MeteoCal

[Design Document]

Mite Ristovski
Dushica Stojkoska
Milica Shulevska

Software Engineering 2 project

Politecnico di Milano
5th School of Engineering

Milan, December 7, 2014

CONTENTS

1 INTRODUCTION

1.1 PURPOSE

This document has the aim to provide the general and specific architecture of the MeteoCal web application, the project of the course Software Engineering 2 at Politecnico di Milano. The document will describe the architectural decisions of the design process, its justifications, also serving as an input for the next phase of the development process of the system.

The intended audience of this document is all the people actively participating in the software engineering course, including professors and tutors. Not only shall the document serve as a reference for the developers to follow the requirements, but it will also serve to the testers to check whether the stated requirements and goals are met or not.

1.2 SCOPE

This is the first iteration of the design process of MeteoCal application. Accordingly the scope of the architecture design is based on the analysis document presented in the analysis phase. This iteration will not provide all the functionalities described as functional requirements in the RASD document.

The MeteoCal architecture will be designed considering the following functionalities:

- **Users**

  MeteoCal will manage registering, log in/out of users, creating events, accepting invitations for the events.
- **Events**

  MeteoCal will manage the creation, update and deletion of an event by a user, and manage invitations sent to participate in an event.
- **Calendar**

  MeteoCal will manage the calendar of each user, including its upcoming events. The calendar can be imported or exported.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

Keyword	Definition
User	All the registered users in the system.
Organizer	A user that has created an event.
Event	A planned public or social occasion, created by a user.
Invited users	Users that have been invited to an event by the organizer
Participants	Invited users that have accepted the invitation.

Table 1: Definitions

1.3.2 Acronyms and abbreviations

Acronym/Abbreviation	Definitions
XML	Extensible Markup Language
RASD	Requirements Analysis and Specification Document
NFR	Non-Functional Requirements
FR	Functional Requirements
QA	Quality Attributes
G	Goal
DBMS	Data Base Management System
AS	Application Server
JEE	Java Enterprise Edition
XHTML	Extensible Hypertext Markup Language
JSF	Java Server Faces
EJB	Enterprise JavaBeans
JDBC	Java Database Connectivity Application Programming Interface

Table 2: Acronyms and abbreviations

1.4 References

1. Requirements Analysis and Specification Document – Mite Ristovski, Dushica Stojkoska, Milica Shulevska
2. IEEE Recommended Practice for Software Engineering Requirements Specification (http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?reload=true&punumber=5841)

1.5 Overview

This document describes the architecture of MeteoCal, general and specific details. It shows as well the architectural decisions of the design process and its justifications. The design was developed using the top-down approach, so the document contains general description of the system at first and specific details of each component in the end.

This document is organized as follows:

1. **Introduction**

   This section describes the purpose of the project and the general functional requirements. It introduces the reader definitions, acronyms and abbreviations used in this document.
2. **Design overview**

   This section provides a general description of the MeteoCal software system, including the functionality and matters related to the overall system and its design.
3. **Design Considerations**

   This section contains all the design assumptions and constraints of the MeteoCal software system.
4. **Software Architecture**

   This section describes the general architecture, basic structure and interactions of the main subsystems.
5. **Detailed Systems Design**

   This section provides through different architectural views, the detailed architecture of the system, specifying all the components of the system with greater detail.
6. **Appendices**

   This section provides the supporting information and all the additional material.

2 DESIGN OVERVIEW

This section provides general description of the MeteoCal software system, including the functionality and matters related to the overall system and its design.

2.1 DESIGN CONTEXT

The design context specifies the limits for the system design, considering the functional and technological context.

2.1.1 Functionalities

The functionalities identified in the RASD are grouped by the following functional areas:

- Managing Users
- Managing Events
- Managing Calendar

2.1.1.1 Managing users

- Functional Requirements

  [FR1] Register to the system.

  [FR2] Login/Logout.

  [FR3] Consult user profiles.

  [FR4] Consult event information.

  [FR5] Accept or decline an event invitation..

  [FR6] Receive notification about the weather.

  [FR7] The organizer can create an event.

  [FR8] The organizer can update an event.

  [FR9] The organizer can delete an event.

  [FR10] The organizer can invite people to an event.

- Non-functional Requirements

  [NFR1] The user password must be stored securely.

  [NFR2] The system must manage high number of users.

  [NFR3] An organizer can manage only one event at a time.

2.1.1.2 Managing events

- Functional Requirements

  [FR11] Consult events.

  [FR12] Events have only one organizer.

- Non-functional Requirements

  [NFR4] Only invited users can attend an event.

2.1.1.3 Managing calendars

- Functional Requirements

  [FR13] Consult calendar.

  [FR14] Consult public calendars of other users.

  [FR15] User can make his/her own calendar public.

  [FR16] User can import calendar.

  [FR17] User can export calendar.

- Non-functional requirements

  [NFR5] Each user can have only one calendar.

  [NFR6] The system must manage high number of calendars.

2.1.2 System Technologies

The MeteoCal web application will be developed using the three tier distributed architectural style. It is composed of client tier, web tier, business logic tier, and persistence tier. Each tier requires usage of specific technologies as shown below:

Client tier – Consists of web browsers . Web browsers tend to talk to the 'web components' on the Server Tier. All major web browsers are considered.

Web tier – It is part of the Java EE server. The web tier consists of components that handle the interaction between clients and the business tier. The web tier will contain the dynamic XHTML web pages, and managed beans. Using JSF the page views will be rendered.

Business tier – It is also part of the Java EE server. The business tier consists of components that provide the business logic for an application. The core functionalities of our application will come from this tier. This part will be implemented using EJB components.
The language that will be used in system implementation is Java Enterprise Edition 7, which supports distributed, multi-tier system based on components. Enterprise Java Beans (EJB) 3.2 is a server-side component architecture that encapsulates the business logic of the system. Glassfish Server 4.1 is used as an application server and it supports all the platform Java EE 7 features.

Persistence tier – To provide database management, MySQL Server 5.6 is used. JDBC will be used for accessing the database by components on the business tier.

System design uses the top-down approach. After identification of four main tiers, the system is decomposed into components that encapsulate related functionalities. Therefore each component is responsible for certain functionalities and interacts with others.

2.1.3 Overall Design

In this section the general design schemas of MeteoCal are presented specifying the basic relations between packages, use cases and users.

2.1.3.1 General Package Design

As the system is distributed in a three-layer fashion and each tier contains a set of functionalities which satisfy the correspondent requirements, we find a mapping between use cases and package design.

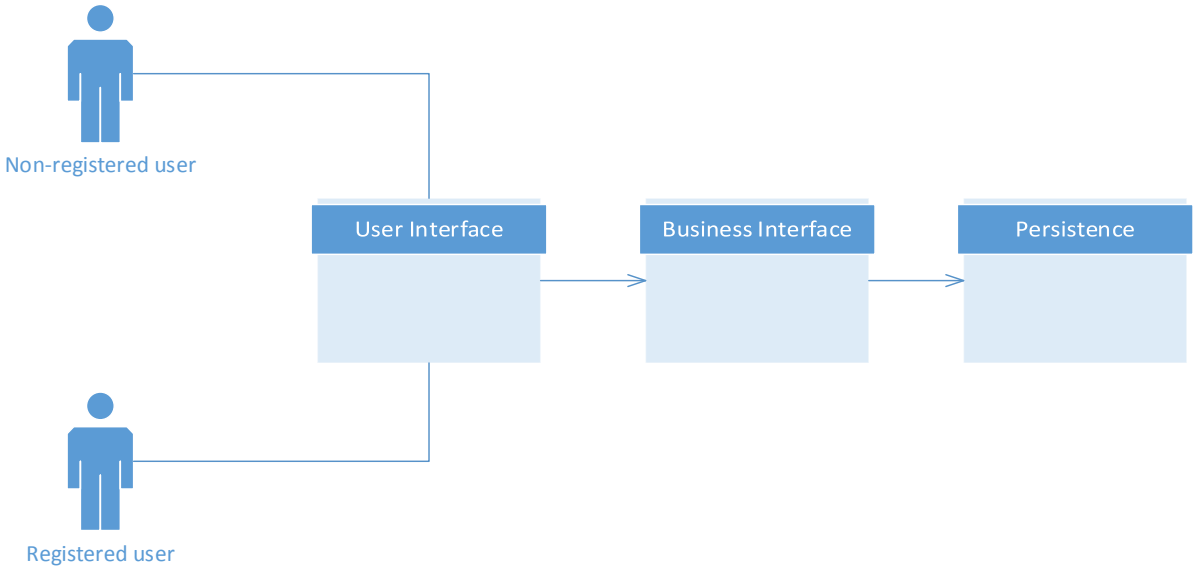
In the diagram we can identify three packages:

- **User UI**
  This package contains the user interfaces. It interacts with the user and it is responsible for obtaining the UI requests, sending them to the Business Logic package and retrieving the data back for displaying it to the user.
- **Business Logic**
  This package contains business logic components. It is in charge of receiving User UI package requests, processing them, accessing the Persistence package when needed and sending a response accordingly.

- **Persistence**

  This package is responsible for managing the data request from the business logic package.

  Non-registered user and registered user access the User UI package directly, where they submit requests to accomplish their tasks.



Package Diagram 1: Basic Package

2.1.3.2 Detailed Package Design

Given the functional requirements identified we can encapsulate them within specific components in the package diagram as follows:

User UI

The set of this sub-package is responsible for capturing the user actions and forwarding corresponding requests to the respective Business Logic set of sub packages.

- *User Managed Bean:* This package implements [FR1], [FR2], [FR3], [FR4], [FR5], [FR6], [FR7], [FR8], [FR9], [FR10].
- *Event Managed Bean:* This package implements [FR11], [FR12].
- *Calendar Managed Bean:* This package implements [FR13], [FR14], [FR15], [FR16], [FR17].

Business logic

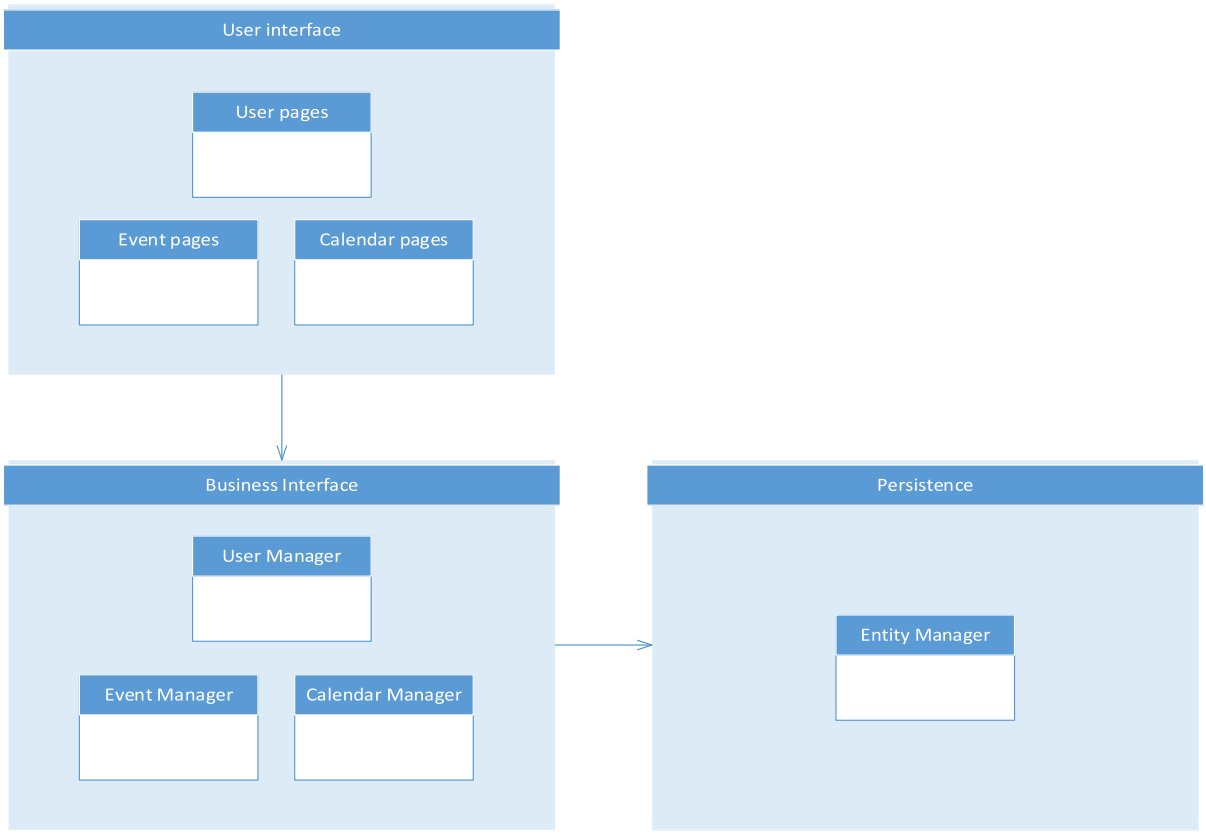
The set of this sub-package is responsible for receiving requests from the User UI package, processing them and send a response back. These packages may access the Persistence package.

- *User EJBs:* This package implements [FR1], [FR2], [FR3], [FR4], [FR5], [FR6], [FR7], [FR8], [FR9], [FR10].
- *Event EJBs:* This package implements [FR11], [FR12].
- *Calendar EJBs:* This package implements [FR13], [FR14], [FR15], [FR16], [FR17].

Persistence

The set of this sub package contains the system's data structure and stored information in the database. It is responsible for receiving requests from Business Logic package, processing them, and returning answers back.

- Persistence entities: This package implements [FR1]-[FR17].



Package Diagram 2: Detailed Package

3 DESIGN CONSIDERATIONS

This section specifies the design assumptions and constraints of the MeteoCal software system.

3.1 ASSUMPTIONS

- Imame assumptions?

3.2 DEPENDENCIES

Dependency	Impact
The Java Virtual Machine is already installed on the operating system.	MeteoCal runs only on operating systems which support the Java Virtual Machine platform.
The supported browsers will be Internet Explorer, Firefox, Chrome or Safari.	MeteoCal works on browsers which support latest web standards; elsewhere the UI experience will be affected.
JEE7 AS is required on the server side.	MeteoCal cannot operate if there is no AS that supports JEE7 platform.

3.3 GENERAL CONSTRAINTS

Assumption	Action
Memory	At least 2 GB
Database server	MySQL
Network	Internet access, HTTP protocol
Security	The system is controlled for each type of user. SSL is not supported.
Hard disk space	At least 40 GB

3.4 PERFORMANCE REQUIREMENTS

3.4.1 Standard Compliance

The software does not have to meet any standard compliance.

3.4.2 Reliability

For ensuring the integrity of users and events, it is necessary to back up the database periodically.

3.4.3 Availability

For guaranteeing the availability of the system, an application server is used. However, for a complete availability of the system, it is necessary to have redundancy in the instances of application. For the current release of the software product we will assume that all the tiers run on the same physical server.

3.4.4 Security

The software product does not support SSL in AS. It supports the hashing of the user password according to sha1 algorithm in the database. It supports authorization according JAAS. It is recommended to implement SSL in future releases.

3.4.5 Maintainability

The architectural style and the component definition ensured to low coupling and high cohesion of the software product. Therefore, the system can be modified and improved easily.

3.4.6 Portability

The software product is developed in Java and can be installed on any operating system which supports Java Virtual Machine and its dependent components.

4 SOFTWARE ARCHITECTURE

This section describes the general architecture, basic structure and interactions of the main subsystems. Furthermore, the section explains the use of JEE technology in the MeteoCal web application.

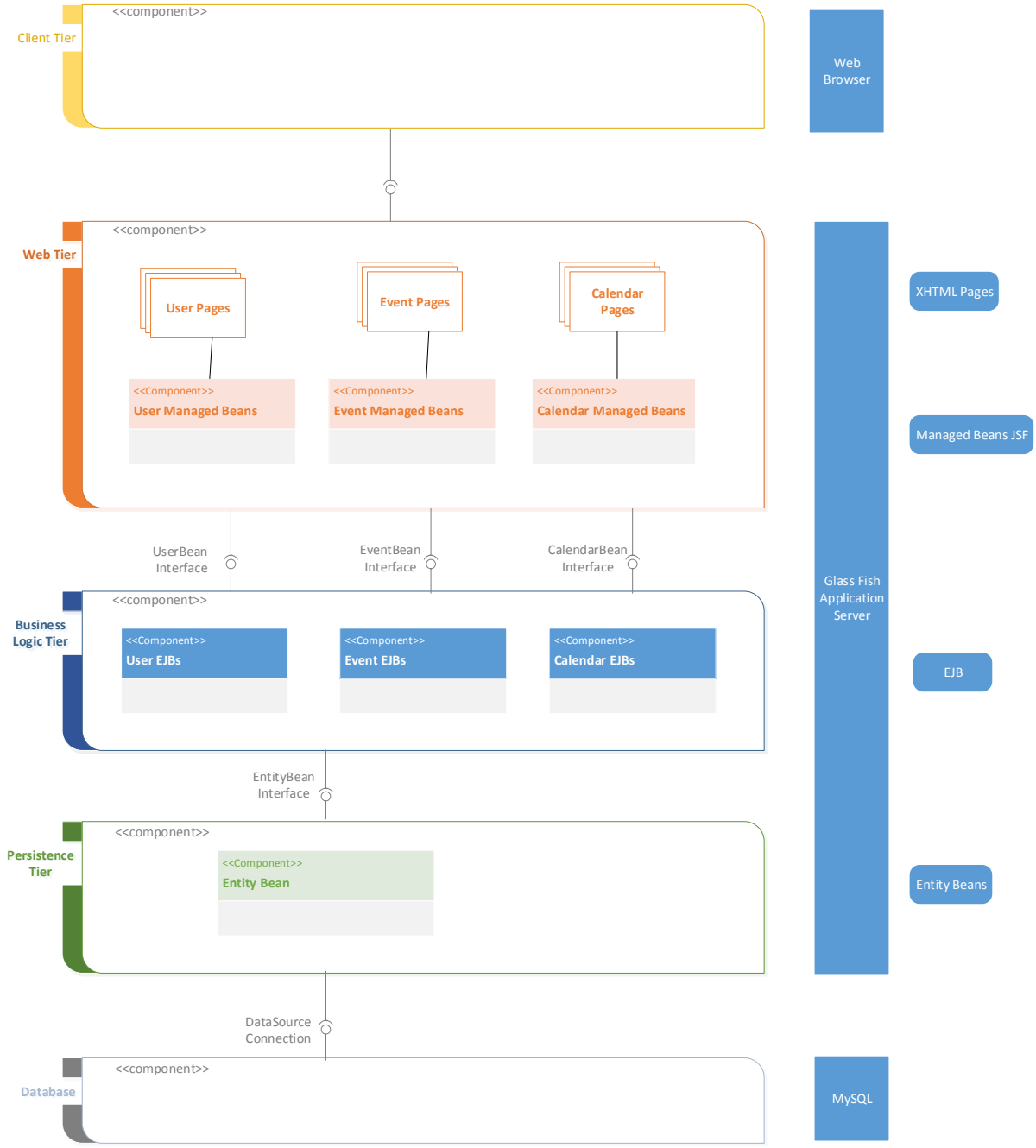
As stated earlier, MeteoCal application will be developed using the three-tier architecture, where each component of the architecture is treated as independent module on separate platforms.

Three-tier architecture of MeteoCal is composed of the following components:

- Web Component represented by Web Tier
- Business Logic Component represented by Business Logic Tier
- Persistence Component represented by Persistence Tier

Furthermore, Data Model is represented by Database.

4.1 CONCEPTUAL DESIGN



Architecture Design 1: General architecture

The above constructed architecture is meant to fulfill the demands of a high availability web application. Web component is responsible for the user interface; the component calls methods in the Business Layer component, where all the logic concerning functionality is located, in order to perform operations requested by the user and shows the result by constructing XHTML pages.

The Persistence component communicates with database and performs create, read, update and delete operations requested by the above mentioned components.

4.1.1 Client tier

The Client Tier consists of application clients with which requests to the server are made. Clients are located on a different machine from the server. After processing the requests made by the clients, the server responds back. Java EE clients can be a web browser, a standalone application, or other servers, and, as stated before, they run on a different machine from the Java EE server.

4.1.2 Web tier

The Web Tier is composed of XHTML pages which are used for displaying the information to the user. In this tier are located the managed beans as well.

The Web Tier is responsible for receiving the requests of the user, where the managed beans are used for listening the events and for displaying data regarding the user requests. In order to retrieve the information, managed beans have to interact with the Enterprise JavaBeans (EJBs) located in the Business Logic Tier.

4.1.3 Business Logic Tier

The Business Logic Tier interacts with Web Tier and Persistence Tier. It is responsible for the logic behind the MeteoCal application and it is composed of the components called EJB Beans.

4.1.4 Persistence Tier

The persistence tier is composed of entity beans responsible for communication with the MeteoCal database. Entities are used for saving, modifying and deleting the system data. Components are in charge of the entity management for the data located in the system.

4.1.5 Database

The database is composed of the tables based on our assumptions and needs of the project.

4.2 SYSTEM SPECIFICATION

Following table displays technologies which will be used in the implementation phase:

Component Name	Technology
Client Tier	Firefox 34.0, IE 11, Chrome 39.0, Safari 8.0.
Web Tier	XHTML in integration with Java Server Faces Technology (Facelets for front-end)
Business Logic Tier	JEE with Glassfish Server 4.1
Persistence Tier, Database Tier	MySQL 5.6

5 DETAILED SOFTWARE DESIGN

5.1 IMPLEMENTATION MODULES/COMPONENTS

MeteoCal project is composed of 3 main components:

- Web component
- Business logic component
- Persistence component

These 3 components will be implemented during the implementation phase.

5.1.1 Web Component

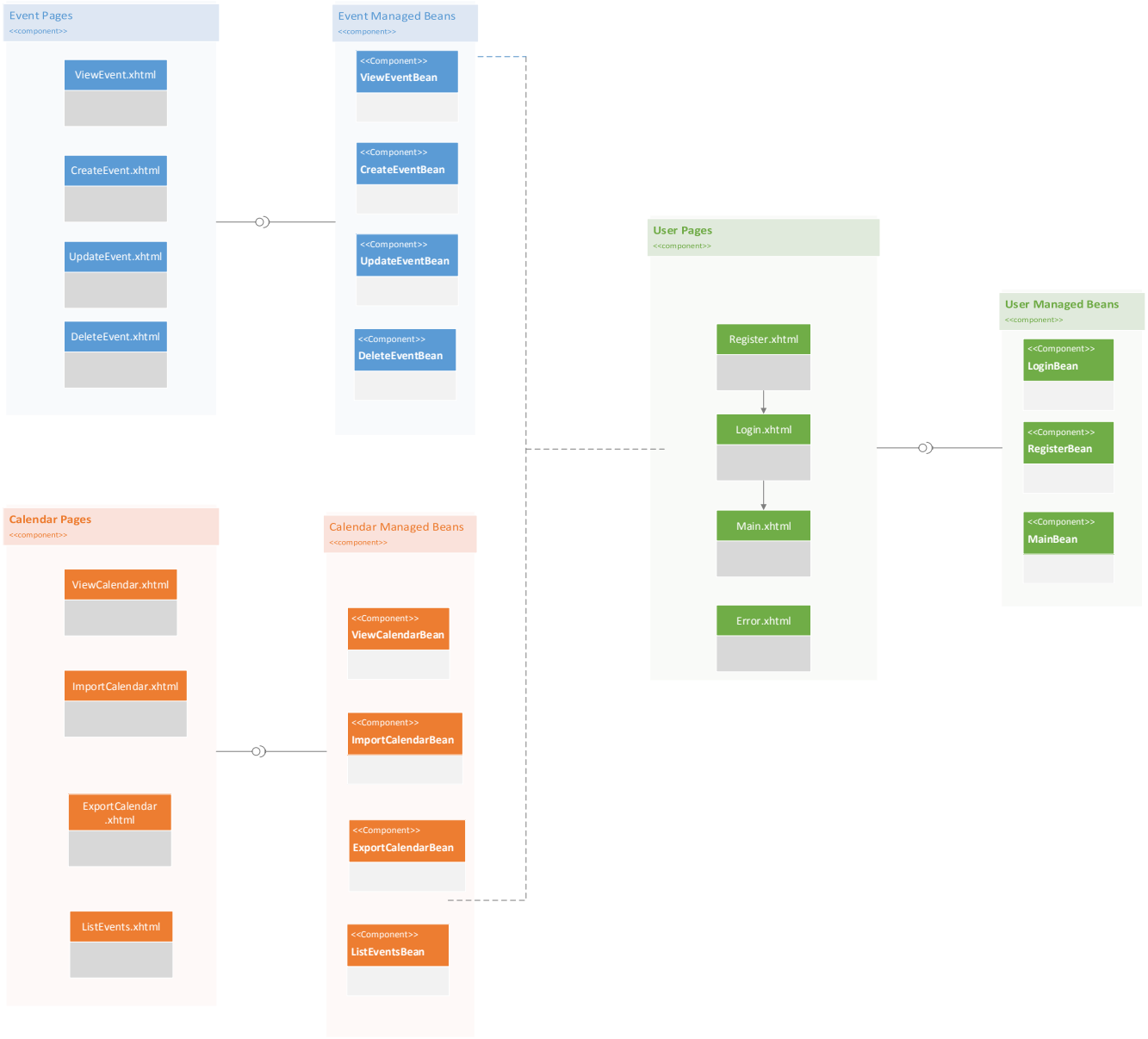
The following general diagram shows the subcomponents and communication between them. In the project, we have identified 3 main subcomponents and their related Managed Beans:

- User Pages
- EventPages
- Calendar Pages

Events in the pages generated by the users are managed by Managed Beans which are composed of 3 sections:

- UserManagedBeans
- EventManagedBeans
- CalendarManagedBeans

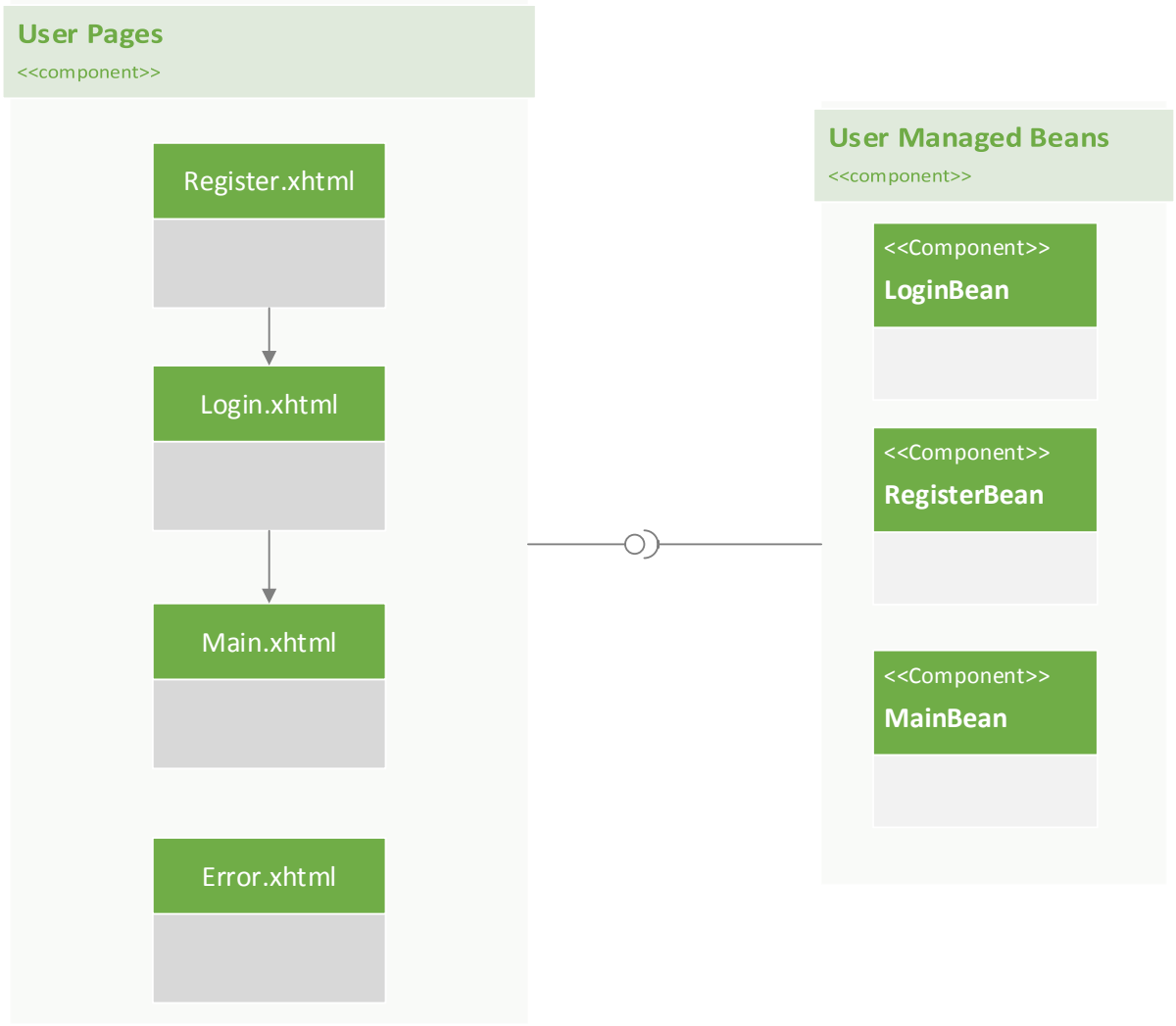
Above mentioned beans represent the conceptual idea of Managed beans, as during the implementation more beans will be needed.



Component Design 1: Web Tier Components

5.1.1.1 User Pages and Managed Beans

Pages and beans in this section are responsible for everything related to users.



Component Design 2: User Pages and Managed Beans

Register	
Classification	Register.xhtml
Definition	User interface for user registration
Responsibilities	- Load registration form

Login	
Classification	Login.xhtml
Definition	User interface for user login
Responsibilities	- Load login form

Main	
Classification	Main.xhtml
Definition	User interface for displaying a homepage for a registered user
Responsibilities	- Display user homepage

Error	
Classification	Error.xhtml
Definition	User interface for displaying error messages
Responsibilities	- Display error message based on user input

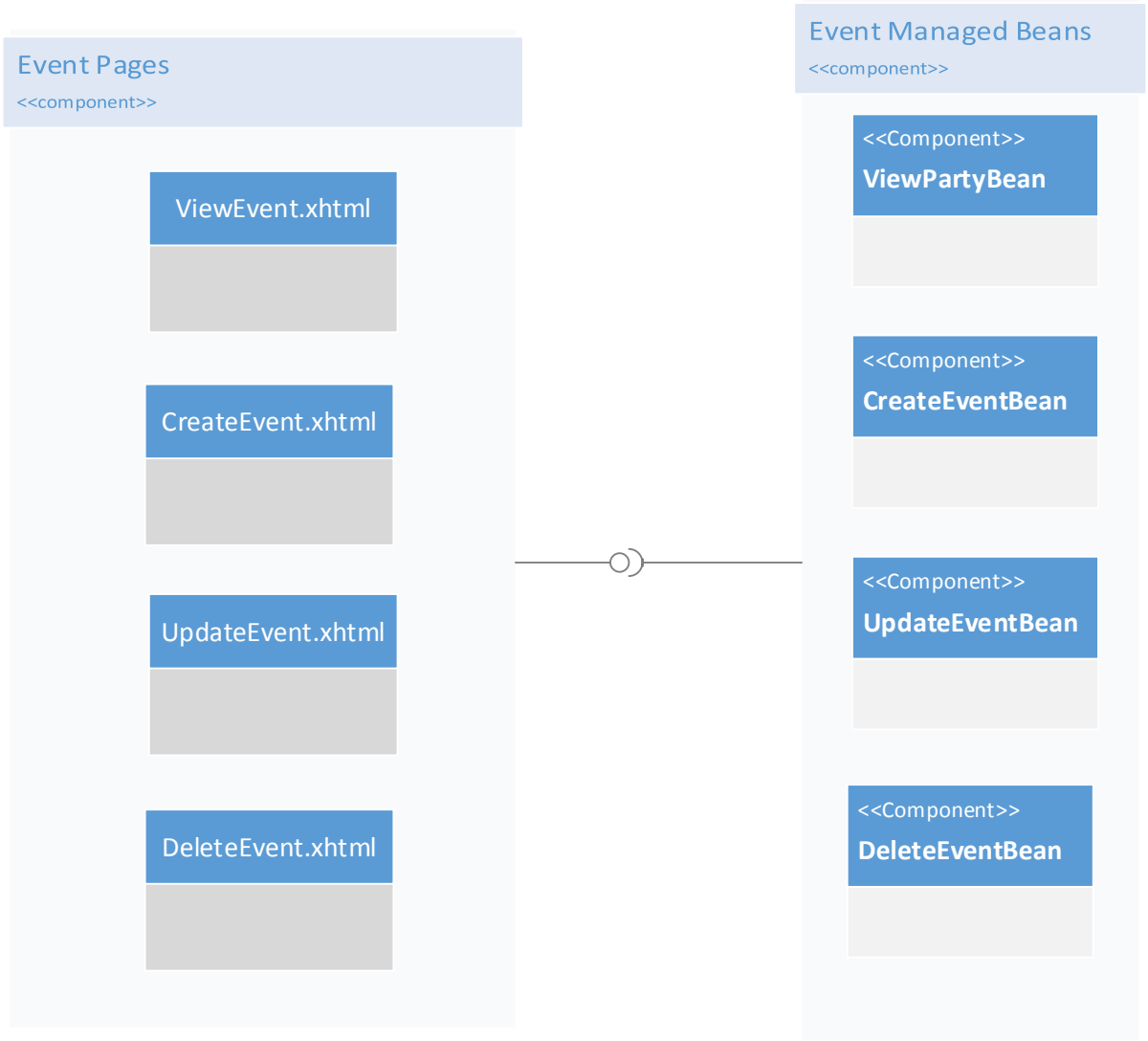
RegisterBean	
Classification	RegisterBean
Definition	Managed bean for user registration
Responsibilities	- Display registration form - Check mandatory data - Redirect to Error.xhtml or Main.xhtml

LoginBean	
Classification	LoginBean
Definition	Managed bean for user login
Responsibilities	- Display login form - Check login credentials - Redirect to Error.xhtml or Main.xhtml

MainBean	
Classification	MainBean
Definition	Managed bean for displaying a homepage for a registered user
Responsibilities	- Load user homepage

5.1.1.2 Event pages and Managed Beans

Pages and beans in this section are responsible for everything related to events.



Component Design 3: Events Pages and Managed Beans

ViewEvent	
Classification	viewEvent.xhtml
Definition	User interface for display of the event
Responsibilities	- Display information about an event

CreateEvent	
Classification	createEvent.xhtml
Definition	User interface for creation of the event
Responsibilities	- Display the form for creation of the event

	- Provide functionalities for creation of the party - Check mandatory data

UpdateEvent	
Classification	updateEvent.xhtml
Definition	User interface for update of the event
Responsibilities	- Display the form for update of the event - Check mandatory data

DeleteEvent	
Classification	deleteEvent.xhtml
Definition	User interface for deletion of the event
Responsibilities	- Display a confirmation message

ViewEventBean	
Classification	viewEventBean
Definition	Managed bean for viewing the event
Responsibilities	- Provide functionalities for loading the details of the event

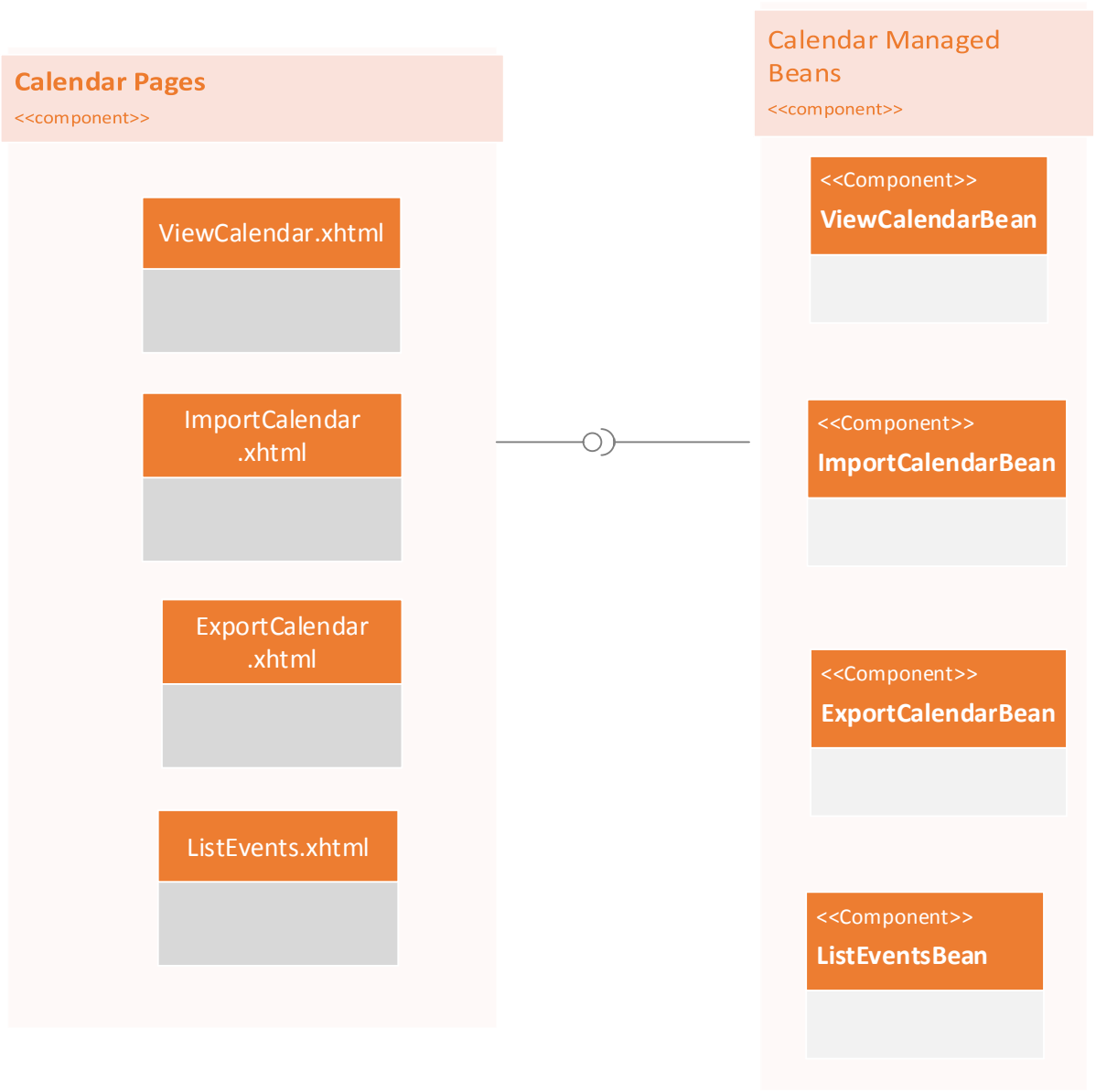
CreateEventBean	
Classification	createEventBean
Definition	Managed bean for creating the event
Responsibilities	- Load the functionalities for the creating of the new event

UpdateEventBean	
Classification	updateEventBean
Definition	Managed bean for updating an event

Responsibilities	- Load the functionalities for updating an event

DeleteEventBean	
Classification	deleteEventBean
Definition	Managed bean for deleting an event
Responsibilities	- Display a confirmation message

5.1.1.3 Calendar pages and Managed Beans



Component Design 4: Calendar Pages and Managed Beans

ViewCalendar	
Classification	viewCalendar.xhtml
Definition	User interface for display of the calendar
Responsibilities	- Display information about the calendar of the user

ImportCalendar	
Classification	importCalendar.xhtml
Definition	User interface for importing a calendar
Responsibilities	- Display the form for importing a calendar

ExportCalendar	
Classification	exportCalendar.xhtml
Definition	User interface for exporting a calendar
Responsibilities	- Display the form for exporting a calendar

ListEvents	
Classification	listEvents.xhtml
Definition	User interface for the events in a user's calendar
Responsibilities	- Display the list of all the events in the person's calendar

ViewCalendarBean	
Classification	viewCalendarBean
Definition	Managed bean for viewing the calendar
Responsibilities	- Check the privacy - Provide functionalities for displaying of the calendar

ImportCalendarBean	
Classification	importCalendarBean

Definition	Managed bean for importing a calendar
Responsibilities	- Provide functionalities for importing a calendar

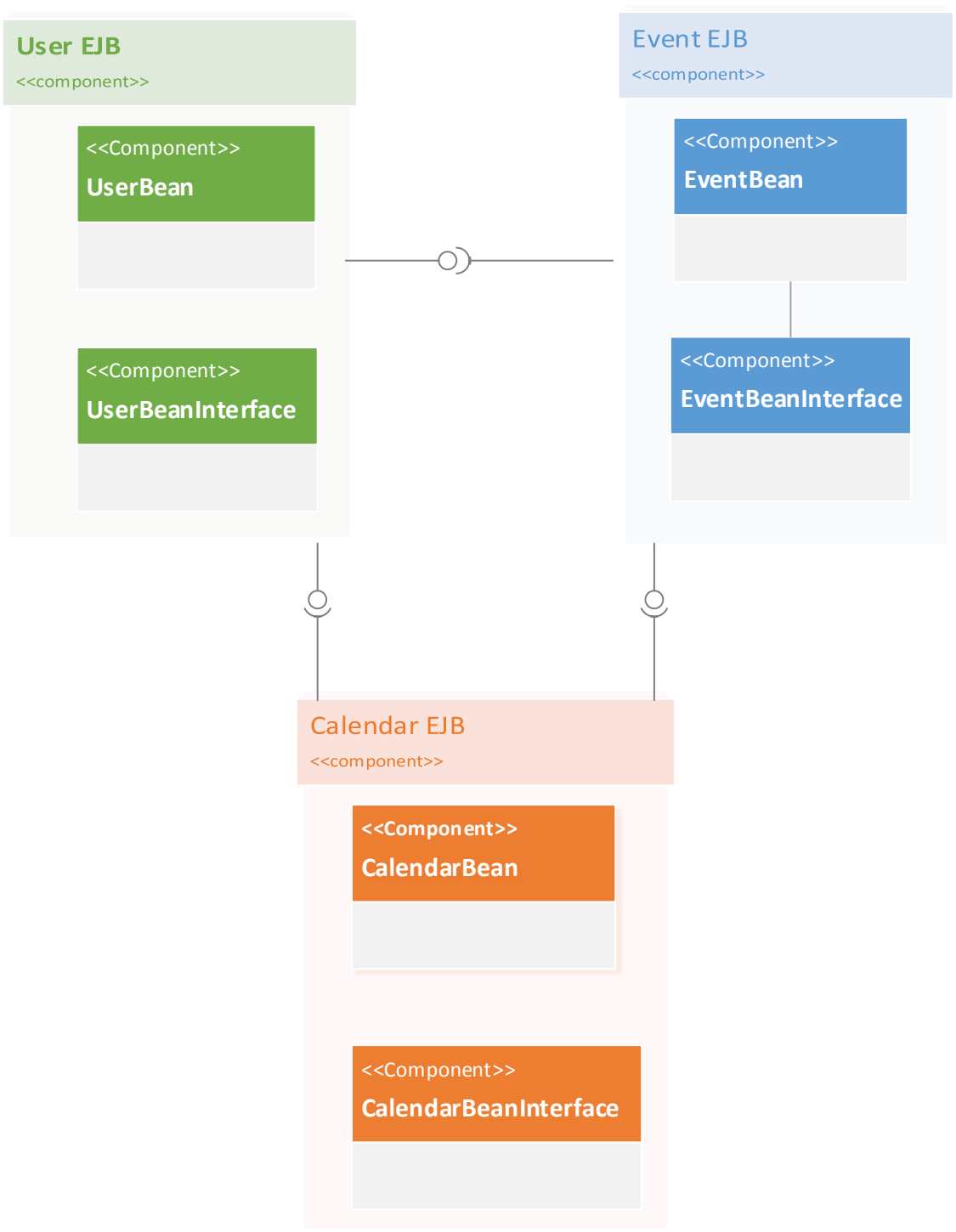
ExportCalendarBean	
Classification	exportCalendarBean
Definition	Managed bean for exporting a calendar
Responsibilities	- Provide functionalities for exporting a calendar

ListEventsBean	
Classification	listEventsBean
Definition	Managed bean for displaying the list of events
Responsibilities	- Check the privacy of the calendar - Provide functionalities for displaying list of events

5.1.2 Business Logic Component

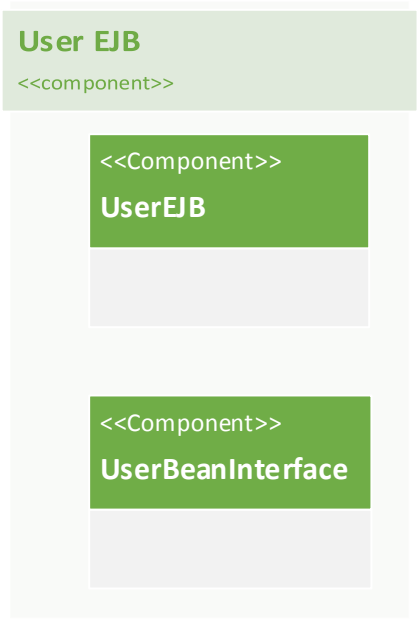
The following diagram shows the planned subcomponents and their communication. In the project we have defined 3 main subcomponents.

- User EJB
- Event EJB
- Calendar EJB



Component Design 5: Business Logic Components

5.1.2.1 User EJB

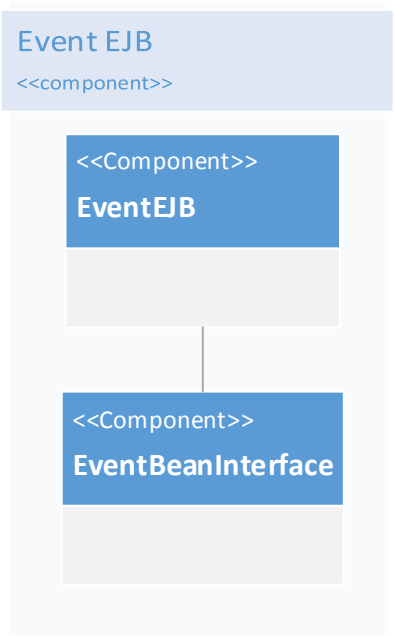


Component Design 6: User EJB

UserBean	
Classification	UserBean
Definition	Bean in charge for all functionalities related to users
Responsibilities	- Login user - Logout user - Register a new user

UserBeanInterface	
Classification	UserBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	- Communicate with CalendarBeanInterface and EventBeanInterface

5.1.2.2 Event EJB

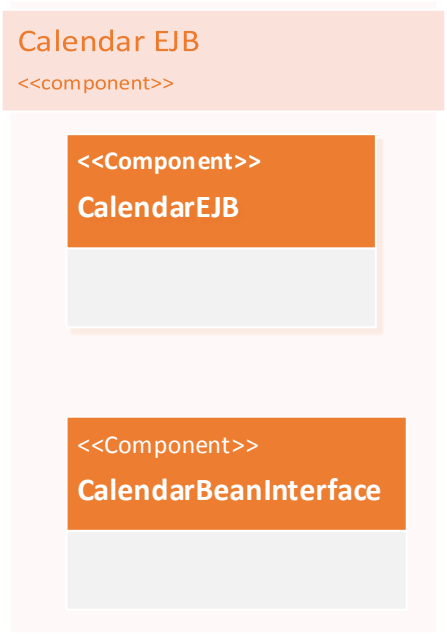


Component Design 7: Event EJB

EventBean	
Classification	EventBean
Definition	Bean in charge for all functionalities related to events.
Responsibilities	- Load list of events - Create event - Update event - Delete event

EventBeanInterface	
Classification	EventBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	- Communicate with UserBeanInterface and CalendarBeanInterface

5.1.2.3 Calendar EJB

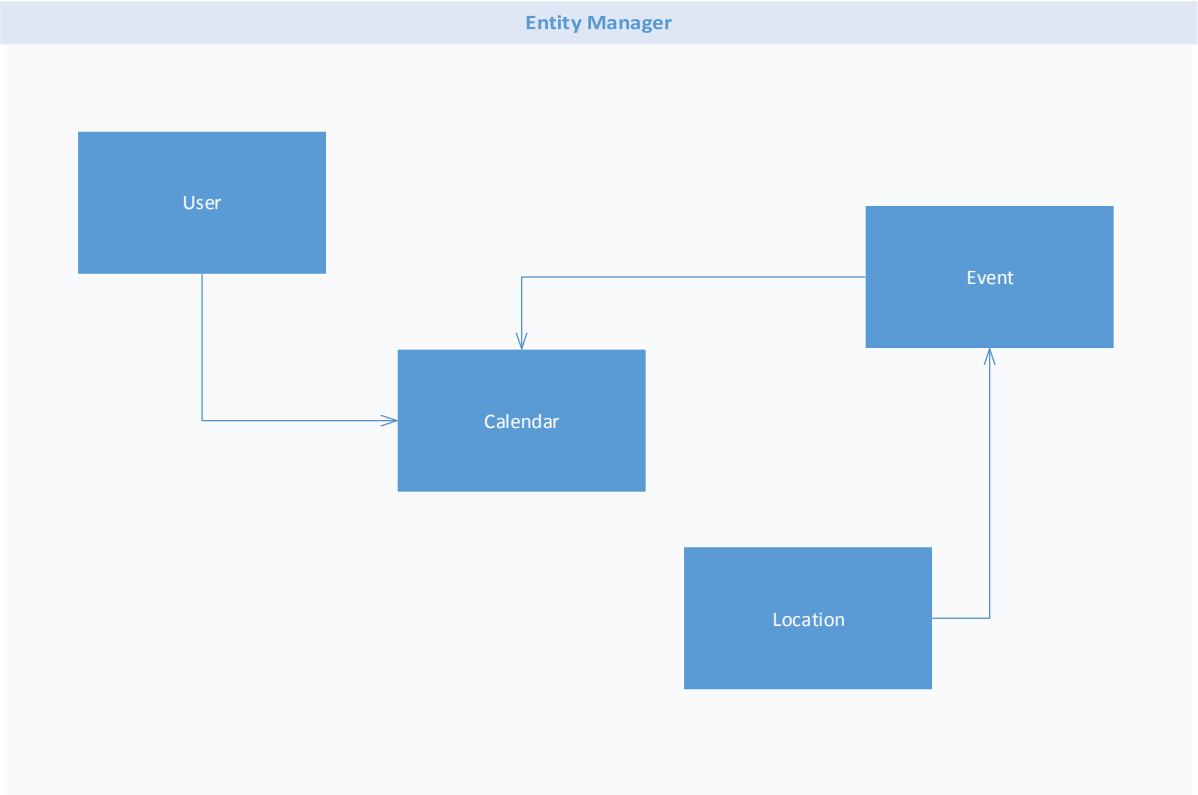


Component Design 8: Calendar EJB

CalendarBean	
Classification	CalendarBean
Definition	Bean in charge for all functionalities related to calendars
Responsibilities	- Load list of events

CalendarBeanInterface	
Classification	CalendarBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	- Communicate with UserBeanInterface and EventBeanInterface

5.1.3 Persistence Component



Component Design 9: Persistence Component

In the Persistence component diagram are shown all the entities in the application. All of these entities represent a high level object view of the tables in the database, and likewise we are providing an object oriented model for the database, that will be used in our application. So, all of the entities have inside all the attributes that are specified in the associated table in the database.

In the following table there is a short description of the responsibilities of each of the entities from above:

Entity	Responsibilities
User	Represents the user and has all the user attributes specified in the database design. Organizer is also a user which can create, update or delete an event. It is connected with the calendar entity.
Event	Represent the event and has all the event attributes specified in the database design. Calendar entity connects with event entity.

	Event entity is connected with location entity.
Calendar	Represents the calendar and has all the calendar attributes specified in the database design. It is connected with event entity and user entity.
Location	Represents the location and has all the location attributes specified in the database design. It is connected with event entity.

5.2 DATABASE MODEL

In the following section we will provide a detail description of the database by providing its both views: the conceptual and the logical design diagrams.

5.2.1 Conceptual Design

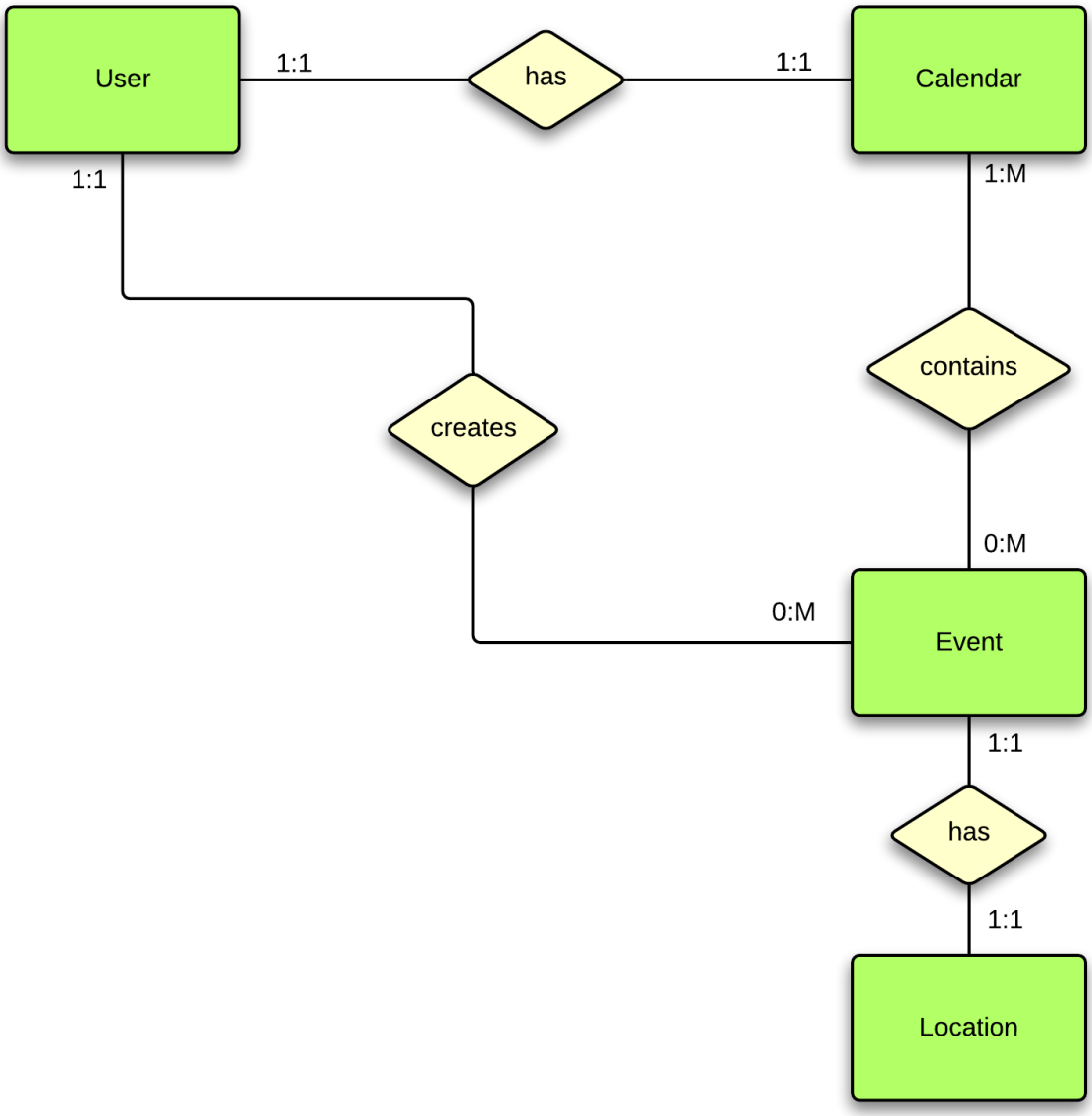
The conceptual design of the database is represented with the Entity Relationship Diagram (ER) given bellow. In the following text we will provide a description of the elements in the diagram.

Let us clarify that:

1:1 - means one to one relation

1:M - means one to many relation

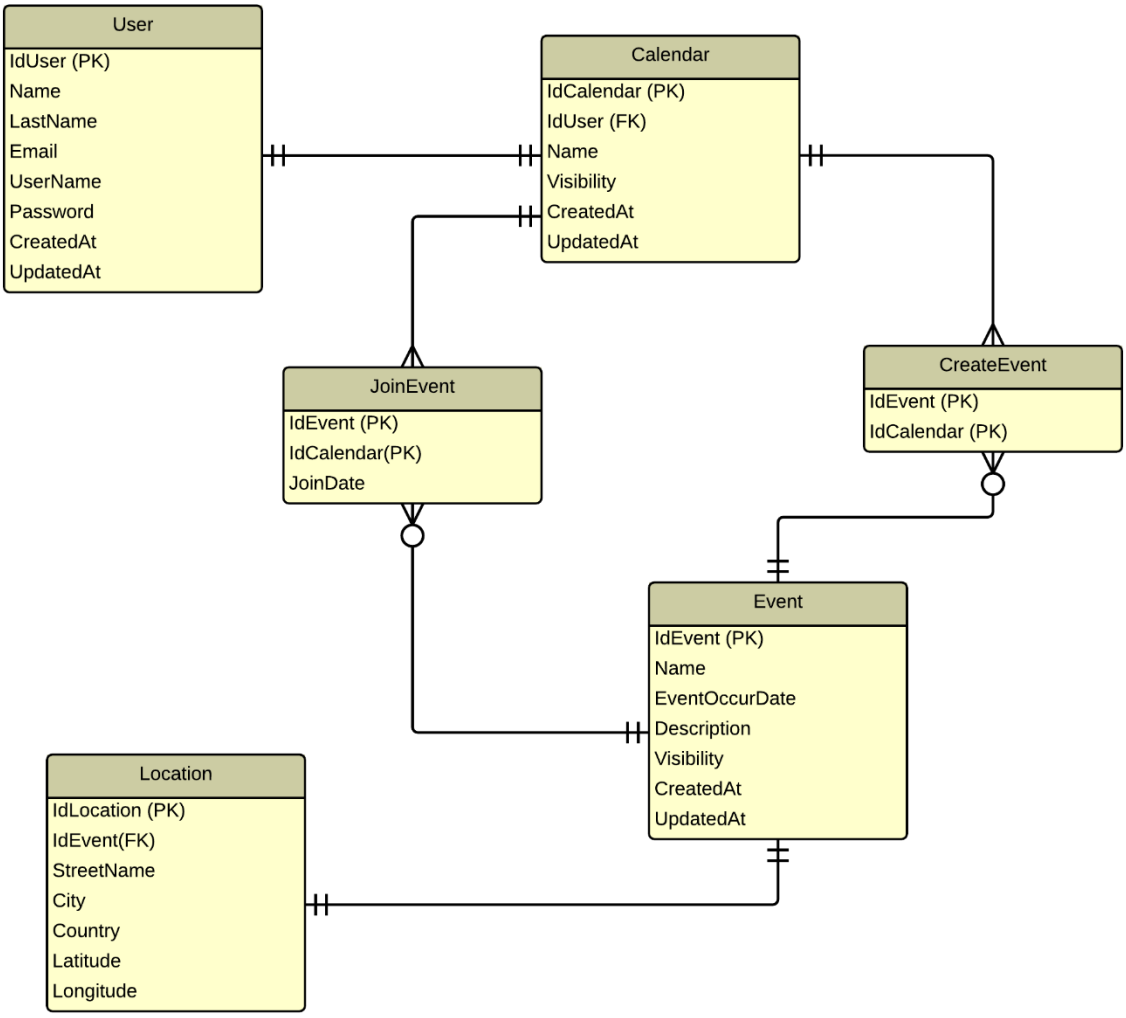
M:M - means many to many relation



Database design 1: Conceptual design

5.2.2 Logical Design

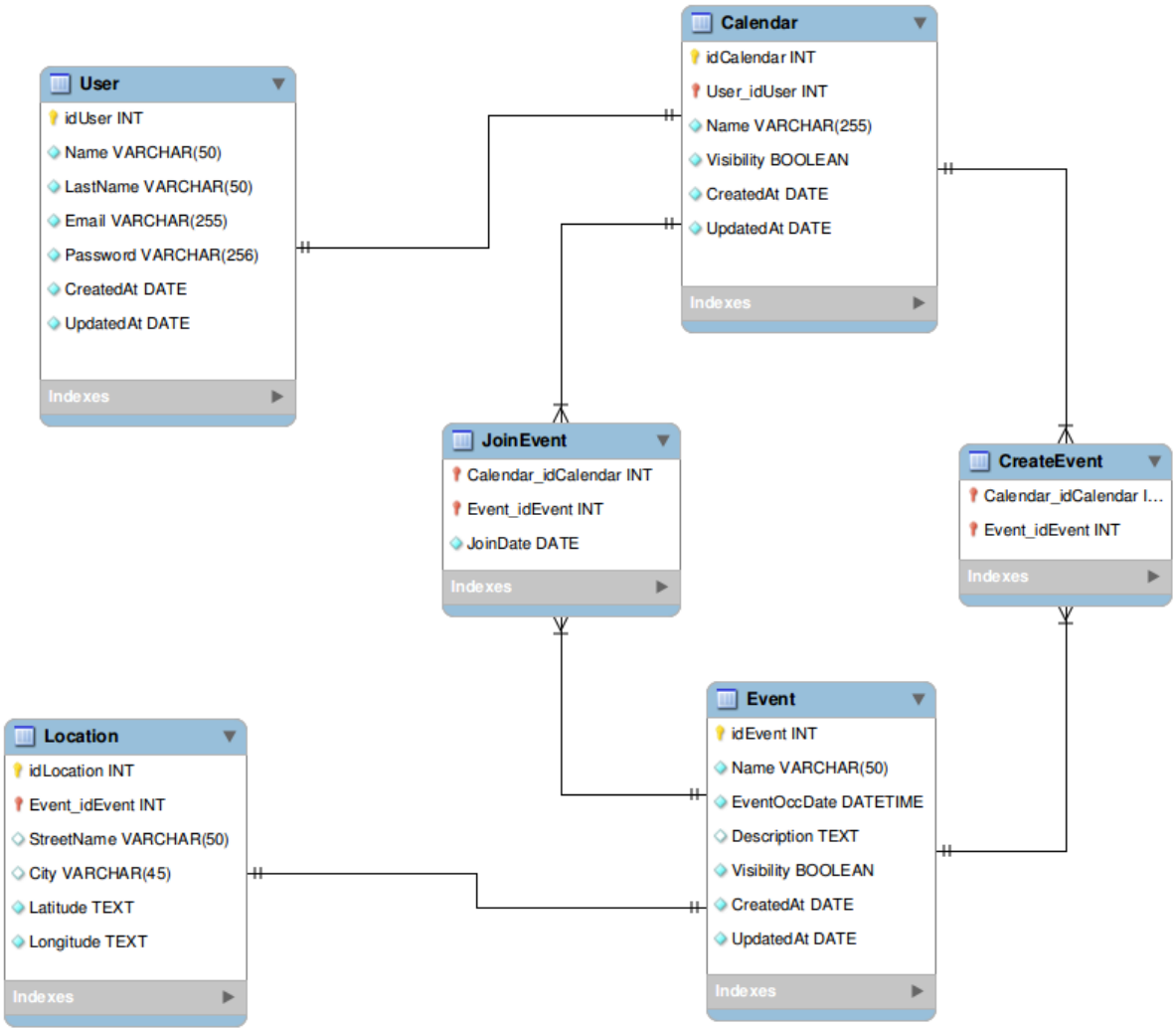
The logical design (LD) diagram represents the final implementation of the database. It is based on the conceptual ER diagram. The LD diagram, shown below describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. The relationships between the entities are created using the Crow's foot notation.



Database Design 2: Logical Design

5.2.3 Physical Design

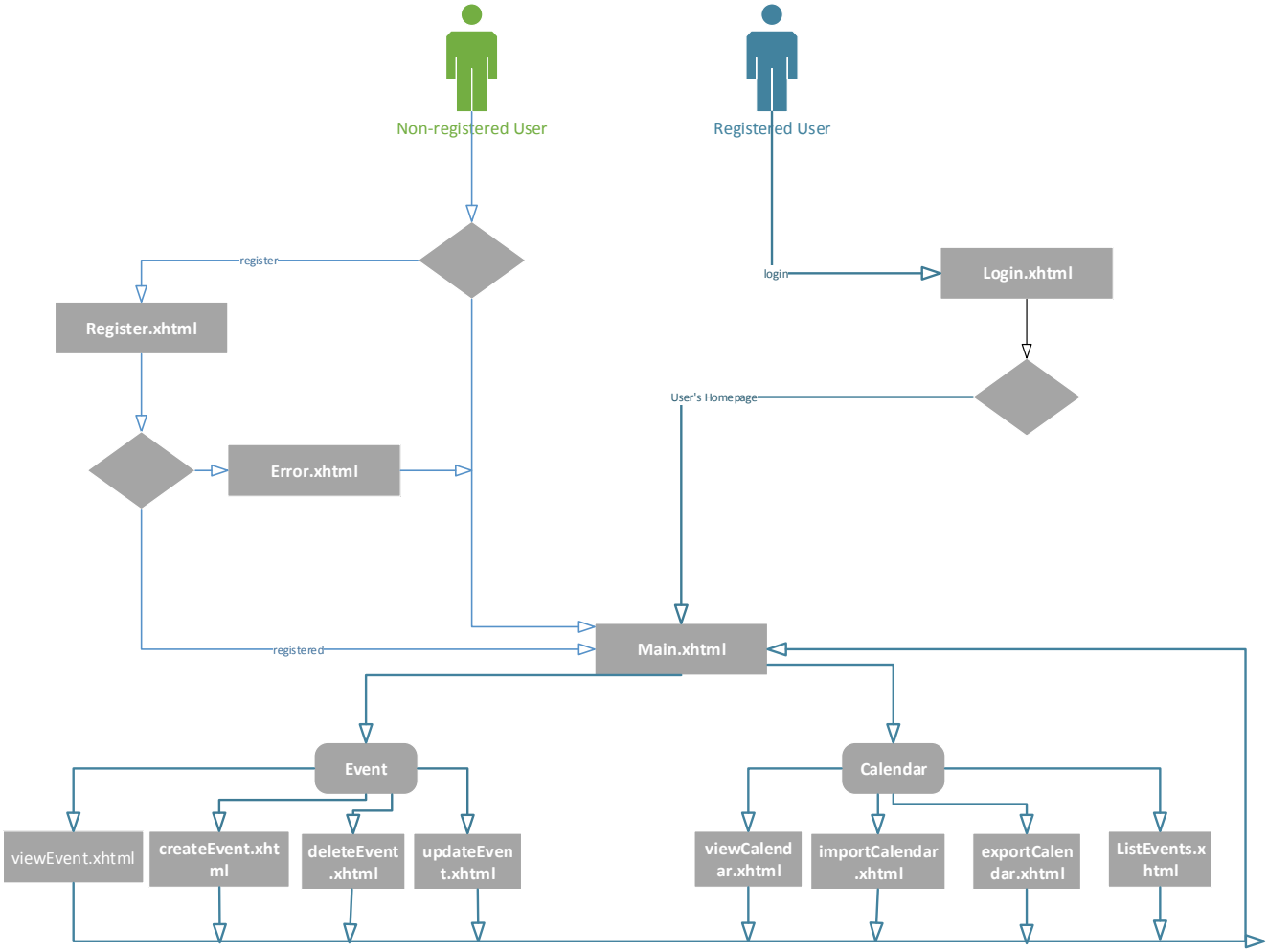
The physical design (PD) represents how the model will be built in the database.



Database Design 3: Physical design

5.3 WEBSITE ORGANIZATION

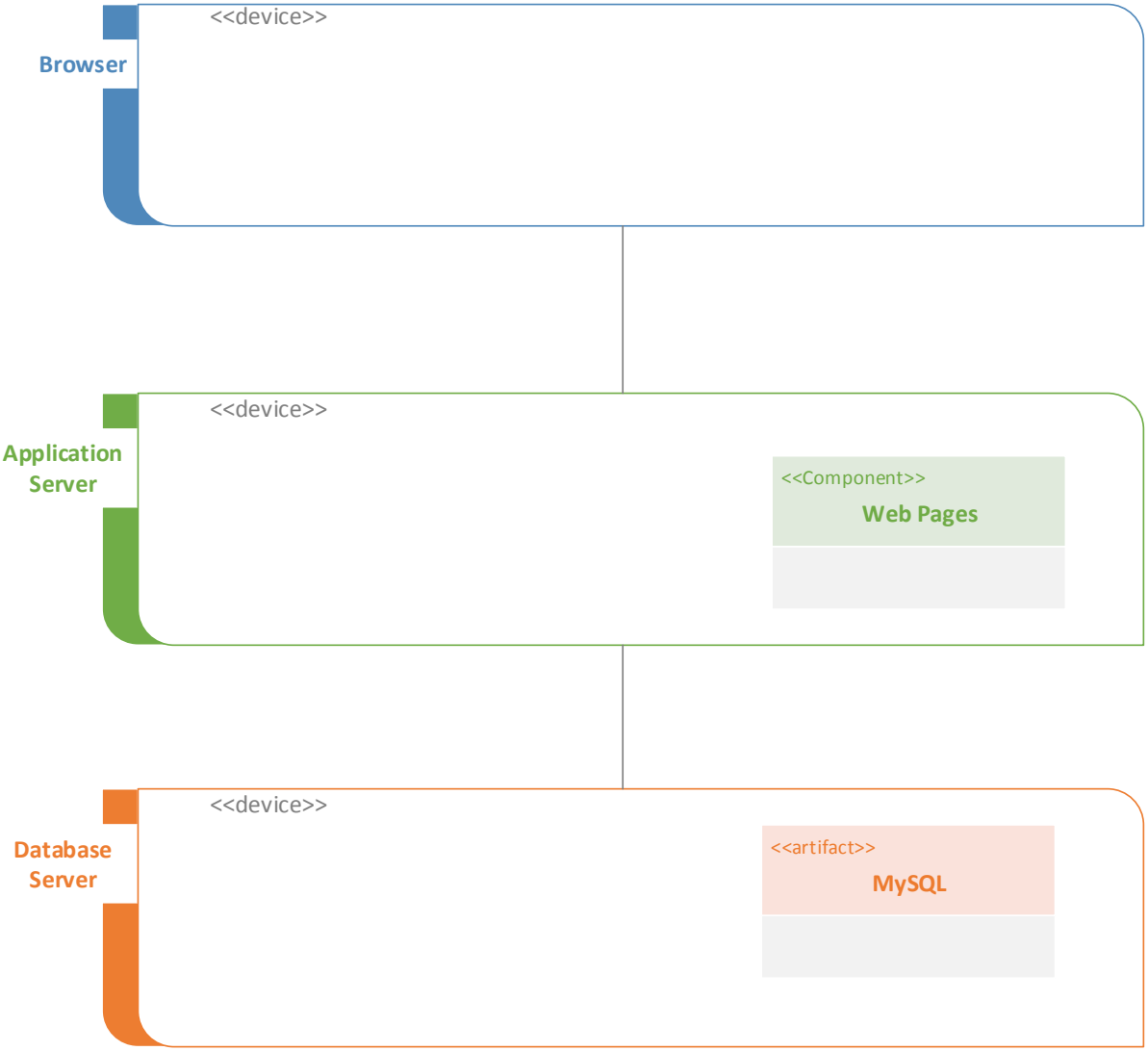
The following diagram explains the website navigation in MeteoCal project.



Navigation diagram 1: Website Navigation

5.4 DEPLOYMENT VIEW

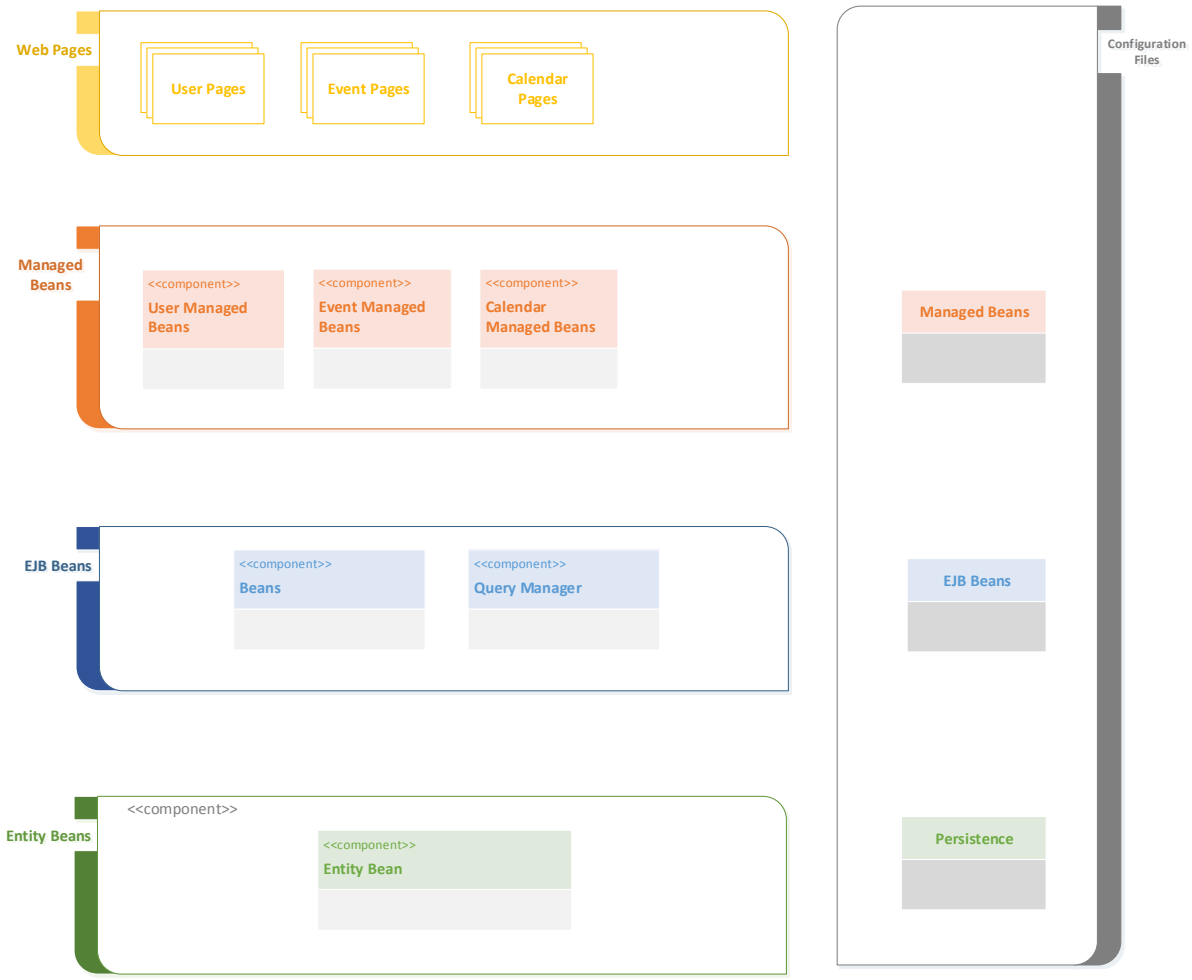
The following diagram displays the deployment view of the MeteoCal software application.



Deployment View 1: MeteoCal deployment view

5.5 MODULE VIEW

The following diagram suggests how the source code will be organized.



Module view 1: Source packages

6 APPENDICES