

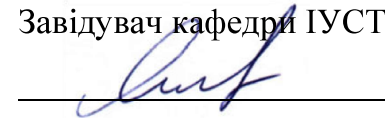
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ


к.т.н, доц. Михелєв І.Л.

" " 2025 р.

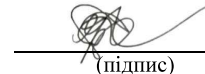
КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему:

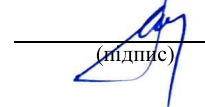
**Системний аналіз технологій віддаленого доступу
систем розумного помешкання**

Виконав: студент групи 6145 м


(підпис) **Фещенко Я.Ю.**

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,
(посада, науковий ступінь вчене звання)


(підпис) **Гайда А.Ю.**

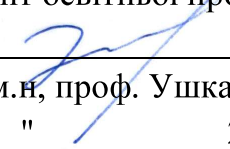
м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 124 "Системний аналіз"
Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


д.ф-м.н, проф. Ушкац М.В.

"__" "____" 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "магістр"

Студенту Фещенку Ярославу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Системний аналіз технологій віддаленого доступу систем розумного помешкання

Керівник роботи к.т.н., доц. Гайда А.Ю.

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів) _____

Аналіз стану інтернет-технологій для створення систем управління контентом, існуючих рішень та засобів розробки, Розробка проєктних рішень по системі, Розробка робочої документації системи управління контентом web-порталу МОІППО, Розрахунок економічного ефекту від впровадження ІС, Охорона праці, Охорона навколишнього середовища

5. Перелік презентаційних матеріалів _____

Content Management System, Структура МОІППО, Існуючі рішення - CMS, Типова архітектура CMS, Порівняння існуючих CMS, Рішення з засобів розробки, Схема функціональної структури, Діаграми варіантів використання, Логічна модель даних, Діаграма діяльності, Діаграма класів, Приклади екранів web-порталу

6. Консультанти розділів роботи

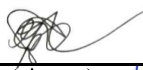
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	К.т.н., доц. Гайда А.Ю.	05.09.2024	22.12.2024
2	К.т.н., доц. Гайда А.Ю.	23.12.2024	15.06.2025
3	К.т.н., доц. Гайда А.Ю.	15.06.2025	25.09.2025
4		25.09.2025	14.11.2025
5		25.09.2025	14.11.2025
6		25.09.2025	14.11.2025

7. Дата видачі завдання 05 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

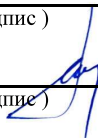
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

Студент


(підпис)

Фещенко Я.Ю.
(ПІБ)

Керівник роботи


(підпис)

Гайда А.Ю.
(ПІБ)

Анотація

Представлена робота присвячена системному аналізу засобів віддаленого доступу для створення системи "розумний дім" з метою поліпшення умов життя.

Під час роботи аналізуються існуючі рішення, формуються вимоги до системи, розроблюються концепції системи і проводиться вибір найкращої, розробляється технічне завдання. Далі проводиться розробка основних проєктних рішень за технічним, інформаційним та програмним забезпеченням, формується робоча документація за програмним забезпеченням.

Дипломна робота виконана на 98 аркушах машинописного тексту, містить 13 малюнків, 4 таблиць, 4 додатків.

Ключові слова: web-застосунок, розумний будинок, Java, Spring Boot, Hibernate

Abstract

The present work is devoted to development of the project of the information system microclimate in Smart house management system with the purpose to improve quality of life.

During operation the requirements to the system are formed, the concepts of the system are developed and the choice by best is made, the requirement specification is developed. Development of main design solutions on technical, information and software further will be carried out, working documentation under the software is formed.

The degree operation is stated on 98 sheets of the typewritten text, contains 13 figures, 4 tables, 4 applications

Keywords: web-application, smart home, Java, Spring Boot, Hibernate

ЗМІСТ

Вступ

1 Формування вимог до системи керування мікрокліматом, розробка концепції та постановка задачі

1.1 Формування вимог до інформаційної системи

1.2 Розробка концепції інформаційної системи

1.3 Постановка задачі

Висновок до розділу 1

2 Розробка проектних рішень по системі

2.1 Загальносистемні рішення

2.2 Рішення з організаційного забезпечення

2.3 Рішення з інформаційного забезпечення

2.4 Рішення з технічного забезпечення

2.5 Рішення з математичного забезпечення. Опис алгоритму

2.6 Рішення з програмного забезпечення. Опис програмного забезпечення

Висновок до розділу 2

3 Розробка робочої документації

Висновок до розділу 3

4 Розрахунок економічної ефективності впровадження інформаційної системи

5 Охорона праці

6 Охорона навколишнього середовища

Висновки

Список використаних джерел

Додатки

Додаток А Технічне завдання

Додаток Б Програма і методика випробувань

Додаток В Керівництво користувача

Додаток Г Текст програмних модулів

Вступ

Будь-який будинок - будь-то адміністративний, виробничий або житловий складається з деякого набору підсистем, що відповідають за виконання певних функцій, які вирішують різні завдання в процесі функціонування цієї будівлі. В міру ускладнення цих підсистем і збільшення кількості, виконуваних ними функцій, управління ними ставало все складніше. Також стрімко зростають витрати на утримання обслуговуючого персоналу, ремонт і обслуговування цих підсистем. Вперше ці проблеми постали при експлуатації великих адміністративних і виробничих комплексів.

Сучасна будівля такого типу - це місто в мініатюрі. Фактично в ньому діють всі служби, які були раніше неодмінними атрибутами міського господарства. У таких будівлях зазвичай існує адміністративна служба або адміністратор, які використовують і обслуговують цю систему практично цілодобово. Хоча є чимало засобів автоматики, які самі справляються з покладеними на них завданнями, такими, як опалення, вентиляція, підтримка мікроклімату, освітлення, пожежна сигналізація, димознищення, контроль входу / виходу і т.п., але управління і обслуговування всіх цих систем вимагає наявності адмініструючого персоналу.

Його обов'язком є контроль роботи цих підсистем і вжиття заходів у разі виходу їх з ладу. Але є ситуації, коли навіть дії кваліфікованого персоналу можуть виявитися неефективними. Це випадки виникнення загрози будівлі і знаходяться в ньому людям, що мають глобальний характер - пожежа, землетрус і інші стихійні лиха, терористичні атаки. Тут потрібно вживати екстраординарних заходів в лічені частки секунди. Реакція і коректність дій людей в критичній ситуації може виявитися недостатньою.

Традиційні системи забезпечення різних аспектів життєдіяльності в минулому проектувалися як автономні. Такі системи, що створювалися окремо для кожної функції і об'єднані для довільної частини будівлі. У будівлях встановлювалися системи тільки з тими можливостями і з тим ступенем складності, які були необхідні на поточний момент побудови будівлі. Подальше розширення і модернізація даних систем були складними і дорогими завданнями через безліч різних чинників. Витрати на експлуатацію такої системи складаються з витрат на експлуатацію кожної

автономної системи в окремо, вартості навчання персоналу Вартість експлуатації цих систем висока - в силу їх автономності кожна з них підтримується окремо. Вартість навчання користувачів настільки ж висока, оскільки вони повинні бути ознайомлені з експлуатацією кожної автономної системи.

1 Аналіз існуючих рішень та формування вимог до інформаційної системи, розробка її концепції і постановка задачі

1.1 Аналіз існуючих рішень

Існує два варіанти перетворення свого житла в розумний будинок. Варіант перший - самостійне створення проекту, установка, об'єднання і настройка певних датчиків в один комплекс. Варіант другий - придбати обладнання на основі готових рішень.

Системи розумного дому можуть бути дротові та бездротові. Виділяють серед систем розумного дому централізовані та децентралізовані. У деяких системах розумного будинку використовується відкритий протокол, а в деяких закритий.

Переваги дротових систем у тому, що вони мають надійний зв'язок з компонентами своєї системи. Бездротові легше встановлювати.

Централізовані системи управляються з єдиного блока, який називається базою. А децентралізовані не мають центральної бази.

Системи розумного будинку ґрунтовані на відкритих протоколах мають можливість розширення компонентами, які підтримують сумісний протокол, але вироблені іншим виробником.

1.1.1 BroadLink

Розумний будинок BroadLink - в представленому списку має найнижчий цінник, але від того не стає гірше. Навпаки, має деякі переваги. Система являє собою безпосередньо блок управління і набір різноманітних датчиків, що підключаються по мережі Wi-Fi. Серед об'єктів управління Broadlink присутні: контроль температури, мікроклімату, музичного обладнання. Відмінною особливістю є управління всіма параметрами через додаток для смартфона.

Переваги:

- Додаток для управління є для різних операційних систем смартфона.
- Зручність додавання нових пристроїв в мережу.
- Зручний і зрозумілий функціонал.

Недоліки:

- Не велика кількість одночасно працюючих пристроїв.
- Відсутні багато функцій в управлінні. Такі, як відеоконтроль.

1.1.2 Smart Bus

Розумний будинок Smart Bus - дана технологія створювалася і розроблялася в Канаді. Сьогодні її реалізували на великій кількості об'єктів в різних країнах. Відрізняється високотехнологічним обладнання і порівняно низькою ціною. В даному ціновому діапазоні тільки ця технологія має розподілений інтелект.

Переваги:

- Дизайн. Оздоблювальними матеріалами для панелей і датчиків служать скло і метал.
- Немає необхідності в центральному процесорі, так як застосовується технологія розподіленого інтелекту.
- Швидко встановлюється і налаштовується.
- Має встановлені режими настройки будинку: робота, гості, відпустка, ніч.

Недоліки:

- В одній мережі одночасно може бути до 64 приладів.

1.1.3 C-Bus

Розумний будинок C-Bus - це преміум рішення від великого виробника з світовим ім'ям. «На борту» у цій системи є відмінне устаткування для керування світловими приладами. Серед всіх користувачів системи виділяються футбольні стадіони, хмарочоси світових корпорація і особисті будинку популярних персон. Це обумовлено високою технологічністю і величезними можливостями. У дорогих система контролю, таких як C-Bus дизайн і програмне забезпечення мають вже більш високу вартість.

Переваги:

- Сумісність. Виробник розробляє не тільки ПО, а й обладнання, завдяки чому виключаються проблеми сумісності.
- Можливість управління необмеженою кількістю пристроїв.
- Є можливо налаштувати сценарну освітлення будівель.
- Наявність систем економії, таких як економія електроенергії, економія води та газу.
- Простота інтеграції. C-Bus без проблем інтегрується у вже готові системи управління або адміністрування будівель.

Недоліки:

- Самостійно встановити таку систему вже навряд чи вийде. Зважаючи на складнощі.
- Дороге обслуговування.
- Висока ціна на обладнання.

1.1.4 Crestron

Розумний будинок Crestron - це абсолютно новий рівень. Якщо цю систему встановлюють і в звичайний житловий будинок, то розгорнуться на повну і показати все спектр своїх можливостей вона може на тільки найдорожчих і вимогливих об'єктах. Система є дуже гнучкою, що дозволяє завжди знайти саме ту конфігурацію, яка повністю влаштує. Crestron – має великий набір інтегрованих можливостей. Висока вартість виправдана наданими можливостями системи.

Переваги:

- В одну систему можна об'єднати різні системи управління будинком.
- Функціонал дисплеїв управління.
- Інтерфейс користувача може бути дуже великим (сенсорні панелі, кнопкові панелі, пульти).
- Управління зовнішніми об'єктами в оточенні будинку (системи безпеки, освітлення по периметру, відкриття і закриття воріт).

Недоліки:

- Вкрай висока ціна.
- Складність в обслуговуванні. У разі виходу з ладу однієї або декількох систем, необхідно залучати фахівця для вирішення.

1.1.5 Порівняння існуючих рішень

Таблиця 1.1 – Порівняння існуючих систем розумного дому.

Назва	Ціна	Функціональність	Складність встановлення	Надійність
BroadLink	10	6	4	10
Smart Bus	7	6	10	8
C-Bus	5	8	5	7
Crestron	3	8	3	7

Таким чином, серед всіх представлених проектів розумних будинків є, як і не дуже дорогі, але з широким базовим функціоналом, так і системи з величезною кількістю можливостей по контролю багатьох параметрів будівлі. Такі системи відрізняються високою вартістю встановлення і налаштування.

Тому є доцільним розглянути варіант створення індивідуального рішення системи "розумний будинок" з віддаленим доступом.

1.2 Опис предметної області

Результатом розроблення інтернет технологій для створення системи "розумний будинок" з віддаленим доступом є інформаційна система, яка повинна допомагати автоматизувати щоденні задачі, наприклад, вимкнення світла, керування мікрокліматом, керування інтенсивністю освітлення.

Управляюча система є середовищем, складовими елементами якого є мікроконтролери, під'єднані до комп'ютерної мережі, сервер, база даних, люди, різного роду технічні і програмні засоби зв'язку і т.д. Основна мета інформаційної системи – організація збереження, обробки і передачі інформації. Інформаційна система являє собою людино-комп'ютерне оточення (систему) для обробки інформації.

Система, що розробляється, є розподіленою, тобто для її функціонування, крім програм-клієнтів, необхідна наявність принаймні одного сервера програм.

Програмне забезпечення сучасних ІС має наступні логічні рівні (див. рисунок 1):

- Рівень користувача (інтерфейс і сервіси користувача);
- Прикладний рівень (прикладні сервіси);
- Рівень даних (сервіси доступу до даних і сховище даних).

Опишемо функції кожного рівня:

- Рівень користувача: відображає дані і дозволяє користувачу редагувати їх. Існує два основних типи інтерфейсу: віконний (реалізується засобами підсистеми інтерфейсу користувача операційної системи) і на основі Web. Web-інтерфейси засновані на HTML, DHTML, XHTML чи XML, у результаті чого вони можуть відображатися будь-яким Web-оглядачем на будь-якій платформі.

- Прикладний рівень: тут реалізовані бізнес-правила й обмеження на дані (перевірка коректності даних). І хоча його сервіси використовуються презентаційним рівнем, він не прив'язаний до якого-небудь клієнта – сервіси прикладного рівня доступні будь-якому клієнту. Бізнес-правила виражаються у формі прикладних алгоритмів, корпоративних правил і т.д. Обмеження на дані гарантують точність і цілісність інформації, що зберігається. Бізнес-правила звичайно реалізуються окремим модулем на централізованому сервері, що дає можливість доступу до нього відразу декільком клієнтам.

- Рівень даних: прикладний рівень не має інформації про те, як і де зберігаються дані, які він обробляє. У цьому питанні він покладається на сервіси доступу до даних, що виконують всю роботу з отримання і передачі даних. Сервіси доступу до даних також реалізуються у виді ізольованих модулів, які знають про місце збереження інформації. Кожен модуль доступу до даних, як правило, відповідає і за цілісність сховища (наприклад, реляційної бази даних). Для багаторівневих програм у якості сховища інформації підходять сервера баз даних (MariaDB, MySQL, PostgreSQL, Oracle та ін.),

необхідні для обслуговування даних у таблицях і високопродуктивної виборки інформації.



Рисунок 1.1 – Логічна архітектура програмного забезпечення інформаційної системи

Реалізація функцій інформаційної системи неможлива без знання орієнтованих на неї інформаційних технологій. Нижче представлений опис технологій, які використовуються на кожному рівні інформаційної системи, що розробляється.

1.2.1 Технології рівня користувача

Web-оглядач і Web-інтерфейс

Web-оглядач чи Web-браузер – це програма, що формує стандартний запит для Web-сервера відповідно до правил протоколу HTTP (Hyper Text Transfer Protocol), одержує документи від Web-сервера, інтерпретує і відображає їх на екрані користувача.

Фактично Web-оглядач є єдиною програмою, яка необхідна користувачу для роботи з інформаційною системою; він надає відповідний Web-інтерфейс.

Web-інтерфейс широко розповсюджений і пропонує готові засоби відображення. Web-оглядачі, що вільно розповсюджуються, мають прості інтерфейси, що збільшує ефективність розгортання програм.

Найпоширеніші Web-оглядачі: MS Internet Explorer, Mozilla FireFox, Opera, Google Chrome.

HTML

Найпоширеніший і простий спосіб створення Web-інтерфейсу – використання HTML (HyperText Markup Language, Мова розмітки гіпертексту). Web-інтерфейс складається з набору HTML-документів, статичних чи генерованих динамічно. HTML являє собою набір команд, що називаються тегами, які розташовуються між фрагментами тексту й усередині них. Команди HTML дозволяють структурувати документ, виділяючи в ньому частини тексту (заголовки різних рівнів, абзаци, перерахування і т.д.), які логічно розрізняються. У результаті кожен Web-оглядач може формувати текст документа таким чином, щоб найкращим способом відобразити його на конкретному дисплеї. Для надання документам більшої виразності текст звичайно форматується з використанням збільшених розмірів шрифту заголовків, застосуванням напівжирного і курсивного накреслень для важливих термінів, виділенням пунктів списків і т.д. Мова HTML дозволяє також включати в документи ілюстративну графіку, що може бути відображена

програмами перегляду, що основані на використанні графічного інтерфейсу користувача

Однією з найважливіших властивостей HTML є можливість включення в документ гіпертекстових посилань. Ці посилання дозволяють користувачу завантажити новий документ на свій комп'ютер, просто клацнувши вказівником миші в тій місці екрана, де розташоване посилання. Будь-який документ може містити посилання на інші документи. Областю документа, що використовується як посилання, може служити слово, група слів, графічне зображення чи навіть заданий фрагмент зображення.

Іншою не менш важливою властивістю HTML є форми. HTML-форми дають розробнику Web-інтерфейсу можливість змішування традиційних елементів, властивих друкованим сторінкам, з інтерфейсом користувача сучасної віконної системи. Механізм форм дозволяє розміщати серед форматowanego тексту з ілюстраціями кнопки, меню, що розкриваються, текстові поля й інше. Ці елементи інтерфейсу дають користувачам, що переглядають сторінки, можливість передачі відгуку, перетворюючи їх із простих глядачів в учасників, що мають право голосу. Головне обмеження HTML-форм пов'язано з тим, що Web-оглядачі змушені здійснювати нове з'єднання з сервером при кожному отриманні HTML-сторінки чи передачі даних форми.

HTML сам по собі не дозволяє створювати інтерактивні документи, тобто такий Web-інтерфейс є статичним. Існує ще одна технологія (DHTML), що дозволяє Web-сторінкам бути більш динамічними. DHTML (Dynamic HTML, динамічний HTML) дозволяє змінювати вид сторінки Web і її стиль після того, як документ завантажився.

Ключовим поняттям DHTML є шари. У звичайному HTML-документі, всі елементи так чи інакше впливають один на одного при формуванні його зовнішнього вигляду. Тобто абзац тексту "зміщує" наступні елементи на деяку відстань униз, картинка, вирівняна вправо, "змушує" текст обтікати її і так далі.

Всі елементи разом утворюють так званий HTML-потік чи потік документа (document flow). Шари у свою чергу – це частини сторінки, що не входять в основний HTML-потік. Для кожного з них створюється свій потік, що ніяк не впливає на основний. При цьому вміст шарів відображається, як правило, над (зверху) основним змістом сторінки. Шари можна пересувати і робити невидимими. Таким чином, за допомогою шарів можна створювати різні інтерактивні ефекти (наприклад, деревоподібне меню як у провіднику Windows).

DHTML заснована на технологіях CSS, JavaScript і HTML.

CSS

CSS (Cascading Style Sheets, каскадні таблиці стилів) – ще один стандарт, розроблений W3C. Існують CSS Level 1 і CSS Level 2. Останнім часом йде активне впровадження CSS Level 3. Але, на жаль не всі браузеры його коректно підтримують. Тому далі розглядатимемо CSS2.

CSS визначив правила опису стилів візуального відображення елементів HTML-документів. Основні особливості CSS:

- CSS — це мова, що дозволяє приєднувати стилі до будь-яких структурованих документів. На сьогодні такими є HTML-документи і XML-документи.
- CSS поширив поняття стилю відображення на друкувальні пристрої, синтезатори мови й інші пристрої відображення документів.

Ціль створення каскадних таблиць стилів полягала в тому, щоб відокремити структуру документа (описану мовою HTML чи XML) від правил його відображення на різних пристроях (що задаються таблицями CSS). За допомогою стилів можна визначити властивості елементів (текст, гіпертекстове посилання, абзац, заголовок, рисунок, таблиця й інші) у HTML-документі.

Існує три способи завдання стилів у HTML-документі: зовнішні таблиці стилів, внутрішні таблиці стилів і таблиці стилів елементів.

JavaScript

Мова JavaScript – інтерпретуєма об'єктно-базована мова високого рівня, що дозволяє писати програми, які виконуються в середовищі Web-оглядача. В даний час JavaScript є найбільш широко розповсюдженою мовою в Web для написання сценаріїв, що виконуються на стороні клієнта.

JavaScript дозволяє вбудовувати команди в HTML-документи. Команди JavaScript можуть виконуватися при завантаженні документа, при заповненні полів форм, при наведенні вказівника миші на елемент документа й інше. Мова JavaScript має досить широкі можливості. Вона не дозволяє працювати на рівні машинних кодів, однак дозволяє отримати доступ до багатьох можливостей Web-оглядачів, HTML-документів, а іноді й операційної системи, у якій працює Web-оглядач. На відміну від Java чи C, програми на JavaScript не потрібно компілювати, а оглядачу не потрібно завантажувати віртуальну машину для виконання програмного коду.

Одна з причин, згідно якої Web-розробники використовують JavaScript, – можливість виконання на стороні клієнта багатьох функцій, що раніше виконувалися винятково на стороні сервера. Кращим прикладом є перевірка HTML-форм. За допомогою JavaScript елементи форми можна перевірити до того, як користувач передасть інформацію Web-серверу. Це приводить до зменшення обсягу транзакцій HTTP.

JavaScript широко використовується для створення DHTML-документів. JavaScript дозволяє переміщатися по дереву (ієрархії) елементів HTML-документа, змінювати його властивості і багато чого іншого.

Крім JavaScript існує ще одна мова для вирішення аналогічних задач – VBScript. Однак вона, на відміну від JavaScript, не отримала такого ж широкого поширення.

1.2.2 Технології прикладного рівня

Web-сервер

Web-сервер (HTTP-сервер) – програма, що сприймає запити клієнтів (наприклад, Web-оглядачів), які приходять, як правило, на стандартний порт TCP/IP (порт 80), і виконує в найпростішому випадку пересилання HTML-документів клієнту. Web-сервер є програмою, яка дозволяє службі Internet, що називається WWW (World Wide Web, Всесвітня інформаційна мережа), знаходити і використовувати гіпертекстові і графічні документи на різних серверах.

Фактично Web-сервер передає інформацію користувачу, який знаходиться на віддаленому комп'ютері. Користувач переглядає інформацію за допомогою Web-оглядача (MS Internet Explorer, Netscape Navigator і інші) і відсилає нові запити Web-серверу.

Web-сервер як правило має вбудовану систему аутентифікації по логіну та пароллю, яку можна використовувати для обмеження доступу до ресурсів сервера.

Найбільш розповсюджені Web-сервера: Apache Server, MS IIS, Netscape Enterprise Server, NGINX сервер.

CGI

CGI (Common Gateway Interface) – це набір правил, згідно з якими програми на стороні сервера можуть через Web-сервер пересилати дані клієнтам. CGI є перевіреним і універсальним способом створення розподілених систем у Internet. Головна задача, яку він вирішує, – це забезпечення викликів віддалених процедур між клієнтом і сервером з використанням сокетів TCP/IP.

Крім CGI існують ще кілька подібних технологій – ISAPI/NSAPI, Servlet і JSP. Ці чотири технології призначені для вирішення приблизно тих самих задач і концептуально дуже схожі. Всі вони дозволяють розширити можливості Web-серверів шляхом розробки програм, зареєстрованих на цьому сервері.

Servlet i JSP

Технологія Servlet була розроблена компанією Sun Microsystems, і була призначена для того щоб серверна частина отримуючи запит від клієнта у якості відповіді повернула згенеровану HTML сторінку яка і відображає дані. Servlet – це спеціальний Java клас, який реалізує необхідні API для отримання запитів від WEB клієнта та після обробки надання результату. Сам по собі Servlet не чого робити не вміє він є частиною системи яка знаходиться у контейнері(Java web сервер) який вміє отримати запит і передати його необхідному додатку. Технологія JSP (JSPx) є „надбудовою” над сервлетами і призначена для більш наглядного конструювання інтерфейсу. JSP є наслідником Servlet і являє собою комбінацію Java коду та HTML, на зразок PHP.

Java web сервери

Існує декілька Java web серверів зокрема такі, WildFly, Jetty, Tomcat, Glassfish, Payara та ін. Майже усі вони виконують одні і ті ж самі задачі:

- трансляція JSP сторінок у сервлети;
- компіляція JSP сторінок;
- зв'язок з базою даних
- обмін даними між програмою та клієнтом;
- обмін даними між програмою та БД;
- автоматизація аутентифікації та авторизації користувачів

1.2.3 Технології рівня даних

Мова SQL

SQL – це скорочена назва структурованої мови запитів (Structured Query Language). SQL – це мова програмування, яка використовується для виконання запитів і модифікації даних, а також для управління СУБД (системами управління базами даних). SQL працює тільки з базами даних одного певного типу, які називаються реляційними.

Стандарти SQL затверджені ANSI і ISO. Хоча в кожній СУБД використовується власний діалект SQL, більшість цих діалектів задовольняють стандарту ANSI SQL. Таким чином, для сумісності програмного забезпечення з різними СУБД, необхідно строго дотримуватися стандарту ANSI SQL.

Мова SQL містить оператори, які підрозділяються на дві категорії: мова опису даних (data definition language, DDL) і мова маніпулювання даними (data manipulation language, DML). DDL використовується для створення і управління такими об'єктами, як бази даних, таблиці і представлення. Оператори DDL містять команди CREATE, ALTER і DROP для кожного з об'єктів. DML використовується для управління даними в об'єктах баз даних за допомогою таких операторів, як SELECT, INSERT, UPDATE і DELETE.

Сервер баз даних

Сервер баз даних – це окрема програма, яка виконуванується як окремий процес (сервіс). Сервер передає обрану з бази інформацію по каналу клієнту. Саме сервер працює з даними, піклується про їхнє розміщення на диску.

Найрозповсюджені сервера баз даних: Oracle, MySQL, PostgreSQL.

EJB

EJB – ще одна технологія від Oracle (Sun Microsystems) яка призначена для реалізації Моделі у Java програмі. Вона реалізує зв'язок з БД незалежно від реалізації, та представлення у вигляді Java об'єктів. EJB поділяється на дві складові

- EntityBeans (класи які є об'єктним представленням інформації з БД)
- SessionBeans (класи та інтерфейси які контролюють, змінюють, і т.д. EntityBeans)

1.2.4 Динаміка роботи інформаційної системи

Маючи загальне уявлення про технології, які використовуються для розробки системи, можна описати динаміку роботи системи:

- Користувач запускає Web-оглядач на віддаленому комп'ютері-клієнті і вводить адресу Web-сервера інформаційної системи.
- Web-оглядач надсилає запит Web-серверу.
- Web-сервер надсилає запит Java-серверу.
- Java-сервер передає запит відповідному додатку (системі).
- Система отримавши запит, передає його відповідному сервлету, або JSP.
- Servlet(JSP) обробляючи запит звертається (якщо необхідно) до SessionBean
- SessionBean звертається до Java-сервера з проханням надати зв'язок з необхідним сервером БД та проводить аутентифікацію користувача по логіну і пароллю та вибирає потрібну базу.
- Отримавши зв'язок SessionBean проводить необхідний запит і отримавши результати повертає їх контролеру (Servlet або JSP)
- Контролер отримав результат генерує необхідну HTML сторінку та повертає її і передає її Web-серверу.
- Web-сервер додає HTTP-заголовок до отриманого документа і відправляє його Web-оглядачу.
- Web-оглядач приймає HTML-документ і відображає його на екрані користувача.
- Користувач отримує Web-інтерфейс і необхідну інформацію. Він може надіслати новий запит Web-серверу.

Таким чином, вимальовується схема взаємодії технологій (рисунок 1.2)

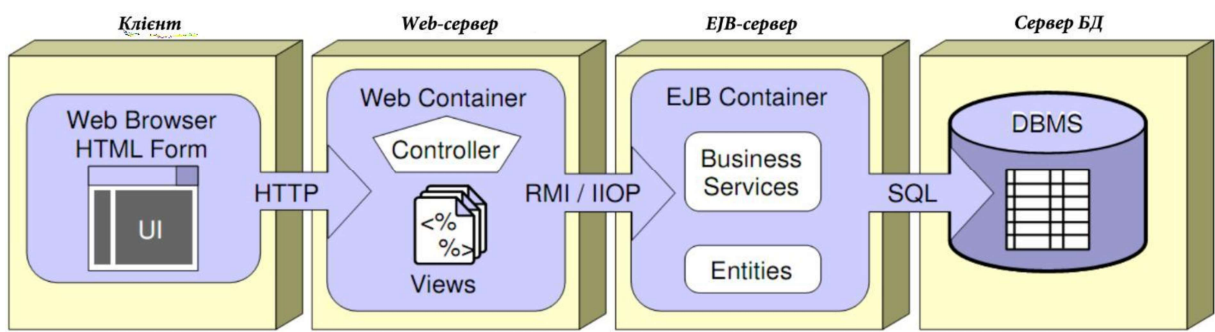


Рисунок 1.2 – Схема взаємодії технологій інформаційної системи

1.3 Формування вимог до системи управління мікрокліматом

Об'єктом автоматизації є мій власний будинок. Мій будинок має потребу у створенні єдиної системи управління кліматом будівлі.

1.3.2 Цілі, критерії й обмеження створення інформаційної системи

Метою розробки інформаційної системи є поліпшення умов життя шляхом автоматизації управління мікрокліматом.

Основні критерії створення інформаційної системи:

- Соціальний критерій. Інформаційна система розрахована на користувачів з високим рівнем володіння комп'ютером, тому достатньо простого інтерфейсу з необхідними функціями

- Економічний критерій. Автоматизована система створюється безкоштовно. Вартість системи буде складатися з витрат на придбання необхідного програмного, технічного й іншого видів забезпечення.

- Критерій мобільності. Користувач повинен мати можливість працювати із системою з будь-якого комп'ютера локальної мережі

- Критерій безпеки. Система повинна бути безпечною з погляду збереження й обробки даних. Втрата даних може привести до серйозних збоїв у роботі системи. Необхідний захист системи від несанкціонованого втручання.

1.3.3 Функції та задачі інформаційної системи

Задачею інформаційної системи, що створюється, є автоматизація наступних функцій:

- Управління системою опалювання
- Управління системою кондиціонування
- Управління вентиляванням
- Управління освітленням
- Підтримка необхідних кліматичних умов у приміщенні

Для створення інформаційної системи необхідно розробити базу даних для збереження інформації про кімнати, відповідальних за температуру в них. Також повинно бути передбачене апаратно-програмне забезпечення для збору інформації про мікроклімат в приміщенні і необхідні бути передбачені засоби для адміністрування системи

1.3.4 Висновки і пропозиції

Таким чином, необхідно розробити автоматизовану інформаційну систему, що дозволить підвищити якість життя в будинку, а також зменшить витрати на електроенергію. Система є глобальної інформаційною системою і вирішує задачі, що пов'язані з управлінням мікроклімату. Інформаційна система повинна включати базу даних, програмне забезпечення жителя, а також відповідне організаційне, технічне та інші види забезпечення.

1.4 Розробка концепції інформаційної системи

1.4.1 Опис результатів вивчення об'єкта автоматизації

Інформаційна система розробляється як система управління кліматом житловому будинку. Для створення системи потрібна розробка бази даних інтерфейсу користувача, а також розробка механізму отримання і передачі інформації про кліматичні умови із приміщень.

1.4.2 Аналіз вимог до інформаційної системи

Крім функціональних вимог до інформаційної системи, необхідно врахувати вимоги до системи в цілому:

- Простий і зручний інтерфейс користувача.
- Доступ до системи через локальну мережу з будь-якого комп'ютера.
- Багатокористувачева робота.
- Висока масштабуємість системи.
- Висока продуктивність.
- Надійність.
- Конфіденційність інформації.

Аналізуючи зазначені вище вимоги, можна зробити висновок, що інформаційна система повинна мати/підтримувати:

- Web-інтерфейс.
- Багаторівневу архітектуру.
- Сучасні стандарти в області інформаційних технологій (протоколи передачі даних, стандарти збереження даних і т.д.).
- Сучасні серверні платформи і програмне забезпечення.
- Сучасні рішення в області інформаційних технологій по розробці розподілених систем.
- Ефективні методи доступу до даних.
- Забезпечити простоту модернізації, можливість повторного використання програмного забезпечення (коду).

1.4.3 Вибір логічної архітектури програмного забезпечення інформаційної системи

Логічна архітектура програмного забезпечення інформаційної системи являє собою концептуальний опис структури програмного забезпечення системи. Кожна система має три рівні: рівень користувача, прикладний і рівень даних. Крім того, в усіх системах мається презентаційне програмне забезпечення (ПЗ), ПЗ обробки бізнес-правил і даних, а також ПЗ, що відповідає за збереження інформації. Відрізняються ж архітектури тільки організацією програмного забезпечення (програмного коду), індивідуальної для конкретної системи.

Багато сучасних інформаційних систем побудовані на базі дворівневої архітектури «клієнт/сервер». При цьому клієнтське програмне забезпечення відповідає за обробку і відображення даних. Самі ж дані централізовано зберігаються на серверах, до яких підключаються клієнти. Часто час використання системи обмежується тривалістю такого з'єднання.

Клієнт-серверні системи добре працюють тільки в контрольованих середовищах, коли число користувачів передбачуване (це необхідно для виділення ресурсів). Однак ця архітектура стає неефективною, при великому числі користувачів або якщо їхнє число невідомо. Крім цього, масштабованість таких систем невисока, тому що кількість безпосередніх з'єднань із сервером обмежена. Невеликі і можливості повторного застосування, оскільки користувачі прив'язані до визначеного формату бази даних. А тому що клієнтське програмне забезпечення містить блок обробки інформації, загальний обсяг програмного забезпечення досить великий (Такий тип клієнтських програм іноді називають «товстим» клієнтом). У цьому випадку при зміні алгоритмів обробки даних приходиться встановлювати нове програмне забезпечення на кожен клієнтський комп'ютер.

Можна отримати невелике поліпшення шляхом переміщення прикладних алгоритмів і блоків обробки даних на сервери даних (наприклад, за

допомогою збережених процедур сервера баз даних). Таку архітектуру іноді називають 2,5-рівневою. Масштабуємість подібних систем трохи краща, але все рівно мала для виконання вимог інформаційної системи. Крім того, можливість повторного використання програмного забезпечення (коду) залишається на невисокому рівні.

Масштабуємість і ступінь повторного використання можна помітно поліпшити, додавши в архітектуру системи третій рівень. У такій багаторівневій архітектурі (її також називають N-рівневою) усі рівні – рівень користувача, прикладний і рівень даних – логічно розділені (рисунок 1.3). Багаторівнева архітектура задовольняє вимогам до інформаційної системи, що розробляється.

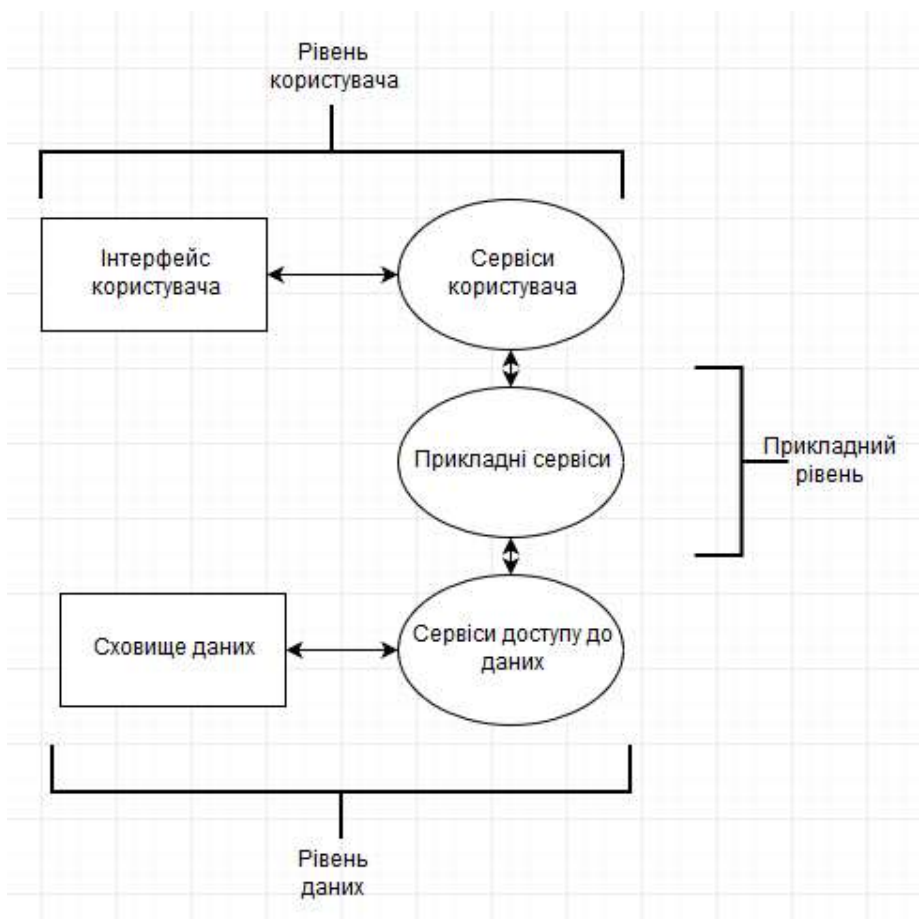


Рисунок 1.3 - Логічна архітектура програмного забезпечення інформаційної системи

Опишемо функції кожного рівня:

- Рівень користувача: відображає дані і дозволяє користувачу редагувати їх. Існує два основних типи інтерфейсу: віконний (реалізується засобами підсистеми інтерфейсу користувача операційної системи) і на основі Web. Web-інтерфейси засновані на HTML, DHTML, XHTML чи XML, у результаті чого вони можуть відображатися будь-яким Web-оглядачем на будь-якій платформі.

- Прикладний рівень: тут реалізовані бізнес-правила й обмеження на дані (перевірка коректності даних). І хоча його сервіси використовуються презентаційним рівнем, він не прив'язаний до якого-небудь клієнта – сервіси прикладного рівня доступні будь-якому клієнту. Бізнес-правила виражаються у формі прикладних алгоритмів, корпоративних правил і т.д. Обмеження на дані гарантують точність і цілісність інформації, що зберігається. Бізнес-правила звичайно реалізуються окремим модулем на централізованому сервері, що дає можливість доступу до нього відразу декільком клієнтам.

- Рівень даних: прикладний рівень не має інформації про те, як і де зберігаються дані, які він обробляє. У цьому питанні він покладається на сервіси доступу до даних, що виконують всю роботу з отримання і передачі даних. Сервіси доступу до даних також реалізуються у виді ізольованих модулів, які знають про місце збереження інформації. Кожен модуль доступу до даних, як правило, відповідає і за цілісність сховища (наприклад, реляційної бази даних). Для багаторівневих програм у якості сховища інформації підходять сервера баз даних (MS SQL Server, Oracle, Sybase, MySQL, PostgreSQL і ін.), необхідні для обслуговування даних у таблицях і високопродуктивної виборки інформації.

1.4.4 Вибір фізичної архітектури програмного забезпечення інформаційної системи

Виходячи з логічної архітектури програмного забезпечення, можна розробити його фізичну архітектуру. Фізична архітектура являє собою опис

структури програмного забезпечення системи з погляду наявних у системі фізичних компонентів: серверів і клієнтських комп'ютерів.

Відповідно до описаної логічної архітектури, можна одержати дві фізичні архітектури, які так чи інакше задовольняють вимогам до системи. Перша фізична архітектура є триланковою (рисунок 1.4), друга – чотириланковою (рисунок 1.5).

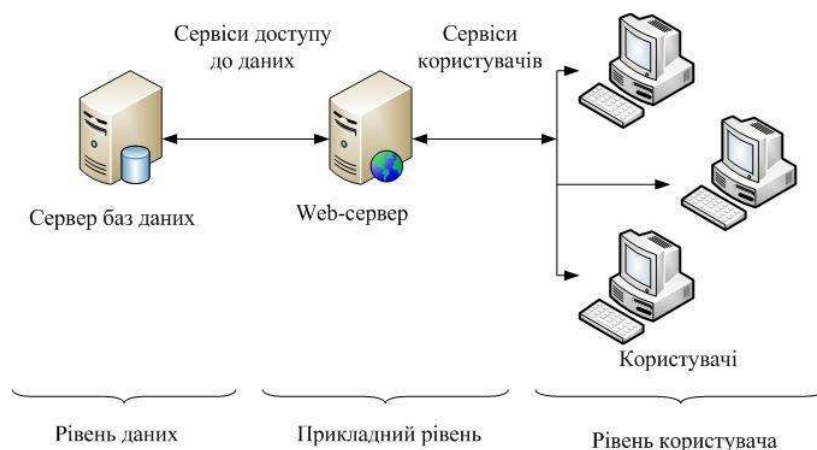


Рисунок 1.4 – Триланкова фізична архітектура програмного забезпечення інформаційної системи

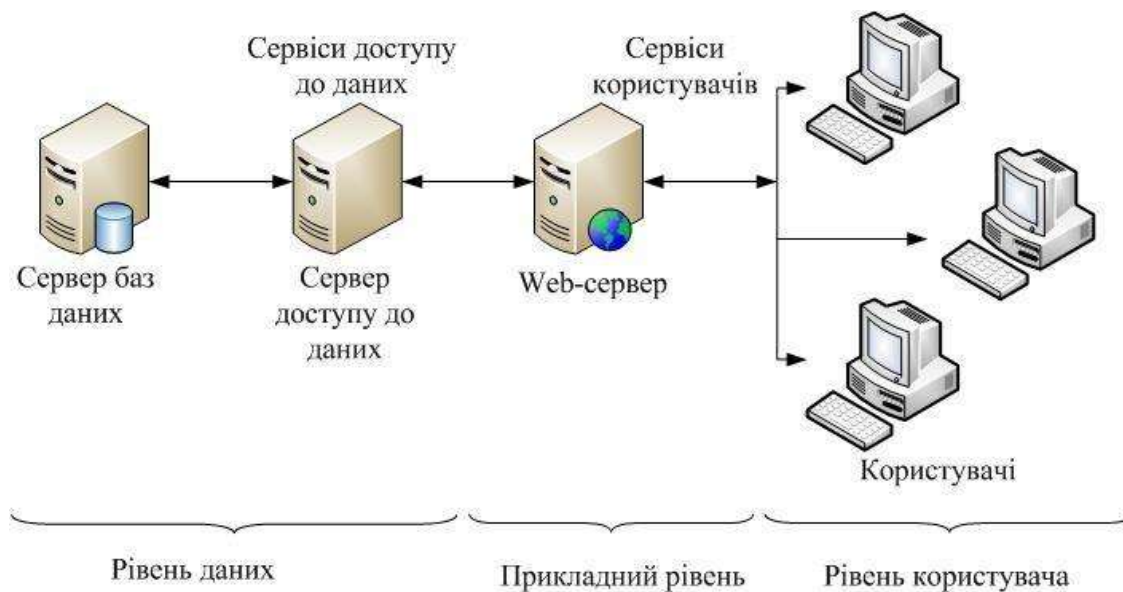


Рисунок 1.5 – Чотириланкова фізична архітектура програмного забезпечення інформаційної системи

Порівняємо отримані фізичні архітектури для інформаційної системи.

Переваги триланкової архітектури над чотириланковою:

- Краща продуктивність при невеликому числі користувачів.
- Порівняльна простота адміністрування системи.

Переваги чотириланкової архітектури над триланковою:

- Краща продуктивність при великому числі користувачів.
- Краща масштабуємість.
- Краща надійність.
- Кращі можливості по модернізації і повторному використанню.

Таким чином, чотириланкова архітектура є більш перспективною.

Вона в більшій мері задовольняє вимогам до інформаційної системи.

1.4.5 Вибір технологій для розробки інформаційної системи

Відповідно до обраної фізичної архітектури для інформаційної системи управління роботами, система є розподіленою і має чотири ланки: сервер баз даних, сервер доступу до даних, Web-сервер і клієнт.

Існують наступні технології для створення розподілених систем:

- DCOM/COM+, REST, CORBA (об'єктно-орієнтовані технології).
- CGI, ISAPI/NSAPI, Servlets і інші.

Всі ці технології тим чи іншим способом організують виклик віддалених процедур (методів) чи його аналог.

Взаємодію між клієнтами і Web-сервером раціонально організувати за допомогою CGI, ISAPI/NSAPI і/або Servlets, тому що ці технології орієнтовані на Web-інтерфейс. Використання DCOM/COM+, CORBA чи RMI у цьому випадку серйозно збільшило б вимоги до програмного забезпечення клієнтських машин, що суперечить вимогам до інформаційної системи (доступ до системи через Internet з будь-якого комп'ютера глобальної мережі). Усі зазначені технології (CGI, ISAPI/NSAPI, Servlets) вирішують однакові задачі і мають приблизно однакові характеристики по продуктивності. Servlets є найгібкішою та одною з найсучасніших технологій, а також реалізується на мові високого рівня Java яка не залежить від операційної системи тому і обираємо цю технологію.

Для взаємодії серверів та доступу до БД потрібно застосовувати більш розвинуті технології.

Найбільш перспективною й універсальною технологією є REST.

REST - підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад, HTML, XML, JSON). Будь-який REST протокол (HTTP в тому числі) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь».

Антиподом REST є підхід, заснований на RPC. Підхід RPC дозволяє використовувати невелику кількість мережевих ресурсів з великою кількістю методів і складним протоколом. При підході REST кількість методів і складність протоколу суворо обмежені, що призводить до того, що кількість окремих ресурсів має бути великою.

REST, як і кожен архітектурний стиль відповідає ряду архітектурних обмежень. Це гібридний стиль який успадковує обмеження з інших архітектурних стилів.

Перша архітектура від якої він успадковує обмеження — це клієнт-серверна архітектура. Її обмеження вимагає розділення відповідальності між компонентами, які займаються зберіганням та оновленням даних (сервером), і тими компонентами, які займаються відображенням даних на інтерфейсі користувача та реагування на дії з цим інтерфейсом (клієнтом). Таке розділення дозволяє компонентам еволюціонувати незалежно.

Наступним обмеженням є те, що взаємодії між сервером та клієнтом не мають стану, тобто кожен запит містить всю необхідну інформацію для його обробки, і не покладається на те, що сервер знає щось з попереднього запиту.

Відсутність стану не означає що стану немає. Відсутність стану означає, що сервер не знає про стан клієнта. Коли клієнт, наприклад, запитує головну сторінку сайту, сервер відповідає на запитання і забуває про клієнта. Клієнт може залишити сторінку відкритою протягом кількох років, перш ніж натиснути посилання, і тоді сервер відповість на інший запит. Тим часом сервер може відповідати на запити інших клієнтів, або нічого не робити — для клієнта це не має значення.

Таким чином, наприклад дані про стан сесії (користувача, який аутентифікувався) зберігаються на клієнті, і передаються з кожним запитом. Це покращує масштабованість, бо сервер після закінчення обробки запиту може звільнити всі ресурси, задіяні для цієї операції, без жодного ризику втратити цінну інформацію. Також спрощується моніторинг і зневаження, бо для того аби розібратись, що відбувається в певному запиті, досить подивитись лише на той один запит. Збільшується надійність, бо помилка в одному запиті не зачіпає інші.

Мінусом цього обмеження є те, що знижується продуктивність через те, що в кожен запит тепер доводиться додавати дані сесії з клієнта. Також збереження стану на різних клієнтах важче підтримувати, бо реалізації клієнтів можуть відрізнятись, тоді як середовище сервера повністю під контролем розробника.

Додатковим обмеженням стилю REST є те, що системи, написані в цьому стилі, повинні підтримувати кешування, тобто дані, які передаються сервером, повинні містити інформацію про те, чи можна їх кешувати, і якщо можна, то як довго. Це дозволяє збільшувати продуктивність, уникаючи зайвих запитів, але також зменшує надійність системи, через те, що дані в кеші можуть бути застарілими.

Всі компоненти в архітектурі REST підтримують однорідний інтерфейс. Це зменшує зв'язність між компонентами і сервісами які вони надають і дозволяє нескладно змінювати компоненти при потребі.

Наступним обмеженням для REST є поділ на шари абстракції. Кожен компонент потрапляє в якийсь шар, і спілкується лише з компонентами в шарі під ним або в шарі над ним. Обмеження знання системи одним шаром зменшує складність компонентів. Саме тому REST є однією з найрозповсюджених технологій.

Основний недолік DCOM/COM+ пов'язаний з необхідністю використовувати Windows-платформу. DCOM/COM+ на практиці довела свою придатність для розробки інформаційних систем. Це безкоштовна технологія і на ній засновано багато інших технологій (наприклад, OLE DB – найефективніший /найпродуктивніший доступ до будь-яких даних). Ця технологія цілком задовольняє вимогам до інформаційної системи, перевірена на практиці, є безкоштовною, тому раціонально використовувати її для взаємодії серверів. Так як ми обрали Servlets то логічно обрати технологію яка реалізується на платформі Java тому наш вибір це EJB, крім цього EJB має багато переваг серед своїх конкурентів.

Як говорилося раніше, на рівні користувача необхідно використовувати Web-інтерфейс (HTML/DHTML, CSS, JavaScript, DOM, Java-аплети). Клієнтське програмне забезпечення – Web-оглядач.

1.4.6 Розробка варіантів концепції інформаційної системи

Виходячи з аналізу вимог до інформаційної системи, її функціональних характеристик, фізичної архітектури і технологій для розробки, можна розробити концепцію інформаційної системи. Основними компонентами системи є:

- серверна платформа і серверне програмне забезпечення;
- система управління базами даних;
- клієнтська платформа і клієнтське програмне забезпечення;

- засоби розробки;
- мережне середовище;
- технічні засоби.

Від вибору перерахованих компонентів залежить ефективність інформаційної системи. Розглянемо кожен компонент інформаційної системи.

Серверна платформа і серверне програмне забезпечення

На Web-сервері повинен бути встановлений HTTP-сервер. Найрозповсюдженими HTTP-серверами, що задовольняють зазначеним вимогам і мають високу продуктивність, є NGINX. На Web-сервері повинна бути встановлена ОС яка підтримує цей сервер.

Так як використовуються технології організовані на Java на комп'ютері, то операційна система на сервері доступу до БД та управління системами опалювання, кондиціонування та вентиляції оптимально буде ОС Debian

Також потрібний Java WEB сервер встановлений на сервері будь який з:

- JBoss
- Payara
- GlassFish
- Tomcat

На сервері баз даних повинна бути встановлена серверна система управління базами даних (СУБД).

Система управління базами даних

Необхідна сучасна серверна реляційна СУБД, що має високу продуктивність, здатна працювати з великими обсягами інформації і підтримує стандарт SQL-92. Крім того, СУБД повинна працювати під управлінням Windows. Серверними СУБД, що задовольняють зазначеним вимогам, на сьогодні є:

- MS SQL Server.
- MySQL.
- PostgreSQL .

- Sybase.
- Oracle (Standard Edition чи Enterprise Edition).

Клієнтська платформа і клієнтське програмне забезпечення

На клієнтських машинах може використовуватися будь-яка операційна система з можливістю встановлення віддаленого мережного з'єднання і графічним інтерфейсом. Операційні системи, що рекомендуються – MS Windows, Linux, тощо.

Програмне забезпечення, з яким буде працювати користувач, являє собою Web-оглядач з підтримкою HTML 5.0, JavaScript, CSS 3 (наприклад, Firefox, Chrome, Opera ...).

Мережне середовище

Будинок має локальну комп'ютерну мережу Ethernet та WIFI, яку можна використовувати як мережне середовище інформаційної системи. Можливо, потрібно буде внести зміни в конфігурацію існуючої комп'ютерної мережі. Основна вимога до мережі – швидкість передачі даних не менше 10 Мбіт/с.

Технічні засоби

Система розробляється для досить невеликого будинку, тому у вигляді серверу можна застосувати старий ноутбук Asus Eee PC 1001PX.

Технічні засоби клієнтських машин повинні відповідати вимогам операційної системи, що використовується. Конфігурація комп'ютера користувача може бути наступною:

- Мікропроцесор Intel Core i5.
- Оперативна пам'ять RAM 8Гб.
- Жорсткий диск з 40 Гб вільного місця і більше.
- Мережна карта Ethernet (100 Мбіт).

Засоби розробки

У якості засобу розробки планується використання об'єктно-орієнтованої мови програмування Java – OpenJDK 8. Це обґрунтовується, по-перше, незалежністю від платформи та, по-друге, безкоштовністю.

1.4.7 Вимоги, що впливають на якість інформаційної системи

Якість інформаційної системи і якість програмного забезпечення не піддаються точному визначенню і виміру. На даний момент, не існує ні метрології якості інформаційних систем, ні реальних методів оцінки якості програмного забезпечення, ні методів оцінки якості процесу проектування, розробки і впровадження.

Проте, задача поставлена і вимагає вирішення. Складемо список вимог (критеріїв), що можуть впливати на якість інформаційної системи (частина з них взята зі стандарту ISO-9126):

- Здатність підвищити ефективність працівників.
- Функціональність.
- Відповідність призначенню.
- Точність.
- Повнота інформації.
- Здатність взаємодіяти із середовищем.
- Відповідність нормам.
- Безпека (захист від взлому даних і інших злочинних дій).
- Надійність.
- Зрілість ("обкатаність").
- Стійкість до відмов.
- Стійкість до помилок користувача.
- Здатність відновлюватися після збоїв.
- Масштабуємість.
- Придатність до використання.
- Зрозумілість.

- Вивчаємість.
- Зручність і простота в роботі.
- Ефективність.
- Продуктивність і час відгуку.
- Споживання ресурсів.
- Супроводжуємість.
- Аналізуємість (діагностика причин помилок і зіставлення з первісним кодом).
- Придатність до змін.
- Стабільність.
- Тестуємість.

Потрібно відзначити, що виконання всіх зазначених вимог ще не гарантує якість інформаційної системи. Також, невідповідність системи якій-небудь вимозі не говорить про те, що система неякісна. Як уже говорилося вище, не існує формальних методів оцінки якості інформаційної системи.

Пропонується наступна конфігурація системи:

- Asus Eee PC 1001PX
 - процесор Intel Atom N450 з частотою 1666 МГц
 - 1 Гб оперативної пам'яті
 - Ethernet (RJ-45) 100 Мбит/с
 - Wi-Fi 802.11n
- Esp8266 NodeMCU в необхідній кількості з датчиками температури та вологості DHT-22 та датчиком вуглекислого газу MQ135 , а також датчиками переміщення(звичайний PIR датчик) і освітленности GY-2561, та радіомодулем з частотою 433 МГц(для управління вимикачами світла)

Висновок

У результаті проведеного аналізу існуючих рішень у сфері систем «Розумне помешкання» встановлено, що попри значну кількість комерційних та відкритих платформ, більшість із них характеризуються низкою обмежень:

фрагментованістю функціональності, закритістю протоколів, недостатньою гнучкістю інтеграції та обмеженими можливостями віддаленого керування в умовах нестабільних мережових з'єднань. Це зумовлює потребу у створенні інформаційної системи, яка забезпечує надійний, безпечний і універсальний доступ до компонентів «Розумного помешкання» незалежно від місця перебування користувача.

На основі виявлених недоліків сформовано вимоги до майбутньої системи, що охоплюють функціональні, технічні, інформаційні та експлуатаційні аспекти. Серед ключових вимог визначено: підтримку віддаленого доступу через захищені канали зв'язку, можливість масштабування та інтеграції з різнорідними пристроями, забезпечення високої доступності сервісів, а також інтуїтивно зрозумілий інтерфейс для кінцевого користувача.

Сформована концепція інформаційної системи передбачає побудову багаторівневої архітектури, у якій модулі збору даних, керування пристроями та віддаленої взаємодії функціонують узгоджено, забезпечуючи цілісність та оперативність роботи. Постановка задачі визначила основні напрями подальшої розробки: створення механізмів безпечної аутентифікації, реалізацію протоколів взаємодії з обладнанням, розробку інтерфейсів віддаленого керування та забезпечення стійкості системи до зовнішніх впливів.

Таким чином, проведений аналіз і сформульовані вимоги стали підґрунтям для розробки концепції інформаційної системи віддаленого доступу до «Розумного помешкання», що відповідає сучасним технічним тенденціям і потребам користувачів та створює основу для подальшої реалізації програмно-апаратного комплексу.

2 Розробка проектних рішень по системі

2.1 Загальносистемні проектні рішення

2.1.1 Організаційна і функціональна структура.

У процесі свого функціонування система взаємодіє з системою управління користувачами з різних підрозділів для виконання ними різних функцій. Схема цієї взаємодії, побудована в нотації варіантів використання системи (use case) показана на рисунку 2.1.

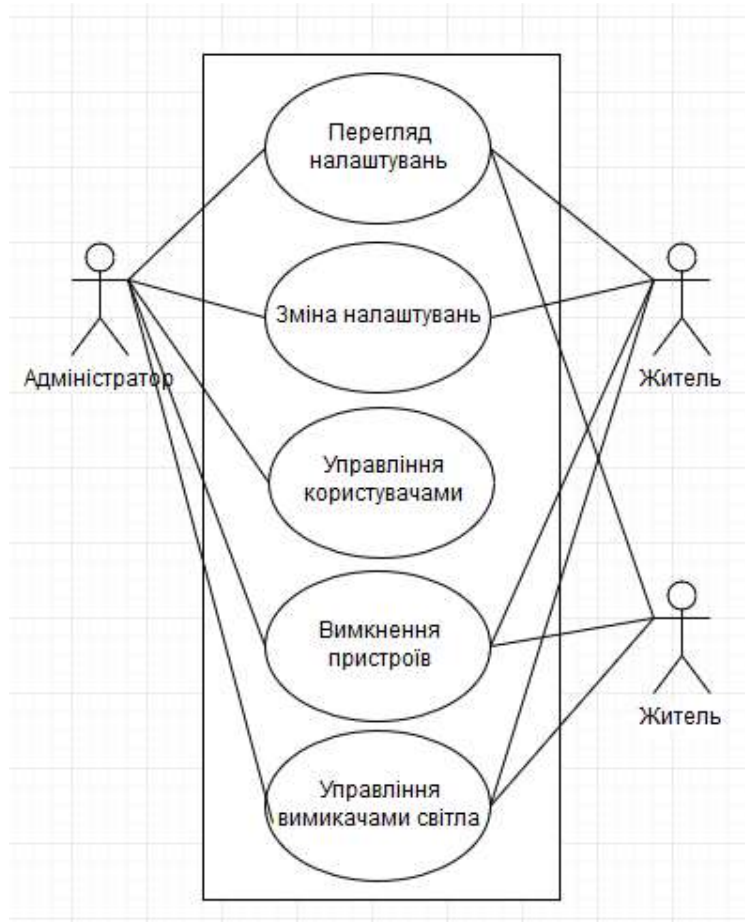


Рисунок 2.1. Схема організаційно-функціональної структури

При роботі системи, протягом дня, жителі будинку використовують наступні функції

Користувач – перегляд та зміна заданих параметрів, вимкнення функцій

Адміністратор – перегляд та управління (зміна, видалення, створення нового) користувачами.

2.1.2 Опис функцій, що автоматизуються.

Задачею системи, що створюється, є автоматизація наступних функцій користувача:

- Моніторинг клімату приміщень
- Управління кліматорегулюючим обладнанням
- Вимкнення світла в приміщеннях
- Моніторинг датчиків руху
- Моніторинг датчиків освітлення
- Усі інші вимоги не є головними і можуть змінюватися під час розробка або експлуатації

Для створення системи необхідно розробити єдину базу даних для збереження інформації про приміщення та їх жителів. Також повинно бути передбачене програмне забезпечення для адміністрування системи.

2.1.3 Опис постановки задачі

На підставі аналізу вимог і обмежень, а також попередніх концепцій реалізації системи, сформулюємо опис постановки задачі. Необхідно розробити систему управління обладнанням, що буде задовольняти перерахованим вище функціональним вимогам. Необхідно також забезпечити надійність збереження і безпеку доступу до інформації Системи. Система повинна обслуговувати одночасну безперебійну роботу багатьом користувачів, причому зміни, внесені одними користувачами повинні бути доступні всім іншим користувачам негайно.

Програмне забезпечення повинне бути побудоване на платформі Debian jessie/nginx/MySQL/Payara/Java/IntelliJ IDEA і реалізовувати графічний інтуїтивно зрозумілий WEB інтерфейс користувача.

2.1.4 Загальний опис системи.

Система являє собою інструмент управління процесом розробки програмного забезпечення, побудований на основі мережі підприємства яке

спеціалізується на розробці програмних продуктів. Основними функціональними характеристиками системи є:

- ведення загальної інформаційної бази з розподіленим доступом;
- авторизований доступ до даних на основі прав доступу;
- відображення змін, внесених з будь-якого робочого місця;
- можливість створення і редагування завдань і оперативного контролю ходу їхнього виконання;

Система має звичний графічний інтерфейс користувача, що знижує витрати на навчання персоналу, і побудована з максимальним використанням існуючого на підприємстві в даний момент устаткування і програмного забезпечення.

2.1.5 Захист від несанкціонованого доступу

Захист системи від несанкціонованого доступу реалізовано на системному рівні. При спробі зміни налаштувань відбувається авторизація користувача. Доступ до даних і функцій дозволяється тільки при правильному введенні користувальницького імені і пароля.

Авторизація відбувається за рахунок вбудованого механізму авторизації серверу Payara, для реалізації цього механізму, програміст повинен лише реалізувати вікно авторизації, що містить форму для введення даних користувача. Крім того, адміністратор системи повинен внести права користувачів у відповідний конфігураційний файл, що використовується сервером Payara.

2.2 Рішення по організаційному забезпеченню

2.2.1 Організаційна структура

У ході впровадження і використання системи, її функціями охоплюються всі приміщення будинку. Схема організаційної структури при взаємодії із системою приведена на рисунку 2.2

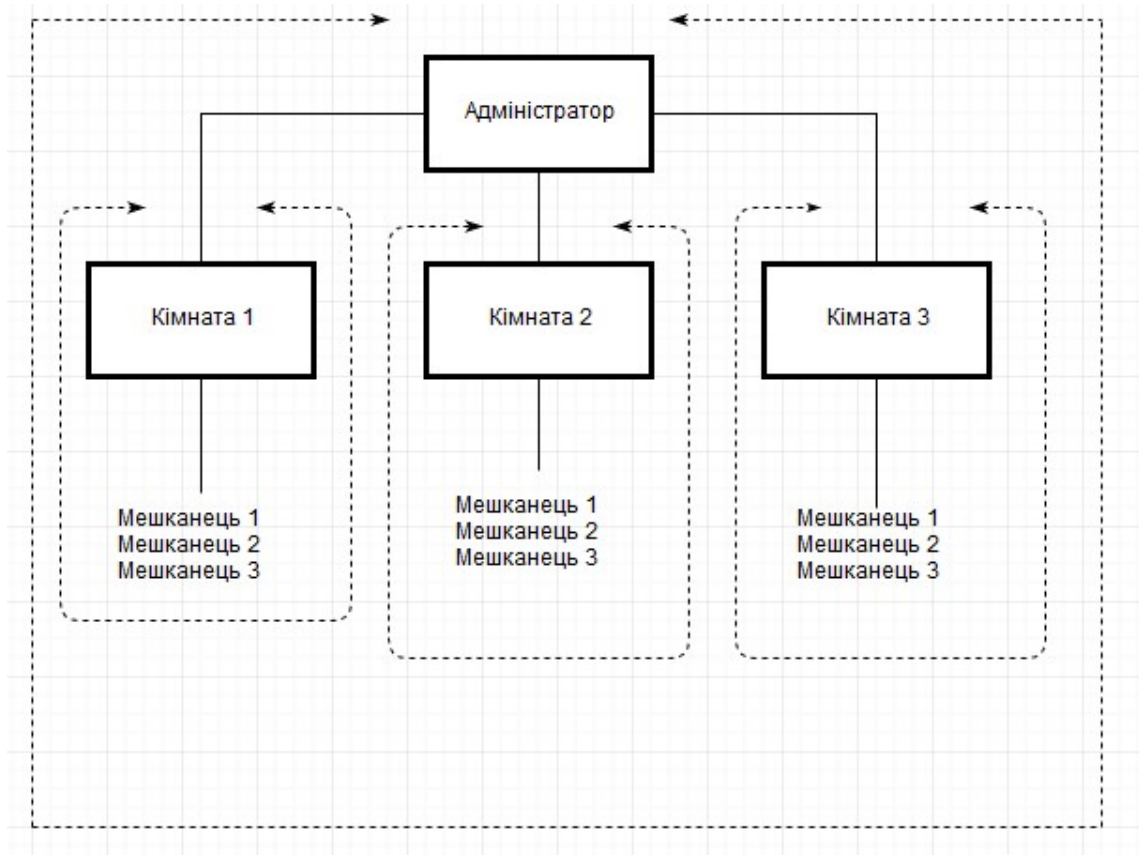


Рисунок 2.2. Структура житлового будинку

2.2.2 Інструкція користувача

Для нормального функціонування системи, всі операції, проведені в ній користувачами, повинні відповідати Інструкції користувача, приведеної в Додатку В.

2.2.3 Заходи щодо організаційного забезпечення

Ефективна робота системи вимагає організаційних заходів, цілями яких є підтримка даних про процеси, що відбуваються, в актуальному стані і

грамотній своєчасній реакції користувачів на інформацію, що їй надається системою:

- навчання користувачів роботі в системі згідно виконуваних функцій;
- мінімізація часу, що проходить між зміною версій документів і зміни їхнього стану (іншими словами, негайна реакція керівництва на зміни у виробничому процесі);
- обов'язкова участь у формуванні даних усіх суб'єктів підприємства; регулярний оперативний контроль зв'язаних процесів керівниками підрозділів і контроль виробничих циклів.

2.2.4 Організаційно-кадрові заходи

Для підтримки апаратних і програмних засобів у робочому стані, а також рішення поточних задач настроювання, установки і відновлення загальносистемного програмного й апаратного забезпечення у будинку повинна бути введена штатна одиниця адміністратора.

2.3 Рішення по інформаційному забезпеченню

2.3.1 Опис інформаційного забезпечення системи

Одну з основ інформаційного забезпечення системи складає визначення її внутрішнього стану, що буде відбито в моделі класів. На даному етапі нас цікавлять довгоживущі і постійні сутності бізнесів-процесів. Цей вид класів будемо називати класами-сутностями (чи класами предметної області).

Для створення моделі класів складемо словник даних, ґрунтуючись на сутностях, виявлених на етапі бізнесу-аналізу.

Житель – людина яка взаємодіє з системою

Дія – зміна параметрів

На даному етапі проектування сутності словника даних стають кандидатами для класів проектованої системи. Модель бізнес-сутностей Системи показана на рисунку 2.3.

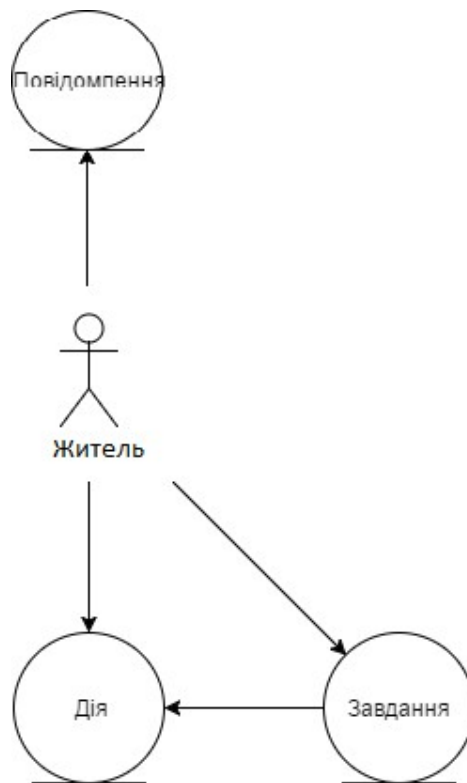


Рисунок 2.3. - Діаграма бізнес-сутностей системи.

На даному етапі проектування сутності словника даних стають кандидатами для класів проектованої системи.

2.4 Рішення по технічному забезпеченню

2.4.1 Схема автоматизації

Раціональний вибір технічних засобів, необхідних для впровадження автоматизованої системи буде значною мірою визначати загальний чи успіх невдачу проекту. На підставі прийнятих загальносистемних рішень були визначені склад і організація комплексу технічних засобів.

Вимоги до продуктивності:

Обчислювальний комплекс повинний забезпечувати достатню продуктивність для рішення передбачуваних задач при бажаній мінімізації фінансових витрат на його придбання. Крім того, потрібно забезпечити сприятний час реакції системи при роботі в інтерактивних режимах, що складають значну частину від всіх експлуатаційних режимів. Цим буде забезпечена комфортна робота користувачів. Проектована система не буде пред'являти підвищених вимог до обчислювальної потужності. Створювану систему можна віднести до класу облікових задач, у яких загальна продуктивність системи визначається швидкістю операцій уведення-висновку при неповному залученні потужності центрального процесора. У такий спосіб будь-яке сучасне апаратне рішення з пропонованих на ринку буде відповідати вимогам до продуктивності. Розподіл програмних компонентів майбутньої системи дозволить не збільшити навантаження на існуючі клієнтські робочі місця і не висувати вимог до їх модернізації.

Вимоги до надійності:

Розв'язувана задача висуває стандартні вимоги до надійності апаратних засобів без вимоги забезпечення повної відмовостійкості. Основною вимогою в даній групі буде вимога до схоронності даних. Передбачаються наступні заходи для його виконання:

- включення сервера в електромережу через пристрій безперебійного живлення (UPS);
- використання на сервері БД апаратного RAID контролера з можливістю відзеркалення дисків (дане рішення має досить сприятну вартість);

Вимоги до експлуатаційних характеристик:

З перерахованих вимог можна зробити висновок про застосовність для сервера БД і клієнтських робочих місць стандартного апаратного забезпечення на платформі x86. До сервера БД пред'являється традиційний набір вимог для такого рішення.

2.4.2 Структурна схема комплексу обчислювальних засобів

Проектована програмна система буде виконана за схемою “Клієнт-сервер”, тому апаратне рішення буде складатися з обчислювальної підсистеми і мережі передачі даних. Принципова схема виконана у виді схеми розміщення наведена на рисунку 2.4.



Рисунок 2.4. - Схема розміщення комплексу технічних засобів.

2.5 Опис алгоритмів функціонування системи

Для уточнення алгоритмів функціонування розроблюваної системи використовуємо засобу об'єктно-орієнтованого аналізу й опису поведінки об'єктів. При цьому будемо розглядати як поведінку системи в цілому (для уточнення виду, характеру і порядку взаємодії між класами), так і реалізацію функціональності окремих класів системи. Діаграма класів системи, побудована на підставі аналізу сутностей і операцій над ними, представлена на рисунку 2.5.

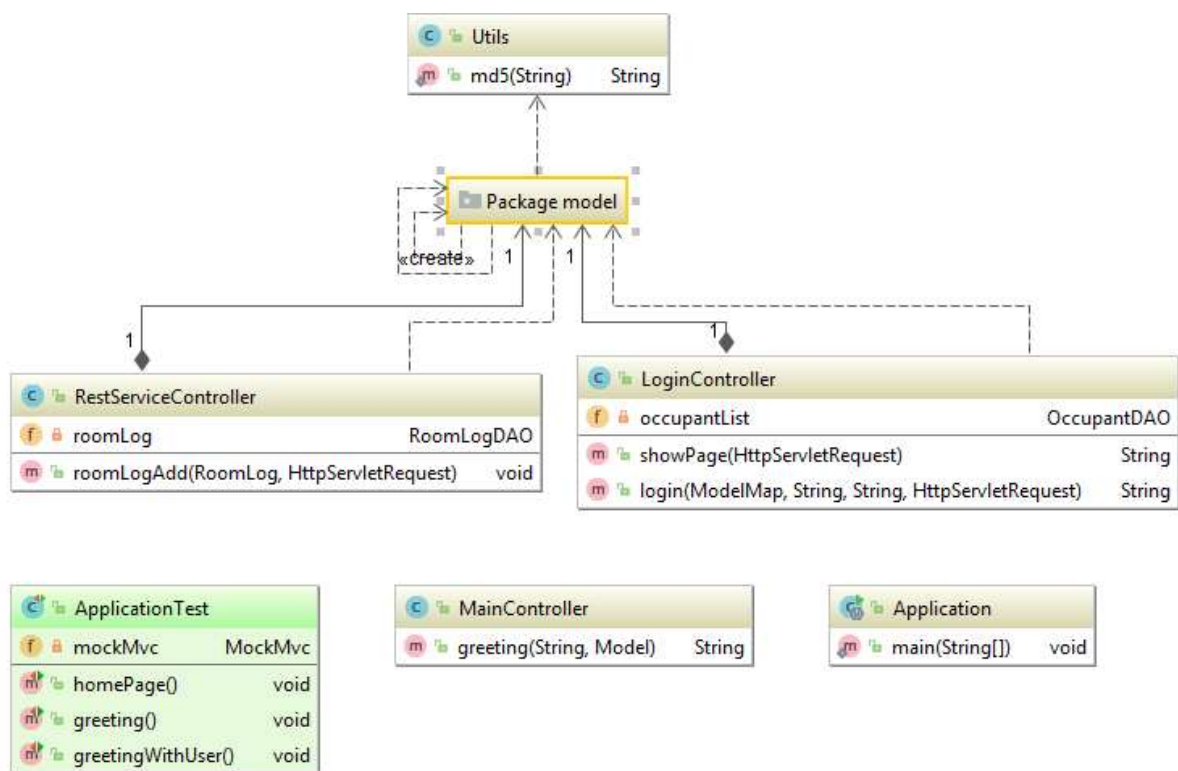


Рисунок 2.5.1 - Діаграма класів системи

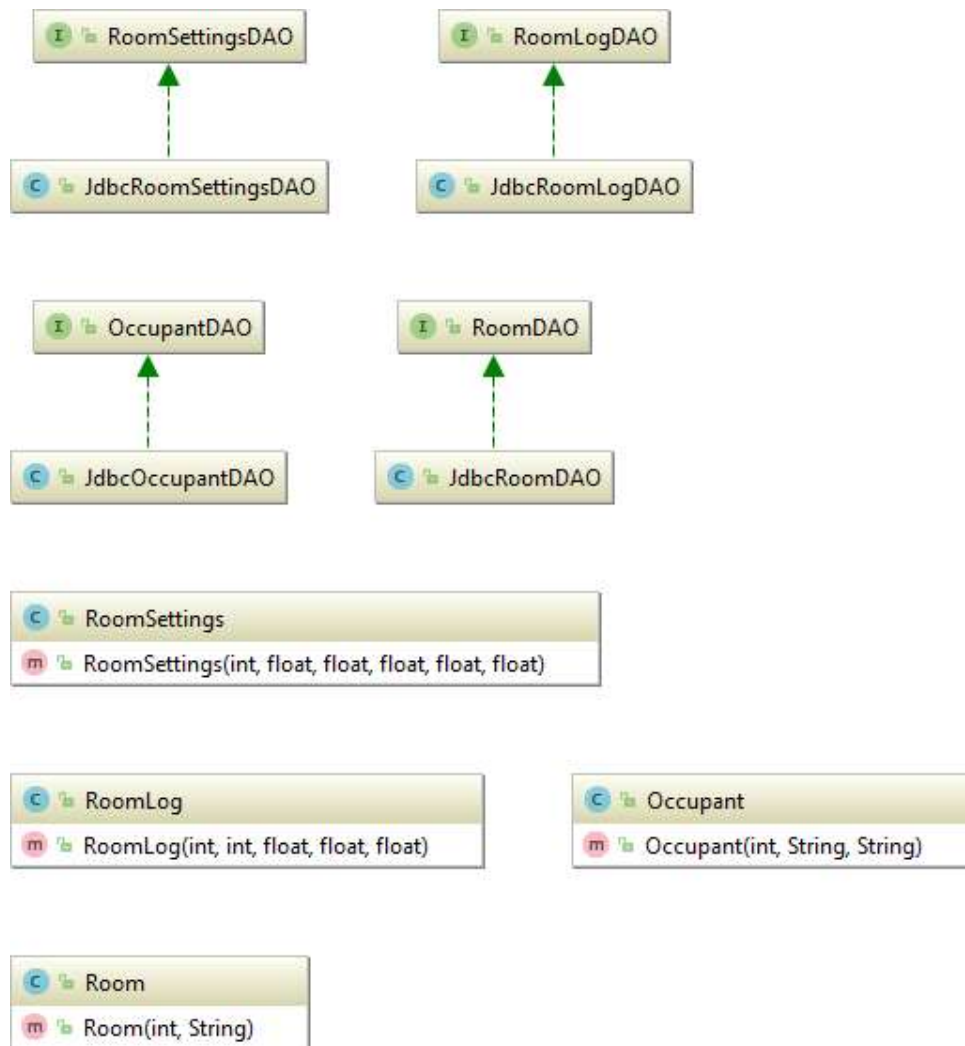


Рисунок 2.5.2 - Діаграма класів пакету `model`

Occupant(Житель) - користувач який може редагувати налаштування

Room(Кімната) – клас кімнати який зберігає в дані про кімнату

`RoomSettings` – клас який зберігає налаштування для кімнати

`RoomLog` – клас який зберігає статистичні дані від кімнат

`LoginController` – сервлет який перевіряє належність жителя до кімнати

MainController – сервлет що займається перевіркою параметрів кімнат

RestServiceController – сервлет приймає повідомлення від датчиків від мікроконтролерів

Розглянута вище діаграма класів являє собою логічну модель статичного представлення системи, що моделюється. Мова йде про те, що на даній діаграмі зображуються тільки взаємозв'язки структурного характеру, що не залежать від часу реакції системи на зовнішні події. На відміну від цього представлення, загальний алгоритм поведінки системи може бути описаний за допомогою діаграми станів (statechart diagram). Загальний алгоритм поведінки важливий для визначення взаємодій між компонентами системи й уточнення причин, що викликають та чи інша відповідна дія. Алгоритм поведінки системи в нотації діаграми станів приведений на рисунку 2.6.

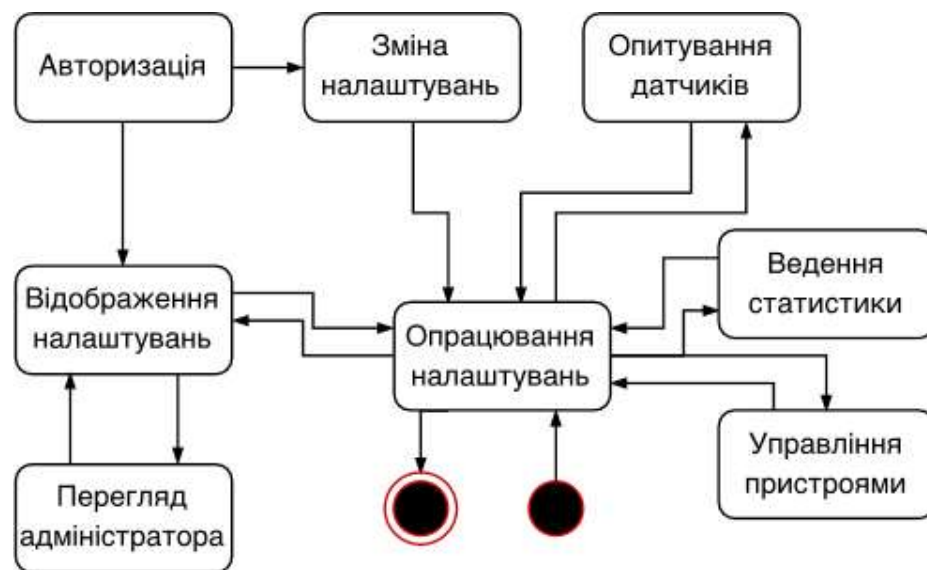


Рисунок 2.6. - Загальний алгоритм поведінки системи.

2.6 Рішення по програмному забезпеченню

2.6.1 Структура і функції частин програмного забезпечення

Функціональність і структуру проектованої автоматизованої системи поряд з моделлю поведінки визначить модель її станів. Будемо будувати модель станів шляхом уточнення моделі бізнесів-сутностей, побудованої нами на підставі вимог користувачів до функцій системи й оброблюваних даних. Вхідними даними для даного етапу проектування будуть моделі сутностей, а

результатом стане модель специфікації проектних конструкцій розроблювального програмного забезпечення. У відповідності з прийнятим у даній роботі підходом об'єктно-орієнтованого аналізу і проектування виконаємо подальшу специфікацію проектних рішень у виді діаграм класів. Класи визначають стан пропонованої програмної системи (через значення своїх атрибутів) і, значною мірою, її поведінку (через операції чи методи класів).

Сформульовані користувацькі вимоги тепер приймемо за обмеження. Поряд з моделлю сутностей для проектування класів будемо користатися побудованими раніше діаграмами прецедентів (рисунок 2.1), у відповідності до рекомендацій методики Rational Unified Process. При цьому модель класів буде відповідати словнику даних, що був отриманий на підставі аналізу прецедентів, і який є текстовим аналогом моделі сутностей.

2.6.2 Фізична модель бази даних

Для забезпечення можливості зберігати дані в реляційній базі даних, з моделі класів була отримана фізична модель даних. Для цього серед усіх класів були визначені постійні. Це такі класи, об'єкти яких повинні зберігати свої значення не тільки під час роботи програми. Далі було проведено відображення постійних класів на таблиці моделі даних. При цьому атрибути класів відобразилися в поля реляційних таблиць. Асоціації між класами змінилися в зв'язки між реляційними таблицями з використанням механізму зовнішніх ключів. Для забезпечення можливості підтримки цілісності даних по сутностям були використані первинні ключі. Обрана СКБД не підтримує зберігання ієрархії спадкування класів, тому при будівництві фізичної моделі даних ієрархії відображалися на таблиці за принципом одна таблиця на клас. Метод згортання типів в супертип при якому ієрархія змінюється в одну таблицю був відхилений із-за великої надмірності даних. Розроблена фізична модель наведена на рисунку 2.7.

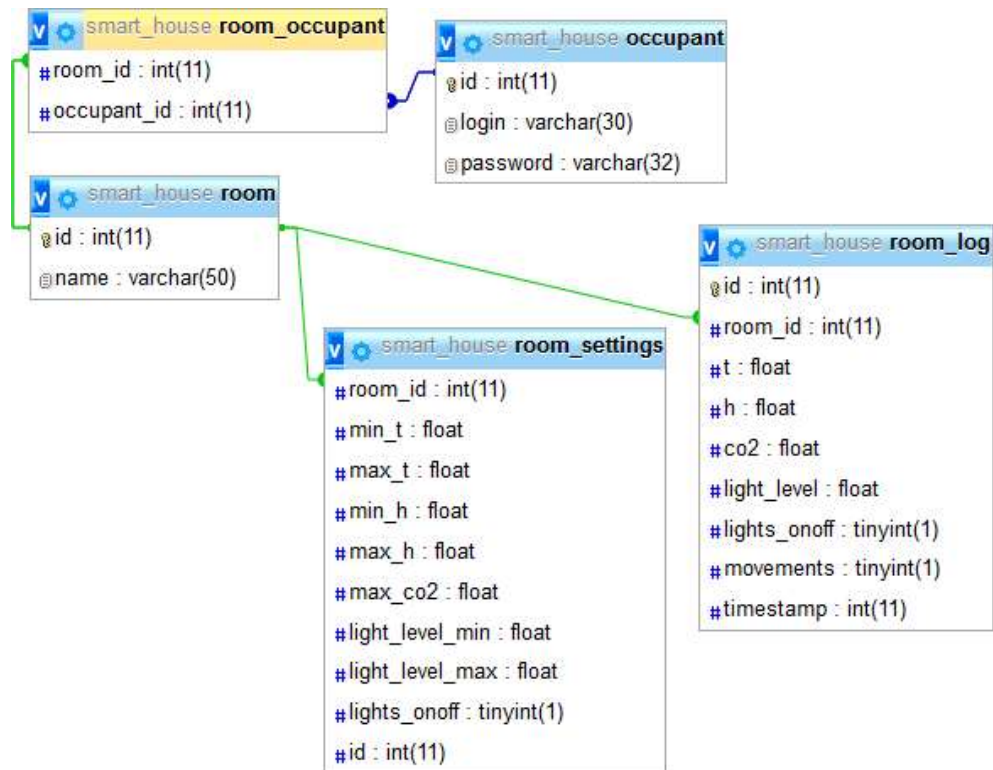


Рисунок 2.7. Фізична модель бази даних

2.6.3 Методи і засоби розробки програмного забезпечення

При аналізі і проектуванні Системи застосовувався об'єктно-орієнтований підхід (ООП), виходячи з чого програмне забезпечення системи повинне бути розроблене з його використанням. Основними принципами ООП є спадкування, інкапсуляція і поліморфізм.

Використання ООП у розробці дозволяє:

- використовувати отримані в результаті аналізу і проектування логічні моделі для первинної кодогенерації;
- скоротити час розробки за рахунок повторного використання коду програмних компонентів;
- проводити налагодження і тестування ґрунтуючись на поводженні об'єктів, без знання їхньої внутрішньої структури;
- цілком використовувати потенціал обраних засобів розробки ПО.

Для проектування й аналізу Системи використовувалася система проектування мовою UML з відкритим кодом - ArgoUML виробництва спільноти Tigris Software – Tigris.org. На сьогоднішній день це один із потужних засобів проектування на ринку. Мова проектування UML - де-факто стандарт галузі. ArgoUML використовується на всіх етапах проектування від аналізу до автоматичної генерації коду програмної системи і має засоби для роботи з всіма інструментами, наданими мовою UML.

Основним засобом розробки програмного забезпечення системи є мова програмування високого рівня Java, використовуються такі технології: Servlets, EJB3, JSPx та JSF. У якості середовища розробки був використаний продукт IntelliJ IDEA 2018.3

На Web-сервері повинен бути встановлений HTTP-сервер. Найрозповсюдженими HTTP-серверами, що задовольняють зазначеним вимогам і мають високу продуктивність, є Apache Server та nginx. На Web-сервері повинна бути встановлена ОС яка підтримує один з них.

Так як використовуються технології організовані на Java а також Raspberry Pi то операційна система на сервері доступу до БД може бути тільки Linux:

Також потрібний Java WEB сервер встановлений на сервері будь який з:

- JBoss
- Payara
- GlassFish
- Tomcat

На сервері баз даних повинна бути встановлена серверна система управління базами даних (СУБД).

Необхідна сучасна серверна реляційна СУБД, що має високу продуктивність, здатна працювати з великими обсягами інформації і підтримує стандарт SQL-92. Крім того, СУБД повинна працювати під управлінням

Windows. Серверними СУБД, що задовольняють зазначеним вимогам, на сьогодні є:

- MySQL.
- PostgreSQL .

На клієнтських машинах може використовуватися будь-яка операційна система з можливістю встановлення віддаленого мережного з'єднання і графічним інтерфейсом. Операційні системи, що рекомендуються – MS Windows, Linux, тощо.

Програмне забезпечення, з яким буде працювати користувач, являє собою Web-оглядач з підтримкою HTML 5, JavaScript, CSS 3 (наприклад, FireFox, Opera, Chrome ...).

Програмне забезпечення для мікроконтролерів для зчитування показань датчиків та відправки їх на сервер розробляється з використанням Arduino IDE for esp8266 з набором сумісних бібліотек

Висновок

У межах розділу було сформовано комплекс проектних рішень, необхідних для створення ефективних засобів віддаленого доступу до системи «Розумне помешкання». На основі аналізу вимог, архітектурних підходів та особливостей функціонування подібних систем визначено загальносистемні принципи побудови, що забезпечують узгодженість, масштабованість та надійність програмно-апаратного комплексу.

Рішення з організаційного забезпечення окреслили структуру взаємодії користувачів, адміністраторів та технічних компонентів системи, а також регламентували порядок доступу, обробки та захисту інформації. У межах інформаційного забезпечення сформовано моделі даних, визначено їхній склад, структуру та правила обміну, що створює основу для коректної роботи всіх функціональних модулів.

Технічне забезпечення охопило вибір апаратних засобів, мережевої інфраструктури та комунікаційних протоколів, які гарантують стабільність

роботи системи та можливість інтеграції з різнорідними пристроями «Розумного помешкання». У частині математичного забезпечення розроблено алгоритм функціонування механізмів віддаленого доступу, що визначає логіку обробки запитів, маршрутизацію даних, контроль станів та забезпечення безпеки.

Рішення з програмного забезпечення включили опис архітектури, функціональних модулів та принципів реалізації програмної частини системи, що забезпечує взаємодію користувача з пристроями помешкання через захищені канали зв'язку та інтуїтивно зрозумілий інтерфейс.

Таким чином, у розділі сформовано цілісну концепцію побудови системи віддаленого доступу до «Розумного помешкання», яка охоплює всі необхідні аспекти – від організаційних і інформаційних рішень до технічної, математичної та програмної реалізації. Це створює методологічне та технологічне підґрунтя для подальшої розробки, впровадження та тестування програмно-апаратного комплексу.

3 Розробка робочої документації системи «Розумний дім» з віддаленим доступом

Система «Розумний дім» з віддаленим доступом призначена для автоматизації управління мікрокліматом, управлінням освітленням, управління автоматичним вимкненням світла. Функції, що автоматизуються системою, пов'язані з управлінням системами опалювання, кондиціювання та управління освітленням.

Основними функціями, що автоматизуються є:

- Управління системами мікроклімату.
- Моніторинг мікроклімату в приміщеннях.
- Управління вимикачами в приміщеннях
- Моніторинг освітлення в приміщеннях
- Усі інші вимоги не є головними і можуть змінюватися під час розробка або експлуатації

При впровадженні програмного забезпечення системи основними цілями є:

- Простий і зручний інтерфейс користувача.
- Доступ до системи через Internet з будь-якого комп'ютера або мобільного пристрою.
- Багатокористувацька робота.
- Висока масштабуємість системи.
- Висока продуктивність.
- Надійність.
- Конфіденційність інформації.

Для ефективного використання програмного забезпечення важливо впровадити організаційні і технічні заходи, що передбачено в п. 2.2 та п. 2.4. При роботі з програмним забезпеченням системи варто керуватися документами «Технічне завдання» (Додаток А), «Програма та методика випробувань» (Додаток Б), «Керівництво користувача» (Додаток В)..

Висновок

Розроблено документацію для системи «Розумний дім» з віддаленим доступом, яка призначена для автоматизації управління мікрокліматом, управлінням освітленням, управління автоматичним вимкненням світла.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

На даний момент сучасні обчислювальні засоби широко використовуються в повсякденній діяльності людини. Сьогодні обчислювальна техніка дозволяє вирішувати найрізноманітніші задачі: від звільнення людини від монотонної ручної праці до розробки складних технічних розрахунків, що не під силу виконати одній людині. Сучасні засоби програмування дозволяють вирішувати задачі, що стояли в недавньому минулому перед цілими колективами наукових співробітників.

Розвиток нових технологій і ріст застосування обчислювальної техніки привели до широкого застосування ЕОМ для вирішення проблем, пов'язаних з обробкою інформації, її контролем і аналізом. Сьогодні на основі засобів обчислювальної техніки розробляються і впроваджуються різні автоматизовані інформаційні системи (управління, проектування, технологічної підготовки виробництва). Успіх у впровадженні цих систем, їхня роль в інтенсифікації розвитку народного господарства нашої країни багато в чому залежить від фахівців із системотехніки, які знають методику аналізу і проектування цих систем, можливості обчислювальної техніки і які володіють математичними методами, що використовуються при постановці та вирішенні задач.

Обґрунтувати доцільність розробки з економічної точки зору можна, розрахувавши собівартість програмного забезпечення і додатковий прибуток, що буде отримана в результаті впровадження. Основним показником, що визначає економічну доцільність витрат на створення програмного забезпечення є річний економічний ефект. Характерним показником економічної ефективності створення і функціонування програмного продукту, що розраховується на стадії створення, впровадження і використання, є ефективність витрат і розрахунковий коефіцієнт економічної ефективності капітальних вкладень.

На першому етапі розраховується собівартість розробки програмного забезпечення. Потім капітальні витрати, необхідні для придбання

устаткування. На підставі цього можна одержати річний економічний ефект і строки окупності даного проекту.

4.1 Розрахунок витрат на створення й експлуатацію інформаційної системи

Розробка програмного забезпечення виконується двома програмістами відділу, місячний оклад кожного складає 1500 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювачів системи 3600 грн./міс, а вартість сучасної ПЗВМ складає 4000 грн. (приведена вартість машини на базі Athlon X2). При вартості кіловат-години електроенергії рівної 0.50 грн., розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в Таблиці 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума грн.
Папір	50.00
Заправлення картриджа до принтера	100.00
Дискові носії (CD/DVD)	30.00
Література	100.00
Непередбачені витрати	20.00
Разом матеріали і комплектуючі вироби.	145.00

Вартість програми розрахуємо по формулі:

$$C_{\text{пр}} = (З_{\text{зп}} + З_{\text{сз}} + З_{\text{зр}} + З_{\text{с}}) * T + З_{\text{м}}, \quad (3.1)$$

де T - тривалість розробки, міс.

$З_{\text{зп}}$ - основна і додаткова заробітна плата обслуговуючого персоналу, грн.

$З_{\text{сз}}$ - відрахування на соціальні заходи (38.5% від основної і додаткової заробітної плати), грн.

$З_{\text{зг}}$ - загальногосподарські витрати (10% від основної заробітної плати), грн.

$З_{\text{е}}$ - витрати на електроенергію.

$З_{\text{м}}$ - витрати на основні і допоміжні матеріали.

При споживанні потужності 0.5 Вт, тривалості роботи на місяць, рівної $21 \cdot 6 = 126$ годин, вартості кіловат-години електроенергії 0.5грн.,

$$З_e = 126 \cdot 0.5 \cdot 0.5 = 31.5 \text{ грн.}$$

Таблиця 3.2 – Витрати на розробку системи

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	Міс.	3
Основна і додаткова заробітна плата	грн.	3600
Відрахування на соціальні заходи	грн.	1350
Загальногосподарські витрати	грн.	360
Витрати на основні і допоміжні матеріали	грн.	145
Витрати на електроенергію	грн.	31.5

Відповідно до формули 3.1 вартість програми:

$$C_{\text{пр}} = (3600 + 1350 + 360 + 31.5) \cdot 3 + 145 = 16169,5 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

$$A_{\text{об}} = 4000 \cdot 0,6 = 2400 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали (гнучкі диски, папір) визначаються в розмірі 5% вартості основного устаткування:

$$B_m = 4000 \cdot 0,05 = 200 \text{ грн.}$$

Річний обсяг робіт ПЗВМ у годинах визначається в такий спосіб:

$$\Phi_m = 253,3 \cdot T_z, \quad (3.2)$$

де T_z - це середнє місячне завантаження устаткування (близько 4 г.) 253,3 середня кількість робочих днів у році. Отже, річний обсяг роботи ПЕОМ складе:

$$\Phi_m = 253,3 \cdot 4 = 1013 \text{ годин}$$

Витрати на електроенергію $З_e$ при 1013 годинах роботи устаткування в рік складуть:

$$З_e = 1013 \cdot 0,24 \cdot 0,5 = 121,5 \text{ грн.}$$

Експлуатаційні витрати для ПЗВМ за рік складуть:

$$З_{\text{оп}} = 2400 + 200 + 121,5 = 2612,5 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію програми складуть:

$$З_{ce} = 16169,5 + 2612,5 = 18782 \text{ грн.}$$

4.2 Розрахунок економічної ефективності

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 2,4 працівника (за експертною оцінкою фахівців підприємства).

Зарплата 2,4 працівника в рік складає:

$$(1300 * 12) * 2,4 = 37440 \text{ грн.}$$

Пряма економічна ефективність впровадження розроблювальної системи показана в Таблиці 3.3.

Таблиця 3.3 – Пряма економічна ефективність впровадження розроблювальної системи

У перший рік	$37440 - 18782 = 9866,325 \text{ грн.}$
У наступні	$37440 - 2612,5 = 15293,5 \text{ грн.}$

4.3 Розрахунок строку окупності програми

Витрати окупляться за:

$$(18782 / 9866,325) * 12 = 22 \text{ місяців.}$$

У такий спосіб витрати на створення програмного продукту окупляться протягом 22 місяців.

Висновки

Виходячи з розрахунків, впровадження системи окупить себе через 22 місяці. Отже можна зробити висновок, що розробка підсистеми для автоматизованого управління проектами на рівні компанії економічно вигідна й обґрунтована.

5. ОХОРОНА ПРАЦІ

Вступ

Охорона праці є невід'ємною складовою організації трудової діяльності в будь-якій сфері. Особливої актуальності питання забезпечення безпечних і здорових умов праці набувають у галузі інформаційних технологій, де фахівці значну частину робочого часу здійснюють професійну діяльність за персональними комп'ютерами. Розроблення інформаційних систем, як один із ключових напрямів діяльності ІТ-спеціалістів, передбачає тривалу роботу з комп'ютерним обладнанням, моніторами та іншими технічними засобами.

Порушення вимог охорони праці під час роботи з комп'ютерною технікою може спричинити розвиток низки професійно зумовлених захворювань, зокрема синдрому зап'ясткового каналу, комп'ютерного зорового синдрому, захворювань опорно-рухового апарату тощо. Крім того, неналежна організація робочого місця, недостатній рівень освітлення, невідповідні параметри мікроклімату у виробничих приміщеннях здатні негативно позначатися на самопочутті та працездатності розробників ІС.

У зв'язку з цим дотримання вимог охорони праці та створення належних умов для збереження здоров'я і безпеки фахівців, задіяних у процесі розроблення інформаційних систем, є одним із пріоритетних завдань керівників підприємств та організацій. Забезпечення безпечних умов праці не лише сприяє поліпшенню фізичного та психологічного стану працівників і зниженню ризиків виникнення професійних захворювань, але й позитивно впливає на продуктивність праці та якість кінцевого програмного продукту.

У цьому розділі дипломної роботи, присвяченої розробленню інформаційної системи підтримки волонтерської діяльності, буде розглянуто

основні аспекти охорони праці в галузі інформаційних технологій, зокрема у контексті проєктування та розроблення ІС. Особливу увагу буде приділено аналізу нормативно-правових актів, що регламентують питання охорони праці, вимогам до організації робочого місця розробника, забезпеченню безпечних умов праці та мінімізації впливу шкідливих виробничих факторів, регламентації режиму праці та відпочинку, а також специфіці застосування заходів електробезпеки.

5.1. Нормативно-правові акти, що регулюють питання охорони праці при роботі з персональними комп'ютерами та іншими пристроями для розробки вебзастосунків

Питання охорони праці під час роботи з персональними комп'ютерами (ПК) та іншими пристроями, що застосовуються для розробки інформаційних систем, регулюються низкою нормативно-правових актів України. Основним документом, що визначає вимоги безпеки й захисту здоров'я працівників при роботі з екранними пристроями, є Наказ Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» від 14.02.2018 № 207. Дія цього наказу поширюється на всіх суб'єктів господарювання незалежно від форми власності та організаційно-правової структури, які здійснюють діяльність, пов'язану із застосуванням екранних пристроїв.

Відповідно до зазначеного документа, роботодавець зобов'язаний забезпечити відповідність робочих місць працівників ергономічним, організаційно-технічним і санітарно-гігієнічним вимогам. Зокрема, роботодавець має інформувати працівників про умови праці та наявність небезпечних або шкідливих виробничих чинників, які залишаються

невирішеними, а також забезпечити проведення інструктажів з питань охорони праці та пожежної безпеки під час роботи з екранними пристроями.

Окрім згаданого нормативного акта, питання охорони праці при роботі з ПК регламентуються й іншими нормативними документами, наприклад, ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», який містить санітарно-гігієнічні вимоги до умов праці, організації робочих місць з ВДТ ЕОМ, параметрів виробничого середовища, освітлення, рівнів шуму тощо.

Електробезпека при експлуатації ПК і периферійних пристроїв забезпечується виконанням положень Правил улаштування електроустановок (ПУЕ), Правил технічної експлуатації електроустановок споживачів (ПТЕЕС), Правил безпечної експлуатації електроустановок споживачів (ПБЕЕС). Згідно з пунктом 1.7.110 ПУЕ, конструкція та розміщення комп'ютерного обладнання повинні відповідати встановленим вимогам електробезпеки і забезпечувати зручність його технічного обслуговування.

Строге дотримання вимог НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів», а також інших державних стандартів і санітарних норм, є обов'язковим для всіх роботодавців та працівників, що використовують ПК і суміжне обладнання для розробки вебзастосунків. Лише системний підхід і суворе виконання положень нормативно-правових актів щодо охорони праці дають змогу створити безпечні та комфортні умови для ІТ-фахівців, істотно знизити ризики виникнення професійних захворювань і травматизму.

5.2. Вимоги до організації робочого місця розробника інформаційної системи

Організація робочого місця розробника вебзастосунків відіграє ключову роль у забезпеченні безпечних та комфортних умов праці, а також у підвищенні продуктивності та ефективності роботи спеціаліста. При облаштуванні робочого простору необхідно враховувати комплекс ергономічних, санітарно-гігієнічних та організаційно-технічних вимог, що регламентуються відповідними нормативно-правовими документами.

Одним із основних нормативних документів, що визначає вимоги до робочого місця з персональним комп'ютером, є ДСТУ ISO 9241-5:2004 «Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 5. Вимоги до компонування робочого місця та до робочої пози». Згідно з цим стандартом, робоче місце розробника вебзастосунків має бути достатньо просторим для зміни робочого положення та забезпечувати зручний доступ до всіх необхідних елементів обладнання.

Робочий стіл повинен мати площу, достатню для розміщення комп'ютера, периферійних пристроїв, документації та інших необхідних матеріалів. Рекомендовані розміри робочої поверхні становлять: ширина – не менше 1200 мм, глибина – не менше 800 мм. Висота робочої поверхні має забезпечувати правильне положення рук користувача на клавіатурі та відповідати антропометричним даним працівника. Оптимальна висота столу варіюється в межах 680–800 мм.

Робоче крісло розробника повинно бути підйомно-поворотним, з регулюванням висоти та кута нахилу спинки, що забезпечує надійну підтримку тіла у зручному положенні. Сидіння та спинка крісла мають бути

м'якими, з заокругленими краями та виготовленими з повітропроникного матеріалу. Для зменшення статичного навантаження на м'язи верхніх кінцівок рекомендується використовувати стаціонарні або регульовані підлокітники.

Монітор слід розташовувати на відстані 600–700 мм від очей користувача, причому верхня точка екрану має знаходитися на рівні очей або трохи нижче. Кут нахилу екрана повинен становити 10–15° при перпендикулярному розташуванні площини екрану до лінії зору користувача. Клавіатуру слід розміщувати на робочій поверхні або на спеціальній полиці на рівні ліктів користувача з кутом нахилу до горизонтальної площини 15–20°.

Для запобігання перенавантаженню зорового аналізатора та розвитку комп'ютерного зорового синдрому необхідно забезпечити належний рівень освітленості робочої зони. Рекомендована освітленість робочого столу становить 300–500 лк, а освітленість екрана монітора не повинна перевищувати 300 лк. Слід уникати прямого світла в очі та відблисків на екрані монітора.

Мікроклімат приміщення, у якому працює розробник вебзастосунків, має відповідати санітарним нормам. Оптимальна температура повітря складає 22–24°C у холодний період року та 23–25°C – у теплий, відносна вологість – 40–60%, швидкість руху повітря – не більше 0,1 м/с. Для підтримки комфортних параметрів мікроклімату рекомендується застосовувати системи кондиціонування або вентиляції.

Дотримання ергономічних та санітарно-гігієнічних вимог до організації робочого місця розробника вебзастосунків створює оптимальні умови для праці, знижує ризик розвитку професійних захворювань опорно-рухового апарату та органів зору, а також підвищує працездатність і ефективність роботи фахівця.

5.3. Забезпечення безпечних умов праці та мінімізація впливу шкідливих факторів при роботі з комп'ютерним обладнанням

Забезпечення безпечних умов праці та мінімізація впливу шкідливих факторів під час роботи з комп'ютерним обладнанням є одним із ключових завдань охорони праці в сфері інформаційних технологій. Незважаючи на те, що професійна діяльність розробника вебзастосунків не належить до категорії робіт з підвищеною небезпекою, вона супроводжується низкою потенційних ризиків для здоров'я працівника.

До основних шкідливих і небезпечних факторів, які можуть негативно впливати на стан здоров'я розробника під час роботи з комп'ютерною технікою, належать: підвищений рівень електромагнітного випромінювання, статичне електричне поле, неіонізуюче електромагнітне випромінювання оптичного діапазону, підвищений рівень шуму та вібрації, незадовільні параметри мікроклімату, недостатнє або нераціональне освітлення робочої зони, а також психофізіологічні фактори, такі як розумове перенавантаження, перевантаження зорового та опорно-рухового апарату.

Для мінімізації впливу зазначених факторів та створення безпечних умов праці необхідно дотримуватися комплексу вимог та рекомендацій. Зокрема, відповідно до ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», рівень електромагнітного випромінювання на відстані 50 см навколо відеотерміналу за електричною складовою не повинен перевищувати 2,5 В/м у діапазоні частот 5 Гц – 2 кГц та 0,25 В/м у діапазоні частот 2 – 400 кГц. Напруженість електростатичного поля на робочому місці з ВДТ не повинна перевищувати гранично допустиме значення – 20 кВ/м.

Для зниження впливу електромагнітного випромінювання рекомендується використовувати монітори, що відповідають сучасним

стандартам безпеки, зокрема MPR II або TCO. Також слід дотримуватися регламентованих перерв при роботі з комп'ютером, під час яких доцільно виконувати комплекс вправ для очей, шиї, плечей, спини та кистей рук.

Важливим аспектом забезпечення безпечних умов праці є підтримка оптимальних параметрів мікроклімату у виробничому приміщенні. Згідно з ДСанПіН 3.3.2-007-98, на робочих місцях з ВДТ необхідно забезпечувати температуру повітря 22–24°C у холодний період року та 23–25°C у теплий, відносну вологість 40–60% і швидкість руху повітря не більше 0,1 м/с. У приміщеннях з комп'ютерним обладнанням слід щоденно проводити вологе прибирання та регулярно провітрювати приміщення.

Для запобігання перенавантаженню зорового аналізатора під час роботи з комп'ютером необхідно забезпечити раціональне освітлення робочої зони. Освітленість робочого столу у зоні розташування документів має становити 300–500 лк. Яскравість світильників загального освітлення в зоні кутів випромінювання 50°–90° відносно вертикалі в подовжній та поперечній площинах не повинна перевищувати 200 кд/м², а яскравість відблисків на екрані ВДТ – 40 кд/м².

Для зниження ризику розвитку синдрому зап'ястного каналу та інших захворювань опорно-рухового апарату рекомендується використовувати ергономічні клавіатури та маніпулятори типу «миша». Крім того, необхідно дотримуватися правильної робочої пози: сидіти прямо, не сутулячись; лікті повинні бути зігнуті під кутом 70–90°, стегна – розташовані горизонтально, а стопи – на підлозі або на спеціальній підставці.

Не менш важливим фактором забезпечення безпеки праці є дотримання правил електробезпеки. Комп'ютерне обладнання та периферійні пристрої повинні підключатися до електромережі через справні штепсельні роз'єми. Забороняється використовувати саморобні подовжувачі, нестандартне

опалювальне електрообладнання, а також залишати увімкнені прилади без нагляду.

Для запобігання ураженню електричним струмом у разі випадкового контакту зі струмопровідними частинами обладнання необхідно забезпечити наявність заземлення або занулення. Крім того, рекомендовано періодично перевіряти стан ізоляції електропроводів та ефективність заземлюючих пристроїв.

Висновок

Дотримання наведених вище вимог і рекомендацій щодо забезпечення безпечних умов праці та мінімізації впливу шкідливих факторів під час роботи з комп'ютерним обладнанням дає змогу зберегти здоров'я і працездатність розробника вебзастосунків, а також підвищити ефективність і якість його професійної діяльності.

6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вступ до проблеми

Охорона навколишнього середовища здійснюється через різні засоби, зокрема й правові механізми. У сфері права захист охоплює майже всі складові природного середовища. Основними сучасними нормативно-правовими актами, які визначають основи організації охорони природи в Україні, є Закони «Про охорону навколишнього природного середовища» (25 червня 1991 р.), «Про охорону атмосферного повітря» (16 жовтня 1992 р.), «Про природно-заповідний фонд України» (16 червня 1992 р.), «Про тваринний світ» (3 березня 1993 р.), «Про карантин рослин» (30 червня 1993 р.) тощо. Крім того, питання використання і захисту природи регулюють також кодекси – земельний, водний, лісовий, про надра, а також закони «Про плату за землю» (3 липня 1992 р.), «Про ветеринарну медицину» (25 червня 1992 р.). Важливим є й затверджений Верховною Радою Порядок щодо обмеження, тимчасової заборони чи припинення діяльності підприємств, установ, організацій та об'єктів у разі порушень законодавства про охорону довкілля.

Види права природокористування включають землекористування, водокористування, лісокористування, користування надрами, тваринним світом та природно-заповідним фондом. Природокористування — це раціональне застосування природних ресурсів людиною для задоволення своїх потреб. Ключові принципи цього права: цільове використання, плановість, тривалість, ліцензування, а також врахування його особливої ролі для суспільства. Виокремлюють загальне й спеціальне природокористування стосовно землі, вод, лісу, надр, тваринного світу тощо. Згідно із Законом «Про охорону навколишнього природного середовища», усі громадяни мають право на загальне користування природними ресурсами безкоштовно та без спеціальних дозволів. Це стосується відвідування парків, скверів, водойм,

лісу, збору дикорослих ягід, грибів, горіхів тощо. Стаття 13 Конституції України гарантує це право кожному. Спеціальне природокористування виникає, коли частину ресурсів держава передає у визначених законом випадках індивідам або організаціям на певний термін і за відповідну плату; передача закріплюється спеціальним дозволом.

Права природокористувачів супроводжуються екологічними правами та обов'язками, що інтенсивно розвиваються в юридичній науці. За Конституцією України, кожен має право на безпечне довкілля і відшкодування шкоди, завданої порушенням цього права. Також гарантується доступ до інформації про стан довкілля та її поширення, що закріплено і в Законі «Про охорону навколишнього природного середовища». Над цим правом лежить обов'язок держави впроваджувати санітарно-гігієнічні заходи для покращення стану довкілля. Екологічні права громадян захищаються у судовому порядку. Разом із цим Закон передбачає певні обов'язки, незалежно від статусу природокористувача: усі зобов'язані берегти природу, використовувати ресурси раціонально та не завдавати їм шкоди. Для суб'єктів спеціального використання природних ресурсів встановлюються додаткові обов'язки, включно зі сплатою за використання та забруднення довкілля згідно з нормативами і лімітами. Контроль у цій сфері здійснюють як державні органи, так і громадські організації шляхом перевірок, нагляду, обстежень та експертиз. Державний контроль здійснюють Ради народних депутатів, державні адміністрації та Міністерство охорони навколишнього природного середовища разом із його місцевими структурами.

Основні пріоритети охорони довкілля та раціонального використання природних ресурсів включають:

- забезпечення екологічної безпеки ядерних об'єктів і захист населення й довкілля від радіації, а також мінімізацію негативних наслідків аварії

на Чорнобильській АЕС;

- покращення екологічного стану річкових басейнів України та якості питної води;
- стабілізацію й поліпшення екологічної ситуації у містах і промислових центрах Донецько-Придніпровського регіону;
- будівництво та модернізацію міських очисних каналізаційних споруд;
- боротьбу із забрудненням Чорного й Азовського морів та покращення їхнього стану;
- формування збалансованої системи природокористування, структурну перебудову економіки, екологізацію процесів у промисловості, енергетиці, будівництві, сільському господарстві та транспорті;
- збереження біологічного й ландшафтного різноманіття, розвиток заповідної справи.

Для досягнення цих цілей передбачено такі завдання:

- зменшення рівня радіаційного забруднення до мінімуму;
- захист повітря від забруднення, особливо у великих містах і промцентрах;
- захист і збереження земель від виснаження, забруднення та нераціонального використання;
- збереження та розширення природних ландшафтів, посилення охорони заповідних і рекреаційних територій;
- підвищення стійкості лісів і їхніх екологічних функцій;
- знешкодження, утилізація та захоронення побутових і промислових відходів;
- запобігання забрудненню вод, скорочення й припинення скиду забруднених стоків у водойми, захист підземних вод;

- збереження й відродження малих річок, управління водними ресурсами за басейновим принципом;
- завершення формування державної системи моніторингу довкілля;
- створення систем прогнозування, запобігання й оперативного реагування на надзвичайні ситуації природного та техногенного характеру;
- екологічний супровід конверсії військово-промислового комплексу;
- контроль за екологічною діяльністю Збройних Сил України;
- розробка ефективних механізмів природокористування;
- впровадження економічних важелів впливу на систему природокористування;
- формування системи екологічної освіти, виховання й інформування.

6.2 Забруднення навколишнього середовища підприємством з розробки програмного забезпечення

Основним джерелом забруднення навколишнього середовища підприємством є використання автотранспорту. Підприємство впливає на стан атмосферного повітря, однак не спричиняє забруднення водного чи ґрунтового середовища.

У сучасних умовах автомобільний транспорт є ключовим фактором погіршення якості атмосферного повітря, особливо в межах великих міст. Атмосфера урбанізованих територій характеризується постійним надходженням забруднюючих речовин, що істотно відрізняє її параметри від природного стану. У міських поселеннях спостерігаються найвищі рівні антропогенних навантажень, унаслідок чого змінюються мікрокліматичні фактори та фізико-хімічні властивості повітряного басейну. Згідно з

метеорологічними дослідженнями, температура повітря у містах переважно вища, ніж у передмісті, а також частіше фіксуються атмосферні опади, тумани та смоги – густі утворення, що містять дим, кіптяву і вихлопні гази. Прозорість міської атмосфери є нижчою порівняно із сільською місцевістю, а запиленість та тумани суттєво зменшують проникнення сонячної радіації.

Якість атмосферного повітря у містах дедалі більше погіршується через викиди автотранспортних засобів. Збільшення кількості автомобілів призводить до зростання об'ємів забруднювачів, зокрема оксидів вуглецю, вуглеводнів, оксидів азоту, сажі, а також свинцевих і сірчаних сполук, концентрація яких значно підвищилась. Міська інфраструктура не розрахована на існуючу кількість автотранспорту, що спричиняє короткі ділянки безперервного руху (400–600 м) та низьку середню швидкість (12–20 км/год), збільшуючи витрати пального у 3–4 рази, відповідно посилюючи викиди шкідливих речовин.

Автотранспорт також є джерелом специфічних видів забруднення, зокрема пилу від стирання шин, який у значних обсягах потрапляє у повітря. Окрім продуктів згоряння пального, автомобільний транспорт сприяє накопиченню свинцю в довкіллі. В Україні продовжують використовувати бензин із вмістом свинцю 0,36 г/л, тоді як у Великій Британії, Німеччині та США цей показник становить 0,013–0,15 г/л.

6.3 Розробка заходів щодо зменшення забруднення

Природоохоронна діяльність охоплює всі заходи, спрямовані на підтримку якості довкілля на рівні, що гарантує стійкість біосфери. Вона включає масштабні державні програми зі збереження унікальних природних територій та різноманіття видів, організацію наукових досліджень, підготовку екологічних фахівців і просвітницьку роботу серед населення. Крім того,

підприємства здійснюють локальні дії: очищають стічні води та відхідні гази, знижують використання природних ресурсів тощо. Здебільшого такі заходи реалізуються інженерними методами.

Виділяють два основні напрями природоохоронної діяльності підприємств. Перший — очищення шкідливих викидів, який сам по собі є малоефективним, оскільки не завжди дозволяє повністю припинити надходження забруднюючих речовин у біосферу. Зменшення забруднення одного компонента часто призводить до посилення забруднення іншого. Наприклад, впровадження вологих фільтрів у системах газоочистки зменшує забруднення повітря, але збільшує забруднення води. Уловлені речовини можуть отруювати значні площі землі. Очищувальні споруди скорочують рівень забруднення, проте цілком проблему не вирішують, адже самі генерують нові, хоча й менш об'ємні, але концентровані відходи. Діяльність більшості очисних установок потребує значних енергетичних ресурсів, створюючи додатковий екологічний ризик.

Значна частина забруднювачів, які витрачають кошти на їхнє знешкодження, є речовинами, що вже мають виробничу цінність і можуть бути використані у господарстві. Для інтеграції еколого-економічних рішень важливо поєднати процес очищення шкідливих викидів із утилізацією уловлених речовин.

Другий напрям — усунення причин забруднення шляхом розробки маловідходних або безвідходних технологій, які дають змогу раціонально використовувати сировину і максимально утилізувати шкідливі речовини. Однак зараз не для всіх галузей знайдено економічно прийнятні рішення щодо зменшення відходів та їх утилізації, тому необхідно діяти комплексно у двох напрямках.

Слід пам'ятати, що жодні очисні споруди чи безвідходні технології не здатні відновити стійкість біосфери, якщо буде перевищено граничні показники перетворення природних систем, згідно із законом незамінності біосфери. Критичними є використання понад 1% енергії біосфери й трансформація понад 10% природних територій (правило одного і десяти відсотків). Тому технічний прогрес не замінює необхідності перегляду суспільних пріоритетів, стабілізації населення, створення заповідників та інших заходів.

Основні напрями захисту атмосфери від автотранспортних викидів:

розвиток і виробництво автомобілів із високоефективними, малотоксичними двигунами, включаючи дизелізацію транспорту;

впровадження дієвих систем нейтралізації вихлопних газів;

зниження токсичності моторних палив;

оптимізація організації транспортного руху в містах та вдосконалення дорожнього будівництва для забезпечення безперервного руху на автомагістралях.

Висновок

Отже, забезпечення ефективного захисту навколишнього середовища вимагає інтегрованого підходу, який поєднує впровадження технологічних інновацій із цілеспрямованою трансформацією суспільних цінностей. Необхідно не лише застосовувати сучасні методи очищення та нейтралізації забруднень, а й оптимізувати використання природних ресурсів і сприяти збереженню екосистем. Лише всебічне поєднання технологічних рішень і свідомої екологічної відповідальності дозволить підтримати стабільність біосфери для майбутніх поколінь.

Висновки

У результаті виконання даної дипломної роботи було розроблено проект системи розумного будинку з віддаленим доступом.

Для досягнення цієї мети були вирішені наступні задачі:

- сформульовані вимоги до автоматизованої системи управління;
- розроблені концепції системи, та проведено вибір оптимального варіанту;
- розроблені проектні рішення по системі з різних видів забезпечення, передбачених етапами ескізного та технічного проектів, зокрема загальносистемні рішення, рішення з організаційного забезпечення, рішення з інформаційного забезпечення, рішення з технічного забезпечення, рішення з програмного забезпечення.
- розроблена робоча документація по системі;
- створено програмно-апаратну реалізацію прототипа системи

Отримані результати дозволяють зробити висновок, що у разі успішної реалізації та впровадження системи управління, підвищиться ефективність управління мікрокліматом та освітленням, що призведе до покращення умов проживання, що призведе загального підвищення якості життя.

Список літератури

1. Методичні вказівки до виконання дипломного проектування зі спеціальності 7.080401 «Інформаційні управляючі системи та технології». – К.В. Кошкін, С.Б. Приходько, С.В. Суслов, В.І. Тимофєєв. – Миколаїв: УДМТУ, 2003. – 52 с.
2. Кріс Шефер, Кларенс Хо, Роб Харроп. Spring 4 для професіоналів, APress, 2016
3. Роберт Лафоре, Структури даних та алгоритми Java, SAMS, 2016
4. Бретт Маклафлін. Об'єктно-орієнтований аналіз та проектування, O'REILLY Media, 2013
5. Сейед Тахагхогхі, Х'ю Є. Вільямс. Посібник з MySQL, O'REILLY Media, 2007
6. Герберт Шилдт. Java8. Повне керівництво, 9-е изд., М.:Вільямс, 2015
7. Внедрение SQL-кода. Вікіпедія — вільна енциклопедія. [Електроний ресурс]. – Режим доступу URL: http://uk.wikipedia.org/wiki/SQL_injection.
8. Межсайтовий скриптинг. Вікіпедія свободная энциклопедии. [Електроний ресурс]. – Режим доступу URL: <http://uk.wikipedia.org/wiki/Xss>.
9. Лановий О.Ф., Кобзєв С.В., Калякін С.В., Системи управління контентом і безпека WEB-сайтів. Системи обробки інформації. Харків; ХНУВС, 2010
10. Безпека життєдіяльності. Під редакцією Білова М.А. – К.: Наука, 2011.– 364с.

Додаток А – Технічне завдання на розробку

1 Вступ

1.1 Назва – Підсистема web-доступу в системі «Розумний дім»

1.2 Призначення програми: автоматизація роботи системи «Розумний дім» засобами web-технологій.

1.3 Область застосування: домашнє використання.

2 Призначення розробки

Призначена для автоматизації роботи кліматорегулюючих засобів, задля підтримання необхідних умов у житлових приміщеннях.

3 Вимоги до системи

3.1 Вимоги до функціональних характеристик

Система буде містити наступні функціональні характеристики :

- Збір інформації про кліматичні умови
- Виведення інформації про користувачів приміщення
- Автоматичне управління кліматичними системами
- Додання інформації в систему.

4.2 Вимоги до надійності

4.2.1 Вимоги до забезпечення надійності функціонування програми

Надійне (стійке) функціонування системи має бути забезпечено надійним (стійким) функціонуванням бездротової мережі і мережі електропостачання в офісному центрі.

4.2.2 Відмова виконання програмного продукту через некоректні дії користувача.

- Перевірка формату вводу даних для коректної роботи.
- Функція автоматичного backup кожен тиждень.

4.3 Умови експлуатації

Данною програмою будуть користуватися орендатори офісного центру, які є просунутими користувачами, та не потребують додаткового навчання по даному ПО.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Вимоги до інформаційних структур і методів розв'язання

Вимоги до інформаційних структур і методів розв'язання відсутні.

4.5.2 Вимоги до вихідних кодів і мов програмування

Додаткові вимоги не пред'являються.

4.5.3 Вимоги до програмних засобів , які використовуються програмою
Програмні засоби що використовуються мають бути вільно поширювані

4.6 Вимоги до маркування та упаковки

4.6.1 Вимоги до маркування :

- Читабельність тексту, чіткість ілюстрацій;
- Достовірність інформації про систему
- Нанесення на Install CD позначок про дане ПО.

4.6.2 Вимоги до упаковки:

Вимоги до упаковки відсутні.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування програми відсутня.

4.8 Спеціальні вимоги

Спеціальні вимоги відсутні.

5 Вимоги до програмної документації

5.1 Попередній склад програмної документації

Попередній склад програмної документації повинен включати в себе:

- технічне завдання ;
- устав проекту .

5.2 Склад програмної документації

Програмна документація складається з наступних програмних документів:

- Устав проекту;
- Технічне завдання ;
- Опис системи ;
- Код програми ;
- Керівництво користувача.

Сертифікація містить список всіх програмних документів .

Технічне завдання містить опис завдання роботи , склад функціональних характеристик програми , опис вимог для розробки.

Опис програми містить опис логічної структури програми , форматів вхідних та вихідних даних , використовуваних технічних засобів.

Код програми містить опис блоків програми і запис блоків в програмному коді.

Керівництво користувача містить інформацію , необхідну для ознайомлення користувача з програмним продуктом.

6 Техніко-економічні показники

6.1 Економічні переваги розробки

Орієнтовна економічна ефективність не розраховується.

7 Стадії та етапи розробки програмного забезпечення

7.1 Стадії розробки

Розробка повинна бути проведена в 5 стадій:

1. Технічне завдання
2. Ескізний проект
3. Технічний проект
4. Робочий проект
5. Впровадження

Додаток Б - Програма і методика випробувань

Об'єкт випробувань

Назва об'єкта випробувань: «Login.java» , надалі називається «програма».
Програма дозволяє користувачеві пройти аутентифікацію та отримати доступ до свого інтерфейсу.

Ціль випробувань

Перевірка придатності програми до використання за призначенням, її відповідність заданим вимогам і програмним документам. Перевірка реально досягнутих характеристик програми.

Вимоги до програми

Програма повинна задовольняти наступним функціональним характеристикам:

- 1) перевірка правильності імені та паролю користувача;
- 2) надання користувачеві доступу до свого інтерфейсу.

Користувач повинен вказати ім'я та пароль.

Вимоги до програмної документації

У комплект програмної документації повинні входити наступні документи:

- Загальний опис системи.
- Посібник користувача – власника товарів.
- Посібник користувача – дилера.
- Посібник користувача – покупця.
- Інструкція з формування і ведення бази даних.
- Посібник адміністратора з установки програмного забезпечення на серверах.
- Програма і методика випробувань.
- Текст програми.

Засоби і порядок випробувань

Вимоги до складу і параметрів технічних засобів:

Сервер на базі комп'ютера с архітектурою x86 або ARM

Вимоги до складу і параметрів програмних засобів :

- 1) операційна система Debian Jessie;
- 2) встановлене з'єднання з сервером баз даних MariaDB 10.1

Група людей, що будуть проводити випробування, повинна складатися з автора програми (розробника) і представника і/або самого замовника. При наявності відповідної можливості, ця група може бути розширена програмістом, спеціалістом-тестувальником і фахівцем-проектувальником, що не брали участь у розробці програми.

Попередньо (не пізніше 3 днів) до початку випробувань уточнюється кінцевий склад групи, що буде проводити випробування. Кожен член групи одержує необхідну документацію до програми (список вимог до програми, опис програми, керівництво користувача, програму і методику випробувань).

Порядок проведення випробувань:

- послідовно виконуються контрольні тести; результати заносяться у відповідний журнал;
- виробляється аналіз отриманих результатів і встановлюється відповідність програми заданим вимогам і програмним документам.

Методи випробувань

Робота програми залежить тільки від двох параметрів, що вводяться користувачем: від імені та паролю користувача.

Іспити будуть проводитися методом «чорної шухляди».

Тести будуть створюватися на основі класів еквівалентності.

Класи еквівалентності виділяються шляхом вибору кожної вхідної умови і розбивкою його на двох або більш груп. Розрізняють два типи класів еквівалентності: правильні класи еквівалентності, що представляють правильні вхідні дані програми, і неправильні класи еквівалентності, що представляють всі інші можливі стани умов (помилкові вхідні значення). При цьому існує ряд правил:

- один тест може покривати тільки один неправильний клас еквівалентності;
- один тест повинний покривати як можна більше число непокритих правильних класів еквівалентності.

Виділені класи еквівалентності для складання тестів мають унікальні номери і приводяться в таблиці Д.1.

Таблица Б.1 – Класи еквівалентності

Операція, що тестується	Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Перевірка імені та паролю користувача	Ім'я правильне	Так (1)	Ні (2)
	Пароль правильний	Так (3)	Ні (4)
Надання користувачеві доступу до свого інтерфейсу	Належність користувача до інтерфейсу встановлена	Так (5)	Ні (6)

Тести

Тести складені на основі класів еквівалентності таблиці Д.1. Тести використовують наступні початкові умови:

- 1) користувач – user1, його пароль – pass1 (власник товару);
- 2) користувач – admin, його пароль – adMpaSSw0rd (адміністратор системи).

Тест 1

Класи еквівалентності:

1, 3, 5

Вхідні дані:

Ім'я користувача: user1

Пароль користувача: pass1

Результат:

Ім'я та пароль користувача пізнані. Належність встановлена. Наданий доступ до інтерфейсу жителя.

Тест 2

Класи еквівалентності:

2, 3, 5

Вхідні дані:

Ім'я користувача: user2

Пароль користувача: pass1

Результати:

Ім'я та пароль користувача не пізнані. Належність до користувачів системи не встановлена. Видане повідомлення про помилку введення імені чи паролю. Доступу до системи не надано.

Тест 3

Класи еквівалентності :

1, 4, 5

Вхідні дані:

Ім'я користувача: user1

Пароль користувача: pass2

Результати:

Ім'я та пароль користувача не пізнані. Належність до користувачів системи не встановлена. Видане повідомлення про помилку введення імені чи паролю. Доступу до системи не надано.

Тест 4

Класи еквівалентності:

1, 3, 5

Вхідні дані:

Ім'я користувача: admin

Пароль користувача: adMpaSSw0rd

Результати:

Ім'я та пароль користувача впізнані. Наданий доступ до адміністрування системи.

Додаток В. - Інструкція користувача автоматизованої системи управління завданнями.

1. Призначення програми.

Програмне забезпечення інформаційної системи управління мікроклімату, яка призначена для автоматизації роботи кліматичних систем на будь-якому рівні. Функції, що автоматизуються системою, пов'язані з управлінням системою опалення, кондиціювання та вентилявання, а також моніторингом кліматичних умов.

Задачею інформаційної системи, є автоматизація наступних функцій користувача:

- Облік кліматичних умов.
- Моніторинг кліматичних умов.
- Формування звітів.

Для створення інформаційної системи необхідно розробити єдину базу даних для збереження інформації про кімнати та їх жителів, необхідні їм кліматичні умови. Також повинно бути передбачене програмне забезпечення для адміністрування системи ПЗ функціонує в мережі житлового будинку, обмін даними будується через єдину інформаційну базу.

2. Умови виконання програми.

Для виконання програми необхідна наявність на в будинку наступних апаратних і програмних засобів:

- робочі місця користувачів, що представляють собою персональні комп'ютери(ПК) з мінімальною конфігурацією Celeron +/- із встановленим веб оглядачем; мінімальний обсяг вільного місця на твердому диску 4Гб;

- локальної бездротової мережі для надання доступу мікроконтролерів до сервера з робочих місць користувачів доступу до даних, із пропускною здатністю 10/100Мбіт/с;

3. Виконання програми.

Виконання програми на кожному з робочих місць починається запуском веб оглядача та вводом адреси системи у мережі. Безпосередньо після цього Користувач системи повинний пройти авторизацію – ввести в діалоговому вікні, що з'явилося, користувальницьке ім'я і пароль. Пароль є унікальною комбінацією буквено-

цифрових символів, відомих тільки користувачу. Якщо ви забули пароль – звернетесь до системного адміністратора.

Після успішного входу в систему користувач бачить екран с інформацією про кліматичні умови а також які кліматичні системи вмикаєні.

На цьому ж екрані користувач може змінити налаштування своєї кімнати

- на екран де може змінити свої налаштунки (кількість секунд для оновлення екрану завданнів, отримувати листа при отриммані завдання, звіту або повідомлення), після заповнення форми користувач повинен натиснути на кнопку „Save” щоб дані буди збереженні у базі;

- на екран статистики де відображується статистика за обраний період.

Після вибору початкового періоду та кінцевого користувач бачить умови що були в вказаний термін,

- на екран повідомлень, де бачить повідомлення адресованні йому, або надіслані ним, він може їх видалити, створити нове натиском відповідної кнопки;

Користувач може вийти з системи натиском на відповідне посиляння з екрану статистики, він може завершити дію веб оглядача, але у цілях безпеки рекомендується виходити з системи.

4. Повідомлення користувачу.

При введенні некоректних даних над полями уведення виводиться діалогове вікно з описом типу і місця виникнення помилки.

При неправильному введенні чи пароля імені користувача виводиться повідомлення «Wrong login or password».

Додаток Г. - Тексти програмних модулів системи.

Occupant.java

```
package app.model;

import java.util.Objects;

public class Occupant {
    private int id;
    private String login;
    private String password;

    public Occupant(int id, String password, String login) {
        this.id = id;
        this.password = password;
        this.login = login;
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, login, password);
    }

    @Override
    public boolean equals(Object obj) {
        return super.equals(obj);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

```
    }  
}
```

OccupantDAO.java

```
package app.model;  
  
public interface OccupantDAO {  
    public void createOccupant(String login, String password);  
    public Occupant authOccupant(String login, String password);  
}
```

Room.java

```
package app.model;  
  
public class Room {  
    private int id;  
    private String name;  
  
    public Room(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

RoomDAO.java

```
package app.model;  
  
import java.util.List;  
  
public interface RoomDAO {  
    public List<Room> getAllRoom();  
}
```

```
        public List<Occupant> getRoomOccupants(int roomId);  
    }  
}
```

RoomLog.java

```
package app.model;  
  
public class RoomLog {  
    private int id;  
    private float t;  
    private float h;  
    private float co2;  
    private int timestamp;  
  
    public RoomLog(int id, int timestamp, float co2, float h,  
float t) {  
        this.id = id;  
        this.timestamp = timestamp;  
        this.co2 = co2;  
        this.h = h;  
        this.t = t;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public float getT() {  
        return t;  
    }  
  
    public void setT(float t) {  
        this.t = t;  
    }  
  
    public float getH() {  
        return h;  
    }  
  
    public void setH(float h) {  
        this.h = h;  
    }  
  
    public float getCo2() {  
        return co2;  
    }  
  
    public void setCo2(float co2) {
```



```

        this.co2 = co2;
    }

    public int getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(int timestamp) {
        this.timestamp = timestamp;
    }
}

```

RoomLogDAO.java

```

package app.model;

import java.util.List;

public interface RoomLogDAO {
    public List<RoomLog> getRoomLog(int roomId);
    public void insertRoomLogEntry(int roomId, RoomLog entry);
}

```

RoomSettings.java

```

package app.model;

public class RoomSettings {
    private int id;
    private float min_t;
    private float max_t;
    private float min_h;
    private float max_h;
    private float max_co2;

    public RoomSettings(int id, float max_co2, float max_h, float
max_t, float min_t, float min_h) {
        this.id = id;
        this.max_co2 = max_co2;
        this.max_h = max_h;
        this.max_t = max_t;
        this.min_t = min_t;
        this.min_h = min_h;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}

```

```

    public float getMinT() {
        return min_t;
    }

    public void setMinT(float min_t) {
        this.min_t = min_t;
    }

    public float getMaxT() {
        return max_t;
    }

    public void setMaxT(float max_t) {
        this.max_t = max_t;
    }

    public float getMinH() {
        return min_h;
    }

    public void setMinH(float min_h) {
        this.min_h = min_h;
    }

    public float getMaxH() {
        return max_h;
    }

    public void setMaxH(float max_h) {
        this.max_h = max_h;
    }

    public float getMaxCo2() {
        return max_co2;
    }

    public void setMaxCo2(float max_co2) {
        this.max_co2 = max_co2;
    }
}

```

RoomSettingsDAO.java

```

package app.model;

public interface RoomSettingsDAO {
    public RoomSettings getRoomSettings(int roomId);
    public void setRoomSettings(RoomSettings settings);
}

```

Utils.java

```
package app;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class Utils {

    private static Connection connection = null;

    public static String md5(String st) {
        MessageDigest messageDigest;
        byte[] digest = new byte[0];

        try {
            messageDigest = MessageDigest.getInstance("MD5");
            messageDigest.reset();
            messageDigest.update(st.getBytes());
            digest = messageDigest.digest();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        BigInteger bigInt = new BigInteger(1, digest);
        String md5Hex = bigInt.toString(16);

        while( md5Hex.length() < 32 ){
            md5Hex = "0" + md5Hex;
        }

        return md5Hex;
    }

    public static Connection getConnection() {
        if (connection != null)
            return connection;
        else {
            try {
                Properties prop = new Properties();
                InputStream inputStream =
Utils.class.getClassLoader().getResourceAsStream("/db.properties");
                prop.load(inputStream);
                String driver = prop.getProperty("driver");
                String url = prop.getProperty("url");
                String user = prop.getProperty("user");
                String password = prop.getProperty("password");
                Class.forName(driver);
            }
        }
    }
}
```

```

        connection = DriverManager.getConnection(url, user,
password);
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return connection;
}

}

}

```

Esp8266-sender-sketch.ino

```

#include <ArduinoJson.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <ESP8266HTTPClient.h>
#include <TaskScheduler.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#include <MQ135.h>

const char *ssid = "wifi-ssid";
const char *password = "password";
const char *urlServer = "http://hostname:port/service";
const char *roomName = "room1";

void jsonTaskCallback();
void serverTaskCallback();

Scheduler runner;
ESP8266WebServer server ( 80 );
HTTPClient http;
Task jsonTask(TASK_SECOND, TASK_FOREVER, &jsonTaskCallback);
Task serverTask(TASK_IMMEDIATE, TASK_FOREVER,
&serverTaskCallback);

#define DHTPIN 2
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define MQ135PIN A0

```

```

MQ135 mq(MQ135PIN);

String getHttpData( String URI ) {
    http.begin( URI.c_str() );
    int code = http.GET();
    if(code > 0) {
        if( code == HTTP_CODE_OK ) {
            return http.getString();
        }
    }
    return "Error";
}

void sendPostData( String data ) {
    http.begin(urlServer);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    http.POST(data.c_str());
    http.writeToStream(&Serial);
    http.end();
}

void serverTaskCallback() {
    server.handleClient();
    yield();
}

void jsonTaskCallback() {
    Serial.println(millis());
    char tchar[6], hchar[6], co2char[6];
    char data[255];
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    float co2ppm = mq.getPPM();
    dtostrf(h, 4, 2, hchar);
    dtostrf(t, 4, 2, tchar);
    dtostrf(co2ppm, 4, 2, co2char);

    StaticJsonBuffer<255> jsonBuffer;
    JsonObject& obj = jsonBuffer.createObject();
    obj["room"] = roomName;
    obj["humidity"] = hchar;
    obj["temperature"] = tchar;
    obj["co2"] = co2char;

    obj.printTo(data, sizeof(data));

    sendPostData(String(data) + String("\r\n\r\n"));
    yield();
}

void handleRoot() {

```

```

    char temp[700];
    char ipaddr[16];
    char clientaddr[16];
    IPAddress ip = WiFi.localIP();
    IPAddress ipcli = server.client().remoteIP();
    int sec = millis() / 1000;
    int min = sec / 60;
    int hr = min / 60;

    String ipJson = getHttpData( "http://api.ipify.org/?format=json"
);
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject& root = jsonBuffer.parseObject( ipJson.c_str() );
    const char* extip = root[ "ip" ];

    snprintf( ipaddr, 16, "%d.%d.%d.%d", ip[0], ip[1], ip[2], ip[3]
);
    snprintf( clientaddr, 16, "%d.%d.%d.%d", ipcli[0], ipcli[1],
ipcli[2], ipcli[3] );

    snprintf ( temp, 600,
"<html>\
<head>\
  <meta http-equiv='refresh' content='5' />\
  <title>ESP8266 Demo</title>\
  <style>\
    body { background-color: #cccccc; font-family: Arial,
Helvetica, Sans-Serif; Color: #000088; }\
  </style>\
</head>\
<body>\
  <h1>Hello from ESP8266!</h1>\
  <p>Uptime: %02d:%02d:%02d</p>\
  <p>Your IP is: %s</p>\
  <p>Connected to: %s</p>\
  <p>IP Address: %s</p>\
  <p>External IP Address: %s</p>\
  <p>Signal strength: %d</p>\
</body>\
</html>",

        hr, min % 60, sec % 60, clientaddr, WiFi.SSID().c_str(),
ipaddr, extip, WiFi.RSSI()
);
    server.send ( 200, "text/html", temp );
}

void handleNotFound() {
    String message = "File Not Found\n\n";
    message += "URI: ";
    message += server.uri();
    message += "\nMethod: ";

```

```

message += ( server.method() == HTTP_GET ) ? "GET" : "POST";
message += "\nArguments: ";
message += server.args();
message += "\n";

for ( uint8_t i = 0; i < server.args(); i++ ) {
    message += " " + server.argName ( i ) + ": " + server.arg ( i
) + "\n";
}

server.send ( 404, "text/plain", message );
}

void setup ( void ) {
    pinMode(D0, OUTPUT);
    digitalWrite(D0, HIGH);
    Serial.begin ( 115200 );
    WiFi.begin ( ssid, password );
    Serial.println ( "" );

    while ( WiFi.status() != WL_CONNECTED ) {
        delay ( 500 );
        Serial.print ( "." );
    }

    Serial.println ( "" );
    Serial.print ( "Connected to " );
    Serial.println ( ssid );
    Serial.print ( "IP address: " );
    Serial.println ( WiFi.localIP() );

    if ( MDNS.begin ( "esp8266" ) ) {
        Serial.println ( "MDNS responder started" );
    }

    server.on ( "/", handleRoot );
    server.onNotFound ( handleNotFound );
    server.begin();
    Serial.println ( "HTTP server started" );
    dht.begin();

    float rzero = mq.getRZero();
    delay(3000);
    Serial.print("MQ135 RZERO Calibration Value : ");
    Serial.println(rzero);

    runner.init();
    runner.addTask(jsonTask);
    runner.addTask(serverTask);
    serverTask.enable();
    jsonTask.enable();
}

```

```
void loop ( void ) {  
    runner.execute();  
}
```