Library Management

Name : CrudAPi

Revision: 1

Date: 03/21/2022

Author: Rishu Ranjan

For further information please contact:   
  
**Rishu Ranjan**

[22rishuranjan@gmail.com](mailto:22rishuranjan@gmail.com)

+91-8287599985

Table of contents

[1. Introduction 2](#_Toc98786259)

[2. Project StRucture 2](#_Toc98786260)

[3. Project DeTails 2](#_Toc98786261)

[3.1. API Project 2](#_Toc98786262)

[3.2. RunTIme Project 6](#_Toc98786263)

[3.3. Domain 6](#_Toc98786264)

[3.4. Persistence 7](#_Toc98786265)

[4. API information 7](#_Toc98786266)

[4.1. GET List for user 7](#_Toc98786267)

[4.2. Get USER BY ID 8](#_Toc98786268)

[4.3. POST Activity 8](#_Toc98786269)

[4.4. Delete API 10](#_Toc98786270)

[4.5. PUT API 10](#_Toc98786271)

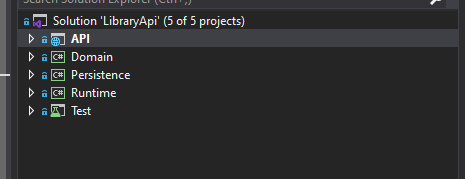
# Introduction

Created a CRUD API for activities. This document covers only the backend part.

# Project StRucture

This project contains of 4 layers,

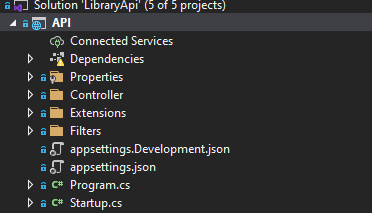
1. **Runtime** – It is the service layer. It interacts directly with controller, and execute work on its behalf. It is the interface between API and the data layer.
2. **API** – It is an API Project. Controller, middleware, services, connection strings are defined here.
3. **Domain** – Class Project, It contains the model definition.
4. **Persistence** – It implements the entity framework core. Dbcontext and related information is implemented here.



# Project DeTails

## API Project

Folder Structure



##### Controllers Folder

* 1. User Controller –

It has following API’s

Crud Apis

1. Get user list - GET Method - https://localhost:44378/api/user
2. Get userbyid - GET Method - https://localhost:44378/api/user/{id}
3. Delete user - DELETE Method - https://localhost:44378/api/user/{id}
4. Update user - PUT Method - https://localhost:44378/api/user/{id}
5. Create user - POST Method - <https://localhost:44378/api/user>

Driver Apis

1. User than issue most books

|  |
| --- |
| GET Method - available routes are below   1. <https://localhost:44378/api/user/mostissued> 2. https://localhost:44378/api/user/mostborrowed |

1. User than issue most books in give number of days

|  |
| --- |
| GET Method, - available routes are below   1. [https://localhost:44378/api/user/mostissued/{days}](https://localhost:44378/api/user/mostissued/%7bdays%7d) 2. [https://localhost:44378/api/user/mostissuedbytime/{days}](https://localhost:44378/api/user/mostissuedbytime/%7bdays%7d) 3. https://localhost:44378/api/user/mostborrowed/{days} |

* 1. Book Controller

It has following API’s

**Crud Apis**

1. Get book list - GET Method - https://localhost:44378/api/book
2. Get bookbyid - GET Method - https://localhost:44378/api/book/{id}
3. Delete book - DELETE Method - https://localhost:44378/api/book/{id}
4. Update book - PUT Method - https://localhost:44378/api/book/{id}
5. Create book - POST Method - <https://localhost:44378/api/book>

Driver Apis

1. Most issued books

|  |
| --- |
| GET Method - available routes are below   1. <https://localhost:44378/api/book/mostissued> 2. https://localhost:44378/api/book/mostborrowed |

1. most issued books in give number of days

|  |
| --- |
| GET Method, - available routes are below   1. [https://localhost:44378/api/book/mostissued/{days}](https://localhost:44378/api/book/mostissued/%7bdays%7d) 2. [https://localhost:44378/api/book/mostissuedbytime/{days}](https://localhost:44378/api/book/mostissuedbytime/%7bdays%7d) 3. https://localhost:44378/api/book/mostborrowed/{days} |

1. Get other issued book by user, if book id and user id is passed

|  |
| --- |
| GET Method - available routes are below   1. https://localhost:44378/api/book/getotherbooks/{bookid}/{userid} |

1. check if book is available and get information on no of available copy

|  |
| --- |
| GET Method, - available routes are below   1. [https://localhost:44378/api/book/available/{bookid}](https://localhost:44378/api/book/available/%7bbookid%7d) 2. https://localhost:44378/api/book/bookavailable/{bookid} |

* 1. Issue Controller

It has following API’s

Crud Apis

1. Get issue list – GET Method - https://localhost:44378/api/issue
2. Get issuebyid – GET Method - https://localhost:44378/api/issue/{id}
3. Create issue – POST Method - https://localhost:44378/api/issue

Driver Apis

1. Get issue list for a book

|  |
| --- |
| GET Method - available routes are below   1. [https://localhost:44378/api/issue/getissuebybookid/{bookid}](https://localhost:44378/api/issue/getissuebybookid/%7bbookid%7d) |

1. Get issue list for a user

|  |
| --- |
| GET Method, - available routes are below   1. [https://localhost:44378/api/issue/getissuebyuserid/{userid}](https://localhost:44378/api/issue/getissuebyuserid/%7buserid%7d) |

* 1. Return Controller

It has following API’s

**Crud Apis**

1. Get return list - GET Method - https://localhost:44378/api/return
2. Get returnbyid - GET Method - https://localhost:44378/api/return/{id}
3. Create return - POST Method - <https://localhost:44378/api/return>

DriverApis

1. Get read rate of a book, that is returned

|  |
| --- |
| GET Method - available routes are below   1. https://localhost:44378/api/return/ReadRate/{ReturnId} |

* 1. BaseApiContreoller – contains operations common to all API.

b. Extension Folder – *To simplify startup.cs class*

1. ApplicationMiddlewareExtenstion – All the middleware services are added here
2. ApplicationServiceExtension – all the services for dependency injection are added here.

##### c. Filter Folder

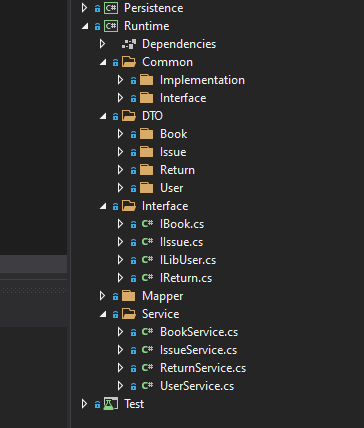
a. ApiExceptionFilter– to check for any exception after action execution.

## RunTIme Project

Methods implemented by Application Service project.

* For each controller there is a service class in runtime project
* Common folder contains interface/implementation for common methods
* DTO folder respective dto for each models for different operation – get, create, updates
* Mapper contains logic to map the dto with the actual domain class

Folder Structure:



Loosly coupled design, each service class is implement thorough respective interface.

## Domain

Domain project inclued two model class.

1. User.cs - model for activity
2. Book.cs – model for books
3. Issue.cs – model for issued books, bookid and user id are foreign key here
4. Return.cs – model for returned books. Issue id is foreign key here
5. ApiResponse.cs – to handle reponse object from API

## Persistence

* This project contains, details about datacontext class and seed.cs class
* Seed.cs will populate data for all the classes initially

# API information For USer Controller

## GET List for user

Endpoint - *https://localhost:44378/api/user*

Response Format –

|  |
| --- |
| {      "data": [          {              "userId": 6,              "name": "Alok Kumar",              "email": "alok.kumar@gmail.com",              "mobile": "+91-882645638",              "city": "Mumbai",              "fullAddress": "City Center,Mumbai, 820001 "          },          {              "userId": 7,              "name": "Aditi Sharma",              "email": "aditi.sharma@gmail.com",              "mobile": "+91-972645638",              "city": "Bengalore",              "fullAddress": "221 Street, Bengalore, 220021"          },          {              "userId": 8,              "name": "John Cena",              "email": "ucantseeme@gmail.com",              "mobile": "+1-72645638",              "city": "Callifornia",              "fullAddress": "st. Lois Street, California, USA"          },          {              "userId": 9,              "name": "S Holmes",              "email": "holmes@gmail.com",              "mobile": "+44-72645638",              "city": "london",              "fullAddress": "221 Baker Street, London, UK"          },          {              "userId": 10,              "name": "J Connor",              "email": "connor@outlook.com",              "mobile": "+44-72645638",              "city": "london",              "fullAddress": "BS Garden, Manchester, UK"          }      ],      "message": "List fectched!!",      "success": **true**,      "status": 200  } |

## Get USER BY ID

Endpoint - *https://localhost:44378/api/user/6*

Response-

|  |
| --- |
| {      "data": {          "userId": 6,          "name": "Alok Kumar",          "email": "alok.kumar@gmail.com",          "mobile": "+91-882645638",          "city": "Mumbai",          "fullAddress": "City Center,Mumbai, 820001 "      },      "message": "User found!!",      "success": **true**,      "status": 200  } |

## Create User

Endpoint - *https://localhost:44378/api/user*

Body –

|  |
| --- |
| {              "name": "Rishu R",              "email": "connor@outlook.com",              "mobile": "+91-8287599985",              "city": "delhi",              "fullAddress": "BS Garden, delhi, India"     } |

Response Format –

|  |
| --- |
| {      "data": [          {              "userId": 6,              "name": "Alok Kumar",              "email": "alok.kumar@gmail.com",              "mobile": "+91-882645638",              "city": "Mumbai",              "fullAddress": "City Center,Mumbai, 820001 "          },          {              "userId": 7,              "name": "Aditi Sharma",              "email": "aditi.sharma@gmail.com",              "mobile": "+91-972645638",              "city": "Bengalore",              "fullAddress": "221 Street, Bengalore, 220021"          },          {              "userId": 8,              "name": "John Cena",              "email": "ucantseeme@gmail.com",              "mobile": "+1-72645638",              "city": "Callifornia",              "fullAddress": "st. Lois Street, California, USA"          },          {              "userId": 9,              "name": "S Holmes",              "email": "holmes@gmail.com",              "mobile": "+44-72645638",              "city": "london",              "fullAddress": "221 Baker Street, London, UK"          },          {              "userId": 10,              "name": "J Connor",              "email": "connor@outlook.com",              "mobile": "+44-72645638",              "city": "london",              "fullAddress": "BS Garden, Manchester, UK"          },          {              "userId": 12,              "name": "Rishu R",              "email": "connor@outlook.com",              "mobile": "+91-8287599985",              "city": "delhi",              "fullAddress": "BS Garden, delhi, India"          }      ],      "message": "Success: User added!",      "success": **true**,      "status": 201  } |

## Delete USER

Endpoint - *https://localhost:44378/api/user/12*

Response Format –

|  |
| --- |
| {      "data": {          "userId": 12,          "name": "Rishu R",          "email": "connor@outlook.com",          "mobile": "+91-8287599985",          "city": "delhi",          "fullAddress": "BS Garden, delhi, India"      },      "message": "User deleted!!",      "success": **true**,      "status": 200  } |

## Update USER

Endpoint -https://localhost:44378/api/user/10

Body –

|  |
| --- |
| {          "userId": 10,          "name": "Jonathan Connor",          "email": "connor@outlook.com",          "mobile": "na",          "city": "london",          "fullAddress": "BS Garden, Manchester, UK"      } |

Response Format –

|  |
| --- |
| {      "data": {          "userId": 10,          "name": "J Connor",          "email": "connor@outlook.com",          "mobile": "+44-72645638",          "city": "london",          "fullAddress": "BS Garden, Manchester, UK"      },      "message": "User found!!",      "success": **true**,      "status": 200  } |

**Note- Similar endpoint/response/body will be for Books, Issues. Returns crud api.**

## Most issued by user

End point - *https://localhost:44378/api/user/MostIssued*

Response Format –

|  |
| --- |
| {      "data": [          {              "userId": 10,              "name": "J Connor",              "email": "connor@outlook.com",              "mobile": "+44-72645638",              "city": "london",              "fullAddress": "BS Garden, Manchester, UK"          }      ],      "message": "List of users that borrowed most book(s) with count 3",      "success": **true**,      "status": 200  } |

## User with most borrowed book in a given time

End point - *https://localhost:44378/api/user/MostIssued/7*

Response Format –

|  |
| --- |
| {      "data": [          {              "userId": 10,              "name": "J Connor",              "email": "connor@outlook.com",              "mobile": "+44-72645638",              "city": "london",              "fullAddress": "BS Garden, Manchester, UK"          }      ],      "message": "User that Borrowed book(s) in given period start date : 3/13/2022 8:15:58 PM, finish date: 3/21/2022 8:15:58 PM with count 3",      "success": **true**,      "status": 200  } |