

School of Computer Science Engineering

2018-02-18

GestUsers: User Management System

Aquilino A. Juan Fuente

José Emilio Labra Gayo

Begoña Cristina Pelayo García-Bustelo

Jordán Pascual Espada

Vicente García Díaz

**INCIDENT MANAGEMENT SYSTEM**

*Software Architecture for GestUsers. Description of the practice work (2018)*

Description of the first practice work to be made by the work teams of the course “Software Architecture” during the academic year 2017-18.

**Grado de Ingeniería Informática del Software**

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

**GestUsers: Incident Management System**

**Authors**:

**Alejandro González Hevia - UO251513**

**Carmen Luisa Sors González - UO252449**

**Eduardo Ulibarri Toledo - UO251436**

**Jaime Fernández Martínez - UO251119**

**Pablo González Balbuena - UO239394**

**Sergio Faya Fernández - UO251005**

**Álvaro Tejido Jardón - UO250821**

**Date**: 05/02/2018

**Version**: ￼

Table of Contents

[0](#_Toc506723794)

[1 Introduction and Goals 5](#_Toc506723795)

[2 Requirements 6](#_Toc506723796)

[2.1 Loader 6](#_Toc506723797)

[2.2 Agents 6](#_Toc506723798)

[3 Methodology 8](#_Toc506723799)

[4 Stakeholders 9](#_Toc506723800)

[4.1 Students that develop the assignment 9](#_Toc506723801)

[4.2 System administrator 9](#_Toc506723802)

[4.3 Agents 9](#_Toc506723803)

[4.4 Developers of the Incident System 10](#_Toc506723804)

[4.5 Course teachers 10](#_Toc506723805)

[5 Quality Attributes 11](#_Toc506723806)

[5.1 List of Quality Attributes 11](#_Toc506723807)

[5.2 Quality Attributes and stakeholders 12](#_Toc506723808)

[6 Architecture Constraints 13](#_Toc506723809)

[6.1 Technical constraints 13](#_Toc506723810)

[6.2 Organizational Constraints 13](#_Toc506723811)

[7 System scope and context 14](#_Toc506723812)

[8 Quality Scenarios 16](#_Toc506723813)

[9 Views 19](#_Toc506723814)

[9.1 Context 19](#_Toc506723815)

[9.1.1 Main overview 19](#_Toc506723816)

[9.1.2 Elements Catalogue 19](#_Toc506723817)

[9.1.3 Rationale 21](#_Toc506723818)

[9.2 Loader 22](#_Toc506723819)

[9.2.1 Main overview 22](#_Toc506723820)

[9.2.2 Catalogue of Elements 22](#_Toc506723821)

[9.2.3 Context Diagram 24](#_Toc506723822)

[9.2.4 Rationale 24](#_Toc506723823)

[9.3 Agents 25](#_Toc506723824)

[9.3.1 Main overview 25](#_Toc506723825)

[9.3.2 Catalogue of elements 25](#_Toc506723826)

[9.3.3 Context Diagram 27](#_Toc506723827)

[9.3.4 Rationale 27](#_Toc506723828)

[9.4 Package View 28](#_Toc506723829)

[9.4.1 Main overview 28](#_Toc506723830)

[9.4.2 Catalogue of elements 28](#_Toc506723831)

[9.4.3 Context Diagram 28](#_Toc506723832)

[9.4.4 Rationale 28](#_Toc506723833)

[9.5 Deployment View 29](#_Toc506723834)

[9.5.1 Main overview 29](#_Toc506723835)

[9.5.2 Catalogue of elements 29](#_Toc506723836)

[9.5.3 Context Diagram 29](#_Toc506723837)

[9.5.4 Rationale 29](#_Toc506723838)

[10 References 30](#_Toc506723839)

# Introduction and Goals

The goal of this document is to describe the structure of an architecture of User Management that will be reused. 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 deliverable of the laboratory assignment of the course "*Software Architecture*" which is taught by the authors. The course is part of the Degree in Software Engineering, School of Computer Science Engineering, University of Oviedo.

The system is divided in two parts: Loader, to load data from agents and Agents, to obtain information about the agents that can participate in the system. 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 Loader module sub-system and the other team will implement the Agents module.

In the next deliverables, the students will create the architecture and implement a prototype of the rest of the Incidence Analysis System of which the current system will be part of.

# Requirements

User Management will be divided in two parts:

* *Loader*: is in charge of loading data from agents that can submit incidences to the system.
* *Agents*: allows to query information about the agents that participate in the system.

## Loader

The System administrator must be able to introduce data from the agents list. That data can be obtained from different sources like people or automatic agents such as sensors or others.

The introduction of data will be made from Excel files that contain a list of rows. Each row (except the first with the headings) contains the following information:

* Name (first and last name in the case of a person)
* Location (geographical coordinates)
* Email (if an automatic agent, eg. sensors, may provide the email of its administrator)
* Identifier (an ID in the case of a person)
* Kind of agent

When importing the agents’ data, the system will create a user (whose login key will be its username) 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 also generate personal letters, but it is not part of it to send these letters to each user by email.

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

## Agents

Agents should be able to login into the system to check that they are in 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 along the agent kind: username name and password and returns the data available about the agent if the information is correct or reports an error if it isn't. All 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

# Methodology

This document employs the ADD (Attribute-Driven Design) methodology (Bass, Clements, & Kazman, 2003) and the SEI ￼￼(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 from Team 1 (Loader).
2. Students from Team 2 (Agents).
3. System administrator.
4. Agents.
5. People responsible of the incident system.
6. Course Teachers.

| Code | Stakeholder | Interests (Modules) |
| --- | --- | --- |
| ST-01 | Students from Team 1 | Loader |
| ST-02 | Students from Team 2 | Agents |
| ST-03 | System administrator | Load files |
| ST-04 | Agents | Check data |
| ST-05 | Developers of Incident System | Check data |
| ST-06 | Course Teachers | Both |

Table 1. List of stakeholders/interests

## Students that develop the assignment

This group is formed by the two teams 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 agents 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
* Be able to use different formats of input files (**Optional**)

## Agents

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 information in a safe way.
* Being able to query their status in the system
* Being able to update or change their information in the system, for example, their password (**Optional**)
* Be able to use the system in a user-friendly way with a simple HTML interface (**Optional**)

## Developers of the Incident System

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

* Have a simple way to detect if an agent can participate in the system as soon as possible
* Use of simple technologies that can interoperate with other systems

## 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

# Quality Attributes

We have identified the following quality attributes:

* **Availability**
  + The system must be able to process data 24x7.
* **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 agents’ data
* **Testability**
  + It must be testable that the agents’ data loading process is correct
  + It must be testable that the web service behaves as expected
* **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 agents’ 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

## 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 | Agents |
| **AT002** | Easily modify some parts of the application: Change the parser of input data | Modifiability | Loader |
| **AT003** | Easily modify some parts of the application: Add an error reporting feature | Modifiability | Loader |
| **AT004** | Easily modify some parts of the application: Add other output files to generate the letters | Modifiability | Loader |
| **AT005** | Easily modify some parts of the application: Enable password change by users | Modifiability | Agents |
| **AT006** | Easily modify some parts of the application: Enable different formats to be used by the web service | Modifiability | Agents |
| **AT007** | The performance of the data loading system is reasonable (not too slow, but not critical) | Performance | Loader |
| **AT008** | The system should warrant the confidentiality of the agents’ data | Security | Loader and Agents |
| **AT009** | It must be testable that the web service behaves as expected | Testability | Agents |
| **AT010** | It must be testable that the user loading process is correct | Testability | Loader |
| **AT011** | The data loading system must be easy to use by system administrator users which are familiar with Unix-like tools. | Usability | Loader |
| **AT012** | The querying web service must be used by automated processes that can query the status of the system. | Interoperability | Agents |
| **AT013** | The system must be simple and easy to develop | Simplicity | Loader and Agents |
| **AT014** | The system should be easily deployable | Deployability | Agents |

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 |
| **AT002** | X |  | X |  |  | X |
| **AT003** | X |  | X |  |  | X |
| **AT004** | X |  | X |  |  | X |
| **AT005** |  | X |  | X |  | X |
| **AT006** |  | 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 |
| **AT013** | X | X |  |  | X | X |
| **AT014** |  | 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** | 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 |
| **TC002** | 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. |
| **TC003** | 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 |
| **TC004** | The output data of the 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. |
|  |  |  |

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 very powerful project management tool for this kind of projects. |

Table 5. Organizational constraints

# System scope and context

The system is decomposed in two modules:

* Loader: 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.
* Agents: This module will allow to query the agents that participate in the system obtaining information from the database.

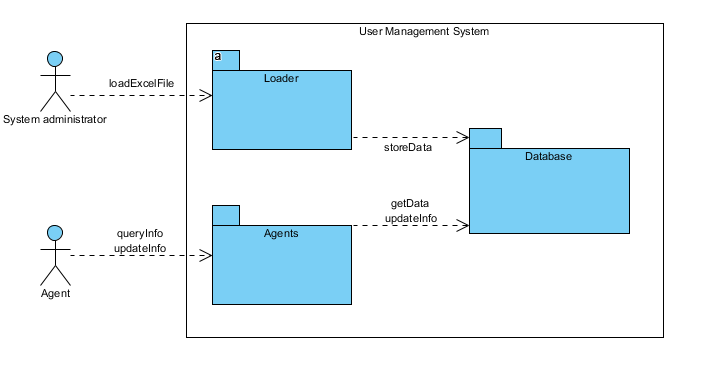


Figure 1. Business Context

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

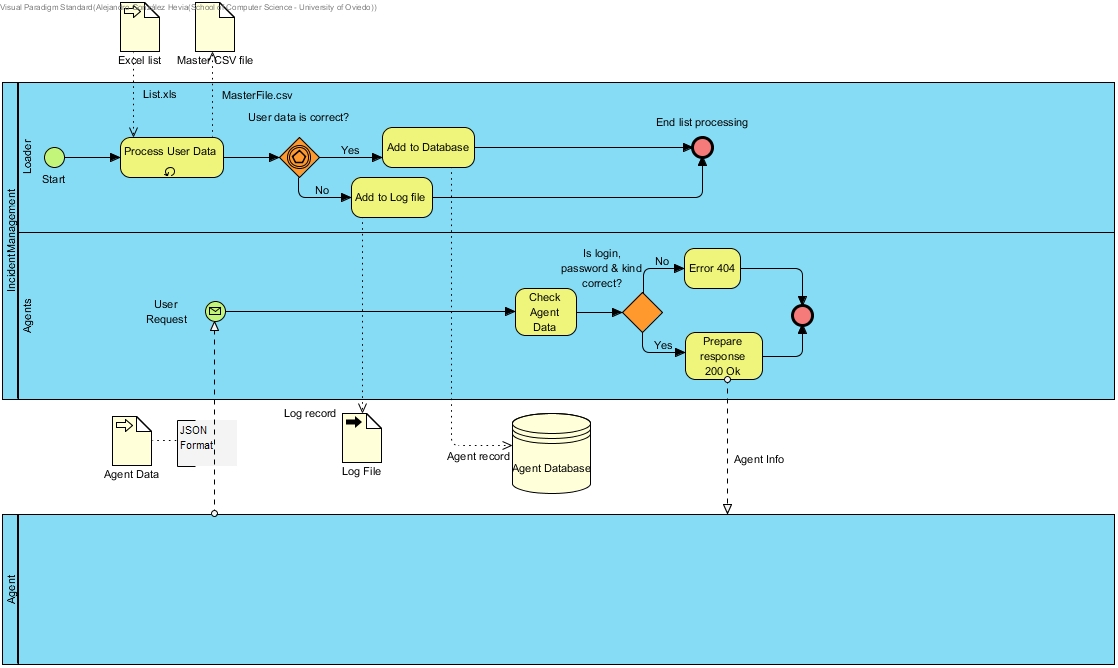


Figure 2. BPMN Diagram

# 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 | Agency System | Ask information about an agent | Runtime | Agents | Agency System obtains the required information in less than 15seg at any time in the day | The required information is obtained in less than 15seg | **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 | Agents 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 | Agents and DBManagement | The password of a user is successfully changed | Less than one day of work | **AT005** |
| 6 | Student developer | A new format is added to the web service | Development | Agents | 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 Agent | **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 | Agents | Get access to the application | Runtime | Agents | An agent can get access to his data but not to other one'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 loader 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 | Angents System | Access to the web service | Runtime | Agents | The agents System requests information about a user by passing a combination of user name and password and specifying the agent's kind. | A JSON 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 | Agents and Loader | The student developers can implement the system | The system can be implemented and tested in 2/3 weeks by third year undergraduate students. | **AT013** |
| 14 | System administrator | Deploys the system | Deployment | Agents and Loader | The system is deployed in a production environment | The system can be deployed by a system administrator in less than an hour. | **AT014** |
|  |  |  |  |  |  |  |  |

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 |
| Loader | ST-01, ST-03, ST-05, ST-06 | AT002, AT003, AT004, AT007, AT008 y AT010, AT011, AT013, AT014 | 2, 3, 4, 7, 8, 10, 11, 13, 14 |
| Agents | ST-02, ST-04, ST-05, ST-06 | AT001, AT005, AT006, AT008, AT009, AT012, AT013, AT014 | 1, 5, 6, 8, 9, 12, 13, 14 |

In the catalogues and views we have described both the mandatory and some optional elements.

## Context

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

### Main overview

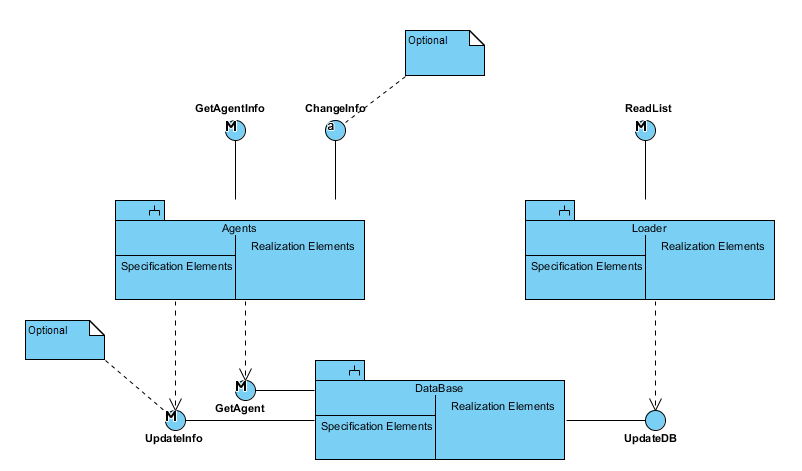


Figura 3. Context view

### Elements Catalogue

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Loader | It introduces agents’ data in the system. It reads an Excel file with data, generates passwords, personal letters and reports any errors. |
| Agents | This is the module used by agents 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. |

#### Relationships

Agents’ data are introduced in the system through the interface *ReadList* from module *Loader*. For each agent, a password is generated as well as a personalized letter with information about the agent.

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

The *Agents* module allows an external system to check the information about an agent through the web service *GetAgentInfo*. In order to check the information, *Agents* asks data to the *DataBase* module through the *GetAgent* 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 *Agents* module requests the *DataBase* to change the password through the *UpdatePasswd* interface.

#### InterfacesPortsPorts

##### Loader

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

##### Agents

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

##### DataBase

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetAgent | Interface | Method invocation | Returns data from agents |
| UpdateDB | Interface | Method invocation | Inserts into the database data about an agent included its password |
| UpdatePasswd | Interface | Method invocation | Updates the password of an agent in the database |

#### Behaviour

##### Loader

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.).

##### Agents

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 incidence system that invokes the Agents 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.

### Rationale

The main design decisions of this sub-system are:

|  |  |  |
| --- | --- | --- |
| Scenario | Quality attributes | Justification |
| 11 | AT011 | The *Loader* module is invoked from the main application as a console program, so its use will be familiar to people who use Unix-like tools. |
| 13 | AT013 | Both modules are simple enough to develop in 3 weeks given the number of students and their current knowledge of software engineering. |
| 14 | AT014 | For the *Loader* module, a batch application can be directly executed without any special needs for deployment. In case of the *Agents* module, the use of Spring Boot framework facilitates deployment. |

## Loader

### Main overview



Figura 4. Loader 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 agent 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 agents. It generates a new password for each agent 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/ Ports

##### Parser

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| ReadList | Interface | Method invocation | Read the Excel file with the agents’ 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 agents’ data in the system obtained from Excel files that contain a row for each agent. Each row (except the first one that contains the headings) contains the following columns:

* Name (string - in the case of a person, first and last name)
* Location (geographical coordinates)
* Email (string – in the case of sensors or other kinds of automatic agents, it can contain the email of the admin person who manages it)
* Identifier: The agent’s ID (string - in the case of a person or an entity it can be the NIF)
* Kind (integer representing the agent’s kind)

The system will also employ a master file in CSV format which will contain all the agent’s kinds available. The file contains two fields separated by commas, where the first field is the numeric code and the second one is the name of the user kind. These numeric codes will be considered to resolve the corresponding agent’s kind.

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 username/password enable a user to enter the system.

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

##### DBUpdate

It updates the database. See 9.1.2.4.3.

##### 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 |

## Agents

### Main overview

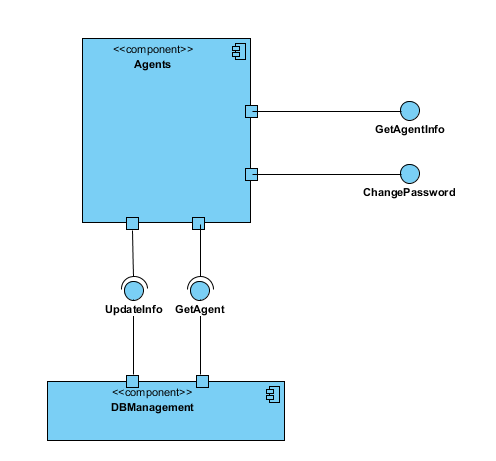


Figure 5. Agents View

### Catalogue of elements

#### Elements

|  |  |
| --- | --- |
| Element | Properties |
| Agents | It offers two web services: *GetAgentInfo*, which allows to obtain information about an agent (person, entity, sensor…) and (Optional) *ChangePassword* that allows to change the password of a user. |
| DBManagement | It offers two interfaces: GetAgent, that returns the data of an agent from the database and (Optional) *UpdateInfo*, to update a password change in the database. |

#### Relationships

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

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

#### Interfacesortss

##### Participants

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Type | Technology | Properties |
| GetAgentInfo | Interface | Web service | Allows to get access to an agent data through the user/password combination. |
| GetPIP | Port |  | Validates a user before asking the data. |
| ChangePassword | Interface | Web service | Allows to change a password using the combination: *user/password/newPasswod*. |
| ChangeInfo | Port |  | Validates a user before asking to change his password. |
| ChangeIP | Port |  | Validates a user before asking to change the password. |
| UpdateInfo | Interface (Required) | Method invocation | Asks a password change for a user. |
| UInfoR | Port |  |  |
| GetParticipant | Interface (Required) | 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 |  |  |
| GetAgent | Interface | Method invocation | Handles the information request for the user. |
| GetPP | Port |  |  |

#### Behaviour

##### Agents

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

<WebServiceURI>/agent

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

{"login": user, "password": password, "kind": agent’s kind}

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

{ "name": Name,

"location": Coordinates (optional),

"email": Email,

"id": identifier,

"kind": agent’s kind,

"kindCode": numeric code that represents the kind,

}

In case that the (username, 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 *username/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 |

## Package View

### Main overview

### Catalogue of elements

#### Elements

#### Relationships

#### Interfaces / Ports

#### Behaviour

### Context Diagram

See …

### Rationale

The main design decisions have been:

| Scenario | Quality Attributes | Justification |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Deployment View

### Main overview

### Catalogue of elements

#### Elements

#### Relationships

#### Interfaces / Ports

#### Behaviour

### Context Diagram

See …

### Rationale

The main design decisions have been:

| Scenario | Quality Attributes | Justification |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 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.