Comics Store

Project documentation

Student(s):

- Sebastian-Antonio Toma

Group: 30431-2

MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

Contents

I Project specification 3

1.1 Domain Model Diagram 3

II Use-Case model 4

2.1 Users and stakeholders 4

2.2 Use-Case identification 4

2.3 UML Use-Case diagram 7

III Architectural design 7

3.1 Conceptual architecture 7

3.2 Package diagram 8

3.3 Class diagram 9

3.4 Database (E-R/Data model) diagram 10

3.5 Sequence diagram 11

3.6 Activity diagram 11

IV Supplementary specifications **Error! Bookmark not defined.**

4.1 Non-functional requirements 13

4.2 Design constraints 14

V Testing 14

5.1 Testing methods/frameworks 15

5.2 Future improvements **Error! Bookmark not defined.**

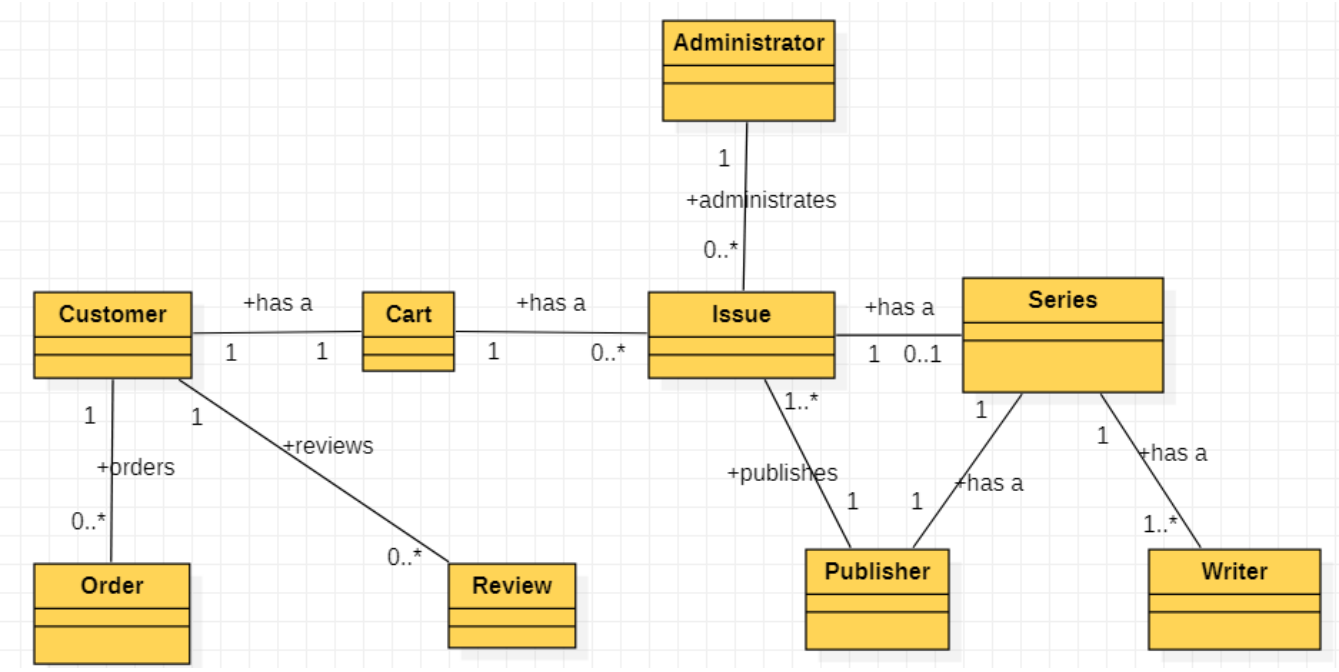
VI Bibliography **Error! Bookmark not defined.**

I Project specification

A physical store is limited to selling to local customers only. Many stores grow their businesses by expanding their market to eCommerce. In this way, their business can reach people all over the country or around the world, increasing the number of customers and, of course, the revenue. To add to this, a web store makes shopping easier for customers, by allowing them to buy and make decisions any time of the day and night, not having the constraints imposed by a physical store schedule. That kind of accessibility is a huge competitive advantage, especially in the context of a pandemic, where shopping online is safer and preferred by people compared to shopping physically. Thus, this application will be a web application.

The users of the application can be admins, publishers or customers, depending on their role, the main specifications of the project would be: for the customer, they can view comics and buy them, they can leave reviews for the bought comics; for the admins, they can add or remove comics from the shop; for the publishers, they can only add issues.

1.1 Domain Model Diagram



II Use-Case model

In this section the different types of users will be presented together with the use cases they introduce.

2.1 Users and stakeholders

Users of this application can be of three types: the customer, the publisher, and the administrator.

The administrator can add issues to shop or can definitely remove them from the shop.

The publisher can only publish, i.e. can only add issues to the shop.

The customer/the client can view the available issues from the shop, can visualize the given review left by other clients, can add them to their cart and buy them, and after they buy, they can also leave reviews to what they bought.

The stakeholders of this application can be the developers and the manager of that application, the publishers that add their products to this application, which is an online shop, also the writers of the sold issues. The competition could be other online comicbook shops or other face to face comicbook shops, that can be negatively impacted by the lack of customers that prefer the commodity of the online shop.

2.2 Use-Case identification

Use case name: Issue adding.

Level: User-Goal.

Main actor: Publisher.

Main success scenario: The publisher logs in the application with his account, then views a special UI where he can press a button to add a new issue, by introducing the required details. After introducing them and finalizing it, a request will be forwarded to the administrator to add the respective issue.

Extension: The publisher might not pass the validation for the fields of the issue to be added, in this case, an error will be displayed.

Use case name: Issue adding accepting.

Level: User-Goal.

Main actor: Administrator.

Main success scenario: The administrator logs in the application with his account, then views a special UI where he can see all the request made by a publisher to add a new issue. After accepting them and finalizing the action, he can see the added issue in the shop together with other ones.

Extension: The administrator can accept multiple request made by different publishers.

Use case name: Issue deleting.

Level: User-Goal.

Main actor: Administrator.

Main success scenario: The administrator logs in the application with his account, then views a special UI where he can select a issue and press a button to delete it. After deleting it and finalizing it, he can see the deleted issue can no longer be seen in the shop together with other ones.

Extension: The administrator can delete multiple issues in a row.

Use case name: User LogIn.

Level: Subfunction.

Main actor: All.

Main success scenario: The user presses the LogIn button on the main page, then introduces the username and password, then presses the button to log in.

Extension: The user will receive an error if the validation for the username, or the password, or the username – password combination fails.

Use case name: Add to cart.

Level: User-Goal.

Main actor: Customer.

Main success scenario: The user logs in with his account by performing the log in use case. After a successful log in, he can view the issues in the shop, he can view more details about them and/or press the button to add them to the cart.

Extension: The user can press multiple times the button to add to cart. By performing this, the number of copies of that issue will be increased.

Use case name: Order.

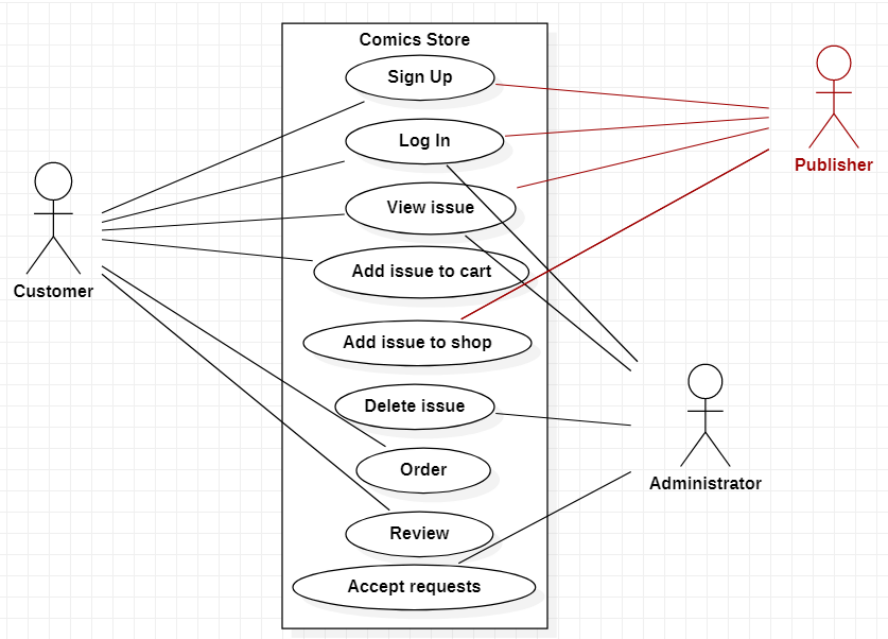
Level: Summary.

Main actor: Customer.

Main success scenario: After logging in and adding a few issues to the cart, the user can navigate to the cart by pressing a button. In the cart window the user can see everything that was added to the cart, and the total price for everything. He can also add or remove some issues that are present in the cart. After pressing order, the details of the shipping will be introduced.

Extension: The customer might not pass the validation for the shipping details introduced, and an error message will be displayed.

2.3 UML Use-Case diagram



III Architectural design

Here the architecture of the project will be presented together with the database description.

3.1 Conceptual architecture

The application is intended to be a web application implemented with Java + Java/Spring, using the Client-Server model.

The chosen architecture is the Layered Architecture. A Layered Architecture represents the organization of the project structure into four main categories/layers:

• Presentation layer

• Business layer

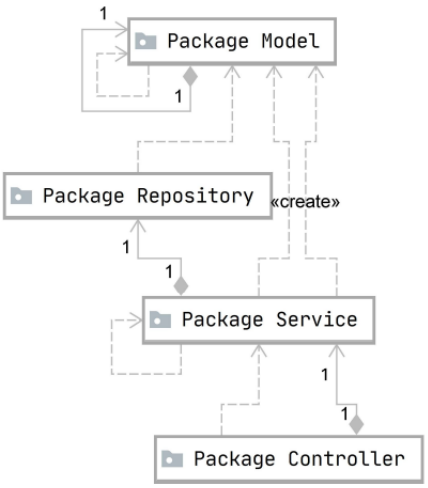
• Persistence layer

• Data layer.

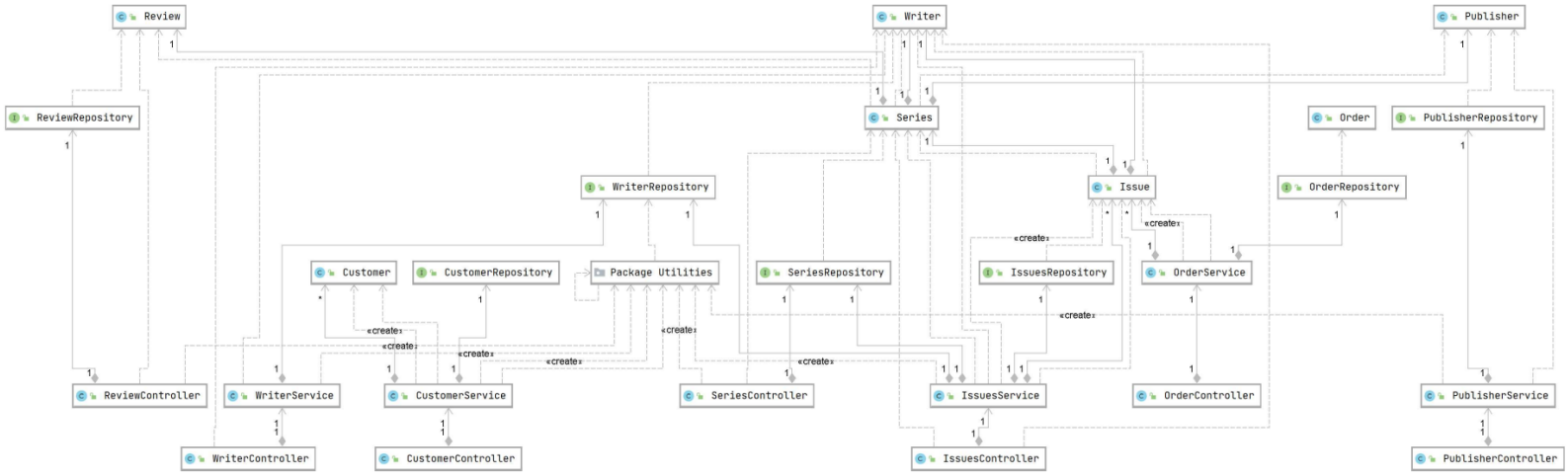
Each of the layers contains objects related to the particular concern it represents and has a special purpose calling functions of the layers below it. The Layered Architecture is suitable for this application, because it is a simple application with not too many features and a Layered Architecture style is the best choice if simplicity is important, especially if the application is not supposed to grow in size and complexity too much.

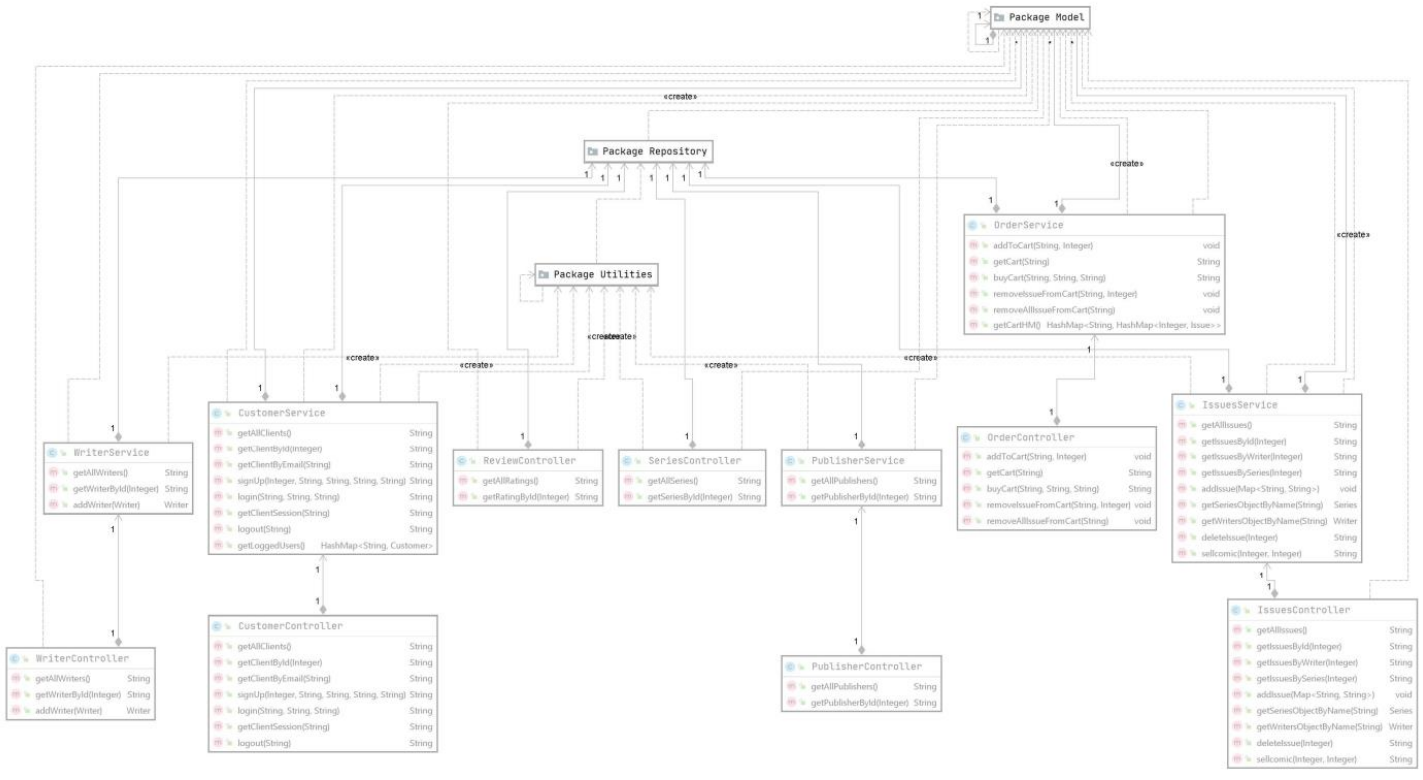
The application will use a relational database as storage (MySQL). The Data Access Layer is going to be implemented using JPA Repository. The database will have a table for each type of user (client, admin, publisher) which all of them will be the specialization of the 'users' table. The orders and the reviews left by the clients will also be stored in tables, together with the products (a comic issue) and the other tables needed for organizing them (writers, publishers, series).

3.2 Package diagram

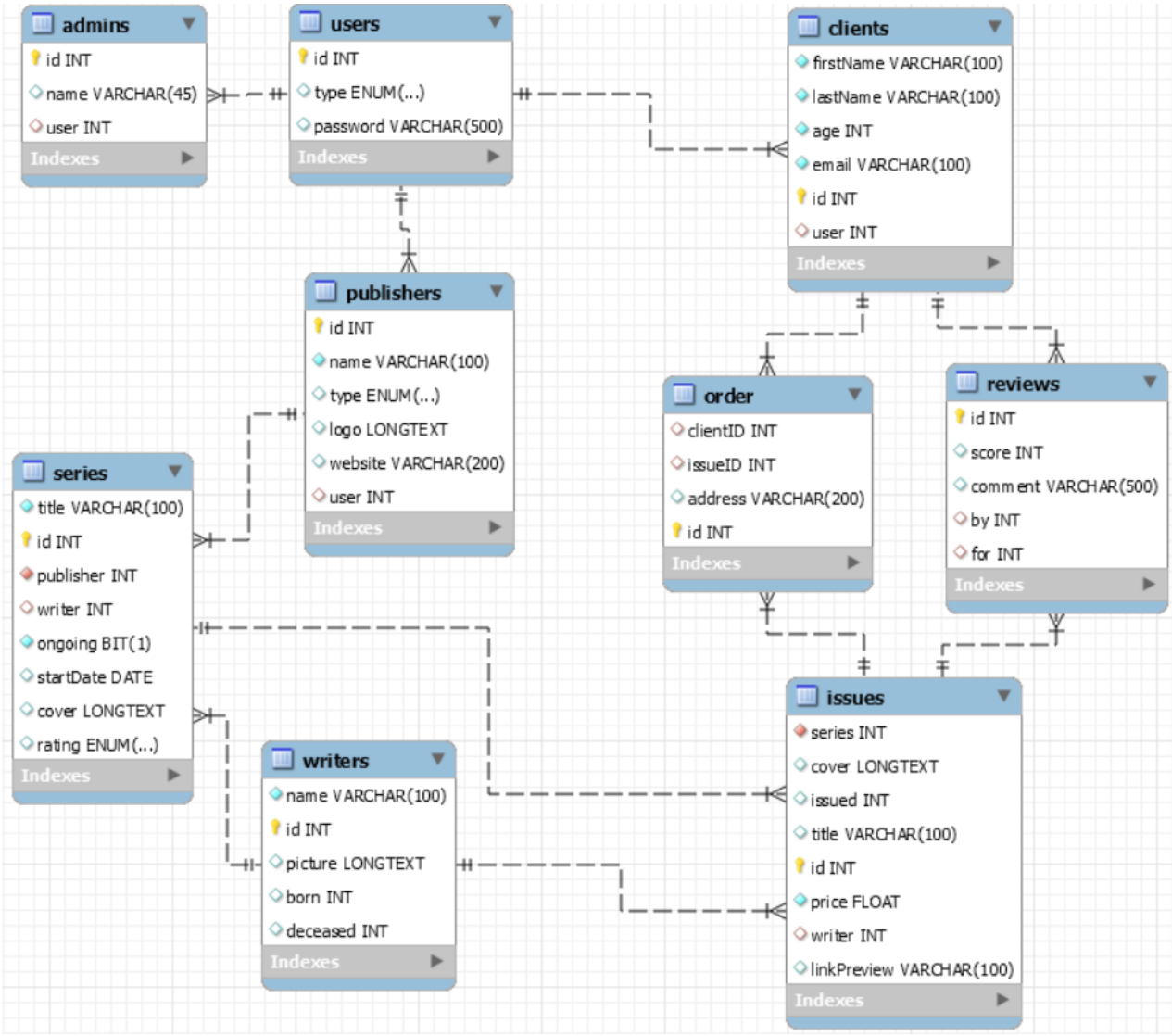


3.3 Class diagram

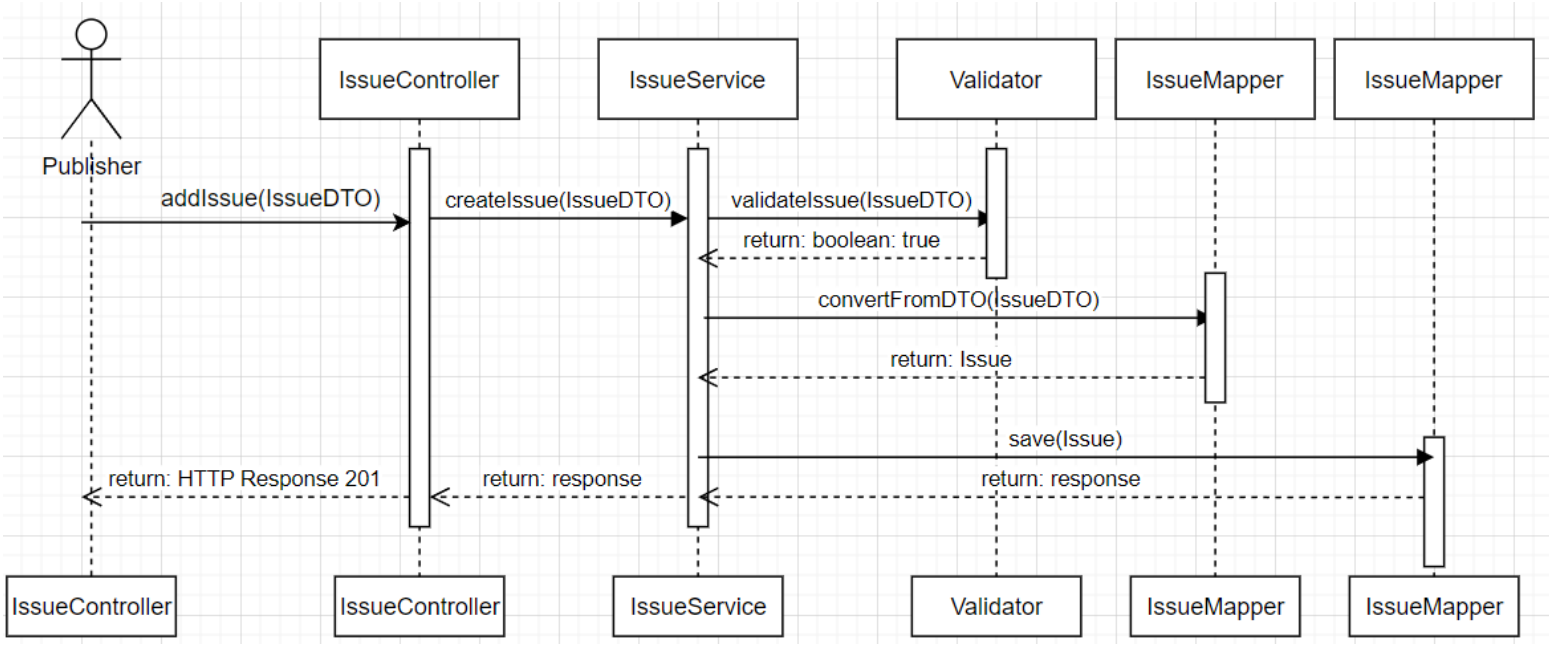




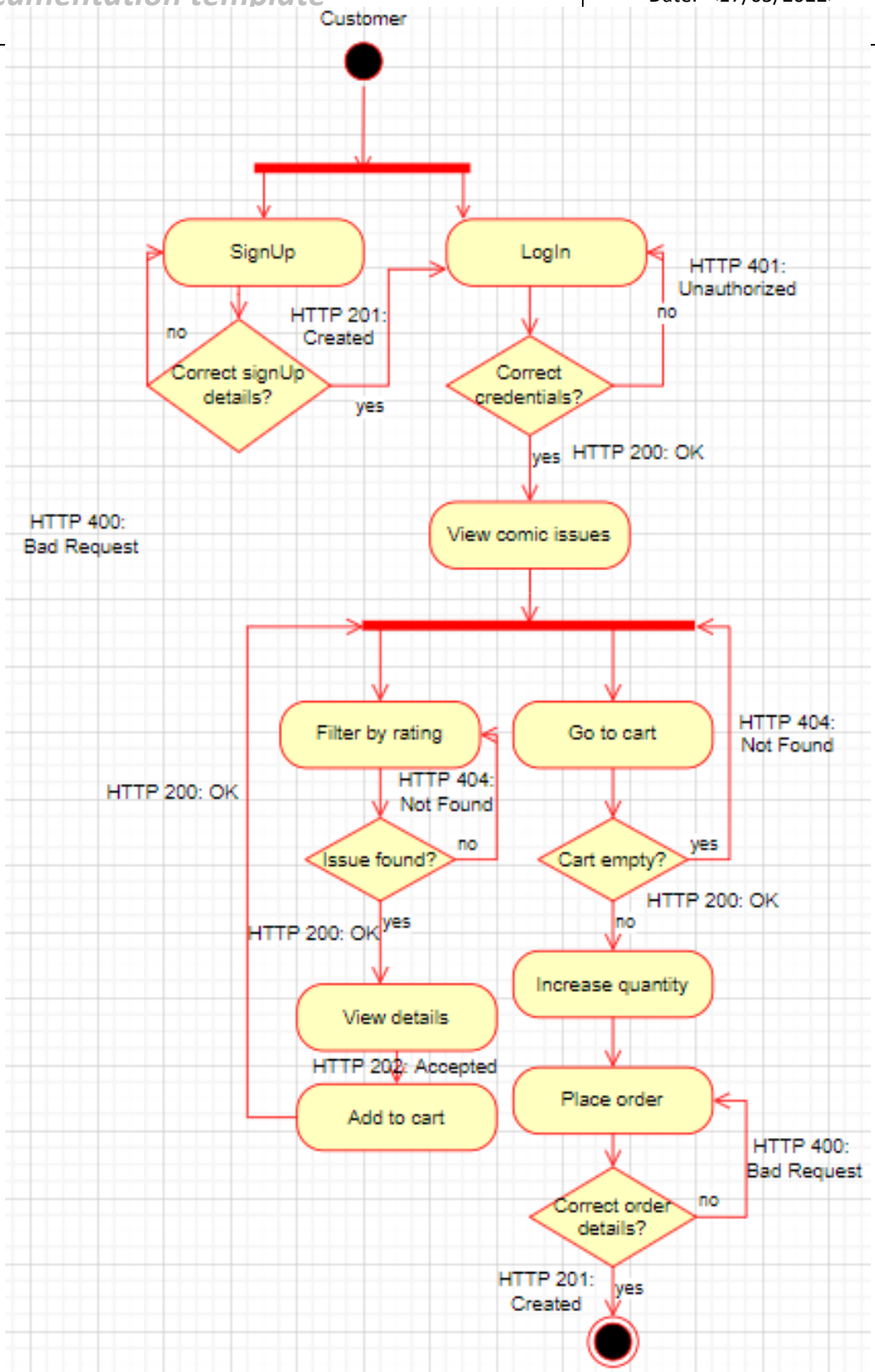
3.4 Database (E-R/Data model) diagram



3.5 Sequence diagram



3.6 Activity diagram


Customer
SignUp
LogIn
HTTP 401: Unauthorized
no
HTTP 201: Created
no
Correct signUp details?
yes
Correct credentials?
yes
HTTP 200: OK
HTTP 400: Bad Request
View comic issues
Filter by rating
Go to cart
HTTP 404: Not Found
HTTP 200: OK
HTTP 404: Not Found
Issue found?
no
Cart empty?
yes
HTTP 200: OK
no
HTTP 200: OK
yes
Increase quantity
View details
HTTP 202: Accepted
Add to cart
Place order
HTTP 400: Bad Request
Correct order details?
no
HTTP 201: Created
yes

IV Supplementary specifications

In this section some of the non-functional requirements and design constraints of the application will be presented.

4.1 Non-functional requirements

Some of the non-functional requirements of this application described in this section are: security, accessibility across devices, availability and scalability.

The security of the application, especially the security of the user accounts (customers, admins, publishers) will be ensured by storing the hash of the password instead of the plain password in the column of the table from the database storing all three types of users. Also, when registering, some validation will be applied to the user's chosen password before allowing it to continue with the account creation. This is needed for user safety and security.

As this application is a web application it should be accessible from all types of devices having internet connection and a browser installed. However, this web application is made especially for desktop devices, so it might not be properly scaled for mobile devices such as smartphones. This is needed for a good user experience when using this application.

The availability of the application will be ensured using version management.

As a Layered Architecture, it can cause some issues regarding the scalability (the property of an application to be further extended), but this can be fixed by diving the layers in multiple independent units (for example the Business Layer).

4.2 Design constraints

The application is written in Java 15, using the Spring Framework, to make it a web application and using Maven as a build automation tool and package management tool and IntelliJ as IDE.

The application will use a relational database as storage (MySQL), stored locally. MySQL Workbench is used for the management of the database. For the connection with the database and the Spring application, JPA Repository will be used to implement the Data Access Layer.

The database will store all three types of users and their accounts, together with their password, which only the hash of the it will be presented in the database, and not the actual password, ensuring a better security for the users.

For testing the application, JUnit will be used together with Mockito.

V Documentation of APIs

POST /localhost:8082/register

Request:

- body (JSON):

```
{
    “username”: “aaa”,
    “password”: “aaaaaaaa”,
    “checkpswd”: “aaaaaaaa”,
    “name”: “aaa”
    “email”: “aaa@gmail.com”
```

“age”: 19

}

Responses:

● 201 CREATED – Customer was successfully registered

● 400 BAD REQUEST – Invalid input data (passwords do not match, empty field, age too low)

GET /localhost:8082/current

Request:

● @AuthenticationPrincipal User user

Responses:

● 200 OK – Current user was successfully returned

body (JSON):

{

“id”: 2,

“name”: “aaa”,

“type”: “Customer”,

“picture”: null

}

● 401 UNAUTHORIZED – No current user is logged in

POST /localhost:8082/series/add

Request:

- body (JSON):

```
{
    "title": "ms marvel",
    "writer": "Stan Lee",
    "publisher": 1
}
```

Responses:

- 200 OK – Series was successfully requested to be added
- 400 BAD REQUEST – Invalid input data
- 401 UNAUTHORIZED – No current publisher is logged in

PUT /localhost:8082/series/rate/2

Request:

- body (JSON):

```
{
    "category": "TEEN"
}
```

Responses:

- 200 OK – Series was successfully rated
- 400 BAD REQUEST – Invalid input data
- 401 UNAUTHORIZED – No current admin is logged in

GET /localhost:8082/order/filter/DELIVERED

Request:

Responses:

- 200 OK – Orders were successfully returned

body (JSON):

[{
“id”: 2,
“cId”: 5,
“price”: 11,
“data”: “16-05-2022”,
“status”: “DELIVERED”
},
{
“id”: 3,
“cId”: 5,
“price”: 17,

“data”: “13-05-2022”,

“status”: “DELIVERED”

},

{

“id”: 4,

“cId”: 6,

“price”: 22,

“data”: “15-05-2022”,

“status”: “DELIVERED”

}]

- 401 UNAUTHORIZED – No current user is logged in
- 404 NOT FOUND – Wrong filter (status)
- 405 METHOD NOT ALLOWED – Publisher is logged in

PUT /localhost:8082/order/change/2/DECLINED

Request:

Responses:

- 200 OK – Order status was successfully changed
- 401 UNAUTHORIZED – No current admin is logged in
- 404 NOT FOUND – Wrong filter (status)

DELETE /localhost:8082/issue/remove

Request:

- body (JSON):

```
{
    “title”: “ms marvel #1”,
    “price”: 7,
    “id”: 22,
    “link”: null
    “series”: “ms marvel”,
    “publisher”: “images/publishers/1.png”
}
```

Responses:

- 200 OK – Issue was successfully deleted
- 401 UNAUTHORIZED – No current publisher is logged in