МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

МАНОЙЛО НАЗАРІЙ ЕДУАРДОВИЧ

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЗОВАНОЇ ВІДПРАВКИ ЛИСТІВ ЗА ШАБЛОНОМ НА ПОШТОВІ СЕРВІСИ

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
БУЛАТЕЦЬКИЙ ВІТАЛІЙ ВІКТОРОВИЧ,
кандидат фізико-математичних наук, доцент кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри
комп'ютерних наукта
кібербезпеки
від_____ 2024 р.
Завідувач кафедри

(_____) Гришанович Т. О.

ЛУЦЬК – 2024

ЗМІСТ

ВСТУП 4
РОЗДІЛ 1 ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ ТА ПОШТОВИХ СЕРВІСІВ 6
1.1 Огляд поштових сервісів 6
1.1.1 Види поштових сервісів 6
1.1.2 Особливості використання різних поштових сервісів 9
1.2 Типові технології та середовища розробки 12
1.2.1 Інтегроване середовище розробки (IDE) 12
1.2.3 Windows Forms (WinForms) 18
1.2.4 Використання бази даних 19
1.2.5 Інтеграція з поштовими сервісами 25
1.2.6 Дизайн і архітектура програмного забезпечення 26
1.2.7 Функціонал Windows Forms 27
1.2 Архітектурні особливості поштових сервісів 29
1.4 Дослідження аналогів застосунку 31
РОЗДІЛ 2 РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗОВАНОЇ ВІДПРАВКИ ЛИСТІВ 36
2.1 Постановка задачі, призначення та вимоги для програми відправки листів 36
2.2 Загальний опис проекту 37
2.3 Вибір моделі розробки програмного засобу 39
2.4 Обґрунтування вибору інструментальних засобів розробки 40
2.4.1 Вибір Visual Studio як основного інтегрованого середовища розробки (IDE): 40
2.4.2 Використання мови програмування C#. 40
2.4.3 Використання Windows Forms для розробки інтерфейсу користувача 41
2.4.4 Використання бази даних SQL Lite. 41
2.5 Особливості програмної реалізації 42
2.5.1 Конфігурація поштових сервісів 42

2.5.2 Вибір поштового сервісу 42
2.5.3 Додавання вибраної електронної адреси 43
2.5.4 Відправка електронних листів 44
2.5.5 Взаємодія з базою даних 46
2.5.6 Оновлення елементів інтерфейсу 46
2.5.7 Таймери для повідомлень 47
2.5.8 Завантаження форми та додавання файлів 47
2.5.8 Завантаження форми та додавання файлів 48
2.5.8 Завантаження форми та додавання файлів 49
2.7 Рекомендації з використання та впровадженню програмного засобу ... 50
ВИСНОВКИ 52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53
ДОДАТОК А 55
ДОДАТОК Б 56

ВСТУП

Актуальність теми. Інтернет перестав бути лише засобом для обміну електронними листами і перетворився на потужну інфраструктуру, що об'єднує головні інформаційні центри, світові бібліотеки, бази даних наукової та правової інформації, а також численні державні та комерційні організації, біржі та банки. Сьогодні Інтернет виступає як універсальна платформа, здатна задовольнити різноманітні потреби людини та значно спростити наше життя. У зв'язку з цим багато підприємств і стартапів прагнуть створювати нові сервіси, що в майбутньому можуть значно покращити наше повсякденне існування.

Автоматизація відправки електронних листів за шаблонами має велике значення для підвищення ефективності та продуктивності користувачів, дозволяючи швидко та точно виконувати рутинні завдання, звільняючи час для більш важливих справ.

Метою роботи є проектування та реалізація програмного забезпечення, яке забезпечить автоматизовану відправку листів на поштові сервіси за шаблоном. Для досягнення цієї мети передбачено розробку інтуїтивно зрозумілого інтерфейсу користувача, реалізацію логіки програми та її інтеграцію з поштовими сервісами.

Об’єкт дослідження – поштові сервіси та програмні засоби автоматизації взаємодії з ними.

Предмет дослідження – процес проектування та реалізації засобів автоматизації роботи з поштовими сервісами, а також технології, методи та інструменти, які використовуються для розробки та впровадження таких програм.

Результати роботи були представлені та обговоренні на наступних конференціях:

- XVIII Міжнародна науково-практична конференція студентів, аспірантів та молодих вчених "Молода наука Волині: пріоритети та перспективи досліджень";
- І Міжнародна наукова конференція "Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем".

РОЗДІЛ 1
ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ ТА ПОШТОВИХ СЕРВІСІВ

1.1 Огляд поштових сервісів

Поштові сервіси є ключовим елементом сучасного цифрового світу, забезпечуючи надійний та ефективний засіб обміну інформацією між користувачами. Розглянемо детальніше різні типи поштових сервісів та їх особливості.

1.1.1 Види поштових сервісів

Існує кілька видів поштових сервісів, які забезпечують різноманітні функції для відправки та отримання електронної пошти. Основні типи таких сервісів наведено на рис. 1.1.

Рисунок 1.1 – Види поштових сервісів

Електронна пошта (e-mail) є не тільки засобом комунікації, але й інструментом для організації робочого процесу та спільного доступу до інформації. Вона дозволяє створювати фільтри та мітки для кращого

сортування повідомлень та автоматичної обробки поточної інформації.

Багато поштових сервісів надають безкоштовне зберігання обмеженої кількості даних, а також платні плани з розширеними можливостями, такими як безлімітне зберігання, підтримка корпоративних доменів тощо. Відправники можуть використовувати аналітичні інструменти для відстеження ефективності розсилок, таких як відсоток відкриття повідомлень, кількість переходів на посилання та інші метрики.

Подібно до електронної пошти, соціальні мережі дозволяють користувачам надсилати приватні повідомлення одне одному, що є основною функцією класичних поштових сервісів. Соціальні мережі надсилають сповіщення про нові повідомлення, коментарі чи реакції, аналогічно до того, як поштові сервіси надсилають сповіщення про нові листи.

У соціальних мережах можна ділитися файлами, фотографіями, документами, що є однією з основних функцій електронної пошти, що дозволяє зберігати та обмінюватися важливими даними. Також, створювання груп для обміну повідомленнями в соціальних мережах, аналогічне груповому листуванню в електронній пошті, сприяє спільній роботі користувачів над проектами або обговоренню різних тем. Соціальні мережі надають можливість зберігати та шукати історію повідомлень, що подібно до функцій архівування та пошуку в електронній пошті (рис.1.2).

Месенджери надають можливість обміну приватними повідомленнями між користувачами, що є аналогом особистої електронної пошти. Створення групових чатів для спілкування в реальному часі нагадує функцію групових листувань у поштових сервісах, дозволяючи командам координуватися та спільно вирішувати питання. Месенджери дозволяють пересилати файли, фотографії та інші документи, що є основною функцією електронної пошти. Вони часто мають вбудовані функції пошуку по історії повідомлень, що схоже на пошукові функції в електронній пошті, дозволяючи швидко знайти необхідну інформацію. Також відбувається інтеграція з іншими сервісами, такими як хмарні сховища, календарі та платіжні системи, що робить їх

схожими на електронну пошту, яка теж часто інтегрується з іншими бізнес-інструментами для покращення продуктивності.

Рисунок 1.2 – Можливості месенджерів

Багато месенджерів надають можливість бачити, коли повідомлення було доставлене та прочитане, що схоже на функції підтвердження доставки та прочитання в електронній пошті.

Поштові списки розсилки відіграють ключову роль у комунікації з аудиторією. Поштові списки розсилки дозволяють створювати повідомлення, які враховують інтереси та потреби аудиторії, що підвищує ефективність комунікації. Для успішної роботи з поштовими списками розсилки необхідно вміти аналізувати дані та використовувати їх для вдосконалення стратегій маркетингу та комунікації. Інструменти автоматизації допомагають відстежувати реакцію аудиторії на розсилки та швидко реагувати на їхні запити чи запитання, що забезпечує ефективну та зручну взаємодію з аудиторією.

1.1.2 Особливості використання різних поштових сервісів

Безпека та конфіденційність. Різні поштові сервіси мають різний рівень захисту даних та приватності. Важливо враховувати ці аспекти під час вибору сервісу для відправки листів.

Електронна пошта відкриває безмежні можливості для спілкування, але використання її може також приховувати певні небезпеки, що варто враховувати. Зокрема, якщо отримано лист від невідомої особи або організації, варто бути обережним. Не можна надіятись на слова або обіцянки без перевірки відправника. Необхідно уникати спаму – це нав'язливої реклами або надмірного розсилання повідомлень. Уникаючи відкриття таких листів, можна зберегти час та зменшите ризик отримання шкідливого вмісту. Потрібно бути обережним з вкладеннями. Ніколи не потрібно відкривати файли або посилання в листах від невідомих джерел. Це може призвести до вірусів або втрати конфіденційної інформації. Перед тим як клікати на посилання у листі, потрібно перевірити URL-адресу. Іноді шахраї можуть намагатися перехопити особисті дані через підроблені посилання. Щоб захистити свій комп'ютер від вірусів і шкідливих програм, потрібно завжди тримати свій антивірусний захист оновленим. Дотримуючись цих простих правил, можна зберегти свою електронну пошту від шкідливих атак та захистити свою конфіденційність.

Кожен поштовий сервіс має свої особливості та можливості. Наприклад, деякі сервіси надають розширені функції фільтрації спаму, інтеграцію з іншими сервісами, можливість планування відправки повідомлень тощо. Розширені функції поштових сервісів можуть значно покращити роботу з електронною поштою. Наприклад, фільтрація спаму дозволяє уникнути надмірного розсилання небажаних повідомлень, що забезпечує чистоту поштової скриньки та збереження затраченого часу. Інтеграція з іншими сервісами, такими як календарі, завдання, зберігання файлів у хмарі, дозволяє ефективно організовувати робочі процеси та зручно управляти усіма завданнями з одного місця.

Можливість планування відправлення повідомлень на конкретний час і

дату є дуже корисною для організації маркетингових кампаній або для того, щоб ваші повідомлення надходили в зручний для отримувача час. Це дозволяє стратегічно планувати комунікації та підвищувати їх ефективність. Також, сервіси, що пропонують створення персоналізованих шаблонів повідомлень, сприяють ефективній комунікації з клієнтами або підписниками, роблячи її більш індивідуальною та професійною.

Для оцінки ефективності розсилок деякі поштові сервіси надають інструменти аналізу, такі як відстеження відкриття повідомлень та кількість переходів на посилання. Ці інструменти допомагають оцінити результативність комунікаційних кампаній, виявити найуспішніші стратегії та підвищити загальну ефективність роботи.

Інтеграція з іншими програмами та платформами робить поштові сервіси більш універсальними та зручними для використання. Наприклад, можливість обговорювати завдання, редагувати документи або співпрацювати над проектами безпосередньо з електронної пошти спрощує комунікацію та співпрацю в команді, підвищуючи продуктивність. Інтеграція також може підвищити продуктивність користувачів шляхом автоматизації деяких рутинних завдань, таких як створення завдань або подій у календарі безпосередньо з поштової скриньки. Це дозволяє швидко організувати робочий час та планувати події, що підвищує ефективність роботи. Деякі інтегровані сервіси надають засоби для аналізу ефективності комунікації та співпраці. Наприклад, можливість переглядати статистику відкриття та відповідей на електронні листи може допомогти визначити найбільш успішні комунікаційні стратегії та підвищити ефективність роботи команди. Інтеграція з іншими сервісами може забезпечити додатковий рівень безпеки та конфіденційності. Наприклад, можливість автоматичного шифрування електронних листів або збереження важливих файлів у безпечному хмарному сховищі забезпечує захист від несанкціонованого доступу до конфіденційної інформації.

Багато поштових сервісів мають мобільні застосунки, які дозволяють

користувачам отримувати доступ до своєї пошти з будь-якого пристрою, що має підключення до Інтернету. Ці застосунки забезпечують користувачам зручний доступ до їхньої електронної пошти навіть у русі, дозволяючи переглядати, відповідати на листи та керувати повідомленнями з будь-якого місця з доступом до Інтернету.

Мобільні застосунки часто синхронізуються з іншими пристроями користувача, такими як комп'ютери або планшети, забезпечуючи легкий доступ до пошти на будь-якому пристрої та однакових повідомлень та даних. Крім того, вони надають можливість отримувати повідомлення в реальному часі про нові листи та інші події в поштовому ящику, що дозволяє швидко реагувати на важливі повідомлення та тримати руку на пульсі подій, навіть коли користувач не перебуває за комп'ютером. Деякі з цих застосунків надають можливість переглядати та редагувати повідомлення навіть без Інтернет-з'єднання, що дозволяє працювати з поштою у ситуаціях, коли доступ до мережі обмежений або відсутній.

1.1.3 Поштові протоколи та їх особливості

При розробці програмних застосунків для роботи з електронною поштою важливо розуміти особливості кожного з основних поштових протоколів: SMTP (Simple Mail Transfer Protocol), POP (Post Office Protocol) і IMAP (Internet Message Access Protocol).

SMTP є стандартним протоколом для відправлення електронних листів між серверами електронної пошти. Він використовується для надсилання повідомлень від клієнтів до поштових серверів та між поштовими серверами. Зазвичай використовуються порти 25, 587 або 465. Порт 25 є стандартним, але порт 587 в більшості використовується для надійних з'єднань, і порт 465 частіше використовується з SSL / TLS.

Шифрування TLS або SSL може бути використане для захисту конфіденційності даних під час передачі через мережу.

POP є протоколом для отримання електронних листів з сервера на

клієнтський пристрій. Він видаляє повідомлення з сервера після завантаження на клієнтський пристрій, хоча можливі варіанти для залишання копій на сервері.

Порт 110 використовується для звичайних з'єднань або порт 995 для захищених (SSL / TLS) з'єднань.

TLS або SSL використовують для захисту з'єднання під час передачі даних.

IMAP також використовується для отримання електронної пошти, але зберігає повідомлення на сервері. Це дозволяє вам організувати та керувати поштою з різних пристроїв.

Порти 143 слугують для звичайних з'єднань або 993 для захищених (SSL / TLS) з'єднань.

Зазвичай використовується TLS або SSL для захисту з'єднання.

1.2 Типові технології та середовища розробки

Розглянемо ключові технології, які використовуватимуться для розробки програми для автоматизованої відправки листів за шаблоном на різні поштові сервіси.

1.2.1 Інтегроване середовище розробки (IDE)

Інтегроване середовище розробки (IDE) є важливим інструментом для розробки програмного забезпечення, в якому розробники можуть писати, відлагоджувати та тестувати свій код. Хоча Visual Studio є найбільш популярним інструментом для розробки програм на мові C#, також існують інші варіанти, включаючи Visual Studio Code.

Visual Studio Code (VS Code) – це легке, але потужне інтегроване середовище розробки, розроблене Microsoft. Однією з переваг VS Code є його легкість використання та налаштування. Він має розширений репозиторій розширень, що дозволяє розробникам налаштувати інструмент під їхні

потреби [9].

Незважаючи на те, що VS Code не має такого широкого спектру можливостей для розробки графічного інтерфейсу, як Visual Studio, він все ще може бути використаний для розробки програм з використанням Windows Forms [10]. Розширення та плагіни, такі як C# for Visual Studio Code, дозволяють розробникам працювати з C# в VS Code з великою ефективністю.

Visual Studio Code надає широкий спектр розширених можливостей редагування коду, включаючи підсвічування синтаксису, автодоповнення коду, рефакторинг та швидкий навігаційний інструментарій. Visual Studio Code надає можливість підсвічування синтаксису, що полегшує розуміння структури коду. Різні елементи мови програмування, такі як ключові слова, змінні, функції та інші конструкції, виділяються різними кольорами, що допомагає розробникам швидше зорієнтуватися в коді та виявити помилки. VS Code має вбудовану функцію автодоповнення коду, яка пропонує варіанти завершення коду на основі контексту історії написання. Це значно прискорює процес написання коду та допомагає уникнути помилок через автоматичне доповнення ключових слів та ідентифікаторів. VS Code надає інструменти для рефакторингу коду, такі як перейменування змінних, виділення фрагментів коду в окремі функції, оптимізація імпортів та інші. Ці функції допомагають зробити код більш структурованим, читабельним та підтримуваним. У VS Code є ряд інструментів для швидкої навігації по коду, таких як швидкий перехід до визначення змінної або функції, викликів функцій, списку методів класу та інші. Ці інструменти дозволяють швидко переміщатися по проекту та швидко знаходити потрібні елементи. Ці функції значно полегшують роботу з кодом та підвищують продуктивність розробника.

Окрім мови C#, Visual Studio Code підтримує широкий спектр інших мов програмування, таких як JavaScript, Python, Java, і багато інших. Це робить його універсальним інструментом для розробників, які працюють з різноманітними технологіями. VS Code відомий своєю легкістю використання та налаштування. Він швидко запускається, має інтуїтивно зрозумілий

інтерфейс користувача та надає можливість налаштовувати робоче середовище згідно із особистими потребами. Visual Studio Code має великий та розмаїтий репозиторій розширень, що дозволяє користувачам налаштувати своє робоче середовище за допомогою додаткових функцій і інструментів. Навіть хоча VS Code не має таких розширених можливостей для розробки графічних інтерфейсів, як Visual Studio, він все ще може бути використаний для розробки програм з використанням Windows Forms або інших бібліотек. Розширення та плагіни, такі як C# for Visual Studio Code, дозволяють розробникам працювати з C# в VS Code з великою ефективністю [11].

Visual Studio Code підтримує функціонал спільної роботи, що дозволяє розробникам працювати над проектами разом, обмінюючись кодом та коментарями безпосередньо в середовищі розробки. Отже, VS Code є привабливим вибором для розробників, які шукають легке та потужне інтегроване середовище розробки для створення програм на мові C# [9].

1.2.2 Мова програмування C#

Мова програмування C# є потужним інструментом для розробки програмного забезпечення на платформі .NET Framework [9]. Особливості та переваги мови C# наведено на рис. 1.3.

C# має чистий та зрозумілий синтаксис, який сприяє простоті розробки і розумінню коду. Це дозволяє розробникам швидко оволодіти мовою та ефективно створювати програми. Синтаксис мови C# має добре розроблену структуру, що дозволяє розробникам швидко та ефективно створювати програми. Чистота та послідовність синтаксису допомагають уникнути зайвих складнощів та забезпечують ефективну роботу над проектами будь-якої складності.

Чіткий та зрозумілий синтаксис мови C# сприяє покращенню читабельності коду. Це дозволяє розробникам швидше розуміти функції програми, виявляти та виправляти помилки та спрощує процес спільної роботи над проектом.

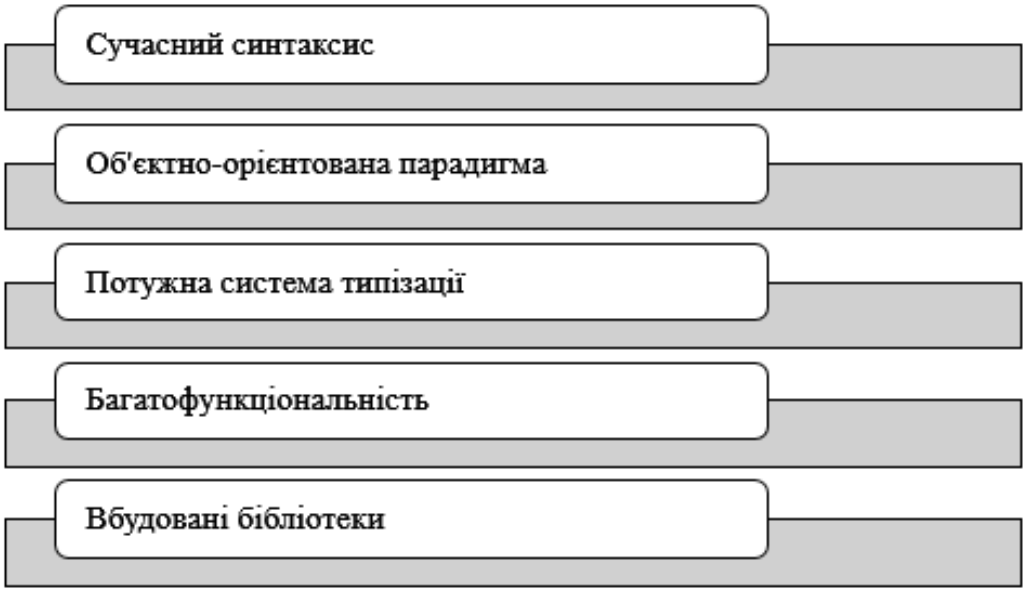

Рисунок 1.3 – Особливості та переваги мови C#

Синтаксис мови C# добре підходить для роботи над великими та складними проектами. Він надає зручні інструменти для організації коду, модульності та структурування, що допомагає розробникам ефективно масштабувати свої проекти з плинним зростанням їхньої складності. C# надає широкий спектр можливостей для розробки різноманітних програм, включаючи веб-застосунки, мобільні застосунки, ігри, робочі програми та багато іншого. Це робить мову C# привабливим вибором для розробників, що працюють у різних сферах індустрії програмного забезпечення. C# підтримує об’єктно-орієнтовану парадигму програмування, що сприяє побудові чіткої та структурованої архітектури програм. Це дозволяє легко управляти складністю програм та забезпечує більшу перевикористовуваність коду. Підтримка ідеалізованої структури програм. Об’єктно-орієнтована парадигма в C# дозволяє програмістам організовувати код у вигляді об’єктів, які представляють реальні або уявні об’єкти в програмі. Це сприяє побудові чіткої та структурованої архітектури програми, що полегшує розуміння та модифікацію коду в майбутньому. C# надає можливості спадкування класів, що дозволяє створювати нові класи на основі вже існуючих. Це допомагає зменшити дублювання коду та сприяє повторному використанню

функціональності. Крім того, поліморфізм дозволяє використовувати об’єкти базового класу у коді, що працює з об’єктами похідних класів, що забезпечує більшу гнучкість та розширюваність програм.

Об’єктно-орієнтована парадигма дозволяє розглядати дані як об’єкти, які мають свої властивості та методи доступу до них. Це дозволяє забезпечити захищеність та конфіденційність даних за допомогою інкапсуляції, обмежуючи доступ до них лише через визначені методи. Об’єктно-орієнтована парадигма дозволяє створювати код, який легко розширювати та модифікувати з мінімальним впливом на існуючий функціонал. Це дозволяє підтримувати та розвивати програму з часом, не потребуючи повного переписування коду. C# використовує статичну сильну типізацію, що дозволяє виявляти помилки під час компіляції та підвищує надійність програм. Це сприяє виявленню та усуненню помилок на ранніх етапах розробки. У C# всі змінні та об’єкти пов’язані з певним типом даних, який визначає їхні можливості та обмеження. Статична сильна типізація означає, що типи даних перевіряються під час компіляції програми, що дозволяє виявляти більшу кількість помилок на ранніх етапах розробки та забезпечує більшу надійність програм.[20]

Статична сильна типізація сприяє підвищенню безпеки програм, оскільки допомагає уникнути помилок, пов’язаних з неправильним типом даних або невірним використанням змінних. Це дозволяє уникнути потенційних проблем виконання програми та забезпечує її стабільну роботу. Такий підхід допомагає розробникам швидше розуміти структуру програми та виявляти помилки під час написання коду. Це збільшує продуктивність розробника та скорочує час, необхідний для розробки та відлагодження програм. Статична сильна типізація сприяє ефективному впровадженню рефакторингу, оскільки забезпечує високий рівень впевненості в безпеці змін. Рефакторинг дозволяє покращити структуру програми та зробити її більш ефективною та зрозумілою.

Завдяки цьому підходу програми на C# зазвичай мають меншу кількість помилок у виробничому середовищі, оскільки більшість помилок виявляються та виправляються на етапі компіляції. C# підтримує широкий спектр функціональних можливостей, включаючи роботу з рядками, масивами, колекціями, файли, мережевими операціями, роботу з базами даних та інше. Це дозволяє розробникам ефективно вирішувати різноманітні завдання в рамках однієї мови програмування.

C# має потужні засоби для обробки рядків і масивів, включаючи різноманітні методи та функції для роботи з текстовими даними, пошуку, заміни, розбиття та з'єднання рядків, а також для маніпулювання елементами масивів. C# надає багато вбудованих структур даних, таких як списки, масиви, стеки, черги, хеш-таблиці та інші, що полегшують роботу з колекціями даних та операції з ними, такі як додавання, видалення, пошук та сортування. Мова програмування C# надає можливості для роботи з файловою системою, включаючи читання, запис, видалення, переміщення та копіювання файлів, а також для роботи з каталогами та шляхами файлів. C# дозволяє легко створювати мережеві застосунки, включаючи роботу з сокетами, HTTP-запитами, веб-службами та іншими протоколами, що дозволяє розробникам створювати різноманітні мережеві застосунки. C# має розширені засоби для роботи з базами даних, включаючи підтримку різних систем управління базами даних (SQL Server, MySQL, PostgreSQL тощо), ORM-фреймворки (Entity Framework, NHibernate) та можливості для виконання SQL-запитів. Окрім цього, C# підтримує багато інших функціональних можливостей, таких як робота з XML, JSON, серіалізація даних, робота з графічними інтерфейсами (за допомогою WPF, WinForms), робота з паралельним програмуванням (за допомогою бібліотеки Task Parallel Library), обробка винятків та багато іншого [7]. C# має велику кількість вбудованих бібліотек, які надають доступ до різних функцій та сервісів, таких як робота з графікою, мережею, шифруванням, асинхронні операції та інше. Це дозволяє розробникам швидко втілювати потрібні функції без необхідності писати власний код з нуля [12].

C# надає доступ до вбудованих бібліотек для роботи з графічними об’єктами та візуалізацією даних. Наприклад, Windows Presentation Foundation (WPF) і Windows Forms дозволяють створювати інтерактивні графічні інтерфейси користувача з використанням готових елементів і компонентів [14]. C# має вбудовані бібліотеки для створення мережевих застосунків, включаючи роботу з сокетами, HTTP-запитами, TCP/IP та UDP-протоколами. Це дозволяє розробникам створювати різноманітні мережеві застосунки без необхідності писати власний код з нуля [15]. C# надає доступ до бібліотек для роботи з шифруванням даних, хешуванням, аутентифікацією та іншими аспектами безпеки. Це дозволяє розробникам створювати безпечні застосунки з ефективними механізмами захисту конфіденційності та цілісності даних. C# підтримує вбудовані бібліотеки для асинхронного програмування, такі як бібліотека Task Parallel Library (TPL) та ключове слово async/await. Це дозволяє розробникам створювати ефективні та швидкодіючі застосунки, що обробляють багатопотоковість та асинхронні запити [12]. Крім того, вбудовані бібліотеки C# надають доступ до інших корисних функцій, таких як робота з XML, JSON, серіалізація даних, робота з базами даних та інші. Це дозволяє розробникам ефективно використовувати готові інструменти для реалізації різноманітних функцій у своїх програмах [13].

1.2.3 Windows Forms (WinForms)

Windows Forms (WinForms) є однією з найпоширеніших технологій для розробки графічного інтерфейсу користувача (GUI) у середовищі .NET Framework. Ця технологія надає розробникам широкий набір інструментів для створення інтерактивних десктопних застосунків зі зручним та привабливим інтерфейсом [18].

Основні характеристики та переваги Windows Forms включають:

- простоту використання;
- багатофункціональність;
- широкий вибір елементів управління;

- інтеграцію з .NET Framework;
- підтримку розширень і сторонніх компонентів.

Windows Forms використовує простий та зрозумілий об'єктно-орієнтований підхід до розробки інтерфейсу, що дозволяє швидко створювати функціональні застосунки навіть для початківців [17]. За допомогою Windows Forms можна легко і швидко створювати застосунки з різноманітними функціями, такими як обробка подій, робота з базами даних, взаємодія з мережею та інші. WinForms містить велику кількість вбудованих елементів управління (controls), таких як кнопки, тексти, списки, таблиці, вкладки тощо, що дозволяє створювати різноманітні та привабливі інтерфейси [1]. Windows Forms є частиною .NET Framework, що робить її ідеальним вибором для розробки застосунків на платформі Windows, забезпечуючи високу сумісність та продуктивність [1]. Розширення та сторонні компоненти дозволяють розробникам значно розширити функціонал Windows Forms, забезпечуючи більшу гнучкість та можливості для реалізації складних завдань [7].

1.2.4 Використання бази даних

База даних – це організована колекція даних, яка зберігається та управляється певною системою управління базами даних (СУБД). Вона використовується для ефективного зберігання, організації та обробки великих обсягів даних у застосунках. Існують різні типи баз даних, включаючи реляційні (наприклад, SQL Server, MySQL, PostgreSQL), NoSQL (наприклад, MongoDB, Cassandra, Redis) та гібридні бази даних. Вибір типу бази даних залежить від потреб застосунку та його конкретних вимог до зберігання даних [2].

Реляційні бази даних − це найпоширеніший тип баз даних, де дані організовані у вигляді таблиць з реляціями між ними. Реляційні бази даних використовують мову SQL для створення, зміни та запитів до даних. Вони надійні, мають сильні переваги в нормалізації даних та забезпечують консистентність даних. Популярні представники цього типу − SQL Server,

MySQL, PostgreSQL, Oracle тощо.

NoSQL бази даних призначені для зберігання та обробки великих обсягів неструктурованих даних. Цей тип бази даних відмінно працює з даними, які можуть мати змінювану структуру або не підлягати строгій нормалізації. Вони зазвичай масштабуються горизонтально, що дозволяє їм працювати з великими обсягами даних та високими навантаженнями. Популярні представники NoSQL − MongoDB, Cassandra, Redis, Couchbase тощо.

Гібридні бази даних поєднують в собі як реляційні, так і NoSQL характеристики. Вони можуть забезпечити гнучкість NoSQL баз даних разом із здатністю до виконання складних операцій, характерних для реляційних баз даних. Гібридні бази даних надають розробникам більше можливостей для оптимізації продуктивності та пристосування до конкретних вимог застосунків.

Реляційні бази даних дозволяють зберігати, оновлювати, видаляти та вибирати дані за допомогою SQL-запитів (Structured Query Language). Вони також надають можливості для виконання операцій з обробки даних, таких як сортування, фільтрація, групування та обчислення. Бази даних можуть підтримувати механізми транзакцій, що дозволяють виконувати групу операцій як атомарну одиницю. Це забезпечує консистентність даних та захист від втрати даних в разі непередбачуваних ситуацій, таких як відмови системи або помилки. Реляційні бази даних дозволяють встановлювати зв’язки між таблицями за допомогою зовнішніх ключів. Це забезпечує цілісність даних та зручність при роботі з взаємозалежними даними. Бази даних надають засоби для резервного копіювання даних та відновлення їх у випадку втрати або пошкодження. Це забезпечує безпеку та надійність збереження даних у випадку непередбачуваних ситуацій. Бази даних використовуються у багатьох типах застосунків [16] (рис.1.4). Вони використовуються для зберігання користувацьких даних, конфігураційних даних, історії та журналів подій, контенту та іншої інформації, необхідної для роботи застосунку.

Рисунок 1.4 – Типи застосунків

Бази даних виконують важливу функцію в різних аспектах зберігання та управління даними. Вони використовуються для зберігання профілів користувачів, історії їх дій та персоналізованих налаштувань, що дозволяє забезпечити індивідуальний підхід та підвищити зручність використання вебзастосунків. Крім цього, бази даних дозволяють зберігати та управляти великим обсягом контенту, включаючи статті, зображення, відео та інші мультимедійні матеріали, що є необхідним для багатьох сучасних веб-сервісів.

Бази даних також відіграють ключову роль у системах керування замовленнями та транзакціями. Вони зберігають і обробляють інформацію про замовлення, транзакції покупок, оплату та доставку, що забезпечує надійне функціонування електронної комерції та інших сервісів, пов’язаних із продажем товарів і послуг.

На мобільних пристроях бази даних використовуються для зберігання різноманітних даних, таких як налаштування застосунку, офлайн-контент і локальна кеш-пам’ять. Це дозволяє застосункам працювати більш ефективно і

надавати користувачам доступ до необхідної інформації навіть без підключення до інтернету. Крім того, бази даних забезпечують синхронізацію даних між мобільним пристроєм та веб-сервером, що гарантує актуальність даних незалежно від пристрою, на якому користувач працює. Це особливо важливо для застосунків, що використовуються на різних платформах і потребують постійного оновлення та узгодження інформації.

Бази даних виконують важливу функцію у забезпеченні можливості роботи застосунків без підключення до мережі. Вони зберігають локальні налаштування, журнали подій, тимчасові дані та іншу інформацію, необхідну для коректного функціонування програм. Завдяки цьому користувачі можуть продовжувати використовувати застосунки і мати доступ до необхідних даних навіть за відсутності Інтернету.

Локальні бази даних особливо корисні для десктопних застосунків, які повинні зберігати та обробляти інформацію автономно. Ці бази даних дозволяють зберігати дані на пристрої користувача, забезпечуючи безперебійну роботу застосунку і збереження інформації, навіть коли немає доступу до Інтернету. Це особливо важливо для застосунків, які повинні надавати швидкий і надійний доступ до даних у будь-який момент, наприклад, програми для обробки документів, управління проектами або ведення фінансової звітності. Крім того, локальні бази даних можуть використовуватися для зберігання офлайн-контенту, що дозволяє користувачам завантажувати та переглядати необхідні матеріали без постійного підключення до мережі. Це може бути особливо корисним у випадках, коли доступ до Інтернету обмежений або коштовний, або коли користувачі знаходяться в місцях з нестабільним зв’язком. Таким чином, локальні бази даних забезпечують гнучкість і автономність у роботі з застосунками, підвищуючи їх ефективність і зручність для кінцевих користувачів.

Бази даних застосовуються для зберігання та управління різноманітними типами контенту, такими як медіафайли, документи, електронні книги та інші

матеріали. Вони також використовуються в застосунках для аналізу даних, генерації звітів, статистики та іншої інформації, що допомагає у прийнятті рішень.

Один із значних переваг баз даних полягає у їхній здатності легко інтегруватися з застосунками, незалежно від платформи чи мови програмування. Інтерфейси програмування застосунків (API) систем управління базами даних (СУБД) дозволяють застосункам взаємодіяти з базою даних, виконуючи запити та отримуючи результати. Це забезпечує ефективний зв'язок між застосунком і базою даних, дозволяючи швидко та надійно опрацьовувати інформацію. Більшість сучасних баз даних надають API для взаємодії з застосунками. Ці API забезпечують стандартний набір операцій для виконання запитів до бази даних, отримання та оновлення даних, а також керування конфігурацією бази даних.

ORM (Object-Relational Mapping) − це технологія, яка дозволяє взаємодіяти з базою даних за допомогою об'єктів програмного забезпечення, а не SQL-запитів. Це спрощує розробку застосунків, оскільки розробники можуть працювати з об'єктами, а не з таблицями бази даних. Існують багато бібліотек та фреймворків, які спрощують інтеграцію з базами даних для різних мов програмування. Наприклад, для мови Python є бібліотеки, такі як SQLAlchemy або Django ORM, для Java - Hibernate або JPA. Бази даних можуть підтримувати різні формати даних для взаємодії з застосунками, такі як JSON, XML, CSV тощо. Це дозволяє застосункам легко обмінюватися даними з базою даних без необхідності перетворення форматів. Під час інтеграції з застосунками бази даних забезпечують механізми шифрування та аутентифікації для захисту конфіденційності та цілісності даних. Безпека даних є важливим аспектом баз даних. Системи управління базами даних надають засоби для захисту даних шляхом автентифікації користувачів, авторизації доступу до даних, шифрування даних та аудиту доступу.

Безпека даних включає в себе захист інформації шляхом шифрування. Бази даних надають можливості для шифрування як самої бази даних, так і

окремих полів або таблиць у базі даних. Шифрування даних забезпечує захист від несанкціонованого доступу навіть у випадку, якщо зловмисник здобуде доступ до файлів баз даних. Бази даних надають засоби для ідентифікації користувачів та контролю їх доступу до різних ресурсів бази даних. Це включає в себе створення користувачів з унікальними обліковими записами, надання їм прав доступу до конкретних таблиць або операцій, а також встановлення паролів та політик паролів для забезпечення безпеки облікових записів. Бази даних можуть вести журнали аудиту, які записують дії користувачів та адміністраторів бази даних. Ці журнали дозволяють виявляти та відслідковувати несанкціонований доступ, зміни в даних або інші підозрілі дії. Аудит доступу допомагає виявляти порушення безпеки та приймати заходи для їх усунення.

Бази даних надають механізми для захисту від SQL-ін’єкцій та інших атак, спрямованих на недостойне використання програмних інтерфейсів або недотримання прав доступу. Це включає в себе використання параметризованих запитів, валідацію введених даних та обмеження доступу до функцій бази даних.

Адміністратори баз даних повинні регулярно виконувати оновлення систем управління базами даних та встановлювати виправлення безпеки для захисту від відомих вразливостей. Крім того, вони повинні моніторити нові загрози та вразливості і приймати відповідні заходи для їх усунення. Бази даних повинні бути масштабованими та відмовостійкими, щоб забезпечити ефективну роботу застосунків у випадку зростання навантаження або виникнення проблем з системою. Тому важливо обирати СУБД, які відповідають вимогам застосунку за масштабованістю та надійністю. Бази даних повинні підтримувати можливість масштабування як в вертикальному, так і в горизонтальному напрямках. Вертикальна масштабованість полягає у збільшенні ресурсів (наприклад, обсягу пам’яті або потужності процесора) на існуючому сервері, тоді як горизонтальна масштабованість передбачає розширення бази даних на декілька серверів або вузлів для розподілення

навантаження. Деякі системи управління базами даних надають можливість автоматичної реплікації даних на різні сервери або вузли для забезпечення резервного копіювання та відмовостійкості. Шарування даних дозволяє розділити базу даних на логічні шари або фрагменти, які можуть бути розподілені між різними серверами або вузлами для оптимізації доступу та роботи з даними. Ефективна масштабованість баз даних вимагає систем моніторингу, які виявляють зростання навантаження та автоматично виконують масштабування ресурсів або реплікацію даних для забезпечення надійності та продуктивності. Бази даних повинні мати механізми для автоматичного відновлення після збоїв або відмов. Це включає в себе резервне копіювання даних, створення точок відновлення, механізми реплікації та кластеризації для забезпечення доступності даних та надійності роботи застосунків.

1.2.5 Інтеграція з поштовими сервісами

Інтеграція з поштовими сервісами дозволяє відправляти листи з програми безпосередньо на адреси отримувачів за допомогою SMTP-протоколу. У реалізації нашої програми можна використовувати стандартні бібліотеки .NET для взаємодії з SMTP-серверами поштових сервісів [8]. При цьому потрібно врахувати такі аспекти:

- безпека та автентифікація.
- обробка помилок та відповідей сервера;
- моніторинг та журналювання;
- тестування та валідація;
- документація та підтримка.

Для надсилання листів через поштові сервіси зазвичай потрібна автентифікація користувача. Це може вимагати використання різних методів автентифікації, таких як токени доступу або паролі, які будуть безпечно збережені в конфігураційних файлах або інших захищених механізмах.

Під час надсилання листів можуть виникати різноманітні помилки, або сервер може повертати різні відповіді. Застосунок повинен мати механізми обробки цих сценаріїв, надсилаючи повідомлення про помилки або вживаючи відповідні заходи в залежності від отриманих відповідей сервера.

Ведення журналу процесу відправлення листів допоможе виявити проблеми та покращити процес в майбутньому. Журнал повинен містити інформацію про успішні та невдалі спроби, а також детальні записи про можливі помилки.

Перед впровадженням програмного застосунку важливо протестувати механізм надсилання листів для підтвердження коректності роботи інтеграції з поштовими сервісами. Також варто провести валідацію надісланих листів для перевірки їхнього успішного отримання та доставки.

Документування процесу інтеграції з поштовими сервісами має включати конфігураційні налаштування та вимоги до безпеки. Крім цього, необхідно забезпечити підтримку користувачам щодо відправлення листів та вирішення будь-яких проблем, які можуть виникнути у процесі їхньої надсилання.

1.2.6 Дизайн і архітектура програмного забезпечення

Визначення дизайну і архітектури програмного забезпечення є критичним кроком у розробці програмних продуктів для автоматизованої відправки листів. Ось які принципи та методи планується використовувати:

- об’єктно-орієнтоване програмування (ООП);
- розділення на модулі та шари;
- використання шаблонів проектування;
- забезпечення тестової покриття.

Принципи об’єктно-орієнтованого програмування (ООП) для створення застосунків базується на взаємодії об’єктів для виконання різних функцій. Це дозволяє створити чітку структуру застосунку та зменшити залежність між її компонентами. Логіка програми повинна бути розділена на окремі модулі та

шари, що дозволить ефективніше керувати розвитком та тестуванням. Наприклад, потрібно створити окремі модулі для роботи з базою даних, взаємодії з поштовими сервісами, обробки даних тощо.

Для підтримки масштабованості та розширюваності застосунку варто використовувати шаблони проектування, такі як фабричний метод, одинак, спостерігач тощо. Наприклад, шаблон "Фабричний метод" дозволить гнучко створювати різні типи листів для відправлення. Активне тестування може застосовуватись для перевірки правильності роботи застосунку та його модулів. Це допоможе уникнути проблем на пізніх етапах розробки та підвищить стабільність застосунку.

Загалом, правильне планування архітектури застосунку на початковому етапі розробки є ключовим для успішної реалізації його функціональності для автоматизованої відправки листів.

1.2.7 Функціонал Windows Forms

Функціонал Windows Forms (WinForms) включає в себе широкий набір можливостей для створення інтерактивних десктопних застосунків у середовищі .NET. [3] (рис.1.5). Windows Forms дозволяє швидко та легко створювати графічний інтерфейс користувача (GUI) за допомогою різноманітних елементів управління, таких як кнопки, тексти, випадаючі списки, таблиці тощо. Розміщення та налаштування цих елементів виконується за допомогою візуального дизайнера, що спрощує процес розробки. Windows Forms дозволяє обробляти різні події, які виникають під час взаємодії користувача з програмою, такі як натискання кнопок, введення тексту, переміщення миші тощо. Це дає можливість створювати реактивні та інтерактивні, які відповідають на дії користувача. Окрім цього, Windows Forms надає можливості для роботи зі зображеннями, малюванням простих графічних об’єктів та налаштуванням їх параметрів, корисних для створення ілюстрацій, діаграм, графіків тощо. Також, використання Windows Forms дозволяє створювати та керувати різними вікнами і діалоговими вікнами

програми, такими як вікна з повідомленнями, підтвердженням операцій, налаштувань тощо.

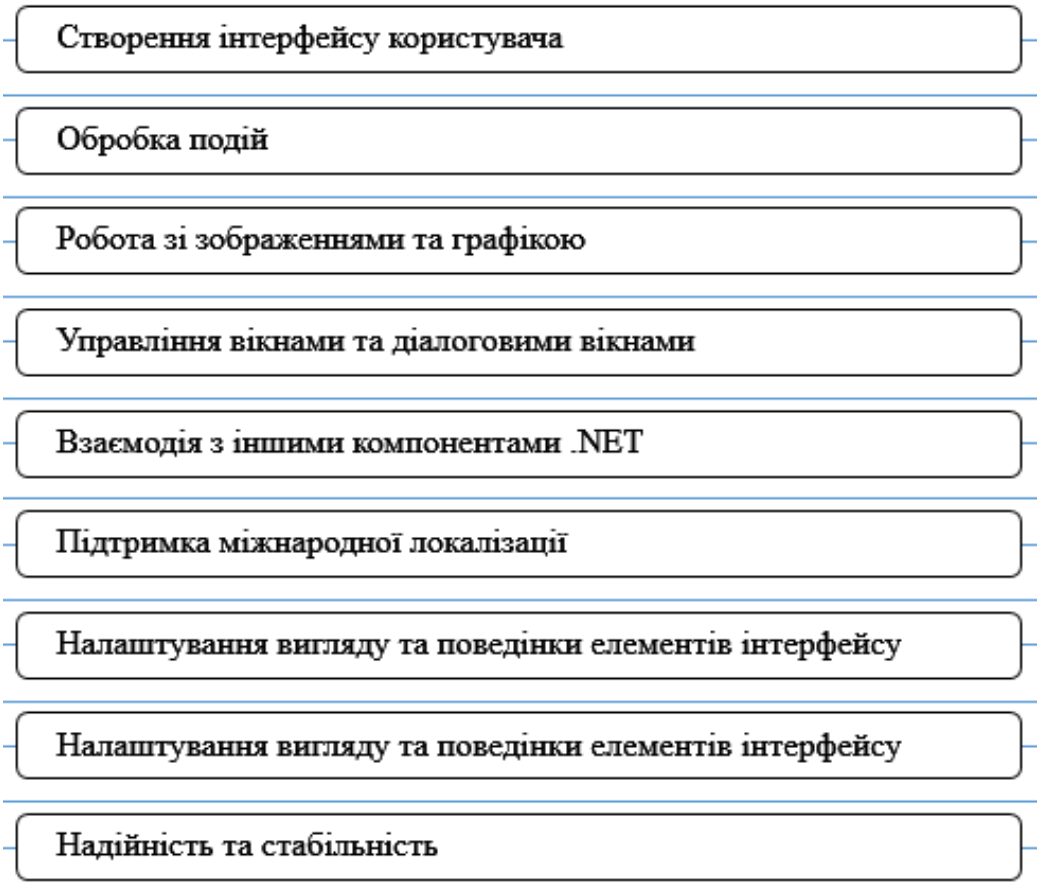

Рисунок 1.5 – Основні можливості Windows Forms

Windows Forms легко інтегрується з іншими компонентами та бібліотеками .NET, такими як ADO.NET для роботи з базами даних, WCF для створення веб-служб, а також з різноманітними сторонніми бібліотеками. Також, Windows Forms дозволяє легко локалізувати програму для різних мов та регіональних налаштувань, що робить її доступною для широкого кола користувачів. Не останнє місце важливості відіграє розширена можливість налаштування вигляду та поведінки елементів інтерфейсу, таких як кольори, розміри, шрифти, вирівнювання тощо. Windows Forms славиться своєю стабільністю та надійністю, роблячи його відмінним вибором для створення надійних та ефективних десктопних застосунків.

1.2 Архітектурні особливості поштових сервісів

Архітектурні особливості поштових сервісів включають різноманітні аспекти, що охоплюють як фізичну, так і логічну організацію сервісу, його робочі процеси, зв'язки між компонентами та інші ключові аспекти. Розглянемо деякі з них більш детально [8].

Більшість поштових сервісів базуються на клієнт-серверній архітектурі, де клієнтська програма (зазвичай поштовий клієнт) взаємодіє з центральним сервером, що зберігає, обробляє та надсилає електронні листи. Ця модель дозволяє забезпечити централізований доступ до електронної пошти з будь-якого пристрою, підключеного до Інтернету.

Архітектура поштових сервісів повинна бути розроблена з урахуванням масштабованості, тобто здатності обробляти велику кількість повідомлень та користувачів. Це може включати розподілені системи, горизонтальне та вертикальне масштабування, кешування та інші стратегії оптимізації продуктивності.

Оскільки електронна пошта містить конфіденційну інформацію, архітектура поштового сервісу повинна забезпечувати високий рівень безпеки. Це включає шифрування передачі даних, аутентифікацію користувачів, захист від спаму та фішингу, а також механізми контролю доступу до поштової скриньки.

Для зменшення навантаження на сервери та прискорення роботи поштового сервісу, може бути використана система кешування, що дозволяє зберігати частину даних локально на клієнтському пристрої або на проміжному сервері.

Деякі поштові сервіси можуть мати можливості інтеграції з іншими сервісами та застосунками, такими як календарі, завдання, зберігання файлів, соціальні мережі тощо. Це дозволяє користувачам зручно керувати своєю електронною поштою та пов'язаними з нею сервісами в одному місці.

Сучасні поштові сервіси повинні бути адаптовані до різних типів пристроїв і розмірів екранів, забезпечуючи зручний доступ до електронної

пошти незалежно від пристрою, на якому вона переглядається.

Архітектура поштових сервісів включає механізми для організації та управління масовими розсилками. Це передбачає налаштування списків розсилки, планування відправки повідомлень, автоматизацію відповіді та відстеження ефективності розсилок через аналітичні інструменти. Вимоги до таких систем включають надійність доставки, уникнення потрапляння в спам, персоналізацію повідомлень і дотримання законодавства про конфіденційність даних. Загальна інформація про розсилки включає наступні особливості функціонування та організації:

- налаштування списків розсилки (формування та підтримка актуальних списків отримувачів, що може включати сегментацію аудиторії за різними критеріями: демографічні дані, інтереси, поведінка);
- планування відправки повідомлень (автоматизовані інструменти для планування часу відправки розсилок з метою максимізації їх відкриття та залученості користувачів);
- автоматизація відповіді (використання автоматичних відповідей для підтвердження отримання повідомлення, подяки за підписку, нагадувань тощо);
- відстеження ефективності розсилок (аналітичні інструменти для моніторингу ключових метрик, таких як відсоток відкриття повідомлень, кількість переходів за посиланнями, конверсії та інші показники ефективності);
- надійність доставки (забезпечення доставки повідомлень до поштових скриньок отримувачів, включаючи обходження фільтрів спаму та інших бар’єрів);
- персоналізація повідомлень (використання даних про отримувачів для персоналізації змісту листів, що підвищує їх релевантність та ефективність).

1.4 Дослідження аналогів застосунку

Порівняння сучасних аналогічних розробок у сфері автоматизованої відправки листів за шаблоном дає можливість з'ясувати різні підходи до цієї задачі та ідентифікувати переваги та недоліки кожного з них. Нижче наведено огляд декількох аналогічних розробок, що не є веборієнтованими хмарними платформами.

GroupMail – це програмне забезпечення для автоматизації розсилок, яке встановлюється локально на комп'ютер (рис.1.6). GroupMail дозволяє створювати та відправляти масові розсилки, управляти списками контактів та відстежувати ефективність кампаній. Програма підтримує інтеграцію з різними серверами SMTP та має потужні інструменти для персоналізації повідомлень.

Рисунок 1.6 – Головне вікно GroupMail

Atomic Mail Sender – локально встановлюване програмне забезпечення для масових розсилок електронної пошти (рис.1.7). Atomic Mail Sender дозволяє відправляти тисячі електронних листів за короткий час, підтримує різні SMTP-сервери та забезпечує можливості для спостереження за ефективністю розсилок, такі як відстеження відкриття листів та кліків.

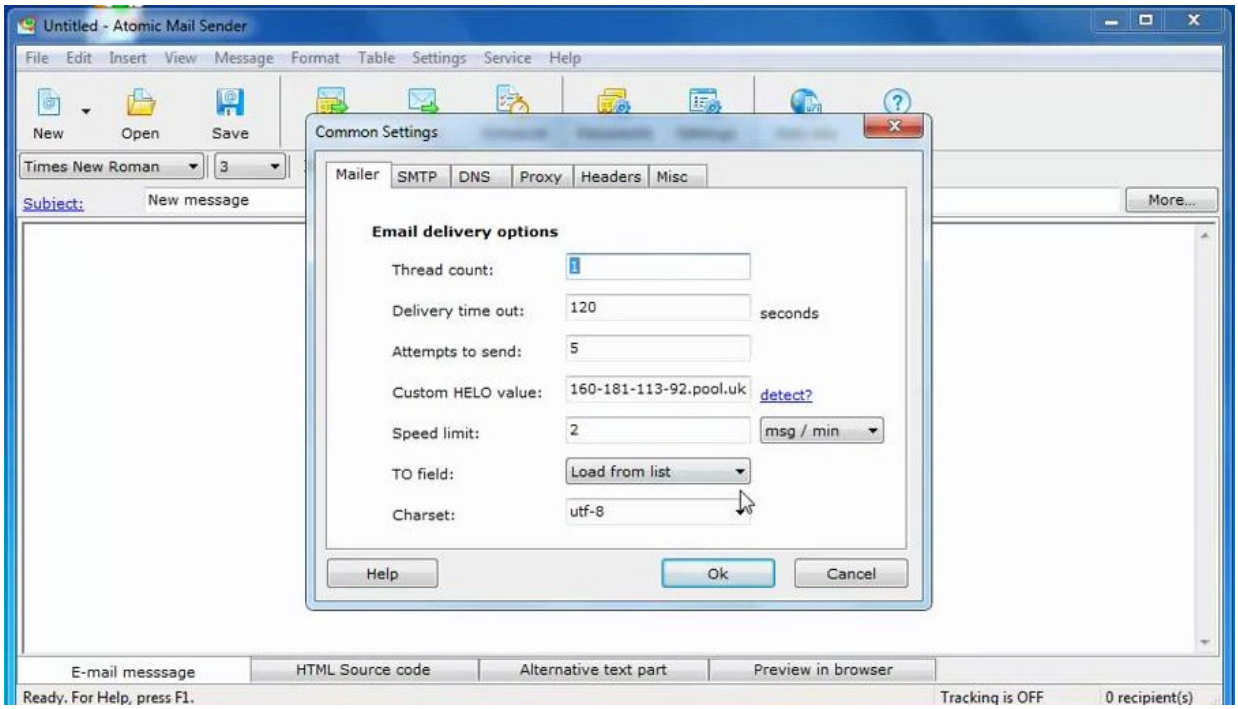

Рисунок 1.7 – Вікно налагодження Atomic Mail Sender

MaxBulk Mailer – це застосунок для розсилки електронних листів, який підтримує роботу на різних платформах, включаючи Windows і macOS. MaxBulk Mailer дозволяє створювати, відправляти та відстежувати масові розсилки з великою кількістю налаштувань для сегментації аудиторії та персоналізації повідомлень (рис.1.8).

SendBlaster – локальне програмне забезпечення для управління та автоматизації масових розсилок електронної пошти. SendBlaster пропонує інтуїтивно зрозумілий інтерфейс для створення шаблонів листів, управління контактами та відстеження результатів розсилок. Застосунок підтримує роботу з різними SMTP-серверами та має інструменти для запобігання потрапляння листів у спам (рис.1.9).

Gammadyne Mailer – це потужний інструмент для автоматизації розсилок електронної пошти, який встановлюється на комп’ютер. Gammadyne Mailer пропонує широкий спектр функцій для створення, надсилання та аналізу масових розсилок, включаючи детальне відстеження метрик, інтеграцію з базами даних та підтримку скриптів для персоналізації. Приклад інтерфейсу Gammadyne Mailer представлено на рисунку 1.10.

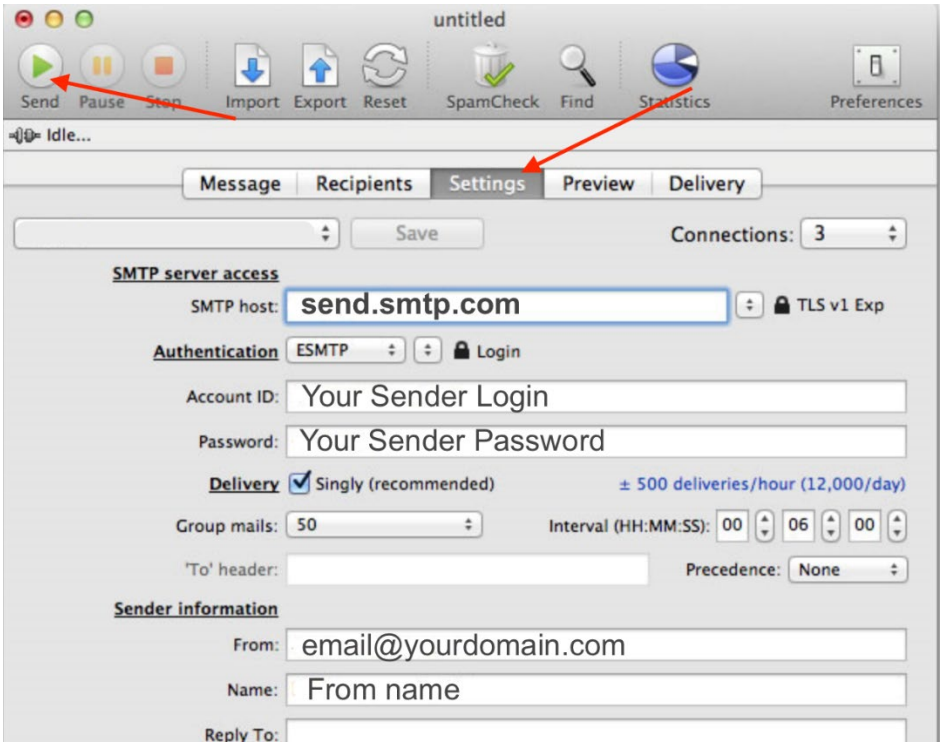

Рисунок 1.8 – Вкладка налагодження MaxBulk Mailer

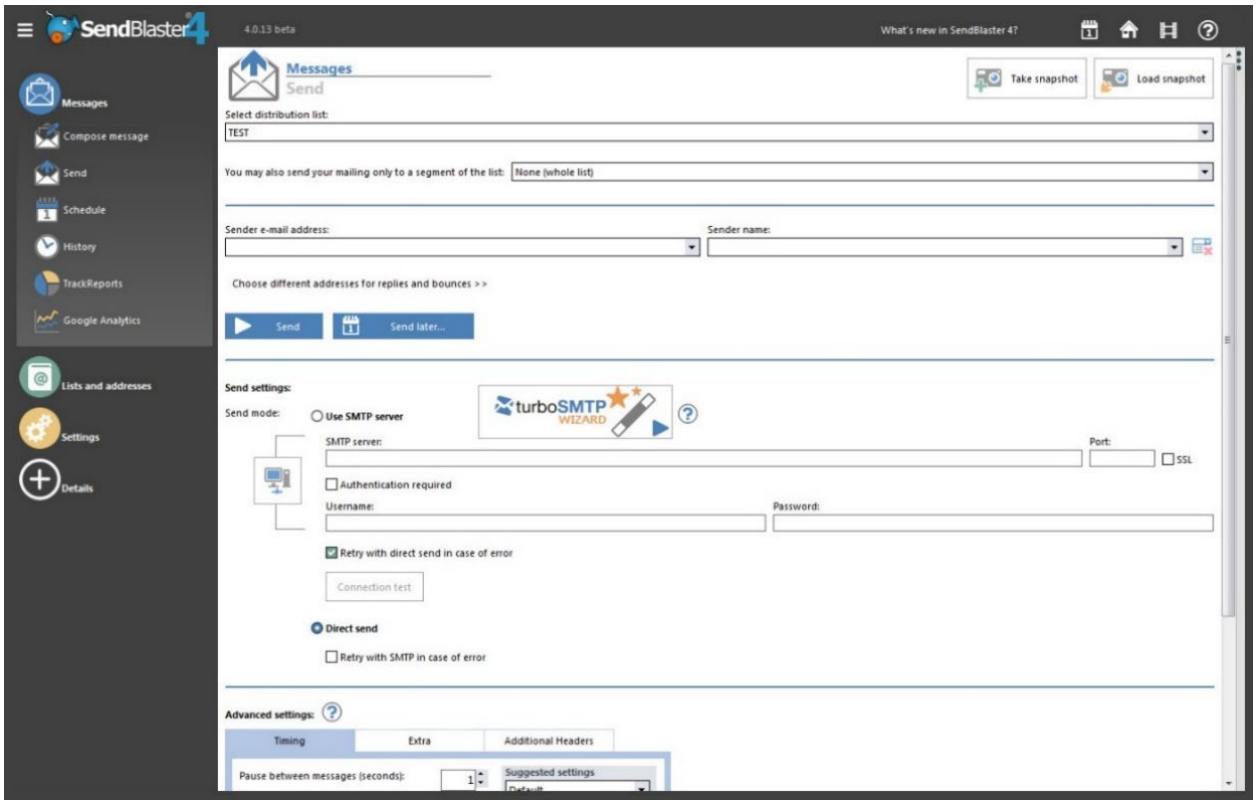

Рисунок 1.9 – Інтерфейс налагодження SendBlaster

Кожен з цих аналогів має свої переваги та обмеження, і вибір конкретного рішення буде залежати від потреб і вимог вашого проекту.

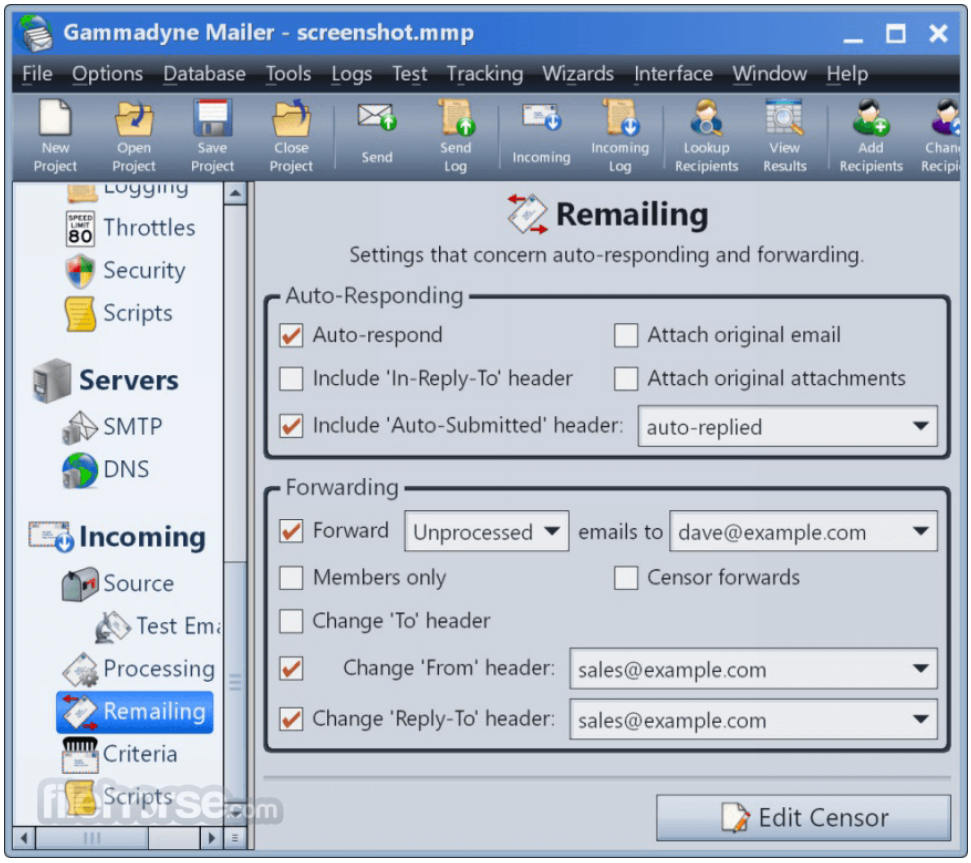

Рисунок 1.10 – Приклад інтерфейсу Gammadyne Mailer

Кожен із розглянутих аналогів має свої унікальні переваги та обмеження, що робить їх придатними для різних завдань та умов використання. Вибір конкретного рішення значною мірою залежить від специфічних потреб і вимог вашого проєкту. Наприклад, один інструмент може забезпечувати кращу інтеграцію з базами даних, інший – пропонувати більш гнучкі можливості для персоналізації, а третій – мати потужні функції для детального відстеження метрик. Щоб допомогти ухвалити обґрунтоване рішення, було створено порівняльну таблицю, яка детально відображає

ключові характеристики кожного з цих застосунків. Цю таблицю подано на рис. 1.11.

Програмне забезпечення	Переваги	Недоліки
GroupMail	Інтеграція з SMTP серверами, потужні інструменти для персоналізації	Потребує локальної установки, можлива крива навчання
Atomic Mail Sender	Висока ємність відправки електронних листів, відстеження відкриттів та кліків	Потребує локальної установки, можливе високе споживання ресурсів
MaxBulk Mailer	Підтримка різних платформ, великі можливості сегментації та персоналізації	Потребує локальної установки, просунуті функції можуть вимагати навчання
SendBlaster	Інтуїтивно зрозумілий інтерфейс, інструменти для запобігання потрапляння листів у спам	Потребує локальної установки, обмежений до локальних ресурсів
Gammadyne Mailer	Широкий спектр функцій, інтеграція з базами даних, підтримка скриптів	Потребує локальної установки, складна конфігурація

Таблиця 1.1 – Порівняльна таблиця розглянутих застосунків

РОЗДІЛ 2
РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗОВАНОЇ ВІДПРАВКИ ЛИСТІВ

2.1 Постановка задачі, призначення та вимоги для програми відправки листів

Проект передбачає створення багатофункціонального застосунку для автоматизованої відправки листів з використанням зазначеного шаблону на різні поштові сервіси. Цей застосунок буде спрощувати та оптимізувати процес відправлення електронних листів, надаючи користувачеві ряд зручних та необхідних функцій.

Серед них:

- інтерактивний інтерфейс;
- управління адресами отримувачів;
- шаблонизація листів;
- підтримка різних поштових сервісів;
- масова відправка листів.

Застосунок забезпечить інтуїтивно зрозумілий інтерфейс, який дозволить користувачу легко вводити необхідну інформацію для відправки листа. Поля для введення адреси отримувача, заголовку листа, тексту повідомлення та прикріплення файлів будуть доступні для заповнення.

Крім того, користувач матиме можливість додавати, видаляти та редагувати адреси електронних скриньок отримувачів. Це дозволить зберігати адреси в обліковому записі додатку і використовувати їх для автоматичного вибору адресата при наступних відправках листів.

Застосунок також повинна надавати можливість використання шаблонів для листа. Користувач повинен мати змогу визначити певні поля в тексті повідомлення, які будуть автоматично заповнюватися при кожній відправці.

Застосунок повинен підтримувати відправку електронних листів як із різних поштових сервісів, так і на різні поштові сервіси, такі як Gmail, Outlook, Yahoo та інші, забезпечуючи гнучкість та можливість вибору для користувача. Це зробить наш застосунок важливим інструментом у сфері комунікацій та бізнесу, спрощуючи процес відправки листів та забезпечуючи надійність та зручність в користуванні.

Також, застосунок повинен дозволяти надсилати повідомлення декільком адресатам одночасно, що спростить процес масової комунікації та підвищить ефективність роботи користувача.

2.2 Загальний опис проекту

Проект полягає в розробці та впровадженні програмного забезпечення, спрямованого на автоматизацію процесу відправки електронних листів як на різні поштові сервіси, так і з різних поштових сервісів. Це інноваційний інструмент, який спрощує та оптимізує масову відправку листів для користувачів будь-якого рівня.

Застосунок повинен надавати користувачеві можливість зручного створення та надсилання листів з використанням попередньо визначеного шаблону. Інтерактивний інтерфейс повинен дозволяти легко ввести адресу отримувача, заголовок листа, текст повідомлення та прикріпити файли.

Основні функціональні можливості повинні включати управління списком адрес отримувачів, що дозволить додавати, видаляти та редагувати контакти для подальшої автоматичної відправки. Також потрібно передбачити можливість використання шаблону для листа, що забезпечує швидке та нестандартне формулювання повідомлень.

Програмне забезпечення повинно підтримувати відправку листів на різні поштові сервіси, що дозволить користувачеві вибирати найзручніший для нього спосіб комунікації. Завдяки попередньому перегляду перед

відправкою, користувач зможе переконатися у правильності введеної інформації та відформатованому вигляді листа.

На рисунку 2.1 зображено UML-діаграму класів для системи автоматизації відправки електронних листів. Загальною метою проекту є створення ефективного інструменту для ведення кореспонденції, який заощаджує час та зусилля користувача, покращує комунікацію та сприяє автоматизації бізнес-процесів.

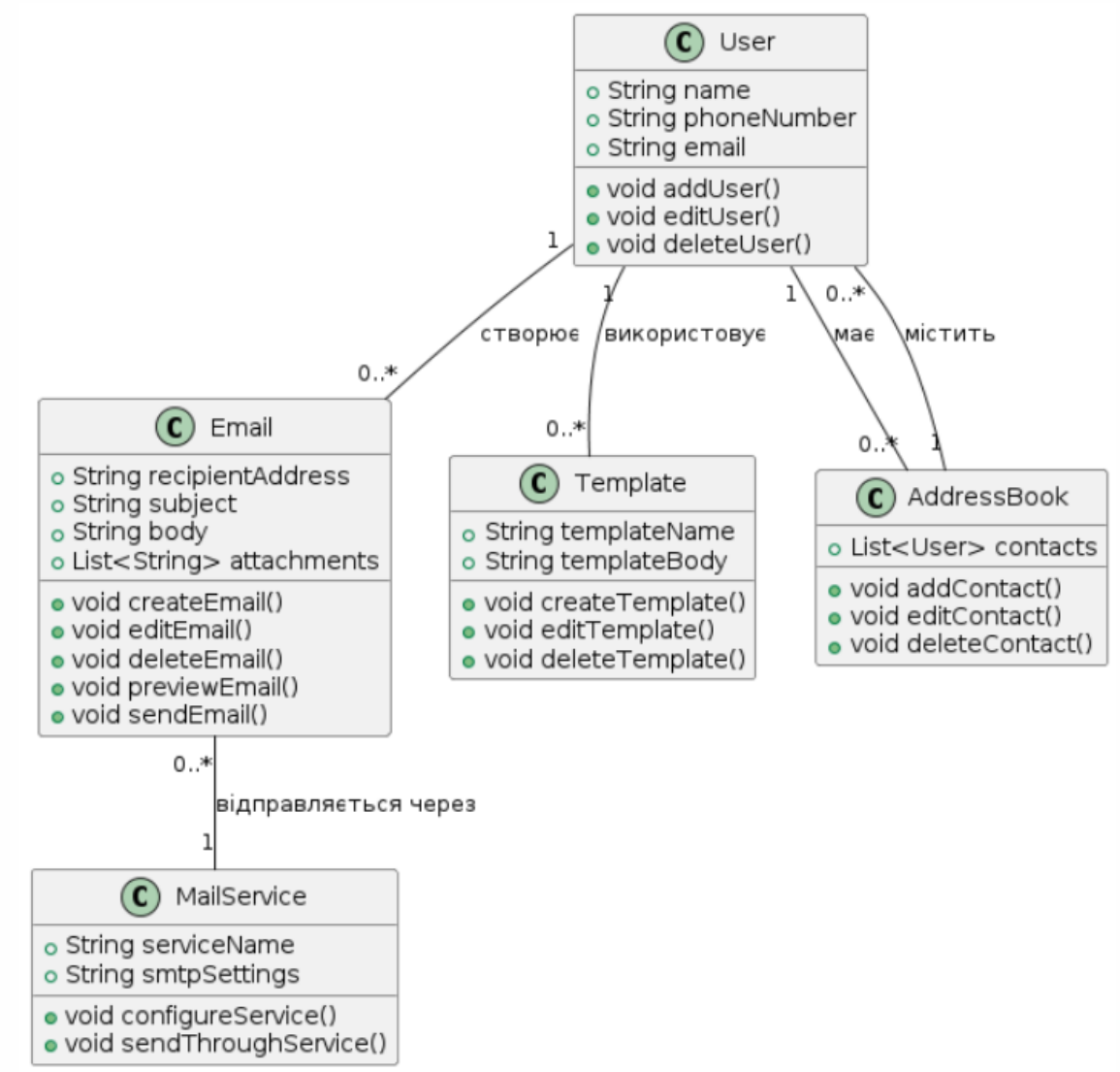

Рисунок 2.1 – UML-діаграма класів для системи автоматизації відправки електронних листів

2.3 Вибір моделі розробки програмного засобу

Оцінюючи попередні рішення та обґрунтовуючи вибір платформи для розробки програмного забезпечення для автоматизованої відправки листів за шаблоном через різні поштові сервіси, було визначено, що для наших потреб найбільш відповідною є гібридна модель розробки.

Гібридна модель полягає в поєднанні каскадної моделі з ітеративним підходом. Було обрано цей підхід через його спроможність забезпечити структурований процес розробки, а також гнучкість, необхідну для адаптації до змін у вимогах чи технічних ускладнень під час проектування та розробки програмного засобу. У нашому випадку, розробка програмного застосунку для відправки листів за шаблоном передбачає поетапне виконання робіт, включаючи аналіз вимог, проектування інтерфейсу користувача та архітектури застосунку, реалізацію функціоналу, тестування та впровадження.

Крім того, гібридна модель дозволяє поєднувати переваги різних методик розробки, що сприятиме якості та швидкості розробки проекту. Наприклад, на початкових етапах ми можемо використовувати каскадний підхід для чіткого визначення вимог та створення загального плану проекту, а наступні етапи можуть включати ітеративні ітерації для постійного вдосконалення функціоналу та адаптації до змін.

Отже, обрана гібридна модель розробки забезпечить нам оптимальне співвідношення між стратегічним плануванням та гнучкістю, необхідною для успішної реалізації нашого програмного забезпечення для автоматизованої відправки листів.

2.4 Обґрунтування вибору інструментальних засобів розробки

2.4.1 Вибір Visual Studio як основного інтегрованого середовища розробки (IDE):

Visual Studio є надійним та широко використовуваним інтегрованим середовищем розробки, яке забезпечує розширені можливості для створення програмного забезпечення. Використання Visual Studio дозволить нам ефективно використовувати всі можливості мови програмування C# та різноманітні функції .NET Framework. Крім того, Visual Studio надає інтегровані інструменти для відлагодження, тестування та підтримки версійного контролю, що спростить розробку нашого програмного продукту.

Visual Studio володіє широким співтовариством розробників та підтримується компанією Microsoft, що гарантує актуальність та надійність середовища. Крім того, Visual Studio пропонує широкий набір плагінів та розширень, які дозволяють розширити його функціональність та адаптувати під конкретні потреби розробника.

Використання Visual Studio дозволить нам підтримувати високий рівень продуктивності та якості в процесі розробки, забезпечуючи при цьому зручне та ефективне середовище для всіх учасників команди.

2.4.2 Використання мови програмування C#.

Мова програмування C# була обрана через свою ефективність, гнучкість та широкий функціонал, що дозволяє розробляти різноманітні програмні продукти. C# є однією з найбільш популярних мов програмування у світі, особливо в контексті розробки програм для платформи .NET. Вона має простий і зрозумілий синтаксис, що сприяє швидкому розвитку та підтримці коду. C# володіє розширеним набором бібліотек та фреймворків, які спрощують роботу з різними аспектами розробки програмного забезпечення. Зокрема, .NET Framework надає широкий спектр інструментів для роботи з базами даних, мережами, графічними елементами та багато іншого.

Використання C# дозволить нам створювати добре структурований, ефективний та легко збережений код, що відповідає вимогам нашого проекту. Ця мова програмування також має велику кількість документації та підтримку спільноти розробників, що дозволить нам ефективно вирішувати будь-які технічні питання, які можуть виникнути під час розробки.

2.4.3 Використання Windows Forms для розробки інтерфейсу користувача.

Windows Forms є однією з найпоширеніших технологій для створення графічного інтерфейсу користувача у середовищі .NET. Використання Windows Forms дозволить нам створити зручний та привабливий інтерфейс для нашого програмного забезпечення з мінімальними зусиллями. Windows Forms надає великий набір стандартних елементів управління, таких як кнопки, тексти, списки та інші, що дозволяє швидко створювати інтерактивні форми. Крім того, Windows Forms підтримує можливість розробки власних елементів управління та налаштування їх зовнішнього вигляду та поведінки.

Використання Windows Forms дозволить нам ефективно реалізувати функціональність інтерфейсу користувача нашого програмного забезпечення, забезпечуючи при цьому зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Крім того, Windows Forms інтегрується з Visual Studio, що дозволить нам забезпечити зручний та безперервний процес розробки та тестування нашого програмного забезпечення.

2.4.4 Використання бази даних SQL Lite.

SQL Lite було обрано для забезпечення потреб зберігання та організації даних в нашому програмному продукті. SQL Lite – це легка, компактна та вбудована реляційна база даних, яка надає широкі можливості для роботи з даними [4]. Використання SQL Lite в нашому проекті дозволить нам забезпечити ефективне та надійне збереження інформації про користувачів, шаблони листів та інші дані, необхідні для роботи програмного продукту [6].

SQL Lite володіє високою швидкістю роботи та малою величиною файлів баз даних, що робить її ідеальним варіантом для використання в мобільних та вбудованих системах. Крім того, SQL Lite не вимагає окремого сервера баз даних, що полегшує розгортання та управління базою даних в нашому програмному продукті.

Використання SQL Lite дозволить нам забезпечити безпеку та цілісність даних, що є критично важливим для успішного функціонування нашого програмного забезпечення. Крім того, SQL Lite підтримує мову запитів SQL, що дозволяє легко взаємодіяти з базою даних та виконувати різноманітні операції з даними [5].

2.5 Особливості програмної реалізації

2.5.1 Конфігурація поштових сервісів

На рис 2.2 подано фрагмент коду, який реалізує метод, що відповідає за початкову конфігурацію сервісів електронної пошти. Створюються два об’єкти EmailConfiguration для двох різних поштових сервісів: UkrNet та Gmail. Кожен об’єкт містить інформацію про SMTP сервер, порт, ім’я користувача та пароль. Початково вибирається сервіс UkrNet для відправки електронних листів, створюючи об’єкт SmtpEmailService з відповідною конфігурацією.

2.5.2 Вибір поштового сервісу

На рис 2.3 подано фрагмент коду, який реалізує функцію, що дозволяє користувачу перемикатися між двома поштовими сервісами (UkrNet та Gmail) за допомогою радіокнопок. Коли вибирається Gmail, метод SetEmailService змінює конфігурацію на Gmail. Аналогічно, коли вибирається UkrNet, конфігурація змінюється на UkrNet. Метод SetEmailService приймає об’єкт конфігурації та оновлює поточний сервіс електронної пошти відповідно до нової конфігурації.

```
private void InitializeMailServices()
{
    // Ініціалізуємо об'єкти EmailConfiguration для кожної поштової служби
    ukrNetConfiguration = new EmailConfiguration
    {
        From = "nazarmanoilo@ukr.net",
        SmptpServer = "smtp.ukr.net",
        Port = 2525,
        UserName = "nazarmanoilo@ukr.net",
        Password = "nFFjvNhQEIUC0z5T"
    };

    gmailConfiguration = new EmailConfiguration
    {
        From = "alsiel2003@gmail.com",
        SmptpServer = "smtp.gmail.com",
        Port = 465,
        UserName = "alsiel2003@gmail.com",
        Password = "utpd xrcz jusb ppsb"
    };

    emailService = new SmtpEmailService(ukrNetConfiguration);
}
```

Рисунок 2.2 – Фрагмент коду для конфігурації поштових сервісів

```
private void radioButtonGmail_CheckedChanged(object sender, EventArgs e)
{
    if (radioButtonGmail.Checked)
    {
        SetEmailService(gmailConfiguration);
    }
}

1 reference
private void radioButtonUkrNet_CheckedChanged(object sender, EventArgs e)
{
    if (radioButtonUkrNet.Checked)
    {
        SetEmailService(ukrNetConfiguration);
    }
}
2 references
private void SetEmailService(EmailConfiguration configuration)
{
    emailService = new SmtpEmailService(configuration);
}
```

Рисунок 2.3 – Фрагмент коду для вибору поштового сервісу

2.5.3 Додавання вибраної електронної адреси

На рис. 2.4 подано фрагмент коду, який реалізує метод, що додає введену користувачем електронну адресу до списку вибраних адрес для відправки.

```
private void AddSelectedEmail()
{
    // Перевіряємо, чи введена адреса не є порожньою
    if (!string.IsNullOrEmpty(comboBoxEmails1.Text))
    {
        // Перевіряємо, чи адреса вже є у списку вибраних
        if (!selectedEmails.Contains(comboBoxEmails1.Text))
        {
            // Додаємо введену адресу в список вибраних
            selectedEmails.Add(comboBoxEmails1.Text);

            // Оновлюємо вміст ComboBox
            UpdateComboBoxEmails();

            // Оновлюємо список вибраних адрес
            UpdateSelectedEmailsList();
        }
        else
        {
            MessageBox.Show("Ця адреса вже була додана.");
        }

        comboBoxEmails1.Text = "";
    }
}
```

Рисунок 2.4 – Фрагмент коду для додавання електронної адреси

Спочатку перевіряється, чи введена адреса не є порожньою. Потім перевіряється, чи ця адреса вже не була додана раніше. Якщо адреса нова, вона додається до списку вибраних адрес, і відповідні елементи інтерфейсу (ComboBox та ListBox) оновлюються для відображення змін. Якщо адреса вже існує у списку, виводиться повідомлення про це.

2.5.4 Відправка електронних листів

На рис. 2.5. подано фрагмент коду, який реалізує метод, що відправляє електронні листи на вибрані адреси. Спочатку перевіряється, чи є вибрані адреси. Якщо їх немає, виводиться повідомлення про необхідність вибору адрес. Потім для кожної вибраної адреси створюється об’єкт Message з темою та текстом листа. Використовується HTML шаблон для форматування листа. Після цього лист відправляється за допомогою методу Send сервісу

SmtpEmailService. Після успішного відправлення очищуються поля вводу та списки, і виводиться повідомлення про успішну відправку.

```
private void buttonSendEmail_Click(object sender, EventArgs e)
{
    //Перевірка, чи є вибрані адреси електронної пошти
    if (selectedEmails.Count == 0)
    {
        MessageBox.Show("Будь ласка, оберіть адреси електронної пошти перед надсиланням.");
        return;
    }
    try
    {
        foreach (var selectedEmail in selectedEmails)
        {
            Message info = new Message()
            {
                Subject = textBoxSubject.Text,
                Body = textBoxBody.Text,
                To = selectedEmail // Використовуємо кожну адресу зі списку вибраних
            };
            // Отримання HTML-шаблону для листа
            string html = File.ReadAllText("html/index.html");
            html = html.Replace("{$Title}", info.Subject);
            html = html.Replace("{$MyBody}", info.Body);
            info.Body = html;
            // Надсилання листа за допомогою SmtpEmailService з конфігурацією відповідного акаунта
            emailService.Send(info, listBoxFiles, formatsList);
        }
        // Очищення полів вводу та списку вибраних адрес
        textBoxSubject.Text = "";
        textBoxBody.Text = "";
        selectedEmails.Clear(); // Очищуємо список вибраних адрес
        UpdateSelectedEmailsList(); // Оновлюємо відображення вибраних адрес в listBoxSelectedEmails
        listBoxFiles.Items.Clear();
        // Показуємо лейбл про успішне надіслання
        labelSend.Visible = true;
        timerSend.Start();
    }
    catch (AuthenticationException ex)
    {
        textBoxBody.Text =  ("Authentication failed: " + ex.Message);
    }
    catch (SmtpCommandException ex)
    {
        textBoxBody.Text =  ($"SMTP command failed: {ex.Message}");
    }
    catch (Exception ex)
    {
        textBoxBody.Text = ("An error occurred: " + ex.Message);
    }

    UpdateComboBoxEmails();
}
1 reference
```

Рисунок 2.5 – Фрагмент коду для відправки електронних листів

2.5.5 Взаємодія з базою даних

На рис. 2.6 подано фрагмент коду, який реалізує метод, що зберігає дані користувачів з таблиці DataGridView до бази даних. Спочатку всі рядки з DataGridView зчитуються і додаються до списку rows. Потім існуючі записи в базі даних видаляються, і додаються нові записи з списку rows. Після цього зміни зберігаються в базі даних, оновлюється ComboBox з електронними адресами, і виводиться повідомлення про успішне збереження.

```
private void btnSave_Click(object sender, EventArgs e)
{
    var rows = new List<UserEntity>();
    foreach (DataGridViewRow row in DGVUsers.Rows)
    {
        rows.Add(new UserEntity
        {
            //Id = Convert.ToInt32(row.Cells[0].Value),
            FullName = row.Cells[1].Value.ToString(),
            NumberPhone = row.Cells[2].Value.ToString(),
            Email = row.Cells[3].Value.ToString()
        });
    }
    foreach (var entity in context.Users)
    {
        context.Users.Remove(entity);
    }

    context.Users.AddRange(rows);

    context.SaveChanges();
    UpdateCheckBoxEmail();
    labelSaveData.Visible = true;
    timer1.Start();

}
```

Рисунок 2.6 – Фрагмент коду для взаємодії з базою даних

2.5.6 Оновлення елементів інтерфейсу

На рис. 2.6 подано фрагмент коду, який реалізує методи, що відповідають за оновлення елементів інтерфейсу після змін у списках електронних адрес. Метод UpdateComboBoxEmails очищує ComboBox і додає туди адреси, які ще не вибрані. Метод UpdateSelectedEmailsList очищує ListBox і додає туди вибрані електронні адреси для відправки.

```
private void UpdateComboBoxEmails()
{
    // Очищаємо вміст ComboBox
    comboBoxEmails1.Items.Clear();

    // Додаємо адреси, які ще не були вибрані, до ComboBox
    foreach (DataGridViewRow row in DGVUsers.Rows)
    {
        if (!selectedEmails.Contains(row.Cells[3].Value.ToString()))
        {
            comboBoxEmails1.Items.Add(row.Cells[3].Value);
        }
    }
}

// Додамо метод для оновлення списку вибраних адрес електронної пошти
3 references
private void UpdateSelectedEmailsList()
{
    // Очищаємо вміст ListBox
    listBoxSelectedEmails.Items.Clear();

    // Додаємо вибрані адреси до ListBox
    foreach (string email in selectedEmails)
    {
        listBoxSelectedEmails.Items.Add(email);
    }
}
```

Рисунок 2.6 – Фрагмент коду для оновлення елементів інтерфейсу

2.5.7 Таймери для повідомлень

На рис. 2.7 подано фрагмент коду, який реалізує методи, що відповідають за тимчасове відображення повідомлень користувачу. Таймери запускаються після виконання певних дій (збереження даних, додавання користувача, відправка листа) і автоматично приховують відповідні лейбли після певного часу. Це забезпечує кращий користувацький досвід, надаючи зворотний зв’язок про виконані дії.

2.5.8 Завантаження форми та додавання файлів

Метод Menu_Load виконується під час завантаження форми Menu. Він автоматично налаштовує розмір стовпців у DataGridView відповідно до вмісту, заповнює DataGridView даними з бази даних, дозволяє користувачеві

додавати нові рядки та оновлює список доступних електронних адрес (рис.2.8).

```
private void timer1_Tick(object sender, EventArgs e)
{
    labelSaveData.Visible = false;
    labelDelete.Visible = false;
    timer1.Stop();
}

1 reference
private void timerAddUser_Tick(object sender, EventArgs e)
{
    labelErroeAdd.Visible = false;
    labelSuccesAdd.Visible = false;

    timerAddUser.Stop();
}

1 reference
private void timerSend_Tick(object sender, EventArgs e)
{
    labelSend.Visible = false;
}
```

Рисунок 2.7 – Фрагмент коду для використання таймерів для повідомлень

Метод buttonAddFile_Click обробляє подію натискання кнопки для додавання файлу. Він відкриває діалогове вікно для вибору файлу. Якщо файл вибраний, шлях до нього зберігається в змінній currentFile. Після цього викликається метод IsCanAdd, щоб перевірити можливість додавання файлу до списку (рис.2.8).

2.5.8 Завантаження форми та додавання файлів

Метод Menu_Load виконується під час завантаження форми Menu. Він автоматично налаштовує розмір стовпців у DataGridView відповідно до вмісту, заповнює DataGridView даними з бази даних, дозволяє користувачеві

додавати нові рядки та оновлює список доступних електронних адрес (рис.2.8).

```
private void Menu_Load(object sender, EventArgs e)
{
    // Автоматичне налаштування розміру стовпців DataGridView відповідно до вмісту
    DGVUsers.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
    DGVUsers.AutoResizeColumns();
    // Заповнення DataGridView даними з контексту бази даних
    DGVUsers.DataSource = context.Users.ToList();
    // Дозвіл на додавання нових рядків користувачем
    DGVUsers.AllowUserToAddRows = true;
    // Оновлення списку доступних електронних адрес
    UpdateCheckBoxEmail();
}
1 reference
private void buttonAddFile_Click(object sender, EventArgs e)
{
    // Відкриття діалогового вікна для вибору файлу
    OpenFileDialog dlg = new OpenFileDialog();

    // Якщо користувач обрав файл, зберегти шлях до нього
    if (dlg.ShowDialog() == DialogResult.OK)
        currentFile = dlg.FileName;

    // Перевірка можливості додавання файлу до списку
    IsCanAdd();
}
2 references
void IsCanAdd()
{
    // Перевірка, чи обрано файл і формат
    if (string.IsNullOrEmpty(currentFile) || string.IsNullOrEmpty(comboBoxFormats.Text))
    {
        return;
    }

    // Увімкнення кнопки для додавання файлу до списку
    buttonAddToList.Enabled = true;
}
```

Рисунок 2.8 – Фрагмент коду для завантаження форми та додавання файлів

2.5.8 Завантаження форми та додавання файлів

Метод Menu_Load виконується під час завантаження форми Menu. Він автоматично налаштовує розмір стовпців у DataGridView відповідно до вмісту, заповнює DataGridView даними з бази даних, дозволяє користувачеві

додавати нові рядки та оновлює список доступних електронних адрес (рис.2.8).

- перевірка на сумісність з різними збірками та версіями операційної системи;
- перевірка на коректність роботи кожного пакетного файлу та інших зовнішніх модулів;
- перевірка коректної роботи та відображення елементів графічного інтерфейсу інтегрованого середовища для різних варіантів роздільної здатності екрану та масштабування шрифтів;
- перевірка операційної системи на працездатність та збереження функціоналу після роботи програми;
- перевірка ефективності роботи (економія місця, зручність та швидкість взаємодії з користувачем).

Налагодження програмного засобу включає такі етапи:

- використання інструментів для виявлення та виправлення помилок у програмному коді, таких як відладчик Visual Studio;
- аналіз стеку викликів та виведення додаткової інформації для виявлення та усунення помилок;
- тестування програмного засобу в реальних умовах експлуатації для підтвердження його працездатності та відповідності вимогам користувачів.

2.7 Рекомендації з використання та впровадженню програмного засобу

Враховуючи досвід тестування та задачі, які ставились в процесі розробки програмного засобу для надсилання листів електронною поштою, можна сформулювати ряд рекомендацій для його повнофункціонального використання.

Наявність апаратних засобів, що відповідають мінімальним системним вимогам:

- процесор двоядерний з частотою не менше 2 ГГц;
- оперативна пам’ять не менше 4 ГБ;
- місце на диску не менше 500 МБ для встановлення застосунку та зберігання даних.

Встановлена одна із операційних систем:

- Windows 10 x64;
- Windows 8 x64;
- Windows 7 x64.

Файлова система системного розділу NTFS версії не нижче 3.1 (v5.1) [19].

Для коректної роботи програми необхідне стабільне підключення до Інтернету на пристрої користувача. Запуск програми з правами адміністратора системи, що забезпечить доступ до необхідних системних ресурсів та коректну роботу всіх функцій програми. Застосунок потребує встановлення максимально версії .NET Framework 4.8.04084. для коректного функціонування [19].

Даний програмний продукт може бути використаний як звичайними користувачами, які мають права адміністратора на локальних системах, так і адміністраторами систем, мережевими адміністраторами.

ВИСНОВКИ

Під час розробки програмного застосунку для автоматизованої відправки листів за шаблоном на різні поштові сервіси використовувалися потужні і надійні технології. Мова програмування C# разом з платформою .NET Framework дозволили створити функціональні можливості програми, а бібліотеки MailKit та MimeKit забезпечили надійність та безпеку відправки листів.

Для зберігання та керування даними користувачів було використано базу даних SQLite, яка є легко налаштовуваною та ефективною для використання в невеликих проектах. Ця база даних дозволяє зберігати інформацію про користувачів і шаблони листів у компактному форматі, що сприяє оптимізації роботи програми [4].

Отже, завдяки цим технологіям програма забезпечує надійну та ефективну відправку листів за шаблоном на різні поштові сервіси. Користувачі можуть легко керувати комунікацією та зекономити час, використовуючи цей програмний засіб.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bischof B. Windows Form Applications. *The .NET Languages: A Quick Translation Guide*. Berkeley, CA, 2002. P. 193–215. URL: https://doi.org/10.1007/978-1-4302-1136-5_11 (date of access: 07.04.2024).
2. Bisson R. SQL injection. *ITNOW*. 2005. Vol. 47, no. 2. P. 25. URL: https://doi.org/10.1093/itnow/bwi039 (date of access: 07.04.2024).
3. C# Programming Language / A. Hejlsberg et al. Pearson Education, Limited.
4. Caldeira C. P. Teaching SQL. *ACM SIGCSE Bulletin*. 2008. Vol. 40, no. 3. P. 340. URL: https://doi.org/10.1145/1597849.1384382 (date of access: 07.04.2024).
5. Copeland R. Essential SQLAlchemy. Beijing : O'Reilly, 2008. 215 p.
6. Eisenberg A., Melton J. SQL. *ACM SIGMOD Record*. 1999. Vol. 28, no. 1. P. 131–138. URL: https://doi.org/10.1145/309844.310075 (date of access: 07.04.2024).
7. Hejlsberg A. The C# programming language. 2nd ed. Upper Saddle River, NJ : Addison-Wesley, 2006. 704 p.
8. Khoshafian S. Intelligent SQL. *Computer Standards & Interfaces*. 1991. Vol. 13, no. 1-3. P. 169–184. URL: https://doi.org/10.1016/0920-5489(91)90025-u (date of access: 07.04.2024).
9. Базурін В. М. Середовища програмування як засіб навчання учнів основ програмування. *Інформаційні технології і засоби навчання*. 2017. Т. 59, № 3. С. 13–27. URL: https://doi.org/10.33407/itlt.v59i3.1601 (дата звернення: 07.04.2024).
10. Байдачный С. С. NET Framework : секреты создания Windows-приложений. Москва : Солон-Пресс, 2004. 496 р.
11. Програмування. К., 1994. 36 с.
12. Libraries. *API Design for C++*. 2011. P. 391–408. URL: https://doi.org/10.1016/b978-0-12-385003-4.00020-8 (date of access: 07.04.2024).

13. Libraries / J. Reinders et al. *Data Parallel C++*. Berkeley, CA, 2020. P. 471–493. URL: https://doi.org/10.1007/978-1-4842-5574-2_18 (date of access: 07.04.2024).
14. Molluzzo J. C. Modular assembler language programming. *ACM SIGCSE Bulletin*. 1984. Vol. 16, no. 3. P. 17–20. URL: https://doi.org/10.1145/989357.989362 (date of access: 07.04.2024).
15. Professional .NET Framework / K. Hoffman et al. Wrox Press, 2001. 1000 p.
16. Sharp J. Microsoft Visual C# 2013 Step by Step. Microsoft Press, 2013.
17. Sharp J. Microsoft Visual C# Etape par Etape. Dunod, 2002. 656 p.
18. Sharp J. Microsoft Visual C# Step by Step. Microsoft Press, 2018.
19. Team W. .Net Framework. CampusPress, 2002. 830 p.
20. Troelsen A. Programming with Windows Form Controls. *C# and the .NET Platform*. Berkeley, CA, 2001. P. 545–604. URL: https://doi.org/10.1007/978-1-4302-1141-9_10 (date of access: 07.04.2024).

ДОДАТОК А

Технічне завдання

Тематика

Програмний засіб призначений для автоматизованої відправки листів за шаблоном на різні поштові сервіси.

Цільова аудиторія

Користувачі, які регулярно відправляють однотипні листи за шаблоном на різні електронні адреси.

Платформа

Програма має бути доступною для використання на операційних системах Windows.

Основні функції

Ця програма створена для надсилання листів електронною поштою та забезпечення зручності та ефективності у відправленні повідомлень.

Дизайн

Інтерфейс програми має бути зрозумілим і легким для використання. Користувач повинен мати зручний доступ до всіх функцій програми і зрозуміти, як користуватися ними без додаткових пояснень. Вивід інформації та взаємодія з користувачем повинні бути інтуїтивно зрозумілими та забезпечувати ефективне використання програмного засобу.

ДОДАТОК Б

1. Загальні відомості

Рисунок Б.1 – Графічний інтерфейс

2. Функціональне призначення

Програма призначена для автоматизованої відправки листів за певним шаблоном на різні поштові сервіси.

3. Умови застосування програми

Дана програма успішно працює на операційних системах Windows.

4. Опис роботи програми

Користувач може додавати нових користувачів у систему, заповнюючи їхні особисті дані, такі як ім’я, номер телефону та адреса електронної пошти (рис. Б.2).

Після додавання користувачів, застосунок надає можливість вибрати адресу, до якої буде відправлено лист. Користувач може вказати заголовок повідомлення, текст повідомлення та прикріпити файли різних форматів (текстові, зображення, аудіо, відео, застосунки тощо) (рис. Б.3).

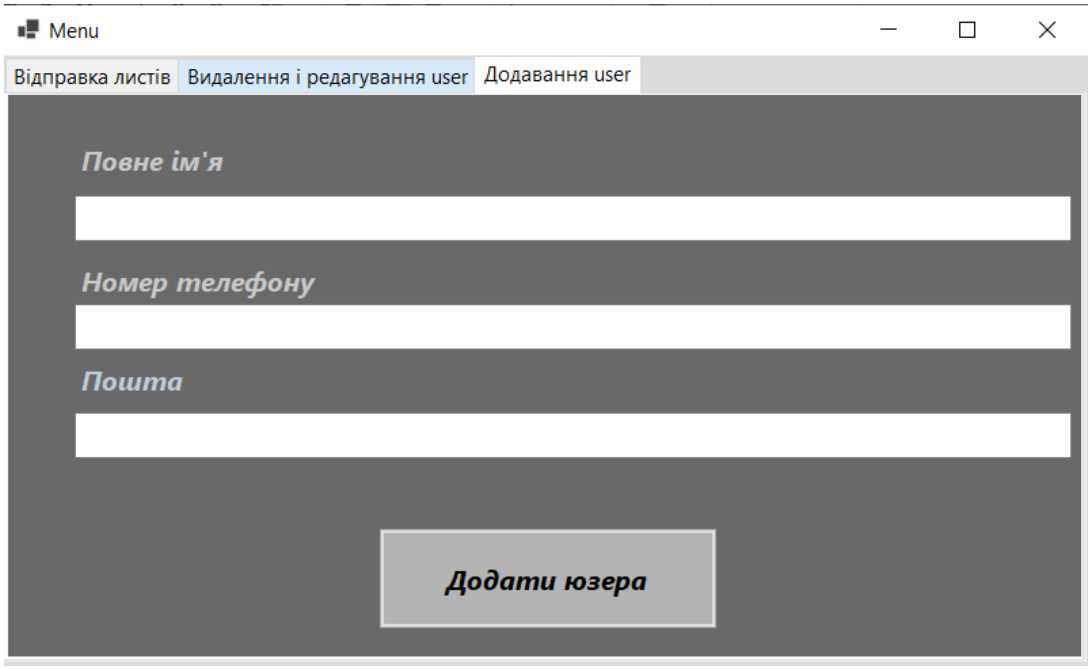

Рисунок Б.2 – Додавання користувачів

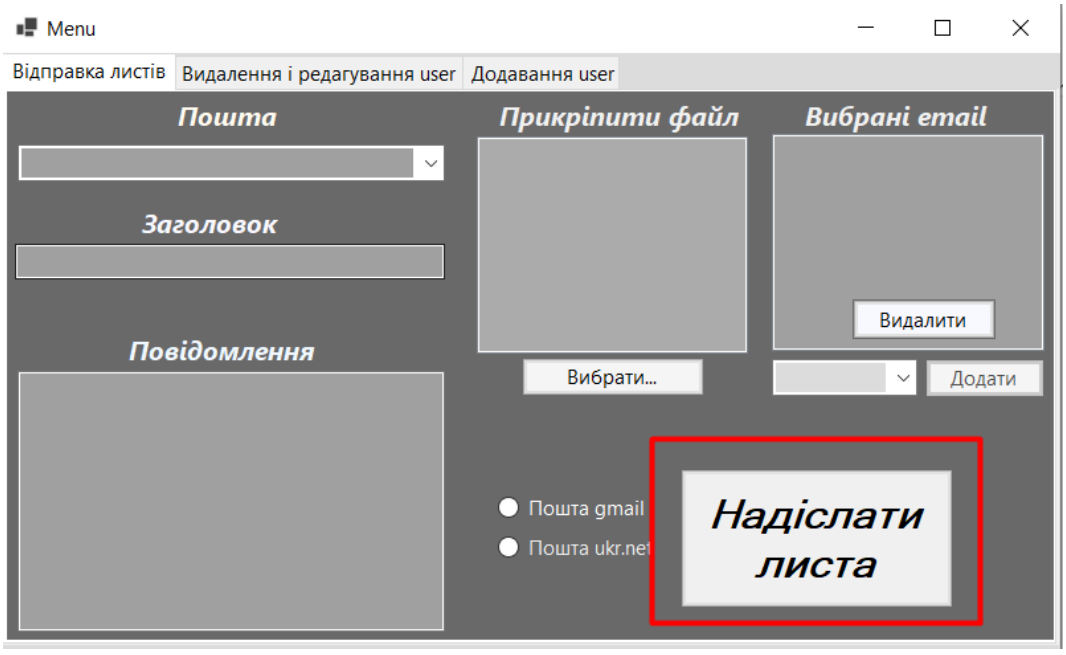

Рисунок Б.3 – Відправлення листа

Користувач також може редагувати та видаляти існуючих користувачів з системи. Для цього доступне відповідне вікно, де можна виконувати редагування особистих даних або видаляти користувачів повністю з системи (рис. Б.4).

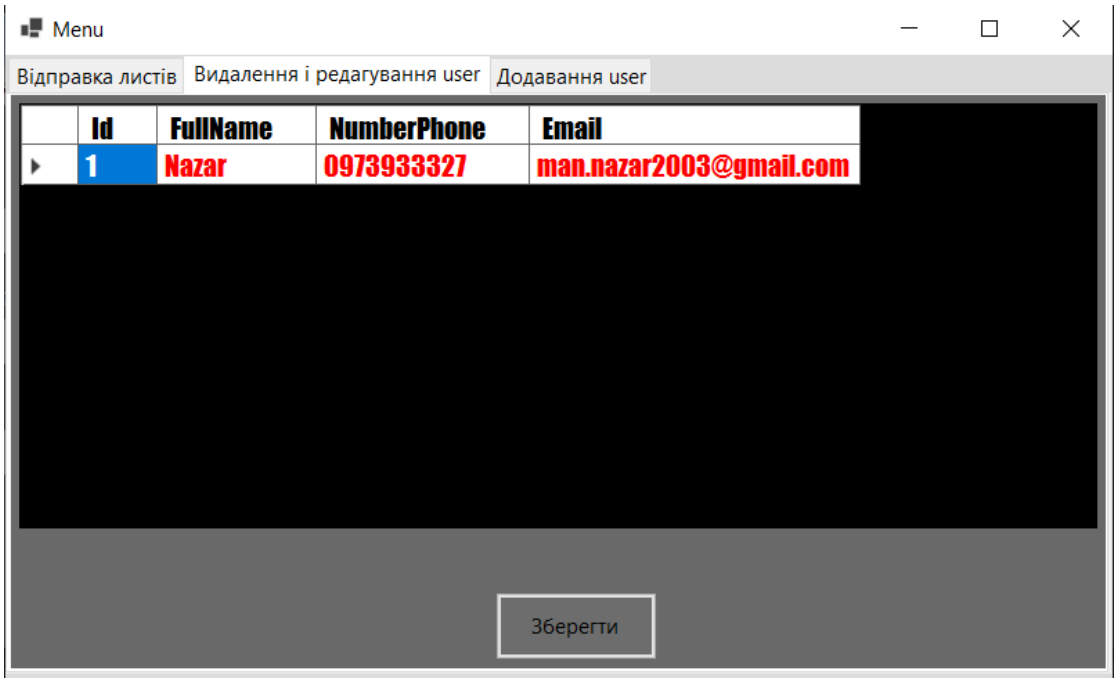

Рисунок Б.4 – Редагування та видалення користувачів

Користувач може надсилати листи з декількох електронних адрес і має можливість вибрати кілька адресатів, яким надішле повідомлення одночасно (рис. Б.5).

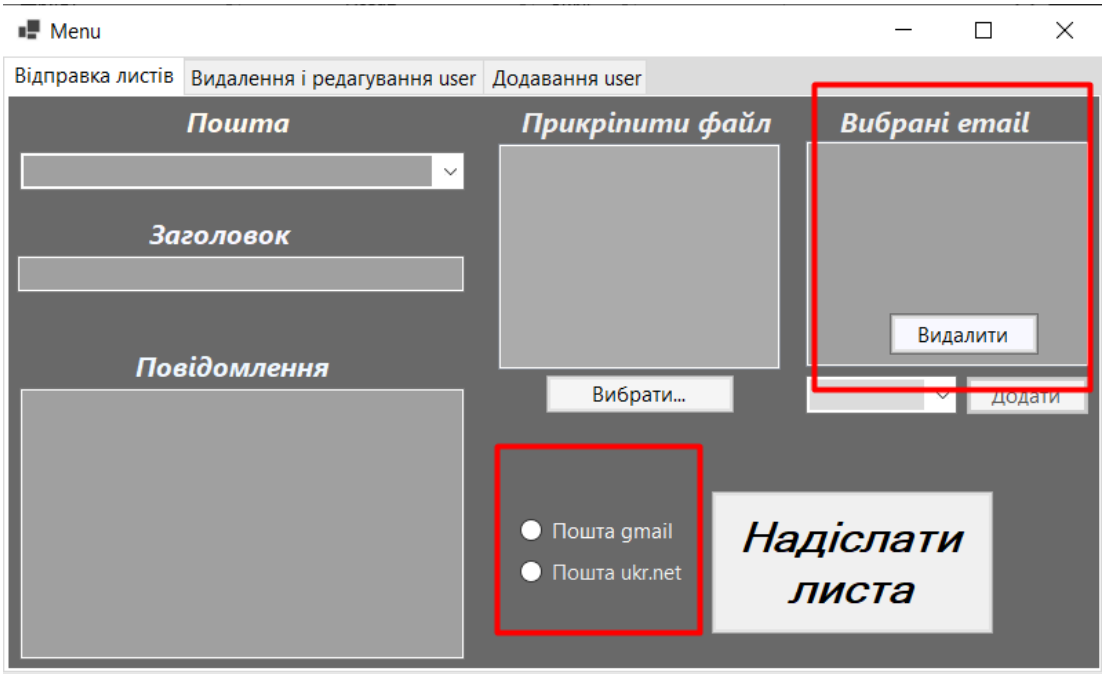

Рисунок Б.5 – Надсилання з кількох адресатів

АНОТАЦІЯ

Манойло Н.Е. Проектування та реалізація програми для автоматизованої відправки листів за шаблоном на різні поштові сервіси. Рукопис

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 122 Комп'ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2024 р.

Кваліфікаційна робота представляє собою детальний огляд процесу дослідження технологій та розробки програмного забезпечення для автоматизованої відправки листів за шаблоном через різні поштові сервіси. В ході дослідження було проведено аналіз наявних рішень та підходів до створення подібних сервісів, а також визначено оптимальні технології для реалізації цієї функціональності.

Однією з ключових складових розробленого продукту є його база даних. Цей компонент відповідає за зберігання та організацію великої кількості даних про листи, що відправляються через систему. Завдяки ефективному керуванню інформацією в базі даних, користувачі можуть швидко та зручно здійснювати відправку листів за шаблоном, не зазнаючи перешкод у роботі.

Програмний продукт вирізняється своїм зручним та інтуїтивно зрозумілим інтерфейсом, що дозволяє користувачам легко та швидко використовувати сервіс для відправки листів. Розробники приділили особливу увагу зручності користування, щоб забезпечити максимально ефективне використання системи.

Система включає компонент, що відповідає за обробку запитів користувачів та забезпечення стабільної роботи програмного забезпечення в цілому. Цей компонент забезпечує швидку та ефективну взаємодію користувачів з системою відправки листів, дозволяючи їм надсилати повідомлення з мінімальними затримками та високою надійністю.

У кваліфікаційній роботі детально проаналізовано різноманітні

технології та методології для створення сервісу відправки листів за шаблоном через різні поштові сервіси. Завдяки використанню відповідних інструментів та уважному аналізу існуючих рішень, було розроблено комплексний та зручний для користувача програмний продукт.

Ключові слова: відправка листів, автоматизація, шаблон, поштові сервіси, тестування, якість.