НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Проект “Розумний Дім”

з дисципліни **«**Сучасні техн. програм.-1**»**

Виконав:

студенти 3 курсу

ФІОТ гр. ІВ-81

Кулініч Віталій

Буцев Богдан

Перевірив:

Шевело Олексій

Київ – 2020 р.

ЗМІСТ

|  |
| --- |
| 1. Overview……………………………………………………...…..3   1.1 Специфіка ……………………………………………....……3  1.2 Вимоги……………………………………………....………..5   1. Основні шляхи дій користувача………………………………...6 2. Порівняння з іншими програмами……………………………...8 3. Дизайн…………………………………………………….………9 4. Розробка………………………………………………....………10 |

**1. Overview**

**1.1. Специфіка**

Нашою задачею буде реалізувати на Django сервер управління розумним домом, маючий web-інтерфейс для настройки і ручного управління, який буде проводити періодичне опитування датчиків і здійснювати автоматичну реакцію в разі певних ситуацій, використовуючи API контролера розумного будинку. Пристрої, підключені до контролера, доступні на запис (зазвичай true - включити/открити, false - виключити/закрити, але бувають і варіанти). І датчики і пристрої доступні на читання. Пристрої, при читанні з них, працюють як датчики і повертають свій стан, який може відрізнятись від записаного.

**Пристрої (запис):**

1. **air\_conditioner** – Кондиціонер (true - вкл, false - викл). При включенні поступово знижує температуру в спальні, поки вона не досягне 16 градусів і сильніше охолодити вже не може.
2. **bedroom\_light** – Лампа в спальні (true - вкл, false - викл).
3. **bathroom\_light** –Лампа у ванній (true - вкл, false - викл).
4. **curtains –** Фіранки string ( "open" - відкрити, "close" - закрити).
5. **boiler** – Бойлер (true - вкл, false - викл). При включенні поступово підвищує температуру води, поки вона не досягне 90 градусів. Для роботи повинен бути відкритий вхідний кран холодної води.
6. **cold\_water** – Вхідний кран холодної води (true - відкрити, false - закрити). Дозволяє відкрити / перекрити подачу холодної води в квартиру.
7. **hot\_water** – Вхідний кран гарячої води (true - відкрити, false - закрити).
8. **washing\_machine** – Пральна машина string ( "on" - вкл, "off" - викл). При включенні починає прати, потім самостійно відключається. Може зламатися і протекти.

**Датчики (читання):**

1. **air\_conditioner** – Кондиціонер. (True - вкл, false - викл).
2. **bedroom\_temperature** – Температура в спальні. Int (0 - 80).
3. **bedroom\_light** – Лампа в спальні. (True - вкл, false - викл).
4. **smoke\_detector** – Датчик задимлення на стелі. (True - задимлення, false - ні).
5. **bedroom\_presence** – Датчик присутності в спальні. (True - є людина, false - ні).
6. **bedroom\_motion** – Датчик руху в спальні. (True - є рух, false - ні).
7. **curtains** – Фіранки. string ( "open" - відкриті, "close" - закриті, "slightly\_open" - відкриті вручну).
8. **outdoor\_light** – Датчик освітленості за вікном (0 - 100).
9. **boiler** – Бойлер. (True - вкл, false - викл).
10. **boiler\_temperature** – Температура гарячої води бойлері. Int (0 - 100 / null). Якщо перекрита холодна вода, то води в бойлері немає, і датчик повертає null.
11. **cold\_water** – Вхідний кран холодної води. (True - відкритий, false - закритий).
12. **hot\_water** – Вхідний кран гарячої води. (True - відкритий, false - закритий).
13. **bathroom\_light** – Лампа у ванній. (True - вкл, false - викл).
14. **bathroom\_presence** – Датчик присутності у ванній. (True - є людина, false - ні).
15. **bathroom\_motion** – Датчик руху в ванній. (True - є рух, false - ні).
16. **washing\_machine** – Пральна машина. string ( "on" - вкл, "off" - викл, "broken" - зламана).
17. **leak\_detector** – Датчик протікання води (true - протікання, false - сухо).

**1.2. Вимоги**

Як і кожну систему, наш розумний дім потрібно буде протестувати. Для цього ми використовуватимемо спеціальний сайт (https://smarthome.webpython.graders.eldf.ru/), де вже реалізовані певні датчики та API для роботи з ними. Відповідну документацію надано на сайті. Для того щоб користуватися сайтом потрібно зареєструватись і отримати API для роботи з контролером. Використовуючи цей же ключ можна буде користуватись “віртуальною камерою” розумного дому, на якій буде наочно видно які пристрої працюють або чому спрацював той чи інший датчик, і вручну керувати пристроями.

**2. Основні шляхи дій користувачів**

Під час експлуатації нашого продукту, користувачеві буде доступний доступ через сайт до нашого додатку, так званий "пульт керування". Тільки таким чином він зможе використовувати функціонал додатка, інших варіантів у міру розробки не передбачається. Отже, коли доступ вже буде дано користувачеві, у нього буде дещо не мало, як три основні шляхи роботи.

Перший це, true / false-залежні датчики, які настільки прості, що у них лише два параметри, якихось додаткових параметрів, які треба виставляти вручну, вони не мають, тому тут і так все ясно. Якщо тобі потрібно щоб цей прилад працював, то перемикач на true, якщо ні - то false.

Другий вид - це true / false - незалежні датчики. Наприклад як датчик руху в спальні, ти його не можеш виставити true або false за замовчуванням, вони будуть називатися, так званими датчиками читання, активність яких можна буде дивитися на тому ж сайті, до якого є доступ у користувача. І крім цього можна буде відстежувати роботу не тільки true / false типу датчиків, а й таких наприклад як, датчик освітлений в кімнаті, який буде вказуватися в процентах.

Третій вид - це вид так званих, встановлюваних контролерів, які якраз таки мають специфіку в своїй роботі і не обмежуються звичайними true / false, а використовують вже показники, введені вручну. Як приклад, та ж температура на кондиціонері, яку користувач повинен виставити самостійно на свій розсуд. У будь-який момент він може змінити її, але в рамках допустимих значень, які будуть передбачені програмою. Також крім цього, звичайно ж буде встановлено якийсь оптимальний параметр за замовчуванням, для успішного запуску роботи програми в перший раз.

І буде ще один додатковий вид датчиків, але який буде прихований системою, під назвою - реагуючий на події датчик. За назвою вже фактично ясна його суть роботи: це оновлюваний кожну секунду датчик, до якого надходить інформація від інших датчиків, і якщо десь відбувається збій, то під нього включається певний алгоритм роботи датчика. Наприклад якщо відбувається витік, то спрацьовує певний датчик, який відключає датчики надходження гарячої і холодної води, тим самим перекриваючи воду в будинку.

**3. Чому буде краще за конкурентів**

У світі вже є аналоги нашого застосування, у вигляді розумних холодильників, колонок, моніторів, пральних машин і т.д. Але це все працює в більшості випадків окремо від різних компаній (і для роботи з кожним приладом доведеться встановити багато додатків), або екосистемою, (яка в себе включає максимум 2-3 приладів). У нашому ж випадку, підключення йде через API кожного приладу і робота відбувається через синхронізацію їх з сервером. У такому випадку можна використовувати дуже багато пристроїв для будинків різної площі і не бути прив'язаним до мережі Bluetooth або якоїсь іншої.

**4. Дизайн**

Що стосується дизайну, то тут все неоднозначно, все буде залежати від нашого часу і можливостей. З одного боку у нас вже буде basic django-дизайн, який хоч і примітивний, але працює ефективно, але для того щоб пустити його в продакшн або співпрацювати з якоюсь компанією, потрібен ширший інтерфейс, такий як django + JS (React), спираючись на приклади які вже працюють за такою схемою, це був би досить таки непоганий вибір, який вважається зараз досить таки прогресивним і робочим. Але як я вже сказав будемо дивитися по ситуації і орієнтуватися на час, який у нас буде. Можливо будемо змушені обійтися тільки першим варіантом.

**5. Розробка**

За адресою / повинна відкриватися веб-форма, з наступними контролями:

1. **bedroom\_target\_temperature** - input type = number, бажана температура в спальні, запам'ятовувати налаштування в базі, поточну настройку з бази виводити на форму. Допустиме значення від 16 до 50, default: 21.
2. **hot\_water\_target\_temperature** - input type = number, бажана температура гарячої води в будинку, запам'ятовувати налаштування в базі, поточну настройку з бази виводити на форму. Допустиме значення від 24 до 90, default: 80.
3. **bedroom\_light** - checkbox, включає / вимикає світло в спальні, синхронізувати значення з контролером.
4. **bathroom\_light** - checkbox, включає / вимикає світло у ванній, синхронізувати значення з контролером.

Там же відображати поточні значення всіх датчиків, прочитані з контролера. Для рендеринга шаблону прочитані з контролера значення повинні бути в словнику в context.data.

Реалізувати автоматично опитування контролера в фоновому режимі кожну секунду (django celery) і реакцію на деякі події.

Реакція на події:

1. Якщо є витік води (leak\_detector = true), закрити холодну (cold\_water = false) і гарячу (hot\_water = false) воду і відіслати листа в момент виявлення.
2. Якщо холодна вода (cold\_water) закрита, негайно вимкнути бойлер (boiler) і пральну машину (washing\_machine) і ні за яких умов не включати їх, поки холодна вода не буде знову відкрита.
3. Якщо гаряча вода має температуру (boiler\_temperature) менше ніж hot\_water\_target\_temperature - 10%, потрібно увімкнути котел (boiler), і чекати поки вона не досягне температури hot\_water\_target\_temperature + 10%, після чого з метою економії енергії бойлер потрібно відключити
4. Якщо штори частково відкриті (curtains == "slightly\_open"), то вони знаходяться на ручному управлінні - це значить їх стан не можна змінювати автоматично ні за яких умов.
5. Якщо на вулиці (outdoor\_light) темніше 50, закрити штори (curtains). Якщо на вулиці (outdoor\_light) світліше 50, відкрити штори. Крім випадків коли вони на ручному управлінні.
6. Якщо виявлений дим (smoke\_detector), негайно вимкнути такі прилади [air\_conditioner, bedroom\_light, bathroom\_light, boiler, washing\_machine], і ні за яких умов не включати їх, поки дим не зникне.
7. Якщо температура в спальні (bedroom\_temperature) піднялася вище bedroom\_target\_temperature + 10% - включити кондиціонер (air\_conditioner), і чекати поки температура не опуститься нижче bedroom\_target\_temperature - 10%, після чого кондиціонер відключити.

Опитування контролера і відправка йому відповіді повинні відбуватися всередині функції core.tasks.smart\_home\_manager. Ця функція повинна викликатися періодично з інтервалом в 5 секунд, наприклад, за допомогою celery. В початку своєї роботи функція запрошує дані з контролера, використовуючи requests.get в API, потім аналізує налаштування користувача по бажаної температурі з БД, і поточну ситуацію, і в кінці, якщо потрібна корекція, робить requests.post в API з командами для контролера, якщо необхідно відправити лист, то відправляє його.

Для відсилання листів потрібно використовувати send\_mail з django.core.mail, а в settings потрібно задати налаштування EMAIL\_HOST, EMAIL\_PORT і інші EMAIL\_ \*, так щоб під час розробки ви відправляли листи через якусь поштову систему і могли перевірити їх роботу. Під час перевірки завдання на сервері ці настройки будуть перевизначені.

Для збереження налаштувань в базі, потрібно використовувати модель Settings (name, value).

Веб-форма для настройки та управління розумним будинком повинна відкриватися в корені сайту, містити 4 input'а c іменами (name = ...): bedroom\_target\_temperature, hot\_water\_target\_temperature, bedroom\_light, bathroom\_light.

У settings потрібно додати змінні SMART\_HOME\_API\_URL і SMART\_HOME\_ACCESS\_TOKEN і задати їх значення, потім використовувати їх для взаємодії з розумним будинком. Ще можна додати EMAIL\_RECEPIENT, в якому задати одержувача листів від системи.

Після того, як реалізуємо функціонал, обов’язково запустимо manage.py test. В даному випадку ми перевіряємо завдання аналогічним чином через тести контролера по URL /, і ручний виклик core.tasks.smart\_home\_manager.