Одеський національний університет імені І.І.Мечникова

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем та технологій

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему: Комплексна тема: Реалізація елементів проекту "Розумний будинок": інтерфейс користувача керування системою вентиляції

(назва українською)

Complex topic: Implementation of elements of the project "Smart Home": ventilation control user interface

(назва англійською)

Виконала: студентка денної форми навчання
спеціальності 123 Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Астаф'єва Олександра Денисівна
(прізвище, ім'я, по-батькові)

Керівник к.ф.-м.н., доц. Якимчук В.І.
(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент д.т.н., проф. Гунченко Ю.О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2022 р.

Завідувач кафедри

_____ _Ю.О. Гунченко
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «_»__ ____ 2022 р.

Оцінка _____/_____/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Н.Ф. Казакова_
(підпис) (прізвище, ініціали)

Одеса - 2022

АНОТАЦІЯ

Дипломна робота є складовою частиною проекту по розробці системи розумної вентиляції.

Метою роботи є розробка інтерфейсу користувача для керування вентиляцією будинку.

Є можливість керування наступними параметрами:

- температура;
- вологість повітря;
- концентрація CO_2 у повітрі.

Данні параметрів збираються за допомогою датчиків та керуються відповідними приладами. Керування параметрами відбувається за допомогою веб-додатку.

Інтерфейс додатку розроблено за допомогою мови html та css. Для розробки програмної частини інтерфейсу користувача використовується мова програмування Python

ANNOTATION

The graduation work is a part of the project to develop a decision support system for smart ventilation.

The aim of the work is to develop a user interface to control the ventilation of the house.

It is possible to control the following parameters:

- temperature;
- air humidity;
- concentration of CO2 in the air.

Parameter data is collected by sensors and controlled by appropriate devices. Settings are managed using a web application.

The application interface is developed using html and css. To develop the software part of the user interface, the programming language Python

ЗМІСТ

ВСТУП

Вентиляція є однією з найважливіших систем забезпечення нормальних умов життєдіяльності людини. Так як повітря впливає на здоров′я людини, потрібно створювати сприятливий клімат, покликаний подачі свіжого повітря з вулиці та видалення забрудненого повітря з приміщень.

Свіже повітря в квартирі обов'язково має бути. Якщо використовується пасивна вентиляція, то його надходження доводиться регулярно провітрювати приміщення. Система вентиляції сприяє підтримці таких важливих для самопочуття параметрів як рівень температури, оптимальний рівень вологості для людини, рухливість повітря, контролює концентрацію в повітрі пилу та аерозолів.

Високий рівень вологості – це одна з головних ознак поганої вентиляції. Конденсат може утворюватися не лише на вікнах, а й на меблях, техніці, стінах. Через це відбувається швидке зношування будівельних матеріалів, меблів та приладів. Крім іншого, проживання у сирому приміщенні негативно позначається на здоров'ї людини.

Суспільство винаходить різноманітні прилади для регулювання різноманітних джерел і приладів будинку, додатково мінімізуючи участь людини в створенні комфортних умов житла. Важливо мати можливість спостерігати за подіями в реальному часі. Для того, щоб не перевіряти по кілька разів на день параметри повітря, можна автоматизувати цей процес або керувати ним з будь-якого місця лише маючи доступ до інтернету.

Маючи апаратно-програмний комплекс, всю розробку було розподілено на етапи роботи. Завдяки діаграмі Ганта (рис. 1) було розраховано оптимальні строки роботи над проектом. Детально діаграма Ганта представлена у Додатку А.

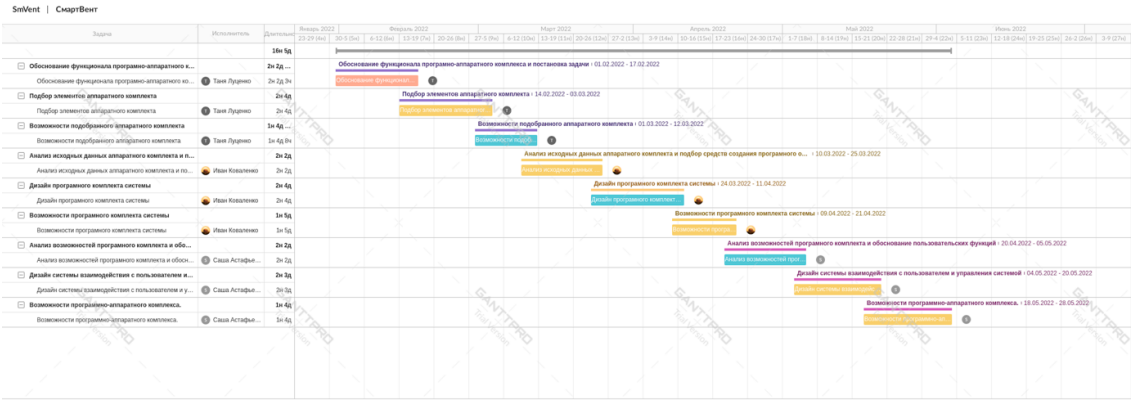


Рисунок 1 – Діаграма Ганта

На діаграмі видно, що розробка додатку починається після обгрунтування функціоналу, вибору компонентів програмно-апаратного комплексу та розробки програмної комплектації системи.

Для системи керування вентиляцією в розумному будинку створено додаток у вигляді веб-інтерфейсу. Оскільки пристрої на сьогодні стають все більш і більш потужними і поступово перетворюються на кишеньковий комп'ютер, рішення керування за допомогою веб додатка є найкращою ідеєю. Користувач не потребує спеціальне устаткування для цього, пізнавати нові технології або складні механізми - все в приємному і лаконічному інтерфейсі знаходиться усередині будь-якого пристрою.

Виходячи з цього, була сформульована мета роботи – розробка веб додатку для керування вентиляцією у своєму будинку.

Для досягнення ключових результатів необхідно:

1) проаналізувати існучі системи керування смарт вентиляцією;

2) розробити незалежний додаток, за допомогою якого можна керувати присторями вентиляції;

3) реалізувати з′єднання додатку з іншими частинами проекту.

1 **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

Перш ніж розпочати розробку додатку керування, слід сформулювати деякі поняття.

Вентиляція забезпечує людей чистим повітрям. Але якщо розширювати можливості вентиляції, можна покращити устаткування, додавши змогу вимірювання температури повітря у приміщенні, вологість, рівень вуглекислого газу. Також вентиляція повинна мати змогу сама відстежувати енергозабезпечення – на той виппадок, якщо станеться ввімкнення подачі електроенергії. Але у нашому проекті розумна вентиляція не лише працює сама по собі, а і користувач має можливість налаштувати роботу пристоів забезпечення усіх вище описаних параметрів: зволожувача повітря, обігрівач, кондиціонер. Про технічну роботу устаткування можна прочитати у роботах Коваленко Івана та Луценко Тетяни.

Програми для комп’терів зазвичай більш повні, ніж Інтернет програми та мобільні. З ними відкривається більше можливостей, тому що вони розраховані на роботу з повноцінними мишею, клавіатурою та великим монітором. Мобільні програми вважаються полегшеними версіями комп'ютерних програм, тому що користувач обмежений невеликим дисплеєм, а працювати з інтерфейсом може лише за допомогою пальців або стілуса. А швидкість веб-програм безпосередньо залежить від швидкості передачі даних, яку забезпечує інтернет-провайдер.

Web-програми в порівнянні з локальними програмами мають такі переваги:

1) простота доступу до програми – будь-яка людина, яка має комп'ютер, підключений до мережі Інтернет, може використовувати веб-додаток;

2) на відміну від локальних програм, веб-програми після завершення розробки не вимагають встановлення на комп'ютерах користувачів. Достатньо лише повідомити їм URL-адресу програми. При

зміні програми всі користувачі відразу починають працювати зі зміненою версією;

3) web-додатки не вимогливі до ресурсів і не висувають жодних вимог до апаратної платформи. Це означає, що немає жодної різниці, скільки мегабайт оперативної пам'яті встановлено на комп'ютері користувача чи з якою операційною системою він працює. Головна вимога- доступність браузера і доступ в Інтернет.

Основною перевагою веб-інтерфейсів є відсутність необхідності встановлення додаткового програмного забезпечення, оскільки популярні операційні системи поставляються з браузером.

Отримавши змогу керуванням мікрокліматом, потрібно забезпечити функціонал програмного забезпечення. Додаток складається з відомих користувачеві елементів керування.

Для того, щоб користувач міг слідкувати за зміною параметрів у часі чи у будь-якому проміжку, у додатку має бути можливість виводити ці зміни у вигляді графиків, наприклад якщо користувач хоче дізнатися середнє значення параметру за деякий період. За допомогою графіків користувач витрачає менше часу для аналізу інформації. Також візуальне предоставлення інформації значно покращує її сприймання.

У додатку також має бути відображення плану будинку у вигляді карти і відображення на ній основних показників датчиків і виконавчих пристроїв.

На веб сайті про розумний будинок є інформація про проект, приклади сценаріїв, відгуки користувачів, важливі статті про різні можливості проекту, рекомендації, контакти технічної підтримки. Тобто веб сайт потрібний не лише для маркетингу, але і для зручнішого зв'язку користувачів з розробниками, це також буде зрозумілий опис самого проекту, його значення, можливостей. Також на сайті є свій особистий кабінет, вхід в який здійснюється з двохфакторною аутентифікацією [4].

У особистому кабінеті веб сайту і в додатку є можливість замовляти додаткові функції для розумного будинку. Це означає, що у разі розширення

площі або купівлі нового устаткування користувачеві не доведеться турбуватися про введення його в систему розумної вентиляції [2].

Натискаючи на кнопки, дані передаються до приладів. За допомогою датчиків вони знаходять параметри, що завдав користувач або ті, що задані за замовчування та дотримуються завданого режиму параметрів користувачем.

Отже, додаток має основне меню, у якому керування параметрами, можливість відстежування графиків та поточні данні розподілені по різним вкладкам. Обравши першу вкладку, що називається «Поточні данні», користувач побачить розподілений на три колонки екран. Перша колонка містить у собі поточний стан рівня кисня у повітрі, наскільки він відповідає заданному користувачем, трохи нижче кнопку для переходу у пункт керування цим елементом. Наступна колонка містить такі ж самі елементи інтерфейсу, але передає стан температури повітря. Та третя – вологість.

Завдання забезпечити куростувача не лише приємним інтерфейсом, а й сконцентрувати його на охопленних параметрах керування та зрозумілості користування інтерфейсом.

Перед тим, як створювати додаток керуванням вентиляцією, повинно оглянути вже існуючі схожі додатки.

1.1 Вимоги до розумної вентиляції

Користувач потрібен мати змогу керувати дистанційно параметрами повітря у своєму будинку. Також потрібно мати змогу спостерігати за показниками, збирати та порівнювати данні за деякий період часу.

Встановивши параметри за замовчуванням, прилади повинні завжди їх підтримувати шляхом керування обладнанням тої чи іншої функціональності та датчиками у приміщенні. Наприклад, якщо температура стає нижче за встановленну за замовчуванням, повітря будет нагріватися за допомогою може термостата, та досягнувши показника, термометр повинен перейти у стан спокою, припинивши свою роботу.

У розумній вентиляції потрібно керувати наступними показниками:

- Температура повітря;
- вологість повітря;
- концентрація CO_2 у повітрі.

1.2 Огляд додатку системи розумної вентиляції HealthBox 3.0

Розумна вентиляція з Healthbox (рис. 1.1) - адаптивна система вентиляції, що працює, коли це необхідно і там, де це дійсно необхідно[6].

Без людей повітря в будинку практично не забруднюється, рівень CO_2 і вологості не зростає, в цей час система автоматично переходить в економічний режим роботи, не віддаючи тепло на вулицю. Після повернення мешканців Healthbox самостійно розпізнає збільшення рівнів CO_2/вологості або летких органічних сполук та переходить у більш інтенсивний режим роботи.

Вентиляція Healthbox контролює наступні показники повітря:

1) рівень вологи;
2) рівень кисню;
3) запахів (voc) окремо для кожного підключеного.

В інтерфейсі користувача відображається невеликий функціонал. В інтерфейсі користувача є три екрани, які відображають інформацію про поточний стан параметрів показника, невеликий графік статистики станів рівню того чи іншого параметру.

Відповідно до показника, екран має свій колір. Це зроблено для того, щоб користувачі, що давно користуються програмою, могли не читаючи тексту, зрозуміти поточне значення якого параметру зараз відображено на екрані.

Рисунок 1.1 – Розумна вентиляція HealthBox 3.0

Маючи невеликий функціонал системи вентиляції, для користувача не пропонується багато дій у додатку або можливостей дізнатися більше інформації. Все виконано у лаконічному інтерфейсі без додаткових дій керування.

1.3 Огляд додатку системи розумної вентиляції Nero

Додаток Nero (рис 1.2) забезпечує керування не тільки розумною вентиляцією. Також він забезпечує керування світлом, розетками та побутовими приладами.

В додатку на перший головний екран розробник вирішив розмістити всі прилади, якими можна керувати, не грубуючи їх за будь-якими особливостями або призначеннями.

Підсвічуючи червоним кольором, розробник показує доступність керування до приладів, сірим кольором показані прилади, що можна керувати, але до них немає доступу або вони не підключні до системи керування ними. Рішення показувати на екрані обидва сценарії змішано погане через наступні причини:

- якщо до системи керування підключено всього декілька приладів, користувач може довго шукати їх в загальному списку, щоб налаштувати;
- не розподіляючі картки по категоріям, розробник збільшує кількість дій на екрані та ускладнює роботу користувача;

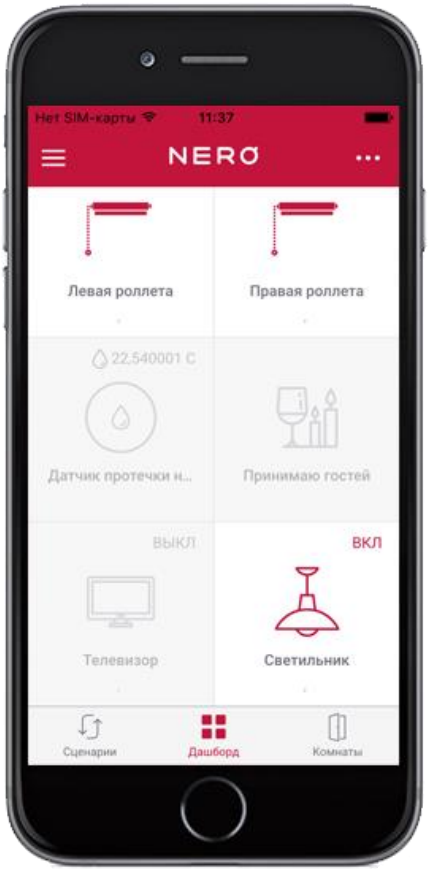


Рисунок 1.2 – Додаток системи вентиляції Nero Electronics

1.4 Огляд системи розумної вентиляції SmartMi Air System

Припливний очищувач повітря - новий тип пристрою в екосистемі розумного будинку Xiaomi (рис. 1.3). Він забирає свіже повітря з вулиці, проганяє його через фільтри та подає свіже та чисте повітря до квартири.

Для фільтрації повітря використовується трикомпонентний фільтр, який може блокувати PM2.5, PM0.3 та інші шкідливі речовини, а також обробляти деякі легкі гази, наприклад, формальдегід і TVOC[7].

В інтерфейсі користувача додатком на головному екрані показано поточний стан параметру повітря. В кольоровому колі найбільшим шрифтом виділені цифри привертають найбльше уваги. Саме це потрібно найпер за все показати користувачу.

Нижче розташовані інші показники. А ще нижче пропонується дія – вимкнути або увімкнути систему вентиляції.

Рисунок 1.3 – Розумна вентиляція SmartMi Air System

1.5 Огляд веб-додатку сервісу Gmail

Розглянемо принципи інтерфейсу кеористувача додатку, яким користуємося щоденно. А саме Google Mail.

Відкриваючи сторінку додатку, користувач побачить перед собою форму входу (рис. 1.4). Форма виділена в рамочку задля того, щоб тримати увагу користувача на заповненні цієї форми. Навіть якщо на сторінці більше нічогісінько не відображається, така форма на білому фоні без рамочки буде гірше тримати фокус уваги, бо сторінка буде необмежена на екрані.

Далі у формі треба вказати логін своєї поштової адресу або увійти за номером телефону. На цій сторінці користувачу не пропонується вводити пароль, це потрібно для того, щоб користувач працював на цьому екрані зі своїм логіном. Тут пропонується відновити його, якщо користувач його забув.

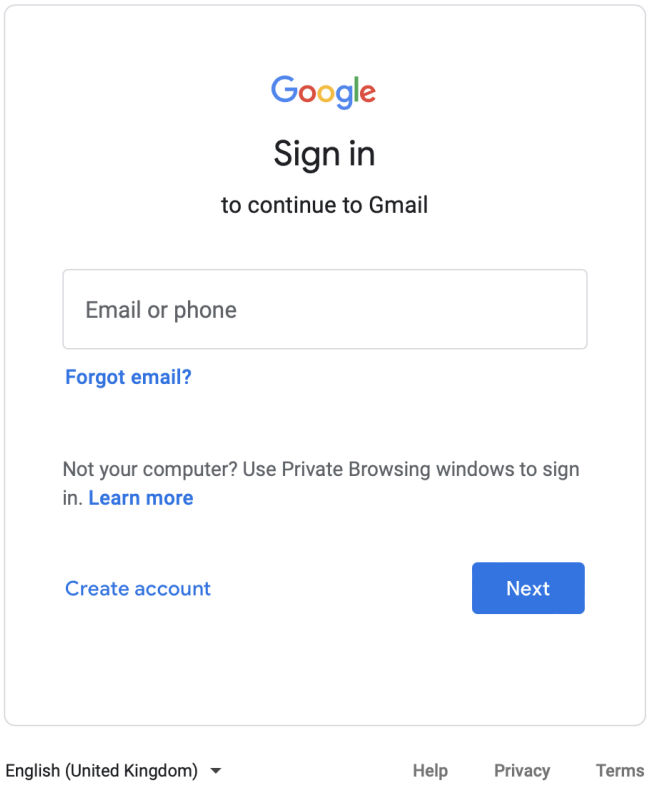


Рисунок 1.4 – Форма входу в додаток пошти

Але якщо це новий користувач в системі, одразу в цій форму пропонується створити акаунт нового користувача. Натиснувши посилання, спочатку відкривається випадаюче меню з можливістю обрати для кого створюється акаунт: для дитини (до 13 років), для себе чи для бізнесу. Відштовхуючись від вибору, алгоритми гугл будуть пропонувати різну політику конфеденційності та відображення медіа.

Наприклад, обравши можливість створити акаунт для дитини, його можна підключити до батьківського акаунту. Батьки зможуть керувати правилами використання тих чи інших сервісів Google для дитини. Це допоможе батькам забезпечити дитину від непотрібного контенту. Або обравши бізнес аканут, користувач буде отримувати більше пропозицій від сервісів Google для просування та роботи з і своїм акаунтом.

Після вводу логіна користувача, відкривається сторінка з привітанням користувача та формою для вводу пароля (рис. 1.5).

На цій сторінці користувачу також пропонується команда для відновлення паролю, якщо користувач його забув. Також тут вже йде пряме звернення до користувача через його особовий логін облікового запису.

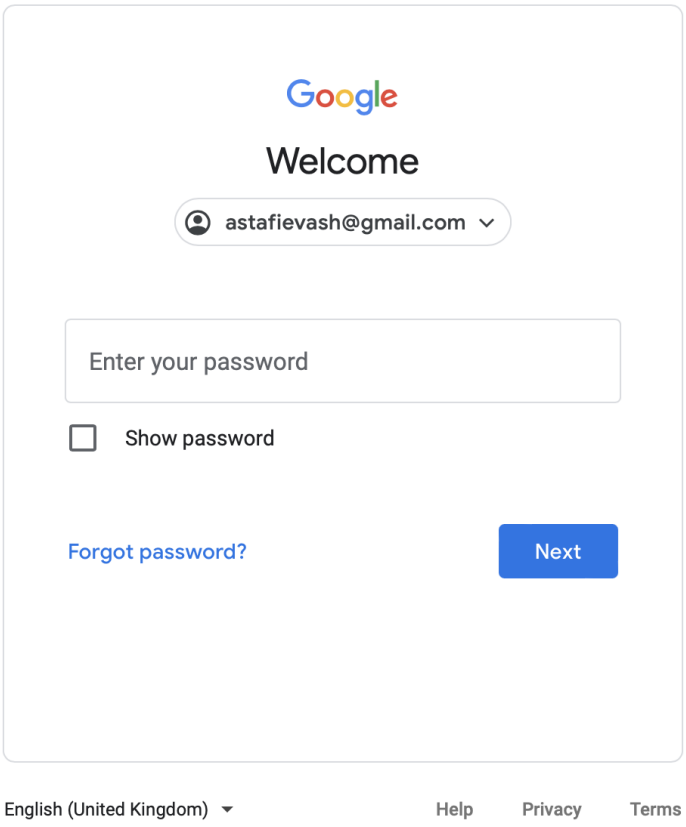


Рисунок 1.5 – Форма вводу пароля користувача

Після того, як користувач увійшов до свого облікового запису, перед ним з'являється перший екран з переліком листів – це основна потреба, за якою користувач користується сервісом Google Mail, тому на першому екрані додатку користувачу на більшій частині екрану показано саме перелік листів (рис. 1.6).

Зліва меншу частину екрану займане головне меню. Через нього можна знайти надіслані листи, отримані, обрані,чернетки та інші.

За допомогою таких категорій не доводиться змішувати на одній сторінці різні види листів, та це полегшує користувачу роботу – йому не доводиться кожен раз запевнюватися, що він читає листа, що вже надіслав, а не варіант в чернетках.

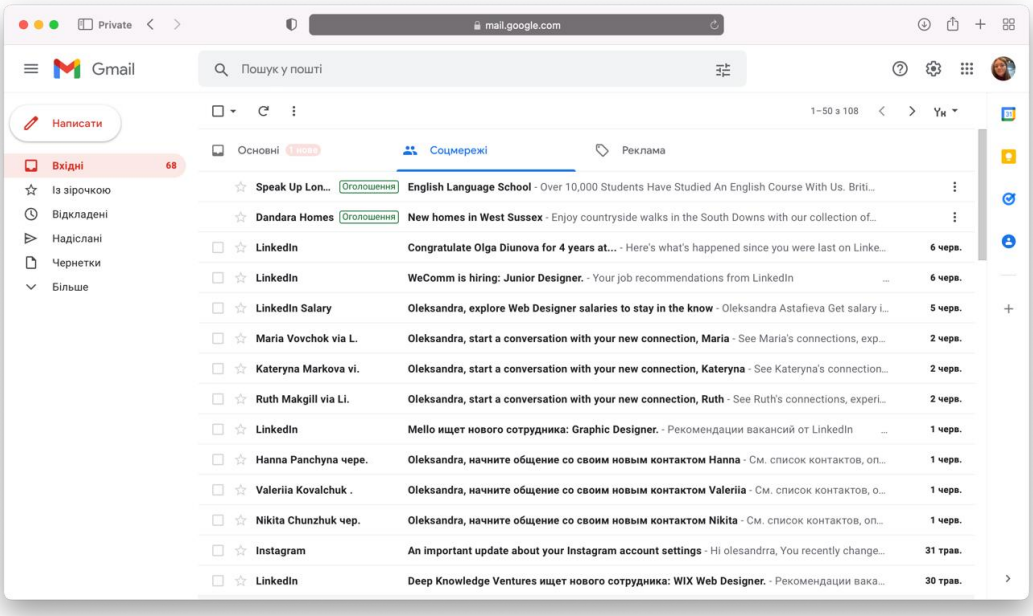


Рисунок 1.6 – Перший екран інтерфейсу користувача Gmail

1.6 Висновки

Проаналізувавши розглянуті вище системи вентиляції, побачимо недоліки та переваги кожної з них для урахування при розробці інтерфейсу користувача. Дійдемо до наступних висновків:

1) більшість платформ не надає користувачеві змогу налаштування параметрів;

2) з огляду на те, що кожна платформа намагається охопити якомога більше параметрів керування, вони концентруються на регулюванні різних важливих параметрів;

3) жодна з систем не має змоги для керування системою вентиляції дистанційно;

4) жодна з систем не має змоги охопити більше параметрів, аніж закладено.

Переглянувши існуючі рішення веб інтерфейсів користувача візьмемо до уваги основні сценарії представлення послуг для користувача та виділимо основні вимоги до проектування інтерфейсів користувача, а саме:

- основна інформацію, що представляє інтерфейс, має бути розташована в центрі екрану;
- основне меню навігації по розділам розташовується зліва;
- всі важливі елементи або основні кнопки, якими найчастіше користуються, мають бути найбільш помітними.

2 ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ КЕРУВАННЯ

На етапі аналізу предметної області було виділено основні завдання, які має вирішувати Web-інтерфейс користувача, та визначено технічні вимоги. Моделювання предметної галузі полегшує вибір інструментів та технологій, які будуть застосовуватись для реалізації програми.

2.1 Архітектура Web-додатку інтерфейсу користувача

Програмний код Web-додатку інтерфейсу користувача виконується на віддаленому сервері, не використовуючи ресурсів комп'ютера користувача.

Розробляюча система складається з архітектури «клієнт-сервер», в якої мережеве навантаження розподілене між постачальниками послуг - сервісами, званих серверами, і замовниками послуг, званих клієнтами.

Клієнтом є браузер користувача, де за допомогою JavaScript забезпечується взаємодія користувача з даними. Функціонування більшості Web-додатків з JS стає більш приємним. Як середовище взаємодії клієнта з сервером використовується мережа Internet (рис. 2.1).

Рисунок 2.1 – Архітектура клієнт-сервер

Основними перевагами архітектури «клієнт-сервер» є:

- можливість розподілити функції обчислювальної системи між кількома незалежними комп'ютерами в мережі. Це дозволяє спростити обслуговування обчислювальної системи - заміна, ремонт, модернізація або переміщення сервера не зачіпають клієнтів;
- всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів. На сервері простіше забезпечити контроль повноважень, щоб дозволяти доступ до даних лише клієнтам із відповідними правами доступу;
- дозволяє поєднати декілька комп’ютерів мережі. Використовувати ресурси одного сервера часто можуть клієнти з різними апаратними платформами, операційними системами тощо.

Основні недоліки архітектури «клієнт-сервер»:

- у разі використання централізованої системи, непрацездатність основного сервера може зробити непрацездатним Web-додаток;
- висока вартість обладнання.

2.2 Програмування інтерфейсу

Мова Python це мова програмування високого рівня, яка підтримує парадігми структурного, процедурного та функціонального програмування. Це інтерпретуєма мова у порівнянні з мовою C++ або Java, які являються компілюємими мовами, що взагалі змушує Python виконувати код повільніше, ніж вони, але з іншого боку надають безкрайні можливості для програмування.

Python, також служить об’єктно-орієнтованою мовою, яка дозволяє розробникам писати короткий і логічний код. Він широко використовується для створення складних веб-додатків. Завдяки доступності бібліотек та фреймворків, призначених для роботи з Python, розробникам не потрібно

створювати веб-додатки з нуля. Великий обсяг фреймворків та бібліотек гарантує, що розробникам буде легко створювати веб-додатки, іноді навіть повністю не використовуючи код, створювати додатки чи інтерфейс користувача візуально, лише налаштовувати логіку там, де це потрібно. Python використовується для машинного навчання чи штучного інтелекту у веб-додатках чи для створення складних програм, які запускаються на стороні додатку, або на стороні серверу. Давайте ознайомимся з деякими за найбільших компаній, які використовували, чи використовують Python для веб-розробки:

- Instagram: Instagram був повністю побудований за допомогою Python, ефективно використовуючи веб-фреймворк Django, і Instagram досі використовує Python 3 як базу коду. Це дозволяє Instagram бути простим і легким в обслуговуванні, а також спрощує оновлення та впровадження нових функцій. Python також дозволив Instagram стати масово масштабованим, що вимагало підтримки мільйонів активних користувачів;
- Netflix: Python також є кращим вибором для багатьох розробників Netflix через простоту використання. Netflix використовує Python для аналізу даних безпосередньо на стороні сервера. Netflix також використовує Python для включення центрального шлюзу сповіщень, який являє собою систему, яка направляє оповіщення необхідним особам без ризику дублювання. Python також використовується Netflix з метою безпеки та відстеження;
- Spotify: Spotify використовує Python в першу чергу для аналізу даних, систем пропозицій і механізмів рекомендацій. Python широко використовується для серверних служб, які взаємодіють один з одним за допомогою фреймворків і бібліотек, написаних на Python. Spotify віддає перевагу Python через те, що він забезпечує більш швидкі конвеєри розробки, а також синхронність. Spotify відомий тим, що використовує Luigi, відомий модуль Python для інтерпретації величезного обсягу аналітики даних.

Також для зручності та спрощення написання коду, в роботі були винайдені та написані веб-фреймворки. Фреймворки - це програмні продукти, які спрощують створення і підтримку технічно складних або навантажених проектів. Фреймворк, як правило, містить тільки базові програмні модулі, а всі специфічні для проекту компоненти реалізуються розробником на їх основі. Тим самим досягається не тільки висока швидкість розробки, але і велика продуктивність і надійність рішень.

Веб-фреймворк - це платформа для створення сайтів і веб-додатків, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. За рахунок широких можливостей в реалізації бізнес-логіки і високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків і веб-сервісів. Перед веб-розробниками часто стоїть вибір між коробковими CMS і фреймворками для реалізації проекту. У кожного з підходів є свої плюси і мінуси, нижче ми розглянемо переваги і недоліки розробки на фреймворках.

Переваги фреймворків:

- розробка на фреймворку на відміну від самописних рішень дозволяє домогтися простоти супроводжуваності проекту;
- можлива і відносно проста реалізація будь-яких бізнес-процесів, а не тільки тих, які спочатку закладені в систему. Також проекти на базі фреймворків легко масштабовані і модернізовані;
- рішення на фреймворках, як правило, працюють значно швидше і витримують велике навантаження, ніж CMS і самописні системи. Саме тому багато популярних інтернет-магазинів працюють не на коробкових CMS, а на фреймворках. За рівнем безпеки рішення на фреймворках значно перевершують самописні системи і порівнянні з CMS (як правило, сайти на фреймворках навіть безпечніше).

Недоліки фреймворків:

- терміни розробки типового функціоналу на фреймворках більше, ніж при використанні CMS. Фреймворки містять тільки базові компоненти

бізнес-логіки рівня програми, тому багато функцій реалізовуються індивідуально;

- для розробки на фреймворку потрібне розуміння бізнес-процесів, які потрібно реалізувати. Наприклад, якщо в CMS вже є якийсь попередньо встановлений процес обробки замовлень, то фреймворки такого не надають.

Для розробки веб-додатку в роботі використано веб-фреймворк Django. Django — (Джанго) - вільний фреймворк для веб-додатків на мові Python, що використовує шаблон проектування MVC. Проект підтримується організацією Django Software Foundation. Та jinja2 Jinja - це шаблонізатор для мови програмування Python. Він подібний до шаблонізатора Django, але надає Python-подібні вирази, забезпечуючи виконання шаблонів у пісочниці. Це текстовий шаблонизатор, тому він може бути використаний для створення будь-якого виду розмітки та вихідного коду. Ліцензовано за ліцензією BSD.

Сайт на Django будується з одного або декількох додатків, які рекомендується робити відчужуваними і підключаються. Це одна з істотних архітектурних відмінностей фреймворку від деяких інших (наприклад, Ruby on Rails). Один з основних принципів фреймворку-DRY (англ. Don’t repeat yourself).

Архітектура Django схожа на «Модель-представлення-контролер» (MVC). Контролер класичної моделі MVC приблизно відповідає рівню, який в Django називається представлення (англ. View), а презентаційна логіка представлення реалізується в Django (англ. Template). Через це рівневу архітектуру Django часто називають " Модель-Шаблон-подання "(MTV).

В роботі проведено аналіз основних можливостей Django:

- ORM, API доступу до БД з підтримкою транзакцій;
- вбудований інтерфейс адміністратора з уже наявними перекладами на багато мов;
- диспетчер URL на основі регулярних виразів;
- розширювана система шаблонів з тегами і спадкуванням;
- система кешування;

- інтернаціоналізація;
- підключаєма архітектура додатків, які можна встановлювати на будь-які Django-сайти;
- generic views - шаблони функцій контролерів;
- авторизація та аутентифікація, підключення зовнішніх модулів аутентифікації: LDAP, OpenID та ін.;
- система фільтрів (middleware) для побудови додаткових обробників запитів, як, наприклад, включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій;
- бібліотека для роботи з формами (спадкування, побудова форм за існуючою моделлю БД);
- вбудована автоматична документація по тегам шаблонів і моделям даних, доступна через адміністративний додаток.

Для написання коду використовуючи мову програмування Python, та фреймворк Django, найкращим варіантом є продукт компанії Jetbrains, а саме PyCharm. PyCharm-це інтегроване середовище розробки для Python, яке має повний комплект засобів, необхідних для ефективного програмування на Python. Перша версія вийшла в 2010 році. Зараз PyCharm поширюється в двох варіантах: платному (PyCharm Professional Edition) і безкоштовному (PyCharm Community Edition)x.

Безкоштовна версія має відкритий вихідний код і поширюється під ліцензією Apache 2. Це полегшене середовище, яке підходить для розробки тільки на Python.

Платний варіант являє собою більш розширену і функціональну версію з можливістю розробки в тому числі багатомовних веб-додатків. Professional Edition підтримує фреймворки:

- Django;
- Flask;
- Google App Engine;
- Pyramid;

- web2py.

І дає можливість віддаленої розробки, а також роботи з базами даних. У таблиці 1 надана порівнювальна характеристика двох версій середовища.

Таблиця 1 - Порівняльна характеристика версій середовища.

Версія продукту Функціонал	PyCharm Professional Edition	PyCharm Community Edition
Функціональний редактор Python	+	+
Інструмент запуску тестів та графічний відладчик	+	+
Навігація по коду та рефакторінг	+	+
Інспекція коду	+	+
Підтримка систем контролю версій	+	+
Інструменти для наукових обчислень	+	-
Веб-розробка	+	-
Веб-фреймворки Python	+	-
Python-профілювальник	+	-
Можливості віддаленої розробки	+	-
Підтримка баз даних та SQL	+	-

2.3 Функціональні вимоги

Інтерфейс користувача керування розумною вентиляцією повинна задовольняти наступним функціональним вимогам:

- система повинна надавати користувачеві змогу регістрації та авторизації з декількох пристроїв;
- система має надавати користувачеві змогу керування приладами у будинку задля забезпечення системи вентиляції;
- система має аналізувати данні за деякий період та показувати користувачеві графіки данних;

- система повинна дозволяти користувачеві налаштовувати параметри повітря у будинку;
- система повинная дозволяти користувачеві змунювати особисті данні;
- система повинна мати змогу забезпечити кросплатформеність.

2.4 Нефункціональні вимоги

У результаті аналізу функціональних вимог та аналізу схожих існуючих проектів, були сформульовані наступні нефункціональні вимоги:

- наявність серверної частини, що допускає передачу даних з бази та їх створення та зміна на запит клієнта;
- наявність клієнтської частини, що відправляє запити на сервер;
- веб-додаток повинен мати кросбраузерну та адаптивну верстку для правильного виведення сторінок на пристрій.

2.5 Варіанти користування системою

Варіант використання додатку двома типами користувачів (рис. 2.2). Перший – незареєстрований користувач. Незареєстрований користувач має можливість зареєструватися. На сторінці головного екрану показано форму реєстрації до додатку та інформацію про додаток.

Другий – зареєстрований у системі користувач. На сторінці головного екрану зареєстрований користувач має можливість увійти у свій обліковий запис за допомогою форми входу, також на сторінці головного екрану можна прочитати інформацію про додаток.

Для будь-якого користувача додатком відображається інформація про додаток. Вона відображається під екраном з формою входу або реєстрації.

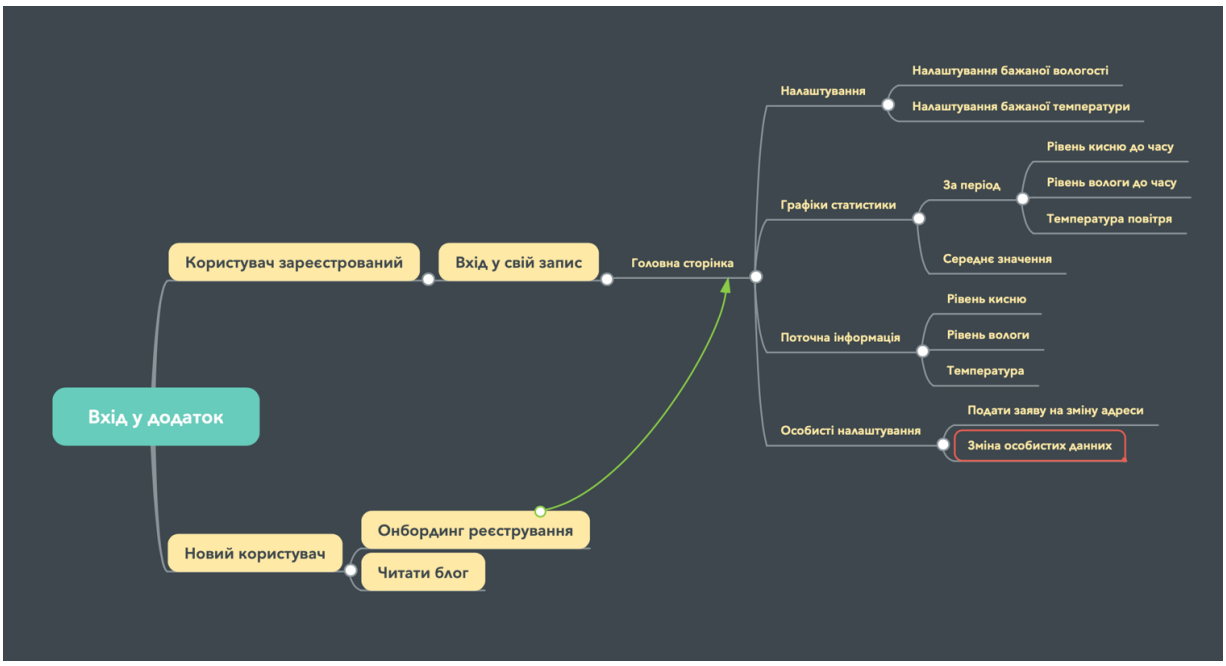


Рисунок 2.2 – Мапа шляху користувача

Незареєстрований користувач має змогу переглянути інформацію про автоматизоване керування вентиляцією.

Зареєстрований користувач може керувати особистими данними – змінювати своє ім’я, дату народження, вік та подати запит на зміну адреси.

Зареєстрований користувач може керувати параметри повітря за замовчуванням, встановлюючи бажані данні.

За допомогою мапи сценарію користування додатком є змога відслідкувати всі можливі сценарії розвитку подій, які доступні користувачу. Та як користувач може дійти до тої чи іншої цілі.

За допомогою мапи сценарів поведінки користувача можна передбачити помилки в користуванні інтерфейсом користувача та сценарії, в яких у користувача можуть виникнути складнощі під час користування інтерфейсом.

Наприклад, якщо користувач хоче дізнатися середнє значення температури повітря у будинку за деякий період йому потрібно зробити наступні рухи: з екрану головної сторінки перейти до вкладки «Графік статистики», там обрати кнопку «За період», а потім обрати рівень якого

параметру потрібно подивитися – температури повітря. Якщо тестування покаже, що у користувача виникли складнощі з пошуком потрібних кнопок, потрібно передивитися мапу сценраію та додати більш очевидного функціоналу та вказівок наступної дії користувача.

3 РЕАЛІЗАЦІЯ ДОДАТКУ КЕРУВАННЯ ВЕНТИЛЯЦІЄЮ

3.1 Проектування інтерфейсу користувача

Інтерфейс користувача насамперед складається з головної сторінки. Головна сторінка, як візитна картка додатку. Тож найкращим рішенням було одразу предоставити користувачу дію, яку він може зробити – заповнити форму входу та увійти у свій додаток (рис. 3.1).

У формі перш за все повинен бути пристуній заголовок та невеликий підзаголовок. Це потрібно для того, щоб користувач бачив дію, яку він зараз буде робити та що ця дія означає. Якщо з заголовку незрозуміло, користувач читає підзаголовок з поясненням. Основою форми є поля для заповнювання. Коли користувач пише в них свої данні, система перевіряє чи існує в базі даних такий користувач. Далі пароль, після натискання кнопки «Увійти» системою перевіряється чи правильний пароль введенний.

Якщо користвач не зареєстрований у базі даних, йому пропонується перейти за посиланням реєстрації під формою. Після чого відкривається вікноз формою реєстрації. Де користувач має вказати свої дані для реєстрації. Це його ім’я та прізвище, адресу, номер телефону, кількість пристроїв, встановлених у приміщенні.

У правій половині екрану зверху знаходитиметься меню з навігацією по сайту та велика ілюстрація для більш приємного вигляду екрану.

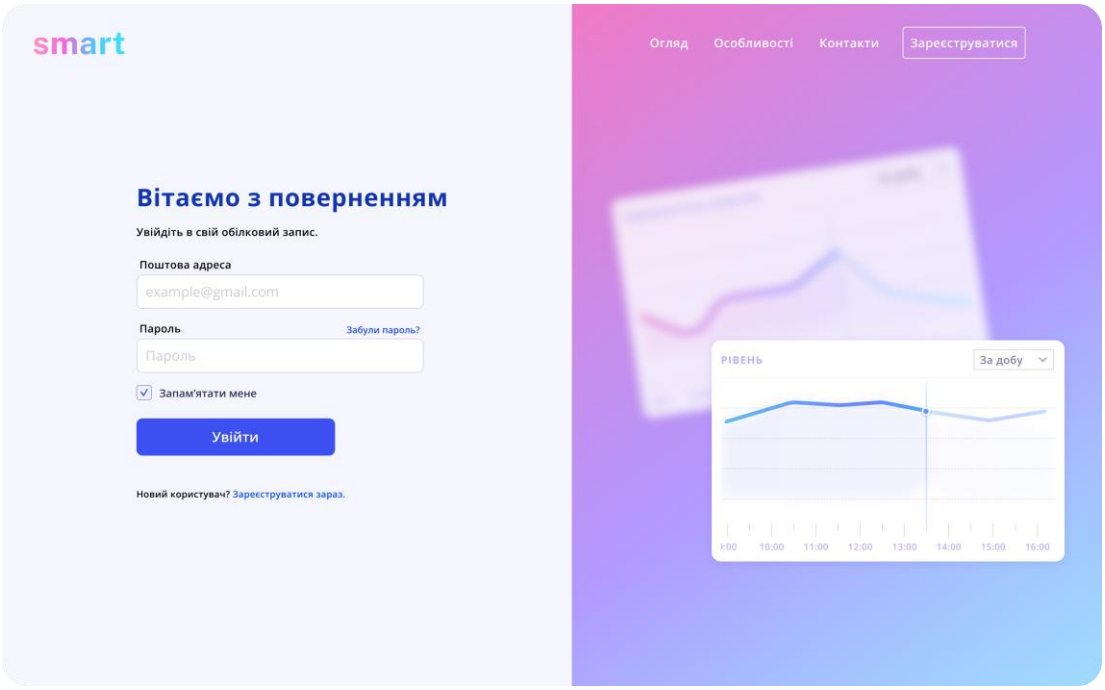


Рисунок 3.1 – Форма входу користувача

Після входу в свій обліковий запис відкривається перша сторінка додатку з поточною інформацією даних показників вентиляції (рис. 3.2).

В лівому меню пристуня навігація по розділам додатку. Перша вкладка «Поточна інформація» відображає інформацію показників на данний момент часу.

На першій сторінці краще за все відображати всю важливу інформацію таку як: показники даних повітря, вологи та кисню. Також показати графіки зміни показників, та дати користувачу змогу змінти їх. Це потрібно для того, щоб користувач не потребував робити багато рухів для досягнення тіє чи іншої мети. Весь основний функціонал винесено на головну сторінку.

Саме те, що в першу чергу задовольняє інтерфейс користувача, а саме:

- перегляд поточних даних параметрів повітря у будинку;
- перегляд динаміки зміни параметрів повітря у будинку;
- змога перейти до екрану налаштування параметру.

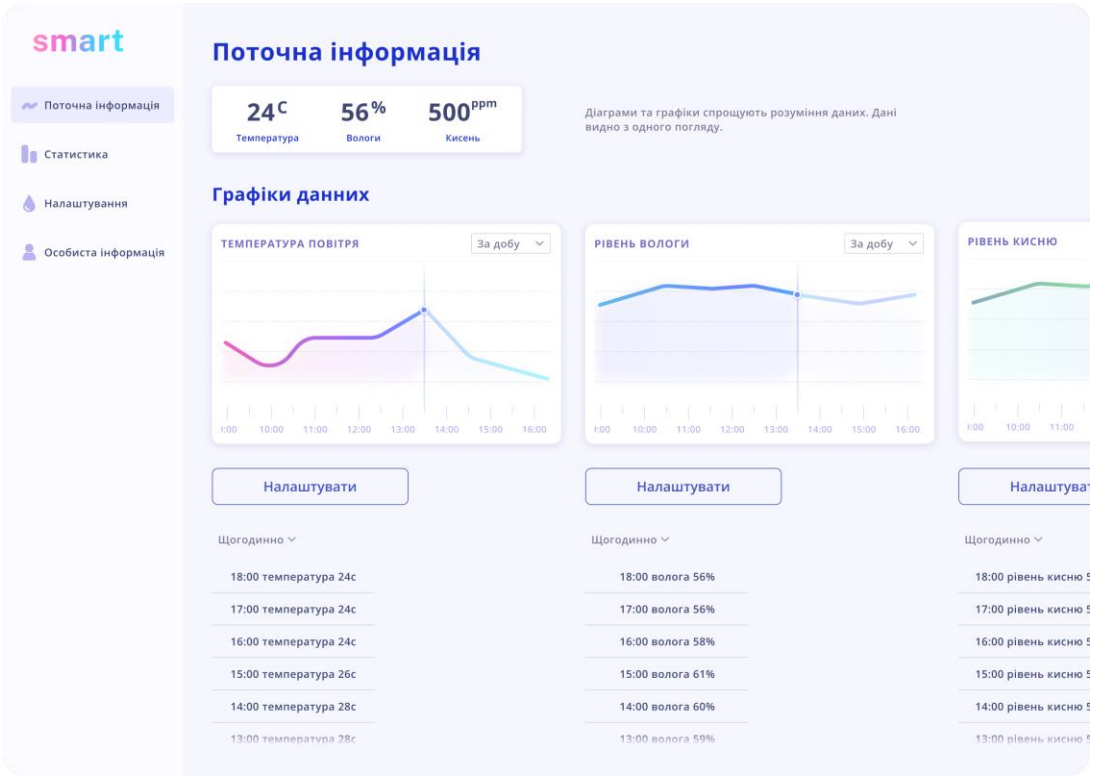


Рисунок 3.2 – Поточна інформація параметрів

На сторінці «Налаштування» користувач може встановити потрібні значення параметрів для забезпечення комфорту у приміщенні. (рис. 3.3) На цій сторінці відображені встановленні поточні значення та пермикачі, за допомогою яких можна змінити значення параметрів повітря у приміщенні.

У кваліфікаційній роботі пермекачі параметрів обрані у вігляді колеса, так як для взаємодії користувача цей елемент буде простішим для розуміння.

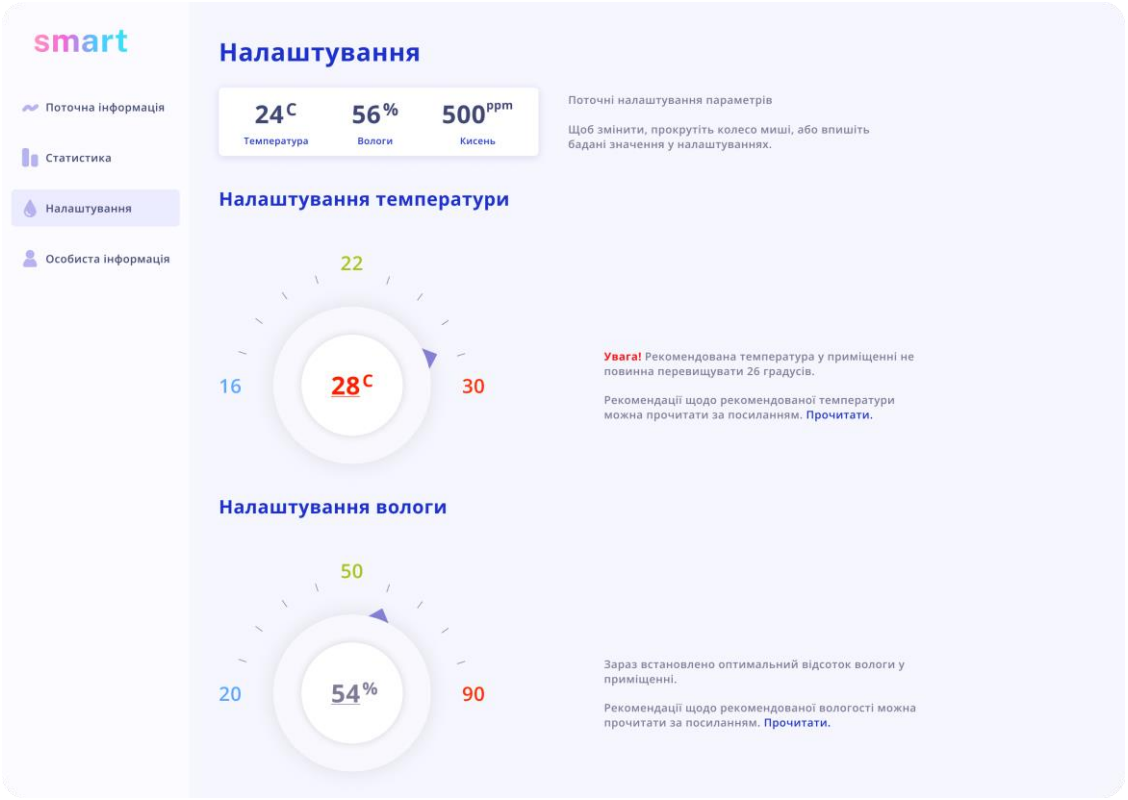


Рисунок 3.3 – Екран «Налаштування»

На сторінці з особистою інформацією користувач може редагувати власні дані (рис. 3.4).

Змінити своє ім’я та прізвище можливо без додаткових дій. Але для зміни адреси потрібно підтвердити особистість, ввівши пароль (рис. 3.5). Це потрібно для забезпечення безпеки прихованної адреси проживання користувача.

Дані адреси є дуже важливими в базі даних, так як аби скористатися обслуговуванням пристроїв та датчиків потрібно знати її місцезнаходження. А також це потрібно для забезпечення безпеки користувача – перегляд адреси після введеня пароля дозволяє не розкривати іншим людям цю інформацію при демонстрації екрану цієї сторінки.

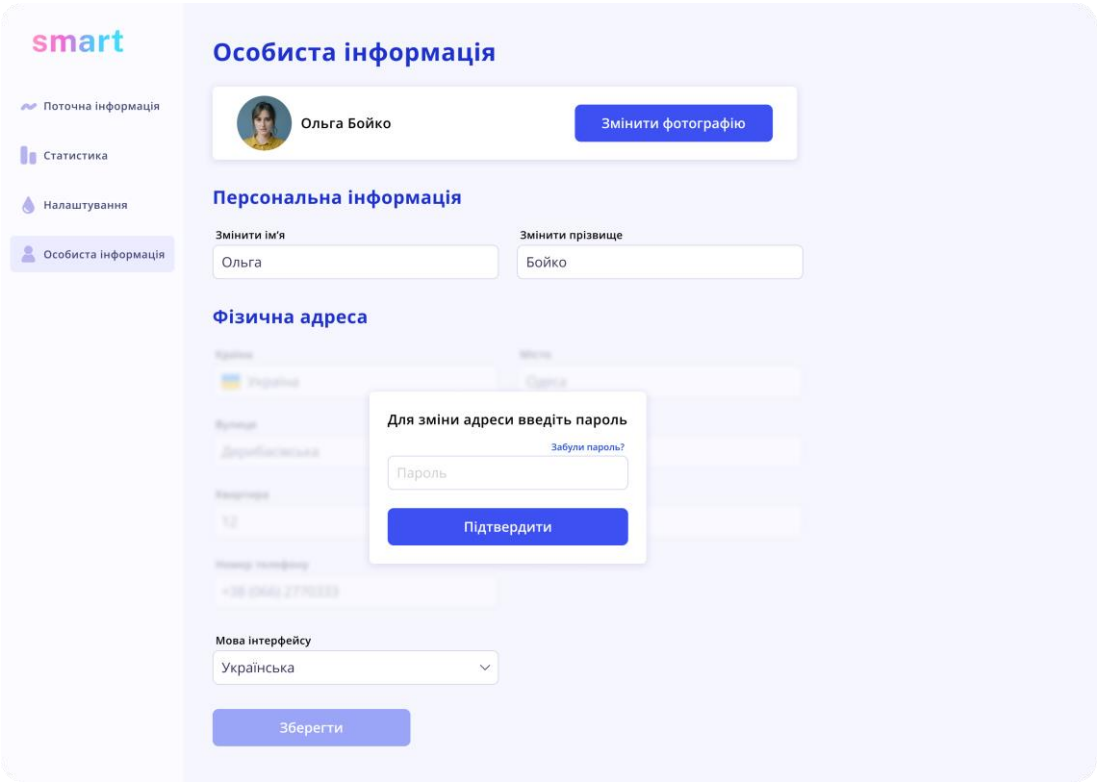


Рисунок 3.4 - Налаштування особистої інформації

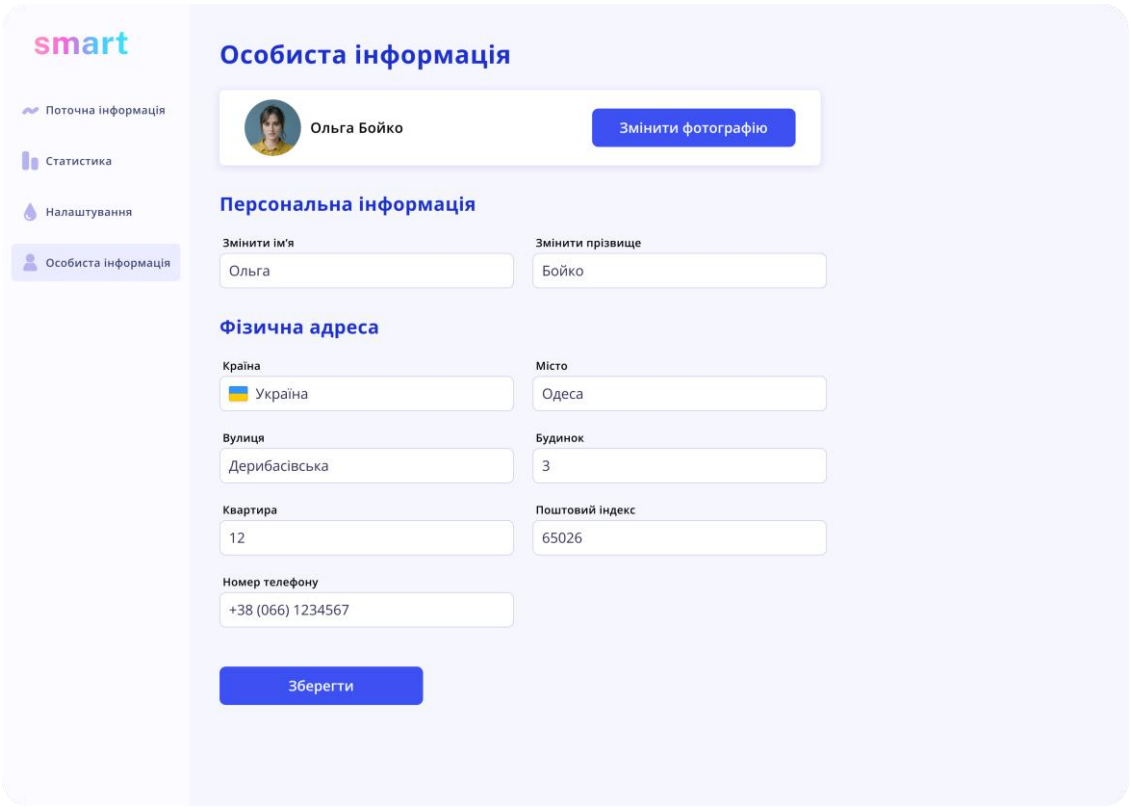


Рисунок 3.5 – Зміна фізичної адреси

Спроектований інтерфейс користувача описано мовами верстування для реалізації у веб-браузерах. Задля більш комфортного користування на різних присторях, верстку адаптовано для мобільної версії.

Кросплатформеність – можливість відображати працю програмного забезпечення на декількох пристроях.

3.2 Програмне забезпечення інтерфейсу користувача

Інтерфейс користувача написано мовою верстування HTML та CSS. Програмна частина інтерфейсу написана мовою Python.

На першій сторінці, що була розглянута у попередньому підрозділі потрібно мати код, що надсилатиме та прийматиме інформацію з форми. Для відправки запиту на реєстрацію, використовуємо технологію Ajax jQuery – це технологія обміну даними з сервером без оновлення сторінки. Обмін даними з сервером відбувається асинхронно.

Лістинг 3.1 - Код функції відправки даних при реєстрації.

```
$(function() {
    $('#btnSignUp').click(function() {

        $.ajax({
            url: '/signUp',
            data: $('form').serialize(),
            type: 'POST',
            success: function(response) {
                console.log(response);
            },
            error: function(error) {
                console.log(error);
            }
        });    });    });
```

Завантажити данні з сервера у обраний елемент можна за допомогою методу jQuery load()

```
$(selector).load(URL,data,callback);
```

Де параметр URL відповідає за адресу, яку потрібно завантажити, а Data – данні у форматі пар ключів і значень для відправки разом з запросом.

Для прийому даних та їх подальшого зберігання у базі даних необхідно написати backend-сторону WEB-додатку. За допомогою «VirtualEnv» створюємо окреме оточення для розробки. VirtualEnv використовується для створення віртуальних оточень для Python програм. Це необхідно для уникнення конфліктів, дозволяючи встановити одну версію бібліотеки для однієї програми, і іншу для другої.

Створення окремого оточення виконується в робочому каталозі за допомогою команд, представлених у лістингу 3.2

Лістинг 3.2 – Команди для створення віртуального оточення.

```
~/<work_folder_name>$ mkdir venv project
~/<work_folder_name>$ virtualenv --
prompt="(venv:<work_folder_name)" ./venv/
New python executable in ./venv/bin/python
Installing setuptools.............done.
Installing pip...............done.
~/<work_folder_name>$ source ./venv/bin/activate
(venv:<work_folder_name>)~/<work_folder_name>$
```

Наступним етапом налаштування оточення розробки є створення файлу залежностей. У файлі на кожному рядку вказується по одній залежності, для початку нам необхідно додати Django до цього файлу. Тому наш файл матиме такий вигляд:

```
Django
```

Якщо необхідна якась спеціальна версія бібліотеки, або додаток не підтримує інші версії, можна явно вказати версію бібліотеки, наприклад.

```
Django==4.0.4
```

Під час розробки, якщо будуть використовуватися додаткові бібліотеки, їх необхідно записувати у файл залежностей для подальшого встановлення їх під час встановлення додатку на сервер.

Для встановлення всіх залежностей необхідно запустити команду

```
(venv:<work_folder_name>)~/<work_folder_name>$ pip install -U -r requirements.txt
```

3.3 Налаштування проекту Django

У фреймворці Django проект – це кінцевий продукт, і він об’єднує у собі один чи декілька додатків. Після встановлення фреймворку генерується скрипт django-admin.py, який використовуєтьсядля для обробки задач саффолдингу. Скаффолдинг - метод метапрограмування для створення веб-додатків, що взаємодіють з базами даних. Метод передбачає завдання розробником специфікації, за якими надалі генерується програмний код для операцій створення певних записів у базі даних, їх читання, оновлення та видалення (CRUD). Через цей скрипт можливо створити проект, який мативе наступну структуру:

```
manage.py
/<project_name>
__init__.py
settings.py
urls.py
wsgi.py
```

- manage.py - є посиланням на скрипт django-admin, але з уже встановленими змінними оточення, що вказують на ваш проект, як для читання налаштувань звідти, так і для управління ним при необхідності;
- settings.py -тут знаходяться налаштування вашого проекту. Файл вже містить кілька розумних налаштувань, але база даних не вказана;
- urls.py -містить URL'И для маппірованія (відображення) уявлень: ми незабаром (в подальших главах) поговоримо про це докладніше;

- wsgi.py - це WSGI обгортка для вашої програми. Цей файл використовується сервером розробки Django і можливо іншими контейнерами, такими як mod_wsgi, uwsgi та ін. на «бойовому» сервері.

Наступним кроком є створення додатку, для цього використовується скрипт manage.py. Структура додатку буде наступна:

- ./<app_name>
- __init__.py
- models.py (містить Django ORM-моделі для програми);
- tests.py (містить написані модульні та інтеграційні тести);
- views.py (містить код подань);
- admin.py (містить модель для адміністративного інтерфейсу);
- migrations (містить файли міграцій)

3.4 Моделі Django

Веб-додатки Django отримують доступ та керують даними через об’єкти Python, які називаються моделями. Моделі визначають структуру даних, типи полів та при необхідності їх максимальний розмір, значення за-замовчуванням, параметри списку вибору, текст довідки та ін. Оголошення моделі не залежить від основної бази даних – після вибору бази даних не є необхідності працювати з нею безпосередньо, необхідно написати структуру моделі та код, а Django зробить решту роботи пов’язану з базою даних.

Моделі визначаються у файлі `models.py`. Вони реалізуються як підкласи `django.db.models.Model`, та можуть включати поля, методи та метадані. У приведеному нижче лістингу 3.4 показана «типова модель» названа `Model_Name`.

Лістинг 3.4 – «типова» модель `Model_Name`

```
from django.db import models
     class Runner(models.Model):
```

```
        date_time_read = models.CharField(max_length=30,
primary_key=True)
        temperature = models.DecimalField(max_digits=5,
decimal_places=2)
        humdity = models.DecimalField(max_digits=5,
decimal_places=2)
```

Також для додавання додаткового функціоналу можливо додати функції одразу до класу моделі. Для збереження змін та створення необхідних таблиць у базі даних необхідно зареєструвати додаток у налаштуваннях проекту, тобто необхідно додати назву нашого проекту у файл `settings.py`. Це налаштування матиме вигляд:

```
INSTALLED_APPS = (
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.messages',
  'django.contrib.staticfiles',
  '<app_name>',
)
```

І лише після додавання додатку у список необхідно створити міграцію, це робиться викликом команди `python ./manage.py makemigrations`, та мігрувати за допомогою команди `python ./manage.py migrate`

3.5 Налаштування jinja

За допомогою jinjia можна описати верстування сторінки з використанням вмісту бекенду, що надходить з попередньо розглянутої кваліфікаційної роботи Коваленко Івана.

В першу чергу відбувається завантеження сторінки. Завантаження сторінки написано за допомгою шаблону jinja2. Базовий клас для всіх

завантажувачів get_TempSource підкласу. Цей метод викликає метод load завантажувальника для отримання об’єкту Template, що містить в собі основне джерело шаблону. Лістинг коду завантаженя сторінок предоставлено у додатку Б.

Клас TemperatureLoader містить в собі функцію для завантаження налаштувань температури. Відповідно до інших параметрів маємо відповідні класи, а саме: клас WetnessLoader для завантаженя налаштувань вологи та клас OxignLoader для завантаженя налаштувань провітрення.

Задля забезпечення повернення користувача до поточного екрану та забезпечення збереження налаштувань, використовується невеликий шаблон, яким поєднується декілька розділів. Тобто повертається строка при кожному виклику.

Лістинг коду функції, що зберігає стан поточного налаштування на сторінці

```
{% set Temperature = joiner("|") %}
{% if categories %} {{ Temperature() }}
    Categories: {{ categories|join(", ") }}
{% endif %}
{% if SmartVentilation %} {{ Temperature() }}
    Author: {{ SmartVentilation() }}
{% endif %}
{% if can_edit %} {{ pipe() }}
    <a href="?action=edit">Edit</a>
{% endif %}
```

У команді {% set pipe = joiner("|") %} створюється контейнер, що дозволяє назначати атрибут тегу.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи розглянуто та сформовано основні вимоги для розробки інтерфейсу користувача керування системою розумної вентиляції у будинку. Було розроблено інтерфейс користувача для керування вентиляцією будинку. За допомогою аналізу існуючих рішень було виділено основні недоліки у більшості з них та сформовані наступні вимоги до розробляючого програмного забезпечення, до яких варто віднести наступне:

- надати користувачу змогу налаштовувати параметри керування приладами для забезпечення мікроклімату у будинку самостійно;
- додати змогу кервування вентиляцією дистанційно;
- розташувати основні елементи керування з огляду на більш важливі частини екрану девайсу користувача;
- розробити незалежний додаток, за допомогою якого можна керувати присторями вентиляції.

Розроблений інтерфейс користувача має наступні переваги:

- забезпечує користувача лаконічним інтерфейсом та вочевидним розташуванням елементів взаємодії з додатком;
- забезпечує користувача кросплатформеністю;
- не викликає у користувача складнощі з керуванням такої системи, як вентиляція.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ren Chevance. Server Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, Storage Solutions / Ren Chevance – Elsevier/Digital Press, 2005. – 690 p.

2. SMART HOUSE CONTROL SWorldJournal, 1(08-01), 28–32. [Електронний ресурс] Режим доступу: https://www.sworldjournal.com/index.php/swj/article/view/swj08-01-033/1216

3. Архітекутра систем [Електронний ресурс] Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/

4. Астаф'єва О.Д., Керування системою «Розумний будинок» / О.Д. Астаф'єва, Т.В. Луценко, І.О. Коваленко // SMART HOUSE CONTROL. SWorldJournal, 1(08-01), 28–32.

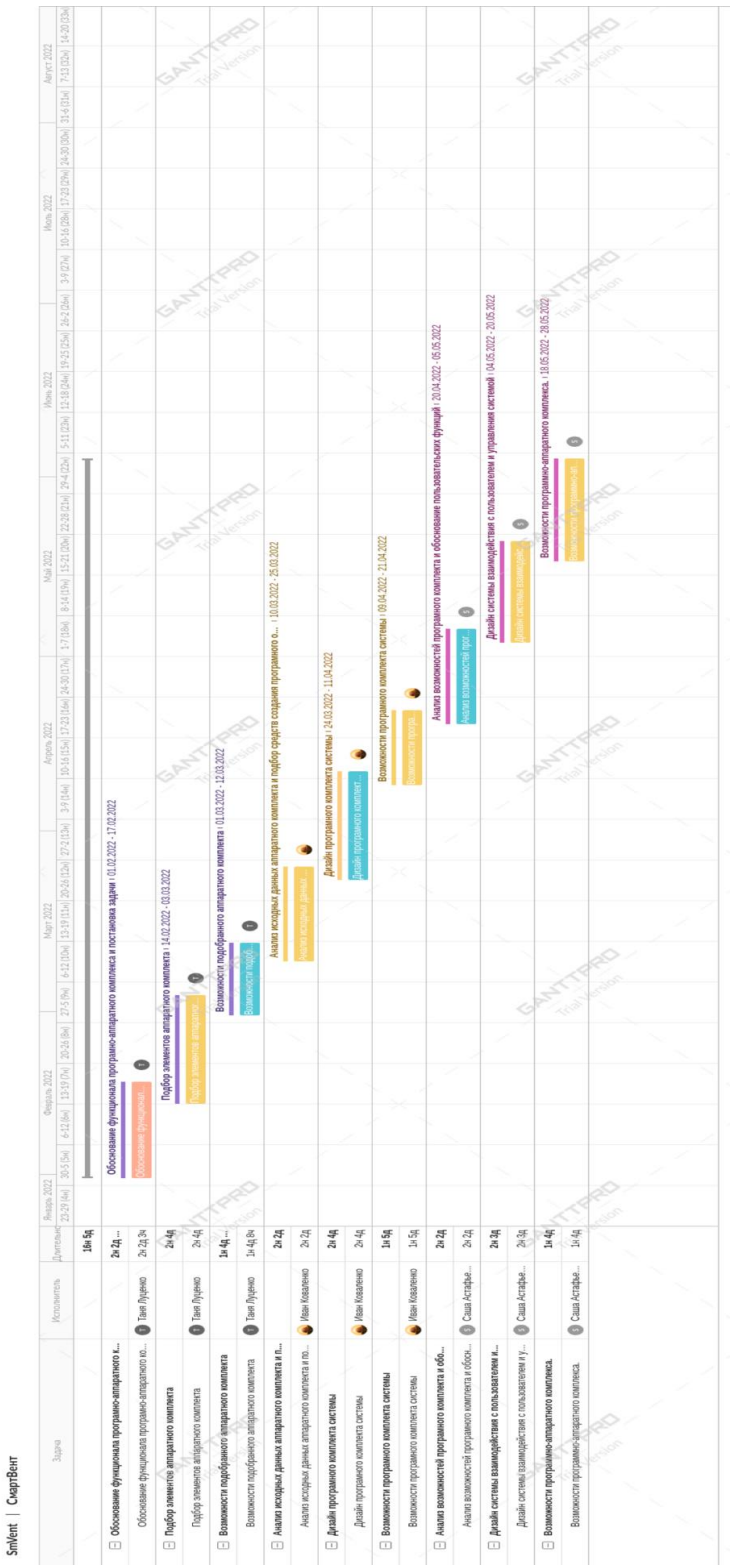
5. Національна бібліотека ім. М. Баумана. [Електронний ресурс]. - 2017.- Режим доступу : https://ru.bmstu.wiki/. –Дата доступпу: квітень 2019.

6. Характеристики Healthbox 3.0 [Електронний ресурс] Режим доступу: https://smart-vent.com.ua/

7. Характеристики Xiaomi Smarthome [Електронний ресур] Режим доступу: https://xiaomi-smarthome.ru/smartmi-air-system/

ДОДАТОК А

Діаграма Ганта



ДОДАТОК Б

Лістинг коду завантаженя сторінок

```
from jinja2 import BaseLoader, TemplateNotFound
from os.path import join, exists, getmtime

class TemperatureLoader(BaseLoader):
    def __init__(self, path):
        self.path = path
    def get_TempSource(self, environment, template):
        path = join(self.path, template)
        if not exists(path):
            raise TemplatetNotFound(template)
        mtime = getmtime(path)
        with open(path) as f:
            TemptSource = f.read()
        return TemptSource, path, lambda: mtime == getmtime(path)
class WetnessLoader(BaseLoader):
    def __init__(self, path):
        self.path = path
    def get_WetSource(self, environment, template):
        path = join(self.path, template)
        if not exists(path):
            raise TemplatewNotFound(template)
        mtime = getmtime(path)
        with open(path) as f:
            TempwSource = f.read()
        return TempwSource, path, lambda: mtime == getmtime(path)
class OxignLoader(BaseLoader):
    def __init__(self, path):
        self.path = path
    def get_OxigSource(self, environment, template):
```

```
            path = join(self.path, template)
            if not exists(path):
                raise TemplateoNotFound(template)
            mtime = getmtime(path)
            with open(path) as f:
                TempoSource = f.read()
            return TempoSource, path, lambda: mtime ==
getmtime(path)
```