Events

Analysis and Design Document

Student: Raluca Pop

**Group: 30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <24/04/2019> | 1.0 | First details | Raluca Pop |
| <01/05/2019> | 1.1 | New improvements | Raluca Pop |
|  |  |  |  |
|  |  |  |  |

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

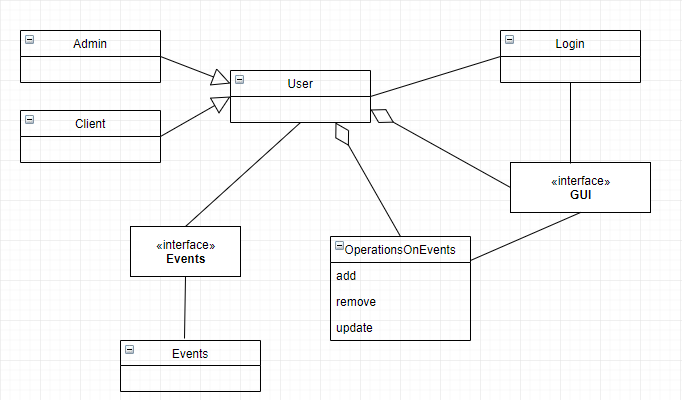
This project will be a web page, where people can book a ticket for an event that is coming. Is will have two users:

-an admin who can add, update and delete events

-a client who can see the events coming soon an can buy a ticket

# Elaboration – Iteration 1.1

# Domain Model



# Architectural Design

## Conceptual Architecture

The following applications design is based on layered architecture pattern, otherwise known as the n-tier architecture pattern.  The layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts.

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request.

**Presentation layer**. This layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.

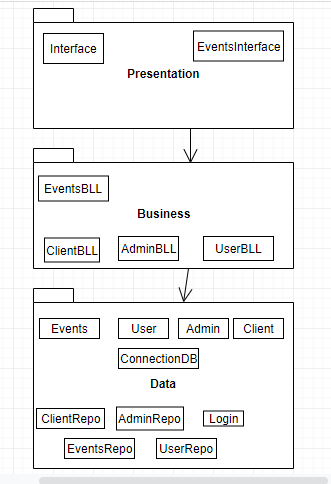
In this layer will be the interface and how the application will look like.

**Business layer**. This layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use**.**

**Data layer**. This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

**Model–View–Controller** (usually known as MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

## Package Design

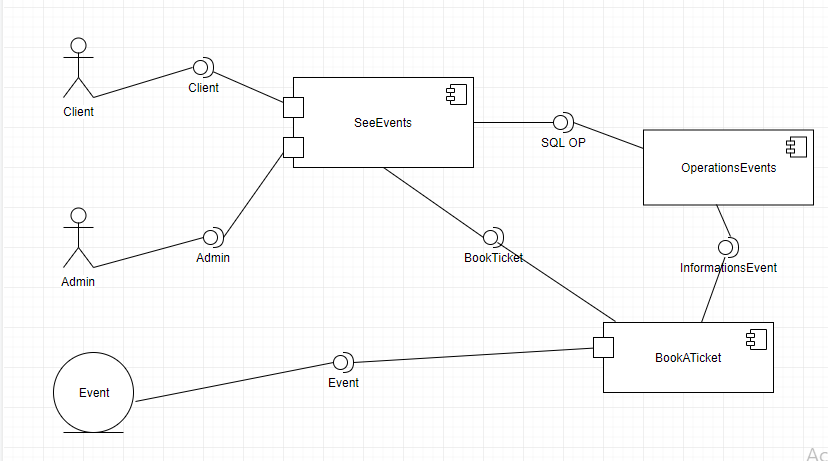


## Component and Deployment Diagrams

Deployment Diagram

# 

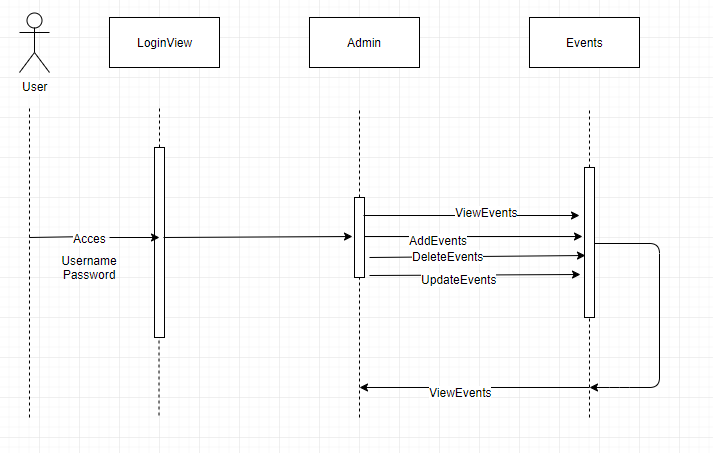
Component Diagram



# Elaboration – Iteration 1.2

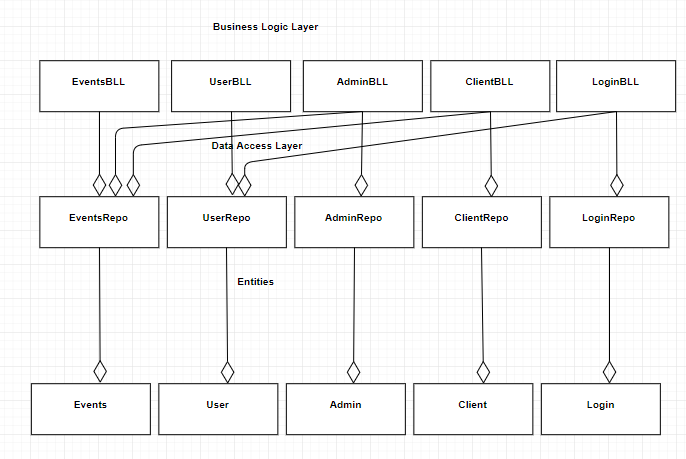
# Design Model

## Dynamic Behavior

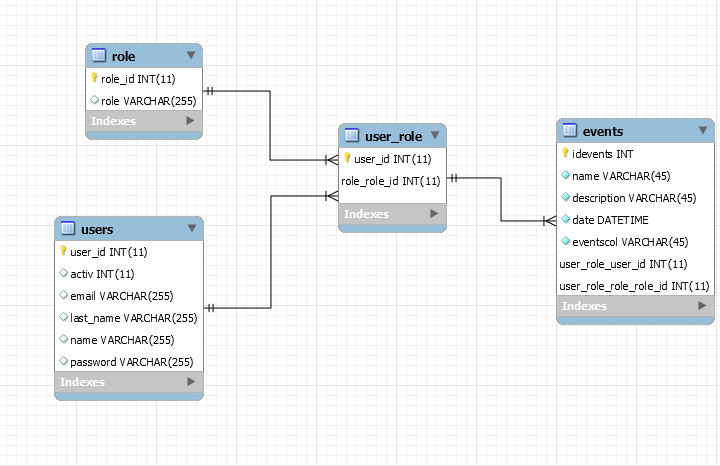


## Class Design

This project will have a model package which will include the business logic layer, the entities and the repositories. The controller package will control the operation that are made on the existing events. The packages will be connected between them, based on the logical representation of the project.

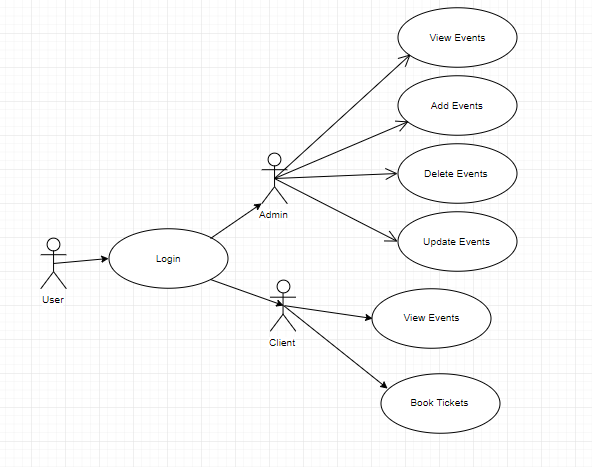


# Data Model

The used databse for this project is MySql Database and the Data Model of this project represents the relations between the main fields of the pro

# Unit Testing

For this project I will use Junit4 for the testing cases. I will test each main operation: add, update, delete. Also, I will test the login operation.



# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

*[Present future improvements for the system]*

# Bibliography