Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів

Кафедра систем управління літальних апаратів

Пояснювальна записка

до дипломної роботи
магістра
(освітньо-кваліфікаційний рівень)

на тему: «Розробка засобів технічної реалізації та інформаційної підтримки розподілених систем термостабілізації»

ХАІ.301.362.24О.151.00183028 ПЗ

Виконав: студент 2 курсу, групи 362

Галузь знань 15 «Автоматизація та приладобудування»

Спеціальність

151 “Автоматизація та комп’ютерно-інтегровані технології

Освітня програма
“Інженерія мобільних додатків”

Горбенко В. Ю.
(прізвище та ініціали студента)

Керівник Джулгаков В. Г.
(прізвище та ініціали)

Рецензент Ковтун І. В.
(прізвище та ініціали)

Харків – 2024

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів (№301)
Рівень вищої освіти___магістр_____
Галузь знань 15 Автоматизація та приладобудування
(шифр і назва)
Спеціальність 151 “Автоматизація та комп’ютерно-інтегровані технології
(шифр і назва)
Освітня програма_____«Інженерія мобільних додатків»
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

к.т.н., доц. _____Костянтин ДЕРГАЧОВ

“____” _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____Горбенко Вадим Юрійович_____
(прізвище, ім’я, по батькові)

1. Тема роботи « Розробка засобів технічної реалізації та інформаційної підтримки розподілених систем термостабілізації »

керівник роботи _доцент кафедри 301 Джулгаков Віталій Георгійович _____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 06.11. 2023 року № 1968-уч
2. Строк подання студентом роботи: _08 січня 2024 року
3. Вихідні дані до роботи: Мобільний додаток має допускати легку і швидку модифікацію, редагування та видалення контенту без необхідності глибоких знань в області програмування.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Стан проблеми. Аналіз і синтез системи. Конструкторська частина. Дослідницька частина. Експериментально-практична частина (розробка мобільного додатку). Економічне обгрунтування розробки.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) – відповідає форматам А1: Стан проблеми (1). Аналіз і синтез системи термостабілізації (4). Конструкторська частина (1). Дослідницька частина (1). Розробка мобільного додатку (2). Економічне обгрунтування розробки (1)
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	доцент Джулгаков В.Г.	12.09.23	08.01.24
2	доцент Джулгаков В.Г.	12.09.23	08.01.24
3	доцент Джулгаков В.Г.	12.09.23	08.01.24
4	доцент Джулгаков В.Г.	12.09.23	08.01.24
5	доцент Джулгаков В.Г.	12.09.23	08.01.24
6	доцент Джулгаков В.Г.	12.09.23	08.01.24

Нормоконтроль _____ Віталій ДЖУЛГАКОВ «10» _січня__2024 р.
(підпис) (ініціали та прізвище)

7. Дата видачі завдання_____12.09.2023_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Початок переддипломної практики	01.09.2023	
2.	Формулювання теми роботи. Розробка технічного завдання	12.09.2023	
3.	Математичний опис системи управління. Аналіз і синтез системи управління. Проведення експериментальних досліджень	20.10.2023	Залік з переддип-ломної практики
4.	Конструкторська частина роботи. Дослідницька частина роботи. Експериментально-практична частина. Економічне обґрунтування розробки. Розробка питань охорони праці і безпеки в надзвичайних ситуаціях	15.12.2023	
5.	Оформлення розрахунково-пояснювальної записки і графічного матеріалу	08.01.2024	
6.	Попередній захист роботи. Рецензування роботи	12.01.2024	
7.	Захист роботи	16.01.2024	

Студент _____ **Вадим ГОРБЕНКО**
(підпис) (ім'я, прізвище,)

Керівник роботи _____ **Віталій ДЖУЛГАКОВ**
(підпис) (ім'я, прізвище)

Міністерство освіти і науки України
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Кафедра 301

«ЗАТВЕРДЖУЮ»
Завідуючий кафедрою 301
к.т.н., доцент
_____Костянтин ДЕРГАЧОВ
«__» _____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на дипломне проектування
Горбенко Вадима Юрійовича

1 Тема проекту: Розробка засобів технічної реалізації та інформаційної підтримки розподілених систем термостабілізації
затверджена наказом по університету від «06» 11 2023 р. № 1968-уч.
2 Строк здачі студентом закінченої роботи «08» січня 2024 р.
3 Область застосування розробки: кондиціювання офісних приміщень.
4 Початкові дані для розроблювальної системи
4.1 Призначення і мета створення системи: забезпечення комфортних умов роботи працівників у приміщенні.
4.2 Загальні відомості номінальна температура в приміщенні (імітаторі приміщення) – +23 °С, номінальний тиск – 0.63 МПа, об'єм приміщення – 150 $м^3$, об'єм імітатора приміщення – $1.5 \cdot 10^{-3}$ $м^3$, номінальна витрата стислого повітря $1.1 \cdot 10^{-3}$ кг/с, потужність нагрівача для імітатора – 15 Вт.
5 Технічні вимоги до каналів системи управління
5.1 Питання, що підлягають розробці: формування структури, функціональної схеми системи забезпечення теплового режиму; математичне описування елементів системи; аналіз і синтез системи стабілізації теплового режиму; розробка макетного зразка системи; експериментальне відпрацювання алгоритмів стабілізації температури.
5.2 Режим роботи системи (безперервний, циклічний, одноразової дії):

безперервний.

5.3 Показники якості системи управління: точність стабілізації ±2 °С в приміщенні, час перехідного процесу ≤ 120с для приміщення, < 20с для імітатора приміщення, перерегулювання 0%, коливальність не допускається.

5.4 Вимоги до приладового складу системи: нагрівач з релейним принципом управління, охолоджувач с лінійним принципом управління, вентилятор охолоджувача, термодатчик, цифровий регулятор на базі промислового контролера.

5.5 Вимоги до взаємозамінності блоків: не задані.

6 Умови експлуатації системи

6.1 Кліматичні вимоги до експлуатації (температура середовища, у якій буде працювати система управління, її вологість, вміст хімічно активних компонентів і т.ін.)

Для реальної системи:

а) діапазон робочих температур: $-10^{o} < t < 50^{o}C$;

б) вологість до 80%.

Для макетного зразка:

а) діапазон робочих температур: $-10^{o} < t < 50^{o}C$;

б) вологість до 80%.

6.2 Механічні вимоги (вібрація, тряска, можливі перекоси, удари, нахили і т.ін.): допускається вібрація до 5g, удари і перекоси недопустимі.

6.3 Наявність перешкод (електричні наведення радіоперешкоди, магнітні впливи): електричні наведення.

6.4 Електричні параметри системи (напруга джерел живлення, потужність, стабільність,частота): живлення 220В 50Гц змінного струму, ±12В ±5В постійного струму.

7 Додаткові функції, реалізовані системою (сигналізація про несправності, реєстрація необхідної інформації, самоконтроль самої системи і т.ін.):
реєстрація режимних параметрів.

8 Обсяг виконуваних розроблювачем робіт.

8.1 Етапи проведення роботи: 1.Стан проблеми системи забеспечення теплового режиму офісного приміщення; 2. Аналіз і синтез системи забезпечення теплового режиму офісного пиміщення; 3. Конструкторська частина; 4. Технологічна частина; 5. Експериментально-практична частина 6. Економічне обгрунтування розробки; 7. Охорона праці.

8.2 Обсяг розробки по кожному етапу: стан проблеми системи забеспечення теплового режиму офісного приміщення – 10 стор.; аналіз і синтез системи

забезпечення теплового режиму офісного приміщення – 35 стор.; конструкторська частина – 12 стор.; дослідницька частина – 10 стор.; експериментально-практична частина – 15 стор.; економічне обгрунтування розробки – 10 стор.; охорона праці – 10 стор.

9 Параметри устаткування системи:

9.1 Граничні габарити: не задані.

9.2 Маса: контролер масою не більше 1 кг.

9.3 Вимоги по конструктивному виконанню та розміщення: контролер має встановлюватись на DIN-рейку.

9.4 Інші вимоги: параметри живлення для імітатора об’єкту управління: живлення обмоток двигуна вентилятора +12В постійного струму, живлення мікроконтролерного блоку керування +5В постійного струму, живлення обмотки керування релейного елементу нагрівача +12В, напруга комутації – 220В змінного струму, величина струму комутації – до 10А

10 Вимоги безпеки: електробезпека і робота зі стислим повітрям. Захист мобільної програми від несанкціонованого доступу та захист конфедіційних даних.

11 Дослідницька частина

11.1 Ефективність та Швидкість: Дослідження продуктивності додатка, виявлення можливих буттівленкоів або місць зайвого навантаження.

11.2 Безпека: Вивчення рівня безпеки додатка, виявлення можливих вразливостей та пропозиції шляхів для покращення безпеки.

11.3 Аналіз Використання: Вивчення та аналіз поведінки користувачів у додатку. Це може включати в себе аналіз часу, який користувачі витрачають в додатку, популярність окремих функцій та інші параметри використання.

11.4 Масштабованість: Вивчення можливостей масштабування додатка в разі зростання користувацької бази чи обсягу даних.

11.5 Аналіз Ринку та Тенденцій: Вивчення ринкових тенденцій у сфері мобільних додатків CMS, виявлення можливих можливостей розвитку на основі попиту.

11.6 Оцінка Інтерфейсу та Взаємодії: Дослідження сприйняття інтерфейсу користувачами, а також вивчення їхньої ефективності та зручності.

12 Експериментально-практична частина:

12.1 Тестування відображення: Перевірка правильності відображення графічних елементів та інтерфейсу на різних розширеннях та типах екранів.

12.2 Адаптивність: CMS може надавати можливість легко адаптуватися до змін потреб користувачів та ринкових умов.

12.3 Тестування безпеки: Проведення аналізу на предмет потенційних загроз безпеці та впровадження заходів щодо їх запобігання.

12.4 Тестування відповідності вимогам: Визначення та перевірка того, чи відповідає функціональність додатка вимогам та специфікаціям.

12.5 Тестування відмовостійкості: Проведення тестів для перевірки, як добре додаток витримує відмови в роботі чи аномалії, такі як втрата з'єднання.

12.6Тестування взаємодії з сервером: Перевірка взаємодії мобільного додатка з серверною частиною через REST API. Врахування різних сценаріїв взаємодії.

12.7 Функціональне тестування: Перевірка коректності роботи всіх основних функцій додатка, включаючи управління термоконтролем, авторизацію та реєстрацію користувачів, взаємодію з базою даних і інші основні операції.

13 Економічна частина

13.1 Розробити (розрахувати, одержати): розрахувати собівартість виготов-лення виробу і ціну виробу з урахуванням ПДВ, рентабельність виробництва.

13.2 Умови і вимоги: одиничний зразок.

13.3 Очікуваний результат: собівартість виготовлення виробу і рентабельність виробництва.

Перелік графічних матеріалів із зазначенням форматів – вказана кількість відповідає форматам А1: Стан проблеми (1). Аналіз і синтез системи термостабілізації (4). Конструкторська частина (1). Дослідницька частина (1). Розробка мобільного додатку (2). Економічне обгрунтування розробки (1)

Керівник проектування	Прийняв до виконання
_____Джулгаков В.Г._____	_____Горбенко В. Ю,_____
(П.І.Б.)	(П.І.Б. студента)
_____	_____
« » _____ 2023 р.	« » _____ 2023 р.

Погоджено з питань:

конструкції	дослідницької частини
_____Джулгаков В.Г._____	_____Джулгаков В.Г._____
(П.І.Б.)	(П.І.Б.)
_____	_____
« » _____ 2023 р.	« » _____ 2023 р.

економіки

_____Джулгаков В.Г._____

(П.І.Б.)

_____

« » _____ 2023 р

РЕФЕРАТ

98 сторінок тексту, 36 зображень, 5 таблиць та 22 використаних джерела.

Магістерський дипломний проект присвячений розробці мобільного додатку для управління системою термоконтролю в офісних приміщеннях. У процесі розробки використовувалися сучасні технології та інструменти, щоб забезпечити ефективність, надійність та зручність використання продукту.

Робота включає в себе детальний огляд вибраних технологій, таких як React Native для розробки мобільної платформи, Node.js для створення серверної частини, а також використання REST API для забезпечення взаємодії між додатком та сервером.

Розглянуті аспекти економічної доцільності розробки, включаючи витрати на серверні ресурси, використання готових фреймворків та оплату вартості розміщення додатка в магазинах для мобільних платформ.

У заключенні роботи наголошено на перспективах майбутнього розвитку проекту, включаючи можливості вдосконалення та розширення функціональності. Проект розроблений з урахуванням сучасних стандартів та вимог, що гарантує його актуальність та відповідність сучасним технологічним вимогам.

КЛЮЧОВІ СЛОВА: ТЕРМОСТАБІЛІЗАЦІЯ, СИСТЕМА УПРАВЛІННЯ, РОЗПОДІЛЕНА СИСТЕМА, МОБІЛЬНИЙ ДОДАТОК, ФРЕЙМВОРК

СПИСОК СКОРОЧЕНЬ

CMS - Система управління контентом
API - Інтерфейс програмування застосунків
GUI - Графічний інтерфейс користувача
SQL - Мова структурованих запитів
SPA - Односторінкова програма
MVP - Мінімально необхідний продукт
UI - Інтерфейс користувача
UX - Враження користувача
HTML - Мова розмітки гіпертексту
CSS - Каскадні таблиці стилів
JS - JavaScript
CDN - Мережа доставки контенту
SDK - Набір розробки програмного забезпечення
HTTP - Протокол передачі гіпертексту
URL - Ресурсний локатор
JSON - Об'єктні записи JavaScript
SSL - Протокол безпеки рівня транспорту
CRUD - Створення, читання, оновлення, видалення
JWT - Жетони JSON
AJAX - Асинхронний JavaScript та XML
IDE - Інтегроване середовище розробки

ЗМІСТ

ВСТУП

Однією з ключових функцій термостатів є можливість програмування та налаштування оптимальних температурних режимів протягом дня. Це дає змогу адаптувати тепловий режим до особливостей робочого графіка та потреб співробітників. Також сучасні термостати зазвичай обладнані датчиками вологості та іншими датчиками, що розширює їхні функціональні можливості. Використання технологій цифрового управління дає змогу створити розумні системи кондиціонування, де термостати можуть взаємодіяти з іншими компонентами інженерних систем будівлі. Такі системи часто включають в себе алгоритми машинного навчання для аналізу і передбачення потреби в опаленні або охолодженні, що сприяє оптимізації енергоспоживання. Сьогодні ринок термостатів надає споживачам великий вибір різноманітних моделей, пропозицій і технологічних рішень від різних виробників. Індустрія термостатів активно розвивається, що створює сприятливі умови для конкуренції між компаніями.

У сучасному суспільстві спостерігається незаперечний факт впливу інтернету на повсякденне життя. Віртуальне середовище стало невід'ємним елементом нашого буття, надаючи нам широкий доступ до інформації, можливість спілкування і зручний механізм оформлення замовлень на різні товари. З урахуванням такого широкого поширення онлайн-платформ важливо, щоб виробники термостатів приділяли особливу увагу наданню користувачам ефективних засобів ознайомлення з їхньою продукцією. Створення зручних інтернет-ресурсів, що містять інформацію про технічні характеристики, переваги та інструкції з використання термостатів, стає стратегічно важливим елементом у просуванні продукції на ринку. Отже, надання детальної та доступної інформації про термостати в онлайн-режимі стає необхідним елементом маркетингової стратегії виробників. Це сприяє не тільки підвищенню рівня обізнаності потенційних споживачів, а й створює умови для більш ефективної взаємодії між виробником і кінцевим користувачем, сприяючи встановленню довірчих відносин і підвищенню конкурентоспроможності продукції в умовах сучасного ринку.

Ефективна взаємодія виробників термостатів з кінцевими користувачами часто здійснюється через веб-ресурси. Однак, щоб забезпечити оперативне і просте оновлення актуальної інформації, включно з описами і характеристиками термостабілізаторів, без необхідності володіння глибокими

знаннями в галузі програмування, стає невід'ємним використання спеціалізованого програмного забезпечення для управління вмістом (CMS). Веб-сайти, як популярний інструмент ознайомлення користувачів з продукцією, вимагають постійного оновлення та модифікації інформації. У цьому контексті CMS є не тільки інструментом управління контентом, а й стратегічно важливим ресурсом для забезпечення оперативного внесення змін до контенту веб-ресурсу.

У майбутньому дипломному проекті планується розробка гнучкого мобільного додатка, здатного інтегруватися з різними веб-сайтами. Для забезпечення ефективної комунікації між веб-ресурсом і сервером буде використовуватися передача даних у форматі JSON. JSON (JavaScript Object Notation) обрано як формат для обміну даними через свою легкість і простоту у використанні. Цей формат надає зручний спосіб структурування і передачі інформації, що особливо актуально в контексті взаємодії між мобільним додатком, веб-сайтом і сервером. Архітектурно проєкт міститиме такі компоненти: сервер, клієнтська частина (веб-сайт) і мобільний застосунок з адміністративною панеллю. Сервер відповідатиме за опрацювання запитів від мобільного застосунку та вебсайту, а також управління JSON-файлами, що містять передану інформацію. Слід підкреслити, що мобільний застосунок виступає в ролі інтерфейсу між користувачем і сервером, забезпечуючи зручність взаємодії та інтуїтивно зрозумілий доступ до функціональності. Одним із ключових аспектів його роботи є опрацювання запитів користувача та ефективна комунікація із сервером для отримання, зміни та передавання даних.

1. СТАН ПРОБЛЕМИ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ ТА ТЕХНІЧНОЇ РЕАЛІЗАЦІЇ СИСТЕМ ТЕРМОРЕГУЛЯЦІЇ

1.1 Загальна характеристика проблеми та існуючі підходи до її вирішення

В сучасному світі, де інтернет є важливою складовою нашого повсякденного життя, необхідність швидкого та простого оновлення інформації на веб-ресурсах стає актуальною проблемою. Одним із способів забезпечення цього є наявність програмного забезпечення, яке дозволяє редагувати та оновлювати веб-сторінки без необхідності у володінні навичками програмування.

Проблема полягає в:

- Складнощах редагування інформації:

Для звичайного користувача, навички програмування можуть бути складними, що ускладнює процес редагування або створення нових сторінок.

- Затримках в оновленні:

Відсутність зручного інструменту може веде до затримок у публікації або оновленні важливої інформації.

Існуючі підходи та їхні обмеження:

- Традиційні CMS-системи:

Багато сайтів використовують системи управління контентом (CMS) для обслуговування вмісту. Проте, багато з них може вимагати технічних навичок та обмежувати гнучкість у редагуванні.

- Індивідуальна розробка:

Розробка власного програмного забезпечення для редагування веб-сторінок вимагає значних ресурсів та фахових знань.

Наш підхід до вирішення проблеми:

В рамках даного проекту передбачено розробку гнучкого мобільного додатку, який дозволить легко і швидко підключати будь-який веб-сайт. Взаємодія між сайтом і сервером буде забезпечена за допомогою json-файлів, що є простим та ефективним форматом. В комплексі буде реалізована серверна, клієнтська частини (веб-сайт), та адміністративна панель додатку, що враховує всі аспекти управління вмістом термостатів.

1.2 Аналіз технічного завдання

Термостабілізатор призначен для забезпечення комфортних умов у офісних приміщеннях. Вимоги включають номінальні параметри, такі як температура, тиск, об'єм, та інші, а також технічні характеристики компонентів, таких як нагрівач, охолоджувач, вентилятор та інші. Розглядаються питання формування структури, функціональної схеми, математичного описування елементів, аналізу та синтезу системи стабілізації. Режим роботи - безперервний, з точністю стабілізації, часом перехідного процесу та іншими параметрами. Визначені кліматичні, механічні та електричні вимоги, включаючи діапазон температур, вологість, вібрацію, електричні параметри тощо. Система повинна вміти реєструвати режимні параметри. Обсяг робіт визначається етапами: від аналізу стану проблеми до охорони праці. Зазначено граничні габарити, масу та вимоги до конструкції. Контролер повинен встановлюватись на DIN-рейку. Технічне завдання визначає всі ключові аспекти розробки системи термостабілізації для офісних приміщень. Передбачає широкий спектр робіт, включаючи аналіз, конструкторську та експериментальну частини, а також економічне обгрунтування.

Проект орієнтований на створення системи адміністрування контенту для веб-сайту виробництва термостатів з метою забезпечення зручності управління та актуалізації інформації. Основною метою системи є надання можливості адміністраторам веб-ресурсу ефективно керувати структурою, розміщенням контенту, а також визначенням та зміною технічних характеристик, опису, зображень. Система повинна враховувати номінальні параметри веб-сайту, включаючи його загальну структуру, спосіб розміщення інформації та технічні характеристики окремих компонентів, таких як блоки тексту та графічні елементи. Врахування цих факторів дозволить адміністраторам з легкістю модифікувати та оновлювати вміст сайту, відповідаючи змінам в бізнес-процесах та вимогах користувачів. Основним завданням системи є надання засобів для легкого та ефективного додавання, редагування та видалення інформації на веб-сайті. Адміністраторам має бути надана можливість керувати розташуванням інформації на сторінках, визначати її структуру та оптимально взаємодіяти з технічними характеристиками різних елементів сайту. Також, передбачається можливість додавання нових батьківських категорій, категорій, підкатегорій та позицій товару, а саме термостатів. Система має бути розроблена з урахуванням високих стандартів безпеки та надійності. Вона повинна гарантувати доступ тільки авторизованим користувачам. Безпека є пріоритетним завданням у розробці системи.

Необхідно впроваджувати ефективні заходи для захисту від несанкціонованого доступу, шифрування конфіденційних даних та впровадження системи контролю доступу. Технічне завдання для системи адміністрування контенту на веб-сайті визначає ключові можливості та вимоги, спрямовані на полегшення процесу управління вмістом. Зазначені функції та характеристики становлять основу для подальшого проектування та реалізації ефективної системи адміністрування.

1.3 Огляд науково-технічної літератури і патентів

У статті [1] було розглянуто докладні інструкції з встановлення React Native на різні платформи, включаючи macOS, Windows і Linux. Також вона включає інформацію про налаштування React Native для розробки мобільних додатків для iOS і Android. Документація охоплює основи React Native, включаючи такі теми, як компоненти, життєвий цикл компонентів, стан, навігація та взаємодія з користувачем. Надає приклади та керівництва з розробки мобільних додатків з використанням React Native. Ці приклади охоплюють широкий спектр завдань, включаючи створення простих додатків, таких як калькулятор, і складних додатків, таких як додатки для електронної комерції.

У статті [2] було оглянуто документацію яка містить докладні інструкції з встановлення Node.js на різні платформи, включаючи macOS, Windows і Linux. Також вона включає інформацію про налаштування Node.js для розробки веб-додатків. Приклади та керівництва з розробки веб-додатків з використанням Node.js. Ці приклади охоплюють широкий спектр завдань, включаючи створення простих додатків, таких як блоги, і складних додатків, таких як корпоративні веб-сайти.

У статті [3] було оглянуто важливий ресурс, який виявився надзвичайно корисним під час розробки мобільного додатка. Цей ресурс надавав обширну інформацію про роботу з REST API, що виявилось критично важливим для взаємодії додатка з сервером. Автор докладно розглянув різні аспекти роботи з REST API, включаючи структуру запитів та відповідей, обробку помилок, автентифікацію та авторизацію. Це надало мені чітке розуміння процесу взаємодії між клієнтом та сервером, що було ключовим у вдалий розвиток мого додатка. Покрокові поради та приклади з реального життя, які надані на цьому ресурсі, значно полегшили для мене впровадження ефективних рішень при

взаємодії з REST API. Я також вивчив найкращі практики та стратегії для оптимізації швидкості та безпеки обміну даними між клієнтом та сервером.

У статті [5] було розглянуто важливий ресурс, який виявився дуже цінним під час моєї роботи з розробки серверної частини додатків на платформі Node.js. Цей ресурс забезпечив мене загальним уявленням про використання Node.js для створення ефективних та масштабованих серверів. Автор надав чіткі та докладні пояснення щодо основ Node.js, а також навігацію по ключовим аспектам, таким як обробка запитів, робота з базами даних, створення API та інші. Зокрема, розділ про Express framework виявився дуже корисним при розробці веб-серверів. Я також отримав інформацію про найкращі практики забезпечення безпеки та оптимізації Node.js додатків. Засвоєні знання допомогли мені створювати ефективний та безпечний серверний шар для своїх проектів.

У статті [6] було оглянуто процеси розробки веб-додатків з використанням Express.js. Цей ресурс надавав докладну інформацію та настанови, необхідні для розуміння та впровадження фреймворку. Автор статті детально розглянув основні концепції Express.js, включаючи структуру додатка, обробку маршрутів, middleware, та обробку запитів та відповідей. Це надало мені чітке уявлення про те, як створювати ефективні та масштабовані веб-сервери з використанням цього фреймворку. Окремі розділи про шаблонізатори та роботу з базами даних також були особливо корисними для мене під час розробки серверної частини додатка. Я отримав чітке розуміння того, як використовувати Express.js для взаємодії з різноманітними компонентами мого веб-додатка.

У статті [8] було оглянуто важливу інформацію, яка допомогла мені у процесі розробки мобільного додатка на React. Автор розглянув застосування useMemo для ефективного управління кешуванням результатів обчислень, що стало важливим для оптимізації продуктивності. Також, було висвітлено використання RSC preload та flushSync для покращення завантаження компонентів та забезпечення плавної взаємодії з користувачем. Я також дізнався про Shadcn UI, MDX та Storybook, що виявилося корисним у створенні більш ефективного та користувацьки орієнтованого інтерфейсу додатка. Ця інформація виявилася неоціненною для розробки та покращення мого мобільного додатка на платформі React.

У статті [11] було оглянуто GitHub репозиторій від Facebook надавав обширну інформацію, яка була невід'ємною для успішного впровадження та

розвитку моїх мобільних проектів. Автори надали докладну документацію, яка охоплює всі ключові аспекти React Native - від налаштування проекту до роботи з компонентами та взаємодії з різними пристроями. Зокрема, інформація про роботу з навігацією, станом, анімацією та використанням сторонніх бібліотек була особливо корисною для мене під час розробки мобільних додатків. Окремі розділи, присвячені відлагодженню, профілюванню та оптимізації продуктивності, допомогли мені створювати ефективні та швидкодіючі додатки для різних мобільних платформ.

У статті [13] було розглянуто репозиторій на GitHub містить детальну документацію та опис функціоналу Hoppscotch - інструменту для тестування API та візуалізації HTTP-запитів. Автори ретельно розглянули основні можливості Hoppscotch, роз'яснили процес відправки та отримання HTTP-запитів, а також подали інформацію про різноманітні функції, такі як робота з параметрами, автентифікація, відладка та багато іншого. Ця інформація виявилася надзвичайно корисною для мене, допомагаючи в зручному та ефективному тестуванні API в моїх проектах. Особливо важливим для мене було вивчення можливостей візуального середовища Hoppscotch та розуміння, як використовувати цей інструмент для ефективного спілкування з різними API. Інструкції та приклади з реального життя, надані в цьому ресурсі, дозволили мені миттєво застосовувати отримані знання у своїй роботі.

У статті [17] було оглянуто завдяки йому я вивчати нові методи та підходи до розв'язання проблем. Запитання та відповіді, представлені у цьому розділі, охоплюють широкий спектр тем, від основ до складних аспектів роботи з Node.js. Окрім того, розділ "Newest" дозволяє мені слідкувати за найновішими питаннями та відповідями, що допомагає тримати руку на пульсі технологічних тенденцій та розвитку Node.js. Різноманітність питань і відповідей також дозволяє мені поглиблювати свої знання та ділитися власним досвідом з іншими учасниками спільноти.

1.4 Постановка задач проектування

В рамках розгляду можливих CMS для потреб підприємств, що виробляють термостати, необхідно враховувати специфічні вимоги та характеристики даної галузі. Одним із ключових аспектів є забезпечення зручного та ефективного управління контентом, що включає в себе не лише веб-сайт, але й можливість керування продуктовою інформацією, характеристиками та іншими аспектами.

На сучасному етапі існує кілька CMS, які можуть відповідати потребам виробників термостатів. Один з прикладів – WordPress. WordPress є популярною та розширюваною платформою, яка надає широкі можливості для розміщення та управління контентом. Він дозволяє легко створювати сторінки, редагувати текст, додавати зображення, та керувати категоріями продуктів.

Іншою альтернативою є Drupal, яка володіє потужними можливостями керування контентом та широким спектром модулів для розширення функціональності. Для виробників термостатів, які мають потребу в розширених можливостях керування та інтеграції, Drupal може виявитися відмінним вибором.

Також слід взяти до уваги Magento, яка спеціалізується на електронній комерції. Вона надає потужні засоби для створення інтернет-магазинів та керування продуктами.

Однак, вибір CMS повинен ґрунтуватися на конкретних вимогах та стратегії підприємства. Необхідно враховувати масштаби бізнесу, потреби в управлінні контентом, а також можливості майбутньої інтеграції з іншими інформаційними системами.

Не дивлячись на широкий спектр можливостей, які надають такі популярні системи управління контентом (CMS), як WordPress, Drupal та Magento, вони також мають свої недоліки, які можуть бути критичними для певних типів бізнесу, зокрема для предприємств, що виробляють термостати.

Складність налаштування та обслуговування: Багато CMS вимагають технічних навичок для встановлення та налаштування. Це може стати проблемою для користувачів без глибоких знань у сфері програмування, зокрема для предприємств, які спеціалізуються на виробництві термостатів, де фахівці можуть бути спрямовані на інші компетенції.

Великі витрати на розробку та підтримку: Використання розширених CMS, таких як Drupal чи Magento, може вимагати значних фінансових витрат на розробку та підтримку. Підтримка та оновлення, зокрема у випадку Magento, можуть вимагати великих ресурсів та фінансів.

Швидкість завантаження та продуктивність: Деякі CMS, особливо при наявності великої кількості розширень чи модулів, можуть впливати на швидкість завантаження веб-сайту. Для предприємства, яке пропонує термостати, важливо мати ефективний та швидкий веб-сайт для задоволення потреб клієнтів. CMS, як будь-яке програмне забезпечення, може бути піддане

загрозам безпеки. Важливо постійно вдосконалювати та оновлювати CMS для захисту від потенційних кіберзагроз.

Обмеження у роботі з великим обсягом товарів: Деякі CMS можуть виявитися менш ефективними при роботі з великою кількістю товарів, характерною для виробників термостатів з різноманітним асортиментом.

У сучасному інформаційному середовищі, де швидкість та гнучкість є ключовими факторами успіху, виробники термостатів виявляють рост інтересу до застосування передових технологій у системах управління контентом (CMS). Потреба у забезпеченні ефективного та прогресивного інструмента для управління вмістом виявляється через критику відносно використання застарілих технологій у наявних CMS. Зокрема, визначається ряд недоліків та відмінностей, які покладають під сумнів їхню актуальність та здатність відповідати потребам ринку.

Першим технічним аспектом, що привертає увагу, є використання застарілої технології, зокрема серверної частини, реалізованої на мові програмування PHP. За сучасними стандартами, такий підхід не є оптимальним та ефективним. Зокрема, розширення можливостей, підтримка масштабування та інтеграція з іншими рішеннями можуть бути значно обмеженими через обрану технологічну платформу.

Другим важливим аспектом є відсутність адаптації до сучасних концепцій мікросервісної архітектури. Сучасні розробники віддають перевагу розділенню функціональності на окремі мікросервіси, що спрощує розгортання, супровід та розвиток систем. Монолітні програмні продукти, до яких можна віднести існуючі CMS, ускладнюють процеси розробки та утримання.

Виправленням цих недоліків є застосування мікросервісного підходу у дипломному проекті. Застосування окремих компонентів для серверної частини, CMS та веб-сайту надає більше гнучкості та легкість у розгортанні. Такий підхід дозволяє розміщувати різні компоненти системи на різних серверах та легко взаємодіяти між ними. Важливим аспектом є також можливість легкого підключення додаткового програмного забезпечення до сервера, що надає виробникам термостатів можливість впроваджувати нові функціональності та автоматизувати бізнес-процеси.

Отже, перехід від застарілих технологій та монолітних систем до сучасного мікросервісного підходу управління контентом дозволяє виробникам термостатів відповідати викликам сучасного ринку, забезпечуючи ефективність та конкурентоспроможність їхніх продуктів.

Так же сипользования разных языков программирования усложняют процесс разработки. В дипломном проекте используется один язык программирования Java script. Так как он поддерживается и на мобильных устройствах, и на сервере и на компьютере. У процесі розробки програмного забезпечення для дипломного проекту особлива увага приділяється вибору язика програмування, який відповідає сучасним стандартам ефективності та універсальності. Важливими аспектами при цьому виступають спрощення самого процесу розробки, а також забезпечення оптимальної сумісності програмного продукту із різними платформами та пристроями.

Однією з труднощів у розробці програмного забезпечення є необхідність використання різних мов програмування для різних компонентів системи. Це може призводити до ускладнення взаємодії між різними елементами програми, а також вимагати великих витрат часу та зусиль для підтримки та розвитку.

У дипломному проекті вирішено цю проблему шляхом використання одного універсального мови програмування – JavaScript. Вибір JavaScript обґрунтовується його широкою популярністю, а також універсальністю застосування. Цей мова програмування підтримується на різних платформах, включаючи мобільні пристрої, сервери та персональні комп'ютери.

Існуючи на багатьох різноманітних пристроях, JavaScript дозволяє підтримувати єдиний кодову базу для розробки як клієнтської, так і серверної частин системи. Це значно спрощує взаємодію між компонентами та забезпечує більшу стабільність системи в цілому.

Одним із важливих переваг використання JavaScript є його підтримка на різних мобільних платформах, що дозволяє розширити аудиторію користувачів та забезпечити оптимальний користувацький досвід на різних пристроях. Вибір JavaScript є стратегічним кроком у напрямку спрощення розробки, оптимізації підтримки та покращення універсальності програмного продукту.

Однією з ключових інновацій є використання сучасних фреймворків – React та Node.js. React, який використовується для клієнтської частини, надає можливість створювати динамічні та ефективні інтерфейси, прискорюючи розробку завдяки його компонентній структурі. Node.js, використовуваний на сервері, гарантує високу швидкодію та масштабованість веб-додатку.

Окремою темою у проекті є питання безпеки та обмеження доступу до адміністративної панелі. Для цього використовуються JWT (JSON Web Token) токени – передовий механізм аутентифікації та авторизації користувачів. Вони генеруються на сервері, підписуються, та містять інформацію про користувача

та його права. Використання JWT токенів дозволяє ефективно захищати дані, передавати їх із запитів клієнтів та надавати контроль доступу до різних функціональних можливостей.

Застосування сучасних фреймворків та захисних механізмів, таких як JWT токени, у дипломному проекті визначається не лише технічними перевагами, але й стратегічною важливістю для забезпечення конфіденційності та ефективності веб-додатку. Використання таких інструментів дозволяє ефективно справлятися з вимогами швидкодії та безпеки, що є критичними для сучасних інформаційних систем.

У процесі створення системи управління контентом (CMS), яка призначена для використання в офісних приміщеннях, необхідно ретельно враховувати специфіку продукта і забезпечувати користувачам, зокрема потенційним покупцям, інформацію, що є необхідною та доступною.

В першу чергу, важливо зазначити, що офісні термостати мають свої унікальні особливості і вимоги. Спрямовані на забезпечення комфортних умов в робочому середовищі, вони повинні володіти певними характеристиками, які потрібно належним чином розкрити в інформаційному вмісті CMS.

Акцентуючи увагу на підходах до представлення інформації, слід розглянути розгорнуті та зрозумілі описи характеристик термостатів. Надання детальної інформації щодо робочих параметрів, енергоефективності, можливостей програмування та інших технічних аспектів стане ключовим елементом, орієнтованим на забезпечення інформованості клієнта. Важливо враховувати актуальність інформації, оскільки характеристики термостатів можуть підлягати змінам внаслідок вдосконалення технологій. Забезпечення можливості швидкого та легкого оновлення контенту є ключовим завданням CMS для забезпечення користувачів актуальною інформацією.

Специфічність продукту також передбачає подачу інформації про преимущества використання даного типу термостатів у офісних приміщеннях. Аналіз функціональних переваг, таких як енергоефективність, можливості програмування режимів, інтеграція з іншими системами, допоможе покупцеві розуміти, чому обрати саме цей продукт.

2. АНАЛІЗ І СИНТЕЗ СИСТЕМИ ТЕРМОСТАБІЛІЗАЦІЇ

2.1 Вибір і обгрунтування функціональної схеми системи

Існує два принципи роботи термостата. У першому випадку, в термостаті створюється й утримується певна температура використовуваного ним теплоносія (масло, газ, вода, спирт тощо). Параметр, який необхідно контролювати за допомогою термостата, контактує з робочою речовиною самого термостата і таким чином регулюється. Причому залежно від діапазону можливих температур вибирається тип робочої речовини.

У другому випадку контрольований параметр підтримується в адіабатичних умовах за постійної температури. Відведення і підведення тепла при цьому здійснюється за рахунок використання спеціального теплового ключа. У цій роботі буде використовуватися перший принцип. Система регулювання температури зображено на рис. 2.1.

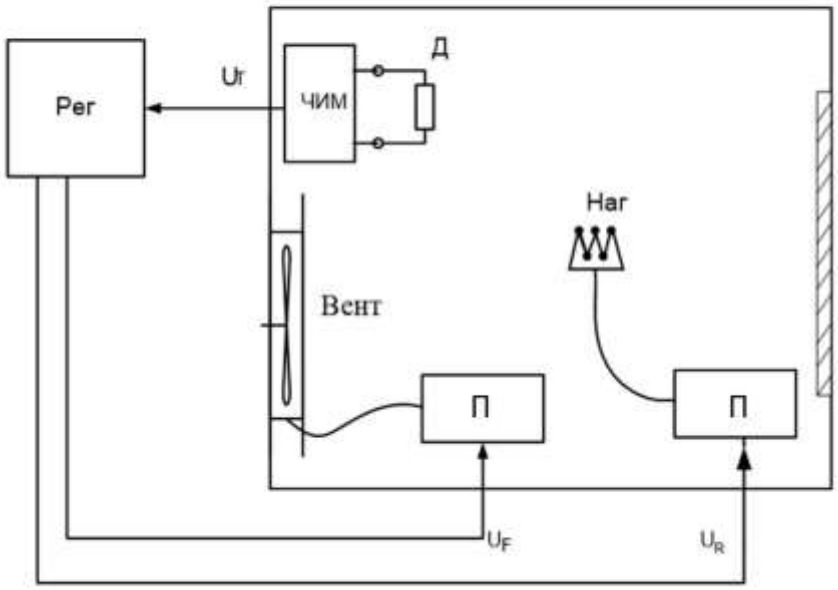


Рисунок 2.1 – Система регулювання температури

На рис. 2.1 введено такі умовні позначення:

Д – датчик;

У – підсилювач;

Рег – регулятор;

Наг – нагрівач;

Вент – вентилятор.

На рис. 2.2 представлена функціональна схема системи керування термостатом.

На рис. 2.2 введены следующие обозначения:

ЗП - задавальний пристрій;

Рег - регулятор;

НЕ - нагрівальний елемент;

ОЕ - охолоджувальний елемент;

БТ - блок теплопередачі;

Д - датчик температури;

$U_{зад}(t)$ – напруга, пропорційна заданій температурі;

e(t) – різниця між заданою і поточною температурами;

$U_{рег1}(t)$ – керувальний вплив нагрівального елемента;

$U_{рег2}(t)$ – керувальний вплив охолоджувального елемента;

$Q_{нэ}(t)$ – кількість теплоти, що надходить із нагрівального елемента;

$Q_{оэ}(t)$ – кількість теплоти, що надходить з охолоджувального елемента;

$U_{д}(t)$ – вихідна напруга датчика, пропорційна поточній температурі;

t^oC – поточна температура.

Згідно з технічним завданням і обраним технічним рішенням розроблено функціональну схему системи, що складається з одного контуру керування.

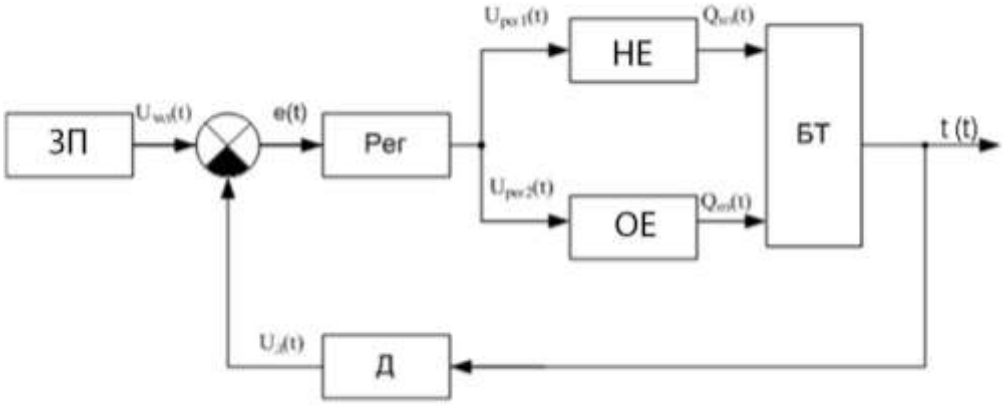


Рисунок 2.2 – Функціональна схема системи управління термостатом

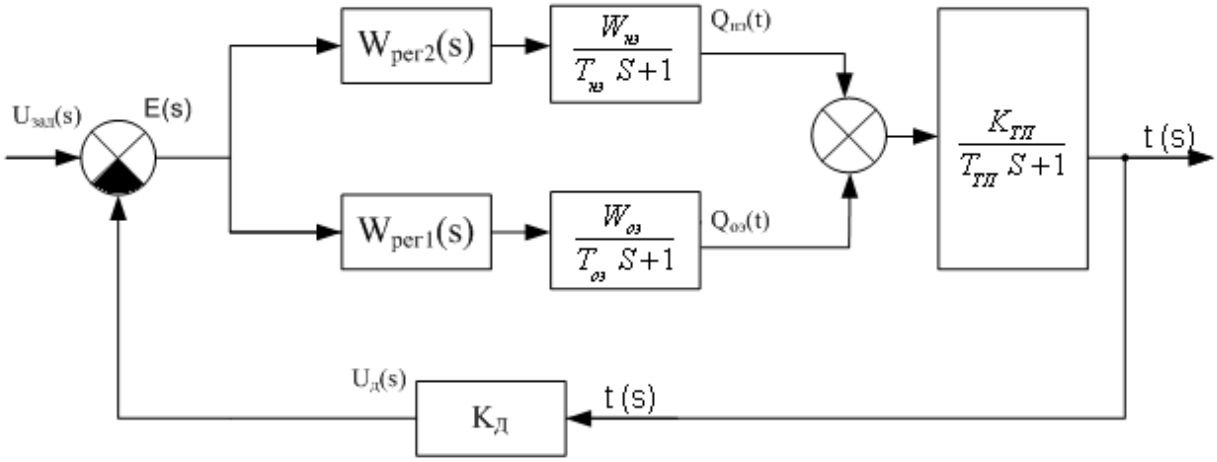


Рисунок 2.3 – Структурна схема керування термостатом

2.2 Розроблення моделей об'єкта автоматичного керування та елементів системи

Математична модель - це модель, побудована за допомогою математичної мови. Нелінійна математична модель ОАС - це математична модель, що враховує нелінійності реального пристрою (зона нечутливості, зона насичення, загальна нелінійність характеристики тощо).

Схема електродвигуна постійного струму, швидкість обертання якого (рад/с) регулюється зміною напруги в ланцюзі якоря $U_в$, показано на рис. 2.4

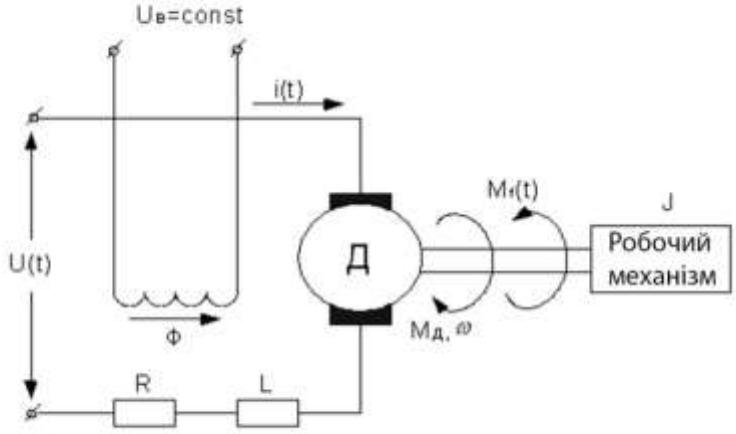


Рисунок 2.4 – Регульований електропривод постійного струму

За допомогою цієї схеми будуємо нелінійну математичну модель електродвигуна. Для цього скористаємося умовою рівноваги механічних моментів на валу двигуна видно із виразу (2.1)

$$M_Д = J\frac{d\omega}{dt} + M_f \tag{2.1}$$

І умовою рівноваги електрорушійних сил і падінь напруг у ланцюзі якоря видно із виразу (2.2)

$$U = L\frac{di}{dt} + Ri + e \tag{2.2}$$

де $M_Д = c_\mu \Phi i$ та $e = c_e \Phi \omega$ визначаються через параметри електродвигуна. В результаті розв'язання системи рівнянь отримуємо нелінійну модель електродвигуна видно із виразу (2.3)

$$(T_{ЭМ}\, T_Э S^2 + T_{ЭМ} s + 1) \cdot\ \omega(s) = k_1 u(s) - k_2(1 + T_Э s) M_f(s) \tag{2.3}$$

де електромеханічна постійна часу видно із виразу (2.4);

$$T_{ЭМ} = JRc_e c_\mu \Phi^2 \tag{2.4}$$

Електромагнітна постійна видно із виразу (2.5);

$$T_Э = \frac{L}{R} \tag{2.5}$$

Постійні коефіцієнти видно із виразу (2.6) та (2.7).

$$k_1 = \frac{1}{c_e \Phi} \tag{2.7}$$

$$k_2 = \frac{R}{c_e c_\mu \Phi^2} \tag{2.8}$$

Лінеаризована математична модель ОАС - це математична модель, представлена у вигляді передавальних функцій елементів та устрою в цілому.

Аналітично отримання передавальних функцій полягає у знаходженні коефіцієнтів передавальних функцій за емпіричними формулами, які використовують паспортні дані.

За заданою потужністю вибираємо потрібний електродвигун. Технічні характеристики електродвигуна П11М представлені у табл. 2.1.

Таблиця 2.1 - Характеристики електродвигуна постійного струму П11М

Параметр, одиниці виміру	Значення
Номінальна потужність, кВт	0,25
Номінальна напруга, В	50
Номінальна частота обертання, об/хв	1500
Номінальний струм на якорі, А	6,85
Момент інерції якоря, кг·см^2	9,15
Опір обмотки якоря, Ом	1,4
Індуктивність якоря L, Гн	0.025

Електродвигун управляє охолоджувальним елементом системи, який є вентилятором.

Для визначення поточної температури термостата вибрано датчик температури. Датчик температури призначений для вимірювання та контролю температури повітря.

У табл. 2.2 наводяться характеристики та параметри датчика температури.

Таблиця 2.2 – Характеристики та параметри датчика температури

Характеристики	Параметри
Діапазон вимірюваних температур	От -40°С до +110°С
Вихідна напруга датчика	От 0 В до 5 В
Напруга живлення	постійне 12 В ± 10%.
Споживаний струм	не більше 12 мА.
Точність вимірювання температури	± 1,2 С

Знайдемо коефіцієнти передачі електродвигуна:

- Коефіцієнт передачі електродвигуна за керуючим впливом видно із виразу (2.9), (2.10) та (2.11)

$$k_{\text{Д}} = \frac{\omega_{\text{ДН}}}{U_H - I_{\text{Я}} \cdot r_{\text{Я}}} \tag{2.9}$$

$$k_{\text{Д}} = \frac{157}{50 - 6.85 \cdot 1.4} \approx 3.9 \tag{2.10}$$

$$\left[K_{\text{Д}}\right] = \frac{1/c}{B - A \cdot O_M} = 1/B \cdot c \tag{2.11}$$

- Коефіцієнт передачі електродвигуна по обурюючому впливу видно із виразу (2.12) та (2.13)

$$k_{ДВ} = k_{Д}^2 \cdot r_{Я} \approx 21.3 \tag{2.12}$$

$$[k_{ДВ}] = \frac{Ом}{B^2 \cdot c^2} = \frac{1}{Н \cdot м \cdot c} \tag{2.13}$$

Постійні часу дорівнюють впливу видно із виразу (2.14) та (2.14)

$$T\text{ээ} = J \cdot k_{\text{Д}^2} \cdot r_{\text{Я}} = 9150 \cdot 15.21 \cdot 1.4 \cdot 10^{-6} \approx 0.2\text{с} \tag{2.13}$$

$$T_{\text{э}} = \frac{L}{r_{\text{Я}}} = \frac{0.025}{1.4} = 0.018\text{с} \tag{2.14}$$

Отримаємо наступну лінеаризовану математичну модель електродвигуна П11М видно із виразу (2.15) та (2.16).

$$\omega(s) = W_U(s) \cdot U(s) + W_f(s) \cdot M(s) = \frac{k_Д}{T_{ЭМ}T_Э s^2 + T_{ЭМ}s + 1} \cdot U(s) + \frac{k_{ДВ}(T_Э s + 1)}{T_{ЭМ}T_Э s^2 + T_{ЭМ}s + 1} \cdot M(s) \quad (2.15)$$

$$\omega(s) = \frac{3.9}{0.0036s^2 + 0.2s + 1} \cdot U(s) + \frac{0.38s + 21.3}{0.0036s^2 + 0.2s + 1} M(s) \quad (2.16)$$

Для спрощення порядку системи візьмемо передатну функцію двигуна видно із виразу (2.17).

$$W_{ДВ}(s) = \frac{\omega_{ЭД}(s)}{U_{УМ}(s)} = \frac{3.9}{0.2s + 1} \quad (2.17)$$

Запишемо передавальні функції основних елементів системи керування термостатом.

Передатна функція охолоджувального елемента видно із виразу (2.18).

$$W_{ОЭ}(s) = \frac{Q(s)}{U_{рег}(s)} = \frac{-0.5}{0.6s + 1} \quad (2.18)$$

Передатна функція нагрівального елемента видно із виразу (2.19).

$$W_{НЭ}(s) = \frac{Q(s)}{U_{НЭ}(s)} = \frac{5}{0.9s + 1} \quad (2.19)$$

Передатна функція термодатчика видно із виразу (2.20).

$$15)\ W_{НЭ}(s) = \frac{t(s)}{U_Д(s)} = 2.6 \quad (2.20)$$

Одним з найважливіших елементів даної системи управління є блок, який відповідає за теплообмін між робочим тілом термостатом та навколишнім середовищем. У реальних умовах тепло передається комбінованим способом. Наприклад, при теплообміні між твердою стінкою та газовим середовищем тепло передається одночасно конвекцією, теплопровідністю та випромінюванням (тепловіддача). Ще більш складним є процес передачі тепла від більш нагрітого газу через поверхню, що розділяє їх (теплопередача). Математична модель процесу теплопередачі представлена нижче.

Розрахунок теплообмінної апаратури включає:

1) визначення теплового потоку шляхом складання та вирішення теплових балансів;

2) визначення поверхні теплообміну із основного рівняння теплопередачі.

Кількість теплоти (Дж/с), яку віддає гарячий теплоносій видно із виразу (2.21)

$$Q_1 = G_1 c_1 (t_{1n} - t_{1k}) \quad (2.21)$$

Аналогічно кількість теплоти (Дж/с), яку отримає холодний теплоносій видно із виразу (2.22)

$$Q_2 = G_2 c_2 (t_{2k} - t_{1h}) \quad (2.22)$$

де G - масова витрата гарячого (холодного) теплоносія, кг/с;

с – питома теплоємність теплоносія, Дж/(кгК);

tн – початкова температура і tк – кінцева температура теплоносія.

Через втрати теплоти в довкілля через зовнішні стінки апарату холодний теплоносій отримає всю теплоту, віддану гарячим теплоносієм видно із виразу (2.22).

$$Q_1 = Q_2 + Q_{nom} \quad (2.22)$$

Рівняння теплового балансу видно із виразу (2.23)

$$G_1 c_1 (t_{1h} - t_{1k}) = G_2 c_2 (t_{2k} - t_{2h}) + Q_{nom} \quad (2.23)$$

Рівняння справедливе, якщо теплоносія не змінює агрегатного стану. Якщо в якості гарячого теплоносія використовують насичену водяну пару, то кількість теплоти, що виділяється при конденсації пари видно із виразу (2.24)

$$Q = Dr \quad (2.24)$$

де D – масова витрата пари, кг/с;

r – прихована теплота пароутворення, Дж/кг пари видно із виразу (2.25)

$$Dr = G_2 c_2 (t_{2k} - t_{1h}) + Q_{nom} \quad (2.25)$$

Якщо в апараті охолоджується конденсат, необхідно врахувати теплоту, що виділяється конденсатом видно із виразу (2.26)

$$Dr + Dc_{kon}(t_n - t_{kon}) = G_2 c_2 (t_{2k} - t_{1h}) + Q_{nom} \quad (2.26)$$

де $\mathbf{c}_{кон}$ – питома теплоємність конденсату, Дж/(кгК); $t_п$ – температура пари, що гріє, рівна температурі конденсації, ^{0}С;

$t_{кон}$ – температура конденсату, що залишає апарат, ^{0}С.

У сучасних теплових апаратах теплові втрати завдяки тепловій ізоляції не перевищують 3...5% від кількості теплоти, що виділяється гарячим теплоносієм, і в наближених розрахунках можуть не враховуватися.

Основне рівняння тепловіддачі видно із виразу (2.27)

$$Q = aF(t_{cm} - t_ж) = aF\Delta t\tau \quad (2.27)$$

Виходячи з рівняння тепловіддачі, зробимо розрахунок передавальної функції блоку теплопередачі.

Передатна функція блоку теплопередачі видно із виразу (2.28)

$$W(s) = \frac{Q(s)}{t(s)} = \frac{0.95}{40s+1} \quad (2.28)$$

2.3 Аналіз властивостей об'єкта автоматичного керування

Розглянемо контур регулювання температури, а саме нагрівання та охолодження, схеми моделювання яких представлені на рис. 2.5 та 2.6. відповідно.

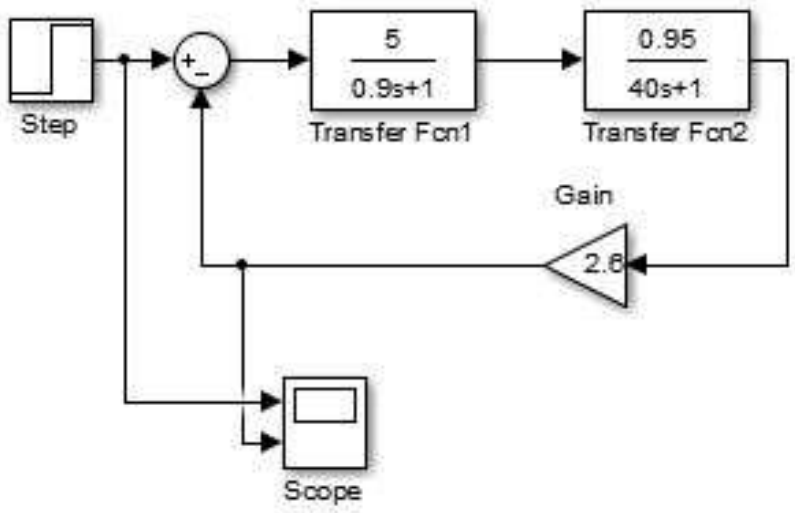


Рисунок 2.5 – Схема моделювання контуру регулювання температури нагрівання.

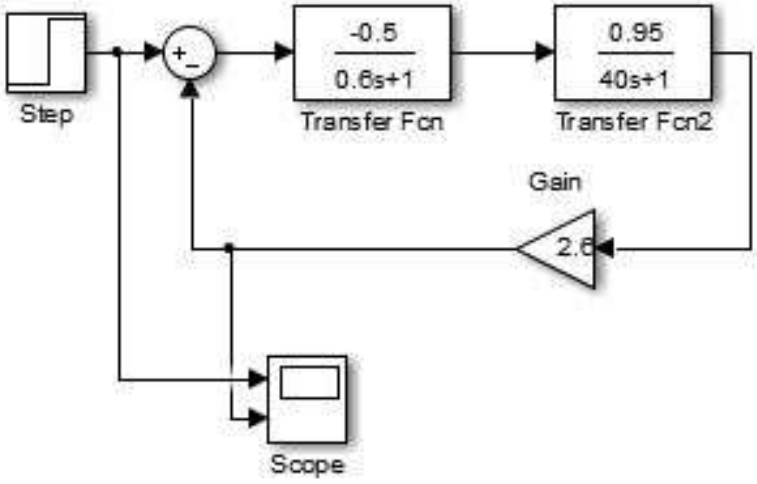


Рисунок 2.6 – Схема моделювання контуру регулювання температури охолодження

Представимо передатну функцію замкнутої системи по впливу, що задає видно із виразу (2.29)

$$W_{PAC\text{П}} = \frac{m(s)}{U_{\text{ЗМ}}(S)} = \frac{48.165}{7.2s^2+41.1s+1} \quad (2.29)$$

Побудуємо в середовищі Matlab перехідну характеристику за впливом, що задає, зобразимо її на рис. 2.7, де а - графік впливу,

б – перехідна характеристика контуру керування температурою

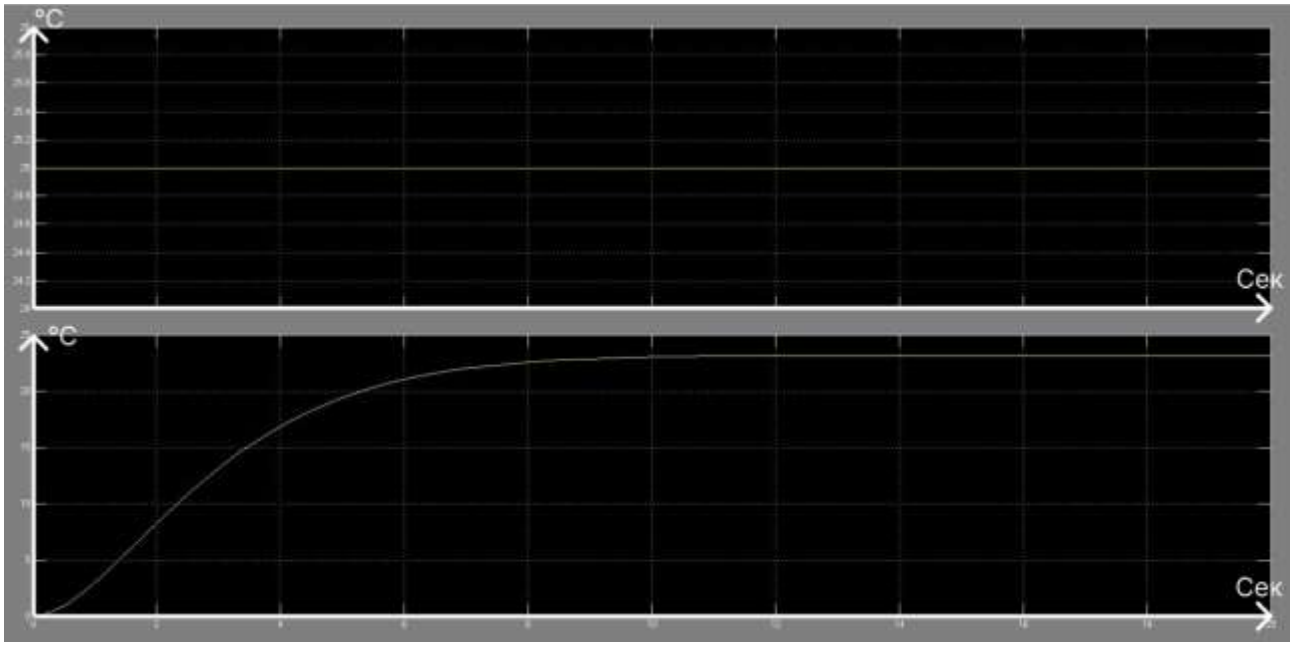


Рисунок 2.7 – Перехідна характеристика по задаючому впливу температури нагріву

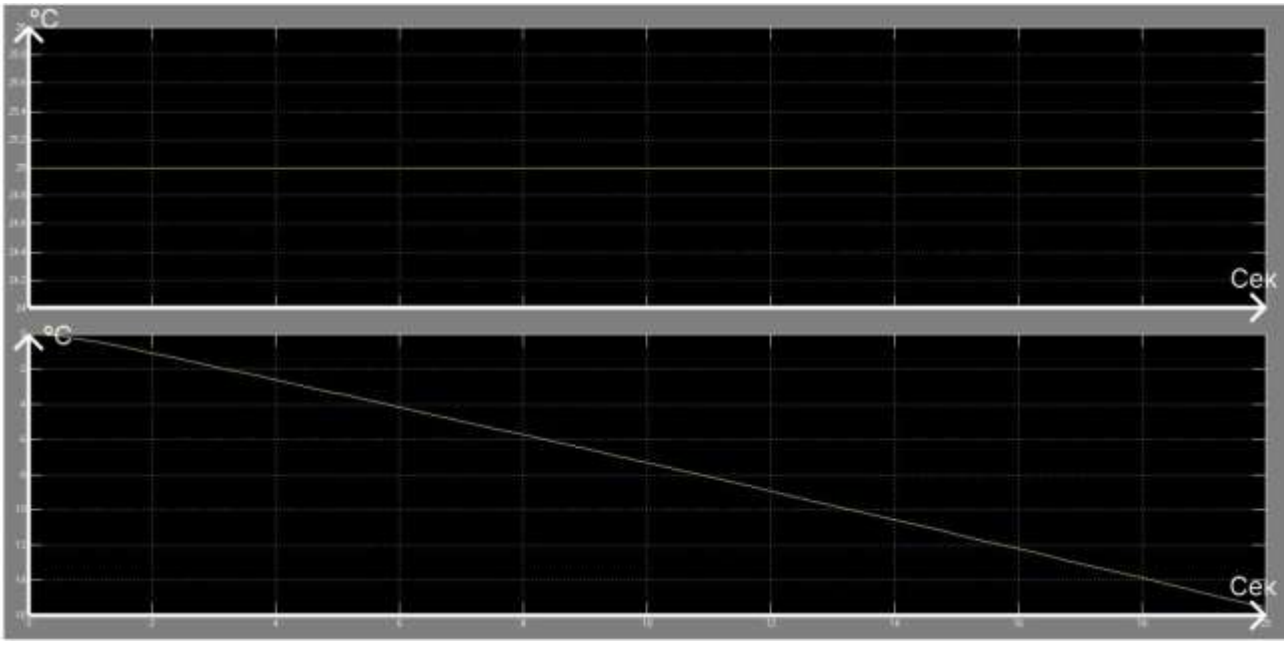


Рисунок 2.8 – Перехідна характеристика по задаючому впливу температури з охолодження

На рис. 2.9 та рис. 2.10 зображено частотні характеристики нескоректованої розімкнутої системи управління від впливу, що задає, по нагріванню і охолодженню відповідно.

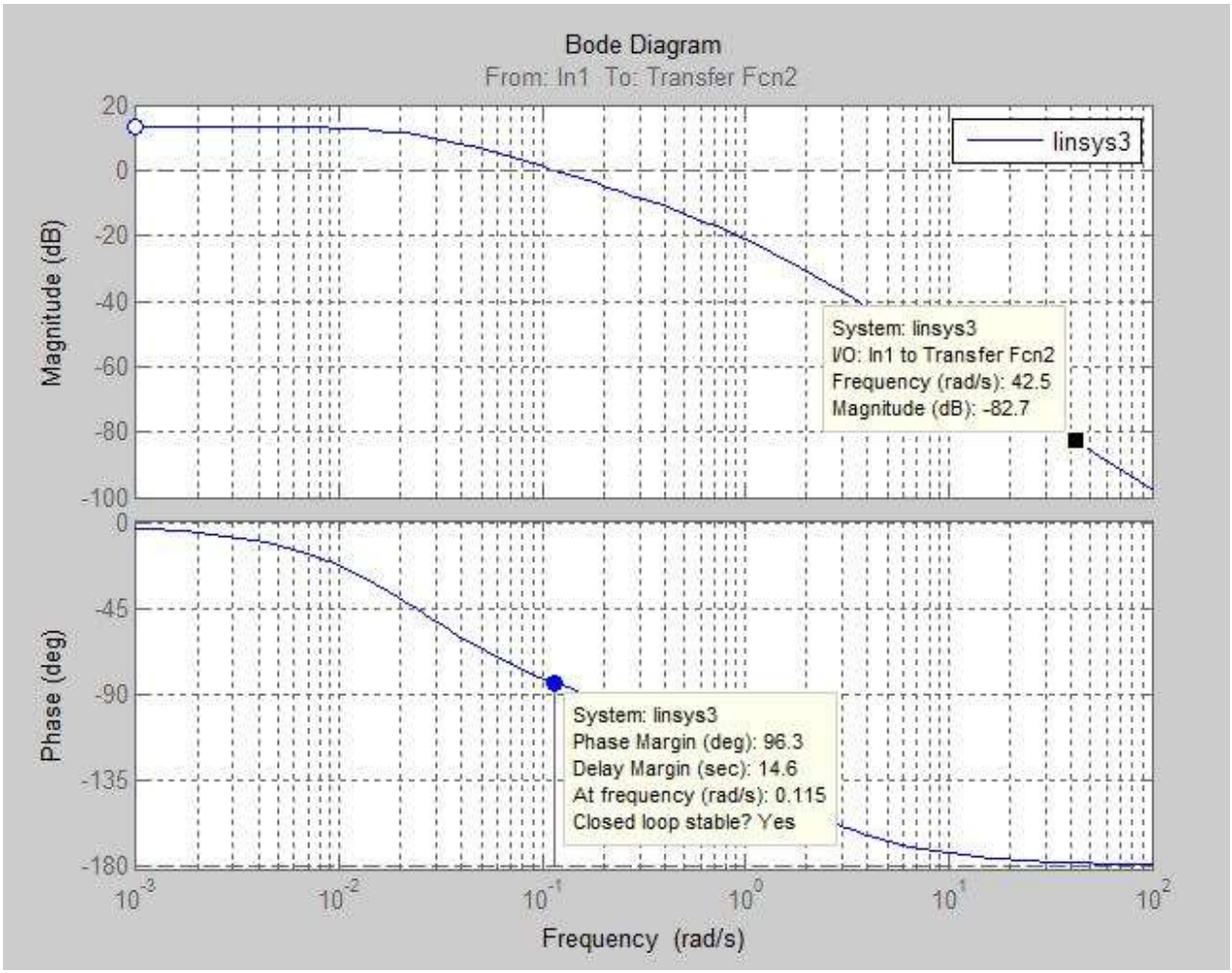


Рисунок 2.9 – Частотні характеристики нескоректованої розімкнутої системи управління нагріванням

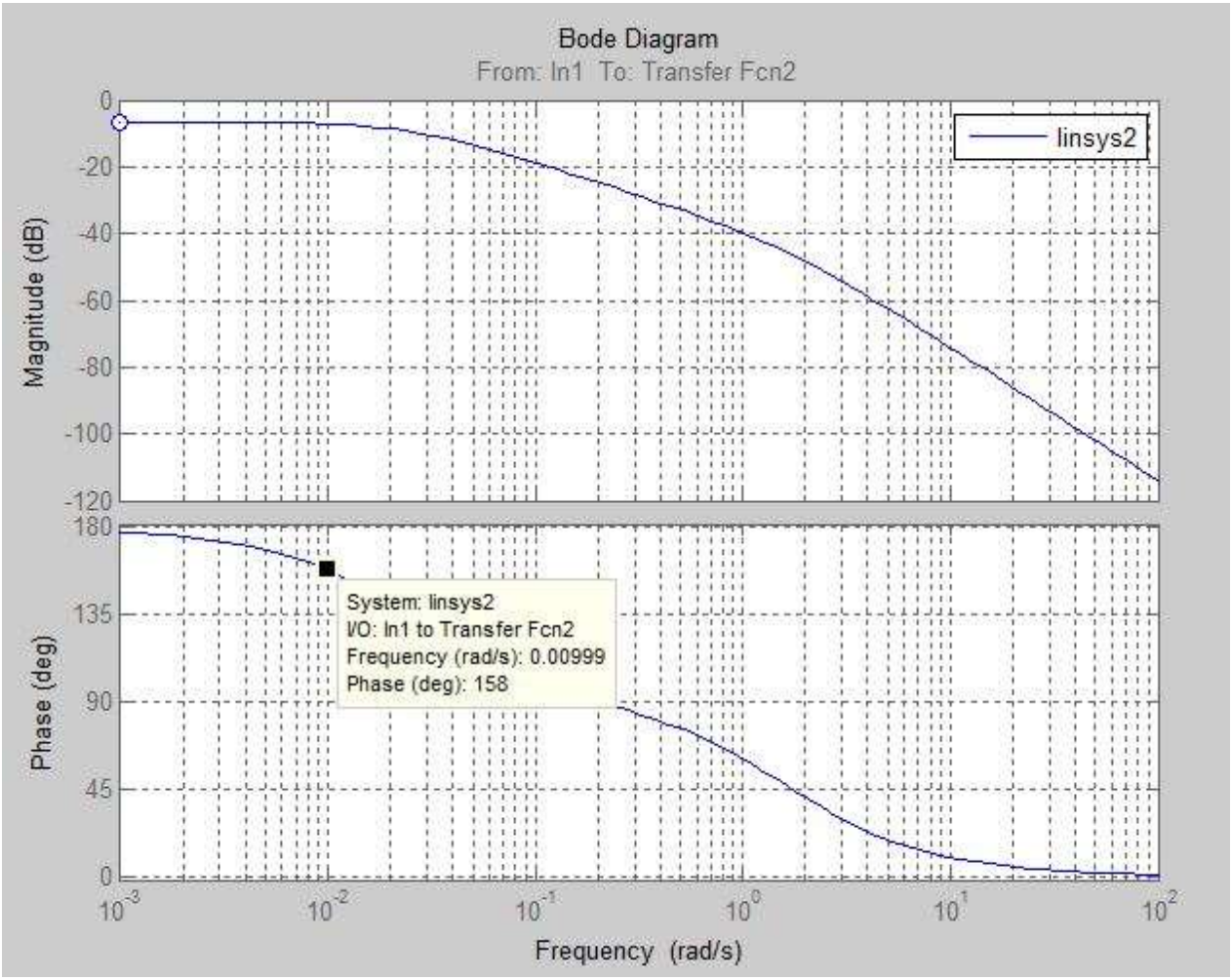


Рисунок 2.10 – Частотні характеристики нескорегованої розімкненої системи управління у режимі охолодження

Проведемо аналіз стійкості вихідної системи

Вирішуючи характеристичне рівняння – знаменник передавальної функції видно із виразу (2.29).

$$D(s) = 7.2s^3 + 44.18s^2 + 41.1s + 1 \qquad (2.29)$$

з'ясували, що коріння цього рівняння знаходиться у лівій площині (s1= -0,025, s2 = -1,11, s3 = -5), що говорить про стійкість цієї замкнутої системи управління.

2.4 Синтез закону управління для пристрою автоматичного управління

Синтез коригувального пристрою провадиться з використанням логарифмічних амплітудно-частотних характеристик – ЛАЧХ. У цій системі необхідно зменшити час перехідного процесу. Система автоматичного керування термостатом є стійкою.

Побудова ЛАЧХ розімкнутої системи управління тепловим режимом.

Передатна функція розімкнутої системи представлена в наступному вигляді видно із виразу (2.30).

$$W_u(S) = \frac{U_{ЗМ}(s)}{m(s)} = \frac{48.165}{0.14(40s+1)(0.9s+1)(0.2s+1)} = \frac{k}{s(T_1 s+1)} = \frac{344.04}{(40s+1)(0.9s+1)(0.2s+1)} \quad (2.30)$$

Частоти сполучення асимптот видно із виразу (2.31), (2.32).

$$\omega_1 = \frac{1}{40} = 0.025\frac{рад}{c}; log(0.025) = -1.6(дек) \quad (2.31)$$

$$\omega_2 = \frac{1}{0.9} = 1.1\frac{рад}{c}; log(1.1) = 0(дек) \quad (2.32)$$

$$\omega_3 = \frac{1}{0.2} = 5\frac{рад}{c}; log(5) = 0.7(дек) \quad (2.33)$$

Побудова починається з низькочастотної ділянки або низькочастотної асимптоти, положення якої визначається коефіцієнтом передачі розімкнутої системи; далі будуємо асимптотичні властивості всіх ланок передавальної функції. Графік ЛАЧХ представлений на рис. 2.11.

Побудуємо бажану ЛАЧХ. Бажану ЛАЧХ збудуємо по ділянках, на підставі вимог до показників перехідного процесу та запасів стійкості. Низькочастотна ділянка будується виходячи з умов забезпечення заданої точності функціонування системи в режимах, що встановилися. Середньочастотна ділянка бажаної ЛАЧХ будується на підставі заданих часу перехідного процесу $t_{ПП}$ та перерегулювання $\boldsymbol{\sigma, \%}$, як видно із виразу (2.34). Для визначення точки перетину середньочастотної асимптоти з віссю частот – частоти зрізу $\omega_{ср}$ користуються номограмою, що пов'язує перерегулювання $\boldsymbol{\sigma, \%}$, максимальний час переходу процесу $t_{Ппmax}$ та максимальне значення речової частотної характеристики замкнутої системи P_{max}. Ця номограма дозволяє за заданою величиною $\boldsymbol{\sigma, \%}$, визначити P_{max} і далі по кривій $t_{Ппmax}$ та заданому $t_{ПП}$ знайти частоту зрізу $\omega_{ср}$ разомкнутой системы (рис. 2.12)

$$t_{ПП} = \frac{K\pi}{\omega_{ср}} \rightarrow \omega_{ср} = \frac{K\pi}{t_{ПП}} \quad (2.34)$$

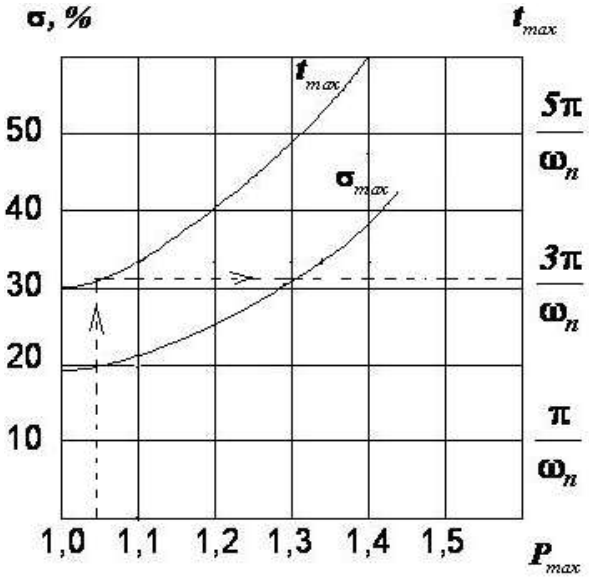


Рисунок 2.11 – Номограма, що служить визначення частоти зрізу

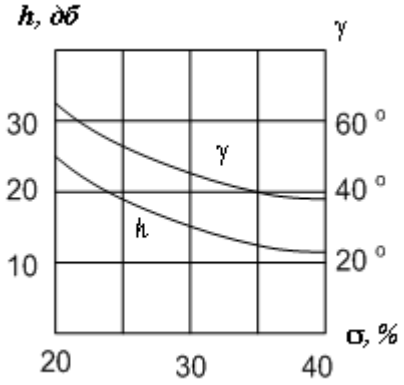


Рисунок 2.12 – Номограма, що служить для визначення запасів за модулем та фазою

Середньочастотна ділянка обмежується умовою необхідного запасу по модулю та фазі. Для цього користуються спеціальною номограмою, яка зв'язує запаси по модулю h і по фазі γ (мал. 1.14): h=25дБ, φ=65^0.

За кривими визначаємо, що середньочастотна ділянка лежить у межах $\pm L_м = 25$дБ

Високочастотна ділянка бажаної ЛАЧХ обмежується частотою $\omega_к \le (6 \div 10)\omega_{ср}$ і може мати будь-який нахил. Ця ділянка істотно не впливає на перехідний процес. Бажану ЛАЧХ зобразимо на рис. 2.13. Була побудована ЛАЧХ коригувального пристрою як різниця між бажаною ЛАЧХ та ЛАЧХ незмінною частиною системи як видно із виразу (2.35) и на (рис. 2.13).

$$L_K = L_ж - L_H \qquad (2.35)$$

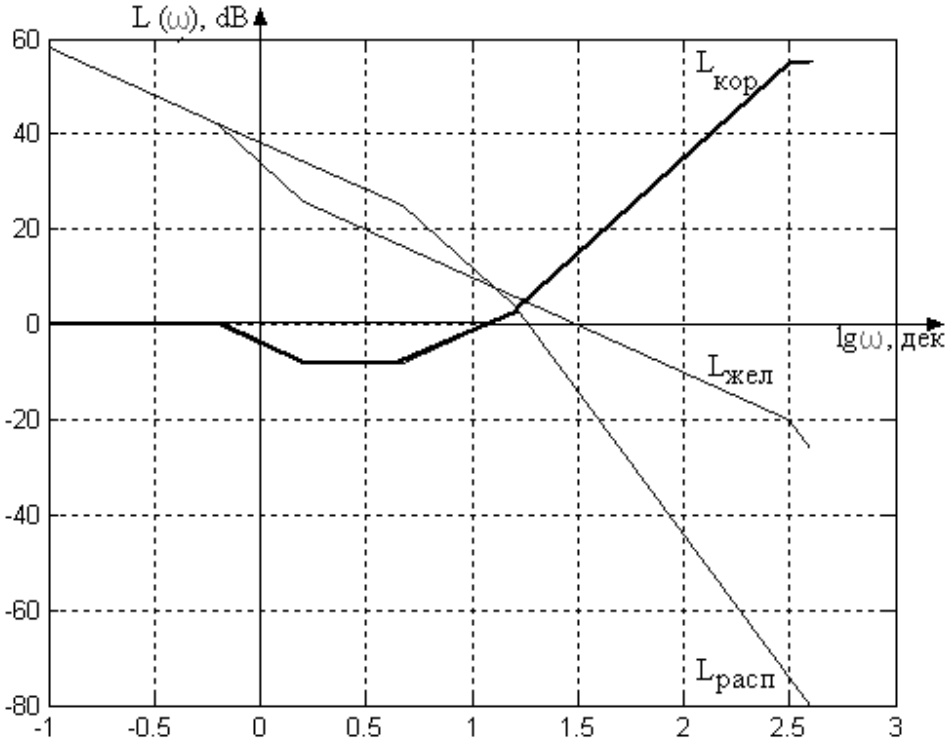


Рисунок 2.13 – Наявна, необхідна ЛАЧХ і ЛАЧХ коригувального елемента

Для побудови передавальної функції коригуючого пристрою, виходячи із частот, що сполучають, визначаємо постійні часу (табл. 2.1).

Таблиця 2.1 - Постійні часу коригувального пристрою

	1	2	3	4	5
lg (w)	- 0.2	0.2	2.5	1.2	0.67
T, c	1.59	0.63	0.00316	0.063	0.21

Передатна функція коригувального пристрою має вигляд як видно із виразу (2.36)

$$W_K(s) = \frac{U_k(s)}{\varepsilon(s)} = \frac{(0.63s+1)(0.21s+1)(0.063s+1)}{(1.59s+1)(0.00316s+1)\text{^}2} = \frac{0.0083s^3+0.1852s^2+0.903s+1}{0.0000159s^2+0.01s^2+1.6s+1} \quad (2.36)$$

Використовуючи Z-перетворення, представимо за допомогою середовища Matlab передатну функцію цифрового коригувального пристрою:

```
w1=tf([0.0083 0.1852 0.903 1],[0.0000159 0.01 1.6 1])
Transfer function:
 0.0083 s^3 + 0.1852 s^2 + 0.903 s + 1
-----------------------------------
1.59e-005 s^3 + 0.01 s^2 + 1.6 s + 1
>> w2=tf(c2d(w1,0.01))
Transfer function:
 522 z^3 - 1125 z^2 + 688.1 z - 84.65
```

---

z^3 - 1.074 z^2 + 0.08122 z - 0.001856

Здійснення вибірки часу: 0.01

2.5 Моделювання динаміки системи управління при заданих початкових умовах та зовнішніх впливах

Побудуємо тимчасові характеристики по дії, що задає, використовуючи середовище Matlab.

Передатна функція скоригованої розімкнутої системи управління як видно із виразу (2.37)

$$W(S) = \frac{m(s)}{U_{ЗМ}(s)} = \frac{78(0.63s+1)}{s(1.59s+1)(0.00316s+1)^2} \qquad (2.37)$$

Передатна функція скоригованої замкнутої системи управління як видно із виразу (2.38)

$$\Phi(S) = \frac{m(s)}{U_{ЗМ}(s)} = \frac{49.14s+78}{0.0000159s^4+0.00991s^3+1.5962s^2+50.14s+78} \qquad (2.38)$$

Графік перехідної характеристики системи управління зображений на рис. 2.14.

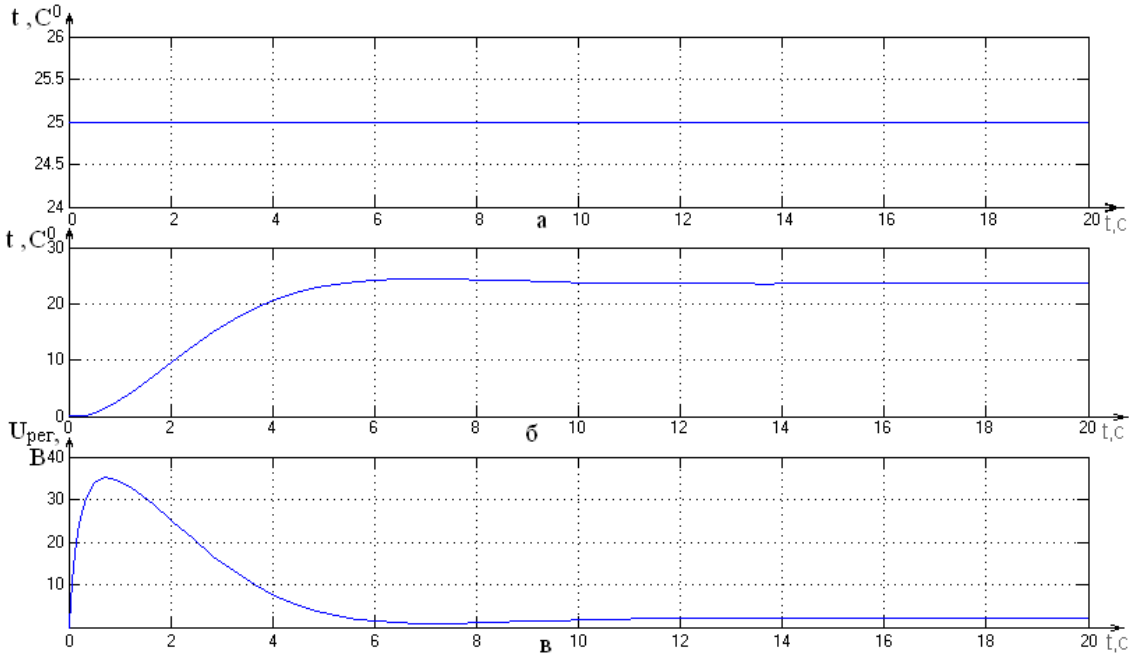


Рисунок 2.14 – Перехідна характеристика системи управління: а – вплив, що задає; б -вихідна температура; в – керуючий вплив.

На рис. 2.15 і 2.16 зображені частотні характеристики скоригованої замкнутої та розімкнутої системи управління від впливу, що задає відповідно.

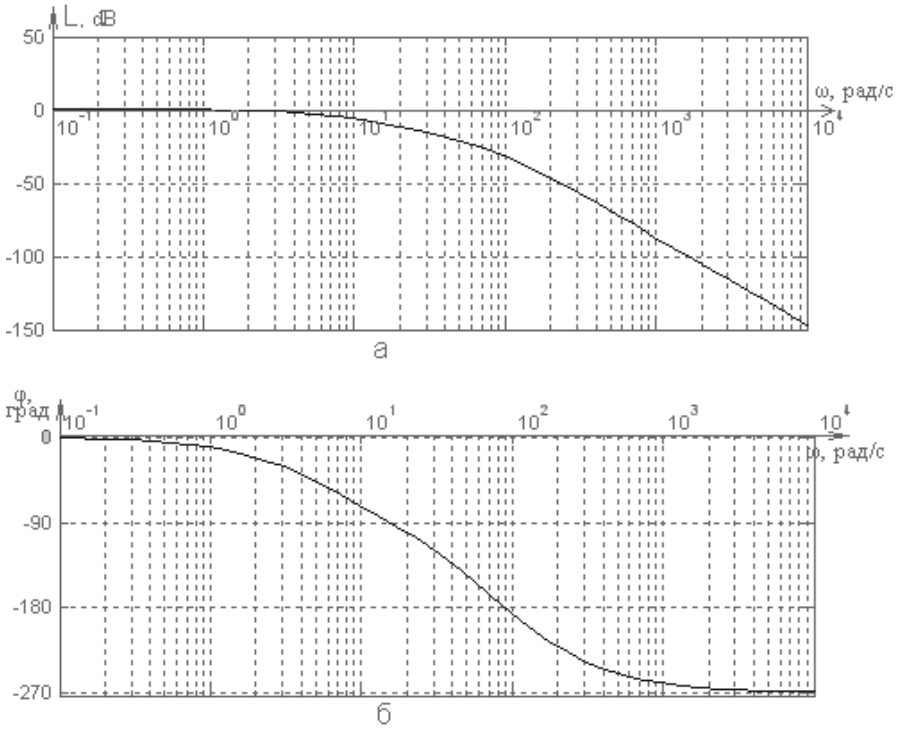


Рисунок 2.15 – Частотні характеристики скоригованої замкнутої системи управління від заданого впливу: а – амплітудно-частотна характеристика; б – фазо-частотна характеристика

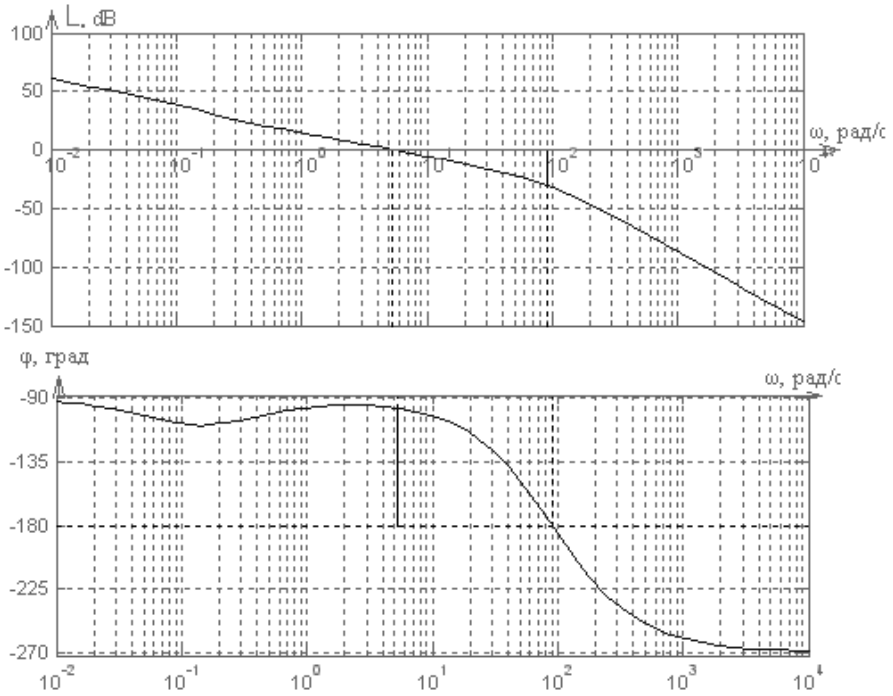


Рисунок 2.16 – Частотні характеристики скоригованої розімкнутої системи управління

Оцінка показників якості за перехідними характеристиками:

– перерегулювання: σ=0%;

– час перехідного процесу: tпп=5,8с.

Оцінка за частотними характеристиками замкнутої системи:

– Показник коливання: M=Amax/A(0)=1;

Оцінка за частотними характеристиками розімкнутої системи:

– Запас стійкості за амплітудою АЗ=30,6дБ

– Запас стійкості по фазі γЗ=82,2 град.

Вирішуючи характеристичне рівняння – знаменник передавальної функції як видно із виразу (2.39)

$$D(s) = 0.0000159s^4 + 0.00991s^3 + 1.5962s^2 + 50.14s + 78 \quad (2.39)$$

з'ясували, що коріння цього рівняння (-381.9808, -200.6277, -39.0215, -1.6404) знаходиться у лівій площині, що говорить про стійкість цієї замкнутої системи управління.

У цьому розділі проаналізували динамічні властивості системи керування термостатом, також здійснили синтез системи, що має задані показники якості. Поліпшили працездатність системи за допомогою введення нового пристрою, що коригує. Проаналізували показники якості нової синтезованої системи. Представили закон управління у наступному цифровому вигляді як видно із виразу (2.40)

$$W_K(Z) = \frac{U_K(Z)}{\varepsilon(Z)} = \frac{522z^3 - 1125z^2 + 688.1z - 84.65}{z^3 - 1.074z^2 + 0.08122z - 0.001856}. \quad (2.40)$$

Записали кінцево-різницеве рівняння закону управління як видно із виразу (2.41)

$$U_K[k] = 1.074U_K[k-1] - 0.08122U_K[k-2] + 0.001856U_K[k-3] + 522E[k] - 1125\,E[k-1] + 688.1E[k-2] - 84.65E[k-3]. \quad (2.41)$$

Для реалізації цього закону необхідні значення помилки та управління з трьох минулих циклів, а отже необхідно поставити нульові умови для U_K[k-1], U_K[k-2], U_K[k-3], E[k-1], E[k-2], E[k-3].

2.6 Висновок за розділом

У результаті виконання всіх етапів конструкторської частини та обгрунтування конструкції блока контролера враховані технічні вимоги, ергономіка та зручність у використанні для реалізації оптимального та функціонального мобільного додатка для управління термостатами в офісних приміщеннях. Це сприятиме забезпеченню стабільного та ефективного функціонування системи, забезпечуючи користувачам комфортні умови.

3. КОНСТРУКТОРСЬКА ЧАСТИНА

3.1 Задачі контролера і визначення складу вхідної і вихідної інформації

У результаті виконання аналізу і синтезу системи керування термостатом було отримано кінцевоо-різницеве рівняння закону керування як видно із виразу (3.1)

$$U_K[k] = 1.074U_K[k-1] - 0.08122U_K[k-2] + 0.001856U_K[k-3] + 522E[k] - 1125\,E[k-1] + 688.1E[k-2] - 84.65E[k-3]. \quad (3.1)$$

Для реалізації цього закону керування необхідно сконструювати цифровий контролер, основними завданнями якого будуть:

1) зчитування вихідного сигналу термодатчика;

2) розрахунок керуючого впливу на основі отриманих значень з датчика;

3) видача керуючого впливу на підсилювач потужності в контур управління нагрівального елемента;

4) видача керуючого впливу на підсилювач потужності в контур управління швидкості обертання електродвигуна охолоджувального елемента;

5) передача установки, сигналу управління і поточного параметра системи на ПЕОМ для відображення процесу управління термостатом.

Після аналізу алгоритмів керування видно, що вхідною інформацією для обчислювача є такий сигнал:

- з термодатчика (датчика температури) - діапазон значень [0,5] В;

Вихідною інформацією є двополярні сигнали $U_{рег1}(t)$ и $U_{рег2}(t)$ керуючого впливу - діапазон значень [-5;5] В.

3.2 Розроблення алгоритмічного забезпечення та оцінка необхідних обчислювальних ресурсів

Одним з етапів розроблення алгоритмічного забезпечення є вибір значення періоду дискретності T_0 і реалізація його в програмі контролера. Проаналізувавши розрахункові показники якості системи (час перехідного процесу $t_{пп}$=1 с), було обрано такий період дискретності T_0=0,1с.

Реалізація цього T_0 може бути виконана двома способами:

1) програмно, використовуючи затримку;

2) за допомогою таймера (за перериванням).

Основною перевагою реалізації часу дискретизації, використовуючи затримку, є простота алгоритму, проте великі ресурси процесора витрачаються

на його реалізацію. Цього недоліку позбавлений другий спосіб реалізації за допомогою таймера, що базується на використанні переривання за переповненням лічильного регістра таймера. Цей спосіб вимагає великих алгоритмічних витрат (необхідна програма для обробки переривання).

Повний алгоритм функціонування обчислювача для розв'язання задачі керування складається з етапів, які виконуються циклічно з періодом T_0.

1) Приймання коду АЦП зі значенням напруги, пропорційної заданій температурі.

2) Масштабування і перетворення формату для отриманих даних з датчика.

3) Реалізація обчислювальних залежностей відповідно до співвідношень.

4) Очікування завершення періоду, реалізоване через очікування сигналу переривання від таймера.

5) Реініціалізація таймера (запис стартового числа) і перехід до пункту 1.

Асинхронними подіями в розроблюваній системі є сигнали переривань. Джерелами переривань є таймер, який реалізує період дискретності обчислень T_0, і послідовний порт, який формує переривання по завершенню приймання байта. Під час роботи МК можлива ситуація, коли одночасно надходять запити на переривання від різних джерел. Для запобігання конфліктам у МК51 реалізовано дворівневу апаратно-програмну шкалу пріоритетів, відповідно до якої пристрій керування вибирає джерело переривання, яке необхідно обслужити першим. Переривання від таймера повинні мати більший пріоритет. Для синхронізації роботи послідовного порту необхідно використовувати другий таймер, однак переривання від нього не потрібно обробляти програмно.

Блок-схему алгоритму керування зображено на рис.3.1.

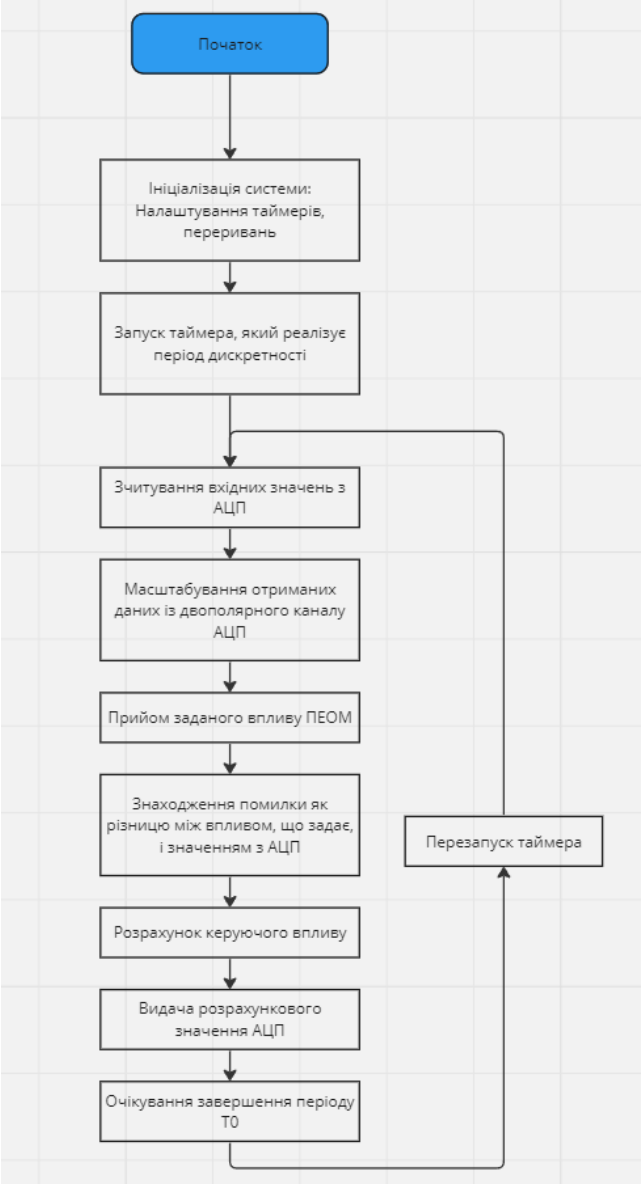


Рисунок 3.1 - Блок-схема алгоритму

3.3 Розроблення структури цифрового контролера

Оскільки всі вхідні параметри алгоритму надходять безпосередньо з датчика, попередня обробка (диференціювання, інтегрування тощо) не потрібна.

Вхідні дані надходять із датчика в аналоговому вигляді, а отже, їх необхідно перетворити на цифровий вигляд. Для цього необхідний блок аналого-цифрового перетворення сигналу. Формат вхідних і вихідних даних визначає точність подання інформації в цифровому вигляді. Необхідна довжина розрядної сітки визначається за такою залежністю як видно із виразу (3.2)

$$n = n_{АЦ} + n_{В} \quad (3.2)$$

де $n_{АЦ}$ - розрядність аналого-цифрового перетворювача (АЦП), $n_В$ - додаткові розряди для компенсації обчислювальної похибки. Значення $n_{АЦ}$ розраховується виходячи із забезпечення точності перетворення не гірше за точність вимірювань, адже $\delta_{АЦ} < \delta$; або $\delta_{АЦ} < 0.5\%$. Абсолютна похибка $\Delta_{АЦ}$ -перетворення видно із виразу (3.3)

$$\Delta_{АЦ} = \delta_{АЦ} * \text{Umax} = 0{,}005 \cdot 12 = 0{,}06\text{В} \qquad (3.3)$$

де U_{max} - максимальне значення сигналу, прийнятого з датчиків. Знайдемо необхідну розрядність аналого-цифрового перетворювача видно із виразу (3.4)

$$n_{АЦ} = \log_2 \alpha_{MAX} - \log_2 (\delta_{АЦ} \cdot U_{max}) = \log_2 10 - \log_2 (0{,}06) = 7{,}64. \qquad (3.4)$$

З урахуванням округлення і похибки перетворення АЦП (2-3 мл.г.) приймаємо значення $n_{АЦ} = 10$ розрядів, тобто АЦП має бути як мінімум десятирозрядним. Виходячи з розглянутої оцінки та аналізу вхідних і вихідних даних, було сформовано структуру цифрового контролера, зображену на рис. 3.2.

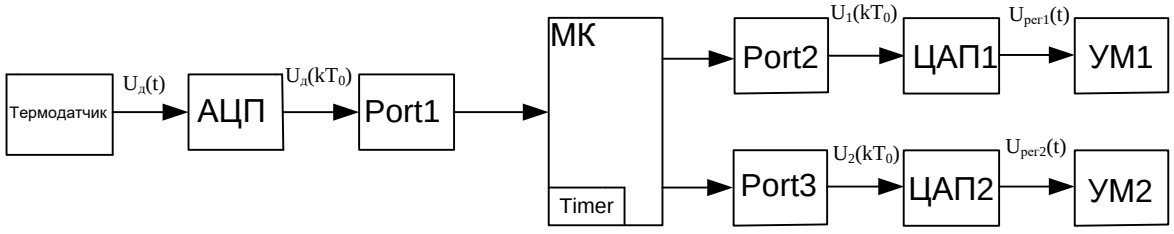


Рисунок 3.2 - Функціональна схема обчислювального регулятора

3.4 Вибір елементної бази для реалізації контролера

Для реалізації контролера було проведено порівняльний аналіз елементної бази, під час якого було обрано такі елементи:

- AT89S52 - мікроконтролер сімейства MCS-51;
- К1113ПВ1 - десятирозрядний АЦП;
- AD7524 - перемножувальний 8-розрядний ЦАП;
- TL074 - чотириканальний JFET операційний підсилювач із низьким рівнем шумів.

Розглянемо детальніше кожен з елементів контролера.

AT89S52 - мікроконтролер сімейства MCS-51, на кристалі якого реалізовано:

- восьмирозрядний процесор (усі регістри та АЛУ мають розрядність 8 біт);
- генератор тактових імпульсів (ГТІ) для процесора;
- внутрішнє ПЗП - резидентна пам'ять програм (РПП) - 4 Кбайт;
- внутрішнє ОЗП - резидентна пам'ять даних (РПД) - 256 байт;
- таймери-лічильники - два однотипних пристрої (таймер Т0 і таймер Т1);
- послідовний порт UART;
- контролер переривань (на п'ять сигналів);
- чотири порти паралельного введення-виведення (кожен по 8 біт) - Р0, Р1, Р2, Р3;
- підсистема керування живленням мікроконтролера.
- Основні електричні параметри мікроконтролерів MCS-51:
- напруга живлення +5В;
- рівні вхідних і вихідних сигналів - ТТЛ;
- навантажувальна здатність ліній портів - 4 входи ТТЛ (8 входів для порту Р0);
- базова тактова частота - 12 МГц.

Розподіл виводів мікроконтролера AT89S52 показано на рис.3.3.

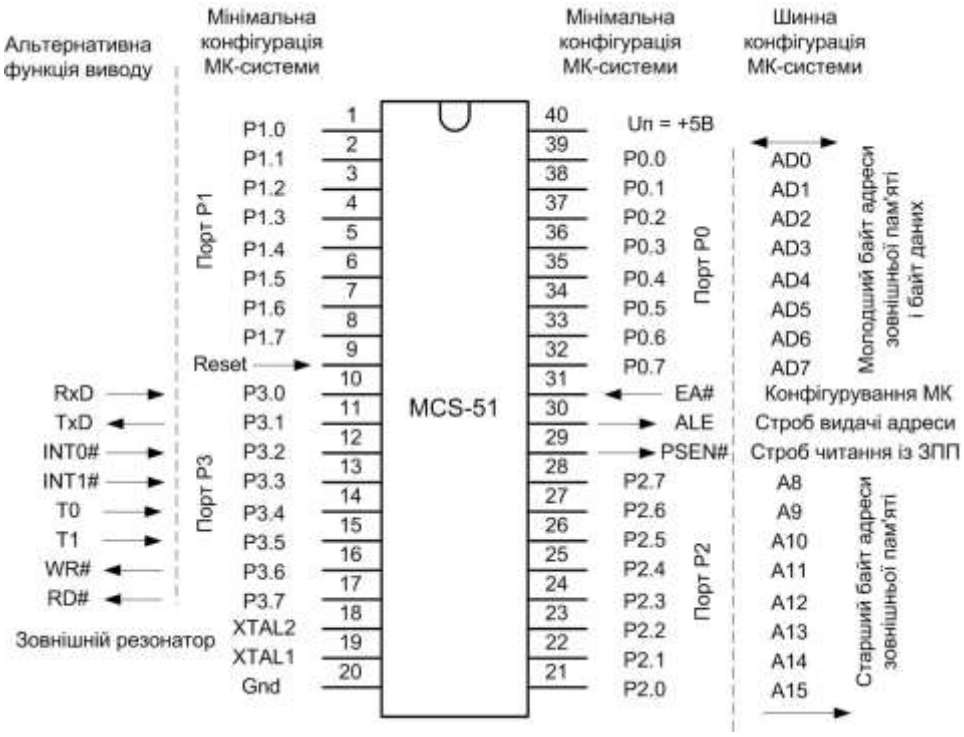


Рисунок 3.3 – Електричний інтерфейс базового мікроконтролера MCS-51

Як видно з рис.3.3, обчислювальна система на основі мікроконтролера (МК-система) може бути виконана у двох варіантах:

1) мінімальна конфігурація - усі зовнішні пристрої системи (датчики, виконавчі пристрої, елементи індикації та керування) під'єднують безпосередньо до портів МК або до окремих виводів портів через додаткові буферні елементи (наприклад, шинні формувачі); таку конфігурацію можна реалізувати, якщо кількість зовнішніх пристроїв можна порівняти з кількістю портів МК;

2) шинна конфігурація - на основі портів P0 і P2 можуть бути організовані системні шини адреси та даних; усі зовнішні пристрої системи підключаються до системних шин через буферні регістри, що адресуються процесором; у такий самий спосіб у системі може бути реалізована зовнішня додаткова пам'ять.

Оскільки передбачається використовувати тільки внутрішню пам'ять мікроконтролера, то елементи АЦП і ЦАП можна під'єднати безпосередньо до портів AT89S52, і всі сигнали для цих елементів формуватимуться програмним шляхом.

Вхідна інформація представлена одним аналоговим сигналом, а вихідна - 2 аналоговими. Таким чином, необхідно використовувати аналого-цифровий перетворювач для вхідного сигналу з датчика.

З метою скорочення кількості допоміжних елементів розроблено пристрій, функціонально закінчений, сумісний із мікропроцесорами, що працюють із TTL-рівнями. АЦП має внутрішнє джерело основної напруги, генератор годинника і компаратор напруги.

У цьому АЦП процес конвертації виконано в "0" на вході $B/\overline{C}$ (блок - перетворення). Для завантаження поточного цільового коду конвертера необхідно подати одиницю (мінімум на 2 мкс) на вході $B/\overline{C}$. Після того, як подається "0" на вхід, починається новий цикл перетворення. Після завершення перетворення на вході $\overline{DR}$ (Готовий) є сигнал "0". Протягом завантаження та перетворення на цій вихідній одиниці підтримані та інформаційні виходи АЦП перебувають в умові високого імпедансу. Після завершення перетворення, одночасно з сигналом готовності інформаційного виходу, встановлено код, що відповідає результату перетворення. Ланцюг увімкнення АЦП К1113ПВ1 подано на мал. 3.4.

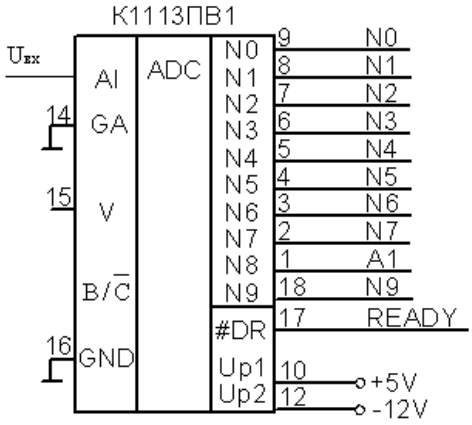


Рисунок 3.4 - Позначення АЦП К1113ПВ1

Інтегральна схема має пристрої виходу з трьома стійкими умовами, які спрощують її інтерфейс на стовбур даних мікропроцесора. Маленький АЦП може служити одному мікропроцесору і навпаки. Час перетворення T_{tr} становить 30 мкс. Зважаючи на завантаження / початок і приймання даних загальний цикл перетворення становить максимально 50 мкс, отже на АЦП такого типу можливо реалізувати опитування сигналів із частотою до 20 кГц.

Основні електричні параметри мікросхеми АЦП К1113ПВ1 наведено в табл. 3.1.

Таблиця 3.1 - Електричні параметри К1113ПВ1

	Електричні параметри	Напруга живлення
1	Номінальна напруга живлення	5 В 5 %
	Uп1,	
	Uп2	-15 В 5 %
2	Вихідна напруга низького рівня	не більше 0,4 В
3	Вихідна напруга високого рівня	не менше 2,4 В
4	Напруга зміщення нуля в однополярному та біполярному режимах від повної шкали	0,3%
5	Струм споживання	
	від джерела живлення Uп1	не більше 10 мА
	від джерела живлення Uп2	не більше 18 мА
6	Вхідний струм високого (низького) рівня	не більше 10 мА 40 мкА

У структурі обчислювача є засоби для мультиплексування вихідних цифрових сигналів, їх перетворення в аналогову форму.

Оскільки в контурі керування термостатом виконавчий пристрій з аналоговим входом, то для перетворення сигналу використовуємо ЦАП, для цього вибираємо мікросхему AD7524 (рис. 3.5).

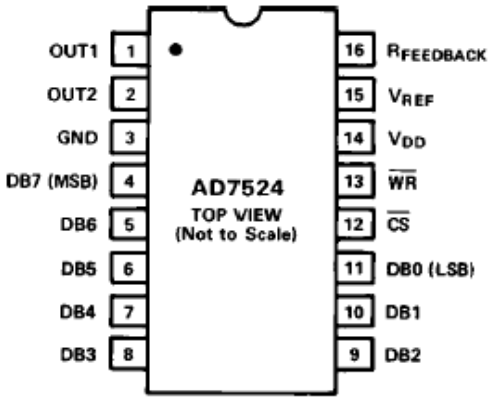


Рисунок 3.5 – Графічне зображення мікросхеми AD7524

Призначення висновків: Vref – вхід джерела опорної напруги; CS(ChipSelect) – вибір мікросхеми; WR(Write) – запис; DB0_DB7 – лінії вводу данних; GND – загальний; Out1,Out2 – виходи ЦАП; RFB – вивід резистора ланцюга зворотнього зв’язку; VDD – вивід живлення.

Для подключеннння пристроїв до контролера необхідні узгоджуючі посилювачі, які представленні мікросхемою TL074 (рис.3.6). Основні параметри відображені у табл.3.2.

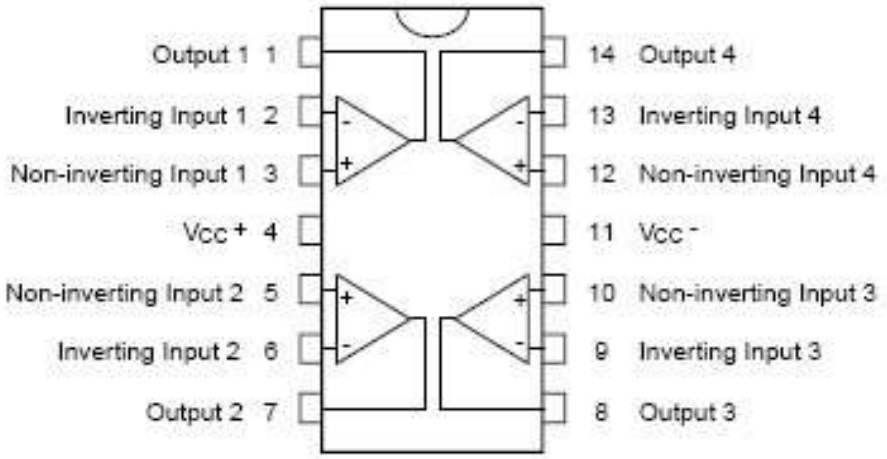


Рисунок 3.6 – Зображення мікросхеми TL074

Таблиця 3.2 – Основні параметри TL074

Параметри модуля	Значення параметрів
Каналів, шт	4
VOS (тип.),мВ	10
IBIAS (тип.),нА	0.2
Смуга пропускання (тип.), МГц	3
Slew Rate (тип.),В/мкс	13
CMRR (тип.),дБ	86
Gain (тип.),дБ	86
Shutdown	Ніт
VCC,В	Від 6 до 36
ICC на канал (макс.),мА	1.4
TA,°C	Від -40 до 105
Корпус	PDIP 14 SOIC-14

Крім того, передбачено зв'язок із ПК через послідовний порт для отримання даних про необхідну температуру і передавання інформації про процес керування. Для виконання цієї функції використовується мікросхема MAX232.

MAX232 - інтегральна схема, що перетворює сигнали послідовного порту RS-232 в сигнали, придатні для використання в цифрових схемах на базі ТТЛ або КМОП технологій. MAX232 працює приймачем і перетворює сигнали RX, TX, CTS і RTS. Схема забезпечує рівень вихідної напруги, який використовується в RS-232 (приблизно ±7.5 В), перетворюючи вхідну напругу +5 В за допомогою внутрішнього зарядового насоса на зовнішніх конденсаторах. Це спрощує реалізацію RS-232 у пристроях, що працюють на напругах від 0 до +5В, оскільки не потрібно ускладнювати джерело живлення тільки для того, щоб використовувати RS-232.

3.5 Формування протоколів обміну даними між елементами контролера

Протокол видавання та приймання сигналів керування для блока АЦП складається з таких дій:

1. Сформувати сигнал скидання АЦП на 3-4 мкс.
2. Видати сигнал запуску АЦП.
3. Опитувати лінію #Ready до появи сигналу з рівнем "0".

4. Прочитати вихідний код АЦП через порти МКС51 у резидентну пам'ять даних, причому спочатку молодший байт через порт Р1, а потім 2 старших біти - через лінії Р2.2, Р2.3.

Протокол видачі сигналів керування на блок ЦАП визначається такими діями:

1. Видати через порт Р0 старші 8 бітів 10-розрядного вихідного коду на ЦАП.

2. Видати через лінії Р2.0, Р2.1 два молодших розряди вихідного коду на ЦАП.

3.6 Висновок за розділом

На основі проведеного аналізу та синтезу системи автоматичного управління термостатом кондиціонування офісних приміщень можна зробити наступні висновки:

Обрана функціональна схема системи базується на детальному аналізі вимог щодо регулювання температури в офісних приміщеннях. Обрані компоненти системи повинні забезпечити ефективне управління термостатом, враховуючи різні зовнішні та внутрішні фактори. Проведена робота з розробки математичних моделей об’єкта автоматичного управління, також враховуючи динаміку елементів системи. Ці моделі дозволяють детально аналізувати та прогнозувати поведінку системи при змінах у вхідних параметрах. Здійснено аналіз властивостей об’єкта автоматичного управління, зокрема стійкості, чутливості до збурень та інших параметрів. Це дозволяє визначити оптимальні умови для стабільної роботи системи. Розроблено оптимальний закон управління, який враховує вхідні сигнали, вимоги до температури та властивості системи. Цей закон має забезпечити точне та стабільне управління термостатом для забезпечення комфортних умов в офісних приміщеннях. Проведено моделювання динаміки системи управління з використанням розроблених моделей. Це дозволяє ефективно тестувати та вдосконалювати систему до впровадження в реальні умови.

4. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ З УПРАВЛІННЯ КОНТЕНТОМ НА САЙТІ

4.1 Аналіз вимог до CMS-додатку

CMS – це абревіатура від "Content Management System" (Система управління контентом). Це програмне забезпечення, призначене для ефективного створення, редагування та організації контенту на веб-сайтах без необхідності глибоких технічних знань з програмування. CMS робить процес управління веб-сайтом більш простим і доступним для користувачів різних рівнів технічної підготовки.

Веб-сайт є електронною платформою, що складається з колекції веб-сторінок, які можуть містити текстовий, графічний, мультимедійний та інший контент. Це простір у Інтернеті, доступний користувачам за допомогою веб-браузера, де вони можуть отримувати інформацію, спілкуватися, здійснювати покупки, а також здійснювати інші онлайн-дії. Веб-сайти можуть виступати як інформаційні ресурси, електронні магазини, соціальні платформи тощо. Ці платформи дозволяють взаємодіяти з інтернет-середовищем та забезпечують доступ до різноманітних сервісів та можливостей в онлайн-середовищі.

Сервер – це комп'ютер або програмне забезпечення, яке забезпечує обслуговування запитів інших комп'ютерів, відомих як клієнти, у мережі. Основна його функція полягає в наданні ресурсів, обробці запитів та забезпеченні з'єднання між користувачами та необхідними інформаційними ресурсами. У науковому контексті сервер визначається як обчислювальний вузол, який відповідає на запити і надає доступ до ресурсів, таких як файли, дані чи послуги. Серверні системи є необхідним елементом в інформаційних технологіях, забезпечуючи ефективний обмін даними та роботу мережі. Вони використовуються для зберігання, обробки, аналізу та надання доступу до інформації, що дозволяє покращити продуктивність та ефективність роботи комп'ютерних систем.

Функціональні вимоги:

Функціональні вимоги до сайту виробника термостатів є визначальним етапом у розробці системи, спрямованої на забезпечення ефективної та зручної інтеракції з користувачем. Зазначені вимоги включають такі аспекти:

1) Інформація про виробника:

Забезпечення розділу на сайті з повною та достовірною інформацією про виробника термостатів, що може включати історію компанії, партнерство, сертифікати якості тощо.

2) Меню для навігації:

Розробка ієрархічної системи меню для зручної навігації користувача на сайті, включаючи родительські категорії, категорії та подкатегорії, які відображають різні аспекти та моделі термостатів.

3) Лінійка товарів:

Створення розділу, що включає повний асортимент термостатів, представлений із заголовками, фотографіями, докладними описами, характеристиками, що дозволяє користувачеві отримати повну інформацію перед покупкою.

4) Контактна інформація:

Введення розділу з контактною інформацією для забезпечення можливості зв'язку користувачів із виробником. Це може включати електронну пошту, телефонні номери та можливість запитань через онлайн-форми.

Функціональні вимоги до CMS-додатку, призначеної для управління вмістом сайту виробника термостатів, визначаються рядом ключових функціональних можливостей:

1) Редагування структури:

Забезпечення можливості редагування родительських категорій, категорій, подкатегорій та карточок товарів з метою ефективного оновлення ієрархії продуктів на сайті.

2) Логування користувачів:

Розробка системи логування для реєстрації та авторизації користувачів для забезпечення безпеки та ідентифікації осіб, які мають доступ до функцій CMS.

3) Управління доступом:

Впровадження функціоналу для управління правами доступу до функцій CMS для різних користувачів. Це включає надання адміністраторських прав та обмеження доступу для інших користувачів.

4) Редагування вмісту:

Забезпечення можливості редагування описів, характеристик та зображень карточок товарів для швидкого та зручного оновлення інформації.

Функціональні вимоги до сервера, що використовується для підтримки CMS-системи управління контентом на сайті виробника термостатів, визначаються низкою ключових можливостей:

1) Забезпечення стабільності та швидкодії:

Забезпечення надійної роботи сервера для забезпечення стабільності та ефективності функціонування CMS-системи.

2) Масштабованість:

Розробка сервера з урахуванням можливостей масштабування для забезпечення оптимальної продуктивності при збільшенні обсягу даних та навантаження.

3) Безпека даних:

Впровадження заходів безпеки для захисту конфіденційної інформації, що включає в себе шифрування даних, захист від несанкціонованого доступу та інші аспекти кібербезпеки.

4) Логування:

Реалізація системи логування для фіксації подій та аудиту для відстеження дій користувачів та забезпечення контролю за діяльністю системи.

5) Управління доступом:

Реалізація системи управління доступом для надання та обмеження прав доступу різним користувачам CMS-системи відповідно до їх ролей та відповідальностей.

4.2 Недоліки існуючих програм CMS

Аналізуючи існуючі системи управління контентом (CMS), такі як WordPress, Joomla та Magento, можна виявити деякі недоліки, які можуть впливати на їхню ефективність та придатність для певних сценаріїв використання.

1. Загальна Продуктивність:

WordPress: Для великих та складних сайтів може виникати проблема з продуктивністю, оскільки WordPress в першу чергу розроблявся для блогів, і його можливості масштабування можуть бути обмеженими.

Joomla та Magento: Також можуть виявитися не такими ефективними для великих магазинів чи складних веб-сайтів через свою структуру та об'ємність коду.

2. Безпека:

WordPress: Завдяки своїй популярності, WordPress є мішенню для кіберзлочинців, і недоліки в безпеці можуть призвести до атак та порушень безпеки.

Joomla та Magento: Також можуть виявитися вразливими перед атаками, особливо якщо не використовуються останні оновлення та патчі для систем безпеки.

3. Специфічність для Великих Магазинів:

WordPress та Joomla: Не завжди найкращий вибір для великих електронних комерційних проектів через свою спрощену структуру та обмежені функціональність.

Magento: Хоча Magento спеціалізується на електронній комерції, він може бути важким у використанні для менших сайтів, оскільки більшість його можливостей не потрібні для менших проектів.

4. Складність та Вивченість:

Joomla: Деякі користувачі можуть виявити Joomla складною у вивченні через свою розгалужену структуру та велику кількість параметрів.

Magento: Магенто, в свою чергу, також може здаватися важким для вивчення та використання, особливо для новачків у сфері електронної комерції.

5. Екосистема та Розширюваність:

WordPress: Хоча велика кількість плагінів доступна, якість деяких може залишати бажати кращого, іноді призводячи до конфліктів та проблем з роботою.

Joomla та Magento: Вони можуть вимагати спеціалізованого програмування для створення додаткового функціоналу, що може зробити їх менш гнучкими для деяких проектів.

4.3 Вибір технологій

В рамках розробки дипломного проекту вибір технологій визначений з урахуванням низки фундаментальних принципів та стратегічних переваг. Весь процес створення реалізований на основі мікросервісної архітектури, яка визначається як система, побудована з невеликих, автономних модулів, що взаємодіють між собою через мережу зображенно на рис 4.1. Це підходить для створення гнучкої та масштабованої системи, де різні функціональні елементи можуть функціонувати незалежно, підвищуючи робочу ефективність та забезпечуючи легкість супроводження.

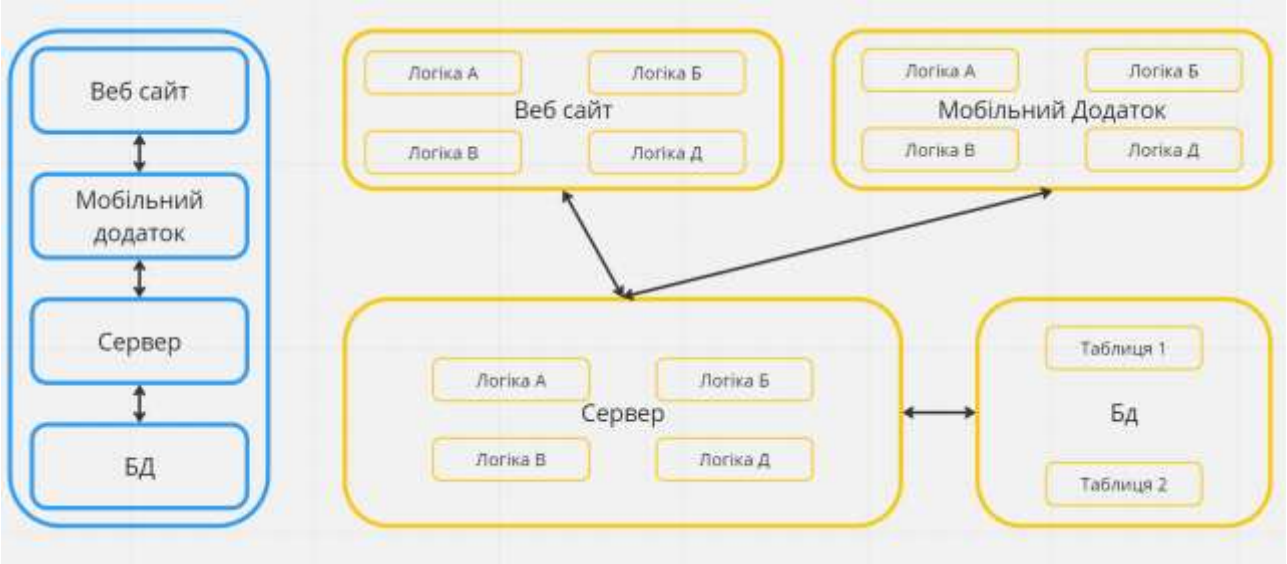


Рисунок 4.1 - Стркутра монолітної програми та мікросервісна архитектура

Одномовний підхід у виборі мови програмування відділяється на користь JavaScript, яка є високорівневою та мовою широкого вжитку. Це забезпечує однорідний технічний стек та уніфікований підхід до розробки різних компонентів системи, а також сприяє спрощенню інтеграції між ними.

У контексті розробки веб-сайту використовується фреймворк React, визнаний своєю ефективністю та зручністю для розробки інтерактивних та динамічних інтерфейсів. Для створення мобільної CMS мобільного додатка використовується React Native, що дозволяє ефективно розробляти мобільні додатки для різних платформ з використанням знайомого JavaScript.

Серверна частина проекту вирішується за допомогою фреймворка Node.js, який відзначається швидкодією та високою масштабованістю, що є важливим у контексті обробки багатьох одночасних запитів.

Взаємодія між різними компонентами системи, а саме між сервером, веб-сайтом та CMS мобільного додатка, реалізована за допомогою REST API та обміном даними у форматі JSON. Це стандарт для взаємодії між веб-сервісами, що дозволяє забезпечити стандартизований та ефективний обмін інформацією.

Безпека бази даних вирішується за допомогою JWT Refresh Token та JWT Access Token. Цей підхід надає додатковий рівень захисту для конфіденційної інформації та забезпечує аутентифікацію користувачів.

4.4 Врахування та Визначення Унікальних Особливостей у Системі CMS для Веб-Сайту

Аналіз технічних характеристик устаткування та їх вплив на управління кондиціонуванням є ключовою частиною дослідження у контексті виробництва термостатів для кондиціонування офісних приміщень. Детальний аналіз цих технічних параметрів дозволяє визначити оптимальні стратегії управління та забезпечити ефективну роботу систем кондиціонування.

Однією з ключових технічних характеристик є точність вимірювання температури. У виробництві термостатів це визначається датчиками, які мають забезпечувати надійне та точне вимірювання температурних показників. Недостатня точність може впливати на якість регулювання кондиціонування та призводити до неефективності системи.

Іншим важливим аспектом є швидкодія реакції технічних систем, особливо у випадках різких змін температури. Швидкість регулювання термостатів визначається характеристиками регуляторів та їх можливістю швидко адаптуватися до змін зовнішніх умов.

Також важливо розглядати інші параметри, такі як діапазон робочих температур, стійкість до впливу електромагнітного випромінювання, енергоефективність та можливість інтеграції з іншими системами автоматизації.

Вплив цих технічних характеристик на управління кондиціонуванням полягає в їхньому взаємодії з системою управління. Наприклад, якщо датчики надто повільно реагують на зміни температури, це може призводити до перегріву або недостатнього охолодження приміщення. Отже, важливо розробити ефективні алгоритми управління, які враховують особливості технічних характеристик термостатів.

4.5 Огляд мікросервісної архітектури та визначення її застосовності для веб-сайту, та мобільного додатку.

В сучасному інформаційному суспільстві стрімко розвиваються технології, що змінюють обличчя програмного забезпечення та веб-розробки. Однією з ключових концепцій, яка знайшла широке застосування в розробці веб-сайтів та мобільних додатків, є мікросервісна архітектура. Цей підхід до розробки програмного забезпечення визначається як розбиття додатку на невеликі, автономні та взаємодіючі сервіси.

Мікросервісна архітектура стає необхідною в умовах постійно зростаючої складності веб-проектів та збільшення вимог до їх гнучкості та масштабованості. Основні принципи цієї архітектури полягають у розподіленні

функціональностей між окремими мікросервісами, що надає можливість незалежно розвивати, масштабувати та підтримувати кожен сервіс.

Важливою особливістю є розбиття додатку на невеликі компоненти, кожен з яких виконує свою конкретну функцію.

Принципи створення компонентів у мобільному додатку, які відповідають за різні функціональні аспекти. Важливим елементом є універсальний підхід до написання компонентів, що дозволяє їм ефективно взаємодіяти та відображати різні дані у додатку.

Кожен компонент призначений для конкретної функції, і відповідає за свої завдання в системі. Особливою рисою є передача інструкцій при виклику компонента, які забезпечують його коректну роботу. Усі ці інструкції описані у вигляді функцій, розташованих у файлі «admin_systemFile.js».

Важливою функціональністю є можливість компонентів взаємодіяти з сервером, як показано у прикладі функції для запиту на сервер для отримання списку категорій. Це підкреслює універсальність та гнучкість підходу до розробки мобільного додатку.

У попередньому зазначеному прикладі, функція приймає аргумент у вигляді змінної, що містить ідентифікатор категорії та параметри для належного отримання даних з сервера. Цей аргумент використовується для налаштування конфігурації запиту, який надсилається до сервера з метою отримання відповідних інформаційних ресурсів.

У наступному прикладі, під час завантаження головної сторінки, за замовчуванням вибирається змінна 'ParentCategories'. Після цього викликається функція, яка проводить пошук подібної змінної. Після знаходження відповідної змінної, вона повертає відповідний компонент для відображення таблиці батьківських категорій у відповіді. Весь код є у додатку

Нижче представлені блок-схеми функціонування програмного коду, та макети сторінок мобільного додатка.

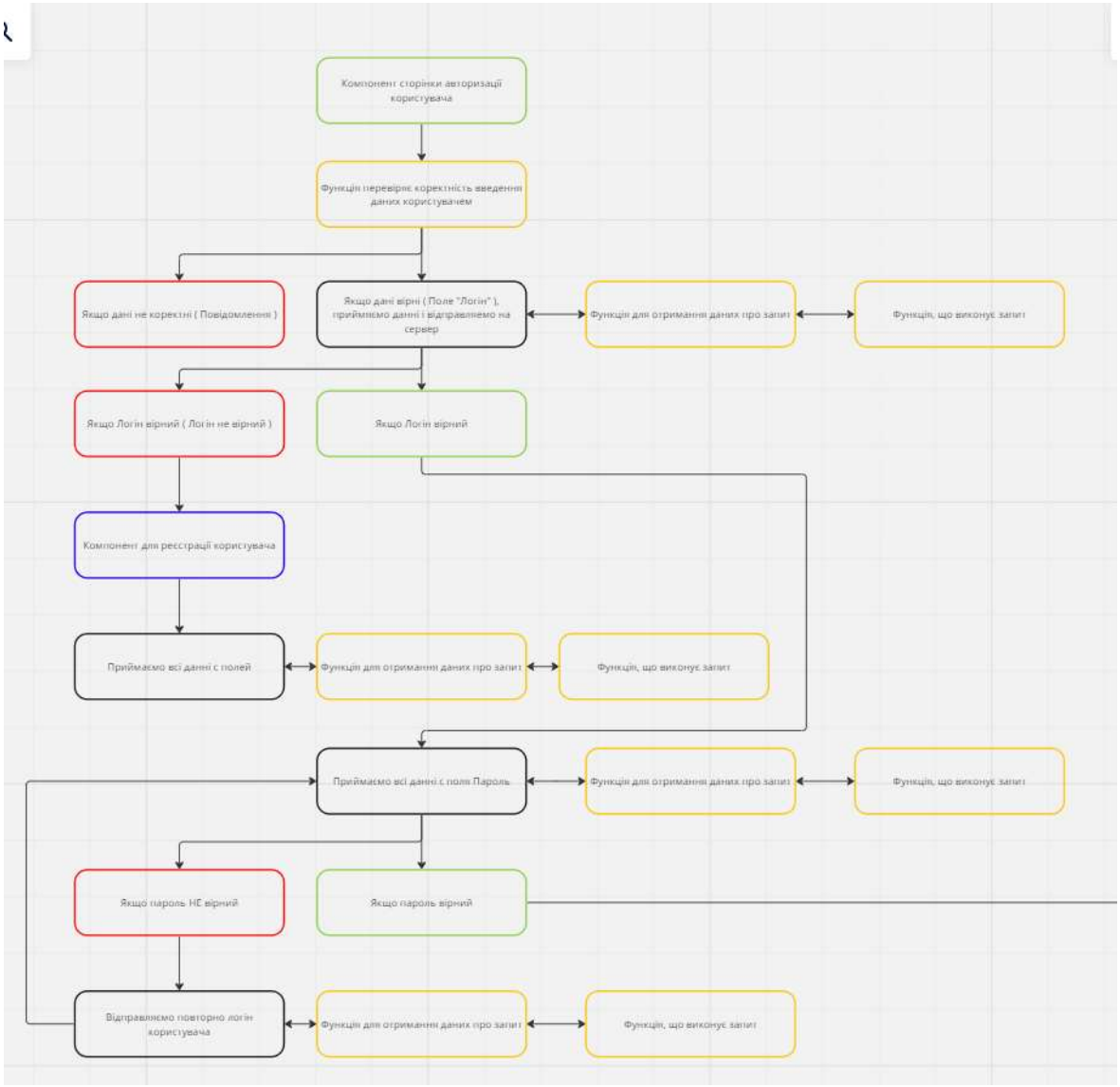


Рисунок 4.2 - Блок-схема компонента для авторизації користувача

Рисунок 4.3 - Макет головної сторінки мобільного додатка

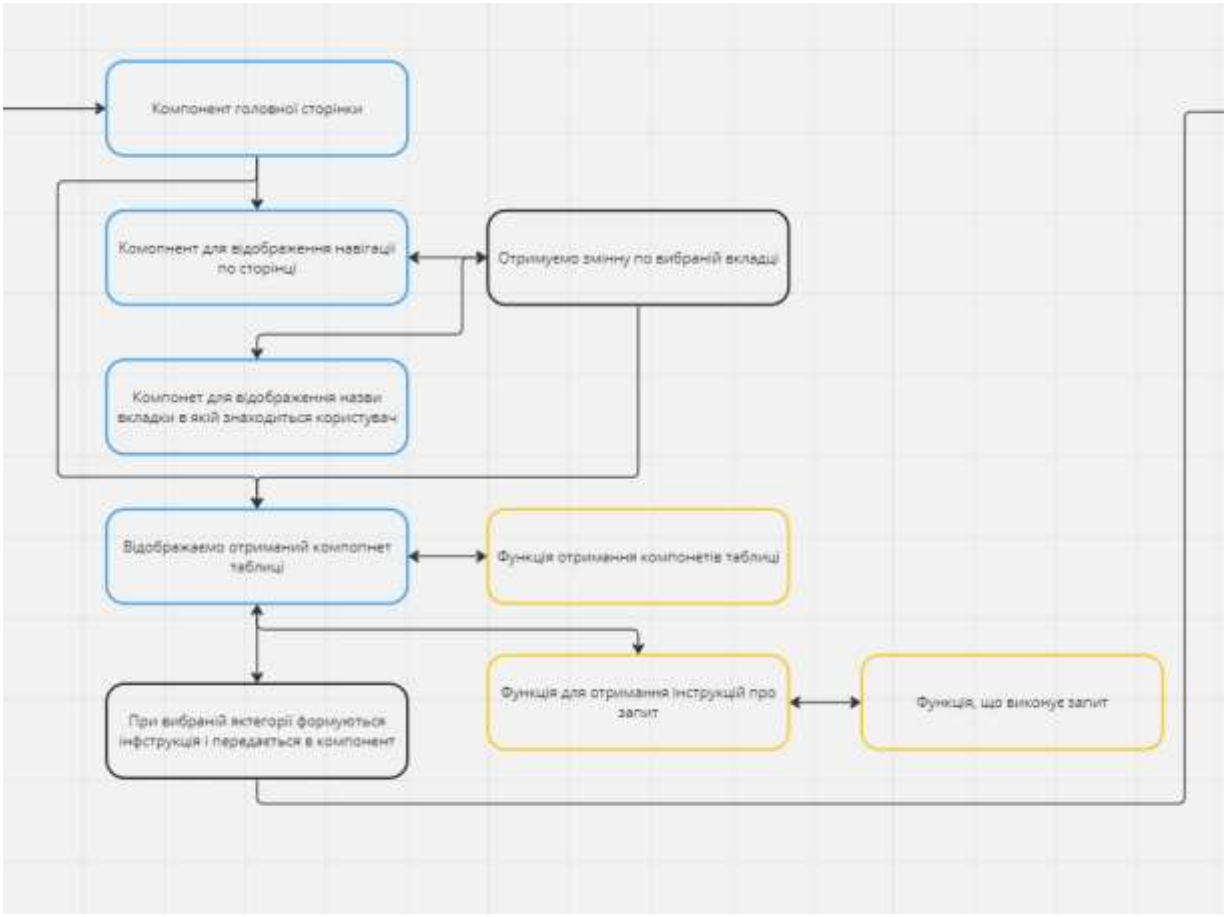


Рисунок 4.4 - Блок-схема компонента для відображення таблиць перелік батьківської категорії, категорії, підкатегорії, товару

Рисунок 4.5 - Макет головної сторінки мобільного додатка

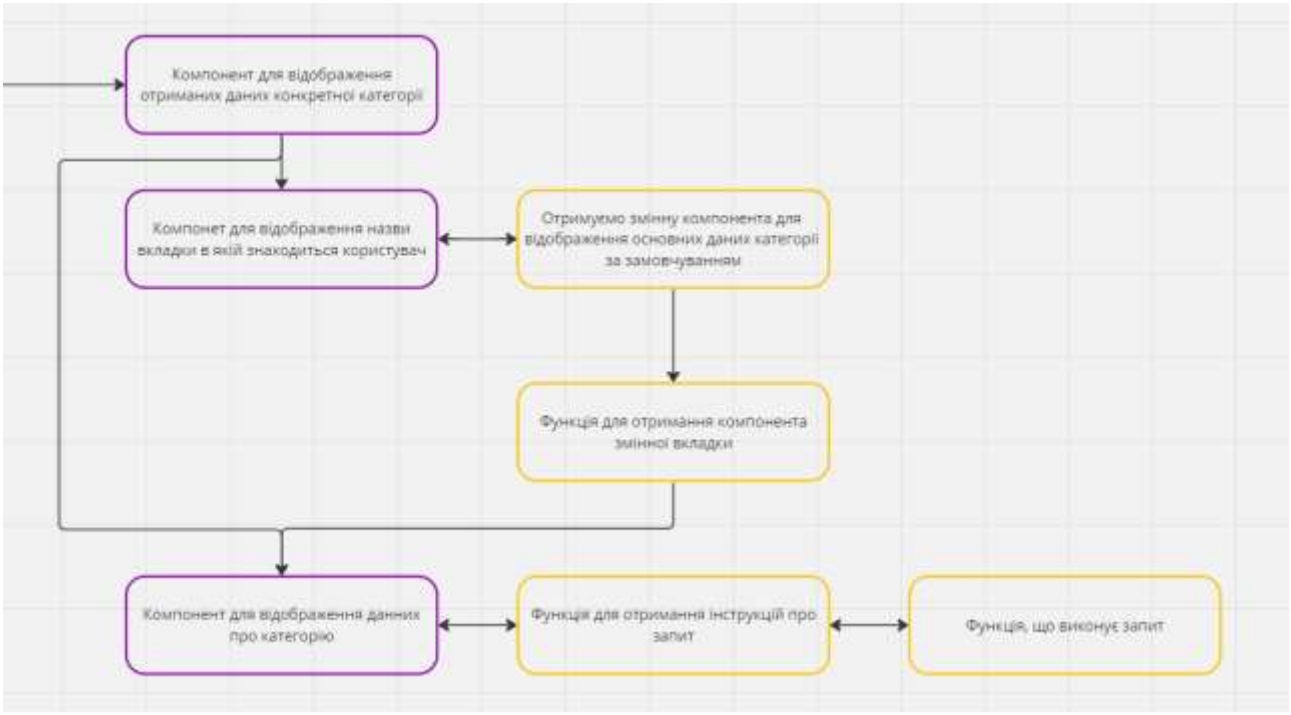


Рисунок 4.6 - Блок-схема компонента для відображення повної інформації батьківської категорії, категорії, підкатегорії, товару

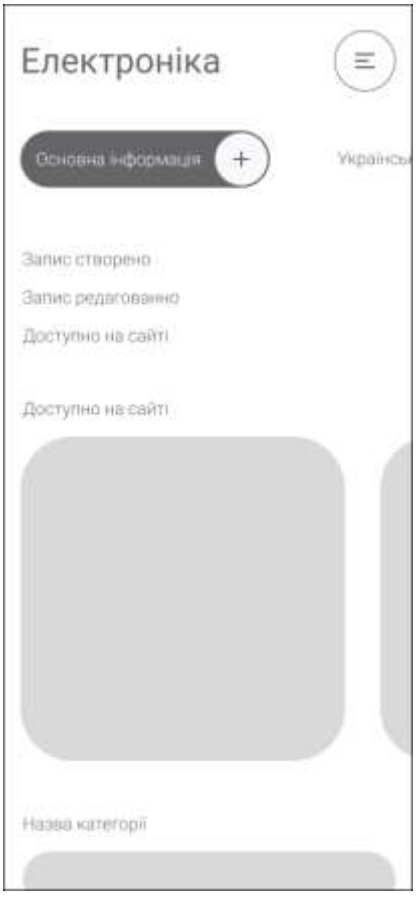


Рисунок 4.7 - Макет сторінки з детальною інформацією

4.6 Обґрунтування вибору фреймворку React для веб-сайту та React Native для CMS мобільного додатку.

Обґрунтування вибору фреймворку React для веб-сайту та React Native для CMS мобільного додатка ґрунтується на численних технічних, ефективності та стратегічних перевагах кожного фреймворку, що відповідає конкретним потребам та цілям проекту.

- React для веб-сайту:

Компонентний підхід: React визначається компонентним підходом, що дозволяє легко розділяти інтерфейс на невеликі та перевикористовувані компоненти. Це полегшує розробку, розширення та підтримку веб-сайту.

Висока продуктивність: Віртуальний DOM у React дозволяє зменшити кількість маніпуляцій з реальним DOM, що призводить до покращеної продуктивності та ефективності роботи веб-сайту, особливо при роботі з великою кількістю даних.

Розширені можливості: Існує широкий вибір сторонніх бібліотек та модулів, що легко інтегруються з React, що дозволяє забезпечити різноманітні функціональні можливості веб-сайту.

React Native для CMS мобільного додатка:

- Крос-платформена розробка:

React Native дозволяє розробляти мобільні додатки для обох платформ, iOS та Android, використовуючи один і той же код. Це зменшує витрати на розробку та утримання, оскільки необхідно писати код лише один раз.

- Висока продуктивність та швидка розробка:

React Native забезпечує швидку розробку за рахунок гармонійної інтеграції з відомими веб-технологіями, такими як React. Крім того, hot-reloading спрощує процес тестування та внесення змін.

- Широкі можливості CMS:

React Native може бути легко інтегрований з різними системами управління контентом (CMS), що дозволяє ефективно керувати вмістом мобільного додатка. Це особливо важливо для CMS, що підтримують багатомовність та велику кількість динамічних данних.

Обираючи React для веб-сайту та React Native для CMS мобільного додатка, ми максимізуємо переваги кожного фреймворку, забезпечуючи високоякісну та ефективну розробку для обох платформ.

4.7 Застосування Node.js як серверного фреймворку та його переваги у відношенні до потреб проекту.

Вибір серверної частини на Node.js ґрунтується на кількох ключових факторах, які визначають його ефективність, масштабованість та сумісність з вимогами проекту.

1) Асинхронність та події:

Node.js працює в асинхронному режимі, що дозволяє ефективно обробляти багатозадачні операції без блокування виконання коду. Це особливо корисно для серверних додатків, які часто взаємодіють з багатою кількістю запитів одночасно.

2) Швидкодія та продуктивність:

Використання JavaScript на обох боціх (клієнтському та серверному) дозволяє ефективно використовувати навички розробників та максимізувати повторне використання коду. Подібний стек технологій сприяє швидкій розробці та підтримці.

3) Велика спільнота та екосистема:

Node.js має велику та активну спільноту розробників, а також обширну екосистему модулів (за допомогою npm), що дозволяє легко і швидко вирішувати завдання розробки та інтеграції.

4) Масштабованість:

Node.js дозволяє ефективно масштабувати серверні додатки, особливо завдяки можливості створювати мікросервіси та використовувати архітектурні патерни для оптимізації продуктивності.

5) Широкі можливості для розробки API:

Node.js надає зручний інтерфейс для розробки API через використання бібліотек, таких як Express.js, які спрощують обробку маршрутів та керування запитами.

Файлова структура CMS додатка

```
control_innovations/
    .expo/
    .idea/
    assets/
        admin_image/
    applications/
        admin_component/
        admin_pages/
```

admin_systemFiles/

locales/

node_modules/

- control_innovations/

Папка "control_innovations" головна папка проекту

- .expo/

Папка ".expo/" містить конфігураційні файли та дані, пов'язані з використанням сервісу Expo у проекті. Вона забезпечує зручний спосіб налаштовувати та взаємодіяти з Expo-специфічними функціями додатка, такими як налаштування збірки та конфігурація додатка.

- .idea/

Папка ".idea/" вказує на використання IDE, ймовірно, IntelliJ IDEA, для розробки проекту. У цій папці зберігаються файли конфігурації та налаштувань проекту, які характеризують специфічні аспекти розробки, такі як налаштування відображення, параметри збірки тощо.

- assets/

Папка "assets/" призначена для зберігання різноманітних ресурсів та активів, таких як зображення, шрифти, аудіо- та відеофайли, які використовуються в додатку. Вона дозволяє організувати та управляти ресурсами додатка. "admin_image/" для зображень.

- applications/

Папка "applications/" містить додатки або компоненти, що входять у структуру проекту. Наприклад, може бути розділена на "admin_component/" для адміністративних компонентів, "admin_pages/" для сторінок адміністратора та "admin_systemFiles/" для системних файлів.

- locales/

Папка "locales/" містить файлову структуру для локалізації додатка. Тут зазвичай розміщуються ресурси для різних мов, дозволяючи створювати мультилінгвальні додатки.

- node_modules/

Папка "node_modules/" містить залежності та бібліотеки, які використовуються в проекті. Node.js використовує цю папку для зберігання встановлених пакетів.

Файлова структура сайта

control_innovations/

node_modules/
public/
src/
site_components/
site_pages/
site_image/
site_systemFile/
locales/

- control_innovations/

Папка "control_innovations" головна папка проекту

- node_modules/

Папка "node_modules/" містить залежності та бібліотеки, які використовуються у проекті. Це стандартна практика для Node.js-проектів, де всі залежності встановлюються в цю папку.

- public/

Папка "public/" може бути використана для зберігання ресурсів, які не потрібно обробляти або компілювати під час робочого процесу, таких як HTML-файли, статичні зображення або інші публічні ресурси.

- src/

Папка "src/" є основним каталогом вихідного коду проекту. Вона містить всі файли, які ви створюєте або редагуєте під час розробки.

site_components/ Підпапка "site_components/" містить компоненти, які використовуються для побудови сторінок веб-сайту. Це може включати заголовки, кнопки, форми та інші переиспользуемі елементи.

site_pages/ Папка "site_pages/" може включати файли, які відповідають за реалізацію конкретних сторінок веб-сайту.

site_image/ Папка "site_image/" призначена для зберігання зображень та графічних ресурсів, які використовуються на сторінках веб-сайту.

site_systemFile/ Папка "site_systemFile/" може містити системні файли, такі як файли конфігурації чи інші системні ресурси.

locales/ Папка "locales/" може містити файли, пов'язані з локалізацією веб-сайту, що дозволяє створювати мультилінгвальні веб-додатки.

Файлова структура сервера

server/

databasesModels/

function/
images/
parentCategory/
category/
subcategory/
product/
node_modules/
telegram_bot/
systemFiles/

- databasesModels/

Папка "databasesModels/" містить моделі баз даних або файли, які визначають структуру та взаємозв'язок даних для використання у серверній частині додатка. Це може включати файли, які відображають таблиці баз даних та взаємодіють із системою керування базами даних.

- function/

Папка "function/" містить файли, які відповідають за функціональність серверної частини додатка. Це може включати обробники запитів, бізнес-логіку, аутентифікацію, авторизацію та інші компоненти, що виконують функції серверу.

- images/

Папка "images/" може бути використана для зберігання зображень, які можуть використовуватися або оброблятися сервером.

parentCategory/ Папка "parentCategory/" може включати файли або моделі, які стосуються батьківських категорій товарів чи послуг у системі.

category/ Папка "category/" містить файли, які стосуються категорій товарів чи послуг.

subcategory/ Папка "subcategory/" може містити файли або моделі для підкатегорій товарів чи послуг.

product/ Папка "product/" включає файли або моделі, що стосуються конкретних товарів чи послуг.

- node_modules/

Папка "node_modules/" містить всі залежності та бібліотеки, які використовуються у серверній частині проекту.

- telegram_bot/

Папка "telegram_bot/" може містити файли, пов'язані з інтеграцією з ботом у Telegram, якщо така інтеграція присутня у проекті.

systemFiles/ Папка "systemFiles/" може включати системні файли, такі як файли конфігурації, ключі для підключення до зовнішніх служб, або інші системні налаштування.

- Підтримка WebSocket:

Node.js має вбудовану підтримку WebSocket, що робить його ідеальним вибором для додатків, які вимагають реального часу та двостороннього зв'язку між клієнтом та сервером.

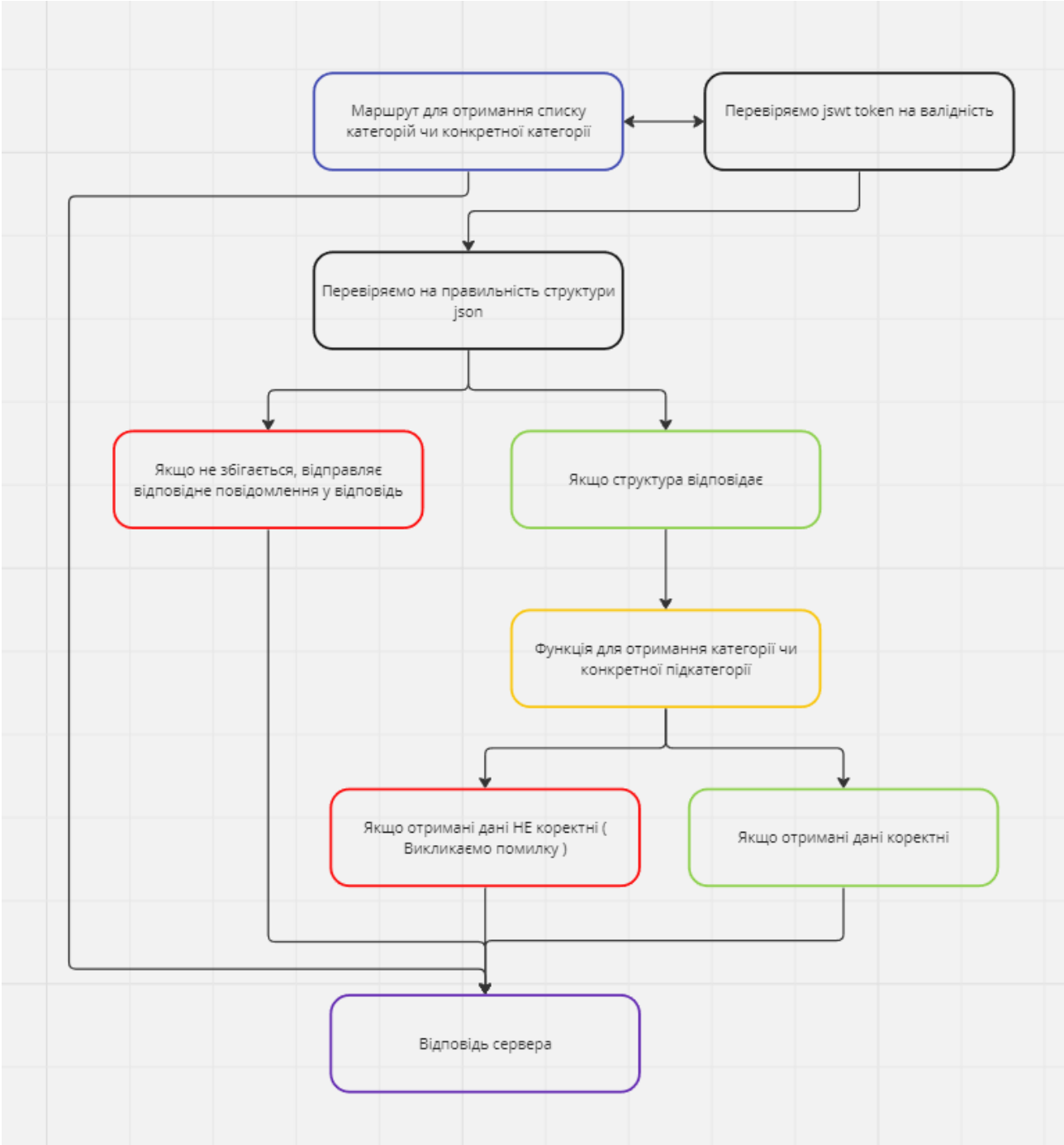


Рисунок 4.8 - Блок-схема роботи маршруту для отримання категорій, категорій, підкатегорій, товару

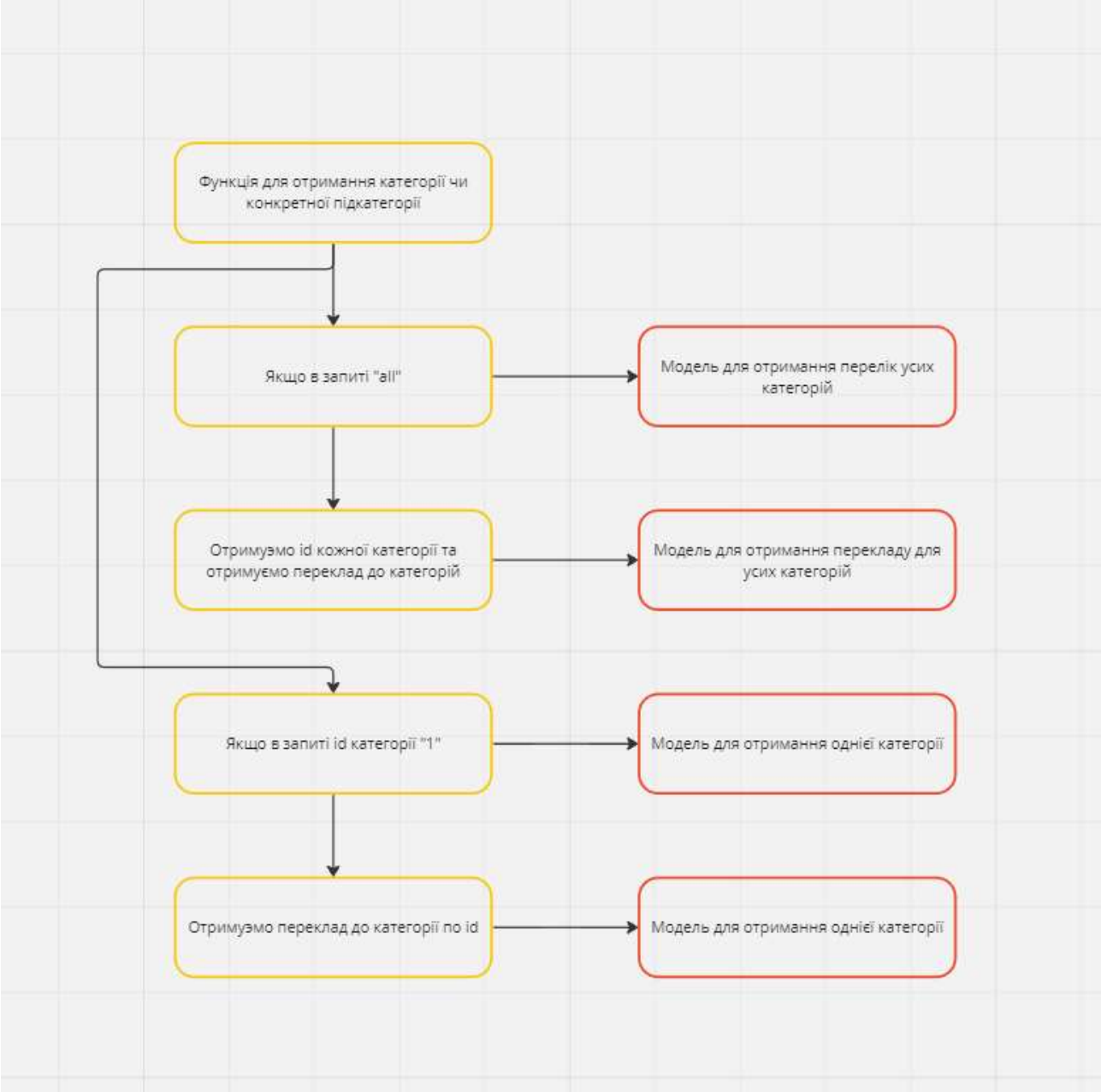


Рисунок 4.9 - Блок-схема для отримання усіх батьківських категорій, категорії, підкатегорії, товару або тільки конкретного запису

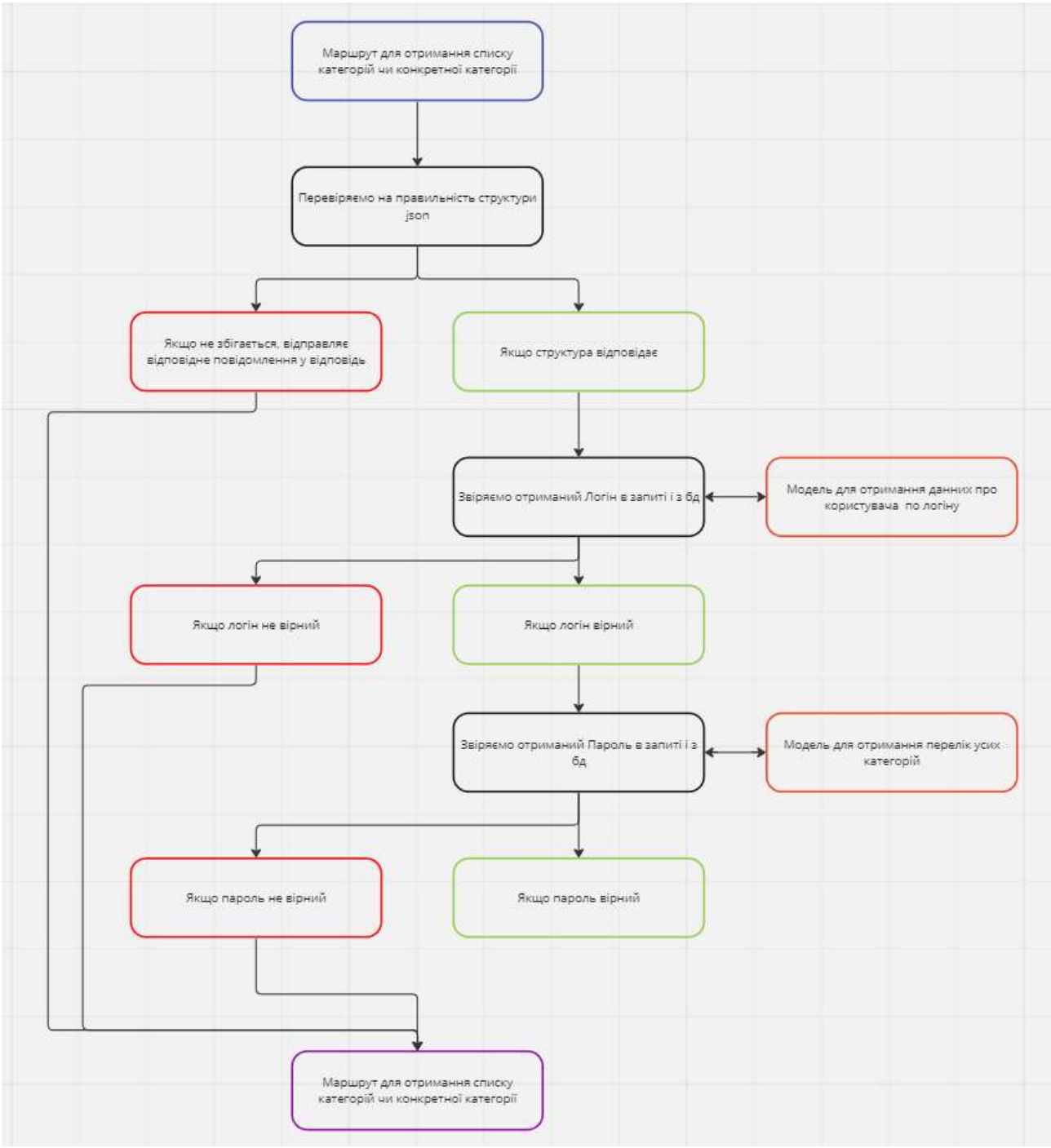


Рисунок 4.10 - Блок-схема Для авторизації користувача

4.8 Визначення принципів взаємодії між компонентами за допомогою REST API та JSON для забезпечення ефективної комунікації.

Визначення принципів взаємодії між компонентами за допомогою REST API та JSON для забезпечення ефективної комунікації є ключовим етапом в розробці систем, що вимагають високофункціональної та ефективної обміну даними. REST API (інтерфейс передачі стану представлення) визначає набір принципів архітектури, що базуються на простоті та легкості використання.

Одним із ключових аспектів є використання JSON (формат обміну даними в текстовій формі) для представлення та передачі інформації між компонентами. JSON дозволяє структурувати дані в легкій для розуміння формі, що робить його ідеальним для використання в комунікаційних цілях. Крім того, він легко читається як людьми, так і комп'ютерами, що сприяє зручності в процесі взаємодії.

REST API використовується для визначення стандартів обміну даними та комунікації між серверами та клієнтами. Він враховує обмеження та принципи, які спрощують розробку, розгортання та підтримку систем. Зокрема, використання HTTP-методів (GET, POST, PUT, DELETE) дозволяє ефективно взаємодіяти з ресурсами та виконувати різноманітні операції.

При використанні REST API та JSON відбувається ефективна передача даних, забезпечуючи легкість розширення та сумісності між різними компонентами системи. Ця архітектурна модель робить взаємодію між компонентами більш прозорою та дієвою, сприяючи розвитку масштабованих та надійних інформаційних систем.

- Маршрут та структура json для отримання батьківських категорій
  Маршрут « /admin/v1.1/parentCategory/receive »
  Метод « Post »
- Маршрут та структура json для створення батьківських категорій
  Маршрут « /admin/v1.1/parentCategory/creation »
  Метод « Post »
- Маршрут та структура json для оновлення батьківських категорій
  Маршрут « /admin/parentCategory/:category »
  Метод « Put »
- Маршрут та структура json для видалення батьківських категорій
  Маршрут « /admin/parentCategory/:language/:category »
  Метод « Delete »
  Структура: Відсутня
- Маршрут та структура json для отримання категорій
  Маршрут « /admin/v1.1/category/receive »
  Метод « Post »
- Маршрут та структура json для створення категорій
  Маршрут « /admin/v1.1/category/creation »
  Метод « Post »

- Маршрут та структура json для оновлення категорій
  Маршрут « /admin/category/:category »
  Метод « Put »
- Маршрут та структура json для видалення батьківських категорій
  Маршрут « /admin/category/:language/:category »
  Метод « Delete »
- Маршрут та структура json для отримання підкатегорії
  Маршрут « /admin/v1.1/subcategory/receive »
  Метод « Post »
- Маршрут та структура json для створення підкатегорій
  Маршрут « /admin/v1.1/subcategory/creation »
  Метод « Post »
- Маршрут та структура json для оновлення підкатегорій
  Маршрут « /admin/subcategory/:category »
  Метод « Put »
- Маршрут та структура json для видалення батьківських категорій
  Маршрут « /admin/subcategory/:language/:category »
  Метод « Delete »
- Маршрут та структура json для отримання батьківських категорій
  Маршрут « /admin/v1.1/product/receive »
  Метод « Post »
- Маршрут та структура json для створення підкатегорій
  Маршрут « /admin/v1.1/product/creation »
  Метод « Post »
- Маршрут та структура json для оновлення підкатегорій
  Маршрут « /admin/product/:category »
  Метод « Put »
- Маршрут та структура json для видалення батьківських категорій
  Маршрут « /admin/product/:language/:category »
  Метод « Delete »
  Структура: Відсутня

4.9 Висновок за розділом

В ході проведення конструкторської частини розглянуто широкий спектр завдань, пов'язаних з розробкою термостата для кондиціювання офісних

приміщень. У даному висновку наведено загальний підсумок роботи за кожним із визначених етапів. На основі розробки CMS мобільного додатка для виробництва термостатів для кондиціювання офісних приміщень, визначено принципи взаємодії компонентів за допомогою REST API та JSON. Це дозволяє ефективно та стандартизовано обмінюватися даними між сервером та мобільним додатком, забезпечуючи ефективну комунікацію. Використання Node.js обрано для серверної частини з урахуванням його асинхронного та подієвого підходу, що відповідає вимогам проекту. Це забезпечує оптимальну обробку великої кількості одночасних запитів та ефективне управління даними.

Для мобільного додатка обрані фреймворки React та React Native з метою перевикористання коду, швидкості розробки та високої продуктивності. React дозволяє розробляти ефективний веб-інтерфейс, в той час як React Native забезпечує можливість швидкої розробки мобільних додатків для різних платформ. Огляд мікросервісної архітектури виявився раціональним для побудови системи. Застосування мікросервісів дозволяє розділити функціональність на невеликі та незалежні сервіси, спрощуючи розробку, розгортання та масштабування.

Аналізуючи особливості виробництва термостатів та вимоги до веб-сайту, було визначено унікальні функціональні та дизайнерські особливості, які повинна підтримувати CMS. Це включає в себе можливість швидкого внесення змін у вміст сайту, інтеграцію з системами аналітики, та забезпечення зручного інтерфейсу для користувачів.

Вибір технологій для розробки веб-сайту та мобільного додатку відбувався на підставі вимог до швидкості реалізації, масштабованості та забезпечення безпеки. Враховано сучасні технічні стандарти, що дозволяють ефективно впроваджувати функціональність та забезпечувати високу продуктивність.

Під час аналізу існуючих CMS виявлено їхні недоліки, такі як обмеження у кастомізації, високий рівень складності для неініційованих користувачів та відсутність певних функціональностей, які є критичними для веб-сайту та мобільного додатку для виробництва термостатів.

5. ДОСЛІДНИЦЬКА ЧАСТИНА

5.1 Задачі і методи проведення досліджень

У цьому розділі визначено програму проведення наукових досліджень для створення мобільного додатка CMS, призначеного для виробництва термостатів для кондиціювання офісних приміщень. Встановлені ключові кроки та етапи виконання досліджень, методології вибору, збору необхідної інформації, а також аналізу та інтерпретації отриманих результатів. У процесі досліджень були задіяні необхідні ресурси, спрямовані на вдосконалення якості та продуктивності мобільного додатка та його серверної частини.

Задачі:

- Оптимізація швидкодії та реакції:

Мета: Визначення оптимальних параметрів продуктивності для максимально швидкого відгуку мобільного додатка та його серверної частини.

Методологія: Впровадження тестів швидкодії та аналізу коду для досягнення оптимальних результатів.

- Надійність та виявлення помилок:

Мета: Забезпечення надійності роботи мобільного додатка та сервера, виявлення та обробка можливих помилок.

Методологія: Проведення тестів на стійкість та розробка механізмів автоматичного виявлення та логування помилок.

- Перевірка даних користувача:

Мета: Забезпечення коректної обробки та перевірки введених користувачем даних.

Методологія: Використання алгоритмів перевірки валідності даних та піддавання випробуванням з різних точок зору введення.

- Захист від несанкціонованого доступу:

Мета: Забезпечення високого рівня безпеки для уникнення несанкціонованого доступу до системи.

Методологія: Впровадження алгоритмів шифрування та аутентифікації, аналіз можливих точок вразливості системи.

- Тестування серверної частини:

Мета: Оцінка ефективності серверної частини для забезпечення відповіді на запити мобільного додатка.

Методологія: Проведення тестів навантаження, визначення об'ємів обробки даних та їх оптимального розподілу.

- Забезпечення сумісності:

Мета: Визначення та врахування сумісності мобільного додатка з різними операційними системами та версіями.

Методологія: Тестування на різних платформах, аналіз особливостей взаємодії з різними ОС.

5.2 Програма проведення досліджень

Для ретельного аналізу ефективності та надійності серверної частини мобільного додатка CMS використовувалась програма Postman показано на рис 5.1, що дозволяє здійснювати HTTP-запити на сервер. Процес дослідження включав в себе виконання запитів на сервер, в тому числі запитів, які містять помилки, з метою оцінки здатності сервера вірно обробляти та повідомляти про виниклі неполадки. Додатково проводились вимірювання часу відповіді сервера для визначення його продуктивності.

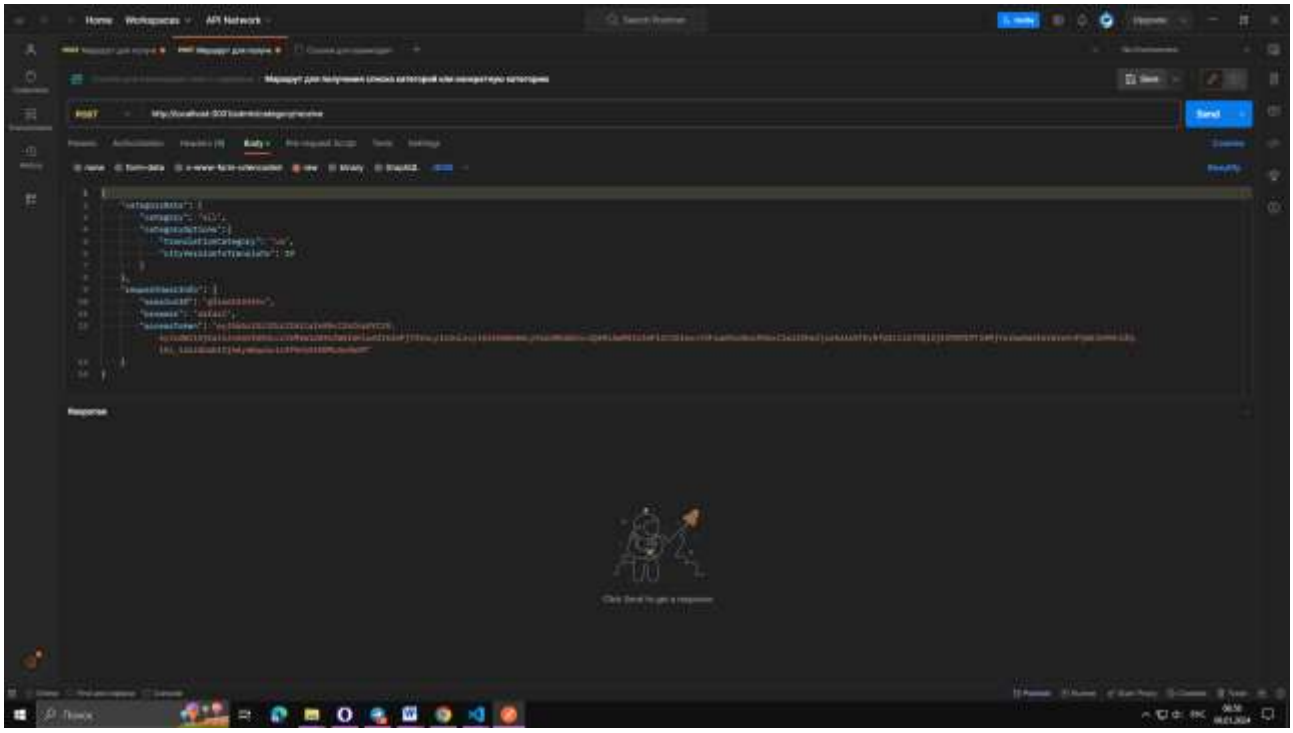


Рисунок 5.1 - Головний екран програми Postman

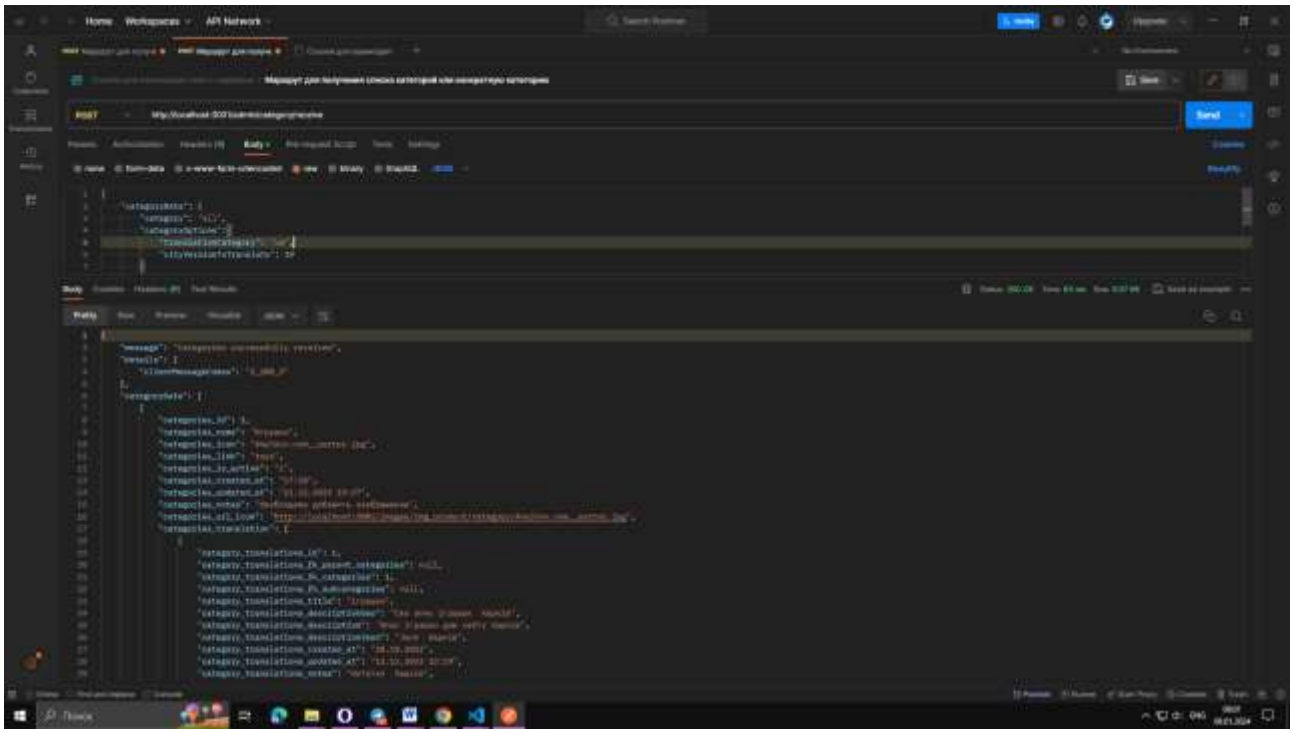


Рисунок 5.2 - Виконання запроса для отримання категорій

Запити на сервер з помилками:

В ході досліджень системи виконувалися запити на сервер, які спеціально містять помилки. Це було важливим етапом для визначення та аналізу реакції сервера на некоректні запити. Цей підхід дозволяв визначити рівень толерантності сервера до помилок та встановлювати, чи надійно сервер обробляє непередбачувані сценарії.

Рисунок 5.3 - Виконня запросу з помилкою

Вимірювання часу відповіді сервера:

Однією з ключових мет дослідження було встановлення часу, за який сервер готовий обробляти та відповідати на різноманітні запити зображенно на рис 5.4. Це включало в себе аналіз часу відправки запиту та часу отримання відповіді. Вимірювання виконувалися з різних точок, а також за умов різного завантаження сервера, щоб отримати повний обсяг даних про продуктивність.

Загальні результати цих досліджень надали цінні відомості про ефективність та стійкість серверної частини мобільного додатка CMS. Отримані дані стануть основою для подальших вдосконалень та оптимізації роботи сервера з метою надання високоякісного та надійного сервісу для користувачів.

Рисунок 5.4 - Час за який сервер оброблює запит

5.3 Результаты досліджень

Дослідження ефективності та надійності серверної частини мобільного додатка CMS виявили вражаючі результати, свідченням чого стали детальні аналізи різних аспектів роботи системи.

- Обробка запитів:

Сервер успішно обробляє всі запити для отримання необхідних даних. Важливою характеристикою є безперебійність обробки запитів, що підтверджує стабільність системи при роботі з великим обсягом даних. Така надійність є ключовим аспектом для забезпечення позитивного користувацького досвіду.

- Час відповіді сервера:

Середній час відповіді сервера становить лише 30 мілісекунд, що свідчить про високу швидкість та ефективність обслуговування запитів. Цей результат є високим і відповідає сучасним стандартам для серверних додатків, забезпечуючи миттєву відповідь на запити користувачів.

- Час завантаження додатка:

Середній час завантаження мобільного додатка становить приблизно 5 секунд. Це показник, який знаходиться в межах прийнятних стандартів для користувацької зручності, але може бути предметом подальшої оптимізації для зменшення часу очікування користувача.

- Обробка помилок:

Система ефективно впоралася з ситуаціями, коли на сервер подавалися запити з помилками. Сервер правильно визначав помилки, відповідно обробляв

їх та надсилав відповідні повідомлення про помилки користувачам. Це важливо для забезпечення коректної та інформативної взаємодії з користувачами.

- Статуси відповідей:

Система надавала вірні статуси відповідей відповідно до результатів обробки запитів. У випадку успішної обробки відправлявся статус 200, при обробці помилки - статус 400, а в разі невідомої помилки - статус 500. Така консистентність статусів сприяє якості та надійності взаємодії між клієнтською та серверною частинами системи.

Усі ці результати свідчать про високий рівень оптимізації та ефективності серверної частини мобільного додатка CMS, що становить важливий крок у забезпеченні високоякісного функціонування та задоволення потреб користувачів.

5.4 Аналіз результатів досліджень

Аналіз даних показав, що сервер успішно обробляє всі запити користувачів для отримання необхідних даних. Це свідчить про високий рівень ефективності системи, що є ключовим аспектом для забезпечення задоволення користувачів та надання їм необхідної інформації. Середній час відповіді сервера в 30 мілісекунд є вражаючим показником, вказуючи на високий рівень продуктивності серверної інфраструктури. Це робить систему конкурентоспроможною та здатною обслуговувати значну кількість користувачів одночасно. Середній час завантаження мобільного додатка залишається прийнятним для більшості користувачів, проте є можливість подальшої оптимізації для зменшення часу очікування. Це важливо для покращення загального користувацького досвіду та збільшення залученості користувачів. Система виявилася досить стійкою до ситуацій, коли на сервер подавалися запити з помилками. Забезпечення вірних статусів відповідей у випадку успішної обробки або виникнення помилки підвищує рівень коректності та передбачуваності для користувачів.

5.5 Висновки за розділом

У результаті проведених досліджень, спрямованих на аналіз технічних вимог, визначення функціональностей та оцінку продуктивності мобільного додатка.

Детальний аналіз технічних вимог до системи дозволив визначити ключові параметри ефективності, надійності та безпеки додатка. Цей етап є

фундаментальним для подальшого успішного розвитку проекту та забезпечення високих стандартів його функціонування.

Застосування методів, таких як аналіз технічних вимог, визначення функціональностей та вивчення продуктивності серверної частини, дозволило систематично підходити до вирішення завдань. В програмі досліджень чітко визначені кроки та етапи виконання робіт, що сприяє відповідному вивченню всіх аспектів системи.

Загальний аналіз результатів вказує на успішність розробленого додатка. Сервер успішно обробляє запити, забезпечуючи низький час відповіді. Обробка помилок і статуси відповідей вказують на високий рівень стійкості та коректності функціонування системи.

Перспективи та рекомендації:

Отримані результати свідчать про перспективний характер мобільного додатка та його готовність до впровадження. Однак для досягнення ще вищого стандарту рекомендується продовжувати дослідження та оптимізацію, зокрема, спрямовані на скорочення часу завантаження додатка та підвищення його інтерфейсної ефективності. Також слід розглядати можливість розширення функціоналу відповідно до змінних потреб користувачів.

6. ЕКСПЕРИМЕНТАЛЬНО-ПРАКТИЧНА ЧАСТИНА

6.1 Задачі і засоби виконання експериментальної розробки

Важливим етапом є оптимізація додатка для різних мобільних пристроїв. Це включає в себе адаптацію до різних розмірів екранів та забезпечення високої продуктивності на різних моделях смартфонів, враховуючи їхні технічні характеристики.

Важливим аспектом розробки є здатність додатка адаптуватися до різних розмірів екранів мобільних пристроїв. Це досягається за допомогою респонсивного дизайну, який автоматично адаптує інтерфейс користувача до конкретного розміру екрану. Така оптимізація забезпечує зручність використання додатка незалежно від типу мобільного пристрою.

Висока продуктивність і оптимізація для старих смартфонів:

Враховуючи різні технічні характеристики смартфонів, важливо забезпечити високу продуктивність додатка навіть на старіших моделях пристроїв. Це може бути досягнуто шляхом ефективного використання ресурсів, оптимізації алгоритмів та уникання надмірного навантаження на процесор та пам'ять пристрою.

Додаток розроблено відповідно до сучасних стандартів та критеріїв для мобільних додатків. Це включає в себе відповідність дизайну з сучасними трендами, дотримання принципів безпеки, швидкодію та ергономіку інтерфейсу. Також враховані вимоги щодо захисту даних та конфіденційності користувачів. Процес розробки не завершується з випуском першої версії додатка. Неперервне вдосконалення відбувається на основі відгуків користувачів, виявлених помилок та змін у вимогах ринку.

6.2 Опис лабораторної установки

У процесі розробки мобільного використовувався домашній комп'ютер, обладнаний операційною системою Windows 10. Обрана операційна система надає широкі можливості для розробки та тестування програмного забезпечення.

- Postman для Виконання Запитів на Сервер: Під час розробки мобільного додатка використовувався інструментарій, що включав в себе програму Postman для виконання запитів на сервер та проведення тестових сценаріїв. Postman надавав можливість взаємодії з сервером, систематизації запитів, аналізу структури відповідей сервера та

ефективного виявлення та виправлення можливих помилок обробки запитів.

- Visual Studio Code для Написання Коду: Процес написання програмного коду виконувався у середовищі Visual Studio Code - інтегрованому розробничому середовищі, яке відзначається зручністю використання та підтримкою різних мов програмування. Застосунок надавав широкий набір інструментів, що полегшував розробку та забезпечував ефективний контроль над кодом.
- Figma для Розробки Дизайну: Для створення естетичного та функціонального дизайну мобільного додатка використовувався графічний інструмент Figma. За допомогою Figma здійснювалася розробка прототипів, створення дизайну елементів і візуалізація концепцій. Використання цього інструменту дозволило створювати ефективний та привабливий інтерфейс.
- Miro для Побудови Блок-Схем: Процес планування та проектування мобільного додатка здійснювався за допомогою інструменту Miro, який відзначається можливістю спільної роботи та побудови блок-схем. Використання Miro сприяло створенню чіткої логіки додатка, покращенню взаєморозуміння в команді та забезпечило ефективний візуальний інструментарій для побудови структури проекту.
- SQLiteStudio для Управління Базою Даних: У роботі над мобільним додатком використовувалася програма SQLiteStudio для зручного та ефективного управління базою даних. SQLiteStudio є інструментом для розробки та адміністрування баз даних SQLite, що відзначається простотою використання та багатофункціональністю. Програма надає можливість створювати та редагувати схеми баз даних, визначати таблиці, їхні зв'язки та поля. Це дозволяє зручно визначати структуру даних та забезпечувати їхню цілісність. SQLiteStudio відмінно підтримує виконання SQL-запитів безпосередньо з інтерфейсу. Це дозволяє взаємодіяти з базою даних, виконуючи запити для отримання, модифікації та видалення даних, а також створення збережених процедур. SQLiteStudio надає зручні інструменти для відслідковування та аналізу даних у базі. Завдяки графічному інтерфейсу, користувач може швидко переглядати вміст таблиць, фільтрувати дані та виконувати запити для здійснення необхідних вибірок.

6.3 Характеристика розробленого програмного забезпечення

Це мобільне додаток виступає в ролі ефективного інструмента для редагування вмісту веб-сайту, при цьому відмінно справляється з завданням навіть без значних навичок у програмуванні. Він створений з урахуванням потреб користувача, який має можливість оперативно та без зусиль вносити зміни в зовнішній вигляд та функціонал свого веб-ресурсу.

Завдяки цьому додатку, процес редагування веб-сайту стає максимально простим та зрозумілим. Його інтуїтивний інтерфейс дозволяє користувачам легко орієнтуватися та оперативно внести необхідні зміни, навіть якщо вони не мають глибоких знань у програмуванні. Цей підхід демократизує процес управління веб-сайтом, відкриваючи його для широкого кола користувачів. Тепер кожен, хто бажає внести зміни у веб-простір, може легко та швидко це робити, не витрачаючи великої кількості часу або ресурсів на програмування.

Мобільний додаток надає розширені можливості управління базою даних, забезпечуючи користувачам широкий функціонал для роботи з інформацією. З його допомогою можна ефективно взаємодіяти з базою даних, використовуючи такі опції як отримання всіх доступних записів, створення нових елементів, редагування, видалення родинних категорій, категорій, подкатегорій, товарів, описів товарів та характеристик товарів.

Унікальна можливість додатку полягає в його здатності надавати великий функціонал для редагування та управління різноманітними аспектами даних у базі. Користувачі можуть зручно керувати товарами, їх характеристиками та відносинами між категоріями. Крім того, додаток включає в себе функціонал авторизації та реєстрації, надаючи можливість користувачам зручно управляти своїм аккаунтом та визначати рівень доступу до різних функцій системи.

Загальний функціонал додатку створений таким чином, щоб максимально спростити та розширити можливості взаємодії з базою даних, що робить його незамінним інструментом для ефективного управління інформацією та ресурсами.

Під час розробки приділялося велике значення аспектам мікроархітектури для забезпечення подальшого успішного масштабування мобільного додатка. У фокусі уваги були такі важливі елементи, як оптимізована структура системи та дбайливе проектування архітектурних компонентів.

Використання фреймворків виявилося невід'ємним чинником у реалізації цих концепцій. Дозволяючи швидко та ефективно розробляти різні частини

додатка, фреймворки створили сприятливе середовище для вдосконалення мікроархітектурних рішень. Це забезпечило високий рівень гнучкості та легкість у впровадженні необхідних змін та розширень. Такий підхід до розробки мобільного додатка не лише забезпечив його поточну ефективність, але і створив надійну основу для подальшого росту та масштабування. Розглядаючи майбутнє, можна впевнено стверджувати, що розробка зосереджена на стійкості та легкості розширення, завдяки вдосконаленим мікроархітектурним рішенням та використанню передових фреймворків.

6.4 Аналіз результатів проведення експериментів і впровадження розробки

Проведений аналіз результатів розробки мобільного додатка свідчить про високий рівень ефективності та відповідність поставленим завданням. Взяті показники продуктивності та відгуки користувачів підтверджують оптимальне функціонування додатка. Середня швидкість відгуку сервера в 30 мілісекунд та час завантаження додатка в 5 секунд вказують на його високу продуктивність та ефективність в роботі.

Застосування мікросервісної архітектури у поєднанні з вибраними технологіями (Node.js, React, React Native) забезпечило не лише високий рівень ефективності, а й гнучкість та можливість швидкого розширення системи. Використання стандартів комунікації (REST API та JSON) дозволяє забезпечити стандартизовану та ефективну взаємодію між сервером та мобільним додатком.

Процес впровадження розробленого мобільного додатка в реальне виробниче середовище планується відбуватися систематично та з урахуванням всіх необхідних аспектів. Враховуючи його велику ефективність та готовність до розширення, впровадження планується проводити відповідно до ретельно розробленого плану.

Важливим етапом буде тестування додатка в реальних умовах залученням обмеженої аудиторії користувачів для збору додаткових відгуків та виявлення можливих несправностей. Поступове впровадження за допомогою пілотних проектів дозволить ефективно вирішувати виникаючі завдання та забезпечити плавний перехід до повномасштабного використання додатка в реальному виробничому середовищі.

6.5 Висновки за розділом

Завданням було точно визначити та чітко сформулювати ті задачі, які вимагало вирішення під час проведення експерименту. Метою нашого дослідження було систематично досліджувати ефективність та функціональність мобільного додатка в умовах його реального використання.

Ключовим аспектом завдань експерименту було визначення рівня відповідності функціоналу додатка та його продуктивності до висунутих технічних та функціональних вимог. Також, важливо було оцінити ефективність обраного підходу до розробки та його здатність відповідати реальним потребам користувачів.

Дослідження зосереджувалося на ретельному аналізі результатів, отриманих під час проведення експерименту, і визначенні їх впливу на загальну ефективність додатка. Крім того, метою було також визначити можливі області для подальшого вдосконалення та оптимізації.

В ході дослідження враховувались питання стійкості та надійності функціоналу додатка під час інтенсивного використання в реальних умовах, щоб впевнитися у відповідності його роботи навіть у вимогливих сценаріях використання.

Отже, експериментальна частина нашого дослідження має на меті висвітлити результати та висновки, що стосуються ефективності та функціональності мобільного додатка, використовуючи підходи, які забезпечують його успішне функціонування в реальних умовах.

7 ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ

7.1 Опис об’єкта і практичних результатів дослідження

У контексті сучасного офісного управління, термоконтролер виступає важливим засобом для забезпечення комфорту працівників та підтримання оптимальних умов праці. Однією з ключових функцій термоконтролера є регулювання температурного режиму, що має різноманітні позитивні впливи на робоче середовище.

Основні Функції Термоконтролера:

Регулювання Температури: Термоконтролер дозволяє точно встановлювати та підтримувати задану температуру в офісних приміщеннях, сприяючи створенню комфортних умов для працівників.

Енергозбереження: Здатність оптимізувати витрати енергії через автоматичне вимикання чи регулювання системи кондиціювання враховує реальні потреби та розподіл робочого часу.

Управління Вентиляцією та Вологістю: Розширені термоконтролери можуть включати функції управління вентиляцією та регулювання вологості для забезпечення оптимального мікроклімату.

Програмовані Режими: Можливість програмованого налаштування температурних режимів, адаптованих до графіка робочого дня та періодів неактивності.

Один із ключових аспектів у розробці було глибоке розуміння функціональності та технічних параметрів термостатів. Це дозволило нам належним чином інтегрувати в наші рішення такі елементи, як регулювання температурного режиму, програмовані режими роботи, системи енергозбереження та інші ключові характеристики, які є особливими для даного класу пристроїв.

Крім того, ми зосередилися на створенні інтуїтивно зрозумілого та ергономічного інтерфейсу мобільного додатка, щоб забезпечити зручність використання та максимальний комфорт для користувачів. Враховуючи специфіку термостатів, ми додали у додаток функції дистанційного керування та моніторингу стану системи, щоб користувачі мали можливість зручно контролювати та налаштовувати терморегуляцію, навіть здалеку.

Окрім цього, важливою частиною нашої розробки стала уважна увага до безпеки та конфіденційності даних, оскільки термостати можуть включати важливу інформацію про температурні умови в приміщенні та зв'язані з цим дані. Наша серверна частина розроблена з використанням найсучасніших

методів шифрування та захисту даних, щоб забезпечити найвищий стандарт безпеки для наших користувачів.

7.2 Аналіз ринку

- WordPress

Завдяки своїй популярності, WordPress може бути об'єктом кібератак, тому безпека є важливим аспектом. WordPress – це не лише проста система управління контентом (CMS); це каталізатор глобальної трансформації в інтернет-просторі. Його найпопулярнішість визначається не лише кількістю веб-сайтів, які використовують цю платформу, але і його впливом на спосіб, яким мільйони користувачів у всьому світі взаємодіють з веб-ресурсами.

Статистика свідчить про те, що WordPress став невід'ємною частиною інтернет-екосистеми. За даними, він служить фундаментом для більшості веб-сайтів, охоплюючи широкий спектр від особистих блогів та невеликих підприємств до великих корпоративних порталів та новинарських сайтів.

В WordPress існує велика кількість плагінів, але їх якість не завжди висока через можливі помилки. ЦМС може працювати не коректно та видає помилки через неякісні плагіни.

Ця платформа пропонує не лише широкий функціонал та можливості для розширення, але і забезпечує великий ком'юнітет користувачів, який активно сприяє розвитку та удосконаленню WordPress. Завдяки активній спільноті та регулярним оновленням, WordPress залишається лідером у світі веб-розробки та контент-менеджменту.

- Joomla

Це визнана та популярна система управління контентом (CMS) в Інтернеті. Вона володіє гнучкістю, що дозволяє користувачам впроваджувати різноманітні функціональність та модулі, але ця гнучкість може призводити до зайвих налаштувань, які іноді можуть бути непотрібними. Із-за високого рівня гнучкості, Joomla має велику кількість параметрів налаштувань, інколи зайвих для звичайного користувача.

Що стосується безпеки, Joomla, подібно до інших популярних CMS, стає об'єктом кібератак через свою велику кількість веб-сайтів, які на ній базуються. Незважаючи на активні заходи забезпечення безпеки та постійні оновлення, її популярність робить її цільовою метою для зловмисників. Команда розробників, хоч і приділяє велику увагу захисту, повинна робити постійні вдосконалення, щоб утримувати велику спільноту користувачів в безпеці.

Важливо зауважити, що, хоча Joomla може викликати певні труднощі через свою гнучкість та кількість параметрів, вона залишається популярним інструментом для великої кількості веб-сайтів. Управління її налаштуваннями вимагає певного рівня експертизи, але, якщо вона належним чином налаштована та захищена, Joomla може надати потужний та гнучкий інструмент для розробки веб-проектів.

- Magento

Magento є визнаною системою управління контентом (CMS) із спеціалізацією на електронній комерції. Ця платформа особливо популярна серед великих та середніх інтернет-магазинів, оскільки надає розширені можливості для управління продажами, складом та іншими ключовими аспектами електронної комерції.

Однією з ключових переваг Magento є її спеціалізовані функції, створені саме для задоволення потреб електронної комерції. Зокрема, платформа пропонує інструменти для ефективного управління продажами, автоматизації складських процесів, аналізу покупців та багато іншого. Це робить Magento відмінним вибором для тих, хто прагне створити потужний та функціональний інтернет-магазин.

Спільнота Magento відіграє важливу роль у розвитку та підтримці цієї платформи. Інновації, регулярні оновлення та підтримка від розробників у спільноті сприяють стабільності та ефективності Magento. Взаємодія спільноти та підтримки компанії дозволяє користувачам отримувати переваги від найновіших технологічних рішень та забезпечує надійну роботу платформи.

Всі ці системи управління контентом (CMS) об'єднує кілька спільних проблем, серед яких застарілі технології та обмежена можливість масштабування. Зазвичай ці системи спрямовані на конкретний функціонал, і при додаванні нового функціоналу необхідно розробляти складні та вартісні рішення, що негативно впливає на продуктивність. Зокрема, проблеми із масштабуванням можуть призводити до обмежень у здатності системи відповідати зростаючим потребам користувачів.

Ще однією загальною проблемою є використання застарілих технологій, які можуть ускладнювати процес розробки та підтримки. Це може впливати на безпеку, продуктивність та загальну здатність системи пристосовуватися до нових вимог та ринкових трендів. Надто деякі з цих CMS можуть виявитися менш гнучкими у контексті впровадження нововведень, що може обмежувати їхню конкурентоспроможність на ринку.

Масштабування є важливим аспектом у сучасних умовах розвитку технологій, оскільки компанії і організації постійно зростають та змінюються. Наявність ефективного та швидкого масштабування може визначати успіх або невдачу в конкурентному середовищі. Тому необхідність у вдосконаленні систем управління контентом полягає в розробці більш гнучких та швидких рішень, які забезпечать найкращі практики в галузі та задовольнять потреби користувачів.

Підтримка також є ключовим фактором у виборі CMS. Наявність ефективної та оперативної підтримки дозволяє швидко вирішувати проблеми, вдосконалювати систему та надавати користувачам необхідну допомогу. Забезпечення постійної та якісної підтримки може визначати стабільність та довготривалість використання CMS у конкретному проєкті чи компанії.

7.3 Розрахунки відповідно до завдання консультанта з економічного розділу

Для підтримки проекту необхідні сервери, існують готові онлайн-рішення, але ціни можуть варіюватися. Середня вартість оренди сервера становить приблизно 800 гривень на місяць. Фреймворки безкоштовні та доступні для завантаження з Інтернету. Для додавання додатка до App Store для iOS вартість становить 3778 гривень на рік, а для розміщення в Google Play необхідно сплатити 954 гривні на рік.

Підтримка проекту включає в себе вартість інфраструктури та сервісів, які забезпечують необхідне середовище для розгортання та функціонування додатка. Оренда серверів є однією з ключових витрат, і ціни можуть різнитися в залежності від обраного постачальника послуг. Готові онлайн-рішення дозволяють швидко розпочати роботу, зменшуючи необхідність власного конфігурування та обслуговування серверів.

Безкоштовні фреймворки, доступні в Інтернеті, є важливим ресурсом для розробників, що дозволяє ефективно використовувати готові рішення та швидко створювати додатки. Це сприяє економії часу та ресурсів на стадії розробки.

Вартість розміщення додатка в магазинах додатків (App Store для iOS та Google Play) є іншою складовою витрат. Додаткові витрати пов'язані із сертифікацією та включенням додатка в офіційні магазини, що забезпечує йому широкий охоплення користувачів.

Загальна вартість підтримки проекту може варіюватися відповідно до конкретних вимог та обраного інструментарію, проте ці фінансові витрати є важливим аспектом забезпечення стабільності та успішної реалізації проекту.

Таблиця 7.1 - Витрат на підтримку мобільного додатку

Витрати	Ціна	Термін
Хостинг	800 грн	На місяць
App Store	3777 грн	На рік
Google Play	954	На рік
Усього витрат		5531

7.4 Висновки за розділом

У данному розділі роботи проведено економічне обгрунтування розробки мобільного додатка CMS для управління системою термоконтролю. Визначено, що підтримка проекту передбачає необхідність інфраструктури та послуг для розгортання, що може бути забезпечено за допомогою оренди серверів. Різноманітні готові онлайн-рішення та безкоштовні фреймворки визначено як важливі елементи для ефективного розвитку додатка.

Вартість оренди сервера становить приблизно 800 гривень на місяць, але ціни можуть варіюватися в залежності від конкретного постачальника послуг. Готові онлайн-рішення та безкоштовні фреймворки дозволяють знизити витрати на розробку та розгортання програмного забезпечення. Вартість розміщення додатка в магазинах додатків визначена як додатковий фактор, що слід враховувати.

Враховуючи всі витрати та важливі аспекти економічного планування, можна зробити висновок, що економічне обгрунтування розробки мобільного додатка є ретельною та збалансованою. Застосування готових рішень та розумне використання ресурсів може значно знизити витрати та підвищити ефективність розробки.

ЗАКЛЮЧЕННЯ

У висновку хочу відзначити, що розробка мобільного додатка для системи управління контентом (CMS) для підприємства, яке спеціалізується на виробництві термостатів для кондиціювання офісних приміщень, була важливим та успішним завданням. Проект спрямовувався на створення універсального та зручного у використанні мобільного додатка, який гармонійно взаємодіє з лінійкою термостатів підприємства, надаючи користувачам розширені можливості управління.

Проект успішно досягнув своїх основних цілей, включаючи розробку та впровадження функціонального мобільного додатка CMS, адаптованого до конкретних потреб підприємства. Додаток сприяє ефективному контролю та моніторингу налаштувань термостатів, покращуючи робоче середовище у офісах.

Застосовуючи передові технології, розробницька команда використовувала React Native для мобільного додатка, забезпечуючи його кросплатформенність та високу якість використання. Backend, реалізований на Node.js, гарантує потужність функцій на стороні сервера, а інтеграція REST API поліпшує взаємодію між мобільним додатком та сервером підприємства.

Дизайн інтерфейсу максимально спрямований на безперешкодний та інтуїтивний досвід. Додаток дозволяє користувачам легко керувати та налаштовувати параметри термостатів, моніторити дані в режимі реального часу та отримувати сповіщення, тим самим забезпечуючи розширені можливості контролю над кліматом в офісах.

Зважаючи на критичність контролю термостатів, безпека виявилася важливою умовою. Розробницька команда використовувала строгі заходи безпеки для захисту даних користувачів та забезпечення цілісності комунікації між мобільним додатком та термостатами.

У майбутньому передбачається можливість вдосконалення та додавання нових функцій. Потенційні покращення можуть включати додаткову інтеграцію зі смарт-офісними екосистемами, розширений аналіз даних для підвищення енергоефективності та підтримку новітніх технологій термостатів. Покращення та інновації роблять мобільний додаток CMS відмінним інструментом для управління термостатами в офісних приміщеннях. З урахуванням розвитку технологій та зростаючого попиту на енергоефективні рішення, цей проект визначає стандарт для майбутніх інновацій в галузі управління кліматом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 React Native Documentation – [Електронний ресурс] URL: https://reactnative.dev/docs/getting-started

1. Node.js Documentation – [Електронний ресурс] URL: https://nodejs.org/docs/latest/api/
2. Express.js Documentation – [Електронний ресурс] URL: https://expressjs.com/
3. REST API Tutorial – [Електронний ресурс] URL: https://www.restapitutorial.com/
4. React Native Tutorial – [Електронний ресурс] URL: https://www.restapitutorial.com/
5. Node.js Tutorial – [Електронний ресурс] URL: https://www.w3schools.com/nodejs/
6. Express.js Guide – [Електронний ресурс] URL: https://expressjs.com/2x/guide.html
7. RESTful API Design Best Practices – [Електронний ресурс] URL: https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/
8. Medium - React Native – [Електронний ресурс] URL: https://medium.com/@sebastienlorber/this-week-in-react-168-tanstack-usememo-rsc-preload-flushsync-shadcn-ui-mdx-storybook-97c47d09ff2c
9. Medium - Node.js – [Електронний ресурс] URL: https://medium.com/@abeythilakeudara3/nodejs-architecture-42a1d0efad8f
10. The Official Node.js Blog – [Електронний ресурс] URL: https://nodejs.org/en/blog
11. React Native GitHub Repository – [Електронний ресурс] URL: https://github.com/facebook/react-native
12. Express.js GitHub Repository – [Електронний ресурс] URL: https://github.com/expressjs
13. Example RESTful API Project – [Електронний ресурс] URL: https://github.com/hoppscotch/hoppscotch?tab=readme-ov-file
14. Eisenman, B. Learning React Native / Board Eisenman. – Addisson Wesley, 2020. – 580 p.
15. Casciaro, M. Node.js Design Patterns / Mario Casciaro. - Berlin, Heidelberg, 2017. - 306 p.

16. Hahn, E. Express in Action / Evan Hahn - Addisson Wesley Publishing Company, 2021. – 382 p.
17. Stack Overflow - React Native – [Електронний ресурс] URL: https://stackoverflow.com/questions/tagged/react-native
18. Stack Overflow - Node.js – [Електронний ресурс] URL: https://stackoverflow.com/questions/tagged/node.js?tab=Newest
19. GitHub Issues - React Native – [Електронний ресурс] URL: https://github.com/facebook/react-native/issues
20. GitHub Issues - Express.js – [Електронний ресурс] URL: https://github.com/expressjs/express/issues

ДОДАТОК А

Лістинг програми для роботи з категорією

Серверна частина:

- Робота з моделями категорії

```
class CategoryModel {
  static getModelCategoryAll() {
    return new Promise((resolve, reject) => {
      const sql = 'SELECT * FROM categories';
      db.all(sql, [], (err, categories) => {
        if (err) {
          console.error(err);
          reject(err);
        } else {
            resolve(categories);
          }
        }
      );
    });
  }
  static getModelCategoryName(category) {
    return new Promise((resolve, reject) => {
      const sql = 'SELECT * FROM categories WHERE categories_link
= ?';
      db.get(sql, [category], (err, categoryData) => {
        if (err) {
          console.error(err);
          reject(err);
        } else {
          resolve(categoryData);
        }
      });
    });
  }
  static getAdminCategoryByValue(categoryId) {
    return new Promise((resolve, reject) => {
      const sql = 'SELECT * FROM categories WHERE categories_id =
?';
      db.get(sql, [categoryId], (err, categoryId) => {
        if (err) {
          reject(err);
        } else {
          resolve(categoryId);
        }
      });
    });
  }

  static getGeneralMethodCategory(CategoryIdOrNameAll) {
    if (typeof CategoryIdOrNameAll === "number") {
      return this.getAdminCategoryByValue(CategoryIdOrNameAll);
    } else if (typeof CategoryIdOrNameAll === "string") {
```

```
    return this.getModelCategoryName(CategoryIdOrNameAll);
  } else {
    return this.getModelCategoryAll();
  }
}
static createAdminCategory(newCategoryData) {
  return new Promise((resolve, reject) => {
    const currentDateAndTime = getCurrentDateTime();
    const sql = `
      INSERT INTO categories (
        categories_fk_parentCategories,
        categories_name,
        categories_icon,
        categories_link,
        categories_is_active,
        categories_created_at,
        categories_notes
      ) VALUES (?, ?, ?, ?, ?, ?, ?)
    `;
    db.run(sql, [
      newCategoryData.categoryFkParentCategories,
      newCategoryData.categoryName,
      newCategoryData.categoryIcon,
      newCategoryData.categoryLink,
      newCategoryData.categoryIsActive,
      currentDateAndTime,
      newCategoryData.categoriesNotes
    ], function (err) {
      if (err) {
        reject(err);
      } else {
        const result = { id: this.lastID, ...newCategoryData };
        resolve(result);
      }
    });
  });
}
static updateAdminCategory(categoryId, updatedCategoryData) {
  return new Promise((resolve, reject) => {
    const currentDateAndTime = getCurrentDateTime();
    const sql = `
      UPDATE categories
      SET
        categories_fk_parentCategories = ?,
        categories_name = ?,
        categories_icon = ?,
        categories_link = ?,
        categories_is_active = ?,
        categories_updated_at = ?,
        categories_notes = ?
      WHERE
        categories_id = ?
    `;
```

```
      db.run(sql, [
        updatedCategoryData.categories_fk_parentCategories,
        updatedCategoryData.categories_name,
        updatedCategoryData.categories_icon,
        updatedCategoryData.categories_link,
        updatedCategoryData.categories_is_active,
        currentDateAndTime,
        updatedCategoryData.categories_notes,
        categoryId
      ], function (err) {
        if (err) {
          reject(err);
        } else {
          resolve({ id: categoryId, ...updatedCategoryData });
        }
      });
    });
  }
  static deleteAdminCategory(categoryId) {
    return new Promise((resolve, reject) => {
      const sql = 'DELETE FROM categories WHERE categories_id =
?';
      db.run(sql, [categoryId], function (err) {
        if (err) {
          reject(err);
        } else {
          if (this.changes > 0) {
            resolve(``);
          } else {
            reject(``);
          }
        }
      });
    });
  }
}
class CategoryTranslationModel {
  static getCategoryTranslationByLanguageAndCity(categoryId,
language, cityName) {
    return new Promise((resolve, reject) => {
      const translationTable =
`category_translations${language.toUpperCase()}`;
      const sql = `
        SELECT *
        FROM ${translationTable}
        WHERE
${translationTable}.category_translations_fk_categories = ?
        AND ${translationTable}.category_translations_cityversion
= (
          SELECT City_version_id FROM City_version WHERE
City_version_name = ?
        )
      `;
```

```
      db.get(sql, [categoryId, cityName], (err, translations) => {
        if (err) {
          console.error(err);
          reject(err);
        } else {
          resolve(translations);
        }
      });
    });

  }
  static getAdminCategoryTranslationByLanguageAndCity(categoryId,
language, cityId) {
    return new Promise((resolve, reject) => {
      const translationTable =
`category_translations${language.toUpperCase()}`;
      const sql = `
        SELECT *
        FROM ${translationTable}
        WHERE
${translationTable}.category_translations_fk_categories = ?
        AND ${translationTable}.category_translations_cityversion
= ?
      `;
      db.get(sql, [categoryId, cityId], (err, translations) => {
        if (err) {
          reject(err);
        } else {
          resolve(translations);
        }
      });
    });

  }
    static getAdminAllTranslationsOfCategory(categoryId, language)
{
      return new Promise((resolve, reject) => {
        const translationTable =
`category_translations${language.toUpperCase()}`;
        const sql = `
          SELECT *
          FROM ${translationTable}
          WHERE
${translationTable}.category_translations_fk_categories = ?
        `;
        db.all(sql, [categoryId], (err, translations) => {
          if (err) {
            reject(err);
          } else {
            resolve(translations);
          }
        });
      });
```

```
    }
  static createAdminCategoryTranslation(categoryId, language,
translationData) {
      const translationTable =
`category_translations${language.toUpperCase()}`;

      const currentDateAndTime = getCurrentDateTime();

      const sql = `
        INSERT INTO ${translationTable} (
          category_translations_fk_parent_categories,
          category_translations_fk_categories,
          category_translations_fk_subcategories,
          category_translations_title,
          category_translations_descriptionSeo,
          category_translations_description,
          category_translations_descriptionJson,
          category_translations_created_at,
          category_translations_notes,
          category_translations_cityversion
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
      `;

      db.run(
        sql,
        [
          translationData.category_translations_fk_parent_categori
es || null,
          categoryId,
          translationData.category_translations_fk_subcategories
|| null,
          translationData.categoryTranslationsTitle,
          translationData.categoryTranslationsDescriptionSeo,
          translationData.categoryTranslationsDescription,
          translationData.categoryTranslationsDescriptionJson,
          currentDateAndTime,
          translationData.categoryTranslationsNotes,
          translationData.categoryTranslationsCityversion
        ],
        (err) => {
          if (err) {
            reject(err);
          } else {
            resolve('Category translation created successfully');
          }
        }
      );
    });
  }
  static updateAdminCategoryTranslation(categoryId, language,
cityId, translationData) {
    return new Promise((resolve, reject) => {
```

```
      const translationTable =
`category_translations${language.toUpperCase()}`;
      const currentDateAndTime = getCurrentDateTime();
      const sql = `
        UPDATE ${translationTable}
        SET
          category_translations_title = ?,
          category_translations_descriptionSeo = ?,
          category_translations_description = ?,
          category_translations_descriptionJson = ?,
          category_translations_updated_at = ?,
          category_translations_notes = ?,
          category_translations_cityversion = ?
        WHERE
          category_translations_fk_categories = ? AND
          category_translations_cityversion = ?;
      `;
      db.run(
        sql,
        [
          translationData.category_translations_title,
          translationData.category_translations_descriptionSeo,
          translationData.category_translations_description,
          translationData.category_translations_descriptionJson,
          currentDateAndTime,
          translationData.category_translations_notes,
          translationData.category_translations_cityversion,
          categoryId,
          cityId
        ],
        (err) => {
          if (err) {
            reject(err);
          } else {
            resolve('Category translation updated successfully');
          }
        }
      );
    });
  }
  static deleteAdminCategoryTranslation(categoryId, language) {
    return new Promise((resolve, reject) => {
      const translationTable =
`category_translations${language.toUpperCase()}`;
      const sql = `
        DELETE FROM ${translationTable}
        WHERE category_translations_fk_categories = ?
      `;
      db.run(sql, [categoryId], function (err) {
        if (err) {
          reject(err);
        } else {
          if (this.changes > 0) {
```

```
            resolve(``);
          } else {
            reject(``);
          }
        }
      });
    });
  }
  static deleteAdminSubcategory(categoryId) {
    return new Promise((resolve, reject) => {
      const sql = 'DELETE FROM subcategories WHERE
subcategories_id = ?';
      db.run(sql, [categoryId], function (err) {
        if (err) {
          reject(err);
        } else {
          if (this.changes > 0) {
            resolve(``);
          } else {
            reject(``);
          }
        }
      });
    });
  }

}
```

//Обробник Post-запиту для отримання категорій

```
app.post('/admin/category/receive', async (req, res) => {
  console.log("", req.body);
  const IDrequestLog = res.locals.IDrequestLog
  const stub = null;
  try {
    const { categoryData } = req.body;

   categoryData
    if (!categoryData) {
      //return res.status(400).json({ error: 'Missing categoryData
in request body' });
      const functionHandleResponse = await handleResponse({status:
400, message: 'Missing categoryData in request body',
clientMessageCodes: '2_400_3', isError: false }, stub,
IDrequestLog);
      return
res.status(functionHandleResponse.system.status).json({...function
HandleResponse.systemAnswer, categoryData:
functionHandleResponse.dataAnswer});
    }
    //const { category, translationCategory,
cityVersionToTranslate } = categoryData;
    const { category, categoryOptions } = categoryData;
    const baseUrl = `${req.protocol}://${req.get('host')}`;
```

```
    const getCategories = await
functionForGettingCategoriesByParameters(category,
categoryOptions, baseUrl)
    const functionHandleResponse = await handleResponse({status:
getCategories.status, message: getCategories.message,
clientMessageCodes: getCategories.clientMessageCodes, isError:
getCategories.isError }, getCategories.data, IDrequestLog);
    return
res.status(functionHandleResponse.system.status).json({...function
HandleResponse.systemAnswer, categoryData:
functionHandleResponse.dataAnswer});
  } catch (error) {
    console.error(категорій ', error);
    const errorFunction = await handleResponse({status: 500,
message: 'Internal server error when retrieving categories',
clientMessageCodes: '2_500_4', isError: true}, error,
IDrequestLog);
    return
res.status(errorFunction.system.status).json({...errorFunction.sys
temAnswer});
  }
});
async function
functionForGettingCategoriesByParameters(categoryMeaning,
categoryOptions, baseUrl) {
  const { translationCategory = null, cityVersionToTranslate =
null } = categoryOptions || {};
    try {
      let filteredCategories;
      if (String(categoryMeaning) &&
String(categoryMeaning).toLowerCase() === 'all') {

        filteredCategories = await
CategoryModel.getGeneralMethodCategory();
      } else if (!isNaN(categoryMeaning)) {
        filteredCategories = await
CategoryModel.getGeneralMethodCategory(categoryMeaning);
      } else {
        throw new Error(, в функции
functionForGettingCategoriesByParameters');
      }
      if (!filteredCategories ||
(Array.isArray(filteredCategories) && filteredCategories.length
=== 0) || (!Array.isArray(filteredCategories) &&
Object.keys(filteredCategories).length === 0)) {
        return { success: 'Запрашиваемая категория не найдена',
message:'The requested category was not found', status:404,
clientMessageCodes: '2_404_1', data: null, isError: false };
      }
     if (Array.isArray(filteredCategories)) {
        filteredCategories = filteredCategories.map(category => ({
          ...category,
```

```
          categories_url_icon: category.categories_icon && typeof
category.categories_icon === 'string' &&
category.categories_icon.trim() !== '' ?
`${baseUrl}/${config.categoryPhotoPath}/${category.categories_icon
}` : null
        }));
      } else {
        filteredCategories.categories_url_icon =
filteredCategories.categories_icon && typeof
filteredCategories.categories_icon === 'string' &&
filteredCategories.categories_icon.trim() !== '' ?
`${baseUrl}/${config.categoryPhotoPath}/${filteredCategories.categ
ories_icon}` : null;
        filteredCategories = [filteredCategories];
      }
      if (categoryOptions === null) {
        return { success: 'Категория без перевода успешно
получена', message:'Category without translation successfully
obtained', status:200, clientMessageCodes: '2_200_3', isError:
false, data: filteredCategories };
      }if (Array.isArray(filteredCategories)) {
              for (const category of filteredCategories) {
                const translation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(category.categories_id, translationCategory,
cityVersionToTranslate);
                category.categories_translation = translation ?
[translation] : null;
              }
            } else {
              filteredCategories = [filteredCategories];
              for (const category of filteredCategories) {
                const translation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(category.categories_id, translationCategory,
cityVersionToTranslate);

                category.categories_translation = translation ?
translation : null;

              }

            }

          */

      for (const category of filteredCategories) {
        const translation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(category.categories_id, translationCategory,
cityVersionToTranslate);
        category.parentCategories_translation = translation ?
(Array.isArray(translation) ? translation : translation) : null;
      }
```

```
      return { success: Категорії з перекладом успішно отримані',
message:'Categories with translation successfully received',
status:200, clientMessageCodes: '2_200_6', isError: false, data:
filteredCategories };
    } catch (error) {
      console.error(', error);
      throw error;
    }
};
```

// Обробник Post-запиту для запису категорій та створення стандартного перекладу

```
app.post('/admin/v1.1/category/creation', async (req, res) => {
  try {
    let generatelinkCategory;
    let existingCategory;
    do {
      generatelinkCategory = generateRandomPassword(10);
categories_link
      existingCategory = await
CategoryModel.getGeneralMethodCategory(generatelinkCategory);
    } while (existingCategory);
    const cityName = "ukraine";
    const language =
config.multilingualManagement.availableTranslationsForSite;
    const standardValueCreatingNewCategory ={
      categoryFkParentCategories: 0,
      categoryName: 'New Category',
      categoryIcon: 'New Category icon',
      categoryLink: generatelinkCategory,
      categoryIsActive: 0,
      categoriesNotes: 'New Category notes',
    }

    const requestCreateNewCategory = await
CategoryModel.createAdminCategory(standardValueCreatingNewCategory
);
    const categoryId = requestCreateNewCategory.id;
    const getFullRecordCategoryId = await
CategoryModel.getGeneralMethodCategory(categoryId);
    const cityVersions = await
FunctionalModels.getGeneralCityVersion(cityName);
    const cityId = cityVersions[0].City_version_id;
    for (const lang of language) {
      const standardTranslationValueCreatingNewCategory = {
        categoryTranslationsTitle: `New title translations
category ${lang}`,
        categoryTranslationsDescriptionSeo: `New description Seo
translations category ${lang}`,
        categoryTranslationsDescription: `New description
translations category ${lang}`,
        categoryTranslationsDescriptionJson: `New description Json
translations category ${lang}`,
        categoryTranslationsNotes: `New notes translations
category ${lang}`,
```

```
        categoryTranslationsCityversion: cityId
      }
      await
CategoryTranslationModel.createAdminCategoryTranslation(categoryId
, lang, standardTranslationValueCreatingNewCategory);
    }
    res.status(200).json({ message: ', categoryData:
[getFullRecordCategoryId] });
  } catch (error) {
    console.error("", error);
    res.status(500).json({ error: 'Внутрішня помилка сервера,
details: error });
  }
});
```

// Обробник Post-запиту для запису перекладу категорій

```
app.post('/admin/category/language/:language/:category', async
(req, res) => {
  try {
    const language = req.params.language;
    const categoryId = req.params.category;
    const translationData = req.body;
    const cityVersion =
parseInt(translationData.category_translations_cityversion, 10);

    const existingTranslation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(
      categoryId,
      language,
      cityVersion
    );
    if (existingTranslation) {
      res.status(200).json({ message: ', translation:
existingTranslation });
    } else {
      await
CategoryTranslationModel.createAdminCategoryTranslation(categoryId
, language, translationData);
      const createdTranslation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(
        categoryId,
        language,
        cityVersion
      );

      if (createdTranslation) {
        res.status(200).json({ message: ', translation:
createdTranslation });
      } else {
        res.status(500).json({ error: ' });
      }
    }
```

```
  } catch (err) {
    res.status(500).json({ error: ', details: err });
  }
});
```

//Маршрут для оновлення записів категорій

```
app.put('/admin/category/:category', async (req, res) => {
  try {
    const categoryId = parseInt(req.params.category, 10);
    const updatedCategoryData = req.body;
    const baseUrl = `${req.protocol}://${req.get('host')}`;
    const updatedCategory = await
CategoryModel.updateAdminCategory(categoryId,
updatedCategoryData);
getGeneralMethodCategory
    const updatedCategoryDetails = await
CategoryModel.getGeneralMethodCategory(categoryId);
    updatedCategoryDetails.categories_url_icon =
`${baseUrl}/${config.categoryPhotoPath}/${updatedCategoryData.cate
gories_icon}`;
    res.status(200).json({ message:, categoryData:
[updatedCategoryDetails] });
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: ', details: err });
  }
});
```

//Маршрут для оновлення записів перекладу категорій

```
app.put('/admin/category/language/:language/:category/:city',
async (req, res) => {
  try {
    const language = req.params.language;
    const categoryId = parseInt(req.params.category, 10);
    const cityId = parseInt(req.params.city, 10);
    const translationData = req.body;
    const existingTranslation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(
      categoryId,
      language,
      cityId
    );
    if (existingTranslation) {
      await
CategoryTranslationModel.updateAdminCategoryTranslation(
        categoryId,
        language,
        cityId,
        translationData
      );
```

```
      const updatedTranslation = await
CategoryTranslationModel.getAdminCategoryTranslationByLanguageAndC
ity(
        categoryId,
        language,
        cityId
      );
      res.status(200).json({ message: ', translation:
updatedTranslation });
    } else {
      res.status(404).json({ error: ' });
    }
  } catch (err) {
    res.status(500).json({ error: ', details: err });
  }
});
// Маршрут для видалення категорії та її перекладу за ID категорії
app.delete('/admin/v1.1/category/delete', async (req, res) => {
  try {
    const { categoryData } = req.body;
    const { category } = categoryData;
    const language =
config.multilingualManagement.availableTranslationsForSite;
    for (const lang of language) {
      await
CategoryTranslationModel.deleteAdminCategoryTranslation(category,
lang);
    }

    await CategoryModel.deleteAdminCategory(category);
    res.status(200).json({ message:  });
  } catch (error) {
    console.log("", error);
    res.status(500).json({ error: ' });
  }
});
```

ДОДАТОК В

Лістинг програми для роботи з категорією

Мобільний додаток:

- Компонент для відображення информації про категорію

```
const HomeAdminPanel = ({ onBuy }) => {
  const { t } = useTranslation();
  const OptionstabsSiteManagement = tabsSiteManagement;
  const defaultTab  = OptionstabsSiteManagement[0];
  const [activeTab, setActiveTab] = useState(defaultTab);
  const renderSelectedTabContent = () => {
    const tabBarParams = {
      functionCallbackWithCategoriesTable:
InfoEditingSectionTabSiteManagement,
      ref: categoryTableRef,
    }
    const renderComponent =
TabBarStructureForSiteManagement(activeTab, tabBarParams);
    return renderComponent;
  };
  const parametersForTabBarComponent = {
    TabBarItems: OptionstabsSiteManagement,
    activeTab: activeTab
  }
  const ClickingButtonComponentTabBar = (tab) => {
    setActiveTab(tab);
  };
  const creatingNewCategoryEntry = async (tab) =>{
    try{
      const resultResultingFunction =
resolveCategoryCreationFunction(tab);
      const response = await apiHelper(resultResultingFunction);
      const categorySpecifications = await
functionGetIdCreatedCategoryOrProduct(tab);
      const categoryCreationRequestMessage =
response.data.message;
      const passedNecessaryParametersComponent = {
        passedNecessaryParametersComponent: {
          ...categorySpecifications,
          externalData:
response.data[categorySpecifications.informationContainer][0][cate
gorySpecifications.responseMappingCategory.categoryId],
        },
      };
      handleShowNotification(categoryCreationRequestMessage,
'success');



      await
setSharedInfoTransmittedFromComponent(passedNecessaryParametersCom
ponent);
      setIsSideMenuVisible(true);
```

```
      updateCategoriesTableData();
    }catch(error){
      console.error(', error);
     error.response.data
      const categoryCreationRequestMessageError =
error.response?.data?.error || 'Неизвестная ошибка';
      handleShowNotification(categoryCreationRequestMessageError,
'error');
    };
  };
  const [activeTabSideMenu, setActiveTabSideMenu] =
useState('Основная информация');
  const handleTabClickSideMenu = (tab) => {
    setActiveTabSideMenu(tab);
  };
  const [name, setName] = useState('');
  const renderSelectedTabContentSideMenu  = () => {
    switch (activeTabSideMenu) {
      case 'Основная информация':
        return <CategoryMainPart
acceptedParametersComponent={sharedInfoTransmittedFromComponent}
callbackFunctionsCategoryMainPart={callbackFunctionsCategoryMainPa
rt} />;
      case 'Украинская версия':
        // Здесь возвращаем компонент для категорий
        const languageCategoryUa = "ua";
        return <CategorySideMenuLanguageVersion
                  acceptedParametersComponent={{
                    passedNecessaryParametersComponent: {
                      ...sharedInfoTransmittedFromComponent.passed
NecessaryParametersComponent,
                      currentLanguageVersion: languageCategoryUa,
                    },
                  }}
                  onNameChange={setName}
                />;
      case '':
        // Здесь возвращаем компонент для подкатегорий
        const languageCategoryEn = "en";
        return <CategorySideMenuLanguageVersion
                acceptedParametersComponent={{
                  passedNecessaryParametersComponent: {
                    ...sharedInfoTransmittedFromComponent.passedNe
cessaryParametersComponent,
                    currentLanguageVersion: languageCategoryEn,
                  },
                }}
                onNameChange={setName}
              />;

      default:
        return null;
    }
```

```
  };
const [isSideMenuVisible, setIsSideMenuVisible] = useState(false);
  //const [selectedCategoryId, setSelectedCategoryId] =
useState(null);
  const [sharedInfoTransmittedFromComponent,
setSharedInfoTransmittedFromComponent] = useState(null);
  //const [parametersForCategoriesForRequest,
setParametersForCategoriesForRequest] = useState(null);
  const InfoEditingSectionTabSiteManagement = (rowData) => {
    setSharedInfoTransmittedFromComponent(rowData);
    setIsSideMenuVisible(true);
  };
  // Создаем ref для CategoriesTable
  const categoryTableRef = useRef();
  const updateCategoriesTableData = () => {
    // Вызываем fetchData через ref в CategoriesTable
    if (categoryTableRef.current &&
categoryTableRef.current.fetchData) {
      categoryTableRef.current.fetchData();
    }
  };
const { showNotification } = useNotification();
  const handleShowNotification = (message, type) => {
    showNotification(message, type);
  };
CategoryMainPart
  const callbackFunctionsCategoryMainPart = {
    onNameChange: setName,
    onDataUpdated: updateCategoriesTableData,
    setIsSideMenuVisible: setIsSideMenuVisible,
  };
  const callbackFunctionsComponentTabBar = {
    categoryCreateButtonClick: creatingNewCategoryEntry,
    onTabClick: ClickingButtonComponentTabBar,
  };
```

- Компонент для роботи з категоріями

```
const CategoryMainPart = ({ /* parametersForCategoriesForRequest
*/ acceptedParametersComponent, callbackFunctionsCategoryMainPart
}) => {
  const { passedNecessaryParametersComponent } =
acceptedParametersComponent;
  const { ancestorCategoryInfo, externalData,
informationContainer, nonDisplayValues, responseMappingCategory,
targetInfoType } = passedNecessaryParametersComponent;
  const {
    categoryId,
    categoryFkParentCategories,
    categoryName,
    categoryIconName,
    categoryLink,
    categoryIsActive,
    categoryNotes,
    categoryUpdatedAt,
```

```
    categoryCreatedAt,
    categoryIconsUrl, } = responseMappingCategory;
  const {
      onNameChange,
      onDataUpdated,
      setIsSideMenuVisible,
  } = callbackFunctionsCategoryMainPart;
  useEffect(() => {
    fetchData();
  }, [acceptedParametersComponent, onNameChange]);
 [objectNamesCategory][responseMappingCategory.categoryName]
  const fetchData = async () => {
    const parameterCategory = {
      meaningCategoryId: externalData,
      parameterCategoryOptions: null
    }
    try {
      const requestCategoryData = await
allowGetCategoryFunction(targetInfoType, parameterCategory);
console.log("Полученный ответ", requestCategoryData);
fetchDataAndUpdateState(requestCategoryData);
    } catch (error) {
    }
  };
  const [categoryDataToDisplay, setCategoryDataToDisplay] =
useState({
    arrayDataCategoryId: '',
    arrayDataCategoriesFkParentCategories: '',
    arrayDataCategoryName: '',
    arrayDataCategoryIconName: '',
    arrayDataCategoryLink: '',
    arrayDataCategoryIsActive: '',
    arrayDataCategoryNotes: '',
    arrayDataCategoryUpdatedAt: '',
    arrayDataCategoryCreatedAt: '',
    arrayDataCategoryIconsUrl: '',
  });

  console.log("", categoryDataToDisplay);
  const fetchDataAndUpdateState = async (response) => {
    const responseData = Array.isArray(response.data) ?
response.data : response.data;
    const categoryResponseContent =
responseData[informationContainer]?.[0];
    const arrayOfReceivedCategoryData = {
      arrayDataCategoryId: categoryResponseContent[categoryId],
      //arrayDataCategoriesFkParentCategories:
categoryResponseContent[categoryFkParentCategories] || null,
        arrayDataCategoriesFkParentCategories:
    categoryResponseContent[categoryFkParentCategories] !==
undefined
      ? categoryResponseContent[categoryFkParentCategories] !==
null
```

```
        ? categoryResponseContent[categoryFkParentCategories]
        : null
      : null,
      arrayDataCategoryName:
categoryResponseContent[categoryName],
      arrayDataCategoryIconName:
categoryResponseContent[categoryIconName],
      arrayDataCategoryLink:
categoryResponseContent[categoryLink],
      arrayDataCategoryIsActive:
categoryResponseContent[categoryIsActive],
      arrayDataCategoryNotes:
categoryResponseContent[categoryNotes],
      arrayDataCategoryUpdatedAt:
categoryResponseContent[categoryUpdatedAt],
      arrayDataCategoryCreatedAt:
categoryResponseContent[categoryCreatedAt],
      arrayDataCategoryIconsUrl:
categoryResponseContent[categoryIconsUrl],
    };
    console.log('', arrayOfReceivedCategoryData);
    setCategoryDataToDisplay(arrayOfReceivedCategoryData);
    if (onNameChange) {
      onNameChange(categoryResponseContent?.[responseMappingCatego
ry.categoryName]);
    }
  };
  const handleFieldChange = (field, value) => {
    setCategoryDataToDisplay(prevData => ({
      ...prevData,
      [field]: value
    }));
  };


  const invertValue = (value) => {
    if (value === '0') {
      return '1';
    } else {
      return '0';
    }
  };
  const handleButtonClick = () => {
    setCategoryDataToDisplay((prevFields) => ({
      ...prevFields,
      arrayDataCategoryIsActive:
invertValue(prevFields.arrayDataCategoryIsActive),
    }));
  };


  const handleSaveChanges = async () => {
```

```
    const parameterCategory = {
      meaningCategoryId: externalData,
      parameterCategoryOptions: null
    }

    const dataForUpdatingRecords = {
      [categoryFkParentCategories]:
categoryDataToDisplay.arrayDataCategoriesFkParentCategories,
      [categoryName]: categoryDataToDisplay.arrayDataCategoryName,
      [categoryIconName]:
categoryDataToDisplay.arrayDataCategoryIconName,
      [categoryLink]: categoryDataToDisplay.arrayDataCategoryLink
|| null,
      [categoryIsActive]:
categoryDataToDisplay.arrayDataCategoryIsActive || null,
      [categoryNotes]:
categoryDataToDisplay.arrayDataCategoryNotes,
    };
    try{
      const requestForDataUpdates = await
allowCategoryUpdateFunction(targetInfoType, parameterCategory,
dataForUpdatingRecords);
      fetchDataAndUpdateState(requestForDataUpdates);
      handleShowNotification(', 'success');
      if (onDataUpdated) {
        onDataUpdated();
      }
      if (onNameChange) {
        onNameChange(responseData.category?.categories_name);
    } catch (error){
 const handleDeleteCategory = async () => {
    try {
      const parameterCategory = {
        meaningCategoryId:
categoryDataToDisplay.arrayDataCategoryId,
        parameterCategoryOptions: null
      }
      const gettingParametersForRequest = await
allowFunctionToDeleteCategory(targetInfoType, parameterCategory);
      handleShowNotification(, 'success');
      setIsSideMenuVisible(false);
      if (onDataUpdated) {
        onDataUpdated();
      }
    } catch (error) {
      console.error('', error);
    }
  };
  const { showNotification } = useNotification();
  const handleShowNotification = (message, type) => {
    showNotification(message, type);
  };
  const [imageSrc, setImageSrc] = useState('');
```

```
  const handleFileChange = async (e) => {
    const file = e.target.files[0];
    if (file) {
      try {
        const response = await
allowGetCategoryFunctionToLoadIcon(targetInfoType, file);
        if (response.data.success) {
         setImageSrc(response.data.imageUrl);
          setCategoryDataToDisplay((prevData) => ({
            ...prevData,
            arrayDataCategoryIconName: file.name,
            arrayDataCategoryIconsUrl: response.data.imageUrl,
          }));
          handleShowNotification(', 'success');
        } else {
          console.error('');
          handleShowNotification(', 'error');
        }
      } catch (error) {
        console.error('', error);
      }
    } else {
      setImageSrc('');
      setCategoryDataToDisplay((prevData) => ({
        ...prevData,
        arrayDataCategoryIconName: null,
        arrayDataCategoryIconsUrl: null,
      }));

    }
  };
  const handleDeleteIcon = (field, value) => {
    setCategoryDataToDisplay(prevFields => ({
      ...prevFields,
      arrayDataCategoryIconName: null
    }));
  };
  const [isOpenTabSelectingParameters,
setOpenTabSelectingParameters] = useState(false);
  const [categoryHierarchyInfo, setCategoryHierarchyInfo] =
useState([]);
  const inputParametersTabSelectingParameters = {
    inputParametersComponent: categoryHierarchyInfo,
  };
  const handleOpenTab = async () => {
    try {
      const parameterCategory = {
        meaningCategoryId: 'all',
        parameterCategoryOptions: null
      }
      const tap = ancestorCategoryInfo.ancestor_targetInfoType;
      const requestParameterContainingCategory = await
allowGetCategoryFunction(tap, parameterCategory);
```

```
      const listOfContainingCategoryNamesAndId =
requestParameterContainingCategory.data[ancestorCategoryInfo.ances
tor_informationContainer].map(item => ({
        id:
item[ancestorCategoryInfo.ancestor_valueGetCategoryId],
        name:
item[ancestorCategoryInfo.ancestor_valueGetCategoryName]
      }));

TabSelectingParameters
      setCategoryHierarchyInfo(listOfContainingCategoryNamesAndId)
;
      setOpenTabSelectingParameters(true);
    } catch (error) {
      console.error("= error);
    }
  };
  const handleCloseTab = (dataToPass) => {
    const valueId = dataToPass.id;

    setCategoryDataToDisplay((prevData) => ({
      ...prevData,
      arrayDataCategoriesFkParentCategories: valueId,
    }));
    setOpenTabSelectingParameters(false);
  };

  const dataOutputFromTheComponent = {
    setDataToPass1: handleCloseTab,
  };
```

Національний аерокосмічний університет ім. М. Є. Жуковського "ХАІ"
Кафедра Систем управління літальних апаратів

Презентація до дипломного проекту магістра на тему:

Розробка засобів технічної реалізації та інформаційної підтримки розподілених систем термостабілізації

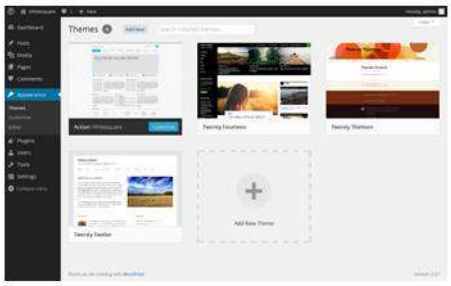
Здобувач: Горбенко Вадим Юрійович Гр. 362

Спеціальність: 151 "Автоматизація та комп'ютерно-інтегровані технології"
Освітня програма: «Інженерія мобільних додатків»

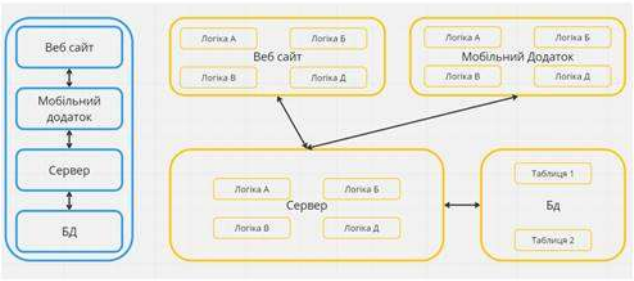
Керівник: доц. Джулгаков В.Г.

Дата захисту: 16 січня 2024 р.

СТАН ПРОБЛЕМИ



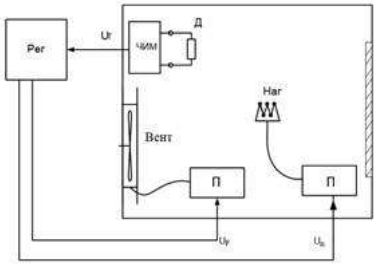
Головний екран CMS WordPress



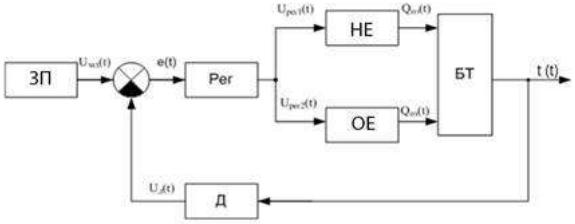
Стркутра монолітної програми та мікросервісна архітектура

Головний екран CMS Joomla

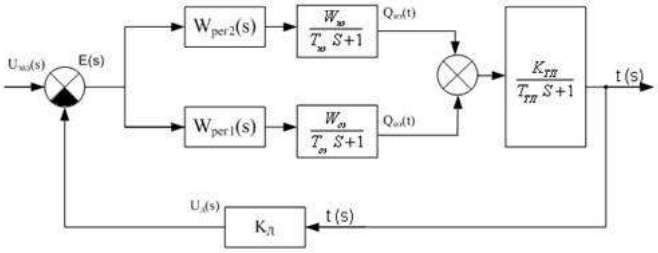
Аналіз і синтез системи термостабілізації



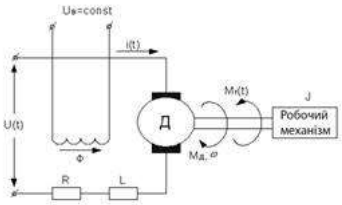
Система регулювання температури



Функціональна схема системи управління термостатом



Структурна схема керування термостатом



Регульований електропривод постійного струму

2

$$Тээ = J \cdot k_Д^2 \cdot r_Я = 9150 \cdot 15.21 \cdot 1.4 \cdot 10^{-6} \approx 0.2с$$

$$T_э = \frac{L}{r_Я} = \frac{0.025}{1.4} = 0.018с$$

Постійні часу дорівнюють

Параметр, одиниці виміру	Значення
Номінальна потужність, кВт	0,25
Номінальна напруга, В	50
Номінальна частота обертання, об/хв	1500
Номінальний струм на якорі, А	6,85
Момент інерції якоря, кг·см²	9,15
Опір обмотки якоря, Ом	1,4
Індуктивність якоря L, Гн	0.025

Характеристики електродвигуна постійного струму П11М

$$k_Д = \frac{\omega_{ДН}}{U_Н - I_Я \cdot r_Я}$$

$$k_Д = \frac{157}{50 - 6.85 \cdot 1.4} \approx 3.9$$

$$[K_Д] = \frac{1/c}{B - A \cdot O_M} = 1/B \cdot c$$

Коефіцієнт передачі електродвигуна за керуючим впливом

$$k_{ДВ} = k_Д^2 \cdot r_Я \approx 21.3$$

$$[k_{ДВ}] = \frac{O_M}{B^2 \cdot c^2} = \frac{1}{H \cdot M \cdot c}$$

Коефіцієнт передачі електродвигуна по обурюючому впливу

Характеристики	Параметри
Діапазон вимірюваних температур	От -40°С до +110°С
Вихідна напруга датчика	От 0 В до 5 В
Напруга живлення	постійне 12 В ± 10%.
Споживаний струм	не більше 12 мА.
Точність вимірювання температури	± 1,2 С

Характеристики та параметри датчика температури

3

$$W_{ДВ}(s) = \frac{\omega_{ЭД}(s)}{U_{УМ}(s)} = \frac{3.9}{0.2s+1}$$

Передатну функцію двигуна

$$W_{ОЭ}(s) = \frac{Q(s)}{U_{per}(s)} = \frac{-0.5}{0.6s+1}$$

Передатна функція охолоджувального елемента

$$W_{НЭ}(s) = \frac{Q(s)}{U_{НЭ}(s)} = \frac{5}{0.9s+1}$$

Передатна функція нагрівального елемента

$$W_{НЭ}(s) = \frac{t(s)}{U_{Д}(s)} = 2.6$$

Передатна функція термодатчика

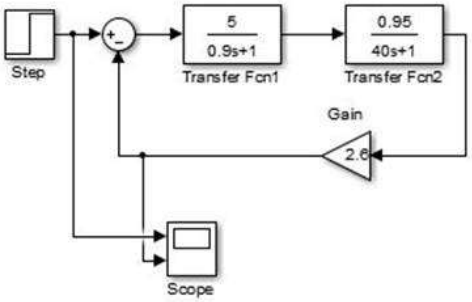
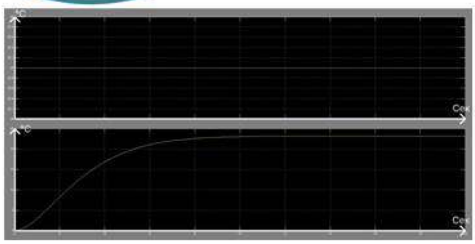


Схема моделювання контуру регулювання температури нагрівання.

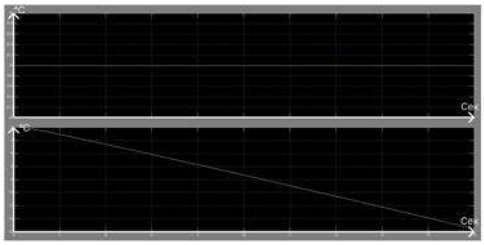
$$\omega(s) = W_U(s) \cdot U(s) + W_f(s) \cdot M(s)$$
$$= \frac{k_Д}{T_{ЭМ}T_Э s^2 + T_{ЭМ}s + 1} \cdot U(s)$$
$$+ \frac{k_{ДВ}(T_Э s + 1)}{T_{ЭМ}T_Э s^2 + T_{ЭМ}s + 1} \cdot M(s)$$
$$\omega(s) = \frac{3.9}{0.0036s^2 + 0.2s + 1} \cdot U(s)$$
$$+ \frac{0.38s + 21.3}{0.0036s^2 + 0.2s + 1} M(s)$$

Отримаємо наступну лінеаризовану математичну модель електродвигуна П11М

4



Перехідна характеристика по задаючому впливу температури нагріву



Перехідна характеристика по задаючому впливу температури з охолодження

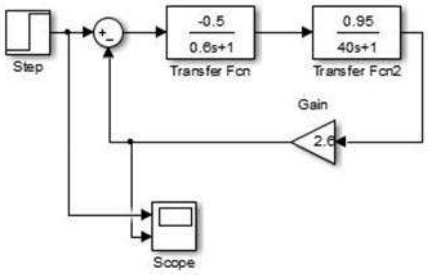
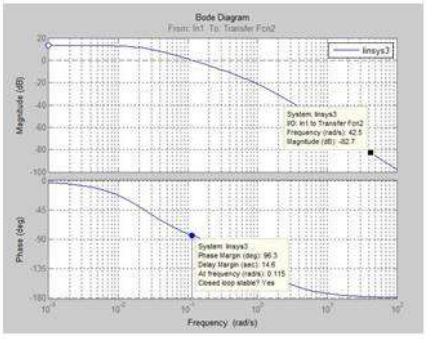


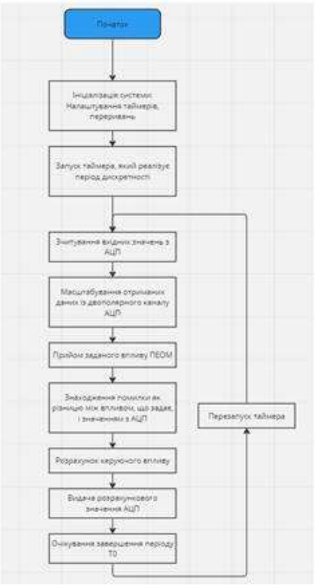
Схема моделювання контуру регулювання температури охолодження



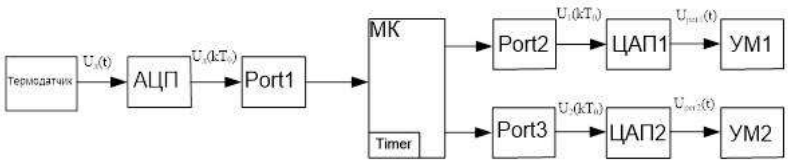
Частотні характеристики нескоректованої розімкнутої системи управління нагріванням

5

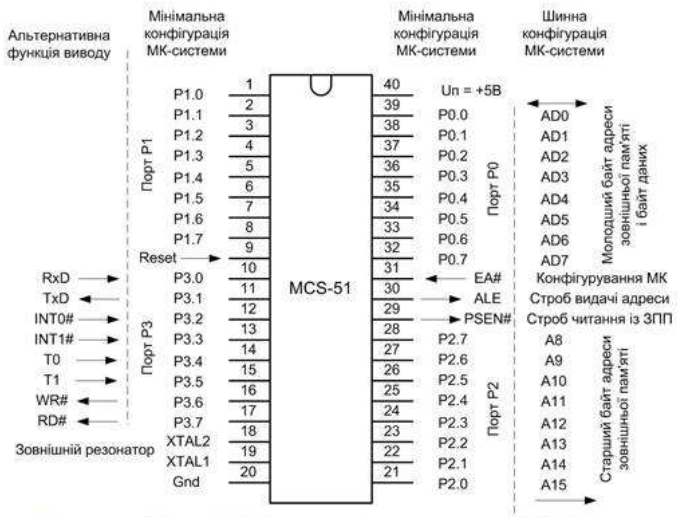
Конструкторська частина



Блок-схема алгоритму



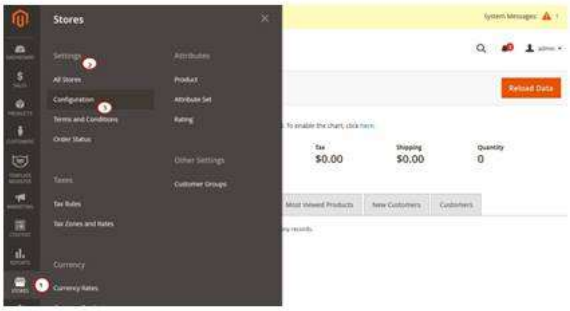
Функціональна схема обчислювального регулятора



Електричний інтерфейс базового мікроконтролера MCS-51

6

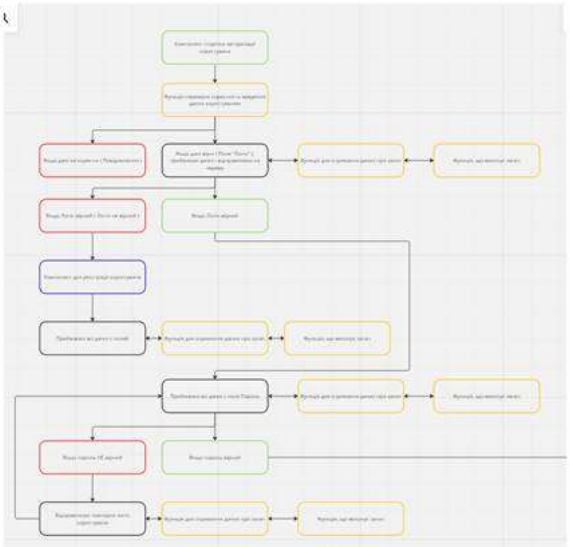
ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ З УПРАВЛІННЯ КОНТЕНТОМ НА САЙТІ



Головний екран CMS Magento

Макет головної сторінки мобільного додатка

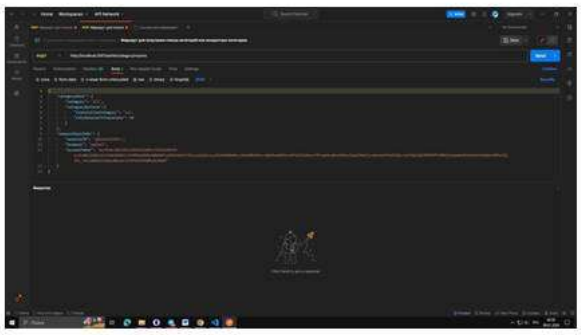
Макет головної сторінки мобільного додатка



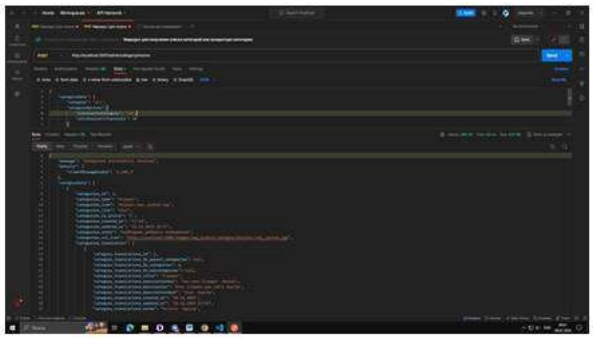
Блок-схема компонента для авторизації користувача

7

ДОСЛІДНИЦЬКА ЧАСТИНА



Головний екран програми Postman

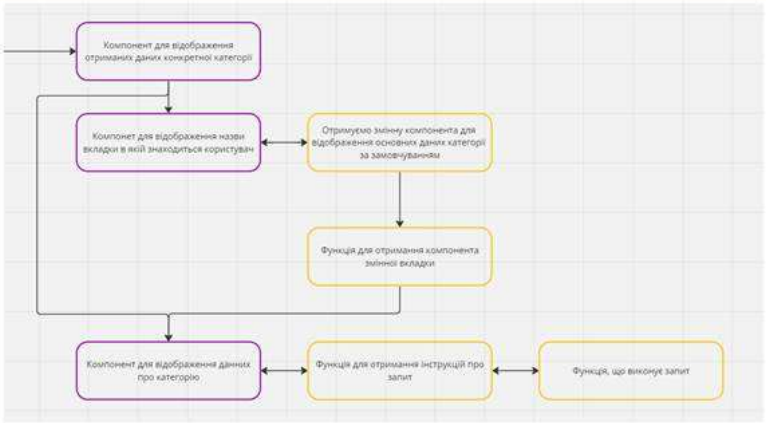


Виконання запроса для отримання категорій

Виконня запросу з помилкою

Час за який сервер оброблює запит

Блок-схема роботи маршруту для отримання батьківських категорій, категорій, підкатегорій, товару



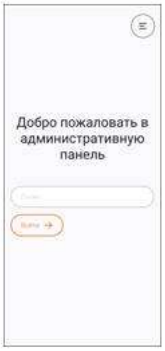
Блок-схема компонента для відображення повної інформаці батьківської категорії, категорії, підкатегорії, товару

Макет сторінки з детальною інформацією

ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ

Витрати	Ціна	Термін
Хостинг	800 грн	На місяць
App Store	3777 грн	На рік
Google Play	954	На рік
Усього витрат		5531 грн

Витрат на підтримку мобільного додатку



Скріншот мобільного додатку старовини погування

Скріншот мобільної програми головної сторінки

10

Дякую за увагу!