Post-Doctoral Application Management System

Architecture specification Document

August 1, 2014
Version 1.0
Iteration 2

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 0.1	Added more detail to information in subsections.	Alfred Ngako
20/05/2014	v 0.2	Added references and citations.	Alfred Ngako
22/05/2014	v 0.3	Transferred architectural scope to architectural requirements specification document. Began editing.	Mathys Ellis
23/05/2014	v 0.4	Added to and edited architectural patterns and style.	Mathys Ellis
23/05/2014	v 1.0	Finalised and edited document for first iteration.	Mathys Ellis

Contents

List of Figures

1 Project Repository

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

2 Document description:

2.1 Document purpose

This document follows on the Architecture requirements specification document. The Architecture specification document provides the architectural descriptions of various architectural factors that the project will need to follow when developing the system. Architecture in this document's context refer to the technological basis and software development styles of the project. Thus this document serves as a contract between SoftServe and the client, Mrs Cathy Sandis of the DRIS of the University of Pretoria in terms of what technologies the project should incorporate as-well the software development infrastructure that the system will be based on.

2.2 Documentation methodology

The documentation and software development methodology used by the project adhere to the guidelines set out by the scum agile methodology. 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:

- Kang, A. August 9, 2002, *Enterprise application integration using J2EE*, Available from: http://www.javaworld.com/article/2074488/enterprise-java/enterprise-application-integration-using-j2ee.html , [Accessed on: 17 May 2014]
- Ali Babar, M., *Architectural Patterns and Frameworks, Week 3, Lecture 3*,Available from: https://blog.itu.dk/MSAR-E2013/files/2013/09/wk3_lect3_patternsframeworktactics.pdf, [Accessed on: 21 May 2014]
- Dr.Phol, K., 2010, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer, Heidelberg.
- Kayal, D., 2008, *Pro Java EE Spring Patterns: Best Practices and Design Strategies Implementing Java EE patterns with Spring Framework*, Apress, New York.
- 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 , [Accessed on: 20 May 2014]
- Oracle. April 2014, *The Java EE 7 Tutorial*, Available from: http://docs.oracle.com/javaee/7/tuto , [Accessed on: 20 May 2014]

3 Architecture Requirements

This section can be found in the architecture requirements specification document in which it discusses the following:

- Architectural scope
- Access channel requirements
- Quality Requirements
- Integration requirements
- Architectural Constraints

4 Architectural Patterns and Styles

The system will employ the Model-View-Controller, MVC, architectural pattern combined with a multi-tier/layered architectural pattern to form what is known as the Java Enterprise Edition system architecture. This will allow the client(s) to be decoupled from the server. Further this allows the view, the controller and the model each to have its own set of layers. Both of these patterns provide various benefits and are widely used. Some benefits include modularity, encapsulation, re-usability of components, decoupling and system maintainability [Ali Babar, M.] [Oracle, 2014][Kayal, D., 2008]

The Java EE system architecture is designed to support highly scalable, distributed, transactional, and portable applications that use the speed, security, and reliability of a Java EE server to provide powerful enterprise applications. [Oracle, 2014] For this reason SoftServe believes this system architecture is well suited for the development of the Post-Doctoral application management system as it will satisfy the requirements in terms of quality and functional as stipulated by the client. The following diagram provides the system architectural that will be employed by the system:

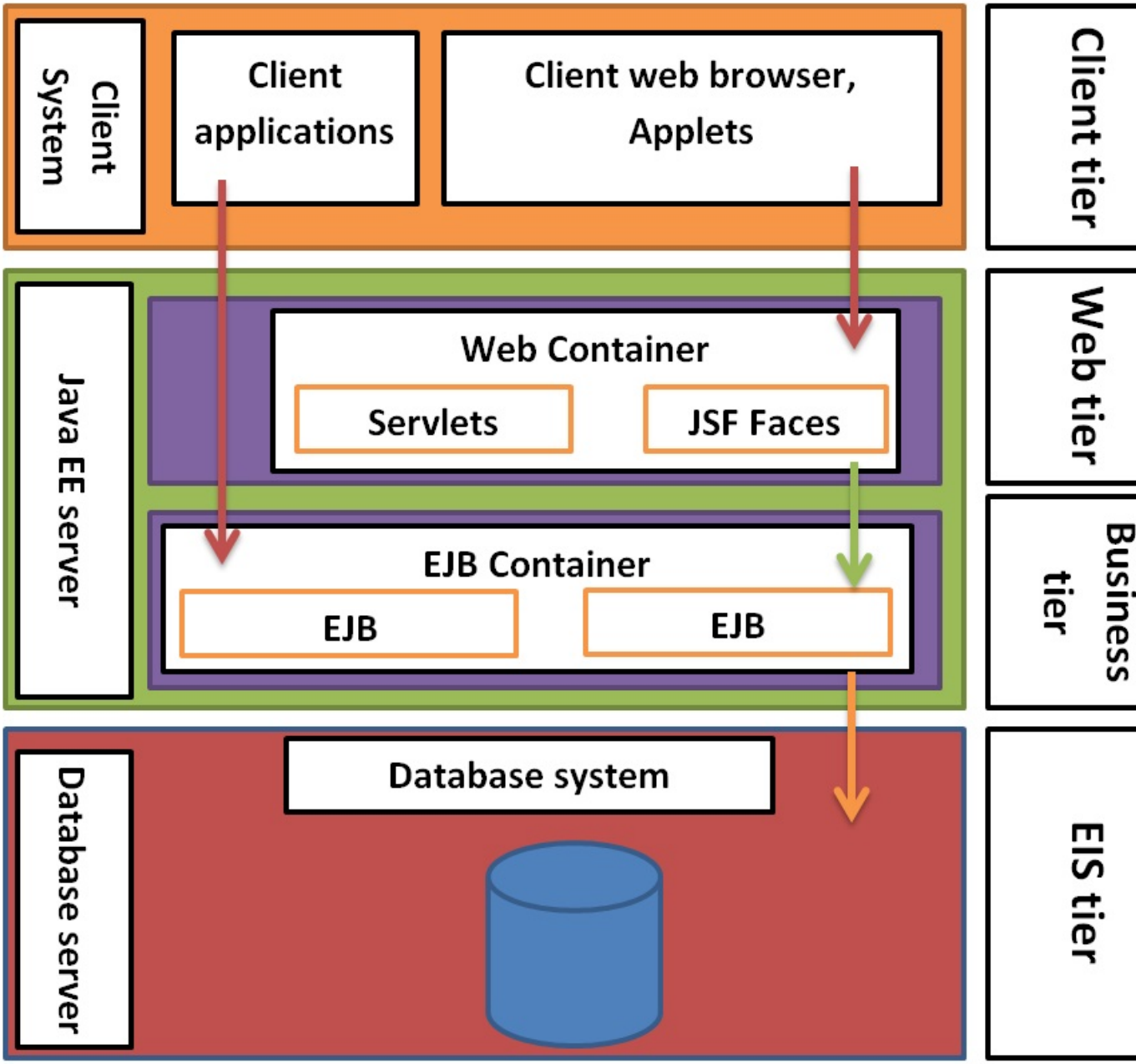

Figure 1: Java EE system architecture

4.1 View

The View component of the system architecture is used to represent and encapsulate the front-end of the system. It contains the client tier of the Java EE system architecture. Thus this allows the system to decouple the front-end and the back-end components. Allowing various clients to access the back-end without having to duplicate certain back-end tiers for each client. [Kayal, D., 2008]

4.1.1 Client Tier

This tier runs on the client system and encapsulates the various components that a client system may use to access the Java EE server-side tiers. These components include dynamic web pages, Java applications and Java applets. In order to make the Post-doctoral application management system accessible to any stakeholder over the internet and provide a uniform user experience the system will only make use of the dynamic web pages component provided by the Java EE client tier. [Oracle, 2014]

4.2 Controller

The controller component of the system architecture is used to provide the business logic and manage and manipulate the view's request in order to provide communication between the Model component and the View. This component therefore hosts the web tier and the business tier of the Java EE application. This component is located on the Java EE server which is a multi-threaded application server. [Kayal, D., 2008] [Oracle, 2014]

4.2.1 Web Tier

This Tier runs on the Java EE server and hosts the Web container. It provides the management and web page generation support for the web pages that the system has to provide to the view through the user of servlets, Java ServerPages and Java ServerFaces Facelets. Facilitates the communication between the business tier and client tier for web browser clients and applets. Client applications do not have to make use of the Web Tier and can directly skip to the business tier. But as stipulated above the other types of client components will not be considered for the project. [Oracle, 2014]

4.2.2 Business Tier

This Tier also runs on the Java EE server and hosts the Enterprise Java Bean container. It provides the business logic section for the Java EE application in the form of Enterprise Java Beans which are simply classes that represent various persistence entities of the data base, system messages, sessions, etc. This tier communicates with EIS tier in order to get access the database and various other lower level infrastructures that the Java EE application requires. [Oracle, 2014]

4.3 Model

The Model component of the system architecture contains the various persistence storage infrastructure and lower level system management features such as transaction processing. It hosts the Enterprise Information System tier of the Java EE application. [Kayal, D., 2008]

4.3.1 Enterprise Information Tier

The EIS tier provides mainly the support for database systems that is used by the Java-EE application. This tier can run on the Java-EE server as a virtual server or on a physically different database server. Due to the project budget and technical constraints the former will used by the system. [Oracle, 2014]

5 Architectural Tactics and Strategies

This section describes the architectural techniques which will be used in order for the system 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 Concurrency Utilities for Java EE

5.1.2 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 ().

5.1.3 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.

5.1.4 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

The core design philosophy of the Java-EE platform is to provide a Java-EE application developer with a set of test and well maintained and reusable APIs as well as frameworks that allow them to focus rather on implementing the actual business logic and UI than focusing on the underlying system technical and management services such as authentication, session management, etc. Thus the Java EE platform provides a runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications. As seen above it also provides a system architecture and various frameworks, discussed below, for implementing services for multi-tier applications that deliver the scalability, accessibility, and manageability needed by a system. Taking all the above into consideration this makes it ideal for the development of the project.

6.1 JavaServer Faces (JSF)

JSF is a web application GUI framework that is based on the JSP, EL and servlet technology that Java-EE provides. It allows the generation of various mark-up languages, such as HTML 4.0.1 and HTML 5, directly from objects and ORM model objects used by the Java-EE application. Thus it is ideal for system as the system needs to provide support for both HTML 5 and 4.0.1 web content.

6.2 Java Persistence API

Java EE is based on Java which is an object-oriented language. Whereas most modern day database management systems, DBMSs, provide relation databases. Thus to bridge this gap the Java Persistence API is used. It provides a Object Relational Mapping solution which allows the relation database to be viewed as a object-oriented database. Thus this critical for the system SoftServe wishes to develop as the system will make use of MySQL which is a relation database management system. The Java persistence API contains the following components:

- Persistence API
- The query language
- ORM

6.3 Java API for RESTful Web Services (JAX-RS)

The JAX-RS API provides a way for the Java-EE application to provide web services or data transfer via the HTTP or HTTPS protocol using the Representational State Transfer, REST, architectural style. This accomplished by the user of various JAX-RS runtime annotations. This will allow the system to provide a set of lightweight web services to various clients across the internet. This will allow the system to easily be accesses by mobile and computer platforms alike and also be accessible over most companies firewalls as it will make use of the HTTPS protocol. Further this will allow for future expansion if client wishes it. To insure security a the POST command will be preferred above GET.[Cervera-Navarro, Evans, Jendrock, Haase and Markito, 2014]

7 Access and Integration channels

This section discusses the requirements for the access channels through which the system can be accessed by humans and other systems. Also it discusses the integration channels which need to be used by the system.

7.1 Access Channels

To provide a system that is as accessible as possible the system provide its services through this essentially makes the system OS independent and accessible over firewalls since it will make use of the HTTPS protocol that is usually not blocked by firewalls. This also improves the usability the system as most computer user or mobile users are aware of how to use web browsers. The system will support the following versions of modern web browsers for both their computer and mobile counterparts:

1. Mozilla Firefox 20+
2. Google Chrome 30+
3. Microsoft Internet Explorer 9+
4. Apple Safari
5. Opera

7.2 Integration Channels

As mentioned in the architecture requirements specification document the IT department of the University of Pretoria will only be willing to provide the SoftServe group with the knowledge of the Peoplesoft system in order to integrate it. Thus the SoftServe group will attempt to accommodate the PeopleSoft system as much as possible by conforming to same user name styling, export of data to formats that are used by the Peoplesoft system to import data. Also the since Peoplesoft is an Oracle enterprise system it would have been developed using Java-EE and the Java programming language. This is therefore a strong motivation for Java-EE to be used by the SoftServe group as a development platform for the system. So to allow integration at a more technical level. Though if the integration was to be done this would be done via the EIS teir using the Java EE Connector Architecture API and the JAX-WS API.

8 Technologies

This section discusses and elaborates on the technologies that the system will use and should also be seen as an extension of the architectural constraints, specified in the Architecture requirements specification document, in terms of elaboration.

8.1 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.

8.2 Development technologies

8.2.1 Java Servlet Technology

A Java servlet is used to extend the Java-EE application server to support HTTP or HTTPS request and respond commands to the business Tier of the application. This is used by the JSF framework to communicate with the business Tier in order to receive and send data.

8.2.2 JavaServer Pages (JSP)

JSP is a technology that is used by the Java-EE platform to provide a native language approach to creating web pages by using HTML or XML to specify static content on a web page and Expression language to provide dynamic content. Thought this will not be used directly but rather by the JSF framework.

8.2.3 Expression Language (EL)

Is a language used by JSP pages and JSF facelets to write servlet code snippets which allow the usage of the data available to the servlet to do calculations or get or set data.

8.2.4 JavaMail API

The JavaMail API provides a email communication infrastructure for Java-EE applications. Thus this will be employed by the system in order to create and provide the email notification infrastructure.

8.2.5 JavaBeans Activation Framework (JAF)

The JavaBeans Activation Framework allows the Java-EE application to determine the data type of some section of data and thus allow the application to provide access to it by encapsulating it and determining the operations that the application may perform on it. This is mainly used by the JavaMail API thus it will need to be included in the system.

8.2.6 Java Database Connectivity API (JDBC)

The Java Database Connectivity API provides the a Java-EE application with the means to execute SQL statements on the database server via programming language commands. This will be used by the system to query the database in order to retrieve and store data on the database via the MySQL DBMS.

8.2.7 Java Transaction API (JTA)

The Java Transaction API provides the Java-EE Application with the means to handle data transactions to the database. The API makes use of an auto commit feature by default that allows any client to see any writing of data to the database automatically. This will be used in the system to improve the data centralisation and data accuracy.

8.2.8 Enterprise JavaBeans (EJB)

The Enterprise JavaBeans is a component used by all Java-EE applications to encapsulate the various business logic modules of the application. Thus it contains the attributes and methods associated with the business logic module and hence can be treated as an stand alone unit that can be reused. The two primary EJBs are Session beans that represent a clients session and the data associated with it and a Message-driven bean that can a allow the component to receive messages asynchronously via event listeners. The session bean will be used by the system to support the Session-Oriented infrastructure.

8.2.9 Java Naming and Directory Interface (JNDI)

The Java Naming and Directory Interface provides Java-EE applications to search for data across LDAP, DNS, etc, directory services. Above this it also allows the application to search for objects that exist within the application and provides access to them so that they can be used by the application. This is used by Java-EE applications usually to locate EJB objects for usage therefore the system will make use of this API.

8.2.10 Java Authentication and Authorization Service (JAAS)

The Java Authentication and Authorization Service provides Java-EE applications with a means to provide user access control. The API allows the application to authorize users with regards to specific user accounts or user groups. Thus this will be used by the system to support the security infrastructure needed by the system.

Build Tools

- **Apache Maven** - This build tool was chosen due to the flexibility of in terms of configurations, its ability to automate tests and the fact that Netbeans IDE provides support for it.

Operating System

The system will be deployed on a single OS but will have clients that will use a variety of OSs. This is one of the main reasons the why the application's UI must be web browser

based so to allow OS independent support. Thus the only prerequisite of the client's OS is that it needs to support a HTML 4 or 5 web browser and be internet accessible.

- **Server OS:**
    1. Linux: Kubuntu 13.10
- **Client OSs**
    1. Windows: All
    2. Linux: All
    3. iOS: All
    4. Android: All

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