BankarNet

Software Architecture Document

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <26/September/14> | 1.0 | Refill of the specification and view on different models of the proyect | Javier Rizo Orozco |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 2

1.5 Overview 2

2. Architectural Representation 2

3. Architectural Goals and Constraints 2

4. Use-Case View 2

4.1 Use-Case Realizations 2

5. Logical View 2

5.1 Overview 2

5.2 Architecturally Significant Design Packages 2

6. Process View 2

7. Deployment View 2

8. Implementation View 2

8.1 Overview 2

8.2 Layers 2

9. Data View (optional) 2

10. Size and Performance 2

11. Quality 2

Software Architecture Document

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

## Scope

This document applies the different diagrams of UML for the development on the project. Any change on the diagrams will affect the document too.

## Definitions, Acronyms, and Abbreviations

## References

Not applied

## Overview

The next topics will represent the architecture for the project software and how this could affecte on the totality system

# Architectural Representation

The architecture is the typical model Client – Server for the project of BankarNet. The users (or employees) will send a petition for diffrente ways to receive an appropriate answer depending on the information and solicitude.

# Architectural Goals and Constraints

Requirement Software:

* Windows(7 on up) or Linux(Any distribituion) SO
* Java 7 installed
* PostgreSQL installed

A logged system will included for the rights of each user.

Strategy

* Implements with the diagrams (Fault of them by this moment, release fasts, we could revise any error before for the code part, is better to having a diagram to have a view and improvement by the moment of work)
* Releasing functions for each user (1 week)
* Graphical Interface (1 Week)
* Log System and any critical details to finish (1 Week)
* Rest of the project before on December (Maybe 6 of December on 2014) were doing tests, review and iterations (for this moment we’re use one iteration)

Development Tools

* Eclipse
* Java SRC
* Git/Github for repository

Not planne schedule because for the hurry. We plane to finish fast but releasing a good product, we can debug any important feature and schedule only if is necessary.

Code will be standar, will use JavaDoc, comments if their necessary, review classes and reorganization

# Use-Case View

There is many features for this part, but the important features for the release are

* Sceneario for Employee Case
* Sceneraio for Teller Case
* Sceneraio on use case “Open Account” for the Executive Case

## Use-Case Realizations

On employee case the system need to create reports by the user because the demand of this. The consult system are used by all the employees and is required by the clients of the bank each time they going to the banks. On teller case all the use case are useful because the same reason (is required for the clients and the employees need to review the accounts or doing the transactions). The manager case is important but not enough and not critical too. The Executive Case on “Open Account” is important because they can work the other member on the consult activities

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

## Architecturally Significant Design Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

# Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

# Deployment View

[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]

# Implementation View

[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

## Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers. ]

## Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

# Data View (optional)

[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

# Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]