

School of Computer Science Engineering

2017-05-09

GestUsers: User Management System

**CITIZEN PARTICIPATION**

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 2015-16.

**Grado de Ingeniería Informática del Software**

|  |  |
| --- | --- |
| logo_eng | **Logo-COLOR-INGLES** |

**GestUsers: User Management System**

**Authors**: **Claudia Castañón Ferreiro**

**Carla Fernández González**

**Pelayo Figaredo García**

**Sara González Tomillo**

**Daniel Ortea Mendéz**

**Alejando Pazos Noval**

**Date**: 28/03/2017

**Version**: 2017.EN.003

Table of Contents

[1 Introduction and Goals 6](#_Toc482114755)

[2 Requirements 7](#_Toc482114756)

[2.1 CitizensLoader 7](#_Toc482114757)

[2.2 Participants 8](#_Toc482114758)

[2.3 Participation system 8](#_Toc482114759)

[2.4 Dashboard 8](#_Toc482114760)

[3 Methodology 10](#_Toc482114761)

[4 Stakeholders 11](#_Toc482114762)

[4.1 Students that develop the assignment 11](#_Toc482114763)

[4.2 System administrator 11](#_Toc482114764)

[4.3 Citizens 11](#_Toc482114765)

[4.4 Course teachers 12](#_Toc482114766)

[4.5 Council 12](#_Toc482114767)

[4.6 Team mates 12](#_Toc482114768)

[5 Quality Attributes 13](#_Toc482114769)

[5.1 List of Quality Attributes 13](#_Toc482114770)

[5.2 Quality Attributes and stakeholders 15](#_Toc482114771)

[6 Architecture Constraints 16](#_Toc482114772)

[6.1 Technical constraints 16](#_Toc482114773)

[6.2 Organizational Constraints 16](#_Toc482114774)

[7 System scope and context 18](#_Toc482114775)

[8 Quality Scenarios 20](#_Toc482114776)

[9 Views 24](#_Toc482114777)

[9.1 Context 24](#_Toc482114778)

[9.1.1 Main overview 24](#_Toc482114779)

[9.1.2 Elements Catalogue 25](#_Toc482114780)

[9.2 Citizens List 28](#_Toc482114781)

[9.2.1 Main overview 28](#_Toc482114782)

[9.2.2 Catalogue of Elements 28](#_Toc482114783)

[9.2.3 Context Diagram 30](#_Toc482114784)

[9.2.4 Rationale 30](#_Toc482114785)

[9.3 Participants 31](#_Toc482114786)

[9.3.1 Main overview 31](#_Toc482114787)

[9.3.2 Catalogue of elements 31](#_Toc482114788)

[9.3.3 Context Diagram 33](#_Toc482114789)

[9.3.4 Rationale 33](#_Toc482114790)

[9.4 Participation System 34](#_Toc482114791)

[9.4.1 Main Overview 34](#_Toc482114792)

[9.4.2 Catalogue of elements 34](#_Toc482114793)

[9.4.3 Context diagram 36](#_Toc482114794)

[9.4.4 Rationale 36](#_Toc482114795)

[9.5 Dashboard 37](#_Toc482114796)

[9.5.1 Main overview 37](#_Toc482114797)

[9.5.2 Catalogue of elements 37](#_Toc482114798)

[9.5.3 Context Diagram 38](#_Toc482114799)

[9.5.4 Rationale 38](#_Toc482114800)

[10 References 39](#_Toc482114801)

# Introduction and Goals

The goal of this document is to describe the structure of an architecture of a Citizen Participation System for a medium size city. Although the system that we describe has its own functionality, the main goal is that it will be part of a general system of citizen participation.

This document describes the first and third deliverables of the laboratory assignment of the course "*Software Architecture*”. The course is part of the Degree in Software Engineering, School of Computer Science Engineering, University of Oviedo.

The system is divided in four parts: CitizensLoader, to load data about citizens; Participants, to check if a user can participate; ParticipationSystem, to allow users to create, vote and comment different proposals; and Dashboard to allow the Council staff to view the participation system in real time.

In the first deliverable, the students implemented the software for CitizensLoader and Participants in two teams of 3 or 4 students during 3 weeks.

And for the next deliverable, each team was responsible to complete the document with the Dashboard and ParticipationSystem modules, following the specifications given by the teachers.

In the last deliverable, our task was to merge all the modules together so that they could work as one, as well as finishing this documentation.

# Requirements

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.
* ParticipationSystem: to allow users to create, vote and comment different proposals.
* Dashboard: to allow the Council staff to view the participation system in real time.

## 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 users can make different proposals that will be supervised by the Council, who will be able to delete the ones which are considered non-appropriate, as well as configure proposal categories, activation dates, not allowed words…. This proposals can be voted by the users in order to make them more or less popular, and also can be commented in order to generate discussion about each one.

To be accepted, a proposal should have a minimum number of votes, which the portal administrators can configure. Once it is accepted, the portal administrators receive a notification and communicate with the corresponding parliament.

Every proposal, as well as its support and comments should be recorded in a log connected to a Kafka Stream.

(**Optional**) It could be possible to order comments by popularity or chronological order.

(**Optional**) Comments can be voted.

(**Optional**) Each user can consult the proposals and comments that he has emitted as well as edit them.

(**Optional**) Users can share proposals in different social platforms in a simple way through a share buttons section in the web page.

## Dashboard

The system will offer a dashboard to the Council staff, councilmen and other authorities in order that they view the evolution of the participation system in real time. The different type 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. 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 ADDAtribute-Driven Design) methodology (Bass, Clements, & Kazman, 2003) and the SEInorm ￼(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. Course Teachers
5. Council
6. Team mates

| Code | Stakeholder | Interests (Modules) |
| --- | --- | --- |
| ST-01 | Students | All modules |
| ST-02 | System administrator | Load files |
| ST-03 | Citizens | Check data and participate |
| ST-04 | Course Teachers | All modules |
| ST-05 | Council | Dashboard |
| ST-06 | Team mates | All modules |

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.

## 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.
* Be able to publish different proposals and vote and comment them.
* Being able to update or change their information in the system, for example, their password (**Optional**)

## 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

## Council

The group of people who are monitoring the dashboard. Some of their goals are:

* The ability to see the ongoing voting on the proposals
* See the comments posted by citizens regarding the proposals
* Being able to see a graphical representation of the proposal’s data (**optional**)
* Receive alarms related to events on the dashboard (**Optional**)

## Team mates

They are responsible for making the complementary parts of the assignment. Their goals are exactly the same as the other students, and they have to work together in order to complete the work.

# Quality Attributes

We have identified the following quality attributes:

* **Availability**
  + The system must be able to process data 24x7.
  + The results and information are shown in the dashboard in real time.
* **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
* **Performance**
  + The performance of the data loading system is reasonable
  + Querying information about a user through the web service should be fast
* **Security** 
  + The system should warrant the confidentiality of the citizens data
* **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 no data is lost in case of an accident
* **Usability**
  + The data loading system must be easy to use by System administrator users which are familiar with Unix-like tools.
  + The participation system must be easy to use by all different kinds of public, independently of their technological skills.
* **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 two modules should be simple and easy to develop
* **Deployability**
  + The system should be easily deployable, especially in a cloud based server
* **Scalability**
  + The system should be easily adapted if the demand for it increases
* **Accesibility**
  + Results can be graphically represented.

## 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 | |
| **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 and Participants CitizensLoader and Participants | |
| **AT014** | The system should be easily deployable | Deployability | Participants | |
| **AT015** | The results and information are shown in the dashboard in real time. | Availability | Dashboard | | |
| **AT016** | It must be testable that in case of an accident no data is lost | Testability | Dashboard, ParticipationSystem | | |
| **AT017** | Results can be graphically represented. | Accesibility | Dashboard | | |
| **AT018** | Querying information about a user through  the web service should be fast | Performance | Participants | | |
| **AT019** | The participation system must be easy to use by all different kinds of public, independently of their technological skills. | Usability | ParticipationSystem | | |
| **AT020** | The system should be easily adapted if the demand for it increases | Scalability | ParticipationSystem | | |

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** | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AT001** | X |  | X | X | X | X | |
| **AT002** | X | X |  |  | X | X | |
| **AT003** | X | X |  |  | X | X | |
| **AT004** | X | X |  |  | X | X | |
| **AT005** | X |  | X |  | X | X | |
| **AT006** | X |  | X | X | X | X | |
| **AT007** | X | X |  |  | X | X | |
| **AT008** | X | X |  |  | X | X | |
| **AT009** | X | X |  |  | X | X | |
| **AT010** | X |  | X | X | X | X | |
| **AT011** | X | X |  |  | X | X | |
| **AT012** | X |  |  | X | X | X | |
| **AT013** | X |  |  | X | X | X | |
| **AT014** | X | X |  |  | X | X | |
| **AT015** | X |  |  | X | X | X |
| **AT016** | X |  |  |  | X | X |
| **AT017** | X |  |  |  | X | X |
| **AT018** | X |  |  |  | X | X |
| **AT019** | X |  | X |  | X | X |
| **AT020** | X | X |  |  | 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** | The logging system will be implemented using Apache Kafka technology | Kafka™ is used for building real-time data pipelines and streaming apps. For this reason is a perfect solution to publish log data that will be updated in real time to the subscribers |

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** | The teams must agree in the format of the log developed in Apache  Kafka | Since the Participation System and the dashboard share the log, the structure must be shared by both teams |

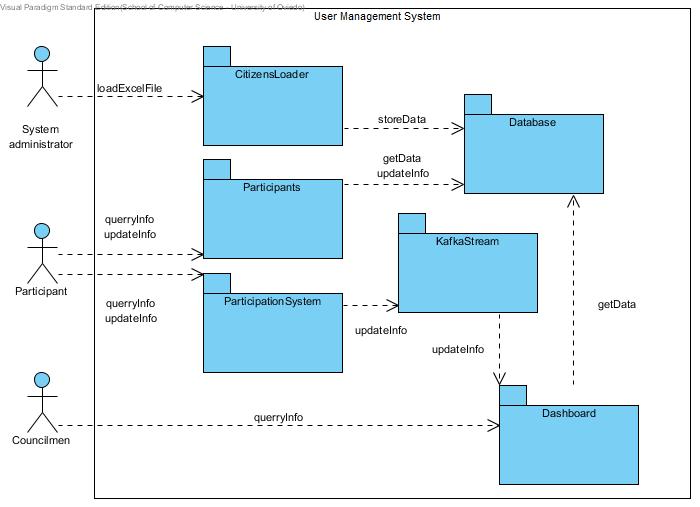
Table 5. Organizational constraints

# System scope and context

The system is decomposed in two modules:

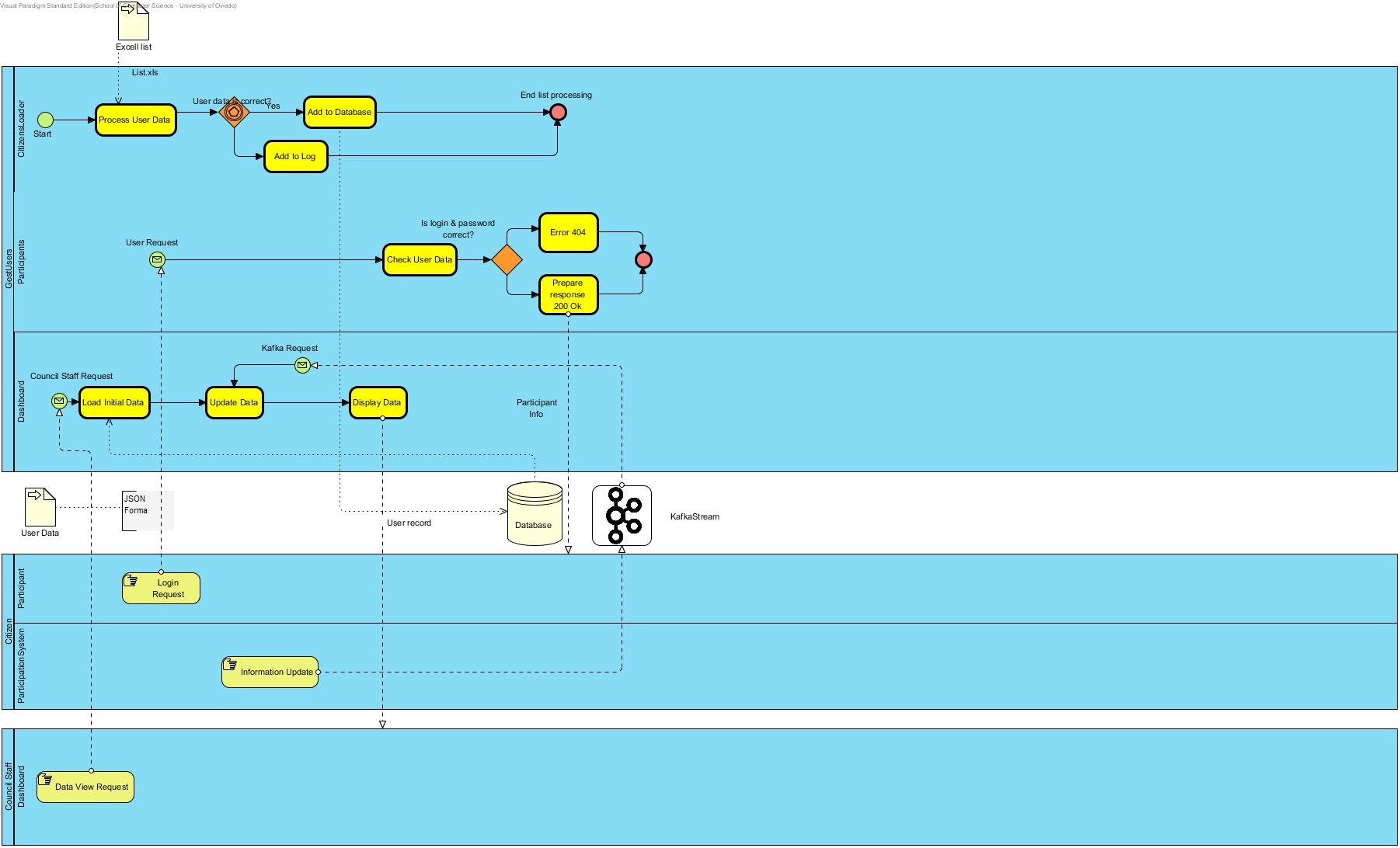
* CitizensLoader: This module will be responsible to convert data from Excel files and load it into the database. The system will be invoked by a system administrator.
* Participants: This module will check if users can participate obtaining information from the database.
* Dashboard: This module will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

Figure 1. Business Context



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

Figure 2. 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 | 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 | The system is deployed in a production environment | The system can be deployed by a system administrator in less than an hour. | **AT014** |
| 15 | Participation System | Change in the participation system | Runtime | Dashboard | The changes in the participation system will be shown in the dashboard | Changes will show in less than a minute in the dashboard | **AT015** |
| 16 | Participation System | Change in the participation system | Runtime | Dashboard | The changes are shown in the dashboard | There are no discrepancies between participation system data and dashboard data | **AT016** |
| 17 | Council | Watches the information in the dashboard | Runtime | Dashboard | The data is shown in graphics. | The graphics show the same information that is shown in the data. | **AT017** |
|  |  |  |  |  |  |  |  |
| 18 | System administrator | Makes a query about an user | Runtime | Participants | Querying information about an user should be fast | The response time is less than 10 seconds | **AT018** |
| 19 | Citizens | Uses the system | Runtime | ParticipationSystem | A very intuitive use of the system | In real time | **AT019** |
| 20 | Citizens | A lot of users in the system | Runtime | ParticipationSystem | Scalable system | Should be the same for one as for 500 | **AT020** |

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,  ST-06 | 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, AT018 | 1, 5, 6, 8, 9, 12, 13, 14,18 |
| Dashboard | ST-01, ST-05, ST-06 | AT014, AT015, AT016, AT017 | 15, 16, 17 |
| Participation System | ST-01, ST-03, ST-05, ST-06, ST-07 | AT015, AT019, AT020 | 15, 19, 20 |

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 five main sub-systems.

### Main overview

Figura 3. 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. |
| Dashboard | This is the module used by the council and other authorities to review data from the system. |
| ParticipationSystem | This module allows citizens to offer, comment and vote different proposals. |
| DataBase | This module encapsulates database access. |
| KafkaStream | This module encapsulated the use of Apache Kafka |

#### 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 *Participation System* module allows users to access the application and use the full proposal system. This includes the ability to make a proposal, as well as vote and comment on other people’s proposals. The user must log in the application using the *LogIn* interface and can then go on to comment and vote on proposals via the *UpdateProposalVotes* interface.

Optionally, it can be possible to order comments by popularity or chronological order or share a proposal via different social media.

The *Dasboard* module allows visualization of data from the system throw *ViewInfo*. The first load of data it’s done through *GetInfo* from the *database* module. The continuous data update will be done by the Kafka cluster.

#### InterfacesPortsPorts

##### 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 |

##### Dashboard

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetInfo | Interface | Method invocation | Retrieves all the relevant data from the database |
| UpdateInfo | Interface | Method invocation | Updates the current data when receiving a message from the KafkaCLuster |
| ViewInfo | Interface | Web Service | This interface will be invoked through an HTTP request |

##### ParticipationSystem

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| LogIn | Interface | Web service | Allows a user access to the application through a combination of email/password |
| CreateProposal | Interface | Method invocation | Creates a new proposal and sends it to the database and the kafka stream |
| VoteProposal | Interface | Method invocation | Updates the proposal votes and sends a message to the kafka stream |
| CommentProposal | Interface | Method invocation | Creates a new comment on a proposal and sends it to the database as well as the kafka stream |
| VoteComment | Interface | Method invocation | Updates the comment votes and sends a message to the kafka stream |
| ViewInfo | Interface | WebService | The different proposals and comments will be shown when 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 |
| UpdatePasswd | Interface | Method invocation | Updates the password of a user in the database |
| CreateProposal | Interface | Method invocation | Inserts into the database data about a new proposal |
| UpdateProposal | Interface | Method invocation | Updates the number of votes a proposal has |
| CreateComment | Interface | Method invocation | Inserts into the database data about a new comment |
| UpdateComment | Interface | Method invocation | Updates the number of votes a comment has |

##### KafkaStream

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| UpdateInfo | Interface | Method invocation | Sends a log message to the KafkaCluster |

#### Behaviour

##### CitizenLoader

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.

##### ParticipationSystem

It allows citizens to comment on proposals, as well as offer their own proposals and upvote or downvote other’s proposals.

##### Dashboard

The dashboard will offer council staff views of the evolution of the proposals, the number of votes of each of them as well as the different comments. It will be updated in real time without user interaction.

##### 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.

##### KafkaStream

The Kafka Stream will be configured to receive events from *ParticipationSystem*, and send messages to Dashboard to update the data.

## Citizens List

### Main overview



Figura 4. 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.

#### Interfaces PortsPorts

##### 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 systemreceive 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

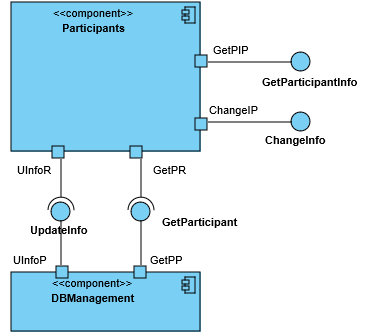


Figure 5. Participants View

### 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 Ports

##### 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 |
| 14 | AT014 | The use of Spring Boot framework facilitates deployment. There are several examples that show how to deploy Spring Boot based applications to production servers |

## Participation System

### Main Overview

Figure 6. Participation System View 1

### Catalogue of elements

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Apache Kafka | Provides the Produce interface, which is part of the Producer API given by Apache. This allows the ParticipationSystem to publish the streams of data needed for the Dashboard module. |
| ParticipationSystem | Offers two webservices: *LogIn,* which allows the user to log in; and *ViewInfo,* which shows the user different information about the proposal and the comments and categories. It also offers four interfaces: *CreateProposal,* which allows the user to create a new proposal; *VoteProposal,* which allows to vote a proposal; *CommentProposal,* which allows the user to create a new comment on a proposal; *VoteComment,* which allows to vote a comment. |
| DBManagement | It offers five interfaces: *GetParticipant*, to handle the information request for the user; *CreateProposal,* which creates a new proposal in the system; *VoteProposal,* which updates the votes of a proposal; *CommentProposal,* which creates a new comment in the system; and *VoteComment,* which updates the information about a comment. |

#### Relationships

The Participation System invokes *GetUser* (sending *email/password*). If the email/password are correct, the user is returned. The Participation System then retrieves all the proposals in the system and offers the user to do one of the 4 actions: *CreateProposal, VoteProposal, CommentProposal*, and *VoteComment.* Each of these invokations will do the appropriate update in the database as well as publish a new message in the kafka stream, which the corresponding topic needed.

#### Interfaces/Ports

##### ApacheKafka

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| Produce | Interface | Method invocation | Allows to publish the logs of data in the streams published in Kafka’s core |

##### ParticipationSystem

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| LogIn | Interface | WebService | Allows a user access to the application through a combination of email/password |
| CreateProposal | Interface | Method invocation | Creates a new proposal and sends it to the database and the kafka stream |
| VoteProposal | Interface | Method invocation | Updates the proposal votes and sends a message to the kafka stream |
| CommentProposal | Interface | Method invocation | Creates a new comment on a proposal and sends it to the database as well as the kafka stream |
| VoteComment | Interface | Method invocation | Updates the comment votes and sends a message to the kafka stream |
| ViewInfo | Interface | WebService | The different proposals and comments will be shown when invoked through an HTTP request |

##### DBManagement

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetParticipant | Interface | Method invocation | Returns data from citizens |
| CreateProposal | Interface | Method invocation | Inserts into the database data about a new proposal |
| UpdateProposal | Interface | Method invocation | Updates the number of votes a proposal has |
| CreateComment | Interface | Method invocation | Inserts into the database data about a new comment |
| UpdateComment | Interface | Method invocation | Updates the number of votes a comment has |

#### Behaviour

##### Apache Kafka

This component receives data from the Participation System, and publishes it in the stream for all its subscribers.

##### ParticipationSystem

This component allows the user to create and vote proposals as well as their comments. The user will log in with a combination of user/password and all the proposals in the system will be shown. The possibility to create a new proposal and new comments will be given by a form, in which the user types in the data.

**(Optional)** The proposals will be able to be sorted by date and number of votes.

**(Optional)** The comments will be able to be sorted by date and number of votes.

##### 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 | AT015 | 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. |
|  |  |  |
| 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 |
| 14 | AT014 | The use of Spring Boot framework facilitates deployment. There are several examples that show how to deploy Spring Boot based applications to production servers |
| 15 | AT015 | The system should be scalable and support a large number of users. |

## Dashboard

### Main overview

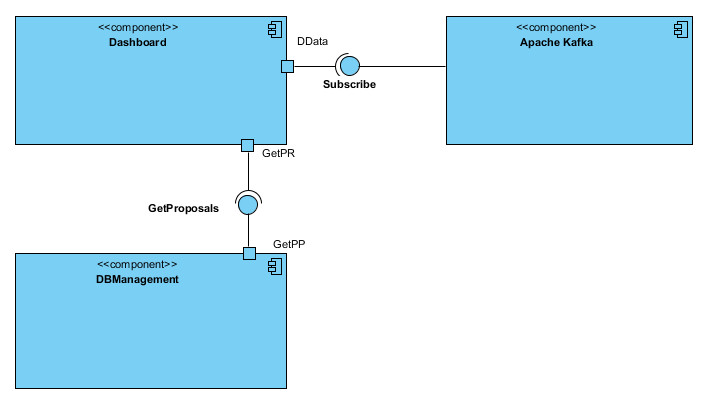


Figure 5. Dashboard View

### Catalogue of elements

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Apache Kafka | Provides the Subscribe interface, which is part of the Subscriber API given by Apache. This allows the Dashboard to receive the streams of data coming from the Participation System module. |
| Dashboard | Displays the information obtained from the Subscribe and GetProposals interfaces. |
| DBManagement | It offers one interface: GetProposals, that returns the proposal’s data from the database. |

#### Relationships

The *Dashboard* system will obtain all the proposals from the *GetProposals* method invokation. Also, the *Dashboard* component invokes the *Subscribe* method to obtain the streams of data related to the topics. After that it deserializes such data and displays it in the screen.

#### Interfaces Ports

##### Apache Kafka

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| Subscribe | Interface | Method invocation | Allows to get the logs of data from the streams published in Kafka’s core |

##### Dashboard

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| Subscribe | Interface | Method invocation | Gets the stream of data related to the topics subscribed |
| DData | Port |  | Deserializzzes the information obtained through the subscribe interface so it can be read |
| GetProposals | Interface | Method invocation | Allows to obtain all the proposal’s information |
| GetPR | Port |  |  |

##### DBManagement

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| GetProposals | Interface | Method invocation | Handles the information request from the dashboa**r**oard. |
| GetPP | Port |  |  |

#### Behaviour

##### Apache Kafka

This component receives a stream of data from the Participation System, the publisher of the data. After that Apache Kafka provides to all the subscribers, in this case the Dashboard component, a serialized data stream containing all the records from the topics that they are subscribed to.

##### Dashboard

This component gets the stream of data from Apache Kafka, the dashboard de-serializes it and then shows the data on the screen to the user in a comprehensible way.

**(Optional)** It shows the data in a graphical way.

**(Optional)** It sends alarms to some users when some specific events happen.

##### 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 |
| --- | --- | --- |
| 15 | AT015 | Using Apache Kafka lets you stream big loads of data in real time. |
| 16 | AT016 | The data shown in the dashboard mustn’t have discrepancies. |
| 17 | AT017 | Showing the data in graphs makes it easier and more comfortable for the users. |

# 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.