Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 3

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Виконала:
студентка групи ІА-31
Кизим Є. К.

Київ 2025

Зміст

Теоретичні відомості 4
Хід роботи 7
Діаграма розгортання використання 7
Опис діаграми розгортання 7
Діаграма компонентів 8
Опис компонентів 8
Діаграма послідовностей 9
Опис сценарію №1: Додавання витрати вручну 9
Опис сценарію №2: Перегляд статистики витрат 11
Контрольні питання 12
Репозиторій 14
ДОДАТКИ 15
Додаток А 15
Додаток Б 15
Додаток В 16
Додаток Г 16
Додаток Ґ 16
Додаток Д 17

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.

Моя тема: 27. Особиста бухгалтерія (state, prototype, decorator, bridge, flyweight, SOA) Програма повинна бути наочним засобом для ведення особистих фінансів: витрат і прибутку; з можливістю встановлення періодичних витрат / прибутку (зарплата і орендна плата); введення сканованих чеків з відповідними статтями витрат; побудова статистики; експорт/імпорт в Excel, реляційні джерела даних;

різні рахунки; ведення єдиного фонду на всі рахунки (всією сім'єю) – на особливі потреби (ремонт, автомобіль, відпустка); можливість введення вкладів / кредитів для контролю банківських рахунків (звірка нарахованих відсотків з необхідними і т.д.).

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Діаграми розгортання показують, де фізично розміщується система та на якому обладнанні працюють її частини. Основою таких діаграм є вузли, які з'єднані між собою каналами зв’язку.

Вузол (node) — це елемент, здатний містити програмне забезпечення. Вони поділяються на два типи:

- **Пристрій (device)** — фізичне обладнання, наприклад комп’ютер чи інший апаратний компонент.
- **Середовище виконання (execution environment)** — програмна платформа, що може містити інші програми (наприклад, операційна система чи вебсервер).

Між вузлами показують лінії зв’язку, які можуть мати назви та атрибути множинності. У назві звичайно вказують спосіб зв’язку — наприклад, протокол (HTTP, IPC) або технологію (.NET Remoting, WCF).

Вузли можуть містити **артефакти (artifacts)** — фізичні файли ПЗ (виконувані файли, бібліотеки, конфігурації, HTML-документи тощо). Їх можуть зображати як окремі прямокутники з позначкою «artifact» або просто перераховувати всередині вузла. Також біля вузлів можна додати додаткову інформацію — виробника, ОС, місцезнаходження й інші деталі.

Часто одна логічна задача вимагає кількох фізичних вузлів — це можна позначити або кількома прямокутниками, або числовою міткою.

Діаграми розгортання бувають двох типів:

1. **Описові** — показують вузли, артефакти та зв’язки між ними без прив’язки до реального обладнання. Такі діаграми корисні на ранніх етапах, коли лише визначають загальні потреби для розгортання.
2. **Екземплярні** — містять конкретні фізичні пристрої, встановлене ПЗ, реальні файли та зв’язки. Використовують на фінальних етапах проєкту, коли вже відомі технічні характеристики обладнання та підготовлена інфраструктура. Такі діаграми фактично слугують графічним планом розгортання системи.

Діаграма компонентів

Діаграми компонентів у UML демонструють систему, поділену на незалежні модулі. Існує три види таких діаграм:

1. **Логічні** — показують концептуальні модулі, що працюють автономно та взаємодіють між собою.
2. **Фізичні** — відображають реальні компоненти та залежності між ними. Класи одного компонента можуть використовувати класи іншого, що зображається залежностями. Такі діаграми допомагають зрозуміти, які частини потрібно об’єднати в інсталяційний пакет, та як зміни в одному модулі можуть вплинути на інші. Зазвичай тут показують файли на кшталт .exe і .dll та групують їх на компоненти серверної частини, клієнтської та загальні (middleware).
3. **Виконувані** — кожен компонент відповідає виконуваному файлу, сторінці HTML, таблиці бази даних чи іншому фізичному ресурсу. Цей різновид нагадує діаграму класів, але на рівні виконуваних елементів.

Фізичні діаграми компонентів у сучасному моделюванні майже витіснені діаграмами розгортання.

Діаграма послідовностей (Sequence Diagram)

Цей тип UML-діаграм призначений для відображення взаємодії між об’єктами в певному часовому порядку. Вона демонструє, як саме передаються повідомлення, викликаються методи та як об’єкти реагують протягом сценарію.

Основні елементи:

- **Актори** — зовнішні користувачі чи системи, що взаємодіють із моделлю.
- **Об’єкти або класи** — розташовані горизонтально; з кожного опускається
- **Лінія життя** (вертикальна пунктирна лінія).
- **Повідомлення** — стрілки між об’єктами, що показують виклики методів; можуть бути синхронними, асинхронними чи сигналами повернення.
- **Активності** — прямокутники на лінії життя, що показують час виконання дії.
- **Контрольні структури** — блоки alt, loop та інші для опису умов, циклів і альтернативних сценаріїв.

Щоб побудувати діаграму, потрібно визначити учасників, додати лінії життя та послідовність обмінів повідомленнями, після чого — у разі потреби — включити логічні блоки.

Діаграми послідовностей широко використовують у моделюванні бізнес-процесів, проєктуванні архітектури та підготовці до тестування, оскільки вони дають чітке уявлення про динаміку взаємодії компонентів.

Хід роботи

Діаграма розгортання використання

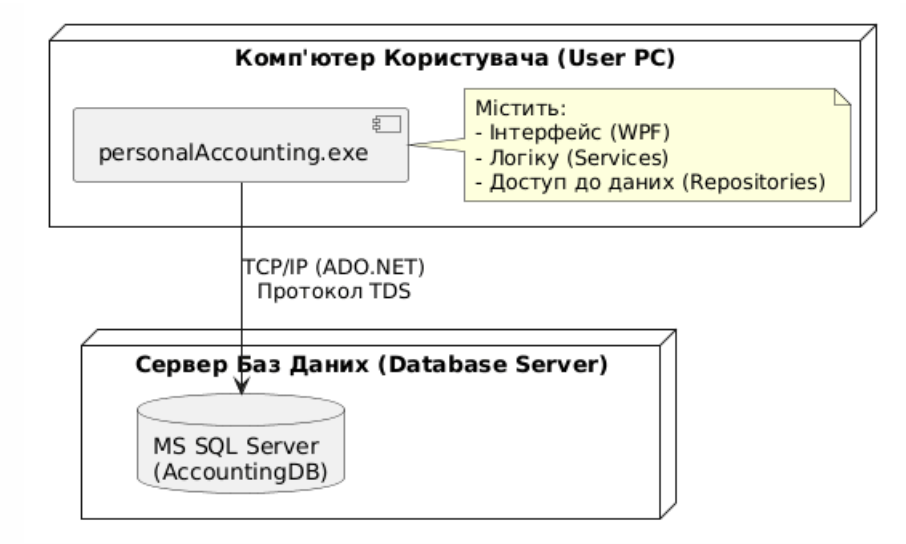


Рисунок 1 – Діаграма розгортання використання

Опис діаграми розгортання:

Вузол Клієнта: Виконує і візуалізацію, і бізнес-логіку.

Сервер БД: Зберігає дані. Зв'язок відбувається через ADO.NET (Microsoft.Data.SqlClient).

Діаграма компонентів

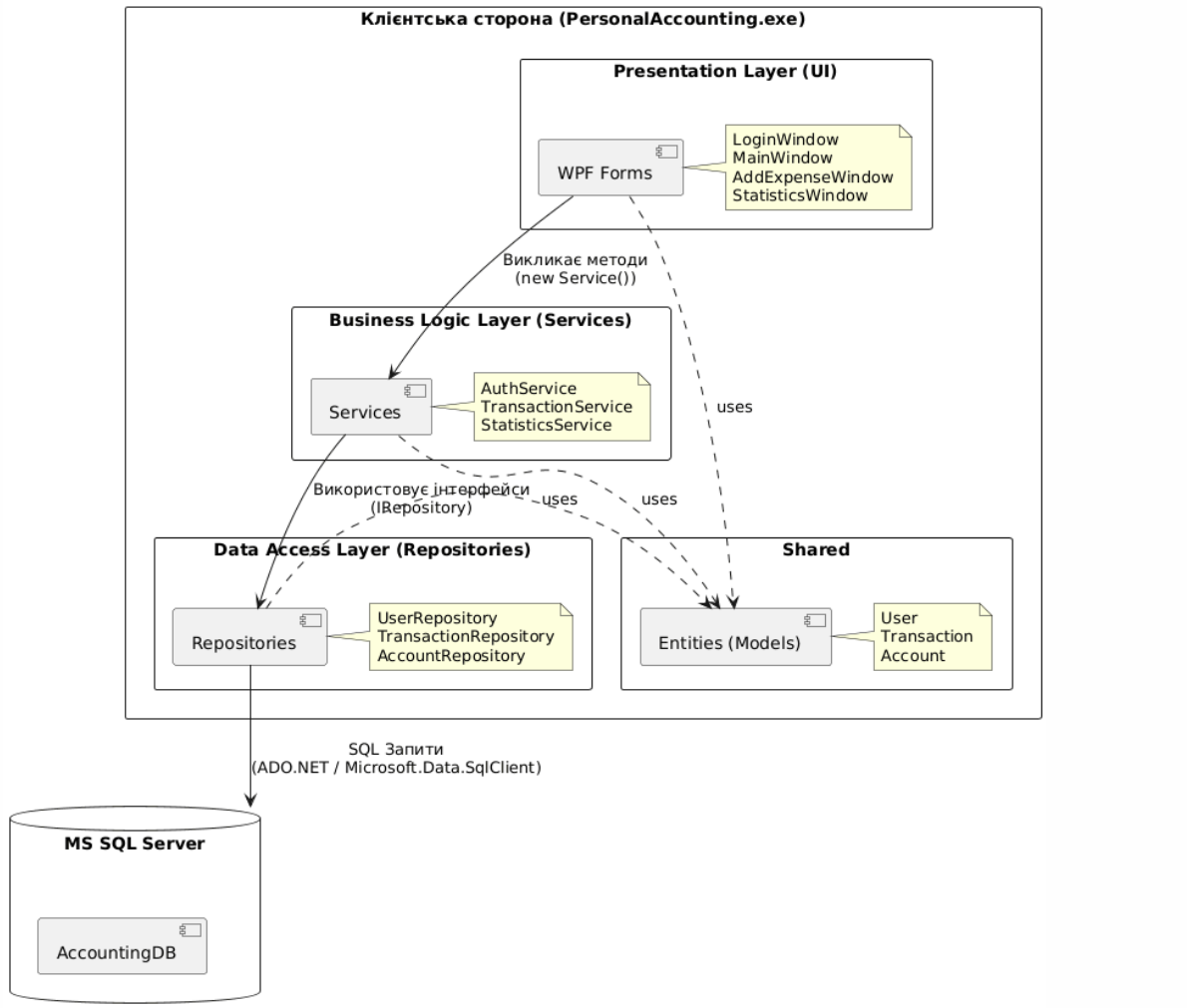


Рисунок 2 – Діаграма компонентів

Опис компонентів:

- **Рівень представлення («Presentation Layer»):** Відповідає за взаємодію з користувачем та візуалізацію даних за допомогою Windows Presentation Foundation (WPF). Цей шар (форми MainWindow, AddExpenseWindow тощо) приймає дані від користувача та ініціює виконання операцій шляхом прямого виклику методів відповідних сервісів.
- **Рівень бізнес-логіки («Business Logic Layer»):** Ядро системи, представлене сервісами (наприклад, TransactionService, StatisticsService). Цей шар виконує валідацію вхідних даних, реалізує бізнес-правила (наприклад, перевірку наявності коштів перед витратою) та делегує операції збереження даних рівню репозиторіїв.

- **Рівень доступу до даних («Data Access Layer»):** Шар репозиторіїв (TransactionRepository, AccountRepository), що інкапсулює логіку взаємодії з базою даних. Він формує та виконує SQL-запити (використовуючи бібліотеку Microsoft.Data.SqlClient) для вибірки, додавання або оновлення інформації.
- **Сутності («Entities»):** Спільні моделі даних (User, Account, Transaction), які описують структуру об'єктів предметної області. Вони використовуються всіма компонентами системи (Інтерфейсом, Логікою та Репозиторіями) для передачі даних між шарами.
- **База Даних:** Зовнішнє реляційне сховище (MS SQL Server) для персистентного (постійного) збереження інформації, до якого надходять SQL-запити від шару доступу до даних.

Діаграма послідовностей

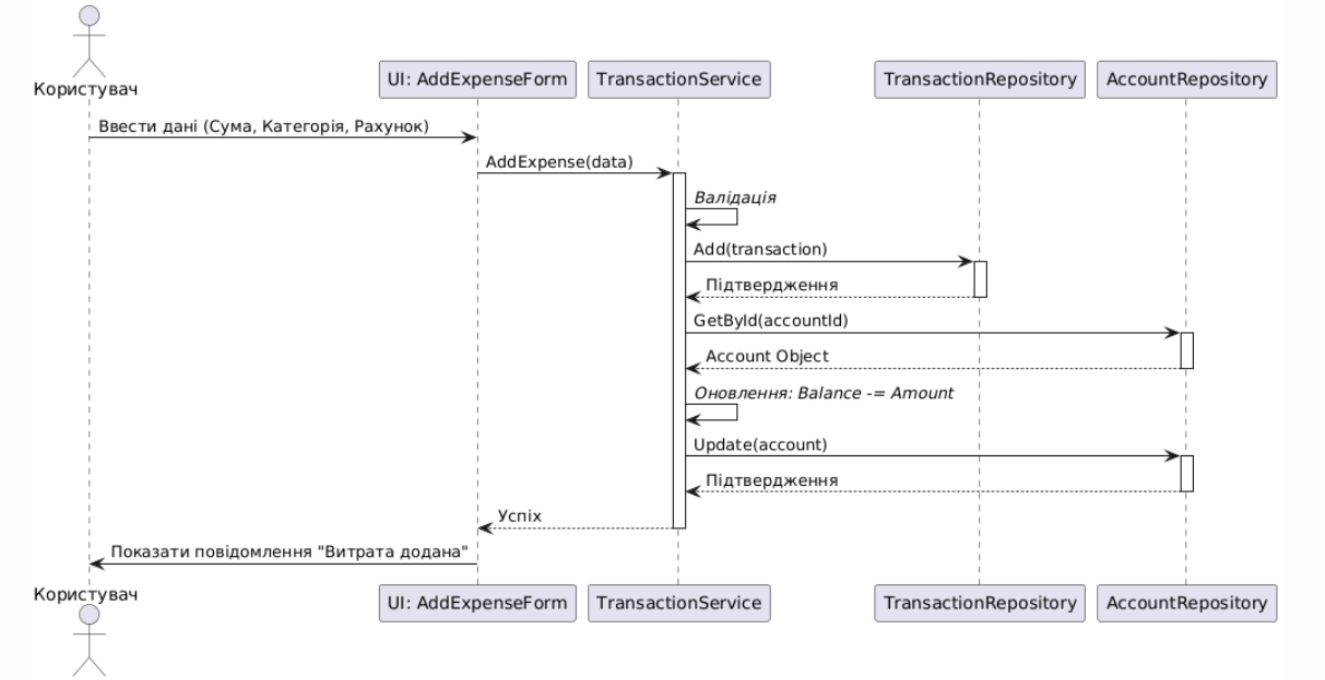


Рисунок 3 – Діаграма послідовності «Додавання витрати вручну»

Опис сценарію №1: Додавання витрати вручну

1. Користувач вводить дані витрати (Сума, Категорія, Рахунок) у формі AddExpenseForm та натискає кнопку «Додати витрату».

2. UI Form надсилає введені дані до TransactionService, викликаючи метод AddExpense(data).
3. TransactionService виконує валідацію отриманої інформації (перевіряє коректність суми, наявність рахунку тощо).
4. Після успішної валідації сервіс передає дані транзакції до TransactionRepository, викликаючи метод Add(transaction) для створення нового запису у БД.
5. TransactionRepository повертає підтвердження, що транзакцію збережено.
6. Далі TransactionService звертається до AccountRepository, викликаючи GetById(accountId), щоб отримати актуальний стан рахунку.
7. AccountRepository повертає Account Object, який містить баланс та інші дані рахунку.
8. TransactionService оновлює баланс рахунку, виконуючи операцію: Balance -= Amount (віднімає витрачену суму).
9. Після оновлення даних сервіс викликає Update(account) у AccountRepository, щоб зберегти зміни балансу в БД.
10. AccountRepository повертає підтвердження, що оновлення виконано успішно.
11. Після успішного завершення всіх операцій TransactionService повертає результат у UI — статус успіху.
12. UI відображає повідомлення користувачу: *«Витрата додана»* та оновлює інтерфейс (баланс, список транзакцій).

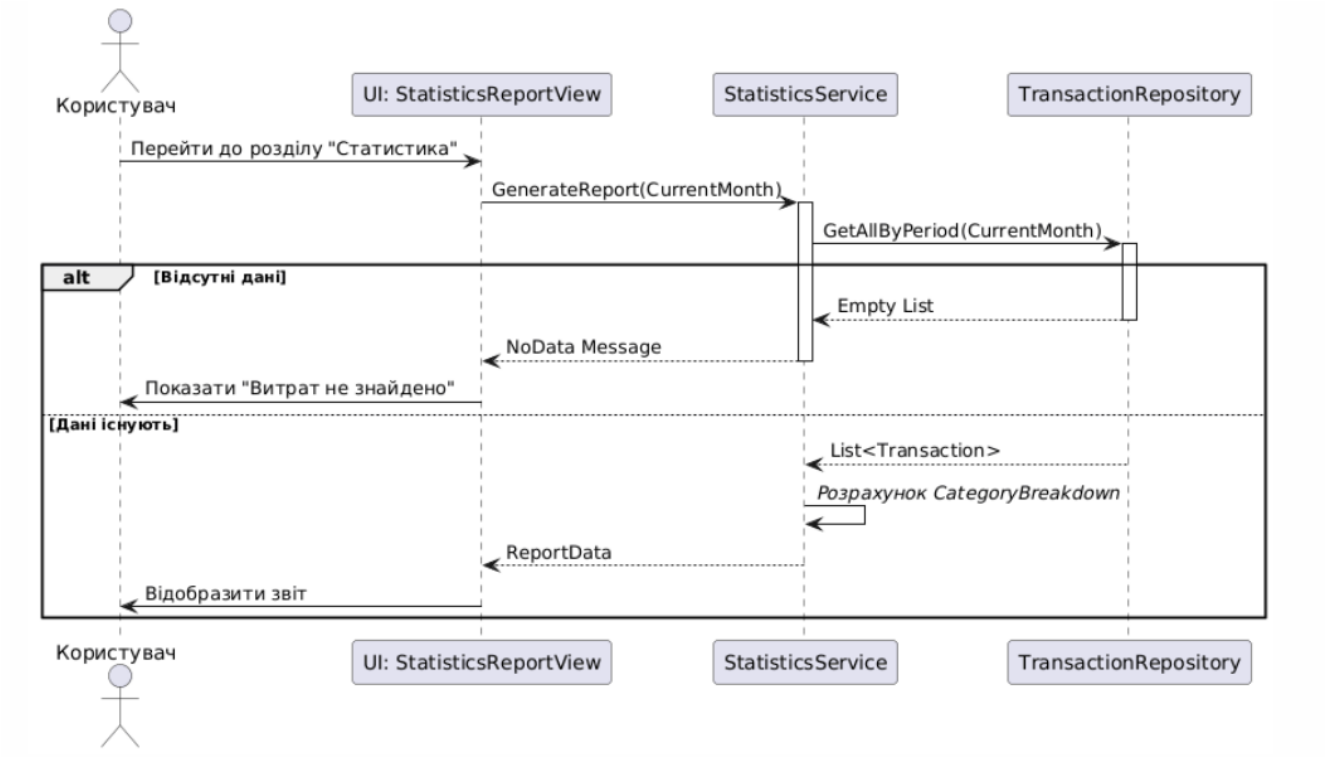


Рисунок 4 – Діаграма послідовності «Перегляд статистикаи витрат»

Опис сценарію №2: Перегляд статистики витрат

Користувач відкриває розділ «Статистика» у інтерфейсі StatisticsReportView.

1. UI викликає метод GenerateReport(CurrentMonth) у StatisticsService, щоб сформувати статистичні дані за поточний місяць.
2. StatisticsService звертається до TransactionRepository, виконуючи запит GetAllByPeriod(CurrentMonth) — отримати всі витрати за визначений період.

Гілка 1: Дані відсутні

4. Якщо репозиторій повертає Empty List, сервіс визначає, що транзакцій за обраний період немає.
5. StatisticsService відправляє у UI повідомлення NoData Message про те, що дані відсутні.
6. UI відображає користувачу повідомлення:
   «Витрат не знайдено».

Гілка 2: Дані існують

4. Якщо репозиторій повертає List<Transaction>, сервіс отримує список витрат за період.
5. StatisticsService виконує Розрахунок CategoryBreakdown — групує транзакції за категоріями, підраховує суми та формує статистичну модель для відображення.
6. Сервіс передає у UI підготовлений ReportData.
7. UI формує та відображає звіт, показуючи статистику витрат за місяць користувачу.

Контрольні питання

1. **Що собою становить діаграма розгортання?**

   Діаграма розгортання (*Deployment Diagram*) представляє фізичне розташування системи, показуючи, на якому обладнанні запускається програмне забезпечення.

2. **Які бувають види вузлів на діаграмі розгортання?**

   Вузли бувають двох типів:
   Пристрій (*device*) - фізичне обладнання (наприклад, комп'ютер) та
   Середовище виконання (*execution environment*) - програмне забезпечення (наприклад, операційна система чи вебсервер).

3. **Які бувають зв’язки на діаграмі розгортання?**

   Зв'язки зображують у вигляді прямої лінії з можливою **назвою**, що вказує на протокол (HTTP, IPC) або технологію зв'язку, а також атрибутами множинності.

4. **Які елементи присутні на діаграмі компонентів?**

На діаграмі компонентів присутні компоненти (окремі модулі) та залежності між ними. Компоненти можуть бути фізичними файлами (наприклад, .exe та .dll).

5. **Що становлять собою зв’язки на діаграмі компонентів?**
   Зв'язки на діаграмі компонентів показують залежності між ними, ілюструючи, що класи з одного компонента використовують класи з іншого компонента.

6. **Які бувають види діаграм взаємодії?**
   Одним із типів діаграм взаємодії є діаграма послідовностей (*Sequence Diagram*).

7. **Для чого призначена діаграма послідовностей?**
   Діаграма послідовностей (*Sequence Diagram*) призначена для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає порядок і логіку виконання операцій, показуючи, як об'єкти обмінюються повідомленнями.

8. **Які ключові елементи можуть бути на діаграмі послідовностей?**
   Ключові елементи: Актори, Об'єкти/Класи (з лінією життя), Повідомлення, Активності та Контрольні структури (наприклад, блоки "alt" або "loop").

9. **Як діаграми послідовностей пов’язані з діаграмами варіантів використання?**
   Діаграми послідовностей розробляються на основі сценаріїв, які, своєю чергою, описують варіанти використання.

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

Діаграми послідовностей моделюють динамічну взаємодію об'єктів (екземплярів класів), структуру яких визначено діаграмами класів.

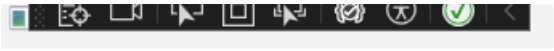
Висновки: Під час виконання лабораторної роботи я навчилася проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі. Також на основі спроєктованих діаграм розгортання та компонентів доопрацювала програмну частину системи. Реалізація системи, додатково до попередньої реалізації, містить чотири візуальні форми. В системі вже 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).

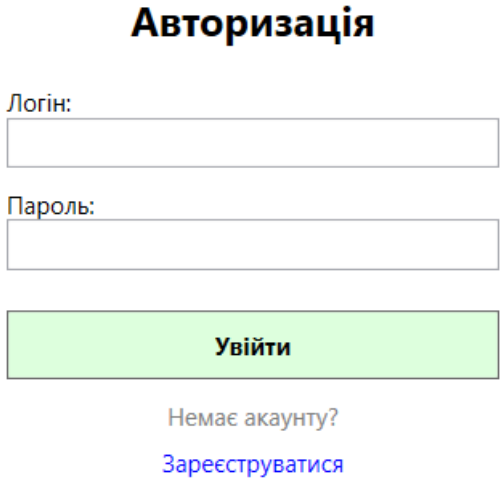
Репозиторій: посилання

ДОДАТКИ

Додаток А

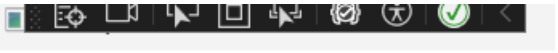
Форма авторизації користувача

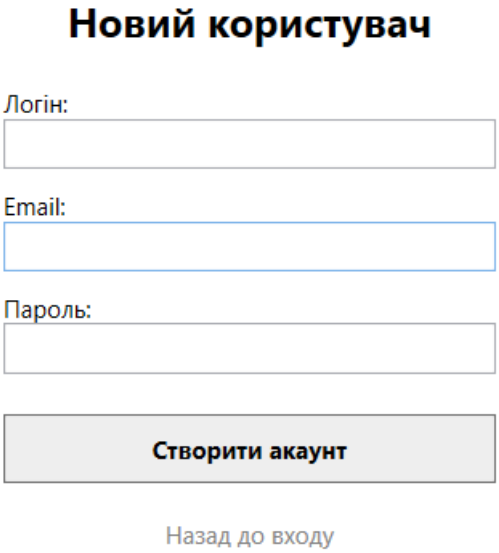




Додаток Б

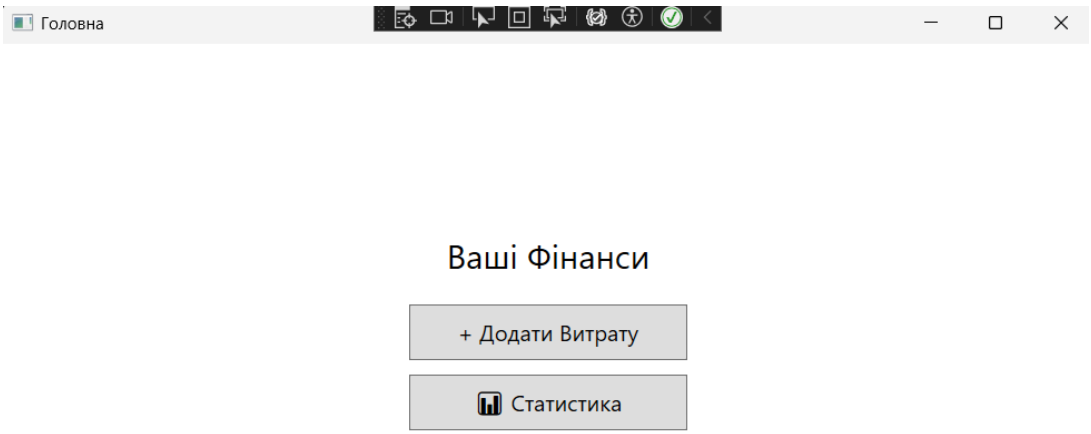
Форма реєстрації користувача





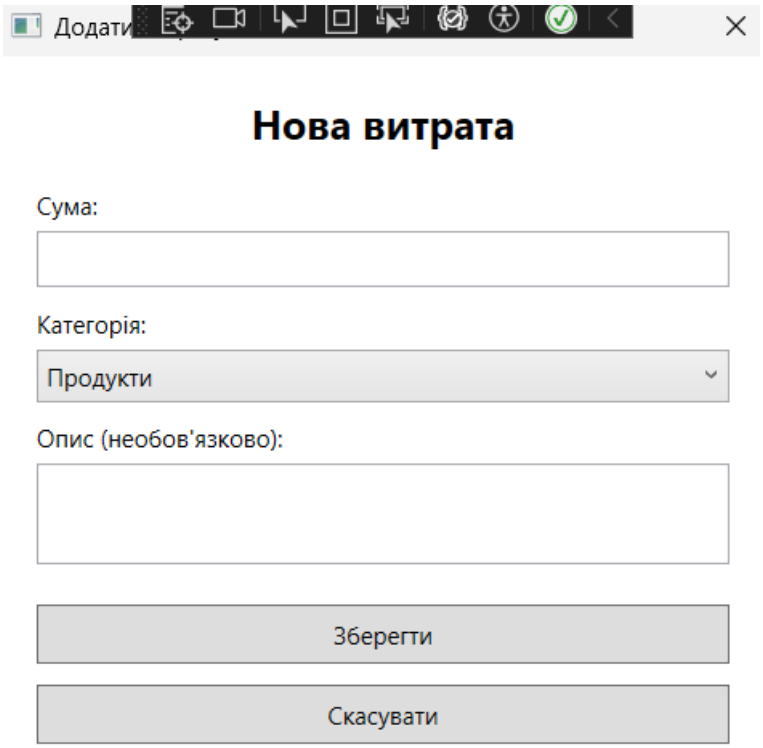
Додаток В

Головне вікно програми з меню навігації



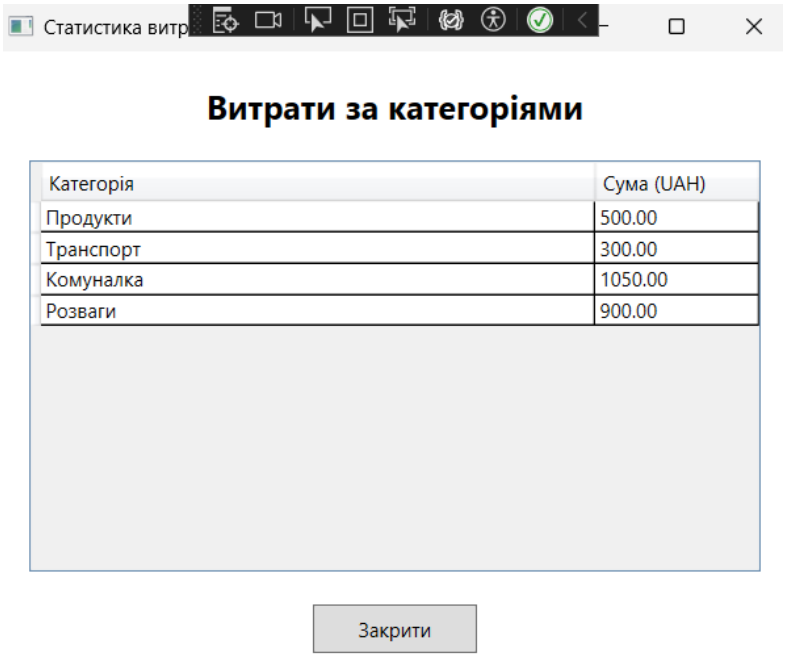
Додаток Г

Процес додавання нової витрати



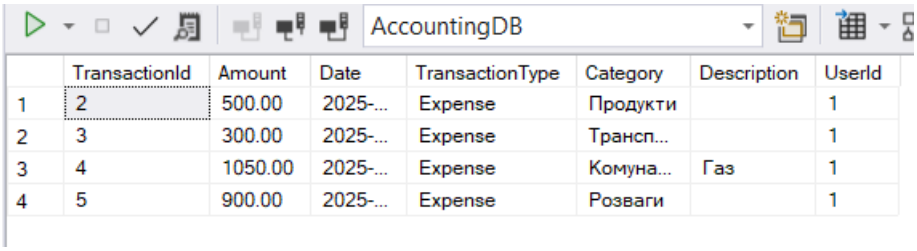
Додаток Ґ

Вікно перегляду статистики витрат за категоріями



Додаток Д

Перевірка збережених транзакцій у базі даних



	TransactionId	Amount	Date	TransactionType	Category	Description	UserId
1	2	500.00	2025-...	Expense	Продукти		1
2	3	300.00	2025-...	Expense	Трансп...		1
3	4	1050.00	2025-...	Expense	Комуна...	Газ	1
4	5	900.00	2025-...	Expense	Розваги		1