teknologi informasi
UNIVERSITAS MUHAMMADIYAH YOGYAKARTA

Modul Praktikum

Pengembangan Aplikasi Basisdata

Apriliya Kurnianti S.T., M.Eng

Daftar Isi

Koneksi Database dalam Aplikasi

Tujuan Praktikum

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami konsep koneksi database menggunakan ADO.NET.
- Menggunakan SqlConnection untuk membuka dan menutup koneksi ke SQL Server.
- Menjalankan kode C# (Console Application) untuk menghubungkan aplikasi ke database.

Alat dan Bahan

Software yang Digunakan:

- Visual Studio (2019/2022)
- Microsoft SQL Server (Minimal 2016)
- .NET Framework 4.7.2 atau lebih baru

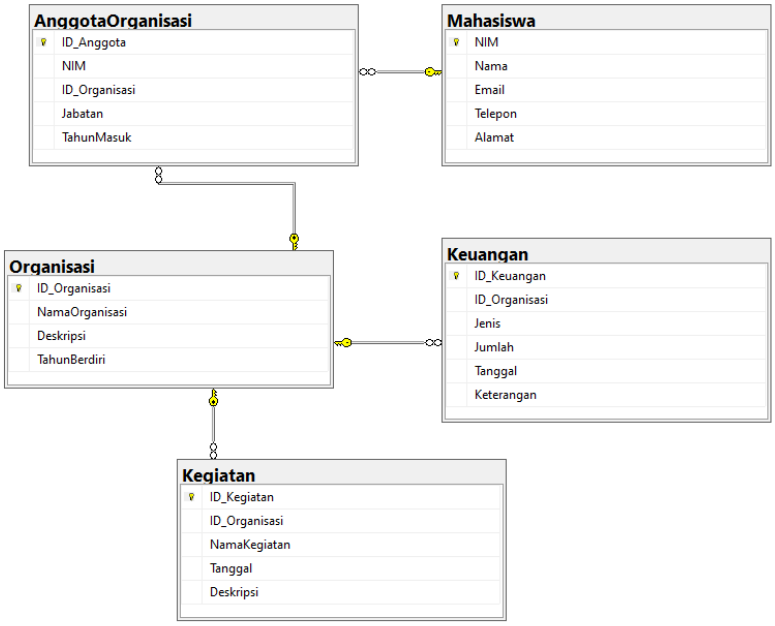
Tools ADO.NET yang Digunakan:

- SqlConnection – Untuk membuka dan menutup koneksi database.
- SqlCommand – Untuk menjalankan perintah SQL.

Langkah-Langkah Praktikum

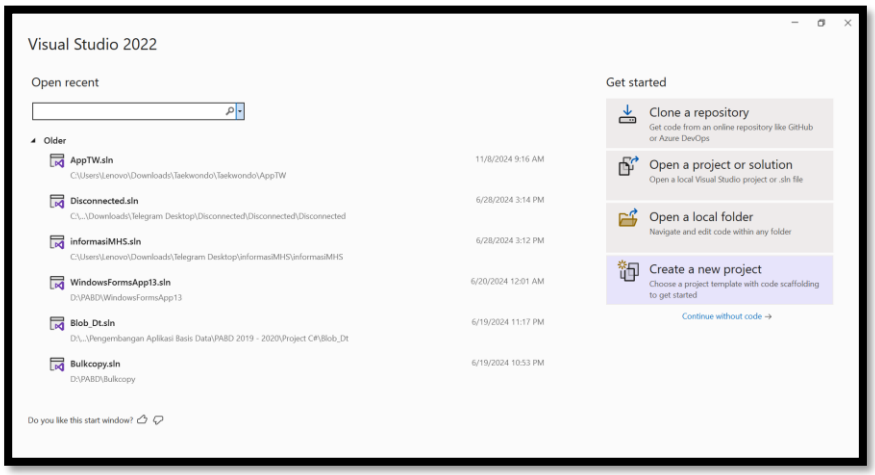
a. Menyiapkan Database di SQL Server

- Buka SQL Server Management Studio (SSMS)
- Buat database baru dengan nama OrganisasiMahasiswa dengan struktur sebagai berikut:

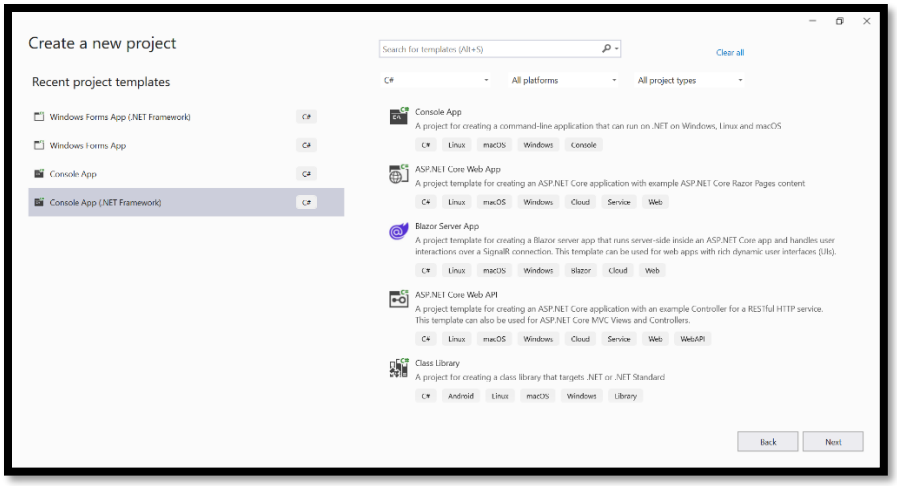


b. Membuat Proyek Console C# di Visual Studio

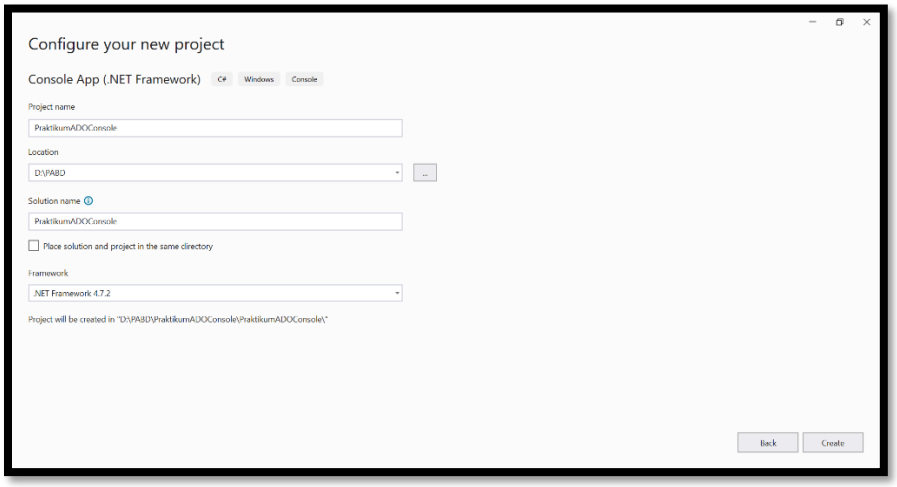
1. **Buka Visual Studio** dan buat proyek baru:



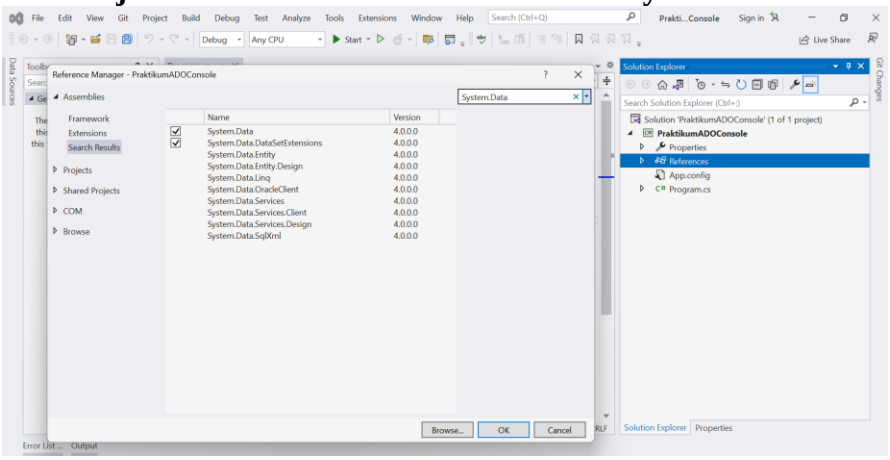
- Pilih **Console App (.NET Framework)**.



- Beri nama proyek: `PraktikumADOConsole`.
- Pilih .NET Framework 4.7.2 atau lebih baru.



2. **Tambahkan referensi ADO.NET**:
   - Klik **Project → Add Reference →** Tambahkan **System.Data**.



c. *Implementasi Koneksi Database dengan ADO.NET (Console C#)*

1. Menampilkan "Koneksi Berhasil" Jika Koneksi Berjalan

Buka `Program.cs` dan ubah dengan kode berikut:

a. Import Library

```
using System;
using System.Data;
using System.Data.SqlClient;
```

Penjelasan:

- `using System;` → Menggunakan fitur dasar C# seperti input/output (`Console.WriteLine`).
- `using System.Data;` → Menggunakan fungsi terkait **database**.
- `using System.Data.SqlClient;` → Menggunakan **ADO.NET** untuk menghubungkan C# dengan SQL Server.

b. Deklarasi Connection String

```
// Connection String untuk menghubungkan ke database SQL Server
static string connectionString = "Data Source=DESKTOP-RAM2OFI\\APRILIYA;" +
    "Initial Catalog=OrganisasiMahasiswa;Integrated Security=True";
```

Penjelasan:

- `Database=OrganisasiMahasiswa;` → Menggunakan database `OrganisasiMahasiswaDB`.

c. Menu Utama (Looping)

```
while (true)
{
    Console.Clear();
    Console.WriteLine("=== Aplikasi Manajemen Mahasiswa ===");
    Console.WriteLine("1. Tambah Data Mahasiswa");
    Console.WriteLine("2. Tampilkan Data Mahasiswa");
    Console.WriteLine("3. Hapus Data Mahasiswa");
    Console.WriteLine("4. Keluar");
    Console.Write("Pilih menu (1-4): ");

    string pilihan = Console.ReadLine();
    switch (pilihan)
    {
        case "1":
            InsertData();
            break;
        case "2":
            RefreshData();
            break;
        case "3":
            DeleteData();
            break;
        case "4":
            Console.WriteLine("Terima kasih telah menggunakan aplikasi ini.");
            return;
        default:
            Console.WriteLine("Pilihan tidak valid, coba lagi!");
            break;
    }

    Console.WriteLine("\nTekan ENTER untuk kembali ke menu utama...");
    Console.ReadLine();
}
```

Penjelasan:

- Menggunakan `while (true)` agar program berjalan terus sampai pengguna memilih keluar.
- `Console.Clear();` → Membersihkan layar console sebelum menampilkan menu.
- `Console.WriteLine(...)` → Menampilkan pilihan menu ke pengguna.
- `Console.ReadLine();` → Menunggu input pengguna (1-4).
- `switch (pilihan)` → Memilih fitur berdasarkan input pengguna.
- `case "1": InsertData();` → Jika pengguna memilih 1, panggil fungsi InsertData().
- `case "4": return;` → Keluar dari aplikasi jika memilih 4.

d. Fungsi Insert Data

```
static void InsertData()
{
    Console.WriteLine("\n=== Tambah Data Mahasiswa ===");
    Console.Write("Masukkan NIM (11 digit)   : ");
    string nim = Console.ReadLine();
    Console.Write("Masukkan Nama             : ");
    string nama = Console.ReadLine();
    Console.Write("Masukkan Email            : ");
    string email = Console.ReadLine();
    Console.Write("Masukkan Telepon (08xxxxx): ");
    string telepon = Console.ReadLine();
    Console.Write("Masukkan Alamat           : ");
    string alamat = Console.ReadLine();
```

Penjelasan:

- `Console.Write("Masukkan NIM: ");` → Meminta input dari pengguna.
- `string nim = Console.ReadLine();` → Menyimpan input pengguna dalam variabel.

```
using (SqlConnection conn = new SqlConnection(connectionString))
{
    string query = "INSERT INTO Mahasiswa (NIM, Nama, Email, Telepon, Alamat) VALUES (@NIM, @Nama, @Email, @Telepon, @Alamat)";
    SqlCommand cmd = new SqlCommand(query, conn);
    cmd.Parameters.AddWithValue("@NIM", nim);
    cmd.Parameters.AddWithValue("@Nama", nama);
    cmd.Parameters.AddWithValue("@Email", email);
    cmd.Parameters.AddWithValue("@Telepon", telepon);
    cmd.Parameters.AddWithValue("@Alamat", alamat);
```

Penjelasan:

- `using (SqlConnection conn = new SqlConnection(connectionString))` → Membuat koneksi ke database.
- `new SqlCommand(query, conn);` → Menyiapkan query SQL INSERT untuk menambahkan data.
- `cmd.Parameters.AddWithValue("@NIM", nim);` → Menggunakan parameterized query untuk mencegah SQL Injection.

```
        try
        {
            conn.Open();
            int result = cmd.ExecuteNonQuery();
            if (result > 0)
                Console.WriteLine("Data berhasil ditambahkan!");
            else
                Console.WriteLine("Gagal menambahkan data.");
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error: " + ex.Message);
        }
    }
}
```

Penjelasan:

- `conn.Open();` → Membuka koneksi ke SQL Server.
- `cmd.ExecuteNonQuery();` → Menjalankan perintah SQL INSERT.
- `catch (Exception ex)` → Menangkap error jika ada kesalahan dalam eksekusi SQL.

e. Fungsi Refresh Data

```
static void RefreshData()
{
    Console.WriteLine("\n=== Daftar Mahasiswa ===");

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        string query = "SELECT * FROM Mahasiswa";
        SqlCommand cmd = new SqlCommand(query, conn);
```

Penjelasan:

- Query : `SELECT * FROM Mahasiswa` digunakan untuk mengambil semua data dari tabel mahasiswa.

```
try
{
    conn.Open();
    SqlDataReader reader = cmd.ExecuteReader();

    if (reader.HasRows)
    {
        Console.WriteLine("\n{0,-11} | {1,-20} | {2,-25} | {3,-13} | {4,-30}",
                          "NIM", "Nama", "Email", "Telepon", "Alamat");
        Console.WriteLine(new string('-', 110));

        while (reader.Read())
        {
            Console.WriteLine("{0,-11} | {1,-20} | {2,-25} | {3,-13} | {4,-30}",
                reader["NIM"], reader["Nama"], reader["Email"], reader["Telepon"], reader["Alamat"]);
        }
    }
    else
    {
        Console.WriteLine("Tidak ada data mahasiswa.");
    }

    reader.Close();
}
catch (Exception ex)
{
    Console.WriteLine("Error: " + ex.Message);
}
```

Penjelasan:

- `SqlDataReader reader = cmd.ExecuteReader();` → Menjalankan query SELECT.
- `while (reader.Read())` → Membaca data per baris dan mencetaknya ke layar.

f. Fungsi Delete Data

```
static void DeleteData()
{
    Console.Write("\nMasukkan NIM yang ingin dihapus: ");
    string nim = Console.ReadLine();

    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        string query = "DELETE FROM Mahasiswa WHERE NIM = @NIM";
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@NIM", nim);
```

Penjelasan:

- Query `DELETE FROM Mahasiswa WHERE NIM = @NIM` menghapus mahasiswa berdasarkan NIM yang diinput.

```
try
{
    conn.Open();
    int result = cmd.ExecuteNonQuery();
    if (result > 0)
        Console.WriteLine("Data berhasil dihapus!");
    else
        Console.WriteLine("Data tidak ditemukan.");
}
catch (Exception ex)
{
    Console.WriteLine("Error: " + ex.Message);
}
```

Penjelasan:

- `ExecuteNonQuery();` → Menjalankan query DELETE.
- Jika `result > 0`, berarti ada data yang berhasil dihapus.

Cara Menjalankan Program

a. Pastikan database OrganisasiMahasiswaDB sudah dibuat di SQL Server.
b. Buka Visual Studio → Jalankan proyek Console App.
c. Pilih menu:
   - 1 untuk menambah data.
   - 2 untuk melihat daftar mahasiswa.
   - 3 untuk menghapus data mahasiswa.
   - 4 untuk keluar dari aplikasi.

Kesimpulan

1. Insert, Refresh, dan Delete Data bisa dilakukan dengan ADO.NET di Console Application C#.
2. Connection String menentukan bagaimana aplikasi terhubung ke SQL Server.
3. Try-Catch digunakan untuk menangani error saat berinteraksi dengan database.
4. Menutup koneksi database setelah digunakan sangat penting untuk mencegah kebocoran memori.