

BOOKSWAP APPLICATION DESIGN DOCUMENT

Author

Ivan Yaremko

C00239239

Supervisor

Dr Chris Staff

Submission date

17/12/2021



1 ABSTRACT

The purpose of this document is to highlight the necessary design features required to develop BookSwap.

2 CONTENTS

1	Abstract.....	2
4	Table of figures	5
5	Introduction	6
6	Project plan.....	7
7	Application architecture.....	9
7.1	BookSwap architecture design.....	9
7.2	BookSwap architecture patterns	10
7.2.1	CQRS	11
7.2.2	Mediator.....	11
7.3	Client side	12
8	Database Schema.....	13
8.1	Book table	13
8.2	AspNetUsers Table	14
8.3	Swap table	15
8.4	Photos Table.....	15
8.5	Message Table.....	16
9	UI MOCK-UPS	17
9.1	Register.....	17
9.2	Search.....	18
9.3	CRUD Book.....	19
9.4	Book details.....	20
9.5	Swap book	21
9.6	Chat with member	22
10	UML Diagrams	23
10.1	UML Class Diagram.....	23
10.2	Book Sequence Diagrams	24
10.2.1	Create Book Sequence Diagram	24
10.2.2	Update Book Sequence Diagram	24
10.2.3	Delete Book Sequence Diagram	25
10.2.4	Details Book Sequence Diagram.....	25
10.2.5	List Book Sequence Diagram	26
10.2.6	List BooksOwned Sequence Diagram	26
10.3	Swap Sequence Diagrams	27

10.3.2	Update Swap Sequence Diagram	27
10.3.3	Delete Swap Sequence Diagram.....	28
10.3.4	Details Swap Sequence Diagram	28
10.3.5	List Swap Sequence Diagram	29
10.3.6	List SwapsIRequested Sequence Diagram	29
10.3.7	List SwapsRequestedFromMe Sequence Diagram	30
10.4	Profile Sequence Diagrams	30
10.4.1	Get Profile Sequence Diagram	30
10.4.2	Update Profile Sequence Diagram	31
10.5	Photos Sequence diagram.....	31
10.5.1	Add Photo Sequence Diagram	31
10.5.2	Delete Photo Sequence Diagram	32
10.6	SignalR Sequence Diagrams	32
10.6.1	Add Message Sequence Diagram	32
10.6.2	List Message Sequence Diagram.....	33
11	References.....	34
12	Plagiarism Declaration	35

4 TABLE OF FIGURES

Figure 1 Gantt Chart.....	7
Figure 2 Clean Architecture [1].....	9
Figure 3 BookSwap architecture design.....	10
Figure 4 CQRS design	11
Figure 5 Mediator Pattern.....	12
Figure 6 Schema of Book table	13
Figure 7 Scehma of AspNettUsers Table	14
Figure 8 Schema of Swap Table	15
Figure 9 Photo schema	15
Figure 10 Message schema	16
Figure 11 UI Register mock-up	17
Figure 12 UI Search marketplace mock-up	18
Figure 13 UI CRUD book mock-up.....	19
Figure 14 UI of book details mock-up.....	20
Figure 15 UI for swapping books.....	21
Figure 16 UI of chatting with members	22
Figure 17 UML Class Diagram	23
Figure 18 Create Book Sequence Diagram.....	24
Figure 19 Update Book Sequence Diagram.....	24
Figure 20 Delete book sequence diagram.....	25
Figure 21 Details book sequence diagram.....	25
Figure 22 List Book Sequence Diagram.....	26
Figure 23 List BookOwned Sequence Diagram	26
Figure 24 Create Swap Sequence Diagram.....	27
Figure 25 Update Swap Sequence Diagram	27
Figure 26 Delete Swap Sequence Diagram	28
Figure 27 Details Swap sequence Diagram	28
Figure 28 List Swap sequence Diagram.....	29
Figure 29 List SwapsIRequested Sequence Diagram.....	29
Figure 30 List SwapsRequestedFromMe Sequence Diagram.....	30
Figure 31 Get Profile Sequence Diagram.....	30
Figure 32 Update Profile Sequence Diagram.....	31
Figure 33 Add Photo Sequence Diagram.....	31
Figure 34 Delete Photo Sequence Diagram.....	32
Figure 35 Add Message Sequence Diagram.....	32
Figure 36 List Message Sequence Diagram	33

5 INTRODUCTION

The following document will demonstrate the design of the BookSwap application. This document is aimed at providing a better understanding of the design of this application.

The following sections will cover application architecture, display the UML context, and class diagrams along with the necessary sequence diagrams.

6 PROJECT PLAN

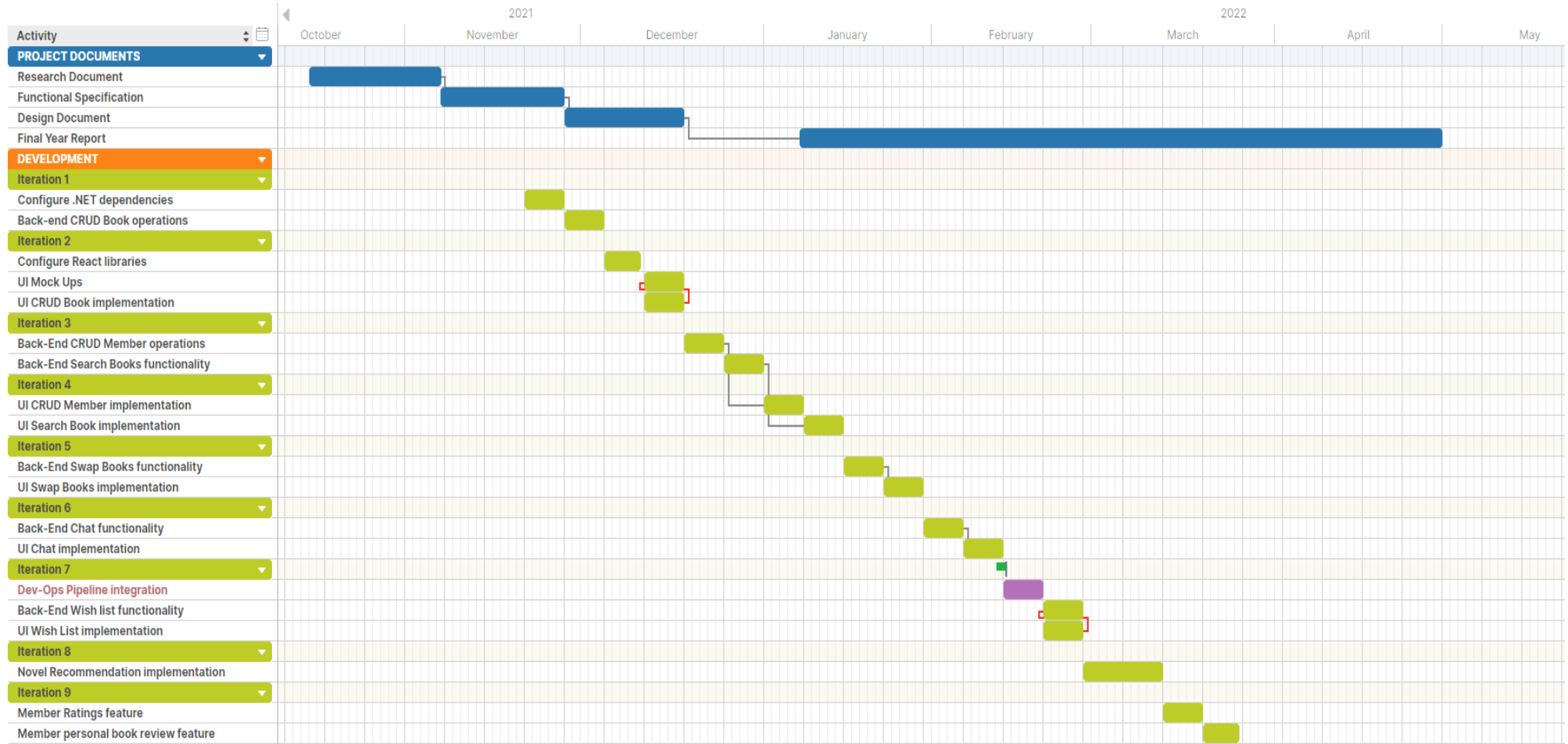


Figure 1 Gantt Chart

Figure 1 illustrates the estimated timeline for this project through a Gantt chart.

The development process is planned in iterations, where each iteration is two weeks long. Each iteration works on improving and adding core functionalities of the project.

The first six iterations focus on implementing the core functionalities as described in the 'Functional Specification' document:

- **Iteration 1.** Works on configuring the .NET dependencies and structuring the application projects into clean architecture. The CQRS and Mediator pattern are also implemented into this iteration along with the CRUD Books functionality.
- **Iteration 2.** The focus for this iteration is to configure the front-end UI libraries, creating mock-ups of the UI and implementing the UI for the CRUD Books.
- **Iteration 3.** The plan for this iteration is to focus on implementing two core functionalities: CRUD Members, along with login, registration, authentication, and Searching for books in the marketplace.
- **Iteration 4.** This iteration focuses in creating the UI for the two core functionalities implemented in the last iteration.
- **Iteration 5.** In this iteration one core functionality is focused on. The swapping of books mechanism, the mechanisms will be implemented in the back end and the UI for it in the front end.
- **Iteration 6.** The core functionality of chatting with other member is focused on in this iteration.

Completing iteration 6 marks a milestone for the project. All the core functionalities required for this project are implemented in local development. Iteration 7 focuses on creating a development operations (DevOps) pipeline for the project. The goal would be to create a continuous integration / continuous deployment process to build, test and deploy onto a live website.

With a DevOps pipeline configured, the remaining iterations focus on developing the non-core functionalities. With each additional feature developed the project can undergo a DevOps process, where the added feature is pushed to a repository, the codebase is tested, build, and then deployed to a live website.

7 APPLICATION ARCHITECTURE

Clean architecture [1] is used to organise the codebase of the project. The purpose of clean architecture is to keep the codebase under control. This is achieved by ensuring that the application code and logic are written so that they don't have any direct dependencies to each other. There should not be any changes to the core of the system if the framework or UI changes. In this way the core of the system is protected and makes external dependencies completely replaceable.

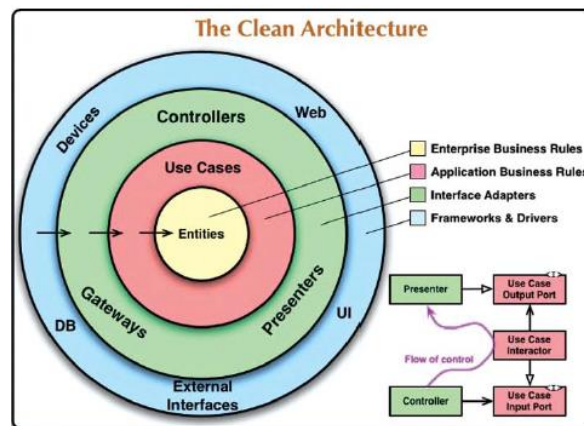


Figure 2 Clean Architecture [1]

Figure 2 illustrates the design of clean architecture. The circles in the figure represent different areas of software.

- The inner circles are the policies.
- The outer circles are the mechanisms.
- The more inwards you go the higher level the software becomes.

BookSwap is designed using the clean architecture along with supporting patterns to facilitate this.

7.1 BOOKSWAP ARCHITECTURE DESIGN

Figure 3 represents BookSwap's clean architecture design. The design follows the dependency rule of clean architecture.

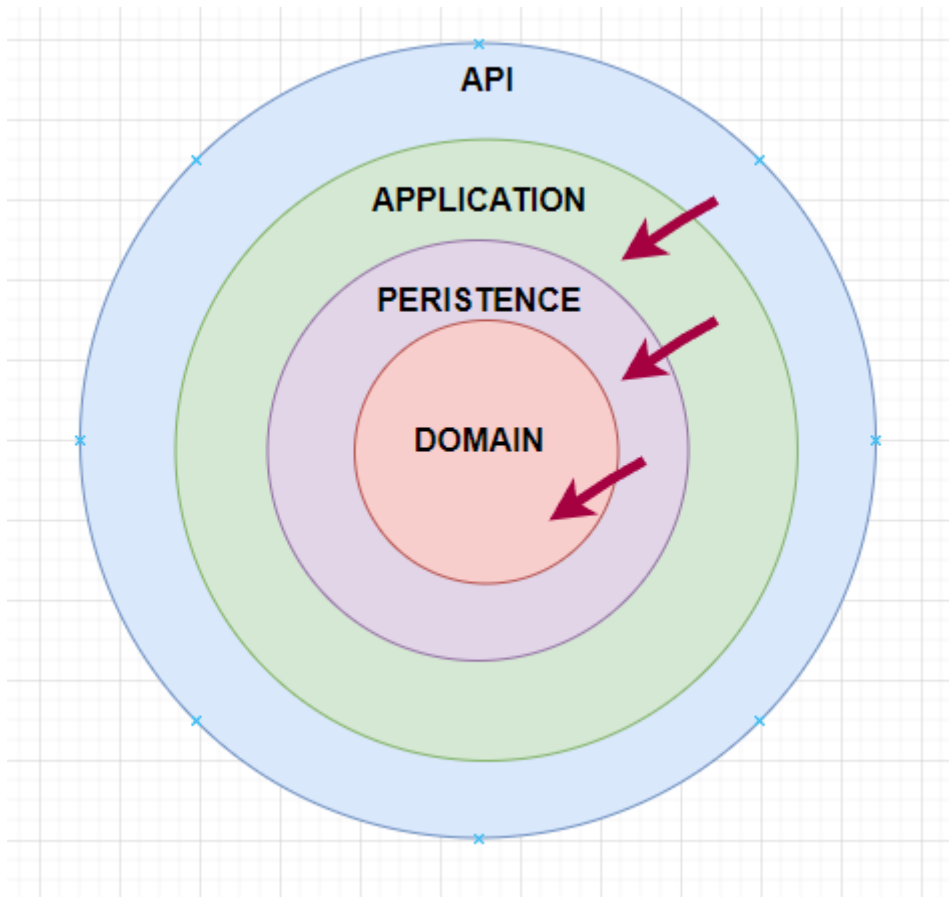


Figure 3 BookSwap architecture design

The circles of Figure 3 are explained below:

- The domain layer contains the enterprise logic.
- The application layer contains the business logic.
- The application layer has a dependency on the domain. Any of the domain entities will be accessible in the application layer.
- The persistence layer which is outside the application but has a dependency on the domain layer provides the connection to the database and translates written code into SQL queries.
- The applications layers job is to use the domain entities, query the database through the dependency on the persistence layer and return the logic to the API.
- The API layer is responsible for the HTTP requests and responds to them.
- The API layer has a dependency on the application layer where the business logic is processed.
- The API has access to both the domain and persistence layer through its transitive dependency via the application layer.
- The infrastructure layer is also outside the application but has a dependency on the application layer and has access to its classes.

7.2 BOOKSWAP ARCHITECTURE PATTERNS

The back-end application of BookSwap follows two patterns:

- 1) CQRS
- 2) Mediator

7.2.1 CQRS

The Command and Query Responsibility Segregation [2] (CQRS) pattern is used to separate the create/update and read operations for a data store.

Create/Update operations are known as commands in this pattern and reads are queries. CQRS separates the commands and queries, commands do something with the database and queries read data from the database.

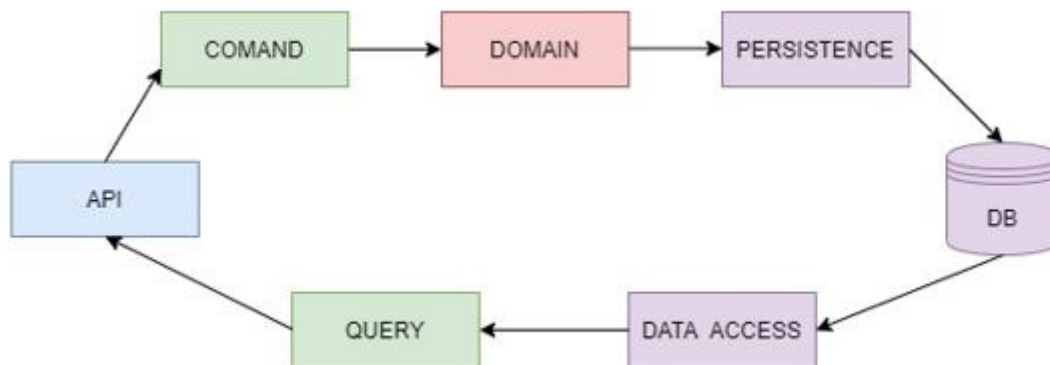


Figure 4 CQRS design

Figure 5 illustrates the CQRS cycle, when a client makes a request to the backend application the API is either going to:

- Make a command request to do something with the domain entities and use the persistence layer to update the database.
- Make a query to retrieve data using the data access to query the database.

7.2.2 Mediator

The mediator pattern is used to mediate between the different layers in the clean architecture design of this application. In the bottom right of Figure 2, clean architecture also specifies the flow of control. Referring with the figure, the green boxes represent the API layer, and the red boxes represent the application layer.

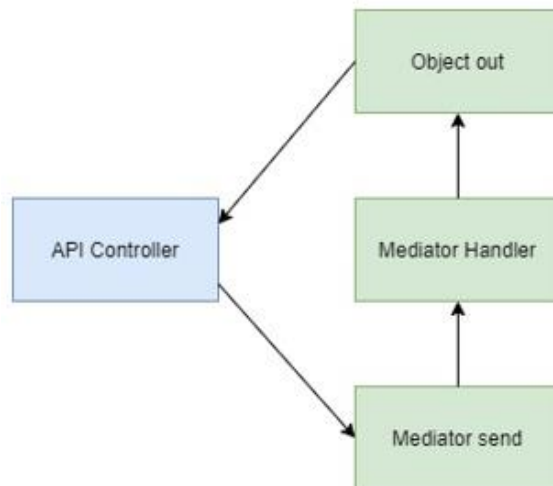


Figure 5 Mediator Pattern

The flow of control will be implemented using a tool called MediatR [3]. This tool allows process management and supports request/response, command, and queries. Figure 5 illustrates how BookSwap implements the mediator pattern. The API controller resides in the API layer and the handler in the application layer.

For example, a client wants to retrieve a book from the database. This query would be sent via the mediator send method to the mediator handler where it is going to process the business logic and return the object back via the API controller.

In this way when a request comes to the API controller:

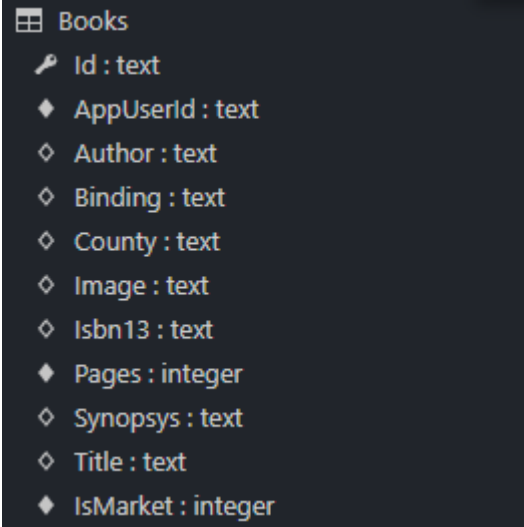
- The API controller will send this request via the mediator send method.
- The request could be either a query or a command.
- The mediator will send this request to the mediator handler.
- The handler will handle the use case.
- The handler will also return an object to the API controller.

7.3 CLIENT SIDE

For the client side of the application BookSwap uses React [4]. The client side is the UI of BookSwap, often referred to as the front end. The end-users, members, of the application use the client to communicate with the backend server.

8 DATABASE SCHEMA

8.1 BOOK TABLE



The image shows a dark-themed screenshot of a database schema for a table named 'Books'. The table has the following columns:

Column Name	Data Type	Primary Key
Id	text	Yes
AppUserId	text	No
Author	text	No
Binding	text	No
County	text	No
Image	text	No
Isbn13	text	No
Pages	integer	No
Synopsys	text	No
Title	text	No
IsMarket	integer	No

Figure 6 Schema of Book table

The book table is used to store books in the database for the application. This table stores information on books such as:

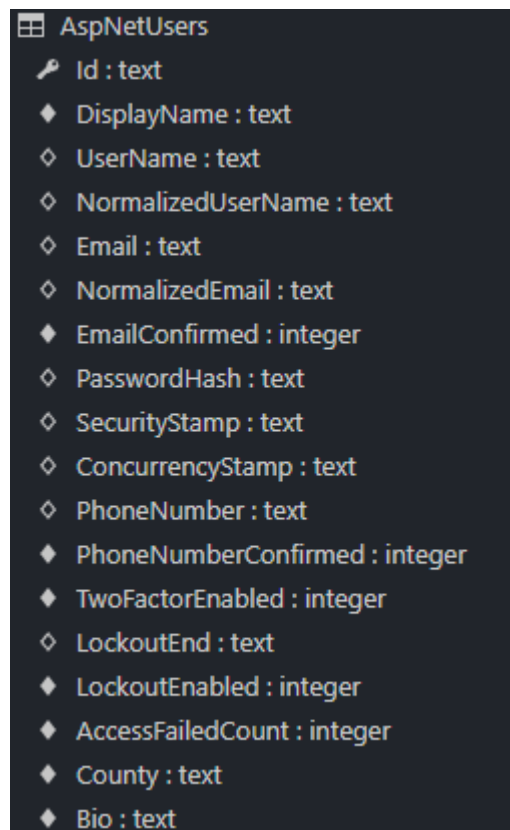
- Book Id
- Owner of Book Id
- Author
- Title
- Synopsys
- Pages

- Binding
- ISBN13
- imageURL

When a member is adding books to the marketplace, this table will be used to populate the data store. The image URL will be provided by ISBNdb [5] API. The AppUserId references the User that owns that book.

8.2 ASPNETUSERS TABLE

The AspNetUsers table is used to store information on the users of the application into the database. This table stores information on members such as:



The image shows a screenshot of a database tool displaying the schema for the AspNetUsers table. The table name is 'AspNetUsers'. The columns and their data types are listed as follows:

Column Name	Data Type
Id	text
DisplayName	text
UserName	text
NormalizedUserName	text
Email	text
NormalizedEmail	text
EmailConfirmed	integer
PasswordHash	text
SecurityStamp	text
ConcurrencyStamp	text
PhoneNumber	text
PhoneNumberConfirmed	integer
TwoFactorEnabled	integer
LockoutEnd	text
LockoutEnabled	integer
AccessFailedCount	integer
County	text
Bio	text

Figure 7 Schema of AspNetUsers Table

This table is auto generated by the .NET Identity library [6] when creating a migration. The only customised column included is the “DisplayName”.

When a user registers with the application, their credentials will be stored in the data store by using this table.

8.3 SWAP TABLE

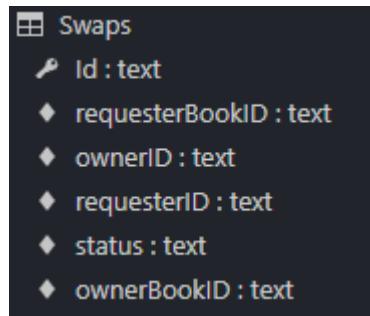


Figure 8 Schema of Swap Table

When a request is made the a swap is created in the database:

- OwnerBookID; The book ID of the book that the requester member making the swap.
- OwnerID: The ID of the owner of the book the requester member making the swap wants
- RequesterID: The ID of the member who is making the request
- Status: The status is used to identify the current state of the swap, when a swap is first created, the status is set to “request”, if the requestee member picks a book to swap from the requester list then the status is updated to “confirmed”.
- RequesterBookID: The ID of the book the ownerID has selected from the book list of the the requester.

For the swapping books use case, the requestee member must select a book from a list of books that the requester member owns. Once the requestee selects a book the swap is confirmed. However, if the requestee does not want to swap his book with any of the books the requester owns they can deny the swap.

8.4 PHOTOS TABLE

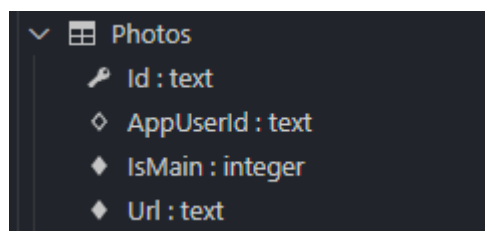
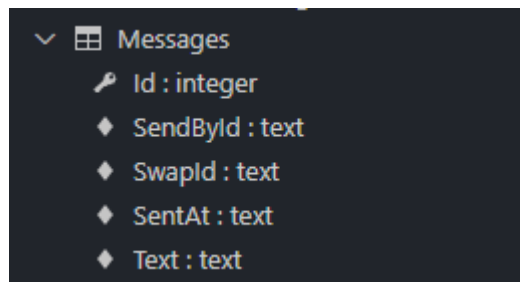


Figure 9 Photo schema

The photos table is used to store members profile Images. The Url is the public url Cloudinary provides. The Photos is an associated ICollection with AppUser.

8.5 MESSAGE TABLE



The image shows a dark-themed interface for a database schema. At the top, there is a dropdown arrow and a table icon followed by the text 'Messages'. Below this, there are five entries, each with a small icon and text: a key icon for 'Id : integer', and diamond icons for 'SendById : text', 'SwapId : text', 'SentAt : text', and 'Text : text'.

Field Name	Field Type
Id	integer
SendById	text
SwapId	text
SentAt	text
Text	text

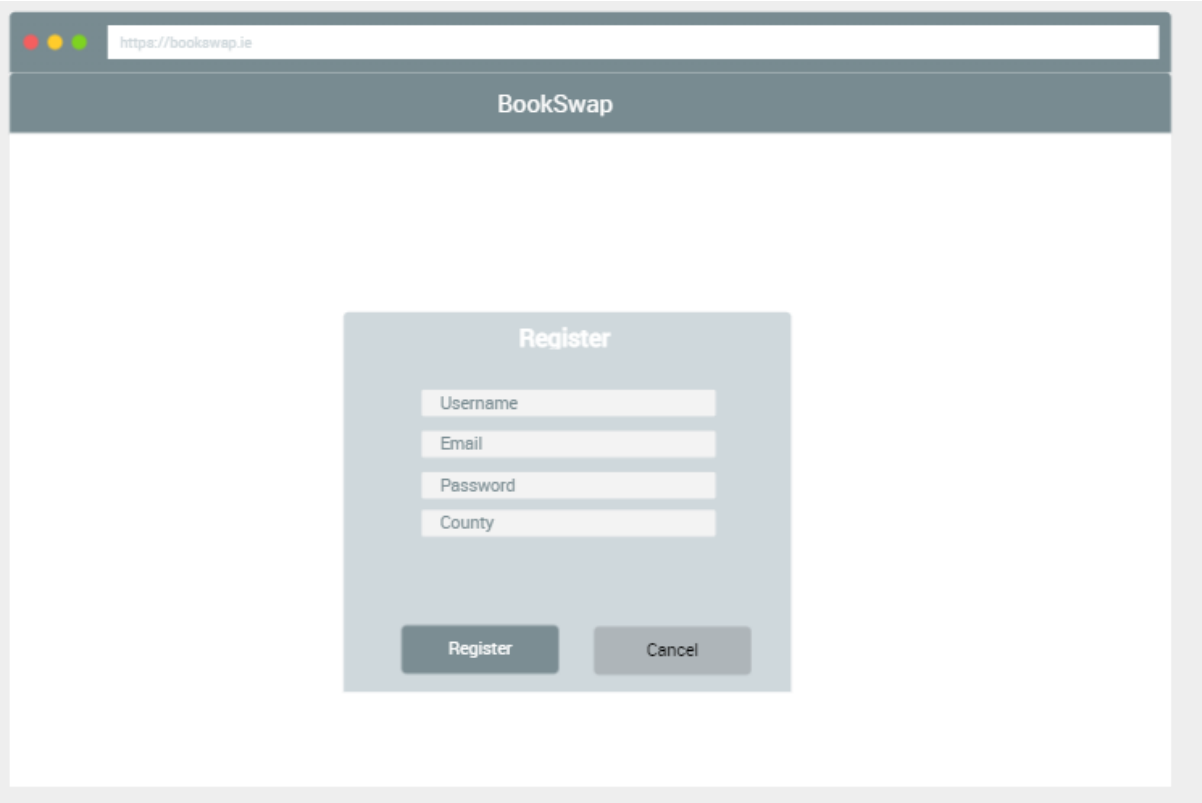
Figure 10 Message schema

The Message table is used to store message member sent to each other. The message table is associated with the Swap table.

9 UI MOCK-UPS

These UI mock-ups are rough designs for the UI of BookSwap. The final UI design of the project will look different compared to these prototypes.

9.1 REGISTER



The image shows a browser window with the URL 'https://bookswap.ie' and a dark header bar labeled 'BookSwap'. In the center, there is a light gray 'Register' form. The form contains four text input fields labeled 'Username', 'Email', 'Password', and 'County'. At the bottom of the form are two buttons: a dark gray 'Register' button and a light gray 'Cancel' button.

Figure 11 UI Register mock-up

Figure 7 illustrates how a user can register with the application. The user must provide a username, email, password, and their county. The user's password will get salted and hashed by the application before storing the information into the database.

Information such as full name, age, and address are omitted due to GDPR concerns; this information is not essential for the use cases of the application

9.2 SEARCH

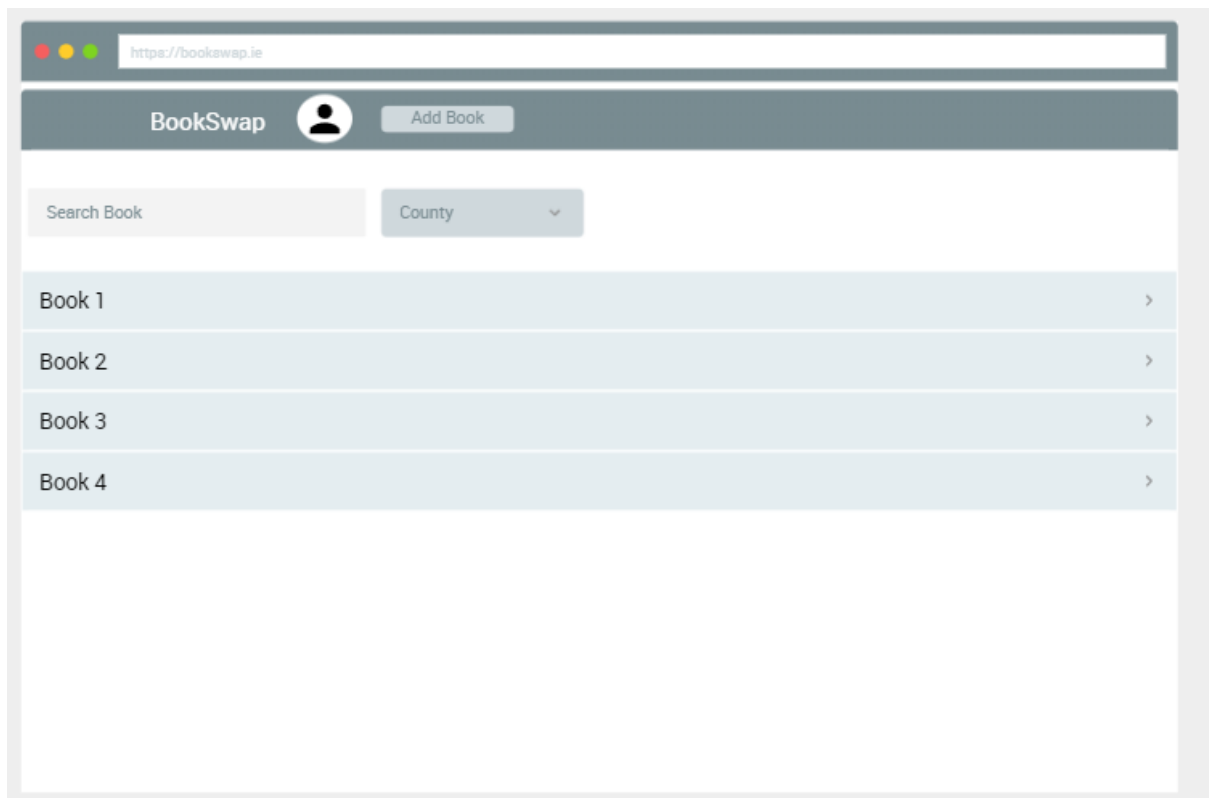


Figure 12 UI Search marketplace mock-up

Figure 8 demonstrates how a member can search for books in the applications marketplace. The county input box will be automatically filled with the member's county. The member may change the location of the county to search for books in other areas. By default, this page will display all books that are on the marketplace for the auto selected county. The member can narrow the list by using the search function. A member can search for a book title, authors name or ISBN.

9.3 CRUD Book

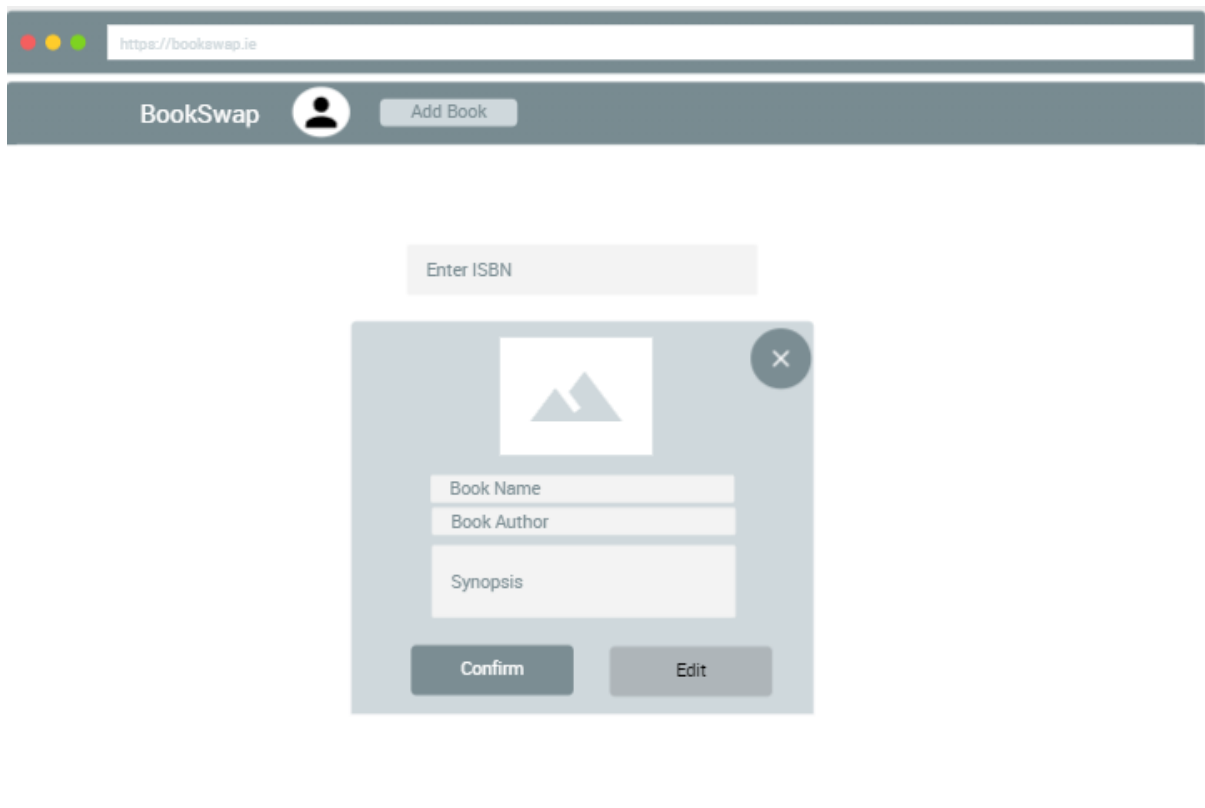


Figure 13 UI CRUD book mock-up

Figure 9 shows a mock-up design of the CRUD book functionality. A member can navigate to this page by clicking on the 'Add Book' button. In this page, the member can input the ISBN of the book they wish to add. The application will use this ISBN to make a request call to ISBNdb to retrieve the details of the books, the application will also auto-fill the necessary details into the form.

9.4 BOOK DETAILS

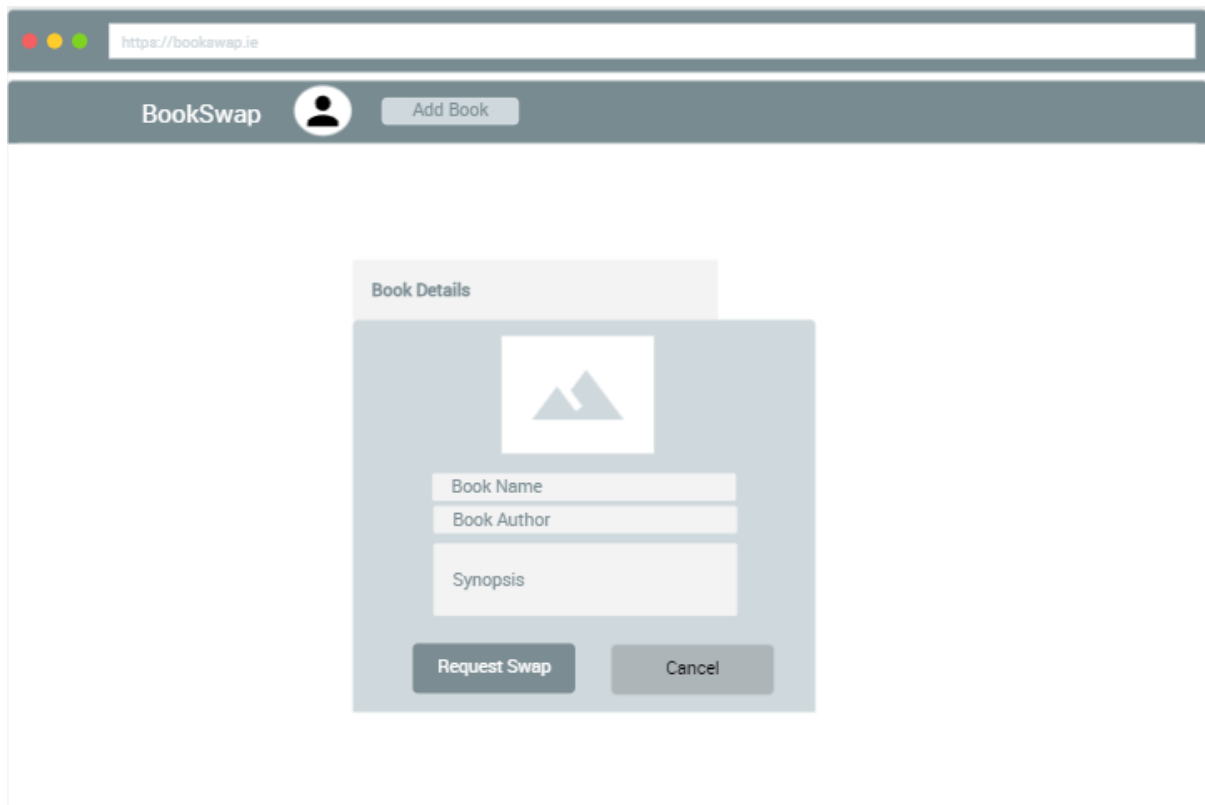


Figure 14 UI of book details mock-up

Figure 10 illustrates the UI mock-up of the book details page. A member navigates this page by selecting a book from the 'Search book' page of the application. In this page a member can see the details of the selected book, as well as request a swap for this book. The requestee member will be notified of the request and can either deny the request or confirm it by selecting a book from the requester's member book list.

9.5 SWAP BOOK

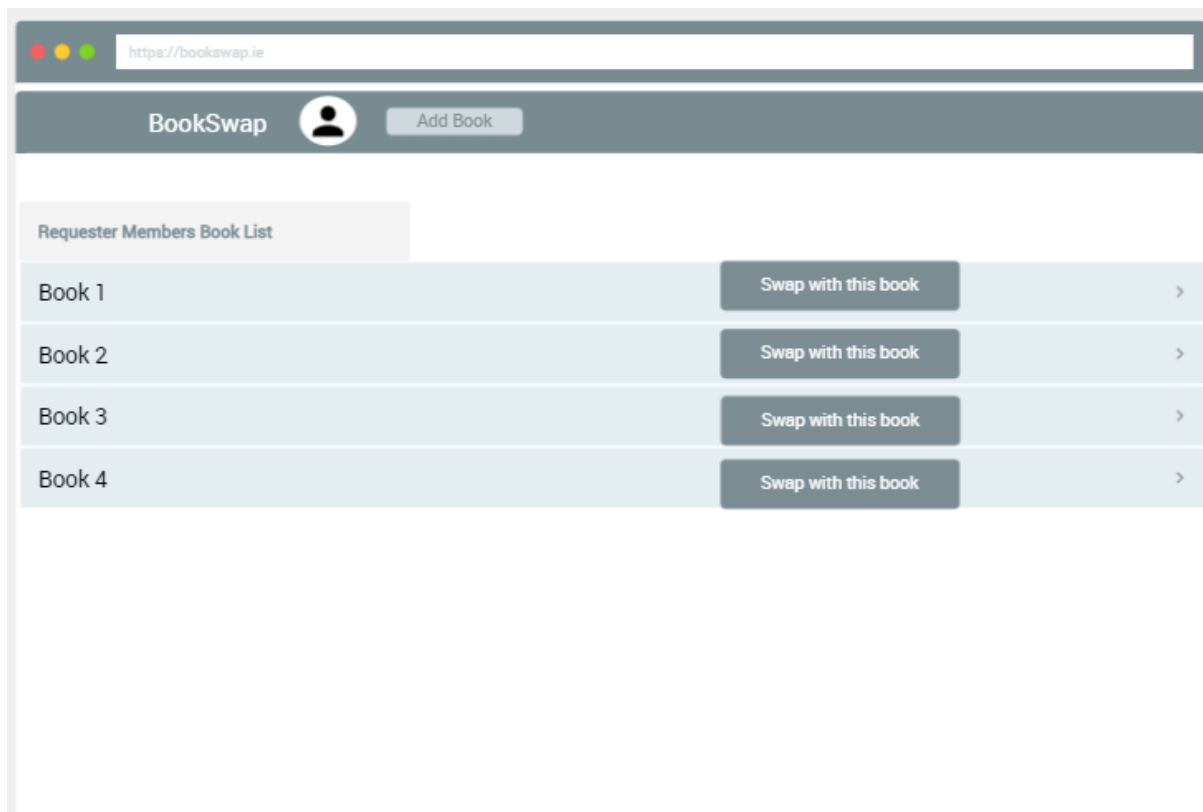


Figure 15 UI for swapping books

The screen shown in figure 11 is accessed by members negotiating a book swap. When another member requests a swap with a book, that book owner member then gets to select a book from a list of books that the requester member owns. Once the book owner member confirms a swap by selecting the “swap with this book” option, the swapped books get removed from the marketplace and the database of the application.

9.6 CHAT WITH MEMBER

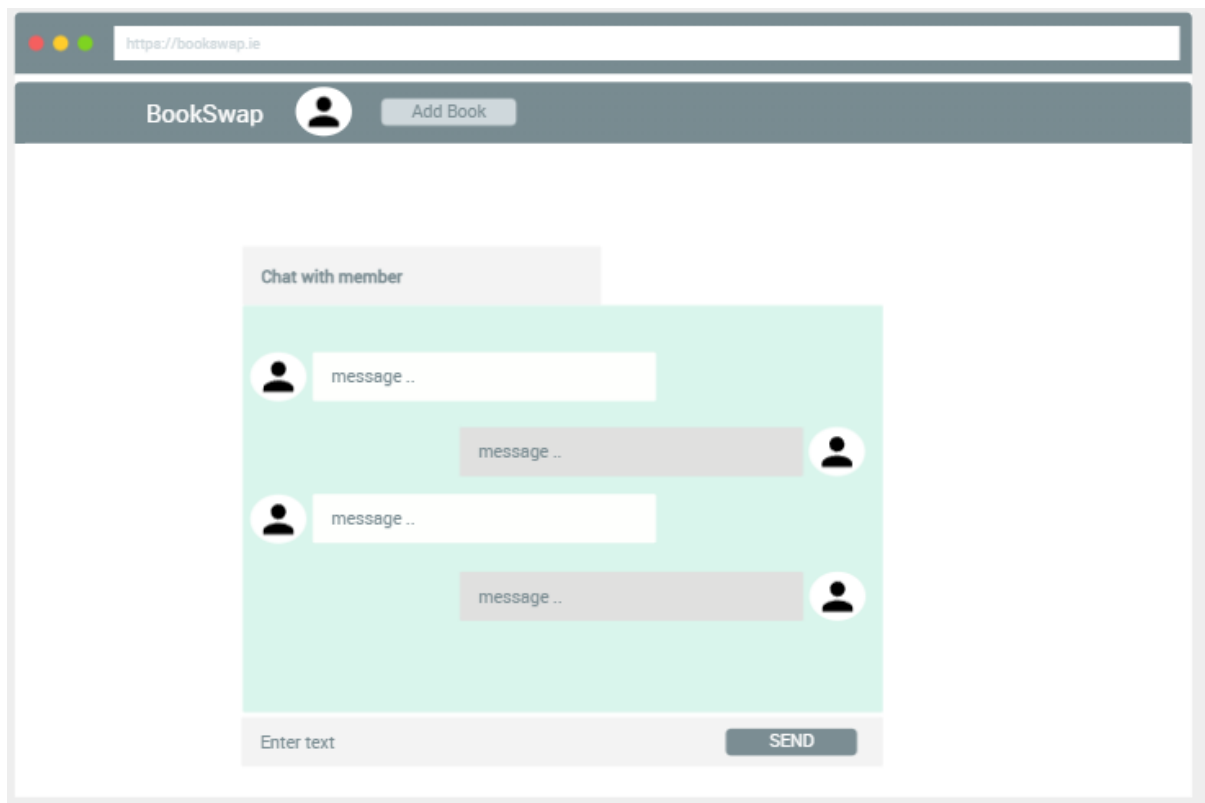


Figure 16 UI of chatting with members

The UI shown in Figure 12 is accessed by members who have confirmed a book swap. This page is used by members to talk with each other. The purpose of the chat system is to help members arrange the swap – it is the responsibility of the members to physically swap books.

10 UML DIAGRAMS

10.1 UML CLASS DIAGRAM

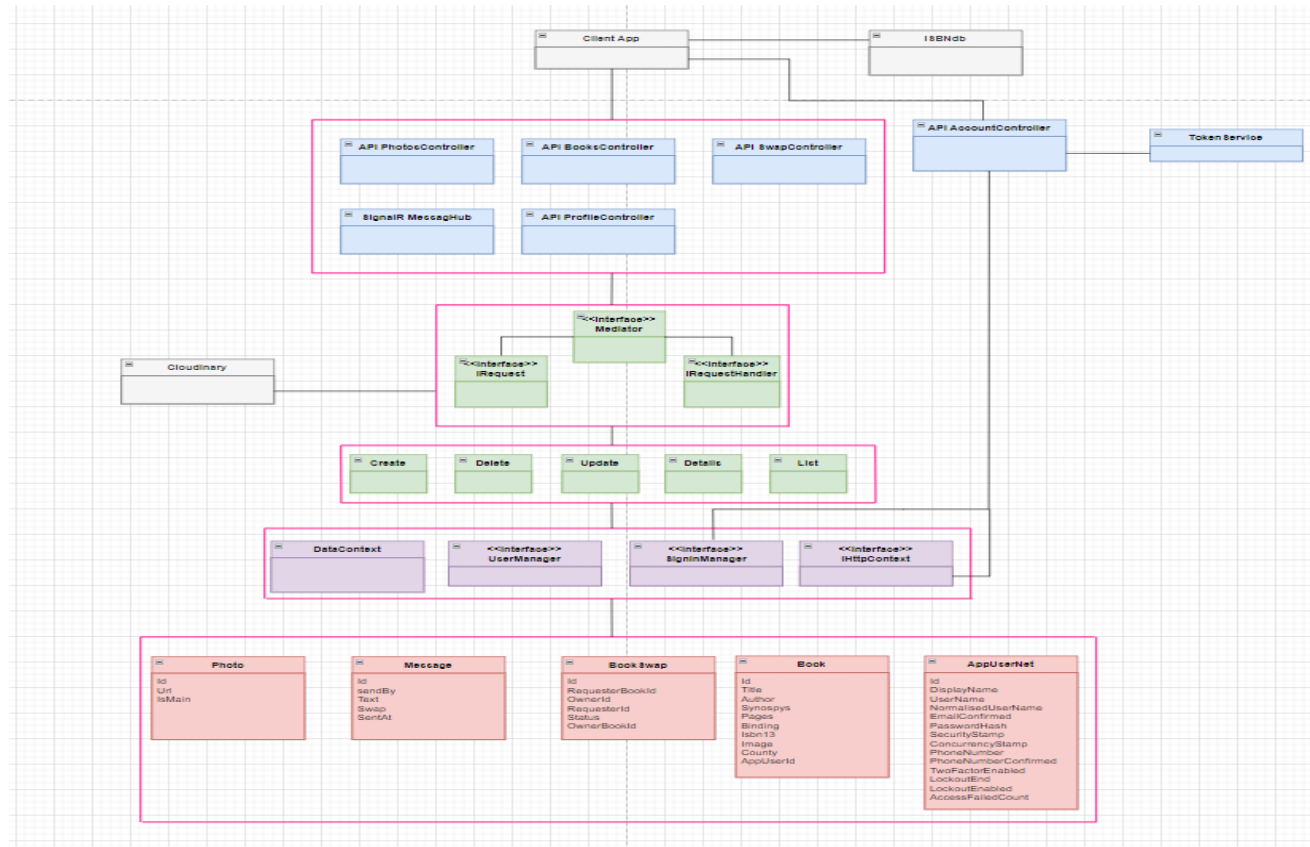


Figure 17 UML Class Diagram

10.2 BOOK SEQUENCE DIAGRAMS

10.2.1 Create Book Sequence Diagram

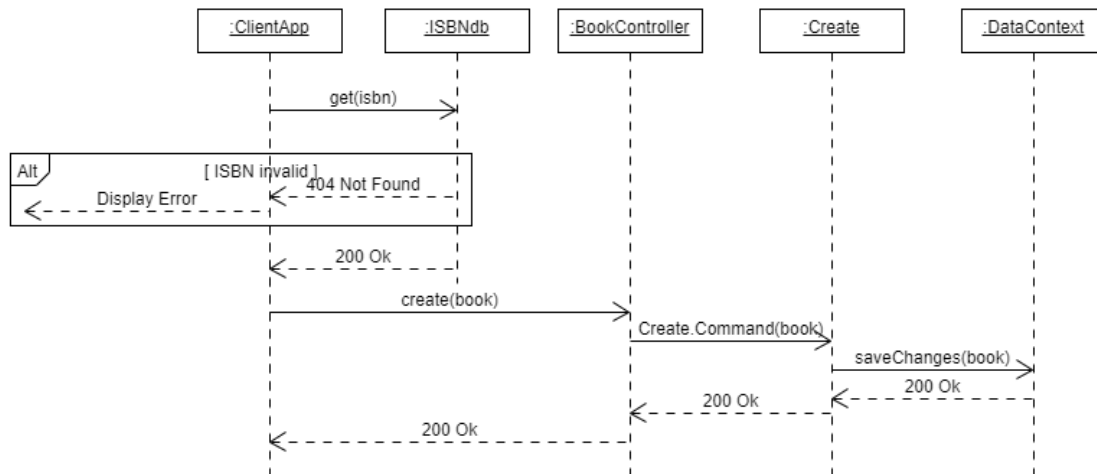


Figure 18 Create Book Sequence Diagram

10.2.2 Update Book Sequence Diagram

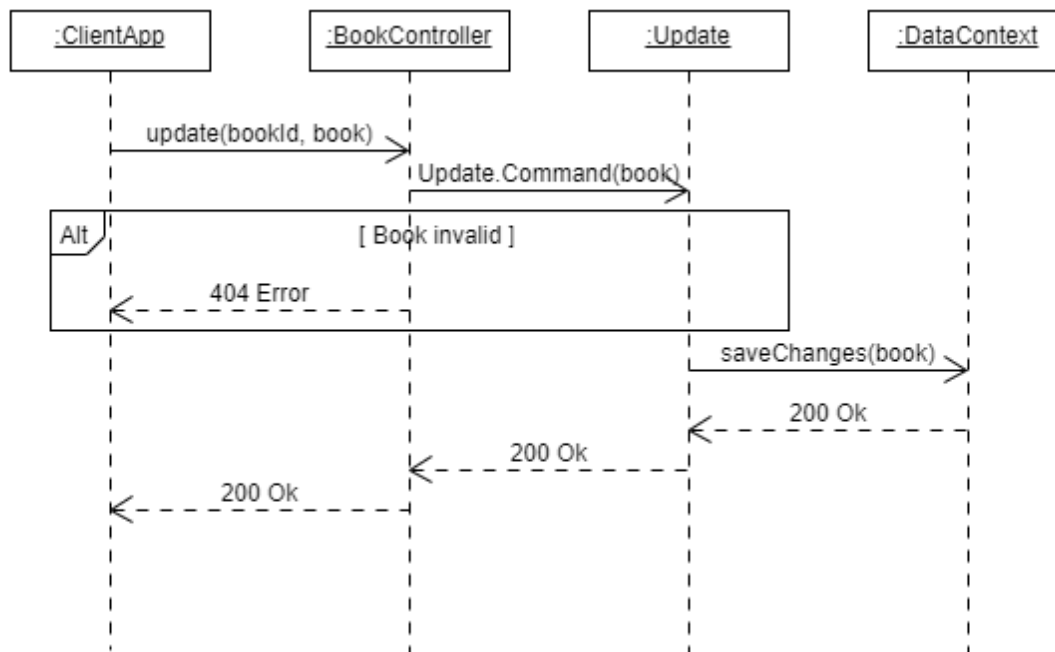


Figure 19 Update Book Sequence Diagram

10.2.3 Delete Book Sequence Diagram

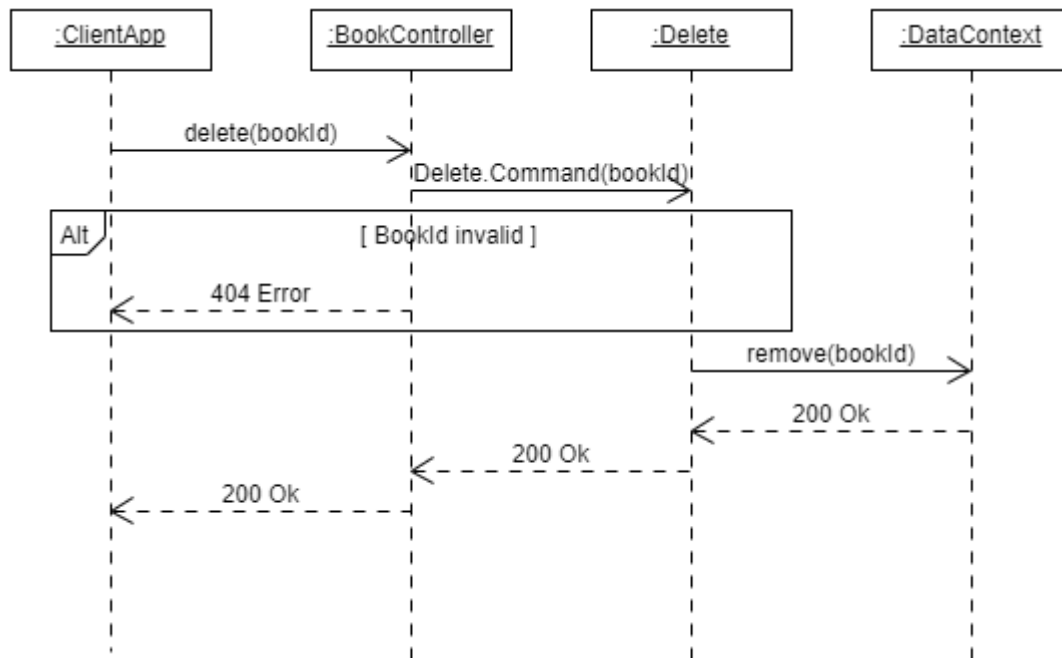


Figure 20 Delete book sequence diagram

10.2.4 Details Book Sequence Diagram

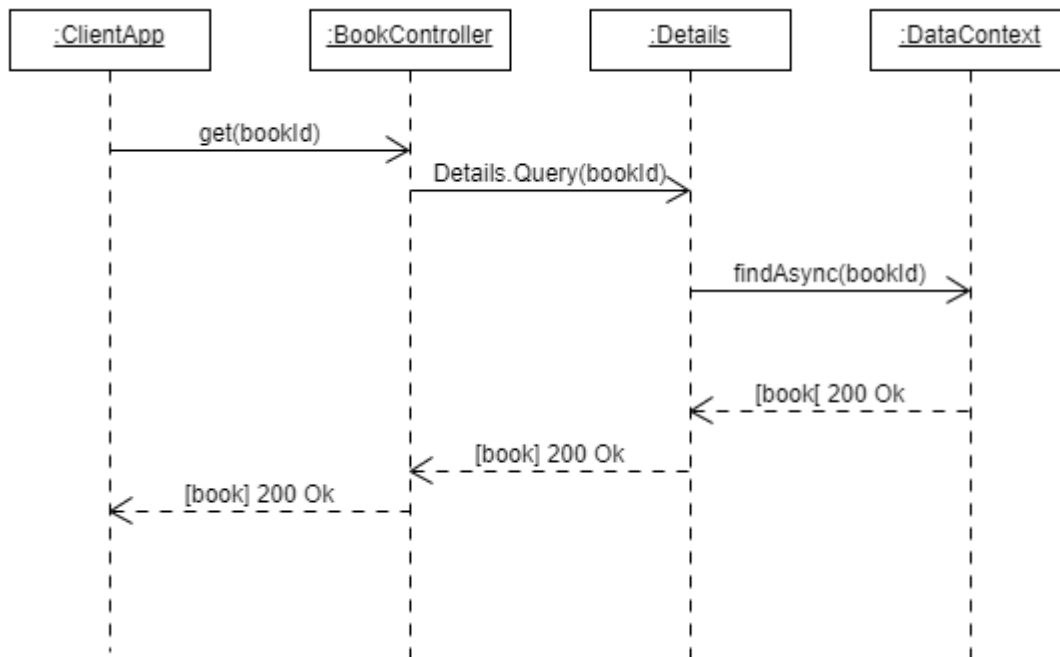


Figure 21 Details book sequence diagram

10.2.5 List Book Sequence Diagram

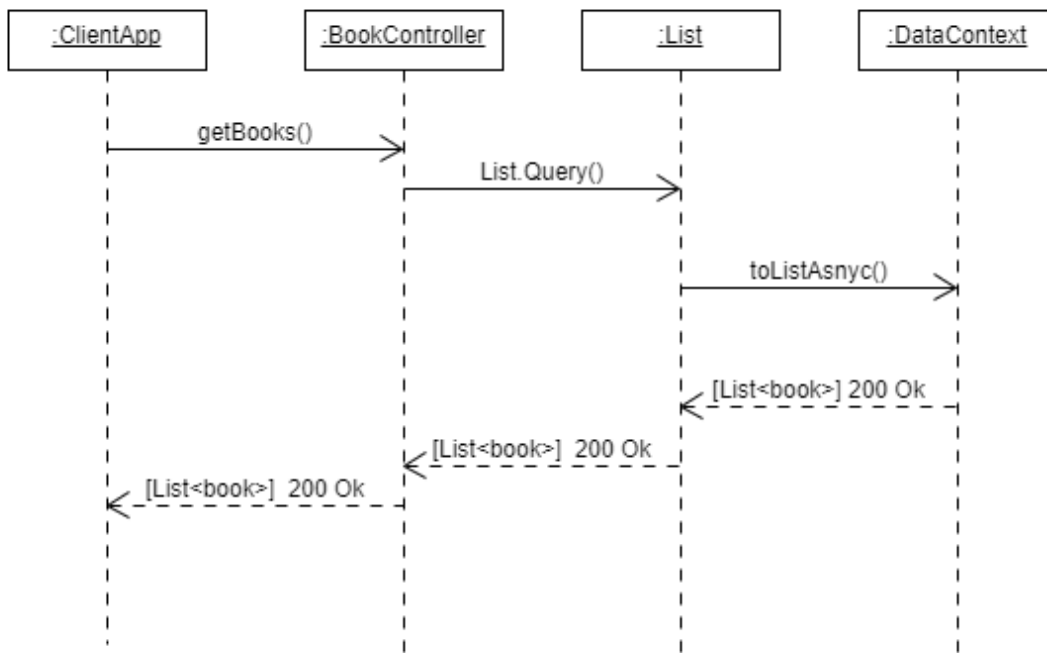


Figure 22 List Book Sequence Diagram

10.2.6 List BooksOwned Sequence Diagram

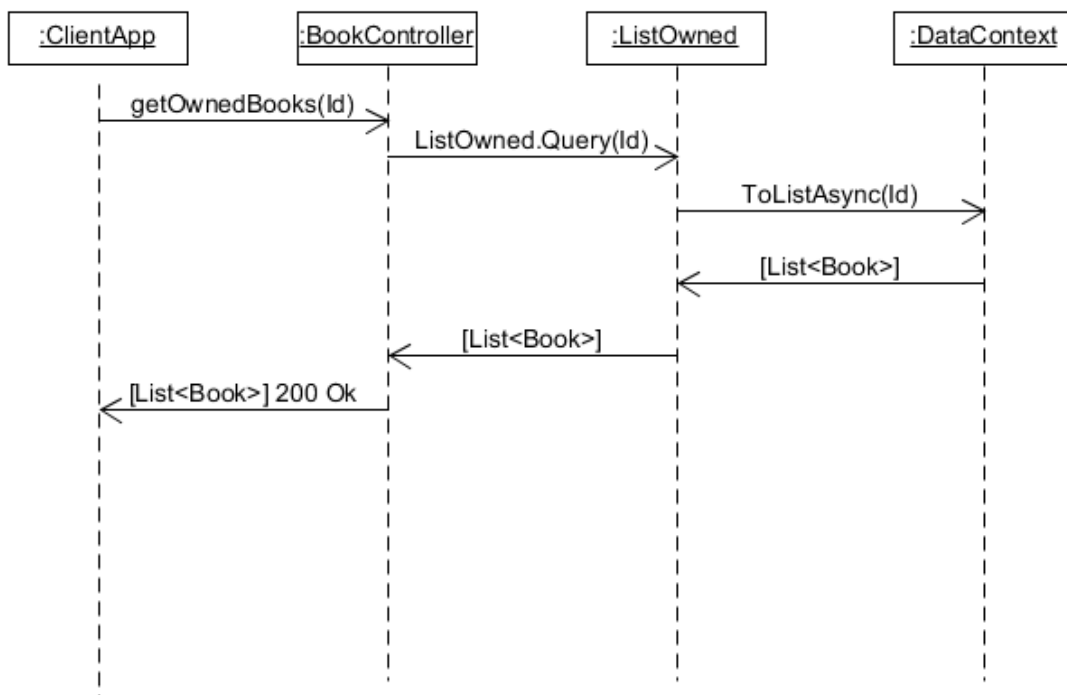


Figure 23 List BookOwned Sequence Diagram

10.3 SWAP SEQUENCE DIAGRAMS

10.3.1.1 Create Swap Sequence Diagram

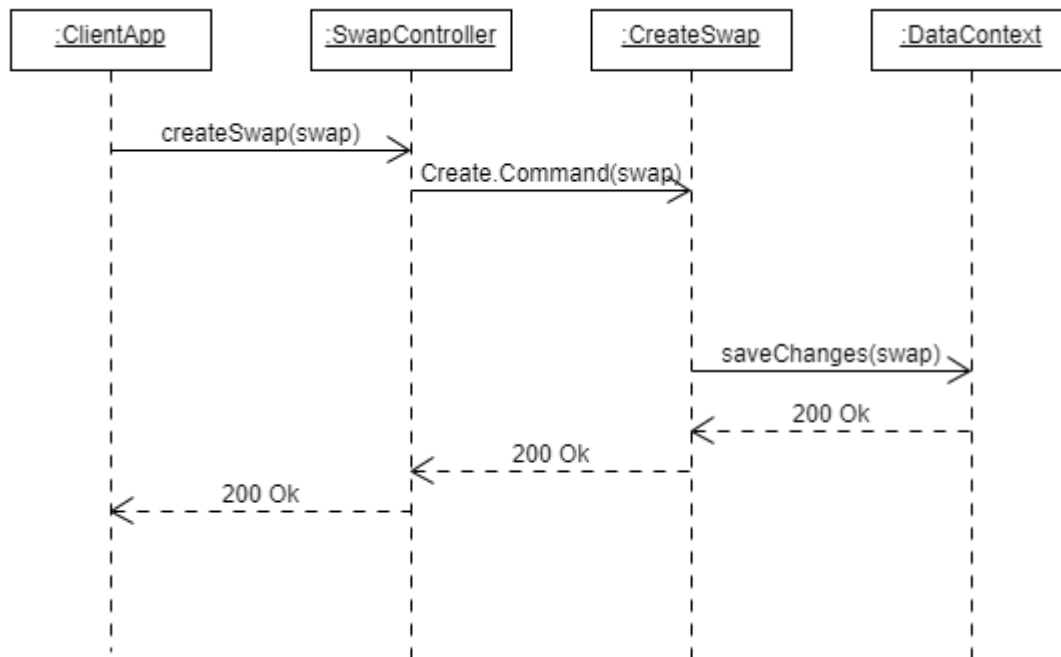


Figure 24 Create Swap Sequence Diagram

10.3.2 Update Swap Sequence Diagram

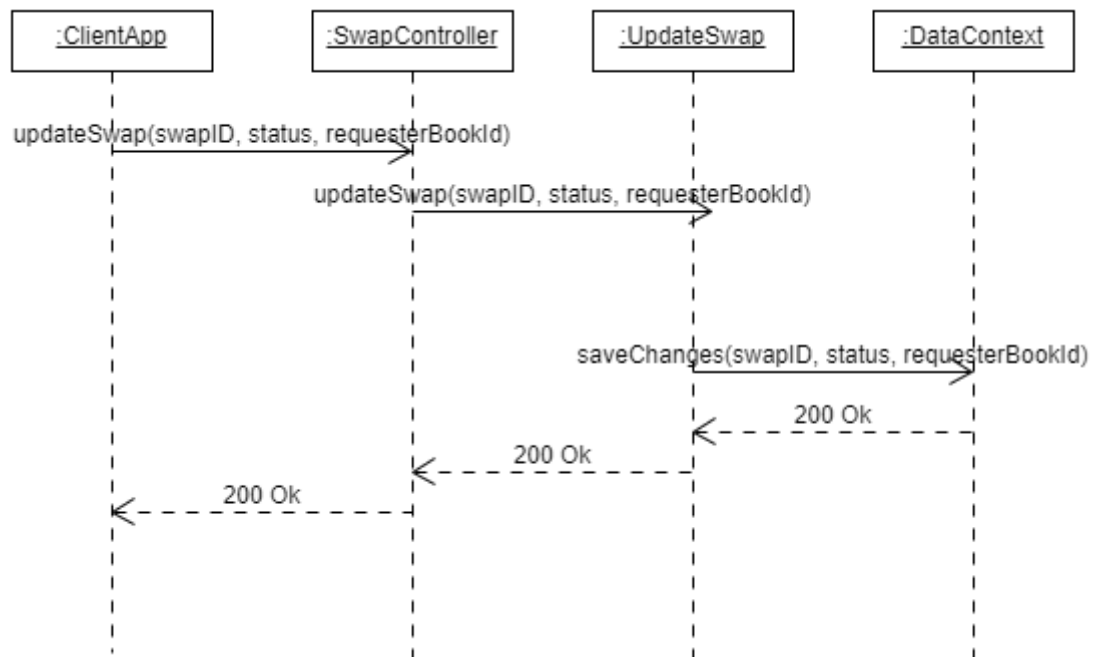


Figure 25 Update Swap Sequence Diagram

10.3.3 Delete Swap Sequence Diagram

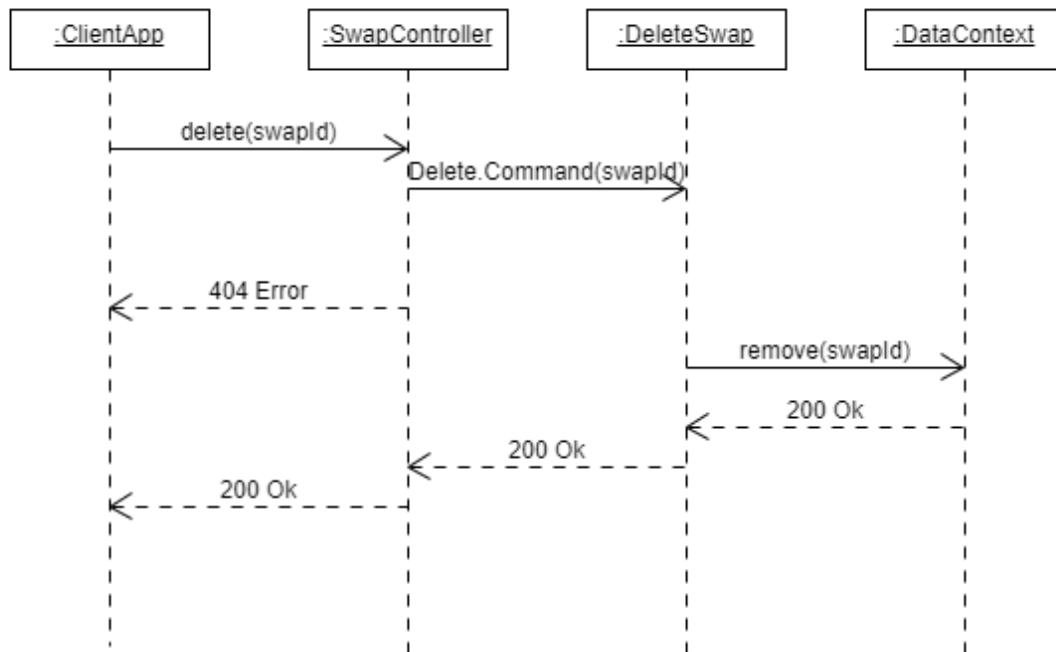


Figure 26 Delete Swap Sequence Diagram

10.3.4 Details Swap Sequence Diagram

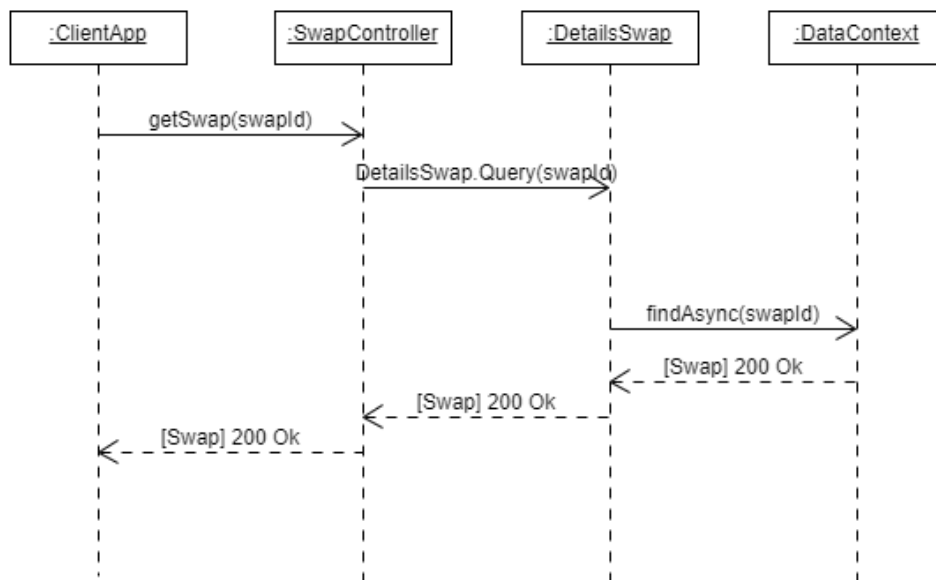


Figure 27 Details Swap sequence Diagram

10.3.5 List Swap Sequence Diagram

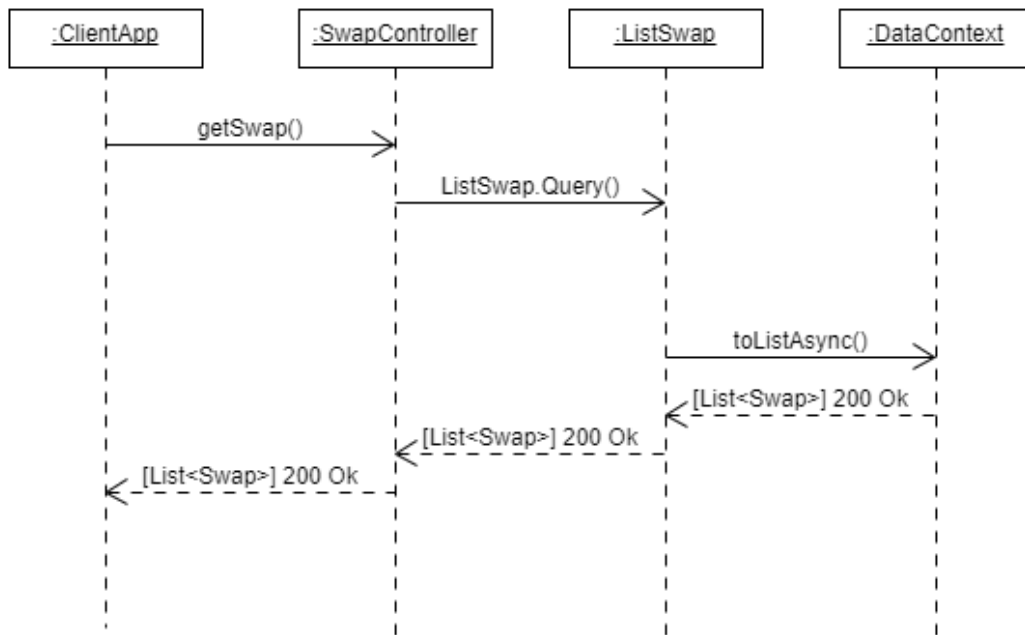


Figure 28 List Swap sequence Diagram

10.3.6 List SwapsIRequested Sequence Diagram

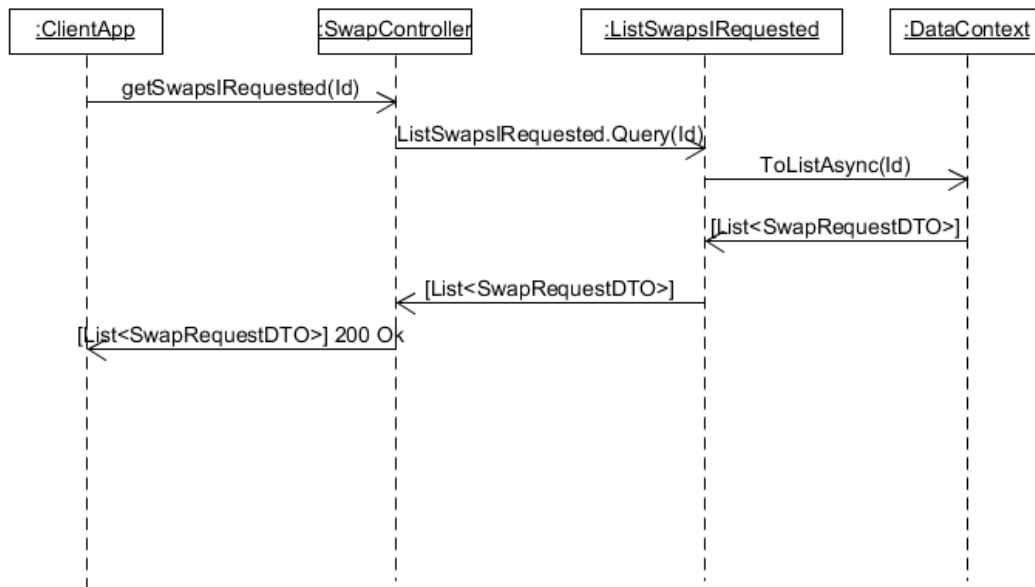


Figure 29 List SwapsIRequested Sequence Diagram

10.3.7 List SwapsRequestedFromMe Sequence Diagram

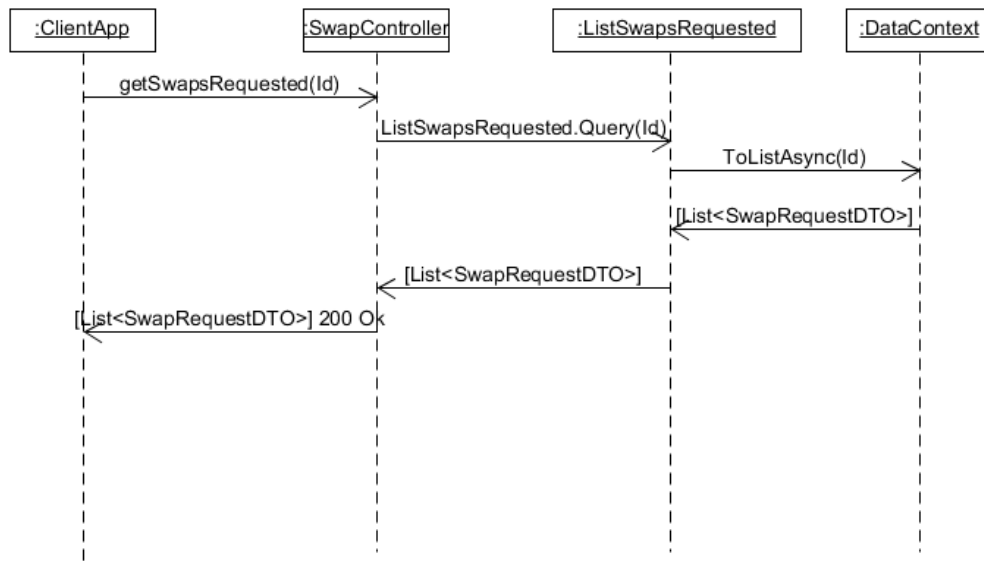


Figure 30 List SwapsRequestedFromMe Sequence Diagram

10.4 PROFILE SEQUENCE DIAGRAMS

10.4.1 Get Profile Sequence Diagram

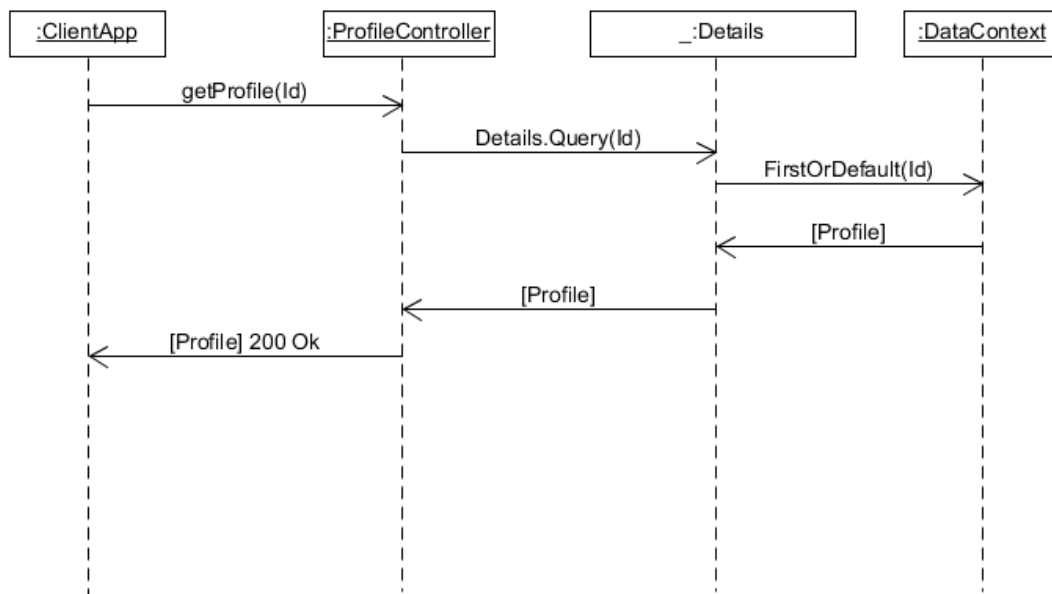


Figure 31 Get Profile Sequence Diagram

10.4.2 Update Profile Sequence Diagram

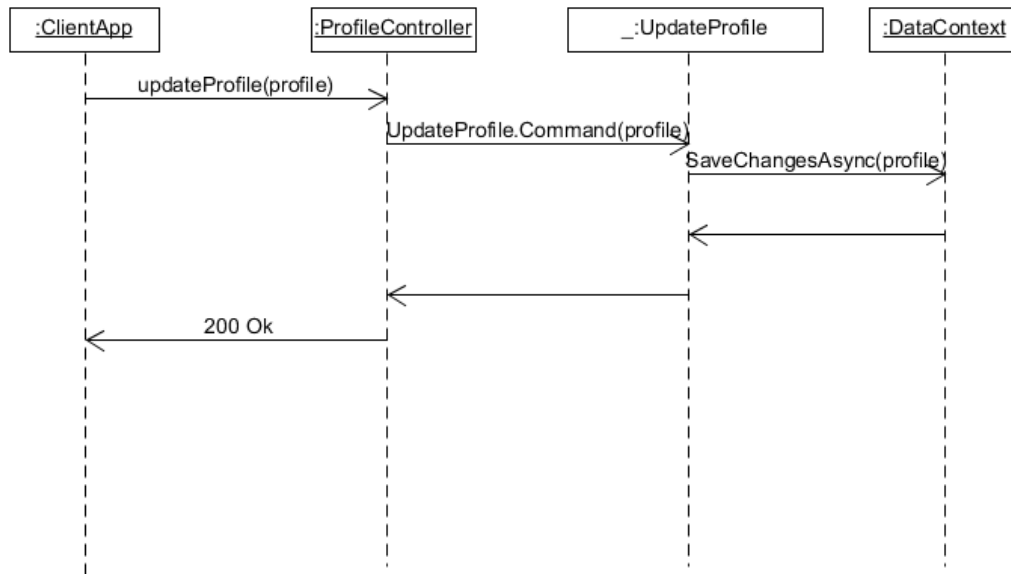


Figure 32 Update Profile Sequence Diagram

10.5 PHOTOS SEQUENCE DIAGRAM

10.5.1 Add Photo Sequence Diagram

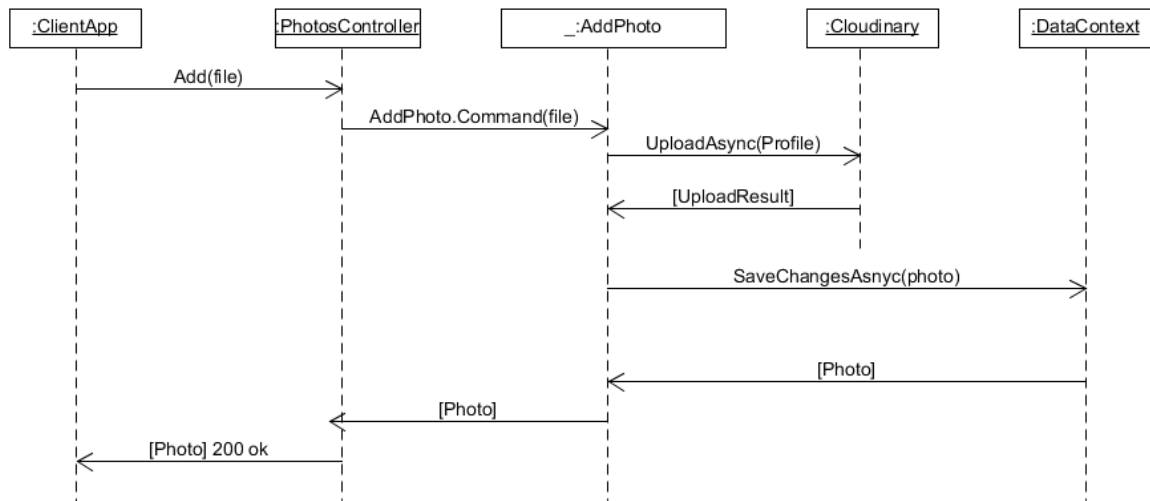


Figure 33 Add Photo Sequence Diagram

10.5.2 Delete Photo Sequence Diagram

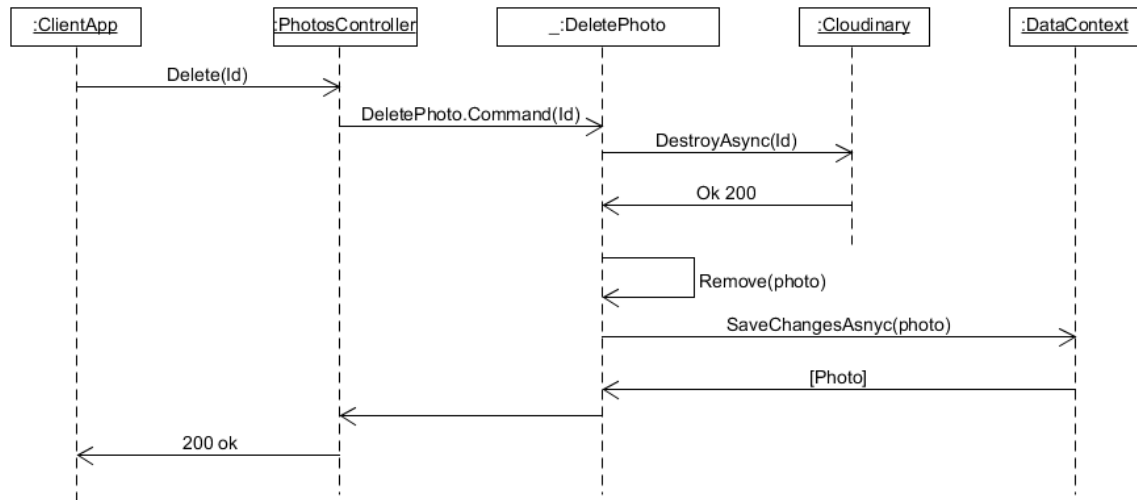


Figure 34 Delete Photo Sequence Diagram

10.6 SIGNALR SEQUENCE DIAGRAMS

10.6.1 Add Message Sequence Diagram

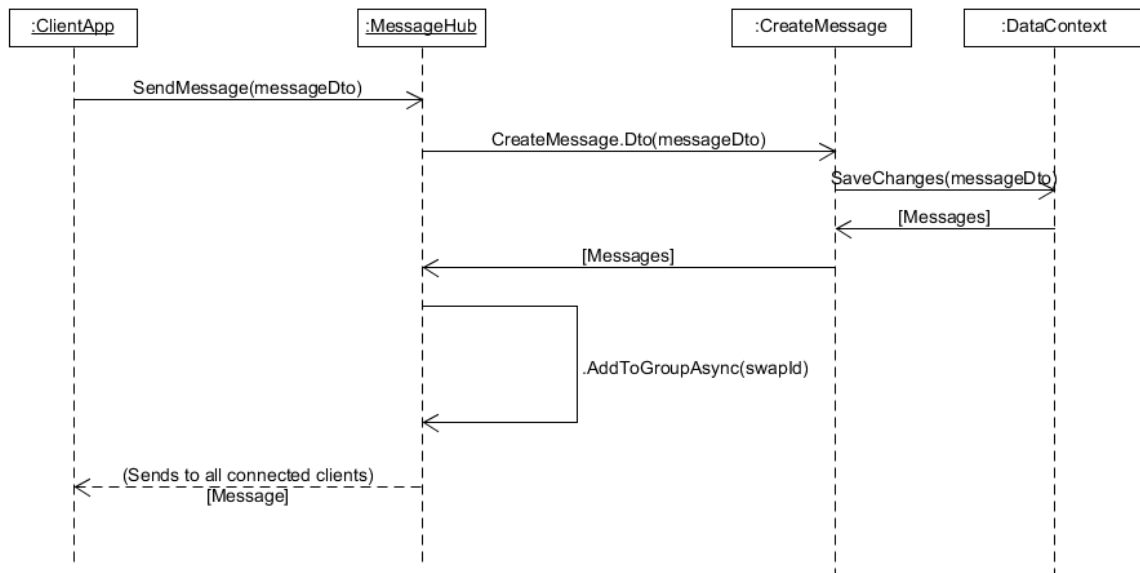


Figure 35 Add Message Sequence Diagram

10.6.2 List Message Sequence Diagram

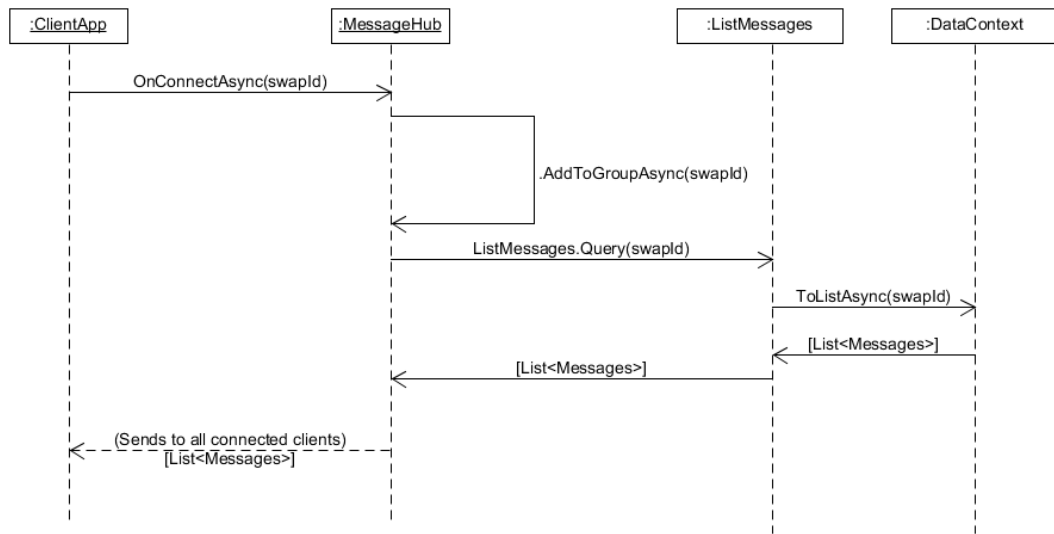


Figure 36 List Message Sequence Diagram

11 REFERENCES

[1]. Martin, R. (2017). Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series) (1st ed.). Pearson.

[2]. Narumoto, M. N. (2021, June 14). *What is the CQRS pattern? - Azure Architecture Center*. Microsoft Docs. <https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>

[3]. Bogard, J., 2021. *GitHub - jbogard/MediatR: Simple, unambitious mediator implementation in .NET*. [online] GitHub. Available at: <<https://github.com/jbogard/MediatR>> [Accessed 28 October 2021].

[4]. Team –. (2020, July 5). React. <https://reactjs.org/community/team.html>

[5]. *ISBNdb API Documentation v2 | ISBNdb*. (2021, August 18). Isbndb.Com. Retrieved October 31, 2021, from <https://isbndb.com/apidocs/v2>

[6]. Docs.microsoft.com. 2022. Introduction to ASP.NET Identity - ASP.NET 4.x. [online] Available at: <<https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity>> [Accessed 2 February 2022].

12 PLAGIARISM DECLARATION



*I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name (Printed) : Ivan Yaremko

Student Number : C00239239

Signature:

X

Ivan Yaremko