Post-Doctoral Application Management System

Architecture specification Document

May 21, 2014
Version 1.0
Iteration 1

Prepared for Ms. Cathy Sandis (UP Research Office)
by SoftServe Group

Group members

Kgothatso Phatedi Alfred Ngako (12236731)

Tokologo "Carlo" Machaba (12078027)

Mathys Ellis (12019837)

Change log			
Date	Version	Description	Person
18/05/2014	v 0.0	Document created.	Alfred Ngako
19/05/2014	v 0.0	Added more detail to information in subsections.	Alfred Ngako
20/05/2014	v 0.0	Added references and citations.	Alfred Ngako

Contents

List of Figures

1 Project Repository

https://github.com/mox1990/Project-Postdoc.git

2 Document description:

2.1 Document purpose

This document provides the documenting for the software architecture infrastructure which the application functionality is deployed and executed.

2.2 Documentation methodology

The documentation and software development methodology used by the project adhere to the guidelines set out by the agile method. Thus this document has undergone and will undergo various iterations that may extend or reduce the contents of the document.
The document was compiled using a software architecture specification document template provided by Dr Fritz Solms as an alternative to the Kruchten 4 + 1 approach to documenting software.
This document was created using the requirement elicitation techniques and requirement definitions as specified by Klaus Pohl's book Requirements Engineering: Fundamentals, Principles, and Techniques [Dr.Phol, K., 2010]. The requirements, vision and scope were elicited from the following sources:

- Numerous interviews with the client.
- On-line research into UP Post doctoral applications.
- Correspondence with the UP IT department.
- Collecting and analysing various documents such as:
  - The initial project request document
  - Application forms
  - Renewal forms
  - CV templates
  - Approval and recommendation forms

2.3 Document conventions:

- Documentation formulation tool: LaTeX

2.4 References:

- Dr.Phol, K., 2010, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer, Heidelberg.
- Abraham Kang. Aug 9, 2002, *Enterprise application integration using J2EE*, Available from: http://www.javaworld.com/article/2074488/enterprise-java/enterprise-application-integration-using-j2ee.html , [17 May 2014]
- Jendrock E, Cervera-Navarro R, Evans I, Haase K, Markito W, *The Java EE 7 Tutorial*, Available from: http://docs.oracle.com/javaee/7/tutorial/doc/home.htm , [20 May 2014]

3 Architecture Requirements

This section discusses the software architectural requirements from the software requirements such as:

- Architectural Scope,
- Quality Requirements,
- Integration and Access Channel Requirements, and
- Architectural Constraints.

All the above mentioned topics are put in place to address the non-functional requirements that were illicited from Ms. Cathy Sandis (UP Research Office).

3.1 Architectural Scope

The scope that the architecture needs to cover include:

- A persistence infrastructure (Database) to facilitate domain objects (e.g CVs, DRIS information, and Applications). This will also allow the implementation of the Audit trail required by the client and the centralised point for all the required documentation shared among stakeholders.
- A session oriented infrastructure to assist in realizing the security requirements of authenticating participants and their actions.
- A web access channel which will provide the client, and other stakeholders, with an interface to the underlying system.
- An infrastructure for the generation of reports.
- A mailing competent infrastructure.

3.2 Quality Requirements

The following are the requirements around the quality attributes of the systems and the services it provides:

3.2.1 Security requirements

The system will need to be fully secured since the system deals with confidential information such as person information, application statuses, financial data and meeting information. Also since the systems main goal is to provide stable and audible application and renewal process flow the system may not be vulnerable to data tampering or any tampering whatsoever.

The system will have to provide different security roles to the registered users on the system. Any number of roles should be assignable to any user by a administrator with the correct role to allow for flexibility. But in essence a stakeholder may only have access to their section of the application process. The system administrator should be able to view all the sections in the system and should be able to modify them except where they may not.

3.2.2 Availability:

The system's availability on designated networks will depend on the availability of the University of Pretoria's servers that host the system. If the University of Pretoria's servers hosting the system are active and provide access over a designated network then the system must be available over that designated network. The designated networks are defined as the internet and the campus network of the University of Pretoria.

3.2.3 Testability:

The system must be testable. This will be done using unit testing and following the test plan that will be laid out in the testing document of this project.

Unit testing will test each unit in regard to:

- **Preconditions**
- **Post conditions**

The project will also have two phases of testing:

- **Offline:** This is the initial phase of testing and debugging which will be done with pseudo data.
- **Online:** This is the final phase of testing and debugging which will be done with active real time data.

3.2.4 Scalability requirements

The current aim is to create a scalable system that can support 500 to 1000 applicants per year with possible growth. This is in line with the figures given by the client and the growth in the research sector of the university.

The system needs to be scalable in regard to the following factors:

- **Performance:** This is regarded as the speed and responsiveness of the system. The system needs to be handle report queries in less than 10 seconds. It should be able to handle any application section processing in less than 3 seconds.
- **Storage:** This is regarded as the growth and shrinking of the data that is stored. The system will need to be able to handle a database that is in the range of 1 GB to 15 GB that has the potential to grow even larger. The reason for this stems from the requirement that the system will support archival functionality and archived data will store the data for long periods of time.
- **Concurrency:** This regarded as the amount of active users on the system at the same time. The system will need to support at least 100+/- concurrent users efficient and effectively since the system requires multiple stakeholders to part take in the application process while there can be multiple applications occurring at the same time.

3.2.5 Auditability:

The system needs to provide an audit trail of all critical actions that occur in the system. Critical actions are considered: user account management operations, login action, logout action and any operation by a user that leads to a change in application data of a particular prospective fellow.

The Audit trail will be in the form of a read-only table stored in the database. It can only be viewed by a user with the correct security role. The system is the only entity that can modify the audit trail where this modification can only be the addition of entries.

3.2.6 Usability requirements

The primary language of the system will be South African English. Any other language support is not considered part of the requirements but the system will be designed to allow for such development in the future.

The system's UI will only consider 2 types of user categories with regard to usability:

- **Trained user:**

  This type of user will have to have training in understanding how the system functions and how to use it. Their computer skills will be assumed to be in the range of basic to intermediate. Thus the user interface can allow for certain complexities but these complexities must be kept at a minimal. This user will be regarded as a system administrator. The stakeholders who fall under this category is the DRIS staff members overseeing the application process.

- **Normal user:**

  This type of user will have no training. Their computer skills will be assumed to be none or minimal. Therefore the UI that they will have access to will be simplistic and will be as user friendly as possible. The stakeholders that fall under this category will be Prospective fellows, Grant holders, HODs, Deans and Post-doctoral committee members.

3.2.7 Robustness:

The system needs to be robust, with the following areas of focus. It will have the ability to detect if the incorrect input has been entered into the system, the user will be notified immediately and not allowed to continue. If there are any defects with any other components with in the system and a task cannot be completed the user will also be notified immediately with the appropriate message sent to them. Users will also re-authenticate using a session object. Emails will be used for this authentication.

3.2.8 Flexibility:

The system is designed in such a way that adding new features can be done without extensive code restructuring. The process of adding new features will require the programmer to create a new module for the new functionality.

3.2.9 Reusability:

The System's components will be used again where applicable. The Login credentials is an example of areas where the system will be reused.

3.2.10 Maintability:

The system administrator will have the responsibility of adding updates that a programmer may wish to add to the system.

3.3 Integration and Access Channel Requirements

This section discusses

1. the requirements for the different channels through which the system can be accessed by humans and other system, and
2. the integration channels which must be supported by this system.

3.3.1 Access Channels

The system must be reachable securely through a web browser.

3.3.2 Integration Channels

Upon completion the system must be integratable with the University of Pretorias PeopleSoft system.

3.4 Architectural Constraints

Ms. Cathy Sandis (UP Research Office)did not specify any Architectural constraints other then the fact that the system must be integratable with the University of Pretoria, packaged software solution, PeopleSoft System.

4 Architectural Patterns and Styles

The system will maintain a MVC architectural pattern specific to how Java EE is structured. Which is a layered structure that automatically decouples the client from the server.

The Java EE structure is designed to satisfy the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. Which is reason to why it suits its use in the development of the system. The following diagram provides the abstracted architectural structure which will be followed:

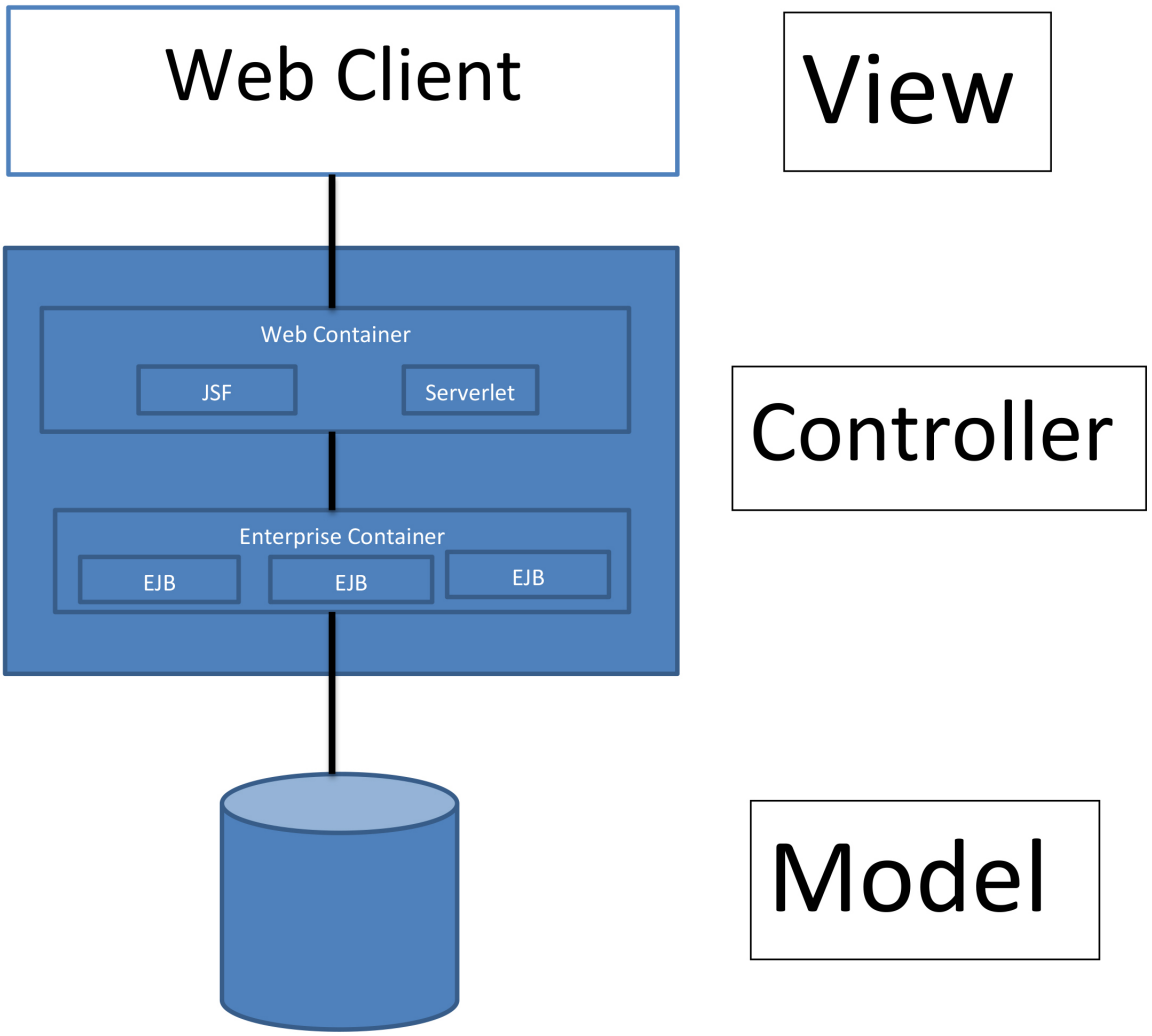

Figure 1: Model View Controller of Architecture

View

This component provides an interface to the under laying system to the user. This client is the web browser. It also incorporates components which are in the Web container(controller), such as Java Sever Faces.

Controller

This component is made up of two containers, the web container and the Enterprise container.

Model

This component is the actual database on which the system runs on. Which will be used as the centralised point of access with regards to applications being processed by the system as required by Ms. Cathy Sandis (UP Research Office).

5 Architectural Tactics and Strategies

This section describes techniques which will be used to satisfy the quality requirements.

5.1 Concurrency

The Java Platform has always offered support for concurrent programming, which is the basis for implementing many of the services offered by Java EE containers. This is realized through the two main concepts of having a multiple threads execute under a single process, in the case of Java EE mulitple beans execute under the JVM. The number of threads that can execute under the JVM can go well beyond thousands depending on factors such as the machine the JVM is running on and how it has been configured.

Even though the concurrent threads will mean better performance and a scalable implementation for the system they may lead to issues that effect the reliability and integrity of the system such as:

- Deadlocks,
- Thread Starvation,
- Concurrent accessing of shared resources, and
- Situations where the program generates incorrect data.

To deal with these issues Java EE provides concurrent utilities that access concurrent resources via JNDI lookup or resource injection. The components in the utilities ensure

that the issues mentioned above are nullified. The primary components of interest in the concurrent utilities for our system are:

- managed executor service,
- managed scheduled executor service,
- managed thread factory, and
- context service.

5.1.1 Managed Executor Service

A managed executor service is used by applications to execute submitted tasks asynchronously. Tasks are executed on threads that are started and managed by the container. The context of the container is propagated to the thread executing the task.

For example, by using an ManagedExecutorService.submit() call, a task, such as the GenerateReportTask, could be submitted to execute at a later time and then, by using the Future object callback, retrieve the result when it becomes available ().

5.1.2 Managed Scheduled Executor Service

A managed scheduled executor service is used by applications to execute submitted tasks asynchronously at specific times. Tasks are executed on threads that are started and managed by the container. The context of the container is propagated to the thread executing the task. The API provides the scheduling functionality that allows users to set a specific date/time for the Task execution programmatically in the application.

5.1.3 Managed Thread Factory

A managed thread factory is used by applications to create managed threads. The threads are started and managed by the container. The context of the container is propagated to the thread executing the task. This object can also be used to provide custom factories for specific use cases (with custom Threads) and, for example, set specific/proprietary properties to these objects (Cervera-Navarro, Evans, Jendrock, Haase and Markito 2014).

6 Use of Reference Architecture and Framework

Java EE provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications. It also provides an architecture for implementing services as multitier applications that deliver the scalability, accessibility, and manageability needed by the system. This is reason to why it is ideal to use it in the development of the project.

It allows easy development of thin-client multi tiried applications without the many lines of complicated code that handles transactions or state management. Business logic is organized into reuseable components and provides underlying services as containers for all component types.

The features in the JavaServer Faces technology provided HTML5 friendly markup which will assist in the implementation of the user friendly UI.

6.1 Java Persistence

Java EE is in itself implemented as a object oriented model, yet it is expected to function with mainly relational databases. To bridge the gap between an object-oriented model and a relational database an object/relational mapping approach will be used to achieve Persistence. Java Persistence consists of the following areas:

- The Java Persistence API,
- The query language, and
- Object/relational mapping metadata

6.2 RESTful web services

RESTful web services are loosely coupled, lightweight web services that are particularly well suited for creating APIs for clients spread out across the internet. Representational State Transfer (REST) is an architectural style of client-server application centered around the transfer of representations of resources through requests and responses. In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are represented by documents and are acted upon by using a set of simple, well-defined operations.

In Java EE 7, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. JAX-RS is a Java programming language API designed to make it easy to develop applications that use the REST architecture.

The JAX-RS API uses Java programming language annotations to simplify the development of RESTful web services. Developers decorate Java programming language class files with JAX-RS annotations to define resources and the actions that can be performed on those resources. JAX-RS annotations are runtime annotations; therefore, runtime reflection will generate the helper classes and artifacts for the resource. A Java EE application archive containing JAX-RS resource classes will have the resources configured, the helper classes and artifacts generated, and the resource exposed to clients by deploying the archive to a Java EE server (Cervera-Navarro, Evans, Jendrock, Haase and Markito 2014).

7 Access and Integration channels

This section discusses the requirements for the channels through which the system can be accessed by humans and other systems. Also making mention about the integration channels which need to be followed.

7.1 Access Channels

The system will be accessible by humans through the recent versions of the following browsers:

1. Mozilla Firefox,
2. Google Chrome,
3. Microsoft Internet Explorer
4. Apple Safari, and
5. Opera

The mobile counterparts of the above mentioned browsers will also be catered for and so no other access channel (such as Android/Apple apps) is to be considered.

7.2 Integration Channels

Upon successful complementation the system must be integratedable with the University of Pretoria's PeopleSoft system. Therefore it is sensible to implement the system in Java EE as it is one of the many components that form part of the multiple tiers that build up PeopleSoft.

The use of build tools such as Maven will provide a central piece of information with regards to the projects build, reporting and documentation. This central point of information will assist in the integration phase since it is implemented through a standardised build approach which can come into play at a later time through the use of EAI.

The EAI will be achieved via the logical integration architecture of Direct point-to-point integration (Kang, 2002). This means that the application management system will make direct JDBC calls to the universities databases tables (which needed to be setup to cater for our system at that point). The Integration method will be pushed-based data-level integration (or if all else fails UI-Level integration).

8 Technologies

The System will use the following technologies:

- Java Enterprise Edition 7,
- MySQL database, open-source relational database management system
- Netbeans 8.0
- Glassfish server

Integrated Development Environment

The system should be buildable independent of an IDE but it will be developed on Netbeans 8.0 to allow for uniformity amongst the development team, with regards to coding style, and provide easy integration with the tools that will be used such as Javadoc, to generate API documentation in HTML format.

API

- Java Persistence API. The Java Persistence API (JPA) is a Java standards–based solution for persistence.
- JavaMail. API used to send email notifications.
- JavaBeans Activation Framework. The JavaBeans Activation Framework (JAF) is used by the JavaMail API. JAF provides standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and create the appropriate JavaBeans component to perform those operations.
- Java Database Connectivity. The Java Database Connectivity (JDBC) API lets you invoke SQL commands from Java programming language methods. You use the JDBC API in an enterprise bean when you have a session bean access the database. You can also use the JDBC API from a servlet or a JSP page to access the database directly without going through an enterprise bean.
- Java Naming and Directory Interface. The Java Naming and Directory Interface (JNDI) API provides naming and directory functionality, enabling applications to access multiple naming and directory services, such as LDAP, DNS, and NIS. The JNDI API provides applications with methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. Using JNDI, a Java EE application can store and retrieve any type of named Java object, allowing Java EE applications to coexist with many legacy applications and systems.
- Java API for XML Web Services. The Java API for XML Web Services (JAX-WS) specification provides support for web services that use the JAXB API for binding

XML data to Java objects. The JAX-WS specification defines client APIs for accessing web services as well as techniques for implementing web service endpoints. The Implementing Enterprise Web Services specification describes the deployment of JAX-WS-based services and clients. The EJB and Java Servlet specifications also describe aspects of such deployment. JAX-WS-based applications can be deployed using any of these deployment models.

- SOAP with Attachments API for Java. The SOAP with Attachments API for Java (SAAJ) is a low-level API on which JAX-WS depends. SAAJ enables the production and consumption of messages that conform to the SOAP 1.1 and 1.2 specifications and the SOAP with Attachments note.

- Java Authentication and Authorization Service. The Java Authentication and Authorization Service (JAAS) provides a way for a Java EE application to authenticate and authorize a specific user or group of users to run it.

Build Tools

- Apache Maven (for reasons explained in the integration channels section).

Operating System

The system will be deployable on:

- Windows 7 and 8.

- Linux based operating systems (specifically Kububuntu 13).

9 Glossary:

- **API** - Application Programming Interface
- **Application** -Both renewal applications or new fellowship applications are seen as applications by this project.
- **CV** - Curriculum Vita
- **EAI** - Enterprise Application Integration
- **HTML** - Hyper Text Mark-up Language
- **Java EE** - Java Enterprise Edition
- **JDBC** - Java Database Connection
- **MVC** - Model View Controller
- **UI** - User Interface
- **UP** - University of Pretoria