Student:Camara Carina

**Group:30235**

Table of Contents

[1](#__RefHeading___Toc1753_1954039511)

[1](#__RefHeading___Toc1755_1954039511)

[Student:Camara Carina 1](#__RefHeading___Toc1757_1954039511)

[Table of Contents 2](#__RefHeading___Toc1759_1954039511)

[1. Requirements Analysis 4](#__RefHeading___Toc1761_1954039511)

[1.1Assignment Specification 4](#__RefHeading___Toc1763_1954039511)

[1.2Functional Requirements 4](#__RefHeading___Toc1765_1954039511)

[1.3Non-functional Requirements 4](#__RefHeading___Toc1767_1954039511)

[1.4Availability 4](#__RefHeading___Toc1769_1954039511)

[1.5Performance 5](#__RefHeading___Toc4725_2083953617)

[1.6Security 5](#__RefHeading___Toc1771_1954039511)

[1.7Testability 6](#__RefHeading___Toc4729_2083953617)

[1.8Usability 6](#__RefHeading___Toc4731_2083953617)

[2. Use-Case Model 7](#__RefHeading___Toc1773_1954039511)

[1.1 Create User 7](#__RefHeading___Toc1775_1954039511)

[1.2 Create Book 7](#__RefHeading___Toc1777_1954039511)

[3. System Architectural Design 8](#__RefHeading___Toc1779_1954039511)

[4. UML Sequence Diagrams 11](#__RefHeading___Toc1781_1954039511)

[5. Class Design 11](#__RefHeading___Toc1783_1954039511)

[6. Data Model 12](#__RefHeading___Toc1785_1954039511)

[7. System Testing 13](#__RefHeading___Toc1787_1954039511)

[8. Bibliography 14](#__RefHeading___Toc1789_1954039511)

1. Requirements Analysis

# Assignment Specification

The project is an implementation of an application for managing a bookstore. The application has two types of users: the administrator and the employee. The employee is represented by the front desk employee. Both users need to provide an username and a password in order to use the application.

The regular user can perform the following operations: search books by genre, title, author and sell books.

The administrator can perform the following operations: create, update, delete and view on books (book information: title, author, genre, quantity, and price) and create, update, delete and view on regular users’ information. Also, the admin can generate two types of reports files, one in pdf format and one in csv format, with the books out of stock.

# Functional Requirements

The application uses the Tomcat8.0 server for displaying the views. The data is stored in xml files as is described in the diagrams. The application uses the Layers architectural pattern for the general organization. Also, all the inputs of the application is validates against invalid data before submitting the data and saving it in the xml file.

# Non-functional Requirements

## Availability

**Quality attribute definition: <<**Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.>>[1]

**Source of stimulus:** External to System (user sending messages to administrator)

**Stimulus:** Unanticipated Message (wrong format)

**Environment:** Normal Operation

**Artifact:** Process

**Response:** Inform Operator; Continue to Operate

**Response measure:** 350 days every year

**Tactics[2]:**

* Avoid fault become failures
* Fault Detection:

- between processes (ping/echo, heartbeat message)

* + - within a process (exceptions)
* Fault Recovery

- preparation and repair (active/passive redundancy, exception handling, retry, ignore faulty behavior, reconfiguration)

* + - reintroduction (state resynchronization)
* Fault Prevention (Removal from service, predictive model, exception prevention)

## Performance

**Quality attribute definition: <<**Performance is an indication of the responsiveness of a system to execute specific actions in a given time interval. It can be measured in terms of latency or throughput.>>[1]

**Source of stimulus:** Users

**Stimulus:** Initiate transactions

**Environment:** Under normal operations

**Artifact:** System

**Response:** Transactions are processed

**Response measure:** minim 1 seconds and maxim 10 seconds

**Tactics[3]:**

* Resource demand

- increase computational efficiency

* + - reduce computational overhead
  + - manage event rate
  + - control frequency of sampling
  + - bound execution times and queue sizes
* Resource management

- introduce concurrency

* + - maintain multiple copies of data/computations
  + - increase available resources
* Resource arbitration

- FIFO scheduling

* + - fixed/dynamic priority scheduling

## Security

**Quality attribute definition: <<**Security is the capability of a system to reduce the chance of malicious or accidental actions outside of the designed usage affecting the system, and prevent disclosure or loss of information.>>[1]

**Source of stimulus:** Unauthorized employee

**Stimulus:** Attempts to modify pay rate

**Environment:** Normal operations

**Artifact:** Data within the system

**Response:** System maintains audit trail

**Response measure:** Correct Data is Restored within 24 hours and source of tampering is identified

**Tactics[2]:**

* Detect attacks

- detect intrusion

* + - detect service denial
  + - verify message integrity
  + - detect massage delay
* Resist attacks

- identify actors

* + - authenticate actors
  + - authorize actors
  + - limit access
  + - limit exposure
  + - encrypt data
  + - separate entities
* React to attacks

- revoke access

* + - lock computers
  + - inform actors
* Recover from attacks

- maintain audit trail

* + - restore

## Testability

**Quality attribute definition: <<**Testability is a measure of how well system or components allow you to create test criteria and execute tests to determine if the criteria are met. Testability allows faults in a system to be isolated in a timely and effective manner.>>[1]

**Source of stimulus:** Unit tester

**Stimulus: C**ode unit completed

**Environment:** Development

**Artifact:** Code unit

**Response:** Results captured

**Response measure:** 50% Path coverage in 3 hours

**Tactics[2]:**

* Control and Observe System Stat

- specialized interfaces

* + - record/playback
  + - localize state
  + - abstract data sources
  + - executable assertions
* Limit Complexity

- limit structural complexity

* + - limit nondeterminism

## Usability

**Quality attribute definition: <<**Usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.>>[4]

**Source of stimulus:** User

**Stimulus:** Downloads a new application

**Environment:** Runtime

**Artifact:** System

**Response:** User uses application productively

**Response measure:** Within 2 min of experimentation

**Tactics[2]:**

* Support User Initiative

- cancel

* + - undo
  + - pause/resume
  + - aggregate
* Support System Initiative

- maintain task model

- maintain user model

- maintain system model

2. Use-Case Model

## 1.1 Create User

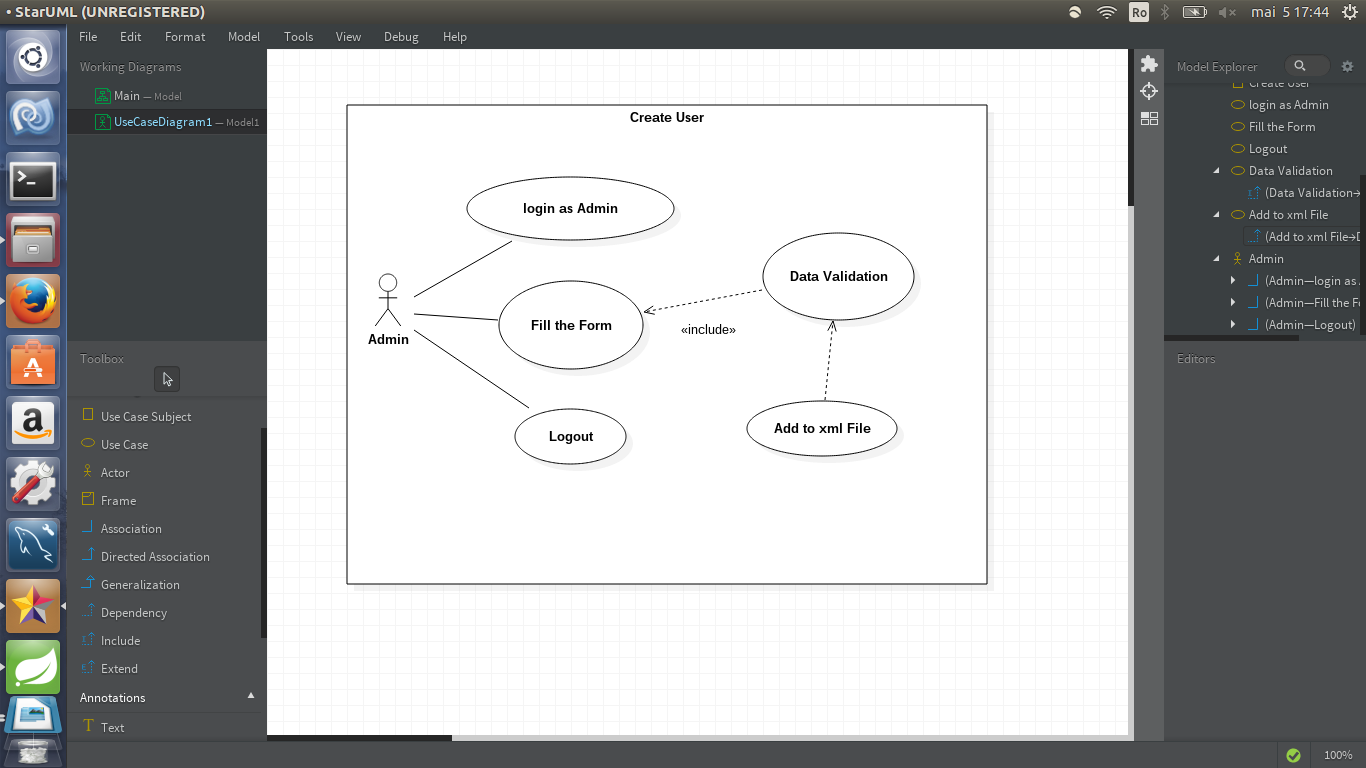
**Use case:** Create User

**Level:** User Goal (level 2 sea)

**Primary actor:** Admin

**Main success scenario:** The admin authenticate successfully in the application. In the main window, he selects the New User button and the page will be redirected to the User form. After he inserts all the valid data, he needs to press the Save button and the costumer will be save in the xml file. The button will redirect the page to the table which contains all the users.

**Extensions:**  If the data is valid, the new user is inserted in the xml file and the information will be visible. In case of invalid data, a message will appear next to the invalid field notifying the employee about the error.



## **1.2 Create Book**

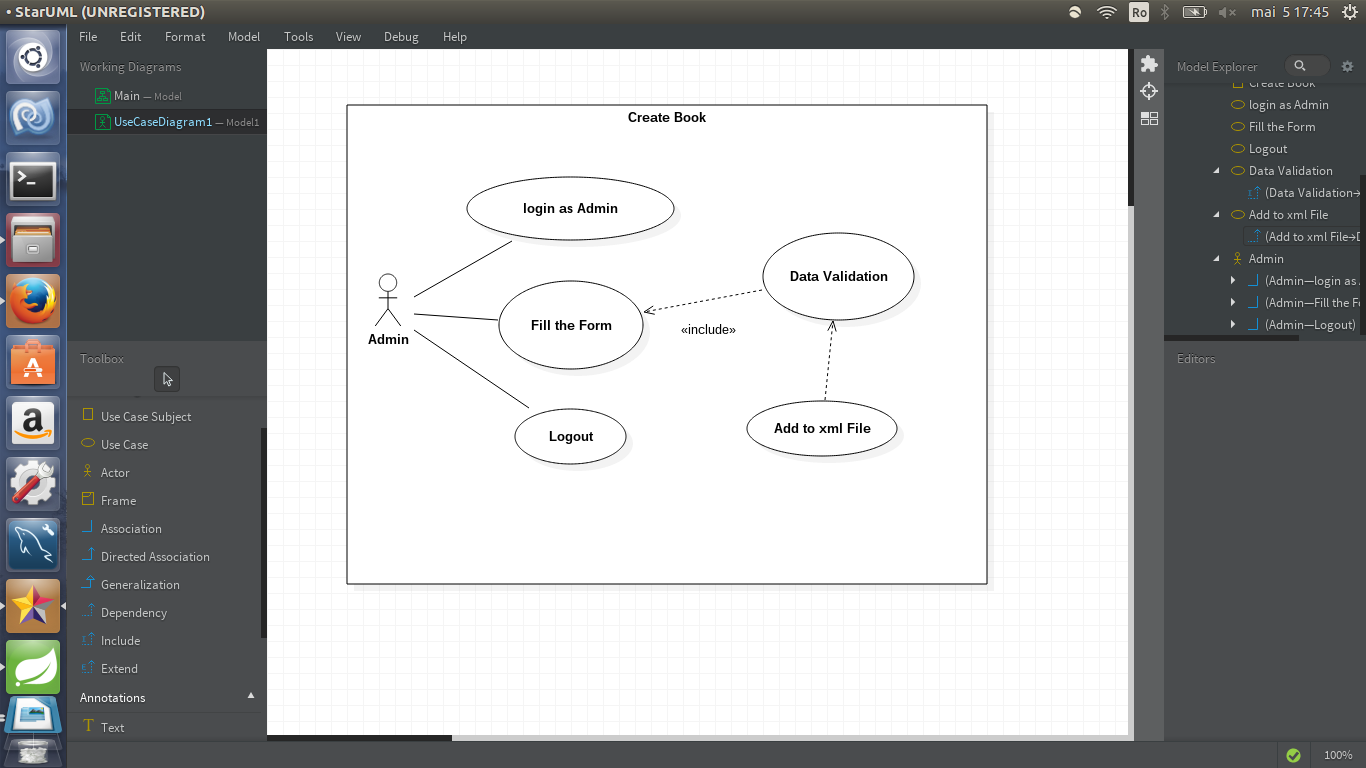
**Use case:** Create Book

**Level:** User Goal (level 2 sea)

**Primary actor:** Admin

**Main success scenario:** The admin authenticate successfully in the application. In the main window, he selects the New Book button and the page will be redirected to the Book form. After he inserts all the valid data, he needs to press the Save button and the transaction will be save in the xml file. The button will redirect the page to the table which contains all the books.

**Extensions:**  If the data is valid, the new transaction is inserted in the xml file and the information will be visible. In case of invalid data, a message will appear next to the invalid field notifying the admin about the error.

****

3. System Architectural Design

**3.1 Architectural Pattern Description**

The Layers architectural pattern has three primary layers: presentation, domain and data source.

Presentation logic is about how to handle the interaction between the user and the software. The primary responsibilities of the presentation layer are to display information to the user and to interpret commands from the user into actions upon the domain and data source.

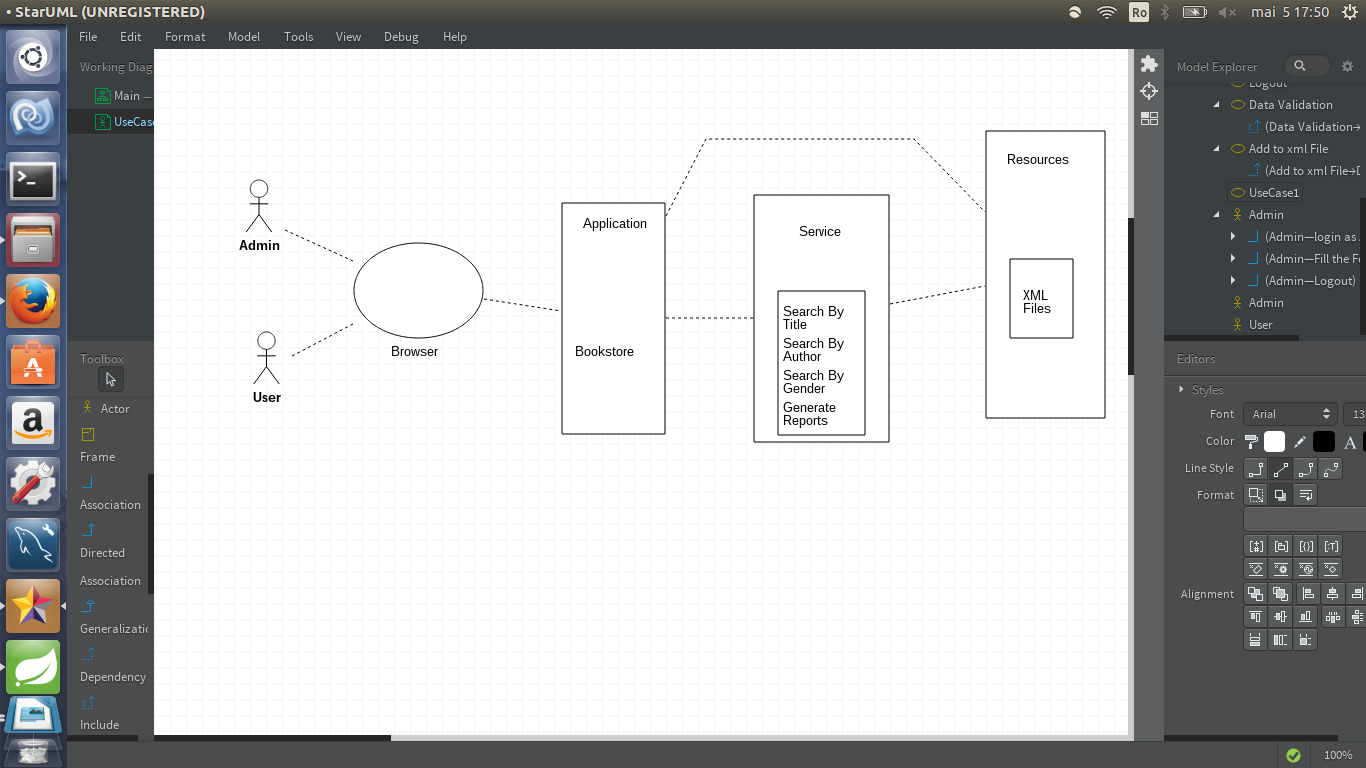
Data source logic is about communicating with other systems that carry out tasks on behalf of the

application. These can be transaction monitors, other applications, messaging systems, and so forth. The biggest piece of data source logic is a database that is primarily responsible for storing persistent data.

The domain logic, also referred to as business logic. This is the work that this application needs to do for the domain we are working with. It involves calculations based on inputs and stored data, validation of any data that comes in from the presentation, and figuring out exactly what data source logic to dispatch, depending on commands received from the presentation.

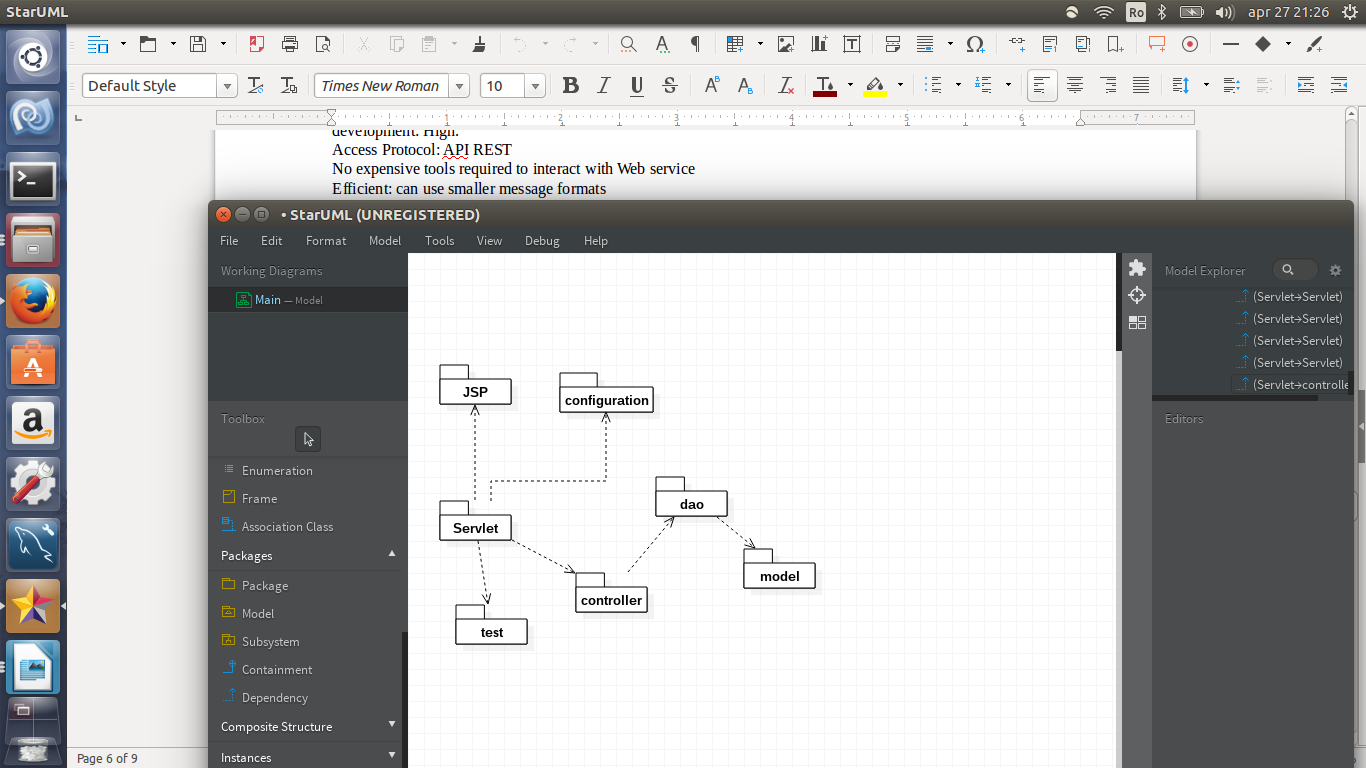
**3.2 Diagrams**

*System’s conceptual architecture*

**

The users interact with the applications through the browser. The navigation is controlled by the application and all the inputs are sent to the application. The service contains all the methods needed and the resources are taken from the database.

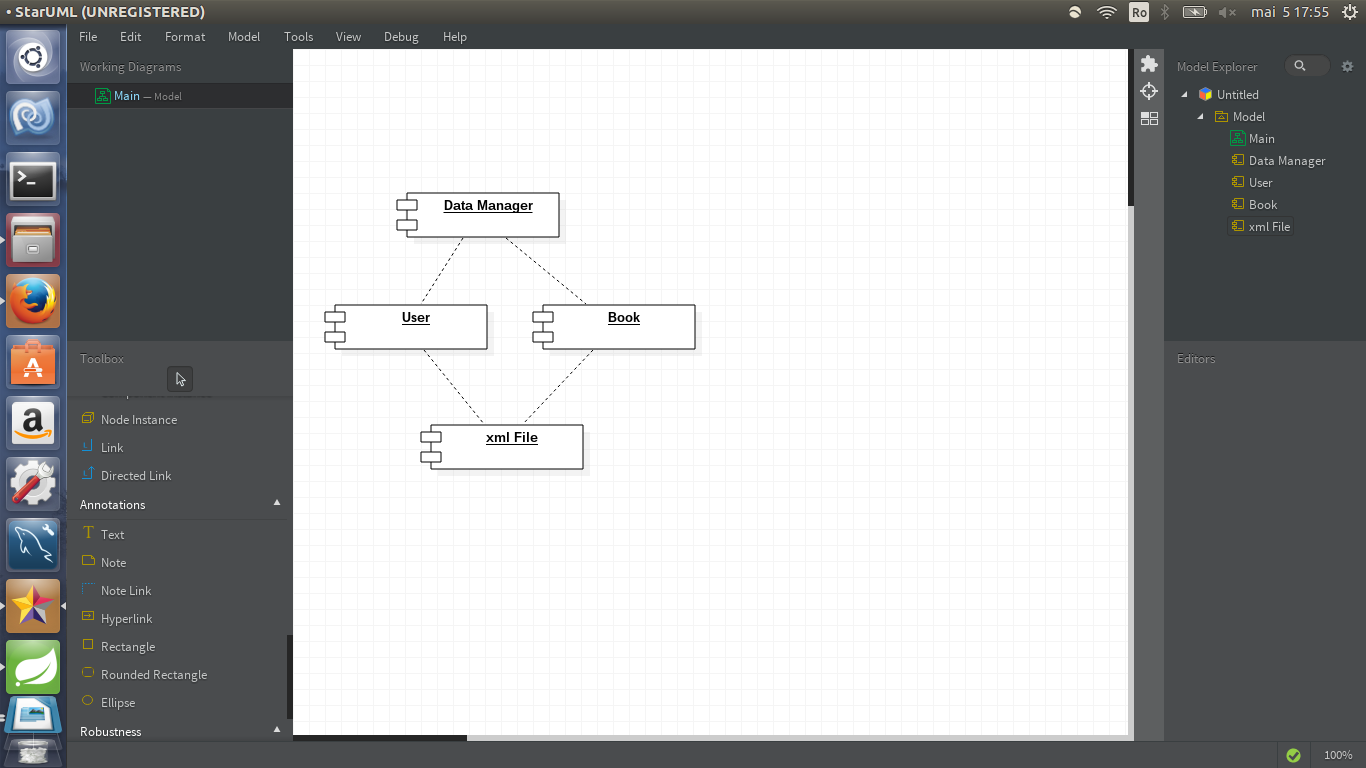
*Package diagram*



As we can see in the package diagram, the project implements an MVC architecture. The package Servlet which contains the web.xml file and the dispatcher file calls the Controller package. This package contains the functions with the annotation @RequestMapping which control what pages to be displayed. The JSP package contains the files for the diplay content (View). The Controller calles the Service package for executing the requested services.

The Test package contains the JUnit testing classes.

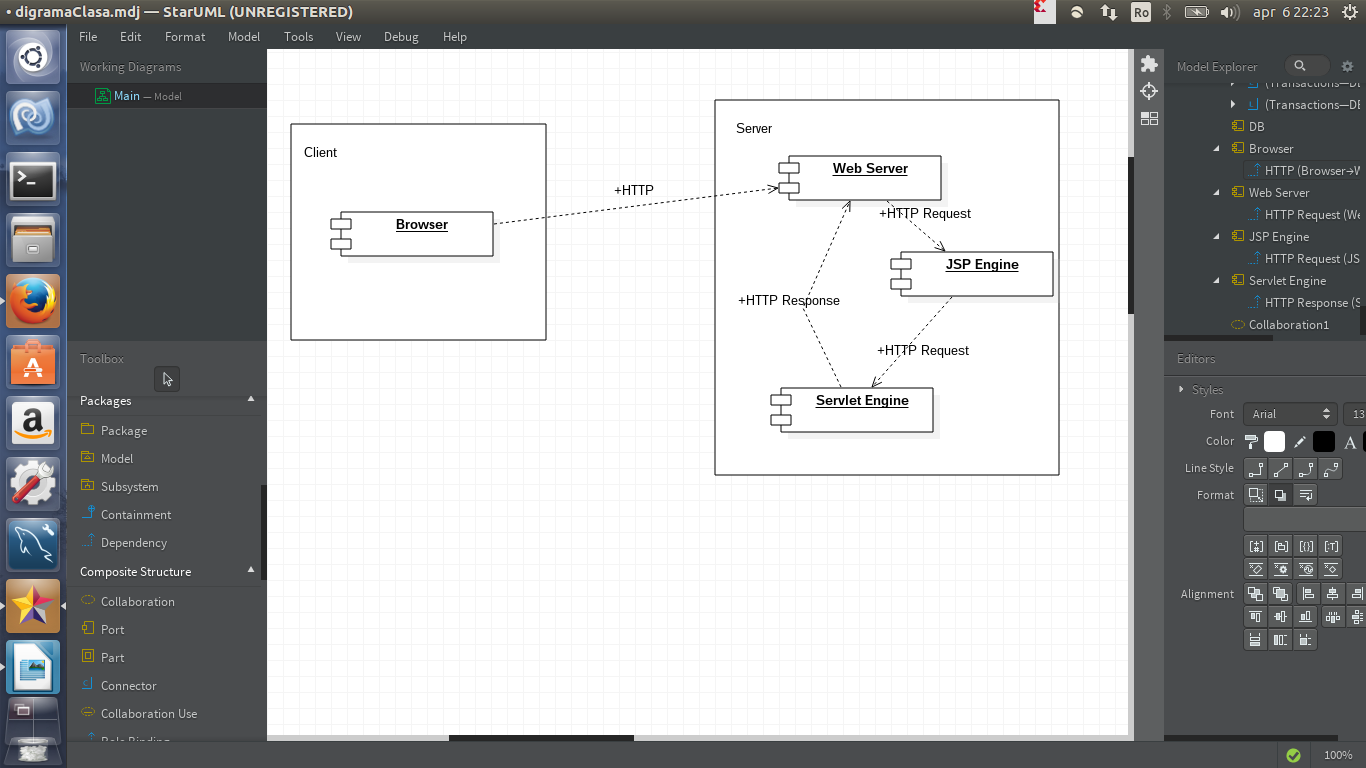
*Component diagram*

**

The component diagram model the architecture of the application in terms of its physical inter-dependent components.

The Client and the Transactions are accessed through the Data Manager called by JSP. Those two components are binded with the Account. All the informations for these components are accessed from the database.

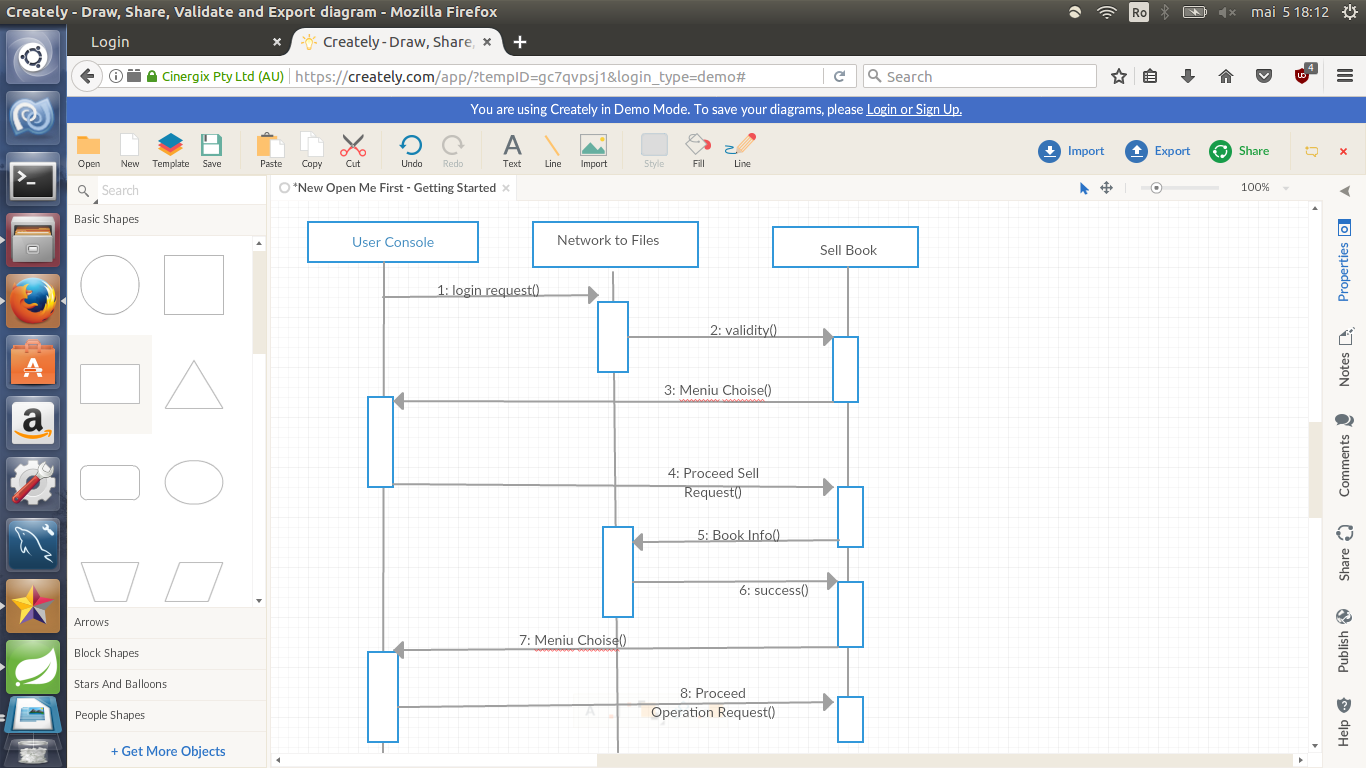
*Deployment diagram*



Deployment diagrams model the hardware used in system implementations, the system components deployed on that hardware, and the associations between those components. Also, it tells what is installed where and with what it communicates. [1]

In this diagram we can see how a JSP file is viewed. The Client, through the browser, connects with the Server. The Web Server sends a HTTP Request to the JSP engine. After the Servlet Engine takes the request, it gives a response to the Web Server which displays the JSP file content.

4. UML Sequence Diagrams



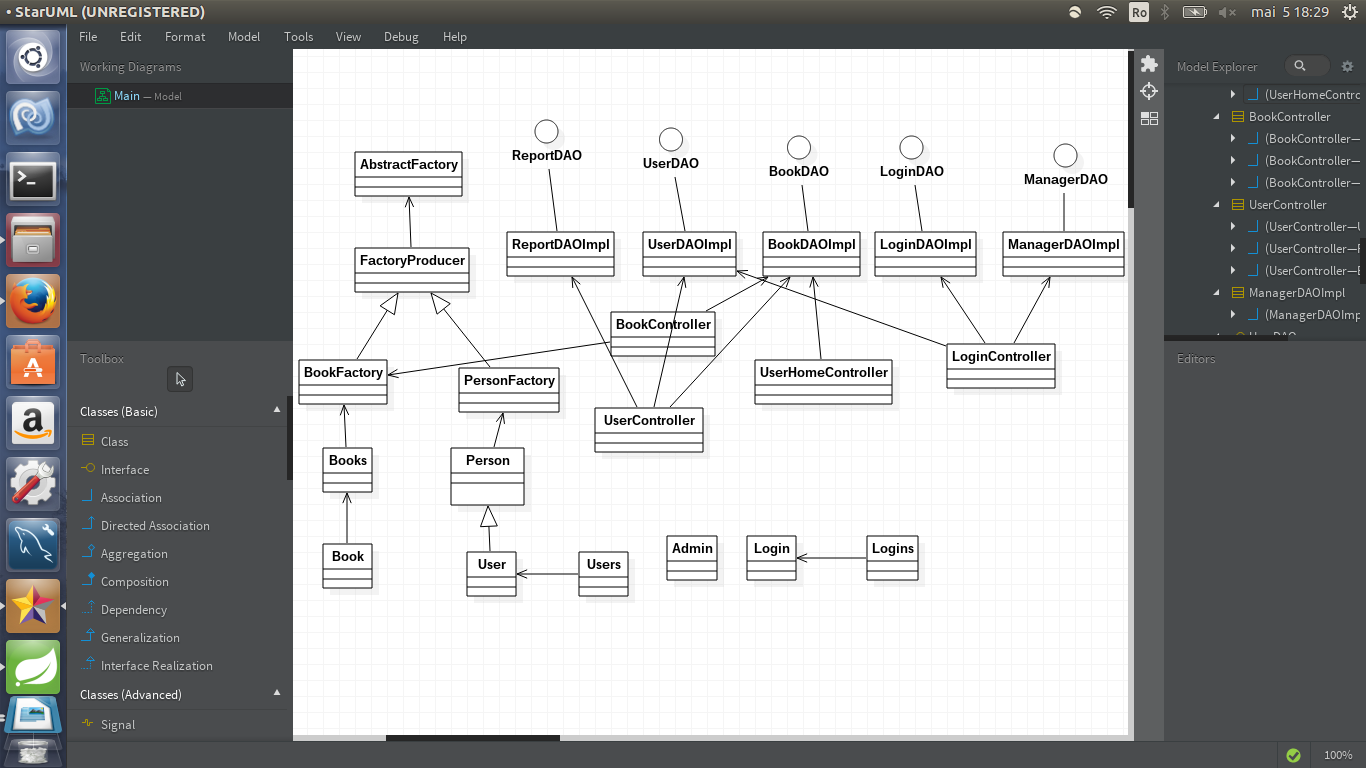
5. Class Design

**5.1 Design Patterns Description**

Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

**5.2 UML Class Diagram**



6. Data Model

*<Books>*

<book>

<idBook>1</idBook>

<title>Insomnii</title>

<gender>Drama</gender>

<author>Irina Binder</author>

<price>22</price>

<stock>3</stock>

</book>

<book>

<idBook>2</idBook>

<title>Hotul de carti</title>

<author>Markus Zusak</author>

<gender>Drama</gender>

<price>18</price>

<stock>2</stock>

</book>

<book>

<idBook>3</idBook>

<title>Poezii</title>

<gender>Poezii</gender>

<author>Mihai Eminescu</author>

<price>10</price>

<stock>2</stock>

</book>

*</Books>*

*<Users>*

<user>

<idUser>1</idUser>

<name>Carina</name>

<username>employeeCarina</username>

<password>password</password>

<email>carina@sql.com</email>

</user>

<user>

<idUser>2</idUser>

<name>Aida</name>

<username>employeeAida</username>

<password>password</password>

<email>aida@sql.com</email>

</user>

<user>

<idUser>3</idUser>

<name>Gia</name>

<username>employeeGia</username>

<password>password</password>

<email>gia@java.com</email>

</user>

</Users>

7. System Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

A test case is a tool used in the process. Test cases may be prepared for software verification and software validation to determine if the product was built according to the requirements of the user.

The application tests all the inputs against invalid data before inserting them in the database.

Also, in the source folder src/java/test are stored the Junit tests. The class CostumerTest contains tests for CRUD operations. First a new costumer is inserted in the database, then the data is modified and deleted. All the tests are completed successfully.

8. Bibliography

<https://www.tutorialspoint.com/software_testing_dictionary/validation_testing.htm>

<https://en.wikipedia.org/wiki/Software_verification_and_validation>