Міністерство освіти і науки України

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"

Виконав

студент групи ІА–33:

Китченко Д.В.

Перевірив:

Мягкий М.Ю.

Київ 2025

Зміст

Вступ

Мета роботи

Метою даної лабораторної роботи є набуття практичних навичок проектування діаграм розгортання та компонентів для системи, що розробляється, а також розробка діаграм взаємодії (діаграм послідовностей) на основі визначених сценаріїв використання. Додатково, необхідно доопрацювати програмну частину системи з реалізацією повного циклу роботи з даними від введення користувачем до збереження в базі даних та відображення результатів в графічному інтерфейсі.

Завдання роботи

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Ознайомитися з теоретичними відомостями щодо діаграм розгортання, компонентів та послідовностей в нотації UML.
2. Проаналізувати діаграми, створені в попередніх лабораторних роботах, та розробити діаграму розгортання для системи "Архіватор".
3. Розробити діаграму компонентів, що відображає модульну структуру системи та залежності між компонентами.
4. Створити мінімум дві діаграми послідовностей для основних сценаріїв використання системи.
5. На основі спроектованих діаграм доопрацювати програмну частину з реалізацією:
    - мінімум двох візуальних форм (GUI)
    - повного циклу роботи з даними (введення → обробка → збереження в БД → виборка → відображення)
6. Підготувати звіт з описом всіх створених діаграм та вихідним кодом доданого функціоналу.

Теоретичні відомості

Діаграма розгортання (Deployment Diagram)

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

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, веб-сервер).

Між вузлами можуть стояти зв'язки, які зазвичай зображують у вигляді прямої лінії. У зв'язків можуть бути атрибути множинності (для показання, наприклад, підключення двох і більше клієнтів до одного сервера) і назва. У назві, як правило, міститься спосіб зв'язку між двома вузлами – це може бути назва протоколу (HTTP, JDBC, TCP/IP) або технологія, що використовується для забезпечення взаємодії вузлів.

Вузли можуть містити артефакти (artifacts), які є фізичним уособленням програмного забезпечення; зазвичай це файли. Такими файлами можуть бути:

- виконувані файли (файли .exe, .jar, .dll)
- конфігураційні файли
- файли даних
- бази даних
- HTML-документи

Діаграми розгортання розрізняють двох видів: описові та екземплярні. На діаграмах описової форми вказуються вузли, артефакти і зв'язки між вузлами

без вказівки конкретного обладнання. Діаграми екземплярної форми несуть у собі екземпляри обладнання з конкретними характеристиками.

Діаграма компонентів

Діаграма компонентів UML є представленням проектованої системи, розбитої на окремі модулі. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів:

- Логічні – система уявляється як набір самостійних, автономних модулів, що взаємодіють між собою.
- Фізичні – показують компоненти та залежності між ними. Залежності показують, що класи з одного компонента використовують класи з іншого компонента.
- Виконувані – кожен компонент являє собою файл (виконувані файли, файли вихідних кодів, сторінки HTML, бази даних).

Діаграма компонентів розробляється для таких цілей:

- візуалізації загальної структури вихідного коду програмної системи
- специфікації виконуваного варіанта програмної системи
- забезпечення багаторазового використання окремих фрагментів програмного коду
- представлення концептуальної та фізичної схем баз даних

Діаграми послідовностей

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML, 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.

Діаграма складається з таких основних елементів:

- Актори (Actors) – користувачі чи інші системи, які взаємодіють із системою.
- Об'єкти або класи – розміщуються горизонтально на діаграмі. Кожен об'єкт має життєвий цикл, який представлений вертикальною пунктирною лінією (лінія життя).

- Повідомлення – лінії зі стрілками, які з'єднують об'єкти. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником).
- Активності – вказують періоди, протягом яких об'єкт виконує певну дію. Позначається прямокутником, накладеним на лінію життя.
- Контрольні структури – блоки "alt" (альтернатива), "loop" (цикл), "opt" (опційно) для відображення умов та циклів.

Основні кроки створення діаграми послідовностей:

1. Визначити акторів і об'єкти, які беруть участь у сценарії
2. Побудувати їхні лінії життя
3. Розробити послідовність передачі повідомлень між об'єктами
4. Додати умовні блоки або цикли за необхідності

Хід роботи

Діаграма розгортання системи "Архіватор"

На основі аналізу вимог до системи "Архіватор" була спроектована діаграма розгортання за клієнт-серверною моделлю (рис. 3.1).

Опис архітектури:

Система складається з двох основних фізичних вузлів, що взаємодіють через мережу TCP/IP.

Клієнтський вузол (<<device>> Клієнтський ПК):

Представлений комп'ютером користувача з операційною системою Windows, Linux або macOS (<<execution environment>> ОС: Windows / Linux / macOS). На клієнтській стороні функціонує середовище виконання JVM (<<execution environment>> JVM), в якому запускається основний виконуваний артефакт Archiver.jar.

Додатково на клієнті присутні допоміжні артефакти:

- postgresql-driver.jar – JDBC драйвер для з'єднання з базою даних PostgreSQL
- config.properties – конфігураційний файл з параметрами підключення до БД

Клієнт має доступ до локальної файлової системи (позначено як "Локальні архіви (*.zip)") для створення, читання та модифікації архівних файлів. Взаємодія з файловою системою здійснюється через механізми File I/O операційної системи.

Серверний вузол (<<device>> Сервер баз даних):

Представлений виділеним сервером з операційною системою Linux (<<execution environment>> ОС: Linux Server). На серверній стороні розгорнуто середовище виконання PostgreSQL Server (<<execution environment>> PostgreSQL Server), яке управляє базою даних ArchiverDB.

База даних містить чотири основні таблиці:

- archive_info – інформація про архіви
- operation_log – журнал операцій

- operationdetail – деталі операцій
- archive_bookmark – закладки в архівах

Мережева взаємодія:

З'єднання між клієнтським та серверним вузлами здійснюється через мережу TCP/IP за протоколом JDBC (порт 5432). Клієнт ініціює з'єднання з базою даних для виконання SQL-запитів (SELECT, INSERT, UPDATE, DELETE). Зв'язок є односпрямованим від клієнта до сервера, що відповідає архітектурі клієнт-серверних додатків.

Переваги такої архітектури:

- Централізоване зберігання даних на сервері
- Можливість роботи декількох клієнтів одночасно
- Незалежність клієнтської та серверної частин
- Легке масштабування системи

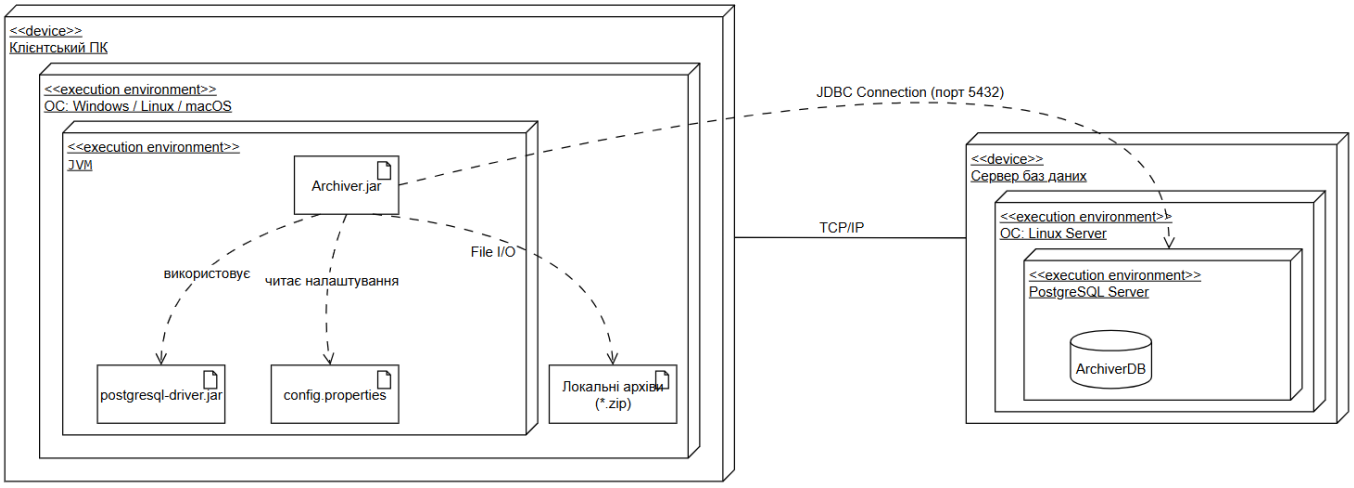


Рисунок 3.1 – Діаграма розгортання системи "Архіватор"

Діаграма компонентів системи

Діаграма компонентів відображає логічну структуру системи "Архіватор" та залежності між модулями (рис. 3.2). Система розділена на три основні частини згідно з багатошаровою архітектурою.

Client Component (Клієнтські компоненти):

Містить три основні модулі:

1. UI Module – модуль графічного інтерфейсу користувача. Реалізований з використанням бібліотеки Swing та включає:
    - SimpleMainWindow – головне вікно зі списком архівів
    - OperationHistoryWindow – вікно перегляду історії операцій
    - Діалогові вікна для введення даних
2. Archiver Core Module – модуль основної бізнес-логіки системи. Включає:
    - ArchiverFacade – фасад для спрощення взаємодії з підсистемами
    - Strategy паттерн (IArchiverStrategy, ZipStrategy)
    - Factory Method (ArchiverStrategyFactory)
    - Координацію роботи між UI та Repository шарами
3. Domain Entities – модуль доменних моделей. Містить класи предметної області:
    - Archive, ArchiveInfo, ArchiveItem
    - ArchivedFile, ArchivedFolder
    - OperationLog, OperationDetail, ArchiveBookmark
    - Енумерації: ArchiveFormat, OperationType, OperationStatus

Data Contract (Контракт даних):

Визначає загальні інтерфейси та схему даних для взаємодії між шарами:

1. DTO Classes – класи передачі даних між шарами. Використовуються для серіалізації та передачі інформації.
2. Database Schema – опис структури таблиць бази даних. Визначає структуру даних для збереження в PostgreSQL.

Server Component (Серверні компоненти):

Містить два модулі:

1. Repository Module – модуль доступу до даних. Включає:
    - Інтерфейси репозиторіїв (IArchiveInfoRepository, IOperationLogRepository, IOperationDetailRepository, IArchiveBookmarkRepository)
    - PostgreSQL реалізації (PostgresArchiveInfoRepository,

PostgresOperationLogRepository, PostgresOperationDetailRepository, PostgresArchiveBookmarkRepository)

2. Database Service – сервіс управління з'єднанням з БД:
    - DatabaseConnector – клас для створення та управління з'єднаннями
    - Завантаження конфігурації з config.properties
    - Пул з'єднань

Залежності між компонентами:

- UI Module залежить від Archiver Core Module (пунктирна стрілка)
- Archiver Core Module використовує Domain Entities
- Archiver Core Module залежить від DTO Classes для передачі даних
- Repository Module залежить від DTO Classes
- Database Service використовує Database Schema для роботи з таблицями
- Repository Module залежить від Database Service для отримання з'єднань

Така архітектура забезпечує:

- Слабку зв'язаність між компонентами
- Можливість незалежного тестування модулів
- Легку заміну реалізацій (наприклад, зміна БД з PostgreSQL на MySQL)
- Дотримання принципу Single Responsibility

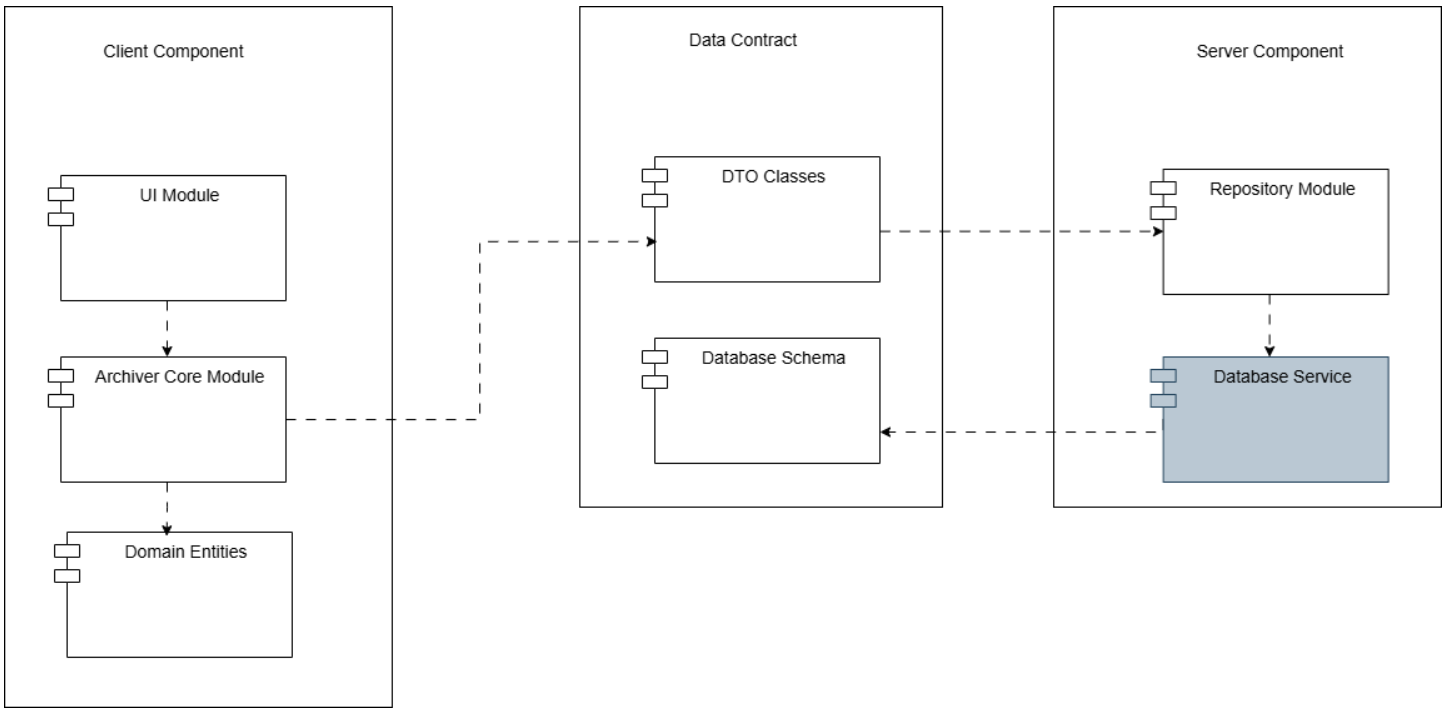


Рисунок 3.2 – Діаграма компонентів системи "Архіватор"

Діаграма послідовності "Створення архіву"

Діаграма описує процес створення нового архіву користувачем (рис. 3.3).

Учасники взаємодії:

- User – користувач системи
- UI – графічний інтерфейс (SimpleMainWindow)
- BusinessLogic – бізнес-логіка (ArchiverFacade + ZipStrategy)
- DataBase – база даних PostgreSQL

Опис основного сценарію:

1. Користувач натискає кнопку "Створити" в головному вікні.
2. UI відкриває діалогове вікно для вибору файлів та папок.
3. Користувач обирає елементи для архівування та натискає "Далі".
4. UI показує діалог збереження, де користувач вказує ім'я архіву, формат (ZIP/TAR.GZ/RAR/ACE), місце збереження та рівень стиснення.
5. Користувач натискає "Зберегти".
6. UI передає команду створення архіву до BusinessLogic з усіма параметрами.
7. BusinessLogic виконує валідацію (перевірка шляху, прав доступу, вільного місця на диску).
8. При успішній валідації BusinessLogic створює архів та виконує транзакцію в базі даних:
    - Відкриває транзакцію (BEGIN)
    - Додає запис про архів в таблицю archive_info
    - Додає запис про операцію в таблицю operation_log
    - Фіксує транзакцію (COMMIT)
9. BusinessLogic повертає UI повідомлення про успішне завершення.
10. UI показує користувачу повідомлення "Архів успішно створено".

Альтернативні сценарії:

- Скасування операції: Користувач може натиснути "Скасувати" на етапах вибору файлів або збереження. Операція перерветься без змін у файловій системі.

- Помилка валідації: Якщо система виявляє проблеми (недостатньо місця, немає прав доступу, файл існує, формат не підтримується), показується відповідне повідомлення про помилку та операція переривається.
- Помилка транзакції: При виникненні помилки під час роботи з базою даних автоматично виконується ROLLBACK для збереження цілісності даних.

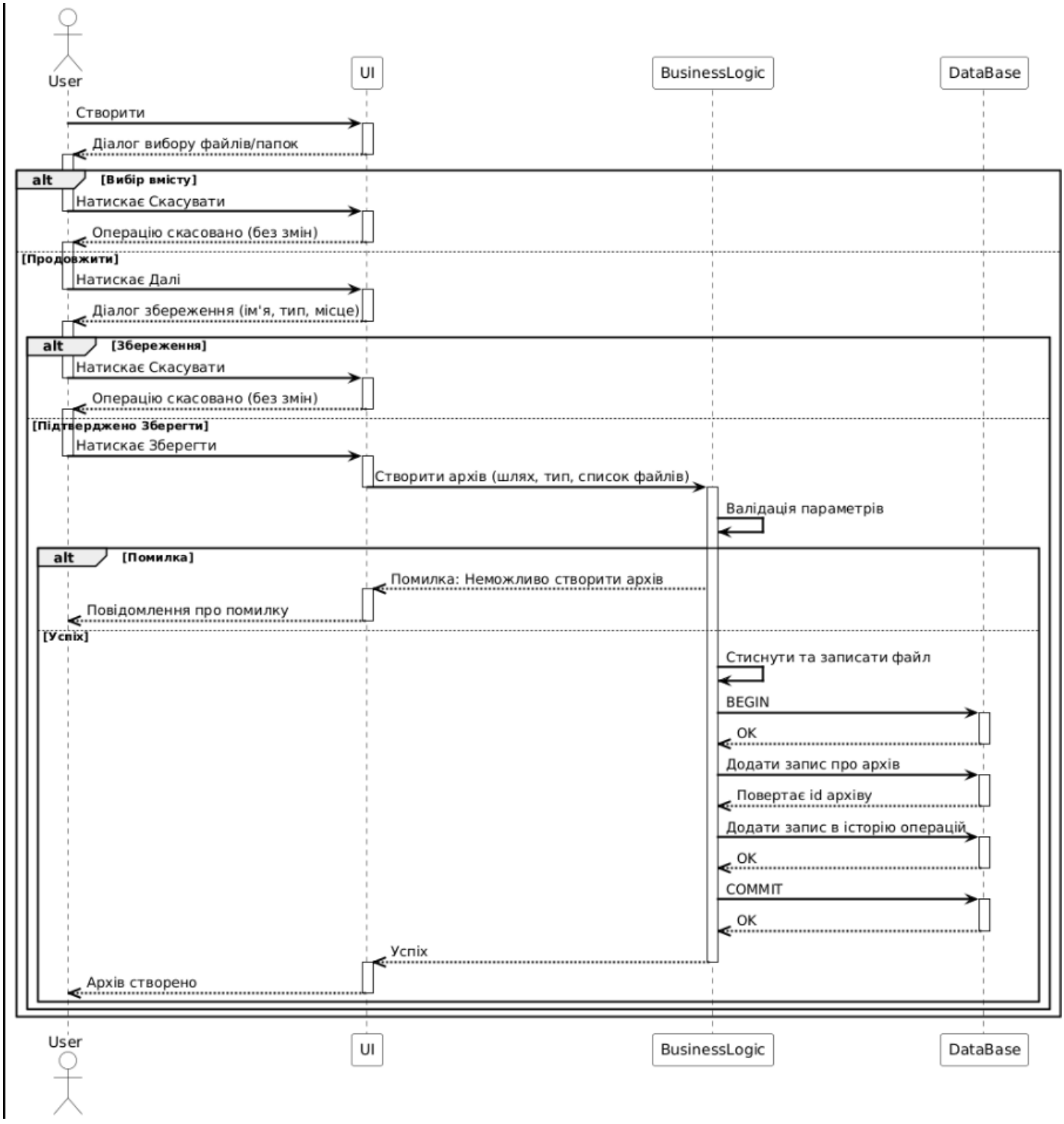


Рисунок 3.3 – Діаграма послідовності "Створення архіву"

Діаграма послідовності "Перевірка контрольної суми"

Діаграма описує процес перевірки цілісності файлів в архіві через обчислення та порівняння контрольних сум (рис. 3.4).

Учасники взаємодії:

- User – користувач системи
- UI – графічний інтерфейс
- BusinessLogic – бізнес-логіка
- DataBase – база даних

Опис основного сценарію:

1. Користувач виділяє один або кілька файлів у вмісті відкритого архіву.
2. Натискає кнопку "Перевірити контрольну суму".
3. UI передає запит до BusinessLogic та показує індикатор виконання операції.
4. BusinessLogic циклічно обробляє кожен вибраний файл:
    - Зчитує дані файлу з архіву через DataBase
    - Обчислює контрольну суму (MD5/SHA-256)
    - Запитує еталонне значення checksum з БД
    - Порівнює обчислену та еталонну суму (якщо еталон є)
    - Повертає результат для файлу: OK, Невідповідність, або тільки обчислену суму
5. Після обробки всіх файлів BusinessLogic повертає загальний результат.
6. UI показує підсумковий список файлів зі статусами та закриває індикатор.

Альтернативні сценарії:

- Скасування операції: Користувач може натиснути "Скасувати" під час перевірки. Операція переривається, результати не зберігаються.
- Помилка читання файлу: Якщо файл пошкоджений або немає прав доступу, він пропускається з відповідним повідомленням. Обробка інших файлів продовжується.

- Еталон відсутній: Якщо в БД немає еталонного значення, система показує тільки обчислену контрольну суму без порівняння.
- Еталон невірний: Якщо еталонне значення некоректне, система пропонує вказати інший еталон або продовжити без порівняння.

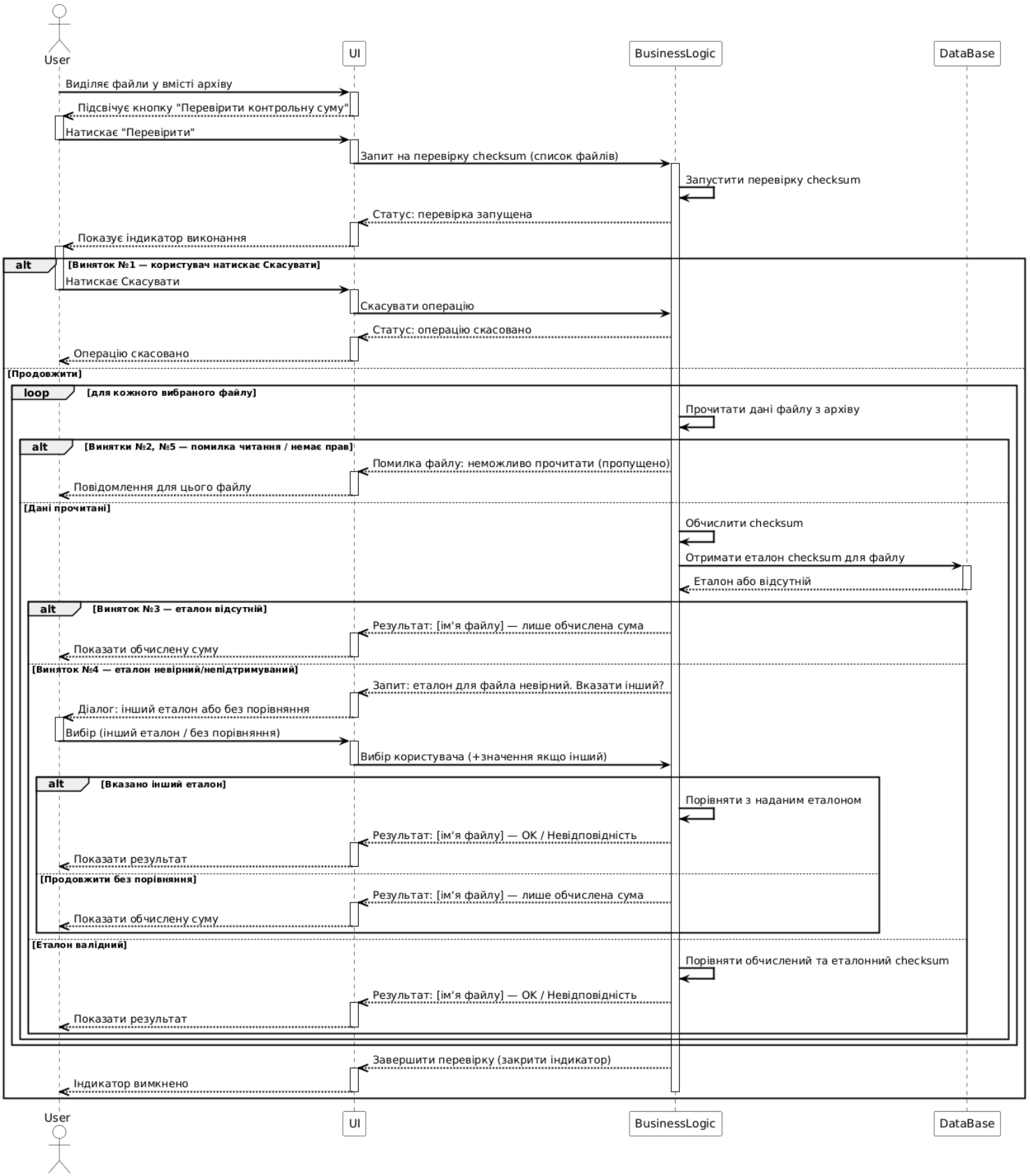


Рисунок 3.4 – Діаграма послідовності "Перевірка контрольної суми"

Реалізація програмної частини

На основі спроектованих діаграм розгортання та компонентів було доопрацьовано програмну частину системи "Архіватор". Реалізація включає повний цикл роботи з даними від введення на формі до збереження в базі даних та відображення результатів.

Система реалізована за багатошаровою архітектурою з чітким розподілом відповідальності:

Presentation Layer (Шар представлення):

- SimpleMainWindow.java – головне вікно програми з таблицею архівів
- OperationHistoryWindow.java – вікно перегляду історії операцій

Business Logic Layer (Шар бізнес-логіки):

- ArchiverFacade.java – фасад для координації всіх операцій системи
- IArchiverStrategy.java – інтерфейс стратегії архівування
- ZipStrategy.java – реалізація стратегії для ZIP формату
- ArchiverStrategyFactory.java – фабрика для створення стратегій

Data Access Layer (Шар доступу до даних):

- Інтерфейси репозиторіїв: IArchiveInfoRepository, IOperationLogRepository, IOperationDetailRepository, IArchiveBookmarkRepository
- PostgreSQL реалізації: PostgresArchiveInfoRepository, PostgresOperationLogRepository, PostgresOperationDetailRepository, PostgresArchiveBookmarkRepository
- DatabaseConnector.java – управління з'єднаннями з БД

Domain Layer (Доменний шар):

- Model класи: Archive, ArchiveInfo, OperationLog, OperationDetail, ArchiveBookmark
- Енумерації: ArchiveFormat, OperationType, OperationStatus

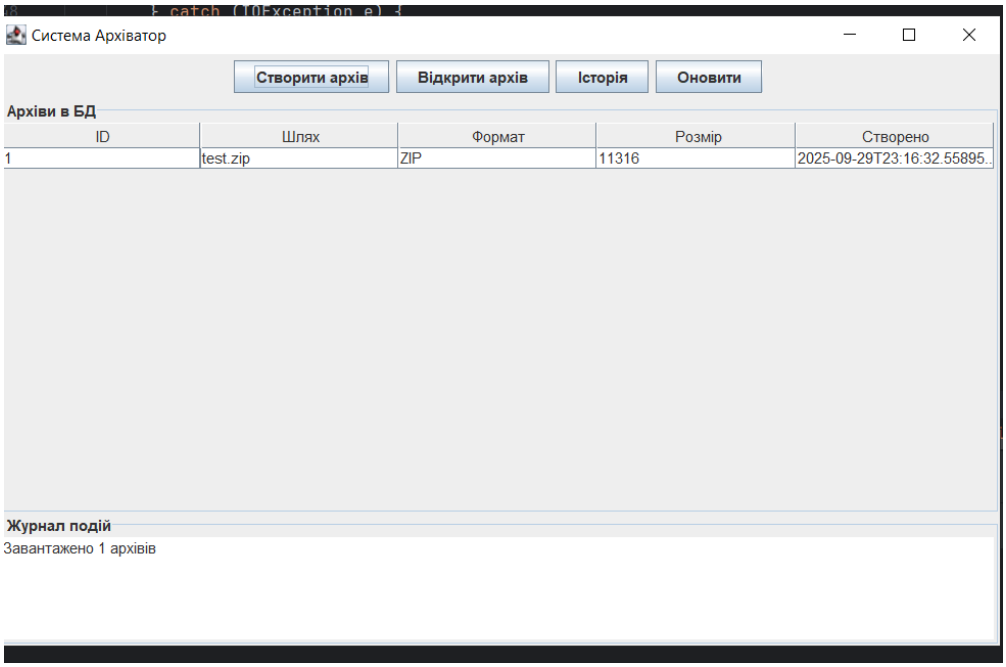
Форма №1: SimpleMainWindow (Головне вікно)

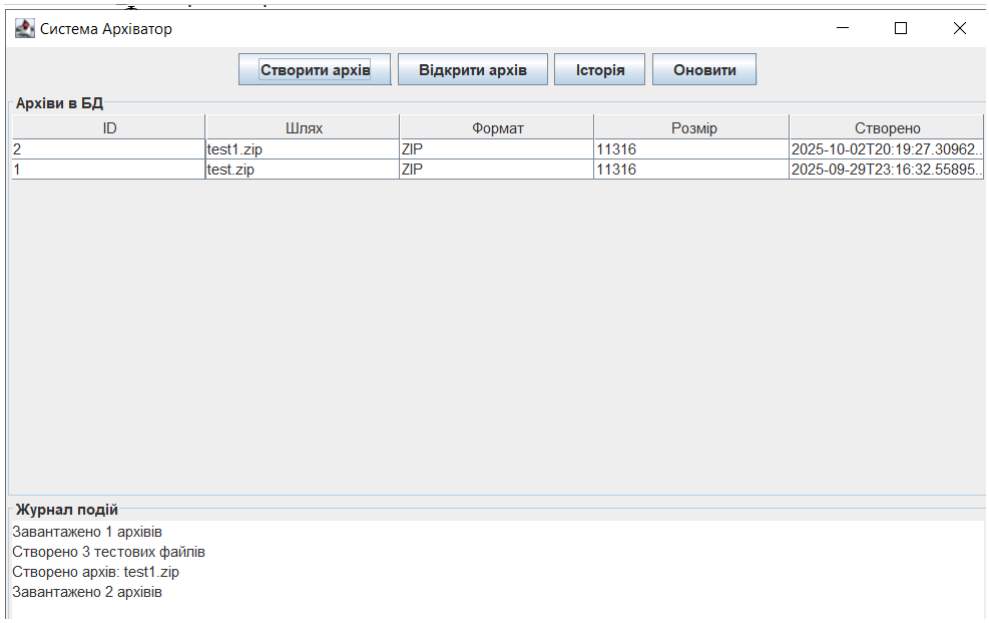
Головне вікно системи містить наступні елементи:

- Панель кнопок управління: "Створити архів", "Відкрити архів", "Історія", "Оновити"
- Таблицю JTable з колонками: ID, Шлях, Формат, Розмір, Створено
- Текстову область JTextArea для відображення журналу подій

Функціональність:

- Відображення списку архівів з бази даних
- Створення нових архівів через діалогові вікна
- Відкриття існуючих архівів
- Перехід до вікна історії операцій
- Оновлення даних з БД





	id [PK] integer	file_path character varying (500)	format character varying (50)	total_size bigint	created_at timestamp without time zone	last_accessed_at timestamp without time zone
1	1	test.zip	ZIP	11316	2025-09-29 23:16:32.55895	2025-09-29 23:41:03.782564
2	2	test1.zip	ZIP	11316	2025-10-02 20:19:27.309625	2025-10-02 20:20:06.968761

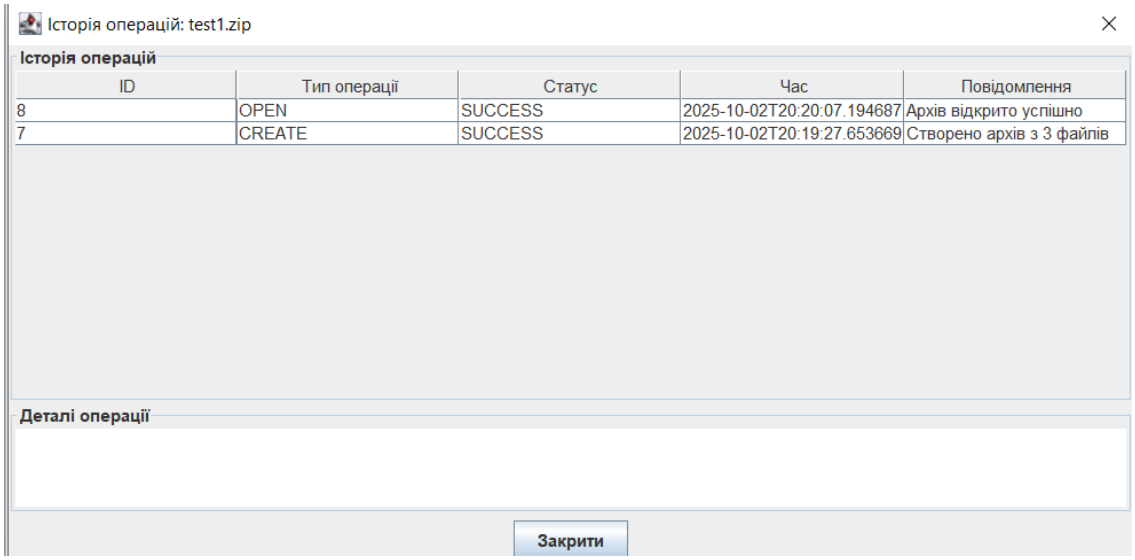
Форма №2: OperationHistoryWindow (Вікно історії операцій)

Вікно історії операцій містить:

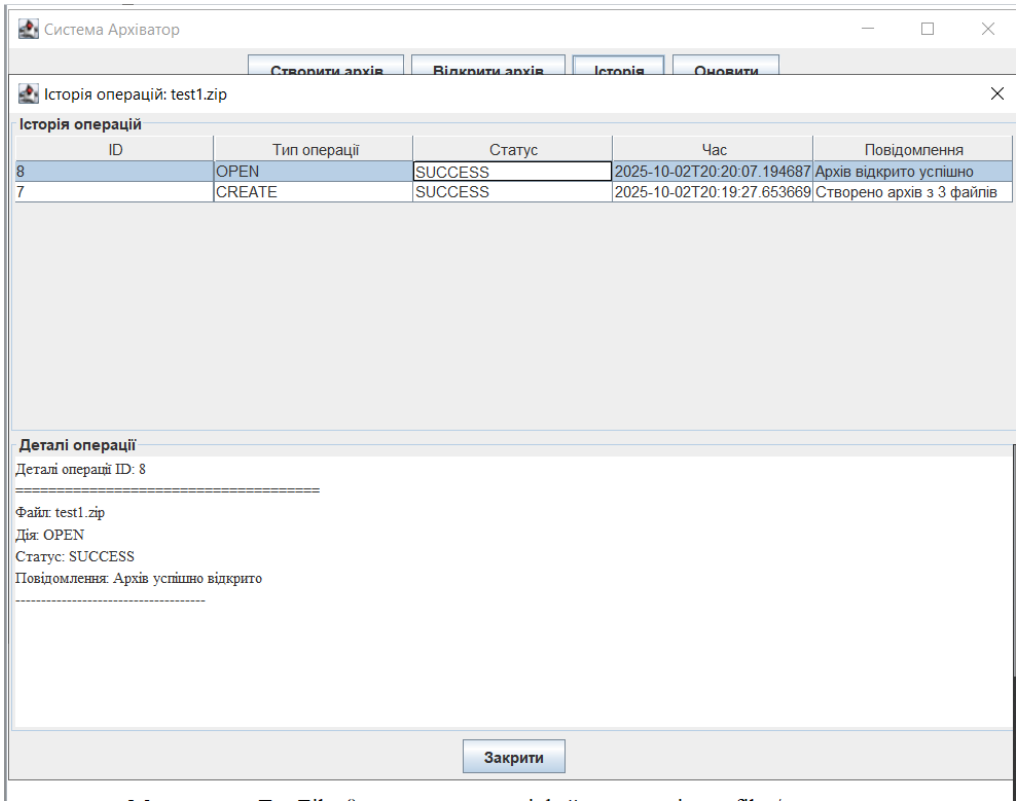
- Таблицю JTable з історією: ID, Тип операції, Статус, Час, Повідомлення
- Текстову область JTextArea для відображення деталей операції
- Кнопку "Закрити"

Функціональність:

- Завантаження історії операцій для вибраного архіву
- Відображення деталей операції при виборі запису
- Перегляд інформації про кожен оброблений файл



	id [PK] integer	archive_id bigint	operation_type character varying (50)	status character varying (50)	timestamp timestamp without time zone	details text
1	1	1	CREATE	SUCCESS	2025-09-29 23:16:32.693951	Створено архів з 3 файлів
2	2	1	OPEN	SUCCESS	2025-09-29 23:16:36.03237	Архів відкрито успішно
3	3	1	OPEN	SUCCESS	2025-09-29 23:16:37.938679	Архів відкрито успішно
4	4	1	CREATE	SUCCESS	2025-09-29 23:25:32.550249	Створено архів з 3 файлів
5	5	1	OPEN	SUCCESS	2025-09-29 23:25:38.053861	Архів відкрито успішно
6	6	1	OPEN	SUCCESS	2025-09-29 23:41:04.023565	Архів відкрито успішно
7	7	2	CREATE	SUCCESS	2025-10-02 20:19:27.653669	Створено архів з 3 файлів
8	8	2	OPEN	SUCCESS	2025-10-02 20:20:07.194687	Архів відкрито успішно



	detailid [PK] integer	logid bigint	itempath character varying (500)	action character varying (50)	status character varying (50)	message text
1	1	1	file1.txt	CREATE	SUCCESS	Файл додано до архіву
2	2	1	file2.txt	CREATE	SUCCESS	Файл додано до архіву
3	3	1	file3.txt	CREATE	SUCCESS	Файл додано до архіву
4	4	2	test.zip	OPEN	SUCCESS	Архів успішно відкрито
5	5	3	test.zip	OPEN	SUCCESS	Архів успішно відкрито
6	6	4	D:\IdeaProjects\untitled11\test_files\test_file1.t...	CREATE	SUCCESS	Файл додано до архіву
7	7	4	D:\IdeaProjects\untitled11\test_files\test_file2.t...	CREATE	SUCCESS	Файл додано до архіву
8	8	4	D:\IdeaProjects\untitled11\test_files\test_file3.t...	CREATE	SUCCESS	Файл додано до архіву
9	9	5	test.zip	OPEN	SUCCESS	Архів успішно відкрито
10	10	6	test.zip	OPEN	SUCCESS	Архів успішно відкрито
11	11	7	D:\IdeaProjects\untitled11\test_files\test_file1.t...	CREATE	SUCCESS	Файл додано до архіву
12	12	7	D:\IdeaProjects\untitled11\test_files\test_file2.t...	CREATE	SUCCESS	Файл додано до архіву
13	13	7	D:\IdeaProjects\untitled11\test_files\test_file3.t...	CREATE	SUCCESS	Файл додано до архіву
14	14	8	test1.zip	OPEN	SUCCESS	Архів успішно відкрито

Система забезпечує наступний повний цикл роботи з даними:

1. Введення даних користувачем:

String archiveName = JOptionPane.showInputDialog("Введіть назву архіву:");

Користувач вводить назву архіву через стандартний діалог Swing.

2. Створення тестових файлів:

```
private List<String> createTestFiles() {
    File testDir = new File("test_files");
    testDir.mkdirs();
```

```
    // Створення 3 реальних файлів з текстовим вмістом
}
```

Метод createTestFiles() створює реальні файли в папці test_files/ для подальшого архівування.

3. Архівування файлів:

```
Archive archive = facade.create(archiveName, ArchiveFormat.ZIP, files);
```

ZipStrategy створює реальний ZIP архів, стискаючи файли за допомогою ZipOutputStream.

4. Збереження в базі даних:

```
// Збереження інформації про архів
archiveRepo.upsert(newInfo);
// Збереження операції
long logId = logRepo.add(logEntry);
// Збереження деталей для кожного файлу
for (String filePath : files) {
    OperationDetail detail = new OperationDetail(...);
    detailRepo.add(detail);
}
```

Дані зберігаються в трьох таблицях PostgreSQL: archive_info, operation_log, operationdetail.

5. Виборка даних з БД:

```
List<ArchiveInfo> archives = facade.listRecentArchives(20);
```

Метод завантажує список останніх 20 архівів з бази даних.

6. Відображення на UI:

```
java
for (ArchiveInfo info : archives) {
    Object[] row = {info.getId(), info.getFilePath(),
                    info.getFormat(), info.getTotalSize(),
                    info.getCreatedAt()};
    tableModel.addRow(row);
```

}

Дані відображаються в JTable головного вікна.\

Вихідний код системи

Нижче наведено ключові фрагменти вихідного коду, доданого в цій лабораторній роботі.

ArchiverFacade

SimpleMainWindow

ZipStrategy

OperationHistoryWindow

PostgresArchiveBookmarkRepository

PostgresArchiveInfoRepository

PostgresOperationDetailRepository

PostgresOperationLogRepository

DatabaseConnector

Питання до лабораторної роботи

1. Що собою становить діаграма розгортання?

Діаграма розгортання - це UML діаграма, яка показує фізичне розташування системи, тобто на якому обладнанні запускається програмне забезпечення. Вона відображає вузли (комп'ютери, сервери), артефакти (файли програм) та зв'язки між ними.

2. Які бувають види вузлів на діаграмі розгортання?

Вузли бувають двох типів:

Пристрій (device) - фізичне обладнання (комп'ютер, сервер, принтер)

Середовище виконання (execution environment) - програмне забезпечення, яке може містити інше ПЗ (операційна система, JVM, веб-сервер, PostgreSQL Server)

3. Які бувають зв'язки на діаграмі розгортання?

Зв'язки зображуються прямими лініями між вузлами. Вони можуть мати:

Назву протоколу або технології (HTTP, TCP/IP, JDBC, .NET Remoting)

Атрибути множинності (показують кількість підключень, наприклад, "2..*" клієнтів до одного сервера)

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів присутні:

Компоненти (модулі системи) - прямокутники з іконкою компонента

Залежності між компонентами - пунктирні стрілки

Інтерфейси - "льодяники" або прямокутники з <<interface>>

Пакети (packages) - для групування компонентів

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

Зв'язки на діаграмі компонентів - це залежності, які показують, що класи з одного компонента використовують класи з іншого компонента. Вони відображаються пунктирними стрілками і показують, які зміни в одному компоненті вплинуть на інші компоненти.

6. Які бувають види діаграм взаємодії?

Діаграми взаємодії бувають двох видів:

Діаграма послідовностей (Sequence Diagram) - показує взаємодію об'єктів у часовій послідовності

Діаграма комунікації (Communication Diagram) - показує структурну організацію об'єктів та їх зв'язки

7. Для чого призначена діаграма послідовностей?

Діаграма послідовностей призначена для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона показує, як об'єкти обмінюються повідомленнями, порядок і логіку виконання операцій. Використовується для моделювання бізнес-процесів, проектування архітектури та тестування.

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Ключові елементи діаграми послідовностей:

Актори - зовнішні користувачі або системи

Об'єкти/класи - учасники взаємодії

Лінії життя - вертикальні пунктирні лінії під об'єктами

Повідомлення - стрілки між лініями життя (синхронні, асинхронні, повернення результату)

Активності - прямокутники на лініях життя, що показують виконання дій

Контрольні структури - блоки alt (альтернатива), loop (цикл), opt (опція)

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

Діаграми послідовностей деталізують сценарії з діаграм варіантів використання. Кожен варіант використання (use case) може мати одну або кілька діаграм послідовностей, які показують покрокову взаємодію об'єктів при виконанні цього варіанту. Діаграма послідовностей показує "як" виконується те, "що" описано в варіанті використання.

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

Діаграми послідовностей використовують об'єкти класів, що визначені на діаграмах класів. Повідомлення на діаграмі послідовностей відповідають методам класів з діаграми класів. Діаграма класів показує статичну структуру (які класи є, які методи), а діаграма послідовностей показує динаміку (як ці методи викликаються під час виконання).

Висновки

Під час виконання лабораторної роботи №3 я спроектував та реалізував систему "Архіватор" з використанням UML діаграм. Було створено діаграму розгортання, яка показує фізичну архітектуру з клієнтським комп'ютером (Java програма) та сервером бази даних PostgreSQL, з'єднаними через мережу. Діаграма компонентів демонструє структуру програми з розподілом на модулі інтерфейсу, бізнес-логіки та роботи з даними. Розроблено дві діаграми послідовностей для сценаріїв створення архіву та перевірки контрольної суми, які показують покрокову взаємодію між користувачем, інтерфейсом, логікою та базою даних. Програмна частина включає два вікна (головне з таблицею архівів та вікно історії операцій), реалізацію створення реальних ZIP архівів, збереження всієї інформації в PostgreSQL та відображення даних користувачу. Застосовано патерни проектування Strategy, Factory Method та Facade для гнучкої архітектури. Система має повний цикл роботи з даними від введення через діалоги до збереження в базі та показу в таблиці. Використання UML діаграм на етапі проектування допомогло спланувати архітектуру перед написанням коду, що спростило розробку та забезпечило якісну реалізацію системи.