User CRUD Service Design Document

BOOTCAMP JAVA 2016

2016

Table of Contents

[1 Introduction 2](#_Toc441423657)

[1.1 Purpose 2](#_Toc441423658)

[1.2 Scope 2](#_Toc441423659)

[1.3 Definitions, Acronyms and Abbreviations 2](#_Toc441423660)

[2 Design Overview 3](#_Toc441423661)

[2.1 Introduction 3](#_Toc441423662)

[2.2 Services Design 3](#_Toc441423663)

[2.3 Services Interfaces 3](#_Toc441423664)

[2.3.1 External User Interface Requirements 3](#_Toc441423665)

[3 Object Descriptions 5](#_Toc441423666)

[3.1 Objects 5](#_Toc441423667)

# Introduction

This document describes all the details necessary for the implementation of the specified services. The first sections portray the purpose and scope of the document. What is more, the process that services perform to satisfice the requirements.

## Purpose

The aim of this document is to specify, document and provide a detailed description of how services work and their interface.

## Scope

This document is for a base level system which will work as a proof of concept for the use of building a set of service. It will affect all the stages of the development process: analysis, implementation and testing.

## Definitions, Acronyms and Abbreviations

UML: Unified Modeling Language

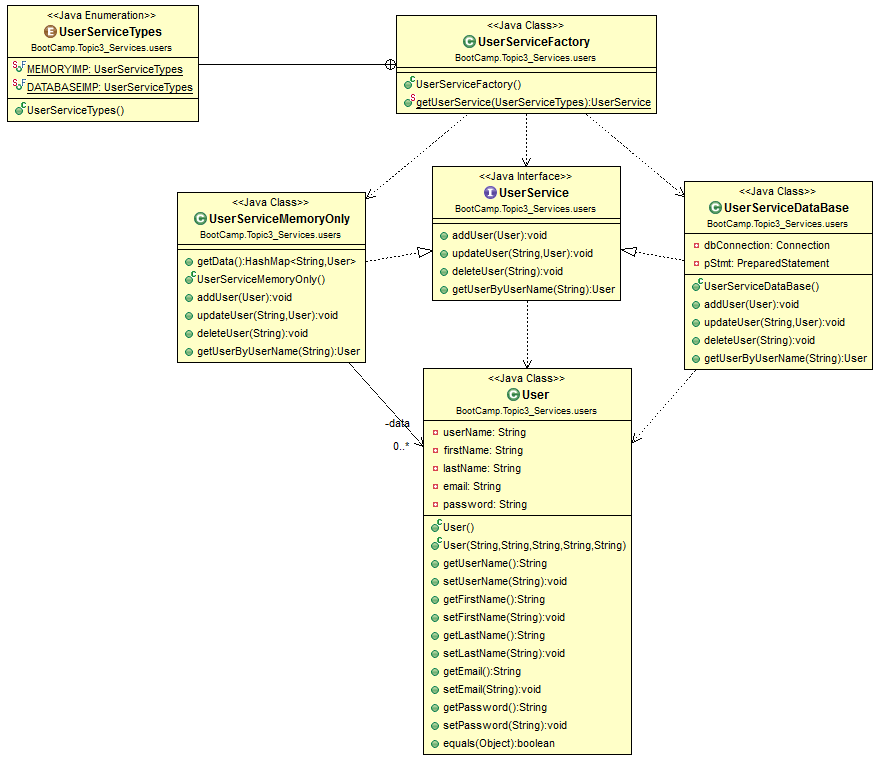
# Design Overview

## Introduction

The Design Overview is a section to introduce and give a brief overview of the design. The Services Design is a way to give the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient themselves to the design and see a summary before proceeding into the details of the design.

## Services Design

UML class diagram of the inherent classes in the services.



## Services Interfaces

### External User Interface Requirements

#### User Interfaces

The user interface for the system will allow the user to easily add, update, remove and get user information.

* addUser: Creates a new user and add this user to appropriate data structure.
* updateUser: Updates the user’s information in the data structure.
* deleteUser: Removes an existent user from the data structure.
* getUserByUserName: Obtains user’s information of an existent user from the data structure.

#### Software Interfaces

The software would need to interface with a persistent data management system to get data from it. This service has two implementations. In one of them the connection will be a standard database connection using JDBC, while the other one, in order to simplify this example, will only manage data memory.

#### Communication Interfaces

The software would need to interface with a persistent data management system to get data from it. This service has two implementations. In one of them the connection will be a standard database connection using JDBC, while the other one, in order to simplify this example, will only manage data memory.

# Object Descriptions

## Objects

|  |  |
| --- | --- |
| **Class name:** UserServiceFactory | |
| **Brief description:** This class is responsible for creating the appropriate user service object that will be used. | |
| **Methods (operations)** | **Method Description** |
| getUserService  (UserServiceType); | This method returns the right UserService implementation based on the UserServiceType that it gets by parameter. |

|  |  |
| --- | --- |
| **Interface name:** UserService | |
| **Brief description:** This interface defines the operations that every UserService should implement. | |
| **Methods (operations)** | **Method Description** |
| addIUser(User) | Creates and add a new user into the appropriate data structure. |
| updateUser(String,User) | Updates the user’s information in the data structure. |
| deleteUser(String) | Removes a specified user from the data structure. |
| getUserByUserName() | Returns the user’s information of an existent user. |

|  |  |
| --- | --- |
| **Class name:** UserServiceMemoryOnly | |
| **Brief description:** This class is responsible for implementing the services provided for the UserService interface. This class uses a hash list to store the users. | |
| **Attributes (fields)** | **Attribute Description** |
| data:HashList<String,User> | List which represents the users stored. |
| **Methods (operations)** | **Method Description** |
| addIUser(User) | Inserts the new user into the hash list. Where the key will be its UserName. |
| updateUser(String,User) | Finds and updates the information of existent user. If it does not exist it will throw a UserNotFoundExeption. |
| deleteUser(String) | Removes an existent user from the hash table. If it does not exist it will throw a UserNotFoundExeption. |
| getUserByUserName(String) | Returns the value of the hash table from a given user name. If it does not exist it will throw a UserNotFoundExeption. |

|  |  |
| --- | --- |
| **Class name:** UserServiceDataBase | |
| **Brief description:** This class is responsible for implementing the services provided for the UserService interface. This class uses JDBC to connect with a data base in order to save user’s data. | |
| **Attributes (fields)** | **Attribute Description** |
| dbConnection:Connection | Link with the Data Base which stores user’s information. |
| pStmp:PreparedStatement | Statement which perform the queries over the Data Base. |
| **Methods (operations)** | **Method Description** |
| addIUser(User) | Inserts the new user into the data base. |
| updateUser(String,User) | Updates the information of existent user. |
| deleteUser(String) | Removes an existent user from the data base. |
| getUserByUserName(String) | Returns a user from the data base |

|  |
| --- |
| **Enumerator name:** UserServiceTypes |
| **Brief description:** This enumerator enumerates the different implementations of the UserService interface in order to be created by the factory |

|  |  |
| --- | --- |
| **Class name:** User | |
| **Brief description:** This class represents all the user’s information. | |
| **Attributes (fields)** | **Attribute Description** |
| username:String | User name that identifies the user. |
| firstname:String | First name of the user. |
| lastname:String | Last name of the user. |
| email:String | Email of the user. |
| password:String | Password account. |
| **Methods (operations)** | **Method Description** |
| getUserName(); | Gets username field |
| setUserName(String); | Sets username field |
| getFirstName(); | Gets firstname field |
| setFirstName(String); | Sets firstname field |
| getLastName(); | Gets lastname field |
| setLastName String (); | Sets lastname field |
| getEmail(); | Gets email field |
| setEmail(String); | Sets email field |
| getPassword(); | Gets password field |
| setPassword(String); | Sets password field |