Київський національний університет імені Тараса Шевченка

ЗВІТ

до Етапу 4. Порівняльний аналіз розробки
практичного завдання (модуль 1)
з дисципліни "Інформаційні технології"
на тему:

Веб-версія системи управління табличними базами даних

Виконав
Студент четвертого курсу
Групи ТТП-41
Факультету комп'ютерних наук та кібернетики
Назарій ЯГОТІН

Київ, 2025

Зміст

1 Вступ

1.1 Контекст роботи

У рамках практичного завдання з дисципліни "Інформаційні технології"було розроблено систему управління табличними базами даних у двох архітектурних парадигмах: десктоп-застосунок на WPF (Windows Presentation Foundation) та веб-орієнтовану версію на базі React та ASP.NET Core Web API.
Обидві реалізації базуються на спільній бізнес-логіці, винесеній у бібліотеку CoreLib, що забезпечує консистентність обробки даних, валідації та серіалізації незалежно від способу доставки функціональності.

1.2 Структура розділу

Розділ організовано таким чином:

1. Спільні елементи
2. Ключові відмінності
3. Переваги та недоліки
4. Порівняльні таблиці
5. Висновки та рекомендації

1.3 Обсяг реалізації

Обидві версії повністю реалізують задану функціональність:

- Базові типи: Integer, Real, Char, String
- Додаткові типи: IntegerInterval, TextFile
- Індивідуальна операція: перейменування та перестановка колонок
- Повний CRUD функціонал для таблиць та рядків
- Валідація даних на всіх рівнях
- Збереження та завантаження баз даних

2 Спільні елементи

2.1 Бізнес-логіка

Обидві версії використовують спільну бібліотеку **CoreLib**, яка містить:

- Моделі даних: `Database`, `Table`, `Column`
- Типи даних: базові та додаткові (`IntegerInterval`, `TextFile`)
- Сервіси: `DatabaseService`, `TableService`, `FileService`
- Валідацію рядків та структури БД
- Серіалізацію JSON з кастомними конвертерами

2.2 Функціональність

Повна реалізація заданої функціональності включає:

- Створення та завантаження БД
- Управління таблицями (CRUD операції)
- Роботу з усіма типами даних
- Валідацію даних при введенні
- Перейменування та перестановку колонок
- Роботу з текстовими файлами

2.3 Зберігання даних

Обидві версії використовують JSON як формат збереження:

- Структура БД: таблиці з колонками та значеннями
- Метадані файлів з усіма атрибутами
- System.Text.Json для серіалізації/десеріалізації

3 Ключові відмінності

3.1 Архітектура

Аспект	**Десктоп (WPF)**	**Веб (React + ASP.NET)**
Структура	Монолітна, однопроцесна	Розподілена клієнт-сервер
Управління сесіями	Одна БД за раз на ПК	Багато одночасних БД через сесії
Користувачі	Один користувач	Багато користувачів через `ISessionService`
Зберігання файлів і БД	Локальна файлова система	Сервер + локальна файлова система
Комунікація	Data Binding	REST API через HTTP

Табл. 1: Архітектурні відмінності

3.2 Управління даними

3.2.1 Десктоп-версія

- Одна активна БД на ПК
- Локальне виконання операцій
- Файли зберігаються у папках `uploads/` та `tempFiles/` на пристрої користувача
- Тимчасове сховище для безпечного завантаження

3.2.2 Веб-версія

- Багато БД одночасно (по одній на сесію)
- Ідентифікація сесії через GUID у заголовку `X-Session-Id`
- Управління екземплярами через `ISessionService` та `MemoryCache`
- Таймаут сесії: 120 хвилин
- Файли зберігаються на сервері у папках `uploads/` та `tempFiles/`

3.3 Технологічний стек

3.3.1 Спільна платформа

Обидві версії побудовані на .NET 9.0 та C# 14.0, що забезпечує:

- Сучасні можливості мови
- Високу продуктивність
- Кросплатформенність бізнес-логіки
- Єдиний підхід до серіалізації

Бібліотека CoreLib ідентична в обох проєктах.

3.3.2 Десктоп-версія: WPF екосистема

- **WPF** — фреймворк для desktop UI
- **XAML** — розмітка інтерфейсу
- **Data Binding** — зв'язування даних
- **NUnit** — unit-тестування
- **Microsoft.Win32** — діалоги файлів

3.3.3 Веб-версія: React + ASP.NET Core

Backend (ASP.NET Core):

- **ASP.NET Core Web API** — RESTful API
- **Controllers** — обробка HTTP запитів
- **Dependency Injection** — управління залежностями
- **ISessionService** — управління сесіями

Frontend (React):

- **React 19** — користувацький інтерфейс
- **TypeScript** — статична типізація
- **Vite** — білд-інструмент
- **Tailwind CSS** — стилізація

3.3.4 REST API архітектура

Веб-версія реалізує 20 точок доступу за RESTful принципами. Ось деякі з них:

Database endpoints:

- POST /api/database/create
- POST /api/database/load
- POST /api/database/download
- GET /api/database/info

Table endpoints:

- POST /api/table/create
- GET /api/table/tableName
- DELETE /api/table/tableName

Row endpoints:

- GET /api/table/tableName/rows
- POST /api/table/tableName/rows
- PUT /api/table/tableName/rows/index

3.3.5 Порівняння складності розробки

Аспект	**Десктоп**	**Веб**
Мови програмування	1 (C#)	2 (C#, TypeScript)
Frameworks	1 (WPF)	2 (ASP.NET Core, React)
Інтеграція	Локальна	Client-Server
API Layer	Не потрібен	20 точок доступу
Управління станом	WPF binding	React hooks + server state

Табл. 2: Складність технологічного стеку

4 Переваги та недоліки

4.1 Десктоп-версія

4.1.1 Переваги

- **Нативний інтерфейс** — потужні можливості WPF
- **Мінімальна затримка** — локальне виконання операцій
- **Повна автономність** — робота без мережі
- **Простота розгортання** — не потребує інфраструктури

4.1.2 Недоліки

- **Однокористувацька** — один користувач та БД
- **Платформна обмеженість** — тільки Windows
- **Розповсюдження** — установка на кожен ПК

4.2 Веб-версія

4.2.1 Переваги

- **Багатокористувацька** — багато користувачів одночасно
- **Кроссплатформеність** — робота через браузер
- **Відсутність установки** — доступ через URL
- **Централізоване управління** — все на сервері

4.2.2 Недоліки

- **Залежність від сервера** — необхідна мережа
- **Мережева затримка** — HTTP запити
- **Обмеження браузера** — немає прямого доступу до файлової системи

5 Порівняльна таблиця функціональності

Функціональність	**Десктоп**	**Веб**
Створення БД	✓	✓
Завантаження БД (з файлу)	✓(Вибір через File Explorer)	✓(завантаження на сервер)
Збереження БД (на диск)	✓(локальна файлова система)	✓(завантаження з браузера)
Управління таблицями	✓	✓
CRUD рядків	✓	✓
Валідація типів	✓	✓
Інтервальний тип	✓	✓
Текстові файли	✓(на диску)	✓(на сервері)
Перейменування колонок	✓	✓
Перестановка колонок	✓(drag-n-drop)	✓(drag-n-drop)
Unit-тестування	✓(NUnit для CoreLib)	✓(NUnit для CoreLib)

Табл. 3: Функціональність обох версій

6 Висновки та рекомендації

6.1 Досвід розробки

Розробка двох версій показала ефективність винесення бізнес-логіки у спільну бібліотеку (CoreLib), що дозволило:

- Уникнути дублювання коду
- Забезпечити ідентичну поведінку системи
- Спростити тестування та підтримку

6.2 Критерії вибору технології

Десктоп-версія доцільна для:

- Індивідуального використання
- Роботи без мережі
- Високих вимог до швидкодії
- Глибокої інтеграції з Windows

Веб-версія доцільна для:

- Багатокористувацьких середовищ
- Кроссплатформенної доступності
- Сучаснішого користувацького досвіду

6.3 Рекомендації для розширення

Для десктоп-версії:

- Додати шифрування БД
- Реалізувати автоматичне резервне копіювання
- Розглянути портування на .NET MAUI

Для веб-версії:

- Впровадити аутентифікацію (JWT, OAuth2)
- Додати HTTPS
- Реалізувати пагінацію для великих таблиць

6.4 Підсумковий висновок

Обидві реалізації є повноцінними. Вибір між ними визначається конкретними вимогами проєкту. Завдяки спільній бібліотеці CoreLib обидві версії гарантують однакову коректність обробки даних.
Досвід підтверджує, що розділення бізнес-логіки від UI дозволяє ефективно розробляти різні інтерфейси до однієї системи без дублювання коду.