Politecnico di Milano
5th School of Engineering

Software Engineering Project

wim v2

Design Document

Irma Metra
Vlado Kragujevski
Javier Hualpa

Milan, December, 2012

Table of Contents

1. Introduction

1.1 Purpose of this document

This document describes the general and specific architecture of the Small World hypothesIs Machine version 2 (SWIMv2), the project of the course of Software Engineering 2 at Politecnico di Milano.
The document will explain the architectural decisions and tradeoffs chosen in the design process and its justifications.

1.2 Scope

The architectural descriptions provided concern the functional view, module view, deployment view, data layer, business logic and the user interface of the RASD. Hence the architecture will consider the following functionalities:

SWIMv2 will provide general functionalities for managing:

- **Profile**
  SWIMv2 will manage personal data of the different types of users. Users can be unregistered, registered and administrator.

- **Connections**
  SWIMv2 will manage the network of connections of the experts.

- **Skills**
  SWIMv2 will manage the list of skills that experts can have.

- **Users**
  SWIMv2 will manage registering, logging in/out of users.

1.3 Definitions and acronyms

1.3.1 Definitions

Keyword	Definitions
Skill	*An ability or area of expertise*
Connection	*A friend or expert that provides a service*
Software product	*System to be developed*
Expert/User	*A registered user*

1.3.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
SWIMv2	*Small World Hypothesis Machine version2*
JEE	*Java Enterprise Edition*
AS	*Application Server*

EJB	*Enterprise Java Beans*
FR	*Functional Requirement*
RASD	*Requirements Analysis and Specification Document*
RDBMS	*Relational Database Management System*
NFR	*Non-functional Requirement*
QoS	*Quality of Service*
XHTML	*Extensible HyperText Markup Language*
ER	*Entity Relationship*
LD	*Logical Design*
JAAS	*Java Authentication and Authorization Service*

1.4 References

1. Analysis document: SWIMv2_RASD_Metra_Kragujevski_Hualpa.pdf
2. IEEE Standard for Information Technology-Systems Design- Software Design Descriptions: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5167255

1.5 Overview

This document specifies the architecture of SWIMv2 spreading from the general into the specific. Also it describes the architectural decisions and tradeoffs and justifies them. The design was guided by a top-down process approach and the document structure reflects this tactic.

The document is organized as follows:

- **Section 1**, Introduction, provides a synopsis of the architectural descriptions.
- **Section 2**, Design Overview, provides a general description of SWIMv2 including its functionality and matters related to the overall system and its design.
- **Section 3,** Design Considerations, describes the design assumptions and constrains of SWIMv2.
- **Section 4**, Software Architecture, specifies the general architecture, describes the basic structure and interactions of the main subsystems.
- **Section 5**, Detailed System Design, specifies in detail the components of the system through different architectural views.
- **Section 6**, Appendixes, provides supporting information and additional material.

2. Design Overview

This section provides a general description of the software system including its functionality and concerns related to the overall system and its design.

2.1 Design context

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

2.1.1 Functionalities

The following functional requirements were identified during RASD. These functionalities are grouped by the following functional areas:

2.1.1.1 Managing Profiles

Functional requirements

[FR1] View personal information
[FR2] Modify personal information
[FR3] Add skill
[FR4] Remove skill

2.1.1.2 Managing Users

Functional requirements

[FR5] Register to the system
[FR6] Login
[FR7] Logout
[FR8] Modify password
[FR9] Recover password

2.1.1.3 Managing Connections

Functional requirements

[FR10] Accept/Deny connection requests
[FR11] View connection requests
[FR12] View pending requests
[FR13] View connections
[FR14] Remove connections
[FR15] Search connections
[FR16] Add connection
[FR17] View suggested connections

2.1.1.4 Managing Skills

Functional requirements

[FR18] Send Request for new skill only for an Expert
[FR19] Add new skill into the set of skills only for Administrator
[FR20] Accept new skills for the set of skills only for Administrator
[FR21] Remove skill from the set of skills only for Administrator
[FR22] Rate the Expert for his skill only for Expert

2.1.2 System technologies

The SWIMv2 will be designed considering the client-server 3-tier distributed architectural style. Each tier requires specific technologies as depicted below:

Web tier

- Dynamic web pages containing XHTML, which are generated by web components.
- Web components developed with Java Server Faces technology, which is a user interface component framework for web applications.

Business Logic tier

- Java Enterprise Edition 7(JEE7) platform supports applications that provide enterprise services in the Java language. It is the common foundation for the various kinds of components in Java.
- Enterprise Java Beans (EJB) 3.1, business components that capture the logic that solves or meets the needs of a particular business domain and persistence entities.
- JBoss AS 7.1, a server that provides services such as security, data services, transaction support, load balancing, and management of distributed applications and supports the JEE7 platform.

Persistence tier

- MySQL Server 5.5, a RDBMS.

2.2 General Design Descriptions

This subsection presents the roadmap followed to model the architecture of SWIMv2, including its functionality and matters related to the overall system and its design.

2.2.1 Design approach

The design approach is based on a client-server 3-tier distributed system, where each tier is described as follows:

- **Client tier:** This tier is responsible of translating user actions and presenting the output of tasks and results into something the user can understand.

- **Business Logic tier:** This tier coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the client and persistence tiers.

- **Persistence tier:** This tier holds the information of the system data model, and is in charge of storing and retrieving information from a database.

The design process followed a top-down process approach, so the outermost tiers were first identified and then broken into components that encapsulate the functionality. Hence each component is responsible for certain functionalities and interacts with others.

2.2.2 Overall design

This subsection presents the design model of SWIMv2, specifying the basic relations between packages, use cases and users.

2.2.2.1 General Package design

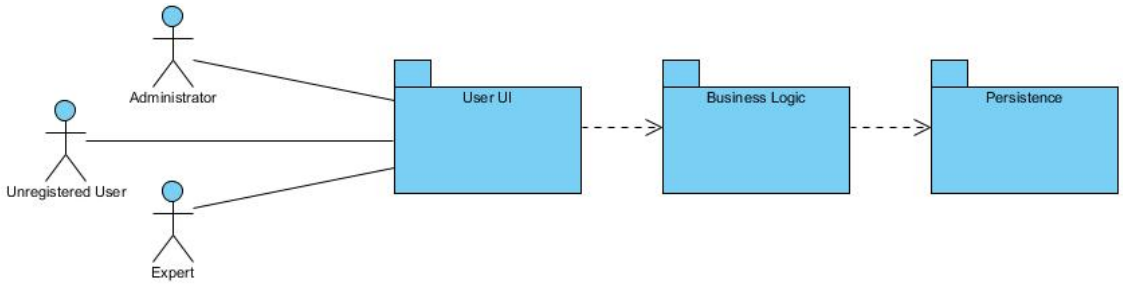
Since each tier is broken into components and each component is responsible for a set of functionalities that fulfill the requirements. There is a correlation between use cases (functionality) and package design. In the diagram we can identify three packages:

- **User UI:** This package contains the user interfaces. It is responsible for the interaction with the user such as getting UI requests, referring them to the Business Logic package and retrieving the data back for displaying.

- **Business Logic:** This package contains the business logic components. This package is responsible for handling the User UI package requests, processing them and accessing the Persistence package if required to provide a response.

- **Persistence:** This package is responsible for managing the data requests from the Business Logic package.

Administrator, Expert and Unregistered user access directly the User UI package and submit requests to accomplish their tasks.



Package Diagram 1: Basic Package

2.2.2.2 Detailed Package design

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

User UI

These set of sub packages are responsible for encapsulating the user actions and forwarding information requests to the Business Logic sub packages.

- Skill Pages: This package implements **[FR3], [FR4], [FR18], [FR19], [FR20], [FR21], [FR22].**
- Connection Pages: This package implements **[FR10] – [FR17].**
- Profile Pages**:** This package implements **[FR1], [FR2].**
- User Pages: This package implements **[FR5] – [FR9].**
-

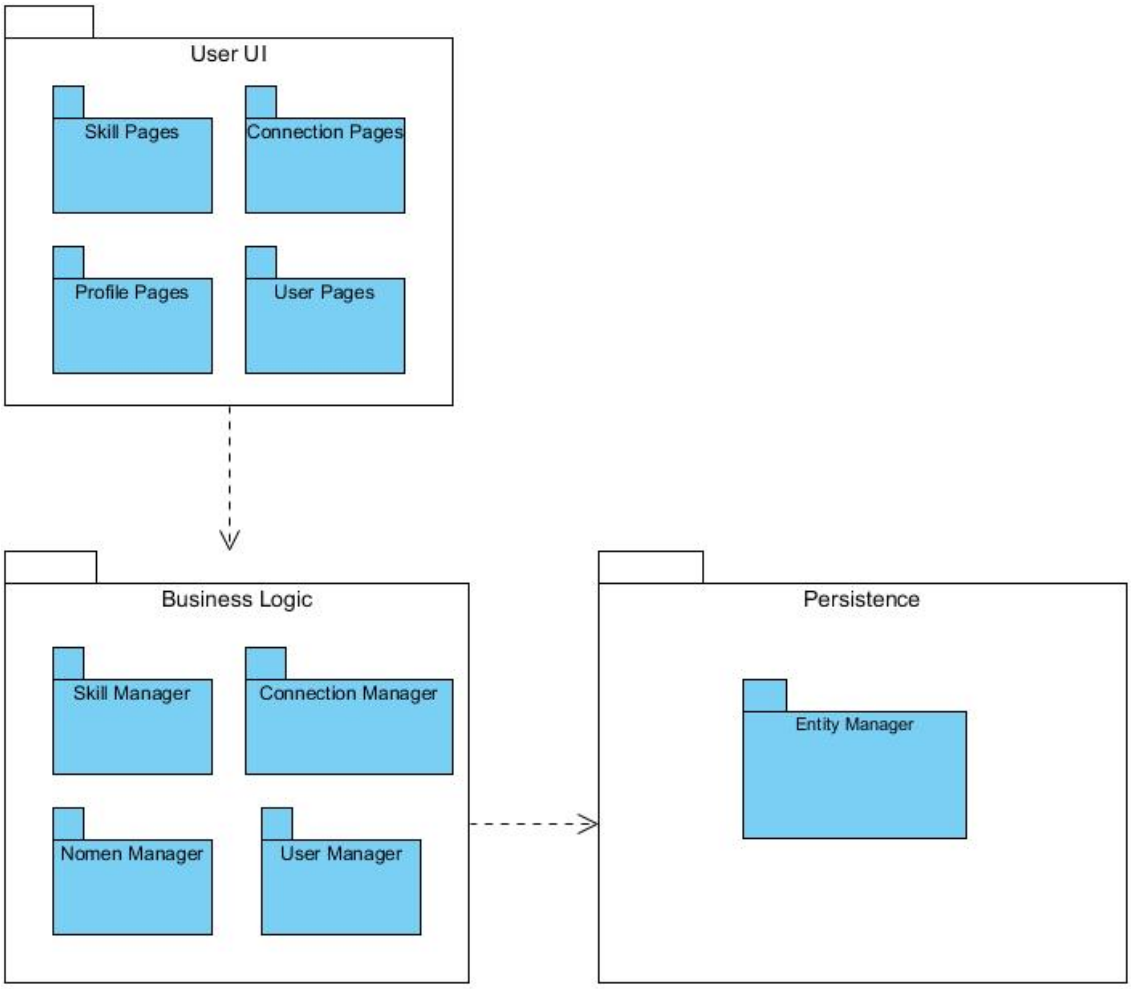
Business logic

These set of sub packages are responsible for handling requests from the User UI package, processing them and send back a response. These packages may access the Persistence package.

- Skill Manager: This package implements: **[FR3], [FR4], [FR18] – [FR22]**.
- Connection Manager**:** This package implements **[FR10] – [FR17].**
- Nomen Manager: This package implements **[FR2], [FR6], [FR22].**
- User Manager: This package implements **[FR1], [FR5] – [FR9].**

Persistence
This sub package contains the data model for the system. It accepts requests from the Business Logic package.

- Entity Manager: This package implements **[F1] – [F22].**



Package Diagram 2: Detailed Package

3. Design Considerations

This section encompasses the design considerations taken into account in the SWIMv2 system design. Assumptions, dependencies, general constraints and performance requirements are clearly stated.

3.1 Assumptions

Assumption	Action
The software product provides one	The administrator credential is provided; hence there is no need for manual set up.

administrator by default	
The software product does not supports more than 1 administrator	Only one individual can perform the administrative tasks. Support for multiple administrators is planned for a future release.

3.2 Dependencies

Dependency	Impact
Java virtual machine that supports JEE7 is already installed on the OS.	SWIMv2 only runs on operating systems that support the JEE7 platform.
The supported browsers will be Firefox and Chrome	SWIMv2 outputs XHTML code that requires browsers that support most of the web standards, elsewhere the UI experience will be affected.
A JEE7 AS is required on the server side	SWIMv2 cannot operate if there is no AS that supports the JEE7 standard.

3.3 General Constraints

This subsection describes the NFRs and the QoS details related to the design of the software product.

Element	Requirement
Memory	2 GB+
Database server	MySQL
Network	Internet Access, HTTP protocol
Security	The software product is controlled for each type of user. SSL is not supported.
Hard disk space	40 GB+

3.4 Performance Requirements

3.4.1 Standard compliance

The software product does not have to meet any standard compliance.

3.4.2 Reliability

For assuring the reliability of the software product, it is mandatory to back up the database periodically.

3.4.3 Availability

An AS is used to guarantee availability of the software product. However in order to eliminate the downtime cause by the dependency on a single point of failure redundancy in the AS instances is recommended. For the first 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.

3.4.5 Maintainability

The architectural style and the component definition described contribute to low coupling and high cohesion of the software product.

3.4.6 Portability

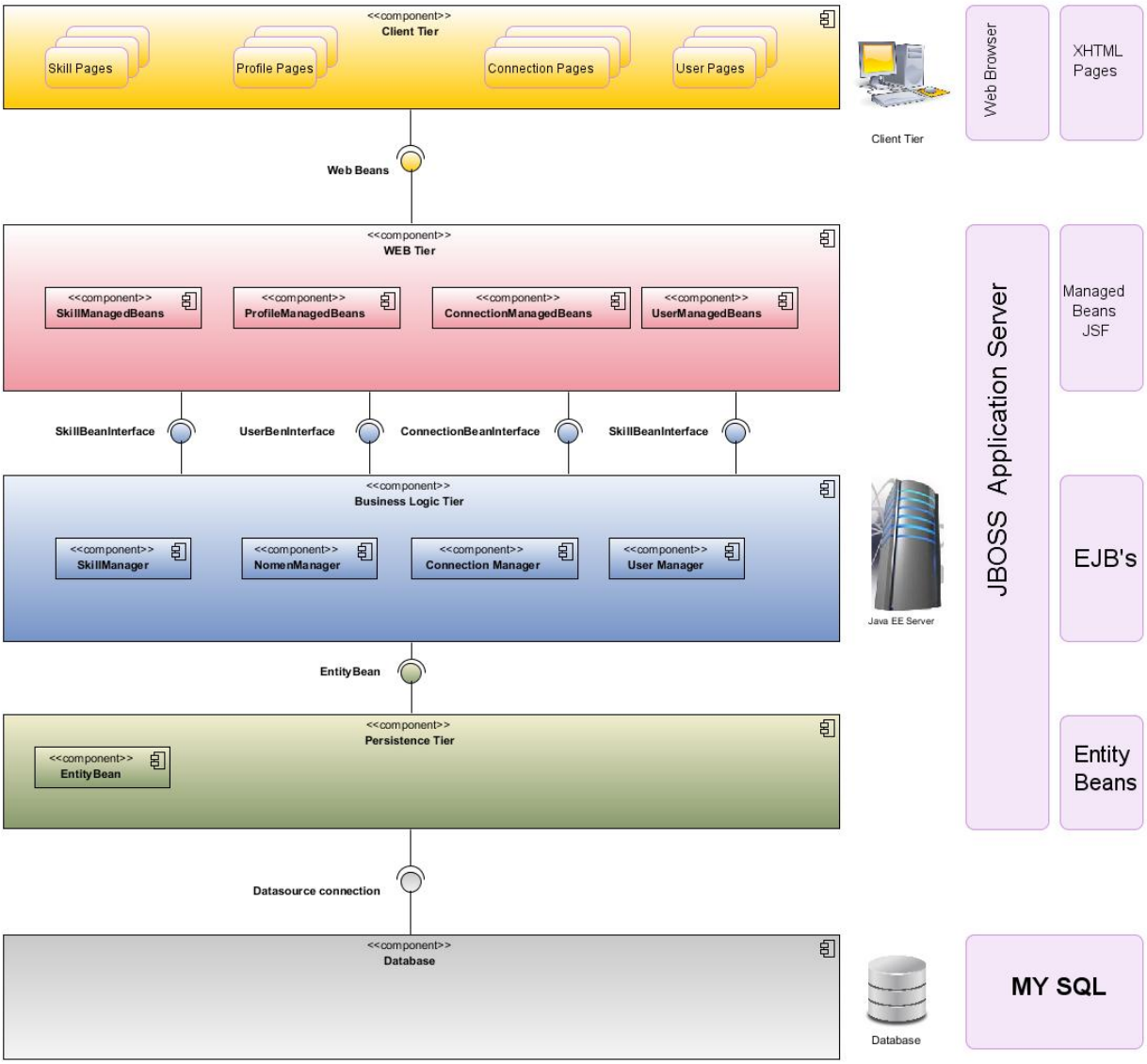
The software product has been developed using the Java language and related dependent technologies. Java is specifically designed to have as few implementation dependencies as possible. Meaning that code that runs on one platform does not need to be recompiled to run on another.

4. Software Architecture

SWIMv2 shall be developed by using a general 3-tier JEE Architecture, as mentioned in section 2.2. We have conceptually identified the components of each, by dividing them in tiers which represent logically a clearer view of what each of them is responsible of. Section 4.1 gives a general description of the architecture we propose for our system and furthermore it provides a small description for each of the components. Detailed description is found in section 5. Detailed software design. Hereby we need to clarify that:

- Client Tier and Web Tier represent the Web Component.
- Business Logic Tier represents the Business Logic component.
- Persistence Tier represents Persistence Component.
- Database represents the data model.

4.1 Conceptual design



Architecture Design 1: General architecture

The above diagram represents our conceptual design of Swim. This diagram depicts all the components of the designed software, by clarifying the logical separation between the tiers (Although we will build our application based on the 3-tier architecture). It is clear that the user will interact with the XHTML pages, which in their side are implemented with beans to manage user interaction. This web interaction is then supported by the business tier, which holds on the information provided by the persistence layer. The persistence layer is the one in charge of the connection to the database and managing all the queries needed from the above layers.
Further explanation on the technologies used is found in section 4.2.

4.1.1 Client Tier

The client tier is composed of the XHTML pages that the normal user (in our case the expert)

will see. Actually this is strictly related to the Web Tier, and in section 5 it is detailed together with the Web Tier.

4.1.2 Web Tier

Web Tier is composed of the web beans. This tier receives the requests of the user and has some specific beans which listen to these events and display data regarding the user requests. They interact with the beans in Business Tier, to retrieve the information. Since we are using JSF these beans are called Managed Beans.

4.1.3 Business logic Tier

The business logic tier is composed of all the logic underlying our application; it is responsible of communication with Web Tier and Persistence Tier. Its components are the EJB Beans, named as Managers in section 5.1, just to differentiate from the web beans.

4.1.4 Persistence Tier

The persistence tier is composed of the entity beans which represent the entities depicted from our RASD document and then further endorsed in our conceptual design. These entities are fundamental as they represent the connection to our database. Since in JEE we are interested in working in an object oriented environment, they represent a high level object view of the database of the application, which from its side connects to the Database Tier, to insert, update, delete, select.

4.1.5 Database

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

4.2 System Specification

The following table displays the technologies we will use during implementation. All of the technologies are free source technologies.

Component Name	Technology
Client Tier- Web Tier	XHTML integrated with Java Server Faces
Business Tier	JEE with AS JBOSS 7.1
Persistence Tier-Database Tier	MySQL 5.5

5. Detailed Software Design

In this section we provide detailed insight into product structure and (intended) implementation regarding the general structure already detailed in section 4.1. We are releasing some of components from some implementation details since we are not very confident in JEE. In the following sections we have provided details about the general subcomponents, which are object of change in the further phase.

5.1 Implementation modules / components

Our project is composed of 3 main components, which shall be implemented during implementation phase.

- Web component
- Business logic component
- Persistence component

5.1.1 Web component

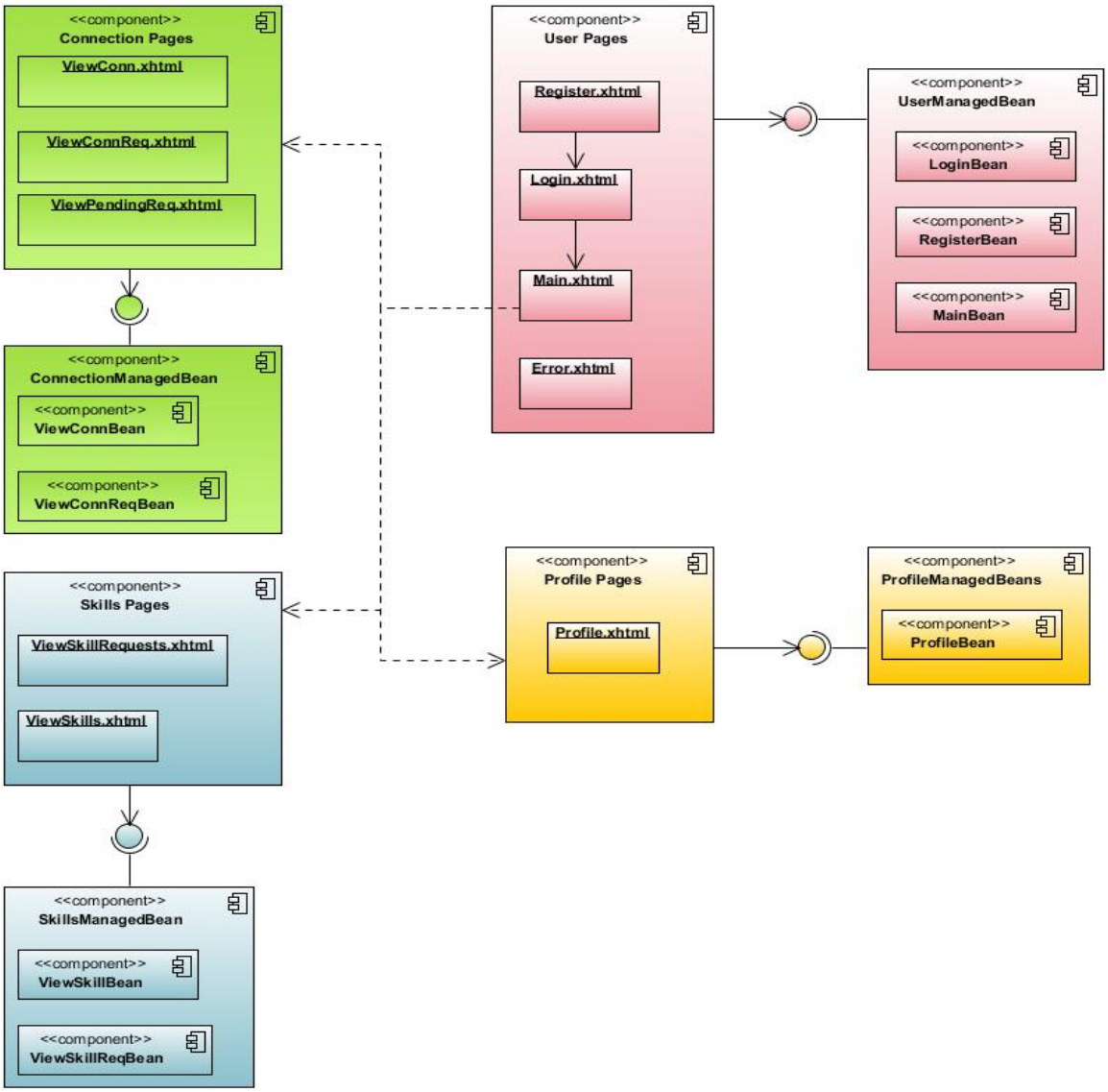
In this section we provide the useful information for what needs to be implemented regarding this component. The following general diagram shows the subcomponents and their communication.
We have identified 4 main Subcomponents and their related Managed Beans.

- Connection Pages
- User Pages
- Skills Pages
- Profile Pages

We hereby remind that we are using JSF and that user events in the pages are managed by Managed Beans divided in 4 sections as well. The related beans Managed Beans are:

- UserManagedBeans
- ProfileManagedBean
- ConnectionManagedBean
- SkillsManagedBean

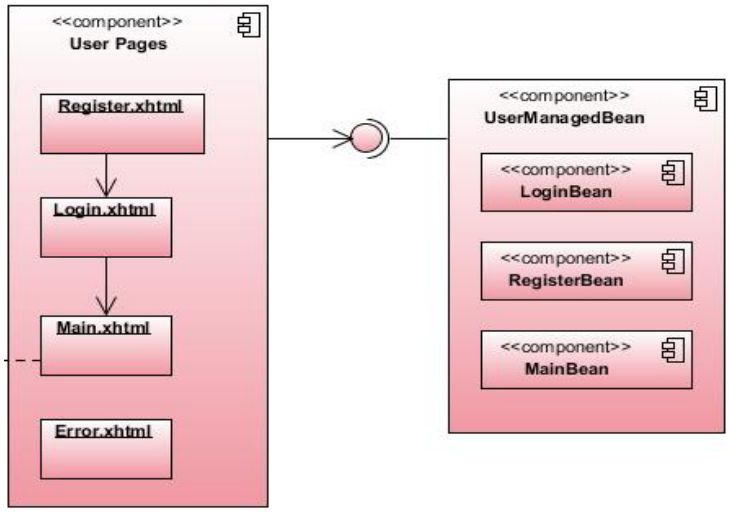
These beans represent the conceptual idea of the Managed Beans, as there will be more managed beans that will be needed during implementation.



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

Component Name	
Classification	Login.xhtml
Definition	User interface for user login
Responsibilities	• Display login form

Component Name	
Classification	Register.xhtml
Definition	User interface for user registration
Responsibilities	• Load registration form

Component Name	
Classification	Main.xhtml
Definition	User interface for displaying main menu for a registered user.
Responsibilities	• Display user main homepage • Display connections to Profile, Connections, Skills pages connections • Display admin main homepage • Display connections to Profile and Skills pages. (Main.xhtml will detect which is the user role (administrator, expert) and will display the appropriate homepage.

Component Name	
Classification	Error.xhtml
Definition	User interface for displaying error messages
Responsibilities	• Load type of error because of user input

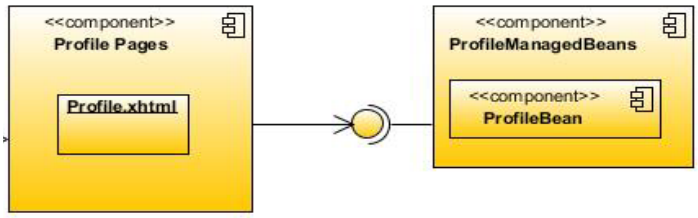
Component Name	
Classification	LoginBean
Definition	Managed been for user login
Responsibilities	• Load login form • Check login credentials • Redirect to Error.xhtml or Main.xhtml

Component Name	
Classification	RegisterBean
Definition	Managed been for user registration
Responsibilities	• Load registration form • Check compulsory data • Redirect to Error.xhtml or Main.xhtml

Component Name	
Classification	MainBean
Definition	Managed bean for displaying main menu for a registered user.
Responsibilities	• Load user main homepage • Load connections to Profile, Connections, Skills pages connections • Load administrator main homepage • Load connections to Profile and Skills pages. (Main.xhtml will detect which is the user role (administrator, expert) and will display the appropriate homepage.

5.1.1.2 Profile pages and managed beans

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



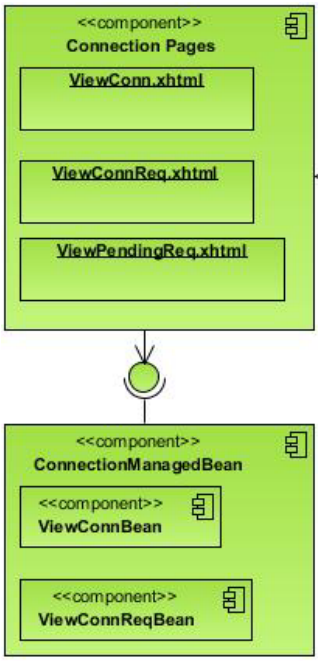
Component Design 3: Profile page and managed beans

Component Name	
Classification	Profile.xhtml
Definition	User interface for displaying user profile information
Responsibilities	• Display profile information

Component Name	
Classification	ProfileBean
Definition	Managed Bean for displaying user profile information
Responsibilities	• Load profile information based on the user.

5.1.1.3 Connection pages and managed beans

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



Component Design 4: Connection pages and managed beans

Component Name	
Classification	ViewConn.xhtml
Definition	User Interface for displaying connections
Responsibilities	• Display in a table the connections.

Component Name	
Classification	ViewConnReq.xhtml
Definition	User interface for displaying connection requests.
Responsibilities	• Display in a table the connection requests

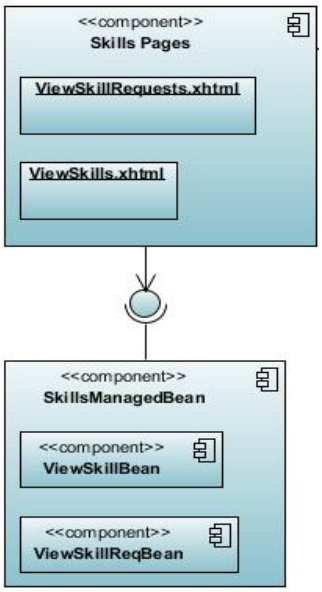
Component Name	
Classification	ViewPendingReq.xhtml
Definition	User interface for displaying pending connection requests.
Responsibilities	• Display in a table the pending connection requests.

Component Name	
Classification	ViewConnBean
Definition	Managed bean for list of connections
Responsibilities	• Load list of the connections of the user that is logged in • Load functionalities, of deleting a connection

Component Name	
Classification	ViewConnReqBean
Definition	Managed bean for list of connection requests
Responsibilities	• Load list of connection requests of the user that is logged in • Provide functionalities of accepting a connection, or denying a connection • Load list of suggestions in case of accepted connection request.

5.1.1.4 Skills pages and managed beans

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



Component Design 5: Skill pages and managed beans

Component Name	
Classification	ViewSkillRequests.xhtml
Definition	User interface for viewing list of skill requests.
Responsibilities	• Display a table of the skill requests, and the user who requested them. (option only for administrator)

Component Name	
Classification	ViewSkills.xhtml
Definition	User interface for viewing list of skills.
Responsibilities	• Display a table of skills.

Component Name	
Classification	ViewSkillBean
Definition	User interface for viewing list of skills.
Responsibilities	• Check the user role with the connection with user bean • Load the user skills, if user • Load all skills if administrator.

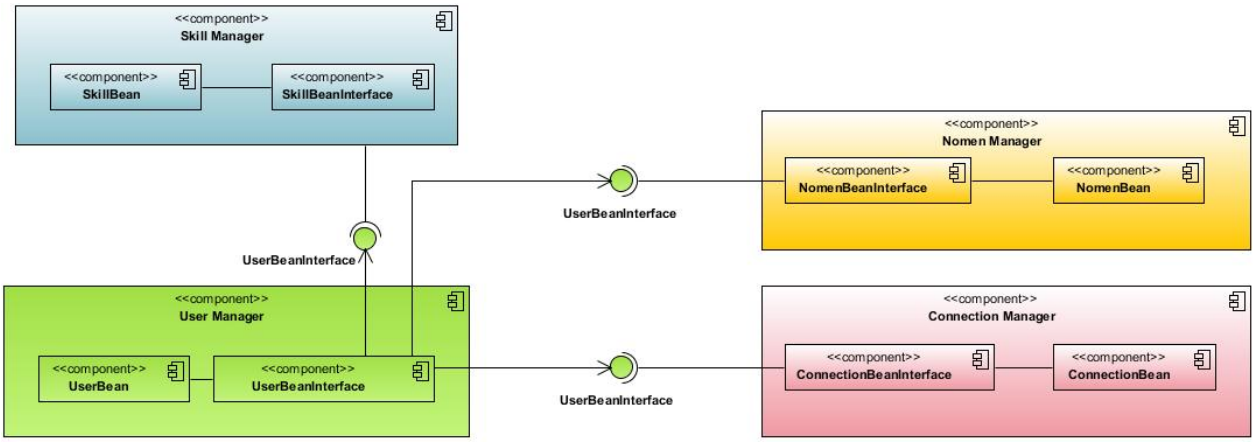
Component Name	
Classification	ViewSkillReqBean
Definition	Managed Bean for viewing list of skill requests.
Responsibilities	• Check the user role with the connection with user bean • Load skill requests.

5.1.2 Business logic component

In this section we provide the useful information for what needs to be implemented regarding this component. The following general diagram shows the subcomponents and their communication. We have identified 4 main subcomponents.

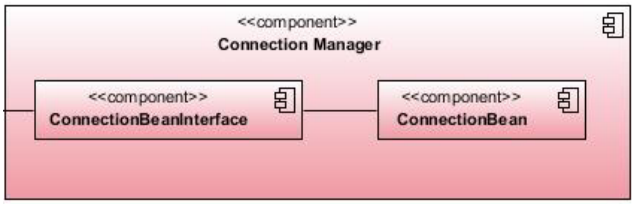
- Connection Manager
- User Manager
- Skill Manager
- Nomen Manager

Each of these beans will be defined as an EJB Bean by defining all the required methods in the specific beans.



Component Design 6: Business Logic components

5.1.2.1 Connection Manager

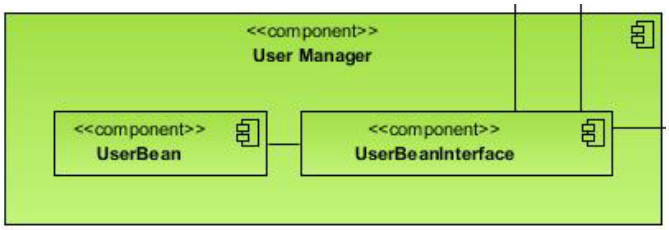


Component Design 7: Connection Manager

Component Name	
Classification	ConnectionBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	• Communicate with UserBeanInterface

Component Name	
Classification	ConnectionBean
Definition	Bean in charge for all functionalities related to connections.
Responsibilities	• Search connection • Create a connection • Delete a connection • Add a connection • Create list of suggested connections • Create list of connections

5.1.2.2 User Manager

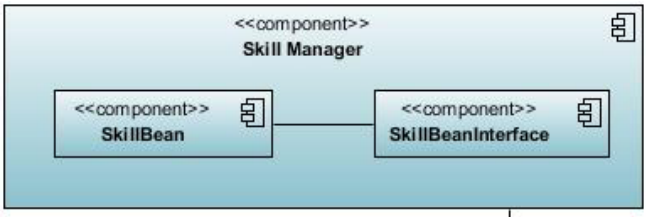


Component Design 8: User Manager

Component Name	
Classification	UserBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	• Communicate with ConnectionBeanInterface, NomenBeanInterface, SkillsBeanInterface

Component Name	
Classification	UserBean
Definition	Bean in charge for all functionalities related to users.
Responsibilities	• Login user/administrator • Logout user/administrator • Register a new user • Load profile information for user/administrator • Search user

5.1.2.3 Skill Manager

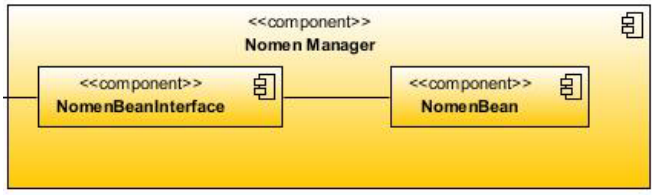


Component Design 9: Skill Manager

Component Name	
Classification	SkillBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	• Communicate with UserBeanInterface

Component Name	
Classification	SkillBean
Definition	Bean in charge for all functionalities related to skills.
Responsibilities	• Load list of user skills • Load list of all skills for administrator • Load list of skill requests for administrator • Add skill • Remove skill • Manage skill requests for administrator • Search skill

5.1.2.4 Nomen Manager

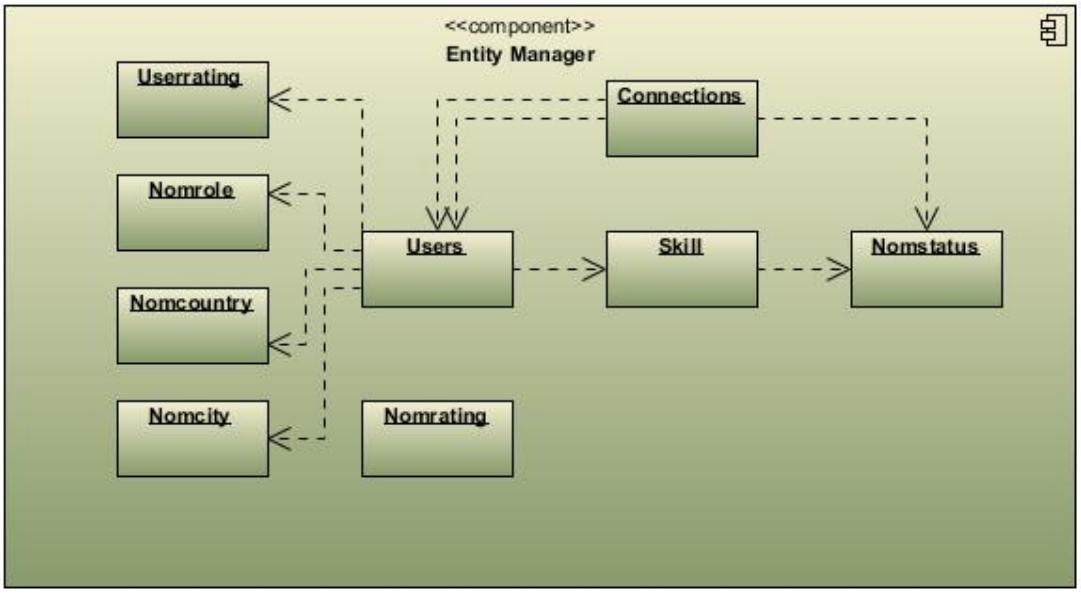


Component Design 10: Nomen Manager

Component Name	
Classification	NomenBeanInterface
Definition	Interface instantiated every time communication between beans is needed.
Responsibilities	• Communicate with UserBeanInterface

Component Name	
Classification	NomenBean
Definition	Bean in charge for all functionalities related to nomenclature entities.
Responsibilities	• Load list of available countries, cities, ratings

5.1.3 Persistence component



Component Design 11: 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
Users	Represents the users in the system
Skill	Represents the skills of the users in the system
Connections	Represents the connections between all pairs of users in the system
Userrating	Represents the rating of each of the users in the system
Nomerole	Represents a nomenclature of all the roles in the system
Nomcountry	Represents a nomenclature of all the countries in the system
Nomcity	Represents a nomenclature of all the cities in the system
Nomstatus	Represents a nomenclature of all the types of statuses that a connection and a skill can have
Nomrating	Represents a nomenclature of all the types of ratings that one user can give to another one

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

Let's start by describing all the elements in the ER diagram. A common characteristic for all entities is that they have primary key named id. In the ER diagram, we can see that all entities have an underlined attribute id.
The diagram has five Nomen entities: Nomcity, Nomcountry, Nomrole, Nomrating and Nomstatus. All of them inherit from the Nomen entity by using the IsA relationship. The use of the Nomen's entities is to represent a specific form of data in our system, called nomenclatures.
The Users entity represents the information for the users in the system. Besides the main attributes, we want to know from which country the user is, so we have a 1:M relation between the Users and Nomcountry, because one user can be from only one country, but the country can have many users. Then, with the same explanation we have 1:M relations between Users and Nomcity, and between Users and Nomrole, because we want to have information for the city of the user, and which role the user belongs to.

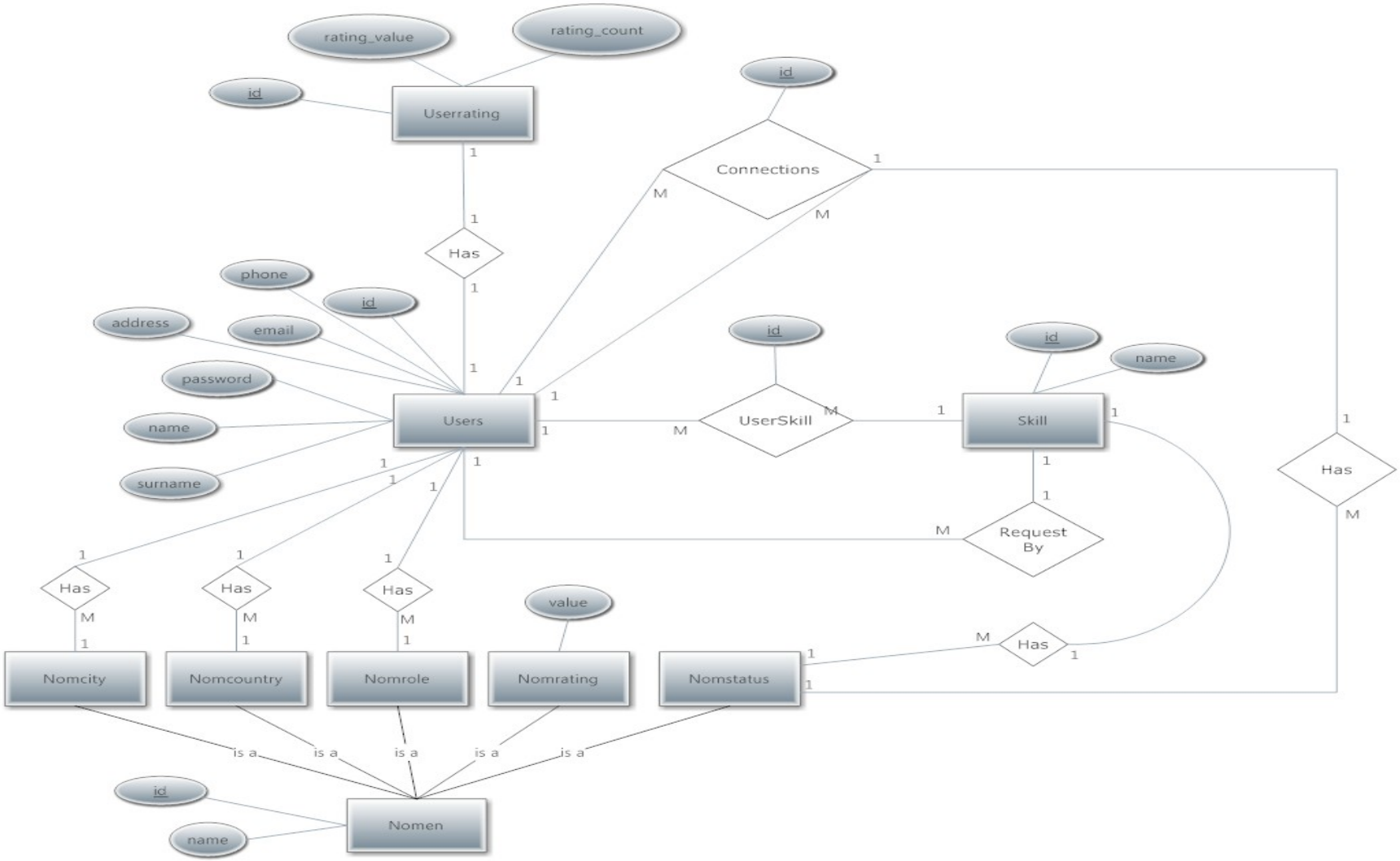
The Userrating entity represents the value of the current rating of each of the users. Here the rating_value attribute is the current value of the rating, and the rating_count attribute is used for storing the information of how many times the user was rated. The Users entity

and the Userrating entity are related with 1:1 relation, because one user can have only one rating, and one rating can be related with only one user.

The Connections entity is an M:M relation, and is a self-loop of the Users entity, because in each connection there are two different users. So, one user can have many connections with different users. Here we want to have information of the current status of the connection, and we have a 1:M relation between Connections and Nomstatus. So one connection can have one Nomstatus, but one Nomstatus can be related with many connections.

The Skill entity represents all the skills in the system. Besides the base attributes, for each skill we want to know who is the person that requested the skill, so we have a 1:M relation between Skill and Users, because one Skill can be requested by only one user, and one user can request many skills. Then, we want to know what is the current status of each skill, so we have a 1:M relation between Skill and Nomstatus, because one skill can have only one status, and one status can be related with many skills.

The UserSkill is a M:M relation, between Users and Skill. So one user can have many skills, and one skill can be related with many users. The objective of this relation is to provide a list of skills for every user in the system.



Database Design 1: Conceptual Desig

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, showed below, is generated automatically by using the MySQL workbench tool. Using the conceptual model, to create the database, will means that the entities will be translated into tables, and the attributes of each of the entities will become columns in the tables. Furthermore, the M:M relations will become tables in the LD diagram, and the 1:M relations will be model by using foreign keys. Let's now explain how we created the LD diagram.

In the LD diagram we see all the tables in the database. For all of them, the common characteristic is that the primary key is of type integer, and is an autoincrement column. That means when inserting new tuples into the table, we are not going to take care about the value of the id column, because it will be assign automatically by the database management system (DBMS).

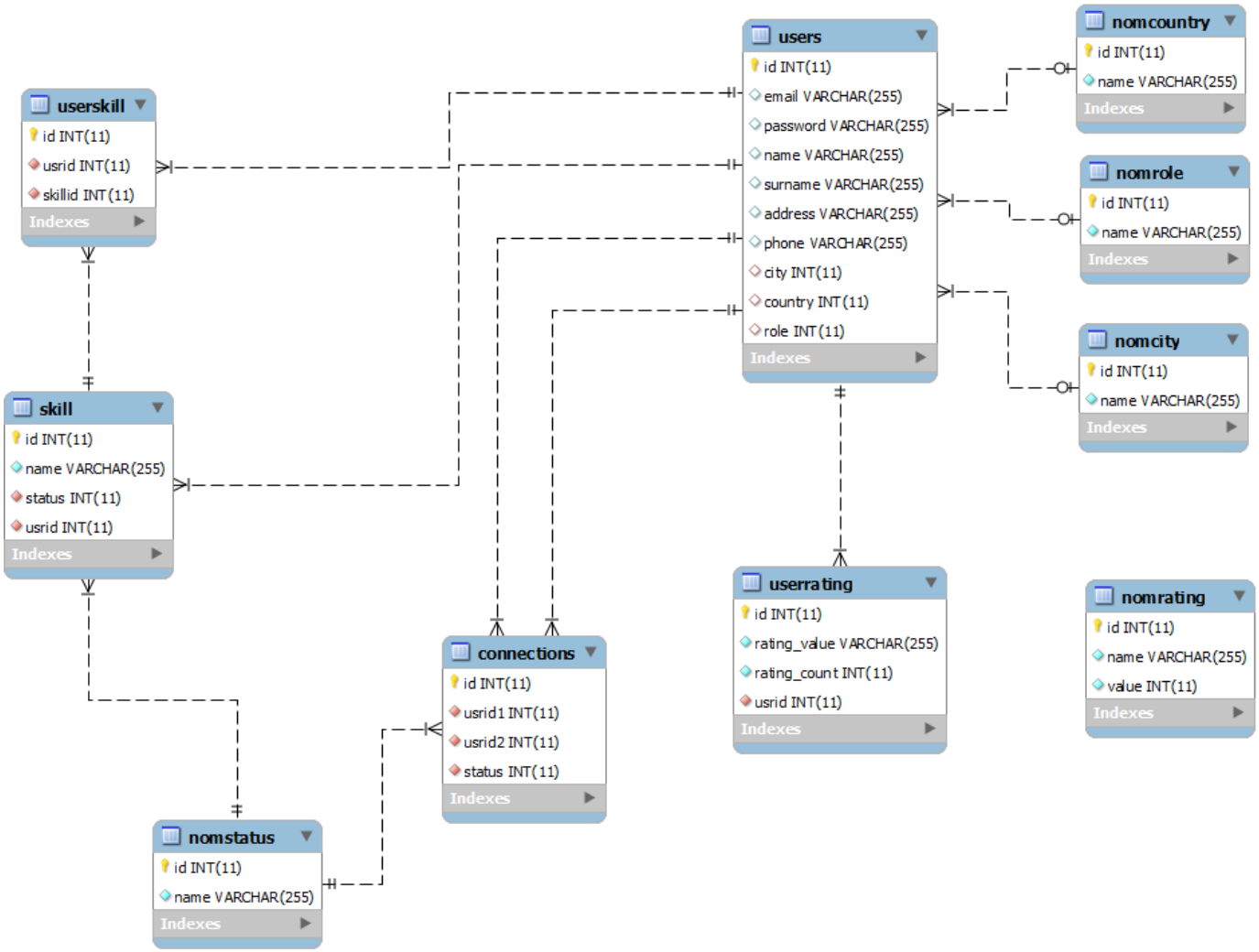
The **users** table, besides the main attributes, has three foreign keys to the tables: **nomcity**, **nomcountry** and **nomrole**, and likewise we are implementing the 1:M relations from the conceptual model of the database. Each of these foreign keys is based on an integer column value, because we created all of the primary keys in the database as integer autoincrement columns. These columns are: city (the id of the city), country (the id of the country) and role (the id of the role).

The **skill** table, besides the main attributes, contains foreign key for modeling the 1:M relation between **skill** and **users**.
This foreign keys is based on the usrid (the id of the user that requested the skill) column. Furthermore, in the **skill** table we have a foreign key for modeling the 1:M relation between **skill** and **nomstatus**, and this key is based on the status column.

The **userskill** table represents the M:M relation between **users** and **skill**. It keeps information of the skills of each of the users in the system. This table, besides the primary key, has only two foreign keys and two columns for each of the keys. The first foreign key is for modeling the relation between the **userskill** and **users**, and is based on the usrid column. The second foreign key is for modeling the relation between the **skill** and **userskill**, and is based on the skillid column.

The **userrating** table stores all the ratings of all the users. Each user has only one rating, so besides the main attributes the table has a foreign key for modeling the 1:1 relation between **users** and **userrating**, and the key is based on the usrid column.

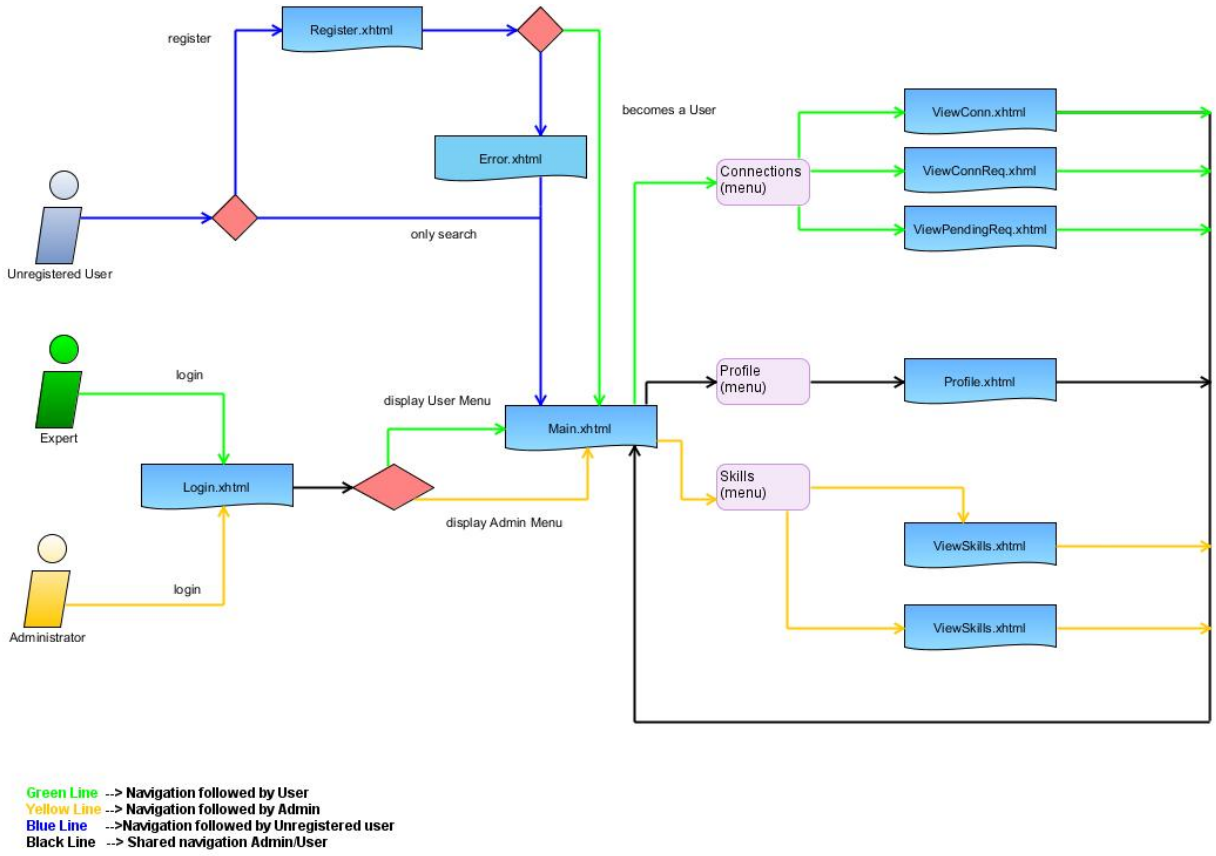
The connections table represents the M:M self-relation of the users table, and it keeps the pair of users that are connected between each other. So, to model the two relations between the **connections** and **users**, the table contains two foreign keys. They are based on the columns usrid1 and usrid2. Furthermore, the **connections** table has another foreign key for modeling the 1:M relation between **connections** and **nomstatus**, and this key is based on the status column.



Database Design 2: Logical Design

5.3 Web site organization

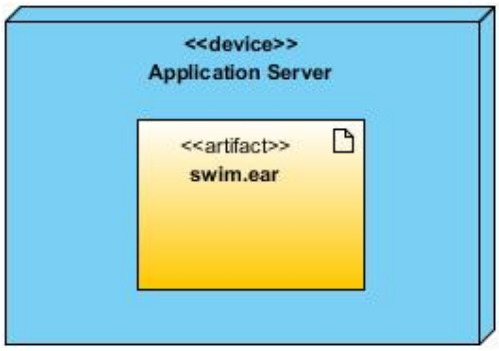
The following diagram describes the site navigation in Swim project.



Navigation Diagram 1: Web site navigation

5.4 Runtime View

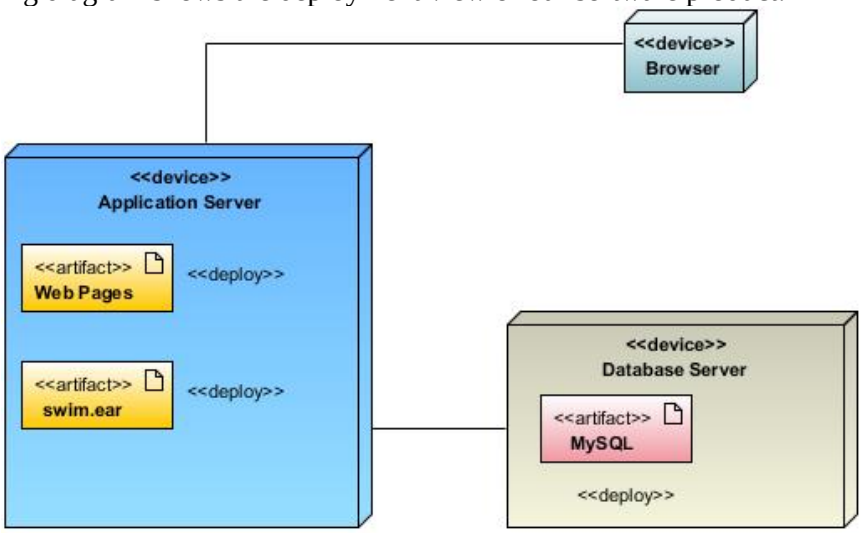
The following diagram describes the runtime view of Swim project. The software product that will be released will be SWIM.ear and can be deployed in any a JEE Application Server.



Runtime View 1: SWIMv2 Runtime view

5.5 Deployment View

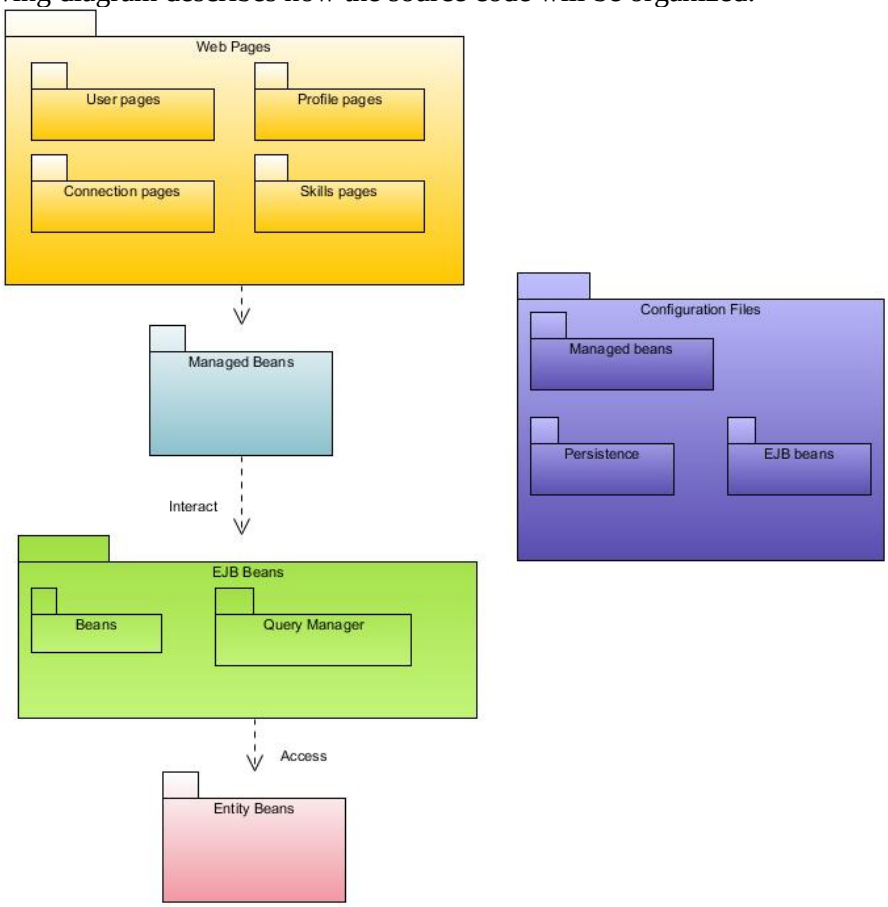
The following diagram shows the deployment view of our software product.



Deployment View 1: SWIMv2 Deployment View

5.6 Module View

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



Module View 1: Source Packages

6. Appendixes

6.1 RASD Modifications

6.1.1 Section 3.2.7.13 Scenario View Pending Requests

Actor should be Expert, not Administrator.

6.1.2 Changes in the Mockup

Skills will not be a separate menu but, it will be inside the profile of the user. Search for registered users will be valid only in the Main page.