

HopeLine Architecture and Design Pattern Considerations

This document is prepared by **Edmel Ricahuerta**

This document is created to help the development team to have better understanding of the architectural designs and patterns of the application.

Audience: **Development Team (Group 7)**

Database and Data Access

The application's database is using Azure's database and server. Since we are using a student starter account, this server will only run for 12 months. Further development must consider deploying it to other services.

To get the user and password or access to database, please send a request with a Subject line – "Azure Access – HopeLine."

Check out <https://azure.microsoft.com/en-ca/services/sql-database/> for more information.

In .NET Core, **datacontext** is used to create connection to database, retrieves data from database, converts objects to SQL queries. For this application code first approach is used. To have a better understanding on how this works, click <https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database>.

Repository and Unit of Work Pattern

This pattern allows an abstraction between data access layer and logic layer. This pattern is useful for separation of concerns and testing. To see examples and concepts, see the following:

- <https://medium.com/@mlbors/using-the-repository-pattern-with-the-entity-framework-fa4679f2139>
- <http://www.tugberkugurlu.com/archive/generic-repository-pattern-entity-framework-asp-net-mvc-and-unit-testing-triangle>
- <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/repository-pattern?view=aspnetcore-2.1>

Dependency Injection

This application is using NET Core 2 framework that supports dependency injection software pattern. To know more about this, see the following:

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.1>

<https://stackify.com/net-core-dependency-injection/>

<https://medium.com/volosoft/asp-net-core-dependency-injection-best-practices-tips-tricks-c6e9c67f9d96>

Razor Pages vs MVC vs SPA

Razor Pages is new concept of building NET application. For this project, we will be using a mix of everything but mostly Razor Pages. These are some links about razor pages:

<https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-2.1>

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/razor-pages-start?view=aspnetcore-2.1>

<https://www.learnrazorpages.com/>

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/xplat?view=aspnetcore-2.1>

MVC stands for Model View Controller that has been around for long time and has different design concept based on what stack is used.

<https://www.google.ca/search?q=razor+pages&oq=Razor+pages&aqs=chrome.0.0j69i61j0j69i60l2j35i39.3904j0j4&sourceid=chrome&ie=UTF-8>

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-2.1>

SPA stands for Single Page Application that is mostly Angular, React, Vue or some other frontend framework with a REST API as backend.

<https://docs.microsoft.com/en-us/aspnet/core/client-side/spa/?view=aspnetcore-2.1>

<https://docs.microsoft.com/en-us/aspnet/core/client-side/spa/angular?view=aspnetcore-2.1&tabs=visual-studio>

Software Architectural Design

This application's architecture is following the .NET guidelines with some mix of other designs.

<https://github.com/dotnet-architecture/eShopOnWeb>

PS: There are more concepts used for this application. If you want to know more just ask or message me

- Edmel