

M12 MATERI 1

1. TUJUAN

CPMK : Mahasiswa dapat mengintegrasikan Frontend React JS dengan Backend ASP.NET Core dan Database Azure SQL.

Sub-CPMK :

- a. Mahasiswa dapat membuat build production dari aplikasi React.
- b. Mahasiswa dapat mengonfigurasi ASP.NET Core.
- c. Mahasiswa dapat melakukan migrasi database menggunakan Entity Framework Core.

2. DURASI WAKTU

2 pertemuan x 4 jam

3. DASAR TEORI

React JS, Single Page Application (SPA), Entity Framework Core.

4. Percobaan

• Membuat Frontend App

1. Buatlah folder baru dengan nama **PRG4_M12_P1_XXX**
2. Lalu pada Visual Studio Code, buka terminal (Command Prompt/PowerShell) lalu jalankan perintah berikut untuk membuat proyek React baru:

```
npx create-react-app frontend-app
```

3. Tunggu instalasi selesai hingga keluar output “Happy Hacking”:

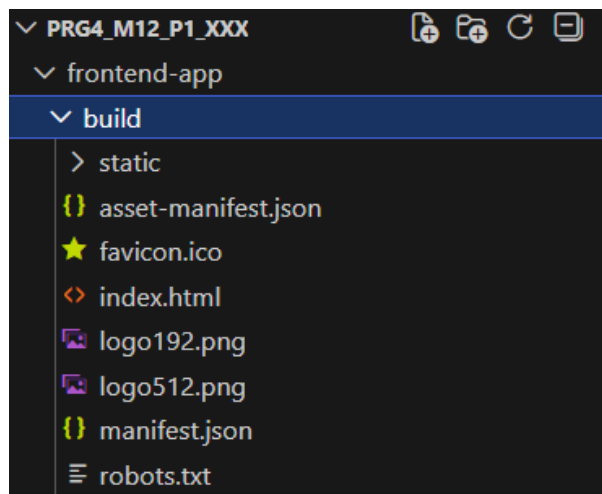
```
We suggest that you begin by typing:  
  
cd frontend-app  
npm start  
  
Happy hacking!
```

4. Buka kembali terminal anda, lalu jalankan perintah berikut untuk pindah folder dan menjalankan proses build untuk mengubah kode React menjadi static file dan siap deploy:

```
PRG4_M12_P1_XXX> cd frontend-app
```

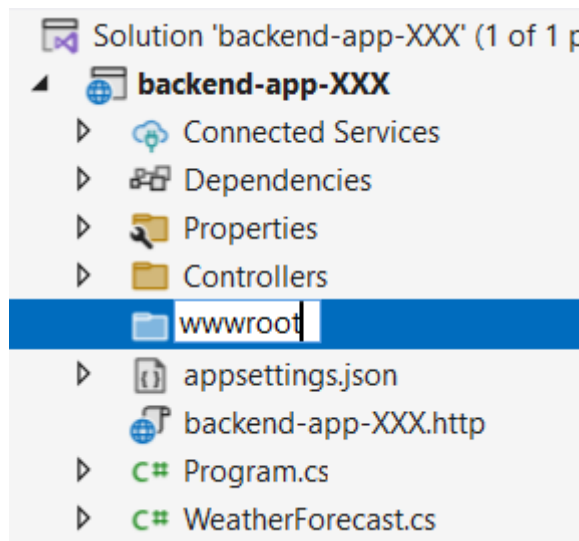
```
PRG4_M12_P1_XXX\frontend-app> npm run build
```

5. Hasil build akan muncul di dalam folder bernama build:

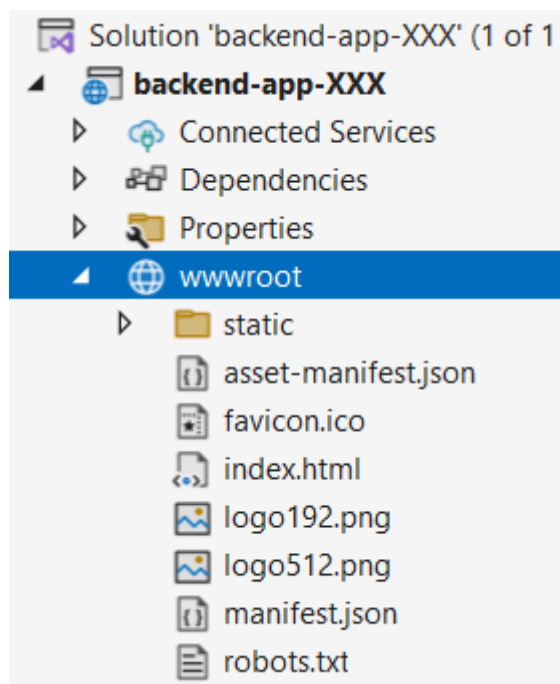


- **Membuat Backend ASP.NET Core Web API**

1. Buatlah project ASP. NET Core Web API baru dengan nama **backend-app-XXX**
2. Buat folder baru bernama wwwroot di dalam project backend:



3. Lalu pindahkan seluruh hasil build yang ada di dalam folder frontend-app/build/* ke dalam folder wwwroot:



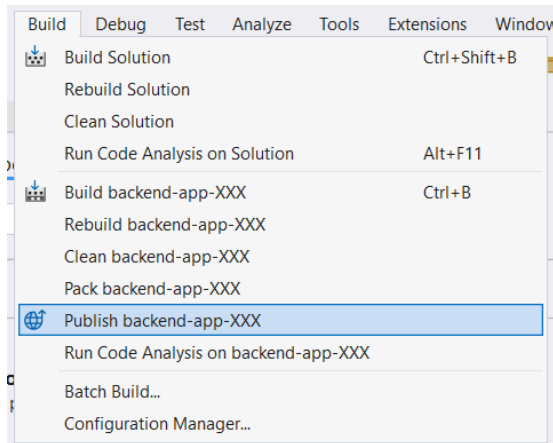
4. Modifikasi file Program.cs untuk mengizinkan akses static file. Tambahkan kode berikut sebelum app.Run():

```

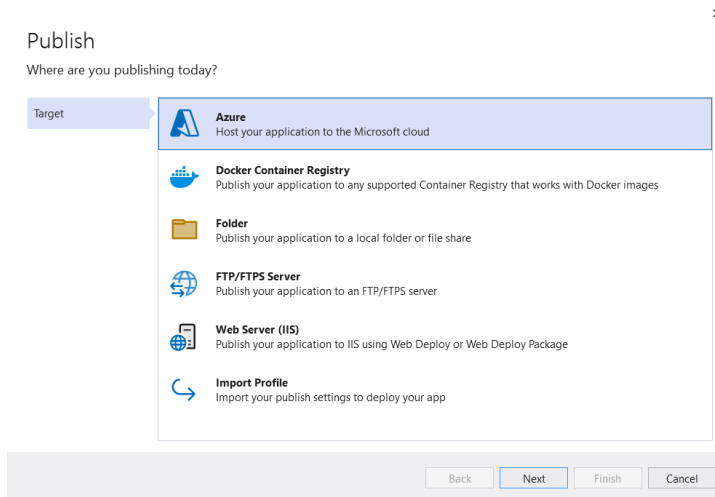
19     app.UseHttpsRedirection();
20
21     app.UseAuthorization();
22
23     app.UseStaticFiles();
24     app.UseRouting();
25     app.MapControllers();
26     app.MapFallbackToFile("index.html");
27
28     app.Run();
29

```

5. Jalankan menu **Build > Publish backend-app-xxx:**



6. Pada pilihan **Target** publish pilih opsi **Azure**:



7. Lalu pada pilihan **Specific Target**, pilih opsi **Azure App Service (Windows)**:

Publish

Which Azure service would you like to use to host your application?

Target

Specific target

- Azure Container Apps (Linux)**
Run scalable containerized applications and microservices on a serverless platform in Azure
- Azure App Service (Windows)**
Publish your application code to a managed infrastructure that is easy to scale
- Azure App Service (Linux)**
Publish your application code to a managed infrastructure that is easy to scale
- Azure App Service Container**
Publish your application as a Docker image to Azure Container Registry and run it on Azure App Service
- Azure Container Registry**
Publish your application as a Docker image to Azure Container Registry
- Azure Virtual Machine**
Manage your own infrastructure

Back Next Finish Cancel

8. Create New pada pilihan App Service:

Publish

Select existing or create a new Azure App Service

Politeknik Astra
fahriel.dwifaldi@polytechnic.as...

Target

Subscription name

Azure for Students

Specific target

App Service

API Management

Search

Create new

Create an Azure App Service

There are no existing instances available

Create a new instance

☐ Deploy as ZIP package

☐ Turn on Basic Authentication (not recommended) ⓘ

Back Next Finish Cancel

9. Sesuaikan nama dari **App Service**, **Resource Group**, dan **Hosting Plan**. Pastikan lokasi pada hosting plan diubah terlebih dahulu ke **Indonesia Central**:

App Service (Windows)

Create new

fahriel.dwifaldi@polytechnic.astra.ac.id (Politeknik Astra)

Name

backend-app-XXX-appService

Subscription name

Azure for Students

Resource group

backend-app-XXX-resourceGroup*

New...

Hosting Plan

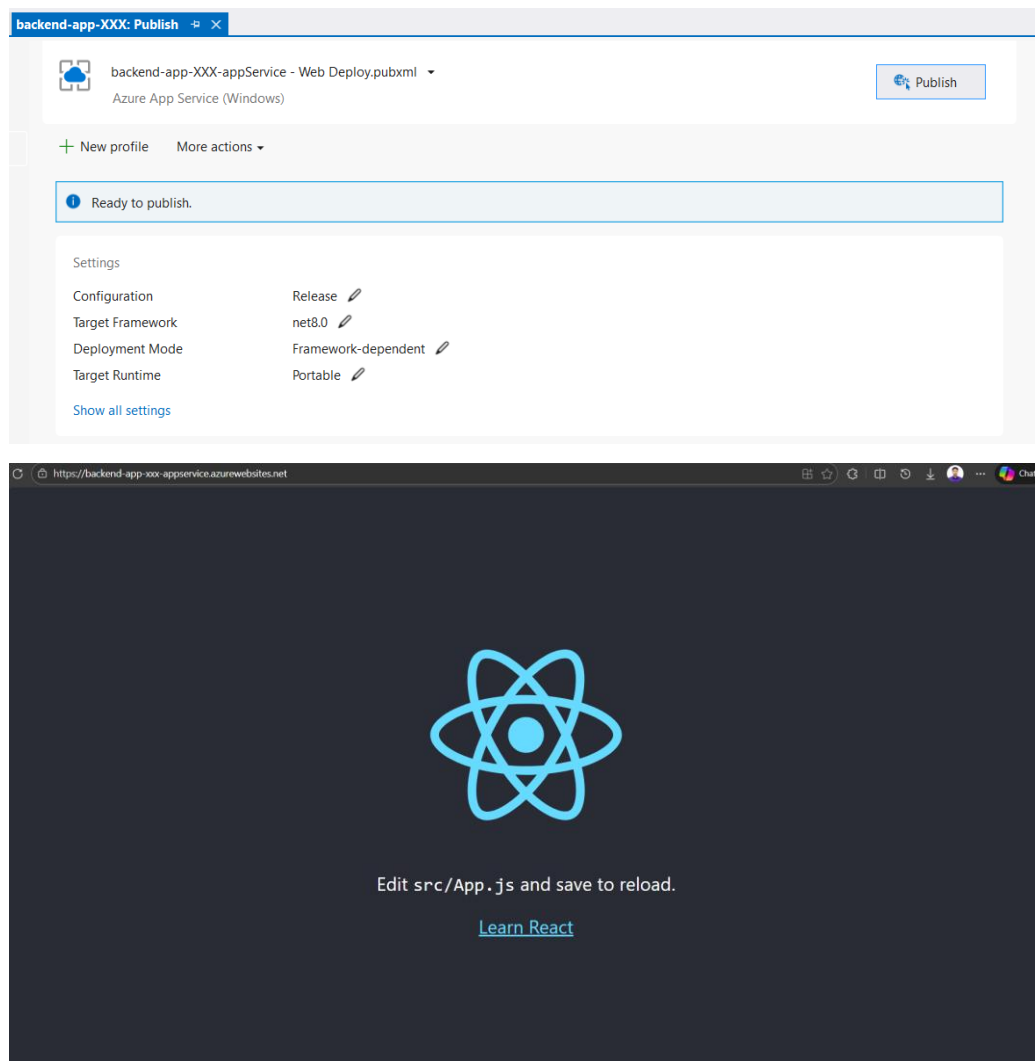
backend-app-XXX-plan* (Indonesia Central, S1)

New...

Export...

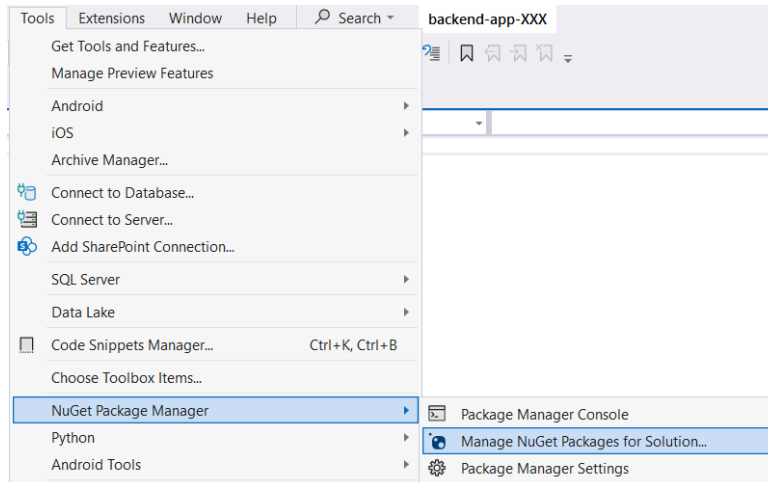
Create Cancel

10. Pilih checkbox **Skip This Step** pada tahap **API Management** lalu tekan Finish dan tunggu hingga proses **Publish profile creation progress** selesai.
11. Tekan Publish, pastikan project sudah berhasil terdeploy menggunakan Azure:

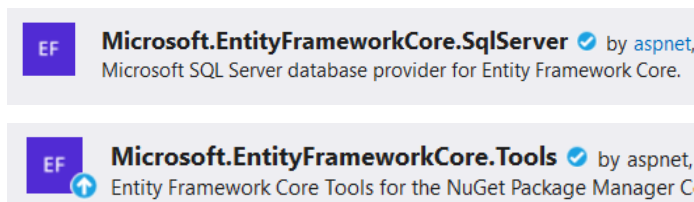


- **Integrasi Menggunakan Azure SQL Database**

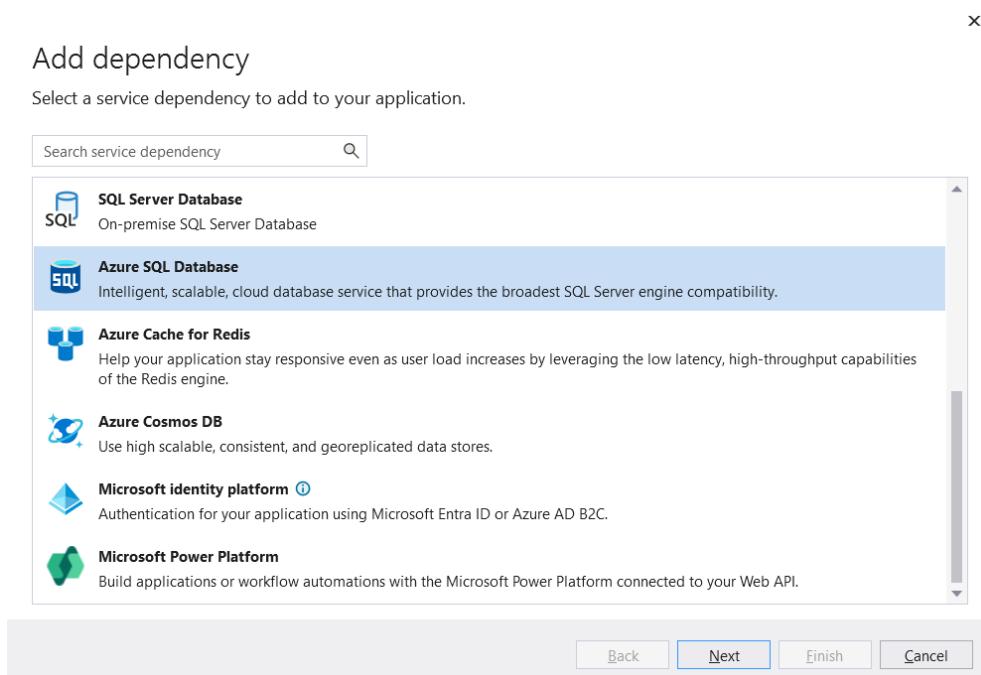
1. Buka menu **Tools > NuGet Package Manager > Manage NuGet Packages for Solution:**



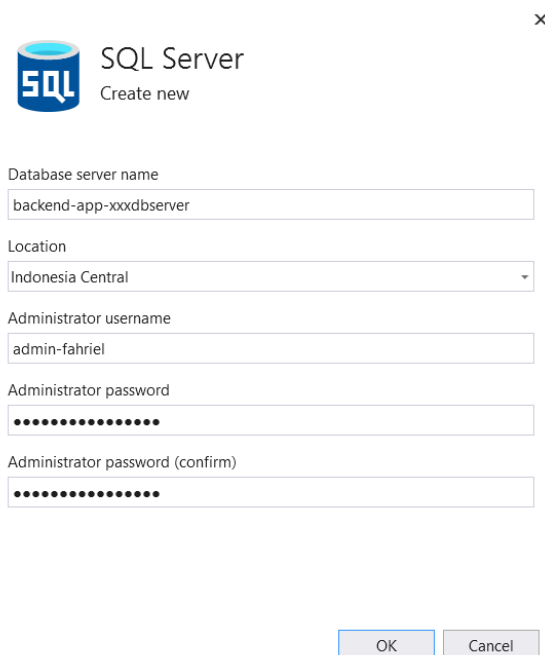
2. Lalu install 2 packages EntityFrameworkCore SqlServer dan Tools:



3. Buka tampilan menu **Publish** yang ada pada langkah sebelumnya, tambahkan **Service Dependencies > SQL Server Database:**



4. **Create New > New Database Server**, lalu masukkan username dan password dari database administrator:



SQL Server
Create new

Database server name
backend-app-xxdbserver

Location
Indonesia Central

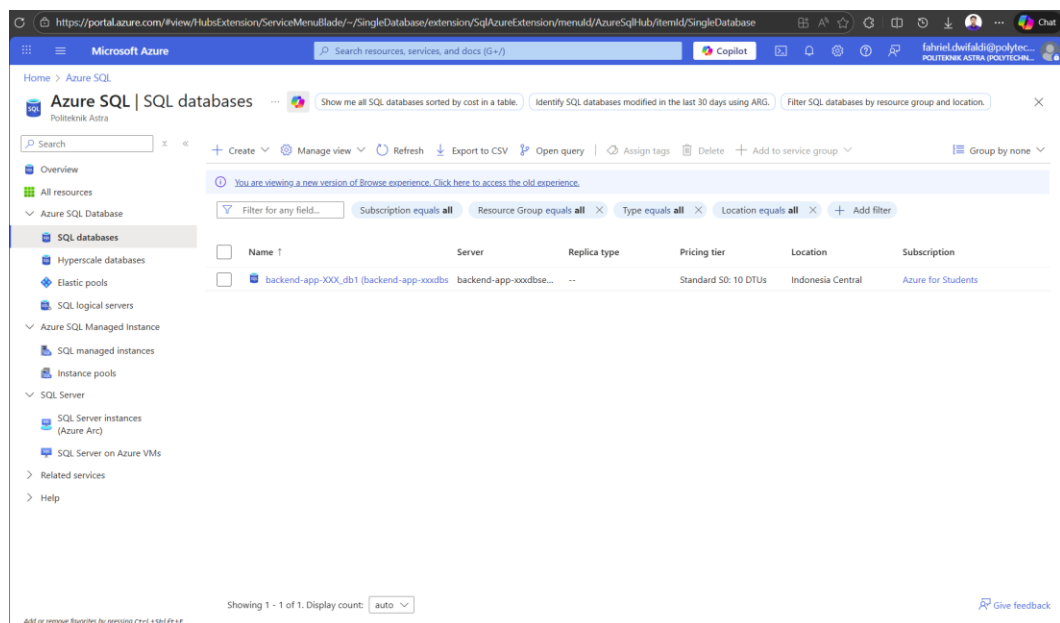
Administrator username
admin-fahriel

Administrator password
.....

Administrator password (confirm)
.....

OK Cancel

5. Pastikan Azure SQL Database berhasil dibuat dengan cara membuka website **Portal Azure > Azure SQL Database**:



6. Buka Azure SQL Database yang baru saja dibuat, lalu pada menu **Settings > Connection Strings** copy value dari ADO.NET (SQL Authentication).
7. Buka file appsettings.json dan tambahkan konfigurasi koneksi database dengan isi DefaultConnection dengan value yang sudah diambil di tahap 6:


```

{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=tcp:SERVER_ANDA.database.windows.net,1433;Initial Catalog=NAMA_DB;User ID=USER;Password=PASS;Encrypt
  }
}

```

8. Buat model baru dan beri nama Product.cs

```

namespace backend_app
{
    4 references
    public class Product
    {
        1 reference
        public int Id { get; set; }

        0 references
        public string Name { get; set; } = string.Empty;

        [Column(TypeName = "decimal(18, 2)")]
        0 references
        public decimal Price { get; set; }
    }
}

```

9. Tambahkan controller baru dan beri nama ProductsController.cs

```

namespace backend_app.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference
    public class ProductsController : ControllerBase
    {
        private readonly AppDbContext _context;

        0 references
        public ProductsController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        1 reference
        public async Task<ActionResult<IEnumerable<Product>>> GetProducts()
        {
            return await _context.Products.ToListAsync();
        }

        [HttpPost]
        0 references
        public async Task<ActionResult<Product>> PostProduct(Product product)
        {
            _context.Products.Add(product);
            await _context.SaveChangesAsync();
            return CreatedAtAction(nameof(GetProducts), new { id = product.Id }, product);
        }
    }
}

```

```

[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteProduct(int id)
{
    var product = await _context.Products.FindAsync(id);
    if (product == null) return NotFound();

    _context.Products.Remove(product);
    await _context.SaveChangesAsync();

    return NoContent();
}
}
}

```

10. Buat file AppDbContext.cs baru dan tulis ulang kode berikut:

```

namespace backend_app
{
    7 references
    public class AppDbContext : DbContext
    {
        0 references
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {
        }

        4 references
        public DbSet<Product> Products { get; set; }
    }
}

```

11. Buka file Program.cs. Tambahkan kode berikut sebelum baris `var app = builder.Build();`:

```

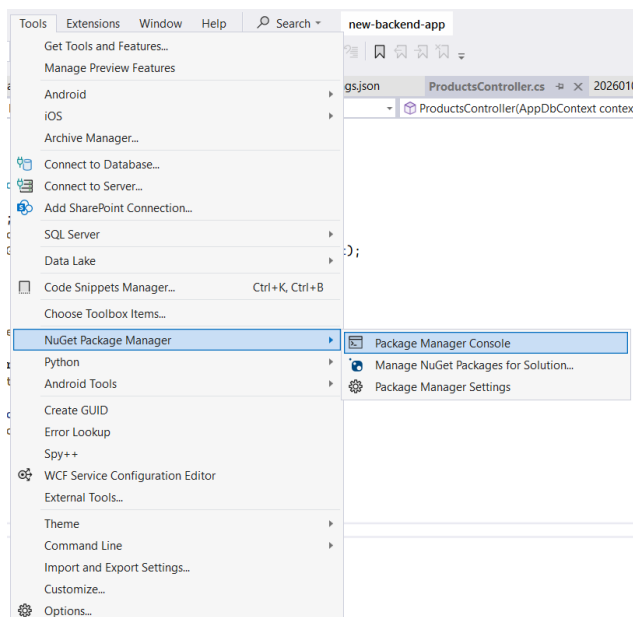
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<AppDbContext>(
    options => options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

```

12. Akses menu **Tools > NuGet Package Manager > Package Manager Console**:



13. Tambahkan migration baru dengan nama InitialMigration untuk membuat tabel di database:

```
PM> Add-Migration InitialCreate
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
```


14. Apabila sudah berhasil, jalankan perintah **Update-Database**:

```
PM> Update-Database
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (31ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (22ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (10ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT 1
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (20ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT OBJECT_ID(N'[__EFMigrationsHistory]');
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (11ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT [MigrationId], [ProductVersion]
  FROM [__EFMigrationsHistory]
  ORDER BY [MigrationId];
Microsoft.EntityFrameworkCore.Migrations[20402]
  Applying migration '20260108002602_InitialCreate'.
Applying migration '20260108002602_InitialCreate'.
Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (10ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  INSERT INTO [__EFMigrationsHistory] ([MigrationId], [ProductVersion])
  VALUES (N'20260108002602_InitialCreate', N'8.0.10');
Done.
```

15. Cek kembali pada **Portal Azure > Azure SQL Database > SQL Databases > { Nama_database } > Query Editor**:

The screenshot displays the Azure Portal interface. The top navigation bar shows the URL <https://portal.azure.com/#@polytechnic.astra.ac.id/resource/subscriptions/228ea...>. The main content area is titled "backend-app-XXX_db1 (backend-app-xxxdbserver)" and is identified as an "SQL database". On the left, a sidebar menu lists various options: Overview, Activity log, Tags, Diagnose and solve problems, Query editor (preview) (which is selected), Mirror database in Fabric (preview), Resource visualizer, and Settings. The "Query editor (preview)" section is active, showing a search bar, a "Login" button, and a "New Query" button. Below this, a list of tables is displayed: "dbo.__EFMigrationsHistory" and "dbo.Products". The "dbo.Products" table is expanded, showing its columns: "Id (PK, int, not null)", "Name (nvarchar, not null)", and "Price (decimal, not null)". A message at the top of the query editor states: "Showing limited object explorer here. For full capability please click here to open Azure Data Studio."

16. Apabila terjadi masalah ketika login, klik bagian **Allowlist ...**




Welcome to SQL Database Query Editor

SQL server authentication

Login *

Password *

✖ Cannot open server 'backend-app-xxxxdbserver' requested by the login.
Client with IP address '139.255.84.186' is not allowed to access the server. To enable access, use the Azure Management Portal or run `sp_set_firewall_rule` on the master database to create a firewall rule for this IP address or address range. It may take up to five minutes for this change to take effect.
[Allowlist IP 139.255.84.186 on server backend-app-xxxxdbserver](#)

OK

17. Buka kembali file App.js yang ada di frontend, dan update kode nya seperti pada gambar berikut:

```
frontend-app > src > JS App.js > App
1 import React, { useState, useEffect } from 'react';
2 import './App.css';
3
4 function App() {
5   const [products, setProducts] = useState([]);
6   const [name, setName] = useState("");
7   const [price, setPrice] = useState("");
8
9   useEffect(() => {
10     fetchProducts();
11   }, []);
12
13   const fetchProducts = async () => {
14     const response = await fetch('/api/products');
15     const data = await response.json();
16     setProducts(data);
17   };
18
19   const handleSubmit = async (e) => {
20     e.preventDefault();
21     if (!name || !price) return;
22
23     const newProduct = { name: name, price: parseFloat(price) };
24
25     await fetch('/api/products', {
26       method: 'POST',
27       headers: { 'Content-Type': 'application/json' },
28       body: JSON.stringify(newProduct)
29     });
30
31     setName("");
32     setPrice("");
33     fetchProducts();
34   };
35
36   const handleDelete = async (id) => {
37     await fetch(`/api/products/${id}`, {
38       method: 'DELETE'
39     });
40     fetchProducts();
41   };
42 }
```

```

43   return (
44     <div className="App">
45       <header className="App-header">
46         <h1>Manajemen Produk</h1>
47
48         <form onSubmit={handleSubmit} style={{ marginBottom: '20px' }}>
49           <input
50             type="text"
51             placeholder="Nama Produk"
52             value={name}
53             onChange={(e) => setName(e.target.value)}
54             style={{ padding: '10px', marginRight: '10px' }}
55           />
56           <input
57             type="number"
58             placeholder="Harga"
59             value={price}
60             onChange={(e) => setPrice(e.target.value)}
61             style={{ padding: '10px', marginRight: '10px' }}
62           />
63           <button type="submit" style={{ padding: '10px 20px', cursor: 'pointer' }}>
64             Tambah
65           </button>
66         </form>
67
68         <table className='table' style={{ width: '50%', margin: '0 auto', color: 'black', background: 'white' }}>
69           <thead>
70             <tr style={{ background: '#007bff', color: 'white' }}>
71               <th style={{ padding: '10px' }}>Nama</th>
72               <th style={{ padding: '10px' }}>Harga</th>
73               <th style={{ padding: '10px' }}>Aksi</th>
74             </tr>
75           </thead>
76           <tbody>
77             {products.map((product) => (
78               <tr key={product.id} style={{ borderBottom: '1px solid #ccc' }}>
79                 <td style={{ padding: '10px' }}>{product.name}</td>
80                 <td style={{ padding: '10px' }}>{product.price}</td>
81                 <td style={{ padding: '10px' }}>
82                   <button

```

```

82                     <button
83                       onClick={() => handleDelete(product.id)}
84                       style={{ background: 'red', color: 'white', border: 'none', padding: '5px 10px', cursor: 'pointer' }}
85                     >
86                       Hapus
87                     </button>
88                 </td>
89             </tr>
90           ))}
91         </tbody>
92       </table>
93     </header>
94   </div>
95   );
96 }
97
98 export default App;

```

18. Buka kembali file App.js yang ada di frontend, dan update kode nya seperti pada gambar berikut:

```
frontend-app > src > # App.css > th
1  .App {
2    text-align: center;
3    font-family: Arial, sans-serif;
4  }
5
6  .App-header {
7    background-color: #282c34;
8    min-height: 100vh;
9    display: flex;
10   flex-direction: column;
11   align-items: center;
12   justify-content: center;
13   font-size: calc(10px + 2vmin);
14   color: white;
15 }
16
17 table {
18   border-collapse: collapse;
19   width: 80%;
20   margin-top: 20px;
21   color: black;
22   background-color: white;
23   border-radius: 8px;
24   overflow: hidden;
25 }
26
27 th, td {
28   padding: 12px;
29   text-align: left;
30   border-bottom: 1px solid #ddd;
31 }
32
33 th {
34   background-color: #007bff;
35   color: white;
36 }
```

19. Pastikan ketika sudah dipublish ulang, kode sudah terupdate dan berhasil menyimpan ke database:

