Документ за софтуерната архитектура

Version <1.0>

История на промените

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| <dd/mmm/yy> | <x.x> | Създаване на документа | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Съдържание

1. Въведение 4

1.1 Цел 4

1.2 Обхват 4

1.3 Дефиниции, Акроними и Абревиатури. 4

1.4 Референции 4

1.5 Кратък преглед. 4

2. Представяне на архитектурата 4

2.1 Географско разположение на системата 4

2.2 Обем на данните 4

2.3 Връзка с чужди системи 4

3. Архитектурни цели и ограничения 5

4. Use-Case View 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

5.3 Use-Case Realizations 5

6. Process View 5

7. Deployment View 5

8. Implementation View 5

8.1 Overview 5

8.2 Layers 6

9. Data View (optional) 6

10. Size and Performance 6

11. Качество 6

Документ за софтуерната архитектура

# Въведение

## Цел

Целта на този документ е да опише софтуерната архитектура на ABM. Той покрива както хардуерa, който е нужен за пълното и функциониране, така и софтуера който ще се използва за реализацията на различните под системи. Методологиите нужни за изграждането и работата и. Архитектурата дава ясна представа на екипа разработващ системата какви стъпки трябва да поеме той, както и предотвратява възможността от възникването на по – нататъшни проблеми за системата.

## Обхват

Документът включва

## Дефиниции, Акроними и Абревиатури.

Всички акроними, дефиниции и абревиатури може да се видят в документа:

ABM-I1-4-Glossary.doc

## Референции

* ABM-I1-5-SoftwareRequirementSpecifications.doc

## Кратък преглед.

[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]

# Представяне на архитектурата

За изготвянето на архитектурата на тези система са взети под предвид следните:

* Софтуерните изисквания, описани в документа за спецификация на допълнителните изисквания, зададени от възложителя на проекта;
* Нефункционални изисквания, описани в гореспоменатия документ;
* Географското разположение на системата;
* Обемът на данните с който системата работи;
* Нуждата от връзки с чужди системи;
* Бюджета нужен за реализация на проекта;
* Живот на системата;
* Наличие на “Legacy” код;
* Подобни системи.

## Географско разположение на системата

Взимайки в предвид големината на проекта и разположението на системата в цяла Европа. Топологията на мрежата, която отговаря най – добре на нашите цели е “Mesh network”. Тя предлага най – надеждна връзка до всяка една точка като потенциално може да загуби скорост за сметка на успешно доставени данни.

Осъществяването на вътрешните мрежи ще се осъществява с топология от тип шина, която предлага бързина, надеждност и ниски разходи.

## Обем на данните

Според количеството на данните който ще постъпват и ще се обработват се налага те да се съхраняват с облачни технологии. Фирмата Amplidata е една от много който предлагат подобни услуги. Взимайки под предвид успешното развитие на фирмата, тя ще е повече от добър съдружник и ще покрива напълно нуждите на системата с десетилетия.

## Връзка с чужди системи

Поради спецификата на работата на тази система се налага осъществяването на връзки с други системи. Най – наложилият се принцип на осъществяване на връзки с други системи е чрез сервизна шина (Enterprise Service Buss). Тя не само позволява общуването между различни приложения и системи, но и надеждно ниво на защита. Чрез нея ще може да се осъществят връзки като с държавни системи така и с други банки или “smart phone” приложения. Въпреки завишените разходи за поддръжката и на фона на конкурентността на други фирми използването и е задължително.

# Архитектурни цели и ограничения

## Използвани технологии

### SVN = GIT

### Maven – dobavqme

### FUCK JAVA BEANS

### Entity beans – ORM – trqbva ni

### JBoss = Apache (трябва да се провери точно)

### SOAP services – трябват ни, ние ще използваме Java Spring – и с по-малко обяснения

# Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

# 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.]

## Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]

# 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.]

# Качество

Софтуерната архитектура ясно очертава възможностите на системата, както и позволява интегрирането на нови системи към нея.

[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.]