Travelling Application

Analysis and Design Document

Student: Matei Stefania

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <17/apr/17> | <1.0> | Document creation | Stefania Matei |
| <05/may/17> | <2.0> | Creating diagrams | Stefania Matei |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

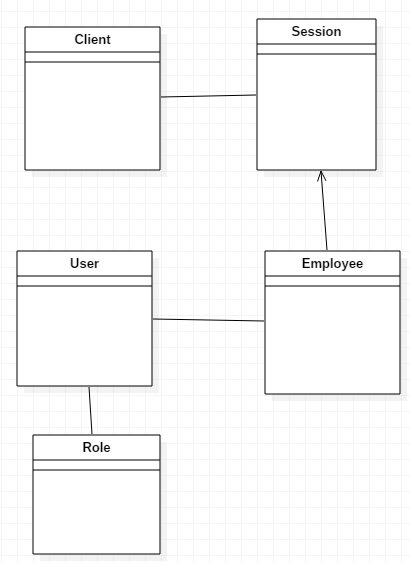
Design an application to hold an evidence about trains and employees. It must have the following functionality:

* CRUD operations on trains—made by regular users
* CRUD operation on employees—made by an admin
* CRUD operations on users—made by an admin
* Provide a way to return the searched trains by an user
* An employee should be able to sell tickets
* User friendly interface
* Export daily reports in PDF format about the sold tickets
* Sending the receipt as a PDF file to a specified address

# Elaboration – Iteration 1.1

# Domain Model

The classes involved in the application will be the following: Client, Employee, User, Role, Session.



# Architectural Design

## Conceptual Architecture

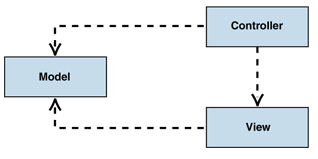
The application will use a MVC architectural pattern. This will ensure the application is easy to maintain over time and provides a flexible development process.

Model–view–controller (MVC) is a [software architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern) for implementing [user interfaces](https://en.wikipedia.org/wiki/User_interface) on computers. It divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient [code reuse](https://en.wikipedia.org/wiki/Code_reuse) and parallel development.

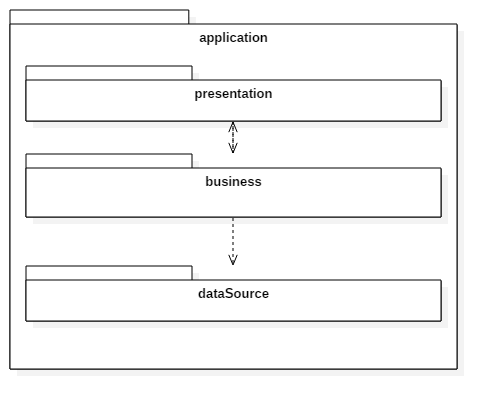
The model is the central component of the pattern. It expresses the application's behavior in terms of the [problem domain](https://en.wikipedia.org/wiki/Problem_domain), independent of the user interface. It directly manages the data, logic and rules of the application.

A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

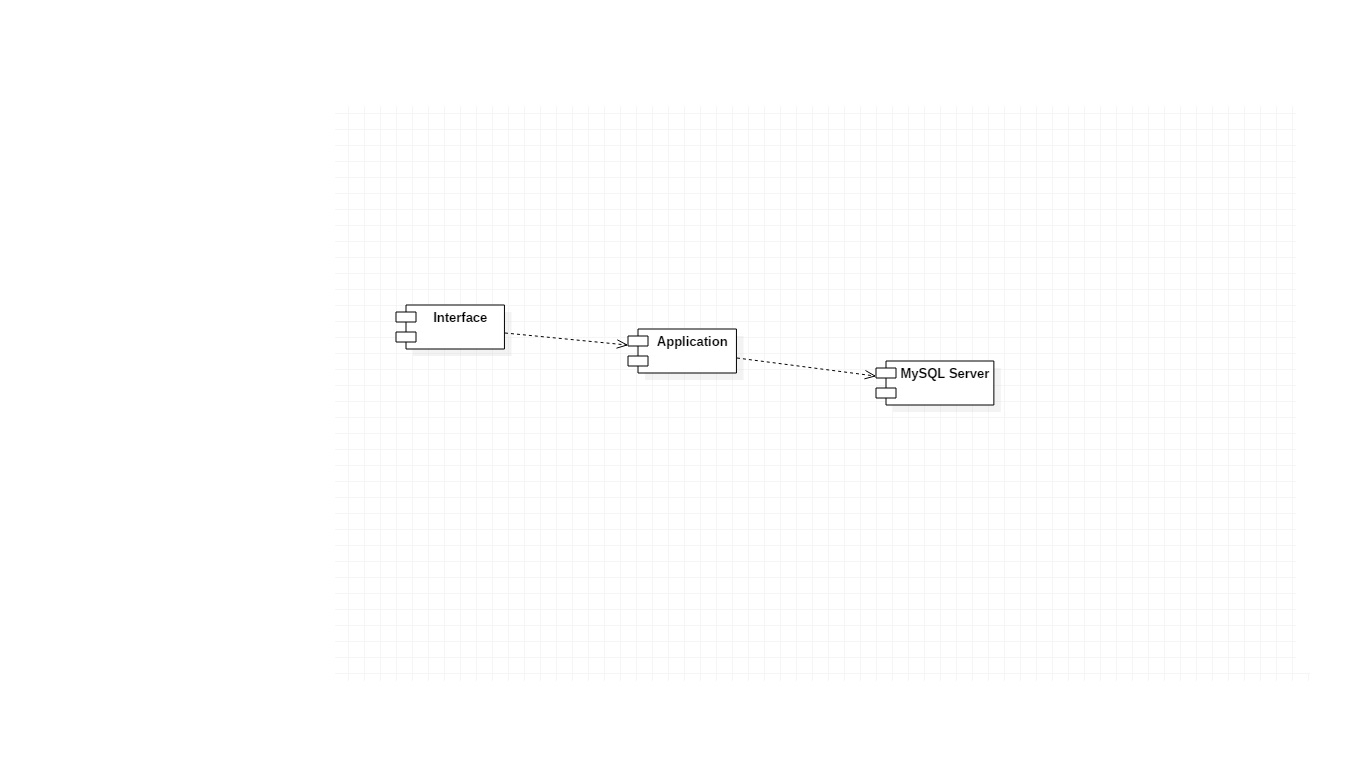
The third part, the controller, accepts input and converts it to commands for the model or view.

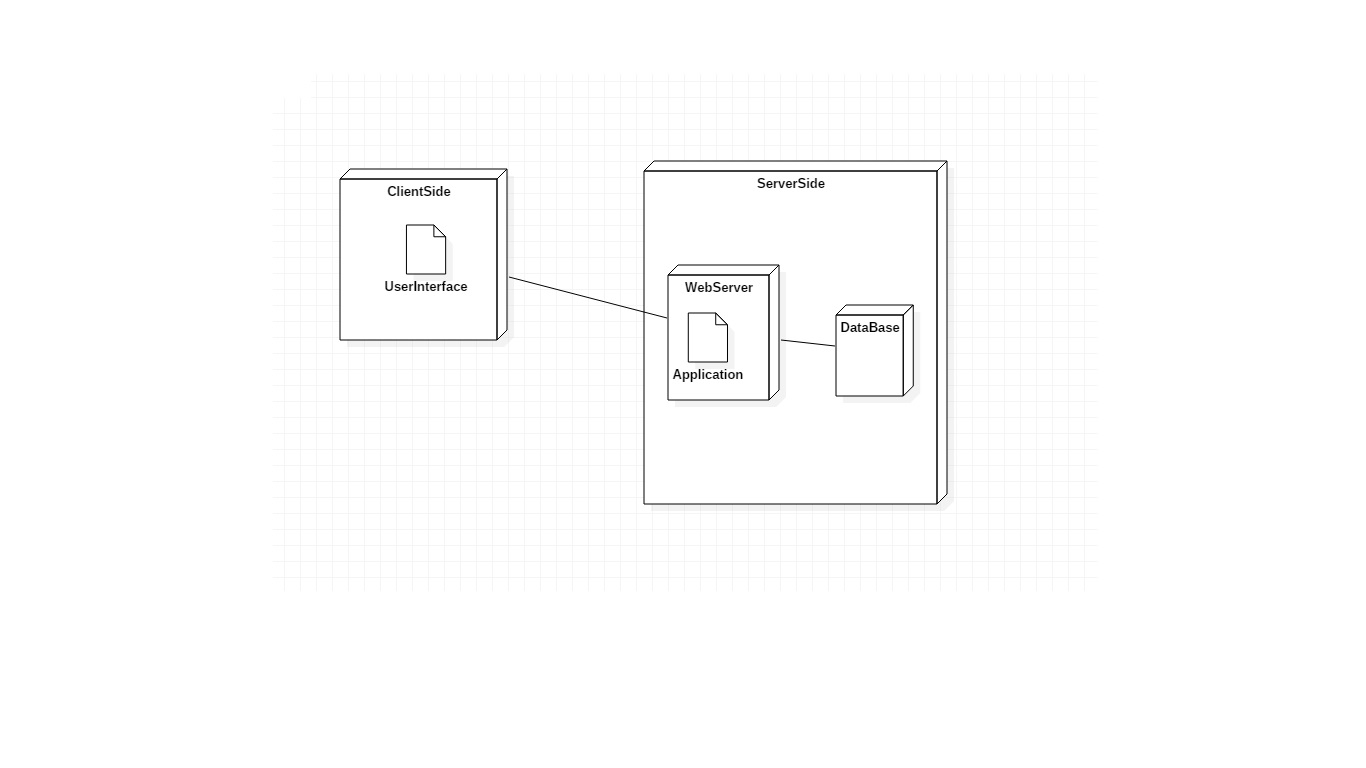


## Package Design



## Component and Deployment Diagrams



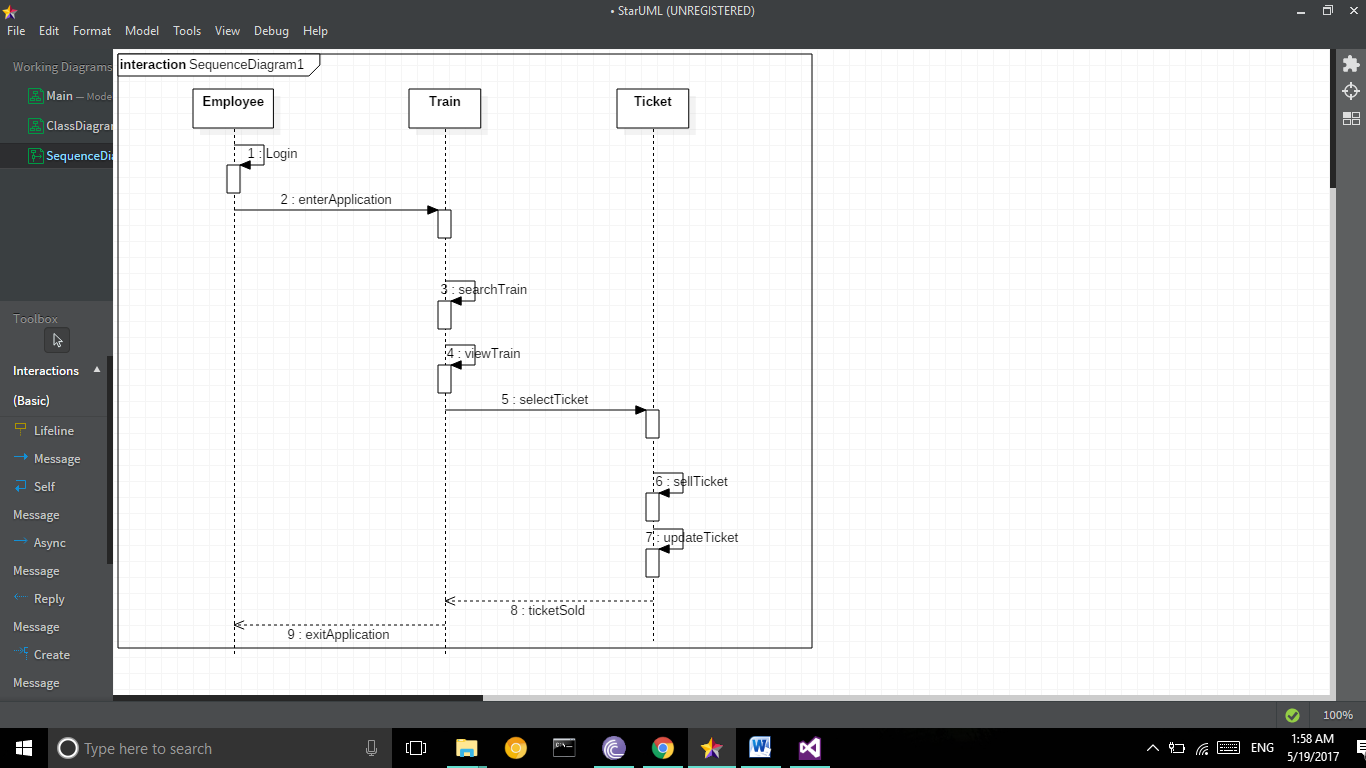


# Elaboration – Iteration 1.2

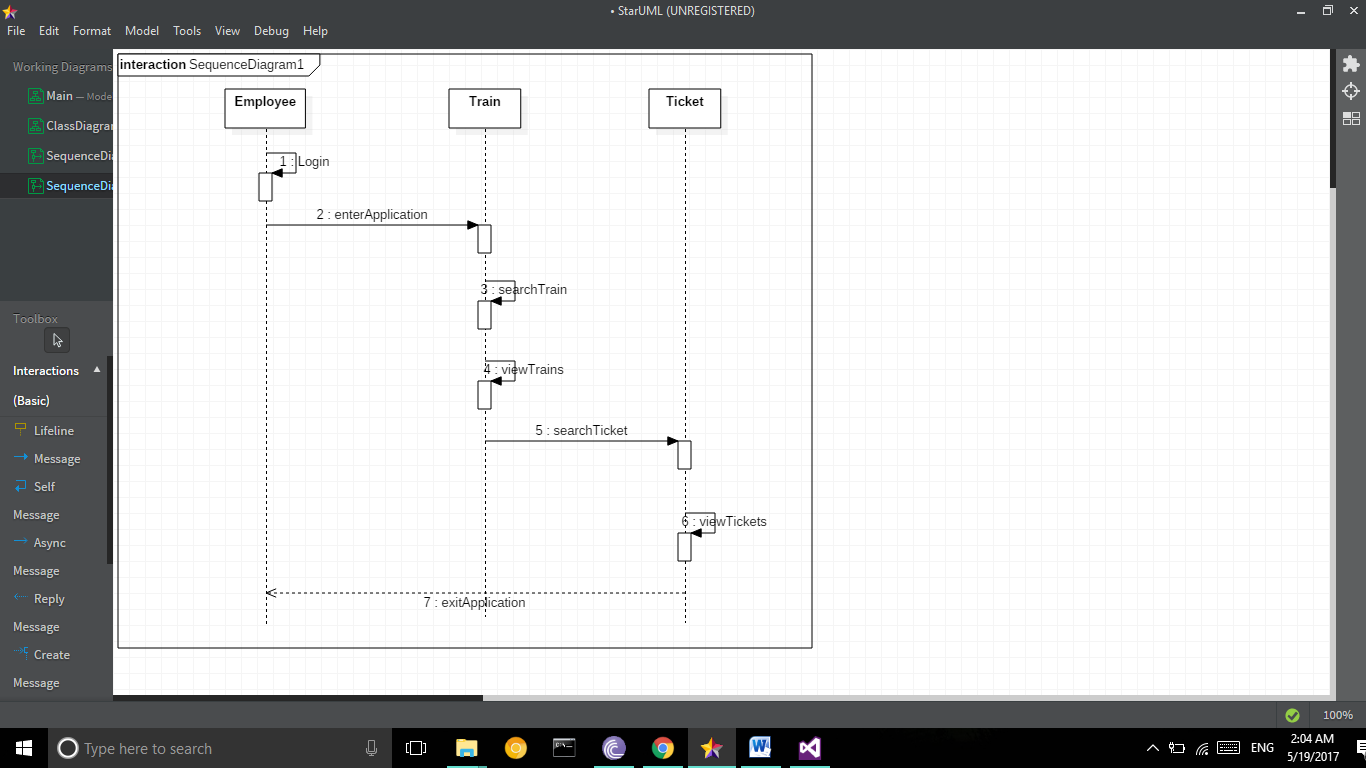
# Design Model

## Dynamic Behavior

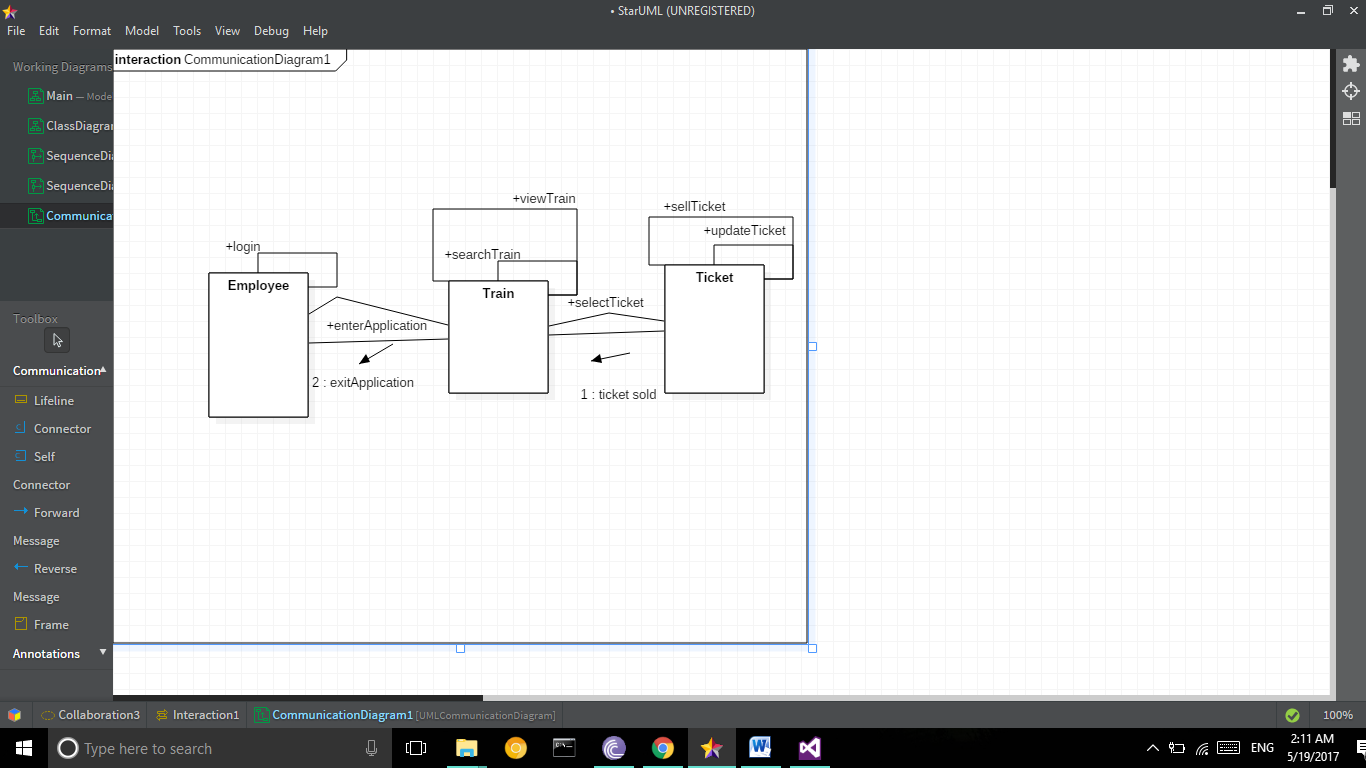
Sequence diagram for sell ticket.



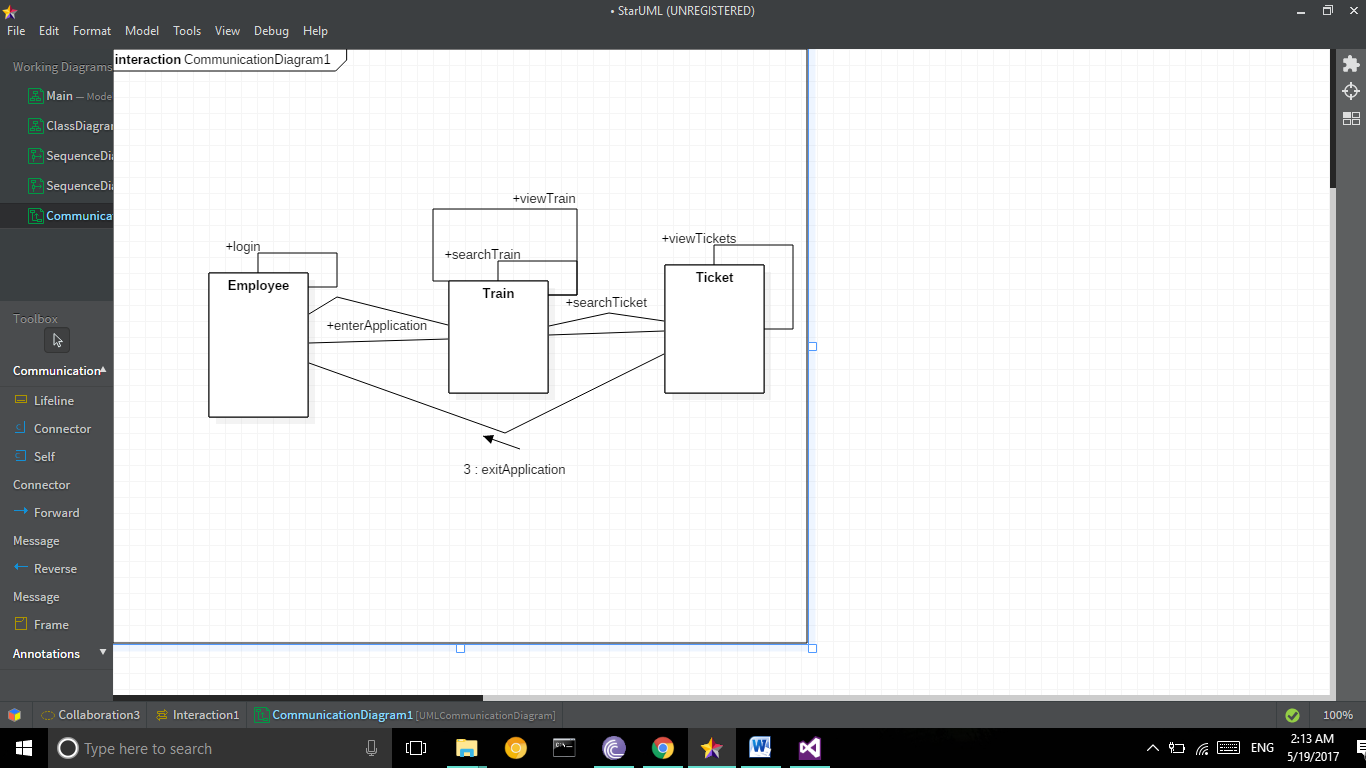
Sequence diagram for search ticket.



Communication diagram for sell ticket



Communication diagram for search ticket



## Class Design

# 

# Data Model

# Unit Testing

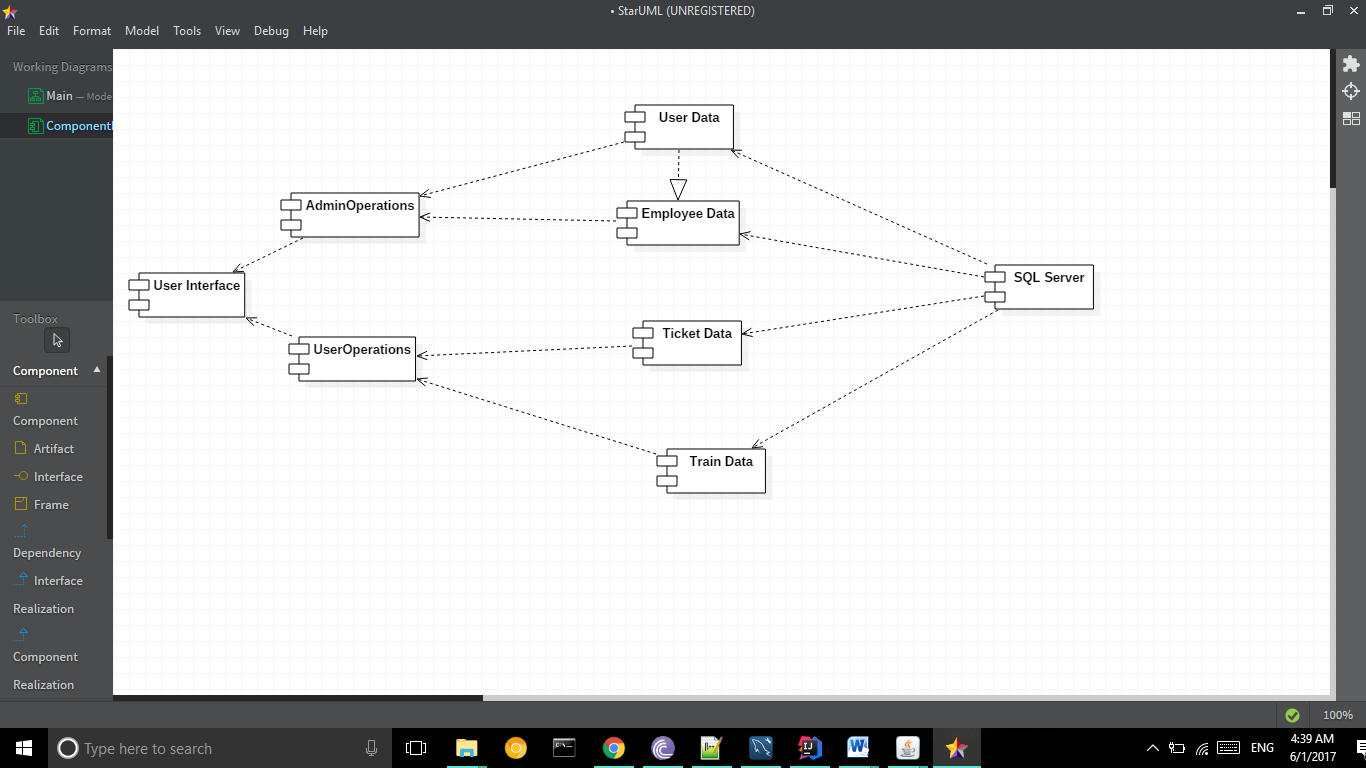
The application will be tested by using Junit4. Some relevant methods will be tested in methods like sellTicketTest, which will return if the method sellTicket works properly. The application will be also tested manually.

# Elaboration – Iteration 2

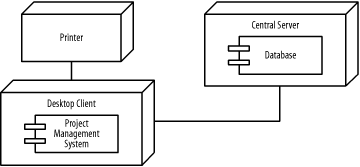
# Architectural Design Refinement

I decided to work with MVC. I worked with 3 packages: Model, View and Controller.

Refined Component Diagram:



Refined deployment diagram:



# Design Model Refinement

# Construction and Transition

# System Testing

The system was tested by using JUnit4 and manual testing.

# Future improvements

The system can be improved by adding the functionality of sending the generated receipt through email. Another one would be to add another type of user, the regular one. This user should be able to access the application and reserve or even buy tickets all by himself.

# Bibliography