

Final Report — Books of Bliss

E-commerce platform for student literature

Benjamin Sannholm, Endrit Karpuzi, Gustav Pihlquist,
Oliver Karmetun, Sophia Pham

Project Group 5

1 Introduction

Books of Bliss is a platform where users can buy and/or sell books and course literature. When a user puts up a book for sale, it needs to specify a price and a condition of the book. Other users can then proceed to buy this listing, provided that they have enough credits in their account balance.

Link to our GitHub repository: <https://github.com/DIT126-Group5/DIT126-Group5>

2 Use cases

No.	Use Case	Actor
#1	Register an account	New user
#2	Log in with existing account	Registered user
#3	Update user settings and password	Signed in user
#4	Deposit money to my account	Signed in user
#5	Search for a book	User
#6	Purchase a book	Signed in user
#7	Give a review of the purchase	Buyer
#8	See seller's reputation	User
#9	Sell a book	Signed in user
#10	Log out	Signed in user

3 User manual

3.1 Navigation

In order to navigate between the pages the user should primarily use the navigation bar at the top of the page, but there are also other ways of navigating from page to page. In particular, buying a book will lead the user through a sequence of pages, starting with the browse page which can be accessed through the navigation bar. When a listing is pressed the user moves to the listing page, followed by the checkout page and finally the purchase confirmation page.

Certain pages are also only available when a user is logged in, in particular the pages under “Mina sidor” in the navigation bar. It is important to note that logging in is a requirement for

reaching the checkout page, unlike other online stores where logging in is not necessary or can be done at checkout.



Figure 1. *Navigation bar when signed out.*



Figure 2. *Navigation bar when signed in.*

3.2 International Standard Book Number (ISBN)

ISBN stands for International Standard Book Number and is usually a 13-digit number that identifies published books. Once assigned to a book, an ISBN can never be reused. There are five parts to an ISBN number — the current prefix is 978 or 979.



Figure 3. *An example of an ISBN.*

On most books, the ISBN can be found on the back cover, next to the barcode. If a book does not show the ISBN on the back cover, look on the page featuring the copyright and publisher information and the ISBN will be found there.

Unfortunately books without ISBN cannot be sold on our site because our system is built on the basis of ISBN. All books that have been published must have an ISBN. But there are some books that have not been published or are too old and therefore lack an ISBN.

3.3 Buy credits

Before buying a book, credits have to be deposited into your account through the “Buy credits” page accessible through the navigation bar at the top of the website. On this page the amount of credits you wish to deposit and valid payment details have to be entered. To perform the deposit, press the “Genomför insättning” button.

Saldo: SEK 0.00

Köp kredit

Belopp

255,00 kr

Betalningsmetod

55555555555555555555 ✓

Justin Bieber ✓

02 ✓ / 21 ✓ 245 ✓

☒ Jag godkänner Books of Bliss köpvillkor

Genomför insättning

Figure 4. *Buy credits form.*

3.4 Search for a book


Any user can access the search function, seen as a text field and a button at the top of all pages in the navigation bar. After entering a search term and clicking “Sök” all listings containing the term in either the title, ISBN, or author will be shown. In the search results or on the home page, a book listing can be pressed to show more information about the listing and the seller.

3.5 Buy a book

Once you have found a book you would like to buy, you may proceed to buy the book by pressing the “Gå vidare till köp” button on the listing information page. The “Gå vidare till köp” button will not be displayed if you are the author of the listing or if you are not logged in.

On the checkout page, please enter the address where you would like the book delivered to and press the “Bekräfta köp” button to confirm your purchase. You may not proceed with the purchase if you do not have enough credits in your account.

Du är på väg att köpa



Computer Systems (2015)
Randal E. Bryant, David R. O'Hallaron
ISBN: 9780134092669
Skick: Begagnad
Kategorier: Computers

Leveransadress

Säljare
Justin Bieber
Omdöme: 0.0/5
Stad: LA

Betalning

Saldo:	SEK 250.00
Pris:	SEK 150.00

[Bekräfta köp](#)

Figure 5. *Checkout page.*

After a successful purchase, you may send a review to tell us how you liked your purchase. Please enter a rating from one to five stars and optionally provide a comment then press the “Skicka” button to send the review.

Vad tyckte du om ditt köp?

Betyg: ★★★★★

[Skicka](#)

Figure 6. *Review purchase form.*

3.6 Sell a book

To publish a listing you need to be signed in to an account. In order to put up a book for sale, enter navigate to the publish listing page by pressing the “Sälj en bok” button in the top navigation bar. On this page, please enter the ISBN of the book in question, which will gather information about the title, publication year, author(s) etc. Thereafter you must specify the book's condition and also give the listing a price. There is also an optional field to enter a description or comment about the listing. When all the required fields are filled, you may press the button “Publicera”, which will publish a new listing and redirect you directly to it.

Vad vill du sälja?

Skriv in bokens ISBN-nummer ☒

Boktitel

Författare

Skriv in ditt pris *

Annonsbeskrivning (max 250 karaktärer)

Valj bokens skick

Figure 7. Publish listing page.

4 Design

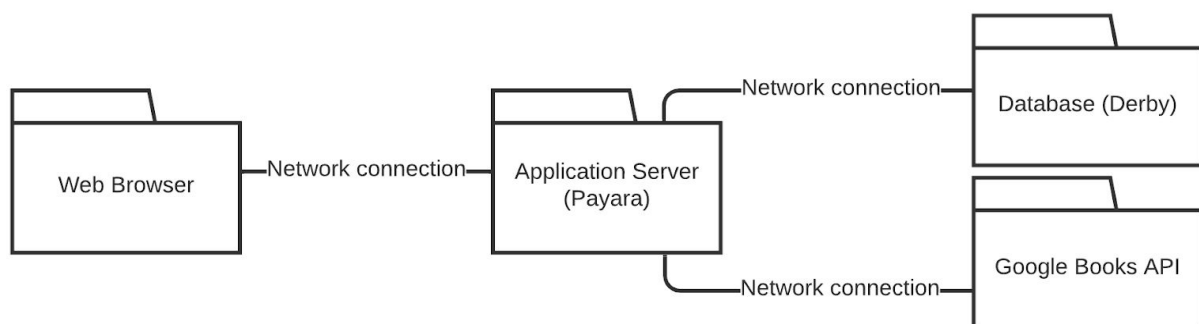


Figure 8. High-level overview of architectural components.

The application is run using a number of components working together. Payara is used as the application server, which implements the Jakarta EE specification and runs the web server. Derby is used as the application's database. Information about books is retrieved from the Google Books API.

When running tests for the application, Arquillian is used for any non-trivial tests. Arquillian starts the application server with Payara, which in turn establishes a connection to a test database. This database is dropped and recreated each time the test suite is executed.

The code is divided into multiple packages each serving a distinct role according to the MVC pattern. The view package consists of a number of backing beans, one for each of the pages on the website. The pages themselves are written in separate XHTML files that interact with the backing beans in order to display information. The controller package contains controller classes that are used when the user does something in the view that requires any updating of information in the model. The controller classes are kept “thin” by delegating work to the model. Most business logic resides either in an entity class or a service.

Most of the responsibility of creating class instances is left to the Contexts and Dependency Injection (CDI) framework of Jakarta EE. The CDI framework could additionally keep classes further decoupled by the use of interfaces. This application does, however, not use any interfaces so this is not the case here.

The following libraries are used in the project:

- Java EE API
 - Java Server Faces (JSF)
 - Contexts and Dependency Injection (CDI)
 - Enterprise Java Beans (EJB)
 - Java Persistence API (JPA) — Object-relational mapping and database connection
 - Bean Validation — For validating entity properties and user input
- Lombok — Generating constructors, getters/setters, hashCode and equals for classes
- JavaMoney Moneta — Used to reliably handle monetary amounts
- PrimeFaces — Component library for JSF
- Bootstrap — CSS framework with utility classes for responsivity, layout and design
- ButterFaces — Used for adding Bootstrap support
- OmniFaces — Utilities for JSF
- Arquillian — For bootstrapping the test environment (e.g. allowing CDI use)
- JUnit — Unit testing framework
- MeanBean — Automatic testing of getters and setters
- EqualsVerifier — Automatic testing of equals and hashCode

5 Application flow

5.1 Use case #1: Register an account

To register an account the user navigates to register.xhtml. Once here they will need to fill in some details in a form, including username, name, address, contact information and a password. To confirm creating the account the “Skapa konto” button is pressed. This will call the method createAccount in RegisterController. This method checks if the information input through the form is valid using validation annotations in the RegisterBackingBean and

whether the username is already taken or not using the AccountDAO. It then creates a new Account object and tries to set the password using AccountService. AccountService checks if the password is valid according to the rules specified by the Account entity in Account.isValidPassword.

The AccountService is additionally responsible for making sure the user's password is securely hashed before saving it. This is performed using the Pbkdf2PasswordHash class, from the Jakarta EE Security API, which implements the PBKDF2 key derivation function.

Once the entered information is valid and the password has been set the account will be added to the database, through AccountDAO, and the user will be notified with a success message. Otherwise an error message will be displayed, notifying the user of what needs to be changed.

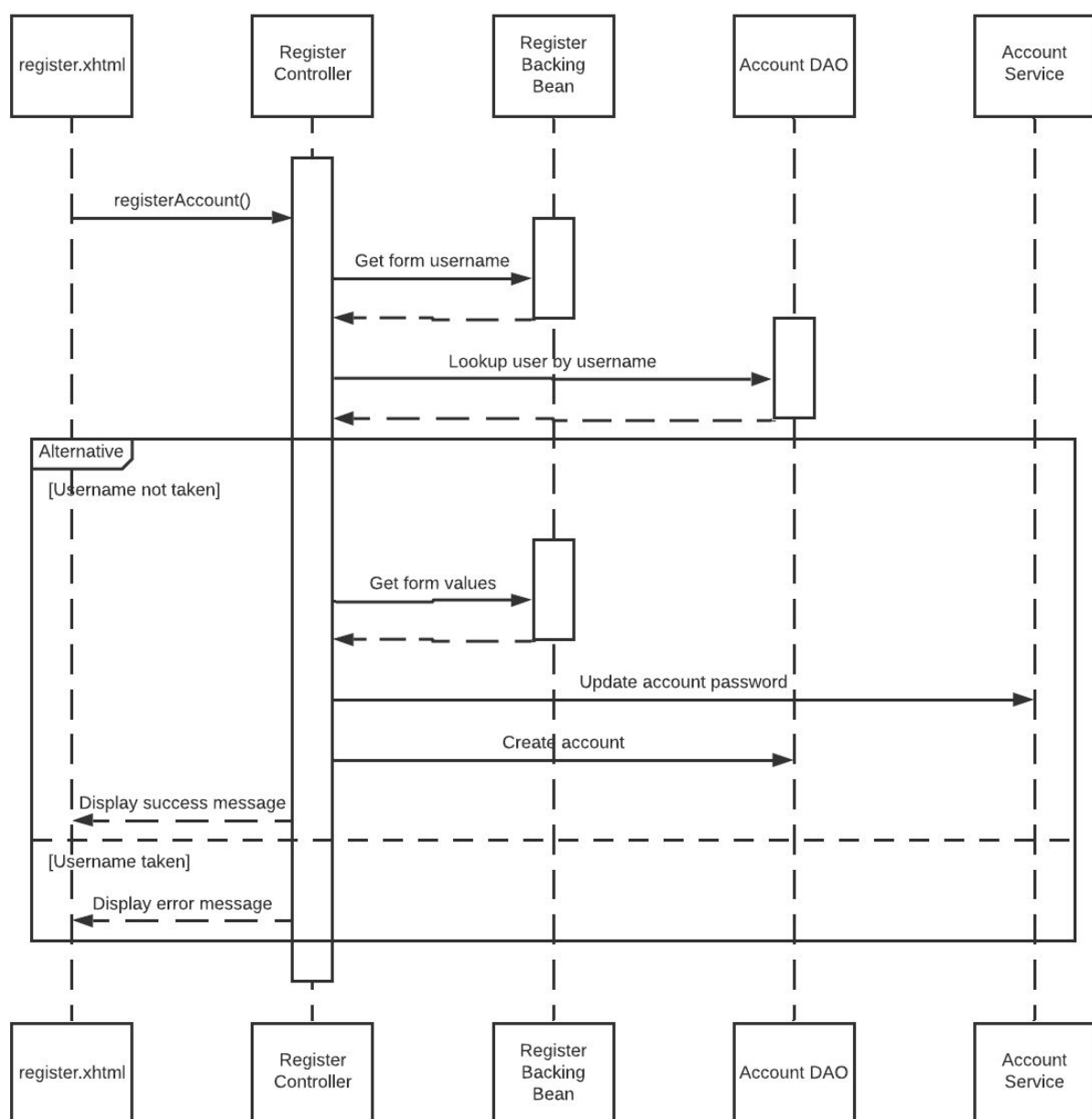


Figure 9. Sequence diagram of account registration process.

5.2 Use case #2: Log in with existing account

The user logs in through the login.xhtml page. Once the user has entered their credentials and pressed “Logga in”, the login method of LoginController is called. LoginController forwards the login request to SecurityContext from the Jakarta EE Security API.

The security API will internally call the validate method of our class BobIdentityStore. The identity store performs a lookup in the database, through AccountDAO, to find the user who is being authenticated. Finally the identity store validates the user’s password using AccountService.validatePassword. Just like when registering an account, AccountService will use the Pbkdf2PasswordHash class but this time to validate the given password against the store password hash.

As soon as the SecurityContext class has given a result back for the authentication request, the LoginController does not need to perform any further actions since the Jakarta EE Security API automatically remembers a successful login in the current user’s session. Depending on the result the controller finally either displays a success or an error message in the view.

5.3 Use case #5: Search for a book

The input area for searching is located in the navigation bar and is accessible from every view. It does not require the user to be logged in to be accessed. When entering a search term and pressing “Sök”, a JavaScript function named “relocateOnSearch” located in common.xhtml is called. The function redirects the user to the browse page with a URL parameter “q” and the search term as the value of the parameter.

The search is carried out in the BrowseBackingBean. If no search term is input, the backing bean retrieves all the listings from ListingDAO and lists them sorted by date. If there is a search term, the backing bean calls the method search in ListingDAO with the search term as the parameter. This method retrieves all listings from the database and filters them by title, ISBN and authors in a case insensitive way. The resulting listings are then returned and shown.

5.4 Use case #6: Purchase a book

When the user begins on the main page, they will see all listings that are available to buy in the form of cards in a list, with the newest first. This list is built by the browse.xhtml file. The XHTML gets all the listings from the BrowseBackingBean, which in turn gets the listings in the database from the ListingDAO method getBuyableListingsSortedByDate().

Every card can be clicked on in order to see additional information about the listing. When clicked the user is navigated to the listing.xhtml page, and the ID of that specific listing is appended at the end of the path as a parameter. The listing.xhtml page will display additional details about the book using the ListingBackingBean. There is also a checkout button that will be rendered if the user is logged in, it is not their own listing and the listing is not already purchased. This will be validated with data from the ListingBackingBean. When pressed it

will take the user to the corresponding checkout page by giving the ID of the listing as a path parameter.

The checkout page will display some information about the book fetched from the CheckoutBackingBean, the user's available credits and a form where the user can input their delivery address. Once they have done so, they can press a confirm purchase button, which will call the confirmPurchase method in CheckoutController. CheckoutController will call the orderListing method in OrderService. The orderListing method will check if the buyer of the listing is also the seller and if so throw an exception. Otherwise, the performTransaction method in BankService will be called and the purchase will be added to the database. The performTransaction method will check that funds are available and if so transact money from the buyer Account to the seller Account and update the database.

At last, if the purchase was successful, the user will be redirected to a success page that will confirm that the purchase was successful and offer the user the possibility of giving a review.

5.5 Use case #9: Sell a book

As described in the user manual, the user has to be logged in to an account to be able to view the publish page. When submitting the ISBN the PublishController class retrieves the entered ISBN value from the PublishBackingBean, and then calls the method BookLookupService.lookupByIsbn, with the ISBN string as a parameter. In this method, an HTTP request to Google Books API is sent, and if successful returns a JSON object with information about the book. Information about the book is extracted (getTitle, getAuthors, etc.) and returned as a LookupResult object.

Required attributes for publishing a book which are retrieved from the API are: title, author(s) and publication year. If the API does not have the information about one of these fields, we do not have enough information about the book which means that it cannot be inserted to the database, and therefore sends this as an error message to the frontend. However, if the LookupResult object is valid, the controller takes the data from the object and puts the values into the variables in the PublishBackingBean which is displayed to the user.

Furthermore, when the user has entered a price and a condition and pressed the button "Publish", the controller will make a call to the class PublishService. This class is responsible for inserting the authors, categories, book and listing into the database. Since the entity Listing is dependent on a Book existing in the database, which in turn is dependent on Author and Category existing in the database, we need to make sure to insert the data in the right order, if the data does not already exist in the database. When everything is assured to be inserted correctly, the user gets redirected to the newly created listing.

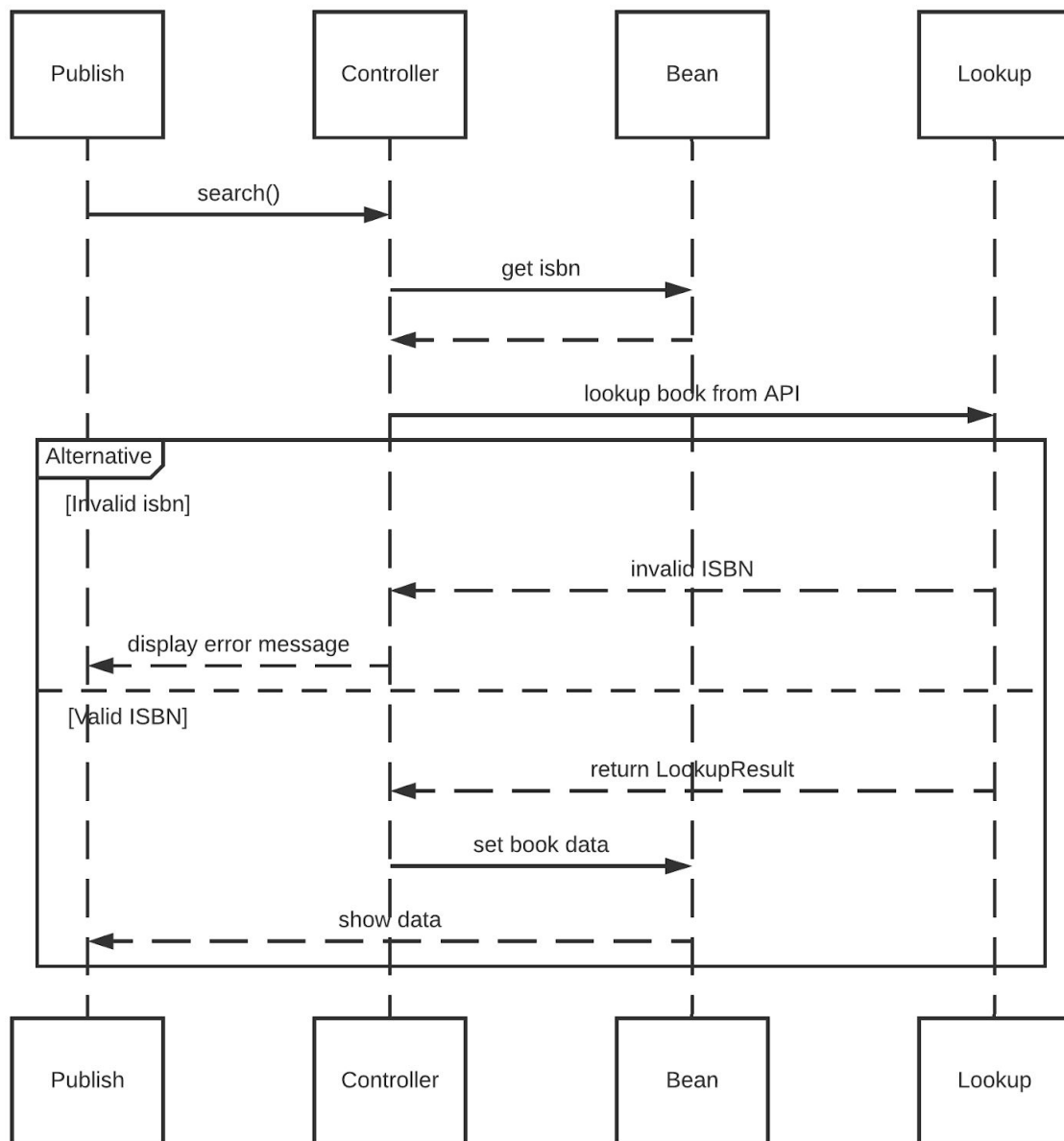


Figure 10. *Sequence diagram of ISBN search process.*

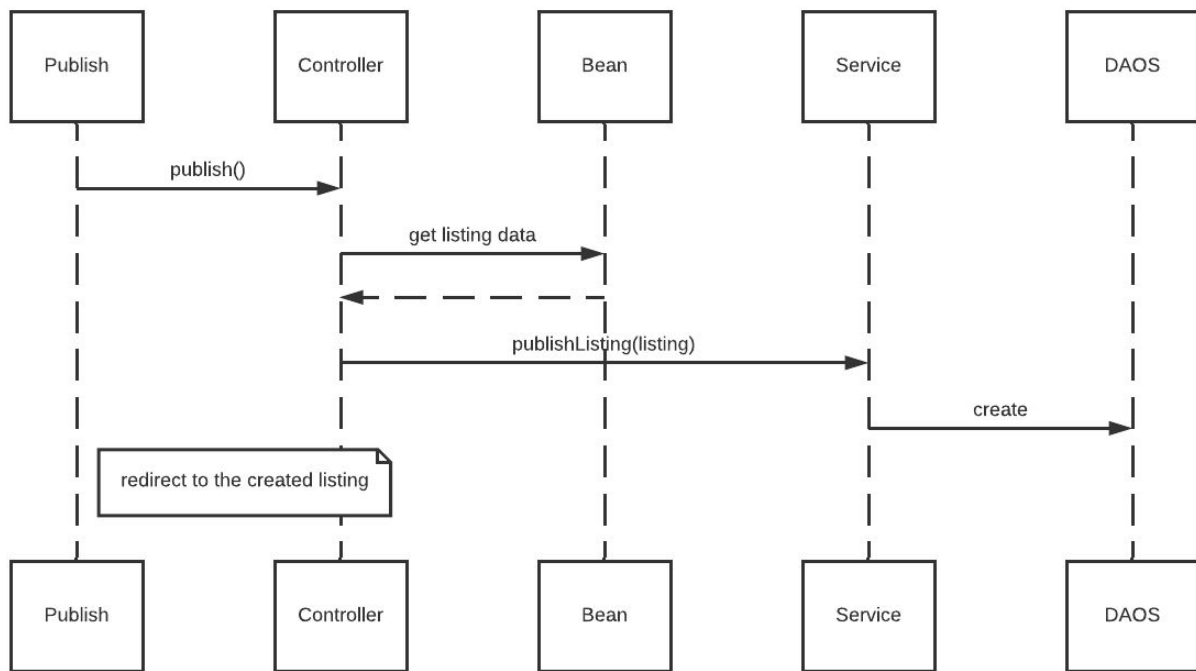


Figure 11. *Sequence diagram of publish listing process.*

5.6 Use case #10: Log out

The possibility to log out is only available for users that are already signed in. It is accessed in the navigation bar under the sub-menu “Mina sidor”. When pressing the button, it calls the method `logout` located in `LogoutController`. This utilizes a function in the class `ExternalContext` to invalidate the current user’s session.

6 Package diagram

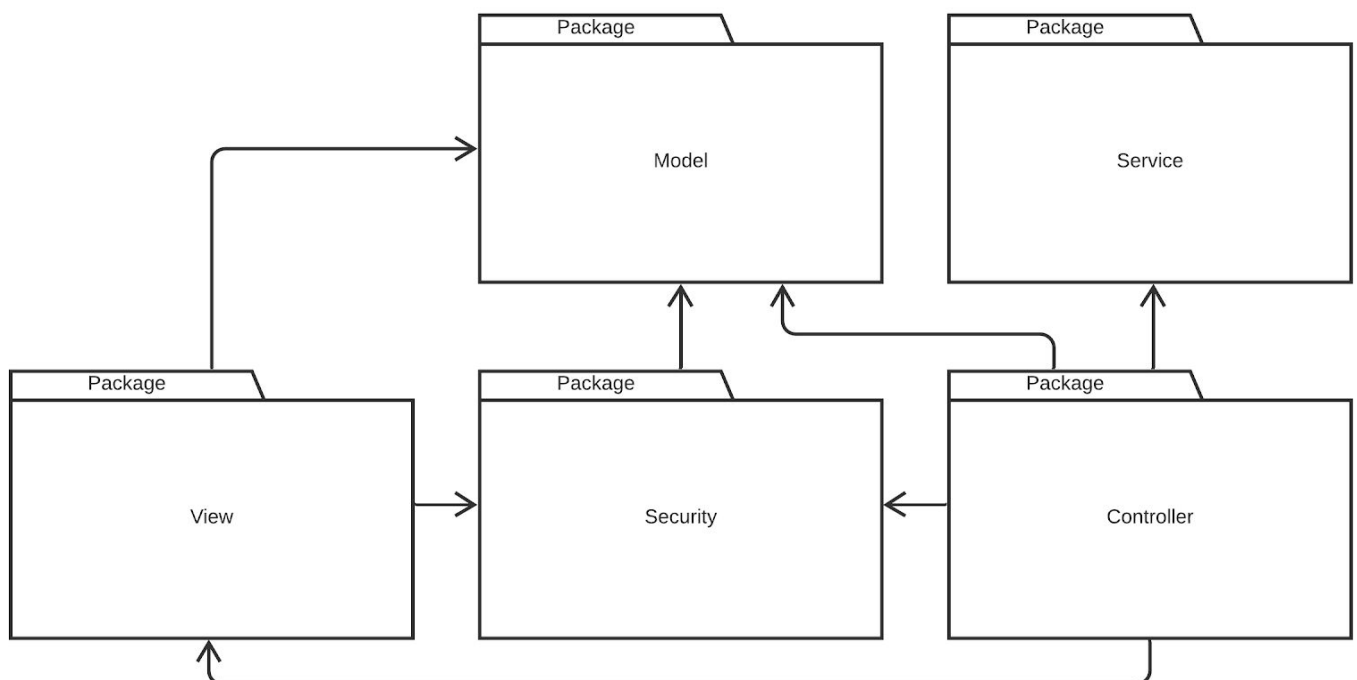


Figure 12. *High-level dependencies between packages.*

7 Code coverage report

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.group05.booksofbliss.controller	<div><div></div></div>	0 %	<div><div></div></div>	0 %	37	37	129	129	23	23	8	8
com.group05.booksofbliss.view	<div><div></div></div>	0 %	<div><div></div></div>	0 %	82	82	64	64	78	78	9	9
com.group05.booksofbliss.model.entity	<div><div></div></div>	88 %	<div><div></div></div>	72 %	59	258	0	121	0	135	0	8
com.group05.booksofbliss.view.converter	<div><div></div></div>	0 %	<div><div></div></div>	0 %	9	9	13	13	3	3	1	1
com.group05.booksofbliss.model.entity.attribute	<div><div></div></div>	85 %	<div><div></div></div>	61 %	13	33	0	6	0	12	0	1
com.group05.booksofbliss.service	<div><div></div></div>	93 %	<div><div></div></div>	94 %	3	41	13	76	0	16	0	2
com.group05.booksofbliss.view.validation	<div><div></div></div>	0 %	<div><div></div></div>	0 %	6	6	8	8	2	2	1	1
com.group05.booksofbliss.model.database.dao.key	<div><div></div></div>	85 %	<div><div></div></div>	65 %	7	20	0	5	0	7	0	1
com.group05.booksofbliss.security	<div><div></div></div>	63 %	<div><div></div></div>	66 %	3	7	5	12	2	4	1	2
com.group05.booksofbliss.model.database.converter	<div><div></div></div>	84 %	<div><div></div></div>	50 %	2	6	2	8	0	4	0	1
com.group05.booksofbliss	<div><div></div></div>	0 %	<div><div></div></div>	n/a	1	1	1	1	1	1	1	1
com.group05.booksofbliss.model.service	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	26	0	57	0	17	0	5
com.group05.booksofbliss.model.dao	<div><div></div></div>	100 %	<div><div></div></div>	100 %	0	36	0	57	0	30	0	10
Total	1 267 of 3 842	67 %	155 of 460	66 %	222	562	235	557	109	332	21	50

Figure 13. JaCoCo testing code coverage report.

8 Model diagram

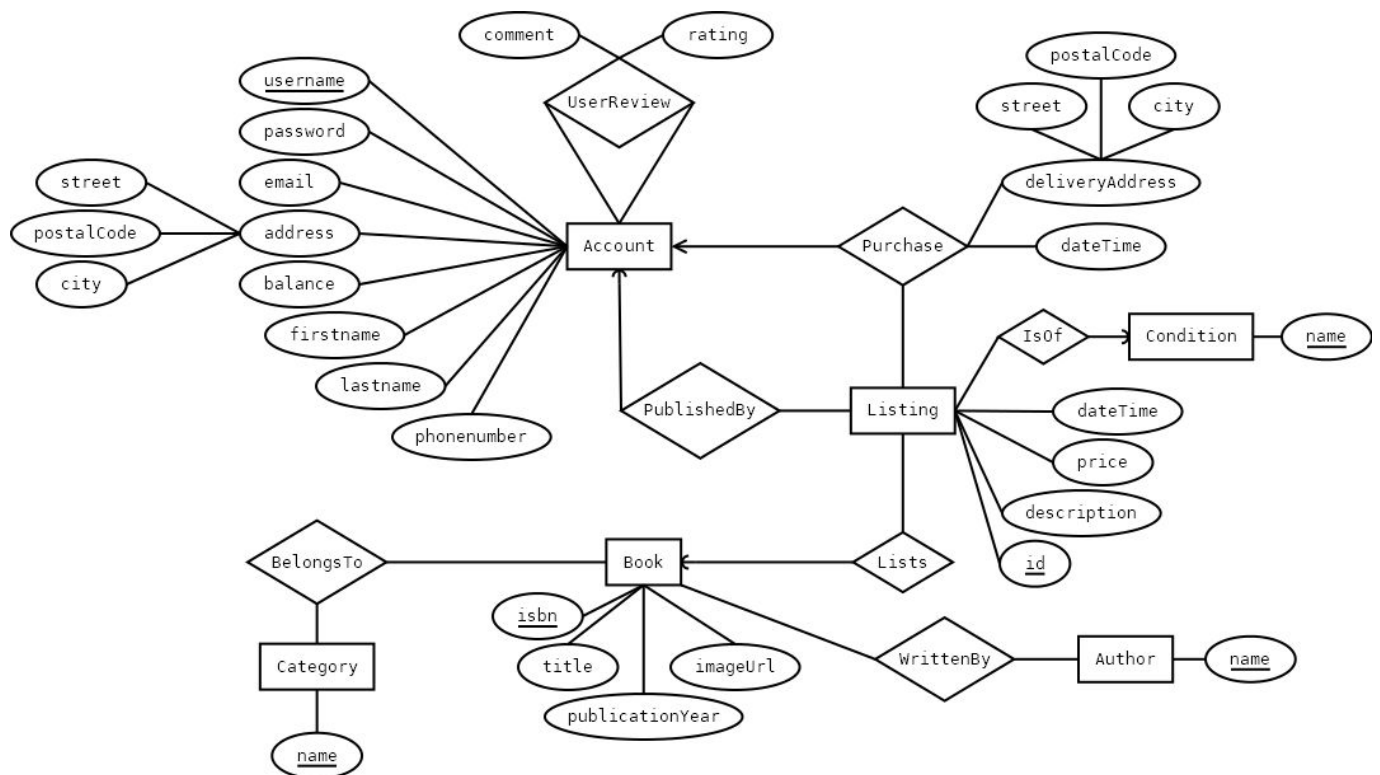


Figure 14. Entity-relationship diagram describing the project's data model.

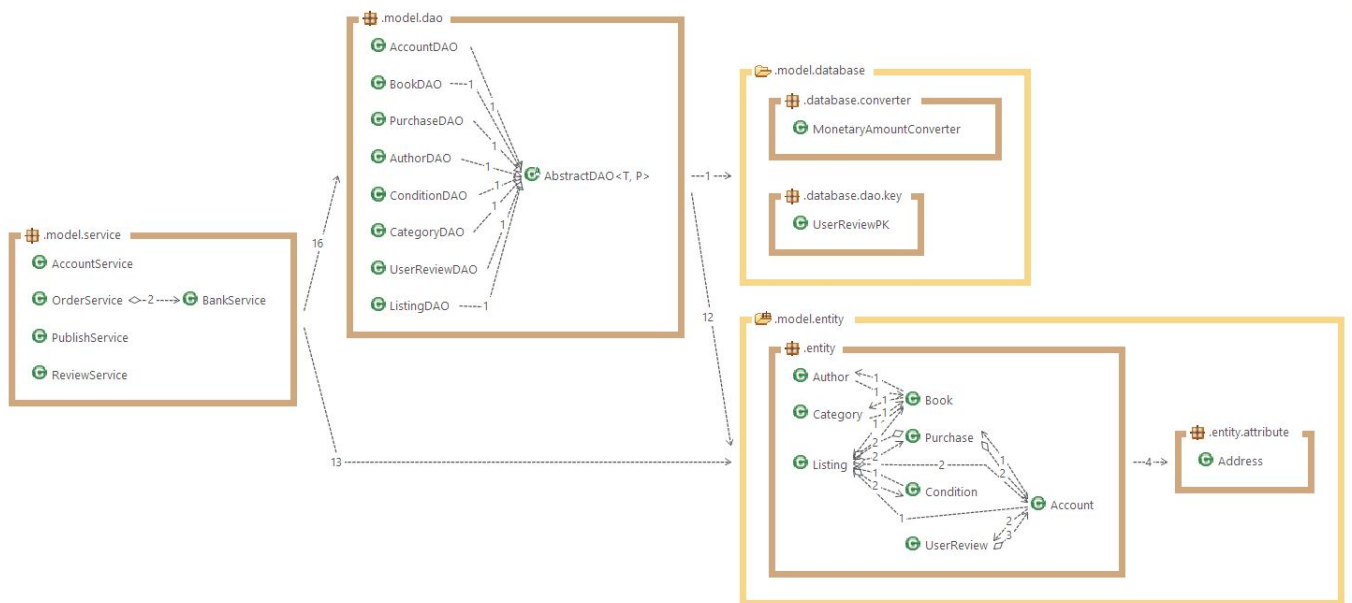


Figure 15. Dependencies between classes in the model package.

9 Responsibilities

- **Benjamin Sannholm:** Checkout page, Buy credits page, Give review page and Authentication. Testing of entity getters/setters/equals/hashcode, BobIdentityStore, UserReview, UserReviewDAO and BookLookupService.
- **Sophia Pham:** Settings page, Part of Account, Registration page, Login page and Logout. Testing of Account and AccountService.
- **Gustav Pihlquist:** Publish listing page, ISBN book lookup service. Testing of PublishListing, ReviewService, BookLookupService and ListingDAO.
- **Oliver Karmetun:** Browse page, Listing page, Testing of ListingDAOTest, OrderServiceTest, UserReviewTest, BookDAOTest, ConditionTest, CategoryTest, BookTest, AuthorTest and BankServiceTest.
- **Endrit Karpuzi:** Listing page, Navbar, Search