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
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Середа М.Л.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Середі Максиму Леонідовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Середа М.Л. Дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи обробки електронної пошти по протоколам IMAP та SMTP.

Метою розробки є дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

Об'єктом дослідження є процес обробки електронної пошти по протоколам IMAP та SMTP.

Предметом дослідження є методи обробки електронної пошти по протоколам IMAP та SMTP.

Методи дослідження базуються на методах теорії телетрафіку, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

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 ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерні науки, IMAP, SMTP

ABSTRACT

Sereda M.L. Research and software implementation of an e-mail processing system based on the IMAP and SMTP protocols. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software designed for the IMAP and SMTP e-mail processing system was developed.

The purpose of the development is the research and software implementation of the e-mail processing system based on the IMAP and SMTP protocols.

The object of the study is the process of e-mail processing using the IMAP and SMTP protocols.

The subject of the study is methods of e-mail processing using the IMAP and SMTP protocols.

Research methods are based on teletraffic theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the e-mail processing system using the IMAP and SMTP protocols.

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 convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: computer science, IMAP, SMTP

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	програмне забезпечення
ASLR	–	Випадкове Розміщення в Адресному Просторі
DEP	–	Запобігання Виконання Даних
DNS	–	Domain Name System
IMAP	–	Internet Message Access Protocol
MIME	–	Multipurpose Internet Mail Extension
POP	–	Post Office Protocol
NRPC	–	віддалений виклик процедур Notes
RM	–	Менеджером Перезавантажень
RSS	–	Really Simple Syndication
SMTP	–	Simple Mail Transfer Protocol
SSL	–	Secure Socket Layer
TLS	–	Transport Layer Security
USENET	–	User Network

ВСТУП

Актуальність теми. Поштовий клієнт – це комп'ютерна програма, яка використовується для читання та надсилання електронних повідомлень. Однак клієнт електронної пошти – це не те саме, що сервер електронної пошти; останнє – це апаратне забезпечення, яке централізовано транспортує та зберігає пошту для багатьох користувачів електронної пошти. Клієнт електронної пошти, навпаки, – це те, з чим взаємодіє такий один користувач, як ви. Як правило, клієнт завантажує повідомлення з сервера для локального використання (або для використання в браузері) і завантажує повідомлення на сервер для доставки одержувачам.

Клієнт електронної пошти дозволяє читати, упорядковувати та відповідати на повідомлення, а також надсилати нові електронні листи.

Щоб упорядкувати електронну пошту, поштові клієнти зазвичай пропонують папки, мітки або те й інше. Інтегрована пошукова система дозволяє знаходити повідомлення за такими деталями, як відправники, теми, час отримання та вміст.

Окрім тексту електронної пошти, клієнти електронної пошти також обробляють вкладення, тому ви можете надсилати та отримувати комп'ютерні файли (наприклад, зображення, документи чи електронні таблиці) електронною поштою.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

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

– Огляд існуючих систем обробки електронної пошти по протоколам IMAP та SMTP.

– Дослідження системи обробки електронної пошти по протоколам IMAP та SMTP.

– Програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

Об'єктом дослідження є процес обробки електронної пошти по протоколам IMAP та SMTP.

Предметом дослідження є методи обробки електронної пошти по протоколам IMAP та SMTP.

Методи дослідження базуються на методах теорії телетрафіку, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод обробки електронної пошти по протоколам IMAP та SMTP.

– Розроблено вітчизняний продукт обробки електронної пошти по протоколам IMAP та SMTP, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі обробки електронної пошти по протоколам IMAP та SMTP.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Поштові клієнти можуть використовувати низку протоколів для надсилання та отримання електронних листів через сервери електронної пошти.

Повідомлення або зберігаються локально на вашому комп'ютері (зазвичай, коли для завантаження пошти із сервера використовується протокол POP або Post Office Protocol), або електронні листи та папки синхронізуються із сервером (зазвичай, коли використовуються протоколи IMAP і Exchange). Завдяки IMAP (Internet Message Access Protocol) і Exchange клієнти електронної пошти, які мають доступ до одного облікового запису, бачать однакові повідомлення та папки, і всі дії автоматично синхронізуються.

Щоб надіслати електронний лист, клієнти електронної пошти використовують майже виключно SMTP (Простий протокол передачі пошти). (З обліковими записами IMAP надіслане повідомлення зазвичай копіюється в папку «Надіслані», і поштові клієнти можуть отримати до нього доступ.)

Існують також інші протоколи електронної пошти. Деякі служби електронної пошти пропонують API (інтерфейси прикладного програмування) для клієнтів електронної пошти для доступу до пошти на своїх серверах. Ці протоколи можуть пропонувати додаткові функції, такі як відкладене надсилання або тимчасове блокування електронних листів.

Історично X.400 був важливим альтернативним протоколом електронної пошти, який використовувався переважно в 1990-х роках. Його складність робить його придатним для використання в уряді та бізнесі, але важчим для реалізації, ніж електронна пошта SMTP/POP.

1.2 Область застосування

Завдяки веб-додаткам, які отримують доступ до електронної пошти на сервері, браузери перетворюються на клієнти електронної пошти.

Наприклад, якщо ви отримуєте доступ до Gmail у Mozilla Firefox, сторінка Gmail у Mozilla Firefox діє як ваш клієнт електронної пошти; це дозволяє читати, надсилати та впорядковувати повідомлення. У цьому випадку для доступу до електронної пошти використовується протокол HTTP.

Чи може автоматизоване програмне забезпечення бути клієнтом електронної пошти?

У технічному розумінні будь-яка програма, яка отримує доступ до електронної пошти на сервері за допомогою POP, IMAP або подібного протоколу, є клієнтом електронної пошти. Таким чином, будь-яке програмне забезпечення, яке автоматично обробляє вхідну електронну пошту, можна назвати клієнтом електронної пошти (навіть якщо ніхто ніколи не бачить повідомлення), особливо стосовно сервера електронної пошти.

Поширені клієнти електронної пошти

До популярних поштових клієнтів належать Microsoft Outlook, Mozilla Thunderbird, macOS Mail, IncrediMail, Mailbox і iOS Mail. Найпопулярнішим веб-клієнтом електронної пошти є Gmail; інші включають Yahoo! Пошта та Outlook.com.

Історично важливі клієнти електронної пошти включали Eudora, Pine, Lotus (і IBM) Notes, nmh і Outlook Express.

Інформація, яка потрібна для налаштування поштового клієнта для облікового запису POP3

Вам потрібно знати налаштування POP-сервера вашого постачальника облікових записів, які можна знайти в довідковій документації про POP. Наприклад, щоб додати Gmail як обліковий запис POP до Outlook, увімкніть доступ POP у Gmail у розділі Налаштування > Переглянути всі налаштування >

Пересилання та POP/IMAP > Увімкнути POP для всієї пошти. Потім додайте обліковий запис в Outlook > виберіть POP у параметрах додаткових параметрів > і введіть налаштування POP Gmail, щоб почати синхронізацію пошти.

Різниця між поштовим клієнтом і веб-поштою

Клієнт електронної пошти – це комп'ютерна програма, прив'язана до одного комп'ютера, тоді як веб-пошта доступна через декілька браузерів. У деяких випадках провайдери пропонують веб-версії своїх поштових клієнтів. Наприклад, Outlook доступний як у версії для настільного ПК, так і у веб-версії з Outlook.com.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Ось 17 найкращих поштових клієнтів, які ви можете переглянути, щоб вибрати, який вам найкраще підходить:

1. Spike

Spike прагне бути гнучкою платформою для електронної пошти та співпраці. Він форматує електронні листи як текстові ланцюжки, щоб їх було легко відслідковувати, і дозволяє вам одночасно спілкуватися в чаті з вашими контактами. Додаток функціонує для виділення важливих електронних листів і впорядкування різноманітних вхідних повідомлень. Усі ці функції використовують шифрування для захисту ваших даних.

2. Mailbird

Mailbird – це служба для керування кількома обліковими записами електронної пошти або адресами. Цей клієнт надає можливість групувати всі ваші електронні листи в одній папці "Вхідні", а також інші функції, такі як швидкі читачі, налаштовувані кнопки відкладення та пошук вкладень. Він також може інтегруватися з декількома платформами соціальних медіа та іншими програмами для зберігання й комунікації, такими як Slack і Dropbox, щоб централізувати ваші зв’язки та бізнес.

3. eM Client

eM Client допомагає людям, які очікують отримати велику кількість електронних листів протягом дня. Ця служба працює з усіма основними постачальниками електронної пошти, а також може працювати в хмарі. Він має функцію миттєвого пошуку в історії електронної пошти та бічну панель, яка

показує ваш розклад, контактну інформацію та історію електронних листів для ваших контактів, щоб допомогти вам керувати продуктивністю. Його користувальницький інтерфейс майже повністю налаштовується, що дозволяє вам вибрати та створити робочий формат електронної пошти, який найкраще відповідає вашим вимогам, від розміщення кнопок до колірної схеми, яку ви віддаєте перевагу.

4. Mozilla Thunderbird

Thunderbird – це клієнт електронної пошти з відкритим вихідним кодом, який Mozilla розробила, щоб бути максимально зручним і ефективним. Оскільки він має відкритий вихідний код, він може інтегрувати плагіни та розширення від різних розробників, які можна ввімкнути безпосередньо з Thunderbird. Служба також може допомогти вам швидше відповідати, дозволяючи створювати шаблони швидкої відповіді та планувати надсилання електронних листів у певний час. Він також підтримує наскрізне шифрування, забезпечуючи безпеку ваших електронних листів і даних.

5. Mailspring

Mailspring прагне надати інтуїтивно зрозумілу послугу, яка адаптується до будь-яких вимог користувача. Він намагається досягти цього за допомогою різноманітних функцій підтримки, включаючи розпізнавання дотиків і жестів, розширені комбінації клавіш, інтегровану перевірку орфографії, переклад у реальному часі та уніфіковану папку "Вхідні" для кількох облікових записів електронної пошти. Mailspring використовує C++ для швидкої роботи навіть на старих комп’ютерах і пристроях. Служба також інформує вас під час спілкування з контактами, отримуючи інформацію та відстежуючи, як одержувачі відповідають на електронні листи, які ви надсилаєте. Цей клієнт сумісний практично з усіма основними операційними системами.

6. Inky

Inky – це поштовий клієнт, який більше зосереджується на блокуванні шахрайства та спуфінгу. Він використовує штучний інтелект (AI) із машинним

навчанням, щоб підвищити безпеку та запобігти потраплянню шахрайських електронних листів до вашої папки "Вхідні". Він також сумісний і простий у використанні з більшістю інтегрованих бізнес-платформ.

7. Airmail

Airmail – це клієнт електронної пошти з широкими можливостями налаштування, який дозволяє кожному користувачеві мати абсолютно унікальний інтерфейс. Ви можете налаштувати його для керування вхідними електронними листами та налаштування додаткових параметрів робочого процесу. Він також підтримує кілька облікових записів, що дозволяє об’єднувати кілька адрес електронної пошти в одну папку «Вхідні». Повітряна пошта може працювати та взаємодіяти з голосовими командами ШІ та сумісна з мобільними пристроями та комп’ютерами.

8. Spark

Spark працює над тим, щоб полегшити командну роботу через програмне забезпечення електронної пошти та підвищити продуктивність. Це досягається завдяки можливостям командного написання, які дають змогу співпрацювати зі своїми контактами електронної пошти, щоб писати чернетки, створювати нотатки та делегувати завдання електронної пошти. Це також дозволяє налаштувати ваші сповіщення так, щоб ви отримували їх лише тоді, коли ви отримуєте електронні листи від відомих відправників, і ви можете використовувати трекер прогресу, щоб керувати своїм часом і завданнями.

9. Hiri

Hiri – це програма, призначена для полегшення керування бізнес-електронною поштою та підвищення загальної продуктивності. Він має намір досягти цього за допомогою різних функцій для підвищення продуктивності, включаючи засіб для видалення безладу, сповіщення про нагадування та інтелектуальну інформаційну панель, яка відображає непрочитані електронні листи та пропонований час відповіді. Hiri також синхронізує свій календар з

іншими службами, позбавляючи вас необхідності використовувати декілька програм для отримання основної та важливої інформації.

10. Bat!

Bat! це поштовий клієнт, який надає пріоритет безпеці за допомогою шифрування в реальному часі для захисту вхідних і вихідних даних. Коли ви вперше створюєте обліковий запис, ви можете встановити головний пароль, щоб заблокувати шифрування ваших електронних листів та інформації, яку ви отримуєте. Він має різноманітні функції, включаючи фільтри, папки та правила, яких програмне забезпечення має дотримуватися, і весь інтерфейс користувача також можна налаштувати.

11. Opera Mail

Opera Software розробила поштовий клієнт Opera Mail, який також є невід'ємною частиною веб-браузера Opera. Він працює з єдиною базою даних, яка підтримує індекс усіх повідомлень і автоматично сортує їх у кількох точках доступу або переглядах. Категоризація повідомлень здійснюється автоматично за типами, наприклад списки розсилки та повідомлення з вкладеннями. Opera Mail може відображати як HTML, так і текст. Він використовує механізм Presto для відображення перегляду HTML.

12. Postbox

Postbox намагається бути інтуїтивно зрозумілим і компетентним. Це дозволяє вам упорядковувати свої електронні листи у вкладках, подібно до веб-браузера, тож ви можете легко керувати кількома потоками. Цей клієнт також пропонує серію навчальних відео в додатку, щоб новачки могли використовувати всі його функції та конфігурації. Він керує великою кількістю файлів і завантажень, інтегруючи ваш улюблений сервіс хмарного зберігання, наприклад Dropbox, і його налаштування пошуку, які дозволяють знаходити всі вкладення та файли.

13. TouchMail

TouchMail централізує всі ваші облікові записи в одному місці та має візуально привабливий кольоровий інтерфейс, який економить ваш час на електронну пошту. Він сумісний із сенсорним екраном, клавіатурою та мишкою. Він пропонує різні функції, зокрема кольорове кодування повідомлень за відправником, миттєвий перегляд повідомлень і швидке сортування та очищення електронної пошти від найчастіших відправників.

14. Zimbra

Програмне забезпечення Zimbra має на меті перетворити електронну пошту на робочий простір для спільної роботи. Він досягає цього, додаючи живий чат, відеоконференції, обмін файлами та екраном. Він також містить кнопку «скасувати надсилання» та можливість відновити видалені електронні листи, якщо ви щось втратите помилково. Zimbra також сумісна майже з усіма основними операційними системами.

15. Superhuman

Superhuman – це клієнт, метою якого є переосмислення електронної пошти шляхом оптимізації процесу та прискорення програмного забезпечення. У ньому стверджується, що кожна дія на клієнті займає менше 100 мілісекунд із потужним штучним інтелектом, який визначає пріоритет ваших електронних листів на основі ваших уподобань. Крім того, він пропонує додаткові функції, такі як можливість скасувати надіслане повідомлення та визначити, коли одержувач відкрив вашу електронну пошту.

16. ProtonMail

ProtonMail – це зручний клієнт електронної пошти з наскрізним шифруванням із відкритим вихідним кодом. Ви можете створювати нові облікові записи без особистої інформації, щоб додати другий рівень безпеки, не дозволяючи записувати свою IP-адресу. Ця конфіденційність також працює з файлами та робочими областями, які ви пов’язуєте зі своїм обліковим записом ProtonMail.

17. Shift

Поштовий клієнт Shift робить електронну пошту більш продуктивною та цікавою. Це дозволяє створювати спільні робочі області та керувати кількома робочими процесами. Розробники Shift розробили його так, щоб він був високоінтегрованим і дозволяв співпрацювати на кількох платформах і розширеннях. Люди також можуть використовувати Shift як веб-браузер.

Поради щодо вибору поштового клієнта

Є багато варіантів для клієнтів електронної пошти, і всі вони мають певні особливості. Якщо ви не знайомі з клієнтами електронної пошти, вам може знадобитися допомога у визначенні того, що таке хороший клієнт електронної пошти. Ось кілька моментів, які ви можете враховувати, вибираючи поштовий клієнт для керування вашими електронними листами:

– Налаштування.
– Доступність.
– Продуктивність електронної пошти.
– Інтеграція програми.
– Доступність з Інтернетом і без нього.
– Оновлені функції.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom

managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це

означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки. Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи обробки електронної пошти по протоколам IMAP та SMTP.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Клієнт електронної пошти – це програма, яка дозволяє надсилати, отримувати та впорядковувати електронні листи на комп'ютері, планшеті чи смартфоні.

Поштові клієнти – це програми з інтерфейсом для читання електронних листів і відповідей на них, сортування повідомлень у папках або додавання міток, написання нових повідомлень і керування вкладеннями.

Приклади поштового клієнта

Зазвичай поштові клієнти – це окремі програми, які ви встановлюєте на свій пристрій.

До популярних поштових клієнтів настільного комп'ютера належать Microsoft Outlook, Mozilla Thunderbird і Apple Mail. Для мобільних пристроїв Apple Mail, Gmail, Outlook і Proton Mail пропонують зручні програми з розширеними функціями, які допоможуть вам бути в курсі папки "Вхідні".

Більшість основних служб електронної пошти також пропонують веб-електронну пошту або веб-пошту. Коли ви отримуєте доступ до Gmail, Outlook або Proton Mail в Інтернеті, ваш браузер стає свого роду клієнтом електронної пошти, що дозволяє надсилати, отримувати та впорядковувати пошту.

Як працюють клієнти електронної пошти?

Поштові клієнти використовують протоколи електронної пошти для підключення до поштових серверів для надсилання та отримання повідомлень:

– SMTP (Простий протокол передачі пошти)(нове вікно)це протокол вихідної електронної пошти, який використовується для надсилання повідомлень із клієнта електронної пошти на поштовий сервер.

– IMAP (протокол доступу до обміну повідомленнями в Інтернеті)(нове вікно) це протокол вхідної електронної пошти, який використовується для отримання та синхронізації електронної пошти з поштовим сервером і кількома клієнтами електронної пошти (пристроями).

– POP3 (протокол поштового відділення)(нове вікно)це старіший протокол вхідної електронної пошти, який дозволяє лише завантажувати повідомлення в клієнт електронної пошти.

Простіше кажучи, ось як клієнт електронної пошти працює з цими протоколами:

1. Ви створюєте повідомлення в поштовому клієнті на телефоні, комп'ютері чи планшеті та натискаєте Надіслати.

2. Ваше повідомлення доставляється на сервер вихідної пошти (SMTP), який передає його через Інтернет на сервер вхідної пошти (IMAP/POP3).

3. Сервер вхідної пошти доставляє ваше повідомлення та вкладені файли одержувачу.

Як працюють поштові клієнти

Поштові клієнти використовують SMTP для надсилання електронних листів. Але якщо ви вручну налаштовуєте свій клієнт і маєте вибір між IMAP і POP3 для вхідної пошти, виберіть IMAP.

Оскільки POP3 працює лише в одному напрямку, завантажуючи повідомлення на пристрій і (зазвичай) видаляючи його з сервера, він не може синхронізувати ваші повідомлення на різних пристроях.

Навпаки, за допомогою IMAP ви можете синхронізувати свою пошту (включаючи вкладення) на всіх своїх пристроях. Наприклад, ви можете:

– Надішліть електронний лист на один пристрій, і його буде додано до папки **«Надіслані»** на всіх пристроях. За допомогою POP надіслані елементи можна зберігати лише на тому пристрої, з якого вони надіслані.

– Позначте повідомлення як прочитане, і воно відображатиметься як прочитане на всіх пристроях. За допомогою POP повідомлення електронної

пошти як прочитане відображається лише на пристрої, на якому ви його позначили.

– Створіть папки або інші налаштування на одному пристрої, і вони будуть доступні на всіх пристроях. За допомогою POP ви повинні вручну налаштувати папки та налаштування на кожному пристрої.

– Отримуйте електронні листи, які надходять на ваш пристрій. За допомогою POP ваш поштовий клієнт періодично перевіряє наявність нових повідомлень і отримує їх.

Коротше кажучи, IMAP у більшості випадків є найкращим протоколом для більшості людей.

Навіщо використовувати поштовий клієнт?

Поштові клієнти для комп'ютерів мають деякі переваги перед веб-поштою. Ось чому ви можете використовувати окрему програму електронної пошти замість або разом із веб-поштою на своєму комп'ютері.

Керуйте кількома обліковими записами

Поштові клієнти дозволяють керувати кількома обліковими записами електронної пошти від різних постачальників послуг електронної пошти. Замість того, щоб входити на різні сторінки веб-пошти для різних облікових записів електронної пошти, ви можете керувати всіма своїми обліковими записами за допомогою однієї програми.

Доступ до електронної пошти в автономному режимі

Ви можете легко завантажувати та зберігати електронні листи на своєму пристрої для доступу до них у режимі офлайн за допомогою поштових клієнтів. Офлайн-доступ може бути корисним, коли у вас немає підключення до Інтернету, наприклад під час подорожі.

Створіть резервну копію повідомлень

Поштові клієнти дозволяють легко створювати резервні копії ваших електронних листів і зберігати їх локально або на зовнішньому жорсткому диску.

Таким чином ви збережете контроль над своїми даними та зможете відновити повідомлення, якщо їх випадково видалено чи втрачено.

Налаштуйте свою електронну пошту

Найкращі постачальники веб-пошти, такі як Proton Mail, дозволяють змінювати макет, тему та інші параметри, щоб персоналізувати свою електронну пошту. Окремі клієнти електронної пошти можуть підтримувати додаткові додатки для подальшого налаштування вашого досвіду.

Підвищте свою продуктивність

Розширені служби веб-пошти, такі як Proton Mail і Gmail, містять інструменти для полегшення робочого процесу, як-от комбінації клавіш, розширені параметри фільтрації та планування надсилання електронних листів пізніше. Окремі поштові клієнти можуть пропонувати додаткові параметри або інтеграцію програм.

Додати розширені функції

Найкращі клієнти електронної пошти також дозволяють додавати розширення або доповнення з більш розширеними функціями. Наприклад, з Mozilla Thunderbird з відкритим кодом(нове вікно), ви можете отримати різні доповнення для підвищення конфіденційності та безпеки(нове вікно).

Навіщо використовувати веб-пошту замість поштового клієнта?

Незважаючи на наведені вище переваги поштових клієнтів, ви можете віддати перевагу використанню веб-пошти замість поштового клієнта, залежно від ваших потреб. Ось деякі переваги використання веб-пошти замість настільного поштового клієнта.

Доступ з будь-якого пристрою

Веб-пошта дозволяє отримати доступ до електронної пошти з будь-якого пристрою зі стандартним браузером і підключенням до Інтернету. Таким чином ви можете керувати своєю електронною поштою на будь-якому пристрої без встановлення додаткового програмного забезпечення.

Налаштуйте свою електронну пошту один раз

Веб-пошту зазвичай легко налаштувати. Після вибору параметрів і налаштувань вони будуть однаковими на всіх пристроях. Навпаки, більшість програм електронної пошти потрібно налаштовувати вручну на кожному пристрої.

Будьте в курсі подій

Хороші постачальники веб-пошти, як-от Proton Mail, гарантують, що їхні служби оновлюються останніми виправленнями безпеки та функціями користувача. На відміну від програм електронної пошти, вам не потрібно турбуватися про оновлення веб-пошти, щоб залишатися в безпеці та бути в курсі подій.

Електронна пошта на всіх платформах

Веб-пошта працює в усіх операційних системах, якщо у вас є веб-переглядач, незалежно від того, користуєтеся ви комп'ютером Windows, Mac чи Linux чи мобільним пристроєм. З клієнтами електронної пошти ви можете не знайти програму, яка добре працює або доступна для всіх ваших пристроїв.

Електронна пошта без додаткових витрат

Сервіси електронної пошти зазвичай надають веб-пошту безкоштовно. Хоча багато хороших поштових клієнтів є безкоштовними, деякі вимагають одноразового платежу або постійної підписки, щоб установити програму або отримати доступ до розширених функцій.

Поштові клієнти та зашифрована електронна пошта

Незалежно від того, який клієнт електронної пошти ви виберете, ви захочете переконатися, що ваша електронна пошта є конфіденційною та безпечною. Однак великі безкоштовні постачальники послуг електронної пошти, такі як Gmail і Outlook, не є приватними(нове вікно)або безпечний. Єдиний спосіб забезпечити справжню конфіденційність – використовувати наскрізне шифрування(нове вікно).

Деякі клієнти електронної пошти, наприклад Mozilla Thunderbird, мають вбудовану підтримку наскрізного шифрування PGP(нове вікно), а інші

підтримують PGP із доповненнями. Ви також можете отримати розширення для браузера, наприклад Mailvelope з відкритим кодом(нове вікно), щоб використовувати PGP із веб-поштою.

Однак для налаштування клієнтів електронної пошти за допомогою PGP потрібні певні технічні знання, а деякі додатки PGP і розширення для браузера мають інші недоліки, як-от відсутність автоматичного шифрування вкладень.

Вибирайте клієнта, залишайтеся в безпеці

Зрештою, вибір окремого клієнта електронної пошти замість веб-пошти на комп'ютері залежить від того, що вам потрібно та що віддаєте перевагу.

Якщо ви користуєтеся розширеною службою електронної пошти, як -от Proton Mail із наскрізним шифруванням, ви можете бути задоволені веб-поштою. Proton Mail в Інтернеті має всі функції, необхідні для керування та впорядкування вашої пошти. Або ви можете віддати перевагу автономному настільному клієнту, наприклад Outlook, Apple Mail або Thunderbird, залежно від вашої операційної системи.

3.2 Розробка структурної схеми

SMTP, IMAP і POP3

IMAP і SMTP – це протоколи, які використовуються клієнтами електронної пошти для надсилання та отримання безпечних електронних листів. POP3 – це застарілий стандарт, який сьогодні використовується все рідше.

SMTP

Простий протокол передавання пошти (SMTP) – це протокол **вихідної** електронної пошти, який використовується для надсилання електронних листів із клієнта електронної пошти на поштовий сервер. Він працює наступним чином:

1. Ви вводите свою електронну пошту (і додаєте будь-які вкладення) локально у своєму поштовому клієнті.

2. Коли ви натискаєте **Надіслати**, ваш поштовий клієнт підключається до поштового сервера SMTP і передає електронну пошту.

3. Після того, як електронний лист буде передано, ваш поштовий клієнт відключається від поштового сервера.

Більшість поштових серверів приймають з'єднання через TCP-порт 587 і TCP-порт 2525. Різні служби електронної пошти можуть використовувати інші порти, хоча багато з них блокують вихідні з'єднання клієнтів через TCP-порт 25, оскільки, хоча цей порт використовується для зв'язку між серверами SMTP, ним часто зловживають. за розсилку спаму(нове вікно).

IMAP

IMAP – це протокол **вхідної** електронної пошти, який використовується для отримання електронних листів із віддаленого поштового сервера до клієнта електронної пошти. IMAP може:

– Згрупуйте пов'язані повідомлення та розмістіть їх у папках.

– Визначте, чи електронний лист було прочитано, на нього відповіли чи видалено.

– Пошук повідомлень на поштовому сервері.

IMAP працює таким чином:

1. Клієнт електронної пошти підключається до поштового сервера (зазвичай через порт TCP 993, припускаючи безпечне з'єднання TLS).

2. Ви переглядаєте та керуєте своїми електронними листами на сервері.

3. Клієнт електронної пошти залишається підключеним до поштового сервера та продовжує оновлюватися в режимі реального часу, доки ви його не закриєте.

За допомогою IMAP електронні листи завантажуються окремо лише тоді, коли ви клацаєте, щоб переглянути їх. Повідомлення не видаляються із сервера (якщо ви не вирішите видалити їх, але навіть тоді вони, швидше за все, за замовчуванням будуть поміщені в папку Кошик, а не видалені відразу).

Під час підключення до поштового сервера за допомогою IMAP будь-які внесені вами зміни, як-от відповідь на електронні листи, їх видалення чи переміщення в інші папки, відбуваються на поштовому сервері. Це означає, що коли ви підключаєтеся до сервера за допомогою IMAP на іншому пристрої, усі зміни синхронізуються.

POP3

POP3 – це остання версія протоколу Post Office(нове вікно)(POP) і є застарілим стандартом, який більше не використовується часто.

Як і IMAP, POP3 є протоколом **вхідної** електронної пошти, тобто він використовується для отримання електронних листів із віддаленого сервера до клієнта електронної пошти. POP3 – це простий протокол, який зазвичай працює таким чином:

1. Клієнт електронної пошти підключається до поштового сервера (зазвичай через порт TCP 995, припускаючи безпечне з'єднання TLS).

2. Завантажує електронні листи.

3. Видаляє копії завантажених електронних листів із сервера електронної пошти.

4. Відключається від сервера.

POP3 підтримує різні безпечні методи автентифікації, і його простота іноді розглядається як одна з його ключових переваг. Мало що може піти не так. Недоліком цієї простоти є те, що POP3 не пропонує розширених функцій керування електронною поштою.

Оскільки POP3 видаляє електронні листи з поштових серверів після того, як їх було отримано, це не дуже корисно для синхронізації електронних листів на різних пристроях.

Це частково вирішується за допомогою опції не видаляти електронні листи з поштового сервера після їх отримання, але IMAP набагато краще обробляє синхронізацію електронних листів між клієнтами.

API

На практиці робота з підключення клієнтів електронної пошти до поштових серверів все частіше виконується програмними інтерфейсами додатків(нове вікно)(API). Це програмне забезпечення, яке, ймовірно, все ще використовує такі протоколи, як SMTP та IMAP, але приховує внутрішні деталі зручним для користувача способом, тому вам не потрібно виконувати будь-яку ручну настройку клієнта електронної пошти.

Proton Mail і IMAP, SMTP. і POP3

Proton Mail підтримує SMTP та IMAP через **Proton Mail Bridge**. Це програма з відкритим вихідним кодом, яка дозволяє вам повністю інтегрувати свій обліковий запис Proton Mail з будь-якою програмою, яка підтримує IMAP і SMTP. Це включає Microsoft Outlook, Mozilla Thunderbird і Apple Mail.

Proton Mail Bridge працює у фоновому режимі на вашому комп'ютері та плавно шифрує та розшифровує вашу електронну пошту під час надходження та виходу з вашого комп'ютера. Жодна інша служба електронної пошти не пропонує функцію, яка захищає вашу конфіденційність таким чином.

Proton Mail Bridge працює з більшістю клієнтів, які належним чином реалізують SMTP та IMAP. Ми перевіряємо, чи Proton Mail Bridge працює в Apple Mail, Outlook і Thunderbird, але інші клієнти електронної пошти також мають бути сумісні. Ми працюємо над забезпеченням більшої сумісності з більшою кількістю клієнтів у майбутньому.

Proton Mail Bridge не підтримує POP3, оскільки Proton Mail хоче забезпечити плавну роботу на кількох платформах.

IMAP проти POP3

Перше, що слід зазначити, це те, що Proton Mail Bridge завжди використовує IMAP, а багато інших служб електронної пошти (зокрема Gmail) використовують API для автоматичного налаштування вашого клієнта електронної пошти.

Якщо ви вручну налаштовуєте поштовий клієнт, виберіть IMAP. Завдяки розширеним функціям керування електронною поштою та легким доступом на кількох пристроях IMAP у більшості випадків є найкращим варіантом для більшості людей.

POP3 особливо корисний, якщо у вас повільне або переривчасте з'єднання з Інтернетом, оскільки всі електронні листи завантажуються безпосередньо на ваш пристрій. POP3 також може бути корисним, якщо на вашому поштовому сервері дуже мало місця для зберігання, оскільки електронні листи зазвичай видаляються з поштового сервера під час завантаження на ваш пристрій.

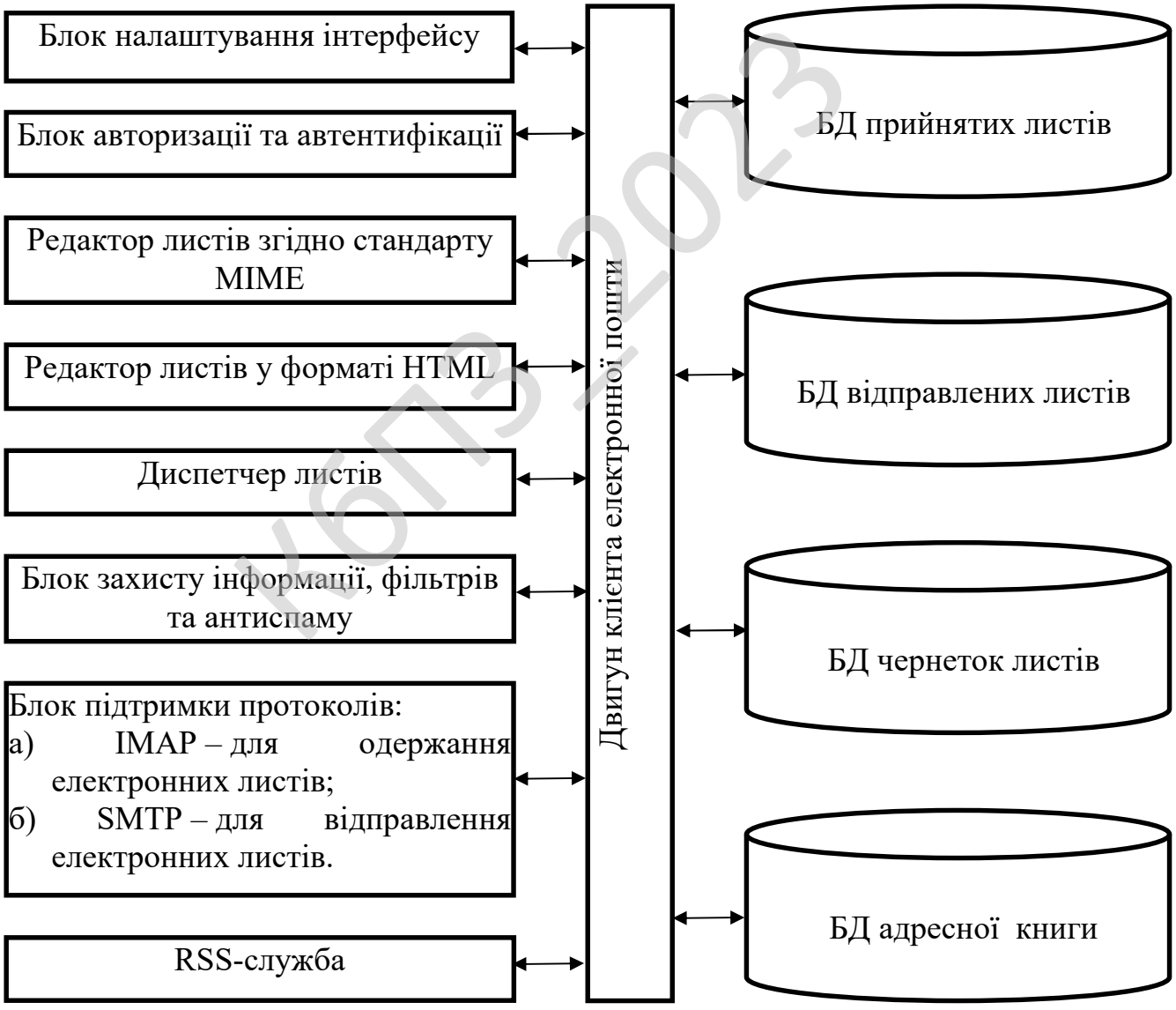

Рисунок 3.1 – Структурна схема розробленої системи

Структурна схема розробленого, у ході виконання магістерського проектування, програмного забезпечення зображена на рисунку 3.1.

Програмне забезпечення поштового клієнту складається з наступних структурних блоків:

– Двигун поштового клієнту системи електронного документообігу по протоколам POP3 та SMTP – основне ядро на якому будується поштовий клієнт.

– Блок підтримки протоколів:

а) IMAP – для одержання електронних листів;

б) SMTP – для відправлення електронних листів.

– БД адресної книги – призначена для зберігання адрес куди відправлені листи, та звідкіля отримані листи.

– БД прийнятих листів – призначена для зберігання листів надісланих користувачеві поштового клієнта.

– БД відправлених листів – призначена для зберігання листів відправлених користувачем поштового клієнта.

– Блок авторизації та автентифікації – призначений для здійснення входу до поштової скриньки згідно логіну та паролю.

– Блок захисту інформації, фільтрів та антиспаму – призначений для забезпечення конфіденційного електронного документообігу по протоколам POP3 та SMTP та захисту від спаму, тобто листів усілякої реклами, які не потрібні користувачу поштової скриньки.

– Диспетчер листів – призначений для управління отриманими, написаними, відправленими листами та чернеток листів.

– БД чернеток листів – призначена для зберігання чернеток листів написаних користувачем поштового клієнта;

– Редактор листів у форматі HTML – призначений для написання та редагування текстових листів у вигляді HTML.

– Редактор листів згідно стандарту MIME – призначений для написання та редагування листів у вигляді тексту, картинок, та інших прикріплених до листа

файлів.

– Блок налаштування інтерфейсу – призначений для індивідуального налаштування інтерфейсу під конкретного користувача.

– RSS-служба – призначена для реалізації служби новин, як поштового клієнта так і глобальних.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональний блок початкових налаштувань

Помічник з міграції

У меню "Довідка" знаходиться функція "Помічник з міграції", яка допоможе настроїти поштового клієнту за вашим смаком. Якщо вам була до вподоби панель інструментів у поштового клієнту, ви можете перенести її до нової версії, використовуючи помічника з міграції. Крім того, за допомогою асистента з міграції ви можете встановити такі додатки, які призначені для змінення вікна повідомлення.

Майстер з налаштування облікового запису

Раніше, до появи цієї функції, вам потрібно було знати свої налаштування IMAP, SMTP і SSL/TLS. Тепер вам необхідно лише вказати своє ім’я, адресу електронної пошти та пароль. Після введення даних, майстер перевірить нашу базу даних і вибере потрібні налаштування.

Адресна книга на відстані одного кліку

Ця функція допомагає швидко та без зусиль додавати контакти до вашої адресної книги. Додавайте контакти, натискаючи значок зірочки в отриманих листах. Клацнувши двічі, ви зможете додати такі деталі, як фотографія, день народження тощо.

Рисунок 3.2 – Функціональна схема системи

Нагадування про вкладення

Функція нагадування про вкладення виконує пошук слова "вкладення"(та інших слів, наприклад назв форматів файлів) у тілі листа. Якщо ключове слово буде знайдено, функція покаже повідомлення з пропозицією додати вкладення, перш ніж лист буде надіслано.

Менеджер діяльності

Менеджер діяльності записує процес взаємодії програми поштового клієнту із вашим постачальником послуг електронної пошти. Вам не потрібно більше блукати в тумані. Зазирніть у відповідний файл, і ви дізнаєтеся всі подробиці взаємодії.

Функціональний блок вкладок та пошуку

Панель інструментів "Швидкий фільтр"

Панель інструментів "Швидкий фільтр" дає змогу швидко відфільтрувати пошту. Почніть введення слова в полі пошуку швидкого фільтра і ви відразу отримаєте результати. Крім того, ви можете відфільтрувати свою пошту за такими параметрами: "Нові листи", "Мітки" або за контактами з адресної книги. Також ви можете "Прикріпити" або зберегти фільтр, щоб використовувати його в інших папках.

Вкладки

Інтерфейс із використанням вкладок дозволяє завантажувати пошту до окремих вкладок і швидко між ними переходити. Припустімо, що ви пишете відповідь на лист і вам потрібна інформація з попереднього листування. Інтерфейс із використанням вкладок дає змогу закласти кілька листів і використовувати інформацію, яку вони містять.

Двічі клацніть листа або натисніть клавішу Enter на повідомленні, щоб відкрити його на окремій вкладці. Натиснувши правою кнопкою на листі або папці, ви зможете відкрити цей елемент на фоновій вкладці.

Під час завершення роботи поштового клієнту видимі вкладки буде збережено й відновлено за наступного запуску програми. Крім того, на панелі інструментів "Вкладки" знаходиться меню, за допомогою якого можна переходити між вкладками.

Пошук

Пошуковий інтерфейс у програмі поштового клієнту містить засоби фільтрування й хронології, які допоможуть знайти потрібного листа. До того ж, поштового клієнту індексує всі ваші листи для пришвидшення пошуку. Результати пошуку відображаються на окремій вкладці, тому ви можете за потреби переходити між цими результатами та іншими повідомленнями.

Архів листів

Якщо ви вважаєте, що повідомлення може знадобитися в майбутньому, але хочете прибрати його з поштової скриньки, заархівуйте його. Функція архівування дає змогу зберегти листа в архіві у відповідній системній папці.

Щоб заархівувати листа, натисніть кнопку "Архівувати" або клавішу A.

Функціональний блок налаштування робочого середовища

Оформлення поштового клієнту

Завдяки графічним оболонкам, які називаються "Персони", ви зможете змінити оформлення програми поштового клієнту умить. До вибору пропонуються сотні оболонок, які присвячені популярним фільмам, відомим географічним принадам або японським татуюванням. Крім того, ви маєте можливість застосувати одну з тем, яка змінить вигляд значків поштового клієнту.

Інтелектуальні папки

Ця функція дає можливість керувати кількома обліковими записами електронної пошти одночасно, поєднуючи спеціальні папки, наприклад "Вхідні", "Надіслані" або "Архів". Замість того, щоб відкривати папку "Вхідні" для кожного облікового запису окремо, ви можете переглядати всю свою вхідну пошту в одному місці.

Менеджер додатків

Знаходьте і встановлюйте додатки безпосередньо в інтерфейсі програми поштового клієнту. Вам більше не потрібно переходити на веб-сторінку додатків – просто запустіть менеджер додатків. Не впевнені який саме додаток вам потрібен? Рейтинги, рекомендації, описи та зображення додатків допоможуть зробити вибір.

Функціональний блок захисту і безпечного користування поштою

Видалення сміття

Популярні засоби програми поштового клієнту для видалення спаму було оновлено з метою ще більшого захисту. Кожен отримуваний лист проходить

через фільтри програми. Щоразу, коли ви позначаєте повідомлення як спам, програма поштового клієнту запам’ятовує ваше рішення і вдосконалює свої фільтри, щоб ви отримували лише важливі листи. До того ж, поштового клієнту може використовувати фільтри, які надає ваш постачальник послуг електронної пошти, щоб попереджувати потрапляння спаму у вашу поштову скриньку.

Потужний захист конфіденційності

Системи захисту конфіденційності користувача поштового клієнту захищають вас від отримання віддалених зображень. З метою захисту конфіденційності, програма поштового клієнту автоматично блокує віддалені зображення в повідомленнях електронної пошти.

Захист від фішингу

Поштовий клієнт сповіщає про спроби фішингу і, таким чином, захищає вас від шахраїв, які намагаються шляхом обману отримати особисту та конфіденційну інформацію користувачів. Другою лінією захисту є повідомлення, які поштового клієнту відображає тоді, коли ви натискаєте посилання, яке може переспрямувати вас на веб-сайт, відмінний від вказаного URL-адресою, наведеною в листі.

Автоматичне оновлення

Служба оновлення поштового клієнту перевіряє версію вашої програми та сповіщає вас про наявність оновлень системи безпеки. Такі оновлення мають незначний розмір (зазвичай 200-700 КБ) і містять лише потрібні елементи. Це значно пришвидшує встановлення оновлень системи безпеки. Система автоматичного оновлення виконує пошук оновлень для програми поштового клієнту для операційних систем Windows, Mac OS X і Linux більш ніж 40 мовами.

Відкритий вихідний код

Розробка програми поштового клієнту ведеться за принципом "відкритого вихідного коду" тисячами пристрасних, досвідчених розробників та експертів із систем безпеки зі всього світу. Наша відкритість та активна спільнота спеціалістів гарантують високий рівень безпеки наших продуктів і швидке їх

оновлення. До того ж, ми можемо вільно користуватися засобами оцінки та перевірки системи безпеки від сторонніх розробників. Ці засоби дозволяють щоразу піднімати безпеку на новий рівень.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3.

Реалізується два основних процеси:

– Процес отримання електронного листа.

– Процес створення електронного листа.

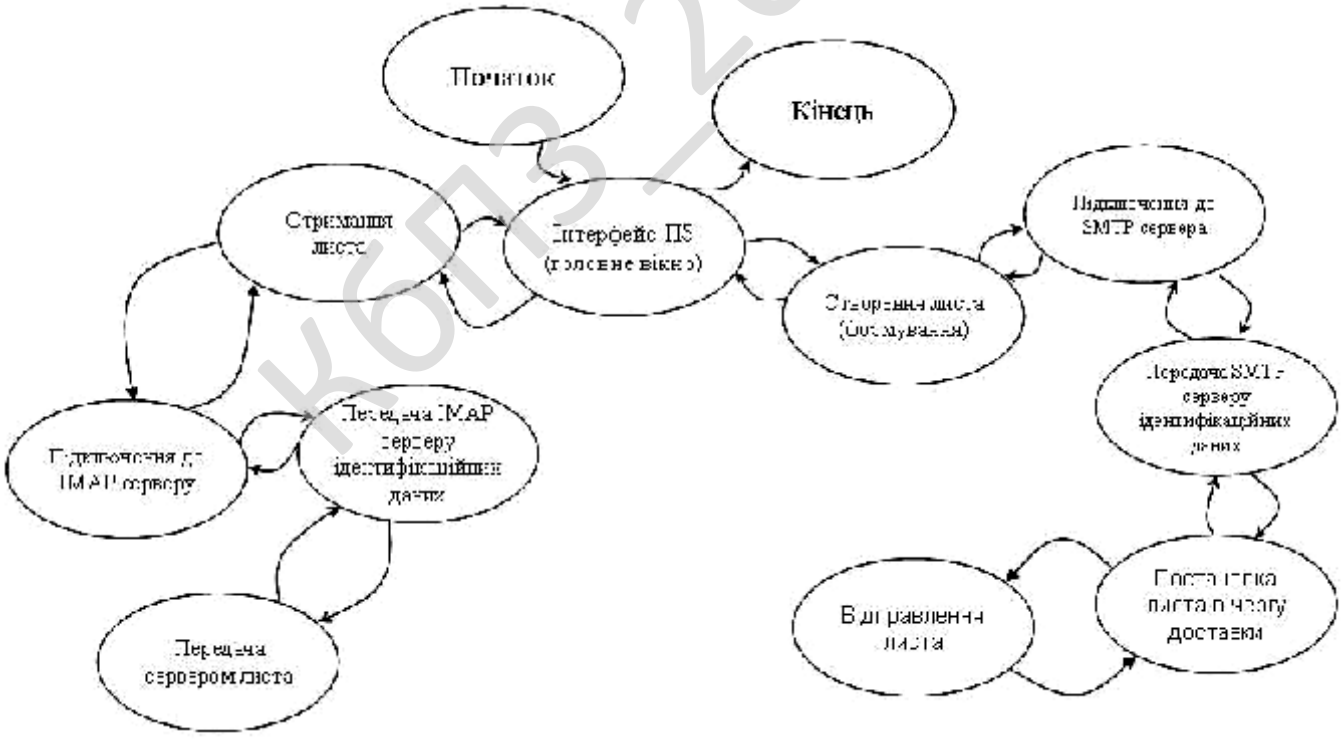

Рисунок 3.3 – Діаграма взаємодії процесів

Процес отримання електронного листа взаємодіє з процесом підключення до IMAP серверу, який, у свою чергу послідовно взаємодіє з процесом передачі IMAP-серверу ідентифікаційних даних, та процесом передачі сервером листа.

Процес створення електронного листа взаємодіє з процесом підключення SMTP-серверу, який у свою чергу взаємодіє послідовно з процесами передачі SMTP-серверу ідентифікаційних даних, постановки електронного листа у чергу доставки та процесом відправлення електронного листа.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних блоків:

1. Оновлення ПЗ. У розробленому ПЗ є механізм оновлення програми через Інтернет. Коли користувач проведе запит на оновлення, з мережі Інтернет по адресі www.EDF.pp.ua/diplom буде завантажено оновлення програми та автоматично встановлено.

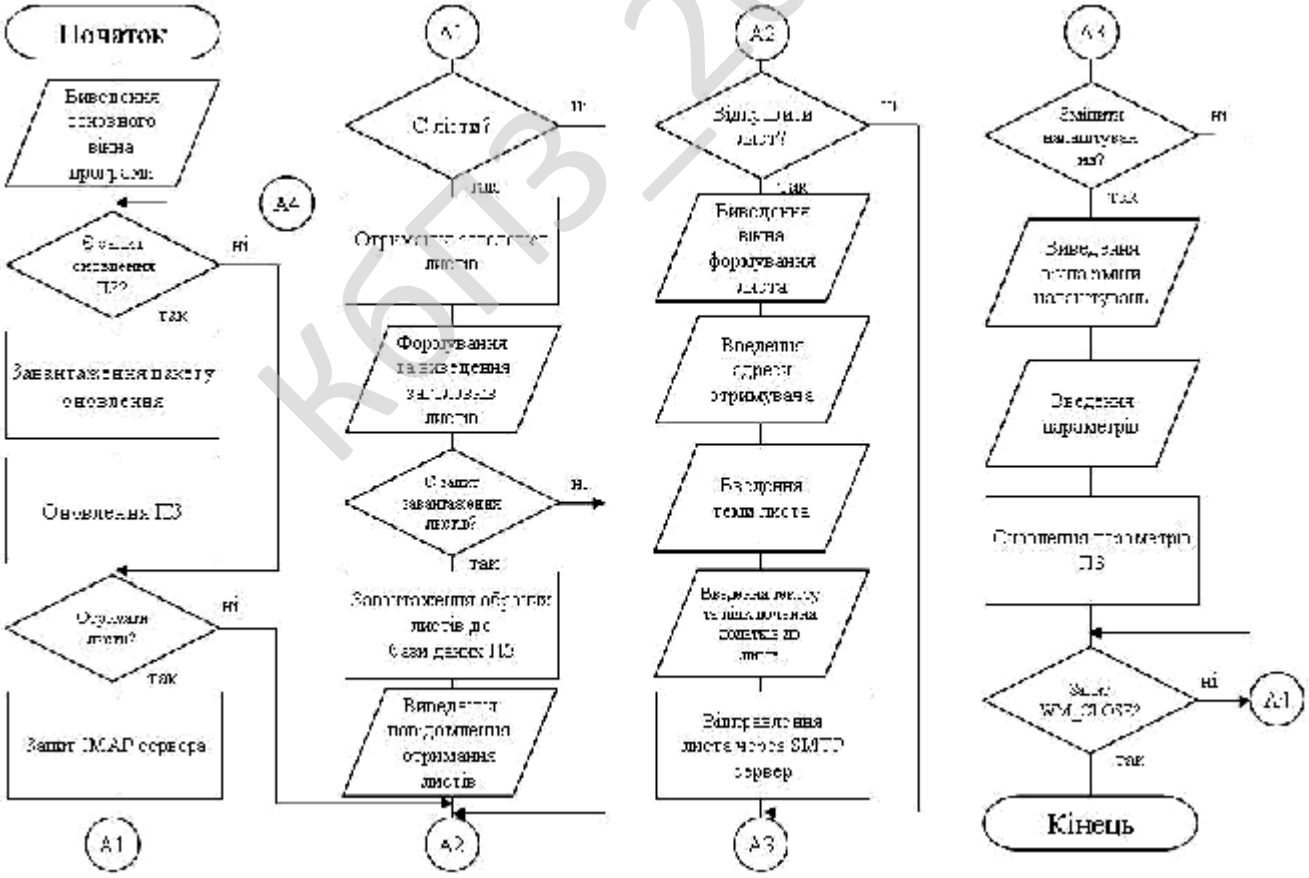

Рисунок 4.1 – Блок-схема основної програми

2. Опис механізму отримання листів.

3. Опис механізму відправлення листів.

4. Блоки налаштування розробленого ПЗ.

На рисунку 4.2 зображено роботу підпрограми формування та відправки листів. Де відображено механізм роботи з SMTP сервером.

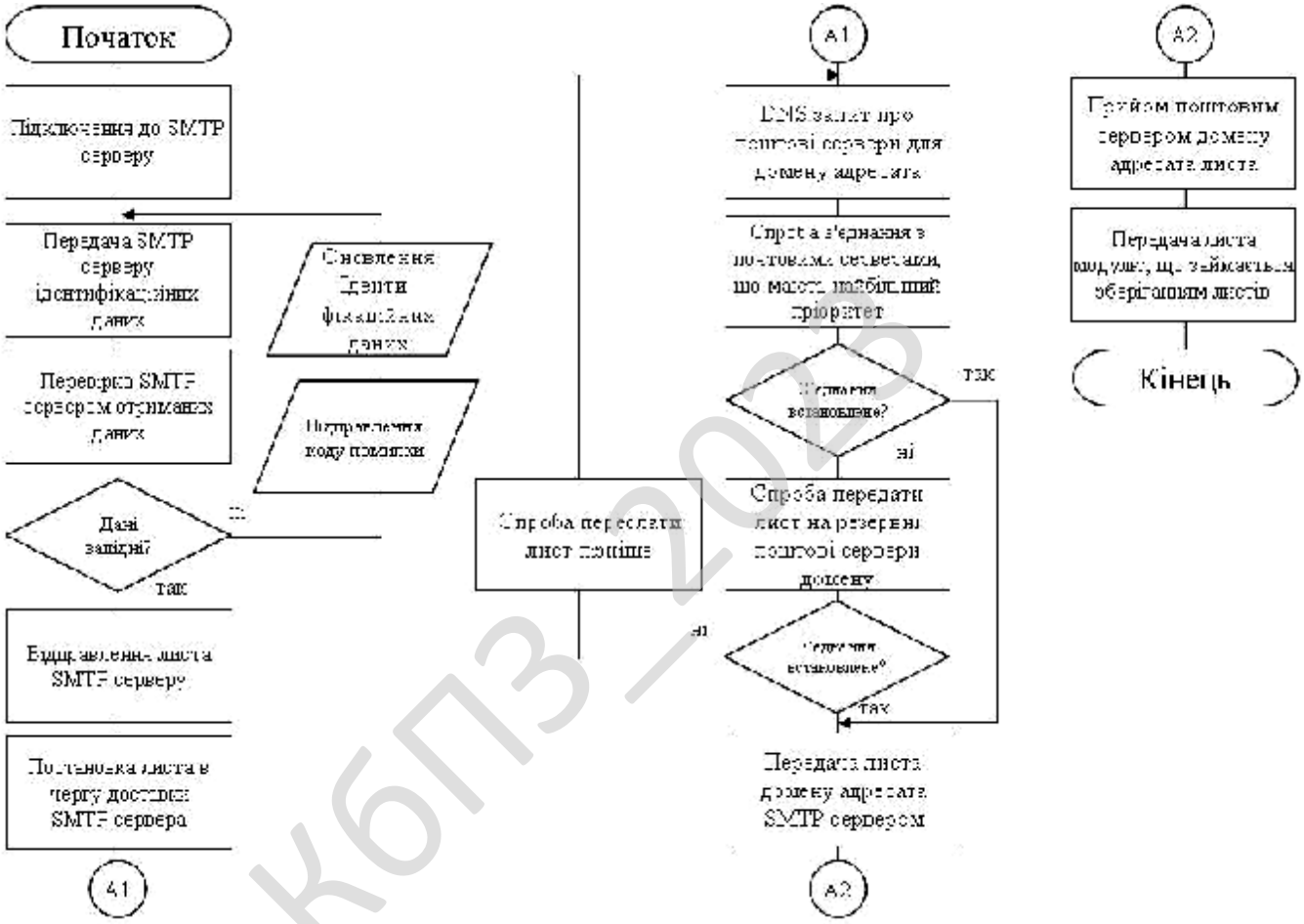

Рисунок 4.2 – Блок-схема підпрограми формування та відправки листів

Розглянемо як було реалізоване розроблене ПЗ.

Реалізація MailSlots – обмін інформацією в мережі Інтернет

Обмін текстовими даними в локальній мережі дуже простий. Для цього необхідні функції:

– CreateMailslot – створення поштового каналу.

– GetMailslotInfo – визначення наявності повідомлення в каналі.

– ReadFile – читання повідомлення з каналу, як з файлу.

– WriteFile – запис повідомлення в канал, як у файл.

Функції роботи з поштовими каналами присутні як в Windows 9x, так і в Windows NT.

Розглянемо частину розробленого вихідного коду – створення поштового каналу (сервер):

```
h := CreateMailSlot(PChar('\\.\mailslot\' + MailSlotName),
0, MAILSLOT_WAIT_FOREVER,nil);
if h = INVALID_HANDLE_VALUE then
begin
  raise Exception.Create('MailSlotServer: Помилка створення каналу !');
// Відправлення повідомлень по поштовому каналі (клієнти).
if not GetMailSlotInfo(h,nil,DWORD(MsgNext),@MsgNumber,nil) then
begin
  raise Exception.Create('TglMailSlotServer: Помилка збору інформації!');
end;
if MsgNext <> MAILSLOT_NO_MESSAGE then
begin
  beep;
// читання повідомлення з каналу й додавання в текст протоколу
  if ReadFile(h,str,200,DWORD(read),nil) then
    MessageText := str
  else
    raise Exception.Create('TglMailSlotServer:
                            Помилка читання повідомлення !');
end;
```

Тепер для зручності використання створимо два компоненти – клієнт і сервер.

```
unit glMSlots;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, extctrls;
type
  TOnNewMessage = procedure (Sender: TObject; MessageText: string) of
                  object;
  TglMailSlotServer = class(TComponent)
  private
    FMailSlotName, FLastMessage: string;
```

```
    FOnNewMessage: TOnNewMessage;
    Timer: TTimer;
//...таймер для прослуховування каналу
    h : THandle;
    str : string[250];
    MsgNumber,MsgNext,read : DWORD;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure Open;
//...створення каналу
    procedure Close;
//...закриття каналу
  protected
    procedure Loaded; override;
    procedure OnTimer(Sender: TObject);
  published
    property MailSlotName: string read FMailSlotName write FMailSlotName;
//... одержання повідомлення
    property OnNewMessage: TOnNewMessage read FOnNewMessage write
                           FOnNewMessage;
  end;
  TglMailSlotClient = class(TComponent)
  private
    FMailSlotName, FServerName, FLastMessage: string;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    function Send(str: string):boolean;
//...відправлення повідомлення
  protected
    procedure Loaded; override;
    procedure ErrorCatch(Sender : TObject; Exc : Exception);
  published
    property ServerName: string read FServerName write FServerName;
    property MailSlotName: string read FMailSlotName write FMailSlotName;
  end;
procedure register;
implementation
procedure register;
begin
```

```
  RegisterComponents('Components', [TglMailSlotServer,
                      TglMailSlotClient]);
end;
constructor TglMailSlotServer.Create(AOwner: TComponent);
begin
  inherited;
  FEnabled := true;
  FMailSlotName := 'MailSlot';
  Timer := TTimer.Create(nil);
  Timer.Enabled := false;
  Timer.OnTimer := OnTimer;
end;
destructor TglMailSlotServer.Destroy;
begin
  Timer.Free;
// закриття каналу
  Close;
  inherited;
end;

procedure TglMailSlotServer.Loaded;
begin
  inherited;
  Open;
end;
procedure TglMailSlotServer.Open;
begin
// створення каналу з ім'ям MailSlotName – по цьому імені до нього
// будуть звертатися клієнти
  h := CreateMailSlot(PChar('\\.\mailslot\' + MailSlotName),
  0, MAILSLOT_WAIT_FOREVER,nil);
h:=CreateMailSlot('\\.\mailslot\MailSlot', 0, MAILSLOT_WAIT_FOREVER,nil);
  if h = INVALID_HANDLE_VALUE then
  raise Exception.Create('TglMailSlotServer: Помилка створення каналу !');
  Timer.Enabled := true;
end;
procedure TglMailSlotServer.Close;
begin
  if h <> 0 then
    CloseHandle(h);
  h := 0;
end;
```

```
procedure TglMailSlotServer.OnTimer(Sender: TObject);
var
  MessageText: string;
begin
  MessageText := '';
// визначення наявності повідомлення в каналі
  if not GetMailSlotInfo(h,nil,DWORD(MsgNext),@MsgNumber,nil) then
    raise Exception.Create('TglMailSlotServer: Помилка збору інформації!');
  if MsgNext <> MAILSLOT_NO_MESSAGE then
  begin
    beep;
// читання повідомлення з каналу й додавання в текст протоколу
    if ReadFile(h,str,200,DWORD(read),nil) then
      MessageText := str
    else
raise Exception.Create('TglMailSlotServer:
                        Помилка читання повідомлення !');
  end;
  if (MessageText<>'')and Assigned(OnNewMessage) then
    OnNewMessage(self, MessageText);
  FLastMessage := MessageText;
end;
constructor TglMailSlotClient.Create(AOwner: TComponent);
begin
  inherited;
  FMailSlotName := 'MailSlot';
  FServerName := '';
end;
destructor TglMailSlotClient.Destroy;
begin
  inherited;
end;
procedure TglMailSlotClient.Loaded;
begin
  inherited;
  Application.OnException := ErrorCatch;
end;
procedure TglMailSlotClient.ErrorCatch(Sender : TObject; Exc : Exception);
var
  UserName: array[0..99] of char;
  i: integer;
begin
```

```
  // одержання ім'я користувача
  i:=SizeOf(UserName);
  GetUserName(UserName,DWORD(i));
  Send('/'+UserName+'/'+FormatDateTime ('hh:mm',Time)+'/'+Exc.message);
  // виведення повідомлення про помилку користувачеві
  Application.ShowException(Exc);
end;
function TglMailSlotClient.Send(str: string):boolean;
var
  strMess: string[250];
  UserName: array[0..99] of char;
  h: THandle;
  i: integer;
begin
  // відкриття каналу : MyServer – ім'я сервера
  // (\\.\\mailslot\xxx – монітор працює на цьому ж ПК)
  // xxx – ім'я каналу
  if FServerName = '' then
    FServerName := '.\';
  h:=CreateFile( PChar('\\' + FServerName + '\mailslot\' + FMailSlotName),
  GENERIC_WRITE, FILE_SHARE_READ, nil, OPEN_EXISTING, 0, 0);
  if h <> INVALID_HANDLE_VALUE then
  begin
    strMess := str;
    // передача тексту помилки (запис у канал і закриття каналу)
    WriteFile(h, strMess, Length(strMess) + 1, DWORD(i), nil);
    CloseHandle(h);
  end;
  Result := h <> INVALID_HANDLE_VALUE;
end;
end.
```

Підключення до серверу за протоколом SMTP та відправлення листа

```
{
Розглянемо параметри які необхідно вказати
smtp – ip адреса smtp сервера
port – порт smtp сервера, за замовчуванням 25
from – адреса відправника
dest – адреса одержувача
subject – тема листа
body – текст листа
Повертає True якщо лист був успішно відправлено.
```

```
}
function mail(smtp: string; port: integer; from, dest, subject,
  body: string): bool;
const
  cl = #13#10;
var
WSAData: TWSAData;
  Host: TSockAddrIn;
  Sock: TSocket;
  res: Integer;
  buff: array[1..255] of Char;
  { відправляємо дані через сокет }
  procedure senddata(str: string);
  var
    i: integer;
  begin
    for i := 1 to Length(str) do
      if send(Sock, str[i], 1, 0) = SOCKET_ERROR then
        exit;
  end;
  { одержуємо відповідь від команди }
  function recvdata(accept: string): bool;
  var
    buff: array[1..255] of Char;
  begin
    res := recv(Sock, buff, SizeOf(buff), 0);
    Result := (Res = SOCKET_ERROR) or (Copy(buff, 1, 3) = accept);
  end;
begin
  try
    result := false;
// ініціалізація сокета
    WSAStartUp(257, WSAData);
    Sock := socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
    if Sock = INVALID_SOCKET then
      Exit;
// встановлюємо хост та порт сервера
    res := inet_addr(PChar(smtp));
    if res <= 0 then
      exit;
    Host.sin_family := AF_INET;
    Host.sin_port := htons(port);
```

```
    Host.sin_addr.S_addr := res;
// підключення до сервера
    if connect(Sock, Host, SizeOf(Host)) > 0 then
      Exit;
    { вітання сервера }
    if not recvdata('220') then
      Exit;
    { EHLO }
    senddata('EHLO' + cl);
    if not recvdata('250') then
      Exit;
    { MAIL FROM: }
    senddata('MAIL FROM:' + from + cl);
    if not recvdata('250') then
      Exit;
    { RCPT TO: }
    senddata('RCPT TO:' + dest + cl);
    if not recvdata('250') then
      Exit;
    { DATA }
    senddata('DATA' + cl);
    if not recvdata('354') then
      Exit;
// відправляємо текст повідомлення
    senddata('Subject:' + subject + cl + cl + body + cl + '.');
    if not recvdata('250') then
      Exit;
    { відключаємося від сервера }
    senddata('QUIT' + cl);
    result := true;
  finally
    { убиваємо сокет }
    closesocket(sock);
    WSACleanup;
  end;
end;
```

Реалізація механізму одержання одиночного листа розробленому ПЗ

```
procedure TForm1.Button1Click(Sender: TObject);
begin
POP3.Host:='mail.58r.ua';
// адреса поштового сервера
```

```
POP3.Port:=110;
// порт по якому буде здійснюватися підключення
POP3.Username:='test.ua';
// Логін користувача
POP3.Password:='11?;
// пароль користувача
IdMessage.Clear;
// очищення буфера для повідомлення
POP3.Connect;
// підключення за протоколом POP3,
/ по налаштуваннях у компоненті IdPOP3, з ім'ям POP3
Memo1.Clear;
// очищення компонента memo для відображення тексту листа.
POP3.Retrieve(1,IdMessage);
// одержання одного повідомлення
Memo1.Lines.AddStrings(IdMessage.Body);
// передача повідомлення з компонента IdMessage в memo
POP3.Delete(1);
// видалення із сервера, отриманого повідомлення
POP3.Disconnect;
// розрив зв'язку
end;
```

Реалізація механізму одержання всіх листів у скрині розробленому ПЗ

```
procedure TForm1.Button2Click(Sender: TObject);
label f;
var
mailcicl:integer;
// лічильник одержуваних листів
addr:string;
// папка в якій будуть зберігатися одержувані листи
begin
POP3.Host:='mail. ua';
// адреса поштового сервера
POP3.Port:=110;
// порт по якому буде здійснюватися підключення
POP3.Username:='test+58r.ua';
// Логін користувача
POP3.Password:='12345666?;
// пароль користувача
addr:=ExtractFilePath(Application.ExeName);
// одержання адреси папки, у якій перебувати наша програма
```

```
POP3.Connect;
// підключення до сервера
for mailcicl:=1 to 10 do
// цикл одержання листів
begin
if POP3.CheckMessages<1 then goto f ;
// Якщо листів немає, то вийти з циклу
IdMessage.Clear;
// очищення буфера для повідомлення
Memo1.Clear;
// очищення компонента memo для відображення тексту листа.
POP3.Retrieve(1,IdMessage);
// одержання одного повідомлення
Memo1.Lines.AddStrings(IdMessage.Body);
// передача повідомлення з компонента IdMessage в memo
Memo1.Lines.SaveToFile(addr+inttostr(mailcicl)+'.txt');
// збереження листа в папці, де перебуває наша програма
POP3.Delete(mailcicl);
// видалення листа на сервері
end;
f:
// мітка виходу із циклу
POP3.Disconnect;
// розрив з'єднання із сервером
end;
```

Кожне повідомлення має атрибути – це тема повідомлення, важливість і інші елементи..

Механізм одержання атрибутів листа

```
procedure TForm1.Button3Click(Sender: TObject);
var
i,numPosts: Integer;
begin
POP3.Host:='mail.ua';
// адреса поштового сервера
POP3.Port:=110;
// порт по якому буде здійснюватися підключення
POP3.Username:='test+58r.ua';
// логін користувача
POP3.Password:='11?;
// пароль користувача
IdMessage.Clear;
```

```
// очищення буфера для повідомлення
POP3.Connect;
// підключення за протоколом POP3, по настроюваннях у
// компоненті IdPOP3, з ім'ям POP3
Memo1.Clear;
// очищення компонента memo для відображення тексту листа.
POP3.Retrieve(1,IdMessage);
// одержання одного повідомлення
Memo1.Lines.AddStrings(IdMessage.Body);
// передача повідомлення з компонента IdMessage в memo
// вивід у компоненти Label інформації про повідомлення.
Label1.Caption := IdMessage.From.Text;
Label2.Caption := IdMessage.Recipients.EmailAddresses;
Label3.Caption := IdMessage.CCList.EMailAddresses;
Label4.Caption := IdMessage.Subject;
Label5.Caption := FormatDateTime('dd mmm yyyy hh:mm:ss', IdMessage.Date);
Label6.Caption := IdMessage.ReceiptRecipient.Text;
Label7.Caption := IdMessage.Organization;
POP3.Delete(1);
// видалення із сервера, отриманого повідомлення
POP3.Disconnect;
// розрив зв'язку
end;
```

Відправлення пошти здійснюється за протоколом SMTP, компонент IdSMTP (Indy Clients). Відправлення пошти здійснюватися після авторизації сервері.

Реалізація механізму відправлення одиночного листа

```
procedure TForm1.Button4Click(Sender: TObject);
begin
SMTP.Host:='mail.58r.ua';
// адреса поштового сервера
SMTP.Port:=25;
// порт по якому буде здійснюватися підключення
SMTP.Username:='test+58r.ua';
// логін користувача
SMTP.Password:='12345666?;
// пароль користувача
SMTP.AuthenticationType:=atLogin;
// тип підключення до сервера – з авторизацією
with IdMessage do
```

```
begin
Body.Assign(Memo1.Lines);
// передача даних з memo у компонент IdMessage, для наступного відправлення
From.Text := 'test@58r.ua';
// майл відправника листа
Recipients.EMailAddresses := 'test@58r.ua';
// майл одержувача листа
Subject := 'Programmersclub.ua';
// теми листа
end;
SMTP.Connect;
// з'єднання з поштовим сервером
try
showmessage('Відбулося підключення до сервера');
// повідомлення про вдале підключення до сервера
SMTP.Send(IdMessage);
// відправлення листа на сервер
finally
SMTP.Disconnect;
// розрив з'єднання
end; end;
```

Перевірка адресатів

```
function IsValidEmail(const Value:string):boolean;
  function CheckAllowed(const s:string):boolean;
  var
    i: integer;
  begin
    Result:= false;
    for i:= 1 to Length(s) do
    begin
      if not (s[i] in ['a'..'z', 'A'..'Z', '0'..'9', '_', '-', '.']) then
        Exit;
    end;
    Result:= true;
  end;
var
  i: integer;
  namePart, serverPart: string;
begin
  Result:= false;
  i:= Pos('@', Value);
```

```
    if i = 0 then
      Exit;
    namePart:= Copy(Value, 1, i - 1);
    serverPart:= Copy(Value, i + 1, Length(Value));
    if (Length(namePart) = 0) or ((Length(serverPart) < 5)) then
      Exit;
    i:= Pos('.', serverPart);
    if (i = 0) or (i > (Length(serverPart) - 2)) then
      Exit;
    Result:= CheckAllowed(namePart) and CheckAllowed(serverPart);
  end;
```

Відправлення листа й вказівка теми, тексту повідомлення

Для відправлення листа за допомогою зареєстрованого клієнта використовується функція Windows – ShellExecute, де як аргумент передається рядок протоколу Mailto. Для цього зробіть наступне:

1. У розділі uses підключите ShellAPI.

2. В оброблювачі OnClick мітки або клавіші введіть наступний код.

```
ShellExecute(Handle, 'open',
'mailto:lalala@lala.ua?par1=value1&par1=value1&...', nil, nil, SW_SHOWNORMAL);
```

Третій параметр функції ShellExecute() – це рядок відповідно до протоколу mailto і правилами оформлення URL:

– mailto: тип протоколу (може бути http: у цьому випадку оставшая URL і параметри запиту).

– lalala@lala.ua – адреса одержувача, можна включати кілька адрес, розділяючи із символом ";".

– ? – роздільник параметрів від адреси.

– par1=value1 – ім'я параметра і його значення.

– & – роздільник параметрів.

Протокол Mailto має наступну форму.

MAILTO:Recipients&Parameters

– Поле Recipients може бути порожньою, одиночною адресою й складатися з декількох адрес, розділених символом ";".

– Поле Parameters додатково і якщо воно є те повинне бути відділене символом "&". Параметри повинні з'являтися у формі пари name/value. Наступний список описує можливі параметри:

PARAMETER DESCRIPTION:

– CC = Carbon copy (додаткові одержувачі).

– BCC = Blind carbon copy (додаткові одержувачі, адреси яких не показуються іншим одержувачам).

– SUBJECT = Subject text (тема).

– BODY = Body text (текст).

Всі дані вказуються в параметрах повинні бути так звані Internet safe characters. Використовуйте %0d для символу переклад рядка (LF), %20 для пробілу й так далі.

mailto:email1;email2&cc=email3?subject=Це%20тема&

body=це%20текст%20листа%0dце%20інший%20рядок

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою національного стандарту захисту інформації на основі алгоритму шифрування/дешифрування ДСТУ 4145-2002 з використанням еліптичних кривих над двійковим розширеним полем Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній. Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$. Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G,\ P_m + k \times P_B\}. \qquad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m. \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було додано відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Нехай задано просте число $p > 4$. Тоді еліптичною кривою E, визначеною над розширеним двійковим полем полем F_{2^m}, називається безліч пар чисел $(x, y), x, y \in F$, що задовольняють тотожності:

$$y^2 \equiv x^3 + a \cdot x + b (\mathrm{mod}\, 2^m), \quad (4.3)$$

де $4 \cdot a^3 + 27 \cdot b^2$ не рівно з нулю по модулю 2^m.

Інваріантом еліптичної кривої називається величина $J(E)$, що задовольняє тотожності:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, 2^m). \quad (4.4)$$

Коефіцієнти a, b еліптичної кривої E, по відомому інваріанту $J(E)$, визначаються таким чином:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, 2^m) \\ b \equiv 2k (\mathrm{mod}\, 2^m) \end{cases} \text{де } k \equiv \frac{J(E)}{1728 - J(E)} (\mathrm{mod}\, 2^m), J(E) \neq 0 \text{ або } 1728. \quad (4.5)$$

Пари (x,y), що задовольняють тотожності (4.1), називаються точками еліптичної кривої E, x та y – відповідно x– та y-координатами точки.

Точки еліптичної кривої позначатимемо $Q(x,y)$ або просто Q. Дві точки еліптичної кривої рівні, якщо рівні їх відповідні x– і y-координати.

На безлічі всіх точок еліптичною кривою Е введемо операцію додавання, яку позначатимемо знаком "+". Для двох довільних точок $Q_1(x_1, y_1)$ та $Q_2(x_2, y_2)$ еліптичної кривої Е, розглянемо декілька варіантів.

Нехай координати точок Q_1 та Q_2 задовольняють умові $x_1 \neq x_2$. В цьому випадку їх сумою називатимемо точку $Q_3(x_3, y_3)$ координати якої визначаються порівняннями:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, 2^m). \quad (4.6)$$

Якщо виконана рівність $x_1 = x_2$ та $y_1 = y_2 \neq 0$, то визначимо координати точки Q_3 таким чином:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, 2^m). \quad (4.7)$$

У разі, коли виконана умова $x_1 = x_2$ та $y_1 = -y_2 (\mathrm{mod}\, p)$, суму точок Q_1 та Q_2 називатимемо нульовою точкою О, не визначаючи її х– і у-координати. В цьому випадку, точка Q_2 називається запереченням точки Q_1. Для нульової точки О виконана рівність:

$$Q + 0 = 0 + Q = Q, \quad (4.8)$$

де Q – довільна точка еліптичної кривої Е.

Щодо введеної операції складання безліч всіх точок еліптичною кривою Е, разом з нульовою точкою, утворюють кінцеву абельову (комутативну) групу порядку т, для якого виконана нерівність:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p} \quad (4.9)$$

Точка Q називається точкою кратності k, або просто – кратною точкою еліптичної кривої E, якщо для деякої точки P виконана рівність:

$$Q = P + ... + P = kP \quad (4.10)$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

ПЗ розроблялось як потужний і в той же час простий поштовий клієнт (рисунок 5.1), який підходить як для професіоналів, так і для простих користувачів ПК.

Багато можливостей включено в нього для автоматизації роботи з електронною поштою:

– Ефективно обробляється велика кількість листів.

– Простий для використання інтерфейс.

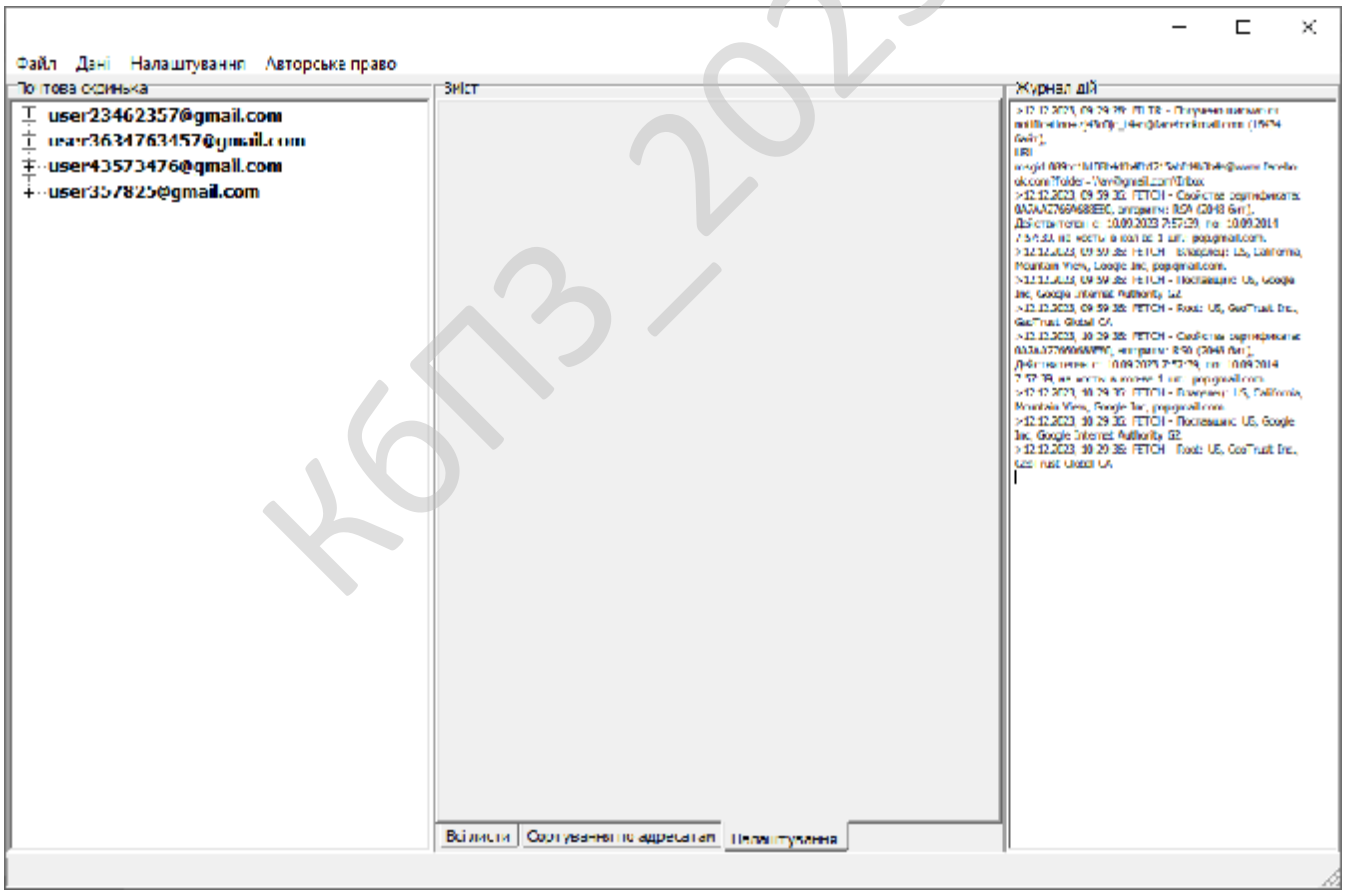

Рисунок 5.1 – Головне вікно програми

– Підтримується велика кількість почтових скриньок.

– Використовуються можливості мультизадачності ОС Windows.

– Попередня обробка пошти на сервері, без завантаження листів на локальний диск.

– Універсальний і зручний редактор листів.

– Удосконалені шаблони і система правил для сортування листів.

– Вбудована адресна книга.

– Підвищена безпека.

Розроблене ПЗ в першу чергу необхідно для ведення ділового листування і, виходячи з цього , являє собою легкий та ефективний засіб керування поштою.

На рисунку 5.2 зображено форму авторського права розробника.

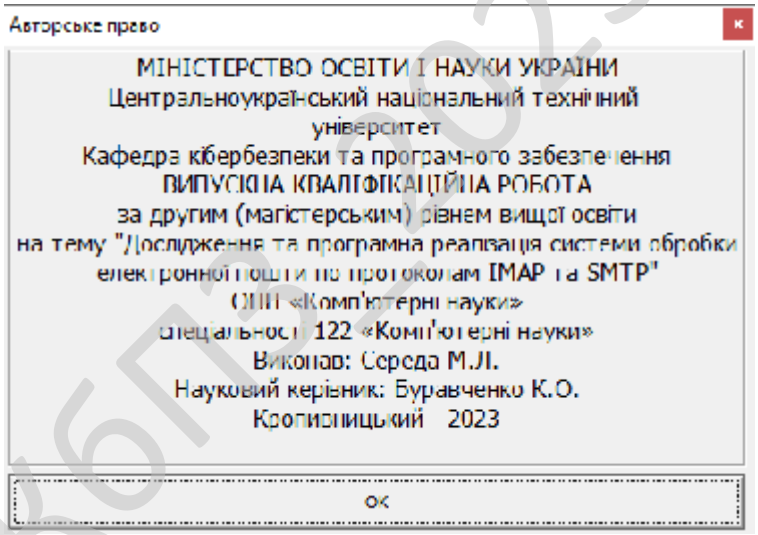

Рисунок 5.2 – Довідка автора

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи обробки електронної пошти по протоколам IMAP та SMTP.

Метою розробки є дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

Об'єктом дослідження є процес обробки електронної пошти по протоколам IMAP та SMTP.

Предметом дослідження є методи обробки електронної пошти по протоколам IMAP та SMTP.

Методи дослідження базуються на методах теорії телетрафіку, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод обробки електронної пошти по протоколам IMAP та SMTP.

– Розроблено вітчизняний продукт обробки електронної пошти по протоколам IMAP та SMTP, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі була досліджена та розроблена програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	20
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	20000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: W_i – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 65 = 109 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	109	Ф 7.1-7.4
Впровадження	13	Д13
Всього	150	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{150 \cdot 1}{48 - 4} = 3,4 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	9	810	13,5
Монітор	60	9	540	9
Клавіатура	30	9	270	4,5
Маніпулятор «мишка»	30	9	270	4,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор– маршрутизатор	30	3	90	1,5
Кабельні господарства ЛВС на 1 м. п.	2,5	280	700	11,67
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	52,33

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{52,33 \cdot 2}{1,2} = 87,2 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 87,2/(48 \cdot 8) = 0,25 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,5	27000	27000
Продакт-менеджер	0,25	20000	10000
Інженер-програміст	3,4	26500	180200
Інженер-електронщик	0,25	20000	10000
Інженер-системотехнік	0,25	20000	10000
Адміністратор мережі	0,5	20000	20000
Системний програміст	0,25	20000	10000
Дизайнер WEB	0,25	20000	10000
Інженер-верстальник	0,25	20000	10000
Бухгалтер-економіст	0,5	20000	20000
Всього за період розробки	$R_{сп}$ = 6,4	-	$Ф_{роб}$ = 307200

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{307200}{6,4 \cdot 48} = 1000 \ грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину SUPERCOMP за 20.10.23 – джерело https://supercomp.kiev.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	AMD Ryzen 3 4100 (100-100000510BOX) AM4, 4 ядра, 8 потоків, 3.8 GHz, 4.0 GHz, TDP – 65 Вт, 7nm, L1: 256KB, L2: 2MB, L3: 4MB, Zen 2, BOX	-
Системна плата	ASRock A520M-HDV сокет – AM4, DDR4, 64 ГБ, 4733 MHz, LAN – 1 Гбіт/с, D-Sub (VGA), DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	GeForce GT710 2048Mb	-
Жорсткий диск	SSD M.2 2280 240GB Apacer (AP240GAS2280P4-1) 240 GB, 3D TLC, M.2, PCI Express 3.0 x4	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot (PSD48G240081) DDR4, 8 ГБ, В наборі – 1	-
Блок живлення	Casecom 450W (CM 450 ATX) ATX, 450 Вт, 24 pin, CPU – 4pin, , SATA – 2, Peripheral – 4, +12V1 – 18A, 1x120 мм, 140 x 150 x 87 мм	-
Корпус	AeroCool Talon-G-BK-v1 Black (ACCM-PV43013.11), Miditower, Front 2x12 см, Top 1x12 см, GPU – до 325 мм, CPU – 155 мм, 198 x 412 x 395 мм	-
Кулер	–	–

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат (МФУ)	1	5965	596,5	6561,5
Всього	–	–	–	199177

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Нематеріальні активи			
4. Нематеріальні активи	20000	10	2000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	5000
Разом	$К_р$ = 1807208		$А_р$ = 176988,5

Примітка: вартість автомобіля взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua/catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 1000 \cdot 150/20 = 7503 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 7503 \cdot 10 \cdot 0{,}01 = 750{,}3 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(7503+750{,}3) = 3054 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 1/2 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1 \cdot 1/2 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 3):

$$З_{М2} = \sum Ц_д\,, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 37,33 грн./шт.

$$З_{М2} = 37{,}33 \cdot 3 = 112 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з.\,, \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 112 + 1702)/20 = 96 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 20 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 176989 \cdot 2/(20 \cdot 12) = 1475 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 7305+730,5+3054+1125+96+1125+1475 = 14910,5 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	7305
2. Додаткова зарплата виконавців	$З_д$	730,5
3. Відрахування на соціальні потреби	$C_{оц}$	3054
4. Загальногосподарські витрати	$Г_{осп}$	1125
5. Витрати на матеріали	$З_М$	96
6. Освоєння нових операційних систем, мов програмування	$О_п$	1125
7. Амортизація основних фондів	$A_м$	1475
8. Повна собівартість програмного забезпечення	$C_п$	14910,5
9. Плановий прибуток	$П_р$	8200,78
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	23111,28
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4622,26
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	27733,54

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 55 \cdot 14910{,}5 = 8200{,}78 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	27733
Всього капітальних витрат	–	27733

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	42273	18117
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	6933
Всього витрат за рік	$І$	42273	25050

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування сервера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилось з 350 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 350 \cdot 90 \cdot 1{,}1 \cdot 1{,}22 = 42273 \text{ грн},$$

до:

$$З_{р\,нов} = 150 \cdot 90 \cdot 1{,}1 \cdot 1{,}22 = 18117 \text{ грн}.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Визначити різницю споживання електроенергії при впровадженні систем не має можливості, тому витрати на електроенергію в розрахунку приймаємо рівними нулю.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	27733	–	6933,25
Всього відрахувань	-	–	27733	–	6933,25

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (23111 - 14910) \cdot 20 - (0,05 \cdot 1408000 + 0,5 \cdot 199177 + \\ +0,2 \cdot 143000 + 0,25 \cdot 37031 + 0,1 \cdot 20000) \cdot 2/12 = 129046 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника.

$$T_в = \frac{1807208}{(23111 - 14910) \cdot 20 \cdot 12 / 2} = 1,8 \text{ років}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	20
2. Повна собівартість розробленої програми	Грн.	14910
3. Ціна розробленої програми	Грн.	23111
4. Плановий прибуток від реалізації розробленої програми	Грн.	8201
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1807208
7. Загальний прибуток від реалізації програмної продукції	Грн.	164020
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	129046
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,8
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	27733
11.Величина економічного ефекту у користувача програмної продукції	Грн.	10289
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Роки	1,6

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (42273\text{-}25050)\text{-}0{,}25 \cdot 27733 = 10289 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{26839}{42273 - 25050} = 1{,}6 \text{ роки}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207 [4], який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [2]. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

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

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	7,3
Довжина	8
Висота	3,1

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,5
Обсяг, V	$м^3$	не менше 20.0	20,1

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працює 9 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (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», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура °С	Воло-гість,%	Швидкість повітря, м/	Температура °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	45-60	0,12
Тепла	23-25	50-70	0,1	23-25	52-70	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP Laser 107a, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути

освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору).

8.5 Розрахункова частина

Початкові дані для розрахунку захисного штучного заземлення: опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*.

Для захисного штучного заземлення застосовуються вертикальні електроди з металевого прутка діаметром 30 мм. (*D=30 мм.=0,03 м.)* довжиною *L*=2,5 *м.* та горизонтальний електрод – металева полоса з перетином 40*4 мм. тип грунта – глина (питомий опір 40 Ом*м). Відстань між вертикальними заземлювачами (електродами) *A*=3 *м*.

Глибина закладення горизонтального контура заземлення *t= 0,8 м*.

Умовна товщина верхнього шару грунта: *H=0,4 м*. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр. 13-16 [6], або аналогічної.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,8+2,5/2=2,05 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho_2 = 1,36*40=54,5 \text{ Ом*м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідной климатичной зони у багатошаровому грунті [8];

$\rho_1 = 50$ *Ом*м.* – табличне значення питомого опору верхнього шару грунта [6];

$\rho_2 = 40$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта [8].

Опір розтіканню електричного струму одного електрода вертикального заземлювача [8]:

$$R_o = 0,366\frac{\rho}{L}\left(lg\frac{2L}{D}+\frac{1}{2}lg\frac{4T+L}{4T-L}\right)= 0,366\frac{54,5}{2,5}\left(lg\frac{2\cdot 2,5}{0,03}+\frac{1}{2}lg\frac{4\cdot 2,05+2,5}{4\cdot 2,05-2,5}\right)= 20,1\ Ом.$$

Відношення *A/L=3/2,5=1,2.*

Визначаємо коефіцієнт екранування вертикальних електродів *Кев=0,8* при попередній (орієтовній) кількості вертикальних електродів, яке дорівнює 4 [8].

Визначаємо необхідну кількість вертикальних заземлювачів (без врахування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом* :

$$N=R_O /(Кев\ R_{ЗН})= 20,1 / (0,8*4)=5,87 \approx 6\ шт.$$

Визначаємо довжину з'єднуючої полоси [8]:

$$L_П =1,05*A*N= 1,05*3*6=18,8\approx 19\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси [8]:

$$R_П =0,366(\rho_2*K_П/L_П)lg(2(L_П *L_П)/(К*t))=$$

$$=0,366(40*5/16)*[lg(2*16*16)/(0,04*0,8)]=16,5\ Ом.$$

де $K_П=5$ – табличне значення коефіцієнта сезонності для відповідної кліматичної зони з'єднуючої полоси: [8].

Загальний опір розтіканню електричного струму заземлювача [8]:

$$R =(R_O*R_П)/(R_O*\eta_П+ N*R_П*Кев)=$$

$$=(20,1* 16,5)/(20,1* 0,75+ 6* 16,5* 0,8)=3,32\ Ом.$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [8].

Умова $R \le R_{ЗН}$ виконується ($3,32 \le 4$).

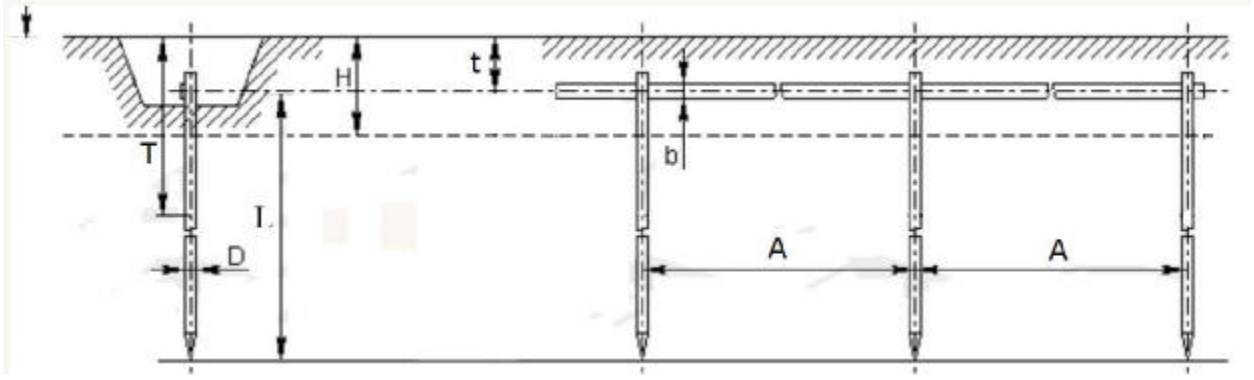

Рисунок 8.1 – Схема штучного заземлення

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного заземлення. Розроблено заходи з охорони праці.

Список використаних джерел інформації

1. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2.007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

4. 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». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

5. 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. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення: 16.06.2023).

6. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

7. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

8. 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. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення: 16.06.2023).

9. Методичні рекомендації до виконання розділу «Заходи з охорони праці та техніки безпеки» у магістерській дисертації / Л.Д. Третякова; М-во освіти і науки України, Національний технічний університет україни «Київський політехнічний інститут» – Київ, КПІ, 2014. – 26 с. – Режим доступу до ресурсу: http://surl.li/dhulo (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи обробки електронної пошти по протоколам IMAP та SMTP.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів обробки електронної пошти по протоколам IMAP та SMTP.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем обробки електронної пошти по протоколам IMAP та SMTP.

– Досліджена система обробки електронної пошти по протоколам IMAP та SMTP.

– На основі отриманих результатів досліджень створена програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання обробки електронної пошти по протоколам IMAP та SMTP.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 4145-2002.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 10289 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,6 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Середа М.Л. Дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

4. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

5. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

6. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

7. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

8. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

9. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

10. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

11. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

12. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

13. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

14. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

15. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

16. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

17. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

18. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

19. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

20. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

21. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

22. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

23. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

24. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

25. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

26. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

27. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

28. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

29. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

30. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

31. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

32. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

33. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

34. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

35. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

36. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

37. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

38. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

39. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних

систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

40. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

41. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

42. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

43. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

44. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

45. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

46. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

47. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

48. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

49. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 121-127.

50. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. – Випуск 2 (46) – Х.: ХУПС – 2016. – С. 146-149.

51. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп’ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

52. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи обробки електронної пошти по протоколам IMAP та SMTP.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 33-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи обробки електронної пошти по протоколам IMAP та SMTP;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 100 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 19.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Буравченко К.О.

Дослідження та програмна реалізація системи обробки електронної пошти по протоколам IMAP та SMTP

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 48

Літера: РП

Кропивницький – 2023 року

Файл MyMail_IMAP_SMTP.dpr – файл проекту

```
program MyMail_IMAP_SMTP;

uses
  Forms,
  frmMain in 'frmMain.pas' {frm_Main},
  DataModule in 'DataModule.pas' {DM: TDataModule},
  frmMail_IMAP_SMTPSettings in 'frmMail_IMAP_SMTPSettings.pas'
{frm_Mail_IMAP_SMTPSettings},
  frmSettings in 'frmSettings.pas' {frm_Settings},
  LetterStorage in 'LetterStorage.pas',
  frmNewLetter in 'frmNewLetter.pas' {frm_NewLetter},
  frmAskOnDuplicate in 'frmAskOnDuplicate.pas' {frm_AskOnDuplicate},
  about in 'about.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.Title := 'Дослідження та програмна реалізація системи обробки
електронної пошти по протоколам IMAP та SMTP, розроблено Студентом Середа Максим
Леонідович';
  Application.CreateForm(Tfrm_Main, frm_Main);
  Application.CreateForm(TDM, DM);
  Application.CreateForm(Tfrm_Mail_IMAP_SMTPSettings,
frm_Mail_IMAP_SMTPSettings);
  Application.CreateForm(Tfrm_Settings, frm_Settings);
  Application.CreateForm(Tfrm_NewLetter, frm_NewLetter);
  Application.CreateForm(Tfrm_AskOnDuplicate, frm_AskOnDuplicate);
  Application.CreateForm(TForm1, Form1);
  Application.HintPause:=200;
  Application.Run;
end.
```

Файл frmMain.pas – основна програма

```
unit frmMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ToolWin, ComCtrls, ExtCtrls, StdCtrls, XPMan,
  IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient,
  IdMessageClient, IdPOP3, ActnList, OleCtrls, SHDocVw,
  {HTTPApp, {HTTPProd,}
  LetterStorage,frmMail_IMAP_SMTPSettings, CoolTrayIcon, about;

type
  Tfrm_Main = class(TForm)
    MainMenu: TMainMenu;
    Splitter1: TSplitter;
    Splitter2: TSplitter;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    CoolBar: TCoolBar;
    Tree_Mail_IMAP_SMTPBoxes: TTreeView;
    ListView_LettersHeaders: TListView;
    ToolBar1: TToolBar;
    btn_CheckMail_IMAP_SMTP: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    XPManifest1: TXPManifest;
    Panel_Content: TPanel;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    ActionList1: TActionList;
    aCheckMail_IMAP_SMTP: TAction;
    aSendMail_IMAP_SMTP: TAction;
    aMail_IMAP_SMTPSettings: TAction;
    aExit: TAction;
    ToolBar2: TToolBar;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    aNewLetter: TAction;
    ToolButton7: TToolButton;
    MemoLog: TMemo;
    Splitter3: TSplitter;
    aSettings: TAction;
    ToolButton8: TToolButton;
    ToolButton9: TToolButton;
    N11: TMenuItem;
    Web_LetterViewer: TWebBrowser;
    Splitter4: TSplitter;
    List_Attachmant: TListView;
    Memo_LetterBody: TMemo;
    Popup_Attachment: TPopupMenu;
    mnuSaveAtt: TMenuItem;
    SaveDialog_Att: TSaveDialog;
    PopupMenuHeaders: TPopupMenu;
    mnu_DeleteLetter: TMenuItem;
    mnu_Answer: TMenuItem;
    mnu_Resend: TMenuItem;
    N12: TMenuItem;
```

```
    aAnswer: TAction;
    aResend: TAction;
    PopupMenu_Mail_IMAP_SMTPBoxes: TPopupMenu;
    N13: TMenuItem;
    Jnghfdbnmgjxne1: TMenuItem;
    N14: TMenuItem;
    About1: TMenuItem;
    procedure aMail_IMAP_SMTPSettingsExecute(Sender: TObject);
    procedure aExitExecute(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure aNewLetterExecute(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure aSettingsExecute(Sender: TObject);
    procedure Tree_Mail_IMAP_SMTPBoxesClick(Sender: TObject);
    procedure aCheckMail_IMAP_SMTPExecute(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure List_AttachmantClick(Sender: TObject);
    procedure Web_LetterViewerBeforeNavigate2(Sender: TObject; const pDisp:
IDispatch; var URL, Flags, TargetFrameName, PostData, Headers: OleVariant; var
Cancel: WordBool);
    procedure ListView_LettersHeadersSelectItem(Sender: TObject; Item:
TListItem; Selected: Boolean);
    procedure List_AttachmantMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
    procedure mnuSaveAttClick(Sender: TObject);
    procedure Popup_AttachmentPopup(Sender: TObject);
    procedure mnu_DeleteLetterClick(Sender: TObject);
    procedure PopupMenuHeadersPopup(Sender: TObject);
    procedure aAnswerExecute(Sender: TObject);
    procedure aResendExecute(Sender: TObject);
    procedure Tree_Mail_IMAP_SMTPBoxesDragDrop(Sender, Source: TObject; X, Y:
Integer);
    procedure Tree_Mail_IMAP_SMTPBoxesDragOver(Sender, Source: TObject; X, Y:
Integer; State: TDragState; var Accept: Boolean);
    procedure TrayClick(Sender: TObject);
    procedure TrayMinimizeToTray(Sender: TObject);
    procedure N13Click(Sender: TObject);
    procedure PopupMenu_Mail_IMAP_SMTPBoxesPopup(Sender: TObject);
    procedure Jnghfdbnmgjxne1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowMail_IMAP_SMTPBoxesInTree;
    procedure LoadLetters;
    procedure AddToLog(s : shortstring);
    procedure HideLetterBody;
    procedure ShowInHtmlViewer(body : TStrings);
  end;


const
    LogFileName   = 'Actions.log';


var
  frm_Main: Tfrm_Main;

  Mail_IMAP_SMTPInfoLoaded  : boolean=false;

  { Головне сховище електроних листів }
  Mail_IMAP_SMTPBoxes       : TMail_IMAP_SMTPBoxList;


function  ExecuteFile(const FileName, Params, DefaultDir: string; ShowCmd:
Integer): THandle;
procedure FillPOPwithMail_IMAP_SMTPBoxSettings(mbSettings :
PMail_IMAP_SMTPBoxSettings);
```

```
implementation

uses shellapi, frmSettings, IdMessage,
     DataModule, frmNewLetter,
     ComObj, frmAskOnDuplicate, IdEMail_IMAP_SMTPAddress;

{$R *.dfm}

//параметри  облікових записів
procedure Tfrm_Main.aMail_IMAP_SMTPSettingsExecute(Sender: TObject);
begin
  frm_Mail_IMAP_SMTPSettings.Show;
end;

// вихід
procedure Tfrm_Main.aExitExecute(Sender: TObject);
begin
halt;
end;



procedure Tfrm_Main.FormShow(Sender: TObject);
{var
   IE: Variant;
begin
IE := CreateOleObject('InternetExplorer.Application');
        IE.Navigate('C:\MyHTML.html');
        While IE.Busy do begin end;
        Memo1.Text:= IE.Document.Body.innerText;
{end; }
begin
  if HideCaret(frm_Main.Handle) then Application.MessageBox('','');
  if not Mail_IMAP_SMTPInfoLoaded then
  begin
    { Show Mail_IMAP_SMTPboxes }

    ShowMail_IMAP_SMTPBoxesInTree;
    LoadLetters;
    Mail_IMAP_SMTPInfoLoaded:=true;

    Memo_LetterBody.Visible:=false;
    Web_LetterViewer.Visible:=false;
    if Settings.ShowInTray then
       begin
       Tray.IconVisible:=true;
       Tray.HideTaskbarIcon;
       frm_Main.Tray.MinimizeToTray:=true;
       end
       else
       begin
       Tray.IconVisible:=false;
       Tray.ShowMainForm;
       frm_Main.Tray.MinimizeToTray:=false;
       end;
  end;
  frm_AskOnDuplicate.ShowModal;
  if frm_AskOnDuplicate.ModalResult = mrIgnore then
Application.MessageBox('','');
end;

procedure Tfrm_Main.ShowMail_IMAP_SMTPBoxesInTree;
var
    i     : word;
    node  : TTreeNode;
begin
  Tree_Mail_IMAP_SMTPBoxes.Items.Clear;
  for i:=1 to TotalMail_IMAP_SMTPBoxes do
```

```
  begin

node:=Tree_Mail_IMAP_SMTPBoxes.Items.Add(nil,Mail_IMAP_SMTPBoxesSettings[i].Name
);
    node.ImageIndex:=0;
    Tree_Mail_IMAP_SMTPBoxes.Items.AddChild(node,'Вхідні').ImageIndex:=1;
    Tree_Mail_IMAP_SMTPBoxes.Items.AddChild(node,'Вихідні').ImageIndex:=2;
    Tree_Mail_IMAP_SMTPBoxes.Items.AddChild(node,'Відправлені').ImageIndex:=2;
    Tree_Mail_IMAP_SMTPBoxes.Items.AddChild(node,'Видалені').ImageIndex:=3;

    AddToLog('Завантажені параметри скриньки
<'+Mail_IMAP_SMTPBoxesSettings[i].Name+'>');
  end;
  AddToLog('Всього облікових записів : '+inttostr(TotalMail_IMAP_SMTPBoxes));
end;


procedure Tfrm_Main.LoadLetters;
var
    i : word;
Begin
  Mail_IMAP_SMTPBoxes.TotalBoxes:=0;
  for i:=1 to TotalMail_IMAP_SMTPBoxes do
  begin
    Mail_IMAP_SMTPBoxes.Add(Mail_IMAP_SMTPBoxesSettings[i].Name);
    AddToLog('Завантажені листи для скриньки
<'+Mail_IMAP_SMTPBoxesSettings[i].Name+'>');
  end;
End;

 // написати листа
procedure Tfrm_Main.aNewLetterExecute(Sender: TObject);
var
  Mail_IMAP_SMTPBoxName : shortstring;
begin
  frm_NewLetter.ClearForm;
  frm_NewLetter.Show;

  if Tree_Mail_IMAP_SMTPBoxes.Selected=nil then
    begin
      Application.MessageBox('Ви повинні виділити яку-небудь скриньку, щоб
відправити з неї листа','З якої поштової скриньки відправляти?');
      exit;
    end;

  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then
Mail_IMAP_SMTPBoxName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Text
     else
     Mail_IMAP_SMTPBoxName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  frm_NewLetter.Caption:='Новий лист  <'+Mail_IMAP_SMTPBoxName+'>';
end;


function ExecuteFile(const FileName, Params, DefaultDir: string; ShowCmd:
Integer): THandle;
var
zFileName, zParams, zDir: array[0..79] of Char;
begin
Result := ShellExecute(Application.MainForm.Handle, nil,
StrPCopy(zFileName, FileName), StrPCopy(zParams, Params),
StrPCopy(zDir, DefaultDir), ShowCmd);
end;


procedure Tfrm_Main.AddToLog(s  : shortstring);
var
    st    :   TFileStream;
    SLog  :   shortstring;
    Mess  :   shortstring;
```

```
    mem   :   TMemoryStream;
Begin
  if (s[1]=#10)and(s[2]=#13) then
     begin
     delete(s,1,2);
     Mess:=#10#13+DateTimeToStr(Now)+' -=> '+s+#10#13;
     end
     else Mess:=DateTimeToStr(Now)+' -=> '+s+#10#13;

  MemoLog.Lines.Add(Mess);

  if Settings.AllowLog then
  begin
    SLog:=ExtractFilePath(Application.ExeName)+LogFileName;
    try
      st:=TFileStream.Create(SLog,fmOpenReadWrite or fmShareDenyNone);
    except
      try
        st:=TFileStream.Create(SLog,fmCreate);
      except
        Application.MessageBox( pchar('Не можу відкрити файл "'+SLog+'" для
запису !'),'Помилка збереження лог файлу');
        System.exit;
      end;
    end;

    { Перевіряємо, якщо виходимо за границю MaxLogSize  }
    { та обрізаємо лог файл у цьому випадку }
    { в випадку вибору більшого MaxLogSize значення ми можемо уповільнити ти
програму}
    { так як кожний раз при записі лог файл буде копіюватися до пам'яті та
обратно }
    if (Settings.MaxLogSize<>0)and(St.Size > Settings.MaxLogSize * 1024) then
        try
          mem:=TMemoryStream.Create;
          mem.Size:=Settings.MaxLogSize*1024;
          mem.seek(0,soFromBeginning);

          st.Seek(-Settings.MaxLogSize*1024,soFromEnd);
          mem.CopyFrom(st,Settings.MaxLogSize*1024);
          st.Seek(0,soFromBeginning);
          mem.Seek(0,soFromBeginning);
          st.CopyFrom(mem,Settings.MaxLogSize*1024);
          st.Size:=Settings.MaxLogSize*1024;
        finally mem.Free; end;

    st.Seek(0,soFromEnd);
    st.Write(Mess[1],length(Mess));
    st.Free;
  end;
End;


procedure DeleteFiles(s : shortstring);
var
  sr  : TSearchRec;
Begin
  try
    findfirst(s, faAnyFile, sr);
    deletefile( IncludeTrailingPathDelimiter(ExtractFilePath(s))+sr.Name );
    while findnext(sr)=0 do deletefile(
IncludeTrailingPathDelimiter(ExtractFilePath(s))+sr.Name );
  except end;
End;


procedure Tfrm_Main.FormCreate(Sender: TObject);
begin
frm_Settings.LoadGlobalSettingsFromFile;
```

```
DeleteFiles(
IncludeTrailingPathDelimiter(ExtractFilePath(Application.ExeName))+'temp_*.html'
);

AddToLog(#10#13' *** Запуск Поштового клієнта'#10#13);
end;


 // параметри програми

procedure Tfrm_Main.aSettingsExecute(Sender: TObject);
begin
  frm_Settings.show;
end;



procedure Tfrm_Main.Tree_Mail_IMAP_SMTPBoxesClick(Sender: TObject);
var
    Mail_IMAP_SMTPBoxName     : shortstring;
    Mail_IMAP_SMTPBoxFolder   : shortstring;
    LetterList      : PLetters;
    i               : word;
begin
  if Tree_Mail_IMAP_SMTPBoxes.Selected = nil then
  begin
    HideLetterBody;
    exit;
  end;
  List_Attachmant.Clear;
  List_Attachmant.Visible:=false;
  Memo_LetterBody.Clear;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then
  begin
    if
Mail_IMAP_SMTPBoxes.FindBox(Tree_Mail_IMAP_SMTPBoxes.Selected.Text).IsBuisyNow
then aCheckMail_IMAP_SMTP.Enabled:=false
                                                                  else
aCheckMail_IMAP_SMTP.Enabled:=true;
    ListView_LettersHeaders.Clear;
    HideLetterBody;
    findMail_IMAP_SMTPBoxAndSelectIt(Tree_Mail_IMAP_SMTPBoxes.Selected.Text);
    exit;
  end;

  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);

  ListView_LettersHeaders.Items.Clear;
  if LetterList = nil then
  begin
    Panel_Content.Caption:='Немає такої папки.'+
                           ' Або вона була завантажена з помилкою. ';
    exit;
  end;

  if Mail_IMAP_SMTPBoxes.FindBox(Mail_IMAP_SMTPBoxName).IsBuisyNow then
aCheckMail_IMAP_SMTP.Enabled:=false
                                             else
aCheckMail_IMAP_SMTP.Enabled:=true;
  findMail_IMAP_SMTPBoxAndSelectIt(Mail_IMAP_SMTPBoxName);

  for i:=1 to LetterList^.Count do
  with ListView_LettersHeaders.Items.Add do
  begin
```

```
    Caption:=LetterList^.Letters[i]^.Subject;
    if LetterList^.Letters[i]^.Name='' then
SubItems.Add(LetterList^.Letters[i]^.From.Address)
                                       else
SubItems.Add(LetterList^.Letters[i]^.From.Name);
    SubItems.Add(DateTimeToStr(LetterList^.Letters[i]^.Date));
    //Розмір
    SubItems.Add( inttostr(LetterList^.LettersSizes[i]) );
    case LetterList^.Letters[i]^.Priority of
      mpHighest : SubItems.Add('Дуже високий');
      mpHigh    : SubItems.Add('Високий');
      mpNormal  : SubItems.Add('Звичайний');
      mpLow     : SubItems.Add('Низький');
      mpLowest  : SubItems.Add('Дуже низький');
      end;
  end;

end;


procedure FillPOPwithMail_IMAP_SMTPBoxSettings(mbSettings :
PMail_IMAP_SMTPBoxSettings);
Begin
  DM.POP.Host :=mbSettings.POPServer;
  DM.POP.Port :=mbSettings.POPPort;
  DM.POP.Username :=mbSettings.POPAccount;
  DM.POP.Password :=mbSettings.POPPass;
End;

(*
procedure RetrieveMail_IMAP_SMTP(let_count  : word);
var
    mes : TIdMessage;
    i   : word;
    LettersNot2beDownloaded : word;
Begin
  try
  LettersNot2beDownloaded:=0;
  for i:=1 to let_count do
    if CurrentMail_IMAP_SMTPBoxSettings^.RetrieveOnlyHeaders then
    // Отримати лише заголовки
    begin
    mes:=TIdMessage.Create(frm_Main);
    DM.POP.RetrieveHeader(i,mes);
    end
    else
    // Отримати весь лист
    begin
      mes:=TIdMessage.Create(frm_Main);
      DM.POP.RetrieveHeader(i,mes);
      if
Mail_IMAP_SMTPBoxes.GetByName(CurrentMail_IMAP_SMTPBoxSettings^.Name).HasLetterW
ithID(mes.MsgId) then
      begin
        inc(LettersNot2beDownloaded);
        continue;
      end;
      if
Mail_IMAP_SMTPBoxes.GetByName(CurrentMail_IMAP_SMTPBoxSettings^.Name).hadLetterW
ithId(mes.msgId) then
      begin
        case CurrentMail_IMAP_SMTPBoxSettings.ifDuplicate of
          Ignore  :  continue;
          Recieve :  ;
          Ask     :  begin
                     frm_AskOnDuplicate.FillInInfo(@mes);
                     frm_AskOnDuplicate.ShowModal;
                     case frm_AskOnDuplicate.ModalResult of
                       mrIgnore : continue;   // не отримувати
```

```
                        mrOk     : ;            // отримувати
                        mrAbort  : begin       // видаляти з сервера

                                   if not DM.POP.Delete(i) then
frm_Main.AddToLog('Помилка видалення листа як отриманого раніше, але видаленого
з комп''ютера(див. параметри) Від: "'+mes.From.Address+'" Тема:
"'+mes.Subject+'"');
                                   continue;
                                   end;
                        end;

                     end;
          DeleteFromServer  : begin
                              if not DM.POP.Delete(i) then
frm_Main.AddToLog('Помилка видалення листа як отриманого раніше, але видаленого
з комп''ютера(див. параметри) Від:  "'+mes.From.Address+'" Тема:
"'+mes.Subject+'"');
                              continue;
                              end;
          end;
      end;

      frm_Main.AddToLog('Отримані листи
('+inttostr(i)+'/'+inttostr(let_count)+') от '+mes.From.Text+' по Теме
:"'+mes.Subject+'"');
      DM.POP.Retrieve(i,mes);

Mail_IMAP_SMTPBoxes.GetByName(CurrentMail_IMAP_SMTPBoxSettings^.Name).Inbox.Add(
@mes,false, DM.POP.RetrieveMsgSize(i));
      if CurrentMail_IMAP_SMTPBoxSettings^.DeleteFromServer then
         if not DM.POP.Delete(i) then frm_Main.AddToLog('Помилка видалення листа
з сервера. Від: "'+mes.From.Address+'" Тема: "'+mes.Subject+'"');
    end;
  finally {mes.Free;} end;

  if LettersNot2beDownloaded<>0 then
     frm_Main.AddToLog('На сервері лежить(ать)' + inttostr
(LettersNot2beDownloaded) + 'листи, які не були завантажені помому що вони були
завантажені раніше і лежать в папках поточної скриньки');
End;
*)

procedure Tfrm_Main.aCheckMail_IMAP_SMTPExecute(Sender: TObject);
var
    CurMail_IMAP_SMTPBox    :   PMail_IMAP_SMTPBox;
begin
  // Перевірити пошту
  Tree_Mail_IMAP_SMTPBoxesClick(self);
  if CurrentMail_IMAP_SMTPBoxSettings =nil then exit;

  aCheckMail_IMAP_SMTP.Enabled:=false;

CurMail_IMAP_SMTPBox:=Mail_IMAP_SMTPBoxes.GetByName(CurrentMail_IMAP_SMTPBoxSett
ings^.Name);
  if CurMail_IMAP_SMTPBox.IsBuisyNow then exit;
  try
    // Імітація переривання таймера для ініціювання процесу перевірки пошти
    CurMail_IMAP_SMTPBox.onTimer_Check(self);
  except
    on e:exception do Application.MessageBox(pchar(E.Message),'Помилка при
запуску служби перевірки скриньки');
  end;

end;


procedure Tfrm_Main.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Mail_IMAP_SMTPBoxes.SaveMail_IMAP_SMTPBoxes;
```

```
end;


procedure Tfrm_Main.HideLetterBody;
begin
  Memo_LetterBody.Hide;
  Web_LetterViewer.Hide;
end;


procedure Tfrm_Main.List_AttachmantClick(Sender: TObject);
var
    Mail_IMAP_SMTPBoxName     : shortstring;
    Mail_IMAP_SMTPBoxFolder   : shortstring;
    LetterList      : PLetters;
begin
  if Tree_Mail_IMAP_SMTPBoxes.Selected = nil then exit;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then
  begin
    findMail_IMAP_SMTPBoxAndSelectIt(Tree_Mail_IMAP_SMTPBoxes.Selected.Text);
    exit;
  end;
  if (ListView_LettersHeaders.Selected=nil)or
     (List_Attachmant.Selected=nil)  then exit;


  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);

  if
LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[List
_Attachmant.ItemIndex].ContentType = 'text/plain' then
     // text/plain
     begin
     Web_LetterViewer.Hide;
     Memo_LetterBody.Clear;

Memo_LetterBody.Lines.AddStrings(TIdText(LetterList^.Letters[ListView_LettersHea
ders.ItemIndex+1]^.MessageParts.Items[List_Attachmant.ItemIndex]).Body);
     Memo_LetterBody.Show;
     end;
  if
LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[List
_Attachmant.ItemIndex].ContentType = 'text/html' then
     // text/html
     begin
     Memo_LetterBody.Hide;

ShowInHtmlViewer(TIdText(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1
]^.MessageParts.Items[List_Attachmant.ItemIndex]).Body);
     Web_LetterViewer.Show;
     end;
end;


// Вхід: рядок з кодом HTML; Вихід: в Web_LetterViewer
procedure Tfrm_Main.ShowInHtmlViewer(body: TStrings);
var
    tempFileName  : shortstring;
    st  :  TFileStream;
    i   :  integer;
    s   :  string;
begin
  if not CurrentMail_IMAP_SMTPBoxSettings^.OpenHTMLasTEXT then
  BEGIN
```

```
    try
       randomize;

tempFileName:=IncludeTrailingPathDelimiter(ExtractFilePath(Application.ExeName))
+'temp_'+inttostr(random(999))+'.html';
       deletefile(tempFileName);
       st:=TFileStream.Create(tempFileName,fmCreate);
    except
       on E:Exception do Application.MessageBox(pchar('Тимчасовий файл
"'+tempFileName+'" Не може бути створений :=> '+e.message),'Помилка створення
тимчасового файлу');
    end;

    st.Seek(0,soFromBeginning);
    for i:=0 to body.Count-1 do
    begin
    s:=body.Strings[i]+#10#13;
    st.Write(s[1], length(s) );
    end;
    st.Free;

    Web_LetterViewer.Navigate('file://'+tempFileName);
  END
  ELSE
  BEGIN
     Web_LetterViewer.Hide;
     Memo_LetterBody.Clear;
     Memo_LetterBody.Lines.AddStrings(Body);
     Memo_LetterBody.Show;
  END;
end;


procedure Tfrm_Main.Web_LetterViewerBeforeNavigate2(Sender: TObject;
  const pDisp: IDispatch; var URL, Flags, TargetFrameName, PostData,
  Headers: OleVariant; var Cancel: WordBool);
begin
  try
    if (pos('temp_',URL)=0)and(URL[2]<>':') then
    Cancel:=true;
  except end;
end;


procedure Tfrm_Main.ListView_LettersHeadersSelectItem(Sender: TObject;
  Item: TListItem; Selected: Boolean);
var
    Mail_IMAP_SMTPBoxName     : shortstring;
    Mail_IMAP_SMTPBoxFolder   : shortstring;
    LetterList      : PLetters;
    intIndex        : integer;
    BodyPart        : word;
begin
  if Tree_Mail_IMAP_SMTPBoxes.Selected = nil then exit;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then
  begin
    findMail_IMAP_SMTPBoxAndSelectIt(Tree_Mail_IMAP_SMTPBoxes.Selected.Text);
    exit;
  end;
  if ListView_LettersHeaders.Selected=nil then exit;

  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);

  //Список листів
```

```
  if pos('TEXT/HTML' ,
ANSIUPPERCASE(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.ContentT
ype))<>0 then
  begin
  Memo_LetterBody.Hide;

ShowInHtmlViewer(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Body)
;
  Web_LetterViewer.Show;
  end
  else
  begin

Memo_LetterBody.Lines:=LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^
.Body;
  Memo_LetterBody.visible:=true;
  end;

  BodyPart:=0;
  List_Attachmant.Items.Clear;
     for intIndex := 0 to
Pred(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Coun
t) do
      begin
         if
(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[in
tIndex] is TIdAttachment) then
            begin //загальні вкладення
               List_Attachmant.visible := true;
               with List_Attachmant.Items.Add do
               begin
                StateIndex:=0;
                Caption :=
TIdAttachment(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageP
arts.Items[intIndex]).Filename;

SubItems.Add(TIdAttachment(LetterList^.Letters[ListView_LettersHeaders.ItemIndex
+1]^.MessageParts.Items[intIndex]).ContentType);
               end;
            end
         else
            begin //текст листа
               if
LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[int
Index] is TIdText then
                  begin
                    List_Attachmant.visible := true;
                    inc(BodyPart);
                    with List_Attachmant.Items.Add do
                    begin
                      StateIndex:=1;
                      if
AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageP
arts.Items[intIndex].ContentType) = 'TEXT/PLAIN'
                          then Caption:='Частина #'+inttostr(BodyPart)
                          else
                          If
AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageP
arts.Items[intIndex].ContentType) = 'TEXT/HTML'
                             then Caption:='HTML #'+inttostr(Bodypart)
                             else
Caption:=LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.
Items[intIndex].ContentType;
                      if
LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Body.Count>0 then
                      if
(AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Body.St
```

```
rings[0])=AnsiUppercase('Це повідомлення складається з кількох частин у форматі
MIME') )
                          and

(AnsiUppercase(LetterList^.Letters[ListView_LettersHeaders.ItemIndex+1]^.Message
Parts.Items[intIndex].ContentType) = 'TEXT/HTML')
                          then begin
                               Selected:=true;
                               List_AttachmantClick(self);
                               end;
                    end;

                 end
            end;
      end;
      if List_Attachmant.Items.Count=0 then List_Attachmant.Visible:=false;
end;

procedure Tfrm_Main.List_AttachmantMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
    it  : TListItem;
begin
if Button = mbRight then
  begin
  it:=List_Attachmant.GetItemAt(x,y);
  if it=nil then exit;
  it.Selected:=true;
  end;
end;


const
  invalid :set of char=['\','/',':','?','*','"','<','>','|'];

function isValidFileName(s:string):boolean;
var
    i : word;
Begin
  isValidFileName:=false;
  for i:=1 to length(s) do
      if s[i] in invalid then exit;
  isValidFileName:=true;
End;


procedure Tfrm_Main.mnuSaveAttClick(Sender: TObject);
var
    Mail_IMAP_SMTPBoxName     : shortstring;
    Mail_IMAP_SMTPBoxFolder   : shortstring;
    LetterList      : PLetters;
Begin
  if Tree_Mail_IMAP_SMTPBoxes.Selected = nil then exit;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then
  begin
    findMail_IMAP_SMTPBoxAndSelectIt(Tree_Mail_IMAP_SMTPBoxes.Selected.Text);
    exit;
  end;
  if (ListView_LettersHeaders.Selected=nil)or
     (List_Attachmant.Selected=nil)  then exit;


  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);
```

```
  if
LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessageParts.Items[List
_Attachmant.ItemIndex] is TIdAttachment then
  begin
    SaveDialog_Att.FileName:=List_Attachmant.selected.Caption;
    if not isValidFileName(SaveDialog_Att.FileName) then
SaveDialog_Att.FileName:='Вкладення '+inttostr(random(99))+'.txt';
    if not SaveDialog_Att.Execute then exit;

TIdAttachment(LetterList.Letters[ListView_LettersHeaders.ItemIndex+1]^.MessagePa
rts.Items[List_Attachmant.ItemIndex]).SaveToFile(SaveDialog_Att.FileName);
  end else Application.MessageBox('Ця частина листа не є додатком!)','Це не
допустимо');

End;


procedure Tfrm_Main.Popup_AttachmentPopup(Sender: TObject);
begin
  if List_Attachmant.Selected = nil then mnuSaveAtt.Enabled:=false
                                    else mnuSaveAtt.Enabled:=true;
end;

procedure Tfrm_Main.mnu_DeleteLetterClick(Sender: TObject);
var
    Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTPBoxFolder : shortstring;
    LetterList                : PLetters;
begin
  if ( Tree_Mail_IMAP_SMTPBoxes.Selected=nil ) or
     ( Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil ) then exit;
  if ListView_LettersHeaders.Selected = nil then exit;

  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);

Mail_IMAP_SMTPBoxes.FindBox(Mail_IMAP_SMTPBoxName).MoveToGarbage(LetterList,List
View_LettersHeaders.ItemIndex+1);
  ListView_LettersHeaders.DeleteSelected;
end;

procedure Tfrm_Main.PopupMenuHeadersPopup(Sender: TObject);
var
    i : word;
begin
  if ListView_LettersHeaders.Selected=nil then
     for i:=0 to PopupMenuHeaders.Items.Count-1 do
        PopupMenuHeaders.Items[i].Enabled:=false
  else for i:=0 to PopupMenuHeaders.Items.Count-1 do
        PopupMenuHeaders.Items[i].Enabled:=true;
end;

procedure Tfrm_Main.aAnswerExecute(Sender: TObject);
var
    Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTPBoxFolder : shortstring;
    LetterList                : PLetters;
begin
  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);

  frm_NewLetter.ClearForm;
```

```
frm_NewLetter.Ed_Reciever.Text:=LetterList.Letters[ListView_LettersHeaders.itemi
ndex+1].From.Text;
  frm_NewLetter.Ed_Subject.Text:='Re:
'+LetterList.Letters[ListView_LettersHeaders.itemindex+1].Subject;
  frm_NewLetter.Caption:='Новий лист с <'+Mail_IMAP_SMTPBoxName+'> [відповідь]';
  frm_NewLetter.Show;
end;

procedure Tfrm_Main.aResendExecute(Sender: TObject);
var
    Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTPBoxFolder : shortstring;
    LetterList                : PLetters;
    i                         : integer;
begin
  Mail_IMAP_SMTPBoxName   :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBoxFolder :=  Tree_Mail_IMAP_SMTPBoxes.Selected.Text;


LetterList:=Mail_IMAP_SMTPBoxes.FindLetters(Mail_IMAP_SMTPBoxName,Mail_IMAP_SMTP
BoxFolder);

  frm_NewLetter.ClearForm;
  frm_NewLetter.Ed_Subject.Text:='Fw:
'+LetterList.Letters[ListView_LettersHeaders.itemindex+1].Subject;
  frm_NewLetter.Caption:='Новий лист с <'+Mail_IMAP_SMTPBoxName+'> [пересилка]';
  frm_NewLetter.Show;

  frm_NewLetter.Memo_Body.Lines.Add('<--- Пересилаємий лист (початок) --->');
  for i:=0 to
LetterList.Letters[ListView_LettersHeaders.itemindex+1].Body.Count-1 do
              frm_NewLetter.Memo_Body.Lines.Add('>
'+LetterList.Letters[ListView_LettersHeaders.itemindex+1]^.Body.Strings[i]);
  frm_NewLetter.Memo_Body.Lines.Add('<--- Пересилаємий лист (кінець)  --->');

end;


procedure Tfrm_Main.Tree_Mail_IMAP_SMTPBoxesDragDrop(Sender, Source: TObject; X,
Y: Integer);
var
    node  : TTreeNode;
    SourceMail_IMAP_SMTPBoxName : shortstring;
    SourceFolderName  : shortstring;
    SourceLetters     : PLetters;

    DestMail_IMAP_SMTPBox,
    DestFolderName    : shortstring;
    DestLetters       : PLetters;

begin
    node:=Tree_Mail_IMAP_SMTPBoxes.GetNodeAt(x,y);
    if (node = nil)or(node.Parent = nil) then exit;
    if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent= nil then
SourceMail_IMAP_SMTPBoxName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Text
                                          else
SourceMail_IMAP_SMTPBoxName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
    SourceFolderName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Text;

SourceLetters:=Mail_IMAP_SMTPBoxes.FindLetters(SourceMail_IMAP_SMTPBoxName,Sourc
eFolderName);

    DestFolderName:=node.Text;
    DestMail_IMAP_SMTPBox:=node.Parent.Text;

DestLetters:=Mail_IMAP_SMTPBoxes.FindLetters(DestMail_IMAP_SMTPBox,DestFolderNam
e);
    DestLetters.Add(SourceLetters.Letters[ListView_LettersHeaders.ItemIndex+1],
```

```
false,SourceLetters.LettersSizes[ListView_LettersHeaders.ItemIndex+1]);

SourceLetters.DeleteLetterWithoutFreeing(ListView_LettersHeaders.ItemIndex+1);
    Tree_Mail_IMAP_SMTPBoxesClick(self);
end;

procedure Tfrm_Main.Tree_Mail_IMAP_SMTPBoxesDragOver(Sender, Source: TObject; X,
  Y: Integer; State: TDragState; var Accept: Boolean);
begin
  if not (Source is TListView) then exit;
  if ListView_LettersHeaders.GetItemAt(x,y) = nil then exit;
  if (Tree_Mail_IMAP_SMTPBoxes.Selected =
nil)or(Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil) then exit;
  Accept:=true;
end;

procedure Tfrm_Main.TrayClick(Sender: TObject);
begin
 // Tray.ShowMainForm;
end;

procedure Tfrm_Main.TrayMinimizeToTray(Sender: TObject);
begin
  if Settings.ShowInTray then
  begin
 //  frm_Main.Tray.IconVisible:=true;
//   frm_Main.Tray.HideTaskbarIcon;
   end
   else
   begin
 ////  frm_Main.Tray.IconVisible:=false;
 //  frm_Main.Tray.ShowMainForm;
   end;
   //Application.Minimize;
end;

// Delete All Letters in the folder
procedure Tfrm_Main.N13Click(Sender: TObject);
var
    Letters : PLetters;
    i       : word;
begin
  if Tree_Mail_IMAP_SMTPBoxes.Selected=nil then exit;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then exit;

Letters:=Mail_IMAP_SMTPBoxes.FindLetters(Tree_Mail_IMAP_SMTPBoxes.Selected.Paren
t.Text, Tree_Mail_IMAP_SMTPBoxes.Selected.Text);
  if Letters = nil then exit;
  if MessageDlg('Ви впевнені, що хочете безповоротно видалити ВСІ листи в цій
папці?"',
              mtConfirmation,[mbYes, mbNo], 0)<> mrYes then exit;

  for i:=1 to Letters.Count do Letters.DeleteLetter(1);
  Tree_Mail_IMAP_SMTPBoxesClick(self);
end;

procedure Tfrm_Main.PopupMenu_Mail_IMAP_SMTPBoxesPopup(Sender: TObject);
begin
  PopupMenu_Mail_IMAP_SMTPBoxes.Items[0].Enabled:=false;
  PopupMenu_Mail_IMAP_SMTPBoxes.Items[1].Enabled:=false;
  PopupMenu_Mail_IMAP_SMTPBoxes.Items[2].Enabled:=false;
  if Tree_Mail_IMAP_SMTPBoxes.Selected=nil then exit;

  PopupMenu_Mail_IMAP_SMTPBoxes.Items[0].Enabled:=true;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent = nil then exit;
  PopupMenu_Mail_IMAP_SMTPBoxes.Items[1].Enabled:=true;
  PopupMenu_Mail_IMAP_SMTPBoxes.Items[2].Enabled:=true;
end;
```

```
// Відправити всі листи в поштовій скриньці
procedure Tfrm_Main.Jnghfdbnmgjxne1Click(Sender: TObject);
var
    BoxName : shortstring;
    Mail_IMAP_SMTPBox : PMail_IMAP_SMTPBox;
    Letters : PLetters;
    i       : word;
begin
  if Tree_Mail_IMAP_SMTPBoxes.Selected=nil then exit;
  if Tree_Mail_IMAP_SMTPBoxes.Selected.Parent=nil then
BoxName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Text
                                        else
BoxName:=Tree_Mail_IMAP_SMTPBoxes.Selected.Parent.Text;
  Mail_IMAP_SMTPBox:=Mail_IMAP_SMTPBoxes.FindBox(BoxName);
  if Mail_IMAP_SMTPBox=nil then exit;
  Letters:=@Mail_IMAP_SMTPBox.Outbox;
  if Letters=nil then exit;

  Tree_Mail_IMAP_SMTPBoxesClick(self);

  try
    try
    FillSMTPwithMail_IMAP_SMTPBoxSettings(CurrentMail_IMAP_SMTPBoxSettings);
    DM.SMTP.Connect(20000);
    frm_Main.AddToLog('З''єднання з сервером
'+DM.SMTP.host+':'+inttostr(DM.SMTP.port)+' успішне.');

    for i:=1 to Letters.Count do
    begin
      frm_Main.AddToLog('Відправка листа від
'+CurrentMail_IMAP_SMTPBoxSettings.Name+' к
'+Letters.Letters[1].Recipients.EMail_IMAP_SMTPAddresses+' за темою:
'+Letters.Letters[1].Subject);
      DM.SMTP.Send( Letters.Letters[1]^ ); // Відправка листа

Mail_IMAP_SMTPBox.Sent.Add(Letters.Letters[1],false,Letters.LettersSizes[1]);
// Додати до папки
      Mail_IMAP_SMTPBox.Outbox.DeleteLetterWithoutFreeing(1);
// Видалити з папки Вихідні
    end;

    finally
      DM.SMTP.Disconnect;
    end;
  except
    on E:Exception do
       begin
       Application.MessageBox(pchar('Помилка при відправці пошти зі скриньки
<'+BoxName+'> :'),pchar(E.Message) );
       AddToLog('Помилка при відправці пошти зі скриньки <'+BoxName+'>
:'+E.Message);
       end;
  end;

end;

// відкриття вікна "про програму..."
procedure Tfrm_Main.About1Click(Sender: TObject);
begin
Form1.Show;
end;

end.
```

Файл LetterStorage.pas – збереження листів

```
unit LetterStorage;

interface
uses idMessage, Classes, ExtCtrls { TTimer }, IdPOP3, frmMail_IMAP_SMTPSettings,
SyncObjs{CS};

const
    LetterDir       = 'Letters\';
    LetterBaseExt   = '.let';


type
{
TMyAttachments = object
      public
      Mail_IMAP_SMTPBoxName,
      FolderName  : shortstring;
      Count : word;
      constructor Init(_Mail_IMAP_SMTPboxname,_foldername : shortstring);
      constructor Load(st : TStream);
      destructor  Done;
      procedure   Save(st  : TStream);
      procedure  Add(fname  : shortstring);
      function   Get(index  : word):string;
      private
      Filenames : packed array [1..512] of ^shortstring;
      end;
}


PIdMessage = ^TIdMessage;
PLetters = ^TLetters;
TLetters = object
      private
      Mail_IMAP_SMTPBoxName,
      FolderName    : shortstring;

      TotalLetters  : word;
      public
      IsOnlyHeader  : {packed} array [1..1024*32] of boolean;
      LettersSizes  : {packed} array [1..1024*32] of longint;
      Letters       : array [1..1024*32] of PIdMessage;
      constructor Init(Mail_IMAP_SMTPBox, Folder : shortstring);
      procedure   Add( mes:PIdMessage; IsHeader: boolean; Size : longint);
      procedure   Load;
      procedure   Save;
      procedure   CreateEmptyLellerArchive(stName : shortstring);
      function    Count:word;
      function    HasLetterWithId( id : string ): boolean;
      procedure   DeleteLetter(index  : integer);
      procedure   DeleteLetterWithoutFreeing(index  : integer);
      procedure   PrepareLetter(Index : integer);
      end;

PBoolean       = ^boolean;
TGetMail_IMAP_SMTPThread = class(TThread)
                private
                    POP : TIdPOP3;
                    Mail_IMAP_SMTPBoxSettings : TMail_IMAP_SMTPBoxSettings;
                    LogString       : string;
                    BalloonMes      : string;
                    IsBuisy         : PBoolean;
                protected
                    procedure Execute; override;
                    procedure CallAddToLog(s:string);
```

```
                    procedure CallShowBalloon(s:string);
                    procedure _ShowBalloon;
                    procedure _Add2log;
                public
                    procedure Init( var mbSettings : TMail_IMAP_SMTPBoxSettings;
IsBuisyFlag : Pboolean );
                end;


PMail_IMAP_SMTPBox  = ^TMail_IMAP_SMTPBox;
TMail_IMAP_SMTPBox  = object
      private
      BoxName   :   shortstring;
      public
      Inbox,
      Outbox,
      Sent,
      Deleted   :   TLetters;
      { Для збереження з іншою частиною інформації }  { Не забувайте зберігати це
також }
      TotalLetters      : word;
      UsedSizeOnServer  : integer;
      DeletedIDs        : TStringList;
      LoadedFolder      : (lfNone,lfInbox,lfOutBox,lfSent,lf,Deleted,lfAll);

      CheckTimer        : TTimer;
      IsBuisyNow        : boolean;
      CheckThread       : TGetMail_IMAP_SMTPThread;

      procedure   onTimer_Check(Sender: TObject);
      procedure   SetTimerInterval( i : word );

      constructor Init(name : shortstring);
      function    GetName   : shortstring;
      function    HasLetterWithID( id : string): boolean;
      function    HadLetterWithID( id : string): boolean;
      procedure   Save;
      function    MoveToGarbage( folder : PLetters; index  : integer ):boolean;
      end;



TMail_IMAP_SMTPBoxList =object
      private
      Mail_IMAP_SMTPBoxes :   array [1..100] of PMail_IMAP_SMTPBox;
      public
      TotalBoxes:   word;
      function  Add(name  : shortstring)       : PMail_IMAP_SMTPBox;
      function  GetByName (name : shortstring) : PMail_IMAP_SMTPBox;
      function  GetByIndex(index: word)        : PMail_IMAP_SMTPBox;
      function  FindLetters( Mail_IMAP_SMTPbox, Mail_IMAP_SMTPfolder :
shortstring ):PLetters;
      function  FindBox( boxname : shortstring): PMail_IMAP_SMTPBox;
      procedure SaveMail_IMAP_SMTPBoxes;
      end;


var
    LettersPath     :     string;
    CS              :     TCriticalSection;


implementation

uses SysUtils,frmMain, frmAskOnDuplicate,Forms, dzlib, Controls {4modal
results},
     Dialogs {for MessageDlg}, CoolTrayIcon;

{ TLetter }
```

```
constructor TLetters.Init(Mail_IMAP_SMTPBox, Folder : shortstring);
Begin
  Mail_IMAP_SMTPBoxName :=  Mail_IMAP_SMTPBox;
  FolderName  :=  Folder;
End;


{ *** Структура файлу TLetters ***}

{ 4 байти       - розмір наступного листа  }
//(*  { 256(5) байт  - ідентифікатор} *)
{   ...  ...   ... }
{   ..   Лист  ..}
{   ...  ...   ... }



procedure   TLetters.Load;
var
    SourceFile    : shortstring;
    stCompressed  : TFileStream;
    zstCompressed : TDecompressionStream;
    stM           : TMemoryStream;

    TotalLetters2Load : word;
    sizeOfALetter     : word;
Begin
  findMail_IMAP_SMTPBoxAndSelectIt(Mail_IMAP_SMTPBoxName);
  if CurrentMail_IMAP_SMTPBoxSettings.DynamicFolders then exit;

  SourceFile:=LettersPath+Mail_IMAP_SMTPBoxName+' '+FolderName+LetterBaseExt;

  try
    stCompressed:=TFileStream.Create(SourceFile,fmOpenRead);
  except
    CreateEmptyLellerArchive(SourceFile);
    frm_Main.AddToLog('Не удалось найти архивные файлы ящика
<'+Mail_IMAP_SMTPBoxName+'>. Они будут созданы заново.');
    Application.MessageBox(pchar('Архів листів "'+SourceFile+'" не може бути
відкритий.'),'Помилка завантаження листів.');
    exit;
  end;

  try
    zstCompressed:=TDecompressionStream.Create(stCompressed);
  except
  on E:Exception do
    begin
    stCompressed.Free;
    Application.MessageBox(pchar('Архів листів "'+SourceFile+'" не може бути
відкритий і розархівований! Exception ='+E.Message),'Помилка завантаження за
архівованих листів.');
    exit;
    end;
  end;

  // Тут починається читання архівних даних
try
  try
      zstCompressed.Seek(0,soFromBeginning);
      zstCompressed.Read(TotalLetters2Load,sizeof(TotalLetters2Load));

      zstCompressed.Read(IsOnlyHeader,sizeof(IsOnlyHeader));
      zstCompressed.Read(LettersSizes,sizeof(LettersSizes));
      TotalLetters:=0;
      while (TotalLetters < TotalLetters2Load) do
```

```
      try
        zstCompressed.Read(sizeOfALetter,sizeof(sizeOfALetter));
        stM:=TMemoryStream.Create;
        stM.Seek(0,soFromBeginning);
        stM.CopyFrom(zstCompressed,sizeOfALetter);
        stM.Seek(0,soFromBeginning);
        inc(TotalLetters);
        new(Letters[TotalLetters]);
        Letters[TotalLetters]^:=TIdMessage.Create(nil);
        Letters[TotalLetters]^.LoadFromStream(stM,false);
      finally
        stM.Free;
      end;
  finally
    zstCompressed.Free;
    stCompressed.Free;
  end;
except
  Application.MessageBox(pchar('Виникла помилка при завантаженні листів з файлу
"' + SourceFile +'".'+
                          # 10 # 13 'Можливо він пошкоджений. Спробуйте видалити
його .'),' Помилка завантаження листів. ')
мети;

End;

function TLetters.Count: word;
begin
  Count:=TotalLetters;
end;

procedure TLetters.Save;
var
    i       : word;
    st      : TFileStream;
    stName  : shortstring;
    zst     : TCompressionStream;
    stM     : TMemoryStream;
    LetterSize : word;
begin
  try
    stName:=LettersPath+Mail_IMAP_SMTPBoxName+' '+FolderName+LetterBaseExt;
    st:=TFileStream.Create(stName,fmOpenWrite);
  except
    try
      st:=TFileStream.Create(stName,fmCreate);
    except
      Application.MessageBox(pchar('Архів листів "'+stName+'" не може бути
створений.'),'Помилка створення листів.');
      exit;
    end;
  end;
  st.Size:=0;

  try
    zst:=TCompressionStream.Create(clMax,st);
  except
    on E:Exception do
    begin
    st.Free;
    Application.MessageBox(pchar('Архів листів "'+stName+'" не може бути
стиснутий. Exception='+E.Message),'Помилка завантаження листів.');
    exit;
    end;
  end;

  { Написати загальну кількість листів }
  zst.Write(TotalLetters,sizeof(TotalLetters));
  zst.Write(IsOnlyHeader,sizeof(IsOnlyHeader));
```

```
  zst.Write(LettersSizes,sizeof(LettersSizes));

  for i:=1 to TotalLetters do
  try
    stM:=TMemoryStream.Create;
    Letters[i].SaveToStream(stM,false);

    stM.Seek(0,soFromBeginning);
    LetterSize:=stM.size;
    zst.Write(LetterSize,sizeof(LetterSize));

    stM.Seek(0,soFromBeginning);
    zst.CopyFrom(stM,stM.Size);
  finally
    stM.Free;
  end;

  zst.Free;
  st.Free;

end;

procedure TLetters.CreateEmptyLellerArchive(stName: shortstring);
var
    st  : TFileStream;
    zst : TCompressionStream;
begin
{  try
    st:=TFileStream.Create(stName,fmOpenWrite);
  except
    try
      st:=TFileStream.Create(stName,fmCreate);
    except
      Application.MessageBox(pchar('Архів листів "'+stName+'" не може бути
створений.'),'Помилка збереження листів.');
      exit;
    end;
  end;

  try
    zst:=TCompressionStream.Create(clMax,st);
  except
    on E:Exception do
    begin
    st.Free;
    Application.MessageBox(pchar('Архів листів "'+stName+'" не може бути
стиснутий. Exception='+E.Message),'Помилка завантаження листів.');
    exit;
    end;
  end;

  // Загальна кількість листів
  TotalLetters:=0;
  fillchar(IsOnlyHeader,sizeof(IsOnlyHeader),0);
  zst.Write(TotalLetters,sizeof(TotalLetters));
  zst.Write(IsOnlyHeader,sizeof(IsOnlyHeader));

  zst.Free;
  st.Free;  }
end;

procedure TLetters.Add(mes: PIdMessage; IsHeader: boolean; Size:  longint);
begin
  inc(TotalLetters);
  new(Letters[TotalLetters]);
  //mes.MessageParts.
  Letters[TotalLetters]^:=TIdMessage.Create(nil);
  Letters[TotalLetters]^:=mes^;
  IsOnlyHeader[TotalLetters]:=IsHeader;
```

```
  LettersSizes[TotalLetters]:=Size;
end;

function TLetters.HasLetterWithId(id: string): boolean;
var
    i : word;
begin
  HasLetterWithId:=true;
  for i:=1 to TotalLetters do
    if Letters[i]^.MsgId = id then exit;
  HasLetterWithId:=false;
end;

procedure TLetters.DeleteLetter(index: integer);
var
    i : word;
    temp  : PidMessage;
    mesID : shortstring;
    Mail_IMAP_SMTPbx: PMail_IMAP_SMTPBox;
    IdIndex : integer;
begin
  temp:=Letters[index];
  mesID:=temp^.MsgId;
  for i:=index to TotalLetters-1 do
      begin
      Letters[i]:=Letters[i+1];
      LettersSizes[i]:=LettersSizes[i+1];
      IsOnlyHeader[i]:=IsOnlyHeader[i+1];
      end;
  dec(TotalLetters);
  temp.Free;
  dispose(temp);

  Mail_IMAP_SMTPbx:=Mail_IMAP_SMTPBoxes.FindBox(Mail_IMAP_SMTPBoxName);
  if Mail_IMAP_SMTPbx = nil then exit;
  IdIndex:=Mail_IMAP_SMTPbx.DeletedIDs.IndexOf(mesID);
  if IdIndex<>-1 then exit
                 else Mail_IMAP_SMTPbx.DeletedIDs.Add(mesID);
end;

procedure TLetters.DeleteLetterWithoutFreeing(index: integer);
var
    i : word;
    temp  : PidMessage;
    mesID : shortstring;
    Mail_IMAP_SMTPbx: PMail_IMAP_SMTPBox;
    IdIndex : integer;
begin
  temp:=Letters[index];
  mesID:=temp^.MsgId;
  for i:=index to TotalLetters-1 do
      begin
      Letters[i]:=Letters[i+1];
      LettersSizes[i]:=LettersSizes[i+1];
      IsOnlyHeader[i]:=IsOnlyHeader[i+1];
      end;
  dec(TotalLetters);
  temp.Free;
  dispose(temp);

  Mail_IMAP_SMTPbx:=Mail_IMAP_SMTPBoxes.FindBox(Mail_IMAP_SMTPBoxName);
  if Mail_IMAP_SMTPbx = nil then exit;
  IdIndex:=Mail_IMAP_SMTPbx.DeletedIDs.IndexOf(mesID);
  if IdIndex<>-1 then exit
                 else Mail_IMAP_SMTPbx.DeletedIDs.Add(mesID);
end;

{ Викликається, коли прийшов час перевірити цю поштову скриньку }
procedure   TMail_IMAP_SMTPBox.onTimer_Check(Sender: TObject);
```

```
Begin
  if IsBuisyNow then exit;
  IsBuisyNow:=true;
  frm_Main.aCheckMail_IMAP_SMTP.Enabled:=false;
  CheckThread.Resume;
End;

constructor TMail_IMAP_SMTPBox.Init(name  : shortstring);
Begin
  BoxName:=name;
  IsBuisyNow:=false;
  findMail_IMAP_SMTPBoxAndSelectIt(name);
  CheckThread:=TGetMail_IMAP_SMTPThread.Create(true);
  CheckThread.Init(CurrentMail_IMAP_SMTPBOxSettings^, @IsBuisyNow);

  if CurrentMail_IMAP_SMTPBoxSettings.DynamicFolders then LoadedFolder:=lfNone
                                          else LoadedFolder:=lfAll;

  CheckTimer:=TTimer.Create(frm_Main);
  CheckTimer.OnTimer:=onTimer_Check;
  if CurrentMail_IMAP_SMTPBoxSettings.CheckEvery<>0 then
  begin
  CheckTimer.Interval:=CurrentMail_IMAP_SMTPBoxSettings.CheckEvery*6000;
  CheckTimer.Enabled:=true;
  end
  else CheckTimer.Enabled:=false;

  Inbox.Init(name,'Вхідні');
  Outbox.Init(name,'Вихідні');
  Sent.Init(name,'Відправлені');
  Deleted.Init(name,'Видалені');

  Inbox.Load;
  Outbox.Load;
  Sent.Load;
  Deleted.Load;

  if not FileExists(LettersPath+BoxName+' DeletedIDs.dat') then
FileClose(FileCreate(LettersPath+BoxName+' DeletedIDs.dat'));
  DeletedIDs:=TStringList.Create;        { Ми повинні визволити це у кінці }
  DeletedIDs.LoadFromFile(LettersPath+BoxName+' DeletedIDs.dat');

End;

function    TMail_IMAP_SMTPBox.GetName   : shortstring;
Begin
  GetName:=BoxName;
End;

procedure TMail_IMAP_SMTPBox.Save;
begin
  Inbox.Save;
  OutBox.Save;
  Sent.Save;
  Deleted.Save;

  deletefile(LettersPath+BoxName+' DeletedIDs.dat');
  DeletedIDs.SaveToFile(LettersPath+BoxName+' DeletedIDs.dat');
end;

function TMail_IMAP_SMTPBox.HasLetterWithID(id: string): boolean;
begin
  if (Inbox.HasLetterWithId(id) or Outbox.HasLetterWithId(id) or
      Sent.HasLetterWithId(id)  or Deleted.HasLetterWithId(id) ) then
HasLetterWithID:=true
                                                              else
HasLetterWithID:=false;
end;
```

```
function TMail_IMAP_SMTPBox.HadLetterWithID(id: string): boolean;
begin
  HadLetterWithID:= DeletedIDs.IndexOf(id)<>-1 ;
end;

procedure TMail_IMAP_SMTPBox.SetTimerInterval(i: word);
begin
  CheckTimer.Interval:=i*60000;
  if i=0 then CheckTimer.Enabled:=false
         else CheckTimer.Enabled:=true;
end;

function TMail_IMAP_SMTPBox.MoveToGarbage(folder: PLetters; index:
integer):boolean;
begin
  if folder.FolderName<>'Удаленные' then
  begin
  Deleted.Add( Folder.Letters[index], false, Folder.LettersSizes[index]);
  folder.DeleteLetterWithoutFreeing(index);
  MoveToGarbage:=true;
  end
  else // Якщо ви хочете видалити лист зі сміттєвого кошику
  if MessageDlg('Ви впевнені(!), що хочете безповоротно видалити цей
лист?',mtConfirmation,[mbYes, mbNo], 0) <> mrNo then
     begin
     folder.DeleteLetter(index);
     MoveToGarbage:=true;
     end
     else MoveToGarbage:=false;
end;

function TMail_IMAP_SMTPBoxList.Add(name  : shortstring):PMail_IMAP_SMTPBox;
Begin
  inc(TotalBoxes);
  new(Mail_IMAP_SMTPBoxes[TotalBoxes]);
  Mail_IMAP_SMTPBoxes[TotalBoxes]^.init(name);
  Add:=Mail_IMAP_SMTPBoxes[TotalBoxes];
End;

function  TMail_IMAP_SMTPBoxList.GetByName (name :
shortstring):PMail_IMAP_SMTPBox;
var
    i : word;
Begin
  GetByName:=nil;
  for i:=1 to TotalBoxes do
      if AnsiUpperCase( Mail_IMAP_SMTPBoxes[i]^.BoxName ) = AnsiUpperCase(name)
then
      begin
      GetByName:=Mail_IMAP_SMTPBoxes[i];
      exit;
      end;
End;

function  TMail_IMAP_SMTPBoxList.FindLetters( Mail_IMAP_SMTPbox,
Mail_IMAP_SMTPfolder : shortstring ):PLetters;
var
    i : word;
Begin
  FindLetters:=nil;
  for i:=1 to TotalBoxes do
      if Mail_IMAP_SMTPBoxes[i].BoxName = Mail_IMAP_SMTPbox then
      begin
        if Mail_IMAP_SMTPfolder = 'Вхідні' then
        begin
        FindLetters:=@Mail_IMAP_SMTPBoxes[i].Inbox;
        exit;
        end;
        if Mail_IMAP_SMTPfolder = 'Вихідні' then
```

```
        begin
        FindLetters:=@Mail_IMAP_SMTPBoxes[i].OutBox;
        exit;
        end;
        if Mail_IMAP_SMTPfolder = 'Відправлені' then
        begin
        FindLetters:=@Mail_IMAP_SMTPBoxes[i].Sent;
        exit;
        end;
        if Mail_IMAP_SMTPfolder = 'Видалені' then
        begin
        FindLetters:=@Mail_IMAP_SMTPBoxes[i].Deleted;
        exit;
        end;

      end;

End;

procedure TMail_IMAP_SMTPBoxList.SaveMail_IMAP_SMTPBoxes;
var
    i : word;
Begin
  for i:=1 to TotalBoxes do Mail_IMAP_SMTPBoxes[i].Save;
End;

function TMail_IMAP_SMTPBoxList.FindBox(boxname :
shortstring):PMail_IMAP_SMTPBox;
var
    i : word;
Begin
  FindBox:=nil;
  for i:=1 to TotalBoxes do
    if Mail_IMAP_SMTPBoxes[i].BoxName = boxname then
    begin
    FindBox:=Mail_IMAP_SMTPBoxes[i];
    break;
    end;
End;

function TMail_IMAP_SMTPBoxList.GetByIndex(index : word):PMail_IMAP_SMTPBox;
Begin
  if (index>=1)and(index<=100) then GetByIndex:=Mail_IMAP_SMTPBoxes[index]
                               else GetByIndex:=nil;
End;

{ TMyAttachments }

constructor TMyAttachments.Init(_Mail_IMAP_SMTPboxname,_foldername :
shortstring);
Begin
  Count:=0;
  Mail_IMAP_SMTPBoxName:=_Mail_IMAP_SMTPBoxName;
  FolderName :=_foldername;

  fillchar(Filenames,sizeof(Filenames),0);
End;

constructor TMyAttachments.Load(st : TStream);
var
    i : word;
    s : shortstring;
Begin
  Count:=0;
  fillchar(FileNames,sizeof(FileNames),0);
  st.Read(Count,sizeof(Count));
  for i:=1 to Count do
    begin
    st.Read(s,sizeof(s));
```

```
    Add(s);
    end;
End;

procedure TMyAttachments.Save(st  : TStream);
var
    i : word;
Begin
  st.write(Count,sizeof(Count));
  for i:=1 to Count do st.write(FileNames[i]^,sizeof(FileNames[i]^));
End;

destructor  TMyAttachments.Done;
var
    i : word;
Begin
  for i:=1 to Count do dispose(FileNames[i]);
End;

procedure  TMyAttachments.Add(fname  : shortstring);
Begin
  inc(Count);
  new(FileNames[Count]);
  FileNames[Count]^:=fname;
End;

function TMyAttachments.Get(index: word): string;
begin
  if index<=Count then Get:=FileNames[index]^;
end;

{ TGetMail_IMAP_SMTPThread }

procedure TGetMail_IMAP_SMTPThread.Init(var mbSettings :
TMail_IMAP_SMTPBoxSettings; IsBuisyFlag : PBoolean );
begin
  POP:=TIdPOP3.Create(frm_Main);
  POP.Host :=mbSettings.POPServer;
  POP.Port :=mbSettings.POPPort;
  POP.Username :=mbSettings.POPAccount;
  POP.Password :=mbSettings.POPPass;

  Mail_IMAP_SMTPBoxSettings:=mbSettings;
  IsBuisy:=IsBuisyFlag;
end;

procedure TGetMail_IMAP_SMTPThread.CallAddToLog(s:string);
Begin
  LogString:=s;
  Synchronize(_Add2log);
End;

procedure TGetMail_IMAP_SMTPThread._Add2log;
Begin
  frm_Main.AddToLog(LogString);
End;

procedure TGetMail_IMAP_SMTPThread.CallShowBalloon(s:string);
Begin
  BalloonMes:=s;
  Synchronize(_ShowBalloon);
End;

procedure TGetMail_IMAP_SMTPThread._ShowBalloon;
Begin
  if BalloonMes[1]='B' then frm_Main.Tray.ShowBalloonHint(
'MyMail_IMAP_SMTP',BalloonMes,bitInfo,30)
                       else frm_Main.Tray.ShowBalloonHint(
'MyMail_IMAP_SMTP',BalloonMes,bitError,30)
```

```
End;

procedure TGetMail_IMAP_SMTPThread.Execute;
var
    CurMail_IMAP_SMTPBox      :   PMail_IMAP_SMTPBox;
    LettersOnServer, i,
    LettersNot2beDownloaded :   word;
    mes             :   TIdMessage;
begin
  inherited;
  while not Terminated do
  BEGIN
  // Перевірка пошти
  if POP.Connected then Pop.Disconnect;

  try
    POP.Connect(20000);              //  20 секунд
  except
    on E:Exception do
    begin
      CallAddToLog('Помилка підключення до сервера <'+pop.host+'
:'+inttostr(POP.port)+'>  Exception='+E.message);
      if Mail_IMAP_SMTPBoxSettings.ShowBalloonOnNewMes then
         CallShowBalloon('Помилка підключення до сервера:'#10#13'<'+pop.host+'
:'+inttostr(POP.port)+'>');
      IsBuisy^:=false;
      exit;
    end;
  end;  // of try


CurMail_IMAP_SMTPBox:=Mail_IMAP_SMTPBoxes.GetByName(Mail_IMAP_SMTPBoxSettings.Na
me);
  LettersOnServer               :=  POP.CheckMessages;
  CurMail_IMAP_SMTPBox^.UsedSizeOnServer  :=  POP.RetrieveMail_IMAP_SMTPBoxSize;

  CallAddToLog('Підключення до'+POP.Host+' для
<'+Mail_IMAP_SMTPBoxSettings.Name+'> успішне. На сервері
'+inttostr(LettersOnServer)+' (нових або непрочитаних) листів
'+inttostr(CurMail_IMAP_SMTPBox^.UsedSizeOnServer)+' байт');

  try
  {   Частина коду, яка відповідає за завантаження кожного повідомлення }

  LettersNot2beDownloaded:=0;
  for i:=1 to LettersOnServer do
    if Mail_IMAP_SMTPBoxSettings.RetrieveOnlyHeaders then
    // Отримати лише заголовки
    begin          {  витягнути тільки заголовки}
    mes:=TIdMessage.Create(frm_Main);
    POP.RetrieveHeader(i,mes);
    end
    else
    // Отримати всі листи
    begin
      mes:=TIdMessage.Create(frm_Main);
      POP.RetrieveHeader(i,mes);
      if
{Mail_IMAP_SMTPBoxes.GetByName(Mail_IMAP_SMTPBoxSettings.Name)}CurMail_IMAP_SMTP
Box^.HasLetterWithID(mes.MsgId) then
      begin
        inc(LettersNot2beDownloaded);
        continue;
      end;
      if
{Mail_IMAP_SMTPBoxes.GetByName(Mail_IMAP_SMTPBoxSettings.Name)}CurMail_IMAP_SMTP
Box^.hadLetterWithId(mes.msgId) then
      begin
        CS.Enter;
```

```
        case Mail_IMAP_SMTPBoxSettings.ifDuplicate of
          Ignore  :  begin
                     continue;
                     end;
          Recieve :  ;
          Ask     :  begin
                     frm_AskOnDuplicate.FillInInfo(@mes);
                     while IsShown do ;
                     IsShown:=true;
                     frm_AskOnDuplicate.ShowModal;
                     IsShown:=false;
                     case frm_AskOnDuplicate.ModalResult of
                       mrIgnore : continue;
                       mrOk     : ;
                       mrAbort  : begin       server
                                  if not POP.Delete(i) then CallAddToLog(Помилка
видалення листа як отриманого раніше, але видаленого з EOM(див. налаштування)
От: "'+mes.From.Address+'" Тема: "'+mes.Subject+'"');
                                  CS.Leave;
                                  continue;
                                  end;
                       end;

                     end;
          DeleteFromServer  : begin
                              if not POP.Delete(i) then CallAddToLog(Помилка
видалення листа як отриманого раніше, але видаленого з EOM(див. налаштування)
От: "'+mes.From.Address+'" Тема: "'+mes.Subject+'"');
                              CS.Leave;
                              continue;
                              end;
          end;
          CS.Leave;
      end;

      CallAddToLog('Отримання листа
('+inttostr(i)+'/'+inttostr(LettersOnServer)+') від '+mes.From.Text+' по Теме
:"'+mes.Subject+'"');
      if Mail_IMAP_SMTPBoxSettings.ShowBalloonOnNewMes then
         CallShowBalloon('Вхідний лист.'#10#13+'Від:
'+mes.From.Text+#10#13+'Тема: '+mes.Subject);
      POP.Retrieve(i,mes);

{Mail_IMAP_SMTPBoxes.GetByName(Mail_IMAP_SMTPBoxSettings.Name)}CurMail_IMAP_SMTP
Box^.Inbox.Add(@mes,false, POP.RetrieveMsgSize(i));
      if Mail_IMAP_SMTPBoxSettings.DeleteFromServer then
         if not POP.Delete(i) then frm_Main.AddToLog('Помилка видалення листа з
сервера. Від: "'+mes.From.Address+'" Тема: "'+mes.Subject+'"');
    end;

  if LettersNot2beDownloaded<>0 then
     CallAddToLog('На сервері лежить(ать) '+inttostr(LettersNot2beDownloaded)+'
листи, які не були завантажені, тому що вони вже були завантажені раніше і
лежать у папках поточної скриньки');

    finally
    IsBuisy^:=false;
    CS.Enter;
      frm_Main.Tree_Mail_IMAP_SMTPBoxesClick(self);
    CS.Leave;
    if POP.Connected then Pop.Disconnect;
    CallAddToLog('Отключение от '+POP.Host+'  -   OK.');

    self.Suspend;
  end;
  END;
end;

begin
```

```
  CS:=TCriticalSection.Create;

LettersPath:=IncludeTrailingPathDelimiter(ExtractFilePath(Application.exename));
  LettersPath:=LettersPath+LetterDir;
  if not DirectoryExists(LettersPath) then ForceDirectories(LettersPath);

end.

finalization
  CS.Free;
end.
```

Файл frmNewLetter.pas – створення та відправка нового листа

```
    unit frmNewLetter;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, ToolWin, ComCtrls, Menus, HTTPApp, HTTPProd, OleCtrls,
  SHDocVw, StdCtrls, ExtCtrls, ImgList, ActnList,
  IdMessage,
  frmMail_IMAP_SMTPSettings,LetterStorage;

type
  PidMessage =  ^TidMessage;

  Tfrm_NewLetter = class(TForm)
    MainMenu: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    StatusBar: TStatusBar;
    CoolBar: TCoolBar;
    ToolBar1: TToolBar;
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Ed_Reciever: TEdit;
    Ed_Copy: TEdit;
    Ed_Subject: TEdit;
    Memo_Body: TMemo;
    Label4: TLabel;
    Ed_HiddenCopy: TEdit;
    Actions_NewLetter: TActionList;
    SendNewLetter: TAction;
    Images_SendLetter: TImageList;
    ToolButton1: TToolButton;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    MoveNewLetter2Outbox: TAction;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    Splitter1: TSplitter;
    List_NewAttachments: TListView;
    ToolButton4: TToolButton;
    NewAttachment: TAction;
    Dialog_OpenAttachment: TOpenDialog;
    PopupM_Attachment: TPopupMenu;
    N6: TMenuItem;
    Ghbrhtgbnmafqk1: TMenuItem;
    Panel_AdditionlSettings: TPanel;
    Group_AdditionalSettings: TGroupBox;
    Label6: TLabel;
    Label7: TLabel;
    Label5: TLabel;
    Combo_Priority: TComboBox;
    Ed_Sender: TEdit;
    Memo_AdditionalHeaders: TMemo;
    ToolButton5: TToolButton;
    Btn_AdvancedPage: TBitBtn;
    Splitter2: TSplitter;
    N7: TMenuItem;
    N8: TMenuItem;
    Label8: TLabel;
    Ed_ReplyTo: TEdit;
    Label9: TLabel;
    Combo_ContexType: TComboBox;
```

```
    procedure SendNewLetterExecute(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure MoveNewLetter2OutboxExecute(Sender: TObject);
    procedure NewAttachmentExecute(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure List_NewAttachmentsMouseDown(Sender: TObject;
      Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
    procedure Btn_AdvancedPageClick(Sender: TObject);
    procedure N8Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    function CheckIfFormIsFilled : boolean;
  public
    { Public declarations }
    IsToBeResend  : boolean;
    ResendMes     : TidMessage;
    Mail_IMAP_SMTPBoxName   : shortstring;
    CURR_Mail_IMAP_SMTPbox_SETTINGS : TMail_IMAP_SMTPBoxSettings;
    CURR_MAIL_IMAP_SMTP_BOX : PMail_IMAP_SMTPBox;
    function CreateLetterFromForm:PidMessage;
    procedure UpdateStatusLine;
    procedure ClearForm;
  end;

procedure FillSMTPwithMail_IMAP_SMTPBoxSettings(mbSettings :
PMail_IMAP_SMTPBoxSettings);

var
  frm_NewLetter: Tfrm_NewLetter;

implementation
uses frmMain, DataModule;
{$R *.dfm}

procedure FillSMTPwithMail_IMAP_SMTPBoxSettings(mbSettings :
PMail_IMAP_SMTPBoxSettings);
Begin
  DM.SMTP.Host :=mbSettings.SMTPServer;
  DM.SMTP.Port :=mbSettings.SMTPPort;
  DM.SMTP.Username :=mbSettings.SMTPAccount;
  DM.SMTP.Password :=mbSettings.SMTPPass;
  DM.SMTP.AuthenticationType:=mbSettings.AuthType;
End;

 // відправити лист
procedure Tfrm_NewLetter.SendNewLetterExecute(Sender: TObject);
var
    mes     : PidMessage;
begin
  if CurrentMail_IMAP_SMTPBoxSettings = nil then
  begin
    Application.MessageBox('Ви повинні вибрати скриньку з якої хочете ','Помилка
відправлення');
    exit;
  end;
  if not CheckIfFormIsFilled then exit;
  mes:=CreateLetterFromForm;
  SendNewLetter.Enabled:=false;
  try
      FillSMTPwithMail_IMAP_SMTPBoxSettings(CurrentMail_IMAP_SMTPBoxSettings);
      frm_Main.AddToLog('Негайна відправка листа від
'+CurrentMail_IMAP_SMTPBoxSettings.Name+' до
'+mes.Recipients.EMail_IMAP_SMTPAddresses);
      DM.SMTP.Connect(20000);
      frm_Main.AddToLog('З''єднання з сервером
'+DM.SMTP.host+':'+inttostr(DM.SMTP.port)+' успішне.');
      try
```

```
        DM.SMTP.Send(mes^);
      finally
        DM.SMTP.Disconnect;
      end;
      frm_Main.AddToLog('Відключення від сервера успішне, повідомлення
відправлене.');
      CURR_MAIL_IMAP_SMTP_BOX.Sent.Add(@(mes^),false,0);
      frm_Main.Tree_Mail_IMAP_SMTPBoxesClick(self);
  except
    on E:Exception do
       begin
       Application.MessageBox(pchar(E.message),'Не вдалося відправити пошту');
       CURR_MAIL_IMAP_SMTP_BOX.Outbox.Add(@(mes^),false,0);
       end;
  end;
  hide;
end;

procedure Tfrm_NewLetter.N4Click(Sender: TObject);
begin
  self.Hide;
end;

// Помістити лист в папку вихідні
procedure Tfrm_NewLetter.MoveNewLetter2OutboxExecute(Sender: TObject);
var
    mes : TIdMessage;
begin
  Application.MessageBox('','');
  mes:=CreateLetterFromForm^;
  CURR_MAIL_IMAP_SMTP_BOX.Outbox.Add( @mes,false,0);
  frm_Main.Tree_Mail_IMAP_SMTPBoxesClick(self);
  hide;
end;

// Додати файл
procedure Tfrm_NewLetter.NewAttachmentExecute(Sender: TObject);
var
    s : ^shortstring;
    st: TFileStream;
begin
  if not Dialog_OpenAttachment.Execute then exit;
  with List_NewAttachments.Items.Add do
  begin
    caption:=ExtractFileName(Dialog_OpenAttachment.FileName);
    try
      st:=TFileStream.Create(Dialog_OpenAttachment.FileName,fmOpenRead or
fmShareDenyNone);
      SubItems.Add(inttostr(st.Size));
      ImageIndex:=2;
      st.Free;
    except
      Application.MessageBox('Не вдалося отримати розмір файлу!','Помилка');
      exit;
    end;
    new(s);
    s^:=Dialog_OpenAttachment.FileName;
    data:=s;

    Splitter1.Visible:=true;
    List_NewAttachments.Visible:=true;
    UpdateStatusLine;
  end;
end;

// Контекстне меню -> Видалити з програми
procedure Tfrm_NewLetter.N6Click(Sender: TObject);
begin
  if List_NewAttachments.ItemIndex<0 then exit;
```

```
  dispose(List_NewAttachments.Items[List_NewAttachments.ItemIndex].Data);
  List_NewAttachments.DeleteSelected;
  UpdateStatusLine;
  if List_NewAttachments.Items.Count<=0 then
     begin
       Splitter1.Visible:=false;
       List_NewAttachments.Visible:=false;
     end;
end;

procedure Tfrm_NewLetter.List_NewAttachmentsMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  Item  : TListItem;
begin
  Item:=List_NewAttachments.GetItemAt(x,y);
  if Item<>nil then Item.Selected:=true;
end;

function Tfrm_NewLetter.CreateLetterFromForm: PidMessage;
var
    PLet : PidMessage;
    i    : integer;
    s    : ^shortstring;
begin
  frm_Main.Tree_Mail_IMAP_SMTPBoxesClick(self);
  if CurrentMail_IMAP_SMTPBoxSettings =nil then exit;

  new(PLet);
  PLet^:=TIdMessage.Create(self);

  PLet^.Subject:=Ed_Subject.Text;
  PLet^.CCList.EMail_IMAP_SMTPAddresses:=Ed_Copy.Text;
  PLet^.BccList.EMail_IMAP_SMTPAddresses:=Ed_HiddenCopy.Text;
  PLet^.Body.Assign(Memo_Body.Lines);
  PLet^.From.Text:= CURR_Mail_IMAP_SMTPbox_SETTINGS.POPEMail_IMAP_SMTP; raises
error on some servers
  PLet^.From.Text:=CurrentMail_IMAP_SMTPBoxSettings^.POPEMail_IMAP_SMTP;
  if Ed_ReplyTo.Text='' then
PLet^.ReplyTo.EMail_IMAP_SMTPAddresses:=CurrentMail_IMAP_SMTPBoxSettings.POPEMai
l_IMAP_SMTP
                      else
PLet^.ReplyTo.EMail_IMAP_SMTPAddresses:=Ed_ReplyTo.Text;
  PLet^.Recipients.EMail_IMAP_SMTPAddresses:=Ed_Reciever.Text;
  case Combo_Priority.ItemIndex of
      0 : PLet^.Priority:=mpHighest;
      1 : PLet^.Priority:=mpHigh;
      2 : PLet^.Priority:=mpNormal;
      3 : PLet^.Priority:=mpLow;
      4 : PLet^.Priority:=mpLowest;
      end;
  PLet^.Sender.Text:=Ed_Sender.Text;
  PLet^.ExtraHeaders.Assign(Memo_AdditionalHeaders.Lines);
  PLet^.ContentType:=Combo_ContexType.Text;

  for i:=1 to List_NewAttachments.Items.Count do
  begin
    s:=List_NewAttachments.Items.Item[i-1].data;
    TIdAttachment.Create(PLet^.MessageParts, s^);
  end;

  CreateLetterFromForm:=PLet;
end;

procedure Tfrm_NewLetter.UpdateStatusLine;
var
  AttSize : longint;
  i       : word;
begin
```

```
  AttSize:=0;
  for i:=1 to List_NewAttachments.Items.Count do
inc(AttSize,strtointdef(List_NewAttachments.Items[i-1].SubItems[0],0));
  if AttSize<>0 then StatusBar.Panels.Items[1].Text:='Розмір вкладених файлів :
'+inttostr(AttSize div 1024)+' Kb'
                else StatusBar.Panels.Items[1].Text:='';
end;

// відкрити/приховати додаткові поля
procedure Tfrm_NewLetter.Btn_AdvancedPageClick(Sender: TObject);
begin
  if Btn_AdvancedPage.Caption<>'Прибрати' then
     begin
     Panel_AdditionlSettings.Show;
     Splitter2.Show;
     Btn_AdvancedPage.Caption:='Прибрати';
     Btn_AdvancedPage.Width:=230;
     end
     else
     begin
     Splitter2.Hide;
     Panel_AdditionlSettings.Hide;

     Btn_AdvancedPage.Caption:='Додатково';
     Btn_AdvancedPage.Width:=170;
     end;
end;

procedure Tfrm_NewLetter.ClearForm;
begin
  Ed_Reciever.Clear;
  Ed_Copy.Clear;
  Ed_Subject.Clear;
  Ed_HiddenCopy.Clear;
  Ed_Sender.Clear;
  Memo_Body.Clear;
  Memo_AdditionalHeaders.Clear;
  List_NewAttachments.Clear;
  Ed_ReplyTo.Clear;
  Combo_Priority.ItemIndex:=2;

  IsToBeResend:=false;
  ResendMes.Clear;
  SendNewLetter.Enabled:=true;
end;

procedure Tfrm_NewLetter.N8Click(Sender: TObject);
begin
  ClearForm;
end;

function Tfrm_NewLetter.CheckIfFormIsFilled: boolean;
begin
  CheckIfFormIsFilled:=false;
  if Ed_Reciever.Text='' then
    begin
    Application.MessageBox('Ви забули вказати Кому ви хочете відправити цей
лист.','Помилка');
    exit;
    end;

  if Ed_Subject.Text='' then
    begin
    if Application.MessageBox('Ви дійсно хочете залишити повідомлення без
Теми?','Підтвердіть...',MB_YESNO    )=IDNo
       then exit;
    end;
  CheckIfFormIsFilled:=true;
end;
```

```
procedure Tfrm_NewLetter.FormCreate(Sender: TObject);
begin
  frm_NewLetter.ResendMes:=TIdMessage.Create(self);
end;

procedure Tfrm_NewLetter.FormShow(Sender: TObject);
begin
  CURR_Mail_IMAP_SMTPbox_SETTINGS:= CurrentMail_IMAP_SMTPBoxSettings^;

CURR_MAIL_IMAP_SMTP_BOX:=Mail_IMAP_SMTPBoxes.FindBox(CurrentMail_IMAP_SMTPBoxSet
tings.Name);
end;

end.
```

Файл frmMail_IMAP_SMTPSettings.pas – параметри облікових записів та створення нових поштових скриньок

```
unit frmMail_IMAP_SMTPSettings;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons,idSMTP;

type

  PMail_IMAP_SMTPBoxSettings = ^TMail_IMAP_SMTPBoxSettings;

  Tfrm_Mail_IMAP_SMTPSettings = class(TForm)
    ListBox_Mail_IMAP_SMTPBoxes: TListBox;
    GroupBox1: TGroupBox;
    GroupBox3: TGroupBox;
    GroupBox2: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Combo_SMTPType: TComboBox;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Ed_SMTPServer: TEdit;
    Ed_SMTPPort: TEdit;
    Ed_SMTPAccount: TEdit;
    Ed_SMTPPassword: TEdit;
    Ed_POPServer: TEdit;
    Ed_POPPort: TEdit;
    Ed_POPAccount: TEdit;
    Ed_POPPassword: TEdit;
    btn_Apply: TBitBtn;
    btn_NewMail_IMAP_SMTPBox: TBitBtn;
    GroupBox4: TGroupBox;
    Label11: TLabel;
    Ed_Subscript: TEdit;
    Ed_Mail_IMAP_SMTPBoxName: TEdit;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Combo_CheckEvery: TComboBox;
    BitBtn1: TBitBtn;
    Check_RetrieveHeadersOnly: TCheckBox;
    Check_DeleteFromServer: TCheckBox;
    Check_DontOpenHTML: TCheckBox;
    GroupBox5: TGroupBox;
    Label15: TLabel;
    Radio_ignore: TRadioButton;
    Radio_Recieve: TRadioButton;
    Radio_Ask: TRadioButton;
    RadioDelete: TRadioButton;
    Label10: TLabel;
    Ed_POPEMail_IMAP_SMTP: TEdit;
    Check_DynamicFolders: TCheckBox;
    Check_ShowBalloon: TCheckBox;

    procedure btn_NewMail_IMAP_SMTPBoxClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Combo_CheckEveryExit(Sender: TObject);
    procedure btn_ApplyClick(Sender: TObject);
```

```
    procedure ListBox_Mail_IMAP_SMTPBoxesClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure LoadSettingsFromRecords;
    procedure LoadFormFromBox(box : PMail_IMAP_SMTPBoxSettings);
    procedure SaveFormToBox(box : PMail_IMAP_SMTPBoxSettings);
    procedure ClearForm;
  end;

  TDuplicateAction = (Ask,Ignore,Recieve,DeleteFromServer);

  TMail_IMAP_SMTPBoxSettings = record
    SMTPServer  :shortstring;
    SMTPPort    :integer;
    SMTPAccount :shortstring;
    SMTPPass    :shortstring;
    AuthType    :TAuthenticationType;

    POPServer   :shortstring;
    POPPort     :integer;
    POPAccount  :shortstring;
    POPPass     :shortString;
    POPEMail_IMAP_SMTP    :shortstring;

    Subscript   :shortstring;
    Name        :shortstring;
    CheckEvery  :word;

    RetrieveOnlyHeaders : boolean;
    DeleteFromServer    : boolean;
    OpenHTMLasTEXT      : boolean;
    ifDuplicate         : TDuplicateAction;
    DynamicFolders      : boolean;
    ShowBalloonOnNewMes : boolean;
    end;

    procedure SaveSettingsToFile;
    procedure LoadSettingsFromfile;
    procedure findMail_IMAP_SMTPBoxAndSelectIt(name  : shortstring);

const
    Mail_IMAP_SMTPBoxSettingsFileName   = 'Mail_IMAP_SMTPboxes.dat';

var
  frm_Mail_IMAP_SMTPSettings         : Tfrm_Mail_IMAP_SMTPSettings;
  TotalMail_IMAP_SMTPBoxes           : word=0;
  Mail_IMAP_SMTPBoxesSettings        : array [1..300] of
PMail_IMAP_SMTPBoxSettings;
  CurrentMail_IMAP_SMTPBoxSettings   : PMail_IMAP_SMTPBoxSettings=nil;
  PrevMail_IMAP_SMTPBoxSettings      : PMail_IMAP_SMTPBoxSettings=nil;

implementation

uses frmMain;
{$R *.dfm}

procedure findMail_IMAP_SMTPBoxAndSelectIt(name  : shortstring);
var
    i : word;
Begin
  CurrentMail_IMAP_SMTPBoxSettings  :=  nil;
  for i:=1 to TotalMail_IMAP_SMTPBoxes do
    if Mail_IMAP_SMTPBoxesSettings[i]^.Name = name then
    begin
```

```
      CurrentMail_IMAP_SMTPBoxSettings:=Mail_IMAP_SMTPBoxesSettings[i];
      exit;
    end;
End;

procedure SaveSettingsToFile;
var
    SFile : shortstring;
    st    : TFileStream;
    i     : word;
Begin
  SFile:=ExtractFilePath(Application.ExeName)+Mail_IMAP_SMTPBoxSettingsFileName;
  try
    deletefile(SFile);
  except end;
  try
    st:=TFileStream.Create(SFile,fmOpenWrite or fmShareDenyNone);
  except
    try
      st:=TFileStream.Create(SFile,fmCreate);
    except
      Application.MessageBox( pchar('Не можу відкрити файл "'+SFile+'" для
запису !'),'Помилка збереження параметрів скриньки');
      exit;
    end;
  end;
  st.Write(TotalMail_IMAP_SMTPBoxes,sizeof(TotalMail_IMAP_SMTPBoxes));
  for i:=1 to TotalMail_IMAP_SMTPBoxes do

st.write(Mail_IMAP_SMTPBoxesSettings[i]^,sizeof(TMail_IMAP_SMTPBoxSettings));
  st.Free;
End;

procedure LoadSettingsFromfile;
var
    SFile : shortstring;
    st    : TFileStream;
    i     : word;
Begin
  SFile:=ExtractFilePath(Application.ExeName)+Mail_IMAP_SMTPBoxSettingsFileName;
  try
    st:=TFileStream.Create(SFile,fmOpenRead or  fmShareDenyNone);
  except
    Application.MessageBox( pchar('Не можу відкрити файл "'+SFile+'" для запису
!'),'Error loading Mail_IMAP_SMTPboxes settings');
    exit;
  end;

  st.Read(TotalMail_IMAP_SMTPBoxes,sizeof(TotalMail_IMAP_SMTPBoxes));
  for i:=1 to TotalMail_IMAP_SMTPBoxes do
      begin
      new(Mail_IMAP_SMTPBoxesSettings[i]);

st.Read(Mail_IMAP_SMTPBoxesSettings[i]^,sizeof(TMail_IMAP_SMTPBoxSettings));
      end;
  st.Free;
End;

procedure Tfrm_Mail_IMAP_SMTPSettings.LoadFormFromBox(box :
PMail_IMAP_SMTPBoxSettings);
Begin
  Ed_SMTPServer.Text  := box.SMTPServer;
  Ed_SMTPPort.Text    := inttostr(box.SMTPPort);
  Ed_SMTPAccount.Text := box.SMTPAccount;
  Ed_SMTPPassword.Text:= box.SMTPPass;

  Ed_POPServer.Text   := box.POPServer;
  Ed_POPPort.Text     := inttostr(box.POPPort);
  Ed_POPAccount.Text  := box.POPAccount;
```

```
  Ed_POPPassword.Text := box.POPPass;
  Ed_POPEMail_IMAP_SMTP.Text    := box.POPEMail_IMAP_SMTP;

  Ed_Subscript.Text   := box.Subscript;
  case box.AuthType of
       atNone : Combo_SMTPType.ItemIndex:=0;
       atLogin: Combo_SMTPType.ItemIndex:=1;
       end;

  if box.CheckEvery<>0 then Combo_CheckEvery.Text:=inttostr(box.CheckEvery)
                       else Combo_CheckEvery.Text:='Ніколи';

  Check_RetrieveHeadersOnly.Checked :=  box.RetrieveOnlyHeaders;
  Check_DeleteFromServer.Checked    :=  box.DeleteFromServer;

  Check_DontOpenHTML.Checked:=box.OpenHTMLasTEXT;
  case box.ifDuplicate of
    Ignore  : Radio_ignore.Checked:=true;
    Recieve : Radio_Recieve.Checked:=true;
    Ask     : Radio_Ask.Checked:=true;
    DeleteFromServer : RadioDelete.Checked:=true;
    end;
  Check_DynamicFolders.Checked := box.DynamicFolders;
  Check_ShowBalloon.Checked    := box.ShowBalloonOnNewMes;
End;

procedure Tfrm_Mail_IMAP_SMTPSettings.SaveFormToBox(box :
PMail_IMAP_SMTPBoxSettings);
Begin
  box.SMTPServer  := Ed_SMTPServer.Text;
  box.SMTPPort    := strtoint(Ed_SMTPPort.Text);
  box.SMTPAccount := Ed_SMTPAccount.Text;
  box.SMTPPass:= Ed_SMTPPassword.Text;

  box.POPServer   := Ed_POPServer.Text;
  box.POPPort     := strtoint(Ed_POPPort.Text);
  box.POPAccount  := Ed_POPAccount.Text;
  box.POPPass     := Ed_POPPassword.Text;
  box.POPEMail_IMAP_SMTP    := Ed_POPEMail_IMAP_SMTP.Text;

  box.Subscript   := Ed_Subscript.Text;
  case Combo_SMTPType.ItemIndex of
       0 : box.AuthType:=atNone;
       1 : box.AuthType:=atLogin;
       end;

  if AnsiUpperCase(Combo_CheckEvery.Text)<>'НІКОЛИ' then
     box.CheckEvery:=strtoint(Combo_CheckEvery.Text)
     else
     box.CheckEvery:=0;

  box.RetrieveOnlyHeaders :=  Check_RetrieveHeadersOnly.Checked;
  box.DeleteFromServer    :=  Check_DeleteFromServer.Checked;
  box.OpenHTMLasTEXT      :=  Check_DontOpenHTML.Checked;

  if Radio_ignore.Checked then box.ifDuplicate:=Ignore;
  if Radio_Recieve.Checked then box.ifDuplicate:=Recieve;
  if Radio_Ask.Checked then box.ifDuplicate:=Ask;
  if RadioDelete.Checked then box.ifDuplicate:=DeleteFromServer;

  box.DynamicFolders      :=  Check_DynamicFolders.Checked;
  box.ShowBalloonOnNewMes :=  Check_ShowBalloon.Checked;

End;

procedure Tfrm_Mail_IMAP_SMTPSettings.ClearForm;
Begin
Ed_SMTPServer.Text  :='smtp.<servername>.ru';
Ed_SMTPPort.Text    :='25';
```

```
Ed_SMTPAccount.Text :='';
Ed_SMTPPassword.Text:='';

Ed_POPServer.Text   :='pop.<servername>.ru';
Ed_POPPort.Text     :='110';
Ed_POPAccount.Text  :='';
Ed_POPPassword.Text :='';
Ed_POPEMail_IMAP_SMTP.Text    :='<nick>@<server>.ru';
Ed_Subscript.Text   :='Вася';

Combo_SMTPType.ItemIndex:=1;
Combo_CheckEvery.ItemIndex:=0;
End;

procedure Tfrm_Mail_IMAP_SMTPSettings.LoadSettingsFromRecords;
Begin
  if TotalMail_IMAP_SMTPBoxes <> 0 then
  begin
    if Mail_IMAP_SMTPBoxesSettings[1]<>nil then
    begin
      CurrentMail_IMAP_SMTPBoxSettings:=Mail_IMAP_SMTPBoxesSettings[1];

      LoadFormFromBox(CurrentMail_IMAP_SMTPBoxSettings);
      ListBox_Mail_IMAP_SMTPBoxes.Selected[0]:=true;
      PrevMail_IMAP_SMTPBoxSettings:=CurrentMail_IMAP_SMTPBoxSettings;
    end;
  end
  else CurrentMail_IMAP_SMTPBoxSettings:=nil;
End;

// створення нової поштової скриньки
procedure Tfrm_Mail_IMAP_SMTPSettings.btn_NewMail_IMAP_SMTPBoxClick(Sender:
TObject);
var
    i : word;
begin
  if Ed_Mail_IMAP_SMTPBoxName.Text<>'' then
  begin
     for i:=1 to ListBox_Mail_IMAP_SMTPBoxes.Count do
         if AnsiUpperCase(ListBox_Mail_IMAP_SMTPBoxes.Items[i-
1])=AnsiUpperCase(Ed_Mail_IMAP_SMTPBoxName.Text) then
            begin
            Application.MessageBox('Таке ім''я облікового запису вже
існує!','Помилка');
            exit;
            end;
     ListBox_Mail_IMAP_SMTPBoxes.Items.Add(Ed_Mail_IMAP_SMTPBoxName.Text);
     GroupBox1.Visible:=true;
     Radio_Ask.Checked:=true;
     inc(TotalMail_IMAP_SMTPBoxes);
     new(Mail_IMAP_SMTPBoxesSettings[TotalMail_IMAP_SMTPBoxes]);

CurrentMail_IMAP_SMTPBoxSettings:=Mail_IMAP_SMTPBoxesSettings[TotalMail_IMAP_SMT
PBoxes];
     CurrentMail_IMAP_SMTPBoxSettings.Name:=Ed_Mail_IMAP_SMTPBoxName.Text;
     if PrevMail_IMAP_SMTPBoxSettings<>nil then
SaveFormToBox(PrevMail_IMAP_SMTPBoxSettings);
     PrevMail_IMAP_SMTPBoxSettings:=CurrentMail_IMAP_SMTPBoxSettings;
     ClearForm;
     SaveFormToBox(CurrentMail_IMAP_SMTPBoxSettings);
     ListBox_Mail_IMAP_SMTPBoxes.Selected[ListBox_Mail_IMAP_SMTPBoxes.Count-
1]:=true;
     {Тепер створити і заповнити нову поштову скриньку}
     Mail_IMAP_SMTPBoxes.Add(Ed_Mail_IMAP_SMTPBoxName.Text);
     frm_Main.AddToLog('Створена поштова скринька
<'+Ed_Mail_IMAP_SMTPBoxName.Text+'>');
     frm_Main.ShowMail_IMAP_SMTPBoxesInTree;
  end
  else
```

```
  Application.MessageBox('Спочатку введіть ім''я вашої поштової
скриньки','Помилка');
end;

procedure Tfrm_Mail_IMAP_SMTPSettings.FormShow(Sender: TObject);
var
    i : word;
begin
  if (ListBox_Mail_IMAP_SMTPBoxes.Count=0)and(TotalMail_IMAP_SMTPBoxes<>0) then
     for i:=1 to TotalMail_IMAP_SMTPBoxes do
ListBox_Mail_IMAP_SMTPBoxes.Items.Add(Mail_IMAP_SMTPBoxesSettings[i].Name);
  LoadSettingsFromRecords;
  if CurrentMail_IMAP_SMTPBoxSettings=nil then GroupBox1.Visible:=false
                        else GroupBox1.Visible:=true;
end;

procedure Tfrm_Mail_IMAP_SMTPSettings.Combo_CheckEveryExit(Sender: TObject);
begin
  if AnsiUpperCase(Combo_CheckEvery.Text)<>'НІКОЛИ' then
  begin
     try
      strtoint(Combo_CheckEvery.Text);
     except
      Application.MessageBox('Невірне число','Помилка');
      Combo_CheckEvery.Text:='Ніколи';
      exit;
     end;
  end;
end;

procedure Tfrm_Mail_IMAP_SMTPSettings.btn_ApplyClick(Sender: TObject);
var
    i : word;
begin
  if ListBox_Mail_IMAP_SMTPBoxes.ItemIndex>=0 then
  begin
  SaveFormToBox(CurrentMail_IMAP_SMTPBoxSettings);
  end;
  SaveSettingsToFile;
  self.Hide;
  for i:=1 to TotalMail_IMAP_SMTPBoxes do
      if Mail_IMAP_SMTPBoxes.GetByIndex(i)<>nil then
Mail_IMAP_SMTPBoxes.GetByIndex(i).SetTimerInterval(Mail_IMAP_SMTPBoxesSettings[i
].CheckEvery);
end;

function FindMail_IMAP_SMTPBox(name : shortstring):PMail_IMAP_SMTPBoxSettings;
var
    i : word;
Begin
FindMail_IMAP_SMTPBox:=nil;

for i:=1 to TotalMail_IMAP_SMTPBoxes do
    if Mail_IMAP_SMTPBoxesSettings[i].Name = name then
    begin
      FindMail_IMAP_SMTPBox:=Mail_IMAP_SMTPBoxesSettings[i];
      exit;
    end;

End;

procedure Tfrm_Mail_IMAP_SMTPSettings.ListBox_Mail_IMAP_SMTPBoxesClick(Sender:
TObject);
begin
  if ListBox_Mail_IMAP_SMTPBoxes.ItemIndex >=0 then
  begin

CurrentMail_IMAP_SMTPBoxSettings:=FindMail_IMAP_SMTPBox(ListBox_Mail_IMAP_SMTPBo
xes.Items[ListBox_Mail_IMAP_SMTPBoxes.itemindex]);
```

```
  if CurrentMail_IMAP_SMTPBoxSettings=nil then
CurrentMail_IMAP_SMTPBoxSettings:=PrevMail_IMAP_SMTPBoxSettings;
  if PrevMail_IMAP_SMTPBoxSettings<>nil then
SaveFormToBox(PrevMail_IMAP_SMTPBoxSettings);
  PrevMail_IMAP_SMTPBoxSettings:=CurrentMail_IMAP_SMTPBoxSettings;
  LoadFormFromBox(CurrentMail_IMAP_SMTPBoxSettings);
  end;
end;

procedure Tfrm_Mail_IMAP_SMTPSettings.FormCreate(Sender: TObject);
begin
  LoadSettingsFromfile;
end;

// видалення поштової скриньки
procedure Tfrm_Mail_IMAP_SMTPSettings.BitBtn1Click(Sender: TObject);
var
    i,j : word;
begin
  if ListBox_Mail_IMAP_SMTPBoxes.ItemIndex >= 0 then
  begin
    if MessageDlg('Ви впевнені, що хочете видалити обліковий запис
"'+ListBox_Mail_IMAP_SMTPBoxes.Items[ListBox_Mail_IMAP_SMTPBoxes.itemindex]+'"
?',
                  mtConfirmation,[mbYes, mbNo], 0)= mrNo then exit;

    for i:=1 to TotalMail_IMAP_SMTPBoxes do
        if Mail_IMAP_SMTPBoxesSettings[i]^.Name =
ListBox_Mail_IMAP_SMTPBoxes.Items[ListBox_Mail_IMAP_SMTPBoxes.itemindex] then
        begin
          for j:=i to TotalMail_IMAP_SMTPBoxes-1 do
Mail_IMAP_SMTPBoxesSettings[j]:=Mail_IMAP_SMTPBoxesSettings[j+1];
          break;
        end;
    frm_Main.AddToLog('Видалено скриньку
<'+ListBox_Mail_IMAP_SMTPBoxes.Items[ListBox_Mail_IMAP_SMTPBoxes.itemindex]+'>')
;
    ListBox_Mail_IMAP_SMTPBoxes.DeleteSelected;
    dec(TotalMail_IMAP_SMTPBoxes);
    try
      ListBox_Mail_IMAP_SMTPBoxes.Selected[0]:=true;
      ListBox_Mail_IMAP_SMTPBoxesClick(self);
      frm_Main.ShowMail_IMAP_SMTPBoxesInTree;
    except end;
  end;
end;

procedure Tfrm_Mail_IMAP_SMTPSettings.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  btn_ApplyClick(self);
end;

end.
```

Файл frmSettings.pas – параметри програми

```
unit frmSettings;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons;

type
  Tfrm_Settings = class(TForm)
    GroupBox1: TGroupBox;
    Check_AllowLog: TCheckBox;
    L_MaxLogSize: TLabel;
    Ed_MaxLogSize: TEdit;
    L_kb: TLabel;
    btn_Close: TBitBtn;
    RadioShowInTaskBar: TRadioButton;
    RadioShowInTray: TRadioButton;
    procedure Check_AllowLogClick(Sender: TObject);
    procedure Ed_MaxLogSizeChange(Sender: TObject);
    procedure Ed_MaxLogSizeKeyPress(Sender: TObject; var Key: Char);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btn_CloseClick(Sender: TObject);
    procedure FormHide(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure LoadGlobalSettingsToForm;
    procedure LoadGlobalSettingsFromFile;
    procedure SaveGlobalSettingsFromForm;
    procedure SaveGlobalSettingsToFile;
  end;

  TSettings = record
    AllowLog  : boolean;
    MaxLogSize: longint;  // kb
    ShowInTray: boolean;
    end;

const
    GlobalSettingsFileName  = 'globalsettings.dat';

var
  frm_Settings: Tfrm_Settings;
  Settings    : TSettings=(AllowLog:true);

implementation

uses frmMain;

{$R *.dfm}

procedure Tfrm_Settings.FormCreate(Sender: TObject);
begin
  // Не працює { Нам потрібно це в frmMain.onCreate }
  LoadGlobalSettingsFromFile;
end;

procedure Tfrm_Settings.FormShow(Sender: TObject);
begin
  LoadGlobalSettingsToForm;
end;
```

```
procedure Tfrm_Settings.Check_AllowLogClick(Sender: TObject);
var
    b : boolean;
begin
  if Check_AllowLog.Checked then b:=true
                            else b:=false;
  L_MaxLogSize.Visible:=b;
  Ed_MaxLogSize.Visible:=b;
  L_kb.Visible:=b;
  Settings.AllowLog:=b;
  Settings.MaxLogSize:=strtointdef(Ed_MaxLogSize.text,500);
end;

procedure Tfrm_Settings.Ed_MaxLogSizeChange(Sender: TObject);
var
    maxs  : longint;
begin
  try
    if Ed_MaxLogSize.Text<>'' then
       maxs:=strtoint(Ed_MaxLogSize.text);
  except
    Application.MessageBox('Розмір лог файлу може бути від 0 до 2147483647
kb','Невірно вказане значення');
    Ed_MaxLogSize.Text:='500';
    exit;
  end;
  Settings.MaxLogSize:=maxs;
end;

const
    Digits :set of char =['1','2','3','4','5','6','7','8','9','0',#8];

procedure Tfrm_Settings.Ed_MaxLogSizeKeyPress(Sender: TObject;  var Key: Char);
begin
if not (key in Digits) then Key:=#0;
end;

procedure Tfrm_Settings.LoadGlobalSettingsToForm;
begin
  Ed_MaxLogSize.Text    :=  inttostr(Settings.MaxLogSize);
  Check_AllowLog.Checked  :=  Settings.AllowLog;
  RadioShowInTray.Checked :=  Settings.ShowInTray;
end;

procedure Tfrm_Settings.SaveGlobalSettingsFromForm;
begin
  Settings.AllowLog   :=  Check_AllowLog.Checked;
  Settings.MaxLogSize :=  strtointdef(Ed_MaxLogSize.text,500);
  Settings.ShowInTray :=  RadioShowInTray.Checked;
end;

procedure Tfrm_Settings.LoadGlobalSettingsFromFile;
var
    st      : TFileStream;
    SFName  : shortstring;
begin
  SFName:=ExtractFilePath(Application.ExeName)+GlobalSettingsFileName;
  try
    st:=TFileStream.Create(SFName,fmOpenRead or fmShareDenyNone);
  except
    try
      st:=TFileStream.Create(SFName,fmCreate or fmShareDenyNone);
    except
      Application.MessageBox(pchar('Неможливо відкрити/створити файл
"'+SFName+'"'),'Помилка завантаження глобальних параметрів');
      exit;
    end;
    fillchar(Settings,sizeof(Settings),0);
    LoadGlobalSettingsToForm;
```

```
    st.Free;
    exit;
  end;
  fillchar(Settings,sizeof(Settings),0);
  st.Read(Settings,sizeof(Settings));
  st.Free;
end;

procedure Tfrm_Settings.SaveGlobalSettingsToFile;
var
    st      : TFileStream;
    SFName  : shortstring;
begin
  SFName:=ExtractFilePath(Application.ExeName)+GlobalSettingsFileName;
  try
    st:=TFileStream.Create(SFName,fmOpenWrite or     fmShareDenyNone);
  except
    try
      st:=TFileStream.Create(SFName,fmCreate or fmShareDenyNone);
    except
      Application.MessageBox(pchar('Неможливо відкрити/створити файл
"'+SFName+'"'),'Помилка завантаження глобальних параметрів');
      exit;
    end;
  end;

  st.Write(Settings,sizeof(Settings));
  st.Free;
end;

procedure Tfrm_Settings.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  SaveGlobalSettingsFromForm;
  SaveGlobalSettingsToFile;
end;

procedure Tfrm_Settings.btn_CloseClick(Sender: TObject);
begin
  self.Hide;
end;

procedure Tfrm_Settings.FormHide(Sender: TObject);
begin
  SaveGlobalSettingsFromForm;
  SaveGlobalSettingsToFile;
  if Settings.ShowInTray then
   begin
   frm_Main.Tray.IconVisible:=true;
   frm_Main.Tray.HideTaskbarIcon;
   frm_Main.Tray.MinimizeToTray:=true;
   end
   else
   begin
   frm_Main.Tray.IconVisible:=false;
   frm_Main.Tray.ShowMainForm;
   frm_Main.Tray.ShowTaskbarIcon;
   frm_Main.Tray.MinimizeToTray:=false;
   end;
end;

end.
```

Файл about.pas – файл довідки

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    Image1: TImage;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```