<Biletixx>

System Design

<1>

<19.12.2021>

Sezin Eliçalışkan
Nazlı Zeynep Uysal
Betül Akgül
Can Saygılı
Altan Öztük

Prepared for
SE301 Software Engineering

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

<Project Name>

Table of Contents

<Project Name>

SYSTEM DESIGN DOCUMENT[1]

1. Introduction

The following section provides an overview of the derived System Design Document for the project “Biletixx”. Purpose of the system, design goals, definitions, acronyms and abbreviations are specified below. This document also includes current and proposed software architectures, subsystem services and references.

1.1. Purpose of the System

Biletixx is an online ticket buying website. The main purpose of this system is to sell tickets to users. The users who are looking for an event can buy a ticket for it easily using the website. The system has some useful features that can help users to buy tickets quite easily. Users can search an event then buy a ticket over the internet at any time. The system is free to use and the registration is only needed for buying a ticket. Users can still search for events without registration. The system also have functions for event holders to add and update events.

1.2. Design Goals

The system is a web-based application that users can buy tickets for events. In the system there are three sides. First one is user side, second one is event holder side and third the admin side. Users can use the system for free. They can register, search for events, look event details. They can also buy tickets for an event if they are registered. Event holder can make changes in the system according to the system rules such as add and delete events.

1.Performance:

Our system is fast and efficient. When user or visitor clicks on something it gets acknowledged instantly without delay. User gets their feedback without waiting. Our system stores user, event holder, event information and ticket information.

2.Dependability:

Our system is robust, when user enters wrong information it gives wrong or invalid input message back to the user, it avoids the crashes by user input. Our system is also reliable, it matches users expectations of a ticket selling website. When they enter our website they can find everything they need within their reach. Our website is available 24/7. Users can interact with the website anytime they want but can only buy tickets if they are not sold out.

3.Maintenance:

To obtain an easily maintainable system, we have chosen file, folder and function names relevant to its task. To be able to easily read and modify the code, we made sure not to clutter all the code in one single page. Instead, we have divided them to sperate pages and used template inheritance.

Our system meets the extensibility criteria since we are able to easily add new functions to our existing system without causing any problems.

4.End User:

The system is easy to use for all end users. The buttons are named accordingly, and necessary explanations are provided through the page. If the users enter wrong inputs, the system can detect and block the wrong input and it provides feedback about the appropriate input. The utilization goals are achieved for this system.

1.3. Definitions, Acronyms, and Abbreviations

Admin: Admin is responsible from the whole system, manages the system.

Visitor: Visitor is a user who is not logged into system, can visit the website.

User: User is a registered user who is logged in to system.

Event Holder: Event Holder is a business owner which manages events.

Form: It is an object that consists of fields containing information that user must fill in.

Event: Event is an activity that takes place in real life such as concerts, shows etc.

Ticket: Ticket is an item containing the price and information for an event.

1.4. References

Biletixx RAD document Document, Biletix website

2. Current Software Architecture

The architecture of similar system which we design Online Ticket Sale Site for example Biletix, contains 4 layered architecture like MVC(Model-View-Controller), these layers are;

- **The Entity Layer :**

This layer contains our main classes, namely real objects, that we will use throughout the project. Entity layer also contains entitites which is abstraction of database or model of database in software languages. For example, user and its features like name, id etc.

<Project Name>

- **The Data Access Layer :**

In this layer, simple database operations are performed. The task of this entry is CRUD(Create-Read-Update-Delete) operations and pull data from the database. No other operations are performed in this layer.

- **The Business Layer :**

Workloads are written in this layer. Business layer is the layer that will process the data that has been pulled into the project by Data Access. The data captured through the data layer cannot be directly integrated into the created application. Because with the data layer, only the data has been pulled into the application environment, but the operations to be performed on it are still unclear. For this, the process of adapting the collected data to the application should be carried out. This is provided by the Business Layer. The data tailored to the program created with the business layer is now ready. Operational (CRUD) operations of the database, user roles, management, authorizations are the responsibility of the projects made in this layer.

- **The Presentation Layer :**

The presentation layer is the layer where the interface of the application interacts with the user. The appearance of the data that we have prepared in the business layer with the presentation layer is now determined to go to the user, and now the application we have created is completely finished.

3. Proposed Software Architecture

In our system we will use 4 layered architecture and MVT which is like MVC , these are ;

- **The entity layer :**

  Entities are User, Event, Admin, EventHolder, Ticket.

- **The data access layer :**

  We will use Django ORM for PostgreSQL , it will provide CRUD operation for business layer .

- **The business layer :**

  This layer takes information from DAL(Data Access Layer), then provides services to UI layer according to business rules, these rules are specified in services, the services of our system in business layer;

  USER SERVICES
  LoginToSystem
  LogoutToSystem
  ChangePassword
  BuyTicket
  PayTicket
  CreateAddress
  DisplayAddress
  UpdateAddress

<Project Name>

DeleteAddress
DisplayTickets
UpdatePersonalInformation
DisplayPersonalInformation

EVENT SERVICES
ListEvent
BookEvent
AddEvent
EditEvent
DeleteEvent

EVENTHOLDER SERVICES
DisplayAllBooking
EditBooking

- **The Presentation layer :**

We will make WEB applications to interact with user we will use HTML, JS, Bootstrap. There will be pages then, there will be forms, buttons, lists in pages to interact with user. This layer will be interact with user and business layer .

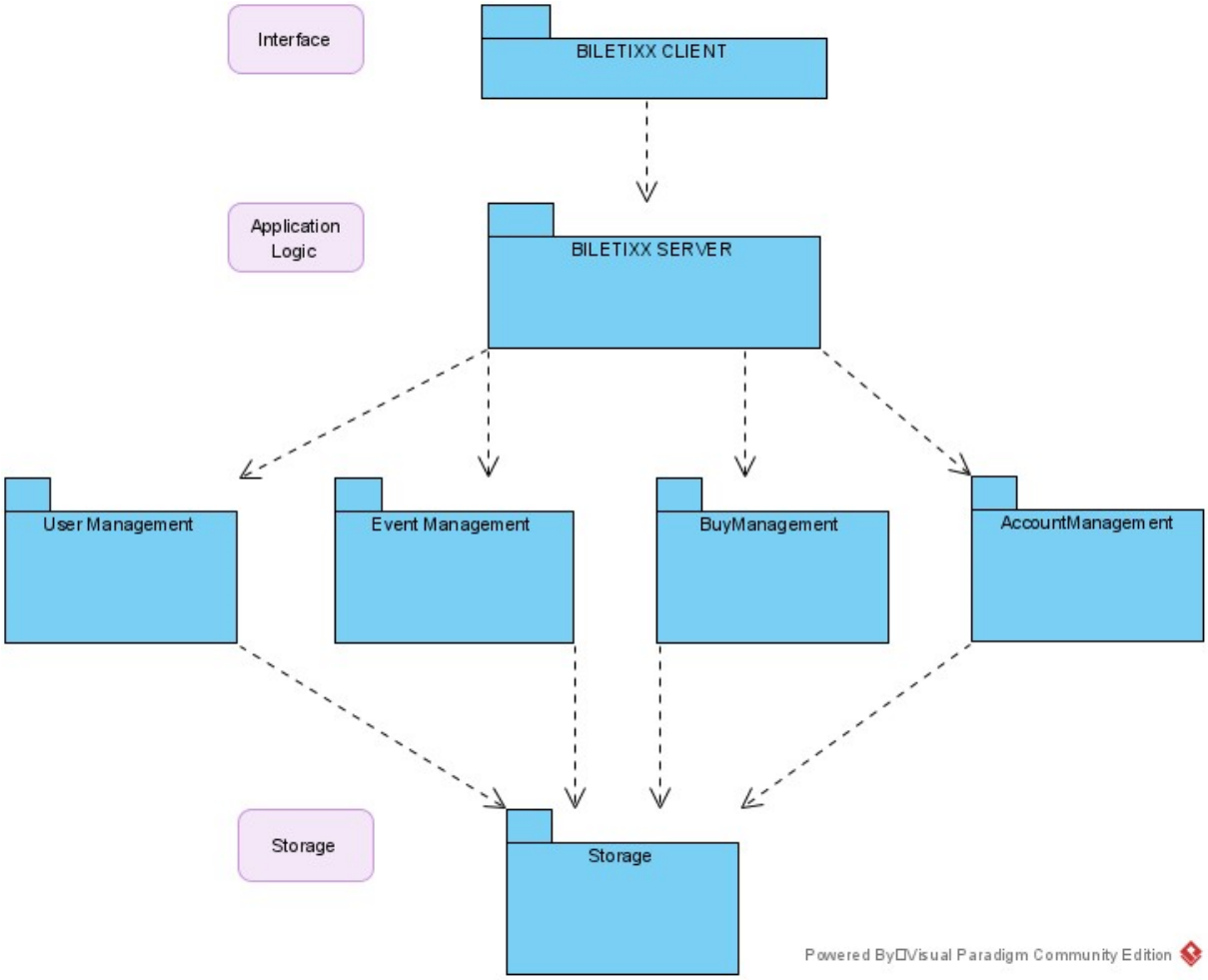
3.1. Overview

The system intends to display the events to user, which are in accordance with their requests and filters. This operation is designed to be user friendlys and it can be done with just a few clicks. In the event side, this part of the system is also user friendly and easy to use. Events are held by EventHolders. Users buy tickets for events which they desired. Eventholders can control event dates, place and pricing. Finally, an admin is always checking the main parts of the system and if necessary, he can change easily.

<Project Name>

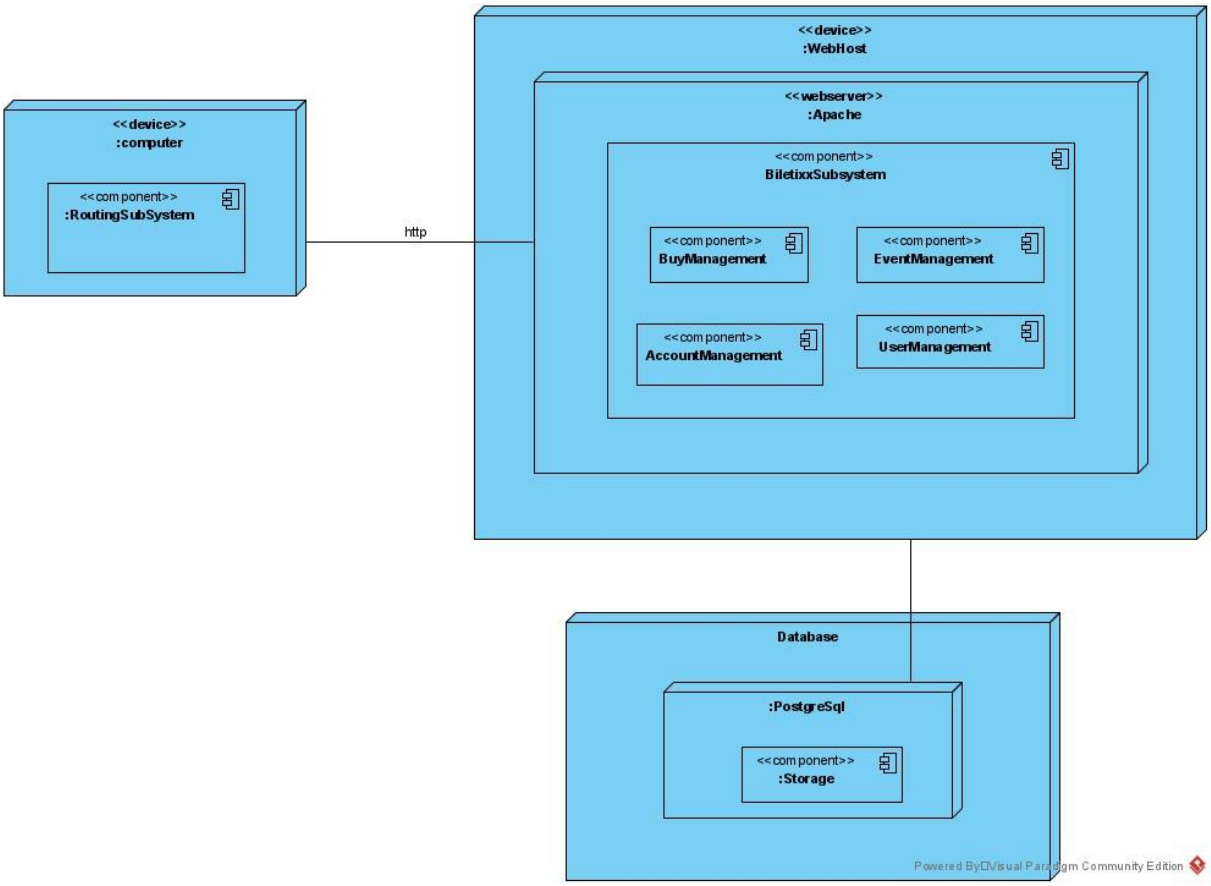
3.2. System Decomposition

Interface: Interface is the website. It will be used by visitor, user, event holder and admin.
User Management: Provides the function which enables admins to manage users.
Event Management: Provides the function which enables admins and event holders to manage events.
Buy Management: Provides the function which enables users to buy tickets and visitors to display events.
Account Management: Provides the function to all users to create and change their account information also login and logout from the website.



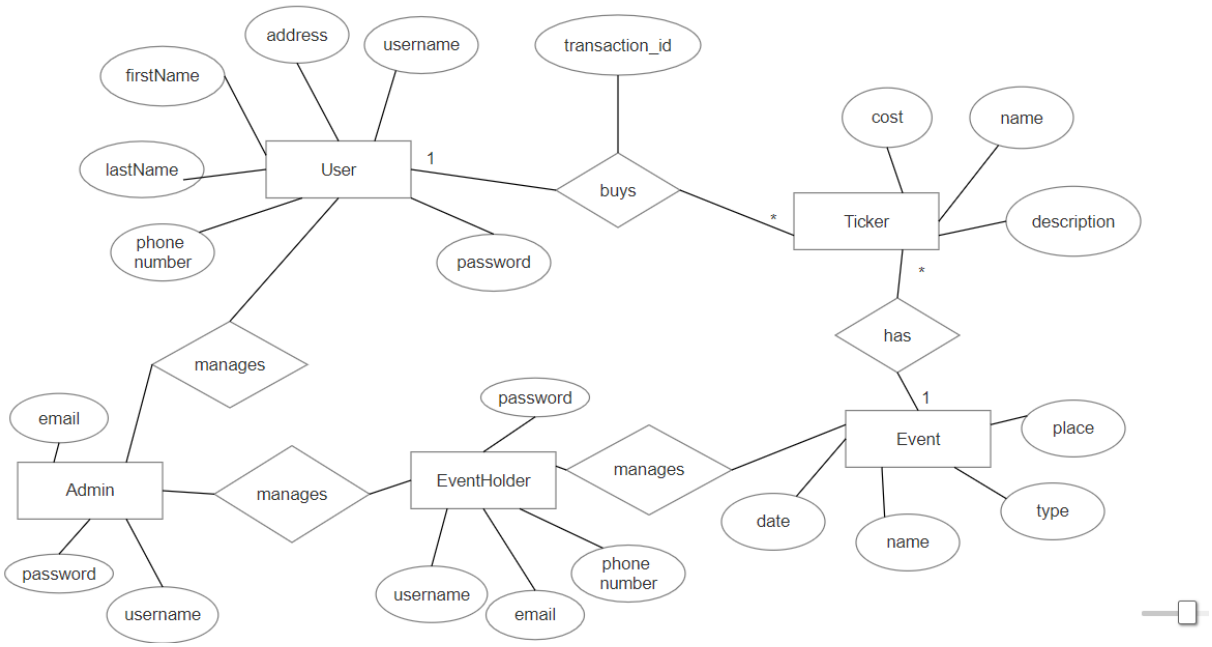
3.3. Hardware Software Mapping

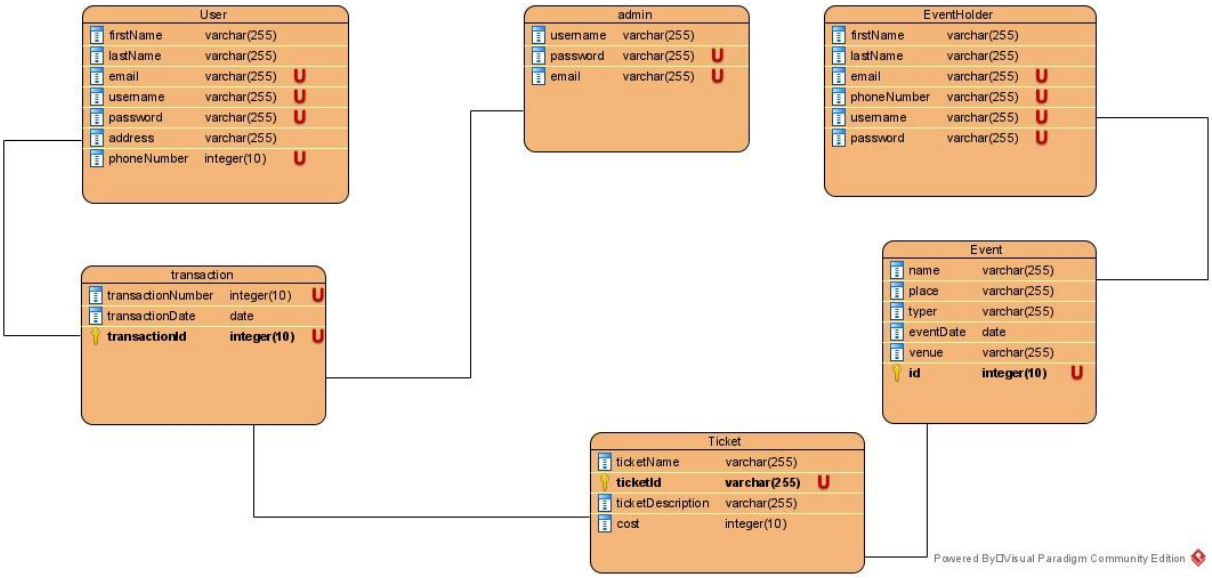
System will use a three-layered architecture. Interface subsystems are mapped in computing hardware and interacts with user. Application Logic subsystems are mapped in Webhost Hardware. Which takes data from Users and delivers them to database. Storage subsystems are mapped in our database.



3.4. Persistent Data Management

We will store the persistent data in 2 ways. The first is File System and the second is Database System. We will store the HTML, Python, CSS and JavaScript data in the file system. We will store the images such as event and ticket related pictures in the file system as well. The other data we want to store are not likely to change so they are also persistent. We will store them in a database because we plan to do searches with those data. The data that we will store in the database are: User information such as the name, surname, username, phone number ,email, password and address. We will also store every data about the events in the database.





3.5. Access Control and Security

Our system is a web based application. In this sytem we have visitor, user, event holder and admin actors. Users can only reach their own personal data and the displayed events in the website, Visitors can see displayed events but they have to create an account before they can buy a ticket. Admins have access to users and event holders information.

Each visitor has to create an account, also each user and event holder has to have an unique password that is stored in the database to ensure security. Passwords on our site cannot be similar to users personal information, must contain 8 characters, cannot be commonly used or be entirely numeric. These passwords can only be reached by the users themselves. Since we used Django, it uses a PBKDF2 algorithm with a SHA256 hash, whish is a password stretching algorithm, as a default.

Our access matrix is shown below. This matrix shows our website functions and what each function can do.

OBJECTS / ACTORS	AccountManagement	BuyManagement	EventManagement	UserManagement
Visitor	createAccount() createEventHolderAccount()	seachEvent() displayEvent()		
User	logIn() logOut() changePassword() forgetPassword() createAddress() displayAddresses() updateAddress() deleteAddress() displayTickets() updatePersonalInformation() displayPersonalInformation()	seachEvent() displayEvent() buyTicket() payTicketCost()		
Event Holder	logIn() logOut() changePassword() forgetPassword()		addEvent deleteEvent	
Admin	logIn() logOut()		deleteEvent	deleteUser deleteEventHolder viewUserInfo () viewEventHolderInfoion()

3.6. Global Software Control

Our system Biletixx is an event driven centralized system that consists of 4 main subsytems, such as:

- User Management subsystem
- Event Management subsystem
- Buy Management subsystem
- Account Management subsystem

-**User management subsytem:** Provides CRUD operations for the user management. It is responsible for Checking the user information and also deleting a spesific user.

-**Event management subsystem**: Provides CRUD operations for the event management. It is responsible for the actions of Adding an event and deleting an event.

-**Buy management subsytem**: Provides CRUD operations for the ticket buying management and it is responsible for the action of searching and buying a ticket.

-**Account management subsytem**: Provides CRUD operations for the user/event holder account management. It is responsible for the actions of login and registreation.

3.7. Boundary Conditions

Startup: Entering the website and logging in.
Shutdown: Logging out and closing the website.
Error Conditions:

- **Login:**
  - Username or password are wrong.
  - Username or password do not match.
  - Account does not exist with the entered credentials.
- **Register:**
  - Username already exists.
  - Mandotary fields are not valid.
- **Buy Management:**
  - Tickets are sold out while trying to buy a ticket.
- **Event Management:**
  - Event has been cancelled.
- **Account Management:**

Visitor tried to buy ticket when they were not registered to the website.

4. Subsystem Services

Biletixx system has multiple subsystem and we have divided into three layers. These are interface, application logic and storage. In the interface layer, there are AdminInterface, VisitorInterface and RegisteredUser Interface subsystems. All interface subsystem has their own homepage and in the RegisteredUser interface there are two homepages determined by the user type. First user type is registered user and the second user type is event holder.

In the application logic layer, there are User management, Account Management,Buying Management and Event Management subsystems.

- Account Management subsystem provides following services:
logIn()
logOut()
createAccount()
createEventHolderAccount()
changePassword()
forgetPassword()
createAddress()
displayAddresses()

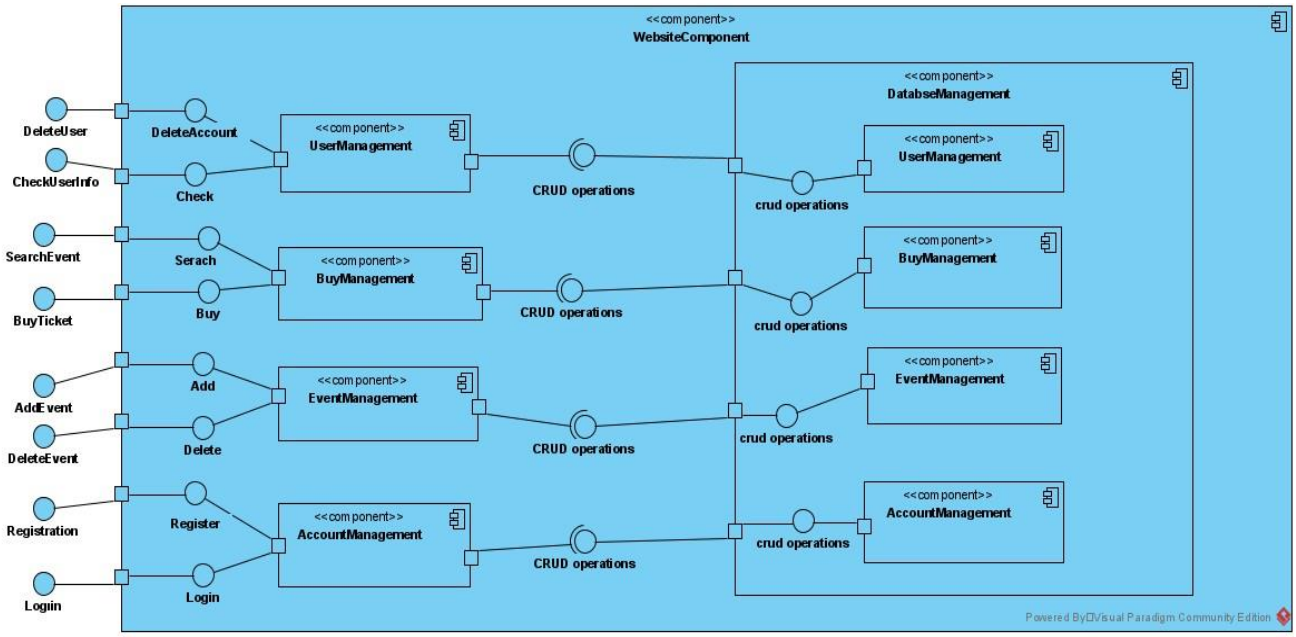
updateAddress()
deleteAddress()
displayTickets()
updatePersonalInformation()
displayPersonalInformation()

-Buying management subsystem provides following services:
seachEvent()
displayEvent()
buyTicket()
payTicketCost()

-Event management subsystem provides following services:
addEvent
deleteEvent

-User management subsystem provides following services to Admin:
deleteUser
deleteEventHolder
viewUserInformation()
viewEventHolderInformation()

Component Diagram:



5. References

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. Biletixx System RAD Document.