SQLite i ADO.NET — książka adresowa

Czas: 2 × 45 min | praca indywidualna | Wykład 2.12

Cel: Zbudujesz książkę adresową z bazą SQLite: tabela tworzona z kodu, dodawanie, edycja, usuwanie i wyszukiwanie wyłącznie przez polecenia z parametrami.

Po zajęciach potrafisz:

- łączy się z bazą SQLite przez Microsoft.Data.Sqlite
- wykonuje polecenia CRUD z parametrami
- obsługuje wartości NULL i odczyt DataReader
- wyjaśnia i sprawdza ochronę przed wstrzykiwaniem SQL

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- pendrive lub konto w repozytorium do zapisania projektu

Oprogramowanie

- **Visual Studio Community 2026 (obciążenie „Programowanie aplikacji klasycznych platformy .NET")** — darmowe dla uczniów i nauczycieli; w pracowni zainstalowane visualstudio.microsoft.com/pl/downloads
- **.NET 10 SDK (LTS)** — instalowany razem z Visual Studio; sprawdzenie: `dotnet --version` dotnet.microsoft.com/pl-pl/download/dotnet/10.0
- **Pakiet NuGet CommunityToolkit.Mvvm 8.4** — instalacja: Zarządzaj pakietami NuGet nuget.org/packages/CommunityToolkit.Mvvm
- **Pakiet NuGet Microsoft.Data.Sqlite** nuget.org/packages/Microsoft.Data.Sqlite
- **DB Browser for SQLite** — podgląd pliku bazy; wersja przenośna nie wymaga instalacji sqlitebrowser.org/dl

Bezpieczeństwo

- Zapisuj projekt w swoim folderze (np. `C:\Users\<login>\source\repos`), a na koniec zajęć skopiuj go na pendrive lub wyślij do repozytorium.
- Nie uruchamiaj programów i pakietów z nieznanych źródeł; pakiety NuGet instaluj tylko te, które wskazuje zadanie.
- Rób przerwy dla oczu (zasada 20-20-20) — praca z projektantem okien wymaga długiego patrzenia w ekran.
- W bazie testowej używaj fikcyjnych danych — nie wpisuj prawdziwych numerów telefonów kolegów.

Przebieg zadania

1 Repozytorium

Projekt WPF `KsiazkaAdresowa`, klasa `Kontakt` i `KontaktyRepo`.

```
cmd.CommandText = "DELETE FROM Kontakty WHERE Id = $id";
cmd.Parameters.AddWithValue("$id", id);
cmd.ExecuteNonQuery();
```

1. `UtworzTabele()` — `CREATE TABLE IF NOT EXISTS` (Id, Imie, Nazwisko NOT NULL, Telefon, Email).
2. `Szukaj(fraza)` — LIKE z parametrem `$f`; znaki % dodaj do wartości parametru.
3. `Dodaj` (z `last_insert_rowid()`), `Zmien`, `Usun(id)` — wspólna metoda dodająca parametry; puste pola jako `DBNull.Value`.

2 ViewModel i okno

Interfejs.

1. DataGrid kontaktów, pole wyszukiwania (wyniki na żywo), formularz po prawej.
2. Zaznaczenie wiersza wypełnia formularz; Nowy czyści; Zapisz dodaje lub zmienia (po Id); Usuń.
3. Plik bazy obok programu: `Path.Combine(AppContext.BaseDirectory, "kontakty.db")`.
4. Błędy bazy (`SqliteException`) pokazuj w komunikacie, nie wyjątkiem.

3 Bezpieczeństwo

Sprawdź ochronę.

1. Dodaj kontakt z nazwiskiem `O'Brien` i z imieniem `x' OR '1'='1` — oba zapisują się poprawnie.
2. Wyszukaj frazę `x' OR '1'='1` — wynik tylko jeden kontakt.
3. Dla porównania napisz (w komentarzu, NIE uruchamiaj na danych) wersję ze sklejaniem tekstu i opisz, co by się stało.
4. Otwórz plik bazy w DB Browser for SQLite i sprawdź zawartość tabeli.
5. Zatwierdź pracę w Git (plik .db dodaj do .gitignore).

W sprawozdaniu umieść

- ☐ kod KontaktyRepo
- ☐ zrzut aplikacji i bazy w DB Browser
- ☐ wyniki testów bezpieczeństwa
- ☐ wyjaśnienie wstrzykiwania SQL

Pytania kontrolne

1. Kiedy używasz ExecuteNonQuery, ExecuteScalar i ExecuteReader?
2. Dlaczego parametry chronią przed wstrzykiwaniem SQL?
3. Jak zapisać do bazy brak wartości?
4. Dlaczego pliku bazy nie dodajemy do repozytorium?

Kryteria oceny

Kryterium	Punkty
Repozytorium z parametrami	7
ViewModel i okno CRUD	6
Testy bezpieczeństwa	4
Pytania	3
Razem	**20**

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).