МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розроблення програмного забезпечення для підтримки комунікаційної діяльності

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконала: студентка групи МгЗІТ-23

Олена МИТЕЛЬСЬКА

Науковий керівник: к.т.н., доцент Тетяна ДЕМКІВСЬКА

Рецензент: д.т.н., професор Олег НІКОНОВ

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерних наук

Спеціальність 122 «Комп'ютерні науки»

Освітня програма «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

_____ Наталія ЧУПРИНКА
"_____" _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ
Мительській Олені Валентинівні

1. Тема роботи: «Розроблення програмного забезпечення для підтримки комунікаційної діяльності», Науковий керівник роботи Демківська Тетяна Іванівна, к.т.н., доцент затверджені наказом вищого навчального закладу від "03" вересня 2024 року №188 -уч.
2. Строк подання студентом роботи: 21.11.2024 р.
3. Вихідні дані до роботи: публікації та статті з даної тематики, нормативна та довідникова література.
4. Зміст дипломної роботи: Вступ. Розділ 1. Аналіз предметної області. Розділ 2. Проєктування мобільного застосунку. Розділ 3. Програмна реалізація. Загальні висновки. Список використаних джерел.

Дата видачі завдання**:** 04.09.2024р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	05.09.2024	
2	Розділ 1 (назва)	20.09.2024	
3	Розділ 2	03.10.2024	
4	Розділ 3	14.10.2024	
5	Розділ 4	25.10.2024	
6	Висновки	28.10.2024	
7	Оформлення дипломної магістерської роботи (чистовий варіант)	31.10.2024	
8	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	01.11.2024	
9	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за10 днів до захисту)	04.11.2024	
10	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (за 7 днів дозахисту)	07.11.2024	

Студент _____ Олена МИТЕЛЬСЬКА

(підпис)

Науковий керівник роботи _____ Тетяна ДЕМКІВСЬКА

(підпис)

АНОТАЦІЯ

Мительська О.В. «Розроблення програмного забезпечення для підтримки комунікаційної діяльності» – Рукопис.

Дипломна магістерська робота за спеціальністю 122 – «Комп'ютерні науки та технології» – Київський національний університет технологій та дизайну, Київ, 2024 рік.

Розроблено програмне забезпечення для підтримки комунікаційної діяльності, яке забезпечує багатоплатформенність та адаптивність інтерфейсу користувача. Програмне забезпечення створено за допомогою технологій Delphi та FireMonkey, що дозволяє їй функціонувати на ПК, планшетах і мобільних пристроях. Інтерфейс розроблено з акцентом на зручність використання, зокрема через підтримку функції Drag-and-Drop, що спрощує процес введення та обробки інформації.

Створено UML-діаграму класів, яка деталізує структуру програмного продукту та взаємодію між його компонентами. Забезпечено перевірку введених даних та надійне підключення до бази даних, що гарантує стабільну роботу програми в реальних умовах.

Під час тестування проведено інсталяційне, продуктивне, навантажувальне тестування та тестування на переривання, які підтвердили стабільність і надійність роботи програмного забезпечення.

Розроблене програмне забезпечення повністю відповідає вимогам користувачів, демонструючи стабільність, адаптивність та зручність у використанні, що робить його ефективним інструментом для управління даними в комунікаційній діяльності різних галузей економіки.

Ключові слова: програмне забезпечення, інтерфейс, інформаційна підтримка, користувач.

ABSTRACT

Mitelska O.V. «Development of software to support communication activities» – Manuscript.

Master's thesis in specialty 122 – «Computer Science and Technology» – Kyiv National University of Technology and Design, Kyiv, 2024.

Developed software to support communication activities, which provides multi-platform and adaptive user interface. The software is built using Delphi and FireMonkey technologies, allowing it to function on personal computers, tablets and mobile devices. The interface is designed with an emphasis on ease of use, in particular through support for the Drag-and-Drop function, which simplifies the process of entering and processing information.

A UML class diagram is created that details the structure of a software product and the interactions between its components. Ensured verification of input data and reliable connection to the database, which guarantees the stability of the robot program in real conditions.

During testing, installation, performance, load testing, and interruption testing are performed to confirm the stability and reliability of robot software.

The developed software fully meets the requirements of users, demonstrating stability, adaptability and ease of use, which makes it an effective tool for data management and communication activities in various sectors of the economy.

Keywords: software, interface, information support, user.

ЗМІСТ

ВСТУП

Актуальність теми кваліфікаційної дипломної роботи. У сучасних умовах розвитку інформаційного суспільства ефективна комунікація є одним із ключових факторів успішного функціонування організацій різних сфер діяльності. Зростання обсягу інформації, швидкість її обробки та необхідність безперебійного обміну даними зумовлюють потребу у впровадженні спеціалізованого програмного забезпечення для підтримки комунікаційної діяльності.

Розроблення таких програмних рішень спрямоване на оптимізацію процесів обміну інформацією, підвищення продуктивності комунікацій та забезпечення безпеки даних. Особливого значення набуває адаптивність і багатоплатформність програмного забезпечення, що дозволяє забезпечити доступність інформаційних ресурсів на різних пристроях – від персональних комп’ютерів до мобільних пристроїв.

Тому розроблення програмного забезпечення для підтримки комунікаційної діяльності є актуальною для різних сфер – бізнесу, освіти, державного управління. Такі рішення допомагають оптимізувати внутрішні та зовнішні комунікації, покращити обмін інформацією між працівниками та підвищити ефективність прийняття управлінських рішень.

Метою даної роботи є розроблення програмного забезпечення для підтримки комунікаційної діяльності, яке забезпечить ефективний обмін інформацією між користувачами, підвищить продуктивність комунікаційних процесів та гарантуватиме безпеку переданих даних.

Завдання дослідження.

1. Проаналізувати сучасні підходи та існуючі програмні рішення для підтримки комунікаційної діяльності.

2. Розробити архітектуру програмного забезпечення для підтримки комунікаційної діяльності.

3. Розробити програмного забезпечення для підтримки комунікаційної діяльності.

Об'єктом дослідження є процес комунікаційної діяльності в організаціях, який забезпечується за допомогою інформаційних технологій, включаючи засоби обміну даними, управління інформаційними потоками та забезпечення безпеки переданих повідомлень.

Предметом дослідження є програмні компоненти, технології та інструменти для розроблення програмного забезпечення, що підтримує комунікаційну діяльність. Зокрема, це засоби забезпечення багатоплатформності, адаптивного інтерфейсу, функціональності Drag-and-Drop, а також механізми захисту та обробки даних.

Методи дослідження: системний аналіз для оцінки існуючих рішень та визначення вимог до програмного забезпечення; моделювання – для побудови UML-діаграм класів і взаємодії компонентів програмного забезпечення з метою чіткого опису архітектури системи; емпіричні методи – інсталяційне, продуктивне, навантажувальне та тестування на переривання для оцінки стабільності, продуктивності та надійності роботи програмного забезпечення;

Інформаційна база. При написанні кваліфікаційної магістерської роботи використано широкий спектр літературних джерел, зокрема наукові публікації з фондів бібліотеки Київського національного університету технологій та дизайну, стандарти та технічна документація, ресурси глобальної мережі Інтернет.

Наукова новизна одержаних результатів. Розроблено багатоплатформне програмне забезпечення для підтримки комунікаційної діяльності, яке поєднує адаптивний інтерфейс користувача з функцією *Drag-*

and-Drop, що забезпечує зручність і простоту введення та обробки інформації. Інтеграція сучасних технологій Delphi та FireMonkey дозволила створити універсальне рішення для роботи на різних пристроях (ПК, планшетах, мобільних телефонах), що підвищує гнучкість та доступність програмного продукту. Розроблено UML-діаграму класів, яка детально описує взаємодію компонентів системи та їх ієрархію, забезпечуючи підвищену прозорість структури програмного забезпечення. Запропоновано підхід до забезпечення безпеки комунікацій шляхом інтеграції надійних механізмів перевірки введених даних та підключення до бази даних, що підвищує надійність обміну інформацією у комунікаційних процесах. Комплексне тестування (інсталяційне, продуктивне, навантажувальне, на переривання) підтвердило стабільність роботи програмного забезпечення в реальних умовах, що є новим підходом до забезпечення якості в комунікаційних рішеннях для різних галузей економіки.

Практичне значення одержаних результатів. Розроблене програмне забезпечення для підтримки комунікаційної діяльності може бути використано у різних сферах, зокрема у бізнесі, освіті, державному управлінні та ін.

Апробація результатів кваліфікаційної роботи: результати доповідались та обговорювались на VIII міжнародної науково-практичної конференції «Мехатронні системи: інновації та інжиніринг» – «MSIE-2024» 07 листопада 2024 року м.Київ.

Структура і обсяг роботи: робота складається зі вступу, 3 розділів, висновків, списку використаних джерел (62 найменування), 3 додатків. Загальний обсяг кваліфікаційної роботи 77 сторінок комп’ютерного тексту (без додатків).

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Сучасні підходи для підтримки комунікаційної діяльності

Сучасні підходи для підтримки комунікаційної діяльності включають використання новітніх технологій та стратегій для покращення ефективності обміну інформацією в організаціях, між окремими людьми та групами [1-6]. Зокрема, виділимо кілька основних напрямків:

1. Цифрові комунікації та нові медіа. Сучасні організації використовують цифрові інструменти для швидкої та ефективної комунікації;

- соціальні мережі (Facebook, Instagram, Twitter, LinkedIn) стали основними каналами для комунікації з аудиторією, формування іміджу бренду, а також для зворотного зв’язку;

- мобільні додатки та месенджери (Viber, WhatsApp, Slack та інші платформи) для оперативного обміну повідомленнями, координації роботи команд і надання підтримки клієнтам;

- вебінари та онлайн-конференції (платформи для проведення онлайн-заходів), що дозволяють різним установам комунікувати з великою кількістю людей в реальному часі.

2. Персоналізація комунікацій. Завдяки аналізу великих даних та технологіям штучного інтелекту (AI), компанії здатні персоналізувати свою комунікацію:

- використання автоматичних систем для створення персоналізованих розсилок, реклами, пропозицій;

- застосування аналітики для адаптації контенту під інтереси та поведінку користувачів.

3. Кросс-канальна комунікація. Сучасні підходи передбачають інтеграцію різних каналів комунікації в єдину стратегію. Кросс-канальна комунікація

дозволяє забезпечити цілісне сприйняття бренду через різні точки взаємодії з клієнтами:

- одночасне використання онлайн (сайти, соціальні мережі) та офлайн каналів (магазини, кол-центри);

- створення єдиного досвіду для клієнтів незалежно від того, через який канал вони взаємодіють з компанією.

4. Інтерактивність та залучення аудиторії. Сучасні підходи підкреслюють важливість взаємодії з аудиторією та її залучення:

- інтерактивний контент (опитування, голосування, конкурси) дозволяє не тільки передавати інформацію, а й активно взаємодіяти з користувачами;

- гейміфікація – використання елементів ігор в комунікаційних процесах (наприклад, балів за взаємодію з контентом або виконання завдань), що підвищує залученість та лояльність.

5. Автоматизація та штучний інтелект. Автоматизація комунікаційних процесів стає невід'ємною частиною сучасних стратегій:

- використання чат-ботів для автоматичного реагування на запити клієнтів;

- автоматизація маркетингових кампаній, що дозволяє своєчасно та точно доставляти повідомлення.

6. Прозорість та етика комунікації. Важливим аспектом сучасних підходів є акцент на етичну та прозору комунікацію:

- організації зобов'язані чітко донести свою політику щодо збору даних, конфіденційності та безпеки;

- врахування соціальних і екологічних аспектів у комунікаціях, що особливо важливо для формування довіри до бренду.

7. Глобалізація та багатомовність. Сучасні комунікаційні стратегії повинні враховувати глобальну аудиторію, тому важливим є:

- адаптація контенту до різних культурних контекстів;

- використання багатомовних платформ для забезпечення доступності інформації для різних груп користувачів.

8. Аналіз і зворотний зв’язок. Використання аналітики та зворотнього зв’язку для оцінки ефективності комунікацій:

- оцінка ефективності компаній через аналіз даних;

- активне використання зворотнього зв’язку від клієнтів для корекції стратегії.

9. Інклюзивність та доступність. Усі стратегії повинні бути інклюзивними та доступними для широкої аудиторії:

- використання інклюзивної мови та адаптація контенту для людей з обмеженими можливостями (наприклад, субтитри для відео, адаптовані сайти).

Таким чином, сучасні підходи для підтримки комунікаційної діяльності зосереджені на інтеграції нових технологій, ефективному управлінні комунікаційними процесами, персоналізації взаємодії з аудиторією та прозорості в усіх етапах комунікації.

1.2. Програмне забезпечення для комунікацій у різних сферах діяльності

Програмне забезпечення для комунікацій широко використовується в різних сферах діяльності для покращення ефективності взаємодії, оптимізації процесів і забезпечення швидкої та зручної комунікації між людьми та організаціями. Залежно від сфери застосування, вибір програмного забезпечення може варіюватися. Розглянемо деякі програмні засоби, які використовуються для комунікацій у різних сферах:

1. Корпоративні комунікації. Для забезпечення ефективної комунікації в межах компаній і між відділами використовуються спеціалізовані платформи:

- *Slack* – популярний інструмент для командної комунікації, який дозволяє створювати канали для обговорення проєктів, обмінюватися файлами та інтегрувати з іншими бізнес-інструментами [7-9];

- *Microsoft Teams* – платформа для організації відеоконференцій, чатів і співпраці в реальному часі, яка інтегрується з іншими продуктами Microsoft [10];

- *Asana* – інструмент для управління проектами, що дозволяє командам спільно працювати над завданнями, слідкувати за дедлайнами та обмінюватися повідомленнями [11];

- *Trello* – система для управління проектами з інтегрованими комунікаційними можливостями, що дозволяє зберігати всі завдання, дедлайни і коментарі в одному місці [12].

2. Маркетингові комунікації. Програмне забезпечення, яке дозволяє організаціям комунікувати з клієнтами та аудиторією для просування продуктів і послуг:

- *Mailchimp* – інструмент для автоматизації email-маркетингу, що дозволяє створювати, відправляти та аналізувати email-розсилки [13];

- *Hootsuite або Buffer* – платформи для керування акаунтами в соціальних мережах, планування та аналітики постів, а також моніторингу взаємодії з аудиторією [14];

- *HubSpot* [15]– повна платформа для автоматизації маркетингу, яка включає CRM-систему, інструменти для email-маркетингу, блогінгу, аналізу та генерації лідів.

3. Навчання та освітні комунікації. В освітній сфері комунікаційні технології дозволяють забезпечити ефективну взаємодію між викладачами, студентами та адміністрацією:

- *Zoom* – платформа для відеоконференцій, що активно використовується в освіті для онлайн-уроків, лекцій та семінарів [16];

- *Google Classroom* – інструмент для організації навчального процесу в онлайн-форматі, що дозволяє викладачам створювати завдання, отримувати зворотний зв'язок і спілкуватися [17].

- *Moodle* – система управління навчанням, яка дозволяє створювати курси, організовувати онлайн-обговорення, тестування та оцінювання.

4. Програмне забезпечення для відеоконференцій. Платформи для організації відеозв'язку та зустрічей, які широко використовуються в бізнесі, освіті, медицині:

- *Skype* – класичний інструмент для відео- та голосових дзвінків, який також підтримує миттєві повідомлення [18];

- *Google Meet* – відеоконференційна платформа від Google, яка є частиною Google Workspace і дозволяє організовувати онлайн-зустрічі [19];

- *Cisco Webex* – платформа для відеоконференцій і вебінарів, яка активно використовується в бізнес-середовищі для великих зустрічей [20].

Програмне забезпечення для комунікацій у різних сферах охоплює широкий спектр інструментів, від простих месенджерів до спеціалізованих платформ для відеоконференцій, обробки запитів клієнтів або управління проектами. Правильний вибір програмного забезпечення дозволяє організаціям оптимізувати процеси комунікації, покращити взаємодію з аудиторією та підвищити ефективність роботи.

1.3. Технології та інструменти для розробки комунікаційних платформ

Розробка комунікаційних платформ [21-29] вимагає використання сучасних технологій і інструментів для забезпечення ефективності, масштабованості та

безпеки. Такі платформи можуть охоплювати різноманітні типи комунікацій: текстові, голосові, відео- та мультимедійні. Ось основні технології та інструменти, що використовуються для створення таких платформ:

1. Технології для розробки користувацьких інтерфейсів (UI/UX). Комунікаційна платформа повинна бути зручна для користувачів, тому важливо забезпечити хороший інтерфейс і досвід взаємодії:

- *React / Vue.js / Angular* – популярні JavaScript-фреймворки для розробки інтерактивних інтерфейсів. Вони забезпечують швидке відображення змін і інтерактивність;

- *Figma / Adobe XD* – інструменти для проектування та прототипування інтерфейсів, які дозволяють дизайнерам створювати зручні і ефективні макети для платформ;

- *Bootstrap / Tailwind CSS* – CSS-фреймворки, що полегшують створення адаптивних та стильних інтерфейсів.

2. Технології для розробки серверної частини (Backend). Серверна частина комунікаційної платформи відповідає за обробку запитів, зберігання даних і управління з'єднаннями між користувачами:

- *Node.js* – серверна платформа на JavaScript, яка широко використовується для створення масштабованих комунікаційних платформ завдяки своїй асинхронній архітектурі;

- *Python (Django, Flask)* – популярні фреймворки для веб-розробки на Python, які можуть бути використані для швидкої розробки бекенд-рішень;

- *Ruby on Rails* – фреймворк для швидкої розробки серверної частини комунікаційних платформ;

- *Java (Spring)* – ще одна потужна технологія для створення бекенду, яка має високу продуктивність і може обробляти великий обсяг одночасних з'єднань;

- *Go (Golang)* – мова програмування, яка підходить для розробки масштабованих і швидких серверних додатків, зокрема для реального часу.

3. Реалізація віртуальних чатів та голосових комунікацій. Для реалізації голосових і текстових чатів використовуються спеціалізовані бібліотеки та інструменти:

- *WebRTC* – технологія для реалізації голосових та відео дзвінків у реальному часі без необхідності в установці додаткових плагінів. Це важливий інструмент для відеоконференцій, голосових дзвінків і чатових функцій.

- *Socket.io* – бібліотека для створення реального часу, яка дозволяє з'єднувати клієнта з сервером для миттєвого обміну повідомленнями (використовується в чатах).

- *Twilio* – платформа для розробки голосових та SMS-комунікацій. Вона дозволяє легко інтегрувати дзвінки, SMS, відеозв'язок та інші функції в додатки.

4. Бази даних. Для зберігання даних про користувачів, повідомлення та іншу інформацію використовуються різні бази даних:

- *MongoDB* – документо-орієнтована NoSQL база даних, що добре підходить для зберігання неструктурованих даних, таких як чати, повідомлення, мультимедійні файли;

- *PostgreSQL* – реляційна база даних, яка є чудовим варіантом для зберігання структурованих даних і забезпечує високу надійність та масштабованість;

- *MySQL* – ще одна популярна реляційна база даних, що використовується для зберігання великих обсягів даних і для високонавантажених платформ.

5. Комунікаційні протоколи та API. Для підтримки комунікацій між клієнтами та серверами, а також між різними частинами системи використовуються наступні технології:

- *REST API / GraphQL* – популярні технології для створення API, через які клієнти можуть обмінюватися даними з сервером. GraphQL дозволяє ефективніше запитувати дані, що особливо важливо для великих платформ з великою кількістю запитів;

- *WebSockets* – дозволяють забезпечити двосторонній зв'язок між клієнтом і сервером для обміну даними в реальному часі. Використовується для чатів, оновлень повідомлень, ігор;

- *MQTT* – легкий протокол обміну повідомленнями для зв'язку в реальному часі, використовується в IoT-пристроях, а також у випадках, коли потрібна мінімальна затримка.

6. Масштабування та обробка великих навантажень. Для забезпечення високої доступності і швидкої роботи при великій кількості користувачів необхідно застосовувати технології масштабування:

- *Kubernetes* – платформа для автоматизованого розгортання, масштабування і управління контейнеризованими додатками, що забезпечує гнучкість і високу доступність;

- *Docker* – технологія контейнеризації, що дозволяє створювати ізольовані середовища для додатків і розгортати їх на різних платформах;

- *Redis* – база даних у пам'яті, яка використовується для кешування та зберігання часто використовуваних даних у реальному часі, щоб зменшити навантаження на основну базу даних.

7. Безпека. Безпека є критично важливим аспектом при розробці комунікаційних платформ:

- *OAuth / JWT* – технології для безпечної автентифікації та авторизації користувачів у системах;

- *SSL/TLS* – забезпечення шифрування даних під час їх передачі через мережу для захисту від перехоплення;

- *End-to-End encryption (E2EE)* – забезпечує шифрування повідомлень на стороні клієнта та декодування їх лише на стороні отримувача (захист особистих даних у чатах, відеодзвінках).

8. Інструменти для тестування та моніторингу. Для забезпечення стабільної роботи комунікаційних платформ важливо застосовувати інструменти для тестування та моніторингу:

- *Jest / Mocha* – інструменти для тестування коду, які допомагають автоматизувати процес перевірки роботи платформи;

- *Prometheus / Grafana* – інструменти для моніторингу і візуалізації даних про продуктивність і навантаження на сервери;

- *New Relic* – платформа для моніторингу та оптимізації продуктивності веб-додатків, яка дозволяє відстежувати час відгуку і ресурси.

Розробка комунікаційних платформ потребує комплексного підходу та використання різних технологій, що охоплюють як фронтенд, так і бекенд, а також забезпечують високий рівень безпеки, масштабованості та ефективності. Вибір правильних інструментів і технологій залежить від специфіки платформи, її цілей та потреб користувачів.

1.4. Порівняльний аналіз існуючих рішень у сфері комунікаційних платформ

Ринок комунікаційних платформ в останні роки швидко розвивається, і існує безліч рішень, що пропонують різні функціональні можливості для комунікацій в бізнесі, освіті, охороні здоров’я, уряді та інших сферах. У цьому порівняльному аналізі розглянемо популярні платформи для організації комунікацій, їх функціональні можливості, переваги та недоліки, щоб допомогти обрати найкраще рішення для конкретних потреб [30-32].

1. Slack. Основні функції: чати в реальному часі (приватні та групові); інтеграція з численними сторонніми додатками (Google Drive, Asana, Trello, GitHub, тощо); пошук по історії повідомлень; канали для організації комунікацій за проєктами або темами; зберігання файлів та обмін документами; додаткові можливості через боти та автоматизацію. До переваг можна віднести: велика кількість інтеграцій з іншими інструментами; легкість у користуванні та інтуїтивно зрозумілий інтерфейс; підтримка різноманітних каналів для співпраці; хороша можливість для автоматизації за допомогою ботів. Недоліки: може бути складно організувати складніші відео конференції (потрібно використовувати додаткові інструменти); можливості управління правами доступу і ролями обмежені порівняно з Teams.

2. Microsoft Teams. Основні функції: повноцінні відео конференції та чати; інтеграція з Microsoft 365 (Word, Excel, PowerPoint, OneNote); канали для співпраці та завдань; вбудовані можливості для обміну документами та зберігання файлів; глибока інтеграція з корпоративними інструментами, такими як SharePoint і OneDrive; організація вебінарів та зустрічей в реальному часі. До переваг відноситься: інтеграція з Microsoft 365; можливості для організації великих відео конференцій і вебінарів; потужні інструменти для управління завданнями і проєктами; додаткові можливості для безпеки та управління доступом. Недоліки**:** може бути складним для освоєння користувачами, які не знайомі з екосистемою Microsoft; обмежена кількість інтеграцій з сторонніми додатками порівняно з Slack.

3. Zoom. Основні функції: Відео конференції з високою якістю зображення та звуку; можливість проведення вебінарів та онлайн-зустрічей; запис зустрічей, чат, екранна демонстрація; інтеграція з календарями (Google, Outlook); підтримка великих зустрічей до 1000 учасників (в залежності від тарифу). До переваг відноситься: висока якість відео та звуку; проста організація вебінарів

та зустрічей; масштабованість для великих аудиторій; інтеграція з іншими інструментами (Google, Microsoft). Недоліки: платний доступ до всіх функцій (обмеження на кількість учасників і час зустрічі в безкоштовній версії); інтерфейс може бути складним для нових користувачів; не має функцій для обміну файлами та управління завданнями.

4. Google Meet. Основні функції: відео конференції та чати; інтеграція з Google Calendar і Google Workspace; можливість проведення зустрічей до 100 учасників в безкоштовній версії; безпека та захищений доступ; легкий доступ до файлів через Google Drive. До переваг відноситься: безкоштовний доступ до основних функцій; інтеграція з іншими продуктами Google (Google Docs, Sheets, Drive); легкість у використанні, особливо для користувачів Google Workspace; простота організації зустрічей для малих і середніх груп. Недоліки: обмежена кількість функцій порівняно з Zoom; меньше налаштувань для великих зустрічей і вебінарів; немає підтримки для складних сценаріїв вебінарів і тренінгів.

5. Trello. Основні функції: візуальне управління проектами через канбан-дошки; створення карток для завдань з можливістю додавання коментарів, файлів і термінів; інтеграція з іншими інструментами (Slack, Google Drive, Zapier); простота у використанні та налаштуванні. До переваг відноситься: простота використання і налаштування; візуальний підхід до управління проектами; безкоштовна версія з основними функціями. Недоліки: обмежені можливості для складних проєктів і аналітики; меньше функцій порівняно з іншими платформами для управління проєктами.

6. Asana. Основні функції: управління проєктами за допомогою списків, календарів і канбан-дошок; детальне управління завданнями (терміни, пріоритети, ресурси); інтеграція з іншими сервісами (Slack, Google Drive, Microsoft Teams); можливість аналізувати продуктивність за допомогою звітів.

До переваг відноситься: більш розширена функціональність для великих проєктів; можливість гнучкого налаштування процесів; підтримка складної аналітики і звітності. Недоліки: вища складність освоєння порівняно з Trello.

Кожна комунікаційна платформа має свої переваги, і вибір оптимального інструменту залежить від конкретних вимог організації, її масштабів та типу діяльності. Для бізнесу, де важлива інтеграція з іншими корпоративними інструментами та безпека, перевагу можуть мати Microsoft Teams або Zoom. Для невеликих команд або особистих комунікацій зручним буде Slack. Якщо ж основна потреба полягає в управлінні проєктами, то Asana і Trello надають потужні можливості для організації роботи і досягнення цілей.

1.5. Проблеми та обмеження сучасних рішень у сфері комунікаційних платформ

Сучасні комунікаційні платформи, незважаючи на свій великий потенціал і розвиток, стикаються з низкою проблем та обмежень, які можуть впливати на їх ефективність і зручність використання. Розглянемо основні проблеми, з якими стикаються ці рішення:

1. Проблеми з безпекою та конфіденційністю:

- багато комунікаційних платформ зберігають великі обсяги особистих даних, документів та повідомлень. Невідповідність стандартам безпеки або вразливості в системах можуть призвести до витоку інформації, що ставить під загрозу конфіденційність користувачів;

- не всі платформи мають повне шифрування кінцевих точок (end-to-end encryption), що може дозволити стороннім особам або навіть самим постачальникам послуг отримувати доступ до особистих даних користувачів;

- злочинці можуть використовувати вразливості платформ для запуску шкідливих програм або фішинг-атак, що може призвести до втрати даних або порушення роботи платформи.

2. Масштабованість і продуктивність:

- платформи, які підтримують велику кількість користувачів або великий обсяг одночасних з'єднань (наприклад, відеоконференції з сотнями учасників), можуть стикатися з проблемами продуктивності, такими як затримки, погіршення якості звуку та зображення, зависання та розриви з'єднання;

- багато платформ мають ліміти на кількість учасників у чатах або відеоконференціях, що може обмежувати їх використання в великих організаціях чи для глобальних заходів;

- деякі платформи, особливо ті, що працюють з великими обсягами мультимедійних даних (відео, зображення, файли), можуть споживати значну кількість системних ресурсів (оперативної пам'яті, процесора), що призводить до уповільнення роботи на менш потужних пристроях.

3. Інтероперабельність і інтеграція з іншими системами:

- у деяких випадках комунікаційні платформи не підтримують інтеграцію з іншими інструментами, такими як системи управління проєктами, CRM або документообіг. Це ускладнює їх використання в бізнес-середовищі, де важлива єдина екосистема інструментів;

- хоча деякі платформи, такі як Slack і Microsoft Teams, пропонують багато можливостей для інтеграції, інші сервіси можуть мати обмежену кількість доступних додатків і плагінів для підключення до сторонніх систем, що обмежує функціональність.

4. Зручність та доступність для користувачів:

- більшість платформ мають велику кількість функцій, але новим користувачам може бути важко орієнтуватися в інтерфейсі і налаштуваннях. Це

може призвести до зниження продуктивності, особливо у великих команд, де є новачки або малодосвідчені користувачі;

- деякі платформи не мають зручних мобільних або веб-версій, що обмежує можливість користуватися ними на різних пристроях, особливо в умовах віддаленої роботи або мобільності співробітників;

- інтерфейси платформ часто оптимізовані для великих моніторів, і їх використання на малих екранах (наприклад, на смартфонах) може бути незручним.

5. Залежність від Інтернет-з'єднання:

- комунікаційні платформи часто вимагають стабільного та швидкого Інтернет-з'єднання для забезпечення якісної передачі даних (особливо для відеоконференцій). У районах з поганим Інтернет-з'єднанням користувачі можуть стикатися з перервами, низькою якістю відео та звуку або навіть неможливістю підключитися до платформи;

- більшість сучасних комунікаційних платформ працюють в хмарі, що означає, що доступ до даних і функціональності залежить від зовнішніх серверів. В разі збоїв або обмежень у хмарних провайдерів це може призвести до простоїв або втрати доступу до важливої інформації.

6. Проблеми з конфіденційністю та відповідністю стандартам:

- збір і обробка особистих даних користувачів (особливо в контексті відповідності таким нормативам, як GDPR) є важливою проблемою для платформ, які не завжди дотримуються вимог щодо захисту даних і приватності;

- багато платформ не надають чітких і зрозумілих умов використання та політик конфіденційності, що може створювати невизначеність у користувачів щодо того, як їхні дані обробляються і зберігаються.

7. Фінансові обмеження та доступність:

- деякі комунікаційні платформи (наприклад, Zoom, Microsoft Teams, Slack) вимагають платних підписок для доступу до повного функціоналу або для великих команд, що може бути дорогим для малих організацій або стартапів;

- безкоштовні версії платформ часто мають суттєві обмеження в функціональності (наприклад, обмеження на кількість учасників, тривалість зустрічей, можливості зберігання файлів тощо), що змушує користувачів переходити на платні тарифи.

8. Етичні та соціальні проблеми:

- платформи, особливо в умовах дистанційної роботи, можуть призвести до залежності від технологій, що знижує особисту взаємодію та створює бар'єри в комунікації між людьми;

- використання одних і тих самих платформ для роботи та особистих потреб може призвести до вигорання або труднощів у встановленні чітких меж між робочим та особистим часом.

Сучасні комунікаційні платформи є важливими інструментами для бізнесу, освіти та особистих комунікацій, але вони мають низку проблем і обмежень, які необхідно враховувати при їх виборі та використанні [33-36]. Від проблем з безпекою та конфіденційністю до обмежень масштабованості та зручності використання – ці фактори можуть значно впливати на ефективність і продуктивність. Рішення цих проблем вимагають постійного оновлення технологій, покращення інтерфейсів, дотримання стандартів безпеки та адаптації до нових вимог користувачів.

Висновки до розділу 1

1. Сучасні підходи для підтримки комунікаційної діяльності спрямовані на використання новітніх технологій для покращення ефективності обміну інформацією в організаціях та з аудиторією. Ключовими напрямками є цифрові комунікації, персоналізація обміну інформацією за допомогою аналітики та

штучного інтелекту, а також інтеграція різних каналів комунікації для створення цілісного досвіду. Крім того, важливими аспектами є залучення аудиторії через інтерактивність та гейміфікацію, автоматизація процесів і застосування чат-ботів, а також дотримання прозорості та етики в комунікації. Глобалізація та багатомовність є важливими для ефективної взаємодії з міжнародною аудиторією, а інклюзивність та доступність забезпечують рівний доступ до інформації для всіх користувачів. Аналіз даних і зворотний зв'язок сприяють постійному вдосконаленню комунікаційних стратегій. Всі ці елементи формують комплексний підхід до сучасної комунікаційної діяльності, орієнтуючись на максимальну ефективність і адаптацію до змінюваних умов зовнішнього середовища.

2. Програмне забезпечення для комунікацій є важливим інструментом у різних сферах діяльності, забезпечуючи ефективну взаємодію між людьми та організаціями. В залежності від потреб конкретної сфери, існують різноманітні інструменти, що дозволяють оптимізувати робочі процеси. У корпоративних комунікаціях популярні платформи для командної роботи та управління проектами, такі як Slack, Microsoft Teams та Asana. Для маркетингових комунікацій використовуються платформи для автоматизації email-розсилок і соціальних мереж, такі як Mailchimp та Hootsuite. В освітньому секторі інструменти, як Zoom та Google Classroom, сприяють ефективній взаємодії між викладачами та студентами. Програмне забезпечення для відеоконференцій, включаючи Skype та Google Meet, є незамінним для бізнесу, освіти та медицини.

Правильний вибір програмного забезпечення дозволяє організаціям значно покращити внутрішню та зовнішню комунікацію, підвищити ефективність роботи та забезпечити зручність для користувачів. З огляду на різноманіття інструментів, кожна організація має можливість підібрати найбільш відповідні рішення для своїх специфічних потреб.

3. Розробка комунікаційних платформ вимагає застосування широкого спектра сучасних технологій та інструментів, які забезпечують ефективність, безпеку та масштабованість. Важливими етапами є розробка зручних користувацьких інтерфейсів, використання потужних серверних технологій для обробки запитів та зберігання даних, а також реалізація віртуальних чатів, голосових та відео комунікацій у реальному часі. Для масштабованих і швидких додатків важливо використовувати технології для обробки великих навантажень, такі як Kubernetes і Docker.

Забезпечення безпеки також є критичним аспектом, де використовуються передові технології шифрування та автентифікації. Правильний вибір інструментів для тестування, моніторингу та обробки даних дозволяє підтримувати високу продуктивність і стабільну роботу платформи. У підсумку, для розробки ефективної комунікаційної платформи необхідно обирати інструменти, які відповідають специфічним вимогам користувачів, забезпечують безпеку і масштабованість, а також інтегруються в єдину ефективну систему.

4. У порівняльному аналізі існуючих рішень для комунікаційних платформ було розглянуто різні інструменти, що використовуються для внутрішніх та зовнішніх комунікацій, організації співпраці в командах, проведення відеоконференцій, а також управління проєктами. Кожна з платформ має свої сильні сторони та обмеження, що визначає їх придатність для різних сценаріїв використання:

- Slack та Microsoft Teams є двома найбільш популярними платформами для бізнес-комунікацій, де Slack виділяється своєю простотою і великим числом інтеграцій з іншими інструментами, а Microsoft Teams має перевагу в контексті організацій, що активно використовують Microsoft 365 і потребують складних можливостей для управління зустрічами та безпеки;

- Zoom і Google Meet – це основні інструменти для проведення відеоконференцій. Zoom забезпечує високу якість відео та підтримує великі аудиторії, що робить його оптимальним вибором для великих вебінарів і зустрічей. Водночас Google Meet пропонує простоту в інтеграції з іншими продуктами Google, є безкоштовним і підходить для малих і середніх груп;

- У сегменті мобільних месенджерів WhatsApp та Telegram також мають значну популярність. WhatsApp відзначається своєю простотою і безпекою, але обмеженням на кількість учасників групи. Telegram, з іншого боку, пропонує безліч функцій для автоматизації та підтримує великі групи, однак має обмежені можливості для відеоконференцій;

- У категорії управління проєктами Trello і Asana пропонують різні підходи до організації роботи. Trello підходить для невеликих команд завдяки своєму простому інтерфейсу і візуальному підходу до управління завданнями, тоді як Asana забезпечує більшу гнучкість і можливості для складних проектів, але має вищий рівень складності.

5. Сучасні комунікаційні платформи, незважаючи на значний розвиток і потенціал, стикаються з низкою проблем та обмежень, які можуть знижувати їх ефективність і зручність використання. Основними проблемами є: безпека та конфіденційність; масштабованість і продуктивність; інтероперабельність і інтеграція; зручність та доступність; залежність від Інтернет-з’єднання; конфіденційність та відповідність стандартам; фінансові обмеження; етичні та соціальні проблеми. Таким чином, при виборі та використанні комунікаційних платформ важливо враховувати ці проблеми та обмеження, а також сприяти постійному оновленню технологій, вдосконаленню інтерфейсів і забезпеченню безпеки для досягнення максимальної ефективності в роботі.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ КОМУНІКАЦІЙНОЇ ДІЯЛЬНОСТІ

2.1. Розробка архітектури програмного забезпечення для підтримки комунікаційної діяльності

Архітектура програмного забезпечення є основою для його подальшого розвитку, підтримки та масштабування. Розробка архітектури для системи, що підтримує комунікаційну діяльність, включає в себе визначення основних компонентів, їх взаємодії, а також вибір технологій, які забезпечують ефективність, масштабованість та безпеку системи [37-41]. Розглянемо основні етапи та принципи розробки архітектури:

1. Основні вимоги до архітектури комунікаційної платформи. При розробці архітектури комунікаційного програмного забезпечення потрібно враховувати наступні вимоги:

- масштабованість (система повинна бути здатна підтримувати збільшення кількості користувачів та обсягів даних, не знижуючи продуктивності);
- продуктивність (забезпечення швидкої обробки даних і реального часу обміну повідомленнями, відео- та голосовими дзвінками);
- надійність і доступність (система повинна працювати 24/7 з високим рівнем доступності і без збоїв);
- безпека (шифрування даних, аутентифікація користувачів, контроль доступу та захист від атак);
- інтероперабельність (платформа повинна забезпечувати інтеграцію з іншими сервісами та системами (CRM, файлові сховища, зовнішні API);

2. Архітектурні патерни та підходи. Залежно від вимог та складності системи, можна обрати кілька архітектурних патернів для комунікаційної платформи:

2.1. Клієнт-серверна архітектура. Основна архітектура для більшості сучасних програм. Вона включає дві основні частини: клієнт, що взаємодіє з користувачем (інтерфейс), і сервер, що обробляє запити та зберігає дані.

Переваги: простота у впровадженні та керуванні; легкість у масштабуванні (додавання нових серверів для обробки запитів); просте управління доступом до даних.

Недоліки: може виникнути навантаження на сервери при великій кількості одночасних підключень; висока залежність від надійності сервера.

2.2. Мікросервісна архітектура. Мікросервіси – це архітектурний стиль, при якому система складається з незалежних малих сервісів, кожен з яких виконує певну бізнес-логіку.

Переваги: гнучкість і масштабованість: кожен мікросервіс можна масштабувати незалежно; легкість у підтримці і розширенні: кожен сервіс можна змінювати без впливу на інші; кожен мікросервіс може бути реалізований за допомогою різних технологій, залежно від вимог.

Недоліки: складність в управлінні кількома мікросервісами, особливо для малих команд; потребує більше ресурсів для підтримки (наприклад, для оркестрації контейнерів, моніторингу і т.д.).

2.3. Архітектура на основі подій (Event-driven architecture). Архітектура, яка реагує на події (наприклад, нове повідомлення або підключення користувача) і запускає відповідні дії.

Переваги: підвищена продуктивність, оскільки система може реагувати на події в реальному часі; покращена інтеграція з іншими системами, оскільки

події можуть бути оброблені іншими мікросервісами; може забезпечити високий рівень масштабованості та відмовостійкості.

Недоліки: може бути складною в реалізації та відладці; потрібен чіткий контроль за порядком подій.

3. Компоненти архітектури комунікаційної платформи

3.1. Інтерфейс користувача (UI). Компонент, через який користувачі взаємодіють з системою. Це можуть бути веб- або мобільні додатки для спілкування через чати, відеоконференції, обмін файлами. Технології: React, Angular, Vue.js для веб-додатків; Swift, Kotlin для мобільних додатків.

3.2. Обробка запитів (API). Система повинна мати API, яке дозволяє клієнтам взаємодіяти з сервером. API є основою для обміну даними між сервером і клієнтом, а також для інтеграції з іншими платформами. Технології: RESTful API, GraphQL для гнучких запитів.

3.3. Модуль зберігання даних. База даних для зберігання всіх користувацьких даних (повідомлення, файли, історія дзвінків, профілі користувачів).Технології: реляційні бази даних (PostgreSQL, MySQL) для структурованих даних, нереляційні (MongoDB) для більш гнучкої роботи з даними, що часто змінюються.

3.4. Модуль комунікацій (чати, дзвінки). Модуль для обробки текстових і мультимедійних повідомлень, голосових і відео дзвінків. Технології: WebRTC для відео- та голосових дзвінків, Socket.IO для реального часу повідомлень.

3.5. Модуль безпеки. Компонент, що відповідає за аутентифікацію та авторизацію користувачів, шифрування даних і захист від несанкціонованого доступу. Технології: OAuth 2.0 для аутентифікації, JWT (JSON Web Tokens) для управління сесіями, TLS/SSL для захищених з’єднань.

3.6. Модуль моніторингу та аналітики. Модуль для збору метрик системи, моніторингу роботи компонентів, збирання логів, а також аналітики для

виявлення аномалій, покращення якості обслуговування. Технології: Prometheus, Grafana для моніторингу, ELK Stack для збору та аналізу логів.

4. Технології для побудови архітектури. Для зручного розгортання та масштабування використовуються технології Docker (контейнеризація) і Kubernetes (оркетстрація). Використання хмарних сервісів (AWS, Google Cloud, Azure) дозволяє масштабувати інфраструктуру та забезпечити надійність. Для створення API можуть використовуватися фреймворки на базі Node.js (Express), Python (Django, Flask), Ruby on Rails.

5. Принципи проєктування архітектури:

- модульність (кожен компонент системи повинен бути незалежним і мати чітко визначену роль);

- гнучкість (архітектура повинна дозволяти легко вносити зміни, оновлювати та розширювати систему);

- розподіленість (для забезпечення масштабованості й доступності слід використовувати розподілені системи);

- висока доступність та відмовостійкість (важливо забезпечити роботу системи навіть у разі збоїв одного з компонентів).

Розробка архітектури програмного забезпечення для підтримки комунікаційної діяльності є ключовим етапом у створенні ефективної, масштабованої та надійної платформи для обміну інформацією. Врахування вимог щодо масштабованості, продуктивності, безпеки та інтеграції з іншими системами дозволяє створити систему, яка здатна ефективно задовольняти потреби користувачів у реальному часі та при великих навантаженнях.

Проаналізувавши основні архітектурні підходи, зокрема клієнт-серверну архітектуру, мікросервісну архітектуру та архітектуру на основі подій, кожна з яких має свої переваги і недоліки для розробки ПЗ було обрано клієнт-серверну архітектуру.

Таким чином, розробка архітектури є основою для створення програмного забезпечення, що відповідає сучасним вимогам щодо комунікаційної діяльності, забезпечуючи високий рівень користувацького досвіду та стабільність роботи на всіх етапах експлуатації системи.

2.2. Опис компонентів та інструментів програмного рішення для підтримки комунікаційної діяльності

Програмне рішення для підтримки комунікаційної діяльності повинно бути побудоване з кількох основних компонентів, кожен з яких відповідає за конкретну функцію або сервіс у системі. Для ефективної розробки та функціонування такої платформи важливо застосовувати різні інструменти та технології, що забезпечують зручність для користувачів, продуктивність системи, безпеку та інтеграцію з іншими сервісами. Нижче наведено основні компоненти та інструменти програмного рішення.

1. Інтерфейс користувача (UI) [42-44]. Інтерфейс користувача є основним компонентом для взаємодії користувачів з платформою. Він може бути представленим як веб- або мобільний додаток, через який користувачі можуть надсилати повідомлення, здійснювати дзвінки, обмінюватися файлами тощо.

Інструменти:

- React / Angular / Vue.js (для веб-додатків) – фреймворки для створення динамічних і інтерактивних веб-інтерфейсів;

- Swift (для iOS) / Kotlin (для Android) – мови програмування для створення нативних мобільних додатків;

- Flutter – кросплатформенний фреймворк для створення мобільних додатків на базі єдиного коду для Android і iOS.

2. API для взаємодії між клієнтом і сервером. API дозволяє забезпечити взаємодію між клієнтською частиною (інтерфейс користувача) і сервером, а також підтримує інтеграцію з іншими сервісами.

Інструменти:

- RESTful API / GraphQL – стандарти для побудови інтерфейсів для взаємодії між клієнтом і сервером. REST використовує стандартні HTTP методи, в той час як GraphQL дозволяє клієнтам запитувати лише необхідні дані;

- OAuth 2.0 – протокол для забезпечення аутентифікації і авторизації користувачів через API;

- JWT (JSON Web Tokens) – технологія для безпечної передачі даних між клієнтом і сервером у вигляді токенів.

3. Модуль обробки та зберігання даних. Модуль відповідає за зберігання всіх даних, включаючи повідомлення, історію дзвінків, профілі користувачів, медіафайли, та надає можливість швидкого доступу до них. Для цього використовуються різні бази даних, залежно від вимог.

Інструменти:

- PostgreSQL / MySQL – реляційні бази даних, що підходять для зберігання структурованих даних (наприклад, профілі користувачів, історія повідомлень);

- MongoDB – нереляційна база даних для гнучкого зберігання великих обсягів неструктурованих даних (наприклад, мультимедійні файли);

- Redis – система кешування для прискорення доступу до часто використовуваних даних (наприклад, сесії користувачів або популярні повідомлення).

4. Модуль комунікацій (чати, дзвінки, сповіщення). Цей компонент є центральним у платформі, оскільки він відповідає за обробку реального часу комунікацій (чати, відео- та голосові дзвінки), а також сповіщення користувачів.

Інструменти:

- WebRTC – технологія для забезпечення відео- і голосових дзвінків через браузери без потреби в додаткових плагінах;

- Socket.IO – бібліотека для реалізації двостороннього зв'язку між клієнтом і сервером, що дозволяє створювати чат-системи з обміном повідомленнями в реальному часі;

- Push Notifications – використання служб для надсилання пуш-повідомлень користувачам (наприклад, Firebase Cloud Messaging для мобільних додатків).

5. Модуль безпеки. Модуль безпеки забезпечує захист даних користувачів, а також забезпечує належний рівень аутентифікації, авторизації та захисту від атак.

Інструменти:

- SSL/TLS – протоколи для забезпечення безпечних з'єднань між клієнтами та серверами;

- OAuth 2.0 / OpenID Connect – протоколи для аутентифікації та авторизації користувачів через сторонні сервіси (наприклад, через Google, Facebook);

- End-to-End Encryption – технології для шифрування повідомлень та медіафайлів, що передаються між користувачами (наприклад, AES або RSA для забезпечення конфіденційності даних).

6. Модуль аналітики та моніторингу. Модуль, який дозволяє моніторити стан системи, відслідковувати її продуктивність, збори логів та даних для аналітики використання платформи.

Інструменти:

- Prometheus – система для збору метрик та моніторингу продуктивності сервісів у реальному часі;

- Grafana – інструмент для візуалізації метрик, що дозволяє відображати дані з Prometheus і здійснювати моніторинг в зручному вигляді;

- ELK Stack (Elasticsearch, Logstash, Kibana) – система для збору, аналізу та візуалізації логів із різних компонентів системи.

7. Модуль інтеграції з зовнішніми сервісами. Цей компонент дозволяє інтегрувати платформу з іншими зовнішніми сервісами, такими як CRM-системи, хмарні сховища файлів, платформи для планування та управління проектами.

Інструменти:

- Zapier / Integromat – платформи для автоматизації робочих процесів і інтеграції з іншими додатками без необхідності програмування;

- API Integration – розробка та використання API для інтеграції з зовнішніми системами, такими як Salesforce (для CRM), Google Drive, Dropbox (для зберігання файлів);

8. Інструменти для тестування та забезпечення якості. Для забезпечення надійності, стабільності та безпеки програмного рішення важливо проводити тестування на різних етапах розробки.

Інструменти:

- Jest / Mocha – фреймворки для тестування JavaScript-коду (особливо корисні для тестування фронтенду);

- Postman – інструмент для тестування API, що дозволяє легко перевіряти та документувати кінцеві точки API;

- Selenium – інструмент для автоматизованого тестування веб-додатків (тестування користувацького інтерфейсу).

Розробка програмного рішення для підтримки комунікаційної діяльності вимагає комплексного підходу до вибору компонентів та інструментів, що дозволяють забезпечити ефективну, безпечну та масштабовану платформу. Вибір правильних технологій для обробки запитів, зберігання даних, інтеграції з іншими сервісами та забезпечення високої безпеки є критичним для успішної

реалізації проекту. Всі ці компоненти повинні працювати в тісній взаємодії для досягнення високого рівня користувацького досвіду та стабільної роботи системи.

2.3. Використання баз даних та сервісів для обробки комунікацій

Використання баз даних [45-46] та сервісів для обробки комунікацій є важливим етапом у створенні програмних рішень для підтримки комунікаційної діяльності. Оскільки комунікації часто вимагають обробки великих обсягів даних у реальному часі (повідомлення, дзвінки, файли), вибір правильних баз даних та сервісів для зберігання та обробки таких даних є критично важливим для ефективності, продуктивності та надійності системи.

1. Типи баз даних для обробки комунікацій

Для зберігання різних типів даних, пов'язаних з комунікаціями, використовуються різні види баз даних в залежності від специфіки запитів та обробки інформації.

1.1. Реляційні бази даних (SQL). Реляційні бази даних є традиційним вибором для зберігання структурованих даних. Вони використовуються для зберігання даних, які мають чітку структуру (наприклад, профілі користувачів, історія чатів, логін/пароль). Приклад: PostgreSQL, MySQL, Microsoft SQL Server.

Переваги: потужні можливості для виконання складних запитів і аналізу даних; забезпечують цілісність даних (ACID-принципи); легко забезпечити зв’язки між таблицями (наприклад, з’єднання між користувачами та повідомленнями).

Недоліки: можуть бути менш ефективними для обробки великих обсягів неструктурованих даних (наприклад, медіафайлів).

1.2. Нереляційні бази даних (NoSQL). Нереляційні бази даних використовуються для зберігання неструктурованих або частково структурованих даних, що змінюються часто (наприклад, повідомлення в чатах, медіафайли). Приклад: MongoDB, Cassandra, CouchDB.

Переваги: гнучкість у зберіганні різноманітних даних; легко масштабується для великих обсягів даних; підходять для зберігання даних з високою швидкістю запису.

Недоліки: відсутність складних запитів або транзакційної підтримки; можуть бути менш оптимізованими для певних типів операцій.

1.3. Бази даних для зберігання медіафайлів (Blob Storage). Для зберігання медіафайлів (фото, відео, аудіо) часто використовуються спеціалізовані системи для зберігання великих обсягів бінарних даних. Приклад: Amazon S3, Google Cloud Storage, Azure Blob Storage.

Переваги: висока доступність і безпека; масштабованість для обробки великих обсягів даних; легко інтегруються з іншими сервісами (наприклад, для трансляції відео).

Недоліки: потрібні додаткові інструменти для обробки метаданих файлів (наприклад, для пошуку за тегами або іншими атрибутами).

2. Сервіси для обробки та зберігання комунікаційних даних.

2.1. Сервіси для реального часу (Realtime Messaging). Сервіси для реального часу забезпечують миттєву доставку повідомлень і даних між користувачами. Вони використовуються для чатів, онлайн-дзвінків і повідомлень в реальному часі. Приклад:

- Firebase Realtime Database – платформа Google для зберігання даних у реальному часі. Призначена для масштабованих чатів та інших реальних часів комунікацій;

- Pusher – інструмент для обміну повідомленнями та обробки подій у реальному часі;

Socket.IO – бібліотека для реалізації двостороннього зв'язку між клієнтом і сервером через вебсокети, що дозволяє створювати чати і месенджери в реальному часі.

Переваги: підтримка масштабованості для великих кількостей користувачів; висока продуктивність для обробки миттєвих повідомлень; легке інтегрування з іншими веб та мобільними платформами.

2.2. Сервіси для голосових і відео дзвінків. Для обробки голосових і відео дзвінків використовуються спеціалізовані сервіси, що забезпечують підтримку різних платформ і стандартів зв’язку. Приклад:

- WebRTC – технологія для безпечного й ефективного забезпечення відео- та аудіоконференцій у реальному часі без додаткових плагінів;

- Twilio – платформа для інтеграції голосових, SMS і відео дзвінків у веб- та мобільні додатки;

- Agora – платформа для голосових і відео дзвінків з підтримкою масштабування та високої якості.

Переваги: проста інтеграція з різними веб та мобільними додатками; підтримка високої якості звуку і зображення; підтримка реального часу та масштабованості.

2.3. Модулі для обробки та зберігання медіафайлів. Для зберігання і трансляції медіафайлів, таких як зображення, відео та аудіо, можна використовувати сервіси для зберігання даних у хмарі. Приклад:

- Amazon S3 – хмарне сховище для зберігання та обробки медіафайлів;

- Cloudinary – платформа для обробки зображень і відео в реальному часі з можливістю масштабування;

- Firebase Storage – сховище для зберігання файлів із високою інтеграцією з іншими сервісами Firebase.

Переваги: легке масштабування для великих обсягів медіафайлів; підтримка трансляції та перетворення медіа; підвищена доступність і безпека файлів.

3. Інструменти для безпеки даних у комунікаційних системах

3.1. Шифрування даних. Для забезпечення конфіденційності комунікацій необхідно застосовувати шифрування на різних рівнях:

- End-to-End Encryption (E2EE) – технологія шифрування, яка гарантує, що лише учасники комунікації можуть прочитати повідомлення, навіть якщо дані передаються через посередників;

- TLS (Transport Layer Security) – протокол шифрування для захищеного обміну даними між серверами і клієнтами.

3.2. Аутентифікація та авторизація. Для забезпечення безпеки доступу до комунікацій та даних важливо використовувати сучасні механізми аутентифікації та авторизації:

- OAuth 2.0 – стандарт для авторизації доступу до даних на сторонніх платформах;

- JWT (JSON Web Tokens) використовуються для безпечної передачі токенів аутентифікації між користувачами та сервісами;

- Multi-factor Authentication (MFA) – додатковий рівень безпеки для аутентифікації користувачів.

2.4. Інтерфейс користувача та дизайн для програмного забезпечення підтримки комунікаційної діяльності

Інтерфейс користувача (UI) та дизайн – це ключові аспекти створення програмного забезпечення для підтримки комунікацій, оскільки саме через

інтерфейс користувач взаємодіє з платформою. Хороший інтерфейс має бути інтуїтивно зрозумілим, привабливим, функціональним та зручним у використанні, щоб забезпечити ефективну комунікацію та мінімізувати складність для кінцевого користувача.

1. Основні принципи дизайну інтерфейсу користувача:

- інтуїтивність (інтерфейс повинен бути таким, щоб користувач міг зрозуміти, як виконати необхідну операцію без додаткових пояснень чи навчання. Ключові функції повинні бути доступні на першому екрані, а навігація повинна бути простою та очевидною);

- простота (Дизайн повинен бути мінімалістичним, без зайвих елементів, що можуть відволікати користувача. Важливо, щоб кожна кнопка, значок або елемент інтерфейсу мав чітку мету і не був зайвим);

- універсальність (інтерфейс має працювати на різних пристроях: мобільних телефонах, планшетах, комп'ютерах. Це досягається через використання респонсивного дизайну, який автоматично адаптується до різних розмірів екранів).

- візуальна привабливість (гармонійне використання кольорів, шрифтів і зображень, що підвищує привабливість і зручність взаємодії. Важливо використовувати контрастні кольори для забезпечення читаємості, а також продумати колірну палітру, що відповідає бренду або загальній темі програми);

- ефективність (інтерфейс має забезпечувати швидку і ефективну взаємодію з користувачем. Наприклад, повідомлення повинні надходити без затримок, а дзвінки мають бути з можливістю швидкого підключення чи відключення).

2. Основні елементи інтерфейсу користувача:

- головна панель навігації (головна панель повинна містити доступ до основних функцій платформи, таких як чат, дзвінки, повідомлення,

налаштування профілю. Вона повинна бути простою та зручною для доступа з будь-якої частини програми);

- чат та повідомлення (інтерфейс для обміну повідомленнями повинен дозволяти користувачам швидко надсилати та отримувати текстові повідомлення, медіафайли, голосові повідомлення тощо. Повідомлення повинні бути відображені у хронологічному порядку, з можливістю їх пошуку і фільтрації);

- іконки та кнопки (іконки повинні бути зрозумілими та однозначними, наприклад, іконка дзвінка для голосових чи відео дзвінків, іконка листа для текстових повідомлень. Всі кнопки повинні бути чітко виділені та легко помітні);

- історія чатів і дзвінків (інтерфейс має надавати можливість зручно переглядати історію чатів та дзвінків, шукати повідомлення за датами чи ключовими словами, а також надавати можливість видаляти або архівувати старі записи);

- система повідомлень (повідомлення в реальному часі повинні відображатися швидко та коректно. Важливо інтегрувати систему сповіщень про нові повідомлення, нові дзвінки або події в системі);

- профіль користувача (користувач має можливість редагувати свій профіль, додавати фото, змінювати налаштування безпеки та конфіденційності. Зручний доступ до налаштувань дозволяє змінювати особисту інформацію, паролі, мовні налаштування та інше).

3. Особливості дизайну для різних платформ:

- мобільні додатки (для мобільних пристроїв дизайн має бути адаптований до невеликих екранів. Респонсивний дизайн дозволяє інтерфейсу змінюватися в залежності від розміру екрану, а також використовувати специфічні можливості мобільних пристроїв, такі як сенсорний екран і жестові команди);

- веб-додатки (веб-інтерфейси зазвичай мають більше простору для розміщення елементів і забезпечують більшу гнучкість у використанні складніших функцій. Веб-версії дозволяють зберігати більше інформації на екрані, тому важливо не перевантажувати інтерфейс і дбати про зручну навігацію);

- кросплатформені рішення (якщо програма має підтримувати декілька платформ (наприклад, iOS, Android і веб), важливо створити дизайн, який буде однаково зручним на всіх пристроях. Використовуються кросплатформенні фреймворки, такі як Flutter, React Native, які дозволяють створювати єдину кодову базу для різних платформ).

4. Інструменти для створення UI та UX дизайну:

- Figma – один з найпопулярніших інструментів для створення інтерфейсів і прототипів, який дозволяє працювати в команді в реальному часі. Підтримує інтеграцію з іншими сервісами для тестування дизайну;

- Sketch – потужний інструмент для дизайна інтерфейсів, зокрема для macOS, який популярний серед дизайнерів для створення веб- і мобільних інтерфейсів;

- Adobe XD – інструмент для створення інтерактивних прототипів, що дозволяє проектувати UI для мобільних додатків та веб-сайтів з можливістю попереднього тестування інтерфейсу;

- InVision – інструмент для створення прототипів і анімацій інтерфейсів. Підходить для інтеграції з іншими платформами для роботи над дизайном;

- Balsamiq – інструмент для швидкого створення каркасних моделей інтерфейсів (wireframes), який дає можливість швидко візуалізувати концепцію інтерфейсу.

Розробка інтерфейсу користувача для програмного забезпечення підтримки комунікаційної діяльності вимагає комплексного підходу, де важливими є не

лише естетичні якості, але й функціональність та зручність для користувача. Хороший дизайн має враховувати потреби користувачів, оптимізувати їх досвід взаємодії з платформою та забезпечити ефективну реалізацію всіх необхідних функцій.

Висновки до розділу 2

1. Розробка архітектури є основою для створення програмного забезпечення, що відповідає сучасним вимогам щодо комунікаційної діяльності, забезпечуючи високий рівень користувацького досвіду та стабільність роботи на всіх етапах експлуатації системи. Проаналізувавши основні архітектурні підходи, зокрема клієнт-серверну архітектуру, мікросервісну архітектуру та архітектуру на основі подій, кожна з яких має свої переваги і недоліки для розробки ПЗ було обрано клієнт-серверну архітектуру.

2. Розробка програмного рішення для підтримки комунікаційної діяльності є складним і багатокомпонентним процесом, який вимагає інтеграції різноманітних інструментів та технологій для забезпечення ефективності, безпеки та стабільності платформи. Кожен компонент, від інтерфейсу користувача до модулів безпеки та аналітики, має важливу роль у створенні комплексної та зручної системи. Вибір правильних технологій для обробки запитів, зберігання даних, забезпечення реального часу взаємодії та інтеграції з іншими сервісами є критичним для успішної реалізації комунікаційної платформи.

3. Використання баз даних та сервісів для обробки комунікацій є критично важливим етапом у створенні ефективних програмних рішень для комунікаційної діяльності. Вибір відповідних технологій для зберігання та обробки даних залежить від типу інформації, що обробляється, та вимог до системи. Збалансований підхід до вибору баз даних, сервісів для обробки даних

і механізмів безпеки дозволяє створити надійну, масштабовану та безпечну платформу для підтримки комунікаційної діяльності.

4. Розробка інтерфейсу користувача та дизайну для програмного забезпечення, що підтримує комунікаційну діяльність, є важливою складовою успіху будь-якої платформи. Інтерфейс повинен бути інтуїтивно зрозумілим, простим, універсальним і привабливим, щоб забезпечити зручну взаємодію та максимізувати ефективність комунікацій. Важливими аспектами є адаптивність дизайну для різних платформ, таких як мобільні додатки, веб-додатки та кросплатформенні рішення. Використання сучасних інструментів для створення та тестування UI/UX, таких як Figma, Adobe XD та Sketch, дозволяє створювати зручні та привабливі інтерфейси.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ КОМУНІКАЦІЙНОЇ ДІЯЛЬНОСТІ

3.1 Програмна реалізація проєкту

Windows – це операційна система, призначена для комунікаторів, планшетів, персональних комп'ютерів і ноутбуків [47]. Десктопні додатки для Windows часто розробляються на Delphi [48] з використанням середовища розробки Rad Studio. Скомпільований програмний код, разом з усіма необхідними ресурсами та файлами, упаковується в спеціальний файл з розширенням *.exe. Цей файл є основним для програми, і він поширюється та інсталюється на пристрої. Кожен такий файл відповідає за одну програму, яка за замовчуванням виконується в окремому процесі, який також займається керуванням пам'яттю.

Мова програмування Delphi є популярною в ряді мов програмування і була розроблена з метою забезпечення легкості освоєння та ефективного застосування програмістами. Її основою є принцип простоти й доступності при створенні додатків та їх прототипуванні. Розробники, спираючись на попередній досвід програмування, намагалися усунути нестабільні або складні для використання можливості. Крім того, наявність потужних інструментів для зручного застосування основ об'єктно-орієнтованого програмування та їх інтеграція в код значно спрощують процес написання програм. Наприклад, створити програму з графічним інтерфейсом у середовищі Rad Studio на Delphi не є складним завданням, завдяки великій кількості готових візуальних компонентів. А можливість створення крос-платформенних рішень дозволяє легко адаптувати програму до різних середовищ.

Для реалізації проєкту було обрано середовище розробки Rad Studio. Rad Studio – це потужне інтегроване середовище розробки (IDE), призначене для

платформи Windows. Воно забезпечує ефективне вирішення типових завдань, що виникають у процесі розробки додатків для цієї операційної системи. Серед його можливостей – інструменти для тестування сумісності програм з різними версіями Windows, а також функції для проектування застосунків, що адаптуються до пристроїв з екранами різної роздільної здатності (планшети, ноутбуки, смартфони тощо). Окрім стандартних можливостей, середовище має додаткові функції, які значно спрощують процес розробки, такі як тестування та розгортання програм на кількох операційних системах (MacOS, Linux, Windows, Android) при збереженні одного коду завдяки крос-платформенним рішенням. Також є можливість віддаленого режиму відладки, що дає змогу ефективно виправляти помилки без необхідності безпосередньо підключати пристрій до розробницького середовища. Це робить Rad Studio зручним інструментом для створення багатофункціональних і крос-платформених додатків.

Для прискорення створення UI-інтерфейсу ПЗ у Rad Studio надається колекція стандартних елементів інтерфейсу та візуальний редактор для їхнього компонування. Це дозволяє зручно переглядати різні варіанти інтерфейсу, наприклад, перевіряти, як він виглядатиме на екранах різних розмірів. Для створення інтерфейсів, що виходять за межі стандартних, є майстер для розробки індивідуальних елементів оформлення, який підтримує використання шаблонів.

До складу середовища входять також спеціалізовані інструменти для рефакторингу, оптимізовані під особливості платформи Windows, а також інструменти для виявлення проблем з продуктивністю, моніторингу використання пам’яті та оцінки зручності використання. Для зручності розробників реалізовано режим швидкого внесення правок. Система підсвічування, статичного аналізу та виявлення помилок була розширена підтримкою API, а також інтегровано оптимізатор коду, що допомагає

покращити ефективність програм. Це робить процес розробки більш зручним і швидким, зменшуючи кількість помилок і підвищуючи загальну продуктивність програм.

Все, що ми бачимо на екрані, візуалізується під час інтерпретації графічного компонування елементів на формі. Це процес, у якому всі елементи інтерфейсу, такі як кнопки, текстові поля, мітки та інші компоненти, розташовуються на вікні згідно з заданим макетом. Нижче наведено приклад коду для вікна авторизації в системі, що визначає основні елементи цього вікна. На рис. 3.1 представлено скріншот початкового вікна системи, де користувач може ввести свої облікові дані для входа.

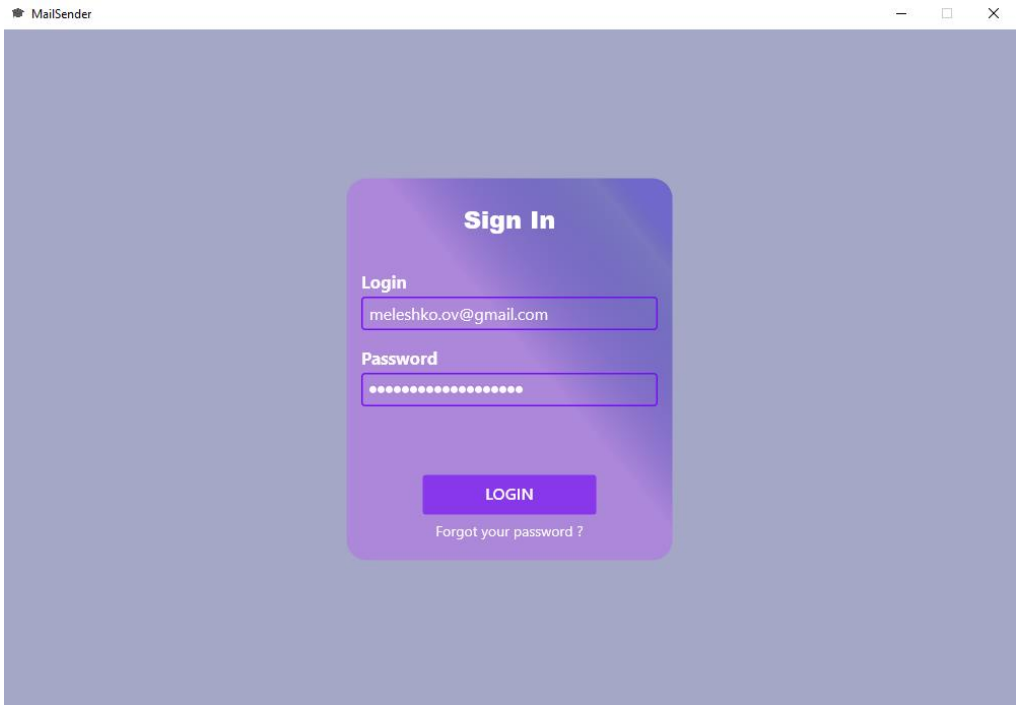

Рисунок 3.1 Вікно авторизації

Програмний код модуля UserAuth, що використовується для перевірки даних користувача при авторизації представлено в програмному лістингу 1.

Програмний лістинг 1. Фрагмент програмного коду, що використовується при авторизації користувача:

```
TUserAuth = class
```

```
 private
      FUsername: string;
      FPasswordHash: string;
      function EncryptPassword(const Password: string): string;
     function ValidatePassword(const Password: string): Boolean;
public
         constructor Create(const Username, Password: string);
         function Authenticate(const Username, Password: string): Boolean;
        property Username: string read FUsername;
         property PasswordHash: string read FPasswordHash;
       end;
implementation
      { TUserAuth }
      constructor TUserAuth.Create(const Username, Password: string);
      begin
        FUsername := Username;
        FPasswordHash := EncryptPassword(Password);
      end;
      function TUserAuth.EncryptPassword(const Password: string): string;
      var
        i: Integer;
        EncryptedPassword: string;
      begin
        EncryptedPassword := '';
      for i := 1 to Length(Password) do
      begin
         EncryptedPassword := EncryptedPassword + Chr(Ord(Password[i]) xor 255);  end;
        Result := EncryptedPassword;
      end;
      function TUserAuth.ValidatePassword(const Password: string): Boolean;
      begin
```

```
  Result := FPasswordHash = EncryptPassword(Password);
end;
function TUserAuth.Authenticate(const Username, Password: string): Boolean;
begin
   Result := (Username = FUsername) and ValidatePassword(Password);
end;
end.
```

На рис. 3.2 представлено скріншот на якому зображено вікно з доданими контактними даними про університети для листування.

Рисунок 3.2 Вікно зі списком університетів та їх контакними даними

Для додавання даних про назву університету та відділ листування, телефон і Email є окреме вікно, що представлене на рис. 3.3.

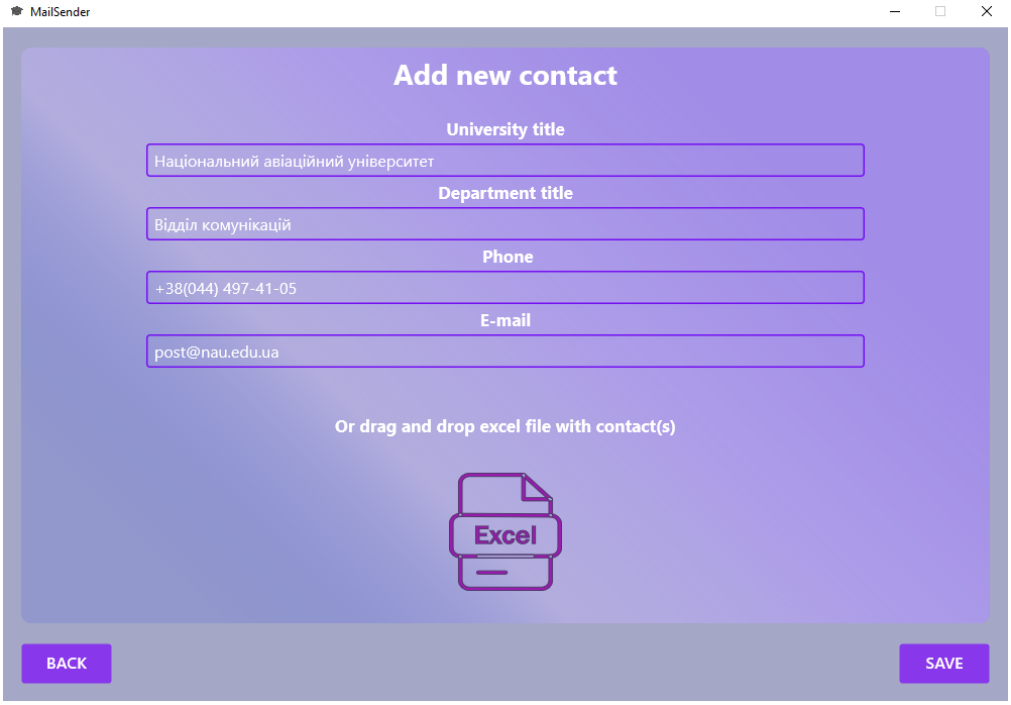

Рисунок 3.3 Вікно з полями для додавання контакних даних

Вікно містить форму для введення даних про університети. Користувач зможе вручну ввести інформацію в поля, а також перетягнути файл, що містить дані, для автоматичного додавання кількох університетів одночасно. Фрагмент коду, який працює з БД та дозволяє додавати університети як вручну через форму, так і через перетягування файлів (Drag and Drop) представлено в програмному лістингу 2.

Програмний лістинг 2. Фрагмент коду взаємодії з БД для додавання даних:

```
procedure TUniversityDatabase.Connect;
begin
  FConnection.DriverName := 'SQLite';
  FConnection.Params.Database := 'universities.db';
  FConnection.Connected := True;
end;

procedure TUniversityDatabase.AddUniversity(University: TUniversity);
```

```
begin
  FQuery.SQL.Text := 'INSERT INTO Universities (UniversityName, Department, Street, Phone, Email) ' + 'VALUES (:UniversityName, :Department, :Street, :Phone, :Email)';
 FQuery.ParamByName('UniversityName').AsString:=University.UniversityName;
  FQuery.ParamByName('Department').AsString := University.Department;
  FQuery.ParamByName('Street').AsString := University.Street;
  FQuery.ParamByName('Phone').AsString := University.Phone;
  FQuery.ParamByName('Email').AsString := University.Email;
  FQuery.ExecSQL;
end;

procedure TUniversityDatabase.DeleteUniversity(UniversityName: string);
begin
  FQuery.SQL.Text := 'DELETE FROM Universities WHERE UniversityName = :UniversityName';
  FQuery.ParamByName('UniversityName').AsString := UniversityName;
  FQuery.ExecSQL;
end;

procedure TUniversityDatabase.GetAllUniversities;
begin
  FQuery.SQL.Text := 'SELECT * FROM Universities';
  FQuery.Open;
end;
```

Вікно для автоматичної генерації звітів та збереження їх у файл представлено на рис. 3.4

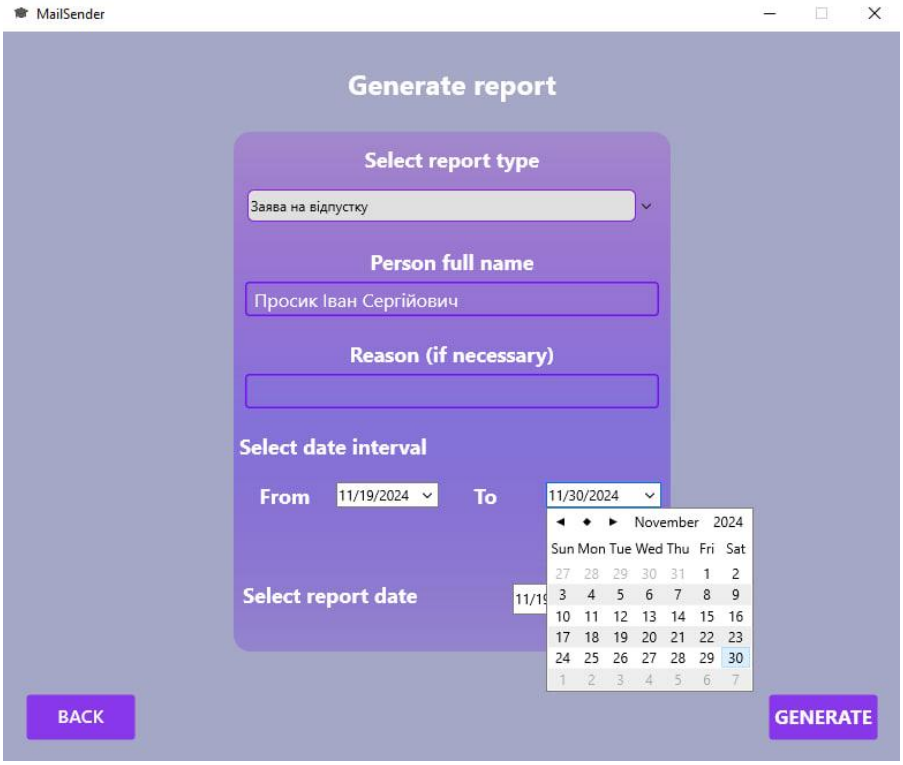

Рисунок 3.4 Вікно для автоматичної генерації звітів

Клас та сигнатура методів використаних у модулі LeaveRequestGenerator представлена в програмному лістингу 3.

Програмний лістинг 3. Клас та сигнатура методів модуля по генерації звітів:

```
TLeaveRequestGenerator = class
  private
    fFullName: String;
    fStartDate: String;
    fEndDate: String;
    fCurrentDate: String;
    fUniversityName: String;
    fRequestType: String;
  public
    constructor Create(const FullName, StartDate, EndDate, RectorName, UniversityName, RequestType: string);
```

```
    procedure GenerateLeaveStatement(const FileName: string);
    procedure GenerateSickLeaveStatement(const FileName: string);
    procedure GenerateTransferStatement(const FileName: string);

  procedure LoadDataFromGET(const URL: string);
    procedure LoadDataFromPOST(const URL: string; const PostData: string);
  procedure SaveStatementToFile(const FileName: string; const StatementText: string);
    property FullName: string read fFullName write fFullName;
    property StartDate: string read fStartDate write fStartDate;
    property EndDate: string read fEndDate write FEndDate;
    property RectorName: string read fRectorName write fRectorName;
    property UniversityName: string read fUniversityName write fUniversityName;
    property RequestType: string read fRequestType write fRequestType;
  end;
```

Для листування та масової розсилки повідомлень розроблено окреме вікно з полями для введення даних, а також функціональні кнопки для прикріплення, додавання та редагування файлів документів і представлено на рис. 3.5.

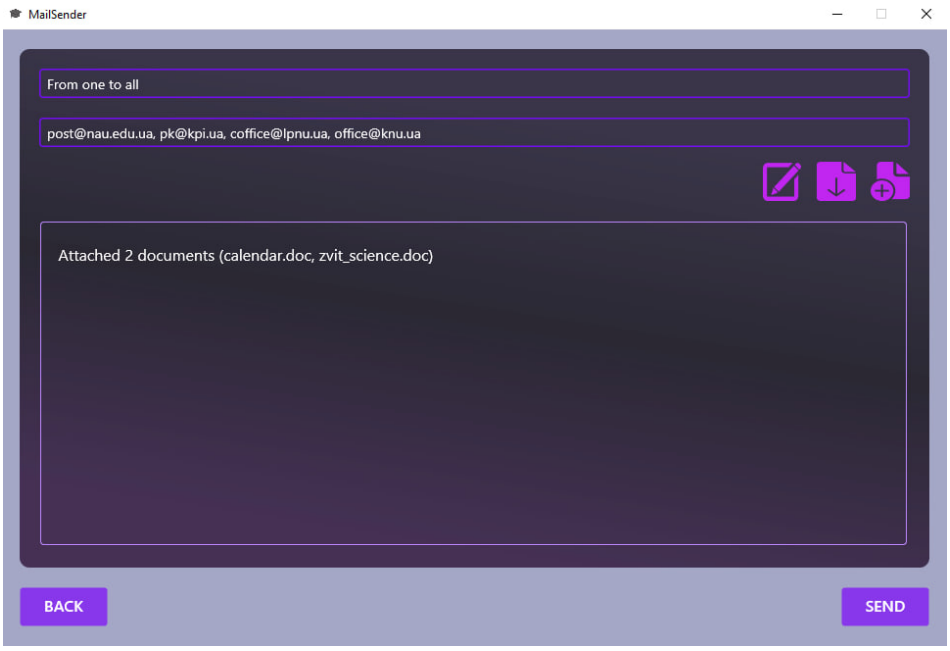

Рисунок 3.5 Вікно для масової розсилки повідомлень

Клас та сигнатура методів використаних у модулі LeaveRequestGenerator представлена в програмному лістингу 4.

Програмний лістинг 4. Клас та сигнатура методів модуля розсилки повідомлень:

```
TEmailSender = class
  private
    FSMTPServer: string;       // SMTP сервер
    FSMTPPort: Integer;        // Порт SMTP
    FSMTPUsername: string;     // Логін SMTP
    FSMTPPassword: string;     // Пароль SMTP
    FSenderEmail: string;      // Адреса відправника
    FSenderName: string;       // Ім'я відправника
    FSSLType: TIdSSLType;      // Тип SSL
  public
    constructor Create(const SMTPServer, SMTPUsername, SMTPPassword, SenderEmail,
SenderName: string; SMTPPort: Integer = 587; SSLType: TIdSSLType = sslTLS);
    destructor Destroy; override;
    function SendEmail(const Subject, Body, RecipientEmail: string; const    Attachments:
TArray<string> = nil): Boolean;
    function SendEmailWithFiles(const Subject, Body, RecipientEmail: string; const Attachments:
TArray<string>; const Files: TArray<string>): Boolean;
    property SMTPServer: string read FSMTPServer write FSMTPServer;
    property SMTPPort: Integer read FSMTPPort write FSMTPPort;
    property SMTPUsername: string read FSMTPUsername write FSMTPUsername;
    property SMTPPassword: string read FSMTPPassword write FSMTPPassword;
    property SenderEmail: string read FSenderEmail write FSenderEmail;
    property SenderName: string read FSenderName write FSenderName;
    property SSLType: TIdSSLType read FSSLType write FSSLType;
  end;
```

3.2 UML -діаграма класів

UML діаграма класів [49] із переліком основних атрибутів та методів розробленого програмного рішення представлена на рис. 3.6.

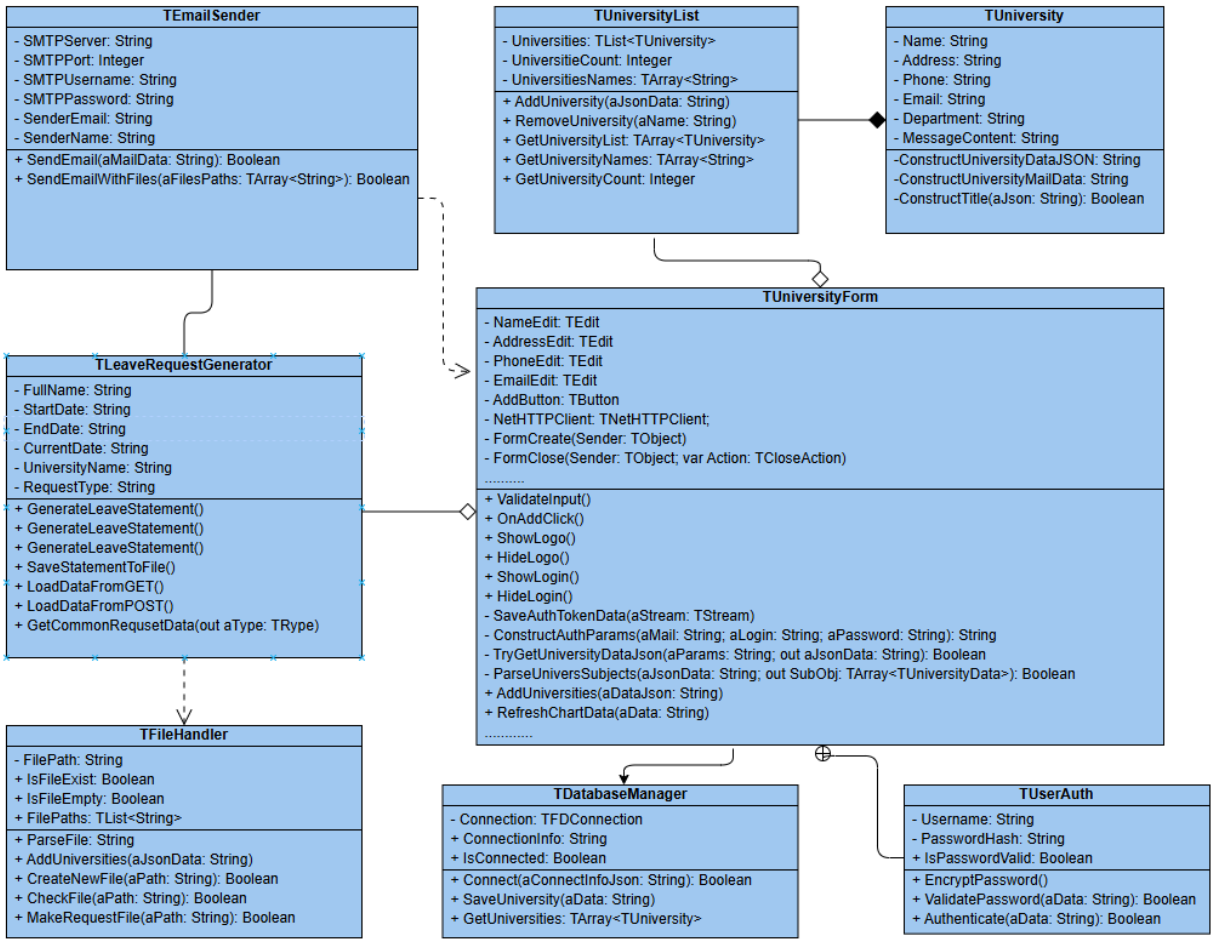

Рисунок 3.6 UML діаграма класів

UML діаграма класів, яка враховує наступні класи:

- TUniversity – опис університету;
- TFileHandler – обробка файлів через Drag-and-Drop;
- TDatabaseManager – взаємодія з базою даних;
- TUniversityForm – форма для введення даних про університети;
- TUniversityList – список університетів;

- TLeaveRequestGenerator – генерація заяв;
- TEmailSender – надсилання електронних листів;
- TUserAuth – авторизація користувача.

Пояснення до функціоналу основних класів та зв’язків між ними:

- TUniversity містить дані про університет (назва, адреса, телефон, email, відділ). Використовується в інших класах, таких як TUniversityForm та TDatabase Manager для введення та збереження даних в базі даних;

- TFileHandler відповідає за обробку файлів, зокрема через механізм Drag-and-Drop. Може зчитувати файли і додавати університети до бази даних через методи, такі як ParseFile() і AddUniversities();

- TDatabaseManager взаємодіє з базою даних для зберігання і витягування інформації про університети. Забезпечує операції для підключення до БД, збереження та отримання даних;

- TUniversityForm – форма для введення даних про університети, з полями для адреси, телефону, email та кнопкою для додавання нового університету. Має методи для валідації введених даних та обробки подій, таких як додавання університету та перетягування файлів;

- TUniversityList містить список університетів і дозволяє виконувати операції з групами університетів, включаючи додавання та видалення;

- TLeaveRequestGenerator – клас для генерації заяв на відпустку, лікарняний чи переведення. Має методи для створення різних типів заяв і збереження їх у файлах, а також для завантаження даних через GET і POST запити;

- TEmailSender відповідає за надсилання електронних листів з можливістю додавати вкладення. Може надсилати листи із файлами та іншими вкладеннями через SMTP;

- TUserAuth відповідає за авторизацію користувача в системі. Шифрує паролі за допомогою методу EncryptPassword() та перевіряє їх через ValidatePassword(). Може аутентифікувати користувача через метод Authenticate().

Діаграма класів добре відображає структуру програми і взаємодію між різними її частинами.

3.3 Програмна реалізація бази даних

Для реалізації модуля який працюватиме з базою даних (БД), та зможе додавати, видаляти і зчитувати дані, було використано компонент FireDAC. FireDAC – це набір компонентів для доступу до баз даних, який підтримує широкий спектр СУБД, таких як MySQL, PostgreSQL, SQLite, MSSQL та інші. Для зберігання інформації про університети та їх контактні дані було створено таблицю в базі даних. SQL запит для створеної таблиці наведений в програмному лістингу 5.

Програмний лістинг 5. SQL запит для таблиці з даними університетів:

```
CREATE TABLE Universities (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    UniversityName VARCHAR(255),
    Department VARCHAR(255),
    Street VARCHAR(255),
    Phone VARCHAR(50),
    Email VARCHAR(255)
);
```

Для реалізації програмної обгортки над БД [50] було написано модуль DatabaseModule, який надає простий інтерфейс для взаємодії з БД, зберіганням і зчитуванням даних. Програмна реалізація класу представлена в програмному лістингу 6.

Програмний лістинг 6. Програмна реалізація класу для роботи з БД та таблицею про університети

```
  TUniversity = class
  private
    FID: Integer;
    FUniversityName: string;
    FDepartment: string;
    FStreet: string;
    FPhone: string;
    FEmail: string;
  public
    property ID: Integer read FID write FID;
    property UniversityName: string read FUniversityName write FUniversityName;
    property Department: string read FDepartment write FDepartment;
    property Street: string read FStreet write FStreet;
    property Phone: string read FPhone write FPhone;
    property Email: string read FEmail write FEmail;
  end;
  TDatabaseModule = class(TDataModule)
    FDConnection: TFDConnection;
    FDQuery: TFDQuery;
  private
    function GetConnectionString: string;
  public
    constructor Create(AOwner: TComponent); override;
    function GetAllUniversities: TArray<TUniversity>;
    function AddUniversity(AUniversity: TUniversity): Boolean;
    function DeleteUniversity(AUniversityID: Integer): Boolean;
  end;
var DatabaseModule: TDatabaseModule;
implementation
```

```
constructor TDatabaseModule.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FDConnection.ConnectionString := GetConnectionString;
  FDConnection.Connected := True;
end;
function TDatabaseModule.GetConnectionString: string;
begin
  Result:='DriverID=MySQL;Server=localhost;Database=your_db_name;
User_Name=your_username;Password=your_password;';
end;
function TDatabaseModule.GetAllUniversities: TArray<TUniversity>;
begin
  FDQuery.SQL.Text := 'SELECT * FROM Universities';
  FDQuery.Open;
  SetLength(Result, FDQuery.RecordCount);
  FDQuery.First;
  for var i := 0 to FDQuery.RecordCount - 1 do
  begin
    Result[i] := TUniversity.Create;
    Result[i].ID := FDQuery.FieldByName('ID').AsInteger;
    Result[i].UniversityName := FDQuery.FieldByName('UniversityName').AsString;
    Result[i].Department := FDQuery.FieldByName('Department').AsString;
    Result[i].Street := FDQuery.FieldByName('Street').AsString;
    Result[i].Phone := FDQuery.FieldByName('Phone').AsString;
    Result[i].Email := FDQuery.FieldByName('Email').AsString;
    FDQuery.Next;
  end;
end;
function TDatabaseModule.AddUniversity(AUniversity: TUniversity): Boolean;
begin
```

```
  try
    FDQuery.SQL.Text := 'INSERT INTO Universities (UniversityName, Department, Street, Phone,
Email) ' + 'VALUES (:UniversityName, :Department, :Street, :Phone, :Email)';
    FDQuery.ParamByName('UniversityName').AsString := AUniversity.UniversityName;
    FDQuery.ParamByName('Department').AsString := AUniversity.Department;
    FDQuery.ParamByName('Street').AsString := AUniversity.Street;
    FDQuery.ParamByName('Phone').AsString := AUniversity.Phone;
    FDQuery.ParamByName('Email').AsString := AUniversity.Email;
    FDQuery.ExecSQL;
    Result := True;
  except
    on E: Exception do
    begin
      Result := False;
    end;
  end;
end;
function TDatabaseModule.DeleteUniversity(AUniversityID: Integer): Boolean;
begin
  try
    FDQuery.SQL.Text := 'DELETE FROM Universities WHERE ID = :ID';
    FDQuery.ParamByName('ID').AsInteger := AUniversityID;
    FDQuery.ExecSQL;
    Result := True;
  except
    on E: Exception do
    begin
      Result := False;
    end;
  end;
```

Клас TUniversity представляє об’єкт університета з його властивостями, такими як назва, відділ, адреса, телефон і електронна пошта. Основний клас модуля це TDatabaseModule, він містить методи для роботи з базою даних через компонент FireDAC.

3.3. Реалізація інтерфейсу користувача

ПЗ розроблене на базі Delphi та FireMonkey [51], інтерфейс користувача створюється з урахуванням специфіки багатоплатформенності та адаптивності. За допомогою FireMonkey можна створювати високоякісні графічні інтерфейси, які працюють на різних пристроях, таких як ПК, планшети та мобільні телефони. Для зручності користувача, інтерфейс організовано в кілька вікон і форм, що забезпечують максимальну функціональність і простоту у використанні.

Основною формою є форма для введення та обробки даних про університети. Вона містить необхідні поля вводу інформації, такі як назва університету, факультет, адреса, телефон, email тощо. Ці поля реалізовані за допомогою компонентів TEdit, що дозволяють користувачеві вводити текстову інформацію. Для зручності користувача форма також має відповідні мітки (TLabel), що пояснюють, яку саме інформацію потрібно ввести в кожне поле. Крім того, форма містить кнопку для додавання введених даних у базу, яка викликає необхідну логіку для збереження цієї інформації. Загальний вигляд структури вкладень Layout –ів інтерфейсу в MailSenderForm показаний на рис. 3.7.

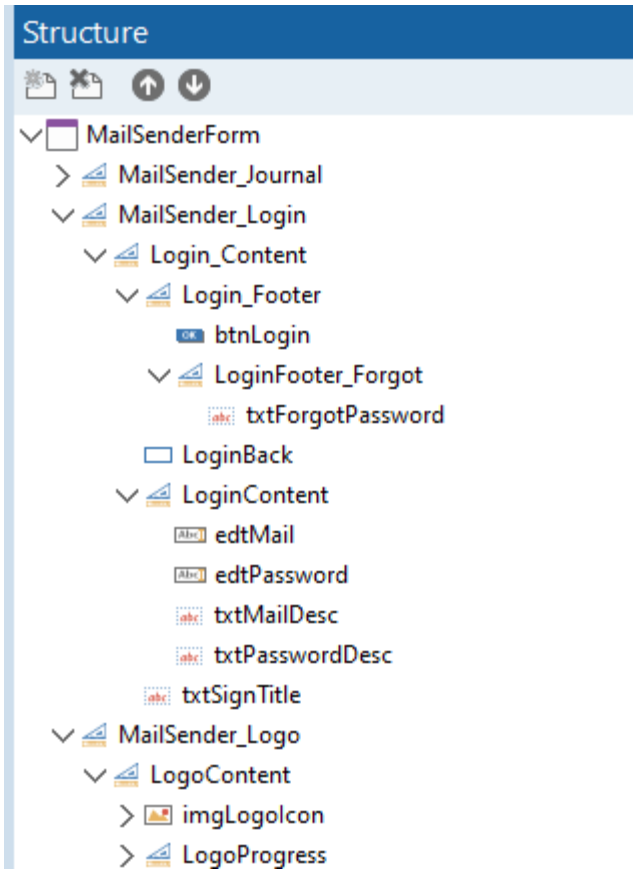

Рисунок 3.7 Загальний вигляд структури інтерфейсу в RadStudio

Для забезпечення можливості масового додавання університетів, реалізовано підтримку Drag-and-Drop. Користувач може перетягнути файл (наприклад, CSV або текстовий файл) у спеціально виділену область на формі, де система зчитує вміст файлу та додає університети з нього в базу даних. Область для перетягування файлів реалізована за допомогою компонента TPanel, в якому можна налаштувати обробку події Drag-and-Drop. Ця можливість спрощує процес додавання даних, дозволяючи автоматично зчитувати інформацію без необхідності вручну заповнювати поля. Щоб полегшити взаємодію користувача з інтерфейсом, форма має механізми валідації введених даних. Перед додаванням університету в базу даних система

перевіряє правильність введеної інформації, таких як формат email або телефонного номера. Якщо дані не відповідають вимогам, користувачу виводиться повідомлення про помилку за допомогою компонентів TMessageDialog або TLabel, що сповіщає про необхідність виправити помилки. Всі введені дані зберігаються в базі даних, яка може бути організована через локальне сховище або через підключення до серверної бази даних за допомогою компонентів FireDAC. Ці дані потім можна використовувати для подальшої обробки, пошуку або редагування.

Для зручності користувача, додаток надає можливість переглядати, редагувати чи видаляти записи, що знаходяться в базі даних. Крім того, форма має адаптивний дизайн, що дозволяє коректно відображати інтерфейс на різних пристроях. Завдяки можливостям FireMonkey, інтерфейс автоматично підлаштовується під розмір екрана, що особливо важливо для мобільних пристроїв та планшетів. Усі компоненти мають налаштування для забезпечення максимального комфорту користувача на різних платформах.

Нижче наведено приклад опису форми для введення та додавання даних про університети представленої вище на рис. 3.3.

Назва форми. Форма має назву TUniversityForm і є стандартною формою Delphi, з усіма необхідними компонентами для введення та управління даними.

Компоненти форми. Текстові поля вводу даних:

- TEdit для Назви університету (EditUniversityName) – користувач вводить назву університету, наприклад, "Київський університет".

- TEdit для Відділу або факультету (EditDepartment) – користувач вводить відділ або факультет, наприклад, "Факультет мехатроніки та компютерних технологій";

- TEdit для Телефону (EditPhone) – користувач вводить номер телефону університету, наприклад, "123-45-67".

- TEdit для Email (EditEmail) – користувач вводить email університету, наприклад, "example@university.com".

Кнопка для додавання університету. TButton для Додавання університету (BtnAddUniversity) – кнопка для додавання введених даних про університет у базу даних.

Компонент для перетягування файлів (Drag and Drop). TPanel або TImage для Області перетягування файлів (PanelDragDrop). Цей компонент слугує як зона для Drag and Drop, куди користувач може перетягнути файл, що містить дані про кілька університетів. Важливо, щоб ця область була візуально виділена, наприклад, змінивши фон на колір або додавши підпис "Перетягніть файл сюди".

Всі ці функціональні елементи взаємодіють між собою для забезпечення зручної роботи з даними про університети. Користувач має можливість як вручну вводити інформацію, так і масово додавати університети через перетягування файлів, що значно спрощує процес збору та обробки даних.

3.4 Тестування та перевірка функціональності програмного забезпечення

Невід’ємною частиною будь-якої реалізації програмного продукту є процес тестування [52-62], або перевірка роботи програми відповідно до вимог, зазначених на етапі початкової розробки. Під час тестування зазвичай виявляються дефекти, які можуть призвести до некоректної роботи системи, її зависання або блокування. У разі виявлення помилки розробник повинен її виправити.

Тестування можна поділити на два основних типи: ручне та автоматизоване. Під час ручного тестування всі операції виконуються вручну,

що дозволяє перевірити, як працює та чи інша функція додатку. Це дозволяє оцінити правильність реалізації функціоналу та перевірити, чи повертаються правильні результати. Кожен з цих видів тестування має свої підкатегорії.

Було проведено такі види тестування системи:

1. Інсталяційне тестування. Під час цього тестування додаток було успішно встановлено на ноутбуки з операційними системами:

- Windows 7;
- Windows 8;
- Windows 10;

2. Тестування продуктивності. В ході тестування система продовжувала успішно працювати при таких умовах:

- низький рівень заряду акумулятора;
- погане покриття мережею;
- низька кількість доступної пам’яті;
- одночасний доступ до сервера кількома користувачами.

3. Навантажувальне тестування. На ноутбуках одночасно запускали кілька сторонніх ПЗ, що використовували підключення до мережі Інтернет.

4. Тестування перериванням. Під час тестування перевіряли поведінку додатка у разі:

- отримання повідомлень під час роботи зПЗ;
- відповіді на дзвінок у Skype під час роботи зПЗ;
- підключення та відключення USB кабелю.

Усі тести пройшли успішно, і ПЗ готове до використання та експлуатації. Фатальних багів виявлено не було.

Для запуску системи програмного продукту необхідна операційна система Windows 7 або новіша. Що стосується апаратних вимог, то вони мінімальні, оскільки ПЗ сумісне з більшістю пристроїв, що працюють на операційних

системах Windows 7 та вище. Для коректної роботи потрібно лише 30 МБ вільного простору на диску.

Висновки до розділу 3:

1. Для розроблення програмного забезпечення для підтримки комунікаційної діяльності використано середовище Rad Studio та мова програмування Delphi. Вибір цієї технології обґрунтований простотою освоєння, наявністю потужних інструментів для об’єктно-орієнтованого програмування та можливістю створення крос-платформених рішень. Рад Studio забезпечує зручність розробки завдяки колекції стандартних компонентів інтерфейсу, підтримці тестування програм на різних операційних системах, а також інструментам для рефакторингу та оптимізації коду.

2. Розробка інтерфейсу користувача зосереджена на забезпеченні зручності взаємодії через прості та функціональні елементи, такі як вікна для авторизації, додавання контактних даних університетів, генерації звітів та масової розсилки повідомлень. Важливими аспектами є інтеграція з базами даних для збереження та обробки даних, а також підтримка функціональності для завантаження даних через Drag-and-Drop.

3. Використано різні методи для обробки та збереження даних, забезпечуючи таким чином не лише інтерактивний, але й ефективний досвід для користувачів. Застосування об’єктно-орієнтованого підходу для обробки запитів, автентифікації користувачів і відправлення повідомлень через SMTP забезпечує стабільність та безпеку роботи системи.

4. Програмна реалізація демонструє високий рівень інтеграції між різними компонентами системи та дає змогу легко масштабувати програму під різні платформи, одночасно зберігаючи зручність для кінцевого користувача.

5. Наведено UML-діаграму класів, яка відображає основну структуру розробленого програмного рішення та взаємодію між його компонентами. Діаграма включає ключові класи, такі як TUniversity, TFileHandler, TDatabaseManager, TUniversityForm, TUniversityList, TLeaveRequestGenerator, TEmailSender, і TUserAuth, кожен з яких виконує специфічні функції в межах системи. Усі ці класи, зокрема їхні методи та атрибути, взаємодіють між собою, утворюючи узгоджену систему для обробки даних, створення звітів, а також забезпечення авторизації користувачів. UML-діаграма є ефективним інструментом для візуалізації структури програмного рішення, що допомагає краще зрозуміти логіку роботи системи та спрощує подальший процес розробки і підтримки проєкту.

6. Інтерфейс користувача для розробленого програмного забезпечення побудовано на основі технологій Delphi та FireMonkey для створення багатоплатформенних та адаптивних графічних інтерфейсів. Завдяки використанню FireMonkey, інтерфейс забезпечує високу якість відображення на різноманітних пристроях, таких як персональні комп'ютери, планшети та мобільні телефони, що дозволяє оптимізувати користувацький досвід незалежно від платформи.

7. Під час тестування були перевірені різні умови роботи ПЗ, такі як низький рівень заряду акумулятора, погане мережеве покриття, обмежена кількість пам’яті та одночасний доступ кількох користувачів до сервера. Усі тести пройшли успішно, і фатальних багів не було виявлено, що підтверджує готовність ПЗ до використання. Апаратні вимоги до програми є мінімальними, і вона сумісна з більшістю пристроїв, що працюють на операційних системах Windows 7 та вище.

ЗАГАЛЬНІ ВИСНОВКИ

1. Розроблено програмне забезпечення, яке використовує сучасні технології для створення багатоплатформенних і адаптивних інтерфейсів користувача, зокрема за допомогою Delphi та FireMonkey. Інтерфейс був спроектований з урахуванням вимог зручності для користувача, а також можливості роботи на різних пристроях – ПК, планшетах та мобільних телефонах. Завдяки підтримці Drag-and-Drop, користувач може зручно додавати дані, що значно полегшує процес введення та обробки інформації.

2. В рамках розробки було реалізовано UML-діаграму класів, яка чітко описує структуру програмного продукту і взаємодію між його компонентами. Класична організація даних, перевірка введених даних і можливість взаємодії з базою даних дозволяють забезпечити надійну роботу програми в реальних умовах.

3. У процесі тестування програмного забезпечення було проведено ряд важливих перевірок, таких як інсталяційне тестування, тестування продуктивності, навантажувальне тестування і тестування на переривання. Всі тестування пройшли успішно, і програмний продукт продемонстрував стабільну роботу в різних умовах. Виявлення і усунення помилок на етапі тестування підтвердило високу якість розробки.

Таким чином, розроблене програмне забезпечення відповідає вимогам користувачів і готове до експлуатації. Воно є стабільним, адаптивним і зручним у використанні, що робить його ефективним інструментом для підтримки комунікаційної діяльності

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мисліцька Н.А., Семенюк Д.С., Колесникова О.А. Мобільне навчання в системі сучасних методичних підходів до організації і проведення учнями фізичних досліджень: Наукові записки Центральноукраїнського державного педагогічного університету імені Володимира Винниченка. 31 Серія: Педагогічні науки №183 (2019). [Електронний ресурс] // Режим доступу: https://www.cuspu.edu.ua/ua/ix-mizhnarodna-naukovo-praktychnaonlain-internet-konferentsiia-problemy-ta-innovatsii-v-pryrodnychomatematychnii-tekhnolohichnii-i-profesiinii-osviti/sektsiia-3

2. Терещук С.І. Технологія мобільного навчання: проблеми та шляхи вирішення. Вісник Чернігівського національного педагогічногоуніверситету. Серія: Педагогічні науки. 2016. Вип. 138. С. 178-180.

3. Заболотний В.Ф., Мисліцька Н.А., Слободянюк І.Ю. Хмаро орієнтовані технології навчання: навчально-методичний посібник. Вінниця: ТОВ «Нілан – ЛТД», 2020.144с.

4. Власюк І. Оптимізація процесу навчання читання англійською мовою через інноваційні технології та мобільні застосунки в епоху інформатизації. Актуальні питання у сучасній науці. 2024. № 5 (23). С 792 – 804. Doi: 10.52058/2786-6300-2023- 12(18).

5. Сущенко Л. О., Андрющенко О. О., Сущенко П. Р. Цифрова трансформація закладів вищої освіти в умовах діджиталізації суспільства: виклики і перспективи. Науковий часопис НПУ вмені Н. П. Драгоманова. 2022. № 2. С. 146 – 151. Doi: 10.31392/NPUnc.series5.2022.spec.2.28

6. Скідан В.В. Використання діджитал-технологій в роботі кураторв академічної групи закладів вищої освіти / В.В. Скідан, А.А. Волівач, О.Я. Ніконов, О.В. Мительська // Вісник Хмельницького національного університету, №6, 2023 (329). Хмельницький: ХНТУ, 2023. С. 92-97.

https://scholar.google.com.ua/citations?view_op=view_citation&hl=uk&user=UTNQCScAAAAJ&citation_for_view=UTNQCScAAAAJ:35N4QoGY0k4C

7. John, Steven. "'What is Slack?' Everything you need to know about the professional messaging program". *Business Insider*. Retrieved May 4, 2022.

8. Pierce, David (June 22, 2022). "Slack added video to Huddles so you don't have to spend your life in Zoom meetings". *The Verge*. Retrieved September 8, 2023.

9. Konrad, Alex (July 8, 2020). "Slack Acquires Corporate Directory Startup Rimeto, Plans To Operate It As Standalone App". *Forbes*. Archived from the original on July 9, 2020. Retrieved July 9, 2020.

10. Нова версія Microsoft Teams тепер у загальному доступі. [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-teams/log-in.

11. Asana: Work in one place. [Електронний ресурс] – Режим доступу до ресурсу:https://play.google.com/store/apps/details/Asana_Work_in_one_place?id=com.asana.app&hl=uk&pli=1

12. Що таке Trello і як програма допомагає в роботі Проджект менеджера. [Електронний ресурс] – Режим доступу до ресурсу: https://wizeclub.education/blog/shho-take-trello-i-yak-programa-dopomagaye-v-roboti-prodzhekt-menedzhera/

13. Mailchimp Email Marketing. [Електронний ресурс] – Режим доступу до ресурсу:https://play.google.com/store/apps/details?id=com.mailchimp.mailchimp&hl=uk&pli=1

14. Hootsuite vs. Buffer: Grow fast with the best social media management solution. [Електронний ресурс] – Режим доступу до ресурсу: https://www.hootsuite.com/hootsuite-vs

15. Grow better with HubSpot. [Електронний ресурс] – Режим доступу до ресурсу: https://www.hubspot.com/

16. Zoom. [Електронний ресурс] – Режим доступу до ресурсу: https://www.dekom.com/ua-uk/videokonferencsvjaz/zoom/

17. Google Classroom is the main address of Education, Training, and Study! [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/view/classroom-workspace/

18. Що таке Skype? [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/skype/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-skype-4ee90a4c-7183-439f-b6dc-dad1254dfd3f

19. Google Meet. [Електронний ресурс] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.google.android.apps.tachyon&hl=uk

20. Webex Meetings. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cisco.com/c/en/us/products/conferencing/webex.html?carousel=1

21. Shahzad, B., Abdullatif, A. M., Ikram, N., & Mashkoor, A. (2017). Build software or buy: A study on developing large scale software. IEEE Access, 5(3), 24262-24274. https://doi.org/10.1109/ACCESS.2017.2763324

22. VersionOne. (2021). 15th Annual State of Agile Report. https://dl.acm.org/doi/full/10.1145/3571849#Bib0154Rd60581057e222

23. Williams, L., Brown, G., Meltzer, A., & Nagappan, N. (2011). Scrum engineering practices: Experiences of three Microsoft teams. In Proceedings of the International Symposium on Empirical Software Engineering and Measurement. IEEE, Los Alamitos, CA, 463–471. https://doi.org/10.1109/ESEM.2011.

24. Maximini, D. (2018). The Scrum Culture: Introducing Agile Methods in Organizations (2nd ed.). Springer International Publishing. https://doi.org/10.1007/978-3-319-73842-0

25. Македон В.В., Валіков В.П., Рябик Г.Є. Розвиток світового ринку ділових інтелектуальних послуг під впливом економіки 4.0. Нобелівський вісник. 2019. № 1. С. 59-72. DOI: 10.32342/2616-3853- 2019-2-12-7

26. Коломієць В. Ф. Роль Internet-технологій у формуванні свідомості [Електронний ресурс] / В. Ф. Коломієць, А. І. Вишневський // Проблеми міжнародних відносин. – 2014. – Вип. 8. – С. 96−108. – Режим доступу: http://nbuv.gov.ua/UJRN/Pmv_2014_8_9.

27. Дмитренко О. Дослідження Інтернет Асоціації. – 2017. – 13 квітня [Електронний ресурс] / О.Дмитренко. – Режим доступу: http://watcher.com.ua/uanetstatistics/

28. Дані досліджень інтернет-аудиторії України – 2017 [Електронний ресурс]. – Режим доступу: http://inau.ua/proekty/doslidzhennya-internet-audytoriyi.

29. Демченко І. Сучасні інформаційно-комунікативні технології як провідний чинник глобальних світоглядних трансформацій [Електронний ресурс] / Інна Демченко // XVIII Міжнародна науково-практична інтернет-конференція. – 2013. – Режим доступу: http://oldconf.neasmo.org.ua/node/2098.

30. Schilirò, D. (2023). Digital platforms and digital transformation. MPRA Paper No. 118006. Retrieved from https://mpra.ub.uni-muenchen.de/118006/

31. Coyle, D. (2018). Practical competition policy implications of digital platforms. Bennett Institute for Public Policy working paper no: 01/2018 Retrieved from https://cutt.ly/9wFggoGX

32. Биков В.Ю. Хмарна комп’ютерно-технологічна платформа відкритої освіти та відповідний розвиток організаційно-технологічної будови ІТ-підрозділів навчальних закладів // Теорія і практика управління соціальними системами: філософія, психологія, педагогіка, соціологія / Щоквартальний науково-практичний журнал. – Харків: НТУ «ХПІ», 2013. – № 1. – С. 81-98.

33. Комунікативна діяльність в державному управлінні : навч. пос. / Н. М. Драгомирецька, К.С. Кандагура, А.В. Букач. Одеса : ОРІДУ НАДУ, 2017. 180 с.

34. Робота із соціальними мережами. Посібник з питань використання соціальних мереж, розроблений Департаментом преси і публічної інформації Консультативної місії ЄС в Україні. Київ : серпень 2020. URL: https://www.euam-ukraine.eu/ wp-content/uploads/2020/09/Working-with-SocialMedia-Compendium-UKRAINIAN-AUGUST2020-FOR PUBLICATION.pdf/.

35. Чуприна А. Роль соціальних мереж у кризовій комунікації в умовах війни. Інформаційне агенство «ЛІГАБізнесІнформ». Вебсайт. URL: https://blog.liga.net/user/achupryna/article/ 45924

36. Коваленко С. В. Месенджер як альтернативна форма офіційного інформування та оповіщення в громаді. Сучасна парадигма публічного управління : зб. матеріалів IV Міжнар. наук.- практ. конф.,10-12 листоп. 2022 р. / за наук. ред. к.е.н., доцента Стасишина А. В. Львів : ЛНУ імені Івана Франка. 2022. С. 958.

37. Рад Б. Я. Архітектура інформаційних систем / Б. Я. Рад, А. І. Водяхо, В. А. Дубенецький, В. В. Цехановскій. – М.: Академія, 2012. 220с

38. Рой філдінг Архітектурні стилі та дизайн мережевих архітектур програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

39. Харченко О. Г. Експерта система проектування архітектури програмного забезпечення / О. Г. Харченко, І. О. Боднарчук, В. В. Яцишин // Комп'ютерні технології друкарства. – № 29. – 2013. – с. 10-26.

40. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура.

41. Скідан В.В., Демківська Т.І. Аналіз архітектурних стилів при розробці WEB-додатків. Інформаційні технології в науці, виробництві та підприємництві:

Збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп'ютерних наук та технологій / загал. наук. ред. В.Ю. Щербань – К.: ТОВ Фастбінд Україна, 2022. – 137-140 с. https://er.knutd.edu.ua/bitstream/123456789/23155/1/C%D0%BA%D1%96%D0%B4%D0%B0%D0%BD.pdf

42. Веб-дизайн та UX/UI: як створити зручний інтерфейс? [Електронний ресурс] – Режим доступу до ресурсу: https://whileweb.com/uk/blog/veb-dizajn-ta-uxui-yak-stvoriti-zruchnij-interfejs/

43. Веб-дизайн, UI/UX, ефективні інтерфейси. [Електронний ресурс] – Режим доступу до ресурсу: https://goldwebsolutions.com/uk/uiux-uk/

44. Frontend. Інтерфейс для взаємодії між користувачем і back end. [Електронний ресурс] – Режим доступу до ресурсу: https://goldwebsolutions.com/uk/blog/frontend/

45. Доценко С. І. Організація та системи керування базами даних: Навч. посібник. – Харків: УкрДУЗТ, 2023. – 117 с.

46. Бази даних : навчально-практичний посібник для самостійної роботи студентів [Електронний ресурс] / М. Ю. Лосєв, В. В. Федько. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 233 с/

47. Операційна система Windows. [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/programnezabezpecenna/operacijna-sistema-windows

48. Delphi. [Електронний ресурс] – Режим доступу до ресурсу: http://delphi.dp.ua/

49. Створення схеми класу UML. [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk

50. Бардус І. О. Бази даних у схемах (на основі фундаменталізованого підходу) : навч. посіб. / І. О. Бардус, М. І. Лазарєв, А. О. Ніценко. − Харків : Вид-во «Діса

плюс», 2017. – 133 с.

51. Безменов М. І. Основи програмування у середовищі Delphi : навч. посіб. – Харків : НТУ «ХПІ», 2010. – 608 с.

52. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2017. – 284 с.

53. Jorgensen P. C., Erickson C. Object-Oriented Integration Testing Communications of the ACM. 37, 9 (Sept, 1994), pp. 30-38.

54. Burnstein I. Practical Software Testing. A process-oriented approach. Springer-Verlag, New York, 2003, – 732 p.

55. Крепич С.Я . Якість програмного забезпечення та тестування: базовий курс. Навчальний посібник / За ред. Крепич С.Я., Співак І.Я. / для бакалаврів галузі знань 12 «Інформаційні технології» спеціальності 121 «Інженерія програмного забезпечення». – Тернопіль: ФОП Паляниця В.А., 2020. – 478с.

56. Білас О.Є. Якість программного забезпечення та тестування. Навчальний посібник / О.Є. Білас – Львів: Видавництво Львівської політехніки, 2011. 216с.

57. Тестування програмного продукту [Електронний ресурс] – Режим доступу до ресурсу: http://lib.mdpu.org.ua/e-book/vstup/L11.htm.

58. Говорущенко Т.О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення : монографія / Говорущенко Т. О. Хмельницький : ХНУ, 2017. 310 с.

59. Золотухіна О. А. Якість та тестування інформаційних систем : Навчальний посібник для самостійної роботи студентів вищих навчальних закладів / Золотухіна О. А., Негоденко О. В., Резник С. Ю., Разіна С. Я. – Київ : ННІІТ ДУТ, 2020. – 128 с.

60. Web-testing. [Electronic resource]. – Access mode: http://www.edb. utexas.edu/minliu/multimedia/PDFfolder/WebTestingPadolina.pdf.

61. Software Testing Help. “Entries Tagged Cookie Testing. Website Cookie Testing, Test cases for testing web application cookies?”. – [Electronic resource]. – Access mode: http://www.softwaretestinghelp.com /category/cookie-testing/

62. Науменко Л.С. Навчальний посібник «Методи тестування та оцінки якості програмного забезпечення» Частина І: Тестування мобільних веб-сайтів та додатків для студентів денної та заочної форми навчання: 122 «Комп’ютерні науки та інформаційні технології» – Полтава: ПолтНТУ, 2018. – 176 с.

ДОДАТОК А

ТЕЗИ КОНФЕРЕНЦІЇ

ДОДАТОК Б

Програмний лістинг 1. Модуль для генерації звітів

```
uses
  System.SysUtils, System.Classes, IdHTTP, IdSSLOpenSSL, System.JSON, System.IOUtils;

const REPORT_TO = '#####';

type
  TLeaveRequestGenerator = class
  private
    fFullName: String;
    fStartDate: String;
    fEndDate: String;
    fCurrentDate: String;
    fUniversityName: String;
    fRequestType: String;
  public
    constructor Create(const FullName, StartDate, EndDate, RectorName, UniversityName,
RequestType: string);

    procedure GenerateLeaveStatement(const FileName: string);
    procedure GenerateSickLeaveStatement(const FileName: string);
    procedure GenerateTransferStatement(const FileName: string);

    procedure LoadDataFromGET(const URL: string);
    procedure LoadDataFromPOST(const URL: string; const PostData: string);

    procedure SaveStatementToFile(const FileName: string; const StatementText: string);

    property FullName: string read fFullName write fFullName;
```

```
    property StartDate: string read fStartDate write fStartDate;
    property EndDate: string read fEndDate write FEndDate;
    property RectorName: string read fRectorName write fRectorName;
    property UniversityName: string read fUniversityName write fUniversityName;
    property RequestType: string read fRequestType write fRequestType;
  end;

implementation

constructor TLeaveRequestGenerator.Create(const FullName, StartDate, EndDate, RectorName,
UniversityName, RequestType: string);
begin
  FFullName := FullName;
  FStartDate := StartDate;
  FEndDate := EndDate;
  FCurrentDate := DateToStr(Date);  // Поточна дата
  FRectorName := RectorName;
  FUniversityName := UniversityName;
  FRequestType := RequestType;
end;

procedure TLeaveRequestGenerator.GenerateLeaveStatement(const FileName: string);
var
  StatementText: string;
begin
  StatementText := Format(
    'ДО %s' + REPORT_TO + sLineBreak +
    'ПРОШУ НАДАТИ МЕНІ ВІДПУСТКУ' + sLineBreak +
    'Я, %s, студент(ка) %s, прошу надати мені академ. відпустку з %s по %s.' + sLineBreak +
    'Заява подана на підставі моїх особистих обставин.' + sLineBreak +
    'Дата подачі заяви: %s.' + sLineBreak +
```

```
    'Підпис: _____' + sLineBreak +
    'Дата: ' + FCurrentDate,
    [FRectorName, FFullName, FUniversityName, FStartDate, FEndDate, FCurrentDate]
  );
  SaveStatementToFile(FileName, StatementText);
end;

procedure TLeaveRequestGenerator.GenerateSickLeaveStatement(const FileName: string);
var
  StatementText: string;
begin
  StatementText := Format(
    'ДО %s' + REPORT_TO + sLineBreak +
    'ПРОШУ НАДАТИ МЕНІ ЛІКАРНЯНИЙ' + sLineBreak +
    'Я, %s, викладач(ка) %s, прошу надати мені лікарняний з %s по %s.' + sLineBreak +
    'Заява подана на підставі медичних показників.' + sLineBreak +
    'Дата подачі заяви: %s.' + sLineBreak +
    'Підпис: _____' + sLineBreak +
    'Дата: ' + FCurrentDate,
    [FFullName, FUniversityName, FStartDate, FEndDate, FCurrentDate]
  );
  SaveStatementToFile(FileName, StatementText);
end;

procedure TLeaveRequestGenerator.GenerateTransferStatement(const FileName: string);
var
  StatementText: string;
begin
  StatementText := Format(
    'ДО %s' + REPORT_TO + sLineBreak +
    'ПРОШУ ПЕРЕВЕСТИ МЕНЕ' + sLineBreak +
```

```
    'Я, %s, студент(ка) %s, прошу перевести мене з %s по %s.' + sLineBreak +
    'Заява подана на підставі моїх особистих обставин.' + sLineBreak +
    'Дата подачі заяви: %s.' + sLineBreak +
    'Підпис: _____' + sLineBreak +
    'Дата: ' + FCurrentDate,
    [FFullName, FUniversityName, FStartDate, FEndDate, FCurrentDate]
  );
  SaveStatementToFile(FileName, StatementText);
end;

procedure TLeaveRequestGenerator.SaveStatementToFile(const FileName: string; const
StatementText: string);
begin
  TFile.WriteAllText(FileName, StatementText);
end;
```

ДОДАТОК В

Програмний лістинг 2. Модуль для відправки повідомлень

```
uses
  System.SysUtils, System.Classes, IdSMTP, IdSSL, IdSSLOpenSSL, IdMessage, IdAttachmentFile,
IdGlobal;

type
  TEmailSender = class
  private
    FSMTPServer: string;
    FSMTPPort: Integer;
    FSMTPUsername: string;
    FSMTPPassword: string;
    FSenderEmail: string;
    FSenderName: string;
    FSSLType: TIdSSLType;
  public
    constructor Create(const SMTPServer, SMTPUsername, SMTPPassword, SenderEmail,
SenderName: string; SMTPPort: Integer = 587; SSLType: TIdSSLType = sslTLS);
    destructor Destroy; override;

    function SendEmail(const Subject, Body, RecipientEmail: string; const Attachments:
TArray<string> = nil): Boolean;

    function SendEmailWithFiles(const Subject, Body, RecipientEmail: string; const Attachments:
TArray<string>; const Files: TArray<string>): Boolean;

    property SMTPServer: string read FSMTPServer write FSMTPServer;
    property SMTPPort: Integer read FSMTPPort write FSMTPPort;
    property SMTPUsername: string read FSMTPUsername write FSMTPUsername;
    property SMTPPassword: string read FSMTPPassword write FSMTPPassword;
    property SenderEmail: string read FSenderEmail write FSenderEmail;
    property SenderName: string read FSenderName write FSenderName;
    property SSLType: TIdSSLType read FSSLType write FSSLType;
  end;

implementation
```

```
constructor TEmailSender.Create(const SMTPServer, SMTPUsername, SMTPPassword,
SenderEmail, SenderName: string; SMTPPort: Integer = 587; SSLType: TIdSSLType = sslTLS);
begin
  FSMTPServer := SMTPServer;
  FSMTPPort := SMTPPort;
  FSMTPUsername := SMTPUsername;
  FSMTPPassword := SMTPPassword;
  FSenderEmail := SenderEmail;
  FSenderName := SenderName;
  FSSLType := SSLType;
end;

destructor TEmailSender.Destroy;
begin
  inherited;
end;

function TEmailSender.SendEmail(const Subject, Body, RecipientEmail: string; const Attachments:
TArray<string> = nil): Boolean;
var
  SMTP: TIdSMTP;
  Msg: TIdMessage;
  i: Integer;
begin
  Result := False;
SMTP := TIdSMTP.Create(nil);
  Msg := TIdMessage.Create(nil);
  try
    SMTP.Host := FSMTPServer;
    SMTP.Port := FSMTPPort;
    SMTP.Username := FSMTPUsername;
    SMTP.Password := FSMTPPassword;
    SMTP.UseTLS := utUseExplicitTLS;
    SMTP.IOHandler := TIdSSLIOHandlerSocketOpenSSL.Create(SMTP);

    Msg.Subject := Subject;
    Msg.Body.Text := Body;
    Msg.From.Address := FSenderEmail;
    Msg.From.Name := FSenderName;
    Msg.Recipients.EMailAddresses := RecipientEmail;
```

```
    if Length(Attachments) > 0 then
    begin
      for i := 0 to High(Attachments) do
      begin
        if TFile.Exists(Attachments[i]) then
        begin
          Msg.AddAttachment(Attachments[i]);
        end;
      end;
    end;

    SMTP.Connect;
    try
      SMTP.Send(Msg);
      Result := True;
    except
      on E: Exception do
        raise Exception.Create('Помилка при надсиланні листа: ' + E.Message);
    end;
  finally
    SMTP.Free;
    Msg.Free;
  end;
end;

function TEmailSender.SendEmailWithFiles(const Subject, Body, RecipientEmail: string; const
Attachments: TArray<string>; const Files: TArray<string>): Boolean;
var
  SMTP: TIdSMTP;
  Msg: TIdMessage;
  i: Integer;
begin
  Result := False;

  SMTP := TIdSMTP.Create(nil);
  Msg := TIdMessage.Create(nil);
  try
    SMTP.Host := FSMTPServer;
    SMTP.Port := FSMTPPort;
    SMTP.Username := FSMTPUsername;
    SMTP.Password := FSMTPPassword;
```

```
    SMTP.UseTLS := utUseExplicitTLS;
    SMTP.IOHandler := TIdSSLIOHandlerSocketOpenSSL.Create(SMTP);

  Msg.Subject := Subject;
    Msg.Body.Text := Body;
    Msg.From.Address := FSenderEmail;
    Msg.From.Name := FSenderName;
    Msg.Recipients.EMailAddresses := RecipientEmail;

    if Length(Attachments) > 0 then
    begin
      for i := 0 to High(Attachments) do
      begin
        if TFile.Exists(Attachments[i]) then
        begin
          Msg.AddAttachment(Attachments[i]);
        end;
      end;
    end;

  if Length(Files) > 0 then
    begin
      for i := 0 to High(Files) do
      begin
        if TFile.Exists(Files[i]) then
        begin
          Msg.AddAttachment(Files[i]);
        end;
      end;
    end;

    SMTP.Connect;
    try
      SMTP.Send(Msg);
      Result := True;
    except
      on E: Exception do
        raise Exception.Create('Помилка при надсиланні листа: ' + E.Message);
    end;
   finally
    SMTP.Free;
```

```
    Msg.Free;
  end;
end;

end.
```