|  |
| --- |
| Database Schema |
|  |

|  |
| --- |
| Solution |
| Graphical user interface  Description automatically generated with low confidence |

|  |
| --- |
|  |
| Diagram  Description automatically generated |

|  |
| --- |
|  |
| Diagram  Description automatically generated |

Graph QL Web API Project:

* Project Name: TitleInfoGQL
* Project Description: GraphQL Web API reads data from SQL Server via Entity Framework Core
* Project Dependencies:
  + Required Framework: .NET 5
  + Requires the following NuGet Packages:

|  |
| --- |
| * <PackageReference Include="GraphQL.Server.Ui.Voyager" Version="5.0.2" /> * <PackageReference Include="HotChocolate.AspNetCore" Version="11.2.2" /> * <PackageReference Include="HotChocolate.Data.Entityframework" Version="11.2.2" /> * <PackageReference Include="Microsoft.EntityframeworkCore.Design" Version="5.0.6"> * <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets> * <PrivateAssets>all</PrivateAssets> * </PackageReference> * <PackageReference Include="Microsoft.EntityframeworkCore.SqlServer" Version="5.0.6" /> |

* + Requires the following Tools:

|  |
| --- |
| * + Entity Framework Core .NET Command-line Tools, Version 5.0.6     - Check to see if you have it via command: dotnet ef     - To install type command: dotnet tool install --global dotnet-ef     - To update type command: dotnet tool update --global dotnet-ef   + Database First script I ran to automatically create Model classes: Scaffold-DbContext "Server={ServerName}\{InstanceName};Database=Titles;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer --output-dir Models     - IMPORTANT NOTES :       * You’ll need to do some tweaking to YOUR DbContext class:         + Remove the empty constructor         + The empty-less constructor remove your type as the Hot Chocolate Framework doesn’t like it and leave just the “DbContextOptions” plain |

* + How to run:

|  |
| --- |
| * + Open the folder in Visual Studio Code   + Click “Yes” to add dependencies   + Open up a Terminal and run this command: dotnet run   + Open up Google Chrome or Edge: <http://localhost:5000/graphql/> |

* How to debug:

|  |
| --- |
| * Download and install the [Insomnia API](https://insomnia.rest/download) as it is AMAZING and your friend for debugging! * In your Startup.cs file do the following:   + In your constructor, in addition to the IConfiguration, also INJECT the IWebHostEnvironment and create your private readonly property for it as well.   + In the ConfigureServices method where you are adding the GraphQLServer, add this to the chain of “services.AddGraphQLServer()” for the Hot Chocolate Framework as it will EXPOSE MORE DETAILS on any exceptions you’re encountering when your environment is set to “IsDevelopment” environment variable:   .ModifyRequestOptions(opt => opt.IncludeExceptionDetails = \_env.IsDevelopment()); |

Testing the GraphQL Web API:

|  |
| --- |
| Here’s how you can use the Insomnia API to test your GraphQL endpoint |
|  |

What is GraphQL?

|  |  |
| --- | --- |
| * Is A Query & Manipulation language for APIs * Is the runtime for fulfilling requests * Everything in GraphQL is a POST request * They just need ONE endpoint that all end in “/graphql” * Was created by Facebook to address both over and under fetching   + No over fetching happens as you can pick only the fields you need and NOT the ENTIRE payload   + No under fetching happens as you can drill down to the children and grab what you need in just ONE request. * Open Source hosted by the Linux Foundation;   It has a Schema that describes the API in full and thus self documenting. | * Is comprised of “Types” which can be any of the following:   + Query,   + Mutation,   + Subscription,   + Objects,   + Enumeration – list of static values   + Scalar – id, int, string boolean, float * Must have a “Root Query Type” * Has Resolvers which can do the following:   + return data for a given field (The Hot Chocolate Framework will do the heavy lifting for you)   + Resolve to ANY data source (Database, Microservice, REST API, etc.) |

Blazor Web Assembly Single Page Application:

* Project Name: TitleInfoClient
* Project Description: SPA Web Application that calls a GraphQL Web API to get Movie Title Information
* Project Dependencies:
  + Required Framework: .NET 5
  + Requires the following NuGet Packages:

|  |
| --- |
| * <PackageReference Include="Microsoft.AspNetCore.Components.WebAssembly" Version="5.0.6" /> * <PackageReference Include="Microsoft.AspNetCore.Components.WebAssembly.DevServer" Version="5.0.6" PrivateAssets="all" /> * <PackageReference Include="Microsoft.Extensions.Http" Version="5.0.0" /> * <PackageReference Include="System.Net.Http.Json" Version="5.0.0" /> |

* + 3rd Party Libraries:

|  |
| --- |
| * + [JQuery DataTables](https://datatables.net/forums/discussion/56389/datatables-with-blazor)     - Easily integrated it to the Blazor app and the table is fully SEARCHABLE, and has pagination and sorting out of the box. |

* + How to run:

|  |
| --- |
| * + Open the folder in Visual Studio Code   + Click “Yes” to add dependencies   + Open up a Terminal and run this command: dotnet run   + Open up Google Chrome or Edge: <http://localhost:5002> (I modified the LaunchSettings.json to have it point to a different port than 5001 as the web api is using it) |

* + How To Debug:

|  |
| --- |
| * + You’ll need to use CHROME or EDGE as currently FireFox does not support the (Shift+Alt+D)   + Run the following command: dotnet run –configuration debug   + Open up Google Chrome or Edge: <http://localhost:5002>   + Press F12   + Press Shift + Alt+D   + A new Tab will open, copy the command provided and run it.   + Another browser window will pop up   + Press F12   + Press Shift+Alt+D again   + In the Developer tools click on the “Sources” tab, you’ll see the wasm folder (expand and you’ll see the C# files as if you were in visual studio) |

|  |
| --- |
| Here’s a visual description of the above debugging process from Google Chrome |
|  |

|  |
| --- |
| Here’s the Blazor Web Application with JQuery DataTables. As you type in the search box it will look at ALL of the data and find anything that matches what you type. |
|  |