**GCI’16 Documentation**

Manfredi Vincenzo N86001560  
Pollastro Andrea N86001233  
Santangelo Andrea N86001703  
Scalella Andrea Francesco N86001625

Contents

[Software requirements document 4](#_Toc497218832)

[Functional model 4](#_Toc497218833)

[Use case diagram 4](#_Toc497218834)

[Cockburn tables 5](#_Toc497218835)

[Mockup 18](#_Toc497218836)

[Gantt diagram 47](#_Toc497218837)

[Domain model 49](#_Toc497218838)

[Analysis class diagram 49](#_Toc497218839)

[Analysis sequence diagrams 50](#_Toc497218840)

[System design document 59](#_Toc497218841)

[Architecture analysis 59](#_Toc497218842)

[Design pattern used & Implementation choices 60](#_Toc497218843)

[Testing document 64](#_Toc497218844)

[System testing 64](#_Toc497218845)

[Performs login test 64](#_Toc497218846)

[Researches contact test 65](#_Toc497218847)

[Alters contract test 66](#_Toc497218848)

[Deletes contract test 67](#_Toc497218849)

[Deletes contract test 68](#_Toc497218850)

[Adds contract test 69](#_Toc497218851)

[Reports errors in bills test 70](#_Toc497218852)

[Deletes injunctions test 71](#_Toc497218853)

[Confirms bill test 72](#_Toc497218854)

[Resends bill test 73](#_Toc497218855)

[Confirms injunctions test 74](#_Toc497218856)

[Resends injunctions test 75](#_Toc497218857)

[JUnit code 76](#_Toc497218858)

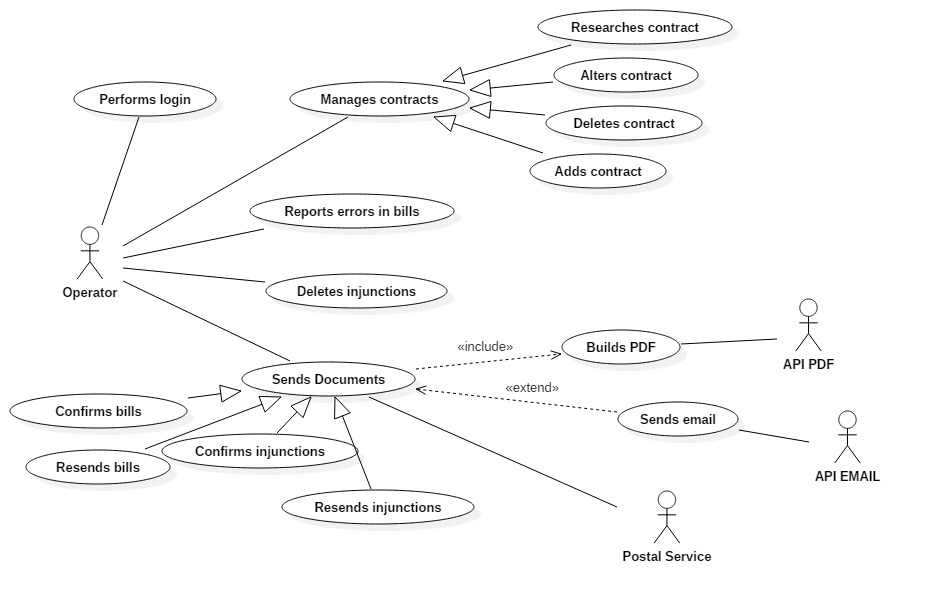
[Source 76](#_Toc497218859)

[Results 79](#_Toc497218860)

# **Software requirements document**

## Functional model

### Use case diagram



### Cockburn tables

#### Performs login

|  |  |
| --- | --- |
| **Use case** | Performs login |
| **Goal in context** | The operator wants to authenticate |
| **Preconditions** | The goal is to let the operator log into the system inserting the right credentials |
| **Success and condition** | The operator gets access to the system |
| **Failed end condition** | The operator can’t access to the system |
| **Operator** | Operator |
| **Trigger** | The operator starts the program |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator starts the program |  |
| 2 |  | The system shows the “Login” mockup |
| 3 | The operator fills all field in the “Login” mockup |  |
| 4 |  | The system enables the “Login” button in the “Login” mockup |
| 5 | The operator presses the “Login” button in the “Login” mockup |  |
| 6 |  | The system shows the “Home” mockup |

EXTENSION n.1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 6.1 |  | The system shows the “Login – error” mockup |
| 7.1 | The operator presses the “ok” button in the “Login – error” mockup |  |
| 8.1 |  | Go to step 3 |

#### Researches contract

|  |  |
| --- | --- |
| **Use case** | Researches contract |
| **Goal in context** | The operator can be able to read information about the con-  tract researched |
| **Preconditions** | The operator must be logged in and he has compiled almost one field in the “Registry management” mockup |
| **Success and condition** | The operator reads the informations about the contract researched |
| **Failed end condition** | There aren’t contracts stored into the system |
| **Operator** | Operator |
| **Trigger** | The operator clicks on “Search” button in the “Registry management” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on “Search” button in the “Registry management” mockup |  |
| 2 |  | The system shows the details of the contracts founded into the table |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 2.1 |  | The system shows the “Registry management – error” mockup |
| 3.1 | The operator clicks on the “Ok” button in the “Registry management – error” mockup |  |

SUBVARIATION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1.2 | The operator leaves all fields blank |  |
| 2.2 |  | The system shows details of all contracts stored into the system |

#### Alters contract

|  |  |
| --- | --- |
| **Use case** | Alter contract |
| **Goal in context** | Modify a pre-existing contract |
| **Preconditions** | The operator must be logged in, he has searched a contract and he has selected it from the table in the “Registry management” mockup |
| **Success and condition** | The operator has modified a pre-existing contract |
| **Failed end condition** | The operator fills the interested fields with invalid characters.  The operator cancels the operation and system shows the “Registry management” mockup |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Alter holder” button in the “Registry management” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on “Alter holder” button in the “Registry management” mockup |  |
| 2 |  | The system shows the “Alter holder” mockup |
| 3 | The operator edits one or more fields |  |
| 4 |  | The system enables the “Alter” button associated with the field edited |
| 5 | The operator clicks on the “Alter” button |  |
| 6 |  | The system shows the “Alter holder – success” mockup |
| 7 | The operator clicks on the “Ok” button in the “Alter holder – success” mockup |  |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 6.1 |  | The system shows the “Alter holder – error” mockup with a reference to the form that contains the error |
| 7.1 | The operator clicks on the “Ok” button in the “Alter holder – error” mockup |  |
| 8.1 |  | Go to step 2 |

#### Deletes contract

|  |  |
| --- | --- |
| **Use case** | Deletes contract |
| **Goal in context** | The operator wants to delete a contract |
| **Preconditions** | The operator must be logged in and he has selected a contract from the table contained in the mockup “Registry management” |
| **Success and condition** | The operator deletes the selected contract |
| **Failed end condition** | The operator can’t delete the selected contract  The operator cancels the operation |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Delete” button in the “Registry management” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on “Delete” button in the “Registry management” mockup |  |
| 2 |  | The system shows the “Delete contract” mockup |
| 3 | The operator clicks on the “Yes” button in the “Delete contract” mockup |  |
| 4 |  | The system shows the “Delete contract – success” mockup |
| 5 | The operator clicks on the “Ok” button in the “Delete contract – success” mockup |  |
| 6 |  | The system shows the “Registry management” mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.1 |  | The system shows the “Delete contract – error” mockup |
| 4.1 | The operator clicks on the “Ok” button in the “Delete contract – error” mockup |  |
| 5.1 |  | Go to step 6 |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.2 | The operator clicks on the “No” button in the “Delete contract” mockup |  |
| 4.2 |  | Go to step 6 |

#### Report errors in bills

|  |  |
| --- | --- |
| **Use case** | Report errors in bills |
| **Goal in context** | The operator wants to report errors in bills |
| **Preconditions** | The operator must be logged in and he has selected a bill in the bills’ table of the “Bills queue” mockup |
| **Success and condition** | The operator reports an error in a bill |
| **Failed end condition** | The operator clicks on the “Cancel” button |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Report error” button in the “Bills” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on “Report error” button in the “Bills” mockup |  |
| 2 |  | The system shows the “Report error” mockup |
| 3 | The operator fills the “Report’s specifications” field in the “Report error” mockup |  |
| 4 |  | The system enables the “Send” button in the “Report error” mockup |
| 5 | The operator clicks on the “Send” button in the “Report error” mockup |  |
| 6 |  | The system shows the “Send” mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.1 | The operator clicks on the “Cancel” button in the “Report error” mockup |  |
| 4.1 |  | Go to step 8 |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 5.2 | The operator clicks on the “Cancel” button in the “Report error” mockup |  |
| 6.2 |  | Go to step 8 |

#### Deletes injunction

|  |  |
| --- | --- |
| **Use case** | Deletes injunction |
| **Goal in context** | The operator wants to delete an injunction that has the “non-issued” state |
| **Preconditions** | The operator must be logged in and he has selected an injunction from the injunctions’ table in the “Injunctions queue” mockup |
| **Success and condition** | The operator deletes one or more injunctions from the injunctions’ table in the “Injunctions queue” mockup |
| **Failed end condition** | The operator doesn’t delete the injunctions from the table |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Delete” button in the “Injunctions queue” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on the “Delete” button in the “Injunctions queue” mockup |  |
| 2 |  | The system shows the “Delete injunction – success” mockup |
| 3 | The operator clicks on the “Ok” button |  |
| 4 |  | The system shows the “Injunctions queue” mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 2.1 |  | The system shows the “Delete injunction – error” mockup |
| 3.1 | The operator clicks on the “Ok” button in the “Delete injunction – error” mockup |  |

#### Confirms bills

|  |  |
| --- | --- |
| **Use case** | Confirms bills |
| **Goal in context** | The operator wants to confirm a single or a group of bills |
| **Preconditions** | The operator must be logged in and he has selected a bill (or more then one bill) in the bills’ table of the “Bills queue” mockup |
| **Success and condition** | The operator confirms a bill |
| **Failed end condition** | The operator cancels the operation |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Confirm” button in the “Bills queue” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on the “Confirm” button in the “Bills queue” mockup |  |
| 2 |  | The system shows the “Build PDF” mockup |
| 3 | The operator click on the “Send PDF” button in the “Build PDF” mockup |  |
| 4 |  | The system shows the “Send PDF” mockup |
| 5 | The operator clicks on the “Ok” button |  |
| 6 |  | The system shows the “Bills queue” mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.1 | The operator clicks on the “Cancel” button in the “Build PDF” mockup |  |
| 3.1 |  | Go to step 6 |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 4.2 |  | The system shows the “Send PDF” mockup in which the field “Log address’ error” is visible |
| 5.2 | Go to step 5 |  |

SUBVARIATION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1.3 | The operator clicks on the “Confirm” button in the “Bills queue” mockup after he has selected more then one row from the table or he has clicked on the “Select all” button |  |
| 2.3 |  | The system shows the “Build PDF – multiple” mockup |
| 3.3 | The operator clicks on the “Send PDFs” button in the “Build PDF – multiple” mockup |  |
| 4.3 |  | The system shows the “Send PDF – multiple” mockup |
| 5.3 | Go to step 5 |  |

#### Resends bills

|  |  |
| --- | --- |
| **Use case** | Resends bills |
| **Goal in context** | The operator wants to resend a bill that has the “Issued” state |
| **Preconditions** | The operator must be logged in and he has selected a bill in the bills’ table of the “Bills” mockup |
| **Success and condition** | The operator must be logged in and he has selected a bill in the bills’ table of the “Bills” mockup |
| **Failed end condition** | The operator cancels the operation  The PDF interface is not available |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Build PDF” button in the “Bills” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on the “Build PDF” button in the “Bills” mockup |  |
| 2 |  | The system shows the “Build PDF” mockup |
| 3 | The operator clicks on the “Send PDF” button in the “Build PDF” mockup |  |
| 4 |  | The system shows the “Send PDF” mockup |
| 5 | The operator clicks on the “Ok” button |  |
| 6 |  | The system shows the “Registry management” mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 4.1 |  | The system shows the “Send PDF” mockup in which the “Log address’ error” form is visible |
| 5.1 | Go to step 5 |  |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.2 | The operator clicks on the “Cancel” button in the “Build PDF” mockup |  |
| 4.2 |  | The system shows the “Bills” mockup |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 2.3 |  | The system shows the “Build PDF – error” mockup |
| 3.3 | The operator clicks on the “Ok” button in the “Build PDF – error” mockup |  |
| 4.3 |  | Go to step 6 |

#### Confirms injunctions

|  |  |
| --- | --- |
| **Use case** | Confirms injunctions |
| **Goal in context** | The operator wants to send an injunction to the customer |
| **Preconditions** | The operator must be logged in and he has selected an injunction from the table contained in “Injunctions queue” mockup |
| **Success and condition** | The system creates and sends the PDF |
| **Failed end condition** | The operator cancels the operation  The PDF interface is not available |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Confirm” button in the “Injunction queue” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on the “Confirm” button in the “Injunction queue” mockup |  |
| 2 |  | The system shows the “Build PDF” mockup |
| 3 | The operator clicks on the “Send PDF” butto in the “Build PDF” mockup |  |
| 4 |  | The system shows the “Send PDF” mockup |
| 5 | The operator clicks on the “Ok” button in the “Send PDF” mockup |  |
| 6 |  | The system shows the “Registry management”mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.1 | The operator clicks on the “Cancel” button in the “Build PDF” mockup |  |
| 4.1 |  | The system shows the “Injunctions queue” mockup |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 4.2 |  | The system shows the “Send PDF” mockup in which the “Log address’ error” form is visible |
| 5.2 | Go to step 5 |  |

EXTENSION #3

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 2.3 |  | The system shows the “Build PDF – error” mockup |
| 3.3 | The operator clicks on the “Ok” button in the “Build PDF – error” mockup |  |
| 4.3 |  | Go to step 6 |

#### Resends injunctions

|  |  |
| --- | --- |
| **Use case** | Resends injunctions |
| **Goal in context** | The operator wants to resend an injunction that has the “Issued” state |
| **Preconditions** | The operator must be logged in and he has selected an injunction in the injunctions’ table of the “Injunction” mockup |
| **Success and condition** | The operator resends an injunction that has the “Issued” state |
| **Failed end condition** | The operator cancels the operation |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Build PDF” button in the “Injunction” mockup |

MAIN SCENARIO

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on “Build PDF” button in the “Injunction” mockup |  |
| 2 |  | The system shows the “Build PDF” mockup |
| 3 | The operator clicks on the “Send PDF” button in the “Build PDF” mockup |  |
| 4 |  | The system shows the “Send PDF” mockup |
| 5 | The operator clicks on the “Ok” button in the “Send PDF” mockup |  |
| 6 |  | The system shows the “Registry management” mockup |

EXTENSION #1

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 3.1 | The operator clicks on the “Cancel” butto in the “Build PDF” mockup |  |
| 4.1 |  | The system shows the “Injunction” mockup |

EXTENSION #2

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 4.2 |  | The system shows the “Send PDF” mockup in which the “Log address’ error” form is visible |
| 5.2 | Go to step 5 |  |

EXTENSION #3

|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 2.3 |  | The system shows the “Build PDF - error” mockup |
| 3.3 | The operator clicks on the “Ok” butto in the “Build PDF – error” mockup |  |
| 4.3 |  | Go to step 6 |

#### Adds contract

|  |  |
| --- | --- |
| **Use case** | Adds contract |
| **Goal in context** | Create a new contract |
| **Preconditions** | The operator must be logged in and he clicked the “Registry management” button from the “Home” mockup |
| **Success and condition** | The operator adds a new contract |
| **Failed end condition** | The operator doesn’t fill all fields  The operator cancels the operation |
| **Operator** | Operator |
| **Trigger** | The operator clicks on the “Add” button in the “Registry management” mockup |

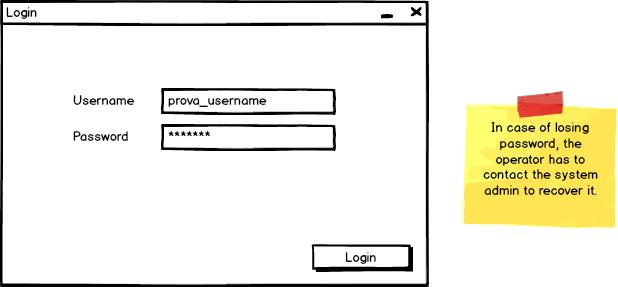
MAIN SCENARIO

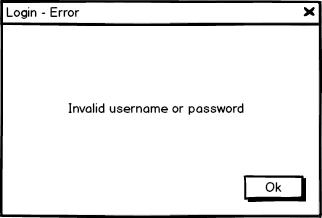
|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 1 | The operator clicks on “Add” button in the “Registry management” mockup |  |
| 2 |  | The system shows the “Add holder” mockup |
| 3 | The operator fills all fields and clicks the “Add” button in the “Add holder” mockup |  |
| 4 |  | The system shows the “Add holder – success” mockup |
| 5 | The operator clicks on the “Ok” button in the “Add holder – success” mockup |  |
| 6 |  | The system shows the “Add holder” mockup |

EXTENSION #1

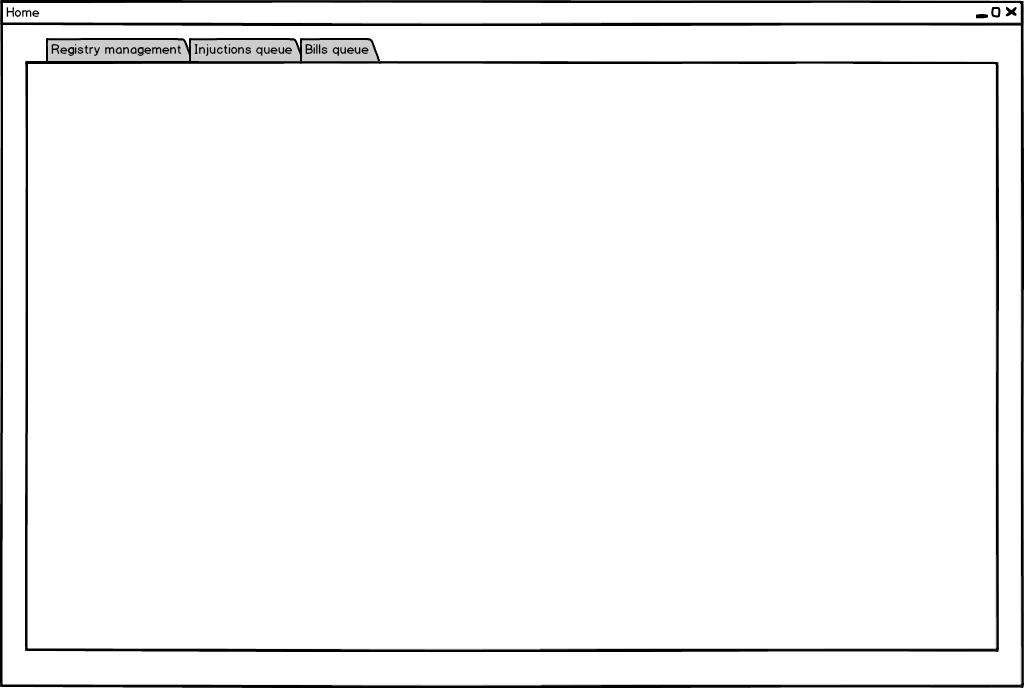
|  |  |  |
| --- | --- | --- |
| **Step n.** | **Operator** | **System** |
| 4.1 |  | The system shows the “Add holder – error” mockup with reference to the field that contains error |
| 5.1 | The operator clicks on the “Ok” button in the “Add holder – error” mockup |  |
| 6.1 |  | Go to step 2 |

### Mockup

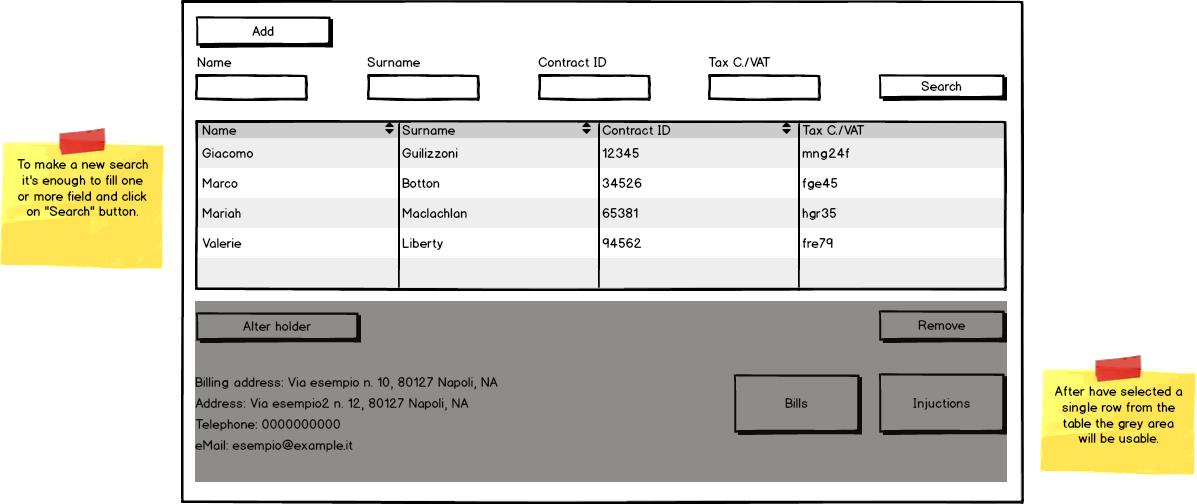


*Login*

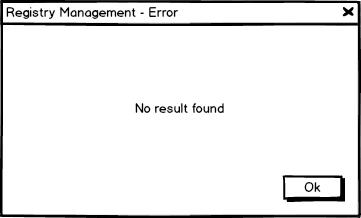
*Login - error*



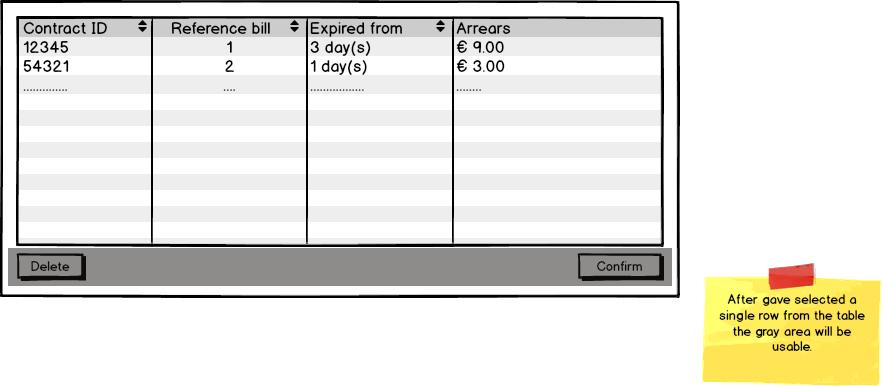
*Home*

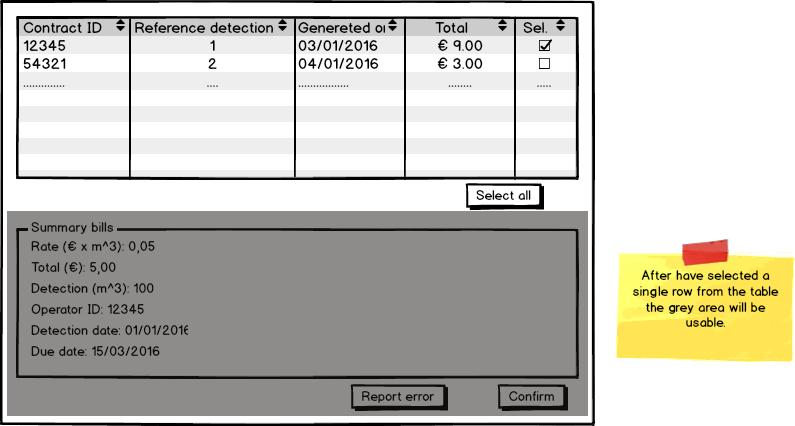


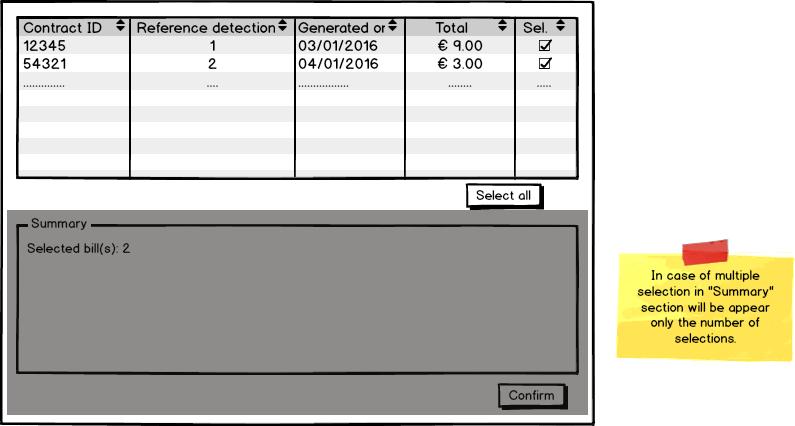
*Registry management*

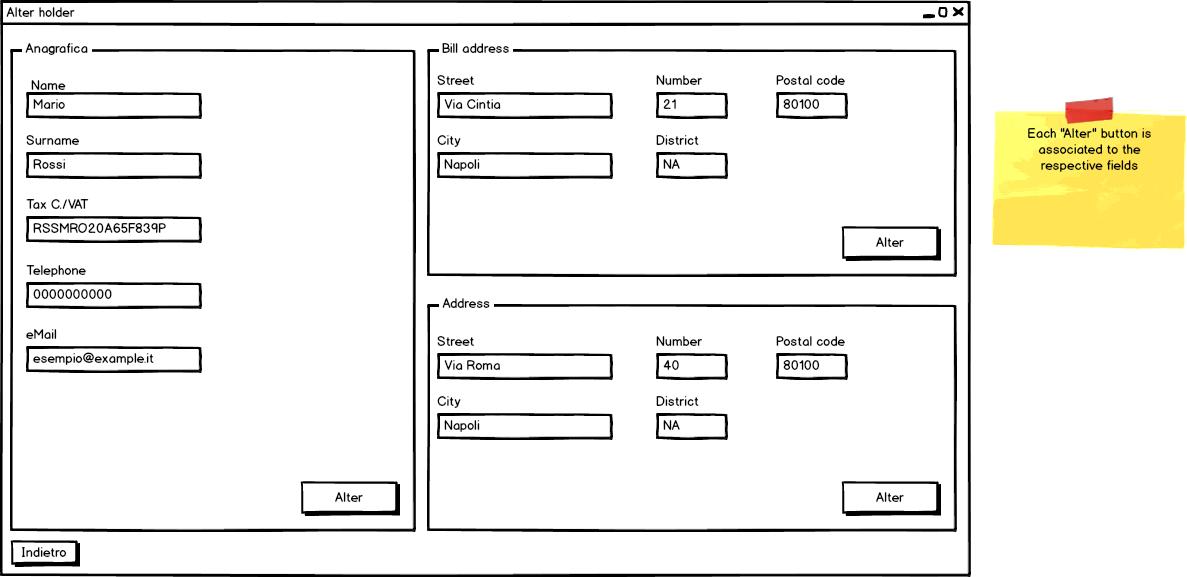


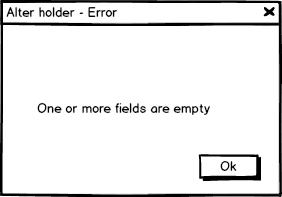
*Registry management – error*

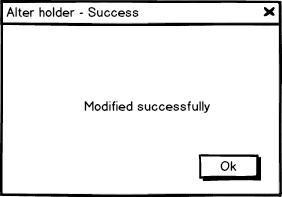


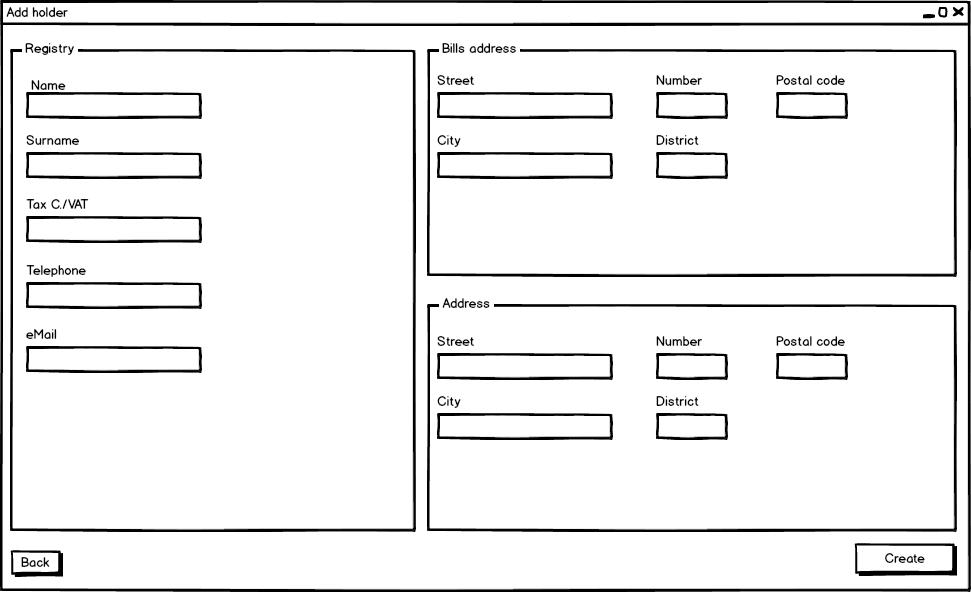
*Injunctions queue*

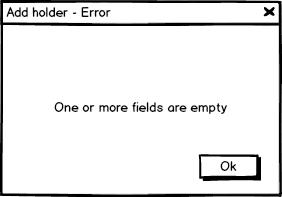
*Bills queue - Single*

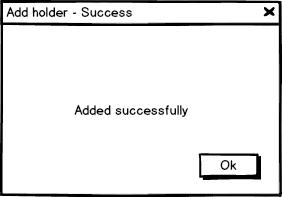
*Bills queue - Multiple*

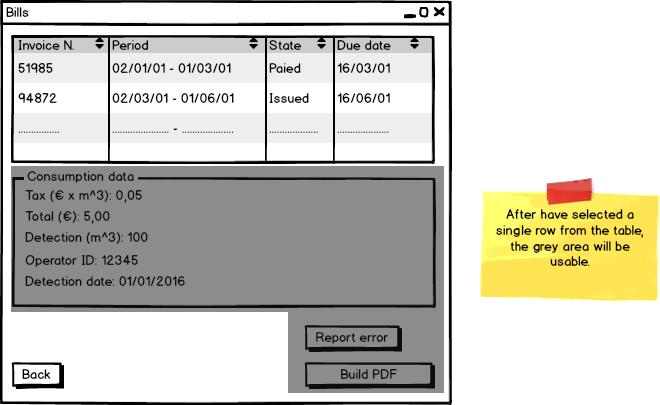
*Alter holder*

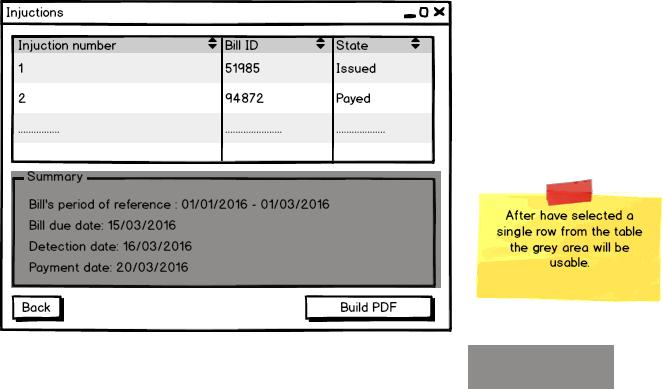
*Alter holder - error*

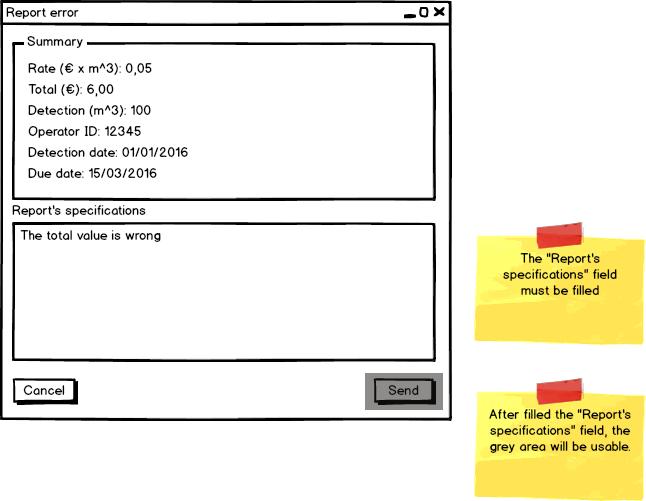
*Alter holder - success* 

*Add holder*

*Add holder - error*

*Add holder - success*

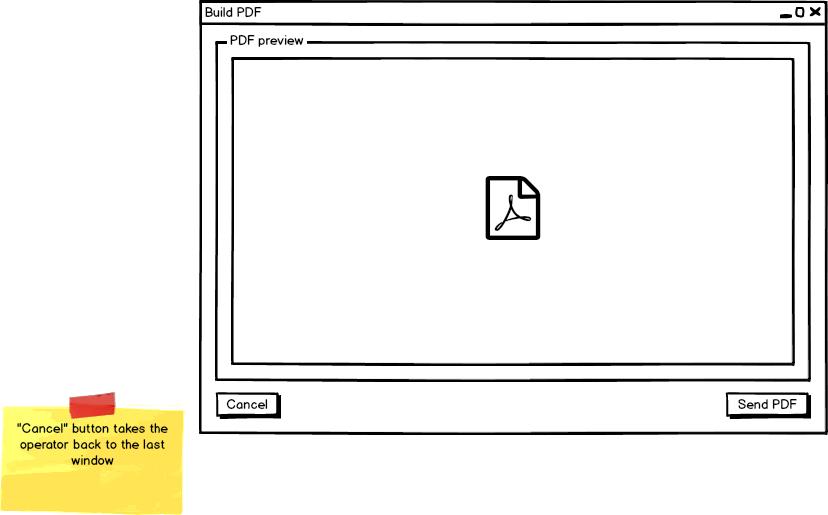
*Bills*

*Injunctions*

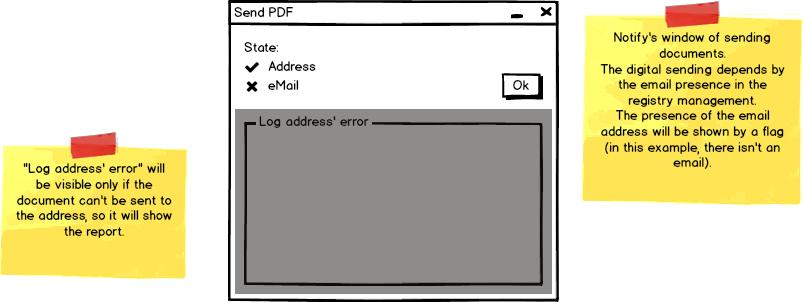
*Report error*



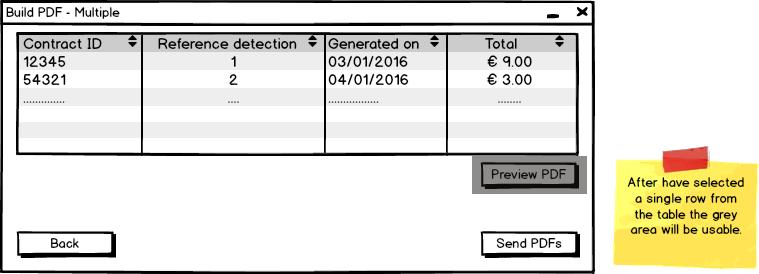
*Send*



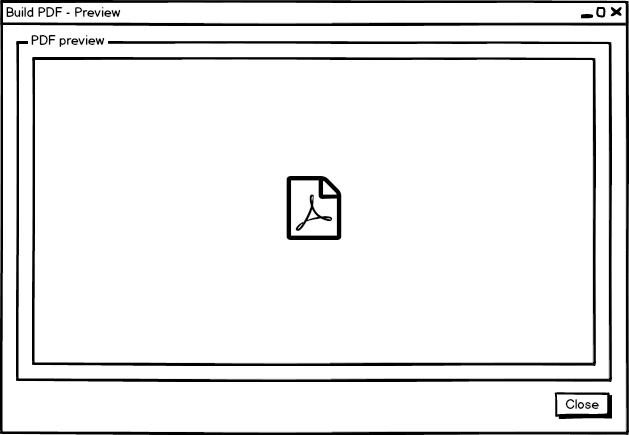
*Build PDF*



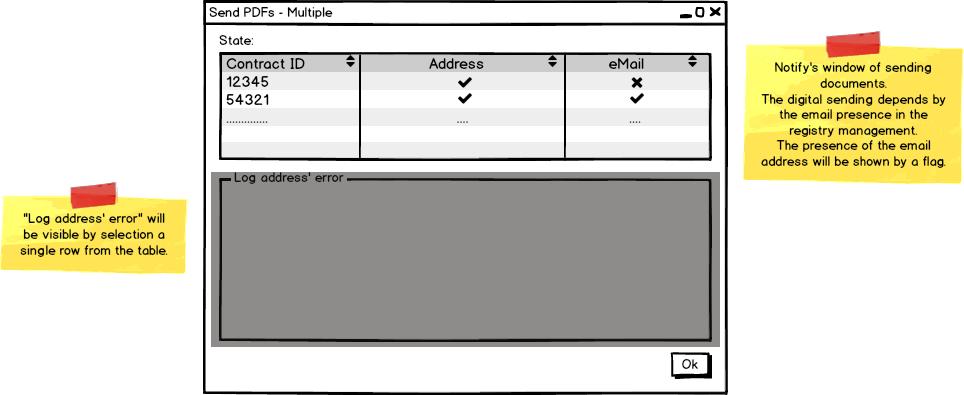
*Send PDF*



*Build PDF - multiple*



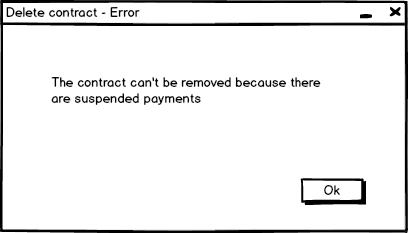
*Build PDF - preview*



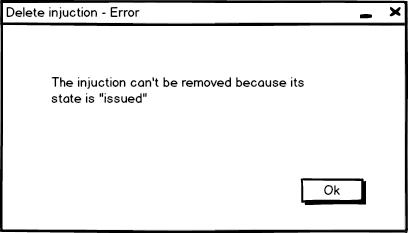
*Send PDFs - multiple*



*Delete contract*



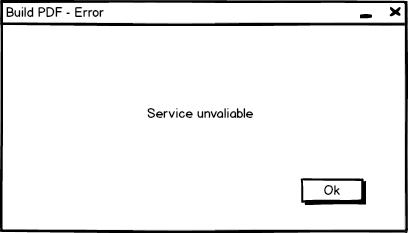
*Delete contract - error*



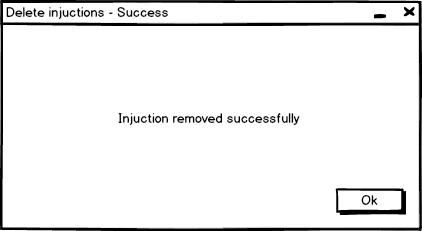
*Delete injunction - error*



*Delete contract - success*



*Build PDF - error*



*Delete injunctions – success*

### Gantt diagram



## Domain model

### Analysis class diagram

#### 

Analysis sequence diagrams

#### Adds contract

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\AddContract SequenceDiagram.png

#### Alters contract

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\AlterContract SequenceDiagram.png

#### Confirms bill

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ConfirmBill SequenceDiagram.png

#### Confirms injunctions

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ConfirmInjunctions SequenceDiagram.png

#### Deletes contract

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\DeleteContract SequenceDiagram.png

#### Deletes injunctions

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\DeleteInjunction SequenceDiagram.pngPerforms login

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Login SequenceDiagram.png

#### Reports errors in bills

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ReportErrorinBill SequenceDiagram.png

#### Researches contract

#### C:\Users\Andrea\AppData\Local\Microsoft\Windows\INetCache\Content.Word\SearchContract SequenceDiagram.png

# System design document

## Architecture analysis

The software architecture used in the application is the Model-View-Controller architecture. The system is partitioned into three subsystems:

* The Model subsystem, it maintains the application domain knowledge
* The View subsystem, it allows the user to view the object of the application domain
* The Control subsystem, it deals with the user interactions

A set of data in a relational database, of which the application uses, and the entity classes be part of the first subsystem. These lasts are the objects that represent the entities of problem.

The View subsystem is formed by the boundary classes. They form the user interfaces. The last subsystem is formed by the control classes. They manage the control logic, the interaction between entity and user interfaces. The application manages the asynchronous event by an event-based management. To boundary objects, that form the user interfaces, are associated listeners that catch an event when it occurs and execute actions depending on which event was triggered, on

which boundary object was triggered and what time of computation. The application is developed using centralized control: only one subsystem, the Control subsystem, is responsible for activating and stopping the others. The Main\_Controller class manages some boundary objects, along with their listeners, and all the classes that implements the Control interface. These lasts manage their boundary objects, but they leave the control of the main user interaction at Main\_Controller class. For this reason, the model used is the Call-Return model.

## Design pattern used & Implementation choices

The design pattern used are the Observer, Singleton and DAO pattern.

* The **Observer pattern** is used to define the Control - Component relationship. Each class that implements the Control interface has their Component objects; the way by which this classes dialogue is defined by the Observer Pattern. For increase the modularity has been included a class Listener. This last is used by Control classes (and by Main\_Controller class) for catching the event triggered by their Component objects. After we talk about the class Listener.
* The **Singleton pattern** is used to define the classes Main\_Controller and Database\_Controller. The application need just one instance of both these classes. Using the Singleton pattern we are sure that, in every moment during the computation, we have at most one instance of Main\_Controller and one instance of Database\_Contoller. The reason to have only one Main\_Controller is that this class manage the main control logic of the application: having more that one Main\_Controller would be a disaster. Instead, the reason to have only one Database\_Controller is that this class is the only one that dialogue with the database: it asks the database to do queries and it manages the results. We want the database receives a request at a time.
* The **DAO** (**Data Access Object**) **pattern** is used to provide some specific operation without exposing details of the database technology implemented. In this way the system is independent from the database technology.   
  Any DAO class is used to access to a specific object stored into the DBMS (all of these classes are described into the DAO Package).   
  In our case we used as technology the mySql DBMS.

During the design, some implementative choices were made. These choices are:

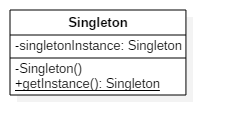
**1:** Class Listener: it has been introduced to increase the modularity.   
This class implements the ActionListener, MouseListener and ChangeListener interfaces, so it implements the methods described into the implemented interfaces.

These methods are invoked by the Component objects and it does something different depending on the object that invoked it.

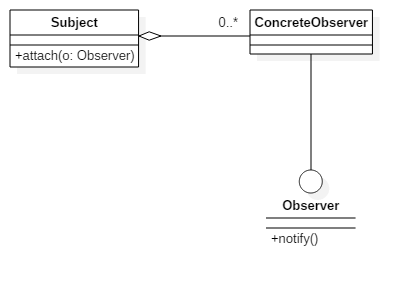
**2:** Controller interface: it has been introduced as tag interface.

**3:** Classes Registry\_Controller, BillsQueue\_Controller, InjunctionQueue\_Controller: MainController class (which is present into the Analysis’ class diagram) was splitted, during the design phase, into these 3 classes: Registry\_Controller, BillsQueue\_Controller, InjunctionQueue\_Controller. In this way we increased the modularity and we patitioned the MainController’s responsabilities: each one of these 3 classes manages one of the 3 panels present into the main view (Registry Management, Bills Queue, Injunctions Queue).

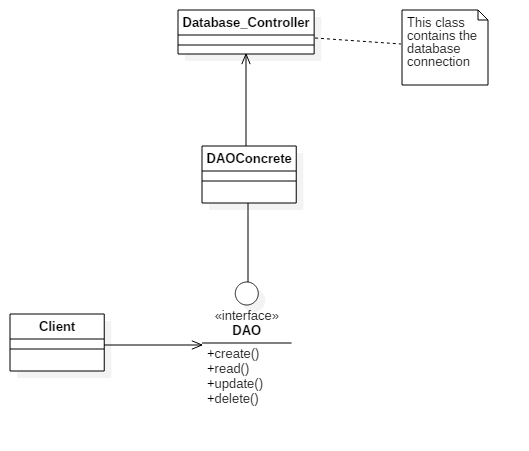
Singleton pattern:



Observer pattern



DAO pattern



# **Testing document**

## System testing

### Performs login test

|  |  |
| --- | --- |
| **Test ID** | 1 |
| **Test name** | Performs login |
| **Test goal** | The goal is to let the operator log into the system inserting  the right credentials |
| **Note** | In this case we consider these credentials:  "User1", "Psw1": right credentials (stored into the DBMS);  "User2", "Psw2": wrong credentials (non-stored into the DBMS) |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator fills the form with credentials: "User1", "Psw1" | System shows the "Registry Management" view | System shows the "Registry  Management" view |
| The operator fills the form with credentials: "User1", "Psw2" | System shows the error's popup with message "Invalid ID and/or Password!" | System shows the error's  popup with message "Invalid  ID and/or Password!" |
| The operator fills the form with credentials: "User2", "Psw1" | System shows the error's popup with message "Invalid ID and/or Password!" | System shows the error's  popup with message "Invalid  ID and/or Password!" |
| The operator fills the form with credentials: "User2", "Psw2" | System shows the error's popup with message "Invalid ID and/or Password!" | System shows the error's  popup with message "Invalid  ID and/or Password!" |

### Researches contact test

|  |  |
| --- | --- |
| **Test ID** | 2 |
| **Test name** | Researches contract |
| **Test goal** | The goal is to let the operator research contracts using forms |
| **Note** | In this case we consider these credentials:  "Name1", "Surname1", "ContractID1","TaxC/VAT1": right credentials  (stored into the DBMS);  "Name2", "Surname2", "ContractID2", "TaxC/VAT2": wrong credentials (non-stored into the DBMS); |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator fills the form with credentials: "Name1",  "Surname1", "ContractID1", "TaxC/VAT1" | System shows the results into the table | System shows the results into the table |
| The operator fills the  form with credentials:  "Name2" | System shows the error's popup with message "No result found" | System shows the error's popup with message "No result found" |
| The operator fills the  form with credentials:  "Surname2" | System shows the error's popup with message "No result found" | System shows the error's popup with message "No result found" |
| The operator fills the  form with credentials:  "ContractID2" | System shows the error's popup with message "No result found" | System shows the error's popup with message "No result found" |
| The operator fills the  form with credentials:  "TaxC/VAT2" | System shows the error's popup with message "No result found" | System shows the error's popup with message "No result found" |

### Alters contract test

|  |  |
| --- | --- |
| **Test ID** | 3 |
| **Test name** | Alters contract |
| **Test goal** | The goal is to let the operator alter a contract selected from the table |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator fills the form with new credentials | System shows a popup with message “Contract altered” displayed | System shows a popup with message “Contract altered” displayed |
| The operator fills the form with invalid characters | System shows a popup with message “Invalid characters” | System shows a popup with message “Invalid characters” |

### Deletes contract test

|  |  |
| --- | --- |
| **Test ID** | 4 |
| **Test name** | Deletes contract |
| **Test goal** | The goal is to let the operator delete a contract selected from the table |
| **Note** | We consider three cases: "Contract1": this contract can be  removed; "Contract2": this contract can't be removed because  it has some pendant bills; "Contract3": this contract can't be  removed because is already closed. |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator selected the "Contract1" | System shows a popup with message "Contract removed" | System shows a popup with message "Contract removed" |
| The operator selected the "Contract2" | System shows the error's popup with a message that explains the reason why it can't be closed | System shows the error's popup with a message that explains the reason why it can't be closed |
| The operator selected the "Contract3" | System shows the error's popup with a message that explains the reason why it can't be closed | System shows the error's popup with a message that explains the reason why it can't be closed |

### Deletes contract test

|  |  |
| --- | --- |
| **Test ID** | 4 |
| **Test name** | Deletes contract |
| **Test goal** | The goal is to let the operator delete a contract selected from the table |
| **Note** | We consider three cases: "Contract1": this contract can be  removed; "Contract2": this contract can't be removed because  it has some pendant bills; "Contract3": this contract can't be  removed because is already closed. |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator selected the "Contract1" | System shows a popup with message "Contract removed" | System shows a popup with message "Contract removed" |
| The operator selected the "Contract2" | System shows the error's popup with a message that explains the reason why it can't be closed | System shows the error's popup with a message that explains the reason why it can't be closed |
| The operator selected the "Contract3" | System shows the error's popup with a message that explains the reason why it can't be closed | System shows the error's popup with a message that explains the reason why it can't be closed |

### Adds contract test

|  |  |
| --- | --- |
| **Test ID** | 5 |
| **Test name** | Adds contract |
| **Test goal** | The goal is to let the operator add new contracts |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator fills all forms interested in the contract's adding | System shows a popup with message "New contract added" | System shows a popup with message "New contract added" |
| The operator fills a form with invalid characters | System shows the error's popup with a message that explains the reason why it can't continue the operation | System shows the error's popup with a message that explains the reason why it can't continue the operation |

### Reports errors in bills test

|  |  |
| --- | --- |
| **Test ID** | 6 |
| **Test name** | Reports errors in bills |
| **Test goal** | The goal is to let the operator reports errors in bills |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator fills the form with the errors | System shows a popup with message "Operation successfully completed" | System shows a popup with  message "Operation successfully completed" |

### Deletes injunctions test

|  |  |
| --- | --- |
| **Test ID** | 7 |
| **Test name** | Deletes injunctions |
| **Test goal** | The goal is to let the operator delete injunctions |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator clicked the “Remove” button | System shows a popup with message "Operation successfully completed" | System shows a popup with message "Operation successfully completed" |
| The operator clicked the “Remove” button but the injunction can’t be removed because it has pending payment | System shows a popup with an error message | System shows a popup with an error message |

### Confirms bill test

|  |  |
| --- | --- |
| **Test ID** | 8 |
| **Test name** | Confirms bill |
| **Test goal** | The goal is to let the operator confirm a selected bill |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator selects a bill from the bill's table and presses "Confirm button” | System shows the bill's preview (PDF) | System shows the bill's preview (PDF) |

### Resends bill test

|  |  |
| --- | --- |
| **Test ID** | 9 |
| **Test name** | Resends bill |
| **Test goal** | The goal is to let the operator resend a selected bill |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator selects a bill from the bills’ table and presses "Confirm button” | System shows the bill's preview (PDF) | System shows the bill's preview (PDF) |
| The operator selects a bill from the bills’ table that has been payed and presses “Confirm button” | System shows a popup that contains an error message | System shows a popup that contains an error message |

### Confirms injunctions test

|  |  |
| --- | --- |
| **Test ID** | 10 |
| **Test name** | Confirms injunctions |
| **Test goal** | The goal is to let the operator confirm an injunction |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator selects an injunction from the injunction’s table and presses "Confirm button” | System shows the bill's preview (PDF) | System shows the bill's preview (PDF) |

### Resends injunctions test

|  |  |
| --- | --- |
| **Test ID** | 11 |
| **Test name** | Resends injunctions |
| **Test goal** | The goal is to let the operator resends an injunction |
| **Note** |  |

|  |  |  |
| --- | --- | --- |
| **Input** | **Desired output** | **Obtained output** |
| The operator selects an injunction from the injunction’s table and presses "Confirm button” | System shows the bill's preview (PDF) | System shows the bill's preview (PDF) |
| The operator selects an injunction from the injunctions’ table that has been payed and presses “Confirm button” | System shows a popup that contains an error message | System shows a popup that contains an error message |
| The operator selects an injunction that refers to a payed bill | System shows a popup that contains an error message | System shows a popup that contains an error message |

## JUnit code

### Source

1. **package** Controller;
3. **import** java.sql.SQLException;
4. **import** java.util.logging.Level;
5. **import** java.util.logging.Logger;
6. **import** org.junit.Before;
7. **import** org.junit.BeforeClass;
8. **import** org.junit.Test;
9. **import** **static** org.junit.Assert.\*;
10. /\*\*
11. \*
12. \* @author Andrea
13. \*
14. \* This class tests the "check" method of "Login\_Controller" class.
15. \* This methods takes 2 parameters in input: USER(String) and PSW(String).
16. \*
17. \* Equivalence classes:
18. \* USER:
19. \*  - Numerical value between 0 and 100 (T)
20. \*  - Numerical value lower then 0  (F1)
21. \*  - Numerical value higher then 100   (F2)
22. \*  - NULL  (F3)
23. \*  - Alphanumerical value  (F4)
24. \*
25. \* PSW:
26. \*  - Stored values (T)
27. \*  - Not stored values (F1)
28. \*  - NULL  (F2)
29. \*
30. \* Method applied: SECT
31. \* Number of tests: 15 (5x3)
32. \*/
33. **public** **class** Login\_ControllerTest {
34. **private** **static** Login\_Controller instance;
35. **private** Boolean result;
36. **private** String user;
37. **private** String psw;
38. **private** Boolean expResult;
40. **public** Login\_ControllerTest() {
41. }
43. @BeforeClass
44. **public** **static** **void** setUpClass() {
45. **try** {
46. instance = **new** Login\_Controller(Main\_Controller.getMain());
47. } **catch** (SQLException ex) {
48. Logger.getLogger(Login\_ControllerTest.**class**.getName()).log(Level.SEVERE, **null**, ex);
49. fail("Login\_Controller instantiation failed");
50. }
51. }
53. @Before
54. **public** **void** setUp() {
55. user = **null**;
56. psw = **null**;
57. result = **null**;
58. expResult = **null**;
59. }
61. /\*
62. \*   EQUIVALENCE CLASSES TESTED:
63. \*   User:   T
64. \*   Psw:    T
65. \*/
66. @Test
67. **public** **void** checkTest1(){
68. user = "2";
69. psw = "ingsw";
70. expResult = **true**;
71. result = instance.check(user, psw);
72. assertEquals(result, expResult);
73. }
75. /\*
76. \*   EQUIVALENCE CLASSES TESTED:
77. \*   User:   T
78. \*   Psw:    F1
79. \*/
80. @Test
81. **public** **void** checkTest2(){
82. user = "2";
83. psw = "ingw";
84. expResult = **false**;
85. result = instance.check(user, psw);
86. assertEquals(result, expResult);
87. }
89. /\*
90. \*   EQUIVALENCE CLASSES TESTED:
91. \*   User:   T
92. \*   Psw:    F2
93. \*/
94. @Test
95. **public** **void** checkTest3(){
96. user = "2";
97. expResult = **false**;
98. result = instance.check(user, psw);
99. assertEquals(result, expResult);
100. }
102. /\*
103. \*   EQUIVALENCE CLASSES TESTED:
104. \*   User:   F2
105. \*   Psw:    T
106. \*/
107. @Test
108. **public** **void** checkTest4(){
109. user = "101";
110. psw = "ingsw";
111. expResult = **false**;
112. result = instance.check(user, psw);
113. assertEquals(result, expResult);
114. }
116. /\*
117. \*   EQUIVALENCE CLASSES TESTED:
118. \*   User:   F2
119. \*   Psw:    F1
120. \*/
121. @Test
122. **public** **void** checkTest5(){
123. user = "101";
124. psw = "ingw";
125. expResult = **false**;
126. result = instance.check(user, psw);
127. assertEquals(result, expResult);
128. }
130. /\*
131. \*   EQUIVALENCE CLASSES TESTED:
132. \*   User:   F2
133. \*   Psw:    F2
134. \*/
135. @Test
136. **public** **void** checkTest6(){
137. user = "101";
138. expResult = **false**;
139. result = instance.check(user, psw);
140. assertEquals(result, expResult);
141. }
143. /\*
144. \*   EQUIVALENCE CLASSES TESTED:
145. \*   User:   F3
146. \*   Psw:    T
147. \*/
148. @Test
149. **public** **void** checkTest7(){
150. user = "2abc";
151. psw = "ingsw";
152. expResult = **false**;
153. result = instance.check(user, psw);
154. assertEquals(result, expResult);
155. }
157. /\*
158. \*   EQUIVALENCE CLASSES TESTED:
159. \*   User:   F3
160. \*   Psw:    F1
161. \*/
162. @Test
163. **public** **void** checkTest8(){
164. user = "2abc";
165. psw = "ingw";
166. expResult = **false**;
167. result = instance.check(user, psw);
168. assertEquals(result, expResult);
169. }
171. /\*
172. \*   EQUIVALENCE CLASSES TESTED:
173. \*   User:   F3
174. \*   Psw:    F2
175. \*/
176. @Test
177. **public** **void** checkTest9(){
178. user = "2abc";
179. expResult = **false**;
180. result = instance.check(user, psw);
181. assertEquals(result, expResult);
182. }
184. /\*
185. \*   EQUIVALENCE CLASSES TESTED:
186. \*   User:   F4
187. \*   Psw:    T
188. \*/
189. @Test
190. **public** **void** checkTest10(){
191. psw = "ingsw";
192. expResult = **false**;
193. result = instance.check(user, psw);
194. assertEquals(result, expResult);
195. }
197. /\*
198. \*   EQUIVALENCE CLASSES TESTED:
199. \*   User:   F4
200. \*   Psw:    F1
201. \*/
202. @Test
203. **public** **void** checkTest11(){
204. psw = "ingw";
205. expResult = **false**;
206. result = instance.check(user, psw);
207. assertEquals(result, expResult);
208. }
210. /\*
211. \*   EQUIVALENCE CLASSES TESTED:
212. \*   User:   F4
213. \*   Psw:    F2
214. \*/
215. @Test
216. **public** **void** checkTest12(){
217. expResult = **false**;
218. result = instance.check(user, psw);
219. assertEquals(result, expResult);
220. }
222. /\*
223. \*   EQUIVALENCE CLASSES TESTED:
224. \*   User:   F1
225. \*   Psw:    T
226. \*/
227. @Test
228. **public** **void** checkTest13(){
229. user = "-1";
230. psw = "ingsw";
231. expResult = **false**;
232. result = instance.check(user, psw);
233. assertEquals(result, expResult);
234. }
236. /\*
237. \*   EQUIVALENCE CLASSES TESTED:
238. \*   User:   F1
239. \*   Psw:    F1
240. \*/
241. @Test
242. **public** **void** checkTest14(){
243. user = "-1";
244. psw = "ingw";
245. expResult = **false**;
246. result = instance.check(user, psw);
247. assertEquals(result, expResult);
248. }
250. /\*
251. \*   EQUIVALENCE CLASSES TESTED:
252. \*   User:   F1
253. \*   Psw:    F2
254. \*/
255. @Test
256. **public** **void** checkTest15(){
257. user = "-1";
258. expResult = **false**;
259. result = instance.check(user, psw);
260. assertEquals(result, expResult);
261. }
262. }

### Results

