

School of Computer Science Engineering

2017-05-08

GestUsers: Citizen Participation System



Aquilino A. Juan Fuente

José Emilio Labra Gayo

Begoña Cristina Pelayo García-Bustelo

Jordán Pascual Espada

Vicente García Díaz

*Software Architecture for GestUsers. Description of the practice work (2017)*

Description of the first practice work to be made by the work teams of the course “Software Architecture” during the academic year 2016-17.

**Grado de Ingeniería Informática del Software**

|  |  |
| --- | --- |
|  |  |

**GestUsers: Citizen Participation System**

**Authors**: **Juan Francisco Piñera Ovejero**

**Gonzalo de la Cruz Fernández**

**Guillermo Rodríguez González**

**Adrian Mirón Cao**

**Paula Tuñón Alba**

**Oriol Invernón Llaneza**

**Date**: 08/05/2017

**Version**: ￼

Table of Contents

[Introduction and Goals 5](#_Toc482009016)

[Requirements 6](#_Toc482009017)

[User Management 6](#_Toc482009018)

[CitizensLoader 6](#_Toc482009019)

[Participants 7](#_Toc482009020)

[Participation System 7](#_Toc482009021)

[Dashboard 8](#_Toc482009022)

[Methodology 9](#_Toc482009023)

[Stakeholders 10](#_Toc482009024)

[Students that develop the assignment 10](#_Toc482009025)

[System administrator 10](#_Toc482009026)

[Citizens 10](#_Toc482009027)

[Developers of the Participation System 11](#_Toc482009028)

[Course teachers 11](#_Toc482009029)

[City council 11](#_Toc482009030)

[Other student developer group 11](#_Toc482009031)

[Quality Attributes 12](#_Toc482009032)

[List of Quality Attributes 12](#_Toc482009033)

[Quality Attributes and stakeholders 14](#_Toc482009034)

[Architecture Constraints 16](#_Toc482009035)

[Technical constraints 16](#_Toc482009036)

[Organizational Constraints 16](#_Toc482009037)

[System scope and context 18](#_Toc482009038)

[Quality Scenarios 21](#_Toc482009039)

[Views 26](#_Toc482009040)

[Context 27](#_Toc482009041)

[Main overview 27](#_Toc482009042)

[Elements Catalogue 28](#_Toc482009043)

[Citizens List 30](#_Toc482009044)

[Main overview 30](#_Toc482009045)

[Catalogue of Elements 31](#_Toc482009046)

[Context Diagram 32](#_Toc482009047)

[Rationale 32](#_Toc482009048)

[Participants 33](#_Toc482009049)

[Main overview 33](#_Toc482009050)

[Catalogue of elements 33](#_Toc482009051)

[Context Diagram 35](#_Toc482009052)

[Rationale 35](#_Toc482009053)

[Dashboard 36](#_Toc482009054)

[Main overview 36](#_Toc482009055)

[Catalogue of elements 36](#_Toc482009056)

[Context Diagram 37](#_Toc482009057)

[Rationale 37](#_Toc482009058)

[References 41](#_Toc482009059)

# Introduction and Goals

The goal of this document is to describe the structure of an architecture of a Citizen Participation System, divided in 3 system modules: User Management system that will be reused; Participation System and Dashboard System. Although each system that we describe has its own functionality, the main goal is that it will be part of the general system of citizen participation.

This document describes the final deliverable of the laboratory assignment of the course "*Software Architecture*" which is taught by the professors. The course is part of the Degree in Software Engineering, School of Computer Science Engineering, University of Oviedo.

The User Management system is divided in two parts: CitizensLoader, to load data about citizens and Participants, to check if a user can participate. The students have to implement the software described in this document in two teams of 3 or 4 students during 3 weeks. One team will implement the CitizensLoader module sub-system and the other team will implement the Participants module.

The Participation System will be in charge of managing the Citizen Participation. It allows users to create, vote and comment proposals. The system will be made by a team of 3 or 4 students during three weeks.

The Dashboard System will offer a dashboard to view the evolution of the participation system in real time. This system will be made by the other team of 3 or 4 students different from the one implementing Participation System. Both systems will be developed in parallel

# Requirements

The project is divided in 3 modules:

1. User Management will be divided in two parts:

* CitizenLoader loads the list of users from the Council, for example the municipal census
* Participants: checks if a citizen can participate

1. Participation System is in charge of managing the citizen participation
2. Dashboard allows the Council staff to monitor the evolution of the participation system.

## User Management

### CitizensLoader

The System administrator must be able to introduce data from the citizens list. That data can be obtained from different sources like the municipal census, lists of immigrants without official residence, bystanders, etc. Those lists will be delivered by some institution to the Council.

The introduction of data will be made from Excel files that contain a list of rows with the following information:

* First name
* Last name
* Email
* Date of birth
* Address
* Nationality
* ID (National ID, the residence card ID, etc.)

When importing the citizens’ data, the system will create a user (whose login name will be the email) and a random password which will enable the user to enter the system to check if the data is correct as well as to later participate in the system. The system will generate personal letters that will be sent to each user by email. This task will be done by the Council and is not part of this system.

If a user appears in two different lists, this event will be recorded and informed in a log file. A user can only be created once. If the data is different from the current data available in the system, the current data will not be modified and an error will be recorded in the log.

[**Optional**] The system could be extended to emit the letters using other formats like Microsoft Word or PDF.

[**Optional**] If the input file contains errors, the system must detect them and report the errors found.

[**Optional**] The input data parser can be configured to accept data in different formats. Although it is mandatory to import data in Excel format, the system should be ready to be extended in the future to accept other formats easily.

(**Optional**) The service can be extended to handle security aspects

### Participants

Citizens should be able to login into the system to check that they can participate once the notification letter has been received. In order to implement that feature, a simple web service will be created that has two parameters passed as a POST message: login name and password and returns the data available about the citizen if the information is correct or reports an error if it isn't. Both the call parameters and the return information will employ JSON format.

(**Optional**) The web service can be extended to offer a simple HTML interface where a user can login and see his information in a human-friendly way.

(**Optional**) Using HTTP content negotiation, the system could handle other formats as XML.

(**Optional**) The service can be extended to enable the user to change his password.

(**Optional**) The service can be extended to handle security aspects

## Participation System

The participants must be able to make new proposals and to vote proposals from other participants. The proposals will be ordered by their popularity or by the date in which they were created. There will be a minimal number of votes for a proposal in order to be approved. This number will be set by the system administrators.

Proposals must allow comments from other users in order to discuss and improve them. The comments can be ordered by chronological order or by popularity. The comments can be also voted by the participants.

The council will be able to configure the different options of the participation system. The council decides what categories are allowed for the proposals. It also has to determine how long proposals are active for voting and discuss about them. The council has to moderate the words they consider offensive or inappropriate in order to maintain an atmosphere of mutual respect between the participants. The council will be able to delete non-appropriate proposals.

When a proposal has overcome the required number of votes to be accepted, the administrators will receive a notification. A proposal that has passed the acceptance phase could be updated by the votes that result from the corresponding parliament.

The proposals, the comments and the feedback they receive will be recorded in the application log that will be connected to a Kafka Stream.

(**Optional**) The Council can add or remove new offensive or inappropriate words at execution time.

## Dashboard

The system will offer a dashboard to the Council staff, councilmen and other authorities so that they view the evolution of the participation system in real time. Different groups of people will be able to see different types of visualizations.

The dashboard must update the information dynamically and in real time without user interaction. It must reflect the changes that appear in the participation system as they are produced as well as update that information in all the concurrent clients that are present/connected at update time.

The dashboard will offer information about the evolution of the proposals, the number of votes of each of them as well as the different comments.

In order to develop the system, a Kafka Stream will be configured from which all the events that will be dynamically visualized will be taken.

To facilitate the independent development of the dashboard from the participation system, a small testing simulator will be created that will generate random events to the Kafka Stream that will be visualized.

(**Optional**) The dashboard can offer graphical visualizations of the different statistics.

(**Optional**) The dashboard can send alarms to some users when some specific events happen.

# Methodology

This document employs the ADD￼(Bass, Clements, & Kazman, 2003)￼￼￼(ANSI/IEEE 1471, 2000).

The templates have also been inspired by the Arc42 templates (<http://arc42.org/>) where documentation architecture templates are defined in English, German and Spanish.

Another project that follows those templates for a biking domain is available at:

<http://biking.michael-simons.eu/docs/index.html>

# Stakeholders

The stakeholders identified are:

1. Students that develop the assignment
2. System Administrator
3. Citizens
4. People responsible of the participation system
5. Course Teachers
6. City council
7. Other student group

| Code | Stakeholder | Interests (Modules) |
| --- | --- | --- |
| ST-01 | Students | All modules |
| ST-02 | System administrator | Load files; Configure participation system |
| ST-03 | Citizens | Check data; Create, vote and discuss proposals |
| ST-04 | Developers of Participation System | Check data |
| ST-05 | Course Teachers | All modules |
| ST-06 | City council | Check data; Configure participation system |
| ST-07 | Other student developer group | Participants; Dashboard |

Table 1. List of stakeholders/interests

## Students that develop the assignment

This group is formed by the team that will develop the system. Some of their goals are:

* Use of known technologies and methodologies minimizing the risks to learn new ones.
* Learn how to develop software collaboratively and in a professional way
* Use similar technologies to the group with whom they will work later to minimize incompatibilities
* Agree on a common structure for the Kafka logs with the other developers

## System administrator

This is the person who is in charge of loading the citizens list.

Some of the goals are:

* Use of simple and well-known technologies for input files
* Files that can be read by humans.
* Be able to automate the loading process.
* Be able to debug the loading process in case of failures

## Citizens

These are the final users of the system. Some of their goals are:

* Get access to the system in a simple way.
* Being able to get participate from their homes in a safe way.
* Being able to query their status in the system.
* Being able to create new proposals.
* Being able to vote and discuss proposals from other users.
* Being able to update or change their information in the system, for example, their password (**Optional**)

## Developers of the Participation System

This is the team that will implement the participation system. Some of their goals are:

* Have a simple way to detect if a citizen can participate in the system as soon as possible
* Use of simple technologies that can interoperate with other systems
* Determine the conditions that a proposal must achieve in order to be accepted

## Course teachers

They are responsible for the results of this assignment. Some of their goals are:

* Use technologies that help students acquire skills related with Software Architecture by developing a practical assignment.
* Introduce the students in collaborative and professional software development through TDD (Test driven development) techniques.
* Show the students an example documentation of a software architecture

## City council

This is the responsible of managing the data obtained in the participation system and of configuring the different parameters of it. Some of their goals are:

* View the evolution of the participation system.
* Get information in real time without user interaction.
* See the evolution of proposals, number of votes of each one and the different comments.
* Determine the categories that are interesting for the citizens
* Determine the periods and dates of the proposals
* Determine what words are inappropriate or inadequate (**Optional**)
* See graphical visualizations of the statistics (**Optional**)
* Get notified when some specific event happens (**Optional**)

## Other student developer group

Student team that will implement the Participants module and the Dashboard:

* Maintain good communication with the other team.
* Develop the modules so they can be easily integrated with the modules developed by the other team.

# Quality Attributes

We have identified the following quality attributes:

* **Availability**
  + The system must be able to process data 24x7.
  + The system must be able to support a large number of users, between one and five million.
* **Modifiability**
  + Easily change some parts of the application: Change the parser of input data
  + Easily change some parts of the application: Add an error reporting feature
  + Easily modify some parts of the application: Add other output files to generate the letters
  + Easily modify some parts of the application: Enable password change by users
  + Easily modify some parts of the application: Enable different formats to be used by the web service
  + Easily modify some parts of the application: Enable to change the categories allowed for the proposals.
  + Easily modify some parts of the application: Enable to change the active period of the proposals.

Easily modify some parts of the application: Enable to delete inappropriate proposals

* **Performance**
  + The performance of the data loading system is reasonable
  + Querying information about a user through the web service should be fast
  + The information of the dashboard should be updated quickly
  + The participation system must support a high number of simultaneous users without delays.
* **Security** 
  + The system should warrant the confidentiality of the citizens’ data.
  + Dashboard users should only see the statistics they're allowed to.
* **Testability**
  + It must be testable that the citizens’ data loading process is correct
  + It must be testable that the web service behaves as expected
  + It must be testable that the participation system behaves as expected
  + It must be testable that the information displayed by the dashboard is correct
* **Usability**
  + The data loading system must be easy to use by System administrator users which are familiar with Unix-like tools.
* **Interoperability**
  + This system will be used by the Participation System which will leverage on it for user management. The Participants web service must be used by an automated process that can query the status of a user.
* **Simplicity**
  + The three modules should be simple and easy to develop
* **Deployability**
  + The system should be easily deployable, especially in a cloud based server

## List of Quality Attributes

The list of quality attribute is the following:

| **Code** | **Description Type of Attribute** | | **Module** |  | |
| --- | --- | --- | --- | --- | --- |
| **AT001** | The system must be able to process data 24x7 | Availability | | Participants, participation system, dashboard | |
| **AT002** | Easily modify some parts of the application: Change the parser of input data | Modifiability | CitizensLoader |
| **AT003** | Easily modify some parts of the application: Add an error reporting feature | Modifiability | CitizensLoader |
| **AT004** | Easily modify some parts of the application: Add other output files to generate the letters | Modifiability | CitizensLoader |
| **AT005** | Easily modify some parts of the application: Enable password change by users | Modifiability | Participants |
| **AT006** | Easily modify some parts of the application: Enable different formats to be used by the web service | Modifiability | Participants |
| **AT007** | The performance of the data loading system is reasonable (not too slow, but not critical) | Performance | CitizensLoader |
| **AT008** | The system should warrant the confidentiality of the citizens’ data | Security | CitizensLoader and Participants |
| **AT009** | It must be testable that the web service behaves as expected | Testability | Participants |
| **AT010** | It must be testable that the user loading process is correct | Testability | CitizensLoader |
| **AT011** | The data loading system must be easy to use by system administrator users which are familiar with Unix-like tools. | Usability | CitizensLoader |
| **AT012** | The querying web service must be used by automated processes that can query the status of the system. | Interoperability | Participants |
| **AT013** | The system must be simple and easy to develop | Simplicity | CitizenLoader, Participants, Dashboard and participation System |
| **AT014** | The system should be easily deployable | Deployability | Participants, Dashboard and participation system |
| **AT015** | The system must be able to support a large number of users, between one and five million. | Availability | Dashboard and Participation System | |
| **AT016** | Dashboard users should only see statistics they're allowed to. | Security | Dashboard | |
| **AT017** | The information of the dashboard should be updated quickly | Performance | Dashboard | |
| **AT018** | It must be testable that the information displayed by the dashboard is correct | Testability | Dashboard | |
|  |  |  |  | |
| **AT019** | Easily modify some parts of the application: Enable to change the categories allowed for the proposals | Modifiability | Participation System | |
| **AT020** | Easily modify some parts of the application: Enable to change the active period of the proposals | Modifiability | Participation System | |
| **AT021** | Easily modify some parts of the application: Enable to delete inappropriate proposals | Modifiability | Participation System | |
| **AT022** | The participation system must support a high number of simultaneous users without delays | Performance | Participation System | |
| **AT023** | It must be testable that the participation system behaves as expected | Testability | Participation System | |

Table 2. List of quality attributes and their types

## Quality Attributes and stakeholders

The following table shows which attribute qualities are interesting for which stakeholder:

| **Attributes**  **vs**  **Stakeholders** | **ST-01** | **ST-02** | **ST-03** | **ST-04** | **ST-05** | **ST-06** | **ST-07** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AT001** | X |  | X | X | X |  | X |
| **AT002** | X | X |  |  | X |  |  |
| **AT003** | X | X |  |  | X |  |  |
| **AT004** | X | X |  |  | X |  |  |
| **AT005** | X |  | X |  | X |  | X |
| **AT006** | X |  | X | X | X |  | X |
| **AT007** | X | X |  |  | X |  |  |
| **AT008** | X | X |  |  | X |  | X |
| **AT009** | X | X |  |  | X |  |  |
| **AT010** | X |  | X | X | X |  |  |
| **AT011** | X | X |  |  | X |  |  |
| **AT012** | X |  |  | X | X |  | X |
| **AT013** | X |  |  | X | X |  | X |
| **AT014** | X | X |  |  | X |  |  |
| **AT015** | X |  |  |  | X | X |  |
| **AT016** | X |  |  |  | X |  |  |
| **AT017** | X |  |  |  | X |  |  |
| **AT018** | X |  |  | X | X | X |  |
| **AT019** | X |  |  |  | X | X |  |
| **AT020** | X |  |  |  | X | X |  |
| **AT021** | X |  |  |  | X | X |  |
| **AT022** | X |  | X | X | X | X |  |
| **AT023** | X |  |  |  | X |  |  |

Table 3. List of stakeholders: interests vs quality attributes

# Architecture Constraints

## Technical constraints

We have detected the following set of technical constraints in the project:

| **Code** | **Constraint** | **Background/Motivation** |
| --- | --- | --- |
| **TC001** | Both systems will be implemented in Java | The developer team (ST001) has knowledge of Java |
| **TC002** | The data will be stored in a relational database. | The developer team (ST001) has knowledge of relational databases and there are a lot of libraries to work with relational databases from Java |
| **TC003** | The web service will be based on REST using JSON format | The REST style of web services using JSON is very popular and easy to implement nowadays. |
| **TC004** | The input data format to load data is Excel | Excel is a popular format for data exchange and there are several libraries to process Excel files |
| **TC005** | The output data of the citizens loader module will be a set of text files | In order to facilitate the implementation, text files are the easier format to generate. However, the developer team can optionally implement other generators. |
| **TC007** | Automated testing | The tests should be run automatically and a continuous integration system must be used |
| **TC008** | The web service will be implemented using the Spring Boot web framework | Spring Boot web framework leverages on Spring, which is a well-known framework very popular in Industry. It contains lots of examples and help info that can help students to learn to use it. |
| **TC009** | Events of the system will be managed using Apache Kafka Stream technology | Kafka Stream is a technology used to manage streams in real time. In this case, it will help us manage logs. It lets you do this with concise code in a way that is distributed and fault-tolerant |

Table 4. Technical constraints

## Organizational Constraints

| **Code** | **Constraint** | **Background/Motivation** |
| --- | --- | --- |
| **OC001** | Each system will be implemented by a small team of student developers. | The size of the teams will be between 3 or 4 students. The goal is that students learn to work collaboratively by developing a simple project |
| **OC002** | The structure of the database will be shared by both teams. | Although the projects are designed to enable independent development by each team. The database acts as a glue between both systems so its structure must be shared by both teams |
| **OC003** | The source code will be available as a Github repository | Github offers a very powerful project management tool for this kind of projects. |
| **OC004** | Logs’ format must be agreed and shared by both teams | Although the modules are independent, the logs sent and parsed using Kafka Stream must be format agreed. |

Table 5. Organizational constraints

# System scope and context

The User Management system is decomposed in two modules:

* CitizensLoader: This module will be responsible to convert data from Excel files and load it into the database. The system will be invoked by a system administrator.
* Participants: This module will check if users can participate obtaining information from the database.

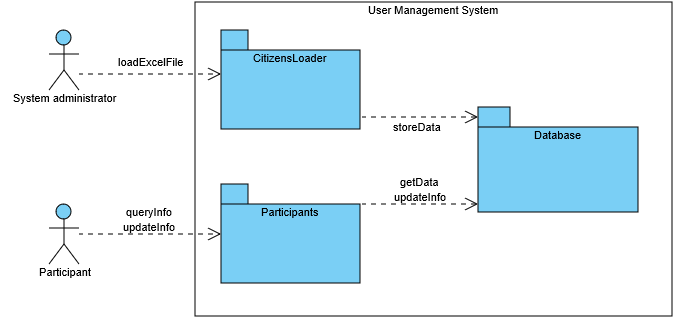


Figure 1. Business Context

The following figure contains a BPMN diagram showing the whole process of all the sub-systems.

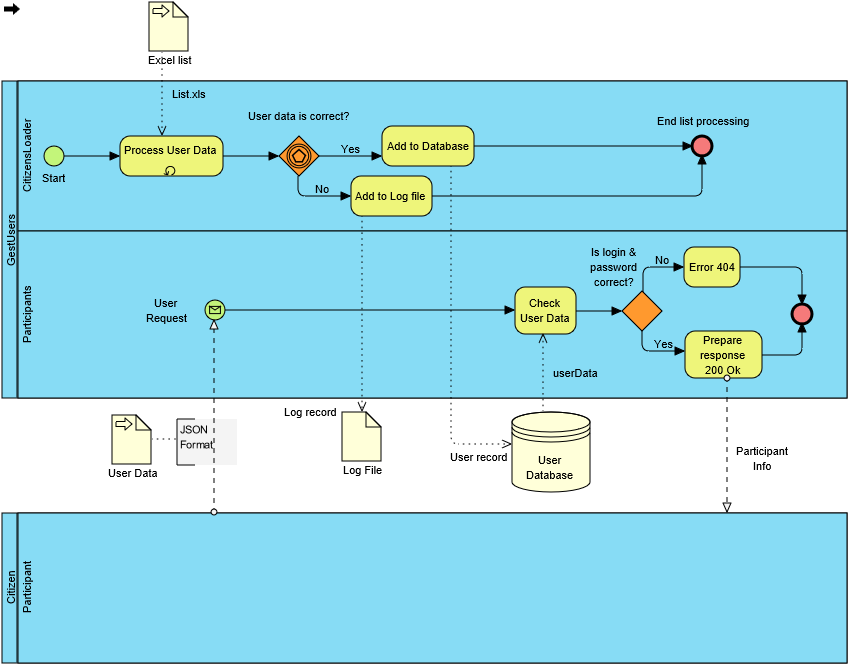


Figure 2. Participants and Citizens loader BPMN Diagram

The Dashboard system will be responsible of showing data of proposals, their number of votes and their comments in real time.

The Participation System will be in charge of managing the creation of proposals and the interaction of the users with the proposals (comment, vote…). It also allows the administrators to moderate proposals, categories and the word filter.

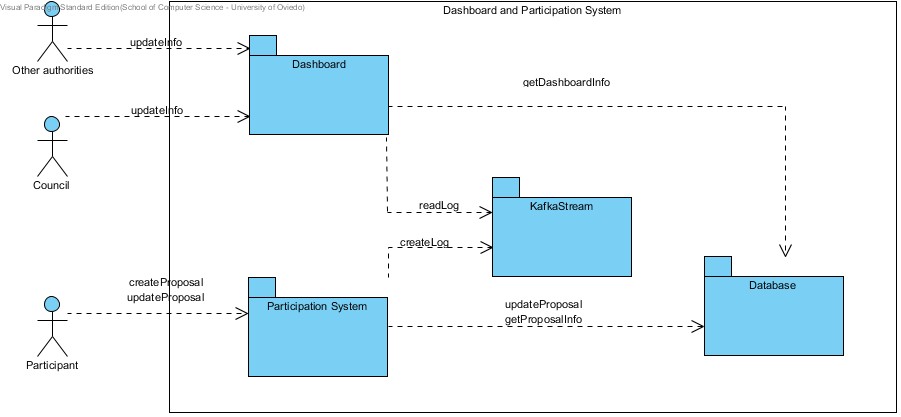


Figure 3. Participation System and Dashboard Business Context

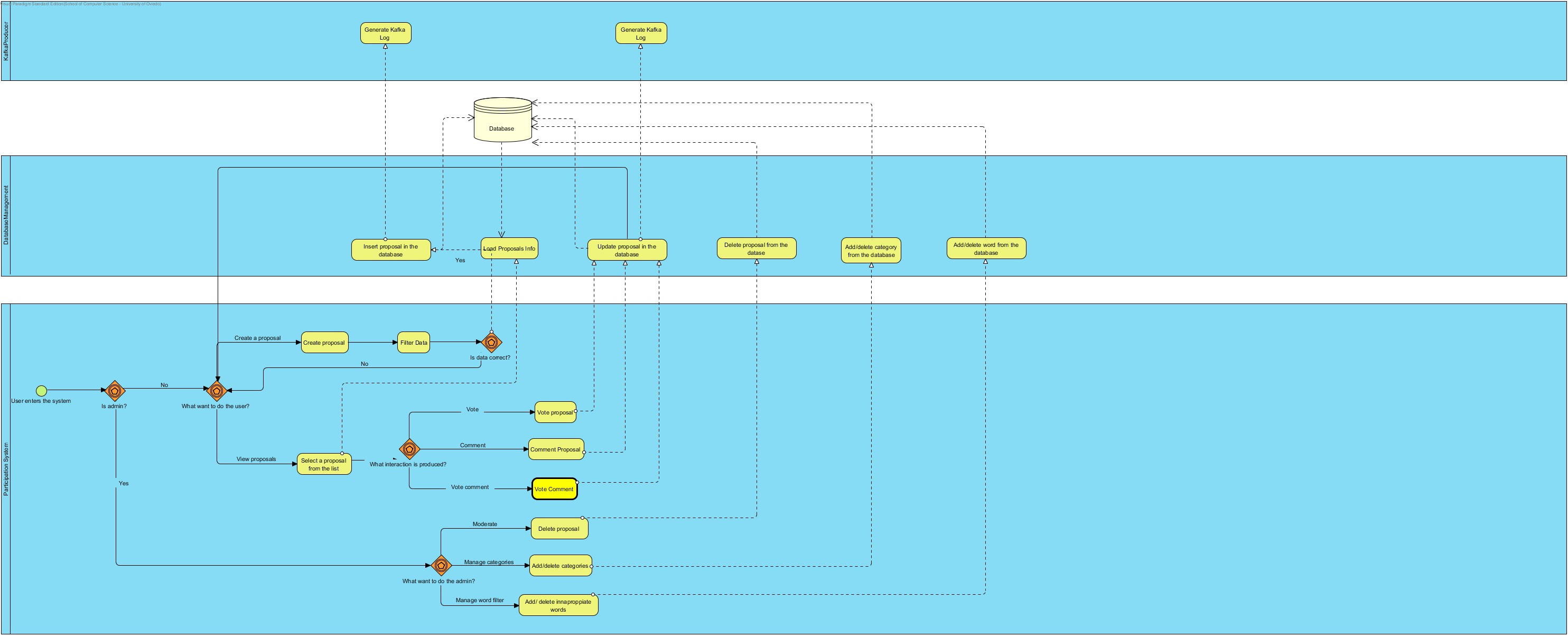


Figure 4. Participation System BPMN Diagram

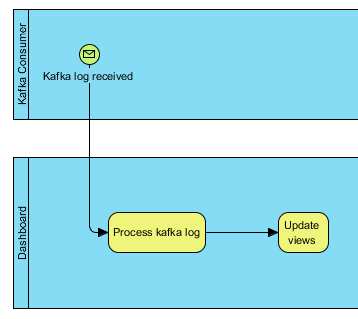


Figure 5. Dashboard BPMN Diagram

# Quality Scenarios

The table below contains the quality scenarios that have been identified:

| Scenario | Source Stimulus | Stimulus | Environment | Artifact | Response | Measure | Affected  Attribute Quality |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Participation System | Ask information about a user | Runtime | Participants | Participation System obtains the required information in less than 15seg at any time in the day | The required information is obtained | **AT001** |
| 2 | Student developer | A new parser is introduced | Development | Parser | Change is successfully introduced | The system is compiled and passes all the tests without errors | **AT002** |
| 3 | Student developer | A new option is implemented for the report file | Development | ReportWriter, DBUpdate and Parser | The option is implemented with minimal changes that affect only the report writer module | Less than one day of work | **AT003** |
| 4 | Student developer | A new output format is added | Development | Participants and DBManagement | The new output format is included with minimal changes to existing code. | Less than one day of work | **AT004** |
| 5 | Student developer | The option to change user's password is introduced | Development | Participants and DBManagement | The password of a user is successfully changes | Less than one day of work | **AT005** |
| 6 | Student developer | A new format is added to the web service | Development | Participants | The new format is implemented | Less than 2 days of work | **AT006** |
| 7 | System administrator | Load an Excel file into the System (DB) | Runtime | Parser, DBUpdate and ReportWriter | Loading an excel file without errors is done in a reasonable time. | < 1 second for each 10 Participantsparticipant | **AT007** |
| 8 | Student developer | Load an Excel file into the system (DB) | Development/  Runtime | Parser, DBUpdate and ReportWriter (Optional) | Loading data should be done in a safe way | It is not possible to get access to the users’ personal data except by the system administrator who cannot get access to the password. | **AT008** |
| 9 | Participants | Get access to the application | Runtime | Participants | A user can get access to his data but not to other user's data | Access to data is enabled only if the pair user name/password is correct | **AT009** |
| 10 | System administrator | Loads an excel file into the DB | Runtime | Parser, DBUpdate and ReportWriter | The loading process is made in a reliable way and it is possible to check that the data has been loaded | There are no errors in the database, no repeated record, and no citizen has less information than expected | **AT010** |
| 11 | System administrator | Loads an excel file into the DB | Runtime | Parser, DBUpdate and ReportWriter | The loading process behaves in a usual way and the options available to run the system are easy to understand | The system shows help options if the user asks for them. The error messages and other information can be understood by technical people | **AT011** |
| 12 | Citizen Participation System | Access to the web service | Runtime | Participants | The participation System requests information about a user by passing a combination of user name and password | A 200 OK response is sent with the correct format if the combination is OK or a failure information is returned | **AT012** |
| 13 | Student developer | Develops the system | Development | Participants  CitizensLoader  Participation System  Dashboard | The student developers can implement the system | The system can be implemented and testes in 2/3 weeks by third year undergraduate students. | **AT013** |
| 14 | System administrator | Deploys the system | Deployment | CitizensLoader, Participants  Participation System  Dashboard | The system is deployed in a production environment | The system can be deployed by a system administrator in less than an hour. | **AT014** |
| 15 | Dashboard System | Monitor statistics | Runtime | Dashboard | The Dashboard System retrieves and displays the information from the Stream logs. | The required information is obtained and updated on real time. | **AT015** |
| 16 | City council | Access to the web system | Runtime | Dashboard | The Dashboard System only shows statistics of the view that the user has defined | Each type of user has a view predefined and he/she can’t modify it. | **AT016** |
| 17 | City council | Statistics modified in real time | Runtime | Dashboard | The information that the user is seeing is updated | Automated updates when the data is changed | **AT017** |
| 18 | City council | View statistics | Runtime | Dashboard | The information shown is the same that is currently in the database | The data that the users see is stored in the database | **AT018** |
| 19 | Participation System | Access the system | Runtime | Participation System | The system must be accessible by the users even when a great number of users are connected at the same time | The system is up and the users can access the system without any delay | **AT015** |
| 20 | Council | Change the proposal categories | Runtime | Participation System | The council is able to add, remove and edit the categories of the system | The council can add and remove categories at any moment | **AT019** |
| 21 | Council | Change the active period of the proposals | Runtime | Participation System | The council is able to change during how many time the proposals are active | The council can change the periods of the proposals at any given time | **AT020** |
| 22 | Council | Delete inappropriate proposals | Runtime | Participation System | The council is able to delete that proposals that are considered inappropriate | The council can delete proposals that are determined as inappropriate at any time with minimal effort | **AT021** |
| 23 | Participation System | High number of users using the system simultaneously | Runtime | Participation System | The users don’t have any delay and will no notice the overload of users in the system. | The response times of the system are the same as when a low number of users are using it | **AT022** |
|  |  |  |  |  |  |  |  |
| 24 | System Administrator | Make proposals, comments, etc... | Runtime | Participation System | The participation system is reliable and can be easily tested by the administrator. | There are no errors in the participation system tests. | **AT023** |

Table 6. List of quality scenarios

# Views

In the following paragraphs the identified the views that will be documented following the learning guide instructions.

|  |  |  |  |
| --- | --- | --- | --- |
| View | Stakeholders | Quality Attributes | Scenarios |
| Context | ST-01, ST-02, ST-03, ST-04, ST-05 | AT011, AT013, AT14 | 11, 13, 14 |
| CitizensLoader | ST-01, ST-02, ST-04, ST-05 | AT002, AT003, AT004, AT007, AT008 y AT010, AT011, AT013, AT014 | 2, 3, 4, 7, 8, 10, 11, 13, 14 |
| Participants | ST-01, ST-03, ST-04, ST-05 | AT001, AT005, AT006, AT008, AT009, AT012, AT013, AT014 | 1, 5, 6, 8, 9, 12, 13, 14 |
| Dashboard | ST-01, ST-05, ST-06 | AT001,AT013,AT014, AT015, AT016, AT017, AT018 | 13,14,15, 16, 17, 18 |
| Participation System | ST-01, ST-02, ST-03, ST-05, ST-06 | AT001, AT008, AT013, AT014, AT019, AT020, AT021, AT022, AT023 | 13, 14, 19, 20, 21, 22, 23, 24 |

In the catalogues and views we have described both the mandatory and some optional elements. The students can ignore those optional elements that they are not going to implement.

## Context

The System view is divided in four main sub-systems.

### Main overview

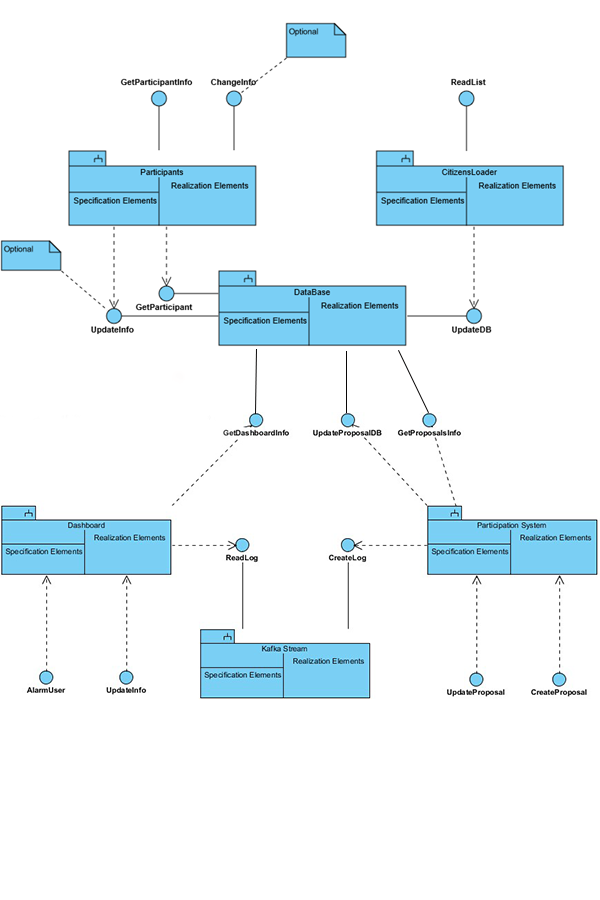


Figure 6. Context view

### Elements Catalogue

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| CitizensLoader | It introduces citizens data in the system. It reads an Excel file with data, generates passwords, personal letters and reports any errors. |
| Participants | This is the module used by citizens to check that their information is available in the system. They can optionally change some of their personal information and their password. |
| DataBase | This module encapsulates database access. |
| Dashboard | This module is used by the councils to check the participation and the results that are in the database. As well as manage the logs produced, and received by Kafka Stream. |
| Participation System | The module is used by the citizens in order to participate in proposals. All the information is stored in the database. It generates the kafka logs to communicate with the Dashboard. |

#### Relationships

Citizens data are introduced in the system through the interface *ReadList* from module *CitizensLoader*. For each user, a password is generated as well as a personalized letter with information about the user.

That interface sends the data to the database through the interface *UpdateDB* from the DataBase module.

The *Participants* module allows an external system to check the information about a user through the web service *GetParticipantInfo*. In order to check the information, *Participants* asks data to the *DataBase* module through the *GetParticipant* interface.

Optionally, it is possible to implement the interface *ChangePassword* that will allow a user to change her password. In order to do that, the *Participants* module requests the *DataBase* to change the password through the *UpdatePasswd* interface.

The Dashboard looks ahead for any new data in the database through GetDashboardInfo interface, for the specified council. It provides the UpdateInfo interface to retrieve the data to the view.

It also provides AlarmUser, an interface used to retrieve the logs to each specified council. It is connected to Kafka Stream through an interface called ReadLog that will manage the Kafka Subsystem and its respective subscriber operations.

Participation System provides two interfaces to introduce information in the system: *CreateProposal* allows the users to create new proposals that will be inserted in the database through the interface *UpdateProposalDB.* The interface *UpdateProposal* manages all the data insertion related to proposals (comments and votes). It also uses the *UpdateProposalDB* interface to store the information in the database and the GetProposalInfo interface to get the information about the proposals and show it to the user.

All the events produced in the participation system are notified by means of Kafka logs using the *createLog* interface provided by Kafka Stream.

#### Interfaces/Ports

##### CitizensLoader

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| ReadList | Interface | Command line invocation | This interface will be invoked from the main application as a console program |

##### Participants

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Interface | Tipo | Tecnología | | Propiedades | |
| GetParticipantInfo | | Interface | Web Service | | This interface will be invoked through an HTTP request |

##### DataBase

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface | Tipo | | Tecnología | | Propiedades | |
| GetParticipant | Interface | | Method invocation | | Returns data from citizens | |
| UpdateDB | Interface | | Method invocation | | Inserts into the database data about a citizen included its password | |
| GetDashboardInfo | | Interface | | Method invocation | | Handles the information of the proposals. |
| UpdateProposalDB | | Interface | | Method Invocation | | Updates the information of the proposals |
| UpdatePasswd | | Interface | | Method Invocation | | Updates the password of a user in the database |
| GetProposalInfo | | Interface | | Method Invocation | | Extracts from the database the information about the proposals |

##### Dashboard

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| UpdateInfo | Interface | Method/Web Service | Allows the view to be updated with the data. |
| AlarmUser | Interface | Web Service | It will alert the corresponding user with the logs it need to see |

##### Kafka Stream

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface | Type | | Technology | | Properties | |
| ReadLog | | Interface | | Method invocation | | Handles the subscriber part of KafkaStream. |
| CreateLog | | Interface | | Method Invocation | | Handles the produce part of KafkaStream. |

##### Participation System

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface | Type | | Technology | | Properties | |
| UpdateProposal | | Interface | | Method/Web Service | | Allows the user to make changes in the information of a proposal |
| CreateProposal | | Interface | | Method/Web Service | | Allows the user to create a new proposal |

#### Behaviour

##### CitizensLoader

See 9.2.2.3.4.

It can also do the following options:

* (**Optional**) the subsystem that generates the letters could implement the Adapter pattern which would enable to generate the letters in different formants in the future (Word, ODT, PDF, RTF, etc.).
* (**Optional**) If the file contains errors, those errors should be detected and a report should be generated for its later treatment
* (**Optional**) The parser of input data should be configurable using an adapter pattern to allow input data in different formats (Excel, TXT, etc.).

##### Participants

It allows users to get access into the system to check if they can participate, using the information that they received in the letter. The users may not get access directly by a web browser, but through an external paticipation system that invokes the Participants module as a web service.

##### DataBase

All the operations done in this module will be integrated in a *Facade pattern* which will contain the operations that offer access to the database. It encapsulates all the operations that affect the database.

## Citizens List

### Main overview



Figure 7. Citizens list view

### Catalogue of Elements

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Parser | Reads data from the Excel file and transforms them into an in-memory object container that can be later iterated to insert the data in the database.  It will also generate the *password* of the citizen as well as the personal letter.  During the design and implementation this component can be divided into the sub-components needed to separate these services following the quality attributes AT002, AT003, AT004 and AT007. |
| DBUpdate | Encapsulates all the database operations using interfaces to allow the database access to be separated from some specific database implementations. |
| ReportWriter | It receives the pieces of data that were not possible to insert into the database as well as the reasons and writes a report containing all that information in a human-readable way |

#### Relationships

The *Parser* component receives the input file in Excel format and reads and converts the information about the different users. It generates a new password for each user and adds the information to the database using the *DBUpdate* component.

(**Optional**) If there are any errors during the loading phase (duplicated DNIs, empty DNI fields, etc.) or if the database component returns an error, this information will be notified to the Reportwriter component through the *WriteReport* interface.

#### InterfacesPortsPortsPorts

##### Parser

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| ReadList | Interface | Method invocation | Read the Excel file with the citizens data. |
| Rlist | Port |  | Creates the needed subcomponents of the parser to process the input file. |
| Insert | Interface (Required) | Method invocation | It calls a method in the *DBUpdate* component to insert the information in the database. |
| InserR | Port |  | Verifies the data and creates the object to send to the *DBUpdate* component. |

##### DBUpdate

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| Insert | Interface | Method invocation | Receives and object with the information to insert in the database. |
| InsertP | Port |  | Verifies input data and generates and error if there is a lack of some mandatory attribute. |
| WriteReport | Interface (Required) | Method invocation | Calls a method from the *ReportWriter* component to write a new item in the report file. |
| WreportR | Port |  | Verifies the data to write |

##### ReportWriter

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| WriteReport | Interface | Method Invocation | Receives the data to write in the report file. |
| WreportP | Port |  | Adds data at the corresponding date and time. |

##### Parser

Introduces the citizencitizens data in the system obtained from Excel files that contain a row for each citizen. Each row (except the first one that contains the headings) contains the following columns:

* First nameName (string)
* Last name (string)
* Birth date
* Email (string that follows the email format conventions)
* ID
* Address
* Nationality
* NIF (string that follows the NIF format with digits followed by a verification letter)
* Polling station code (an integer)

Invocation will be done through a batch program executed in the command line by the system administrator. During the import process a password will be generated so the combination of email/password enable a user to enter the system and participate in the system receive information about the polling station code where the user can participate.

This component will also generate personal emailsletters communicating the user that he has been added to the system with a user name (his email) and a password.

##### DBUpdate

It updates the database. See 9.1.2.4.5.

##### ReportWriter

(**Optional**) It stores in a text file information about the errors that were produced by the conversion process. The basic information to store is:

* Date
* Time
* Original Excel file
* Error information (with all the needed information)

### Context Diagram

See 9.1.

### Rationale

The main design decisions of this sub-system are:

| Scenario | Quality attributes | Justification |
| --- | --- | --- |
| 2 | AT002 | Access to the parser using an Adapter pattern facilitates to change the implementation without affecting other parts of the application. |
| 3 | AT003 | Defining an interface and an object for error reporting allows to add this functionality later. |
| 5 | AT005 | Using a relational database will improve the performance of accessing information about users. |
| 6 | AT006 | Using a relational database that offer security aspects can improve the security of the system. Sending the login name and password by regular mail avoids that the information can be accessed electronically. |
| 8 | AT008 | Using a standard database which can be queried using SQL can allow the students to verify that the data has been correctly loaded. |
| 10 | AT010 | The use of a batch application that can be executed manually or configured for its automatic execution is a common practice for system administrators. |
| 14 | AT014 | A batch application can be directly executed without any special needs for deployment |

## Participants

### Main overview

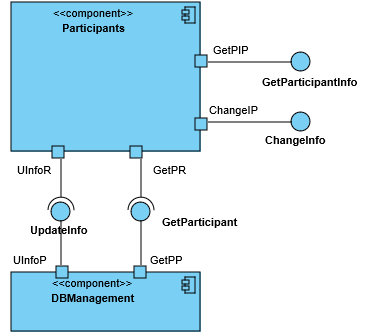


Figure 8. Participants View

### Catalogue of elements

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Participants | It offers two web services: *GetParticipantInfo*, which allows to obtain information about a user and (Optional) *ChangePassword* that allows to change the password of a user. |
| DBManagement | It offers two interfaces: GetParticipant, that returns the data of a participant from the database and (Optional) *UndateInfo*, to update a password change in the database. |

#### Relationships

The ParticipantParticipation System invokes *Participants* using a web service call which is processed by *GetParticipantInfo* (sending *email/password*) and it gets access to the DBManagement system using the interface *GetParticipant*. If the email/password are correct the data is returned as a JSON response.

(**Optional**) The user can invoke *Participants* through a web browser to change his password invoking *ChangePassword* and sending the parameters *email/password/newPasswod*. It will invoke the interface *UndateInfo* to modify the password using the *DBManagement* component.

#### Interfaces

##### Participants

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| GetParticipantInfo | Interface | Web service | Allows to get access to a citizen data through the email/password combination |
| GetPIP | Port |  | Validates a user before asking the data. |
| ChangePassword | Interface | Web service | Allows to change a password using the combination: *email/password/newPasswod*. |
| ChangeInfo | Port |  | Validates a user before asking to change his password. |
| ChangeIP | Port |  | Validates a user before asking to change the password |
| UndateInfo | Interface (Required) | Method invocation | Asks a password change for a user. |
| UInfoR | Port |  |  |
| GetParticipant | Interface (Requerida) | Method invocation | Asks information for the user |
| GetPR | Port |  |  |

##### DBManagement

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| UndateInfo | Interface | Method invocation | Handles the password change of a user. |
| UInfoP | Port |  |  |
| GetParticipant | Interface | Method invocation | Handles the information request for the user. |
| GetPP | Port |  |  |

#### Behaviour

##### Participants

It implements a REST web service to handle requests of information about users. The POST HTTP request will be done to the following address:

<WebServiceURI>/user

where <WebServiceURI> represents the URI where the web service has been deployed. The POST request contains JSON data with the following structure:

{"login": email, "password": password}

In case that the (email, password) combination are available in the database the response will be 200 OK with the a JSON body of the form:

{ "firstName": Nombre,

"lastName": Apellidos,

"age": Age (will be obtained from user's birth date and current time)

"ID": User ID,

"email": email

}

In case that the (email, password) is incorrect, the response will be 404 Not found.

(**Optional**) It is possible to implement some HTML interface so the web service can be used by humans through a web browser.

(**Optional**) The web service can be extended to allow users to change their password.

##### DBManagement

This component encapsulates all the database access so it can be easy to change the underlying database system.

### Context Diagram

See 9.1.

### Rationale

The main design decisions have been:

| Scenario | Quality Attributes | Justification |
| --- | --- | --- |
| 1 | AT001 | Using a REST Web Service leverages on HTTP technology and makes it easier to deploy the system in some infrastructure with high availability. |
| 4 | AT005 | The encapsulation of model features that affect the database during development and the use of a MVC framework will facilitate the addition of functionalities like password change. |
| 6 | AT006 | Using a Web framework like Spring Boot will facilitate the development of common web features like content negotiation |
| 8 | AT008 | Accessing by *email/password* is considered secure enough for this process. Passwords should be stored encrypted. |
| 9 | AT009 | The development of a REST web service based on JSON formats will facilitate the development of tests. The Spring Boot framework contains several tools for unit and integration testing of web applications that can be used. |
| 12 | AT012 | The use of a REST web service enables the automatic access to the system through a software client |
| 13 | AT013 | The web service API defined is simple and contains the minimal functionality. Leveraging on Spring Boot web framework will facilitate the development by the students given that the framework has solutions for all the required functionality |

## Dashboard

### Main overview

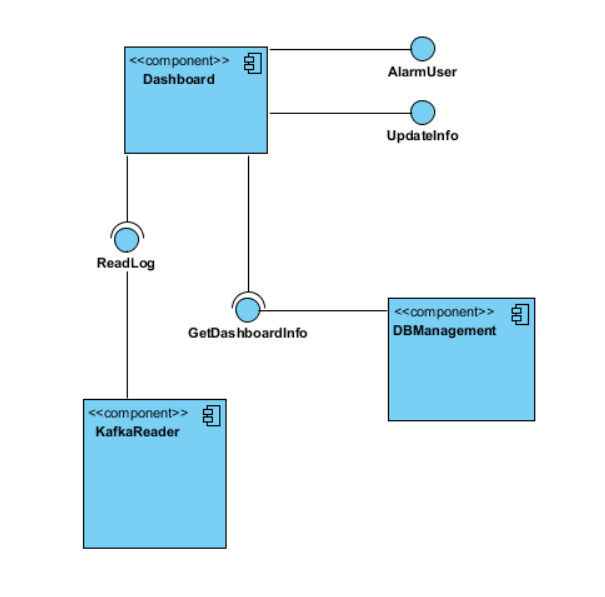


Figure 9 Dashboard View

### Catalogue of elements

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Dashboard | It offers an interface: *UpdateInfo*, which allows the view to update the data taken from the database. Inside it should compare the actual data and the data from the database and update only what it is needed. (Optional) Another web service called AlarmUser would allow to send an alarm to a user depending on the logs. |
| KafkaReader | It offers an interface ReadLog that reads data from KafkaStream. |
| DBManagement | It offers an interface GetDashboardInfo that returns the data of all the proposals. |

#### Relationships

The Dashboard System invokes *Dashboard* using a recursive web service call which is processed by *UpdateInfo* and it gets access to the DBManagement system using the interface *GetDashboardInfo*. If there is new information, the view is updated.

(**Optional**) The user can invoke *Dashboard* and the information can be shown using a graphical format.

(**Optional**) The user can invoke Dashboard and while the information is updated, AlarmUser can be called if a specific event happens in the ReadLog to show a message to the user.

#### Interfaces

##### Dashboard

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| AlarmUser | Interface | Web Service | It will alert the corresponding user with the logs it need to see |
| UpdateInfo | Interface | Method/Web Service | Allows the view to be updated with the data. |

##### DBManagement

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface | Type | | Technology | | Properties | |
| GetDashboardInfo | | Interface | | Method invocation | | Handles the information of the proposals. |

##### KafkaReader

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface | Type | | Technology | | Properties | |
| ReadLog | | Interface | | Method invocation | | Handles the subscriber part of KafkaStream. |

#### Behaviour

##### Dashboard

It implements a method that will be called recursively to update information.pdate information.pdate

The information should be delivered back to the view and updated dynamically without the interaction of the user.

(**Optional**) A graphical view of the data returned in a JSON could be displayed.

(**Optional**) If some event on KafkaStream happens, the log related to a user should be sent to that user.

### Context Diagram

See 9.1.

### Rationale

The main design decisions have been:

| Scenario | Quality Attributes | Justification |
| --- | --- | --- |
| 1 | AT001 | Using a REST Web Service leverages on HTTP technology and makes it easier to deploy the system in some infrastructure with high availability. |
| 13 | AT013 | Using Spring Boot framework allows us to program faster using the infrastructure that the framework offers. |
| 14 | AT014 | With Spring boot it is easy and fast to deploy the system |
| 15 | AT015 | Using a MVC pattern will allow us to increase the concurrency of the system to support a lot of users |
| 16 | AT016 | Using credentials for authentication will allow us to separate which user sees what kind of information. |
| 17 | AT017 | Using a web service allows a fast update.e allows us to quickly upde allows A web service provides fa |
| 18 | AT018 | Using Spring Boot will allow us to test the displayed information using unit testing. |

**Participation System**

**Main Overview**

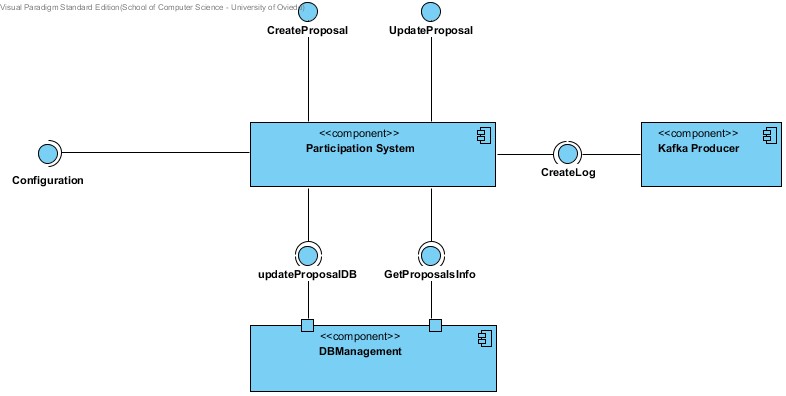


Figure 10 Participation System View

**Catalogue of elements**

***Elements***

| Element | Properties |
| --- | --- |
| Participation system | It offers three services: GetConfiguration, which allows to obtain information about the configuration of the system. Council members can configure different parameters like proposal categories, dates in which they will be active, not allowed words, etc. The portal administrator can modify the minimal number of support votes. CreateProposal allows the users to create new proposals. UpdateProposal allows the users to interact with the proposals (create comments, vote etc) |
| DBManagement | It offers two interfaces: CreateProposal, that inserts the data of a new proposal into the database and UpdateProposal that modifies the number of votes, comments, etc. |
| Kafka Producer | It offers one service: CreateLog, which generates Kafka logs for the different events that are produced in the Participation System |

***Relationships***

The Participation System invokes the method getConfiguration which will receive the configuration parameters established by the council and the portal administrator.

It gets access to the DBManagement system using the interfaces *CreateProposal* and *UpdateProposal*.

***Interfaces***

Participation System

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| GetConfiguration | Interface | Web service | Allows to obtain information about the configuration of the system. Council members can configure different parameters like proposal categories, dates in which they will be active, not allowed words, etc. The portal administrator can modify the minimal number of support votes. |
| CreateProposal | Interface | Method/Web Service | Allows the users to create new proposals configuring different parameters. |
| UpdateProposal | Interface | Method/Web Service | Allows the users to interact with the proposals, vote and comment them and also to vote comments. The votes can be positive or negative. |

DBManagement

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| InsertProposal | Interface (Required) | Method invocation | Inserts a new user proposal if it fullfills the requirements specified by the Council. |
| UpdateProposal | Interface (Required) | Method invocation | Updates a proposal (vote and/or comment). |

##### KafkaProducer

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Interface | Type | | Technology | | Properties | |
| CreateLog | | Interface | | Method invocation | | Handles the producer part of KafkaStream. |

***Behaviour***

Participation System

It allows users to create proposals and interact with the existing ones. The users can vote proposals positively or negatively. The can also comment proposals to generate feedback. Comments can be also voted. The system implements a filter to avoid inappropriate content in the platform.

KafkaProducer

For each one of the events produced in the Participation System (create proposal, vote proposal, comment proposal, vote comment…) the kafka producer must generate a log in order to inform the dashboard about the update.

DataBaseManagement

All the operations done in this module will be integrated in a *Facade pattern* which will contain the operations that offer access to the database. It encapsulates all the operations that affect the database.

**Context diagram**

See 9.1.

**Rationale**

|  |  |  |
| --- | --- | --- |
| Scenario | Quality attributes | Justification |
| 13 | AT013 | Using Spring Boot framework allows us to program faster using the infrastructure that the framework offers. |
| 14 | AT014 | With Spring boot it is easy and fast to deploy the system |
| 19 | AT015 | Using a well-optimized database for concurrency will allow to have a high performance accessing the information about the proposals. |
| 20 | AT019 | Using DAO pattern for the database management allow us to make modifications over the proposals stored in the database in a very easy way. |
| 21 | AT020 | Using DAO pattern for the database management allow us to make modifications over the proposals stored in the database in a very easy way. |
| 22 | AT021 | Using DAO pattern for the database management allow us to make modifications over the proposals stored in the database in a very easy way. |
| 23 | AT022 | Making code that is prepared for managing concurrency and a good control of the concurrent accesses of the database will allow us to manage simultaneous users. |
| 24 | AT023 | Using Junit tests we can prove the well-functioning of the participation system and all their functionalities. |

# References

ANSI/IEEE 1471. (2000). *Recommended Practice for Architectural Description of Software-Intensive Systems.* ANSI/IEEE.

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice, Second Edition.* Boston: Addison Wesley.