Linked List

Sprint Report

Team: Partigiano

1 Introduction

The following article is the report of a Software Engineering project, that took place in 2025. The source code and file files can also be found in Github,

1.1 Purpose

The purpose of the project is to develop a robust ***Linked List***. Other than the different functionalities of this application, the project should be designed in such a way that it is maintainable and extendable, as well fully documented. This shall result in fast and correct development and deployment of new features in the future. Another target of the project was to implement agile methods to develop software. In our case, I used **scrum.**

1.2 Document Structure

The rest of this document is structured as follows:

- Section 2 specifies the technologies used during the development of the project.
- Section 3 describes our Scrum team and specifies this Sprint's backlog.
- Section 4 dives into the Use Cases, which derive from User Stories.
- Section 5 explains the project's Architecture and Design.

2 Tools and Technologies

- **Java**: A high-level, class-based, object-oriented **programming language**. Widely used and famously quoted as “write once, run anywhere” as she has very few dependencies.
- **Spring Boot**: An open-source Java framework used for programming standalone, production-grade Spring-based applications.
- **Bootstrap:** An open-source **CSS framework** directed at responsive front-end web applications.
- **GitHub**: A **developer platform** that allows development teams to create, store and manage their code. It uses and extends the Git software.
- **Junit**: A **unit testing framework** for java.
- **PostgreSQL**: A free and open-source **relational database** management system (RDBMS) emphasizing extensibility and **SQL** compliance.
- **Jira**: A product development tool that allows bug tracking, issue tracking and agile project management.
- **IntelliJ**: An IDE for developing computer software written in Java and other JVM-based languages.
- **Maven**: A **build automation** tool used for, among others, java projects.

3 Scrum team and Sprint Backlog

3.1 Scrum team

Product Owner	Alex Pournaras
Scrum Master	Alex Pournaras
Development Team	Alex Pournaras

3.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories	Description
1	**9/3/2024**	**22/3/2024**	**2**	**None**	**Determine the tools and technologies to be used to build the project. Create use cases, domain classes and UML diagrams.**
2	**24/3/2024**	**07/04/2024**	**2**	**US1, US2, US3**	**Implement basic user functionalities such as creating and logging in/out of an account.**
3	**07/04/2024**	**21/04/2024**	**2**	**US5, US6, US7, US8, US9**	**Complete book offer functionalities:** **Add, remove book offers with info for them.** **Browse book request list. Browse interested users list for each book.** **Select a user to hand each book**

4	**24/04/2024**	**05/05/2024**	**2**	**US4, US10, US11**	**Add User Profile editing and book search functionalities**
5	**06/05/2024**	**19/05/2024**	**2**	**None**	**Create separate database for testing and test each class.**

4 Use Cases

4.1 Create Linked List

Use case ID	UC1
Actors	User
Preconditions	-
Main flow of events	1. The use case starts when the user calls the **create()** function.
Post conditions	2. A new **Linked List** has been **created** with empty contents.
Alternative flow 1	1. There is no space in the *heap* to allocate memory.
Post conditions	The programs exits, throwing an error message.

Add Node

Use case ID	UC2
Actors	User

Pre conditions	The user has created a Linked List.
Main flow of events	3. The use case starts when the user calls an **add_node** function. 3.1. add_at_head() 3.2. add_at_tail() 3.3. add_at_index() 3.3.1. The user will define the index of the insertion, default: head.
Post conditions	4. A new **node** has been **inserted** to the Linked List.
Alternative flow 1	5. There is **no space** in the ***heap*** to allocate memory .
Post conditions	6. The programs exits, throwing an error message.
Alternative flow 2	7. The user-provided **index is out of bounds**. 7.1. The software asks the user to dictate its next action 7.1.1. Ignore and do nothing. 7.1.2. Insert at head. 7.1.3. Insert at tail.
Post conditions	8. The software throws out of bounds message. 9. The node is inserted or not, depending on user's choice.

Delete Node

Use case ID	UC3
Actors	User
Pre conditions	The user has created a Linked List. For the sake of the main flow, it will have to be non-empty.
Main flow of events	10. The use case starts when the user calls a delete function. 10.1. delete_at_head(). 10.2. delete_at_tail(). 10.3. delete_at_index().

	10.3.1. It's matter mandatory for the user to define the index.
Post conditions	11. The corresponding node has been deleted.
Alternative flow 1	12. The user-provided **index is out of bounds**.
Post conditions	13. No effect to the data structure, a message is thrown, informing the user about that.

Get Node

Use case ID	UC4
Actors	User
Pre conditions	The user has created a Linked List. For the sake of the main flow, it will have to be non-empty.
Main flow of events	14. The user calls the **get_node()** function. It is mandatory for the user to define the index.
Post conditions	15. The corresponding **node** is being **returned**.
Alternative flow 1	16. The user-provided **index is out of bounds**.
Post conditions	17. Void is being returned. A message is thrown, informing the user about that

5 Design

5.1 Architecture

The project consists of **5 packages.**

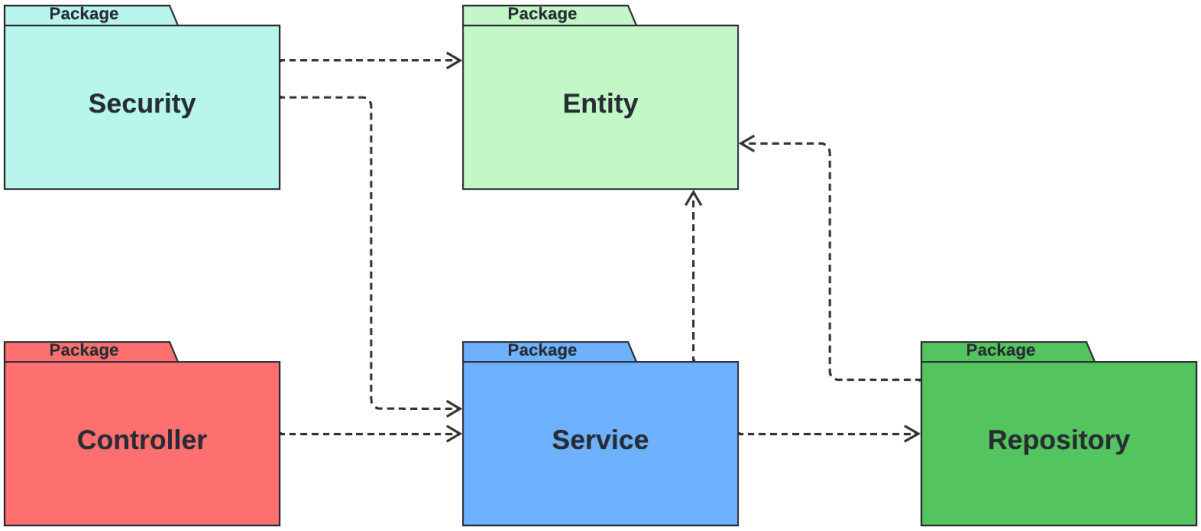
1. **Entity :** The entity package contains the domain classes of the project. Those classes represent real world objects (user, book, etc.).

2. **Repository:** The repository package contains the classes that extend the **JpaRepository** Interface. The allow Spring to map Entity classes into SQL tables. They also provide query methods to save, delete and update java object data into the SQL schema of the project.

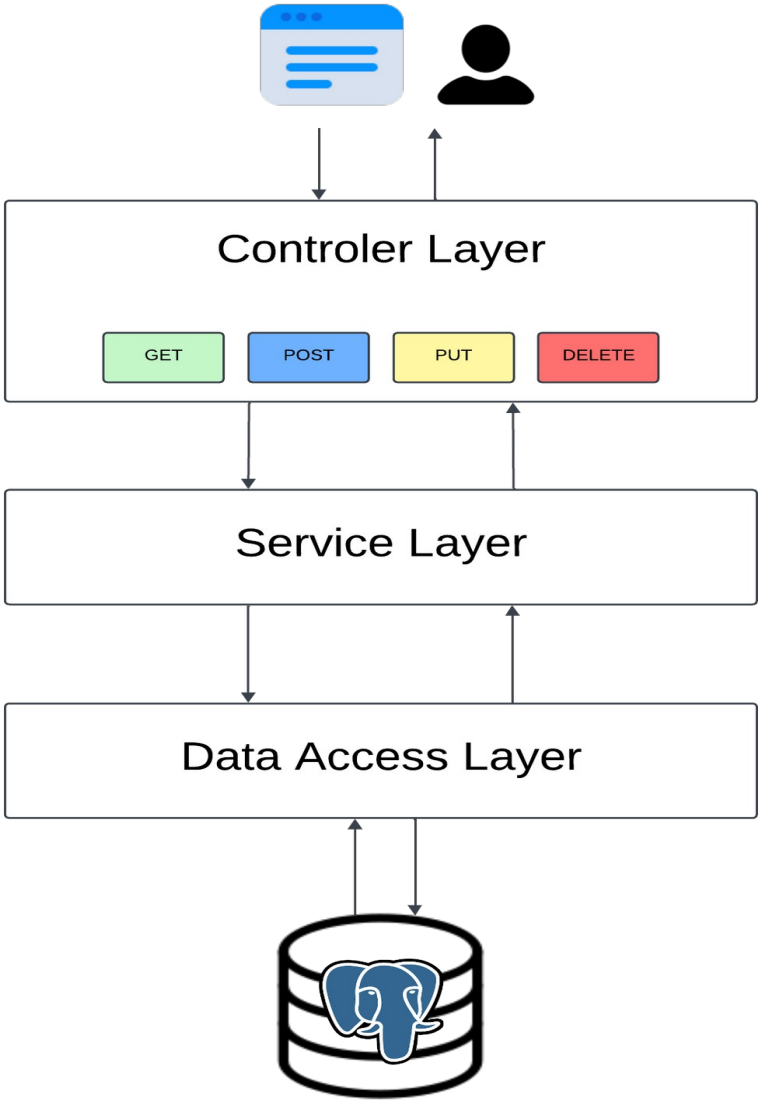
3. **Service:** The Service package contains classes responsible for transferring between the backend and the front-end. The classes apply domain login to determine whether the incoming data requests are valid.

4. **Controller:** The controller package contains classes that take care of **HTTP requests** made on the projects front-end. Each class is responsible for calling the appropriate Service class based on incoming **CRUD** operations and loading the view of each endpoint if needed.

5. **Security:** The security package contains the configurations needed to securely authorize user access. It determines which endpoints should be public and/or accessible for a specific type of user.


Package
Security
Package
Entity
Package
Controller
Package
Service
Package
Repository

The project uses the **3-Tier architecture** which means that the controller requests data from the service layer. The service layer determines if the controller should have access to the requested data given the user credentials. Finally, the service queries the data from the repository and sends it back to the controller.



5.2 Design

The **Entity package** consists of 4 subpackages:

1. Book
2. Recommend
3. Search
4. User