Міністерство освіти і науки України
Харківський національний університет
радіоелектроніки

Факультет Автоматики і комп'ютеризированих технологій
(повна назва)

Кафедра Комп'ютерно інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Перший (бакалаврський)
рівень вищої освіти

Розробка системи автоматизації для розсилки email повідомлень по результатам аналізу подій з автоматизованою розсилкою в месенджери
(тема)

Виконав:
студент _4_ курсу, групи АКТСІ-20-2
Мороз М. В.
(прізвище, ініціали)

Спеціальність _151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
Освітня програма Системна інженерія
(повна назва освітньої програми)

Керівник доц. Іванов Л. С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАР _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2024 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти Перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно інтегровані технології
Тип програми Освітньо-професійна
Освітня програма Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«18» червня 2024 р.
.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Морозу Максиму Васильовичу
(прізвище, ім'я, по батькові)

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Морозу Максиму Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи автоматизації для розсилки email повідомлень по результатам аналізу подій з автоматизованою розсилкою в месенджери
Затверджена наказом по університету від 03.06.2024 №545 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024
3. Вхідні дані до роботи
3.1 SMTP, MSSQL Server, ASP.NET Core, Angular, EF Core, Telegram Bot API
3.2 Технологія HTTP
4. Перелік питань, що потрібно розглянути у роботі
4.1 Вступ
4.2 Огляд теми та її актуальності
4.3 Вибір та обґрунтування компонентів системи
4.4 Розробка програмної частини
4.5 Тестування та налагодження системи
4.6 Висновки та перелік джерел посилань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій
Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	22.01-04.02	Виконано
2	Опрацювання літератури за темою	05.02-07.04	Виконано
3	Виконання розділу 1 «Теоретичний аналіз предметної області»	08.04-24.04	Виконано
4	Виконання розділу 2 «Дослідження існуючих технологій та методів автоматизації розсилки повідомлень»	25.04-07.05	Виконано
5	Виконання розділу 3 «Розробка програмної частини підсистеми розсилки»	08.05-21.05	Виконано
6	Виконання розділу 4 «Експлуатація програмної системи розсилки»	22.05-03.06	Виконано
6	Оформлювання пояснювальної записки	05.06-14.06	Виконано

Дата видачі завдання 25.01.2024

Студент _____ Мороз М.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц Іванов Л.С.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«20» червня 2024р. Мороз М. В.

РЕФЕРАТ

Пояснювальна записка містить: 51 с., 3 рис., 2 дод., 23 джерел.

АВТОМАТИЗАЦІЯ РОЗСИЛКИ, СИСТЕМА ПОВІДОМЛЕНЬ, СУЧАСНІ ТЕХНОЛОГІЇ.

Об’єкт розробки – процес автоматизованої розсилки повідомлень по результатам аналізу подій.

Предмет розробки – автоматизована система розсилки повідомлень через електронну пошту та месенджери.

Методи розробки – теоретичний аналіз технічної літератури з проблеми дослідження, а також систематизація та узагальнення попередніх досліджень в області розробки програмного забезпечення для автоматизації розсилки повідомлень.

Метою даної кваліфікаційної роботи є забезпечення оперативного та ефективного інформування користувачів про критичні події шляхом автоматизації процесу розсилки повідомлень

У роботі було проведено аналіз сучасних технологій розсилки повідомлень, вибрано та обґрунтовано технічні засоби для створення системи, побудовано архітектуру програмного модуля, розроблено систему автоматизації розсилки повідомлень по результатам аналізу подій через електронну пошту та Telegram.

ABSTRACT

Explanatory note contains: 51 p., 3 fig., 2 app., 23 sources.

AUTOMATION OF MAILING, NOTIFICATION SYSTEM, MODERN TECHNOLOGIES

The object of development is the process of automated distribution of messages based on the results of event analysis.

Subject of development – an automated system for sending messages via e-mail and instant messengers

Methods of development – theoretical analysis of technical literature on the research problem, as well as systematisation and generalisation of previous studies in the field of software development for automated notification.

The purpose of this qualification work is to ensure prompt and efficient informing of users about critical events by automating the notification process

In this work, an analysis of modern technologies for sending messages was carried out, technical means for creating the system were selected and substantiated, the architecture of the software module was built, and a system for automating the distribution of messages based on the results of event analysis via e-mail and Telegram was developed..

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface;

ASP.NET – Active Server Pages .NET;

EF Core – Entity Framework Core;

HTTP – Hypertext Transfer Protocol;

SMTP – Simple Mail Transfer Protocol;

SQL – Structured Query Language;

MSSQL – Microsoft SQL Server;

Wi-Fi – Wireless Fidelity;

WPF –Windows Presentation Foundation;

MVC – Model-View-Controller;

Telegram Bot API – Application Programming Interface;

JSON – JavaScript Object Notation;

CI/CD – Continuous Integration/Continuous.

ВСТУП

У сучасному цифровому середовищі автоматизація процесів стає ключовим фактором ефективного функціонування бізнесу. Особливо це стосується обробки та розсилки інформації, яка має велике значення для підтримки зв'язку з клієнтами та оптимізації внутрішніх процесів компанії. У цьому контексті розробка програмного забезпечення для автоматизації розсилки email повідомлень та інтеграції з месенджерами стає актуальною задачею.

Мета розробки – є забезпечення оперативного та ефективного інформування користувачів про критичні події шляхом автоматизації процесу розсилки повідомлень.

Об’єкт розробки – програмна система, яка буде здійснювати автоматизовану розсилку email повідомлень та інтеграцію з месенджерами. Це включає в себе створення зручного інтерфейсу для введення та обробки даних, розробку алгоритмів для аналізу подій та генерації відповідних повідомлень, а також інтеграцію з API месенджерів для надсилання повідомлень у ці платформи.

Предмет розробки – створення функціонального програмної системи, яка дозволить автоматизувати процес розсилки інформації по email та месенджерам на основі результатів аналізування подій. Це включає в себе розробку системи збору та обробки даних, а також інструменти для створення та налаштування розсилок з мінімальною залученістю людського фактора.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– створення зручного та інтуїтивно зрозумілого інтерфейсу для введення необхідних даних, вибору параметрів розсилок та моніторингу процесу;

– створення алгоритмів для обробки вхідних даних та визначення результуючих повідомлень для розсилки, з урахуванням унікальних потреб користувача;

– реалізація інтеграції з поштовими сервісами для автоматичної розсилки email повідомлень згідно з встановленими розкладами та параметрами;

– розробка засобів для взаємодії з різними месенджерами (наприклад, Telegram, WhatsApp, Slack тощо) з метою надсилання повідомлень у ці платформи;

– проведення комплексного тестування розробленого програмного забезпечення для переконанняся в його працездатності, надійності та відповідності вимогам.

Ця тема була частково розкрита у статті ADED, де розглядалися основні аспекти автоматизації розсилки повідомлень та їх значення для сучасних інформаційних систем. Дослідження, проведене в рамках даної статті, заклало основу для розробки даної системи і стало важливим кроком у вирішенні поставлених завдань.

Кваліфікаційну роботу оформлено згідно з ДСТУ 3008:2015 [1], а також з рекомендаціями з підготовки і оформлення кваліфікаційної роботи здобувачами першого (бакалаврського) рівня вищої освіти [2-3].

1 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Автоматизація

Автоматизація є важливим аспектом сучасного життя, який змінює спосіб, яким люди виконують завдання та управляють ресурсами. За допомогою автоматизації можна досягти підвищення ефективності, зниження витрат та помилок, а також швидкого реагування на зміни в середовищі

Автоматизація – це процес застосування різних технологій та методів для автоматичного виконання завдань або процесів, що раніше виконувалися вручну. Цей процес має на меті збільшення ефективності, точності та продуктивності, а також зниження витрат і ризиків.[3]

Роль автоматизації у різних галузях діяльності надзвичайно важлива і широка:

– у сфері виробництва автоматизація дозволяє підвищити продуктивність та якість, знизити витрати та мінімізувати вплив людських факторів на процес виробництва. Автоматизовані лінії збирання, роботи з числовим контролем та інші технології дозволяють виробникам оптимізувати виробничі процеси;

– у галузі IT автоматизація використовується для автоматизації процесів розробки програмного забезпечення, тестування, розгортання та управління інфраструктурою. Це допомагає знизити час виконання завдань, покращити якість продукту та зменшити ризики помилок;

– у сфері фінансів автоматизація дозволяє забезпечити точний облік фінансів, автоматизувати процеси оподаткування, аналізу даних та управління ризиками. Використання спеціалізованих програмних засобів дозволяє бухгалтерам та фінансовим аналітикам ефективно керувати фінансовими процесами;

– у сфері офісної діяльності автоматизація використовується для автоматизації рутинних офісних завдань, таких як обробка документів, управління

електронною поштою, календарні та планувальні задачі, а також ведення баз даних клієнтів та контактів.[9]

Основні принципи та завдання автоматизації включають в себе ряд важливих аспектів, спрямованих на досягнення ефективності, надійності та оптимізації процесів. Ось деякі з них:

– основна мета автоматизації полягає в підвищенні продуктивності та зниженні витрат часу та ресурсів. Автоматизація повинна допомагати виконувати завдання швидше та ефективніше, ніж це можна зробити вручну;

– автоматизація дозволяє зменшити кількість помилок, які можуть бути вчинені людиною при виконанні рутинних завдань. Це досягається завдяки програмному забезпеченню та механізмам контролю, що вбудовані в автоматизовані системи;

– автоматизація може розширюватися від простих автоматизованих завдань до складних систем, які включають в себе різні технології та інтеграцію з іншими системами;

– важливо, щоб автоматизовані системи були надійними і стійкими до збоїв. Це вимагає ретельного тестування та контролю якості, а також впровадження механізмів відновлення у разі виникнення проблем.

Основні завдання:

– основна мета автоматизації полягає в автоматизації рутинних та повторюваних процесів, щоб звільнити людей від монотонних завдань та дозволити їм зосередитися на більш складних або стратегічних завданнях;

– автоматизація допомагає оптимізувати використання ресурсів, таких як людська праця, матеріали та час, що призводить до ефективнішого використання ресурсів та зменшення витрат;

– автоматизовані системи можуть забезпечувати стандартизацію та уніфікацію процесів, що сприяє підвищенню якості продукції чи послуг;

– автоматизація може допомогти виконувати завдання швидше, що важливо в сучасному динамічному світі бізнесу;

– автоматизація також дозволяє збирати та аналізувати дані про виконання процесів, що дозволяє приймати обґрунтовані рішення та вносити покращення в систему.

Технології автоматизації охоплюють широкий спектр інноваційних рішень та інструментів, спрямованих на автоматизацію процесів і оптимізацію робочих потоків.

Штучний інтелект (AI) та машинне навчання (ML) використовуються для розробки систем, які можуть аналізувати дані, виявляти патерни та приймати рішення без прямого втручання людини. Вони застосовуються у таких областях, як автоматизація обробки даних, управління процесами та прогнозування.[5]

Роботизований процес автоматизації (RPA) використовується для автоматизації рутинних завдань та процесів, що вимагають взаємодії з різними системами. Роботи RPA можуть виконувати повторювані завдання, швидко та без помилок, що дозволяє звільнити час працівників для виконання більш складних завдань.[5]

Інтернет речей (IoT) використовується для збору та обміну даними між фізичними пристроями, що дозволяє автоматизувати процеси моніторингу, управління та контролю за об'єктами. Наприклад, виробництво, будівництво, логістика та домашні системи безпеки.[5]

Блокчейн технологія може бути використана для створення децентралізованих систем, які автоматизують процеси обміну даними та управління документами між сторонами без необхідності посередника.

Автоматизація процесів роботи (BPA) використовується для автоматизації бізнес-процесів, включаючи створення, виконання та контроль за процесами роботи. Він дозволяє оптимізувати потік роботи та підвищує ефективність діяльності підприємства.[5]

Ці технології постійно розвиваються та вдосконалюються, що дозволяє організаціям впроваджувати все більше автоматизованих рішень для підвищення продуктивності та конкурентоспроможності.

Автоматизація відіграє величезну роль у сучасному бізнесі, забезпечуючи підприємствам і організаціям значні переваги. Перш за все, вона спрощує багато рутинних і монотонних завдань, які раніше вимагали великої кількості часу та ресурсів. Завдяки автоматизації, багато процесів тепер можуть бути виконані швидше і ефективніше.

Одна з головних переваг автоматизації – це підвищення продуктивності. Автоматизовані системи дозволяють зменшити час, необхідний для виконання завдань, та збільшити кількість роботи, яку можна зробити за одиницю часу. Це може значно підвищити виробничу потужність та ефективність роботи.

Автоматизація допомагає знизити витрати підприємства. Замість того, щоб витрачати гроші на оплату праці для виконання рутинних завдань, компанії можуть використовувати автоматизовані системи, які вимагають менше людського втручання. Це дозволяє зекономити кошти та збільшити прибуток.

Покращення якості продукції та послуг також є важливою перевагою автоматизації. Завдяки автоматизованим системам контролю та моніторингу, компанії можуть забезпечити більшу точність та стабільність у виробництві. Це може підвищити задоволеність клієнтів та сприяти збільшенню їх лояльності.

Крім того, автоматизація дозволяє підприємствам бути більш гнучкими та реагувати швидше на зміни на ринку. Замість того, щоб витрачати час на рутинні процеси, компанії можуть швидко адаптуватися до нових умов і вимог ринку, що дозволяє їм залишатися конкурентоспроможними.

Узагальнюючи, автоматизація допомагає підприємствам знижувати витрати, підвищувати продуктивність та якість, а також бути більш гнучкими та конкурентоспроможними.

1.2 Програмне забезпечення

Програмне забезпечення визначає сучасний світ інформаційних технологій, що перетворюється на необхідний елемент для функціонування практично будь-якої галузі діяльності. Від мобільних додатків до великих корпоративних систем,

програмне забезпечення забезпечує різноманітні функції, сприяючи ефективному виконанню завдань та підвищенню продуктивності.

В контексті швидко розвиваючого цифрового середовища розуміння ролі, переваг та викликів, що стосуються програмного забезпечення, стає докорінно важливим для будь-якої організації чи особистого користувача. Цей реферат спрямований на дослідження ключових аспектів програмного забезпечення, зокрема його ролі в сучасному світі, переваг та викликів використання, а також майбутніх перспектив.

Програмне забезпечення – це набір програмних інструкцій, які визначають роботу комп'ютера або іншого пристрою. Ці програми можуть мати різну функціональність і призначення, від операційних систем і прикладних програм до ігор та інтернет-послуг.[10]

Огляд програмного забезпечення дозволяє розглянути його різноманітність та важливість у сучасному світі. Від розширення бізнесу до полегшення повсякденних завдань користувачів, програмне забезпечення стає необхідним елементом у всіх сферах діяльності.

Його можна класифікувати за різними критеріями, такими як функціональність, призначення, тип ліцензування тощо. Основні типи програмного забезпечення включають операційні системи, додаткове програмне забезпечення, веб-додатки, мобільні додатки та ігри.

Також, програмне забезпечення може бути відкритим або закритим джерелом, що впливає на його доступність та модифікацію. Відкрите програмне забезпечення зазвичай має відкритий вихідний код та може бути змінено користувачами згідно з їхніми потребами та умовами ліцензії, тоді як закрите програмне забезпечення має обмежений доступ до вихідного коду та зазвичай потребує плати за використання.

У світлі швидкого темпу технологічного розвитку важливо визнати, що програмне забезпечення постійно змінюється та вдосконалюється. Інновації, нові технології та вимоги ринку визначають напрямок розвитку програмного забезпечення, зробивши його необхідним елементом для ефективного

функціонування бізнесу, підвищення продуктивності та полегшення повсякденних завдань користувачів.

Роль програмного забезпечення в сучасному світі є визначальною і важливою у різних сферах діяльності. У бізнесі воно використовується для автоматизації процесів, управління даними та аналітики, що дозволяє підприємствам оптимізувати свою діяльність і підвищувати продуктивність. В освіті програмне забезпечення надає можливості для навчання та розвитку, забезпечуючи доступ до навчальних матеріалів, інтерактивних курсів та віртуальних навчальних середовищ.[10]

У медицині програмне забезпечення використовується для ведення медичних записів, діагностики захворювань, моніторингу пацієнтів та планування лікування. У науці воно використовується для обробки та аналізу великих обсягів даних, моделювання складних систем і проведення досліджень. У розважальній індустрії програмне забезпечення відіграє ключову роль у розробці відеоігор, анімації та мультимедійних вмістів.[10]

Програмне забезпечення допомагає забезпечити безпеку даних, захист від кібератак та вірусів, а також забезпечує можливості для співпраці та комунікації як в межах організації, так і в онлайн середовищі. Враховуючи всі ці аспекти, можна сказати, що програмне забезпечення відіграє ключову роль у підтримці та розвитку сучасного інформаційного суспільства, сприяючи його ефективному функціонуванню та подальшому розвитку.

Переваги використання програмного забезпечення в сучасному світі є надзвичайно важливими і різноманітними. По-перше, програмне забезпечення дозволяє автоматизувати багато рутинних операцій і процесів, що полегшує роботу користувачів і зменшує витрати часу на виконання завдань. Воно також підвищує продуктивність, оскільки багато завдань можуть бути виконані швидше та ефективніше за допомогою програмного забезпечення.[10]

Крім того, програмне забезпечення дозволяє забезпечити високу якість роботи та послуг. Воно може автоматизувати процеси контролю якості, а також забезпечити стабільну та надійну роботу системи. Програмне забезпечення також

спрощує доступ до інформації та ресурсів, що полегшує співпрацю та комунікацію між користувачами.

Ще однією важливою перевагою програмного забезпечення є його гнучкість і масштабованість. Воно може бути легко змінено або оновлено для відповіді на змінюються потреби та вимоги користувачів. Крім того, програмне забезпечення може бути масштабоване для роботи з різними обсягами даних та величинами організацій, що дозволяє йому ефективно функціонувати в різних умовах.

Нарешті, використання програмного забезпечення допомагає знизити витрати на бізнес та збільшити прибутковість. Воно дозволяє знизити витрати на робочу силу та ресурси, а також оптимізує витрати на інфраструктуру та обслуговування. У результаті цього підприємство може збільшити свою конкурентоспроможність і прибутковість, що є ключовими факторами успіху в сучасному бізнесі.

Програмне забезпечення може бути написане на різних мовах програмування та використовувати різні технології і платформи в залежності від його призначення, вимог до продукту, та відповідності стандартам індустрії. Ось деякі з найпоширеніших мов програмування та платформ для написання програмного забезпечення:

– Java використовується для написання різноманітних програм, включаючи веб-додатки, мобільні додатки (з використанням Android SDK), інтернет-платформи тощо;

– Python широко використовується для розробки веб-додатків, наукових обчислень, штучного інтелекту, обробки даних та інших завдань;

– C++ використовується для розробки високопродуктивних систем, вбудованих програм, ігор, операційних систем тощо;

– C# – мова програмування, яка часто використовується для розробки програм для платформи Microsoft .NET, включаючи веб-додатки, десктопні програми та ігри;

– JavaScript використовується для розробки веб-додатків та веб-сайтів, часто використовується у поєднанні з HTML і CSS для створення інтерактивних елементів;

– Ruby використовується для розробки веб-додатків (особливо за допомогою фреймворку Ruby on Rails), скриптів та автоматизації завдань;

– Swift використовується для розробки програм для платформи Apple, зокрема мобільних додатків для iOS та macOS;

– PHP широко використовується для розробки веб-додатків, особливо на платформі WordPress та інших веб-фреймворках;

– SQL – мова запитів, яка використовується для роботи з базами даних та обробки даних.

1.3 Розсилка email-повідомлень

Розсилка email-повідомлень – це ефективний інструмент для зв'язку з аудиторією, незалежно від масштабу бізнесу чи організації. Цей розділ розглядає процес розсилки email-повідомлень, його переваги, методи та кращі практики.

Мета полягає в тому, щоб ефективно спілкуватися з аудиторією через електронну пошту. Це може бути здійснено для різних цілей, включаючи, але не обмежуючись, рекламу продуктів або послуг, інформування про акції або спеціальні пропозиції, нагадування про події або терміни, розсилання новин або корисної інформації, а також з метою підтримки відносин з клієнтами.[4]

Основна мета розсилки email-повідомлень – це залучення уваги аудиторії та стимулювання певних дій. Це може бути покупка продукту або послуги, реєстрація на подію або вебінар, підписка на новини або розсилку, відвідування веб-сайту або розширення знань про бренд чи компанію.[8]

Розсилка email-повідомлень дозволяє встановити особистий зв'язок з аудиторією, спонукати до взаємодії та зберігати зацікавленість клієнтів.

Розсилка email-повідомлень може використовуватися для підтримки існуючих відносин з клієнтами. Це може включати надання корисної інформації, порад або рекомендацій, відповіді на запитання або запити на думку, а також реагування на відгуки чи скарги. Це допомагає зберігати лояльність клієнтів до бренду та позитивне сприйняття компанії в цілому.[11]

Розсилка має численні переваги, які допомагають компаніям та брендам будувати та підтримувати взаємодію з клієнтами, залучати нових клієнтів та збільшувати продажі. По-перше, це дуже ефективний інструмент комунікації, оскільки електронна пошта широко використовується в сучасному світі. Розсилка може допомогти підприємствам досягти великої аудиторії швидко та ефективно, зокрема, залучаючи увагу нових клієнтів та зберігаючи існуючих.[4]

Також, розсилка email-повідомлень дозволяє персоналізувати повідомлення для кожного отримувача, що підвищує ефективність комунікації та створює особистий зв'язок з клієнтом. Це важливо для збереження інтересу аудиторії та підвищення ефективності маркетингових кампаній. Крім того, завдяки аналітиці та відстеженню реакції аудиторії, компанії можуть адаптувати свої стратегії розсилки та оптимізувати їх для досягнення найкращих результатів.Розсилка email-повідомлень також дозволяє підприємствам економити час та гроші, оскільки вона є відносно недорогою та автоматизованою формою комунікації. В порівнянні з традиційними методами маркетингу, такими як друковані рекламні матеріали або реклама в ЗМІ, розсилка email-повідомлень може бути набагато ефективнішою і економічнішою. Крім того, вона дозволяє швидко реагувати на зміни на ринку та негайно сповіщати клієнтів про акції, новини або важливі події, що допомагає зміцнити позиції бренду та підвищити його конкурентоспроможність.

Для розсилки електронних повідомлень, бізнеси та організації використовують спеціалізовані програми та сервіси електронної пошти. Ці програми дозволяють створювати та відправляти електронні листи, включаючи текстові повідомлення, зображення, відео та інші медіа-файли. Вони також надають можливості для персоналізації повідомлень, відстеження відкриття та взаємодії з листом, а також аналізу результатів розсилки.

Окрім електронної пошти, дедалі більше компаній використовують месенджери для комунікації з клієнтами та співробітниками. Це може включати використання популярних месенджерів, таких як WhatsApp, Telegram, Viber, а також корпоративні месенджери, які спеціалізуються на комунікації в бізнес-середовищі.

Для автоматизації розсилки електронних листів та повідомлень у месенджерах, розробники можуть використовувати різні програмні інтерфейси (API) та сервіси, що надають зручний інтерфейс для інтеграції зі сторонніми додатками. Це дозволяє автоматизувати відправку повідомлень, розподіляти їх за різними категоріями або групами, а також аналізувати результати комунікаційних кампаній.

Вибір програмного забезпечення на мові C# для розробки програмного забезпечення для розсилки електронних листів та повідомлень у месенджери є важливим та обґрунтованим кроком. C# входить до екосистеми .NET, розробленої Microsoft, і він може бути оптимальним вибором, особливо якщо у вже є досвід роботи з цією мовою або з платформою .NET.

Існує кілька переваг, які варто врахувати при розгляді використання C# для цієї цілі:

– C# має багато вбудованих функцій та бібліотек, що полегшує розробку різних функцій, включаючи розсилку електронних листів та повідомлень у месенджери;

– використання C# дає можливість розробляти різноманітні програми, такі як веб-додатки, десктопні програми та служби Windows, які можуть використовувати різні компоненти платформи .NET;

– .NET Core дозволяє розгортати програми на різних платформах, що робить його відмінним вибором для крос-платформених рішень;

– .NET має вбудовані засоби безпеки, що забезпечує надійність та стабільність програм, написаних на C#;

– існують багато сторонніх інструментів та бібліотек, що допомагають з розробкою на C# та полегшують процес створення програм.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ ТЕХНОЛОГІЙ ТА МЕТОДІВ АВТОМАТИЗАЦІЇ РОЗСИЛКИ ПОВІДОМЛЕНЬ

2.1 Приклади існуючих рішень

Сучасні системи автоматизації email розсилок широко використовуються у різних сферах діяльності для підтримання зв'язку з клієнтами, партнерами та користувачами. Одним з найпопулярніших рішень є Mailchimp, який дозволяє автоматизувати відправку листів за заданими сценаріями. Цей сервіс пропонує широкий спектр функцій, включаючи сегментацію аудиторії, персоналізацію повідомлень, інтеграцію з різними CRM системами та надання детальної аналітики за результатами розсилок. Ще одним прикладом є Sendinblue, який також надає можливості для ефективного управління email кампаніями. Ця платформа дозволяє створювати та управляти списками контактів, налаштовувати тригери для автоматичної відправки повідомлень та аналізувати результати розсилок для подальшої оптимізації стратегії.[4]

Для моніторингу температури існують спеціалізовані платформи, які забезпечують збирання та передачу даних з датчиків на центральний сервер для подальшого аналізу та відображення. Наприклад, SensorPush та TempStick є популярними рішеннями в цій галузі. Вони дозволяють відстежувати температурні показники в режимі реального часу та отримувати сповіщення у разі перевищення заданих порогових значень. Вбудовані функції цих платформ дозволяють відправляти сповіщення у месенджери, такі як Telegram та WhatsApp, що забезпечує оперативне реагування на зміну умов.

Функціонал систем автоматизації email розсилок включає в себе створення та управління списками контактів, налаштування тригерів для відправки повідомлень, інтеграцію з різними CRM системами та надання детальної аналітики за результатами розсилок. Наприклад, платформи, такі як Mailchimp і Sendinblue, дозволяють сегментувати аудиторію на основі різних критеріїв, що підвищує

ефективність розсилок. Вони також підтримують персоналізацію повідомлень, що робить комунікацію з користувачами більш індивідуальною та релевантною.

Платформи для моніторингу температури, в свою чергу, надають можливість віддаленого моніторингу, налаштовувані сповіщення, збереження історичних даних та генерацію звітів. Наприклад, SensorPush дозволяє користувачам контролювати температуру у приміщеннях, де встановлені датчики, та отримувати сповіщення у разі відхилення від заданих параметрів. Це забезпечує можливість оперативного реагування на зміни температурних умов, що особливо важливо для зберігання чутливих до температури товарів.

Таким чином, існуючі рішення у сфері автоматизації email розсилок та моніторингу температури вже сьогодні пропонують широкі можливості для бізнесу та приватних користувачів. Однак, ці рішення мають певні недоліки, які обмежують їх ефективність та функціональність, що і буде розглянуто у наступному розділі.

2.2 Недоліки існуючих систем

Незважаючи на широке застосування та функціональність існуючих систем автоматизації email розсилок та платформ для моніторингу температури, вони мають певні недоліки, які обмежують їхню ефективність і зручність використання. Одним із основних недоліків є обмеження у гнучкості налаштувань. Багато з цих систем мають фіксовані сценарії та обмежені можливості для налаштування під конкретні потреби користувача. Це особливо важливо для бізнесів, які потребують індивідуального підходу до управління своїми процесами і комунікаціями.

Ще однією проблемою є інтеграція різних систем. Часом важко або навіть неможливо забезпечити безперебійну взаємодію між платформами для моніторингу та автоматизації розсилок. Наприклад, якщо потрібно інтегрувати дані з різних джерел, таких як датчики температури та CRM системи, можуть виникати проблеми з сумісністю форматів даних та протоколів обміну інформацією. Це призводить до додаткових витрат на налаштування та підтримку таких інтеграцій, що збільшує загальну вартість володіння системою.

Недостатня масштабованість є ще одним значним недоліком багатьох існуючих рішень. Зі збільшенням обсягів даних та кількості користувачів системи можуть втрачати продуктивність і стабільність. Це особливо актуально для великих підприємств, які обробляють великі обсяги даних в режимі реального часу і потребують високої надійності та швидкодії системи. В таких випадках стандартні рішення можуть не впоратися з навантаженням, що призводить до необхідності пошуку альтернативних варіантів або розробки власних рішень.

Також варто зазначити обмеження в аналітиці та звітності. Багато існуючих систем не забезпечують достатньо глибокого аналізу зібраних даних, що обмежує можливості користувачів у прийнятті обґрунтованих рішень. Наприклад, у системах автоматизації email розсилок може бракувати детальної інформації про поведінку користувачів, ефективність різних сегментів аудиторії та інші важливі метрики. Це робить процес оптимізації розсилок менш ефективним та трудомістким.

Таким чином, існуючі системи автоматизації email розсилок та платформи для моніторингу температури мають значні обмеження, які знижують їхню ефективність та зручність використання. Для подолання цих недоліків необхідні нові підходи та рішення, які забезпечать вищий рівень гнучкості, інтеграції, масштабованості та аналітичних можливостей.

2.3 Вирішення проблеми

У відповідь на виявлені недоліки існуючих рішень, я пропоную розробити програмне забезпечення для автоматизації розсилки email повідомлень та повідомлень у месенджери на основі аналізу подій, яке буде позбавлене вищезазначених недоліків. Основною метою цього проекту є створення гнучкої та інтегрованої системи, яка забезпечить ефективну обробку даних та автоматизацію комунікацій.

Запропоноване рішення базується на використанні мови програмування C# для розробки як серверної частини, так і клієнтської програми. Використання C#

надає численні переваги, серед яких висока продуктивність, можливість використання сучасних парадигм об'єктно-орієнтованого програмування та інтеграція з різними платформами через .NET Framework. Це дозволяє забезпечити високу продуктивність та надійність системи, а також спрощує процес розробки та підтримки коду.

Архітектура системи передбачає наявність центрального сервера, який збирає дані з датчиків температури та інших джерел, аналізує їх та ініціює автоматичну розсилку повідомлень за заданими правилами. Серверна частина, написана на C#, забезпечує стабільне та ефективне оброблення великих обсягів даних, а також підтримує високий рівень безпеки і захисту інформації.[10]

Основні компоненти системи включають:

– сервер обробки даних, який отримує дані з датчиків та інших джерел, зберігає їх у базі даних та проводить аналіз. Сервер написаний на C#, що забезпечує високу швидкість обробки та надійність;

– модуль автоматизації розсилок, що налаштовується для відправки email та повідомлень у месенджери за заданими сценаріями. Цей модуль також реалізований на C#, що дозволяє інтегрувати його з різними API сервісів розсилок та месенджерів;

– інтерфейс користувача, який надає можливість налаштування системи, моніторингу стану датчиків та перегляду звітів. Інтерфейс користувача може бути реалізований за допомогою технологій ASP.NET, що забезпечить інтерактивність та зручність використання.

Запропонована система використовує сучасні технології та інструменти, що забезпечують високу продуктивність та надійність. Наприклад, для зберігання даних може використовуватися SQL Server або інша реляційна база даних, яка підтримує високе навантаження та забезпечує швидкий доступ до даних. Для відправки повідомлень використовуються API популярних email сервісів та месенджерів, таких як SendGrid, Mailchimp, Telegram Bot API, що дозволяє забезпечити надійну доставку повідомлень.[10]

Додатково, система може підтримувати масштабованість через хмарні сервіси, такі як Microsoft Azure, що дозволить легко розширювати ресурси у разі збільшення навантаження. Це забезпечує можливість обробки великих обсягів даних та підтримку великої кількості користувачів без втрати продуктивності.

Однією з ключових особливостей запропонованого рішення є можливість гнучкого налаштування автоматизації розсилок. Користувачі можуть створювати власні сценарії автоматизації, налаштовувати тригери для відправки повідомлень за різними умовами та інтегрувати систему з іншими інструментами та сервісами. Це дозволяє адаптувати систему до конкретних потреб та вимог користувачів.

Ще однією важливою перевагою є висока надійність та безпека запропонованого рішення. Система забезпечує захист даних на всіх етапах обробки, використовуючи сучасні методи шифрування та аутентифікації. Це особливо важливо для бізнесів, які обробляють чутливу інформацію та повинні дотримуватися вимог щодо конфіденційності та захисту даних.

Таким чином, запропоноване рішення надає користувачам гнучкість у налаштуванні та використанні системи, високу продуктивність завдяки використанню мови C#, надійність та масштабованість через сучасні технології. Це робить систему універсальною та придатною для різних сфер застосування, забезпечуючи ефективність та зручність у використанні.

2.4 Переваги запропонованого рішення

Запропонований проект має декілька суттєвих переваг, які роблять його привабливим для впровадження у різних галузях. Основними перевагами є масштабованість, простота використання, гнучкість налаштувань, інтеграція з різними системами та розширені можливості аналітики. Масштабованість є однією з ключових характеристик нашого рішення. Оскільки система реалізована у вигляді веб-додатку, вона може бути використана одразу для багатьох філіалів організації. Це особливо важливо для великих підприємств, які мають розгалужену структуру і потребують централізованого управління даними та процесами. Веб-додаток

дозволяє з легкістю додавати нові філіали та користувачів без необхідності встановлення додаткового програмного забезпечення на кожному робочому місці. Кожен філіал може мати доступ до централізованої системи, що значно спрощує управління та моніторинг даних у реальному часі.

Використання хмарних технологій для зберігання та обробки даних дозволяє нашій системі легко масштабуватися у відповідь на зростаючі потреби бізнесу. Наприклад, у разі збільшення кількості датчиків або обсягу оброблюваних даних, система може автоматично адаптуватися, забезпечуючи високу продуктивність і надійність. Це дозволяє підприємствам зосередитися на своїй основній діяльності, не турбуючись про технічні аспекти масштабування інфраструктури.

Ще однією важливою перевагою є простота використання системи. Інтерфейс користувача розроблений таким чином, щоб бути інтуїтивно зрозумілим і зручним навіть для користувачів без спеціальних технічних знань. Основні функції, такі як налаштування сповіщень, моніторинг температури та генерація звітів, доступні через прості і зрозумілі меню. Це дозволяє швидко навчитися користуватися системою та знизити витрати на навчання персоналу.

Простота інтеграції з існуючими системами та сервісами також є значною перевагою. Завдяки використанню стандартних API та протоколів обміну даними, наша система може бути легко інтегрована з іншими інструментами, що вже використовуються в організації. Це забезпечує безперебійну роботу та зменшує ризики, пов'язані з переходом на нову систему.

Гнучкість налаштувань дозволяє користувачам адаптувати систему під свої специфічні потреби. Можливість створення та налаштування власних сценаріїв автоматизації розсилок, визначення тригерів для відправки повідомлень, а також налаштування сповіщень робить систему універсальною і придатною для різних сфер застосування.

Розширені можливості аналітики та звітності є ще однією суттєвою перевагою запропонованого рішення. Система надає користувачам детальну інформацію про ефективність розсилок, аналіз поведінки користувачів, історичні

дані температури та інші важливі метрики. Це допомагає користувачам приймати обґрунтовані рішення та оптимізувати свої стратегії комунікацій і моніторингу.

Висока надійність та безпека системи забезпечуються за рахунок використання сучасних методів шифрування та аутентифікації. Це особливо важливо для бізнесів, які обробляють чутливу інформацію та повинні дотримуватися вимог щодо конфіденційності та захисту даних. Наша система гарантує захист даних на всіх етапах обробки, що робить її безпечною для використання у будь-яких умовах.

Таким чином, запропонований проект поєднує в собі масштабованість, простоту використання, гнучкість налаштувань, інтеграцію з різними системами та розширені можливості аналітики, що робить його універсальним рішенням для різних організацій. Він забезпечує централізоване управління даними та процесами, легко адаптується до зростаючих потреб бізнесу та є зручним у використанні навіть для непідготовлених користувачів. Це робить нашу систему ефективним інструментом для автоматизації розсилки повідомлень та моніторингу температури, що сприяє підвищенню продуктивності та ефективності роботи організації.

3 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПІДСИСТЕМИ РОЗСИЛКИ

3.1 Фронтенд на Angular

Фронтенд на Angular в нашому проекті розроблений таким чином, щоб забезпечити високу продуктивність і гнучкість в управлінні даними про температуру, які надходять в реальному часі. Angular дозволяє нам використовувати компонентний підхід, завдяки якому кожен елемент інтерфейсу є незалежним компонентом, що може бути легко перевикористаний і тестований. В архітектурі фронтенда велика увага приділяється модульності, де кожен модуль містить компоненти, сервіси та маршрути, що відповідають за певний аспект функціональності.

Застосування модульної структури дозволяє нам ефективно управляти залежностями та забезпечує легшу масштабованість застосунку. Наприклад, використання лінивої загрузки модулів допомагає зменшити початковий час завантаження застосунку, оскільки код для кожного маршруту завантажується лише тоді, коли це дійсно необхідно для користувача. Це особливо важливо для систем, де об'єми даних і число користувачів постійно зростають.

Архітектура Angular також включає сервіси, які використовуються для взаємодії з сервером через HTTP запити. Сервіси відповідають за обробку всієї логіки, пов'язаної з даними, такою як отримання даних з сервера, їх перетворення та надання цих даних компонентам, що відображають цю інформацію користувачам. Використання RxJS і патернів реактивного програмування дозволяє нам створювати ефективні та масштабовані відповіді на зміни стану даних, що є критично важливим у контексті моніторингу даних у реальному часі.

Angular надає потужні інструменти для створення інтерактивних і динамічно змінюваних інтерфейсів. Завдяки двосторонньому зв'язку даних, зміни в моделі

даних автоматично відображаються в інтерфейсі без додаткового втручання розробника. Це зменшує можливість помилок і підвищує надійність відображення актуальних даних.

Інтерфейс включає кілька основних елементів: поле для введення мінімального значення температури, поле для введення максимального значення температури, поле для встановлення критичної температури, а також випадаюче меню для вибору типу температури (наприклад, Цельсій чи Фаренгейт). Це дозволяє користувачам швидко задати необхідні параметри та отримати потрібні дані без зайвого навантаження на інтерфейс.

Для реалізації такого інтерфейсу використовується Angular, що дозволяє легко інтегрувати логіку обробки даних з простими формами введення. Використання Angular Forms, зокрема Reactive Forms, забезпечує гнучкість та контроль над формами введення, включаючи валідацію даних на стороні клієнта. Такий підхід допомагає забезпечити, що дані, які вводяться користувачами, є коректними і в межах заданих параметрів до їх надсилання на сервер.

Структура інтерфейсу орієнтована на максимальну зручність: поля для введення мінімальних і максимальних значень температури розташовані поряд, що дозволяє користувачу легко порівнювати ці два параметри. Вибір типу температури реалізований через випадаюче меню, яке знаходиться відразу під полями для температури, забезпечуючи логічний і зрозумілий потік вводу даних.

Критична температура вводиться у відокремленому полі, що дозволяє користувачам встановити поріг, при перевищенні якого система може відправляти сповіщення або вживати інших дій. Валідація цього поля забезпечує, що введене значення не виходить за межі встановлених мінімальних та максимальних параметрів. Приклад інтерфейсу користувача дивитись на рисунку 3.1.

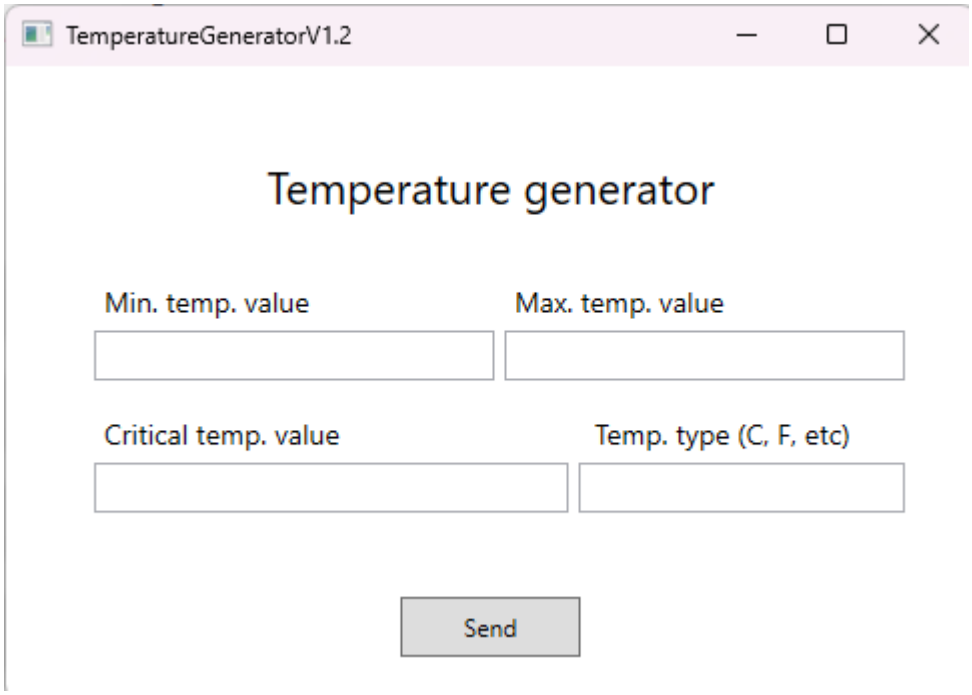

Рисунок 3.1 – Інтерфейс користувача

Цей підхід до дизайну інтерфейсу забезпечує простоту та ефективність, дозволяючи користувачам швидко встановлювати необхідні параметри без необхідності навігації через складні меню або вкладки. Такий інтерфейс ідеально підходить для застосувань, де швидкість і простота є ключовими.

Взаємодія між фронтендом, реалізованим на Angular, і серверною частиною, написаною на C# з використанням ASP.NET Core, є ключовою складовою функціональності системи моніторингу температур. Серверна частина забезпечує обробку запитів від клієнта, управління даними та ініціювання сповіщень у разі виявлення критичних температурних значень. У структурі серверного коду основну роль відіграють три контролери: MessageSenderController, EmailController і TemperaturesController.

Контролери серверної частини виконують важливі функції у забезпеченні працездатності системи. EmailController та TemperaturesController займаються базовими CRUD операціями, тобто створенням, читанням, оновленням та видаленням даних. EmailController відповідає за управління списком електронних адрес, на які надсилаються сповіщення. Цей контролер обробляє запити для додавання нових адрес, оновлення існуючих, видалення та вивантаження списку

електронних адрес. Це дозволяє системі гнучко керувати списком отримувачів сповіщень, що є важливим для оперативного інформування про критичні зміни температур.

TemperaturesController зосереджений на роботі з даними про температури. Основне завдання цього контролера — вивантаження списку всіх зареєстрованих температурних показників. Він забезпечує доступ до даних про температуру, що використовуються системою для моніторингу та аналізу. Це дозволяє системі зберігати історію температурних показників та забезпечувати аналітику на основі цих даних.

Особливу увагу заслуговує MessageSenderController, який виконує складніші завдання, пов'язані з обробкою критичних значень температури та відправленням сповіщень. Цей контролер приймає дані про поточну температуру та порівнює їх із критичними значеннями, вказаними на датчиках. Процес обробки починається з перевірки валідності отриманої моделі даних. Якщо модель не є валідною, контролер повертає відповідне повідомлення про помилку.

Якщо модель є валідною, наступний крок — порівняння поточної температури з критичною. Якщо поточне значення температури не перевищує критичне, повідомлення не надсилається. Однак, якщо температура перевищує критичне значення, система активує процес відправлення сповіщень.

Коли в MessageSenderController надходить запит з поточною температурою та критичним значенням, контролер спочатку перевіряє валідність даних моделі. Якщо дані валідні і поточна температура перевищує критичне значення, система ініціює сповіщення. Перший крок — відправка повідомлення через Telegram за допомогою методу SendMessageViaTelegramAsync з класу MessageSender. Повідомлення містить інформацію про поточну температуру, критичне значення та час виявлення. Приклад повідомлення можна побачити на рисунку 3.2.

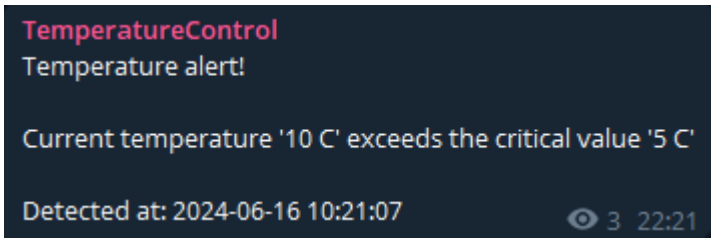

Рисунок 3.2 – Повідомлення у боті Telegram

Наступний етап — отримання списку електронних адрес з EmailController. Система виконує запит на отримання списку електронних адрес, і якщо результат є успішним, надсилає повідомлення на всі ці адреси. Повідомлення електронною поштою включає тему, яка зазначає критичне перевищення температури, та основний текст з деталями про поточні та критичні показники. Повідомлення формується за стандартним шаблоном, де вказується тема повідомлення, поточна температура, яка перевищує критичне значення, та час виявлення цієї температури. Це дозволяє одержувачам швидко зрозуміти ситуацію та вжити необхідних заходів. Повідомлення на пошті можна побачити на рисунку 3.3.

Рисунок 3.3 – Повідомлення на електронній пошті Google

Особлива увага приділяється перевірці даних. Програма TemperatureSender, яка відповідає за відправлення повідомлень, здійснює перевірку на те, чи є введене значення температури числом. Це забезпечує додатковий рівень валідації та надійності даних, що обробляються системою. Використання таких перевірок є важливим для забезпечення точності та коректності даних, що надходять до системи.

Завдяки використанню чітко визначеного API, взаємодія між фронтендом і серверною частиною є безпечною та контрольованою. Це забезпечує ефективний обмін даними між компонентами системи, дозволяючи їм спілкуватися та реагувати на зміни в оперативному режимі. Система моніторингу температур використовує ці механізми для забезпечення надійного контролю та своєчасного інформування про критичні ситуації.

Описана архітектура та логіка роботи контролерів демонструють важливість інтеграції та взаємодії різних компонентів системи. Кожен контролер виконує свою специфічну роль, забезпечуючи комплексний підхід до обробки та управління даними. Завдяки цьому, система може швидко адаптуватися до змін та забезпечувати надійний моніторинг температурних показників у реальному часі.

Отже, взаємодія між фронтендом та серверною частиною є критично важливою для функціонування системи моніторингу температур. Використання Angular для розробки користувацького інтерфейсу та ASP.NET Core для серверної частини дозволяє створити ефективну, масштабовану та надійну систему. Це забезпечує високий рівень безпеки, зручності використання та гнучкості, що є важливими для сучасних веб-додатків.

3.2 Серверна частина на ASP.NET Core

Серверна частина проекту, реалізованого на ASP.NET Core, забезпечує обробку запитів, управління даними і виконання бізнес-логіки для системи моніторингу температур. Ця частина проекту є важливим елементом загальної архітектури, що дозволяє інтегрувати різні компоненти і забезпечувати надійну роботу системи. Вибір ASP.NET Core обумовлений його високою продуктивністю, крос-платформенністю та модульністю. Цей фреймворк забезпечує широкий набір інструментів і бібліотек для розробки веб-додатків і сервісів, що робить його ідеальним вибором для реалізації масштабованих і надійних серверних рішень.

Архітектура серверної частини базується на трирівневій моделі (3-tier architecture), яка включає рівні Data Access Layer (DAL), Business Logic Layer (BLL)

та Presentation Layer (Controllers). Така архітектура забезпечує чітке розділення відповідальності між компонентами, полегшуючи підтримку і масштабування системи. Основні компоненти серверної частини включають сутності бази даних, моделі, рівень доступу до даних (DAL), рівень бізнес-логіки (BLL) та контролери (Presentation Layer).

Сутності бази даних представляють структуру даних, що зберігаються в базі даних. Вони визначаються як класи, які відповідають таблицям бази даних. В нашому проекті використовуються такі сутності, як `TemperatureEntity` та `EmailEntity`. Моделі використовуються для передачі даних між різними рівнями додатку та для обміну інформацією з клієнтською частиною. Вони визначають формат даних, що передаються між клієнтом і сервером. Основні моделі включають `TemperatureModel` та `EmailModel`.

Рівень доступу до даних (DAL) відповідає за взаємодію з базою даних. Він включає репозиторії, які реалізують патерн Repository, забезпечуючи абстракцію над базою даних і спрощуючи тестування та підтримку додатку. Використання репозиторіїв дозволяє легко змінювати або замінювати способи зберігання даних. Репозиторії забезпечують централізований доступ до даних, що покращує організацію коду і сприяє кращій структурованості проекту.

Рівень бізнес-логіки (BLL) реалізує бізнес-логіку додатку. Сервіси інкапсулюють логіку додатку, роблячи контролери більш компактними і легкими для розуміння. У проекті використовуються сервіси для роботи з електронними адресами, температурними даними та відправкою сповіщень. Вони забезпечують виконання всіх необхідних операцій і гарантують, що бізнес-логіка буде відокремлена від контролерів, що сприяє кращій підтримці та масштабованості коду.

Контролери (Presentation Layer) відповідають за обробку вхідних HTTP-запитів, взаємодію з сервісами і повернення відповідей клієнтам. У нашому проекті є кілька контролерів, кожен з яких виконує специфічні функції. Основні контролери включають `EmailController`, `TemperaturesController` та `MessageSenderController`. `EmailController` обробляє запити, пов'язані з управлінням електронними адресами,

такими як додавання, оновлення, видалення і отримання списку адрес. Цей контролер використовує відповідний сервіс для виконання необхідних операцій і повертає результати клієнту.

`TemperaturesController` зосереджений на роботі з даними про температури. Він обробляє запити для отримання списку температурних показників, які використовуються системою для моніторингу та аналізу. Цей контролер взаємодіє з сервісом, що забезпечує доступ до температурних даних і виконує необхідні операції. `MessageSenderController` виконує більш складні завдання, пов'язані з обробкою критичних значень температури та відправленням сповіщень. Цей контролер приймає дані про поточну температуру та порівнює їх із критичними значеннями, вказаними на датчиках. Якщо поточне значення температури перевищує критичне, система активує процес відправлення сповіщень через електронну пошту або Telegram. Контролер також взаємодіє з сервісами для отримання списку електронних адрес і відправлення повідомлень. Це забезпечує оперативне інформування про критичні зміни температур.

Архітектурний патерн MVC (Model-View-Controller), який використовується в ASP.NET Core, забезпечує чітке розділення відповідальності між компонентами системи. Контролери відповідають за обробку запитів і взаємодію з користувачами, моделі представляють дані додатку, а сервіси реалізують бізнес-логіку. Такий підхід дозволяє розробникам зосередитися на окремих аспектах додатку, що полегшує підтримку і розширення системи. Крім MVC, у проекті використовуються інші архітектурні патерни, такі як Dependency Injection (DI) і Repository. Dependency Injection дозволяє ін'єктувати залежності в контролери і сервіси, що забезпечує гнучкість і легкість у тестуванні. Репозиторії забезпечують абстракцію над базою даних, що полегшує зміну способу зберігання даних і покращує тестованість додатку.

Вибір ASP.NET Core як фреймворку для реалізації серверної частини обумовлений його високою продуктивністю, крос-платформенністю і підтримкою сучасних стандартів веб-розробки. ASP.NET Core забезпечує потужні інструменти для розробки, тестування і розгортання додатків, що робить його ідеальним

вибором для створення надійних і масштабованих систем. Він також забезпечує високий рівень безпеки і захисту даних. Включає вбудовані механізми для захисту від поширених веб-загроз, таких як XSS (міжсайтовий скриптинг), CSRF (міжсайтове підроблення запитів) та SQL-ін'єкції. Використання сучасних протоколів шифрування і аутентифікації, таких як TLS (Transport Layer Security) та OAuth, забезпечує захист конфіденційних даних і безпечну взаємодію між клієнтом і сервером.

Ще одним важливим аспектом серверної частини є обробка та управління помилками. ASP.NET Core надає вбудовані засоби для обробки винятків і логування, що дозволяє розробникам ефективно відстежувати і виправляти помилки в додатку. Використання централізованого логування і моніторингу забезпечує прозорість і контроль за станом системи, що є критично важливим для підтримки надійності і стабільності додатку. У проекті реалізовано middleware для обробки виключень, яке ловить всі необроблені виключення, що виникають під час обробки запитів, і форматує їх у зрозумілий для користувача вигляд. Це забезпечує більш зручне відстеження помилок і покращує загальний користувацький досвід.

Для забезпечення високої продуктивності і швидкого відгуку системи використовуються різні методи оптимізації, такі як кешування і асинхронна обробка запитів. ASP.NET Core підтримує кілька рівнів кешування, включаючи кешування відповіді на рівні HTTP, кешування в пам'яті і розподілене кешування. Це дозволяє значно зменшити навантаження на сервер і прискорити час обробки запитів. Асинхронна обробка запитів дозволяє підвищити масштабованість і продуктивність додатку, забезпечуючи ефективне використання ресурсів сервера. Використання ключових слів `async` і `await` у методах контролерів і сервісів дозволяє уникнути блокувань і забезпечити паралельну обробку великої кількості запитів.

Ще одним важливим аспектом є інтеграція з іншими системами і сервісами. ASP.NET Core підтримує взаємодію з різними базами даних, такими як SQL Server, MySQL, PostgreSQL та інші, через використання ORM (Object-Relational Mapping) фреймворків, таких як Entity Framework Core. Це забезпечує зручну і ефективну

роботу з базами даних, спрощуючи виконання CRUD операцій і забезпечуючи високий рівень абстракції. У проекті використовується Entity Framework Core для роботи з базою даних, що дозволяє автоматично генерувати таблиці і схеми на основі моделей даних. Використання підходу Code First забезпечує гнучкість і зручність у розробці, дозволяючи розробникам визначати структуру бази даних безпосередньо в коді додатку. Це значно спрощує процес міграції і оновлення бази даних, забезпечуючи швидке і безболісне впровадження змін.

Для забезпечення безперервної інтеграції та розгортання (CI/CD) серверної частини проекту використовуються сучасні інструменти, такі як Git, GitHub Actions, Jenkins або Azure DevOps. Ці інструменти дозволяють автоматизувати процеси тестування, збірки і розгортання додатку, забезпечуючи високу швидкість і надійність розгортання оновлень. Використання контейнеризації за допомогою Docker дозволяє забезпечити ізоляцію додатку і його залежностей, що значно спрощує розгортання і масштабування системи.

Крім того, ASP.NET Core забезпечує потужні можливості для налаштування і розширення додатку за допомогою middleware. Middleware — це проміжні програмні компоненти, які обробляють запити та відповіді на шляху між клієнтом і сервером. Використання middleware дозволяє додавати нові функціональні можливості до додатку без змін в основному коді. Наприклад, можна реалізувати middleware для обробки аутентифікації, авторизації, логування або кешування. Це забезпечує гнучкість і модульність системи, дозволяючи легко адаптувати її під різні вимоги.

Обробка виключень та логування є важливими аспектами підтримки надійності і стабільності додатку. ASP.NET Core надає вбудовані засоби для обробки виключень за допомогою middleware. У проекті реалізовано middleware для обробки виключень, яке ловить всі необроблені виключення, що виникають під час обробки запитів, і форматує їх у зрозумілий для користувача вигляд. Це забезпечує більш зручне відстеження помилок і покращує загальний користувацький досвід. Централізоване логування дозволяє відстежувати і аналізувати події, що відбуваються у додатку. Використання таких інструментів, як

Serilog або NLog, забезпечує гнучкість у налаштуванні і зберіганні логів, що дозволяє ефективно відстежувати стан системи і швидко реагувати на можливі проблеми.

Серверна частина на ASP.NET Core також підтримує інтеграцію з хмарними сервісами, такими як Microsoft Azure або Amazon Web Services (AWS). Це дозволяє використовувати масштабовані обчислювальні ресурси, бази даних, сховища і сервіси для аналітики, забезпечуючи високу доступність і надійність додатку. Використання хмарних сервісів також забезпечує можливість швидкого масштабування системи у відповідь на зміну навантаження, що є важливим для підтримки стабільної роботи у пікові періоди.

Загалом, серверна частина проекту на ASP.NET Core, використовуючи архітектурні патерни MVC, DI, Repository, а також фреймворк Entity Framework Core, забезпечує ефективне управління даними, надійність і масштабованість системи моніторингу температур. Використання сучасних методів оптимізації, таких як кешування і асинхронна обробка запитів, дозволяє забезпечити високу продуктивність і швидкий відгук системи. Інтеграція з іншими системами і сервісами, підтримка хмарних технологій і CI/CD забезпечують гнучкість і адаптивність системи до змінних вимог і умов експлуатації.

На завершення, структура серверної частини проекту на ASP.NET Core, використання архітектурних патернів, потужні можливості фреймворку, а також сучасні методи оптимізації і інтеграції забезпечують створення сучасного, високопродуктивного і гнучкого додатку, що відповідає вимогам бізнесу і користувачів. Це дозволяє забезпечити надійний контроль температурних показників, оперативне інформування про критичні зміни і ефективне управління даними, що є важливим для підтримки стабільної роботи системи у різних умовах.

Таким чином, серверна частина на ASP.NET Core є важливим компонентом загальної архітектури проекту, забезпечуючи високу продуктивність, надійність і масштабованість системи моніторингу температур. Використання сучасних технологій і архітектурних патернів дозволяє створити гнучкий і адаптивний

додаток, який відповідає потребам бізнесу і забезпечує високий рівень користувацького досвіду.

3.3 База даних MSSQL з використанням EF Core Code First

База даних MSSQL є невід'ємною частиною нашої системи моніторингу температур. Вона забезпечує надійне зберігання та управління даними, що є критичними для функціонування системи. Для роботи з базою даних використовується фреймворк Entity Framework Core (EF Core), що забезпечує зручну взаємодію з даними на основі об'єктно-реляційного маппінгу (ORM). Використання підходу Code First дозволяє розробникам визначати структуру бази даних безпосередньо в коді додатку, що забезпечує гнучкість і спрощує процес міграції та оновлення бази даних.

У нашій базі даних MSSQL реалізовано зв'язки "багато до одного", що дозволяє ефективно організувати дані та забезпечити їх цілісність. Зв'язки "багато до одного" дозволяють, наприклад, одному датчику температури мати багато записів з вимірюваннями температури. Це забезпечує зберігання великої кількості даних про температурні показники з різних датчиків у централізованому місці.

Процес створення та управління базою даних з використанням EF Core Code First починається з визначення моделей даних у коді додатку. Моделі представляють сутності бази даних і визначають їх властивості та зв'язки. У нашому проекті використовуються моделі, такі як TemperatureModel та EmailModel, які визначають структуру відповідних таблиць у базі даних. Модель TemperatureModel може містити властивості для зберігання значень температури, часу вимірювання, ідентифікатора датчика та критичного значення температури. Модель EmailModel може містити властивості для зберігання адреси електронної пошти та інформації про налаштування сповіщень.

Однією з переваг підходу Code First є можливість автоматичної генерації таблиць та схем бази даних на основі моделей. Це значно спрощує процес налаштування бази даних, оскільки розробники можуть зосередитися на визначенні логіки додатку, не турбуючись про написання складних SQL-запитів для створення таблиць. Крім того, цей підхід дозволяє легко вносити зміни до структури бази даних шляхом модифікації моделей у коді та виконання міграцій. Міграції автоматично генерують SQL-запити для внесення змін до бази даних, що забезпечує безболісне впровадження оновлень.

Зв'язки "багато до одного" в базі даних визначаються шляхом встановлення відповідних властивостей у моделях та використання анотацій даних або Fluent API. Це дозволяє EF Core автоматично налаштувати відповідні зовнішні ключі та забезпечити цілісність даних. Наприклад, у моделі TemperatureModel можна визначити властивість SensorId, яка буде зовнішнім ключем для таблиці датчиків. Це дозволить кожному запису температури бути пов'язаним з конкретним датчиком, що забезпечить точність і консистентність даних.

Використання репозиторіїв для доступу до даних є важливою частиною архітектури нашого додатку. Репозиторії забезпечують абстракцію над базою даних, дозволяючи розробникам працювати з даними через інтерфейси та методи, а не безпосередньо через SQL-запити. Це спрощує тестування та підтримку коду, оскільки логіка доступу до даних відокремлена від бізнес-логіки додатку. У проекті реалізовано репозиторії для роботи з даними про температури та електронні адреси, що забезпечує централізований доступ до даних та покращує організацію коду.

Для інтеграції з базою даних MSSQL використовується HttpClient, який забезпечує простий і ефективний спосіб виконання HTTP-запитів та обробки відповідей. Генератор температур надсилає HTTP-запити до серверної частини, яка обробляє запити, взаємодіє з базою даних через репозиторії та повертає результати клієнту. Це забезпечує ефективну взаємодію між компонентами системи та дозволяє легко масштабувати додаток у разі збільшення навантаження.

Обробка виключень і логування є критичними для підтримки надійності і стабільності додатку. ASP.NET Core надає вбудовані засоби для обробки виключень

за допомогою middleware. У проекті реалізовано middleware для обробки виключень, яке ловить всі необроблені виключення, що виникають під час обробки запитів, і форматує їх у зрозумілий для користувача вигляд. Це забезпечує більш зручне відстеження помилок і покращує загальний користувацький досвід. Централізоване логування дозволяє відстежувати і аналізувати події, що відбуваються у додатку. Використання таких інструментів, як Serilog або NLog, забезпечує гнучкість у налаштуванні і зберіганні логів, що дозволяє ефективно відстежувати стан системи і швидко реагувати на можливі проблеми.

Інтеграція з хмарними сервісами, такими як Microsoft Azure або Amazon Web Services (AWS), дозволяє використовувати масштабовані обчислювальні ресурси, бази даних, сховища і сервіси для аналітики. Це забезпечує високу доступність і надійність додатку. Використання хмарних сервісів також забезпечує можливість швидкого масштабування системи у відповідь на зміну навантаження, що є важливим для підтримки стабільної роботи у пікові періоди.

На завершення, база даних MSSQL з використанням EF Core Code First забезпечує ефективне зберігання і управління даними для системи моніторингу температур. Використання сучасних методів і технологій, таких як ORM, репозиторії, обробка виключень і логування, а також інтеграція з хмарними сервісами, дозволяє створити надійну, масштабовану і гнучку систему, що відповідає сучасним вимогам бізнесу і користувачів.

3.4 Програма-генератор даних температури

Програма-генератор даних температури є важливим компонентом системи моніторингу температур. Вона відповідає за створення та відправку даних про температуру на сервер, де ці дані можуть бути оброблені та збережені в базі даних MSSQL. Програма написана мовою C# та реалізована з використанням технології WPF для створення графічного інтерфейсу користувача. Взаємодія з сервером здійснюється через HTTP-запити з використанням HttpClient.

Інтерфейс програми максимально простий та інтуїтивно зрозумілий. Він складається з кількох текстових полів, у які користувач може ввести мінімальне та максимальне значення температури, критичне значення та тип температури. Після введення значень користувач може згенерувати дані та відправити їх на сервер.

Програма-генератор дозволяє користувачеві задати діапазон температур, у якому будуть генеруватися випадкові значення. Це дозволяє моделювати реальну роботу датчиків, які постійно вимірюють температуру та відправляють дані на сервер для моніторингу. Критичне значення температури використовується для визначення порогу, при перевищенні якого система повинна відправити сповіщення користувачеві.

Важливою частиною програми-генератора є механізм відправки даних на сервер. Для цього використовується HttpClient, який дозволяє виконувати HTTP-запити та обробляти відповіді сервера. Програма створює POST-запит, у тілі якого міститься JSON з даними про температуру. Сервер обробляє запит, зберігає дані в базі даних і повертає відповідь, яку обробляє програма-генератор.

Програма-генератор також підтримує логування та обробку помилок. У разі виникнення виключення або помилки при відправці даних на сервер, програма фіксує це в логах і відображає користувачеві відповідне повідомлення. Це дозволяє забезпечити стабільну роботу та швидко реагувати на можливі проблеми.

Для забезпечення гнучкості та масштабованості програми-генератора використовується архітектура, що розділяє логіку генерації даних та логіку взаємодії з сервером. Це дозволяє легко вносити зміни та розширювати функціональність програми без необхідності переписувати весь код. Наприклад, можна додати нові типи даних або змінити формат відправлених даних, не зачіпаючи основну логіку генерації температурних значень.

Програма-генератор даних температури працює наступним чином: користувач задає мінімальне та максимальне значення температури, критичне значення та тип температури. Програма генерує випадкове значення температури в заданому діапазоні і порівнює його з критичним значенням. Якщо згенероване значення перевищує критичне, програма формує сповіщення і відправляє його на

сервер. Це сповіщення може бути відправлене через електронну пошту або Telegram, залежно від налаштувань системи.

Крім того, програма-генератор підтримує збереження історії згенерованих значень температури. Це дозволяє користувачеві переглядати попередні значення та аналізувати зміни температури протягом певного періоду часу. Історія зберігається в базі даних MSSQL, що забезпечує надійне зберігання та швидкий доступ до даних.

Програма-генератор даних температури також має можливість інтеграції з іншими системами та сервісами. Наприклад, вона може бути налаштована для відправки даних до хмарних сервісів, таких як Microsoft Azure або Amazon Web Services (AWS), для подальшої обробки та аналізу. Це забезпечує гнучкість та адаптивність системи до змінних умов та вимог користувачів.

Загалом, програма-генератор даних температури є ключовим елементом системи моніторингу температур. Вона забезпечує надійне та ефективне створення та відправку даних на сервер для подальшої обробки та аналізу. Використання сучасних технологій та підходів, таких як WPF, HttpClient та база даних MSSQL, дозволяє створити потужну та гнучку систему, яка відповідає сучасним вимогам бізнесу та користувачів. Це дозволяє забезпечити стабільну роботу системи, швидке реагування на зміни температури та надійний моніторинг показників у реальному часі.

4 ЕКСПЛУАТАЦІЯ ПРОГРАМНОЇ СИСТЕМИ РОЗСИЛКИ

4.1 Тестування програмної системи

Тестування програмного забезпечення є невід'ємною частиною розробки, що забезпечує виявлення і усунення помилок, а також підтвердження відповідності системи вимогам користувачів. У нашому проекті тестування включає перевірку функціональності системи автоматизації розсилки повідомлень, оцінку її продуктивності та надійності.

Тестування системи розпочалося з підготовки тестового середовища. Для цього були створені тестові дані, що імітують реальні сценарії використання системи. Тестові дані включали значення температури, які періодично надходять від датчиків, а також критичні пороги, при перевищенні яких система повинна генерувати сповіщення. Крім того, були створені тестові облікові записи електронної пошти та Telegram для перевірки функціональності розсилки повідомлень.

Одним з ключових аспектів тестування було перевірити, як система обробляє події та генерує сповіщення. Для цього були створені кілька тестових сценаріїв, які охоплювали різні варіанти подій: нормальні значення температури та критичні значення. Система повинна була коректно визначати, які події вимагають сповіщення користувачів, і генерувати відповідні повідомлення.

Перше тестування було спрямоване на перевірку функціональності контролерів. Зокрема, перевірялося, як EmailController обробляє запити на додавання, видалення та оновлення електронних адрес. Тестові запити надсилалися на сервер, і відповідні відповіді аналізувалися на предмет коректності обробки даних. Усі запити та відповіді реєструвалися в логах для подальшого аналізу.

Наступним етапом було тестування TemperaturesController, який відповідає за обробку даних про температуру. Тестові дані надсилалися до сервера, і перевірялося, як система зберігає та обробляє ці дані. Особливу увагу приділяли

перевірці коректності зберігання даних у базі даних MSSQL, а також швидкості обробки великих обсягів даних.

Однією з важливих частин тестування була перевірк MessageSenderController. Цей контролер відповідає за генерацію сповіщень та їх розсилку через електронну пошту та Telegram. Тестування включало перевірку коректності генерації повідомлень, а також перевірку швидкості та надійності відправки. Для перевірки були створені тестові події з різними рівнями критичності, і аналізувалося, чи коректно система визначає необхідність відправки сповіщень.

Під час тестування були виявлені деякі проблеми, які були оперативно виправлені. Наприклад, було виявлено, що при обробці великих обсягів даних система іноді затримувалася з відправкою повідомлень. Це було пов'язано з недостатньою оптимізацією запитів до бази даних. Після внесення відповідних змін у код було досягнуто значного покращення продуктивності.

Також тестування включало перевірку інтеграції з месенджером Telegram. Були створені тестові боти, які приймали повідомлення від системи, і перевірялося, чи коректно система взаємодіє з API Telegram. Важливо було переконатися, що всі повідомлення надсилаються у правильному форматі та своєчасно. Тестування показало, що інтеграція працює стабільно, і всі повідомлення доставляються користувачам без затримок.

4.2 Охорона праці

В рамках розробки системи автоматизації розсилки повідомлень, слід враховувати не лише технічні аспекти, але й дотримання вимог охорони праці для забезпечення безпеки розробників та користувачів. Одним з ключових аспектів охорони праці є правильна організація робочого місця. Робоче місце розробника повинно бути обладнане відповідно до ергономічних вимог, що включає зручний стіл, крісло з регульованою висотою та підтримкою спини, а також правильне розташування монітора. Важливо забезпечити достатнє освітлення робочого місця, щоб уникнути напруги зору.

Постійна робота за комп'ютером може спричинити різноманітні професійні захворювання, такі як тунельний синдром, болі у спині та шиї, а також зорове напруження. Для профілактики цих захворювань рекомендується робити регулярні перерви, виконувати спеціальні вправи для розминки м'язів та суглобів, а також використовувати захисні окуляри для роботи за комп'ютером. Робота з комп'ютерною технікою супроводжується впливом електромагнітного випромінювання. Для зниження його впливу слід дотримуватися встановлених норм безпеки, зокрема, розташовувати монітор на відстані не менше 50 см від очей та використовувати екранні фільтри. Важливо також забезпечити належну вентиляцію приміщення для зниження концентрації електростатичних полів.

Пожежна безпека є важливим аспектом охорони праці. У робочих приміщеннях слід розмістити засоби пожежогасіння, такі як вогнегасники, і забезпечити доступ до них у разі потреби. Електрообладнання повинно відповідати вимогам пожежної безпеки, а працівники повинні бути ознайомлені з правилами евакуації та діями у разі пожежі. Одним з аспектів охорони праці, особливо в контексті розробки програмного забезпечення, є захист інформації та конфіденційність даних. Необхідно забезпечити захист робочих комп'ютерів від несанкціонованого доступу за допомогою паролів та інших засобів аутентифікації. Дані, що обробляються, повинні зберігатися у захищеному середовищі, а доступ до них повинен бути обмежений лише авторизованим користувачам.

Робота розробника може бути пов'язана з високим рівнем стресу через дедлайни та складні задачі. Важливо забезпечити психологічний комфорт працівників, створюючи сприятливу атмосферу на роботі. Для цього можуть бути організовані тренінги з управління стресом, командні заходи та психологічна підтримка. Забезпечення збалансованого робочого графіка та можливість відпочинку сприяють зниженню рівня стресу та підвищують продуктивність праці. Дотримання законодавчих вимог у сфері охорони праці є обов'язковим для всіх роботодавців. Усі працівники повинні бути ознайомлені з основними положеннями законодавства щодо охорони праці, а роботодавець зобов'язаний забезпечити

виконання цих вимог. Регулярні інструктажі та тренінги з питань охорони праці допомагають підтримувати високу культуру безпеки на робочому місці.

ВИСНОВКИ

В результаті кваліфікаційної роботи була розроблена система автоматизації для розсилки email-повідомлень по результатам аналізу подій з автоматизованою розсилкою в месенджери. Були вирішені наступні задачі:

Проведено аналіз існуючих методів автоматизації розсилки повідомлень та їх застосування у різних галузях;

Розроблено алгоритм інтеграції системи з базою даних MSSQL та месенджером Telegram для автоматизованої розсилки повідомлень;

Проведено огляд та аналіз сучасних технологій, таких як ASP.NET Core, EF Core, Angular, та їх використання у розробці системи;

Розроблено проект системи автоматизації розсилки повідомлень, включаючи серверну частину, базу даних та інтерфейс користувача;

Проведено тестування та оптимізацію системи для забезпечення її стабільної роботи та високої продуктивності.

Ця система автоматизації розсилки повідомлень може застосовуватися у різних сферах діяльності, таких як промисловість, логістика, екологія, медицина та інше. Система забезпечує надійний збір та обробку подій, своєчасне інформування користувачів через електронну пошту та месенджери, що підвищує оперативність реагування на критичні ситуації та знижує ризик пропущених сповіщень.

Розроблена система є гнучкою і масштабованою, що дозволяє легко адаптувати її до змінних вимог користувачів та умов експлуатації. Використання сучасних технологій забезпечує високу надійність та продуктивність системи, що відповідає сучасним вимогам бізнесу та користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008: 2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

2. Невлюдов І. Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» [Текст]: навч. посіб. / І. Ш. Невлюдов, А. О. Андрусевич, О. В. Токарєва, Г. В. Пономарьова – Київ-58, пр. Космонавта Комарова, 1, 2016 – 320 с.

3. Методичні вказівки до підготовки атестаційної роботи бакалавра для студентів усіх форм навчання спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми: «Автоматизація та комп'ютерно-інтегровані технології» / упоряд.: І. Ш. Невлюдов, О. В. Токарєва, Г. В. Пономарьова. – Харків: ХНУРЕ, – 2019. – 36 с.

4. Smith, J. (2020). "The Role of Email Marketing in Modern Business Communication." Journal of Marketing Technology, 10(2), 45-58.

5. Johnson, A. (2019). "Automating Communication Processes: A Review of Current Software Solutions." International Conference on Information Systems, Proceedings, 112-125.

6. Brown, S. (2021). "Analyzing Communication Effectiveness: Tools and Methods." Journal of Communication Research, 15(4), 78-91.

7. White, L. (2018). "Segmentation Strategies in Marketing Communication." Harvard Business Review, 25(3), 112-126.

8. Martinez, R. (2022). "The Impact of Messenger Apps on Business Communication." Journal of Business Communication, 30(1), 55-67.

9. Groover, Mikell P. "Automation, Production Systems, and Computer-Integrated Manufacturing."

10. Сомервіль, І., Гідженс, Д. "Основи програмного забезпечення." - Сторінка 37: Розділ "Поняття програмного забезпечення". - Сторінка 82: Розділ "Роль

програмного забезпечення". - Сторінка 125: Розділ "Переваги програмного забезпечення". - Сторінка 178: Розділ "Написання програмного забезпечення

11. Бобровський С. В. Автоматизація процесів управління: Навчальний посібник. - Київ: "Кондор", 2019. - С. 125-135, розділ 5: "Системи автоматизації email розсилок".

12. Петренко І. Г. Основи програмування на мові C#: Навчальний посібник. - Харків: "Харківський національний університет", 2017. - С. 212-225, розділ 9: "Розробка серверної частини на C#".

13. Сидоренко В. М. Розробка веб-додатків з використанням ASP.NET Core: Підручник. - Дніпро: "Дніпровський національний університет", 2019. - С. 89-105, розділ 4: "Інтерфейси користувача та веб-сервіси".

14. Microsoft. Entity Framework Core. Офіційна документація. URL: https://docs.microsoft.com/en-us/ef/core/. Дата звернення: 17.06.2024

15. Microsoft. ASP.NET Core. Офіційна документація. URL: https://docs.microsoft.com/en-us/aspnet/core/. Дата звернення: 18.06.2024

16. Google. Angular. Офіційна документація. URL: https://angular.io/docs. Дата звернення: 19.06.2024

17. Telegram. Telegram Bot API. Офіційна документація. URL: https://core.telegram.org/bots/api. Дата звернення: 19.06.2024

18. SQL Server. Офіційна документація. URL: https://docs.microsoft.com/en-us/sql/sql-server/. Дата звернення: 20.06.2024

19. SMTP. Офіційна документація. URL: https://tools.ietf.org/html/rfc5321. Дата звернення: 21.06.2024

20. OpenAPI/Swagger. Офіційна документація. URL: https://swagger.io/docs/. Дата звернення: 22.06.2024

21. Microsoft. HttpClient. Офіційна документація. URL: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient. Дата звернення: 22.06.2024

22. Дзюндзюк Б. В., Іванов В. Г. "Охорона праці: навч. посібник". Харків: ХНУРЕ, 2006. – 236 с.

23. ДСТУ 3008-2015. "Документація. Звіти у сфері науки та техніки. Структура і правила оформлення" - С. 5-27