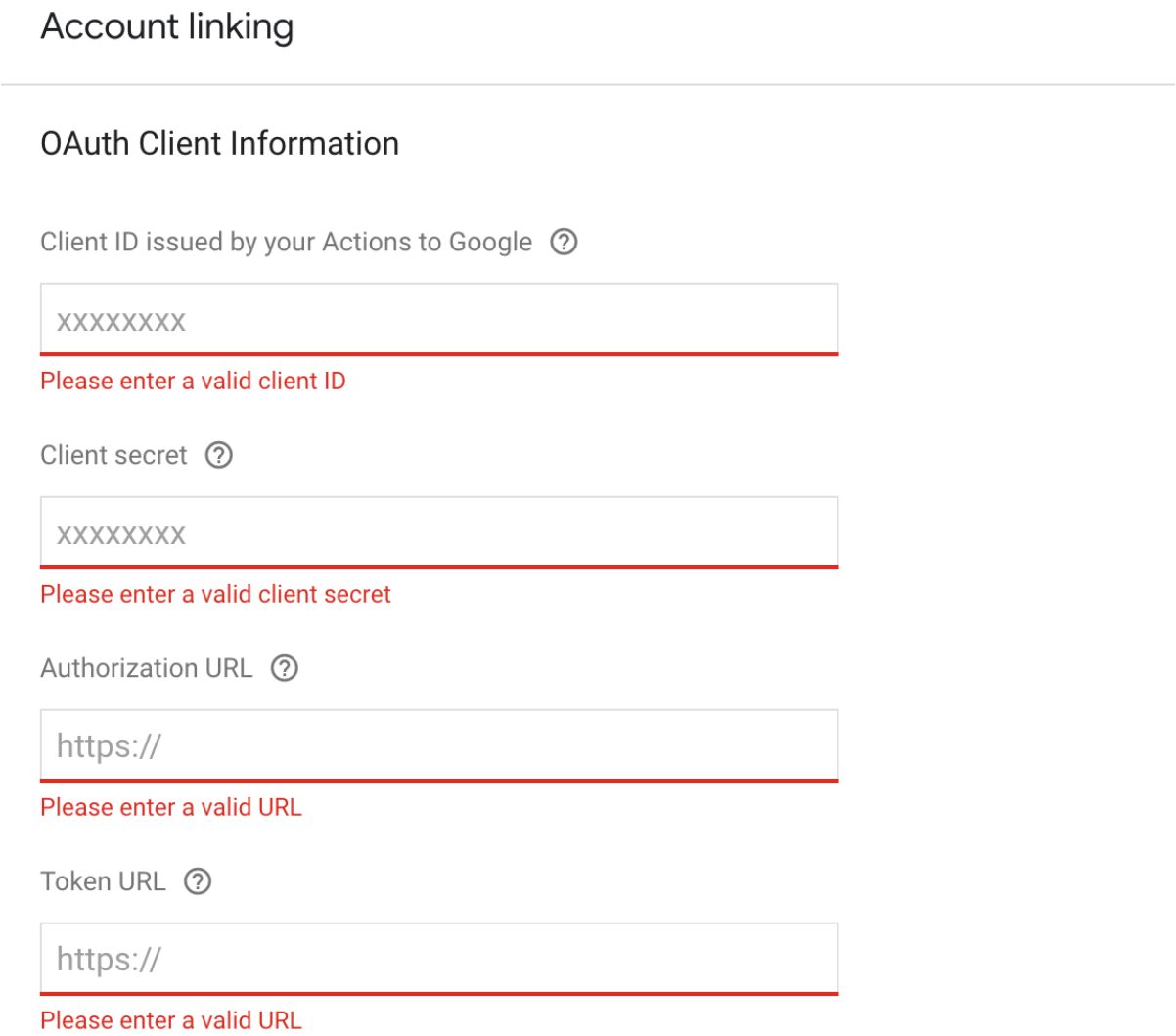

Рисунок 5.3.2

Для заповнення усіх полів потрібно отримати Client ID та Client Secret. Їх можна отримати безпосередньо у сервісах Google.Client id слугує ідентифікатором сервіса який намагається ініціювати авторизацію . Client Secret відомий тільки серверу на який іде запит і серверу з якого іде запит, він слугує підтвердженням того, що саме сервіс який має даний Client id надіслав запит.

Також потрібно мати протокол з'єднання HTTPS[6]. За замовчуванням Java застосунки не дають протокола HTTPS, як і хмара Amazon. Отже, треба змусити сервера на яких розгорнутий веб-застосунок використовувати HTTPS. Для цього є багато способів , але для цього випадку обирається найпростіший – прописати у термінал Ec2 instance команду « ssh -R 80:localhost:8080 nokey@localhost.run » після запуска сервіса , який запускає jar файл. Дана команда створює проксі на порті 8080 (порт за замовчуванням для java застосунків , написаних на tomcat . Саме на tomcat даний веб-застосунок і був написаний) і створює безпечне з'єднання із ним , після чого надає згенероване посилання на сервіс (рисунок 5.3.3), який запущений на цьому порті

```
e5c7449fd4934c.lhrtunnel.link tunneled with tls termination, https://e5c7449fd49
34c.lhrtunnel.link
```

Рисунок 5.3.3

Після отримання усіх потрібних даних і їх запису у відповідні поля у консолі Google можна переходити до тестування інтеграції.

Спочатку тестування буде відбуватися за допомогою телефона у додатку Google Home від Google. Треба пройти авторизацію у Google з того ж аккаунту, на якому було створено Smart Home проект. Після відкриття і проходження авторизації користувач попадає на стартовий екран додатку (рисунок 5.3.4)

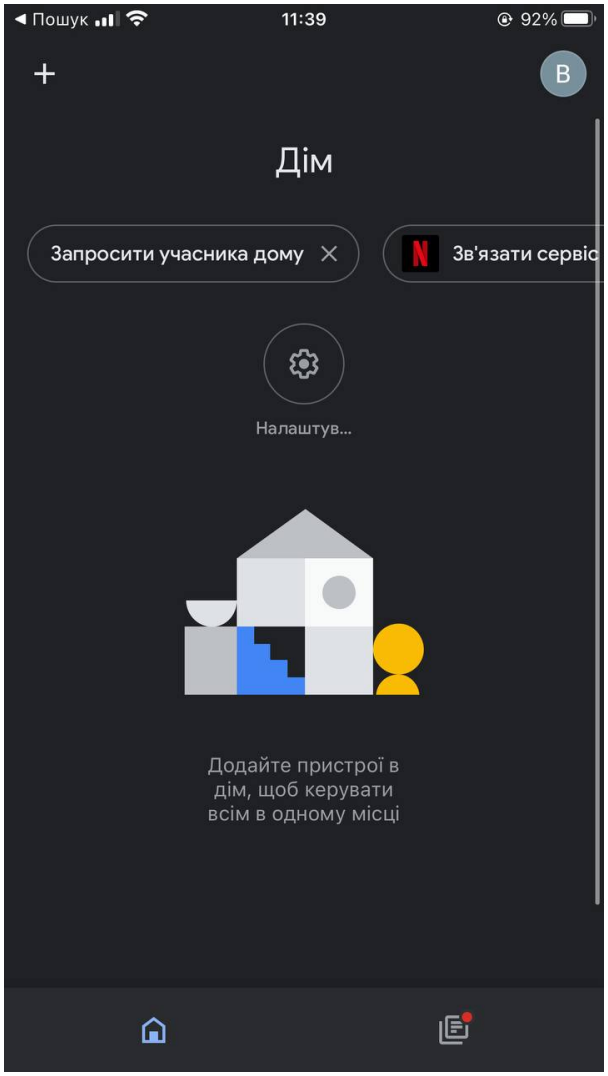

Рисунок 5.3.4

У верхньому лівому куті є кнопка + , потім «Налаштувати пристрій» (малюнок 5.3.5) і «працює з Google» (рисунок 5.3.6) можна побачити усі застосунки , які мають інтеграцію з Google , серед яких буде і створений у консолі розробників власний проект (рисунок 5.3.7).

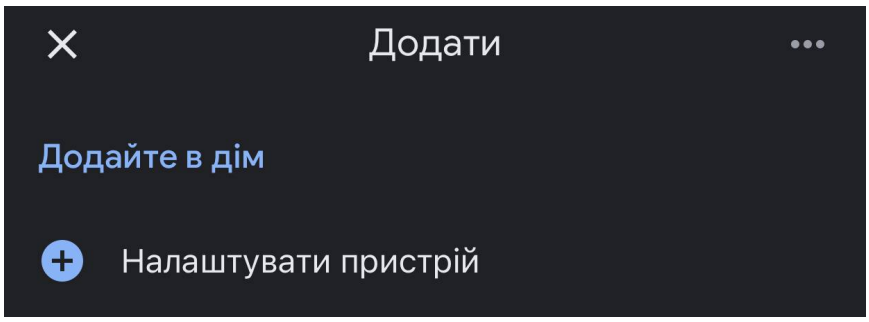

Рисунок 5.3.5

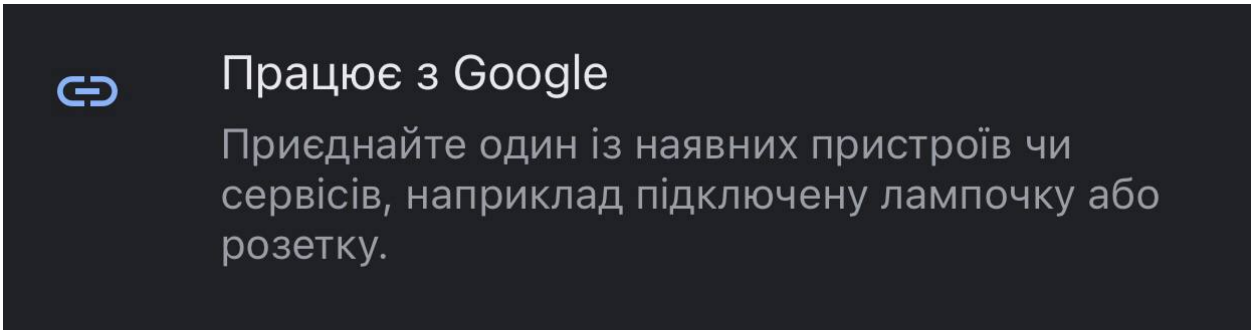

Рисунок 5.3.6

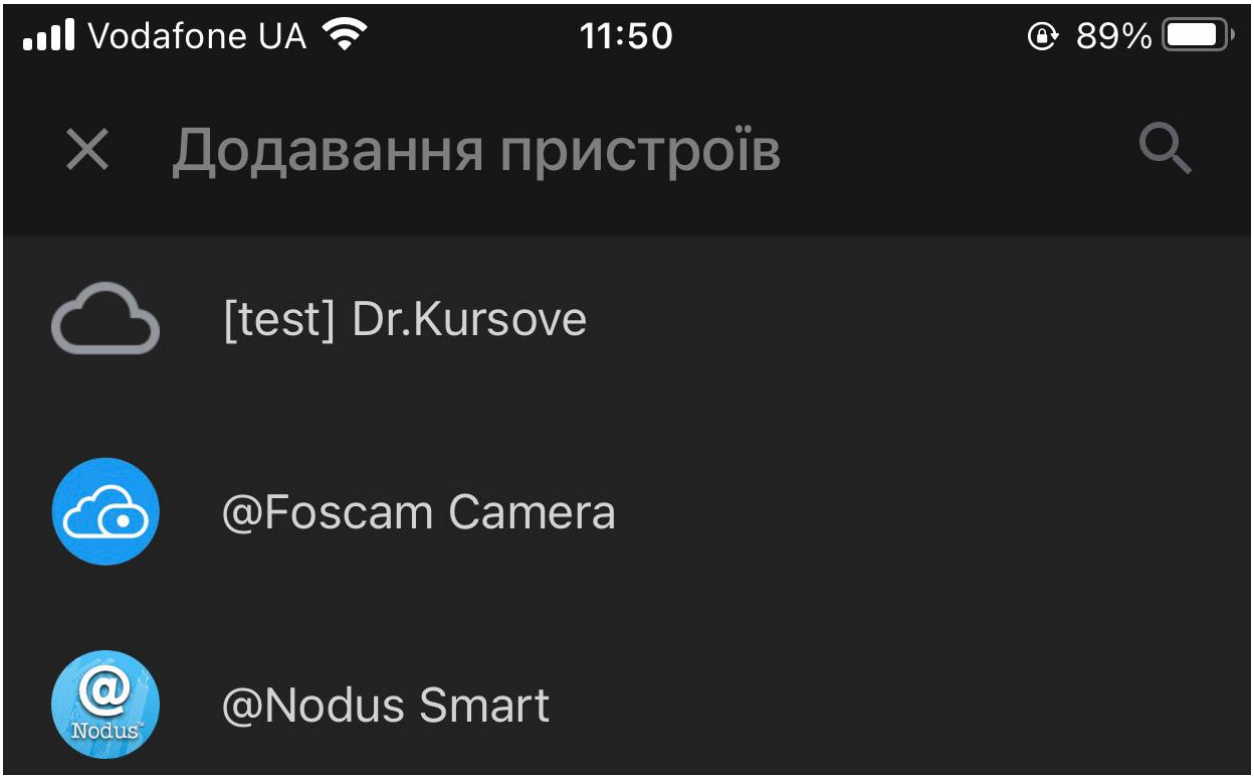

Рисунок 5.3.7

Обираємо створений проект (у даному випадку він має назву Dr.Kursove) і потрапляємо на сторінку авторизації (рисунок 5.3.8) яка знаходиться на сервері для Account Linking веб-застосунку.

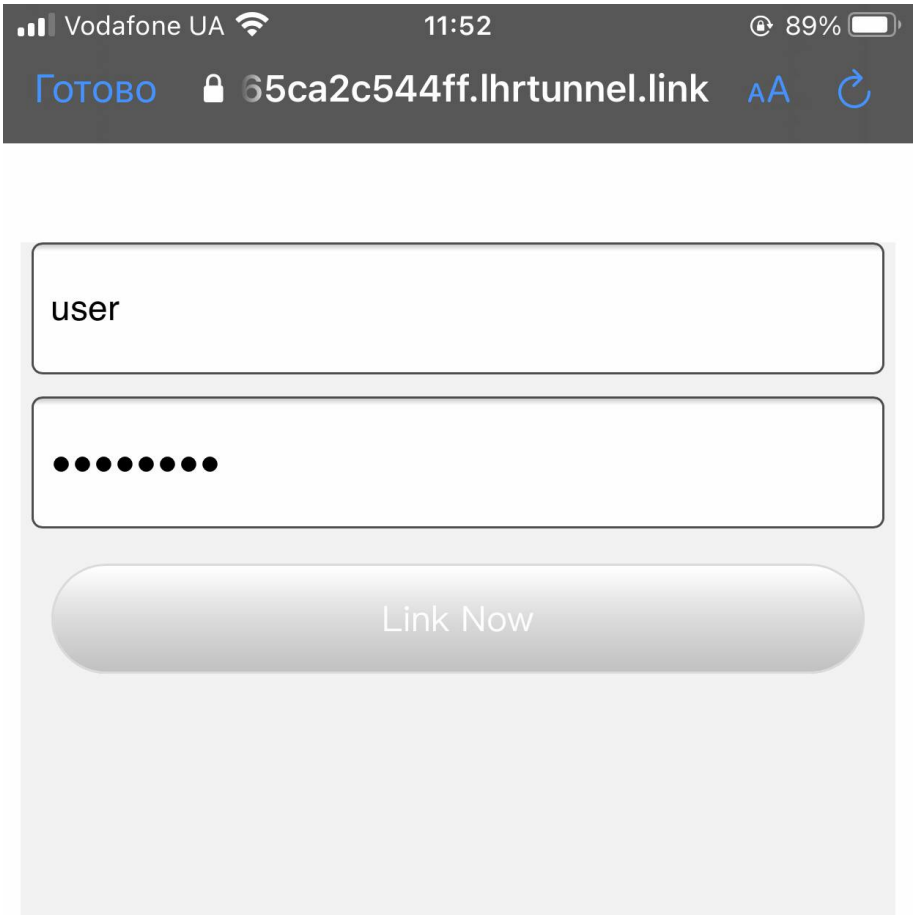

Рисунок 5.3.8

Після вводу даних і натискання на кнопку “Link Now” починається процес лінкування аккаунту веб-застосунку до аккаунту Google (рисунок 5.3.9). Після успішного приєднання вилазить сповіщення про завершення лінкування аккаунту (рисунок 5.3.10).

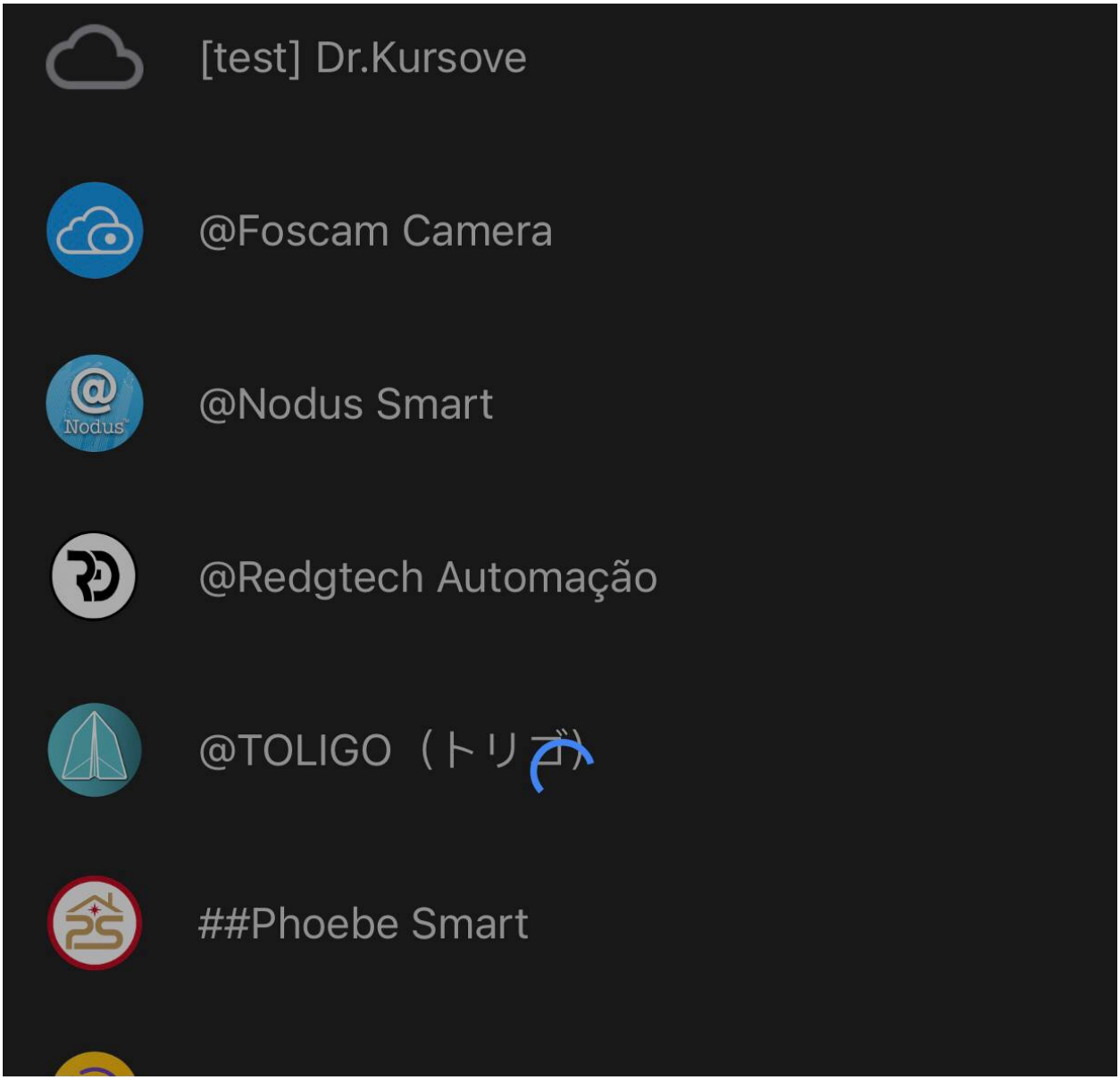

Рисунок 5.3.9

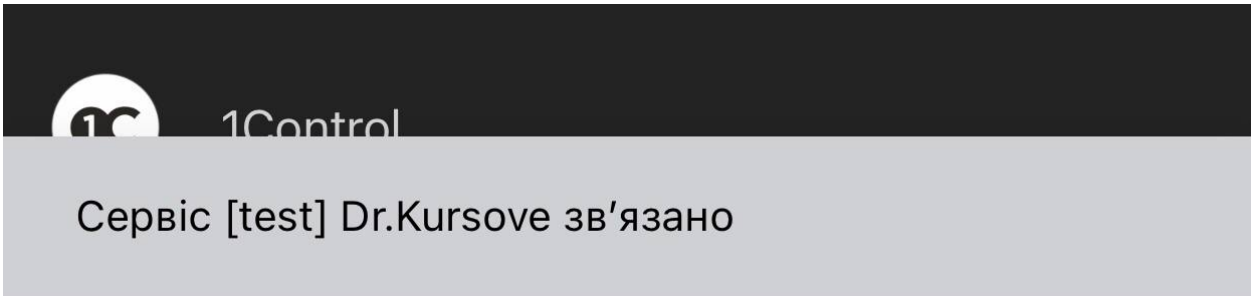

Рисунок 5.3.10

Починається синхронізація розумних предметів доступних користувачу у веб-застосунку(рисунок 5.3.11).

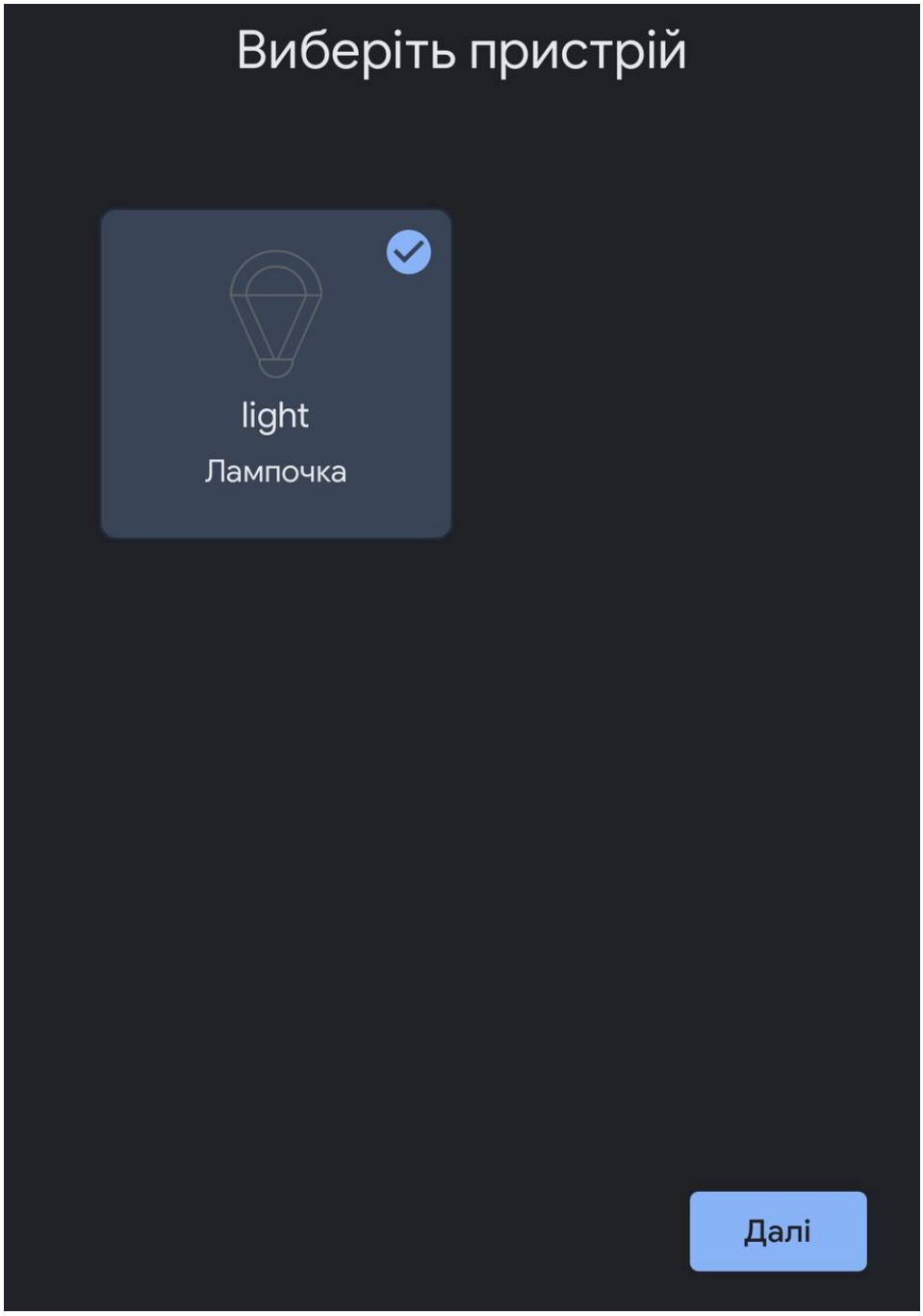

Рисунок 5.3.11

Після обрання речей, які користувач хоче додати до Google Home , на сервер відправляється запит про дізнання стану розумної речі (Рисунок 5.3.12).

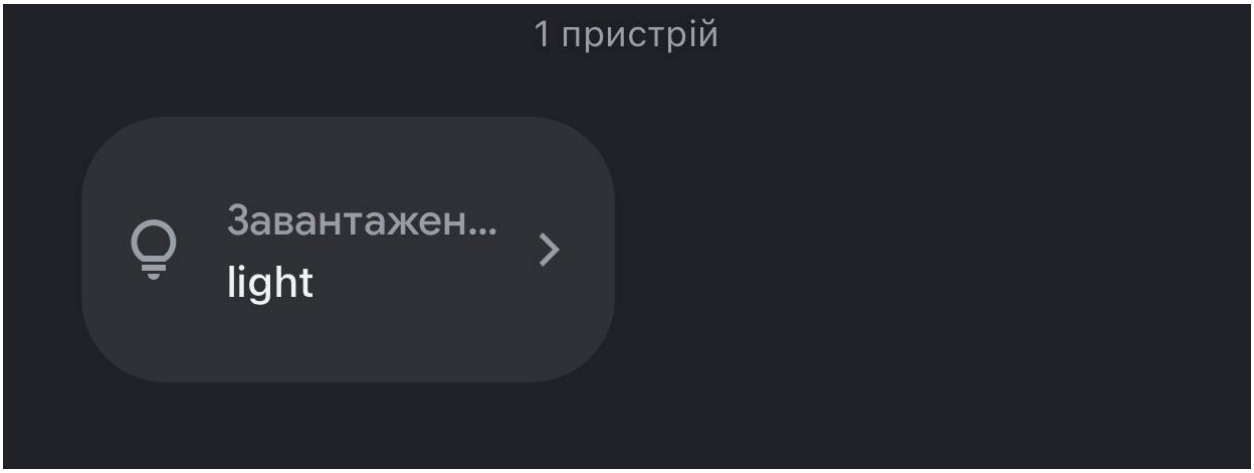

Рисунок 5.3.12

У випадку данного веб-застосунку це запит про стан «ввімкнено-вимкнено» та стан «онлайн-офлайн» і стан який наразі має девайс відображається у Google Home (рисунок 5.3.13, рисунок 5.3.14, рисунок 5.3.15).

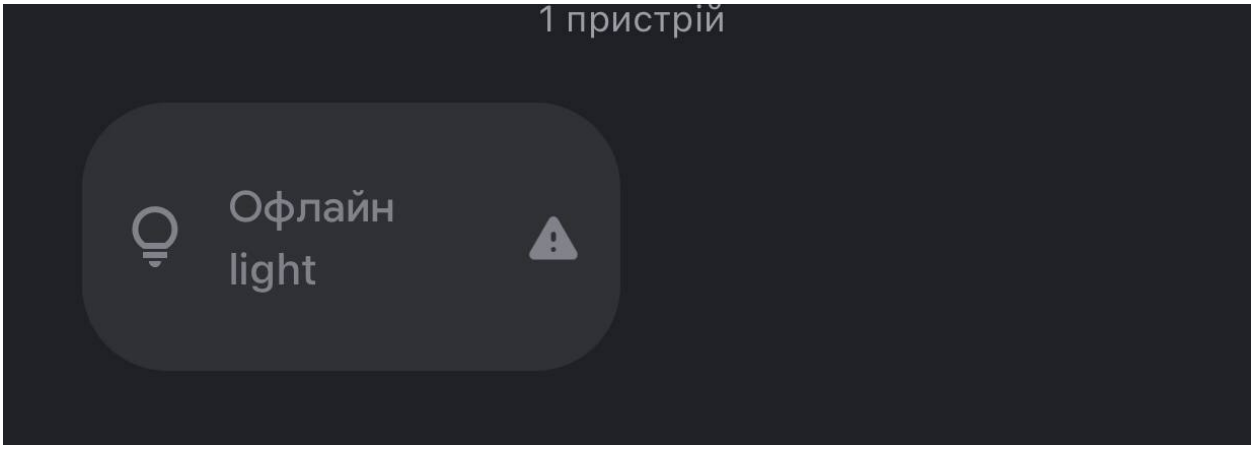

Рисунок 5.3.13

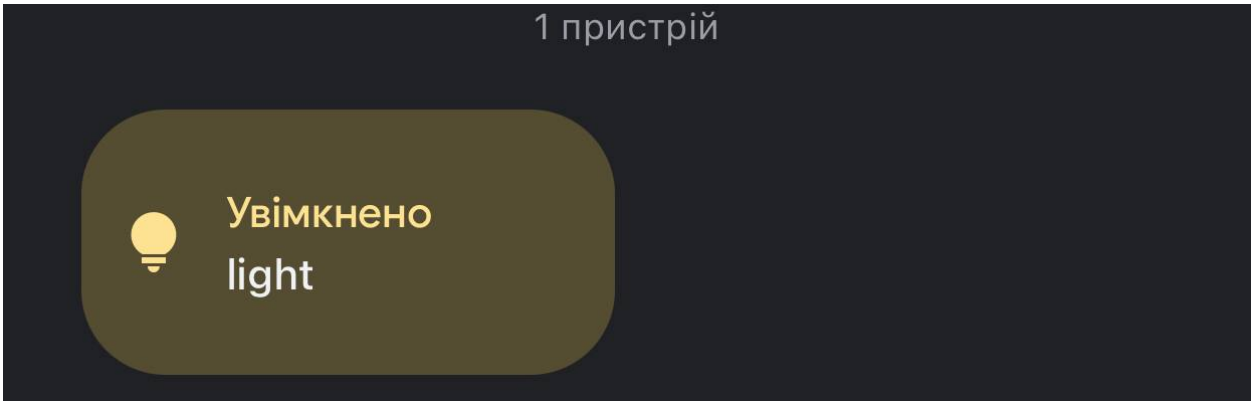

Рисунок 5.3.14

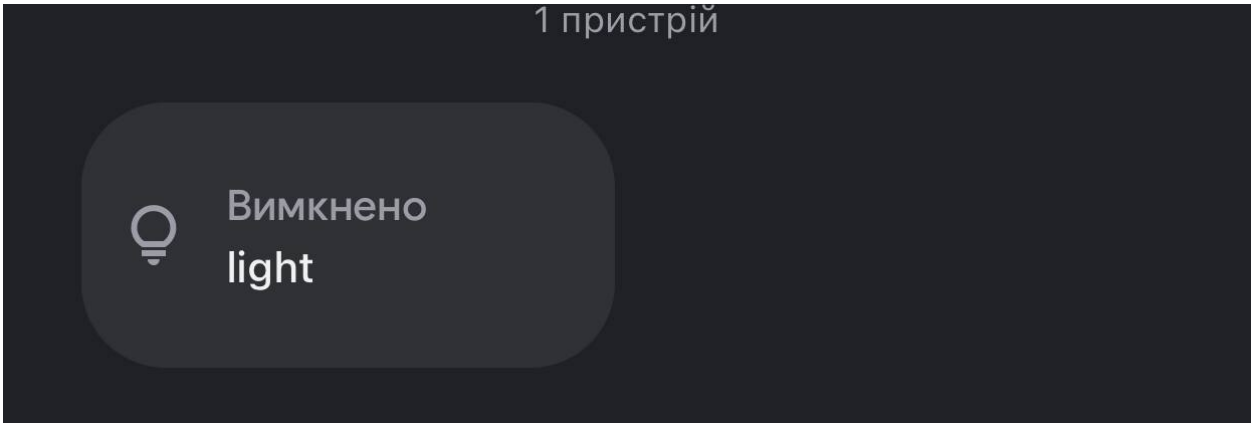

Рисунок 5.3.15

В першому випадку при опитуванні девайс не надсилає нічого , адже він вимкнений

У другому і третьому девайс виводить серверу веб-застосунку повідомлення про свій стан у термінал (рисунок 5.3.16)

```
admin@MacBook-Air ~ % telnet 194.44.33.192 8181
Trying 194.44.33.192...
Connected to 194.44.33.192.
Escape character is '^]'.
2
>turned offConnection closed by foreign host.
```

Рисунок 5.3.16

При натисканні на іконку девайсу він змінює свій стан «ввімкнено-вимкнено» на протилежний , якщо не знаходиться офлайн.

Висновок

Результатом цієї роботи став веб-застосунок розумних речей, яикй має інтеграцію з сервісом Google Home. За допомогою цього застосунку можна контролювати розумні речі в основі яких лежить плата ESP8266 і автоматизувати власне житло .

В ході розробки був досліджений процес роботи із Arduino IDE, програмування мікроконтролерів ESP8266, процес інтеграції із сервісами Google, був вивчений спосіб розгортання додатку з HTTPS протоколом.

Найбільшими труднощами були:

Створення бібліотеки для спілкування із розумними девайсами через telnet та її публікація до локального репозиторію системи автоматичної збірки Gradl;

Програмування і завантаження коду серверу на мікроконтролер ESP8266;

Перспективами розвитку є створення застосунку з користувацьким інтерфейсом для контролю розумних девайсів, збільшення кількості девайсів, покращення існуючого девайсу, інтеграція з іншими розумними домами такими як Alexa або AWS iot.

Список використаної літератури

1. Arduino documentation loop() [Електронний ресурс] – режим доступу:

   https://www.arduino.cc/reference/en/language/structure/sketch/setup/

2. Arduino documentation setup() [Електронний ресурс] – режим доступу:

   https://www.arduino.cc/reference/en/language/structure/sketch/loop/

3. Google Account Linking documentation [Електронний ресурс] – режим доступу:

   https://developers.google.com/assistant/smarthome/concepts/account-linking

4. Google Fulfilment documentation [Електронний ресурс] – режим доступу:

   https://developers.google.com/assistant/smarthome/concepts/fulfillment-authentication

5. Google Intent fulfilment documentation [Електронний ресурс] – режим доступу:

   https://developers.google.com/assistant/smarthome/develop/process-intents

6. Http over TLS documentation [Електронний ресурс] – режим доступу:

   https://datatracker.ietf.org/doc/html/rfc2818

7. Hypertext Transfer Protocol documentation [Електронний ресурс] – режим доступу:

   https://datatracker.ietf.org/doc/html/rfc2616

8. Jar overview [Електронний ресурс] – режим доступу:

   https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jarGuide.html#:~:text=JAR%20stands%20for%20Java%20ARchive,applets%20and%20their%20requisite%20components%20(.

9. Oauth2.0 documentation [Електронний ресурс] – режим доступу:

   https://datatracker.ietf.org/doc/html/rfc6749

10. Oauth2.0 documentation Authoriztaion Grant [Електронний ресурс] – режим доступу:

    https://datatracker.ietf.org/doc/html/rfc6749#section-1.3