



Berner Fachhochschule  
Haute école spécialisée bernoise

BACHELOR PROJECT

FALL TERM 2014

---

# Online Inspector for UniBoard

---

*Author:*  
Priya BIANCHETTI

*Supervisor:*  
Rolf HAENNI

January 16, 2015

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Present Situation . . . . .	5
1.2	Project Goal . . . . .	5
1.3	Contribution . . . . .	6
1.4	Report Structure . . . . .	6
<b>2</b>	<b>Technical Background</b>	<b>7</b>
2.1	Web Services . . . . .	7
2.1.1	JAX-WS . . . . .	8
2.2	JavaEE Web Application . . . . .	9
2.3	Certificates and Signatures . . . . .	10
<b>3</b>	<b>UniBoard</b>	<b>12</b>
3.1	Properties of UniBoard . . . . .	12
3.2	The Two Basic Operations of UniBoard . . . . .	13
3.2.1	Post Operation . . . . .	13
3.2.2	Get Operation . . . . .	14
3.3	UniBoard Inspector . . . . .	15
3.3.1	Data Queried by the Application . . . . .	15
3.3.2	Format of a Post . . . . .	17
<b>4</b>	<b>Requirements Analysis and Design</b>	<b>18</b>
4.1	Use Cases . . . . .	18
4.1.1	Use Case Diagram . . . . .	18
4.1.2	<b>UC 1:</b> View Post . . . . .	19
4.1.3	<b>UC 2:</b> Inspect Posts Using Basic Search . . . . .	19
4.1.4	<b>UC 3:</b> Inspect Posts Using Advanced Search . . . . .	19
4.1.5	<b>UC 4:</b> Inspect Posts Using Public Key . . . . .	20
4.2	Mock-ups . . . . .	21
4.2.1	UniBoard Inspector Dashboard . . . . .	21
4.2.2	Inspection Using Basic Search . . . . .	22
4.2.3	Inspection Using Advanced Search . . . . .	23
4.2.4	Inspection Using Public Key . . . . .	24
4.2.5	View Post . . . . .	25

<b>5</b>	<b>Implementation</b>	<b>26</b>
5.1	UniBoard Inspector Project Components . . . . .	26
5.2	Constructing a Query . . . . .	28
5.3	SSD System Sequence Diagrams . . . . .	29
5.3.1	SSD Basic Search . . . . .	29
5.3.2	SSD Advanced Search . . . . .	29
5.3.3	SSD Search by Public Key . . . . .	29
5.4	Page Flow Diagram . . . . .	33
5.5	Exception Handling and Testing . . . . .	34
5.6	Tools Used . . . . .	36
<b>6</b>	<b>Work Methods and Project Organization</b>	<b>37</b>
6.1	Project Plan . . . . .	38
6.2	Weekly Goals Achieved . . . . .	39
<b>7</b>	<b>Conclusion</b>	<b>40</b>
<b>8</b>	<b>Annexes</b>	<b>41</b>
8.1	UniBoard Inspector User Interfaces . . . . .	41
8.1.1	View Post Dialog . . . . .	41
8.1.2	Advanced Search Dialog . . . . .	42
8.1.3	Search by Public Key Dialog . . . . .	43
8.1.4	Basic Search Results Page . . . . .	44
8.2	JSON Messages . . . . .	45
8.2.1	JSON Message for the Post Candidate . . . . .	45
8.2.2	JSON Message for the Post ElectionData . . . . .	46
8.2.3	JSON Message for the Post Ballot . . . . .	47
8.2.4	JSON Message for the Post DecryptedVote . . . . .	48
8.2.5	JSON Message for the Post ElectionResult . . . . .	49

# Online Inspector for UniBoard

**Abstract.** UniBoard is a system that deals with the implementation of a public bulletin board where data can be stored. Currently, UniBoard offers a web service interface to post messages to the board and read previously posted messages. Data stored to the board are called posts. The nature of the posted messages may vary from being a clear text message to something much more complicated like a Base64 encoded message.

The goal of this project is to develop an independent web application that allows to inspect the content of the board. In order to do so the application communicates with UniBoard over the existing web service interface. A query is sent to the board and the corresponding results are obtained. The application allows a user to view the content of the board without any technical knowledge of the existing web service. It acts as a black box between the user and UniBoard. The technical complexity of building a query, communicating with UniBoard over a web service interface and displaying the results in a clear and convenient way is handled by the application. The application is a view-only application and does not allow modifications to be made to the content displayed.

# 1 Introduction

In this section, information concerning the actual situation of the UniBoard system, the goal of the project and its contribution to the existing system is provided.

## 1.1 Present Situation

UniBoard is implemented by the current e-voting projects, UniVote and UniCert. For example UniVote uses a public bulletin board to register information about an election, the ballots, the voters etc. Information posted to the board is not always in a readable format as it contains data that is encoded.

An author is a user who posts a message to the board and a reader is a user reading from the board. The existing web service interface called *UniBoardService* exposes two principle operations, the Post and the Get operations that allow to post messages to the board and read previously posted messages respectively. All communication with UniBoard is done over the existing web service interface.

## 1.2 Project Goal

The work of this project includes the implementation of an application to inspect the content of the board. The application fulfills the following properties:

- It is a web service client application that consumes the services of the existing web service interface. More specifically, it invokes the GET operation of the web service in order to read from the board.
- It provides a web user interface for users to enter the search criteria.
- It provides different search methods in order to filter out the posts from less restrictive to more restrictive levels.
- It queries the board for content matching the search criteria and displays the results of the search in a viewer.

### **1.3 Contribution**

The development of the web application possessing the above mentioned properties will allow users to monitor what is happening on the board. Imagine that you posted a message to the board and now you would like to see if the board contains your message.

The option available at present is to look through all the existing posts on the board until you find your posted message. Such an approach is time consuming and not optimal. This application will allow users to inspect the content of the board in a easy and user friendly way. The results will be displayed in a clear and convenient way for the users.

### **1.4 Report Structure**

The sections in this report are organized in a way to firstly familiarize the reader about the technologies used, followed by providing information about UniBoard and the application to be developed UniBoard Inspector. Later, the artifacts concerning the design of the application and its implementation is provided. To terminate the report, the working methods and the organization of the project is provided, followed by a conclusion and scope for future development.

## 2 Technical Background

In this section, the basic knowledge that is required in order to understand the technologies used in the project is presented. You may skip this section if you are familiar with the concept of web services, JavaEE web applications, cryptographic certificates and digital signatures.

### 2.1 Web Services

Web services allow the communication between two devices over the Internet. The idea of web services relies on the re-usability of programs and easy updates whenever changes are made. It is a machine to machine interaction where one side acts as a client and the other side acts as a service provider.

A web service has an interface which is in a machine-readable format that describes the rules for communication. This interface is described in a WSDL (Web Service Description Language) file which is a XML format to describe the services provided by a web service. The services are described as a collection of network endpoints or ports. A client connecting to the web service may determine the operations available on the server from the WSDL file.

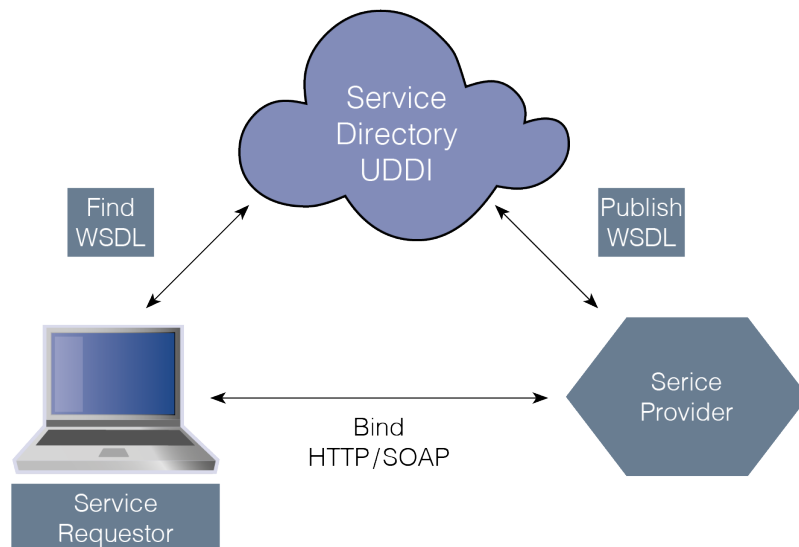


Figure 1: Providing and Consuming Web Services [1]

The service provider sends the WSDL file to a directory called UDDI (Universal Description, Discovery and Integration). The service requester contacts the UDDI to find out the services provided and then contacts the service provider using the SOAP (Simple Object Access Protocol) protocol over HTTP. [2] [3]

The application developed as a work of this project is a JavaEE web application. The two types of JavaEE web service technologies are JAX-WS (Java API for XML Web Services) and JAX-RS (Java API for RESTful Web Services). The web service provider UniBoardService has been set up using JAX-WS technology and is the focus of the following subsection.

### 2.1.1 JAX-WS

In JAX-WS technology the web services and clients communicate using XML. The calls to the operations of the web service by the client and the responses are transmitted by an XML-based protocol called SOAP (Simple Object Access Protocol). These SOAP messages are transmitted over HTTP.



Figure 2: Communication between a JAX-WS Web Service and a Client [4]

With JAX-WS, the application developer does not have to deal with the complexity of SOAP messages. The JAX-WS runtime does the conversion of API calls and responses to and from SOAP messages. The web service operations are specified by defining methods in a Java class or interface. A WSDL generating tool exposes these methods from the class as a web service. [2] [4]



## 2.2 JavaEE Web Application

The JavaEE platform provides a multi-tiered approach for implementing services where the application logic is divided into the tiers. The client tier components run on the client machine. The web and the business tier components run on the JavaEE server. The information system tier runs on the database server. The figure given below shows the different components contained in the different tiers.

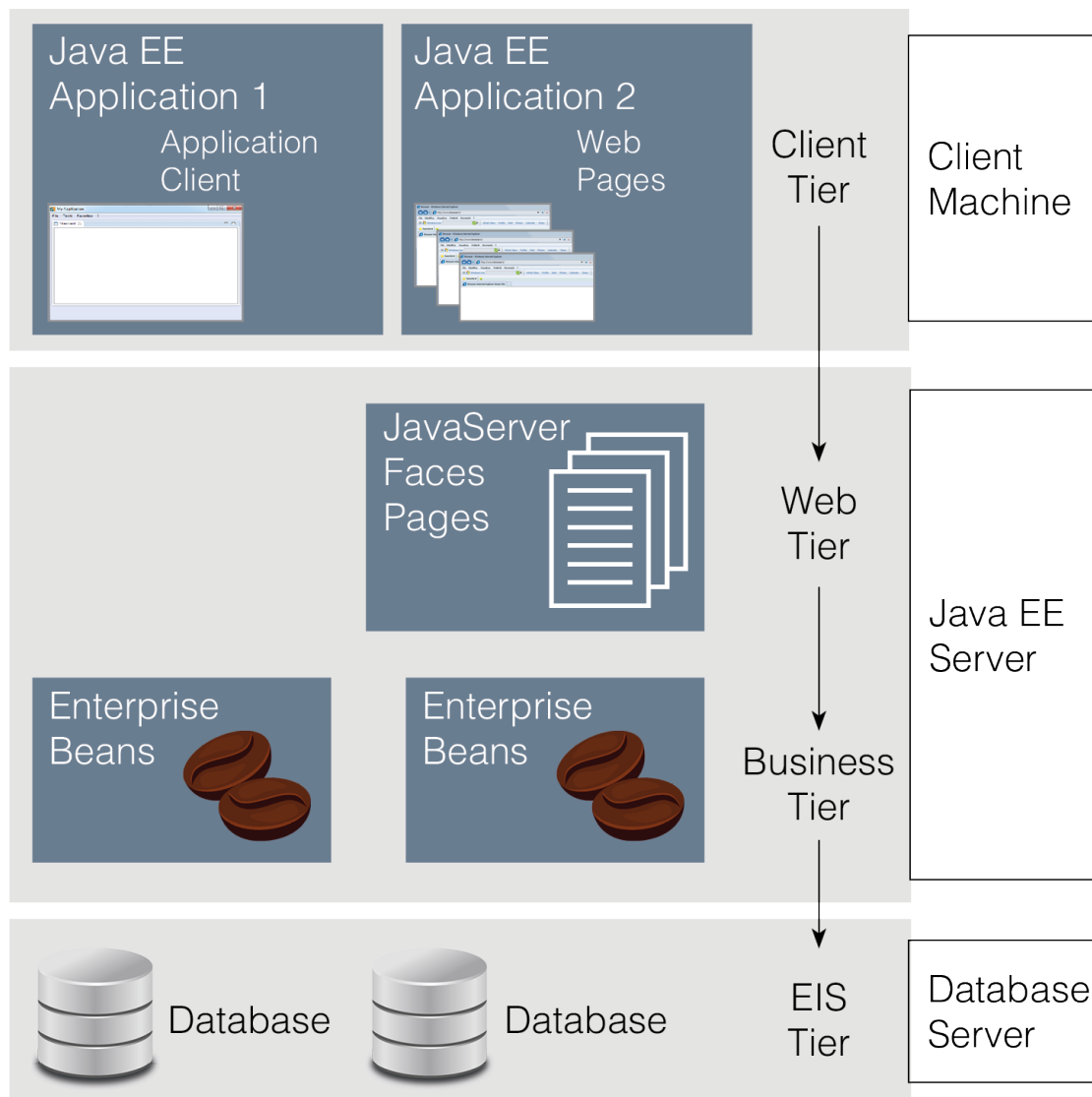


Figure 3: Multitiered Applications [5]

The tiers of interest to this project is the web and the business tiers. The web tier

contains the JSF (Java Server Faces) pages that is a Java specification for building user interfaces for web applications.

The business tier contains the Enterprise Beans where the logic of the application lies. An Enterprise Bean receives and processes data from client programs. It can also retrieve data and send it back to the client. [5] [6]

### **2.3 Certificates and Signatures**

In this section, a brief explanation about cryptographic certificates and digital signatures is provided. A digital certificate is a form of identification in the digital world just like passports and driver's licenses are in the real world. Just like a document that is signed with a handwritten signature, in the digital world a digital signature is used to attest the true identity of an entity.

Firstly, in order to understand digital certificates one needs to understand the concept of public key cryptography. A person or entity who owns a digital certificate basically owns a key pair consisting of a public key and its corresponding private key. The private key can be considered as a secret that is exclusive to the owner of the certificate. The public key is known by everyone. The question that arises is, how does one obtain these key pairs i.e a certificate?

A certificate can be obtained from a certification authority (CA) that provides you the key pair. The key pair is generated by a series of mathematical calculations. You may refer to the RSA algorithm for any further information on generating key pairs. An entity that wishes to obtain a certificate provides information about itself to the CA. An entity may represent a client or a server. The CA verifies this information and issues a certificate with the public and private key pair.

The aim of having a certificate is to secure the communication between two communicating parties through encryption and to prove the authenticity of the communicating parties by signing the messages sent. Both encryption and digital signatures are mathematical operations that slightly differ from one another with respect to the parameters used. When two parties wish to communicate by sending encrypted messages, the sender encrypts the message using the public key of the receiver. The receiving end decrypts the message using his private key. Hence, the sender can be sure that the message can be read only by the intended receiver, since only he possesses the private key to decrypt

the message.

But to what extent can one trust a certificate? When one obtains a certificate the CA signs this certificate using its private key. The signing of the certificate by the CA enables to verify the authenticity of the certificate. One can verify using the public key of the CA if the certificate presented is indeed a certificate issued by the claimed CA. This is important since one can trust a certificate only to the point that one can trust the issuing certification authority. Some examples of trusted CA's are VeriSign, GeoTrust etc. [7] [8]

## 3 UniBoard

UniBoard is a secure public bulletin board where data can be stored. A message that is posted to the board needs to follow the so called posting properties in order for it to be considered as a valid post. These posting properties define the structure of the messages posted to the bulletin board. [9]

The bulletin board is sectioned and grouped. It is access controlled, ordered, chronological, append-only and allows certified posting. Each of these properties is explained below.

### 3.1 Properties of UniBoard

In this section, the posting properties are explained in detail.

- **Sectioned:** The bulletin board consists of multiple sections where related messages are grouped together. These sections are equally shaped in structure. When posting a message, the author has to provide the section to which the post belongs.
- **Grouped:** A section consists of multiple groups. When posting a message, the author has to provide the group to which the message belongs. Messages belonging to the same group are similar in shape and content. Every section has the same set of group.
- **Typed:** In a grouped bulletin board, each group defines the type of the message to be stored in the group. Messages of the defined type only are accepted and all others are rejected.
- **Access Controlled:** In order to authorize only certain users to post messages to the board, a message to be accepted by the board must be signed. The signature and the public key must be included in the post. The set of authorized public keys is known to the board and access is controlled by verifying the signature.
- **Ordered:** A sequence number called the rank is added by the board such that all the published posts have a total order.
- **Chronological:** The board adds a time-stamp to every message posted to the board in order to denote when the message was received by the board.
- **Append-Only:** In order to ensure that no posted message is removed from the board, the board creates a hash value which is a function of the preceding post and other parameters. This value is added to the current post.

- **Certified Posting:** Upon successful posting of a message, the board creates a signature and adds it to the post. The user receives a receipt from the board. [9]

## 3.2 The Two Basic Operations of UniBoard

UniBoard is a public bulletin board that is sectioned and grouped. It is typed, access controlled, ordered, chronological and allows certified posting. Review the previous section for a detailed explanation of these posting properties if necessary. Although in this project we do not deal with the post operation, in order to read from the board it is important to know the structure and the nature of the information posted or published to the bulletin board. The following sections describe the Post and the Get operations that allow to publish and read the content of the bulletin board respectively. [9]

### 3.2.1 Post Operation

Data can be published to the bulletin board by calling the Post operation of the web service interface UniBoardService. Messages posted on the board need to follow the posting regulations for UniBoard in order for it to be considered as a valid message.

In order for a post to be valid, an author posting a message to UniBoard needs to provide four other user attributes along with the message. These user attributes constitute the following posting properties: Sectioned, Grouped, Typed and Access Controlled. The message is received by UniBoard which in turn adds four board attributes to the message. These board attributes constitute the following posting properties: Ordered, Chronological, Append-Only and Certified Posting.

Hence a valid post consists of a message, four user attributes and four board attributes. A brief specification of the user and board attributes follows below.

The user attributes consists of the following information:

- **The Section:** UniBoard defines a set of available sections. When posting a message the author indicates the section for the message.
- **The Group:** Messages belonging to a group are similar in nature. The author indicates the group for the message when posting it.
- **The Type:** Each group defines its own set of valid messages. A message of type T and group G is valid if the type T belongs to the set of valid messages for the group G.

- The Signature: The signature serves to control the access to the bulletin board. In order to identify the author of a message, a digital signature is provided and the corresponding public key.

The board attributes consists of the following information:

- The Order: A sequence number is added to the post so that all the messages published on the board have an order.
- The Time-stamp: A time-stamp is added to the post to denote when the message was received by the board.
- The Hash: In order to ensure that no previously posted messages can be removed or changed a hash value is created which is a function of the previous post and other parameters.
- The signature: All user and board attributes including the message is signed and this signature is added to the post. [9]

### 3.2.2 Get Operation

The Get operation of the web service interface UniBoardService, allows to send a query to the board and obtain the resulting posts that corresponds to the query. All messages published on the bulletin board can be retrieved using the get operation. What is a query and how is it constructed? A query Q is an object that represents the constraints in the following format:

Q = Get posts WHERE constraint1 AND constraint2

A constraint or a search criteria can be constructed using the message, user attributes and board attributes. In the case of UniBoard, a query that is sent to the board specifies additional properties such as the order and the number of results to be returned. [9]

### 3.3 UniBoard Inspector

Now that you are familiar with the concept of UniBoard, its operations and the posting properties of the data stored to the board, lets proceed to see the form of data that is queried by the application UniBoard Inspector and the format of the posts that are stored to UniBoard.

#### 3.3.1 Data Queried by the Application

UniBoard Inspector queries the board for information posted to the board in the context of e-voting. The user can inspect the board for information concerning an election that is being held on-line. Lets consider elections that are held in different universities in Switzerland. In order to set up an election one first defines information about an election. Followed by defining the voters and the candidates in any order. Once the election is open, the voters may cast their votes. These votes are encrypted before being posted to the board. The votes are decrypted and finally the result of the election is obtained.

The figure given below describes the order in which the information for an election is posted to the board.

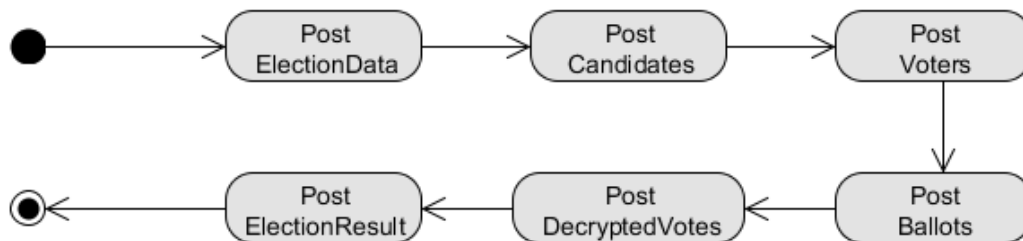


Figure 4: Election Cycle

As mentioned earlier UniBoard is a sectioned and grouped bulletin board. In order to put the above information of an on-line election, one would consider the different universities to be the sections and the different states of the election process to be the groups.

Lets consider the following three universities: Bern University of Applied Sciences BFH, University of Bern Unibe and University of Zurich UZH. These universities form the three

sections. The different groups will be formed by the information concerning an election i.e, ElectionData, Candidates, Voters, Ballots, DecryptedVotes and ElectionResult.

Recall that a Post contains a message, Alpha attributes and Beta attributes. In the following figure, the different groups are represented as object types in order to show the content of the messages belonging to the different groups.

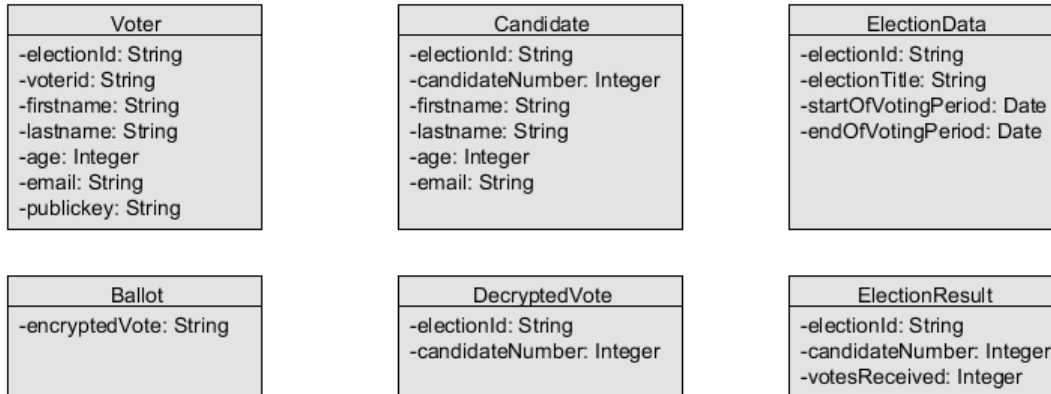


Figure 5: Object Type Representation of the Groups

The following figure shows a graphical representation of a structured bulletin board with the three sections i.e, BFH, Unibe and UZH and the six groups i.e, ElectionData, Candidates, Voters, Ballots, DecryptedVotes, ElectionResult.

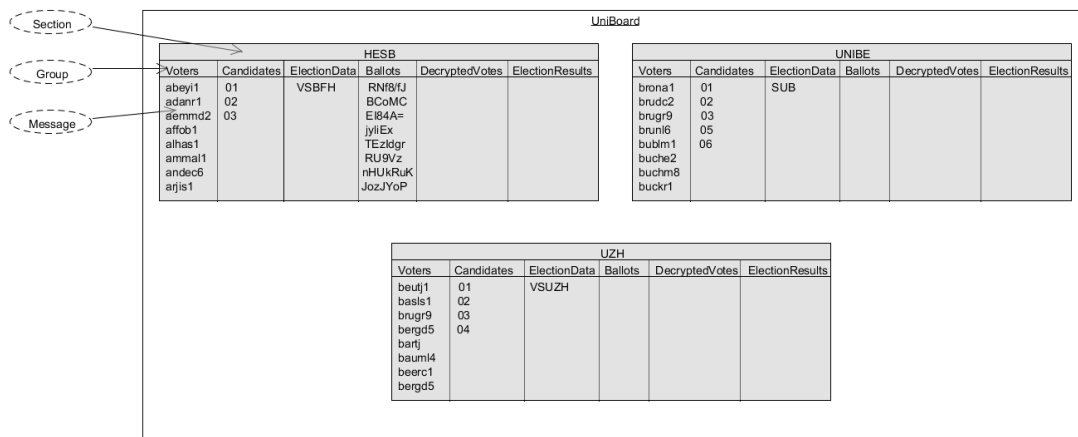


Figure 6: UniBoard: A Sectioned and Grouped Bulletin Board.



### 3.3.2 Format of a Post

A post is a JSON (JavaScript Object Notation) message that stores all information as attribute-value pairs. A post contains a message, user attributes and board attributes. The JSON message for a post belonging to the group Voters is presented below:

```
{
  "message": {
    "electionid": "VSBFH",
    "voterid": "gab2@bfh.ch",
    "firstname": "Gabriel",
    "lastname": "Honnegger",
    "age": 23,
    "email": "Gab2@student.bfh.ch",
    "publickey": "nZvbmJlcmd1bkBiZmguY2gWDxxzQhm1RAM\naKumjBW7"
  },
  "alpha": {
    "section": "BFH",
    "group": "Voters",
    "signature": "0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAXIjAgBgNVBAMMGXBoaWx1bW9uLnZvbmJlcmd1bkBiZmguY2gxEzAR\nBgNVBAQMCnZvb1BCZXJnZW4xFDASBgNVBCoMC1BoalwzDg8Kpbw9uMQ8w",
    "key": "RNf8/fJqfB1v27VGxWDxxzQhm1RAM\naKumjBW7z1YnjcYJ5By3ptSLNQ1jct00vh6b0sQ"
  },
  "beta": {
    "timestamp": ISODate("2014-10-21T20:00:00.000Z"),
    "rank": 1,
    "boardSignature": "iBCZXJnZW4xFDASBgNVBCoMC1BoalwzDg8Kpbw9uMQ8wDQYDVQQK\nnDAZiZmguY2gwgEgEfMA0GCS"
  }
}
```

Figure 7: JSON Message for Voters.

A list of such JSON messages are stored to the board for each of the different sections and groups. Querying the board for posts involves providing values for the different search parameters. For example, some of the possible search parameter names are section, group, key, time-stamp, rank etc.

The JSON messages for posts belonging to the different groups can be found in the Annexe section.

## 4 Requirements Analysis and Design

The aim of requirements analysis and design is to have a well reflected plan at the beginning stage of the project that will help in the implementation of the project. The requirements analysis is presented with the help of use cases that define a step by step procedure to obtain the functionalities desired by the application. The design is presented with the help of mock-ups that provide a conceptual solution fulfilling the requirements.

### 4.1 Use Cases

The use cases help representing the different operations that is supported by the system and the interaction of the user with the system. In the following, the use case diagram is presented followed by the description of the use cases.

#### 4.1.1 Use Case Diagram

The use case diagram shown below identifies the different search methods that is supported by UniBoard Inspector and other functionalities.

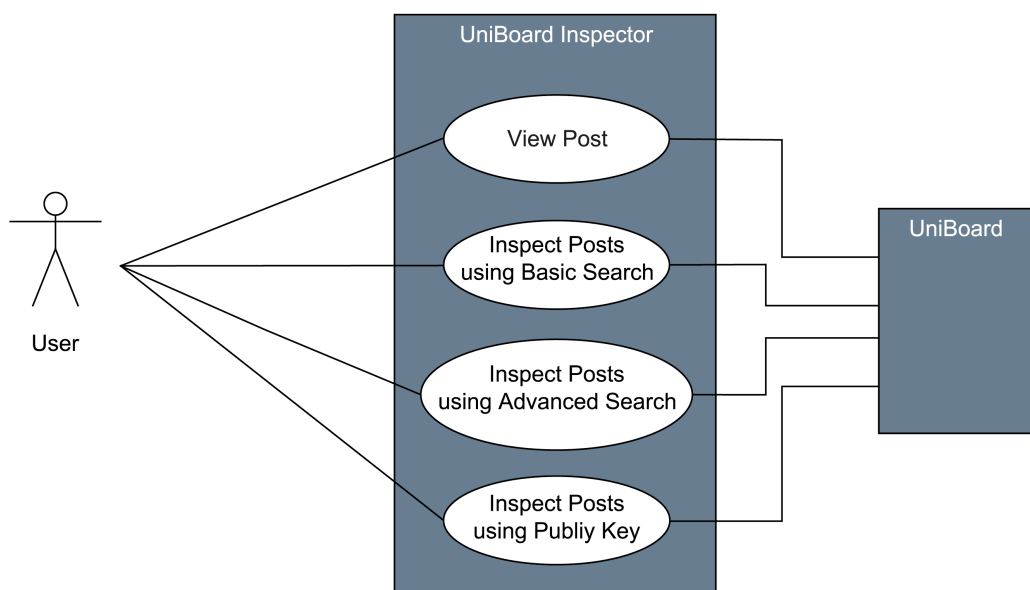


Figure 8: Use Case Diagram UniBoard Inspector

In the following you will find the description of the use cases for each of the search methods. Only the main success scenario is documented.

#### **4.1.2 UC 1: View Post**

**Actor:** User.

**Preconditions:** A list of posts are displayed in the viewer.

**Postconditions:** The user views the details of a post.

##### **Main Success Scenario**

1. The user interacts with a component of the application.
2. The details of the post which contains the section, group, rank, timestamp, public key, board signature etc is displayed.

#### **4.1.3 UC 2: Inspect Posts Using Basic Search**

**Actor:** User.

**Preconditions:** The name of the section to be queried must be chosen.

**Postconditions:** The resulting posts are displayed in the viewer.

##### **Main Success Scenario**

1. The user provides the search parameters to query the board.
  - a) He provides the name of the section for which he would like to view all the posts.
  - b) He provides the name of the group in the section for which he would like to view the posts.
  - c) He provides a time interval to view all the posts that were inserted during this time period.
  - d) He limits the number of results displayed in the viewer. If not a default limit of 50 is chosen.
2. The posts corresponding to the query are obtained and displayed in the viewer.

#### **4.1.4 UC 3: Inspect Posts Using Advanced Search**

**Actor:** User.

**Preconditions:** One or more section names to be queried must be chosen.

**Postconditions:** The resulting posts are displayed in the viewer.

##### **Main Success Scenario**

1. The user provides the search parameters to query the board.

- a) He provides one or more section names for which he would like to view all the posts.
  - b) He may provide one or more groups belonging to these sections for which he would like to view the posts.
  - c) He may provide a time interval to view all the posts that were inserted during this period.
  - d) He may provide rank values to obtain all posts that are either equal to, less than, more than or between the values.
  - e) He may provide a public key in order to obtain all the posts that were inserted by himself using this public key or any other person to whom the key belongs to.
  - f) He may limit the number of results displayed in the viewer. If not a default limit of 50 is chosen.
2. The posts corresponding to the query are obtained and displayed in the viewer.

#### **4.1.5 UC 4: Inspect Posts Using Public Key**

**Actor:** User

**Preconditions:** None.

**Postconditions:** The resulting posts are displayed in the viewer.

##### **Main Success Scenario**

1. The user provides his public key to search for posts.
2. He may limit the number of results displayed in the viewer. If not a default limit of 50 is chosen.
3. All posts that were posted using this public key is displayed in the viewer.

## 4.2 Mock-ups

In this section, the mock-ups for UniBoard Inspector is presented. The mock-ups correspond to the different search methods and functionalities presented in the use cases.

### 4.2.1 UniBoard Inspector Dashboard

The dashboard is the start page of the application. The dashboard consists of a title bar, a side menu where the basic search parameters can be chosen to inspect the board and a centrally placed viewer where the top 50 most recent posts that were posted to the board is displayed.

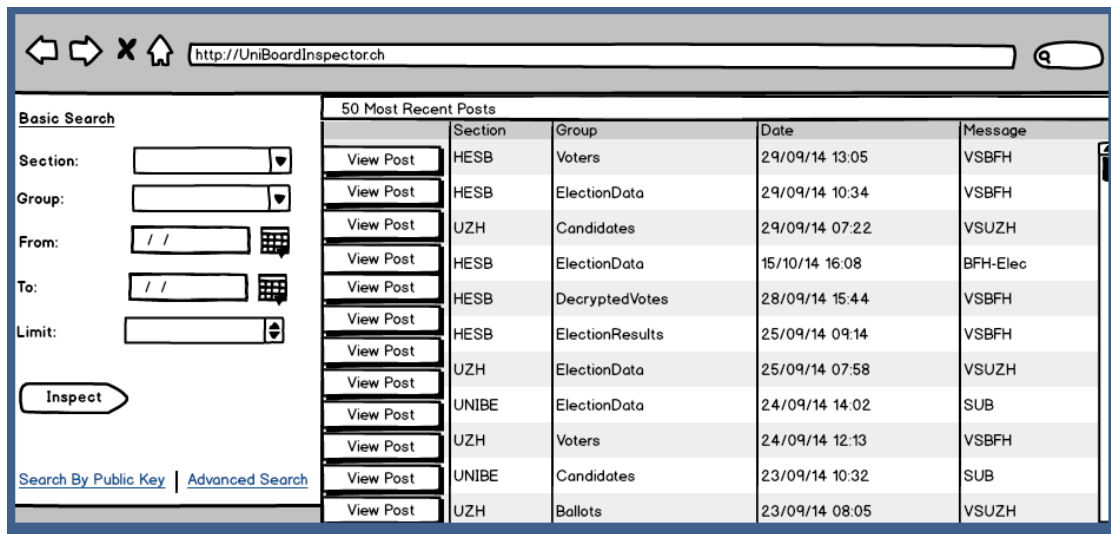


Figure 9: UniBoard Inspector Dashboard

#### 4.2.2 Inspection Using Basic Search

Below is the mock-up that shows the results of a search. The basic search parameters are chosen by selecting a section, group, date and limit. The results of the query that is sent to the board using the basic search parameters is displayed in the viewer.

The screenshot shows a web browser window with the URL `http://UniBoardInspector.ch`. The page is titled "Basic Search" and displays search results for "20 Most Recent Posts from Section HESB".

**Search Parameters:**

- Section: HESB
- Group: ElectionData
- From: 20/09/2014
- To: 30/10/2014
- Limit: 20

**Search Results Table:**

Rank	Date	Election Id	Election Title	Start of Voting Period	End of Voting Period
1	25/09/14 09:14	VSBFH	IT Elections 2014	12/11/2014 08:00	30/01/2015 24:00
2	23/09/14 10:32	BFH-Elec	Chairman Election	01/11/2014 08:00	30/12/2014 24:00

The interface includes an "Inspect" button and links for "Search By Public Key" and "Advanced Search".

Figure 10: Basic Search Results Page

### 4.2.3 Inspection Using Advanced Search

The advanced search provides more restriction compared to the basic search and allows the user to filter the posts more selectively.

By clicking on the Advanced Search link a search dialog box is opened where the advanced search parameters can be chosen. A list of sections and groups can be chosen. The posts can be filtered further by providing a time interval, rank values and public key. The number of results can be limited by choosing a desired value for the limit.

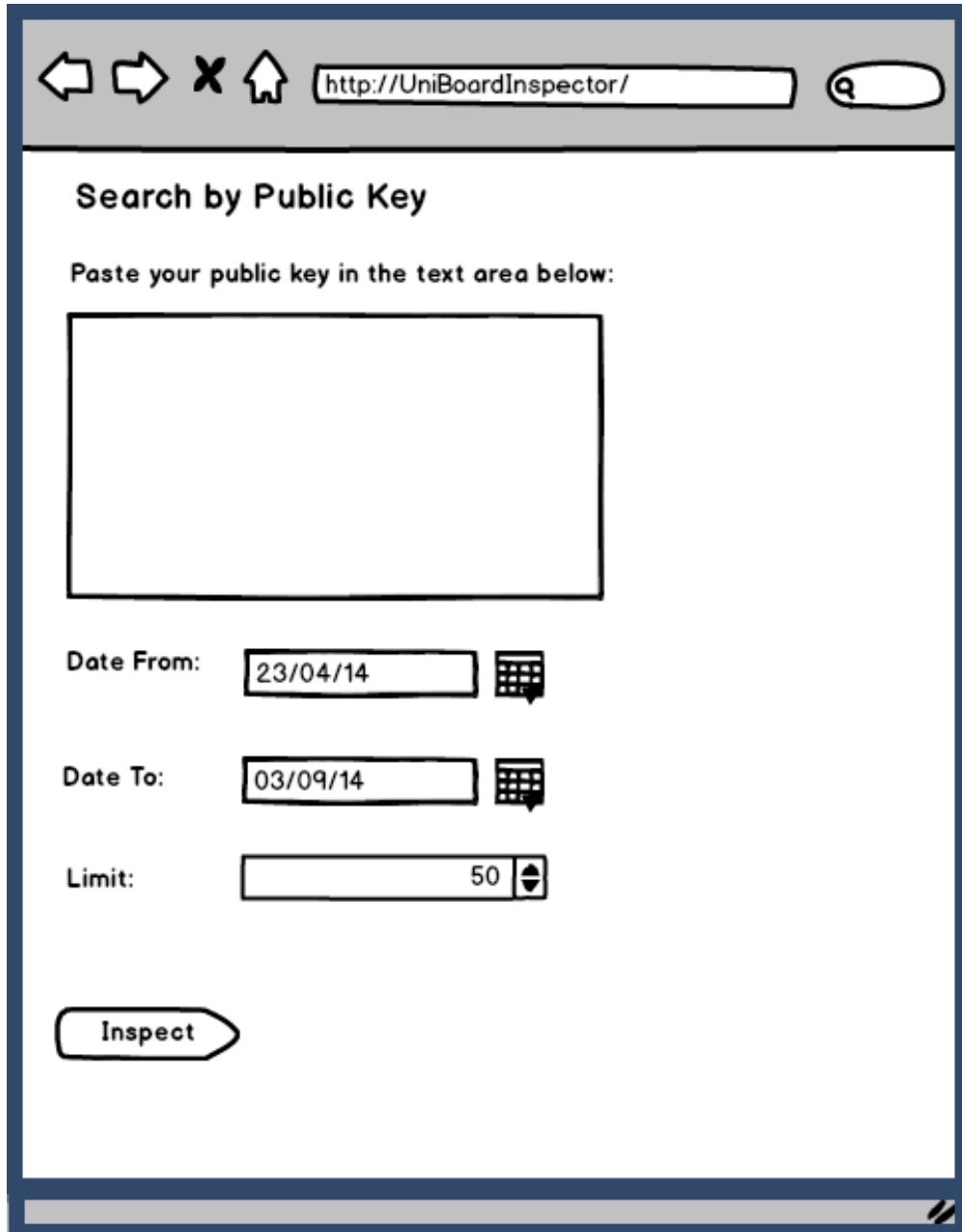
The screenshot shows a web browser window with the address bar containing `http://UniBoardInspector/advancedsearch`. The main content area is titled "Advanced Search" and contains the following fields and controls:

- Section:** A dropdown menu with options "In", "All", "In", and "Equals". To its right is a list of checkboxes:  HESB,  UNIBE, and  UZH. Further right is a "Selected sections:" list containing "HESB" and "UZH".
- Group:** A dropdown menu with options "Equals", "All", "In", and "Equals". To its right is a list of checkboxes:  ElectionData,  Ballots,  Voters,  DecryptedVotes,  Candidates, and  ElectionResult. Further right is a "Selected groups:" list containing "Candidates" and "ElectionResult".
- Date:** A dropdown menu with options "Between", "From", "Between", and "Before". To its right are two date input fields: "23/04/14" and "03/09/14".
- Rank:** A dropdown menu with options "Equal To", "Less Than", "More Than", and "Between". To its right are two numeric input fields: "25" and an empty field.
- Public Key:** A text input field containing "certificate.pem" and an "Upload Certificate" button.
- Limit:** A numeric input field containing "50".
- Inspect:** A button located at the bottom left of the dialog box.

Figure 11: Advanced Search Dialog Box

#### 4.2.4 Inspection Using Public Key

The user can search for posts by copying and pasting the public key in a text area. This allows a quick search for all posts that were posted using a public key.



The image shows a web browser window with the address bar containing 'http://UniBoardInspector/'. The main content area is titled 'Search by Public Key'. Below the title, there is a prompt: 'Paste your public key in the text area below:'. This is followed by a large, empty rectangular text input field. Below the text area, there are three rows of form controls: 'Date From:' with a text box containing '23/04/14' and a calendar icon; 'Date To:' with a text box containing '03/09/14' and a calendar icon; and 'Limit:' with a text box containing '50' and a spinner control. At the bottom left of the form area, there is a button labeled 'Inspect'.

Figure 12: Search by Public Key Dialog Box



#### 4.2.5 View Post

The user may click on the View Post button in order to view the details of the selected post. The entire message including the user attributes and the board attributes of the post can be viewed.



Figure 13: The details of a post

## 5 Implementation

In this section, the artifacts supporting the realization of the web application UniBoard Inspector is presented. The different implementation details and the outcomes are presented with the help of Class Diagrams, System Sequence Diagrams, Domain Models, Page Flow Diagrams etc. In the following sub-section the structure of the project is presented and the details of the project are further deepened as you continue to the other sections.

### 5.1 UniBoard Inspector Project Components

UniBoard Inspector is a JavaEE web application. Hence it is a multi-tiered application. The Java Server Faces pages are used to build the user interfaces for UniBoard Inspector. The web tier contains the JSF pages. The search parameters entered by the user are assigned to variables in the enterprise bean classes where validation is done if necessary. The enterprise beans along with other classes contains the business logic of the application.

The application communicates with UniBoard over a web service interface that is built using the JAX-WS technology(Java API for XML Web Services). Refer to the figure in the next page.

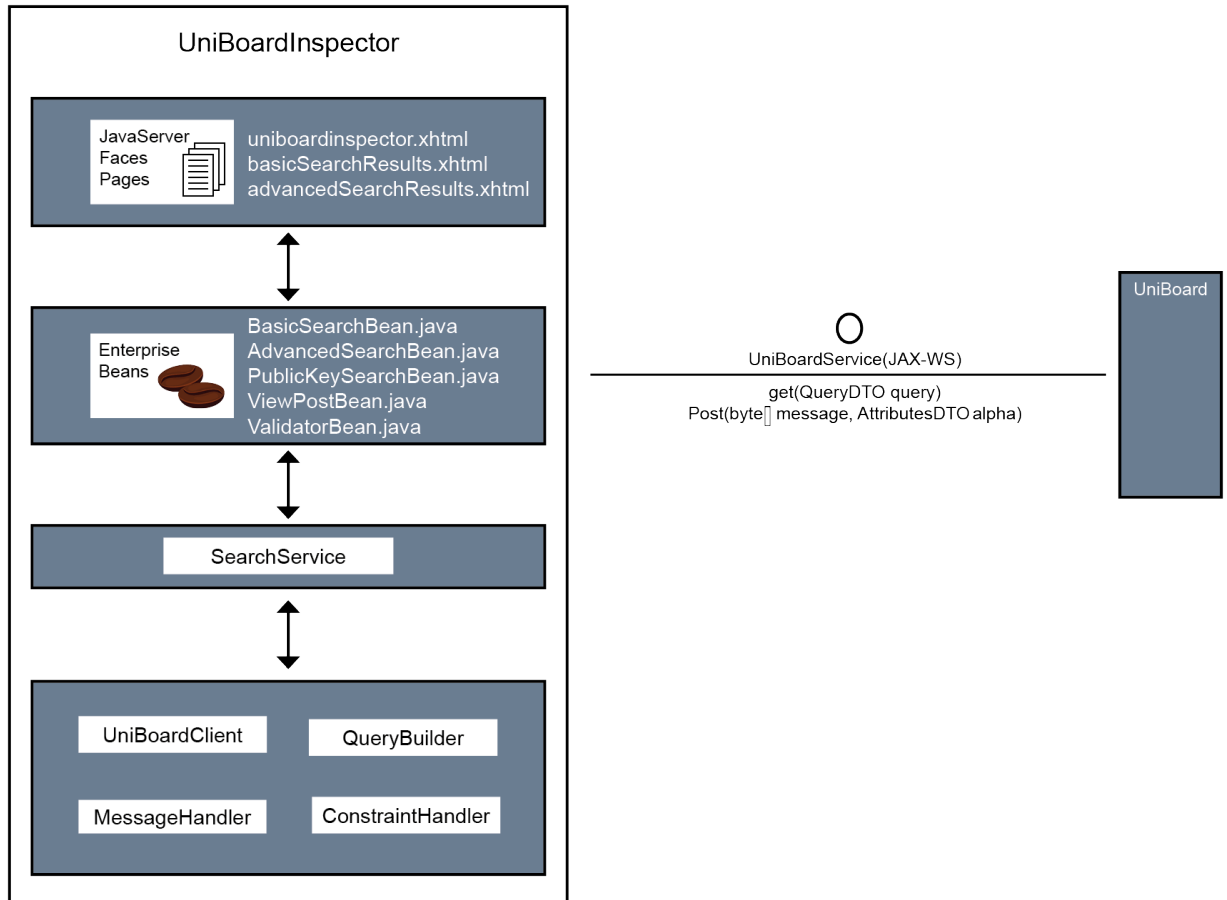


Figure 14: Detailed Overview of Systems

## 5.2 Constructing a Query

The domain model below describes the classes that are used to build a Query object. The Query object is sent to UniBoard using the get operation of the web service interface UniBoardService. The list of posts corresponding to the query is obtained and displayed in a viewer by UniBoardInspector.

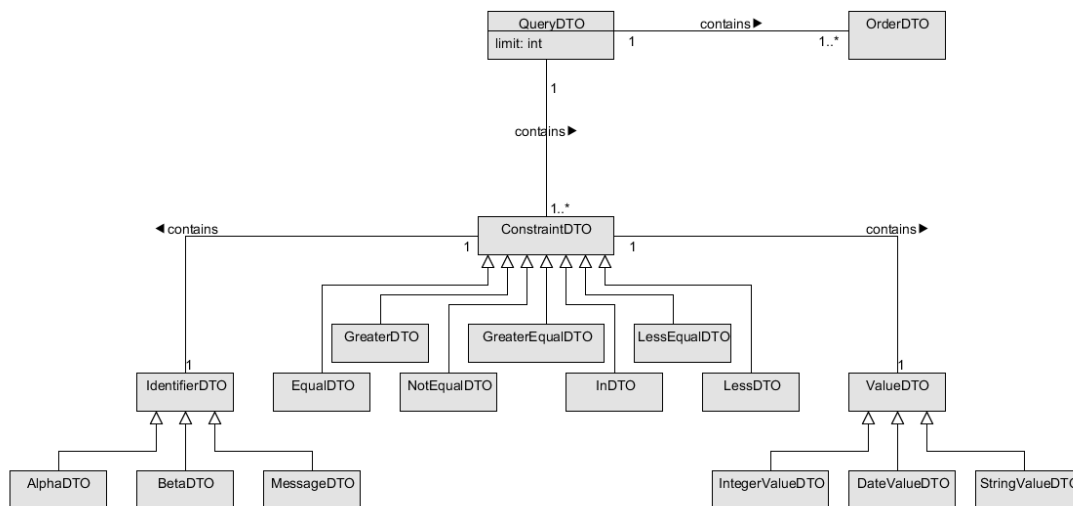


Figure 15: Domain Model

A query is made up of one or more constraints, an order and a limit. A Constraint object type is made up of an Identifier object type and a corresponding Value type assigned to the Identifier.

The Constraint type can be an Equal To, Less Than, More Than etc constraint type. The Identifier type can be an Alpha, Beta or Message Identifier type. Recall that an Alpha Identifier contain information like the section and group to which a message belongs to. A Beta Identifier contains information added by the board like the timestamp and the rank of a post in the given group. The Value types can be an Integer, String, Date etc Value types.

Hence, a Constraint such as the following, *ConstraintA: The name of the section is BFH*, will be constructed using an Alpha Identifier object and a String value type that contains

the name of the section(i.e. BFH). The constraint type itself is an EqualTo object type.

A query consists of one or more constraint objects, an Order object and a Limit. A query such as the following: *Get all posts where ConstraintA, Order them by the most recent and show only 50 of the results* will be constructed using a Constraint object, an OrderBy Date object and a Limit which is an Integer with value 50.

### **5.3 SSD System Sequence Diagrams**

The System Sequence Diagrams describes how various classes interact with each other in order to execute a functionality of the application. The user can inspect the content of the board by using different search methods. In this section, the System Sequence Diagrams for the available search methods i.e, Basic Search, Advanced Search and Search By Public Key is presented. Below you will find a brief introduction for each of the search methods followed by the three system sequence Diagrams.

#### **5.3.1 SSD Basic Search**

In order to inspect the board using Basic Search, the user must provide a section for which he would like to view all the posts. He optionally provides a group, a time period to view the messages posted during this period and limits the number of results that is returned by the board.

#### **5.3.2 SSD Advanced Search**

The user provides one or more sections for which he would like to inspect the board. He provides groups, a rank interval, a time interval, the public key etc. The advanced search provides a wider range of choices for the search parameters.

#### **5.3.3 SSD Search by Public Key**

The user provides the public key for which he would like to inspect the board in order to view all the posts corresponding to this key. He optionally provides a date and time interval and limits the number of results that is displayed.

The implementation of these three is shown with the help of the sequence diagram given below.

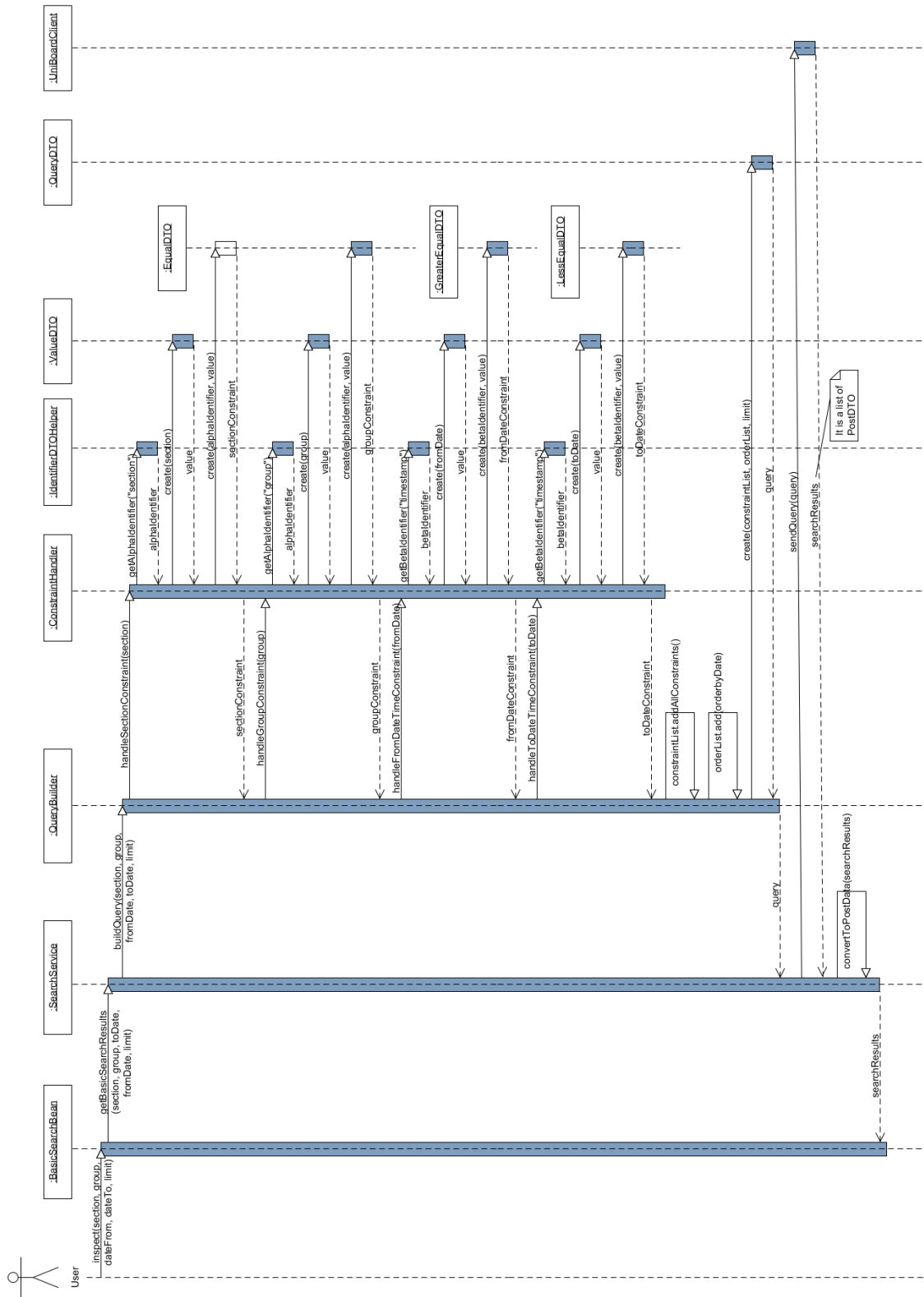


Figure 16: SSD for Inspection Using Basic Search

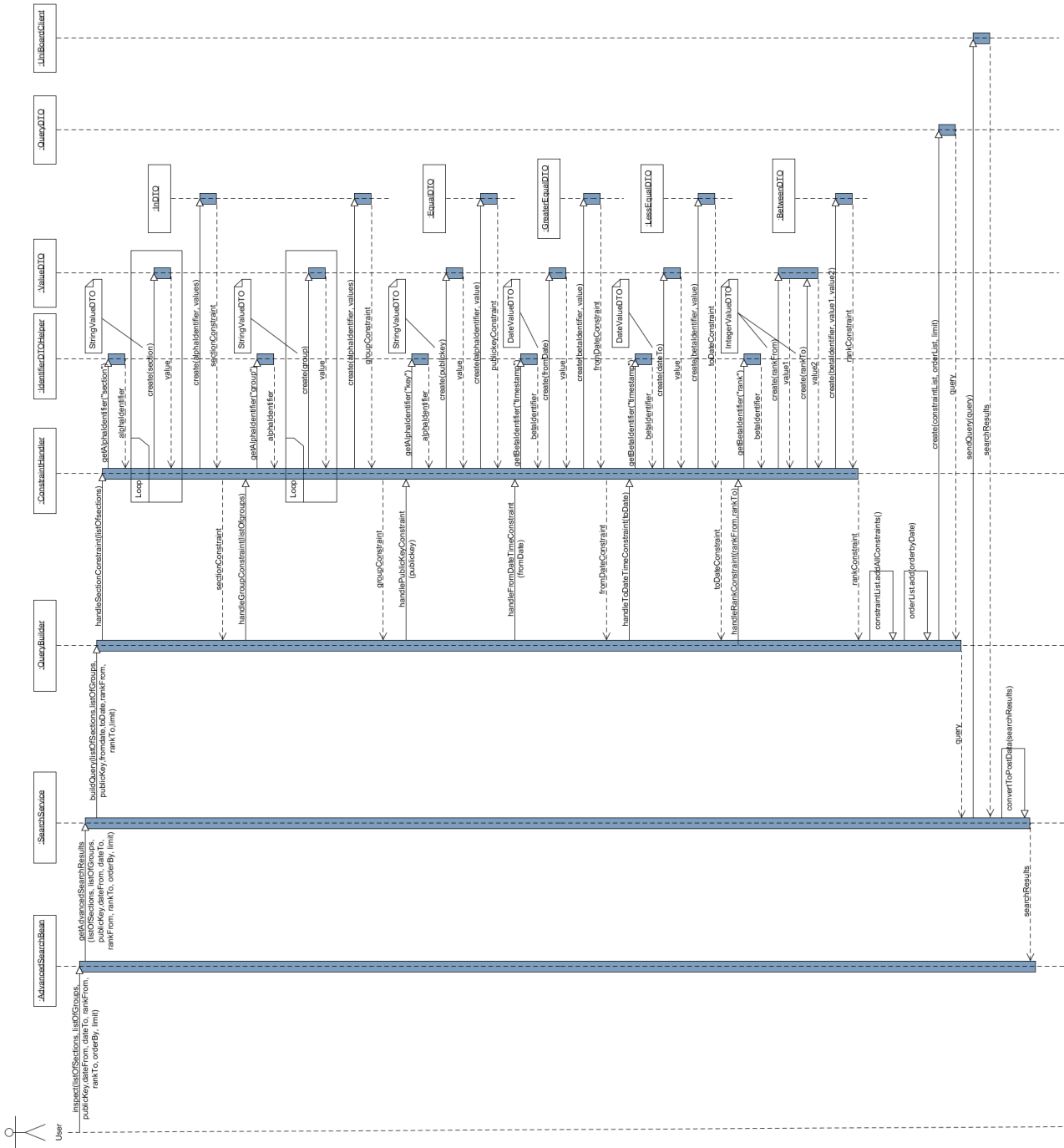


Figure 17: SSD for Inspection Using Advanced Search

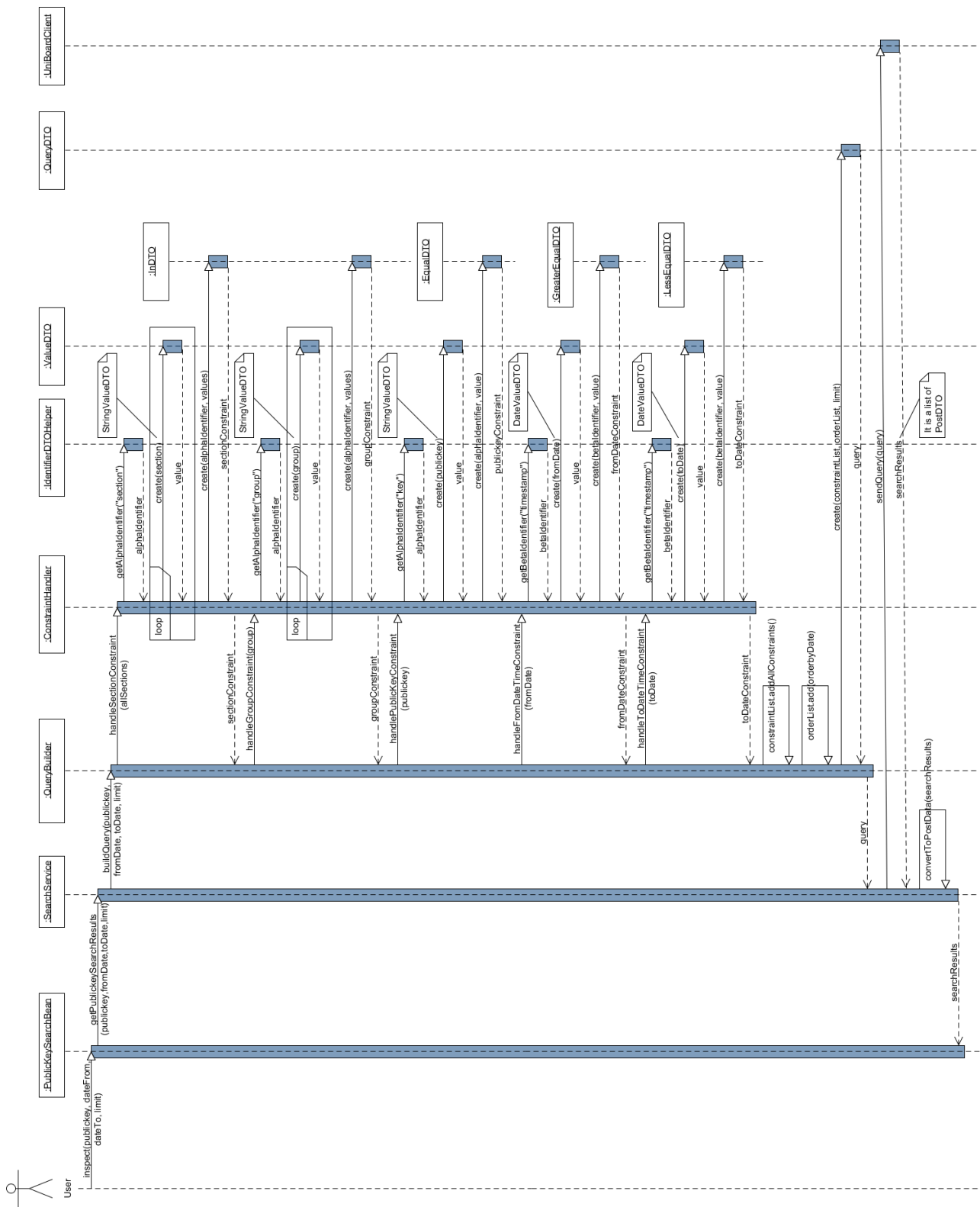


Figure 18: SSD for Inspection Using Public Key



## 5.4 Page Flow Diagram

The page flow diagram of the application UniBoard Inspector is shown below. The arrows show the interaction of the user with the application, such as clicking on a button or a link. The blue ovals show the different pages that are displayed by the application.

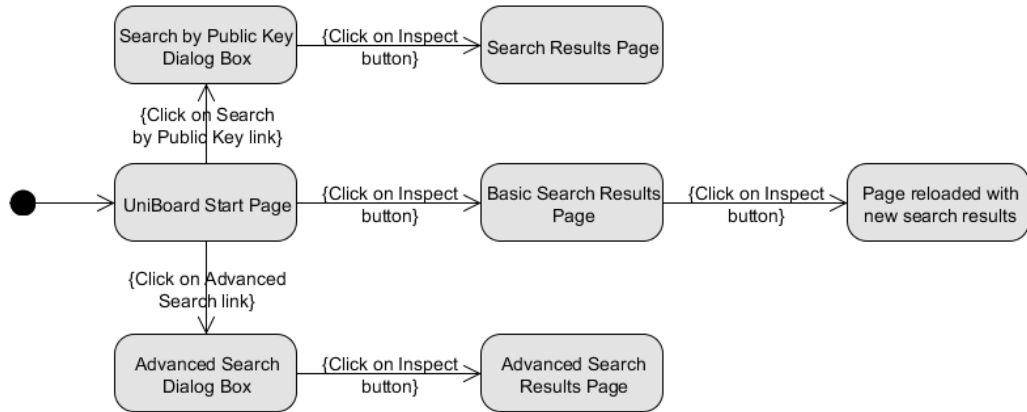


Figure 19: Page Flow Diagram for UniBoard Inspector

The home page of the application UniBoard Inspector is shown below. Screen shots of the remaining pages are included in the Annexe section.

The screenshot shows the UniBoard Inspector interface. On the left, there are search filters: Section (dropdown), Group (dropdown), From Date, To Date, and Limit (set to 50). Below these are buttons for 'Inspect', 'Advanced Search', and 'Search By Public Key'. The main content area is titled 'Top 50 Most Recent Posts' and contains a table with columns for Section, Group, Date, and Message. Each row includes a 'View Post' button.

	Section	Group	Date	Message
<a href="#">View Post</a>	BFH	Voters	29/11/2014 20:00	[B@30e1b30b
<a href="#">View Post</a>	UNIBE	Voters	03/11/2014 11:50	[B@48ec4ee6
<a href="#">View Post</a>	BFH	Voters	21/10/2014 20:00	[B@12300371
<a href="#">View Post</a>	BFH	Candidates	21/10/2014 20:00	[B@e31f90b
<a href="#">View Post</a>	BFH	ElectionData	12/10/2014 20:00	[B@72d54874
<a href="#">View Post</a>	UNIBE	Voters	24/08/2014 08:57	[B@1067292
<a href="#">View Post</a>	BFH	Candidates	29/07/2014 16:15	[B@4d4657b8
<a href="#">View Post</a>	UNIBE	ElectionData	25/04/2014 15:42	[B@452918aa
<a href="#">View Post</a>	UZH	Voters	05/02/2014 20:23	[B@d12645f
<a href="#">View Post</a>	UZH	ElectionData	01/01/2014 20:00	[B@327b9fad
<a href="#">View Post</a>	UZH	Voters	03/08/2012 20:00	[B@523030bf

Figure 20: UniBoard Inspector Home Page

## 5.5 Exception Handling and Testing

In the following, a brief explanation of how error handling was done at the interface and business logic levels is presented. An error or an exception that occurs during the execution of the program changes the normal expected behavior of the program. This requires processing and treating the occurred error.

The user interacts with the application and the parameters entered by the user is validated by the ValidatorBean. The ValidatorBean handles errors at the user interface level and sends an appropriate error message to the user. The error messages are generated by the the MessageFactory. Once the parameters are validated, the enterprise bean classes communicate with the SearchService in order to obtain the search results corresponding to the search parameters.

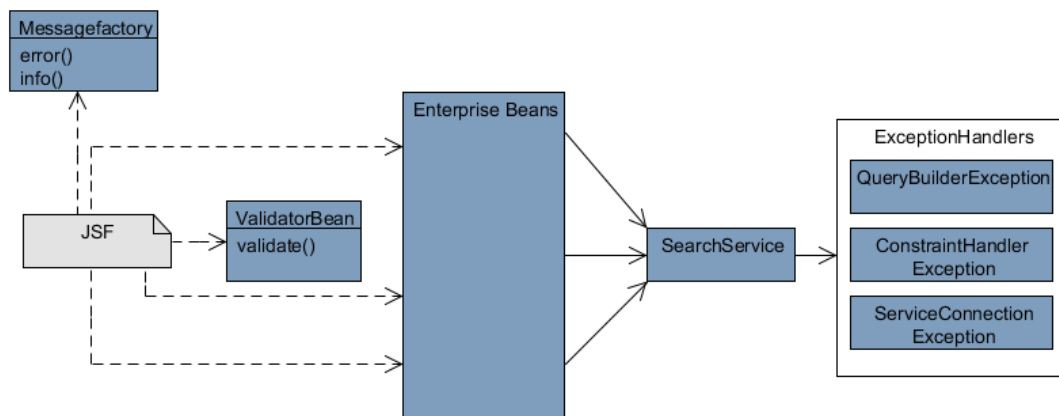


Figure 21: Exception Handling

Exceptions and errors encountered while building the query and connecting to the web service are handled by the exception handlers at the business logic level of the application. Hence, exception handling is done at the user interface and business logic levels.

The main functionalities of the application were tested with the help of unit tests. The results of the tests shows that the application behaves as expected. The results of the unit tests are shown below.

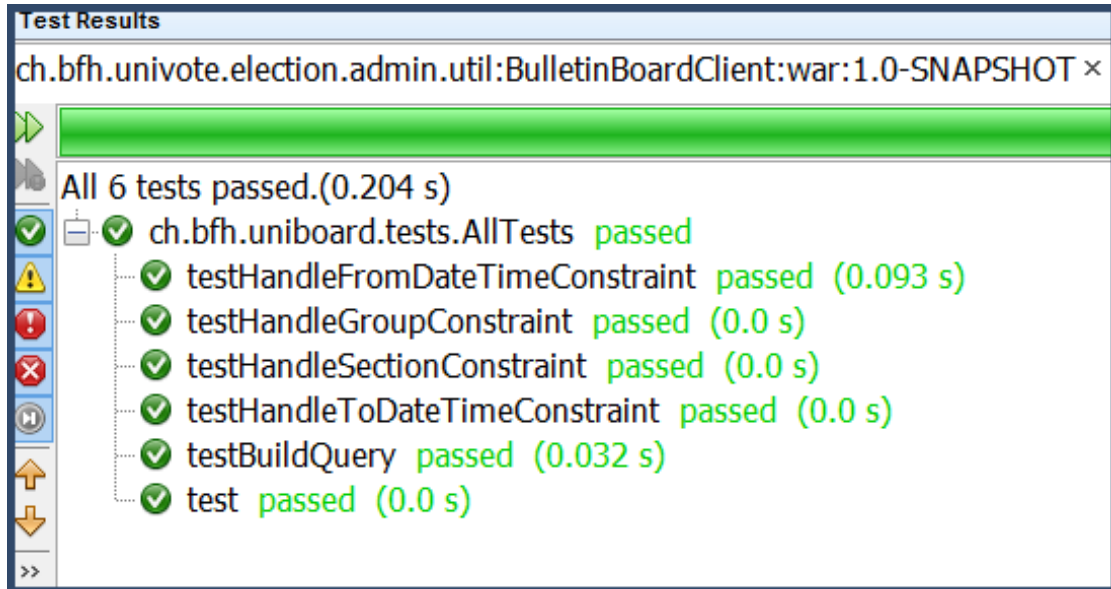


Figure 22: Test Results

The code coverage report is shown below. A coverage of around 50 percent is obtained. This covers the code from the classes that handle the principle functionalities of the application. The classes that have not been covered are the enterprise beans classes that require a different testing framework.

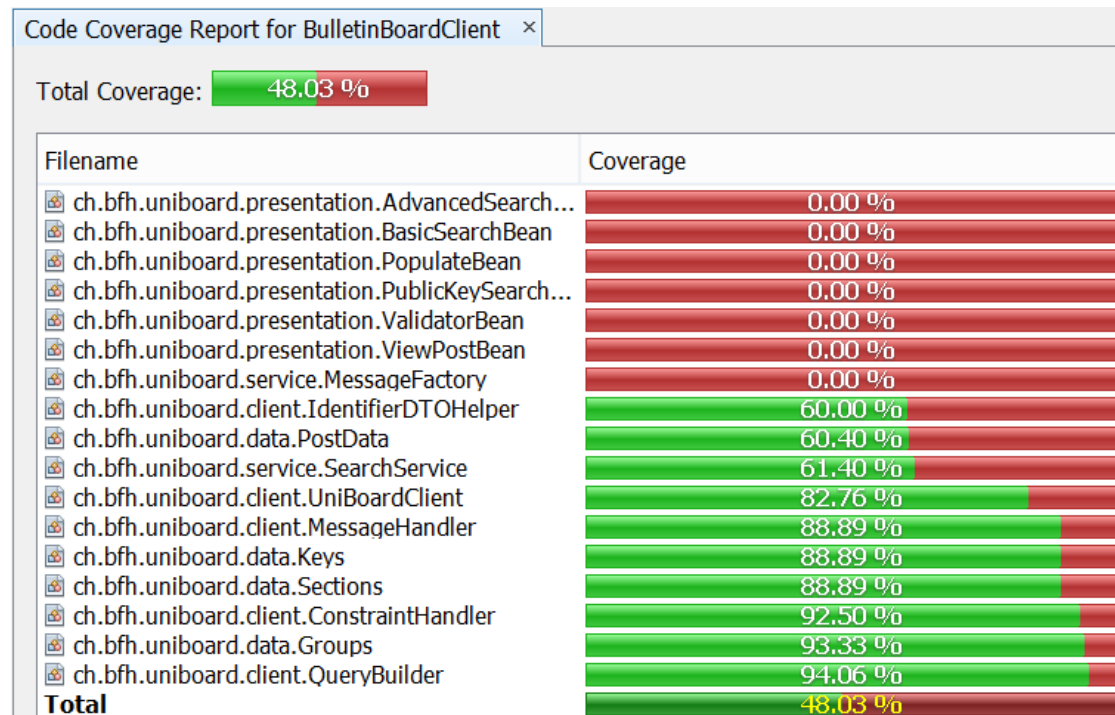


Figure 23: Code Coverage Report

## 5.6 Tools Used

The tools used to develop the application, design documents and to edit the report is presented below.

- Development Environment: Netbeans 8.0 with Primefaces 5.
- Editing Design Documents: UMLet, Visual Paradigm, Balsamiq Mockups.
- Testing: Maven Code Coverage with Cobertura.
- Editing report: TexStudio  $\text{\LaTeX}$ .

## 6 Work Methods and Project Organization

A project such as a Bachelor thesis needs a considerable level of planning and organization of the work. I have dedicated this section to briefly present the working methods and goals achieved during the semester.

An initial project plan was established at the start of the semester. Even though all requirements cannot be fully collected in an initial plan, it was an advantage to have a step-by-step plan and a time-line to guide the work done during the semester. The supervision meetings took place once a week where ideas were discussed and progress reports were submitted.

During the software development process attention was given to the design and coding was done in order to have a working software. Once a working functionality was obtained, any required refactoring and testing of the code was done. The refactoring involved better organization of the code, changes requested during the supervision meetings and the incorporation of new ideas. Unit tests were done for the main functional parts of the code. The software development consisted of a continuous process of coding, refactoring and testing.

Equal importance was given to the project report as for software development. Content to the report was added periodically starting from the first week itself. Whenever a new functionality of the software was obtained, documentation supporting the design and the implementation was added.

In the following, you will find the project plan that was established at the start of the semester. A complementary document showing the goals achieved every week is presented as well, which notes down more precisely the work done during the week and the deadlines that were met. It is more informative towards the technical aspects of the project and in the progress of the project report.

## 6.1 Project Plan

- Week 1, 15.09:    1. First meeting with supervisor.  
                    2. General discussion of project, ideas and work to be done for the coming week.  
                    3. System set-up.
- Week 2, 22.09:    1. Work organization planning.  
                    2. Report writing: Introduction, Project Goal.  
                    3. First version of mock-ups.
- Week 3, 29.09:    1. Second version of mock-ups.  
                    2. Report writing: The Get operations that will be made available, the  
                       format of the request and response messages, json and json schema.
- Week 4, 06.10:    1. Finalization of design with mock-ups.  
                    2. Getting a first web service client side code running.  
                    3. Report writing.
- Week 5-6:  
13.10 - 20.10    1. Design the main page with JSF Primefaces.  
                    a. Realization of page flow according to the mock-ups.  
                    b. Realization of the components contained in the web  
                       pages (text fields, buttons, tabular views etc).  
                    2. Report writing.
- 20.10.2014:       First Milestone. Completion of design.
- Week 6-11:  
27.10 - 01.12    Coding and design.  
                    a. Invoking web service with an appropriate query.  
                    b. Obtain the JSON response and extract the parameters.  
                    c. Display it to the user in a view which is clear and easy to understand.
- 26.11.2014:       Second Milestone. Completion of Basic Search with error handling and unit tests.
- Week 12, 08.12:   Code Review and testing.  
                    Report writing.
- Week 13, 15.12:   Code review and testing.  
                    Report writing.
- Week 14, 22.12:   Code review and testing  
                    Report writing.
- 31.12.2014:       Third Milestone. Submission of first version of the report.
- Week 15, 05.01:   Revise all documents and deliverables according to feedback.
- Week 16, 12.01:   Submit project.

## 6.2 Weekly Goals Achieved

- Week 1, 15.09: 1. System set-up.
- Week 2, 22.09: 1. Report writing(Added sections Introduction and Project Goal)  
2. Work organization plan.
- Week 3, 29.09: 1. Example of a bulletin board.  
2. Class diagram and graphical representation for the example.  
3. First version of mock-ups.  
4. Report writing (Added sections UniBoard, Basic Operations of UniBoard)
- Week 4, 06.10: 1. Finalization of design with mock-ups.  
2. Report writing (JSON and JSON Schema).  
3. Started design of web pages with JSF.
- Week 5, 13.10: Design of pages with JSF.  
a. A data table to display the information.  
b. Definition of a temporary dataset to populate the data table.  
c. A dialog with various components to provide search fields for the advanced Search.
- Week 6, 20.10: 1. Added new components and functionalities to existing pages.  
a. "View Post" buttons for every post that is displayed in the data table.  
b. A dialog to display detailed information for a post that appears by clicking on the "View Post" button.  
c. New components added to "Advanced Search" dialog.  
2. Provided Json schemas for the test data.
- Week 7, 27.10: Report writing (Added section: Web Services).  
Refactoring of code to be more generic.  
Added error handling for user interface.
- Week 8, 03.11: Preparation for meeting with the expert (short presentation).  
Added web service client(UniBoardClient.java) that queries the board.
- Week 9, 10.11: Completion of web client with the different queries for Basic Search.  
Added Java classes QueryBuilder, ConstraintHandler and other helper classes.
- Week 10, 17.11: Report writing: added section design documents.  
Added unit tests for java classes.  
Added error handling for java classes.
- Week 11-12,  
22.11 - 31.11: Worked on basic search, Code refactoring, Unit tests,  
End of Basic search
- Week 13-16,  
07.12 - 14.01: Completed Advanced Search and Search by Public Key.  
Final report.

## 7 Conclusion

The aim of the project was very clear and the ideas of how to achieve them as well. This helped in the organization of the work and in meeting the goals that were set for the project without any major difficulties. The work of this project gave me a deeper insight on the existing technologies such as web services, web applications, message formats, communication protocols and querying a NoSQL database.

The application UniBoard Inspector developed as a work of this project is an independent web application but at the same time has dependencies with the research project UniBoard. This needed some amount of integration with respect to the project UniBoard. Hence, working on this project helped me gain considerable knowledge and understanding of the already existing system UniBoard.

Since UniBoard is a public bulletin board, the messages that are posted to the board are public messages and they can be viewed by anyone without any security concerns. The application UniBoard Inspector sends a query to the board and displays the resulting posts in a viewer. A question that arises is about the authenticity of the data displayed by the application. This can be easily verified since every post contains a digital signature from the board. By using the corresponding public key one can verify if the post was returned by the board and not by a third party that intercepted the communication.

Verification of the board signature in order to confirm that the content displayed by the application is indeed the content returned by the board is a functionality that can be implemented as a future development of this project.



## 8 Annexes

### 8.1 UniBoard Inspector User Interfaces

In this section, the different web user interfaces of the application UniBoard Inspector is presented.

#### 8.1.1 View Post Dialog

**Post Details**

**Section:** HESB  
**Group:** ElectionData  
**Rank:** 12  
**Timestamp:** 12/10/2014 20:00

**Message:**

End: 2014-12-15 24:00  
Start: 2014-10-21 12:00  
Election Id: VSBFH  
Title: Best Teacher Award 2014

**Signature:**

0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAxjAgBgNVBAMMGXBoaWxlbW9uLnZvbmJlcmdlbkBiZmguY2gxZzARBgNVBAQMCnZvbiBCZXJnZW4xZDASBgNVBCoMC1BoaWzDg8KpbW9uMQ8w

**Public Key:**

RNf8/fJqfBlv27VGxWDxxzQhmlRAM aKumjBW7zIYnjYJ5By3ptSLNQLjctO00vh6b0sQ

**Board Signature:**

iBCZXJnZW4xZDASBgNVBCoMC1BoaWzDg8KpbW9uMQ8wDQYDVQKQ DAZIZmguY2gwggEiMA0GCS

**Hash:**

d41d8cd98f00b204e9800998ecf8427 75cdbfeb70a06d42210938da88c42991 6e0b7a1676ec0279139b3f39bd65e41a c74c812e4d2839fa9acf0aa0c915e022

**Confirmation:**

-----BEGIN SIGNATURE----- IQB1AwUBMVSIA5QYCuMfgNYjAQFAKgL/ZkBfBEsbthba4BlrcnjqabcKgNv+ a5kr4537y8RCd+RHm75yYh5xxA1ojELwNhhb7cltrp2V7LIONAelws4S87UX80c LBTBcN6AACf11qymC2h+Rb2j5SU+rmXWru+=QFMx -----END SIGNATURE-----

Figure 24: View Post Dialog

### 8.1.2 Advanced Search Dialog

**Advanced Search**

**Section:**  All  BFH  UZH  UNIBE

**Group:**  All  Voters  ElectionData  Candidates  
 Ballots  DecryptedVotes  ElectionResult

**Date:** From   
To

**Rank:**  Equal To   
 Less Than   
 More Than  
 Between

**Limit:**

**Copy and paste your public key in the text area below:**

```
RNf8/fJqfBlv27VGxWDxxzQhmlRAMaKumjBW7zIYnjcYJ5B  
y3ptSLNQljctO00vh6b0sQ
```

Figure 25: Advanced Search Dialog

### 8.1.3 Search by Public Key Dialog

**Search by Public Key** [X]

Copy and paste your public key in the text area below:

```
RNf8/fJqfBlv27VGxWDxxzQhmlRAM\naKumjBW7zIYnjcYJ5By3ptSLNQljctO00vh6b0sQ
```

**From Date:** 01/12/2014 00:00

**To Date:** 31/12/2014 00:00

**Limit:** 50

Inspect

Figure 26: Search by Public Key Dialog

## 8.1.4 Basic Search Results Page

The screenshot displays the UniBoard Inspector interface. On the left, the 'Basic Search' section includes filters for Section (BFH), Group (ElectionData), From Date (14/01/2014 00:00), To Date (01/01/2015 00:00), and Limit (50). An 'Inspect' button is located below these filters. The main area, titled 'Basic Search Results', contains a table with the following data:

View Post	Date	Rank	End	Start	Message	Election Id	Title
<a href="#">View Post</a>	12/10/2014 20:00	12	2014-12-15 24:00	2014-10-21 12:00	VSBFH		Best Teacher Award 2014

Figure 27: Basic Search Results Page

## 8.2 JSON Messages

The JSON messages for all the different groups are listed below.

### 8.2.1 JSON Message for the Post Candidate

The corresponding JSON message for the post Candidate is presented below:

```
{
  "_id": ObjectId("54475149b0b388cd1f52b6fb"),
  "message": {
    "electionid": "VSBFH",
    "candidatenummer": 2,
    "firstname": "peter",
    "lastname": "mulhouse",
    "age": 56,
    "email": "peter.mulhouse@bfh.ch"
  },
  "alpha": {
    "section": "HESB",
    "group": "Candidates",
    "signature": "0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAxIjAgBgNVBAMMGXBoawx1bW9uLnZvbmJlcmd1bkBiZmguY2gxZzAR\nBgNVBAQMCnZvbiBCZXJnZW4xFDASBgNVBCoMC1BoawzDg8Kpbw9uMQ8w",
    "key": "RNf8/fJqfBlv27VGxWDxxzQhm1RAM\naKumjBW7z1YnjcYJ5By3ptSLNQ1jct000vh6b0sQ"
  },
  "beta": {
    "timestamp": ISODate("2014-10-21T20:00:00.0Z"),
    "rank": 7,
    "boardSignature": "iBCZXJnZW4xFDASBgNVBCoMC1BoawzDg8Kpbw9uMQ8wDQYDVQK\nnDAZiZmguY2gwgEgEfMA0GCS"
  }
}
```

Figure 28: JSON Message for Candidate.

## 8.2.2 JSON Message for the Post ElectionData

The corresponding JSON message for the post ElectionData is presented below:

```
{
  "message": {
    "electionid": "VSBFH",
    "title": "Best Teacher Award 2014",
    "start": ISODate("2014-10-21T20:00:00.000Z"),
    "end": ISODate("2014-12-15T20:00:00.000Z")
  },
  "alpha": {
    "section": "HESB",
    "group": "ElectionData",
    "signature": "0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAXIjAgBgNVBAMMGXBoawx1bW9uLnZvbmJlcmd1bkBiZmguY2gxZzAR\nBgnVBAQMCnZvbiBCZXJnZW4xFDASBgNVBCoMC1BoawzDg8KpbW9uMQ8w",
    "key": "RNf8/fJqfB1v27VGxwDxxzQhm1RAM\naKumjBw7z1YnjcYJ5By3ptSLNQ1jct000vh6b0sQ"
  },
  "beta": {
    "timestamp": ISODate("2014-10-12T20:00:00.000Z"),
    "rank": 12,
    "boardSignature": "iBCZXJnZW4xFDASBgNVBCoMC1BoawzDg8KpbW9uMQ8wDQYDVQK\nnDAZiZmguY2gwggEfMA0GCS"
  }
}
```

Figure 29: JSON Message for ElectionData

### 8.2.3 JSON Message for the Post Ballot

The corresponding JSON message for the post Ballot is presented below:

```
{
  "_id": ObjectId("54475149b0b388cd1f52b6fb"),
  "message": {
    "encryptedvote": "ewogICAgImdyb3VwIjogImJhbGxvZCIsc3R5cHRvIjogewogICAgICAidHlwZSI6ICJETC
    IsCiAgICAgICJwIjogIjE2MTkzMTQ4MTE5ODQ4MDYzOTIyMDIxNDZmZTU5NTkzMTQ0MTA5NDU4NjMw
    NDkxODQwMjg3MzUwNjUxMDU0NzIzNzIyMzc4Nzc3NTQ3NTQyNTk5MTQ0MzkyNDk3NzQxOTMzMDY2Mz
    E3MDIyNDU2OTc0ODAxOTkwMDE4MDA1MDExNDQ2ODQzMDQxMzkwODY4NzMyOTg3MTI1MTEwMTI4MDg3
    ODc4NjU4ODUxNTY2ODAxMjc3Mjc5ODI5ODUxMTYyMTYzNDE0NTQ2NDYwMDYyNjYxOTU0ODgyMzIzOD
    E4NTM5MDAzNDg2ODM1NDkzMzA1MDEyODUxNTY2MjY2MzY1Mzg0MTg0MjY5OTUzNTI4Mjk4NzIzMDYw
    MDg1MjU1MDE4NDc4NDk0MDE2NDgwNzYwNjMwNDI5NyIsCiAgICAgICJxIjogIjY1MTMzNjg3ODI0Mz
    gxNTAxOTgzNTIzNjg0Nzk2MDU3NjE0MTQ1MDEwNDI3NzUyNjkwODk3NTg4MDYwNDYyOTYwMzE5MjUx
    NzIzMDIxIiwKICAgICAgImciOiAiMTA5Mjg0MjY5OTUzNTIzNzA5NDk0",
  },
  "alpha": {
    "section": "BFH",
    "group": "Ballot",
    "signature": "0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAXIjAgBgNVBAMMGXBoawxlbW9uLnZvbml1cmd1bk
    BiZmguY2gxEzAR\nBgNVBAQMCnZvbiBCZXJnZW4xZDASBgNVBCoMC1BoawzDg8KpbW9uMQ8w",
    "key": "RNf8/fJqfB1v27VGxWDxxzQhm1RAM\naKumjBw7z1YnjcYJ5By3ptSLNQLjct00vh6b0sQ"
  },
  "beta": {
    "timestamp": ISODate("2015-01-01T08:15:00.0Z"),
    "rank": 3,
    "boardSignature": "iBCZXJnZW4xZDASBgNVBCoMC1BoawzDg8KpbW9uMQ8wDQYDVQK\nnDAZiZmguY2gwggEfmA0GCS"
  }
}
```

Figure 30: JSON Message for Ballot.

## 8.2.4 JSON Message for the Post DecryptedVote

The corresponding JSON message for the post DecryptedVote is presented below:

```
{
  "message": {
    "electionid": "VSBFH",
    "candidatenum": 1,
  },
  "alpha": {
    "section": "BFH",
    "group": "DecryptedVote",
    "signature": "0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAXIjAgBgNVBAMMGXBoaWw1bW9uLnZvbmJ1cmd1bkBiZmguY2gxZzAR\nBgNVBAQMCnZvbiBCZXJnZW4xFDASBgNVBCoMC1BoaWwzDg8Kpbw9uMQ8w",
    "key": "RNf8/fJqfB1v27VGxWDxxzQhm1RAM\naKumjBw7z1YnjcYJ5By3ptSLNQ1jct00vh6b0sQ"
  },
  "beta": {
    "timestamp": ISODate("2014-11-01T07:25:00.000Z"),
    "rank": 9,
    "boardSignature": "iBCZXJnZW4xFDASBgNVBCoMC1BoaWwzDg8Kpbw9uMQ8wDQYDVQK\nnDAZiZmguY2gwggEfMA0GCS"
  }
}
```

Figure 31: JSON Message for DecryptedVote.



### 8.2.5 JSON Message for the Post ElectionResult

The corresponding JSON message for the post ElectionResult is presented below:

```
{
  "message": {
    "electionid": "VSBFH",
    "candidatenum": 1,
    "votesreceived": 98
  },
  "alpha": {
    "section": "BFH",
    "group": "ElectionResult",
    "signature": "0xNDEwMjEwOTM5MDZaFw0xNjEwMjEwOTM5MDZaMGAxIjAg8gNVBAMMGXBoawx1bW9uLnZvbmJlcmd1bkBiZmguY2gxZzAR\nBgnVBAQMcNzVbiBCZXJnZW4xFDASBgNVBCoMC1BoawzDg8KpbW9uMQ8w",
    "key": "RNf8/fJqfBlv27VGxwDxxzQhm1RAM\naKumjBW7z1YnjcYJ5By3ptSLNQ1jct000vh6b0sQ"
  },
  "beta": {
    "timestamp": ISODate("2015-01-01T010:55:00.000Z"),
    "rank": 15,
    "boardSignature": "iBCZXJnZW4xFDASBgNVBCoMC1BoawzDg8KpbW9uMQ8wDQYDVQQK\nnDAZiZmguY2gwgEfmA0GCS"
  }
}
```

Figure 32: JSON Message for ElectionResult

## **Declaration of Authorship**

I hereby certify that I composed this work completely unaided, and without the use of any other sources or resources other than those specified in the bibliography. All text sections not of my authorship are cited as quotations, and accompanied by an exact reference to their origin.

Name: Priya Bianchetti

Date: 16-01-2015

Signature:

## References

- [1] Providing and Consuming Web Services  
[https://help.sap.com/saphelp\\_erp2005/helpdata/en/d6/f9bc3d52f39d33e10000000a11405a/frameset.htm](https://help.sap.com/saphelp_erp2005/helpdata/en/d6/f9bc3d52f39d33e10000000a11405a/frameset.htm)
- [2] Java API for XML Web Services.  
[http://en.wikipedia.org/wiki/Java\\_API\\_for\\_XML\\_Web\\_Services](http://en.wikipedia.org/wiki/Java_API_for_XML_Web_Services)
- [3] Web Services Description Language.  
[http://en.wikipedia.org/wiki/Web\\_Services\\_Description\\_Language](http://en.wikipedia.org/wiki/Web_Services_Description_Language)
- [4] Creating a Simple Web Service and Clients with JAX-WS  
<http://docs.oracle.com/javaee/6/tutorial/doc/bnayn.html>
- [5] Distributed Multitiered Applications <http://docs.oracle.com/javaee/6/tutorial/doc/bnaay.html>
- [6] JavaServer Faces [http://en.wikipedia.org/wiki/JavaServer\\_Faces](http://en.wikipedia.org/wiki/JavaServer_Faces)
- [7] James F.Kurose, Keith W.Ross, *Computer Networking*, Fifth Edition
- [8] Understanding Digital Certificates [http://technet.microsoft.com/en-us/library/bb123848\(v=exchg.65\).aspx](http://technet.microsoft.com/en-us/library/bb123848(v=exchg.65).aspx)
- [9] E. Dubuis, S. Fischli, R. Haenni, S. Hauser, R. E. Koenig, P. von Bergen, *UniVote System Specification*, Technical Report, Bern University of Applied Sciences, Biel, Switzerland