

BookStore

TECHNICAL IMPLEMENT

LINH NGUYEN

Github URL

<https://github.com/linhnguyenhp88/BookShop/tree/feature/Dev/Sprint0.0.1>

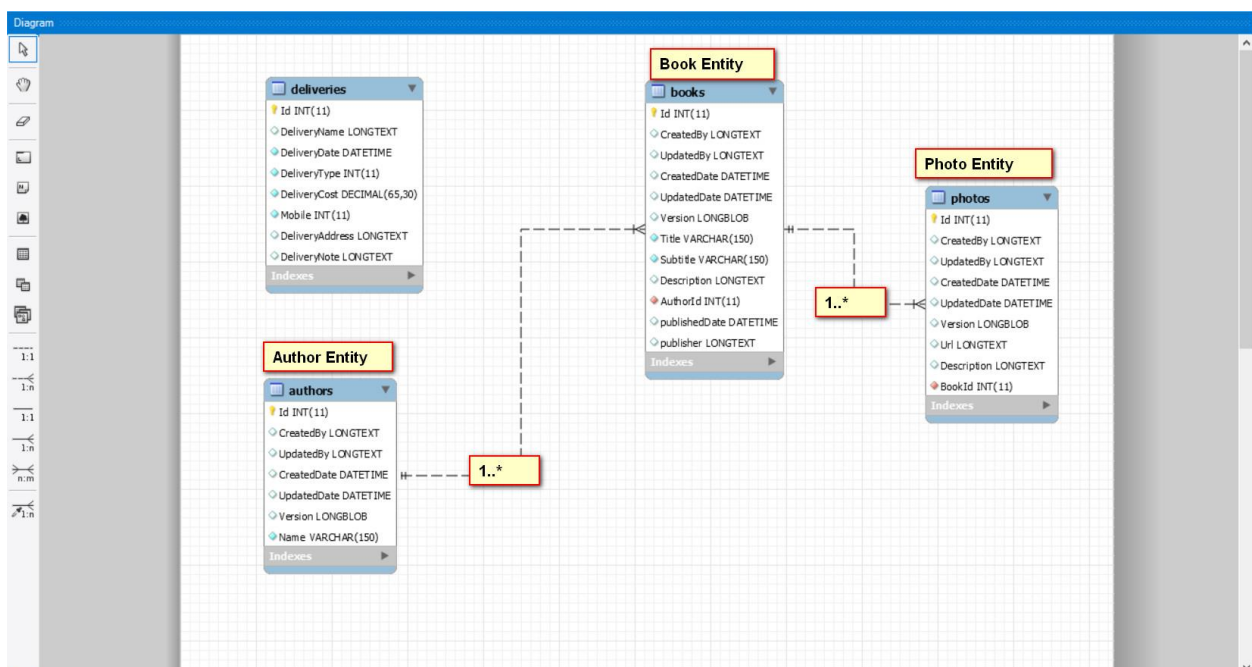
Prerequisite

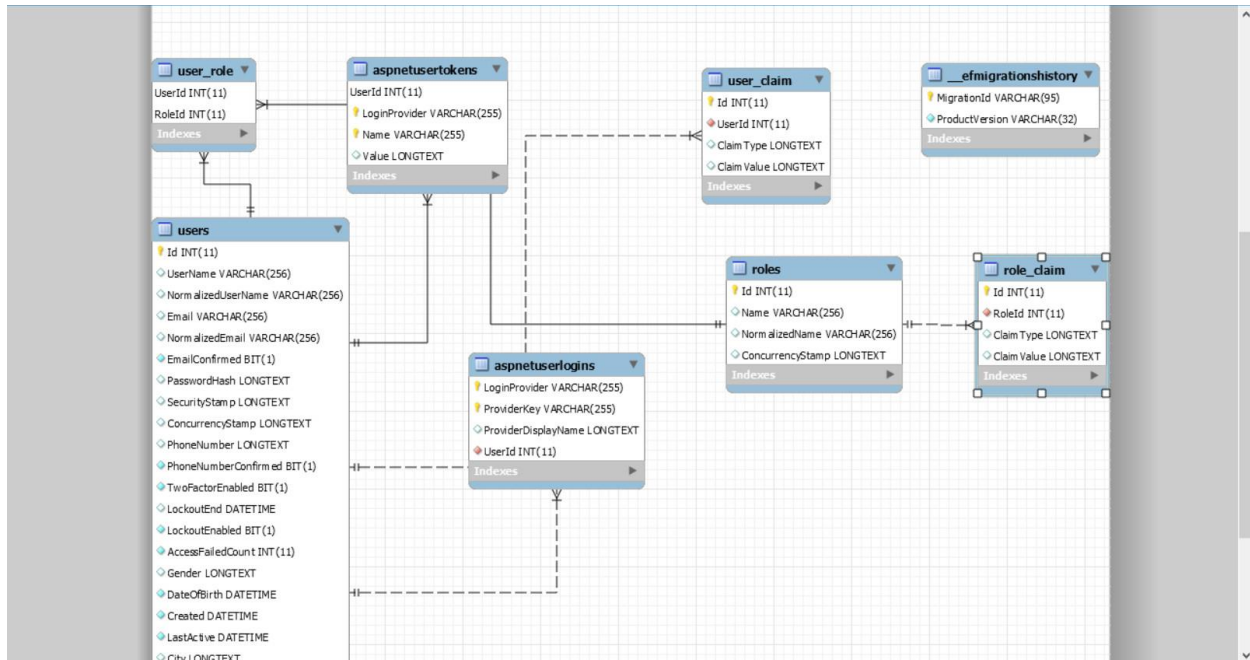
- .NET Core SDK 2.2.0
- Visual Studio 2019 (16.3.0)
- MySQL or SQLite
- Visual Studio Code
- Angular
- NodeJs
- Typescript

BookStore Website is a monolith architecture application consisting of services based on .NET Core, NodeJS, Angular Client-Side and more running. It demonstrates how to wire up into a larger application using architectural principals.

Business Context

Conceptual Model

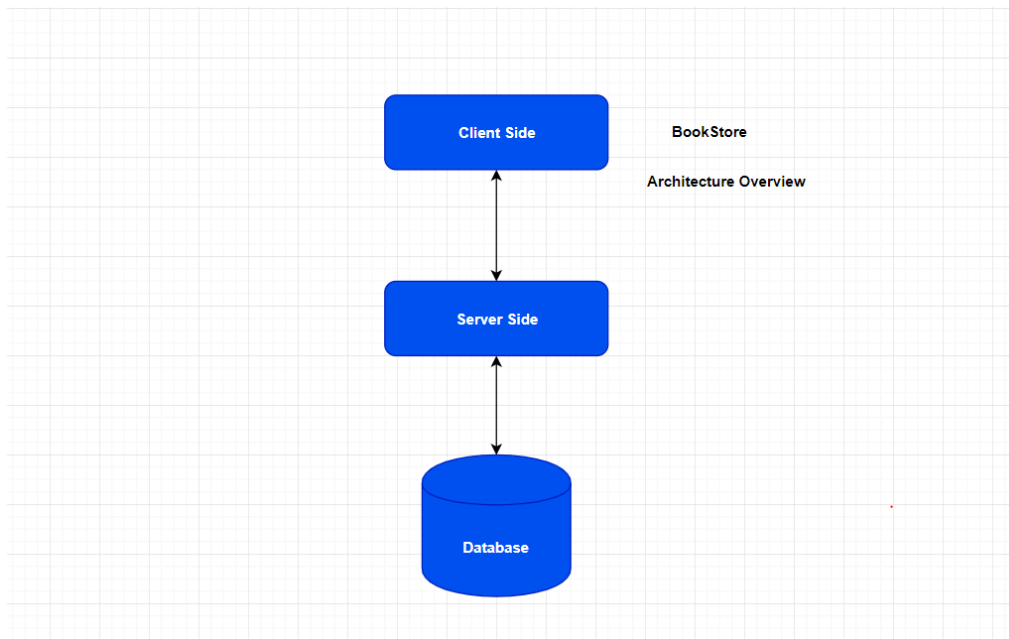




Identity Entities Model implement from ASP.NET Core Identity (in this application using asp.net core Entity Framework Code First)

More Detail please click here: <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.0&tabs=visual-studio>

Architecture Overview

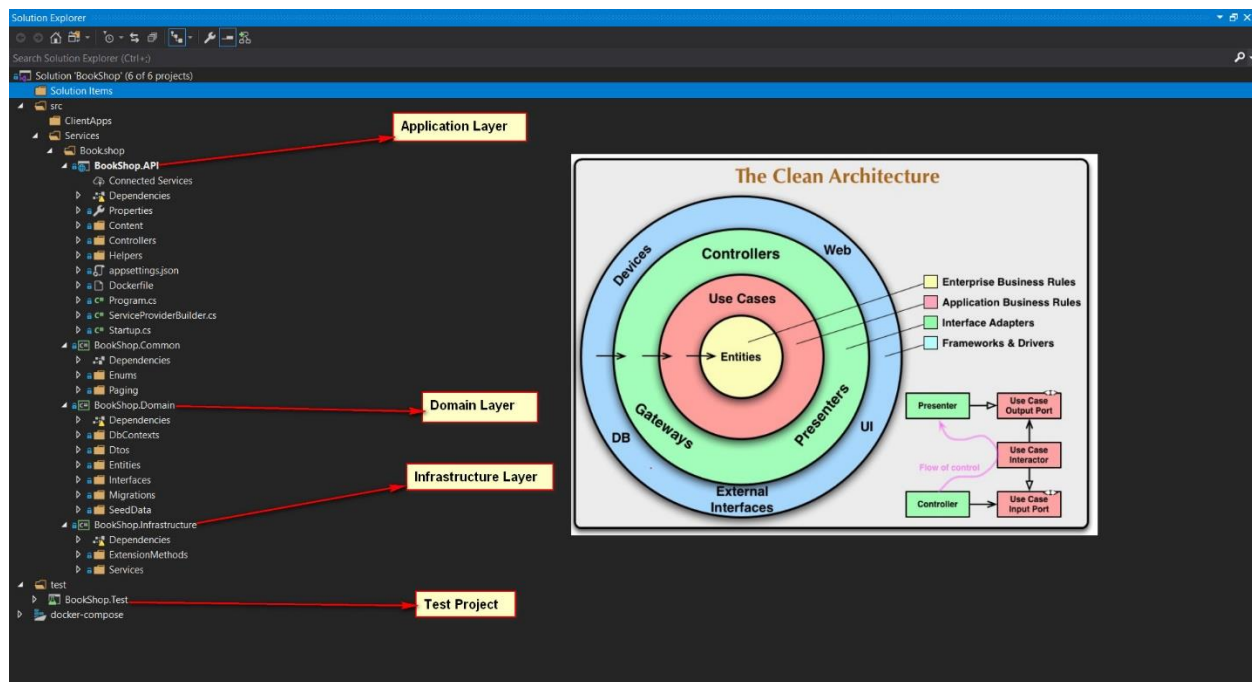


Backend Development

Open visual studio BookShop.sln for a solution containing all projects.

In the backend apply Domain-Driven Design Pattern to the application, because with this pattern has many benefits.

- Organization gains a useful model of its domain.
- Determining where to place boundaries between Bounded Contexts balances two competing goals.
- Easily move or modify the small parts with little or no side effects.
- A clear and manageable path through a very complex problem.
- Coding it's generally well organized and easily tested, and the business logic all lives in one place.



Application Layer

- Asp.net Core Web API
- Network access to services
- API contracts/Implementation
- Queries using Entity Framework Core ORM

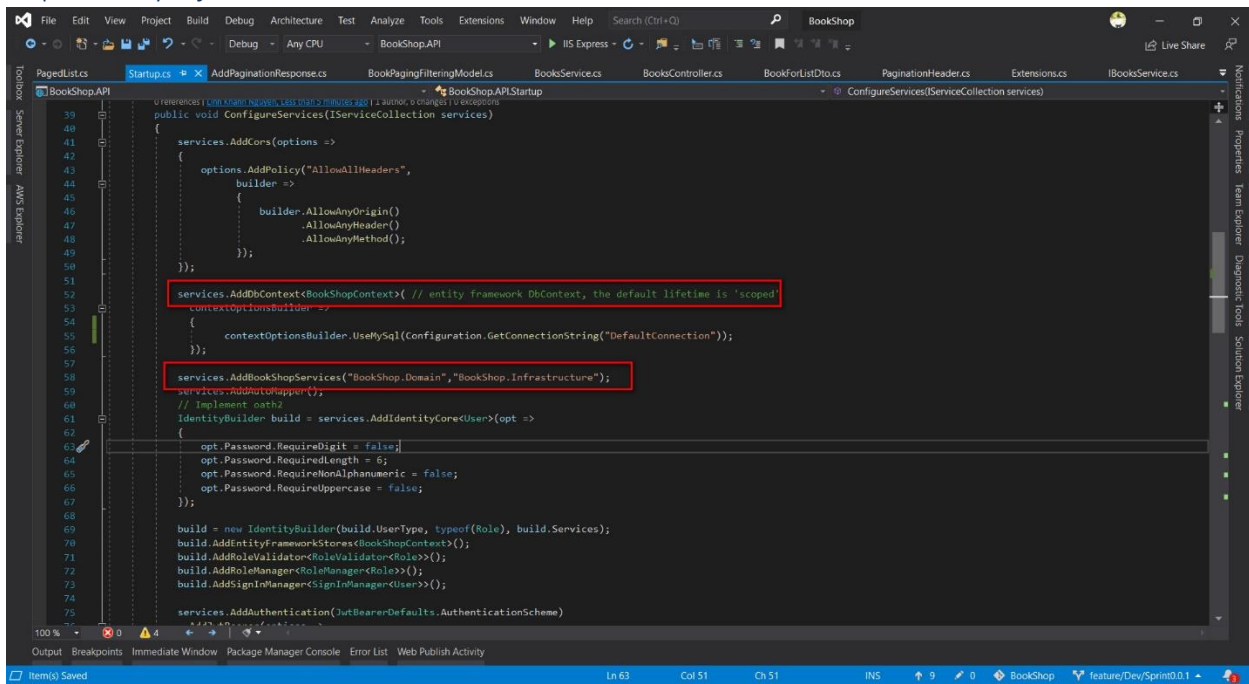
Infrastructure Layer

- Data persistence infrastructure
- This simple demo so that no need to implement Repository, use services instead of repository.
- Use ORM Entity Framework Core

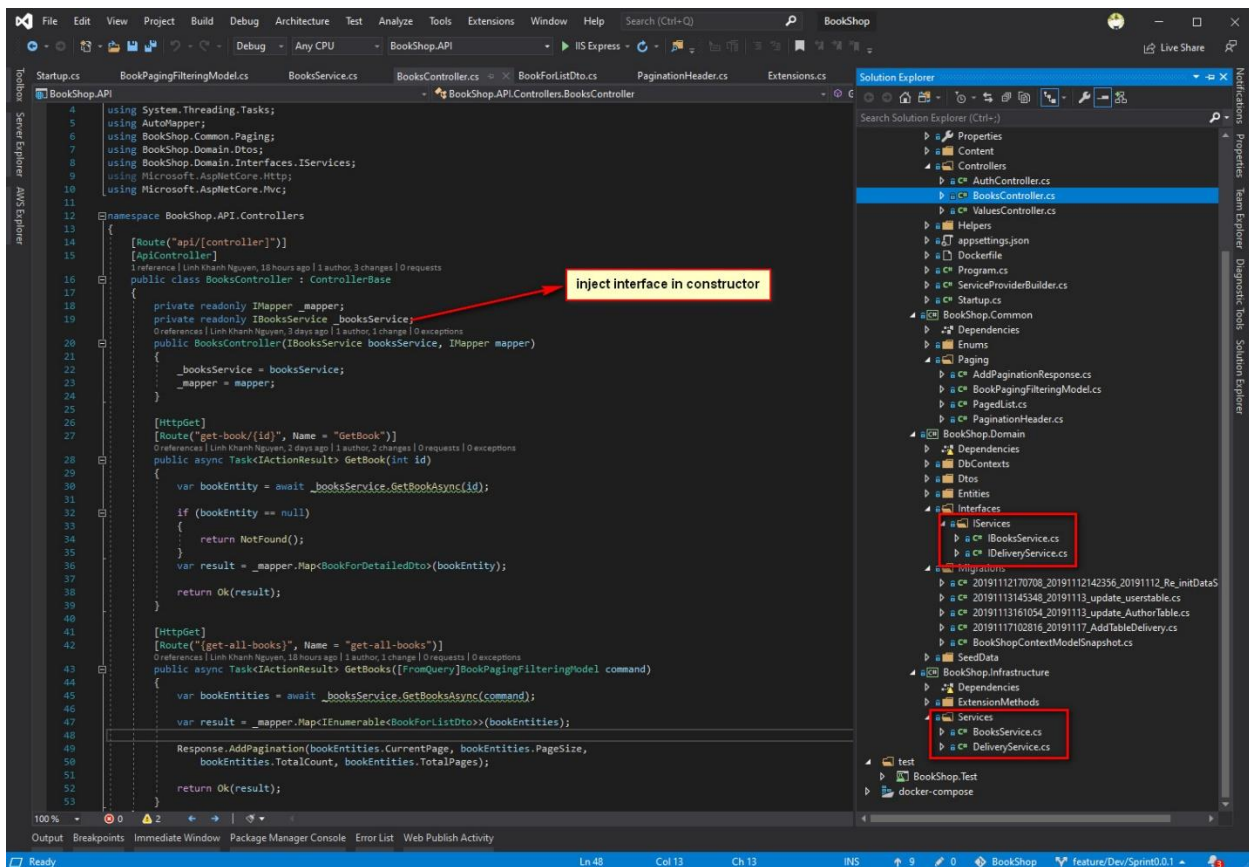
Domain model Layer

- Domain entity model
- POCO entity class (clean code C#)
- DDD pattern Repository interfaces

Dependency injection



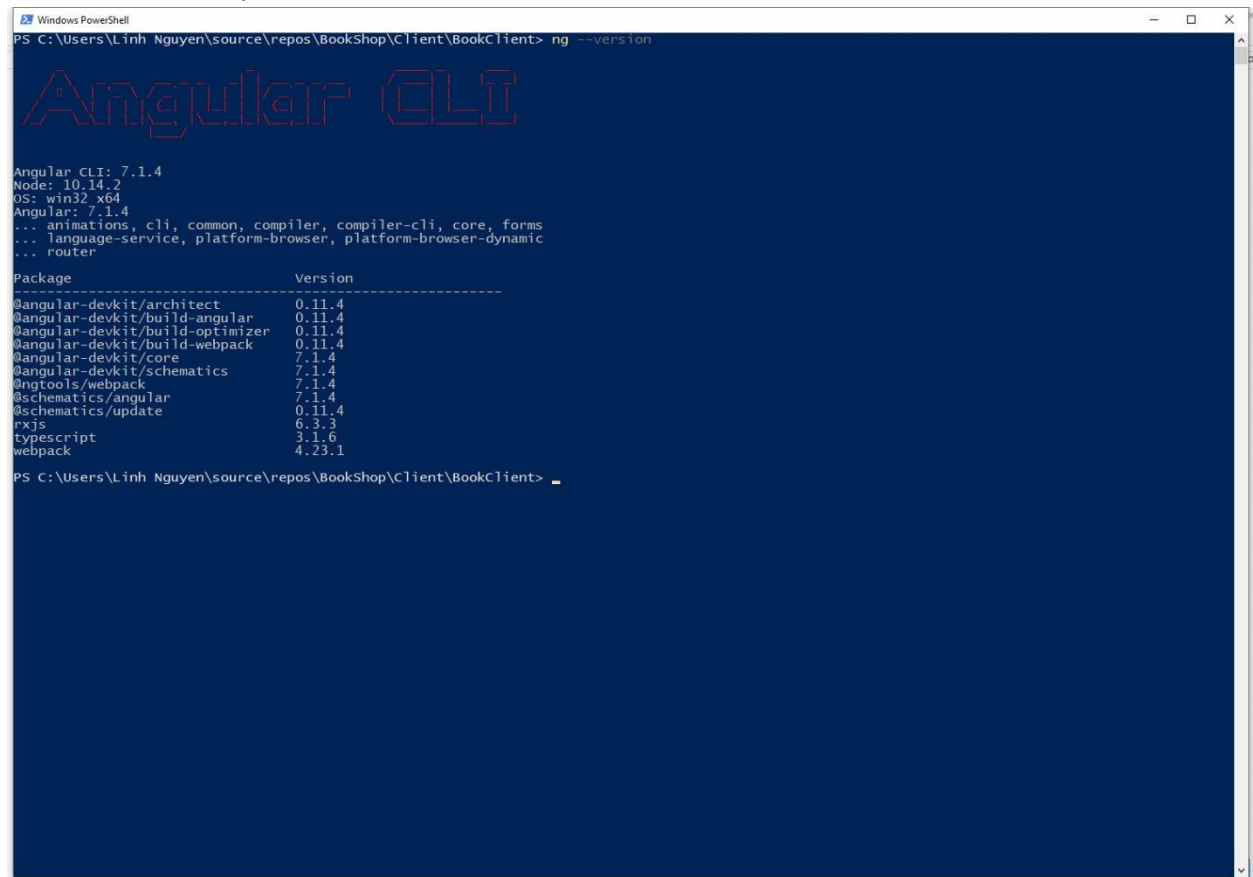
Register services



Reason apply DI

- DI makes it simple for me to manage dependencies between objects.
- Loosely couple architecture. (High Level module and Low Level module)
- Testing can be performed using mock objects.

Frontend Development



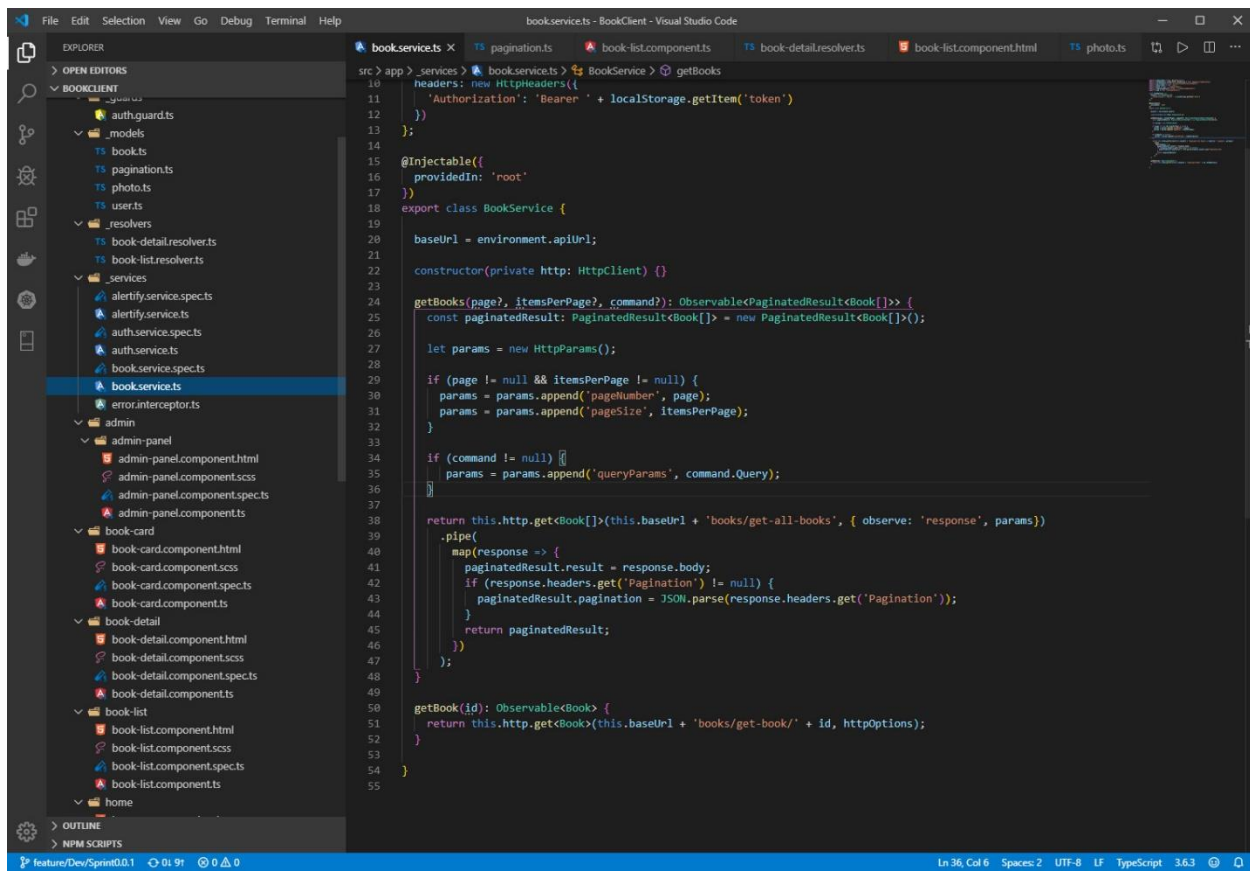
```
Windows PowerShell
PS C:\Users\Linh Nguyen\source\repos\BookShop\Client\BookClient> ng --version

Angular CLI
Angular CLI: 7.1.4
Node: 10.14.2
OS: win32 x64
Angular: 7.1.4
... animations, cli, common, compiler, compiler-cli, core, forms
... language-service, platform-browser, platform-browser-dynamic
... router

Package                                  Version
-----
@angular-devkit/architect                0.11.4
@angular-devkit/build-angular            0.11.4
@angular-devkit/build-optimizer           0.11.4
@angular-devkit/build-webpack            0.11.4
@angular-devkit/core                     7.1.4
@angular-devkit/schematics               7.1.4
@ngtools/webpack                         7.1.4
@schematics/angular                      7.1.4
@schematics/update                       0.11.4
rxjs                                     6.3.3
typescript                               3.1.6
webpack                                  4.23.1

PS C:\Users\Linh Nguyen\source\repos\BookShop\Client\BookClient>
```

Check version Apps run: `ng --version`



Using Angular, Typescript, SCSS .

The image shows a screenshot of the Visual Studio Code editor. The top bar displays the menu (File, Edit, Selection, View, Go, Debug, Terminal, Help) and the current file path: `book-list.component.ts - BookClient - Visual Studio Code`.

The left sidebar shows the Explorer view with a project structure for `BOOKCLIENT`. The file `book-list.component.ts` is selected under the `book-list` folder.

The main editor area displays the content of `book-list.component.ts`. The code is a TypeScript class for a component that interacts with a book service. It includes a `route` property, an `ngOnInit` method, a `pageChanged` method, and a `loadBooks` method.

The bottom panel shows the Output view with the following content:

```
! node
C:\Users\Linh Nguyen\source\repos\BookShop\Client\BookClient>clean
'clean' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Linh Nguyen\source\repos\BookShop\Client\BookClient>serve
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2019-11-18T12:55:08.749Z
Hash: 902a032efe44d77f6cd
Time: 9900ms
chunk (main) main.js, main.js.map (main) 73.5 kB [initial] [rendered]
chunk (polyfills) polyfills.js, polyfills.js.map (polyfills) 237 kB [initial] [rendered]
chunk (runtime) runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk (styles) styles.js, styles.js.map (styles) 1.56 MB [initial] [rendered]
chunk (vendor) vendor.js, vendor.js.map (vendor) 5.3 MB [initial] [rendered]
! fadn: Compiled successfully.
[]
```

The screenshot shows the Visual Studio IDE with the 'BookShop' project selected in the Solution Explorer. The project file 'BookShop.csproj' is open in the editor, showing the 'Project Sdk' and 'PackageReference' elements. A red arrow points from the 'BookShop' project in the Solution Explorer to the 'Project Sdk' element in the project file.

Book Shop

Username

admin

Password

password

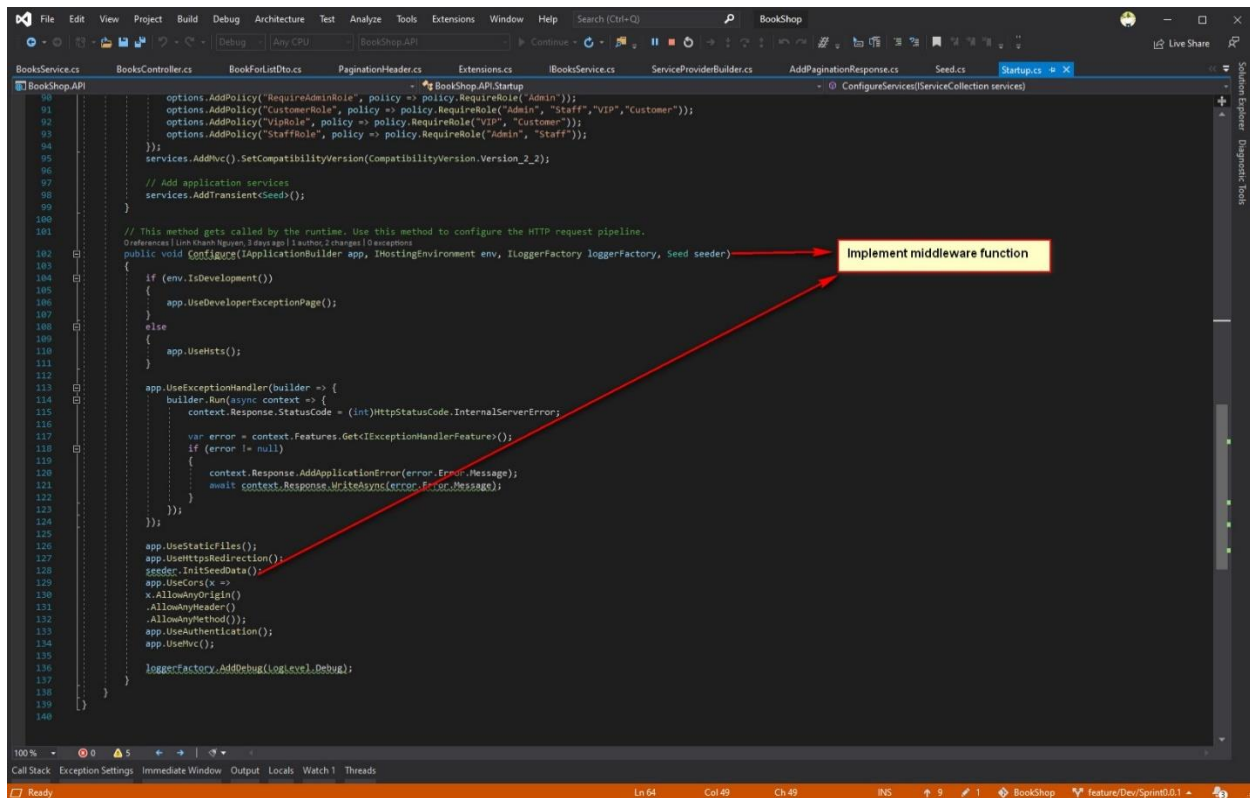
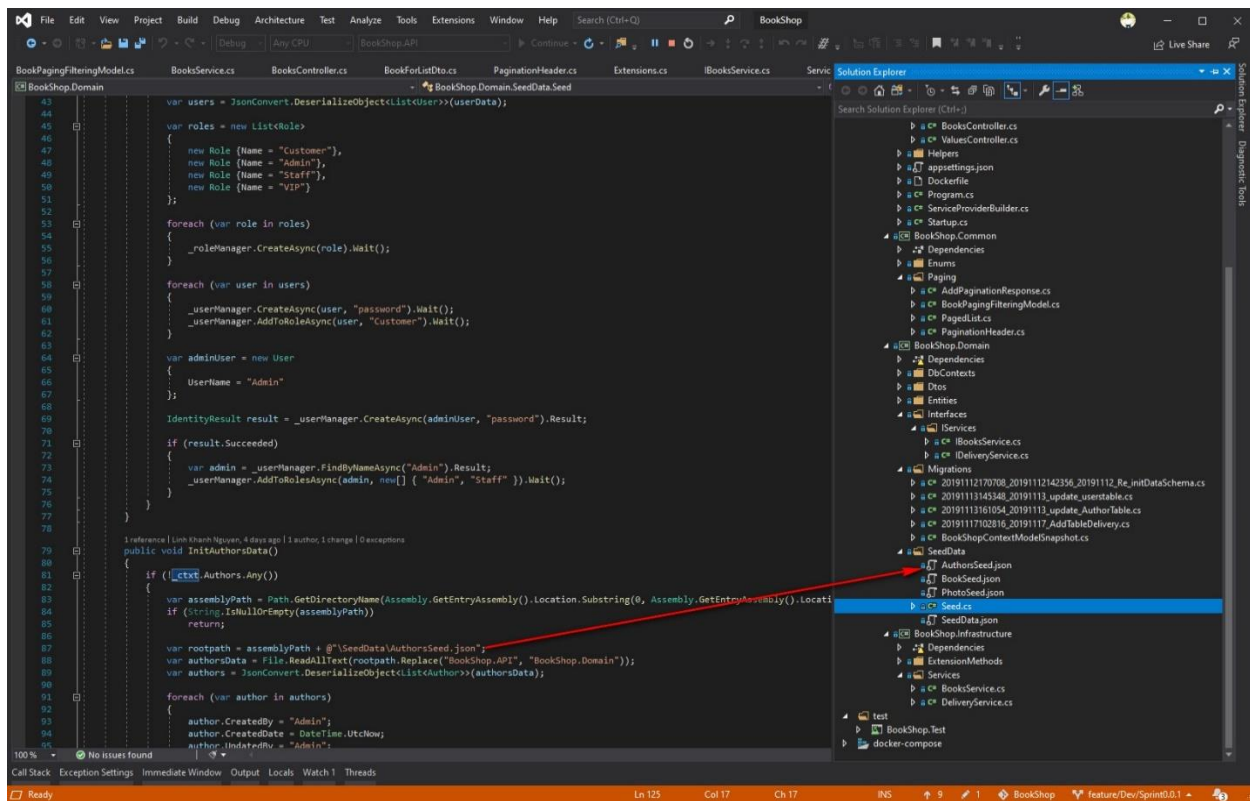
Login

Find your Books

A lot of book wait for you... All you need to do is click List Book menu..!

Learn more

With the first time login user input username and password, this is an extended features.



Search by book title or author

Search



Total spent time

Design and implement database schema: 30 minutes

Design and implement backend structure 1 h

Implement Entities class and Db-context 1h

Implement data migrations and initial data from Json files then insert to database using code first 1h

Implement Class and services business login then register them (DI) 2h

Implement DTO Class and Auto Mapper 30 minutes

Implement Controller then implement functions API Restful 1h

Implement authentication backend JWT token 2h

Initial angular and create services and components angular apps html, CSS 6h

Integration FE and BE testing around 1h

Summary 16 hours

Thank you very much!

