Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____О.А. СМІРНОВ
“____” _____ 20 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерні науки»
спеціальності 123 «Комп’ютерна інженерія»
_____М.А. Томчак
«____» _____ 20 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Н.М. Якименко
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Томчак М.А. Дослідження та програмна реалізація системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

Метою роботи є дослідження та програмна реалізація системи мережевої розсилки електронної пошти з трирівневим захистом.

Об'єктом дослідження є процес розсилки електронної пошти з трирівневим захистом.

Предметом дослідження є методи розсилки електронної пошти з трирівневим захистом.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на будь якому пристрої або сервері з встановленим JDK 21+, не залежно від операційної системи.

Програму розроблено в середовищі Java.

Ключові слова: комп'ютерна інженерія, мережева розсилка, трирівневий захист

ABSTRACT

Tomchak M.A. Research and software implementation of a network email distribution system with a three-level data protection model. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed that is intended for a network e-mail distribution system with a three-level data protection model.

The purpose of the work is to study and programmatically implement a network email distribution system with three-level protection.

The object of the research is the process of email distribution with three-level protection.

The subject of the study is methods of sending e-mail with three-level protection.

The research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of a network e-mail distribution system with a three-level data protection model.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described

A user-friendly interface has been developed. Instructions for working with the software are provided.

The program can be used on any device or server with JDK 21+ installed, regardless of the operating system.

The program is developed in the Java environment.

Keywords: computer engineering, network distribution, three-level security

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ

TLS (Transport Layer Security) – це криптографічний протокол, який забезпечує захищене шифроване з'єднання між клієнтом і сервером у мережі.

AES (Advanced Encryption Standard) – це симетричний алгоритм шифрування даних, який забезпечує високий рівень захисту та використовується для безпечного зберігання і передачі інформації.

SHA-256 (Secure Hash Algorithm 256-bit) – криптографічна хеш-функція, що генерує 256-бітний унікальний хеш із будь-яких даних.

REST API (Representational State Transfer Application Programming Interface) – це інтерфейс для взаємодії між програмами через HTTP, який використовує стандартні методи (GET, POST, PUT, DELETE) для роботи з ресурсами.

API (Application Programming Interface) – набір інструментів і правил для взаємодії програм.

SMTP (Simple Mail Transfer Protocol) – протокол для відправки електронної пошти.

CRM (Customer Relationship Management) – система для управління взаємовідносинами з клієнтами.

Go – мова програмування від Google, оптимізована для швидких і масштабованих сервісів.

MIT – ліцензія на програмне забезпечення з відкритим кодом, дозволяє вільне використання і модифікацію.

PostgreSQL – об'єктно-реляційна система управління базами даних.

HTML (HyperText Markup Language) – мова розмітки для створення веб-сторінок.

MySQL – популярна реляційна система управління базами даних.

DNS (Domain Name System) – система для перетворення доменних імен у IP-адреси.

SPF (Sender Policy Framework) – механізм перевірки, чи має сервер право відправляти пошту від імені домену.

DKIM (DomainKeys Identified Mail) – технологія підпису електронної пошти для перевірки її достовірності.

DMARC (Domain-based Message Authentication, Reporting & Conformance) – політика захисту пошти, що поєднує SPF і DKIM.

SES (Simple Email Service) – сервіс від AWS для відправки та отримання електронної пошти.

AWS (Amazon Web Services) – хмарна платформа з набором сервісів для обробки, зберігання та управління даними.

Java – об’єктно-орієнтована мова програмування.

Maven – інструмент для управління залежностями і збірки Java-проектів.

Spring Boot – фреймворк для швидкої розробки Java-додатків.

H2 database – легка вбудована реляційна база даних на Java.

Gmail SMTP server – сервер Gmail для відправки пошти через SMTP.

JVM (Java Virtual Machine) – середовище виконання Java-програм.

username – ім’я користувача для входу в систему.

email – електронна пошта користувача.

JSON (JavaScript Object Notation) – формат обміну даними у вигляді тексту.

HTTPS (HyperText Transfer Protocol Secure) – захищена версія HTTP з шифруванням TLS.

Docker – платформа для контейнеризації і запуску додатків у ізольованих середовищах.

Gmail – сервіс електронної пошти від Google.

ВСТУП

Актуальність теми. У сучасних умовах електронна пошта досі лишається одним із найважливіших інструментів обміну інформації. Вона використовується не лише для особистого спілкування, а й у сфері бізнесу, освіти, державного управління, а також у критично важливих галузях, пов’язаних із національною безпекою. Саме тому питання захисту електронного листування є актуальним.

Одним з ефективних напрямів підвищення безпеки інформаційних систем є впровадження багаторівневих моделей захисту, що поєднують кілька незалежних механізмів шифрування та перевірки правдивості даних. У межах даної роботи розглядається реалізація трирівневої системи захисту, яка включає:

– використання TLS (Transport Layer Security) для захищеної передачі даних через HTTP(S) у REST API;

– застосування AES (Advanced Encryption Standard) для шифрування бази даних користувачів;

– реалізацію SHA-256 для контролю цілісності логів.

Таким чином, розробка системи мережевої розсилки електронної пошти з трирівневим захистом є актуальною задачею, що відповідає сучасним тенденціям розвитку інформаційної безпеки та вимогам до надійності комунікаційних сервісів.

Мета й завдання дослідження. *Метою роботи* є дослідження та програмна реалізація системи мережевої розсилки електронної пошти з трирівневим захистом.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем мережевої розсилки електронної пошти.

– Дослідження системи мережевої розсилки електронної пошти з трирівневим захистом.

– Програмна реалізація системи мережевої розсилки електронної пошти з трирівневим захистом.

Об'єктом дослідження є процес розсилки електронної пошти з трирівневим захистом.

Предметом дослідження є методи розсилки електронної пошти з трирівневим захистом.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

– Розроблено вітчизняний продукт мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність полягає у створенні прототипу системи мережевої розсилки електронної пошти з трирівневим захистом. Розроблювана система забезпечуватиме захищену передачу даних та безпечне зберігання службової інформації й логів, що підвищує рівень надійності та безпеки електронного листування в мережевому середовищі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Обсяги електронної комунікації невпинно зростають, і потреба в захисті листування стає особливо актуальною. З розвитком підприємств та розширенням мережі клієнтів і партнерів необхідність убезпечити поштові скриньки й зберігання даних у них зростає щодня.

Одним із основних векторів атак залишається викрадення облікових даних, що дозволяє зловмисникам отримати повний контроль над поштовою скринькою. Це дає змогу витягти списки контактів, фінансові документи та інші чутливі дані або використовувати скриньку для шахрайства та розповсюдження шкідливих повідомлень. Навіть без повного контролю, витік окремих листів може призвести до шахрайств і викрадення важливої інформації.

Ще однією загрозою є маніпуляція журналами подій. Зловмисник з доступом до облікового запису може підробляти або видаляти записи логів, приховуючи сліди перебування. Цілісність логів є критичною для оперативного реагування на інциденти та розслідування порушень безпеки.

Розроблена система мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту покликана забезпечити централізоване, безпечне та ефективне керування процесами надсилання повідомлень. Система автоматизує розсилку пошти, контролює їхню доставку, зберігає цілісність даних і знижує ризик втрати або підроблення інформації під час передавання.

Вона впроваджує SMTP через TLS для передачі повідомлень, AES для захисту бази даних користувачів та SHA-256 для цілісності логів, що формує трирівневу модель безпеки. Система забезпечує централізоване та захищене логування, збір статистики та моніторинг стану розсилок. Завдяки REST API вона

легко інтегрується з іншими сервісами забезпечуючи високу адаптивність і централізоване керування.

Масштабованість системи дозволяє використовувати її як на локальному рівні підприємства, так і в повноцінній інфраструктурі, здатній обробляти тисячі листів. Поєднання безпеки, стабільності та ефективності дозволяє підвищити якість сервісу, своєчасно реагувати на проблеми доставлення та оптимізувати навантаження на сервер.

Таким чином, призначення системи полягає у створенні надійного, гнучкого та безпечного програмного продукту, який автоматизує процес надсилання електронної пошти, інтегрується з іншими системами та забезпечує захист відповідно до сучасних вимог кібербезпеки.

1.2 Область застосування

Для багатьох підприємств ручне формування й розсилання великої кількості листів стає неефективним, призводить до втрат часу й зниження якості комунікації. Саме тому зростає попит на спеціалізовані програмні рішення, які дозволяють централізовано керувати електронною поштою, контролювати її доставку, вести статистику розсилок і забезпечувати надійний захист інформації.

Сфера застосування системи розсилки охоплює широкий спектр галузей. Завдяки добре спланованій архітектурі й REST API система легко інтегрується у вже наявні інформаційні рішення підприємств: CRM, ERP, сервіси підтримки клієнтів, платіжні системи, вебпортали та інші корпоративні платформи. Така інтеграція дозволяє автоматизувати не лише сам процес відправлення листів, а й збір аналітики. Безпека є не менш важливою частиною системи, зважаючи на кількість атак, спрямованих саме на електронну пошту. Саме тому створення системи розсилки, що має вбудовані механізми захисту TLS, AES, SHA-256, є зрозумілим і логічним рішенням.

У корпоративних структурах така система може відігравати роль елемента внутрішньої інфраструктури безпеки. Наприклад, використання централізованого серверного рішення дозволяє уникнути ситуацій, коли співробітники надсилають листи через особисті поштові акаунти, що може призвести до витоку даних або підробки документів. Система може бути використана також у сфері електронної комерції для автоматизованого надсилання підтверджень замовлень, сповіщень про оплату, оновлень статусу доставки або акційних пропозицій. Для таких застосувань особливо важливими є стабільність, масштабованість і контроль за доставкою повідомлень. В освітніх установах система може забезпечити ефективну взаємодію між адміністрацією, викладачами та студентами, зменшивши навантаження на персонал. Наприклад, автоматизовані повідомлення можуть інформувати про зміни в розкладі, результати сесій, події чи новини університету. Такі розсилки можна налаштовувати централізовано та адаптувати їхній зміст для різних груп користувачів.

У державному секторі система може використовуватися для організації розсилок у межах програм електронного самоврядування. Це дасть можливість швидко та зручно розсилати повідомлення громадянам про рішення, довідки, сповіщення, пільги або електронні квитанції. У час, коли багато послуг переходять у цифрову площину, такі системи є необхідними для забезпечення швидкої, захищеної та достовірної комунікації між державними органами й населенням. Розроблена система може стати складовою частиною інформаційно-аналітичних платформ. Завдяки збору статистики, аналізу логів і моніторингу доставок вона дозволяє не лише здійснювати комунікацію, а й накопичувати дані для подальшої аналітики.

Таким чином, область застосування розробленої системи охоплює різноманітні сфери діяльності. Її використання дозволяє оптимізувати процеси обміну електронними повідомленнями, підвищити рівень інформаційної безпеки та контроль над доставкою й ефективність управління даними.

2. ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень з профілю теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

У межах теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти «Дослідження та програмна реалізація системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних» в цьому розділі буде розглянуто сучасні реалізації та архітектурні підходи існуючих систем мережевої розсилки електронної пошти.

Listmonk

Listmonk – це система управління масовими розсилками електронної пошти, що поєднує високу продуктивність, масштабованість і розширювані можливості завдяки архітектурі на базі Go та PostgreSQL. Вона належить до класичних серверних рішень, які розгортаються локально на стороні користувача, забезпечуючи повний контроль над інфраструктурою, безпекою та даними. Програмне забезпечення має відкритий вихідний код і поширюється за ліцензією MIT, що дозволяє його вільно модифікувати, адаптувати та інтегрувати у власні системи.

Listmonk призначена для централізованого управління підписниками, кампаніями, шаблонами листів, а також забезпечення аналітики й безпечного доставляння повідомлень. Система орієнтована на корпоративне та організаційне використання, де важливими є стабільність, прозорість і незалежність від сторонніх постачальників сервісів.

Встановлення та запуск Listmonk здійснюється через контейнеризацію Docker або локальну інсталяцію на Linux, Windows чи macOS. Під час налаштування користувач задає параметри підключення до бази даних

PostgreSQL, SMTP-сервера та адміністративного вебінтерфейсу. Завдяки мінімальним залежностям і невеликим системним вимогам інсталяція триває лише кілька хвилин і не потребує складного адміністрування.

Архітектура Listmonk базується на чіткому поділі компонентів за функціональністю. Бекенд, реалізований мовою Go, відповідає за логіку, обробку запитів, планування розсилок і керування чергами. База даних PostgreSQL зберігає інформацію про користувачів, кампанії, шаблони, історію відправлень і статистику. Вебінтерфейс забезпечує адміністрування системи, створення кампаній і моніторинг результатів. REST API дозволяє інтегрувати систему з іншими сервісами, такими як CRM, аналітичні платформи чи CMS.

Модуль керування підписниками підтримує створення сегментів, груп, фільтрів та імпорт або експорт контактів у форматі API чи через API. Користувачі можуть самостійно керувати своїми підписками за допомогою спеціальних посилань opt-in / opt-out, що забезпечує відповідність політикам конфіденційності. Модуль кампаній дає змогу створювати як одноразові, так і повторювані розсилки, використовувати шаблони листів у форматах HTML і Markdown, а також планувати їх за розкладом. Для відправлення повідомлень система може підключатися до одного або кількох зовнішніх SMTP-серверів, підтримуючи шифрування TLS для захищеної передачі даних. Асинхронна модель обробки завдань, реалізована через внутрішню чергу повідомлень, дозволяє надсилати десятки тисяч листів без блокування основного сервісу.

Безпека в Listmonk реалізована на кількох рівнях. На транспортному рівні всі з'єднання вебінтерфейс, API, SMTP захищені шифруванням TLS, що запобігає перехопленню даних. Для доступу до адміністративного інтерфейсу та API застосовується токен-авторизація, що дозволяє контролювати права користувачів. Крім того, система веде журнали активності, які фіксують дії адміністраторів і користувачів, що забезпечує можливість моніторингу та аудиту змін.

Listmonk має розвинену систему аналітики: вона відстежує статуси доставлення, відкриття листів, переходи за посиланнями, показники відписок і

відмов. Усі статистичні дані зберігаються локально, що дозволяє аналізувати ефективність кампаній без залучення сторонніх сервісів. Звіти формуються у режимі реального часу, а отримані метрики можуть використовуватися для сегментації підписників або повторних кампаній.

Завдяки багатопотоковості Go і можливості розгортання у кластерному середовищі Listmonk здатна обробляти великі обсяги даних. У конфігурації можна визначати кількість одночасних SMTP-з'єднань, обмеження швидкості відправлення rate limit і правила черги. Система оптимізована для масштабування тому може використовуватися як для невеликих локальних розсилок, так і для великих комерційних кампаній.

Інтерфейс користувача зручний та інтуїтивно зрозумілий. Він надає можливість створювати й редагувати кампанії через візуальний редактор, налаштовувати сегменти аудиторії, керувати шаблонами листів і відстежувати статистику. Панель керування підтримує багатомовність і може бути адаптована до корпоративних стандартів. Завдяки відкритому коду Listmonk легко розширюється через плагіни чи власні модифікації. Також можливо підключати зовнішні аналітичні інструменти, інтегрувати системи аутентифікації LDAP, OAuth2 або створювати кастомні звіти.

Розглядаючи всі плюси системи, варто також відзначити основні мінуси. Система не має розвиненої повноцінної безпеки. Всі дані зберігаються у відкритому вигляді, без шифрування бази користувачів, а також відсутній захист від підроблення журналів активності. Механізм контролю доступу є мінімальним, що може бути недостатньо для складних або великих систем. Крім того, для адміністрування та резервного копіювання потрібні додаткові технічні знання, оскільки система не передбачає автоматичного захисту даних чи перевірки їхньої цілісності.

Postal

Postal – це система управління масовими розсилками електронної пошти, побудована на фреймворку Ruby on Rails і призначена для самостійного

розгортання на власних серверах. Вона належить до класичних серверних рішень, що дозволяє організаціям мати повний контроль над інфраструктурою, даними користувачів та безпекою. Postal підходить для корпоративного використання та великих команд, яким важливо керувати масовими розсилками без залучення сторонніх сервісів.

Postal призначена для централізованого керування масовою розсилкою електронної пошти, включаючи адміністрування доменів відправників, обробку вхідних і вихідних повідомлень, контроль репутації серверів і моніторинг доставляння. Система дозволяє відстежувати всі етапи обробки листів, включаючи доставлення, відкриття, кліки та bounce-повідомлення.

Встановлення Postal здійснюється на серверах Linux з використанням PostgreSQL або MySQL для збереження даних та Redis для черг повідомлень. Налаштування включає підключення SMTP-серверів, конфігурацію DNS для підтримки SPF, DKIM і DMARC, а також створення користувачів і ролей. Postal оптимізована для обробки великого обсягу листів, що робить її придатною для підприємств з високим навантаженням.

Архітектура Postal передбачає три основні компоненти: бекенд на Ruby on Rails, базу даних для зберігання інформації про кампанії, користувачів, домени та статистику, а також вебінтерфейс для адміністрування та моніторингу. REST API дозволяє інтегрувати Postal із зовнішніми системами та автоматизувати управління розсилками.

Модуль управління доменами дає змогу підключати власні домени відправників, налаштовувати DNS-записи для підвищення репутації та забезпечення високого процента доставлення листів. Модуль кампаній дозволяє створювати одноразові та повторювані розсилки, працювати з шаблонами листів у форматі HTML, планувати відправлення та контролювати швидкість надсилання rate limiting. Postal підтримує асинхронну обробку завдань через черги, що дозволяє ефективно розподіляти ресурси сервера та уникати блокування процесів.

Безпека в Postal реалізована через шифрування TLS для всіх з'єднань, контроль доступу на основі ролей, а також ведення журналу дій користувачів і адміністраторів. Система підтримує механізми автентифікації користувачів та захист від несанкціонованого доступу до API й вебінтерфейсу.

Postal включає детальну аналітику по розсилках: відстежується статус доставлення, відкриття листів, переходи за посиланнями, bounce-повідомлення та відписки. Аналітичні дані зберігаються на власних серверах і доступні для генерації звітів у режимі реального часу. Це дозволяє оцінювати ефективність кампаній, сегментувати аудиторії та оптимізувати стратегію розсилок.

Інтерфейс користувача Postal забезпечує інтуїтивну роботу з доменами, кампаніями та підписниками. Через вебпанель можна створювати та редагувати розсилки, контролювати статуси повідомлень і переглядати детальну статистику по всіх операціях. Завдяки відкритому коду Postal легко модифікується, підтримує інтеграції з додатковими інструментами для аналітики, кастомізацію шаблонів та автоматизацію процесів.

Основна увага у Postal приділена продуктивності та стабільності роботи, а не комплексній безпеці. Всі дані користувачів зберігаються у відкритому вигляді оскільки база даних користувачів не шифруються, що створює ризик несанкціонованого доступу. Система не передбачає механізмів перевірки цілісності логів. Крім того, налаштування та інсталяція платформи потребують досвіду системного адміністратора, оскільки потрібно правильно конфігурувати DNS-записи, Redis, PostgreSQL та сервери SMTP. Підсумовуючи, у Postal присутній лише транспортний рівень безпеки TLS, тоді як захист на рівні зберігання даних та контроль логів відсутній.

Amazon Simple Email Service

Amazon Simple Email Service (SES) – це хмарна інфраструктура для надсилання електронної пошти, що входить до складу платформи Amazon Web Services (AWS). На відміну від класичних серверних систем, SES не надає вебінтерфейс для керування кампаніями, а працює як API-рівень для розробників

та інтеграторів, що дозволяє будувати власні сервіси розсилок або інтегрувати електронну пошту у додатки, CRM, маркетингові платформи та інші системи. Amazon SES є платною послугою, оплата здійснюється залежно від кількості надісланих листів і додаткових функцій, таких як перевірка доменів та зберігання вхідних повідомлень.

Amazon SES призначена для забезпечення надійного та масштабного доставлення електронної пошти для транзакційних або маркетингових повідомлень. Система орієнтована на компанії, розробників і проєкти, які хочуть інтегрувати масові розсилки у власні сервіси без необхідності самостійно підтримувати інфраструктуру SMTP.

SES працює через API або SMTP-інтерфейс, що дозволяє програмно надсилати листи, керувати чергами, отримувати статуси доставляння, обробляти відмови bounces та відписки complaints. Розробники можуть автоматизувати надсилання транзакційних повідомлень, формувати персоналізовані шаблони та керувати великою кількістю підписників через власні сервіси.

Безпека та надійність у SES реалізовані на рівні AWS. Передача даних підтримує шифрування TLS, а авторизація здійснюється через AWS Identity and Access Management (IAM), що дозволяє точно налаштовувати права доступу до ресурсів та API. Сервіс автоматично обробляє bounce-повідомлення та скарги користувачів, захищаючи репутацію домену та IP-адреси від блокувань.

Amazon SES підтримує горизонтальне масштабування: користувачі можуть надсилати мільйони листів на день без зміни налаштувань серверів. Система інтегрується з іншими сервісами AWS, такими як Amazon CloudWatch для моніторингу статистики, S3 для зберігання вхідних або вихідних повідомлень, а також Lambda для автоматичної обробки подій, що виникають під час розсилок.

Хоч SES не надає графічного інтерфейсу для кампаній, він забезпечує повну прозорість і контроль для розробників: усі події надсилання, доставляння та відмов логуються, а статистика доступна у форматах, які легко інтегруються у власні аналітичні системи або додатки.

Разом з тим, Amazon SES має низку обмежень. Користувач не має прямого контролю над зберіганням даних, оскільки вся інформація розміщується в хмарній інфраструктурі AWS. Сервіс не забезпечує прикладного шифрування даних AES або захисту логів на рівні користувача, а безпека повністю залежить від політик Amazon і механізмів AWS. Крім того, SES є платним і не має власного вебінтерфейсу.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації системи мережевої розсилки електронної пошти через REST API обраний такий стек технологій: Java, Maven, Spring Boot, H2 database та Gmail SMTP server. Такий набір засобів забезпечує стабільну роботу, високу продуктивність, гнучкість налаштування й тирівневу модель безпеки.

Мова програмування Java

Java – це стабільна, об’єктно-орієнтована мова програмування, яка має довгострокову підтримку LTS, що гарантує регулярні оновлення безпеки та сумісність з чинними бібліотеками та фреймворками. Це критично важливо для серверного застосунку, який повинен працювати без збоїв протягом тривалого часу. Довгострокова підтримка означає не лише регулярні оновлення безпеки, а й стабільність роботи з інструментами, що мінімізує ризик втрати даних або некоректної обробки запитів.

Віртуальна машина Java (JVM) забезпечує високу продуктивність навіть при великому навантаженні, що особливо важливо для багатопотокових задач. У системі передбачено одночасну обробку численних REST API-запитів та регулярне виконання cron jobs для автоматичного відправлення листів. Використання Java гарантує, що ці процеси працюють ефективно та стабільно без зниження продуктивності.

Розвинена екосистема бібліотек і фреймворків дозволяє швидко інтегрувати сторонні модулі для роботи з базами даних, відправлення пошти, планування завдань і валідації даних. Крім того, Java має велику спільноту розробників і багату документацію, що спрощує розв'язання технічних питань і зменшує ризик помилок під час розробки.

Статична типізація дозволяє виявляти помилки ще на етапі компіляції, роблячи систему більш надійною та захищеною від неправильного використання API. Це особливо важливо для системи, яка обробляє дані користувачів: username, email, cron expression та інші поля. Типи даних, перевірка null-значень і строгі правила компіляції допомагають запобігти некоректним запитам, які могли б порушити роботу сервісу або пошкодити дані.

JVM забезпечує стабільну роботу під час обробки великої кількості одночасних запитів. Це критично для системи, яка має одночасно обробляти REST API-запити, виконувати планові завдання для автоматичної розсилки листів і вести логування в базі даних. Java підтримує багатопотоковість «з коробки» і має широкий набір інструментів для керування потоками та синхронізацією.

Якщо в майбутньому кількість користувачів зросте з сотень до тисяч або мільйонів, архітектура на Java зможе обробляти більший обсяг запитів без зміни основного коду. Java підтримує масштабування через кластеризацію, розподілене кешування та інтеграцію із зовнішніми чергами повідомлень. Це дозволяє ефективно планувати автоматичне відправлення листів у великих системах, де одночасно працюють тисячі cron jobs.

Java підтримує сучасні стандарти REST API та JSON. Це забезпечує коректну роботу клієнт-серверного обміну даними, сумісність із вебдодатками та мобільними клієнтами. Також Java пропонує стандартизовані механізми валідації та централізованої обробки помилок. Це відповідає всім вимогам завдання: сортування, пошук, статистика і логування.

Java має потужні бібліотеки для безпеки: шифрування AES, хешування SHA-256 і TLS для захищених HTTPS-з'єднань. Це дозволяє реалізувати трирівневу модель захисту даних, що відповідає темі проєкту. Таке поєднання технологій забезпечує безпечну систему без потреби у додаткових зовнішніх рішеннях.

Для керування залежностями та збірки використовується система Maven. Вона забезпечує стандартизований життєвий цикл розробки, спрощує інтеграцію в CI/CD-процеси та дозволяє автоматизувати збірку, тестування і розгортання застосунку.

Maven

Maven – це один із найпопулярніших інструментів керування проєктами та збірки у світі Java. Він створений для автоматизації процесів компіляції, тестування, пакування та розгортання застосунку. Для проєкту системи мережевої розсилки електронної пошти через REST API вибір Maven є логічним, адже він забезпечує стабільність, простоту налаштування та сумісність з іншими інструментами екосистеми Java.

Maven дозволяє повністю автоматизувати процес створення програмного продукту. Замість ручної компіляції коду, копіювання залежностей та архівації, розробник виконує команду mvn clean install в терміналі або може згенерувати проєкт через сайт start.spring.io, де при обранні Maven створюється файл pom.xml. Це мінімізує ризик помилок і значно прискорює розробку.

Однією з головних переваг Maven є потужна система керування залежностями. Завдяки файлу pom.xml Maven автоматично підключає всі необхідні залежності з центрального репозиторію, забезпечуючи однаковий результат збірки у будь-якому середовищі. Крім того, підтримка transitive dependencies дає змогу автоматично підключати бібліотеки, від яких залежать інші модулі, усуваючи конфлікти версій і зменшуючи кількість ручної роботи.

Maven застосовує стандартизовану структуру каталогів, що робить проєкт зрозумілим і впорядкованим для будь-якого розробника. Такий підхід спрощує

навігацію та підтримку коду. У контексті системи мережевої розсилки REST API це особливо важливо, оскільки модулі логування, безпеки та обробки запитів мають бути чітко відокремлені.

Maven використовує офіційні центральні репозиторії, які проходять перевірку безпеки. Це гарантує, що всі підключені бібліотеки перевірені, стабільні й не містять шкідливого коду. Для системи, яка обробляє персональні дані користувачів і взаємодіє з поштовими сервісами, це є обов'язковою вимогою.

Завдяки цьому Maven виступає як надійна основа для підтримки довготривалих проєктів, що потребують стабільності й сумісності між поколіннями Java.

Spring Boot

Spring Boot – це сучасний фреймворк для розробки серверних застосунків на Java, який значно спрощує створення, налаштування та запуск вебсервісів. Його головна перевага полягає у можливості швидко розробити повноцінний REST API без зайвої конфігурації та ручного керування залежностями. Для системи мережевої розсилки електронної пошти через REST API використання Spring Boot є цілком обґрунтованим, адже він поєднує стабільність, модульність, безпеку та масштабованість.

У Spring Boot передбачено вбудований вебсервер. На відміну від класичного підходу, коли застосунок потрібно розгортати в Tomcat або Jetty вручну, Spring Boot уже містить їх у собі. Такий підхід не лише спрощує розгортання, а й забезпечує кращу портативність, що дає змогу розмістити застосунок на будь-якому сервері, у контейнері Docker або навіть запустити локально без додаткових залежностей.

Для системи мережевої розсилки електронної пошти Spring Boot надає всі необхідні інструменти для реалізації REST API. За допомогою анотацій можна легко створювати маршрути для керування розсилками, користувачами, чергами повідомлень і логами. Spring Boot підтримує автоматичне перетворення об'єктів

Java у формат JSON або XML через бібліотеку Jackson, що спрощує обмін даними з клієнтською частиною або зовнішніми сервісами.

Важливим компонентом будь-якого серверного застосунку є безпека. Spring Boot має потужний модуль Spring Security, який інтегрується практично без додаткових зусиль. У межах трирівневої моделі захисту даних, де використовується TLS для захисту транспортного рівня, AES для шифрування бази даних користувачів і SHA-256 для хешування логів, Spring Security відіграє роль координатора доступу до API. Таким чином, система мережевої розсилки отримує надійний захист як на рівні трафіку, так і на рівні збереження даних.

Для системи мережевої розсилки важливою є також автоматизація завдань. Spring Boot надає модуль Spring Scheduler, який дозволяє запускати завдання за розкладом. За допомогою простої анотації можна створити механізм автоматичного відправлення електронної пошти, перевірки стану черг або створення резервних копій. Такий підхід робить систему гнучкою, самостійною і стійкою до збоїв.

Spring Boot інтегрує сучасну систему логування через SLF4J, Logback або Log4j2. Це дозволяє вести детальний облік усіх подій. Використання алгоритму SHA-256 для хешування записів журналу гарантує, що дані логів не можуть бути змінені без виявлення, навіть у разі спроби стороннього втручання можна перевірити цілісність усіх подій.

Якщо навантаження на систему зростає, можна розгорнути кілька копій застосунку в кластері або мігрувати з вбудованої H2 бази на PostgreSQL, MySQL чи інше промислове рішення. Завдяки підтримці контейнеризації та хмарних платформ Docker, Kubernetes, Google Cloud, AWS застосунок можна легко адаптувати до будь-якого середовища.

Завдяки поєднанню цих переваг Spring Boot є надійним фундаментом для створення серверної частини системи мережевої розсилки електронної пошти. Він забезпечує високу швидкість розробки, стабільність роботи та комплексну модель безпеки.

H2 Database

H2 Database – це сучасна реляційна система керування базами даних, створена на Java, яка вирізняється легкістю використання, високою продуктивністю та гнучкою інтеграцією з іншими компонентами застосунку. Її головна перевага полягає в тому, що вона може працювати у вбудованому режимі без необхідності окремого серверного середовища, зберігаючи дані локально або в оперативній пам'яті. Для системи мережевої розсилки електронної пошти через REST API використання H2 Database є цілком виправданим, адже вона поєднує швидкість, зручність тестування, сумісність зі Spring Boot та повну підтримку стандарту SQL.

H2 Database працює без проміжних шарів, безпосередньо всередині JVM, тому час доступу до даних мінімальний, що забезпечує високу ефективність при роботі з REST API. Це особливо важливо для системи мережевої розсилки, де швидкість обробки запитів і відповідей безпосередньо впливає на продуктивність. Завдяки цьому можна швидко перевіряти працездатність логіки без ризику затримок, характерних для важчих систем.

H2 повністю підтримує стандарт SQL-92, що дозволяє описувати структуру даних у вигляді Java-класів, а таблиці створюються автоматично під час запуску застосунку. Такий підхід спрощує розробку, робить код чистішим і зменшує кількість потенційних помилок. Якщо в майбутньому виникне потреба перейти на більш потужну базу даних, наприклад PostgreSQL або MySQL, цей процес буде максимально простим, оскільки H2 сумісна з основними діалектами SQL.

H2 підтримує всі базові механізми безпеки: автентифікацію користувача, шифрування з'єднань, а також збереження зашифрованих баз. При потребі база може бути захищена алгоритмом AES, а доступ до неї дозволено лише через захищений пароль. У поєднанні з TLS на рівні REST API та SHA-256 для журналів система отримує багаторівневий захист даних.

Отже, вибір H2 Database є слушним рішенням, оскільки вона забезпечує швидку розробку, просте налагодження, зручне тестування та належний рівень

безпеки. У поєднанні зі Spring Boot і Java H2 створює надійну основу для побудови сучасної системи мережевої розсилки електронної пошти, яку можна легко масштабувати та адаптувати до промислових умов.

Gmail SMTP Server

Gmail SMTP Server – це надійний поштовий сервіс від компанії Google, який забезпечує стабільну, безпечну та масштабовану платформу для відправлення електронної пошти через стандартний протокол SMTP (Simple Mail Transfer Protocol). Його головна перевага полягає у високому рівні довіри поштових систем до серверів Google, завдяки чому листи доставляються користувачам практично миттєво і рідко потрапляють у спам. Для системи мережевої розсилки електронної пошти через REST API вибір Gmail SMTP є доцільним, адже він забезпечує стабільну роботу сервісу та просту інтеграцію зі Spring Boot.

Використання Gmail SMTP не потребує складних налаштувань для інтеграції у Spring Boot. Достатньо вказати параметри підключення у файлі application.properties або application.yml, де визначаються хост, порт, логін та пароль додатка. Google також підтримує створення спеціальних «паролів додатків», які дозволяють безпечно взаємодіяти із сервісом без передачі основного пароля облікового запису.

Gmail SMTP має глобальну інфраструктуру з розподіленими датацентрами, що забезпечує стабільну роботу системи незалежно від місця розташування користувачів. У разі тимчасових збоїв сервіс автоматично виконує повторне доставлення, що особливо важливо для масових розсилок і регулярних повідомлень.

Ще однією перевагою є можливість детального логування та моніторингу. Кожне відправляння листа може фіксуватися у базі даних або логах системи разом із відповіддю сервера, яка включає код статусу, час відправлення та результат операції. Це дозволяє розробнику аналізувати помилки, відстежувати недоставлені листи та своєчасно реагувати на проблеми. Інтеграція Gmail SMTP

зі Spring Boot забезпечує централізоване керування відправленнями та контроль стану кожної розсилки.

Таким чином, Gmail SMTP Server є доцільним, безпечним і ефективним рішенням для системи мережевої розсилки електронної пошти через REST API. Він забезпечує гнучке налаштування та повну сумісність із Java, Spring Boot і Maven, формуючи стабільну та сучасну інфраструктуру для поштового сервісу.

3. ОПИС І ОБҐРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Система мережевої розсилки електронної пошти через REST API є сучасним продуктом, який реалізується з урахуванням вимог до безпеки, продуктивності, та масштабності. Основною метою розроблення системи є забезпечення конфіденційності, цілісності та захищеності інформації відповідно до сучасних вимог безпеки. Система створена для підтримання регулярної та надійної відправки електронної пошти, одночасно обробляючи великі обсяги запитів і забезпечуючи збереження інформації.

Для реалізації системи обрана мова програмування Java, система збірки Maven, фреймворк Spring Boot, реляційна база даних H2 та сервер Gmail SMTP. Цей набір інструментів забезпечить стабільну роботу, високу продуктивність і гнучке налаштування компонентів. Використання цих технологій дозволить легко оновлювати або розширювати систему.

Архітектурні особливості

Система мережевої розсилки електронної пошти через REST API створена за стандартною трирівневою архітектурою, що включає рівень взаємодії з користувачем, рівень логіки та рівень збереження даних. Така структура забезпечує модульність, гнучкість у налаштуванні та можливість масштабування системи залежно від обсягів оброблюваних повідомлень.

Особливе місце в архітектурі системи відводиться забезпеченню трирівневої моделі захисту даних, що включає захищене передавання інформації з використанням криптографічних протоколів, шифрування даних у сховищі та застосування механізмів контролю цілісності даних.

Для реалізації системи обрана мова програмування Java, яка забезпечує стабільність, об’єктно-орієнтовану структуру та довгострокову підтримку.

Використання JVM гарантуватиме високу продуктивність навіть під значним навантаженням і дозволяє ефективно виконувати багатопотокові задачі, характерні для масових розсилок. Сувора типізація Java сприяє зменшенню кількості помилок ще на етапі компіляції та забезпечує коректну обробку даних користувачів.

Рівень взаємодії з адміністратором реалізований через графічний інтерфейс та REST API, який приймає та повертає дані у форматі JSON. Адміністратору надається змога створювати нових користувачів, редагувати, видаляти та переглядати існуючих, а також переглядати статистику відправлень. Цей рівень створюється на базі фреймворку Spring Boot, що спрощує обробку HTTPS-запитів і конфігурацію вебсервісу, забезпечуючи зручний та стандартизований інтерфейс для інтеграції з іншими додатками або клієнтськими інтерфейсами.

Рівень логіки відповідає за обробку запитів, управління чергами повідомлень, реалізацію правил системи та автоматизацію процесу розсилки. Основні функціональні модулі розроблені за призначенням і взаємодією через внутрішні інтерфейси. Для забезпечення одночасної обробки великої кількості завдань використовується багатопотоковість, що дозволятиме виконувати паралельні процеси без перевантаження системи.

Рівень збереження даних відповідає за надійне зберігання інформації про користувачів, їхні розсилки та журнали подій. У системі використовується реляційна база даних H2, що забезпечує швидкий доступ до інформації. У разі масштабування можливим буде перехід на зовнішні бази, такі як PostgreSQL або MySQL, що забезпечуватиме стабільну роботу системи навіть при обробці великих обсягів даних та численних паралельних запитах.

Для керування залежностями та структури проєкту використовується Maven, що забезпечує чітку організацію всіх етапів життєвого циклу збірки. Надсилання електронної пошти відбувається через JavaMail API за допомогою SMTP, що дозволяє інтегрувати зовнішні поштові сервіси та забезпечити стабільне та надійне доставлення повідомлень.

Модулі та компоненти

Система мережевої розсилки електронної пошти у рамках цієї розробки створена як сукупність п’яти основних модулів: модуль інтерфейсу, модуль управління користувачами, модуль управління розсилками, модуль роботи з базою даних та модуль безпеки й логування. Кожен модуль виконує чітко визначені функції, проте всі вони взаємодіють між собою, утворюючи єдиний механізм роботи системи.

Модуль інтерфейсу реалізує взаємодію адміністратора із системою через веб-сторінки, де основну роботу виконують HTML-шаблони з елементами керування, такими як кнопки, форми та таблиці. WebController виступає посередником між браузером і модулем управління користувачами, приймає запити від браузера і відображає сторінки. RestController своєю чергою виступає логічним посередником між браузером та модулем управління користувачами. Він приймає запит у форматі JSON та передає в модуль управління користувачами для відповідних дій, після чого повертає відповідь або дані теж у форматі JSON. Створений RestController через REST API дуже зручний для інтеграції з любими сервісами.

Модуль управління користувачами відповідає за створення, редагування, видалення та перегляд облікових записів користувачів. Операції реалізовані через гарфічний інтерфейс та REST API, які ідентифікують користувачів за іменем або електронною адресою. Створена можливість посторінкового перегляду списку користувачів та сортування за датою, що полегшуватиме керування великою кількістю отримувачів. Цей модуль фактично є точкою входу до серверу системи, формуючи базу користувачів для подальших розсилок.

Модуль управління розсилками забезпечує безпосереднє надсилання листів через графічний інтерфейс та REST API. Система підтримує як ручне відправлення конкретному користувачу, так і автоматичну масову розсилку за заданим розкладом, який задається cron-завданнями. Модуль також веде

моніторинг виконання розсилок, відображаючи, коли та кому були відправлені повідомлення, що підвищує прозорість і контроль роботи системи.

Модуль роботи з базою даних відповідає за збереження та обробку всієї інформації про користувачів, розсилки та cron-завдання. У системі використовується база H2, яка забезпечує швидкий доступ до даних. Це дозволяє легко розширювати функціонал або при необхідності перейти на іншу базу даних, не змінюючи основну логіку системи.

Модуль безпеки та логування забезпечує контроль доступу, шифрування даних та ведення журналу подій. Впроваджено трирівневу модель безпеки, що включає TLS для захищеної передачі даних, AES для шифрування інформації користувача у базі та SHA-256 для контролю цілісності логів. Події створення користувачів, розсилок за завданням та відправлення листів фіксуються у журналах, що дозволяє швидко реагувати на збої та контролювати стан системи.

Завдяки поділу на чіткі модулі система залишається зрозумілою, легкою у підтримці та масштабуванні. Кожен компонент виконує свою функцію, а спільна робота всіх модулів формує гнучку, надійну та готову до подальшого розвитку структуру системи.

Масштабування та продуктивність

Система розроблена з урахуванням можливого зростання кількості користувачів та обсягів оброблюваних даних. Архітектура передбачає гнучкі механізми масштабування, що дозволяють адаптувати роботу платформи до зростаючих навантажень як вертикально, так і горизонтально.

Вертикальне масштабування передбачає збільшення ресурсів окремого сервера, зокрема оперативної пам’яті, обчислювальної потужності процесора та дискового простору для бази даних. Такий підхід дозволяє підтримувати стабільну роботу системи при збільшенні обсягів оброблюваних повідомлень без зміни архітектури застосунку.

Горизонтальне масштабування передбачає розгортання кількох копій застосунку в кластері. Це дає змогу паралельно обробляти тисячі REST API-

запитів та одночасно виконувати численні cron-завдання для автоматичної розсилки електронної пошти. Така організація роботи забезпечує ефективну обробку масових розсилок та підтримує безперервність функціонування системи навіть під високим навантаженням.

Для підвищення продуктивності використовується багатопотоковість, що дозволяє виконувати операції одночасно без блокування інших процесів. Це важливо для модулів управління розсилками, які обробляють великі черги повідомлень, де незначні затримки впливають на своєчасність доставлення листів. Використання JVM забезпечує ефективне керування потоками та ресурсами, оптимально використовуючи наявні апаратні потужності.

Масштабованість на рівні бази даних також врахована. У разі значного збільшення навантаження можливий перехід на більш потужні системи управління базами даних, такі як PostgreSQL або MySQL, що гарантує стабільну роботу навіть при обробці великих обсягів даних та численних паралельних запитах.

Особливості безпеки

Безпека розглядається як ключовий аспект функціонування системи мережевої розсилки електронної пошти. Для її забезпечення використовується трирівнева модель захисту даних, яка охоплює всі основні точки взаємодії та обробки інформації.

Перший рівень, який відповідає за транспортування, забезпечує захищену передачу даних між користувачем та системою. Усі запити та відповіді передаються через HTTPS із застосуванням протоколу TLS. Такий підхід гарантує конфіденційність інформації під час передачі та запобігає перехопленню або модифікації даних сторонніми особами.

Другий рівень, який відповідає за безпечне зберігання інформації про користувачів у базі даних H2. Дані зберігаться у зашифрованому вигляді із застосуванням алгоритму AES. Така організація забезпечить захист даних та спрощуватиме адміністрування системи.

Третій рівень – це контроль за операціями відправлення листів що виконує система. Для захисту цілісності записів відправлень використовується хешування за алгоритмом SHA-256, що дозволяє відстежувати спроби несанкціонованого втручання. Логування дає можливість оперативно реагувати на помилки чи збої та забезпечує повний контроль за станом системи.

Комплексний підхід до захисту даних дозволяє мінімізувати ризики несанкціонованого доступу та гарантує передбачувану, надійну і безпечну роботу системи навіть під час високого навантаження, зокрема при масових розсилках або одночасній обробці великої кількості запитів.

Обробка користувацьких даних

Система виконує роботу з різними типами даних необхідних для планування та управління розсилками. Статична типізація Java дозволяє на ранньому етапі виявляти потенційні помилки обробки даних, а додаткові перевірки на null-значення та відповідність формату забезпечують цілісність інформації в базі та стабільність логіки системи.

Модуль обробки черг повідомлень створений як компонент системи, що відповідає за порядок виконання розсилок та контроль повторних спроб у разі помилок. Для ефективної роботи застосовується багатопотоковість, що дозволяє одночасно обробляти велику кількість повідомлень і запобігатиме блокуванню у системі.

Кожна операція виконується логічно ізольовано. Така організація спрощує локалізацію можливих помилок і підвищує прозорість роботи системи. Відправлення фіксуються в логах із зазначенням часу та типу дії, що забезпечує контроль історії операцій та можливість аналізу ефективності обробки повідомлень.

Механізм надсилання електронної пошти

Система інтегрується із зовнішнім сервісом Gmail SMTP, який виступає транспортним рівнем для доставлення повідомлень користувачам. Такий підхід

дозволяє уникнути розгортання власного поштового сервера, водночас забезпечує надійне та безпечне доставлення листів.

Використання протоколу SMTP через бібліотеку Spring Boot Starter Mail є основним способом взаємодії з поштовим сервісом. Конфігурація включає налаштування хосту, порту, облікових даних і параметрів шифрування, що дозволяє забезпечити безпечне з'єднання та захист даних користувачів.

Процес відправлення повідомлення включає ініціалізацію сесії, перевірку аутентифікації та встановлення захищеного з'єднання за допомогою TLS. Далі формується структура листа із зазначенням одержувача, теми та тіла повідомлення. Після цього дані передаються на сервер Gmail, який виконує доставлення листа до адресата. Для підвищення безпеки передбачені «паролів додатків», що дозволяє взаємодіяти із сервісом без застосування основного пароля облікового запису.

Архітектура системи побудована з використання загального інтерфейсу MailService і передбачає інтеграцію інших поштових сервісів шляхом реалізації нового класу-відправника. Такий підхід забезпечує гнучкість системи та можливість її розвитку без зміни основного коду.

Висновок

Реалізована система виконує основні функції масової розсилки електронної пошти. Її архітектура побудована на принципах модульності та розділення відповідальності, що забезпечує чітку організацію компонентів і спрощує подальший розвиток системи.

Графічний інтерфейс та REST API використовуються як інтерфейси для обробки запитів та взаємодії між користувачем і серверною частиною. Це забезпечує зручне використання та інтеграцію з іншими вебзастосунками та клієнтськими інтерфейсами без необхідності змінювати внутрішню логіку системи.

Автоматизоване відправлення листів за допомогою cron-задач дозволяє системі виконувати заплановані операції автономно, без постійної участі

користувача. Це підвищує ефективність розсилок і забезпечує своєчасне доставлення повідомлень.

Логування та формування статистики створюють основу для контролю та аналізу роботи системи. Це відкриває перспективу для подальшого розвитку функціоналу, зокрема впровадження аналітичного модуля для більш детальної обробки даних та оптимізації розсилок.

Забезпечення трирівневої безпеки, дозволяє контролювати доступ, фіксувати зміни та забезпечувати безпечну передачу інформації, що особливо важливо під час масових розсилок або обробки великої кількості запитів.

Отже, розроблена система є гнучким, надійним і масштабованим рішенням для організації масової розсилки електронної пошти із забезпеченням трирівневого захисту даних під час їх передачі та зберігання. Вона поєднує простоту використання з механізмами захисту інформації, що гарантує безпечну передачу повідомлень користувачам. Система зберігає стабільність роботи під навантаженням і підтримує як ручне, так і автоматичне відправлення листів. Продумана архітектура дозволяє легко адаптувати її до нових вимог і інтегрувати з іншими сервісами у майбутньому.

3.2 Розробка структурної схеми

Всупереч розвитку месенджерів і соціальних мереж, електронна пошта залишається одним з основних способів офіційної комунікації для багатьох компаній світу. Вона дозволяє швидко обмінюватися повідомленнями та файлами забезпечуючи інформаційний міст між користувачами.

Щодня багато організацій стикаються з проблемою обробки великого обсягу листів та своєчасного доставлення. Зі збільшенням кількості користувачів та обсягів даних зростає потреба в автоматизації процесів, що забезпечує ефективність і швидкість обміну повідомленнями. Саме тому розробка

автоматизованих систем мережевої розсилки електронної пошти набуває актуальності.

У рамках даної роботи розглядається концепція системи мережевої розсилки електронної пошти через REST API, яка розроблялася з урахуванням вимог безпеки, продуктивності та масштабованості. Основною метою створення системи є гарантування конфіденційності, цілісності та безпечного зберігання інформації відповідно до сучасних стандартів безпеки. Подібні системи можуть стати невіддільною частиною роботи організацій, що прагнуть до швидкого та надійного обміну даними, зокрема для масових розсилок, внутрішніх сповіщень та інтеграції з іншими вебзастосунками.

Для реалізації системи використані сучасні технології, такі як мова програмування Java, фреймворк Spring Boot, система збірки Maven, реляційна база даних H2 та зовнішній SMTP-сервер Gmail. Така комбінація рішень забезпечує надійну платформу, здатну зберігати дані користувачів, обробляти черги повідомлень та виконувати розсилку відповідно до заданих правил і розкладів.

Система розроблена як трирівнева архітектура з виділенням рівня користувача, рівня логіки та рівня збереження даних. В архітектурі особливе місце відводиться трирівневій моделі захисту даних із забезпеченням безпечної передачі інформації, шифруванням даних користувачів у сховищі та контролем цілісності логів. Такий підхід гарантує стабільність роботи системи під високим навантаженням та забезпечує гнучкість для її подальшого розвитку.

Головною метою проєктування структурної схеми є визначення логіки взаємодії всіх компонентів від створення листів до доставлення їх у поштові скриньки одержувачів. Структурна схема наочно демонструє послідовність передачі інформації, що сприяє модульності, зрозумілості та гнучкості при масштабуванні системи.

Рисунок 3.1 – Структурна схема системи

Загальна характеристика підкомпонентів структурної схеми

Система мережевої розсилки електронної пошти у рамках цієї розробки побудована як сукупність фізичних підкомпонентів, кожен із яких виконує конкретну роль та забезпечує стабільну роботу платформи.

Перший підкомпонент – це інтерфейс користувача. Він дозволяє створювати, переглядати, редагувати та видаляти користувачів, формувати тему та вміст листа та надсилати як окремому користувачу так і всім за corn-розкладом. Інтерфейс також відображає статистику розсилок, що дозволяє розробляти нові стратегії відправлень.

Другий підкомпонент – це серверна частина на Spring Boot. Він приймає HTTPS-запити від вебінтерфейсу, перевіряє коректність даних, виконує дії над ними та передає їх на SMTP-сервер. Сервер підтримує багатопотокову обробку,

що дозволяє одночасно обслуговувати велику кількість запитів та забезпечує стабільну роботу системи.

Третій компонент – це база даних. Вона створена для збереження інформації про користувачів, розкладу запланованих відправлень з темами й листами та логів відправлених листів. Така організація даних забезпечує можливість відновлення інформації у разі збоїв та спрощує аналіз ефективності розсилок.

Четвертий компонент – це сервер SMTP. Використовується зовнішній сервіс Gmail SMTP для відправлення листів, керування чергою повідомлень, виконання повторних спроб доставлення у разі тимчасових проблем та передавання статусів. Використання готового сервісу забезпечує високий рівень надійності та мінімізує ризик потрапляння листів у спам, водночас дозволяючи зосередитися на розробці та логіці системи.

Така концептуальна організація компонентів може забезпечити безпеку передачі, надійне зберігання даних та контроль цілісності, одночасно обробляючи великі обсяги повідомлень, підтримуючи стабільність роботи та своєчасне доставлення.

Створення запиту користувачем системи

Система взаємодіє з користувачем через графічний інтерфейс та REST API, які слугують інтерфейсом для формування та управління електронними повідомленнями. Запити на відправлення повідомлень включають адресу одного або всіх одержувачів, тему, текст повідомлення з можливістю зазначення запланованого часу відправлення.

Виконується попередня перевірка даних на коректність email-адрес, що дозволяє своєчасно виявляти помилки. Для підвищення продуктивності та надійності обробки запитів у системі використовується багатопотоковість, що дає змогу одночасно готувати та обробляти велику кількість повідомлень.

Такий підхід забезпечує передбачуване та структуроване формування запитів і створює основу для подальшої безпечної обробки та доставлення повідомлень.

Підготовка листа

Для підвищення продуктивності та ефективності обробки використовується багатопотоковість, що дозволяє одночасно відправляти велику кількість листів, оптимізуючи використання ресурсів і скорочуючи час.

Система автоматично формує статистику відправлених листів на основі логування всіх подій відправлення. Кожен факт надсилання повідомлення зберігається у базі даних, де фіксується ім'я та пошта користувача, тип запуску операції, а також дата й час створення запису. Така модель забезпечуватиме можливість перегляду статистики та відстеження стану системи при зростанні навантаження. Записи формуються автоматично, що гарантує повноту та правдивість статистичних даних.

Така організація забезпечує безпечну, передбачувану і структуровану підготовку повідомлень.

Передача листа у мережі Інтернет

Лист, створюваний у системі мережевої розсилки, проходитиме багаторівневий шлях через мережу Інтернет до поштової скриньки одержувача. Після генерації у внутрішній системі відбувається формування повідомлення у форматі, сумісному з протоколом SMTP, зі створенням усіх необхідних заголовків, MIME-структур та унікальних ідентифікаторів. Далі повідомлення інкапсулюється у транспортні TCP-сегменти та мережеві IP-пакети, що забезпечує можливість його передачі у мережевому середовищі.

Система доменних імен DNS виконує визначення поштового сервера домену одержувача на основі MX-записів. Враховуються їхні пріоритети, щоб обирати найбільш доступний і надійний маршрут. Після отримання IP-адреси цільового сервера повідомлення спрямовується мережею через інфраструктурні вузли, проходячи локальні та магістральні маршрутизатори, транзитні шлюзи та

точки обміну трафіком. На кожному з них маршрутизатор аналізує IP-заголовки та використовує таблиці маршрутизації для визначення подальшого напрямку руху пакета.

Надійність передавання даних забезпечується протоколом TCP, що контролює доставлення пакетів, перевірку контрольних сум і за потреби ініціює повторну передачу втрачених сегментів. Проміжним етапом маршрутизації лист проходить через поштовий сервер, який відповідає за тимчасове зберігання, повторні спроби доставлення та базову валідацію повідомлення. На цьому рівні також проводяться первинні перевірки автентичності відправника та фільтрація для запобігання доставлення небажаних або шкідливих повідомлень.

Перед завершенням доставлення повідомлення надходить на поштовий сервер одержувача, де знову виконуються перевірки безпеки й сортування, після чого лист доставляється до відповідної теки поштової скриньки користувача. Для забезпечення конфіденційності та цілісності на всіх мережевих етапах застосовується TLS-шифрування.

Таким чином, мережевий етап доставлення електронного повідомлення забезпечує його надійне та оптимальне маршрутизування через інфраструктуру Інтернет, включаючи визначення доменних імен, встановлення поштових серверів, переходи між мережевими вузлами та автономними системами.

Сервер SMTP

На проміжному етапі лист надходить на SMTP-сервер Gmail, який виступає хабом для подальшого доставлення. Сервер SMTP (Simple Mail Transfer Protocol) розглядається як один із ключових компонентів системи мережевої розсилки електронної пошти, оскільки він відповідає за передачу сформованих повідомлень між поштовими серверами. Це дозволяє зосередитися на реалізації основного функціоналу системи без необхідності розгортання власного поштового сервера та постійного його адміністрування, а також використовувати вже перевірені механізми безпеки та високу доступність ресурсу.

Встановлення захищеного TLS-з’єднання між системою та сервером Gmail, забезпечує захищений обмін даними під час сеансу та захист від перехоплення або модифікації листа на проміжних етапах передачі. Також проводиться автентифікація доменів і серверів за стандартами SPF, DKIM та DMARC, що зменшує ймовірність потрапляння повідомлень у спам.

Система SMTP-сервера формує внутрішнє представлення повідомлення, яке включає всі необхідні атрибути: адреси одержувачів, тему та текст. Під час підготовки листа плануються MIME-заголовки з адресами, часом створення, типом контенту та унікальним ідентифікатором для відстеження стану листа на наступних етапах.

Окремо виділяються операції відправлення повідомлення через сервер Gmail. Такий підхід дозволяє ізолювати потенційні збої і забезпечує стабільність роботи системи. Відправлення фіксуються у логах із зазначенням ім’я та пошти користувача часу та типу відправлення, що створює основу для контролю та подальшого аналізу ефективності розсилок.

Система MailService відстежує статус доставлення та здійснює повторні спроби у разі тимчасових збоїв або недоступності сервера. Взаємодія із сервером дає змогу отримувати коди стану. Така організація дозволяє контролювати процес доставлення, забезпечує можливість повторного відправлення у разі помилок і підтримує ефективність при великих обсягах розсилки.

Архітектура системи залишає простір для майбутніх удосконалень, таких як інтеграція з кількома SMTP-серверами для підвищення надійності доставлення, оптимізація часу повторних спроб або впорядкування листів у великих чергах.

Отримувачі повідомлень

Після проходження проміжного SMTP-сервера повідомлення надходить на поштовий сервер одержувача, де проходить його комплексна перевірка для забезпечення безпечного та коректного доставлення до поштової скриньки користувача. На цьому етапі сервера виконується автентифікацію та перевірка

домену відправника, включно з оцінкою записів SPF, DKIM та DMARC, що дозволяє підтвердити легітимність джерела та знизити ризик потрапляння листа у спам. У разі невдалої автентифікації повідомлення буде позначене як потенційно небезпечне або заблоковане.

Далі відбувається проходження спам-фільтрів та перевірка за чорними списками IP-адрес, доменів і вмісту листа. Сервером аналізується повідомлення на відповідність внутрішнім політикам безпеки, перевіряючи вкладення, посилання та текст на наявність шкідливого коду, підозрілих елементів або порушень правил організації. При виявленні потенційних загроз лист буде заблокований, відправлений у карантин або позначений для подальшого аналізу.

Усі дії автоматично фіксуються, включно з часом отримання повідомлення, результатами автентифікації, проходженням спам-фільтрів, статусом доставлення та кодами помилок у разі відхилення. Після успішного проходження всіх перевірок повідомлення доставляється до поштової скриньки одержувача.

3.3 Розробка функціональної схеми

Головною метою проєктування схеми є визначення принципів функціонування системи та взаємодії її компонентів. Для наочного розуміння доцільно розглянути систему в розрізі п’яти основних модулів, серед яких модуль інтерфейсу, модуль управління користувачами, модуль управління розсилками, модуль роботи з базою даних, модуль безпеки й логування. Кожен із цих модулів виконує конкретні функції, взаємодіє з іншими компонентами через інтерфейс користувача та забезпечує виконання загальних завдань системи. Такий підхід дозволяє чітко простежити розподіл функцій, логіку обробки даних і взаємозв’язки між користувачем та системою.

Модуль інтерфейсу

Модуль інтерфейсу реалізує взаємодію адміністратора через графічний інтерфейс та REST API. WebController приймає запити від браузера та формує

HTML-сторінки з кнопками, формами та таблицями для створення, редагування та видалення користувачів, відправлення листів і перегляду статистики. Для динамічного доступу до даних і взаємодії з іншими модулями використовується RestController, який через REST API повертає інформацію у форматі JSON. Завдяки цьому адміністратор може працювати із системою через сторінки браузера залишаючи логіку виконання серверу через RestController.

Модуль управління користувачами

Цей модуль пов'язаний з модулем інтерфейсу та надає можливість створення та підтримки облікових записів користувачів, що забезпечує іншим компонентам системи необхідну інформації для розсилок. Тобто він надає можливість додавання нового користувача з ім'ям та електронною поштою, редагування цих даних, видалення користувачів та перегляду списку всіх зареєстрованих облікових записів. Дані про користувачів зберігаються у реляційній базі H2, де кожному користувачу призначено унікальний ідентифікатор та дату створення.

Модуль управління розсилками

Цей модуль відповідає за відправлення електронної пошти користувачам. Листи можуть надсилатися як за запитом конкретному користувачу, так і автоматично всім користувачам за заданим cron-завданням. Для автоматичної розсилки можлива реалізація будь-якого графіка, наприклад кожні 5 хвилин, щодня о певній годині або щотижня в конкретний день. Модуль відповідає за створення, редагування, видалення та перегляд cron-завдань, щоб автоматизація була максимально гнучкою. Використовується багатопотоковість для одночасної обробки великої кількості повідомлень, що дозволяє уникати блокувань і забезчує своєчасне доставлення листів. Модуль також пов'язаний з механізмами логування та базою даних.

Модуль роботи з базою даних

Цей модуль відповідає за збереження інформації системи. Для зберігання використовується база H2, яка забезпечує швидкий доступ до даних. Модуль

забезпечує додавання нових записів, оновлення чинних та надання даних для відображення статистики. Логи ведуться з інформацією про те кому був надісланий лист, чи він відправлений ручним, чи автоматичним способом, а також дату та час виконання операції. У разі масштабування планується можливість підключення зовнішніх реляційних баз даних, таких як PostgreSQL або MySQL, що забезпечить стабільну роботу системи при збільшенні обсягів даних і паралельних запитів.

Модуль безпеки та логування

Цей модуль відповідає за забезпечення безпеки системи та контроль цілісності даних . У ньому впроваджена трирівнева модель безпеки, яка включає TLS для захищеної передачі даних через REST API, AES для шифрування даних користувачів у базі та SHA-256 для контролю цілісності логів. Модуль веде журнал надсилань, включно з типом відправлення листів, часом, імен та поштою користувача. Статистика включає дату і час першого та останнього листа, що забезпечує контроль активності та своєчасне виявлення проблем.

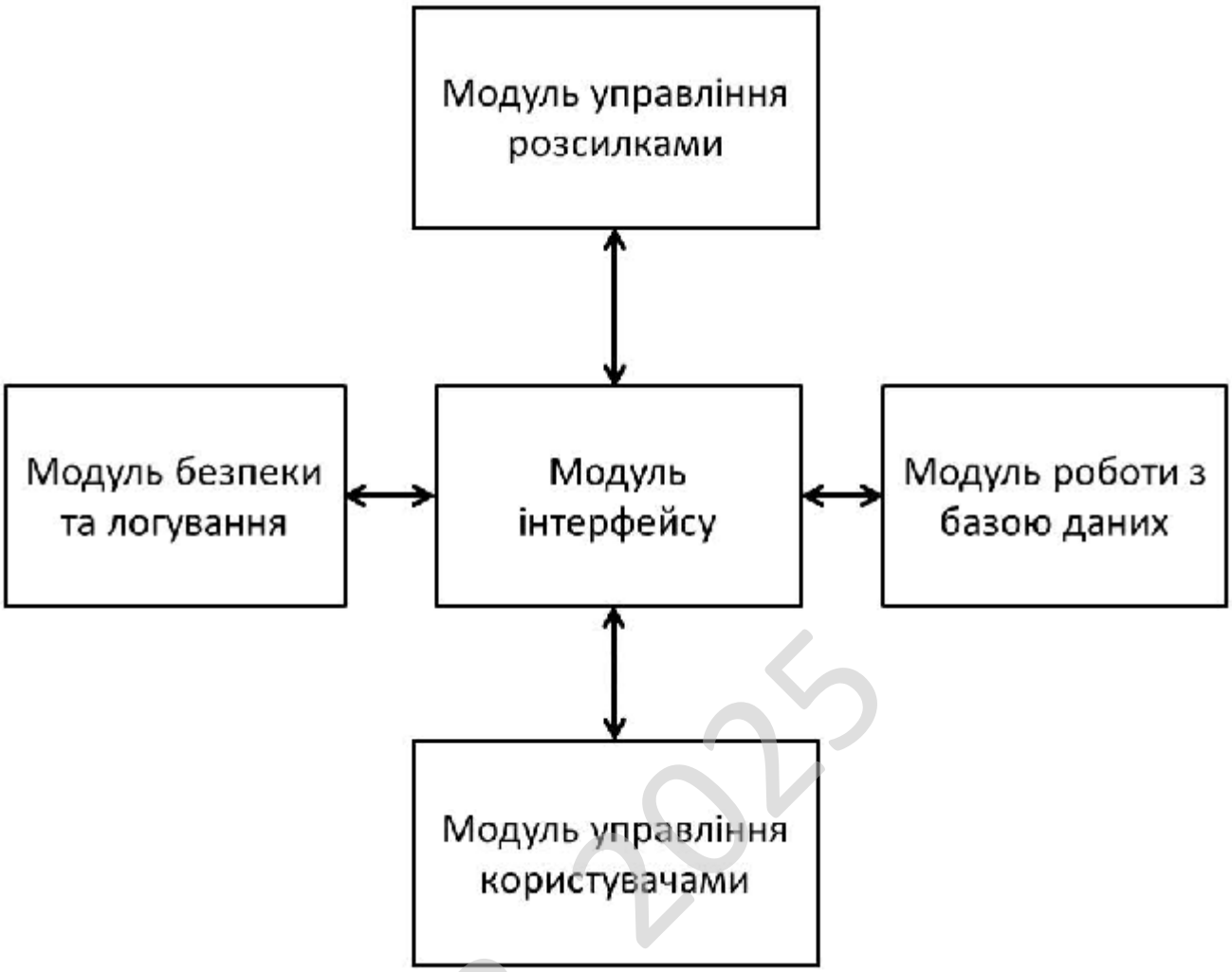

Рисунок 3.2 – Функціональна схема системи

3.4 Розробка діаграми процесів

Діаграма процесів слугує для наочного відображення процесів обробки даних у розроблюваній системі. На етапі проєктування системи формується діаграма взаємодії на рівні контексту, що дасть можливість показати загальну логіку роботи та зовнішні взаємозв'язки системи. Надалі вона може деталізуватися шляхом уточнення процесів і потоків даних, що дозволить повністю описати роботу функціональних компонентів.

Розроблена діаграма процесів наведена на рисунку 3.3 дає змогу зрозуміти, яким чином відбувається взаємодія всіх основних елементів системи під час виконання розсилки електронної пошти.

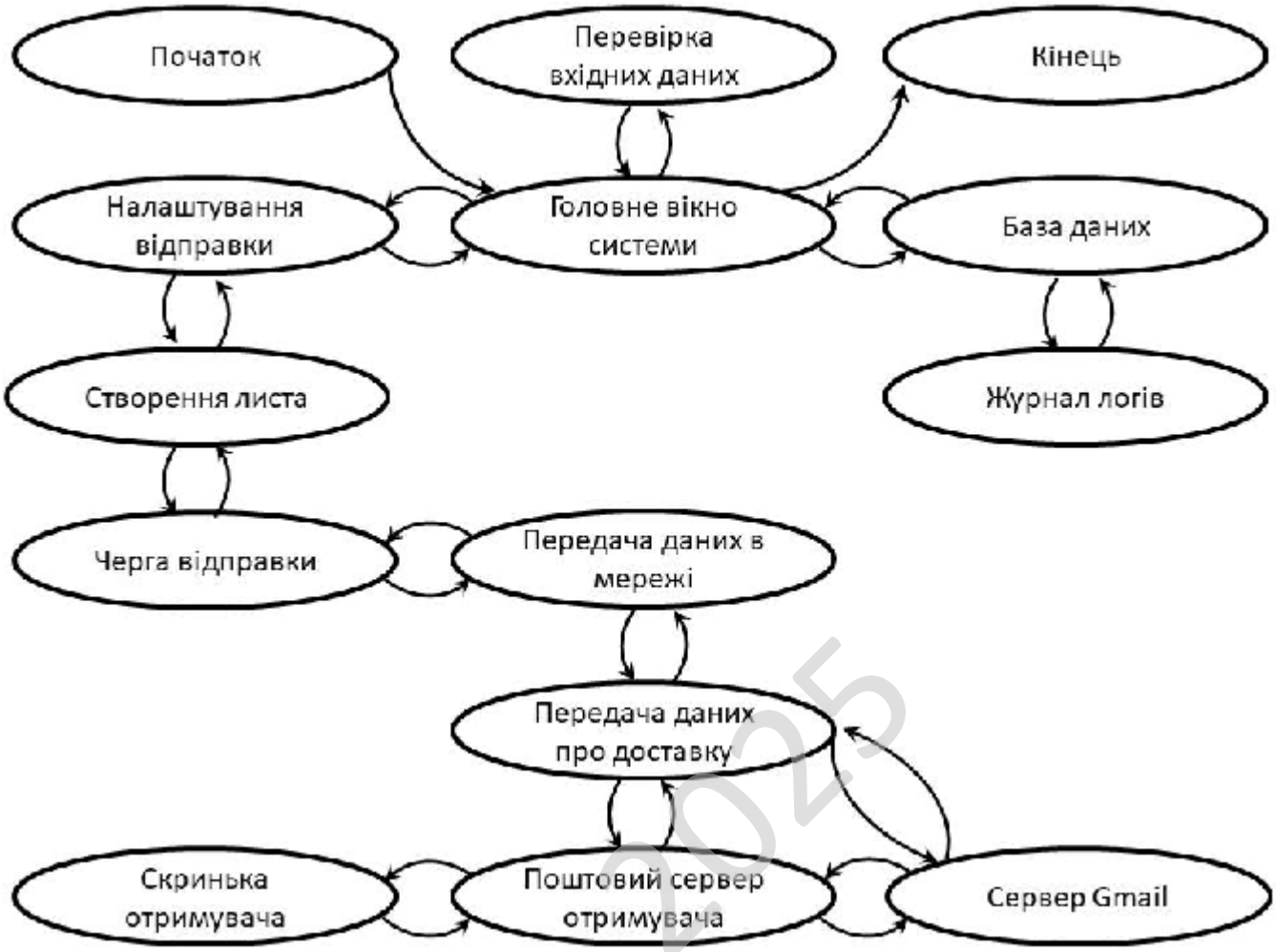

Рисунок 3.3 – Діаграма процесів системи

4 РЕАЛІЗАЦІЯ ПРОЄКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЄКТНИХ РІШЕНЬ

4.1 Розробка блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є важливою частиною програмного забезпечення. Від того, наскільки точно і детально вони складені, залежить якість роботи всієї програми. На схемі функціональні блоки позначаються прямокутниками з підписами, які показують, яку саме функцію вони виконуватимуть. Зв'язки між блоками зображаються лініями, що показують напрям передачі даних або керування.

Система мережевої розсилки електронної пошти реалізована для безпечного надсилання електронних повідомлень. Основним завданням системи є забезпечення безпеки розсилки листів з можливістю взаємодії через інтерфейс. Алгоритм функціонування системи забезпечує поетапну обробку даних, що охоплює виконання послідовних функцій та підпрограм.

Загальна логіка роботи системи відображається на блок-схемі рисунок 4.1, яка демонструє послідовність основних операцій та умовні переходи. Алгоритм починається з ініціалізації системи, під час якої запускаються базові модулі, зокрема модуль інтерфейсу, модуль управління користувачами, модуль роботи з базою даних та модуль безпеки і логування. На цьому етапі встановлюється захищене з'єднання та формується стартова сторінка на локальному сервері. Адміністратор отримує можливість створювати, редагувати або видаляти облікові записи, відправляти одиночні або масові розсилки, а також переглядати списки користувачів та логів.

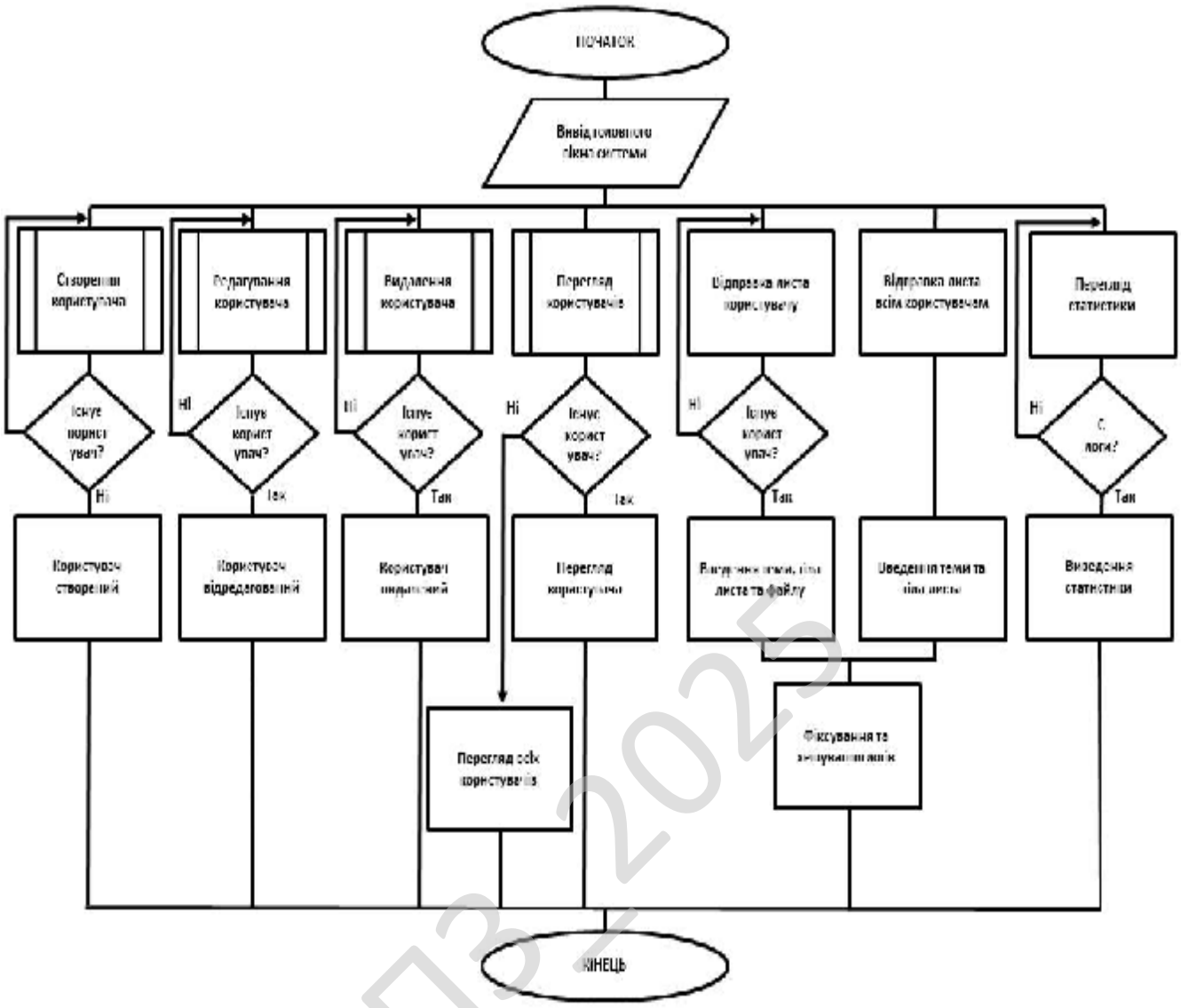

Рисунок 4.1 – Блок-схема програми

Головне вікно виступає стартовою точкою взаємодії з системою. Для створення користувача адміністратор повинен вести ім’я та пошту користувача. Після введення даних система переходить до етапу валідації. Поступово перевіряються поля імені та пошти, вони не можуть бути порожніми. Формат пошти повинен відповідати стандарту, а формат імені повинен відповідати заданим вимогам. У разі введення некоректних даних система формує повідомлення про помилку та виводить правила введення.

Після успішної перевірки даних система виконує ініціалізацію користувача. На цьому етапі встановлюється зв’язок із базою даних H2 для виконання запиту

на перевірку існування користувача за параметрами ім’я та пошти. Якщо запис відсутній, відбувається створення нового користувача шляхом присвоєння id, який автоматично генерується як унікальний ідентифікатор, при цьому заноситься дата та час створення.

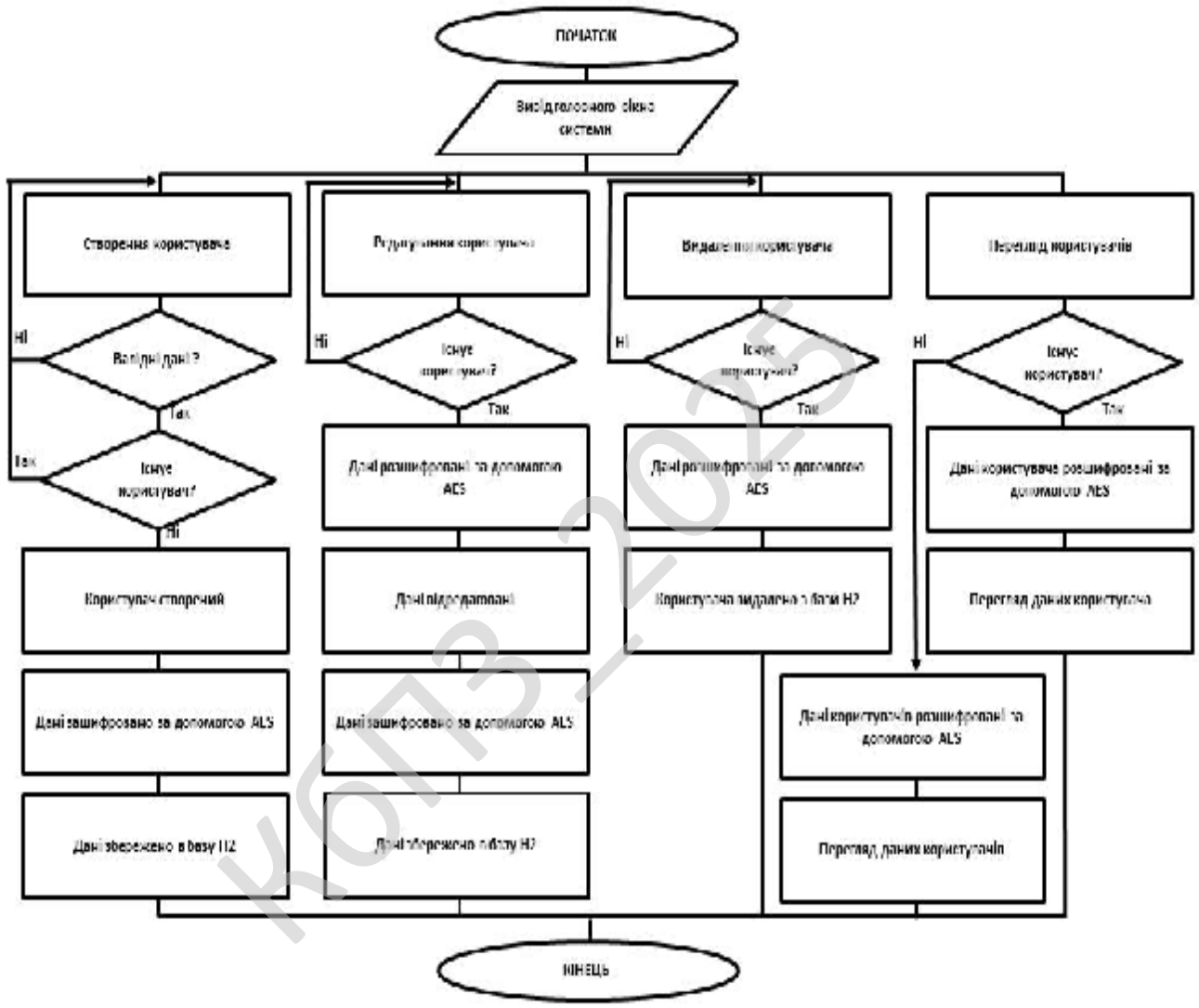

Рисунок 4.2 – Блок-схема підпрограми

Якщо користувач існує в базі, система повідомляє про це і процес редагування продовжується до введення коректних даних. Додатково присутня можливість редагування, видалення та перегляду користувачів й статистики відправлень. Для зручності реалізований посторінковий перегляд та сортування

користувачів за датою створення. Функція пошуку за іменем, поштою забезпечує швидкий доступ до потрібного запису.

Після того, як база даних заповнена користувачами, адміністратор переходить до відправлення листів. Він має можливість обрати, чи надіслати лист окремому користувачу або всім користувачам одразу за заданим розкладом. Система забезпечує формування та відправку повідомлень, а також логування результатів, що дозволяє контролювати стан розсилки.

Процес відправлення реалізується через Gmail SMTP сервер із використанням захищеного TLS-з'єднання. Це забезпечує конфіденційність і цілісність переданих даних. Усі відправлення фіксуються відповідним записом у журналі логів, що дозволяє аналізувати статистику. Всі записи сортуються за спаданням кількості відправлених листів, а також підтримується посторінковий перегляд для зручності роботи з великими обсягами даних.

Нижче наведено код класу RestController, який є логічним посередником між браузером та модулем курування користувачами:

```
import org.diplomamagistra.email_sender.service.dataService.entity.CronJob;
import org.diplomamagistra.email_sender.service.dataService.entity.LogType;
import org.diplomamagistra.email_sender.service.dataService.entity.User;
import org.diplomamagistra.email_sender.service.dataService.entity.UserDTO;
import org.diplomamagistra.email_sender.service.emailService.CronJobService;
import org.diplomamagistra.email_sender.service.emailService.EmailService;
import
org.diplomamagistra.email_sender.service.securityAndLogService.LogService;
import org.diplomamagistra.email_sender.service.userService.UserService;
import org.springframework.data.domain.Page;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

/**
 * Цей клас створено як логічний посередник між сервером програми та файлами
html. Сервер програми чекає на
 * HTTPS запит браузера, після чого обробляє його за допомогою своїх
сервісів та дає відповідь у вигляді файлу JSON.
 */
```

```
@org.springframework.web.bind.annotation.RestController
@RequestMapping("/rest/api/users")
public class RestController {

    /**
     * Ці поля створено для взаємодії з сервісами програми.
     */
    private final UserService userService;
    private final EmailService emailService;
    private final CronJobService cronJobService;
    private final LogService logService;

    /**
     * Цей конструктор створений для ініціалізації полів сервісів.
     *
     * @param userService сервіс для роботи з користувачами
     * @param emailService сервіс для роботи з поштою
     * @param cronJobService сервіс для роботи з завданнями на відправлення
листів
     * @param logService сервіс для роботи з логами
     */
    public RestController(UserService userService, EmailService
emailService, CronJobService cronJobService,
                          LogService logService) {
        this.userService = userService;
        this.emailService = emailService;
        this.cronJobService = cronJobService;
        this.logService = logService;
    }


    /**
     * Цей метод створений для передачі запиту створення користувача від
браузера до сервісу користувачів.
     *
     * @param request тимчасовий об’єкт для швидкого перетворення даних
     * @return відповідь у форматі JSON про результат створення користувача
     */
    @PostMapping("/createUser")
    public ResponseEntity<Map<String, String>> createUser(@RequestBody
UserDTO request) {
        Map<String, String> response = new HashMap<>();
        try {
            userService.createUser(request.getUsername(),
request.getEmail());
```

```
                response.put("message", "Користувача створено!");
                response.put("messageType", "success");
                return ResponseEntity.ok(response);
            } catch (Exception e) {
                response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
                response.put("messageType", "error");
                return ResponseEntity.badRequest().body(response);
            }
        }

        /**
         * Цей метод створений для передачі запиту редагування даних користувача
від браузера до сервісу користувачів.
         *
         * @param request тимчасовий об’єкт для швидкого перетворення даних
         * @return відповідь у форматі JSON про результат редагування даних
користувача
         */
        @PostMapping("/editUser")
        public ResponseEntity<Map<String, String>> editUser(@RequestBody UserDTO
request) {
            Map<String, String> response = new HashMap<>();
            try {
                userService.editUser(request.getId(), request.getUsername(),
request.getEmail());
                response.put("message", "Дані успішно змінено!");
                response.put("messageType", "success");
                return ResponseEntity.ok(response);
            } catch (Exception e) {
                response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
                response.put("messageType", "error");
                return ResponseEntity.badRequest().body(response);
            }
        }

        /**
         * Цей метод створений для передачі запиту на видалення користувача від
браузера до сервісу користувачів.
         *
         * @param request тимчасовий об’єкт для швидкого перетворення даних
         * @return відповідь у форматі JSON про результат видалення користувача
         */
        @DeleteMapping("/deleteUser")
```

```
    public ResponseEntity<Map<String, String>> deleteUser(@RequestBody
UserDTO request) {
        Map<String, String> response = new HashMap<>();
        try {
            userService.deleteUser(request.getId());
            response.put("message", "Користувача успішно видалено!");
            response.put("messageType", "success");
            return ResponseEntity.ok(response);
        } catch (Exception e) {
            response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
            response.put("messageType", "error");
            return ResponseEntity.badRequest().body(response);
        }
    }

    /**
     * Цей метод створений для передачі запиту на перегляд всіх користувачів
посторінково або одного користувача за
     * його іменем або поштою від браузера до сервісу користувачів.
     *
     * @param request тимчасовий об’єкт для швидкого перетворення даних
     * @return відповідь у вигляді користувачів посторінково
     */
    @PostMapping("/viewUsers")
    public ResponseEntity<Page<User>> viewUsers(@RequestBody UserDTO
request) {
        Page<User> users;
        if (request.getQuery() == null || request.getQuery().isEmpty()) {
            users = userService.getAllUsersInPages(request.getPage(),
request.getSize());
        } else {
            users = userService.searchByNameAndEmail(request.getQuery(),
request.getPage(), request.getSize());
        }
        return ResponseEntity.ok(users);
    }

    /**
     * Цей метод створений для передачі запиту на відправлення листа
користувачу від браузера до
     * сервісу поштового сервісу.
     *
     * @param request тимчасовий об’єкт для швидкого перетворення даних
     * @return відповідь у форматі JSON про результат відправлення листа
```

```
     */
    @PostMapping("/sendEmail")
    public ResponseEntity<Map<String, String>> sendEmail(@RequestBody
UserDTO request) {
        Map<String, String> response = new HashMap<>();
        try {
            emailService.sendEmailToUser(request.getUser(),
request.getSubject(), request.getBody(), LogType.REST);
            response.put("message", "Лист надіслано!");
            response.put("messageType", "success");
            return ResponseEntity.ok(response);
        } catch (Exception e) {
            response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
            response.put("messageType", "error");
            return ResponseEntity.badRequest().body(response);
        }
    }

    /**
     * Цей метод створений для передачі запиту на відправлення листа усім
користувачам за заданим cron-завданням
     * від браузера до сервісу керувань завданнями.
     *
     * @param request тимчасовий об’єкт для швидкого перетворення даних
     * @return відповідь у форматі JSON про результат задання завдання на
відправлення листів
     */
    @PostMapping("/sendEmailAll")
    public ResponseEntity<Map<String, String>> sendEmailAll(@RequestBody
UserDTO request) {
        Map<String, String> response = new HashMap<>();
        try {
            CronJob cronJob = new CronJob();
            cronJob.setSubject(request.getSubject());
            cronJob.setBody(request.getBody());
            cronJob.setExpression(request.getExpression());
            cronJobService.saveCronJob(cronJob);
            cronJobService.sendOnSchedule(cronJob);
            response.put("message", "Листи заплановано!");
            response.put("messageType", "success");
            return ResponseEntity.ok(response);
        } catch (Exception e) {
            response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
```

```
                response.put("messageType", "error");
                return ResponseEntity.badRequest().body(response);
            }
        }

        /**
         * Цей метод створений для передачі запиту на перегляд статистики від
браузера до сервісу керування логами.
         *
         * @param request тимчасовий об’єкт для швидкого перетворення даних
         * @return відповідь у форматі JSON про результат задання завдання на
відправлення листів
         */
        @PostMapping("/viewStatistics")
        public ResponseEntity<Page<LogService.UserLogStats>>
viewStatistics(@RequestBody UserDTO request) {
            Integer pageInteger = request.getPage();
            Integer sizeInteger = request.getSize();
            int page =  pageInteger != null ? request.getPage() : 0;
            int size = sizeInteger != null ? request.getSize() : 5;
            Page<LogService.UserLogStats> statistics =
logService.getUsersLogsStatistics(page, size);
            return ResponseEntity.ok(statistics);
        }

        /**
         * Цей метод створений для передачі запиту на перегляд cron-завдань від
браузера до сервісу керування завданнями.
         *
         * @param request тимчасовий об’єкт для швидкого перетворення даних
         * @return відповідь у форматі JSON про результат задання завдання на
відправлення листів
         */
        @PostMapping("/viewCronJobs")
        public ResponseEntity<Page<CronJob>> viewCronJobs(@RequestBody UserDTO
request) {
            Page<CronJob> cronJobs =
cronJobService.getAllCronJobsInPages(request.getPage(), request.getSize());
            return ResponseEntity.ok(cronJobs);
        }

        /**
         * Цей метод створений для передачі запиту на редагування cron-завдання
від браузера до сервісу керування завданнями.
         *
```

```
     * @param request тимчасовий об’єкт для швидкого перетворення даних
     * @return відповідь у форматі JSON про результат редагування завдання
на відправлення листів
     */
    @PostMapping("/editCronJob")
    public ResponseEntity<Map<String, String>> editCronJob(@RequestBody
UserDTO request) {
        Map<String, String> response = new HashMap<>();
        try {
            cronJobService.editCronJob(request.getId(),
request.getSubject(), request.getBody(), request.getExpression());
            response.put("message", "Дані успішно змінено!");
            response.put("messageType", "success");
            return ResponseEntity.ok(response);
        } catch (Exception e) {
            response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
            response.put("messageType", "error");
            return ResponseEntity.badRequest().body(response);
        }
    }

    /**
     * Цей метод створений для передачі запиту на видалення cron-завдання
від браузера до сервісу керування завданнями.
     *
     * @param request тимчасовий об’єкт для швидкого перетворення даних
     * @return відповідь у форматі JSON про результат видалення завдання на
відправлення листів
     */
    @DeleteMapping("/deleteCronJob")
    public ResponseEntity<Map<String, String>> deleteCronJob(@RequestBody
UserDTO request) {
        Map<String, String> response = new HashMap<>();
        try {
            cronJobService.deleteCronJob(request.getId());
            response.put("message", "Користувача успішно видалено!");
            response.put("messageType", "success");
            return ResponseEntity.ok(response);
        } catch (Exception e) {
            response.put("message", e.getMessage() != null ? e.getMessage()
: "Сталася невідома помилка");
            response.put("messageType", "error");
            return ResponseEntity.badRequest().body(response);
        }
```

```
        }

        /**
         * Цей метод створений для передачі запиту на пошук користувачів за
іменем або поштою від браузера до
         * сервісу користувачів.
         *
         * @param request тимчасовий об’єкт для швидкого перетворення даних
         * @return відповідь у вигляді користувачів
         */
        @PostMapping("/searchByNameAndEmail") // Обробляє GET-запити за /search
для динамічного пошуку
        public ResponseEntity<Page<User>> searchByNameAndEmail(@RequestBody
UserDTO request) {
            Page<User> users =
userService.searchByNameAndEmail(request.getQuery(), request.getPage(),
request.getSize());
            return ResponseEntity.ok(users);
        }

        /**
         * Цей метод створений для передачі запиту на пошук користувача за
іменем його id від браузера до
         * сервісу користувачів.
         *
         * @param request тимчасовий об’єкт для швидкого перетворення даних
         * @return відповідь у вигляді користувача або null
         */
        @PostMapping("/searchById")
        public ResponseEntity<User> searchById(@RequestBody UserDTO request) {
            Optional<User> user = userService.searchUserById(request.getId());
            return ResponseEntity.ok(user.orElse(null));
        }
    }
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою симетричного алгоритму шифрування AES. Симетричний алгоритм AES використовує єдиний секретний ключ, який застосовується для шифрування і для розшифрування даних.

Спочатку формується секретний ключ K фіксованої довжини, який може бути заданий в ручну або згенерований за допомогою генератора випадкових чисел. Даний ключ може мати довжину 128 біт, 192 біта або 256 біт. Від довжини ключа залежить кількість раундів шифрування, тобто відповідно до розміру ключа раундів може бути 10, 12 або 14.

Для шифрування файл M попередньо обробляється та ділиться на блоки фіксованої довжини 128 біт. Через фіксований розмір, алгоритм представляє кожний блок у вигляді двовимірного масиву 4x4. Для кожного блока виконується криптографічне перетворення з використанням секретного ключа K. Це перетворення включає заміну байтів (SubBytes), зсув рядків (ShiftRows), змішування стовпців (MixColumns), додавання ключа раунду (AddRoundKey).

Процедура SubBytes проводить нелінійну заміну байтів використовуючи таблицю S-box. Процедура ShiftRows циклічно зсуває рядок на кількість байтів, що відповідає номеру рядка. Процедура MixColumns змішує байти кожного стовпця за допомогою лінійної трансформації. Процедура AddRoundKey генерує раундові ключі з секретного ключа K відповідно до кількості раундів та по розміру блока розбитих даних 4x4. Кожен байт ключа додається до відповідного байту розбитих блоків за допомогою операції XOR.

При шифруванні ці процедури повторюються відповідно до кількості раундів, але перед першим раундом додається нульовий ключ раундів, а на кінцевому раунді не виконується процедура MixColumns.

У результаті формується зашифрований файл C, який визначається:

$$C = AES_k(M)$$

Зашифроване файл C передається у захищеному вигляді. Без знання секретного ключа K відновлення початкового повідомлення є неможливим.

Для розшифрування повідомлення C виконується зворотне криптографічне перетворення з використанням того самого секретного ключа K, у результаті чого відновлюється початкове повідомлення:

$$M = AES^{-1}(C)$$

Якщо в процесі розшифрування отримане повідомлення M є коректним і придатним для подальшої обробки, то вважається, що дані не були змінені під час передавання, а доступ до них мав лише власник секретного ключа K.

Стійкість алгоритму AES ґрунтується на складності підбору секретного ключа. За сучасного рівня розвитку обчислювальної техніки повний перебір можливих значень ключа AES є неможливим, що забезпечує високий рівень конфіденційності інформації.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Процес впровадження системи мережевої розсилки електронної пошти в промислову експлуатацію передбачає налаштування створеного продукту під умови використання, перевірку його працездатності в реальному середовищі та підготовку користувачів до роботи із системою. Впровадження системи охоплює етапи, які роблять розроблену систему готовою до експлуатації в практичній діяльності підприємства. Цей процес є важливим етапом життєвого циклу будь-якого програмного продукту, тому що саме на цьому етапі перевіряється його якість, зручність інтерфейсу, стабільність та відповідність вимогам.

Розглядаючи розроблену систему мережевої розсилки електронної пошти, методика впровадження включатиме такі етапи, як підготовка та налаштування середовища, встановлення компонентів, тестування, документування, навчання користувачів та передачу в експлуатацію. Проходячи ці етапи, система поступово наближатиметься до робочого стану.

Спершу проводиться аналіз характеристик обладнання, де планується розміщення системи. Визначаються мінімальні вимоги до продуктивності, обсягу пам’яті, швидкодії мережі та обсягу зберігання даних. Важливо забезпечити стабільну роботу сервера та підтримку сучасних стандартів безпеки. Підготовка середовища включатиме встановлення операційної системи та налаштування віртуальної машини або контейнера Docker.

Після встановлення віртуальної машини JVM програму можна запустити двома способами. Перший спосіб, включає завантаження файлу email-sender-0.0.1-SNAPSHOT.jar та виконання команди java -jar target/email-sender-0.0.1-SNAPSHOT.jar у терміналі. Другий спосіб, який буде розглядатися в цьому розділі, вимагає завантаження всього проєкту, при цьому потрібне середовище розробки до прикладу IntelliJ IDEA. Після завантаження

проєкту та середовища розробки потрібно відкрити файл EmailSenderApplication.java, який автоматично відкриє середовище розробки в якому для запуску програми потрібно запустити цей же файл EmailSenderApplication.java через команду RUN.

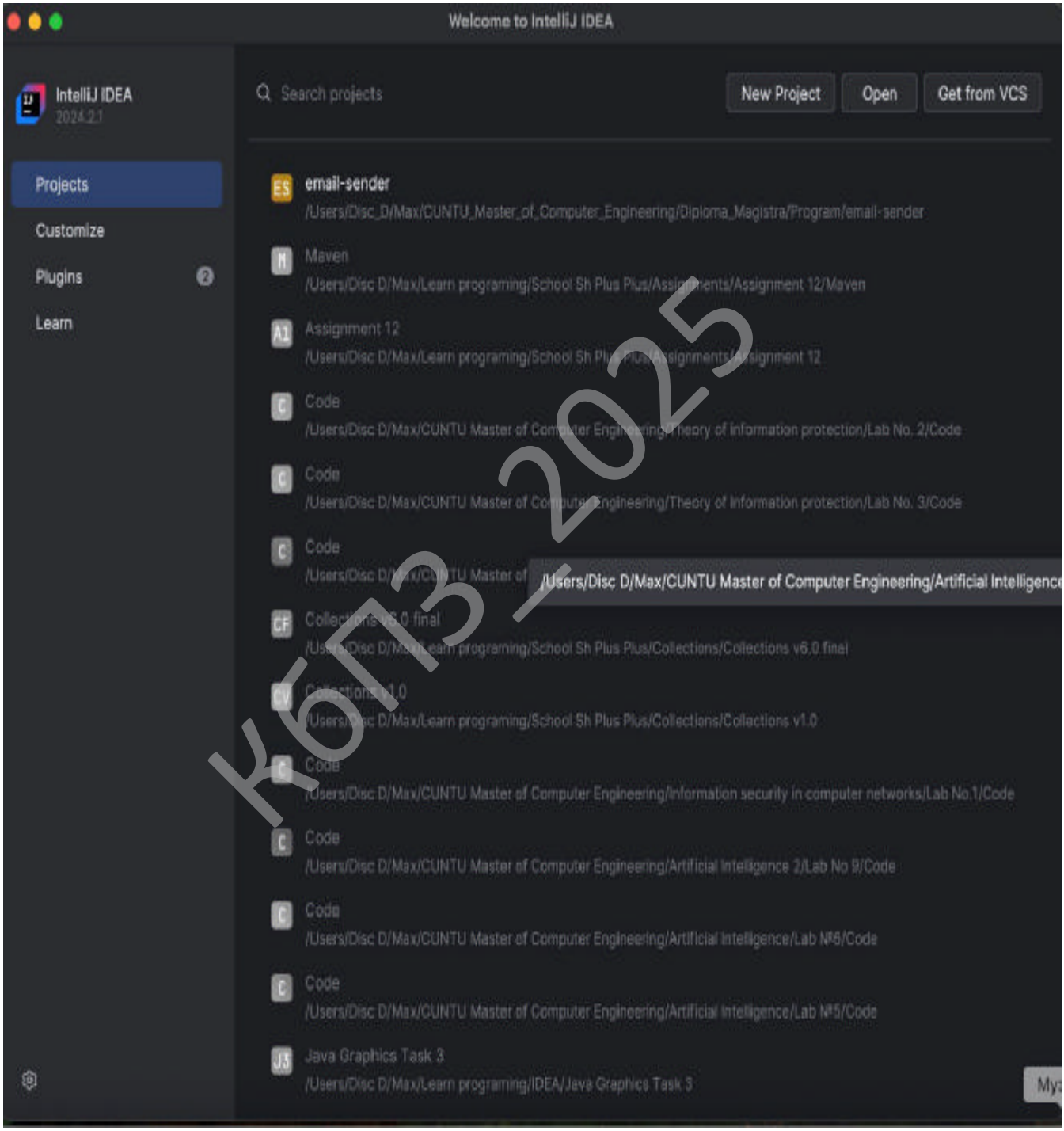

Рисунок 5.1 – Вікно запуску проєктів IntelliJ IDEA

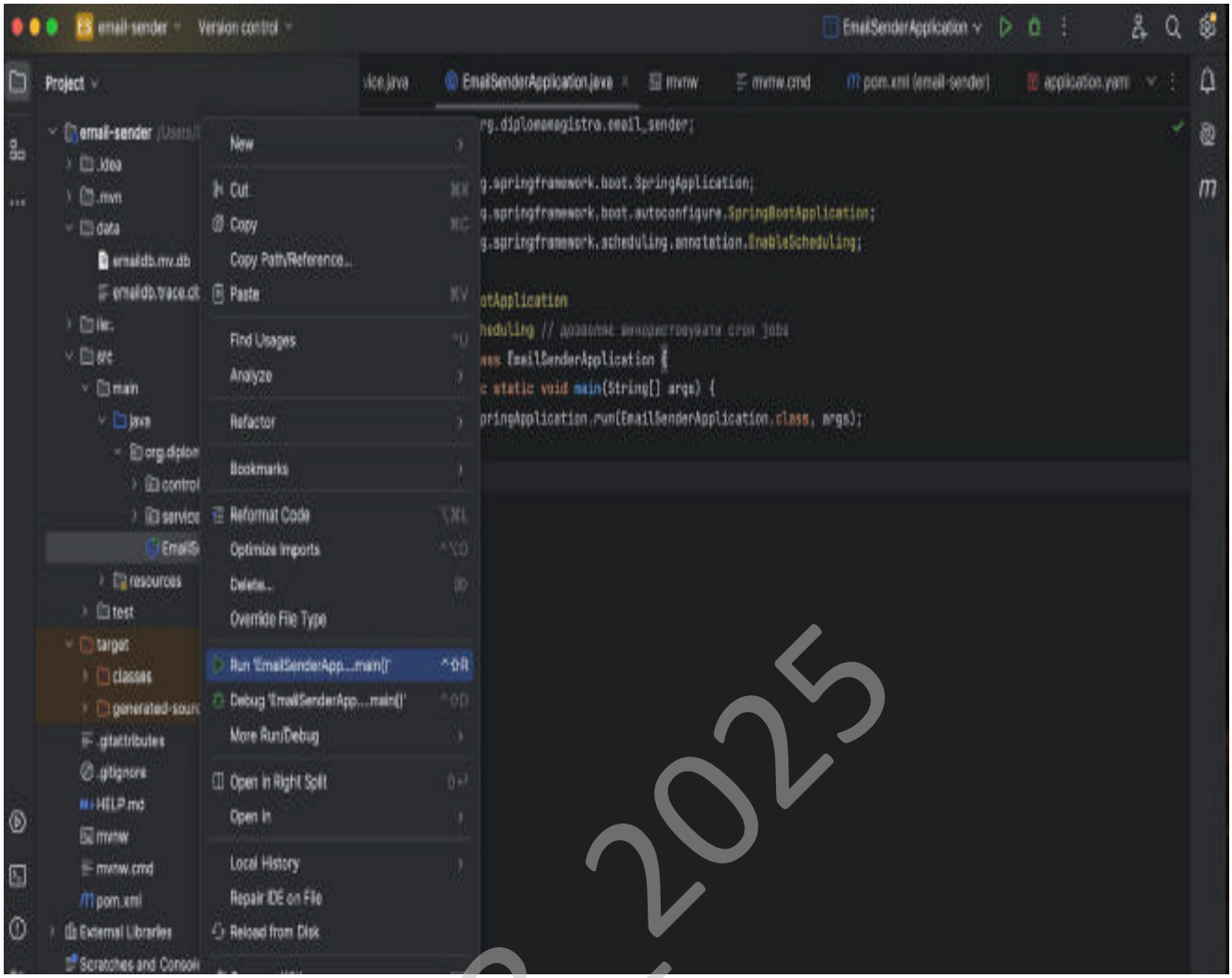

Рисунок 5.2 – Вікно запуску системи

З запуском системи сформується база даних H2, яка використовується для зберігання облікових записів користувачів, завдань розсилок та логів для виведення статистики. Цей етап не потребує додаткового налаштування, оскільки база створюється під час першого запуску програми, але налаштування можливі за допомогою файлу application.yaml.

У конфігураційному файлі application.yml можливо налаштувати файлову базу даних, автоматичне оновлення схеми бази даних за допомогою Hibernate, вебконсоль для перегляду даних, SMTP-сервер Gmail для розсилки електронної пошти, захищене HTTPS-з'єднання з використанням TLS і власного сертифіката та багато іншого.

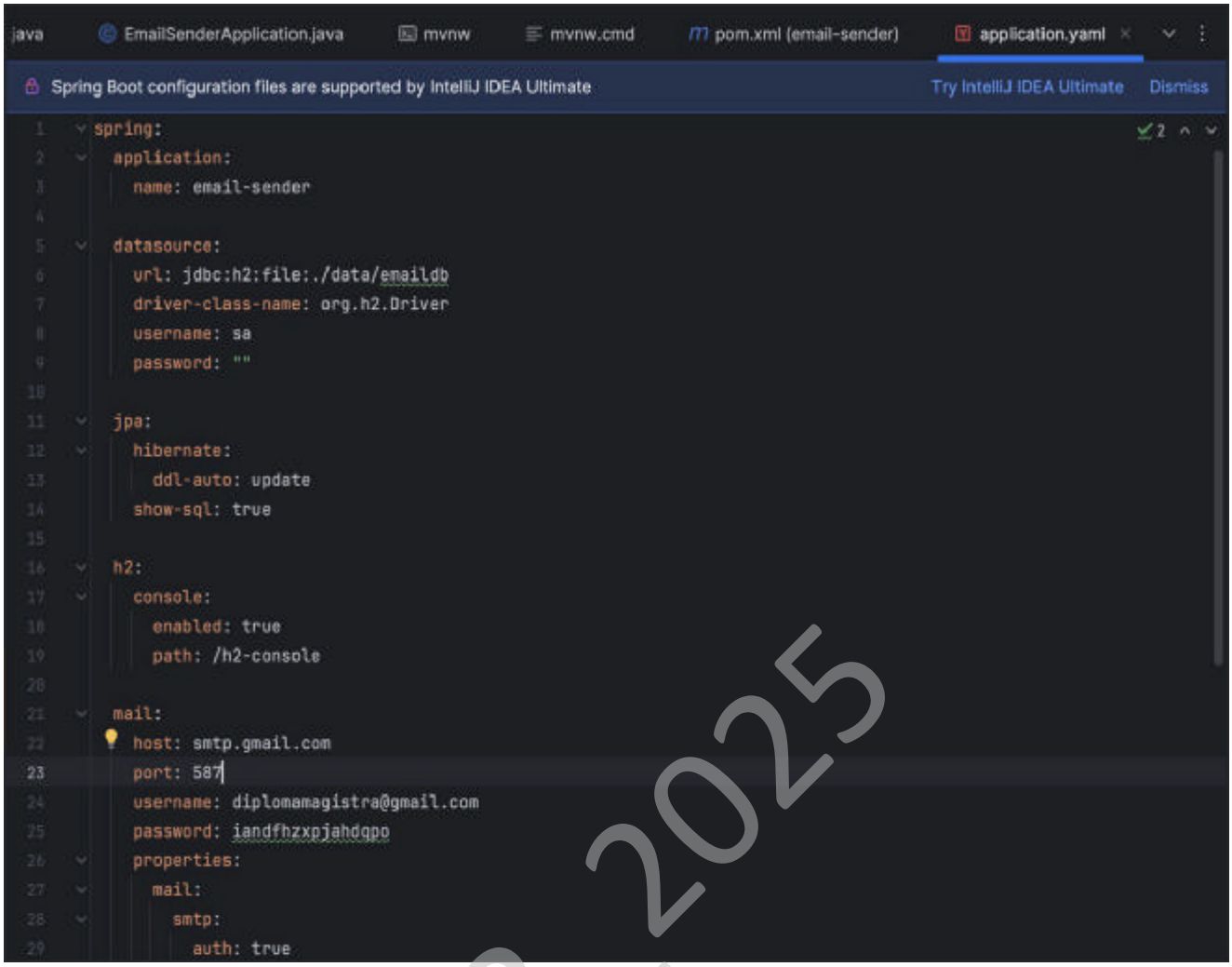

Рисунок 5.3 – Перша частина конфігураційного файлу application.yml

У системі використовується протокол SMTP для взаємодії з сервером Gmail, що забезпечує своєючергою використання захищеного з’єднання через протокол TLS. Система вже налаштована та підключена до сервера Gmail за допомогою двоетапної автентифікації та паролів додатків для автоматичної взаємодії з сервером. Це дозволяє гарантувати безпечне та стабільне відправлення повідомлень.

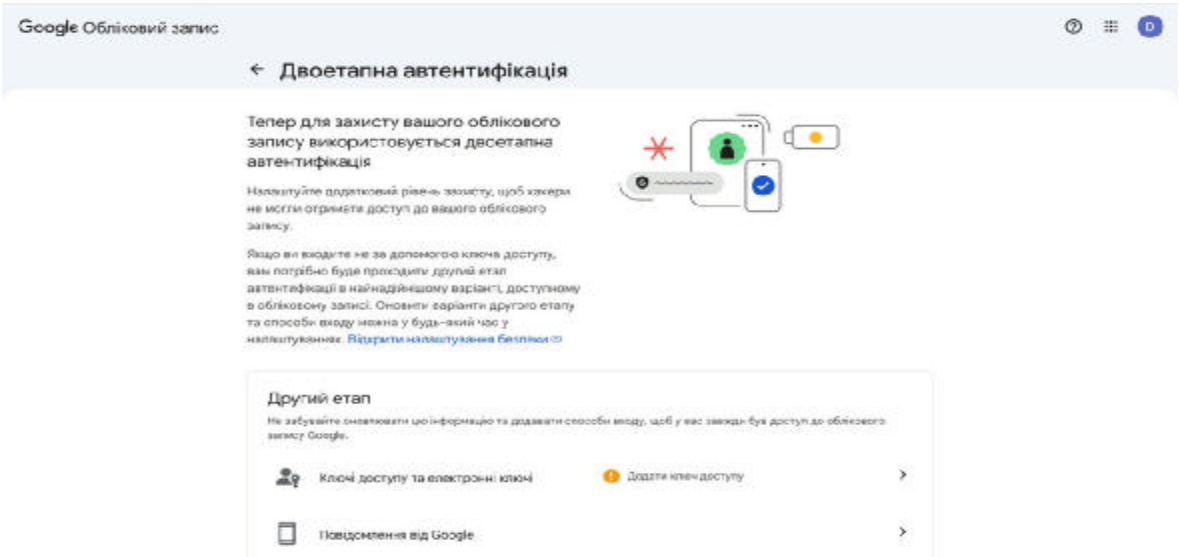

Рисунок 5.4 – Підключення двоетапної автентифікації

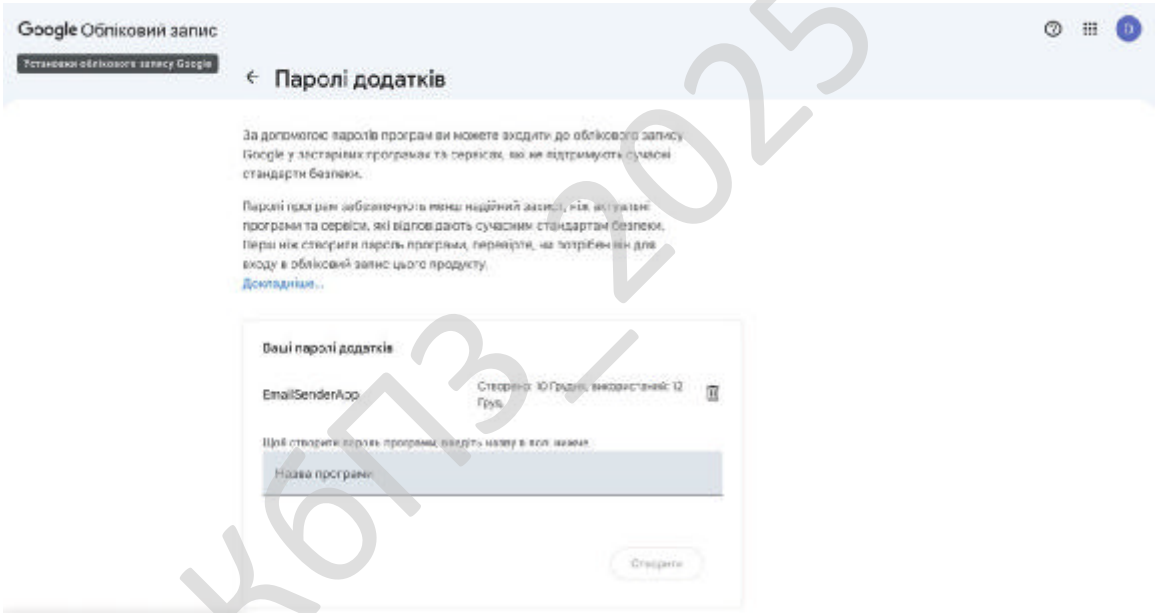

Рисунок 5.5 – Створення паролів додатків

Після того як програма запущена, а з'єднання з мережею стабільне для роботи, необхідно відкрити браузер та перейти за посиланням https://localhost:8443/, де з'явиться головне вікно системи. Створення користувачів, редагування, видалення та перегляд це перше що потрібно зробити та протестувати на правильність роботи. Для цього створюється, редагується переглядається та видаляться тестовий користувач.

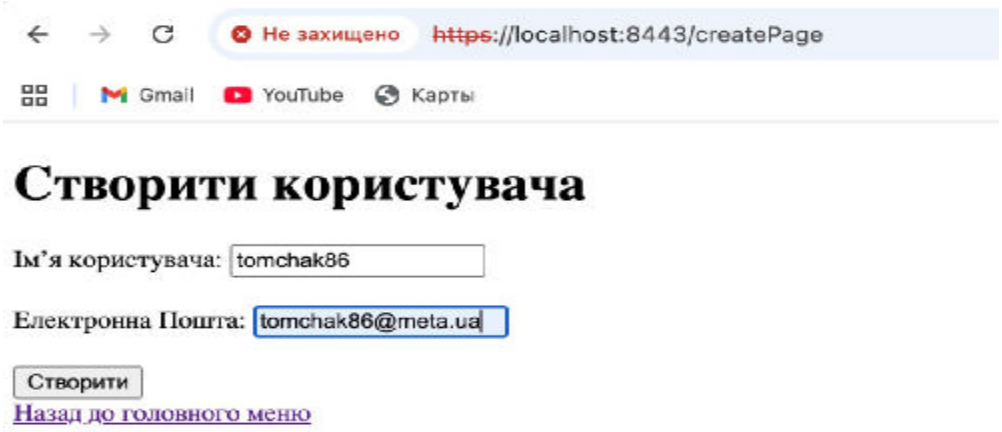

Рисунок 5.6 – Створення користувача в системі

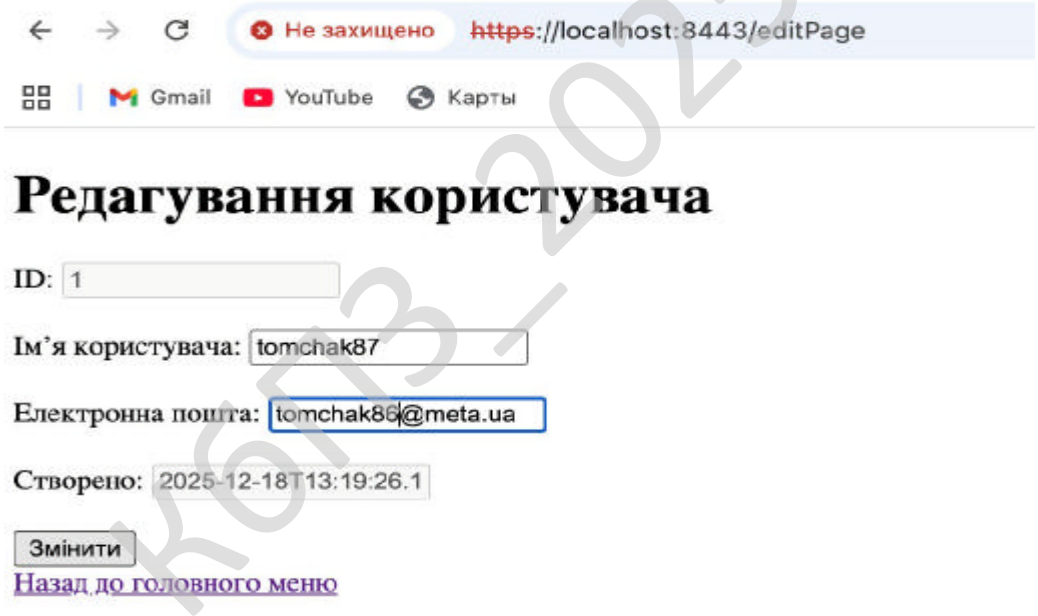

Рисунок 5.7 – Редагування користувача в системі

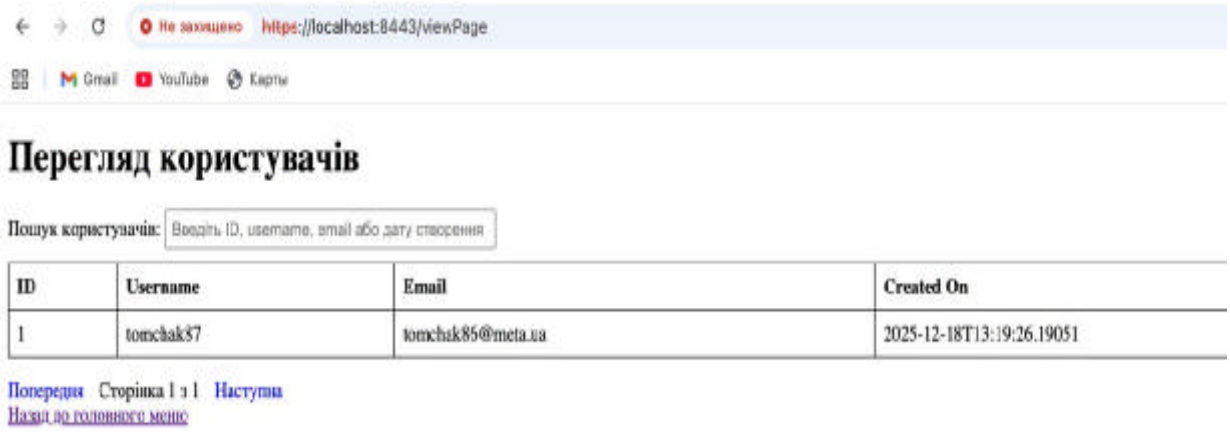

Рисунок 5.8– Перегляд користувача в системі

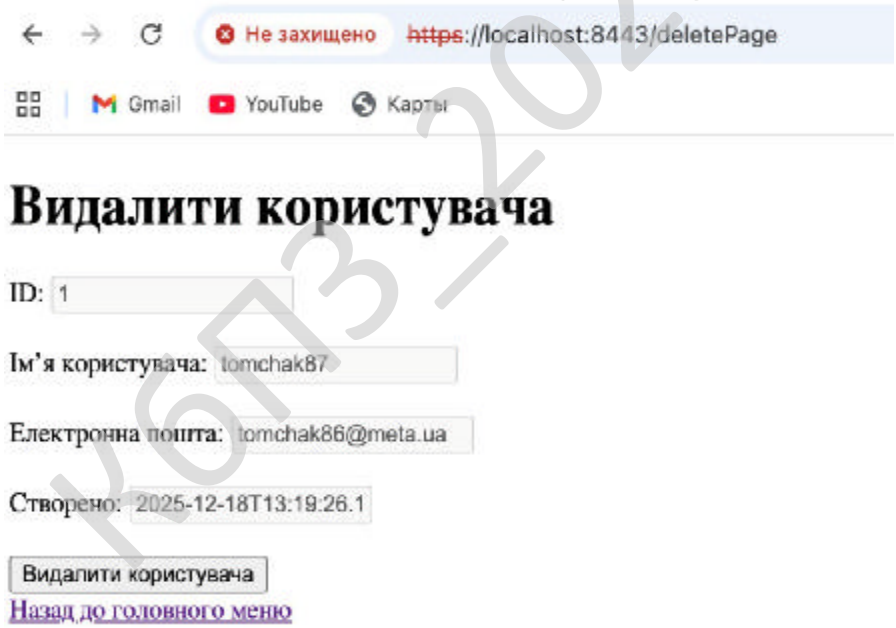

Рисунок 5.9 – Видалення користувача в системі

Далі проводяться тесові надсилання спочатку конкретному користувачу. Тоді автоматично через cron-завдання в запланований час. Це дозволяє системі самостійно виконувати розсилку за заданим графіком, без участі користувача. Для контролю роботи cron-процесів створюється журнал, який записує час

відправлення, ім’я та пошту користувача та тип відправлення. Це допомагатиме переглядати статистику відправлень та аналізувати ефективність роботи системи.

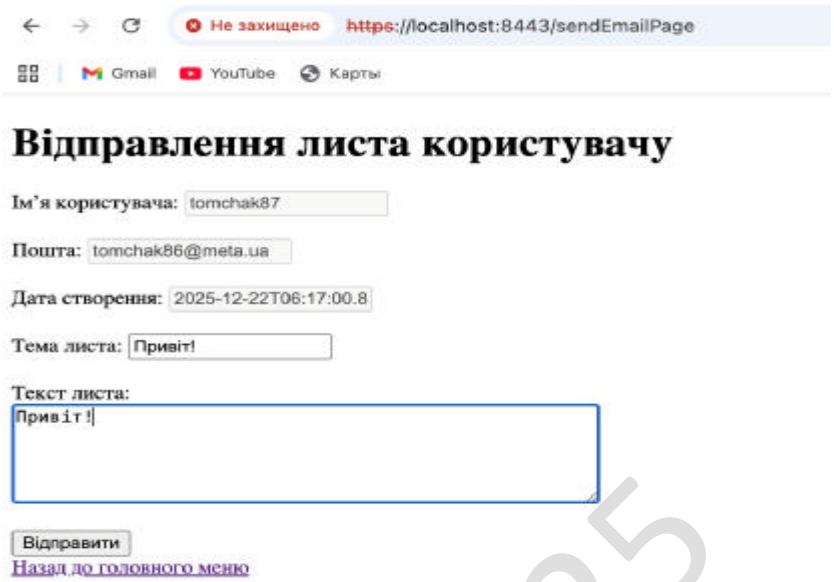

Рисунок 5.10 – Відправка листа користувачу

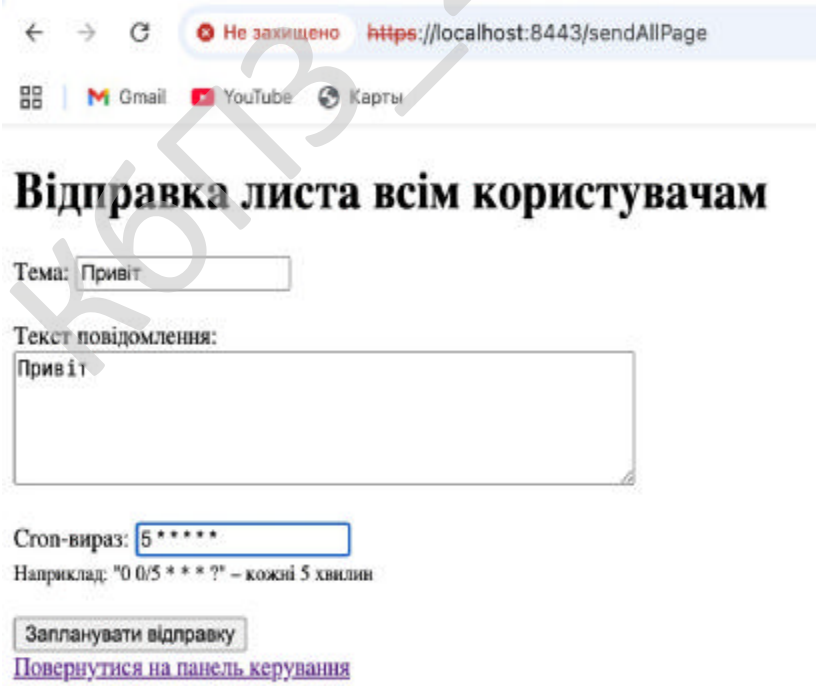

Рисунок 5.11 – Відправка листа всім користувачам за розкладом

В той час коли якась кількість листів надіслана різними способами має сенс перевірити, що статистика відправлень теж працює.

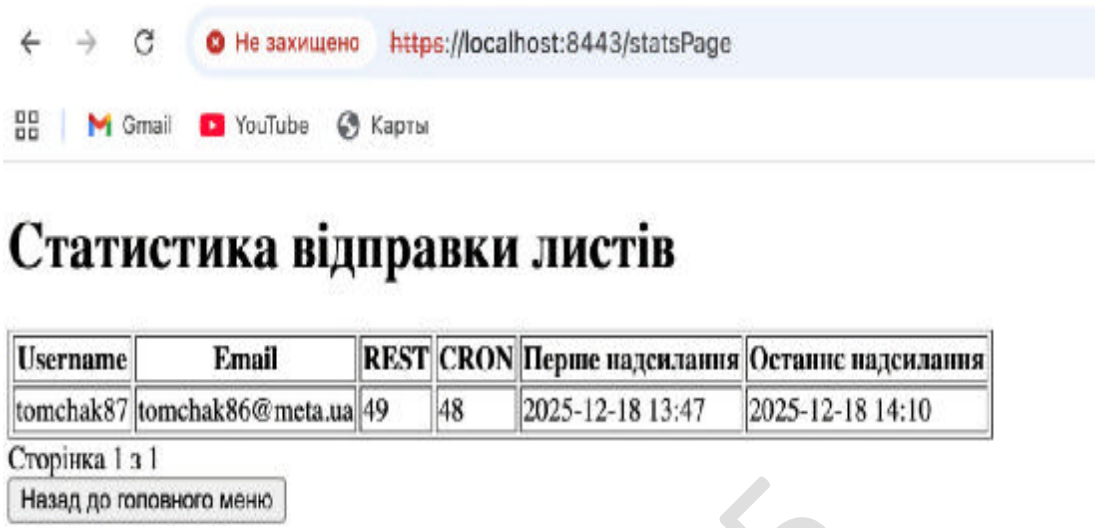

Рисунок 5.12 – Статистика відправлень системи

Після базових тестувань, створюється фокус група яка буде користуватися системою в тестовому режимі. На цьому етапі перевірятиметься, що система функціонує стабільно, запити обробляються коректно, дані передаються без втрат, а користувач отримує передбачений результат. Особливу увагу потрібно приділити перевірці сценаріїв із помилковими вхідними даними, щоб переконатися, що система правильно реагує на непередбачені ситуації. Протягом певного часу збирається зворотний зв'язок від користувачів, оцінюється зручність інтерфейсу, швидкість роботи та точність виконання завдань. У разі виявлення помилок чи неточностей вносяться необхідні зміни, оновлюються модулі та оптимізується код.

Паралельно оновлюватиметься документація, яка міститиме інструкції з інсталяції, налаштування, експлуатації системи, а також опис основних функцій і можливих ситуацій, у яких може знадобитися технічна підтримка. За допомогою цієї документації передбачається проводити навчання користувачів. Їм демонструватимуть основні можливості системи, порядок входу в систему, створення розсилок та перегляд журналу відправлень. У процесі навчання користувачі отримують практичні навички роботи з програмою, що дає змогу розпочати використання системи без додаткових труднощів.

Після перелічених етапів проводиться підсумкова перевірка готовності, яка включає контроль усіх налаштувань, оцінку продуктивності та безпеки. Якщо всі критерії відповідають поставленим завданням, систему офіційно можна вважати введеною в експлуатацію. У ході експлуатації здійснюватиметься технічна підтримка, оновлення компонентів, усунення виявлених помилок і додавання нових функцій за потреби. Це дозволятиме підтримувати систему в актуальному стані, підвищувати її стабільність і відповідність сучасним вимогам.

Таким чином, методика впровадження системи мережевої розсилки електронної пошти передбачає комплексну підготовку програмного забезпечення до роботи, у результаті якої система стає повністю готовою до використання, забезпечуючи безпечну, автоматизовану та ефективну розсилку електронних повідомлень.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

Метою розробки є дослідження та програмна реалізація системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

Об'єктом дослідження є процес мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

Предметом дослідження є методи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних.

– Розроблено вітчизняний продукт мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних, який має більш широкі можливості, на відміну від існуючих аналогів.

Наукова новизна розробленої системи полягає у створенні комплексного підходу до безпеки в системі мережевої розсилки електронної пошти. Сучасні поштові платформи вже давно використовують протоколи шифрування та базові засоби захисту даних, але переважно один або два рівні безпеки. У розробленій системі поєднано три незалежні механізми. Це поєднання створює цілісну трирівневу модель захисту що в свою чергу дозволяє одночасно забезпечити

захист каналу передачі даних, захист збереженої інформації користувачів та контроль цілісності логів.

У більшості сучасних корпоративних систем поштових розсилок основна увага приділяється саме захисту передавання даних за допомогою транспортного шифрування TLS та контролю доступу через API-ключі. Такий підхід є непоганим безпековим рішенням, але він не робить систему повністю захищеною. У розробленій системі також застосовується TLS як базовий механізм шифрування даних під час передачі запитів між клієнтською і серверною частинами. Це відповідає сучасним стандартам і забезпечує захист від перехоплення чи несанкціонованого втручання. В той самий час система передбачає розширення цього рівня безпеки.

Другим важливим елементом є застосування алгоритму AES для шифрування даних користувачів у базі даних, що забезпує конфіденційність навіть у разі, якщо зловмисник отримає фізичний доступ до бази. Це створює додатковий рівень захисту і дозволяє відповідати сучасним вимогам до збереження персональних даних.

Третій рівень безпеки вибудовується за рахунок використання алгоритму SHA-256 для контролю цілісності логів. Це рішення є особливо важливим, оскільки в більшості систем розсилок логування розглядається лише як технічна функція, а не як об’єкт захисту. Використання хешування для перевірки логів дозволяє гарантувати, що записи не були змінені після створення. Таким чином, адміністратор може мати повну впевненість у достовірності подій.

У практичному сенсі наукова новизна проявляється у створенні системи, яка поєднує в собі принципи безпеки з реальними задачами автоматизації. Важливо також відзначити, що розроблена система дозволяє збирати і аналізувати дані без порушення принципів конфіденційності. Це робить систему придатною для роботи з великими обсягами інформації та аналітичних завдань.

Таким чином, наукова новизна полягає не лише у використанні відомих криптографічних алгоритмів, а в тому, як вони поєднані, узгоджені між собою і

впроваджені в архітектуру системи. Розроблене рішення демонструє можливість створення безпечної поштової системи нового покоління, у якій безпека не є окремим компонентом, а є невід'ємною частиною всієї структури. Такий підхід можна вважати основою для подальших досліджень у сфері комплексного захисту комунікацій.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати цього дослідження можуть стати корисними для організацій, які активно працюють із розсилками: маркетинговими, інформаційними або для внутрішньої комунікації. Особливо актуальними вони будуть для тих, хто вже стикався з проблемами безпеки або шукає ефективні способи її посилення.

Окрему групу потенційних користувачів становить корпоративний сектор. Для малих та великих організацій важливим є управління та контроль вхідної й вихідної кореспонденції. Це дозволить автоматизувати бізнес-процеси, зменшити ризики людських помилок і запобігти використанню працівниками особистих поштових акаунтів у робочих цілях.

Цим проєктом також можуть зацікавитися стартапи, що працюють у сфері email-маркетингу. Їм часто бракує якісних рішень із захистом на рівні каналу зв’язку, контенту та доступу до даних. Модель, яка пропонує трирівневий захист, виглядає переконливо та має конкурентні переваги на ринку, де більшість рішень зосереджені на доставленні, а не на безпеці самих даних.

Суттєвий інтерес до системи може проявити й сфера електронної комерції. Інтернет-магазини щодня генерують значну кількість листів для підтвердження замовлень, інформації про оплату, зміни статусу доставки, акційних повідомлень. Для цього сегмента особливо важливими є стабільність, масштабованість та гарантована безпечна доставка електронних листів, адже будь-які збої безпосередньо впливають на якість сервісу та рівень довіри покупців.

Розробка буде цікавою також для державних установ, освітніх закладів чи медичних центрів. У цих сферах обмін електронними листами часто містить персональні або чутливі дані, і порушення безпеки може мати серйозні юридичні

й фінансові наслідки. З огляду на тенденцію до цифровізації державних послуг попит на захищені системи електронних повідомлень лише зростає.

Крім того, система може бути корисною для внутрішнього використання у компаніях, які хочуть оптимізувати власні інструменти розсилки, не передаючи дані стороннім сервісам. Власне програмне забезпечення дає більше контролю.

Також окремим ІТ-фахівцям, які займаються питаннями кіберзахисту, буде цікаво оцінити, як на практиці реалізована ідея розсилки з багаторівневим захистом, для застосування як прототипу або бази для подальших розробок.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості ідеї впровадження даної системи застосовано метод експертних оцінок. Важливим було отримати зворотний зв’язок не лише від викладачів, а й від фахівців, які працюють у сферах ІТ, маркетингу та кібербезпеки.

Було проведено невелике опитування серед семи експертів із досвідом у сфері email-технологій. Їм були запропоновані ключові параметри для оцінки: рівень інноваційності, масштабованість, потенціал комерціалізації, простота реалізації та значимість захисту даних.

Кожен із експертів оцінював параметри за шкалою від 1 до 10. На основі отриманих оцінок було розраховано середні бали, що дозволило визначити загальну привабливість проєкту – вона склала 8,3 із 10. Найвищі оцінки отримали безпекові аспекти та актуальність системи розсилки.

Застосування цього підходу дало можливість отримати аналіз його сильних сторін. Результати підтверджують, що проєкт не лише технічно важливий, а й представляє реальний інтерес для ринку.

7.3 Вибір методу оцінки вартості ПЗ

Для аналізу вартості розробки та реалізації даного проєкту застосовано метод витрат, або так званий “cost-based approach”. Він дозволив врахувати всі прямі та непрямі витрати на етапах розробки, тестування, впровадження та супроводу системи.

Застосування даного підходу є доцільним, тому що проєкт поки що не має повноцінної комерційної реалізації і не приносить доходів, на які можна було б спиратися при оцінці ринкової вартості. Оцінювання потенційної ринкової ціни без наявної бази могло б бути ризикованим.

Було враховано витрати на хостинг, робочий час програміста, можливі витрати на дизайн, інтеграцію SMTP-серверів та сертифікатів шифрування. Отримана вартість формує фундамент для подальшого визначення цінової стратегії.

Разом з цим, метод витрат дозволяє побачити варіанти для економії та оптимізації ресурсів без зниження якості захисту та функціональності системи.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Хмарні сервіси масової електронної розсилки зазвичай використовують сторонні інструменти, які або мають обмежені механізми безпеки, або значно підвищують вартість при підключенні розширеного функціоналу захисту даних. Це створює низку потенційних ризиків, таких як витік персональної інформації через недостатній рівень авторизації, перехоплення контенту листів у разі відсутності шифрування, компрометація логіки доставки, відсутність контролю за внутрішнім доступом до бази розсилки, а також додаткові витрати на сторонні сервіси при масштабуванні.

Впровадження власної системи масової розсилки електронної пошти з трирівневою системою захисту дозволяє скоротити витрати на використання сторонніх сервісів, знизити ймовірність витоку або спотворення даних, прискорити процес доставки критичних повідомлень, централізовано адмініструвати списки розсилки, покращити репутацію домену розсилки завдяки точному контролю SPF, DKIM та DMARC, а також зменшити ризик санкцій і штрафів за порушення регламентів захисту даних. Вхідні дані зафіксовано в таблиці 7.1.

Розрахунок економічного ефекту демонструє наступне:

– економія на відмові від зовнішнього сервісу розсилки 420 000 грн/рік;

– економія на безпекових надбудовах 150 000 грн/рік;

– зменшення потенційних ризикових витрат 65 000 грн/рік;

– економія на трудових витратах 85 год × 350 грн = 29 750 грн/рік;

– загальна щорічна економія 664 750 грн/рік;

– чистий економічний ефект у перший рік (з урахуванням початкових інвестицій) 664 750 – 320 000 = 344 750 грн;

– термін окупності ≈ 0,48 року (менше ніж шість місяців);

– рентабельність інвестицій (ROI) ≈ 108 %;

Додаткові нефінансові переваги:

– підвищення рівня інформаційної безпеки;

– автономність;

– покращення доставлення;

– зниження рівня відмов;

– можливість подальшої комерціалізації рішення;

Таблиця 7.1 – Вихідні дані для розрахунку

Показник	**До впровадження**	**Після впровадження**	**Економічний ефект**
Вартість використання зовнішнього сервісу розсилки (щорічно), грн	420 000	–	-420 000 грн
Вартість витрат на безпекові плагіни та ліцензії (щорічно), грн	150 000	–	-150 000 грн
Час відправки повної розсилки (для 100 тис. адрес), хв	45	12	-33 хв
Рівень доставлення (delivery rate), %	85%	97%	+12 п.п.
Потенційні штрафи за витік/недоставку, грн/рік	80 000	15 000	-65 000 грн
Адміністративні витрати на налаштування/супровід, людино-год/рік	180	95	-85 годин
Вартість 1 людино-години адміністратора, грн	350	350	–
Початкові інвестиції у розробку та впровадження системи, грн	–	–	320 000 грн

Впровадження власної системи захищеної e-mail розсилки дозволяє скоротити витрати на понад 660 тис. грн щороку, окупитися менш ніж за півроку та забезпечити високий рівень контролю, надійності та масштабованості. Це робить розроблене ПЗ економічно привабливим, конкурентоспроможним і практично доцільним для широкого застосування як в бізнесі, так і в публічному секторі.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Просування проєкту передбачається через цільову презентацію на платформах для стартапів та технологічних хабах. Це не лише технічна розробка, а і рішення для реальних проблем, тому позиціонування має підкреслювати його практичну цінність і користь для бізнесу.

Першим кроком може бути створення короткого демо-відео, у якому просто і зрозуміло показується трирівневий захист, навіщо він потрібен і як працює система. Користувачі наочно зможуть зрозуміти практичну користь системи.

Далі варто презентувати продукт на хакатонах, освітніх заходах, бізнес колах та інших заходах де активно ведеться e-mail маркетинг . Це можуть бути невеликі конференції, круглі столи, навіть вузівські виступи для майбутніх фахівців.

Також доцільно публікувати інформацію на технічних форумах і платформах, таких як GitHub, ProductHunt або Stack Overflow. Це дозволяє отримати зворотний зв’язок, пропозиції щодо вдосконалення продукту та можливості співпраці.

Використання професійних спільнот у LinkedIn, спеціалізованих Telegram-каналах та тематичних групах у Facebook не буде зайвим. Реклама в таких каналах повинна бути короткою та зрозумілою з акцентом на безпеці та ефективності розсилок. Додатково можна залучати експертів та блогерів у сфері

ІТ-безпеки для рецензій та статей. Це підвищує авторитетність системи та допомагає формувати позитивну репутацію на ринку.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для таких цифрових проєктів важливо з самого початку визначити оптимальні канали збуту. Одним із найбільш доцільних рішень є запуск системи як SaaS-платформи, яку можна легко інтегрувати без складних налаштувань.

Додаткову цінність може дати розробка розширень для популярних CRM-систем та платформ електронних розсилок. Це дасть змогу користувачам підключати продукт без необхідності змінювати наявні робочі системи.

Іншим варіантом є застосування моделі white-label, за якої компанії можуть адаптувати систему під власний бренд. Це розширює потенційний ринок і забезпечує додаткові джерела доходу.

Для залучення невеликих компаній, освітніх установ та організацій із обмеженим бюджетом потрібно запровадити безкоштовний тарифний план з обмеженою кількістю відправок, але з усіма рівнями захисту. Такий підхід збільшить охоплення і створить умови для подальшого переходу на повний тариф.

До обов’язкових елементів можна віднести й створення презентаційної сторінки з чітко описаними перевагами, практичними кейсами та відгуками. Демонстрація прикладів використання підвищить розуміння та зацікавлення потенційних користувачів.

7.7 Визначення ключових факторів успіху конкретного проєкту

Перше, що впливає на успіх цього проєкту – це актуальність проблеми, яку він вирішує. Кібератаки стають частішими, а безпека електронної пошти

залишається одним із слабких місць багатьох організацій. Тобто попит на подібні рішення поступово зростає.

Друге – це проста та зрозуміла реалізація. Система має бути доступною не лише для технічних фахівців, а й для маркетологів, менеджерів та керівників, які очікують безперебійної роботи без надмірної складності у використанні.

Третій фактор – це формування довіри. Якщо користувачі бачать, що система забезпечує реальний рівень захисту завдяки механізмам шифрування, авторизації та ведення журналів подій, вони будуть готові її використовувати. Тут важливу роль відіграють якісна документація, демонстраційні матеріали та приклади практичного застосування. Ще одним моментом є гнучкість. Якщо продукт легко адаптувати до різних задач, галузей і мов, то й аудиторія буде значно ширшою, а шанси на комерційний успіх будуть вищими.

Та головним чинником є швидка підтримка та розвиток. У сфері безпеки неможливо залишатися на одному місці, тому регулярні оновлення, активність у спільнотах і швидка реакція на зворотний зв’язок формують довіру та бажання інвестувати або хоча б використовувати систему на постійній основі.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

У сфері охорони праці основні правила визначає Закон України «Про охорону праці», який встановлює загальні вимоги до безпечних умов роботи. Детальніше ці положення розкриває Наказ Міністерства соціальної політики №207 від 14.02.2018, який затверджує «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Для оцінки умов праці також важливі галузеві норми, такі як ДБН В.2.5-28:2018, що встановлює вимоги до природного та штучного освітлення приміщень, та ДСН 3.3.6.042-99, що визначає допустимі параметри мікроклімату.

Щоб оцінити, які умови можуть призводити до професійних захворювань, слід орієнтуватися на чинні нормативні документи: Закон України «Про охорону праці», Наказ №207/2018, ДБН В.2.5-28:2018, ДСН 3.3.6.042-99.

Під час роботи з комп'ютером можуть діяти такі шкідливі та небезпечні фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території;

– пожежна небезпека;

– ризик ураження електричним струмом;

– надмірне навантаження на зір;

– не правильна освітленість робочого місця;

– несприятливі умови мікроклімату;

– електромагнітні випромінювання, у тому числі високочастотні;

– нервово-емоційна напруженість та інтелектуальні перевантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– статичні навантаження на опорно-руховий апарат;

– монотонність роботи;

– шум.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Для дослідження санітарно-гігієнічних умов праці необхідно визначити, чи відповідає робоче середовище встановленим нормам безпечної та комфортної роботи. В офісному приміщенні оцінюють площу та об’єм, які припадають на одного працівника, а також параметри мікроклімату, рівень шуму, освітленість робочих місць, джерела тепла та пил.

У таблиці 8.1 наведені фактичні геометричні параметри приміщення, у якому одночасно працюють семеро програмістів. У таблиці 8.2 подано розрахунок площі та об’єму в розрахунку на одного працівника, а також нормативні значення, з якими далі порівнюються фактичні показники.

Таблиця 8.1 – Розміри приміщення

НАЙМЕНУВАННЯ	ЗНАЧЕННЯ, М
Ширина	5
Довжина	9
Висота	3.3

$S_{загальна} = (5 \times 9) = 45м^2$

$S_{на\ людину} = 45 / 7 = 6,43м^2$

$V_{загальний} = 5 \times 9 \times 3.3 = 148,5м^3$

$V_{на\ людину} = 148,5 / 7 = 21,21м^3$

Таблиця 8.2 – Площа та обсяги приміщення на одного працюючого

ГЕОМЕТРИЧНА ХАРАКТЕРИСТИКА	ОДИНИЦЯ ВИМІРУ	НОРМАТИВНЕ ЗНАЧЕННЯ	ФАКТИЧНЕ ЗНАЧЕННЯ
Площа, S	$м^2$	Не менше 6,0	6,43
Об’єм, V	$м^3$	Не менше 20,0	21,21

Відповідно до даних, наведених у таблицях 8.1 та 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце відповідають нормативним вимогам. Також вони узгоджуються з положеннями Наказу Міністерства соціальної політики України № 207 від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», які визначають параметри безпечної та зручної організації робочої зони. Розмір приміщення забезпечує достатній простір для розміщення обладнання, свободу рухів працівника та можливість дотримання ергономічних вимог під час виконання роботи.

Температура повітря в приміщенні формується під впливом зовнішніх погодних умов, тепловиділення від обладнання та людей, а також роботи системи опалення. У теплий період, коли зовнішня температура ще не дуже низька, у розрахунок береться тепло від комп’ютерної техніки, офісного обладнання, освітлення та від самих працівників. У холодний період основним джерелом підтримання нормативної температури є опалювальні прилади, проте також враховуються виділення тепла від техніки, освітлення та людей. У денний час додаткове надходження тепла забезпечує сонячне випромінювання, що проникає через вікна.

Згідно з чинними вимогами Постанови № 42 від 01.12.1999 (ДСН 3.3.6.042-99), роботи, що виконуються у даному офісному приміщенні, належать до категорії Іа. В цій категорії людина витрачає енергії до 120 ккал на годину. Для таких умов повинні підтримуватися оптимальні параметри температури,

вологості та швидкості руху повітря, оскільки працівники тривалий час перебувають у статичній позі й чутливо реагують на відхилення мікроклімату.

Аналіз мікроклімату в приміщенні здійснюється шляхом оцінки взаємодії різних показників та їх порівняння з вимогами Постанови № 42 від 01.12.1999 (ДСН 3.3.6.042-99). Вимірюють температуру, відносну вологість, рух повітря, інтенсивність теплового випромінювання, запиленість та можливу загазованість повітря, оскільки надлишок цих факторів може підвищувати втому, знижувати концентрацію уваги та сприяти розвитку професійних захворювань. Оцінка цих показників дозволяє визначити рівень комфорту та безпеки робочої зони, а також вплив мікроклімату на самопочуття та працездатність працівників.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату для встановленої категорії робіт.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

ПОРА РОКУ	ОПТИМАЛЬНІ			ФАКТИЧНІ		
	Температура, °С	Вологість, %	Шв. повітря, м/с	Температура, °С	Вологість, %	Шв. повітря, м/с
Холодна	22-24	40-60	0,1	21-22	50-60	0,1
Тепла	23-25	50-70	0,1	23-24	60-70	0,1

Проведений аналіз умов праці показує, що в цілому мікроклімат приміщення відповідає рекомендованим нормам. У холодний період року підтримання комфортної температури забезпечує система опалення та примусова вентиляція, проте іноді температура може бути на 1 °С нижче рекомендованого рівня. У теплий сезон застосовується кондиціонування повітря та природне або примусове вентилювання. Для підтримки чистоти повітря та зменшення запилення регулярно проводяться провітрювання та вологе прибирання приміщення.

У приміщенні присутні джерела шуму, такі як вентилятори комп'ютерної техніки, система кондиціонування, вхідні та вихідні отвори системи примусової вентиляції та багатофункціональний пристрій. Проте рівень шуму залишається в межах допустимих норм і не впливає на працездатність та концентрацію співробітників.

8.3 Розробка заходів з умов поліпшення охорони праці

Проведений аналіз умов праці в розглянутому офісному приміщенні дозволяє оцінити безпечність та комфортність робочого середовища та надати деякі пропозиції що до поліпшення. Оцінка площі та об'єму у розрахунку на одного працівника показала достатньо місця для комфортного розміщення робочих місць, обладнання та техніки, що дозволяє співробітникам вільно пересуватися та дотримуватися ергономічних вимог. У цьому секторі можна рекомендувати збільшення площі, проте лише для організації зони відпочинку, що сприятиме кращій стимуляції та відновленню нервової системи співробітників.

Мікрокліматичні умови оцінювалися за параметрами температури повітря, відносної вологості, швидкості руху повітря, інтенсивності теплового випромінювання та чистоти повітряного середовища. Встановлено, що температура та інші показники в приміщенні загалом перебувають у комфортних межах, проте у холодний період температура інколи може бути на 1 °C нижче рекомендованих значень. У зв'язку з цим рекомендовано перевірити термостат або стан системи опалення загалом, а за потреби розглянути встановлення багатозонального термостатного обладнання для більш точного підтримання комфортної температури.

Основними джерелами шуму в приміщенні є вентилятори комп'ютерної техніки, система кондиціонування, вентиляційні канали та багатофункціональний пристрій. Рівень шуму не перевищує нормативних меж і не впливає на

працездатність, концентрацію уваги та загальне самопочуття співробітників. Додатковою рекомендацією для цього сектору може стати впровадження системи аудіостимуляції, наприклад прослуховування спеціальної музики Brainwave, що сприяє підвищенню концентрації та продуктивності роботи.

З огляду на зазначені умови, основними чинниками, які потенційно можуть знижувати ефективність праці програміста, є психофізіологічні фактори: перевтома, стрес та недостатній відпочинок. Тому заходами для підтримки продуктивності та збереження здоров'я є створення позитивної психологічної атмосфери в колективі, а також впровадження додаткових технік організації робочого часу та відпочинку, таких як методика «Помодоро».

Основною рекомендацією з комплексу заходів безпеки є регулярне ознайомлення персоналу з маршрутами евакуації та дотримання плану евакуації. Рекомендується включити до колективного договору мінімальний набір медичних засобів першої допомоги. Необхідна періодична перевірка та вимірювання параметрів заземлення й занулення всіх електроприладів, оскільки ураження електричним струмом може мати серйозні наслідки, включно з фібриляцією шлуночків серця. Для підвищення безпеки доцільно мати у приміщенні дефібрилятор та персонал, навчений його використанню.

8.4 Техніка безпеки та протипожежна профілактика

Навіть офісна діяльність містить низку потенційних небезпек, пов'язаних із використанням електротехнічного обладнання, перебуванням у закритих приміщеннях, дією психофізіологічних факторів, а також можливими пожежними ризиками. Техніка безпеки в офісних приміщеннях спрямована на створення таких умов праці, за яких повністю або максимально усуваються ризики виникнення травм, аварій, професійних захворювань та надзвичайних ситуацій. Дотримання правил безпеки в офісному середовищі є обов'язковим для всіх працівників незалежно від їхньої посади.

Електробезпека є одним з ключових аспектів безпеки офісного приміщення. Робота програміста безпосередньо пов'язана з використанням комп'ютерної, мережевої та офісної техніки, яка працює від електромережі. Неналежна експлуатація електроприладів може призвести до ураження електричним струмом, коротких замикань та перегрівання обладнання.

Усі електроприлади, що використовуються в офісі, мають відповідати класам електрозахисту та бути технічно справними. Персонал не повинен користуватися обладнанням із пошкодженими кабелями, відкритими контактами або дефектами корпусу. Заборонено підключати пристрої до розеток із механічними пошкодженнями або ознаками оплавлення.

До заходів електробезпеки також належить правильне розташування кабельних ліній. Кабелі не повинні перетинати проходи, створювати петлі чи бути натягнутими вздовж підлоги без захисних коробів. Також повинно бути присутнім заземлення та занулення електромережі. Регулярна перевірка цих параметрів є обов'язковою та повинна проводитися спеціалізованими службами. Неправильне або відсутнє заземлення може спричинити небезпечні для життя струмові удари, особливо при роботі з металевими корпусами комп'ютерів.

Системні блоки повинні мати вільний доступ повітря для охолодження, не допускається їх встановлення в закриті ніші без вентиляції. Монітори слід розміщувати на відстані 50–70 см від очей, а освітлення повинно виключати появу відблисків на екрані. Усі пристрої повинні підключатися через справні розетки або сертифіковані мережеві фільтри.

У разі виникнення аварійної ситуації, такої як запах горілого пластику, поява диму, іскріння або сильне нагрівання обладнання, слід негайно вимкнути відповідний пристрій, знеструмити його та повідомити відповідальну особу. Категорично забороняється намагатися розбирати електроприлад самостійно – це повинні виконувати лише кваліфіковані спеціалісти.

Пожежна безпека є базовим елементом. Офісні приміщення містять значну кількість електротехніки, меблів, паперу, кабельних трас та оздоблювальних

матеріалів, що створює ризик займання при неправильній експлуатації або аваріях.

Приміщення повинно бути обладнане вогнегасниками відповідного класу, які мають розміщуватися у легкодоступних місцях. Найбільш придатними для офісу є порошкові та вуглекислотні вогнегасники, що ефективні для гасіння електричних пожеж. Необхідно здійснювати регулярну перевірку їх термінів придатності та сервісного обслуговування.

Евакуаційні виходи мають бути вільними від будь-яких перешкод. Заборонено встановлювати біля виходів меблі, коробки, техніку або інші предмети, які можуть ускладнити евакуацію. На видимих місцях повинні бути розміщені схеми евакуації та інструкції щодо дій у випадку пожежі.

Організаційні заходи є невід'ємною частиною системи техніки безпеки. До них належить проведення регулярних інструктажів, навчань з пожежної безпеки, тренувальних евакуацій та перевірок знань персоналу. Співробітники повинні знати розташування евакуаційних виходів, вогнегасників, аптечки та інших засобів безпеки.

В офісі доцільно мати аптечку першої допомоги, що містить базові медикаменти та перев'язувальні матеріали. Враховуючи ризики ураження електричним струмом, доцільним є також розміщення автоматичного зовнішнього дефібрилятора та навчання персоналу базовим навичкам серцево-легеневої реанімації.

Організаційні заходи також включають ведення журналів технічного огляду обладнання, систематичні перевірки стану електромережі, інвентаризацію вогнегасників та контроль за дотриманням вимог безпеки.

Для підвищення загальної безпеки в офісному приміщенні можна впровадити додаткові заходи:

– встановлення системи автоматичного контролю за температурою обладнання;

– використання протипожежних кабелів та коробів;

– застосування систем раннього виявлення диму;

– проведення регулярних тренінгів для персоналу.

8.5 Розрахункова частина

Розрахунок штучного освітлення.

Завдання: розрахувати штучне освітлення робочого приміщення.

Початкові дані:

Ширина приміщення: A = 5,0 м.

Довжина приміщення: B = 9,0 м.

Висота приміщення: H = 3,3 м.

Висота робочої поверхні від підлоги: 0,8 м.

Розрахункова висота підвісу світильників над робочою поверхнею:

h = 3,3 − 0,8 = 2,5 м.

Площа приміщення: S = 5,0 × 9,0 = 45,0 м².

Нормована освітленість для робочих місць за комп'ютером: E = 300 лк.

Сумарний світловий потік F визначається за методом коефіцієнта використання світлового потоку:

$$F = (E \times S \times K \times Z) / n,$$

де:

K – це коефіцієнт запасу, враховує старіння ламп і забруднення світильників,

Z – це коефіцієнт нерівномірності,

n – це коефіцієнт використання світлового потоку, залежить від індексу приміщення та відбиття стін і стелі, ρ стелі ≈ 50%, ρ стін ≈ 50% (згідно методичним рекомендаціям до виконання розділу "Заходи з охорони праці та техніки безпеки").

Приймаємо (згідно методичним рекомендаціям до виконання розділу "Заходи з охорони праці та техніки безпеки"):

$$K = 1,4.$$

$$Z = 1,1.$$

Індекс приміщення використовується для визначення коефіцієнта n:

$$i = S / (h \times (A + B)) = 45,0 / (2,5 \times (5,0 + 9,0)) = 45,0 / 35,0 \approx 1,29.$$

Для такого індексу і зазначених коефіцієнтів відбиття табличне значення n для типової LED-панелі 0,62 (згідно методичним рекомендаціям до виконання розділу "Заходи з охорони праці та техніки безпеки").

Розрахунок сумарного світлового потоку:

$$E \times S = 300 \times 45 = 13\ 500 \text{ лм}.$$

З урахуванням коефіцієнтів K і Z: $13\ 500 \times 1,4 \times 1,1 = 20\ 790$ лм.

Ділимо на n: $F = 20\ 790 / 0,62 \approx 33\ 532$ лм.

Сумарний необхідний світловий потік: $F \approx 33\ 532$ лм.

Для розрахунку кількості використовуються LED панелі.

600х600мм 40Вт 4000К 6200К IP20.

Світловий потік: $Фℓ = 4\ 000$ лм.

Кількість світильників: $N = F / Фℓ = 33\ 532 / 4\ 000 \approx 8,38$.

Округляємо до 9 світильників.

Обчисляєм фактична середню освітленість за формулою:

$$E \text{ факт} = (N \times Фℓ \times n) / (S \times K \times Z).$$

E факт $= (9 \times 4\ 000 \times 0,62) / (45 \times 1,4 \times 1,1) \approx 322$ лк.

Норма E = 300 лк забезпечена з невеликим запасом.

Рекомендоване розташування світильників 3 × 3:

По довжині 9 м → три світильники через 3 м.

По ширині 5 м → три світильники через 1,67 м.

8.6 Висновки до розділу

У ході виконання розділу "Заходи охорони праці та техніки безпеки" були розглянуті питання, що стосуються безпечних і комфортних умов праці для працівників, які працюють за персональними комп’ютерами.

Проведено аналіз санітарно-гігієнічних умов праці. Під час аналізу враховано вимоги чинних нормативних документів. Також ідентифіковано основні шкідливі та небезпечні фактори, що можуть впливати на фахівців під час роботи з ПК. На основі отриманих даних розроблено заходи щодо покращення умов праці.

Розкрито питання техніки безпеки й протипожежної профілактики. Розглянуто правила безпечної експлуатації електрообладнання, порядок дій у випадку загоряння, необхідність забезпечення приміщення первинними засобами пожежогасіння та вимоги щодо електробезпеки й організаційних заходів.

У розрахунковій частині виконано розрахунок штучного освітлення приміщення. На основі обраних LED-панелей розрахована потрібна кількість, що гарантує відповідний рівень світлового комфорту.

Підводячи підсумок, можна зробити висновок, що проведений аналіз та розроблені технічні й організаційні заходи забезпечують відповідність чинним нормам охорони праці та сприяють збереженню здоров’я працівників у процесі професійної діяльності.

9 ОСНОВНІ ВИСНОВКИ

У ході виконання пояснювальної записки було виконано повний цикл дослідження та проєктування системи мережевої розсилки електронної пошти з забезпеченням трирівневої моделі захисту даних. Було розглянуто сервіс, який забезпечує управління користувачами, відправлення листів, створення та налаштування cron-задач, а також перегляд статистики відправлених повідомлень.

Проєктування структури бази даних дозволяє забезпечити логічну цілісність даних та ефективність виконання запитів. Реалізація забезпечує взаємодію клієнтської частини з сервером та спрощує інтеграцію з іншими сервісами.

Під час виконання роботи було розглянуто схожі системи, такі як Listmonk, Postal і Amazon SES. Вони демонструють хороші можливості для організації розсилок, проте обмежені лише транспортним шифруванням (TLS). Це означає, що хоча дані передаються захищеним каналом, всередині самих систем інформація залишається незахищеною. У жодній із платформ не реалізовано шифрування бази даних користувачів або перевірку цілісності журналів, а механізми контролю доступу переважно спрощені чи залежні від зовнішніх інструментів.

Зважаючи на вище перераховане було прийнято рішення приділити окрему увагу забезпеченню безпеки системи. Розроблено трирівневу модель захисту, що поєднує використання TLS для шифрування з’єднань між клієнтом і сервером, AES для захисту чутливих даних у базі користувачів та SHA-256 для хешування логів.

Також було проведено аналіз і підбір літературних та технічних джерел, серед яких документація Listmonk, Postal, Amazon SES, Gmail SMTP, а також наукові й навчальні видання з програмування на Java, побудови REST API та

проєктування баз даних. Це дало змогу сформувати міцну теоретичну основу проєкту, обґрунтувати архітектурні рішення та забезпечити наукову новизну розробки.

Результати роботи можуть бути використані для подальшого розвитку програмних рішень у сфері масових електронних розсилок, систем підтримки користувачів та інформаційних платформ. Виконана робота має практичну цінність, оскільки продемонструвала повний цикл запланованої розробки сучасної системи на базі Java та Spring Boot, підтвердила можливість побудови безпечних і масштабованих комунікаційних сервісів та сформувала основу для подальших досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт Listmonk URL: https://listmonk.app

2. Офіційна документація Postal URL: https://docs.postalserver.io

3. Офіційний сайт Amazon Simple Email Service (SES) URL: https://docs.aws.amazon.com/ses

4 Java “Back to Basics” Tutorial URL: https://www.baeldung.com/java-tutorial

5. Lesson: Language Basics
URL: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/

6. Гід-посібник для розробників Java URL: https://roadmap.sh/java

7. Spring Web MVC URL: https://docs.spring.io/spring-framework/reference/web/webmvc.html

8. Java Bean Validation URL: https://docs.spring.io/spring-framework/reference/core/validation/beanvalidation.html

9. Hibernate ORM URL: https://hibernate.org/orm/documentation/

10. Building a RESTful Web Service URL: https://spring.io/guides/gs/rest-service/

11. Guide to Spring Email URL: https://www.baeldung.com/spring-email

12. Як надсилати електронні листи з додатка
URL: https://support.google.com/a/answer/176600

13. Task Execution and Scheduling URL: https://docs.spring.io/spring-framework/reference/integration/scheduling.html

14. Cron Expression Generator & Explainer
URL: https://www.freeformatter.com/cron-expression-generator-quartz.html

15. HTTPS using Self-Signed URL: https://www.baeldung.com/spring-boot-https-self-signed-certificate

16. Java AES Encryption and Decryption URL: https://www.baeldung.com/java-aes-encryption-decryption

17. Hashing in Java URL: https://www.baeldung.com/sha-256-hashing-java

18. JDK 21 Documentation URL: https://docs.oracle.com/en/java/javase/21/

19. Martin, R. C. Clean Code. Prentice Hall, 2008 рік, 464 сторінки.

20. Martin, R. C. Clean Architecture. Pearson, 2017 рік, 432 сторінки.

21. Schildt, H. Java: A Beginner's Guide, 8th Edition. McGraw-Hill, 2018 рік, 720 сторінок.

22. Schildt, H. Java: A Complete Reference, 12th Edition. McGraw-Hill, 2022 рік, 1248 сторінок.

23. Heckler, M. Spring Boot: Up and Running. O'Reilly, 2021 рік, 320 сторінок.

24. Subramanian, S. Pro REST API Development with Java. Apress, 2018 рік, 350 сторінок.

25. Elmasri, R., & Navathe, S. Fundamentals of Database Systems. Pearson, 2015 рік, 1200 сторінок.

26. Dinh, T., & Sethi, R. JavaMail API Developer's Guide. Oracle, 2012 рік, 300 сторінок.

27. Johnson, R. SMTP Protocol and Implementation Guide. Independently published, 2025 рік, 480 сторінок.

28. Смірнов О.А., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Проєктування комп'ютерних систем та мереж: навчальний посібник. Кропивницький: вид. Лисенко В.Ф., 2019 рік, 264 сторінок.

29. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології: навчальний посібник. Кіровоград: РВЛ КНТУ, 2016 рік, 159 сторінок.

30. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі: навчальний посібник. Кіровоград: РВЛ КНТУ, 2016 рік, 233 сторінок.

31. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Кіровоград: КНТУ, 2012 рік, 454 сторінок.

32. Інноваційний розвиток діяльності суб'єктів господарювання в умовах воєнного та повоєнного стану: теорія, практика, аналітика: монографія / [Пономаренко В.С., Малярець Л.М., Внукова Н.М. та ін.]; за заг. ред. Пономаренка В.С. – Харків : ХНЕУ ім. Кузнеця, 2024 рік, 428 сторінок.

33. Менеджмент. Маркетинг. Підприємництво: навч. посіб. / Рябоволик Т.Ф., Андрощук І.О., Доренська А.О. та ін. – Кропивницький : ЦНТУ, 2024 рік, 208 сторінок.

34. Нова світова економіка: менеджмент, технології, стратегії: навч. посібник / Зінчук Т.О., Куцмус Н.М., Прокопчук О.А., Данкевич В.Є. та ін.; за ред. Зінчук Т.О., Куцмус Н.М. – Київ : Центр учбової літератури, 2022 рік, 372 сторінок.

35. Страпчук С.І. Менеджмент: навчальний посібник. Львів : Видавництво «Новий Світ – 2000», 2024 рік, 356 сторінок.

36. Krasteva, Iva. 17 Project Management Trends to Navigate 2024 and Beyond. URL: https://businessmap.io/blog/project-management-trends

37. Digital Tiger: the Power of Ukrainian IT – 2023.

URL: https://mind.ua/publications/20270953-it-industriya-v-cifrah-najcikavishi-dani-z-doslidzhennya-digital-tiger

38. Snyder, D. C. Hybrid Project Management. – New Jersey : John Wiley & Sons, 2022 рік, 320 сторінок.

39. Міністерство цифрової трансформації України. Результати цифрової трансформації в регіонах України за 2023 рік.

URL: https://thedigital.gov.ua/news/rezultati-tsifrovoi-transformatsii-v-regionakh-ukraini-za-2023

40. Розвиток IT бізнесів в Україні та світі – основні тенденції.

URL: https://finance.bigmir.net/ua/business/7215755-rozvitok-it-biznesiv-v-ukrayini-ta-sviti-osnovni-tendentsiyi

41. Державні будівельні норми України: ДБН В.2.5-28:2018.

42. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12

43. Зеркалов Д.В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011рік, 551 сторінок.

44. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». URL: https://zakon.rada.gov.ua/laws/main/z0508-18?utm_source=chatgpt.com#Text

45. Охорона праці. Ч.1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997 рік, 20 сторінок. URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

46. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

47. Розрахунки з електробезпеки. Розрахунок захисного заземлення. URL: https://cpo.stu.cn.ua/Oksana/rozrah_rozd_OP_DP_bak_spec_mag/90.html

48. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2022 рік, 175 сторінок. URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161

49. Охорона праці. Ч. 2. Занулення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ − сумісного типу. / Оришака О.В., Солових Є.К., Оришака В.О., Солових А.Є., Катеринич С.Е.; М-во освіти і науки України,

Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2019 рік, 27 сторінок.
URL: https://dspace.kntu.kr.ua/items/d4fddab2-5b10-445d-a873-8febf465fe61

50. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим(магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальності 123 “Комп’ютерна інженерія” [укл. О.В. Оришака, К.М. Марченко]; М-во освіти і науки України, Центральноукраїн. нац.техн. ун-т. – Кропивницький : ЦНТУ, 2022 рік, 20 сторінок.