Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Лабораторна робота №3
«Технології розроблення програмного забезпечення»
Тема: « E-mail клієнт»

Виконав:
Студент групи ІА-34
Марченко А.О.
Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Основи проектування розгортання.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

• Розробити діаграму компонентів для проєктованої системи.

• Розробити діаграму розгортання для проєктованої системи.

• Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

• На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

Теоретичні відомості

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення [3].

Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення.

Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер).

Між вузлами можуть стояти зв'язки, які зазвичай зображують у вигляді прямої лінії. Як і на інших діаграмах, у зв'язків можуть бути атрибути множинності (для показання, наприклад, підключення 2х і більше клієнтів до одного сервера) і назва. У назві, як правило, міститься спосіб зв'язку між двомавузлами – це може бути назва протоколу (HTTP, IPC) або технологія, що використовується для забезпечення взаємодії вузлів (.NET Remoting, WCF).

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути виконувані файли (такі як файли .exe, двійкові файли, файли DLL, файли JAR, збірки або сценарії) або файли даних, конфігураційні файли, HTML-документи тощо. Перелік артефактів усередині вузла вказує на те, що на даному вузлі артефакт розгортається в систему, що запускається.

Діаграми розгортань розрізняють двох видів: описові та екземплярні. На діаграмах описової форми вказуються вузли, артефакти і зв'язки між вузлами без вказівки конкретного обладнання або програмного забезпечення, необхідного для розгортання. Такий вид діаграм корисний на ранніх етапах розроблення для розуміння, які взагалі фізичні пристрої необхідні для функціонування системи або для опису процесу розгортання в загальному ключі.

Діаграми екземплярної форми несуть у собі екземпляри обладнання, артефактів і зв'язків між ними. Під екземплярами розуміють конкретні елементи – ПК із відповідним набором характеристик і встановленим ПЗ; цілком може бути, у межах однієї організації це може бути якийсь

конкретний вузол (наприклад, ПК тестувальника Василя). Діаграми екземплярної форми розробляють на завершальних стадіях розроблення ПЗ – коли вже відомі та сформульовані вимоги до програмного комплексу, обладнання закуплено і все готово до розгортання. Діаграми такої форми являють собою скоріше план розгортання в графічному вигляді, ніж модель розгортання.

Діаграма компонентів UML є представленням проєктованої системи, розбитої на окремі модулі [3]. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів: логічні; фізичні; виконувані.

Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою.

Коли на діаграмі представляють фізичне розбиття, то в такому разі на діаграмі компонентів показують компоненти та залежності між ними. Залежності показують, що класи в з одного компонента використовують класи з іншого компонента. Фізична модель використовується для розуміння які компоненти повинні бути зібрані в інсталяційний пакет. Також така діаграма показує зміни в якому компоненті будуть впливати на інші компоненти.

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій.

Основні кроки створення діаграми послідовностей:

• визначити акторів і об'єкти, які беруть участь у сценарії;

• побудувати їхні лінії життя;

• розробити послідовність передачі повідомлень між об'єктами;

• додати умовні блоки або цикли за необхідності.

Діаграми послідовностей є корисними для моделювання бізнес-процесів, проєктування архітектури систем і тестування. Вони дають змогу візуалізувати логіку взаємодії компонентів та виявити потенційні проблеми ще на етапі проєктування.

Хід роботи

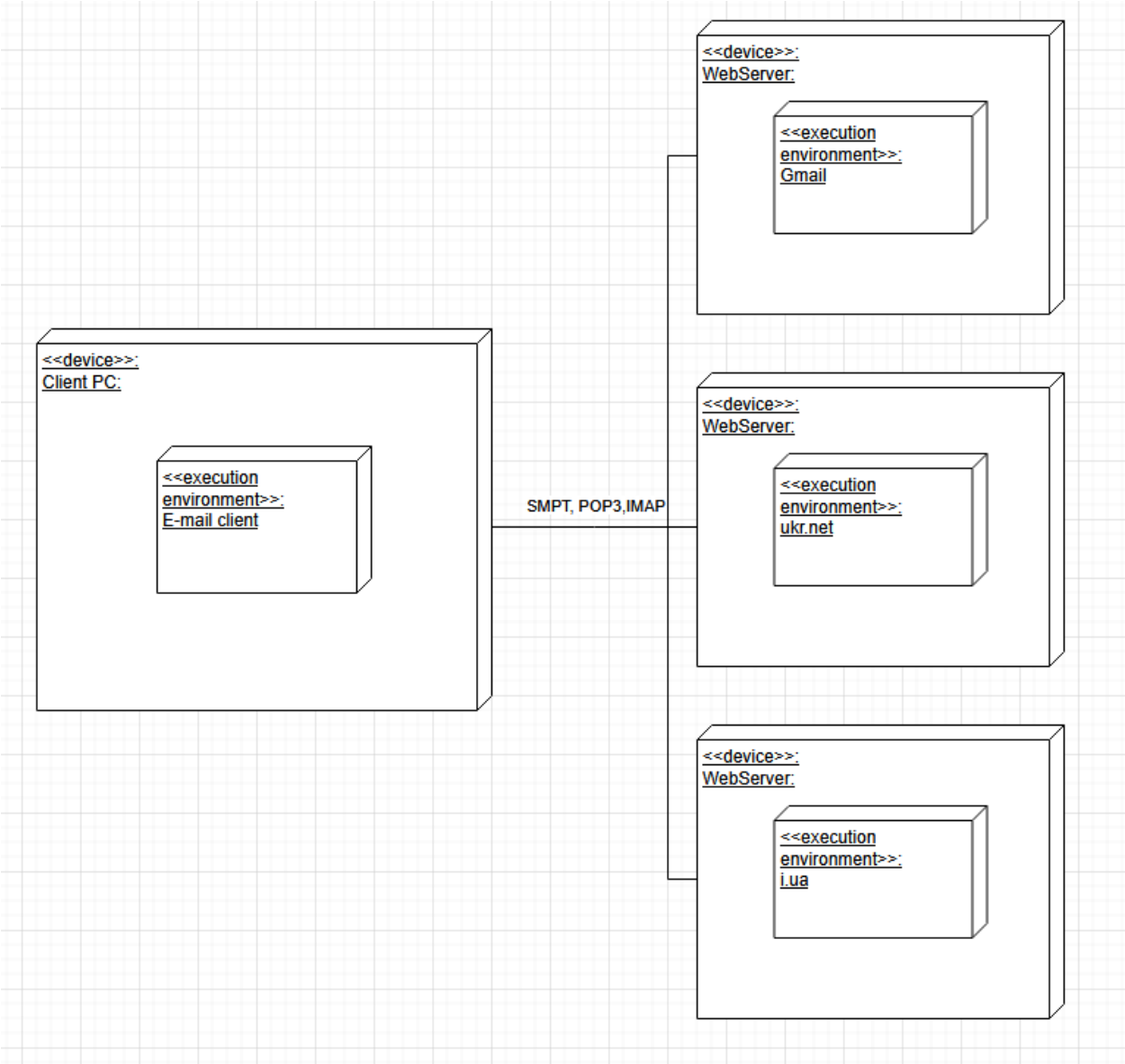

Рис. 3.1 - Діаграма розгортання

На діаграмі представлено чотири основні вузли:

- Client PC - це кінцевий пристрій користувача, з якого здійснюється доступ до електронної пошти.

- WebServer - цей вузол повторюється тричі, представляючи сервери різних поштових провайдерів.

SMTP, POP3, IMAP - це протоколи, які використовуються для обміну даними між E-mail client на ПК та поштовими серверами. SMTP (Simple Mail Transfer Protocol): Використовується для надсилання листів. POP3 (Post Office Protocol version 3) та IMAP (Internet Message Access Protocol): Використовуються для отримання та доступу до листів.

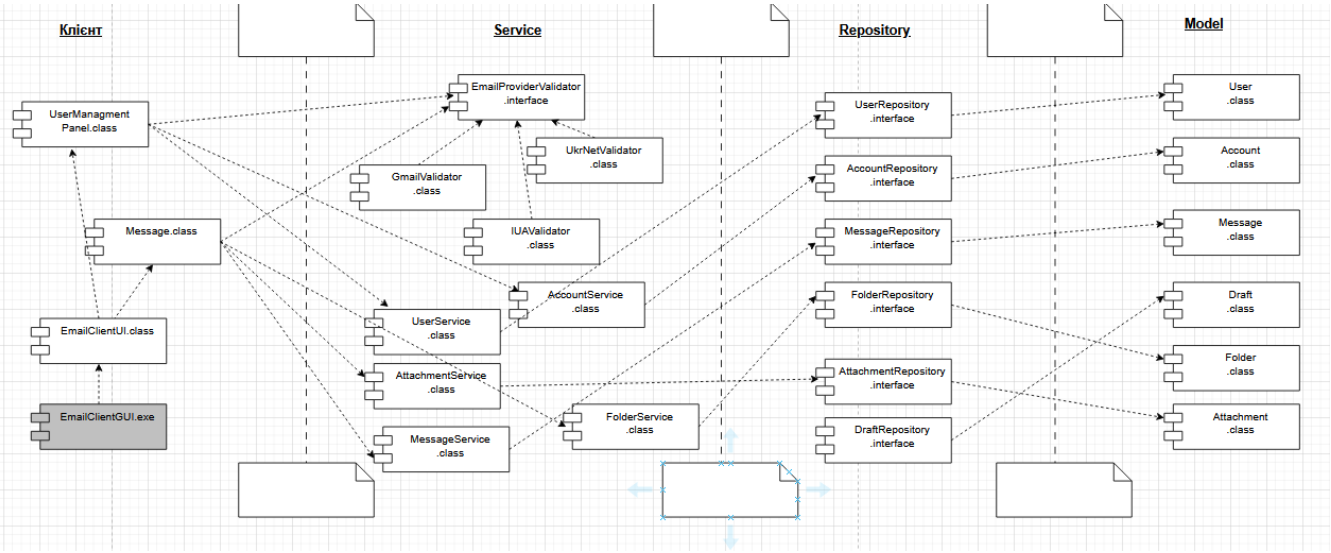

Рис. 3.2 - Діаграма компонентів

У клієнтській частині розміщені компоненти, які відповідають за взаємодію з користувачем та базову логіку інтерфейсу. EmailClientGUI.exe — головний застосунок, який взаємодіє з класами EmailClientUI.class, Message.class та UserManagementPanel.class. Ці компоненти забезпечують відображення інформації, керування повідомленнями та управління користувачами.

Service-рівень містить логіку бізнес-процесів та валідації. Інтерфейс EmailProviderValidator використовується різними реалізаціями валідаторів (GmailValidator, UkrNetValidator, IUAValidator), які відповідають за перевірку правильності налаштувань різних поштових провайдерів. Тут також знаходяться сервіси UserService, AccountService, AttachmentService,

MessageService та FolderService, які реалізують основні операції роботи з користувачами, акаунтами, повідомленнями, вкладеннями та папками.

На Repository-рівні визначені інтерфейси доступу до даних: UserRepository, AccountRepository, MessageRepository, FolderRepository, AttachmentRepository та DraftRepository. Вони виступають проміжною ланкою між бізнес-логікою та моделлю даних, забезпечуючи абстракцію від конкретної реалізації зберігання інформації (наприклад, бази даних).

Model-рівень містить класи доменної моделі: User, Account, Message, Draft, Folder, Attachment. Вони представляють структуру та сутності системи, які зберігаються у сховищі та опрацьовуються сервісами. Repository-рівень напряму взаємодіє з цими класами, забезпечуючи отримання та збереження даних.

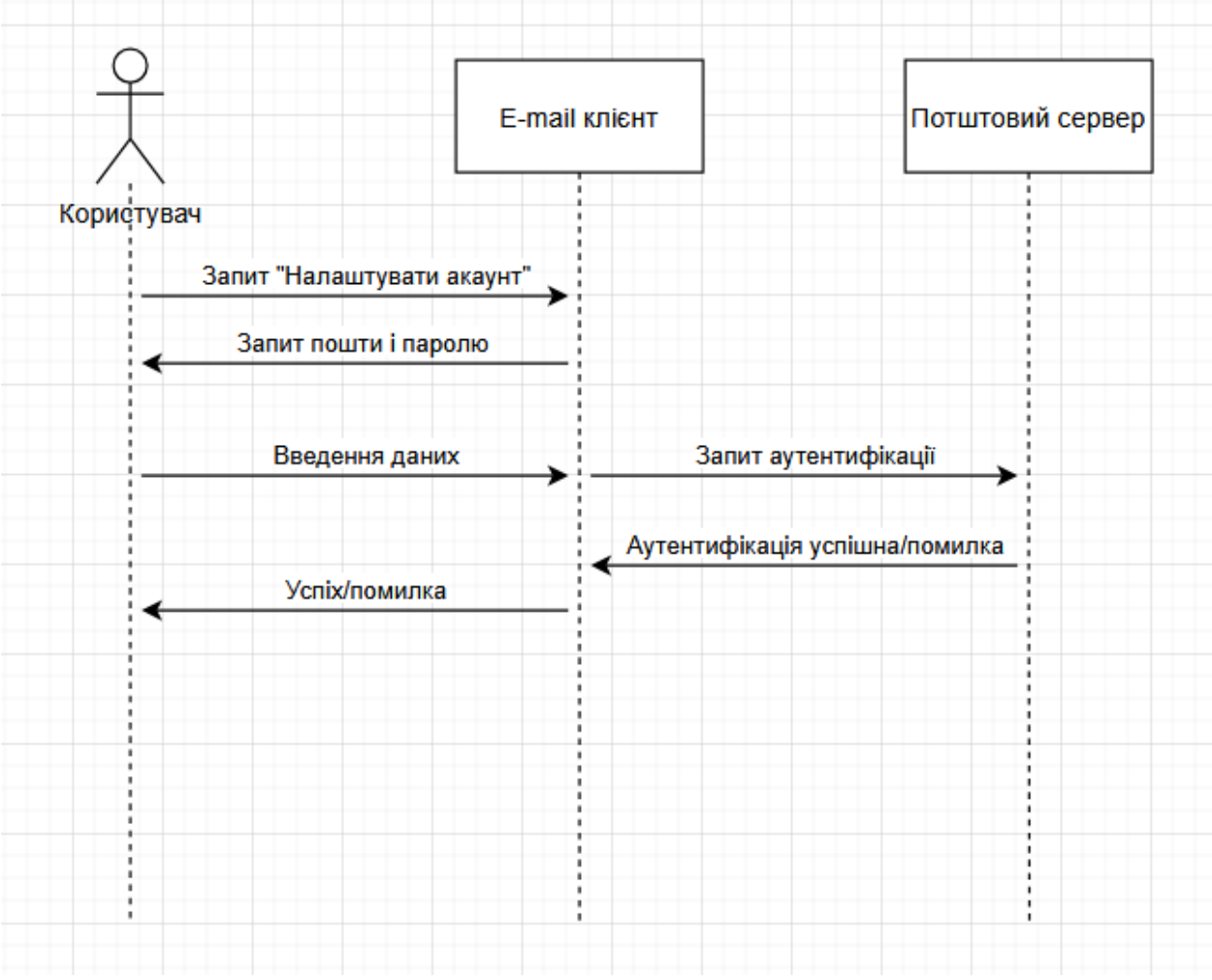

Рис. 3.3 - Діаграма послідовностей (Налаштування акаунта)

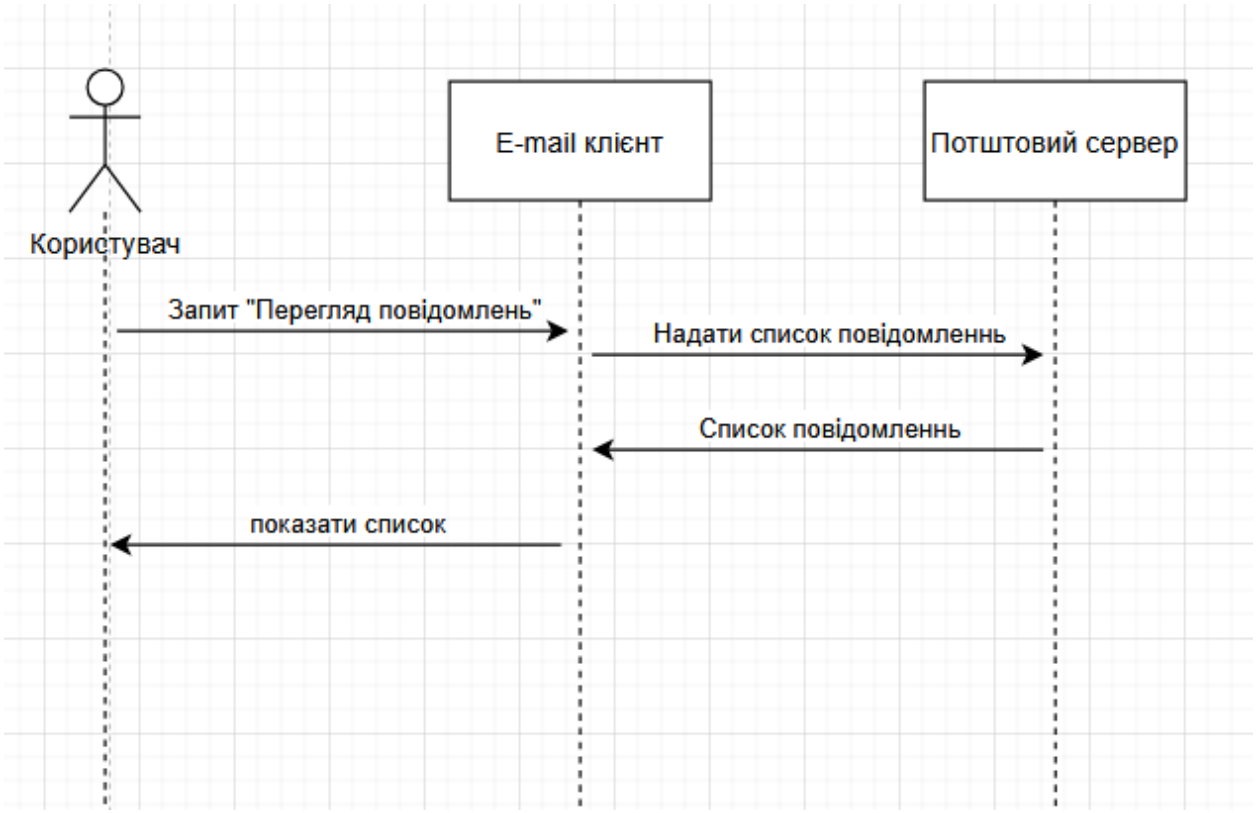

Рис. 3.4 - Діаграма послідовностей (Перегляд повідомлень)

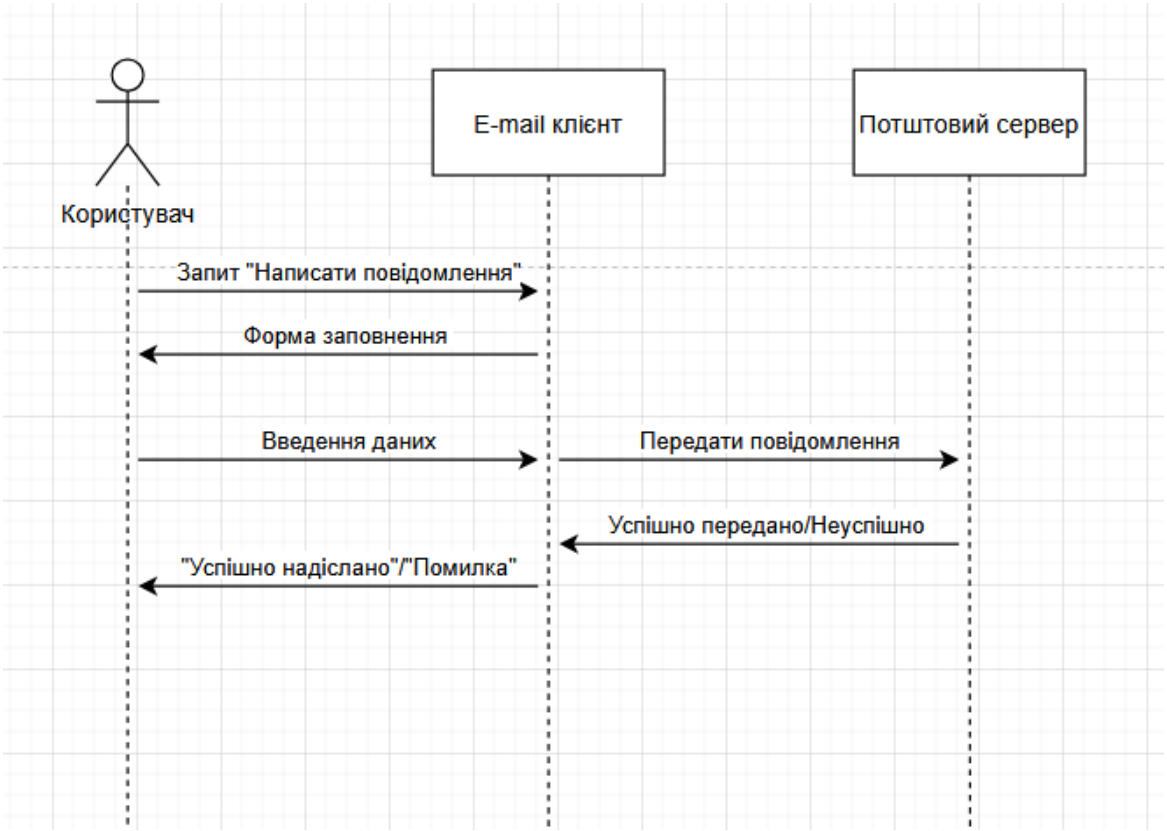

Рис. 3.5 - Діаграма послідовностей (Написати повідомлення)

Відповіді на питання

1.Що собою становить діаграма розгортання?

Діаграма розгортання - це діаграма в мові UML, яка візуалізує архітектуру системи, показуючи, як програмні компоненти розміщуються на фізичних обчислювальних вузлах під час роботи. Вона відображає вузли, а також компоненти та об'єкти, які виконуються на цих вузлах, та зв'язки між ними.

2. Які бувають види вузлів на діаграмі розгортання?

Існують два основні види вузлів на діаграмі розгортання: фізичні вузли та виконавчі вузли. Фізичні вузли представляють фізичні апаратні пристрої, такі як сервери або комп'ютери, а виконавчі вузли - програмні компоненти, які виконуються на цих пристроях, наприклад, програмне забезпечення, бази даних чи веб-сервіси.

3. Які бувають зв’язки на діаграмі розгортання?

На діаграмах розгортання бувають різні зв'язки, які показують залежності та комунікацію між елементами. Основні типи - це зв'язки, що зображують розміщення ,залежність та комунікацію.

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів присутні компоненти**,** залежності та зв'язки між ними. Компонентами можуть бути програмні файли, модулі, пакети тощо, а також їх інтерфейси та елементи, з яких вони складаються.

5. Що становлять собою зв’язки на діаграмі компонентів?

Зв'язки на діаграмі компонентів представляють залежності між компонентами системи і показують, як один компонент взаємодіє з іншим, часто за допомогою інтерфейсу. Залежності можуть бути різними, наприклад, один компонент використовує функціональність іншого.

- Залежності - зв'язок показує, що один компонент залежить від іншого. Це означає, що для роботи йому потрібні функції або послуги, які надає інший компонент.
- Інтерфейси - зв'язок часто встановлюється через інтерфейс. Компонент може "надавати" один інтерфейс і "використовувати" інший. Діаграма компонентів може відображати ці інтерфейси, показуючи, як компоненти взаємодіють.
- Типи зв'язків - залежності можуть бути представлені пунктирними лініями, які з'єднують компоненти, що залежать один від одного.
- Відображення - зв'язки також можуть відображати, як компоненти взаємодіють, надаючи або споживаючи послуги. Це допомагає зрозуміти загальну структуру системи та її поведінку

6. Які бувають види діаграм взаємодії?

Існують різні види діаграм взаємодії, але найпоширенішими є: діаграми послідовності (які показують взаємодію об'єктів у часі) та діаграми комунікації (які акцентують увагу на взаємодіях між об'єктами, показуючи, як вони пов'язані між собою).

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей призначена для візуалізації взаємодії між об'єктами в системі в часі, показуючи, яку послідовність повідомлень вони надсилають один одному в рамках певного сценарію використання. Вона допомагає моделювати логіку, уточнювати деталі сценаріїв та документувати поведінку системи, особливо для одного конкретного варіанту використання

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Ключовими елементами діаграми послідовностей є об'єкти (або актори), лінії життя, повідомлення (синхронні та асинхронні), активаційні скриньки (або виклики), значення, що повертаються та області життєвого циклу (наприклад, цикли або ітерації).

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграми послідовностей і діаграми варіантів використання пов’язані тим, що діаграма послідовності деталізує поведінку одного конкретного сценарію, який описано в діаграмі варіантів використання. Тобто, діаграма варіантів використання надає загальне уявлення про функціональність системи, а діаграма послідовності показує, як саме ця функціональність реалізується у вигляді взаємодії об’єктів у часі.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграми послідовностей і діаграм класів взаємопов’язані, оскільки перша показує динамічну взаємодію об'єктів (екземплярів класів) в часі, а друга описує статичну структуру цих класів, їхні атрибути та зв'язки. Діаграма класів надає структуру, а діаграма послідовностей — це один зі способів візуалізувати, як об'єкти, що належать до цих класів, взаємодіють для виконання конкретного сценарію чи функціональності.