IPROX TV Show Data Sync and Web API Development Report

By: Sanjith Gunaratne

Completed Work

.NET 8 Web API, Minimal API & Azure Function

1. **Clean Architecture:** The project has been structured using clean architecture principles to ensure scalability and maintainability, enabling the easy addition of features in the future.
2. **SOLID Principles:** The project follows the SOLID principles, ensuring that the system is easy to maintain, extend, and refactor. This includes proper handling of the Single Responsibility, Open/Closed Principle, Dependency Inversion, and more.
3. **Repository Pattern & Unit of Work:** Implemented the repository pattern for better separation of concerns and easier testing. The Unit of Work pattern was added to call services, ensuring better transaction management.
4. **DRY Principle (Don't Repeat Yourself):** The DRY principle has been implemented to eliminate code duplication, creating reusable methods and shared components for common tasks.
5. **API Versioning (v1.0):** Added API versioning to maintain backward compatibility while adding new features. This is available only in the minimal API project (v1.0), with default settings also available in controller-based API projects.
6. **Customized API Documentation (Swagger):** Swagger has been configured to provide interactive API documentation for easy testing and understanding of endpoints. This is available only in the minimal API project, with default settings available in controller-based API projects.
7. **Enums:** Enums were utilized for managing categorical values, enhancing code readability and maintainability.
8. **Entity & LINQ:** Used entities and LINQ queries for efficient database interaction and querying.
9. **Seeding & Migrations:** Implemented data seeding and database migrations, using a base class for initialization and schema management. Created tables for **TvShow, Genre, and TvShowGenres**.
10. **Caching:** Implemented **MemoryCache** for improved performance on the **GET api/show** endpoint when no filter parameters are provided, fetching data from memory cache when available.

11. **Pagination:** Implemented pagination to efficiently handle large data sets, improving server load and performance.
12. **Unit Tests:** Wrote unit tests, including **CreateAsync_ShouldLogError_WhenExceptionIsThrown()**. Additional tests for other areas are planned.
13. **Search for a Show by Name:** Implemented a feature to allow users to search for a show by name in the database.
14. **Display Shows in Descending Order of Premiere Date:** Ensured that shows are displayed in descending order of their premiere date.
15. **Ensure Database Foreign Keys for Genres:** Created a separate **genre master table** and a **mapping table** between shows and genres to ensure proper foreign key relationships.
16. **Controller and Minimal API:** Both controller-based and minimal API approaches have been implemented for CRUD operations, offering flexibility in API design.
17. **Async Programming:** Applied **asynchronous programming** for all I/O-bound operations (e.g., database access, API calls) for non-blocking performance.
18. **Azure Function for TV Show Data Sync:** An **Azure Function** with a **timer trigger** has been created to sync TV show data from the **TvMaze API**. The function is triggered **every minute**, fetching only the **latest sync ID** from the database to update new data without hitting API rate limits. About **250 movies** are retrieved per API call and updated in the database accordingly.

Pending Work

1. **Implement Configurable Parameters:**
    - Currently, only **connection strings** are configurable. Other parameters, such as **pageCount**, are still hardcoded.
    - Moving these parameters to configuration settings (e.g., **appsettings.json**) will enhance flexibility.
    - Implementing the **IOptions pattern** for reading configuration settings will allow for better separation of concerns and allow easy adjustments without code modifications.
2. **Stored Procedures:**
    - Stored procedures have been implemented for **complex queries**, improving performance and maintainability.
3. **Timeouts:**
    - Timeout management needs to be implemented for both **external API calls** and **database queries**, ensuring better management of long-running operations.

Running the Project Locally

1. Setting Up the Database

- Use the provided **sample database** located in the `Meta/Database` folder.
- Restore the database and update the **connection string** in `appsettings.json` accordingly.
- Alternatively, run database migrations using the following steps:
    1. Set the startup project as **Iprox.Presentation.TvShows.Api**.
    2. In **Package Manager Console**, select **Iprox.Infrastructure.Persistence** as the default project.
    3. Run the command:
       `Update-Database`

2. Configuring Connection Strings and App Settings

For Web API and Minimal API:

Modify the `appsettings.json` file with the appropriate database connection string:

```
"ConnectionStrings": {
    "ApplicationDbContext":
"Server=.;Database=IproxDb;Trusted_Connection=True;MultipleActiveResultSets=tr
ue;TrustServerCertificate=True;"
},
"Logging": {
    "LogLevel": {
        "Default": "Information",
        "Microsoft.AspNetCore": "Warning"
    }
},
"AllowedHosts": "*"
```

For Azure Function:

Modify the `local.settings.json` file with the correct storage connection string:

```
{
    "IsEncrypted": false,
    "Values": {
        "AzureWebJobsStorage":
"DefaultEndpointsProtocol=https;AccountName=iproxtvshowstorage;AccountKey=7Dd0
TIzb+YDQyqIZorIxRyUCvLzHre2U7DK9rZt0ioM+CPu5+cyQ1+3br75XRJGEKTDcIXVC1LAw+AStLN
LTVw==;EndpointSuffix=core.windows.net",
        "FUNCTIONS_WORKER_RUNTIME": "dotnet-isolated"
    },
    "ConnectionStrings": {
        "ApplicationDbContext":
"Server=.;Database=IproxDb;Trusted_Connection=True;MultipleActiveResultSets=tr
ue;TrustServerCertificate=True;"
    }
}
```

3. Running the APIs and Azure Function Locally

To Run Web API (Controller-Based)

- Navigate to `Iprox.Presentation.TvShows.Api`.
- Run the project using **Visual Studio** or the following command in the terminal:

  ```
  dotnet run
  ```

To Run Minimal API

- Navigate to `Iprox.Presentation.TvShows.Minimal.Api`.
- Run the project using:

  ```
  dotnet run
  ```

To Run Azure Function

- Navigate to `Iprox.Presentation.Functions.TvShowFunctions`.
- Start the function using:

  ```
  func start
  ```