# Requirements’ document

## Requirements List

### Functional requirements

The Back Office Operator must be able to create new payment orders, specifying a bill by a list of candidate payment orders.

The Back Office Operator can issue not-issued payment orders.  
When a payment order is issued, it is given a protocol number generated incrementally.

The Back Office Operator can delete not-issued payment orders.

The Back Office Operator can reissue suspended payment orders.  
When a payment order is reissued his protocol number is not changed.

The Back Office Operator can save as not pertinent suspended payment orders.

The Back Office Operator can save as paid notified payment orders.

The Back Office Operator can save as suspended notified payment orders.

When a payment order is issued the system must generate and store a PDF document with all the useful communications for the contributor.

A user must be able to log in as a Back Office Operator in desktop application.

A user must be able to log in as a Readings Operator in mobile application.

The Readings Operator must have the list of the assigned readings updated automatically.

The Readings Operator can save readings specifying a meter and the water consumed.

The Readings Operator must be able to send saved readings via Internet.

### Non-functional Requirements

Payment orders can be seen and searched specifying a protocol, a debtor, a year, a trimester and/or a status.

The readings are first saved locally on smartphone and sent later only if internet connection is available.

To use the system every user must log in.

### Domain Requirements

Not issued payment orders can only be issued or deleted.

Issued payment orders can only be archived.

Once saved as paid or saved as not pertinent, a payment order must be archivied.

Suspended payment orders can only be reissued or saved as not pertinent.

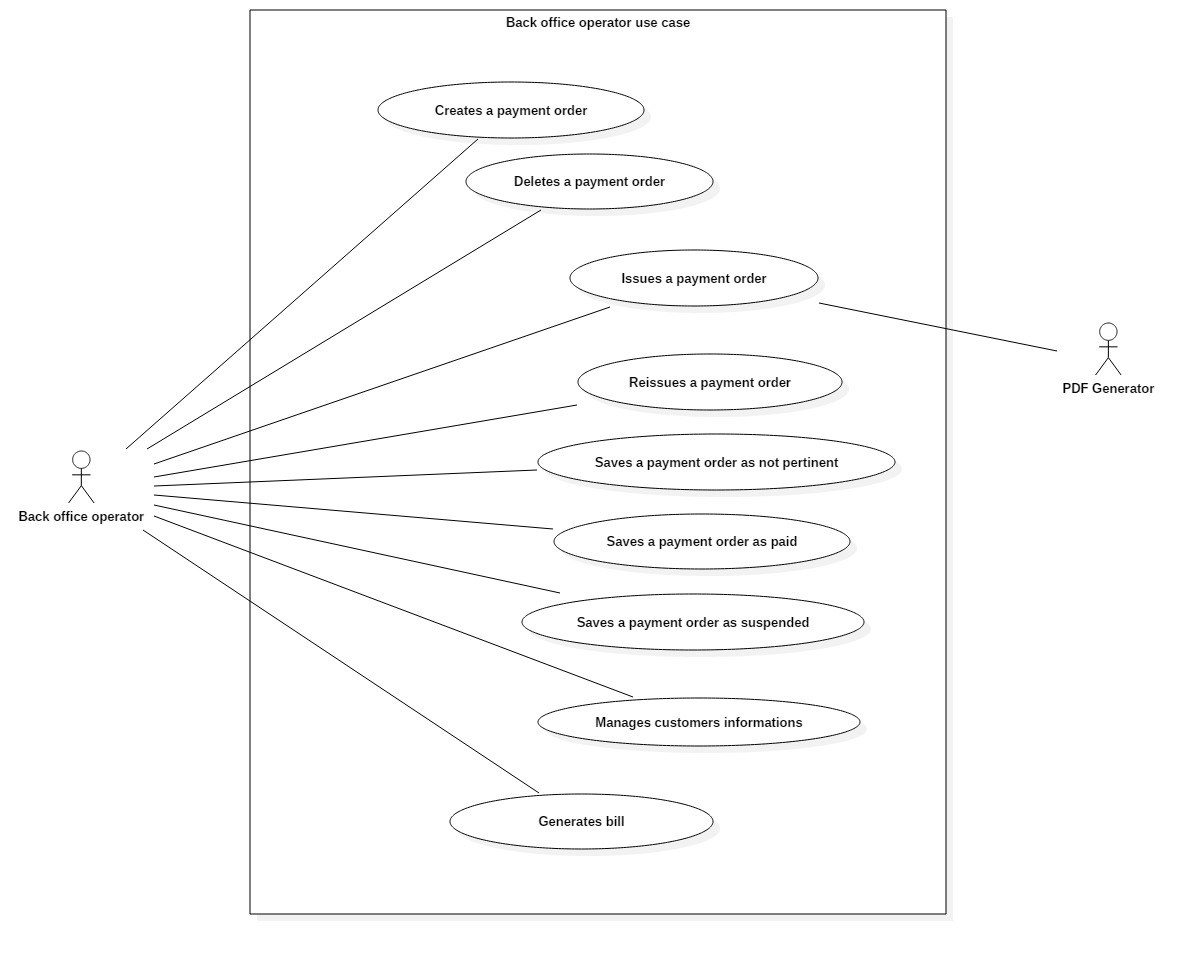
Every issued payment order is identified by a protocol number.

Notified payment orders can only be saved as paid or as suspended.

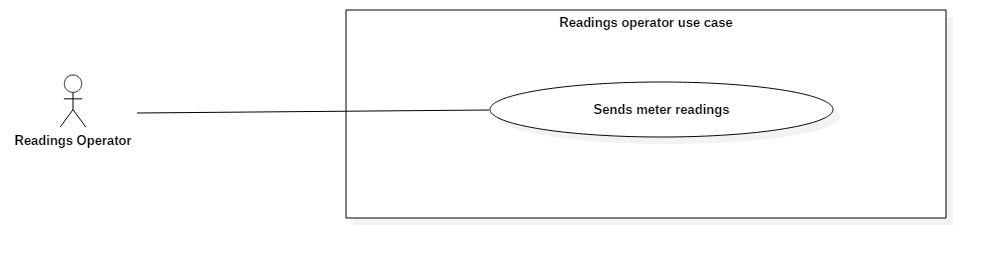
The considered unit of measurement of water is m3.

## Use Case Diagrams

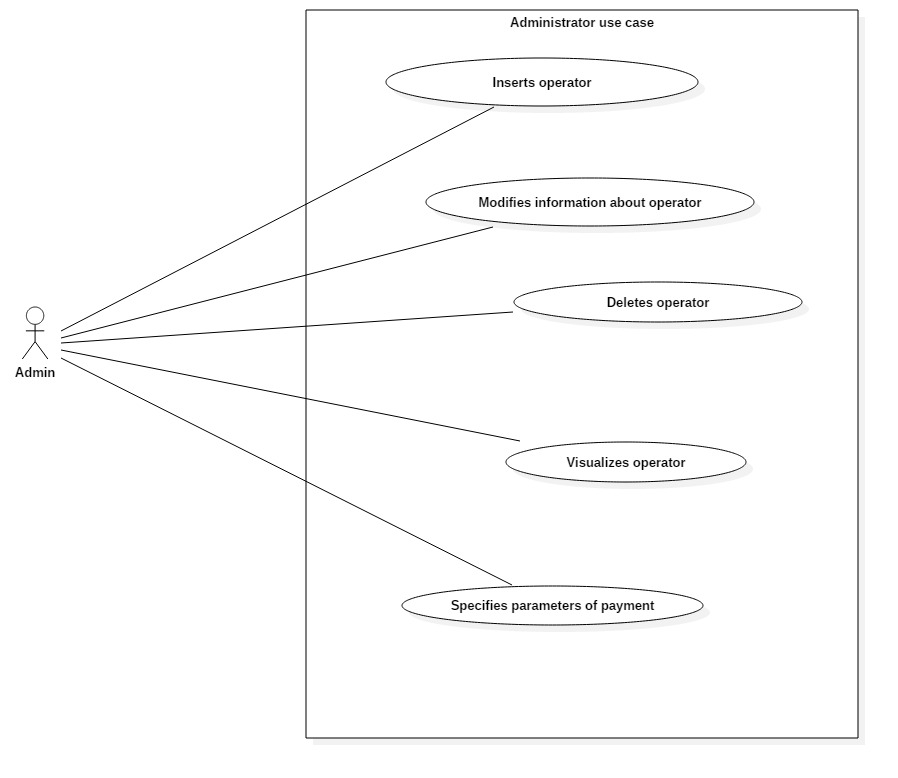
### Back office operator use case



### Readings operator use case



### Administrator use case



## Cockburn Diagrams for Use Cases

### Create payment order

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#1* | *Create payment order* | | |
| Goal in Context | Generate a new payment order. | | |
| Scope & Level | System under design ; Level = User goal | | |
| Preconditions | The user must be logged in the system. | | |
| Success End Condition | A payment order is created and stored in the system. | | |
| Failed End Condition | The user presses “Home” in “Payment Orders” mockup. | | |
| Primary Actor | Back Office Operator. | | |
| Trigger | The operator presses “Payment Orders” in “Main Menu” mockup. | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** |
| *1* |  | Shows mockup “Payment Orders” |
| *2* | Presses “New payment order” |  |
| *3* |  | Shows mockup “Candidate Payment Orders” |
| *4* | Selects a bill |  |
| *5* |  | Enables “Create Payment Order” |
| *6* | Presses “Create Payment Order” |  |
| *7* |  | Shows mockup “Confirm” |
| *8* | Presses “Yes” |  |
| *9* |  | Shows mockup “Operation Success” |
| *10* | Presses “OK” |  |
|  | *11* |  | *Returns to mockup “Payment Orders”* |

|  |  |  |  |
| --- | --- | --- | --- |
| EXTENSIONS | **Step n°** | **Back Office Operator** | **System** |
|  | *2.1* | Presses “Home” |  |
|  | *2.2* |  | *Shows mockup “Main Menu”* |
| SUBVARIATIONS | *Step* | **Back Office Operator** | **System** |
|  | *2.2* | Sets filtering parameters and presses “Filter” |  |
|  | *3.2* |  | *Shows the filtered table and return to step #2* |

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#2* | *Delete payment order* | | |
| Goal in Context | Delete a selected payment order. | | |
| Scope & Level | System under design ; Level = User goal | | |
| Preconditions | The user must be logged in the system. | | |
| Success End Condition | A payment order is deleted from the system. | | |
| Failed End Condition | The user presses “Home” in “Payment Orders” mockup. | | |
| Primary Actor | Back Office Operator. | | |
| Trigger | The operator presses “Payment Orders” in “Main Menu” mockup. | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** |
| *1* |  | Shows mockup “Payment Orders” |
| *2* | Selects a payment order |  |
| *3* |  | Enables “Delete” |
| *4* | Presses “Delete” |  |
| *5* |  | Shows mockup “Confirm” |
| *6* | Presses “Yes” |  |

### Delete payment order

|  |  |  |  |
| --- | --- | --- | --- |
|  | *7* |  | Shows mockup “Operation Success” |
| *10* | Presses “OK” |  |
|  | *11* |  | *Returns to mockup “Payment Orders”* |
| EXTENSIONS | **Step n°** | **Back Office Operator** | **System** |
|  | *2.1* | Presses “Home” |  |
|  | *3.1* |  | *Shows mockup “Main Menu”* |
| SUBVARIATIONS | *Step* | **Back Office Operator** | **System** |
|  | *2.2* | Sets filtering parameters and presses “Filter” |  |
|  | *3.2* |  | *Shows the filtered table and return to step #2* |

### Issue payment order

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#3* | Issue payment orders. | | |
| Goal in Context | User issues a payment order. | | |
| Scope & Level | System Under Design; Level = “User goal” | | |
| Preconditions | User must be logged. | | |
| Success End Condition | A payment order is issued. | | |
| Failed End Condition | User presses ‘Home’ in mockup ‘Payment Orders’. | | |
| Primary Actor | Back office operator. | | |
| Trigger | Back office operator presses ‘Payment Orders’ in mockup ‘Main menu’. | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** |
| *1* | Presses ‘Payment Orders’ in mockup ‘Main’. |  |
| *2* |  | Shows mockup ‘Payment Orders’. |
| *3* | Selects a not issued payment order. |  |
| *4* |  | Enables button ‘Issue’. |
| *5* | Presses ‘Issue’. |  |
| *6* |  | Shows mockup ‘Confirm’. |
| *7* | Presses ‘Yes’. |  |
| *8* |  | Shows mockup ‘Successful operation’. |
| *9* | Presses ‘Ok’. |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | *10* |  | Shows mockup ‘Payment Orders’ in which the payment order concerned shall be issued, and UC successfully ends. |
| EXTENSIONS | ***Step*** | **Back Office Operator** | **System** |
|  | *\*.1* | Presses ‘Home’ in mockup ‘Payment Orders’. |  |
| *\*2.1* |  | Shows mockup ‘Main’ and UC fails. |
|  | *5.2* | Presses ‘No. |  |
| *6.2* |  | Back to step 2. |
| SUBVARIATIONS | ***Step*** | **Back Office Operator** | **System** |
|  | *3.3* | Filters a not issued payment order, selects. |  |
| *4.3* |  | Back to step 4. |

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#4* | Archives payment orders. | | |
| Goal in Context | User archives a suspended payment order. | | |
| Scope & Level | System Under Design; Level = “User goal” | | |
| Preconditions | User must be logged. | | |
| Success End Condition | A payment order is archived. | | |
| Failed End Condition | User presses ‘Home’ in mockup ‘Payment Orders’. | | |
| Primary Actor | Back office operator. | | |
| Trigger | Back office operator presses ‘Payment Orders’ in mockup ‘Main menu’. | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** |
| *1* | Presses ‘Payment Orders’ in mockup ‘Main menu’. |  |
| *2* |  | Shows mockup ‘Payment Orders’. |
| *3* | Selects a suspended payment order. |  |
| *4* |  | Enables button ‘Archive’. |
| *5* | Presses ‘Archive’. |  |
| *6* |  | Shows mockup ‘Confirm’. |
| *7* | Presses ‘Yes’. |  |
| *8* |  | Shows mockup ‘Successful operation’. |
| *9* | Presses ‘Ok’. |  |

### Archive payment order

|  |  |  |  |
| --- | --- | --- | --- |
|  | *10* |  | Shows mockup ‘Payment Orders’ in which there isn’t the payment order concerned because it shall be archived, and UC successfully ends. |
| EXTENSIONS | ***Step*** | **Back Office Operator** | **System** |
|  | *\*.1* | Presses ‘Home’ in mockup ‘Payment Orders’. |  |
| *\*2.1* |  | Shows mockup ‘Main menu’ and UC fails. |
|  | *5.2* | Presses ‘No. |  |
| *6.2* |  | Back to step 2. |
| SUBVARIATIONS | ***Step*** | **Back Office Operator** | **System** |
|  | *3.3* | Filters suspended payment order, selects it. |  |
| *4.3* |  | Back to step 4. |

### Reissue payment order

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#5* | Reissue payment orders. | | |
| Goal in Context | User reissues a suspended payment order. | | |
| Scope & Level | System Under Design; Level = “User goal” | | |
| Preconditions | User must be logged. | | |
| Success End Condition | A payment order is reissued. | | |
| Failed End Condition | User presses ‘Home’ in mockup ‘Payment Orders’. | | |
| Primary Actor | Back office operator. | | |
| Trigger | Back office operator presses ‘Payment Orders’ in mockup ‘Main menu’. | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** |
| *1* | Presses ‘Payment Orders’ in mockup ‘Main menu’. |  |
| *2* |  | Shows mockup ‘Payment Orders’. |
| *3* | Selects a suspended payment order. |  |
| *4* |  | Enables button ‘Reissue’. |
| *5* | Presses ‘Reissue’. |  |
| *6* |  | Shows mockup ‘Confirm’. |
| *7* | Presses ‘Yes’. |  |
| *8* |  | Shows mockup ‘Successful operation’. |
| *9* | Presses ‘Ok’. |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | *10* |  | Shows mockup ‘Payment Orders’ in which the payment order concerned shall be issued, and UC successfully ends. |
| EXTENSIONS | ***Step*** | **Back Office Operator** | **System** |
|  | *\*.1* | Presses ‘Home’ in mockup ‘Payment Orders’. |  |
| *\*2.1* |  | Shows mockup ‘Main menu’ and UC fails. |
|  | *5.2* | Presses ‘No. |  |
| *6.2* |  | Back to step 2. |
| SUBVARIATIONS | ***Step*** | **Back Office Operator** | **System** |
|  | *3.3* | Filters suspended payment order, selects it. |  |
| *4.3* |  | Back to step 4. |

### Saves payment order as paid

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#6* | *Saves payment order as paid* | | |
| Goal in Context | Saving a payment order as paid | | |
| Scope & Level | System under design ; Level = User goal | | |
| Preconditions | User must be logged in. | | |
| Success End Condition | The payment order is saved in the system as paid. | | |
| Failed End Condition | There is no payment order that can be saved as paid. The user interrupts the use case. | | |
| Primary Actor | Back Office Operator. | | |
| Trigger | The operator presses button “Payment Order” in “Main Menu” mockup. | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** |
|  | *1* |  | Shows mockup “Payment Orders” |
| *2* | Selects a “notified” payment order from the table and presses “Save as paid” button |  |
| *3* |  | Shows mockup “Confirm” |
|  | *4* | Presses OK |  |
| *5* |  | Shows mockup “Operation Success” |
| *6* | Presses OK |  |
| *7* |  | *Returns to mockup “Payment Orders”* |

|  |  |  |  |
| --- | --- | --- | --- |
| EXTENSIONS | **Step n°** | **Back Office Operator** | **System** |
|  | *2a* | Presses “Home” |  |
|  | *3a* |  | *Shows mockup “Main Menu”* |
| SUBVARIATIONS | ***Step n°*** | **Back Office Operator** | **System** |
|  | *2b* | Sets filtering parameters and presses “Filter” |  |
| *3b* |  | *Shows the filtered table and return to step #2* |

### Saves a payment order as suspended

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USE CASE *#7* | *Saves a payment order as suspended.* | | | |
| Goal in Context | Saving a payment order as suspended. | | | |
| Scope & Level | System under design; Level = User goal. | | | |
| Preconditions | User must be logged in. | | | |
| Success End Condition | The payment order is saved in the system as suspended. | | | |
| Failed End Condition | There is no payment order that can be saved as suspended. The user interrupts the use case. | | | |
| Primary Actor | Back Office Operator. | | | |
| Trigger | The operator presses button “Payment Order” in “Main Menu”. | | | |
| DESCRIPTION | **Step n°** | **Back Office Operator** | **System** | |
|  | *1* |  | Shows mockup “Payment Orders” |
| *2* | Selects a “notified” payment order from the table and presses “Save as suspended” button |  |
| *3* |  | Shows mockup “Confirm” |
| *4* | Presses OK |  |
| *5* |  | Shows mockup “Operation Success” |
| *6* | Presses OK |  |
| *7* |  | *Returns to mockup “Payment Order”* |

|  |  |  |  |
| --- | --- | --- | --- |
| EXTENSIONS | **Step n°** | **Back Office Operator** | **System** |
|  | *2a* | Presses “Home” |  |
|  | *3a* |  | *Shows mockup “Main Menu”* |
| SUBVARIATIONS | ***Step n°*** | **Back Office Operator** | **System** |
|  | *2b* | Sets filtering parameters and presses “Filter” button |  |
| *3b* |  | *Shows the filtered table and return to step #2* |

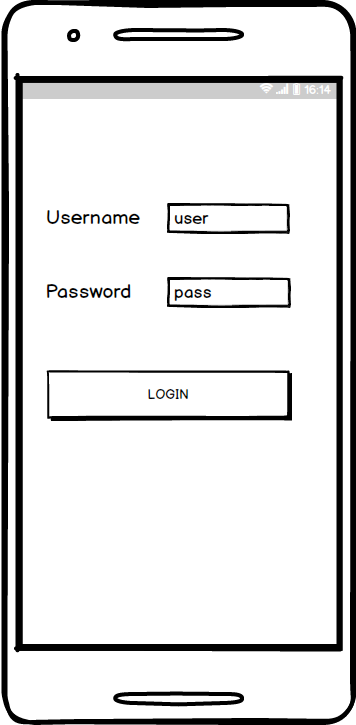
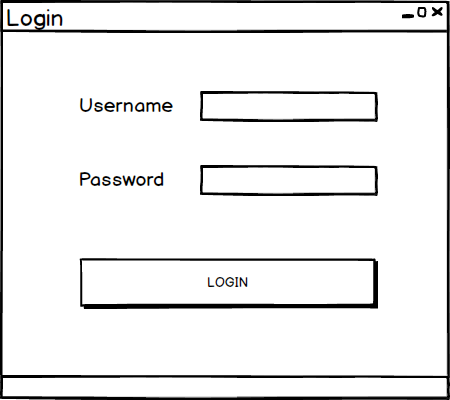
### Sends meter readings

|  |  |  |  |
| --- | --- | --- | --- |
| USE CASE *#8* | *Sends meter readings.* | | |
| Goal in Context | Reading the water consumption of a customer. | | |
| Scope & Level | System under design ; Level = User goal | | |
| Preconditions | User must be logged in. | | |
| Success End Condition | The Reading Operator saves the water consumption and sends it to the server. | | |
| Failed End Condition | The Readings Operator closes the application.  The Readings Operator doesn’t have readings left to do.  There is no Internet connection to send the readings done. | | |
| Primary Actor | Readings Operator | | |
| Trigger | User logs in the mobile app. | | |
| DESCRIPTION | **Step n°** | **Readings Operator** | **System** |
|  | *1* |  | Shows mockup “Readings Main” |
| *2* | Selects a customer from the list and clicks “Save reading” button |  |
| *3* |  | Shows mockup “Save Reading” |
| *4* | Inserts the amount of water consumed and presses “Save” button |  |
| *5* |  | Shows mockup “Confirm Reading” containing all the reading’s informations |
| *6* | Presses “Yes” button |  |
| *7* |  | *Returns to mockup “Readings Main”* |
| *8* | Presses “Send readings” button |  |

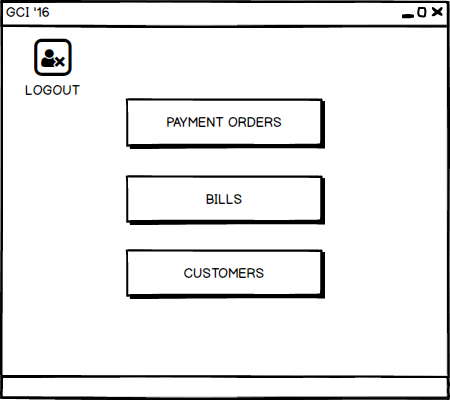
|  |  |  |  |
| --- | --- | --- | --- |
|  | *9* |  | *Shows mockup “Successful Sending”* |
| *10* | Presses OK |  |
| *11* |  | *Returns to mockup “Readings Main”* |
| EXTENSIONS | **Step n°** | **Readings Operator** | **System** |
|  | *9a* |  | *Shows mockup “Failed Sending”* |
| *10a* | Presses OK |  |
| *11a* |  | *Returns to mockup “Readings Main”* |
| SUBVARIATIONS | ***Step n°*** | **Readings Operator** | **System** |
|  | *8b* | Returns to step #1 |  |

## User Interface Mockups

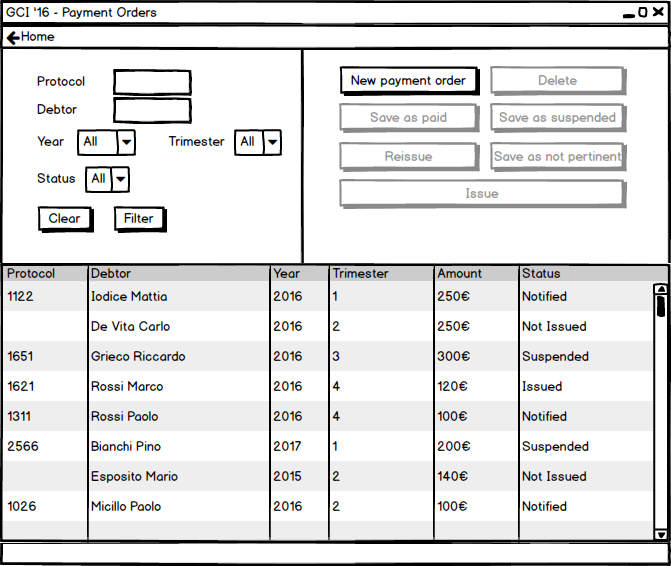
### Login



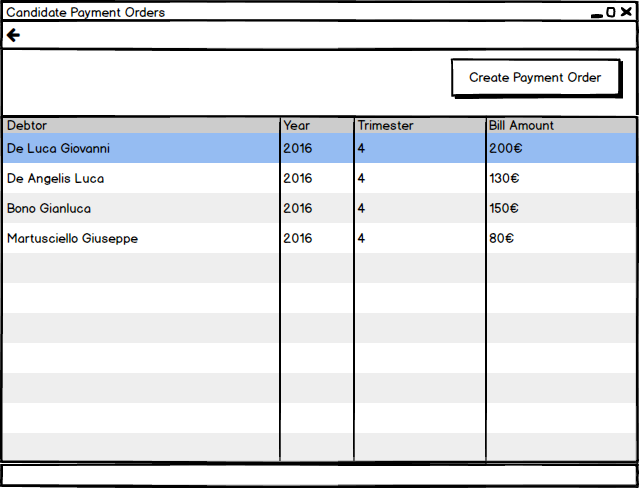
### Main Menu



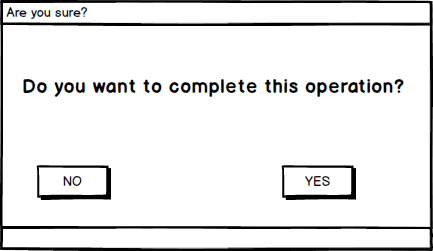
### Payment Order



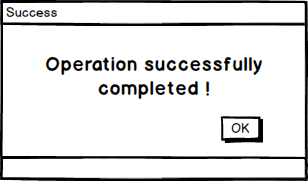
### Candidate Payment Orders



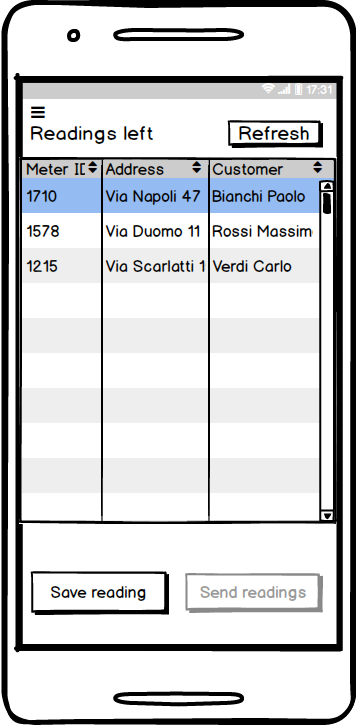
### Confirm



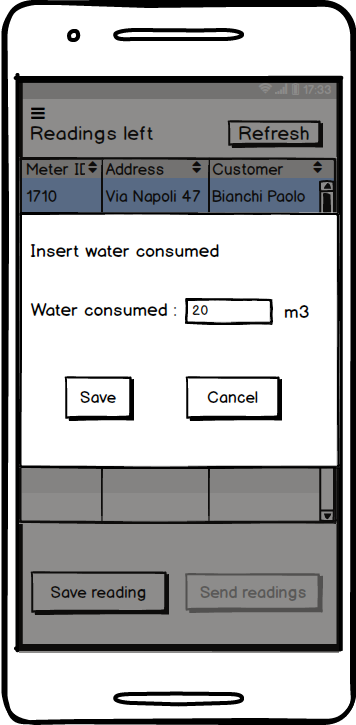
### Operation Success



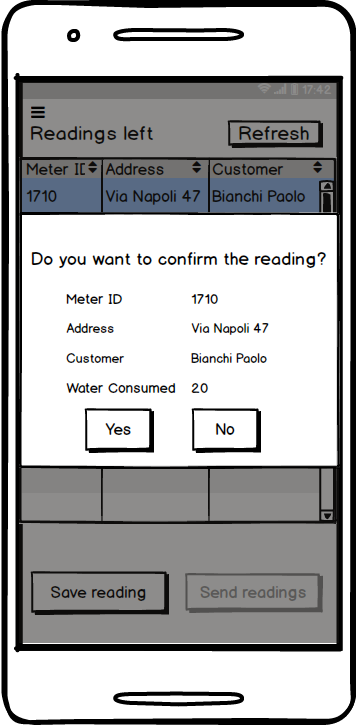
### Readings Main



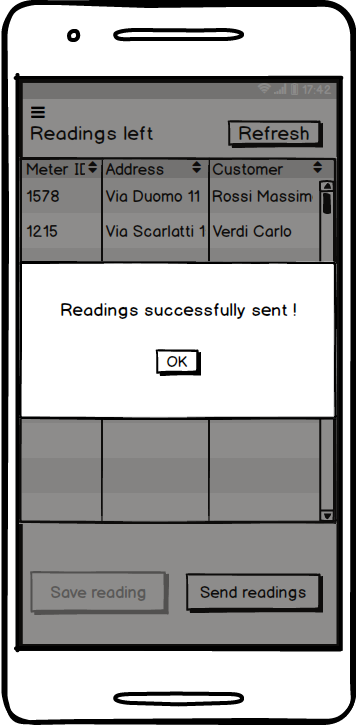
### Save Reading



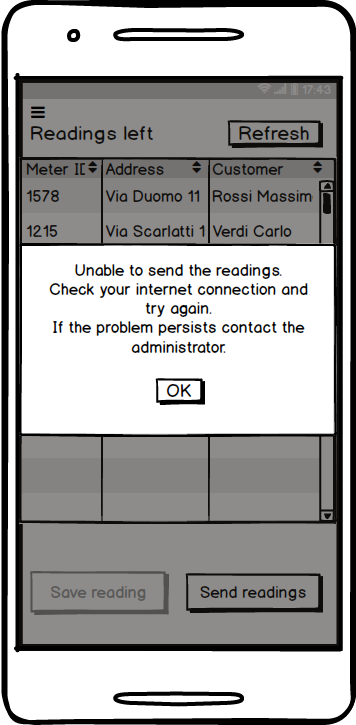
### Confirm Reading



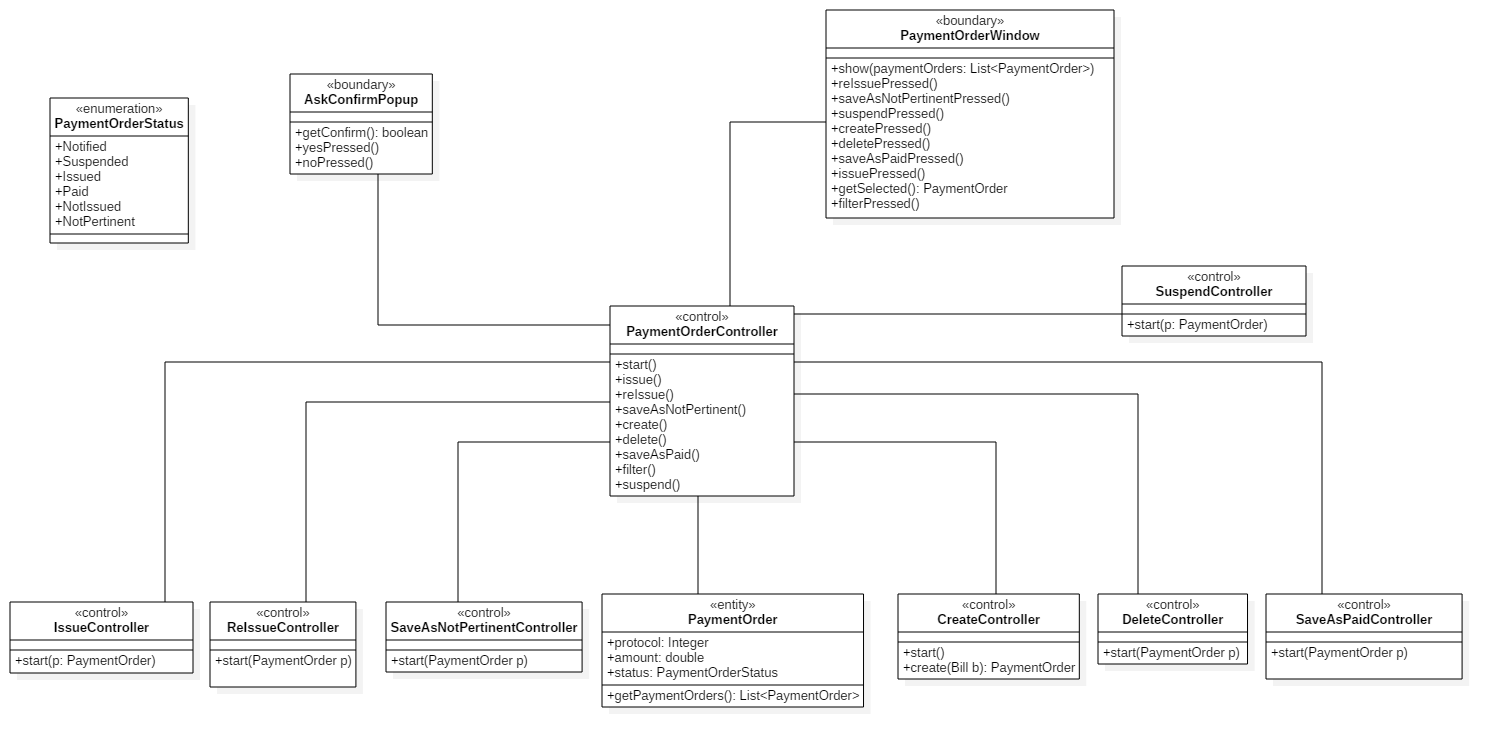
### Successful Sending



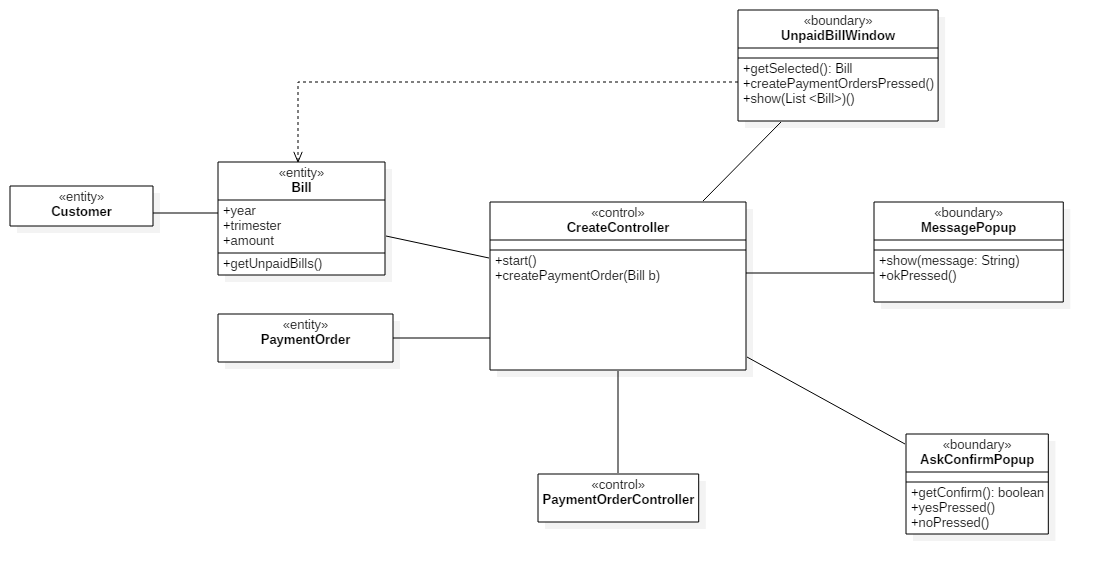
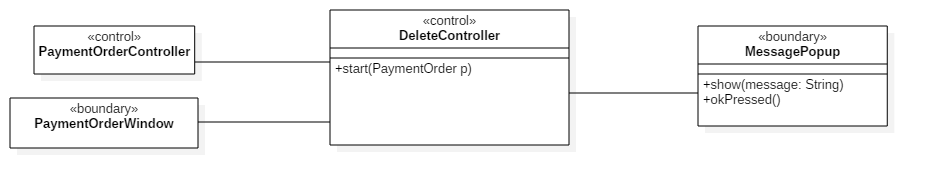
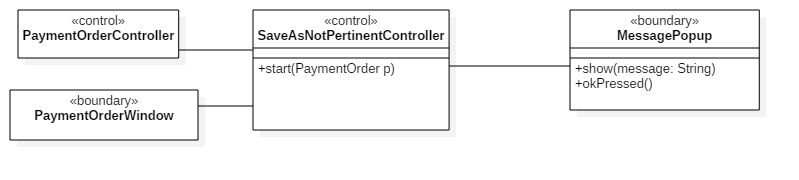
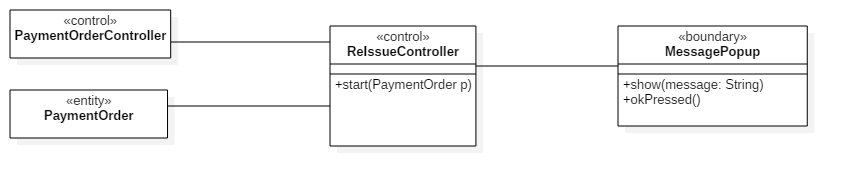
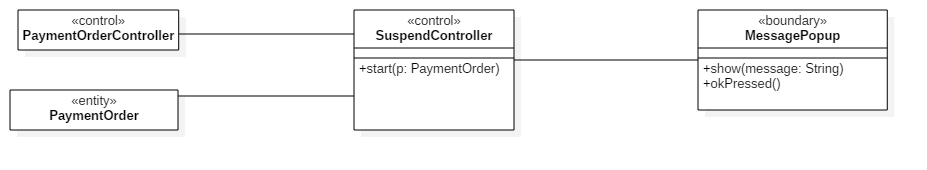
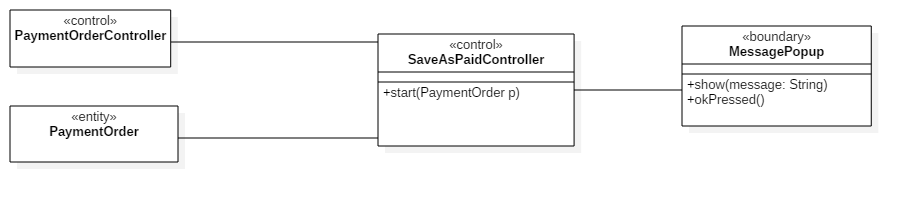
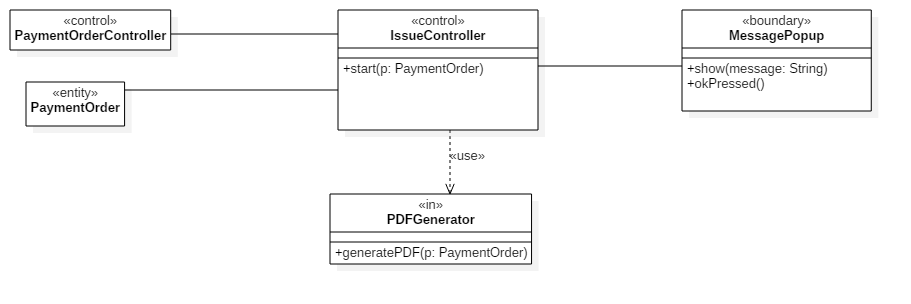
### Failed Sending



## Entity-Boundary-Control Class Diagram

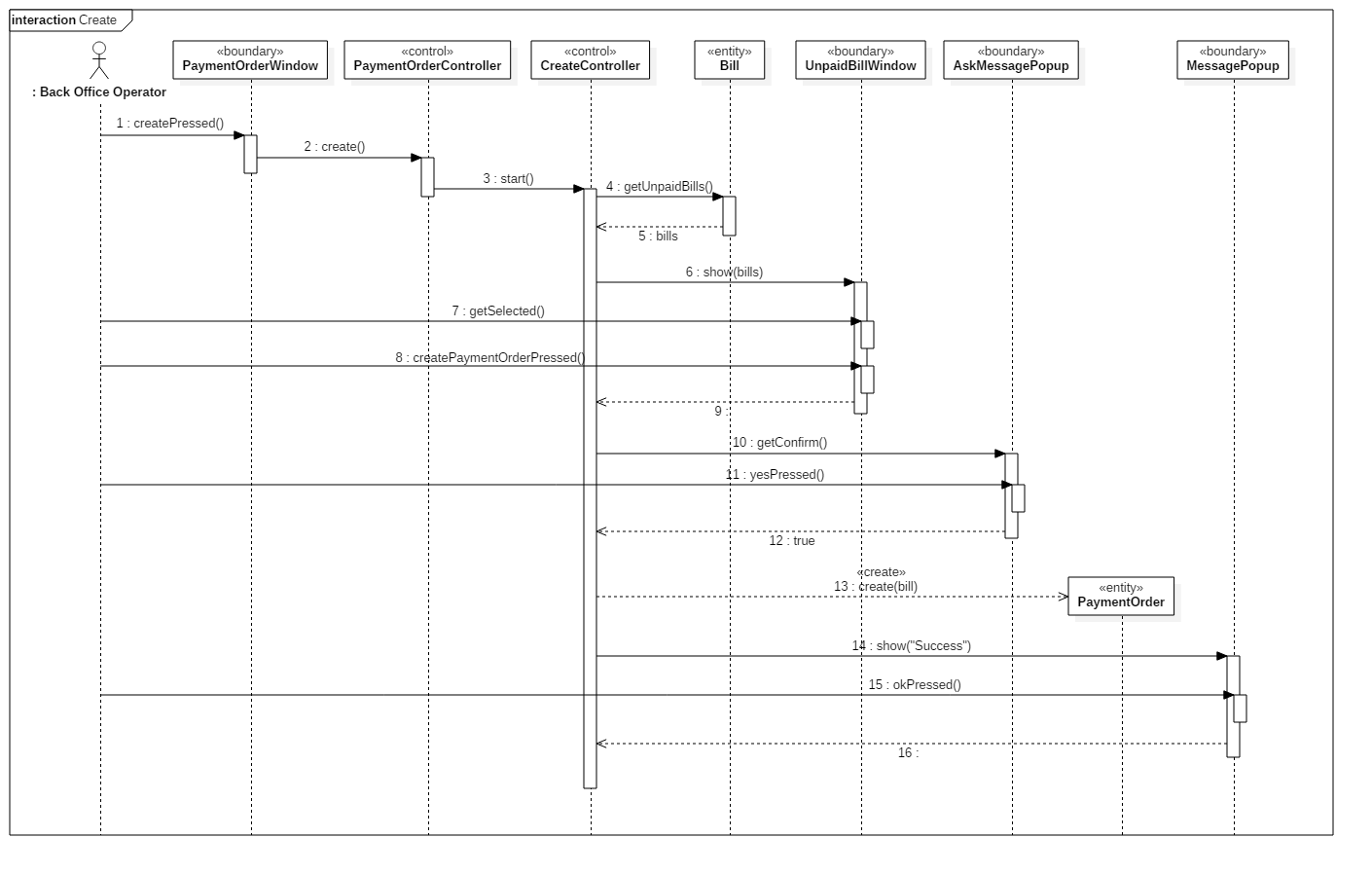


The class diagrams below specify the behaviour of controllers.

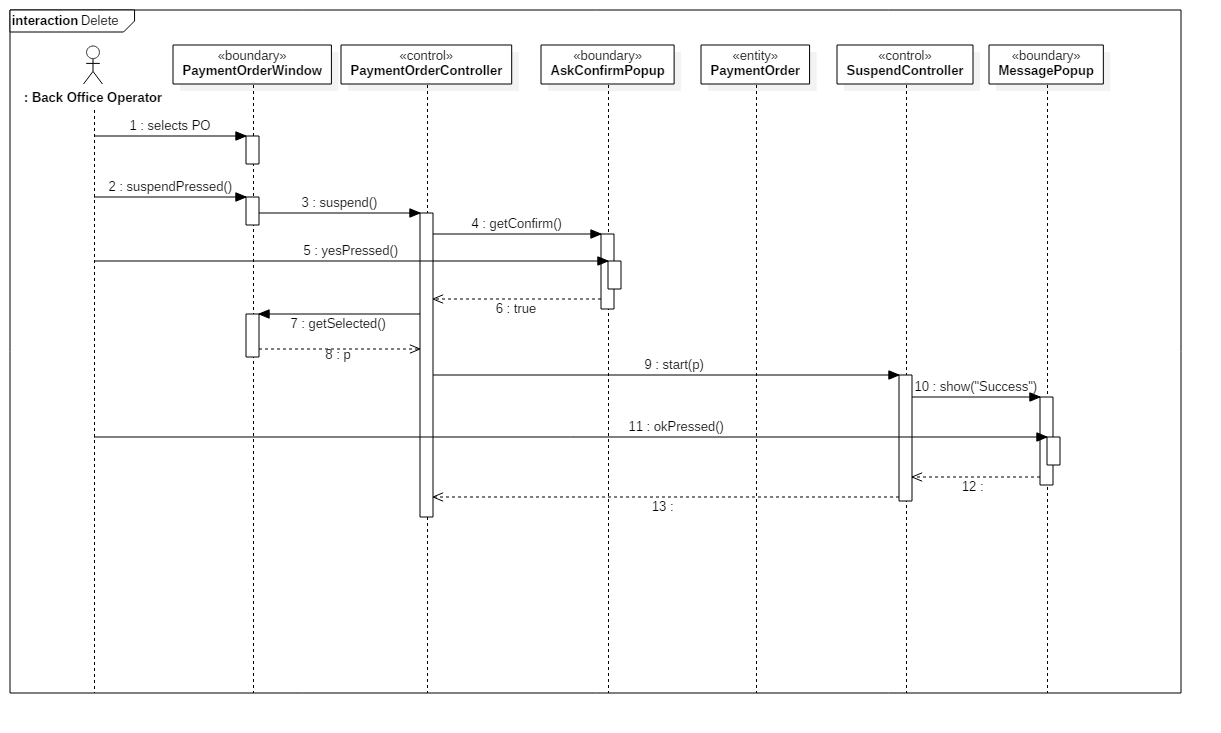


## Sequence Diagrams

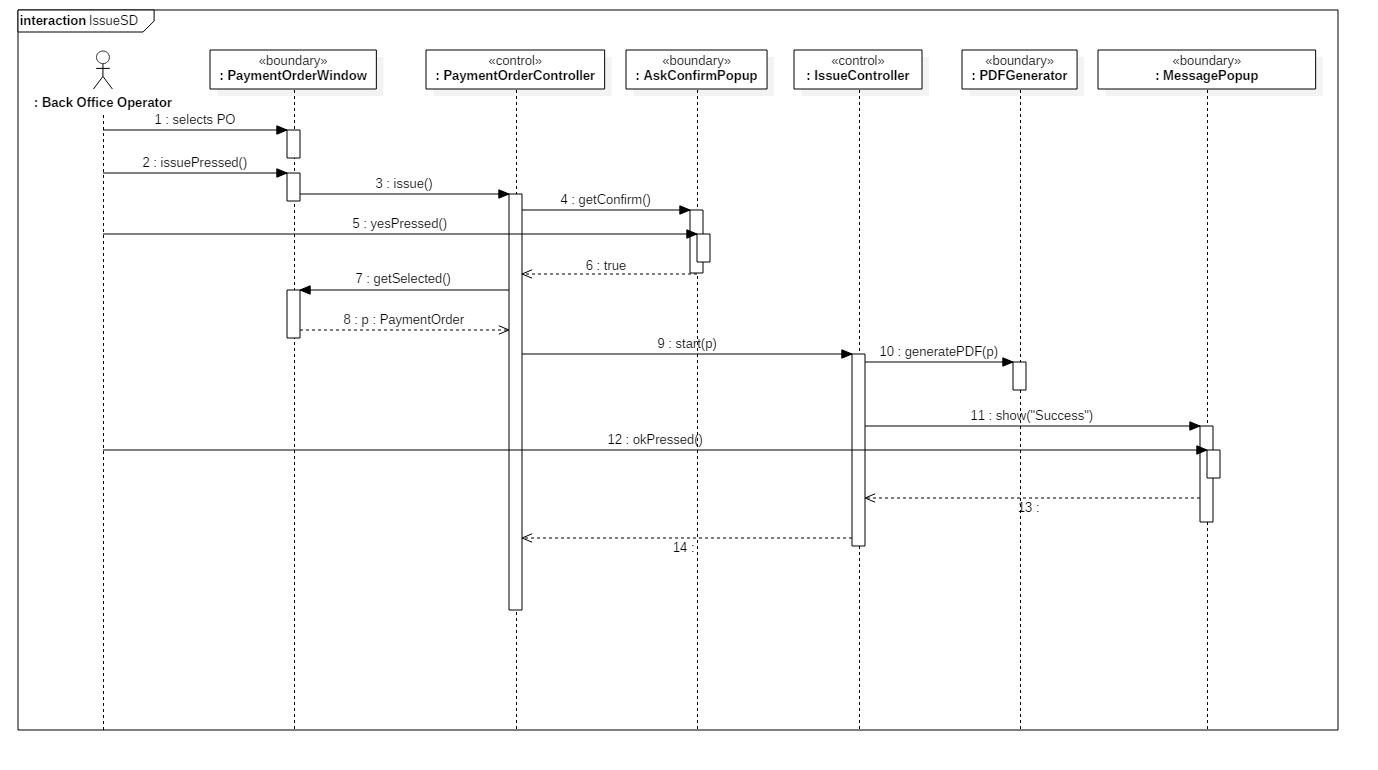
### Create payment order



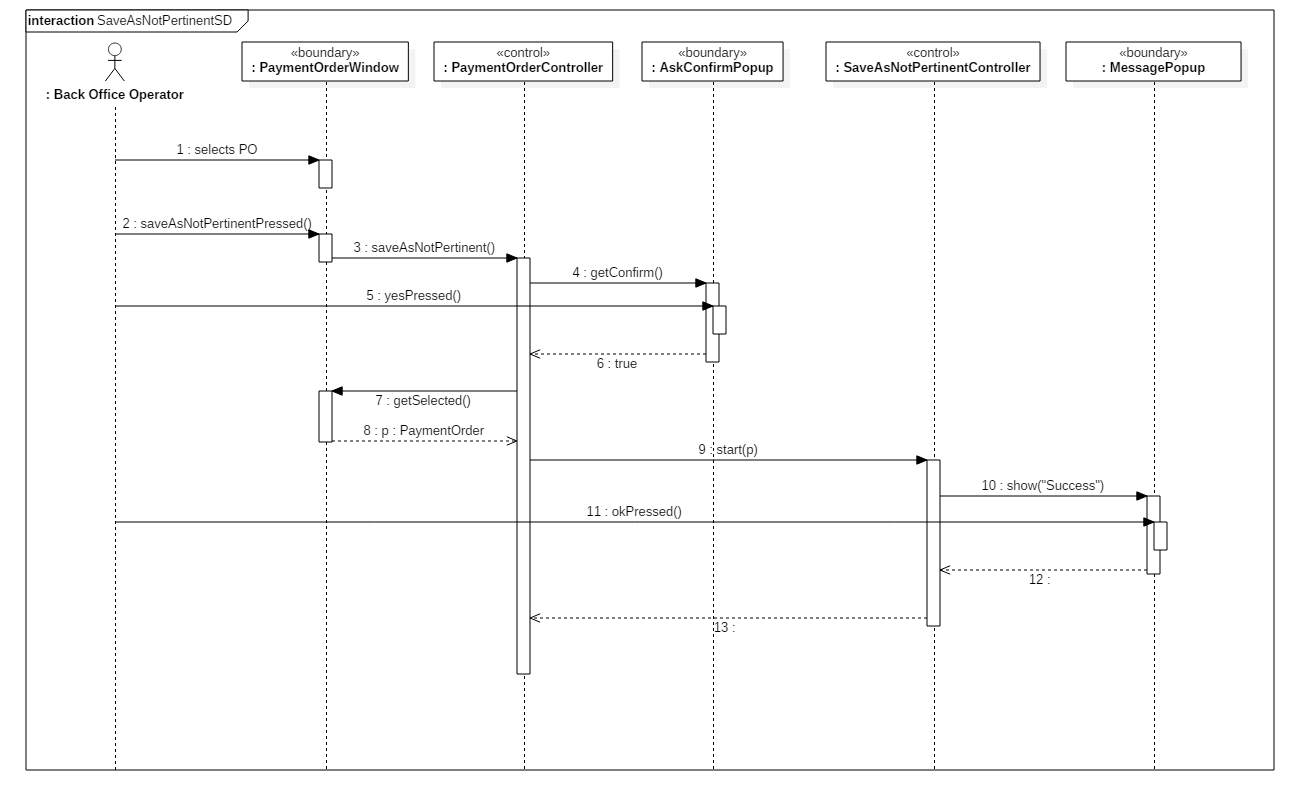
### Delete payment order



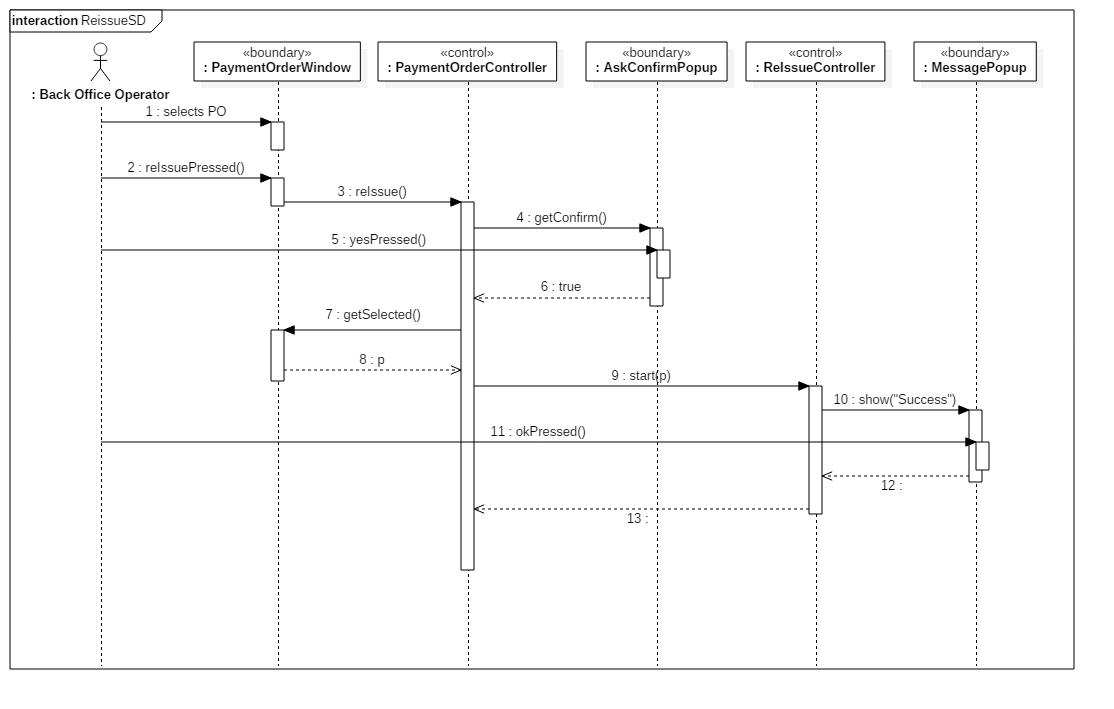
### Issue payment order



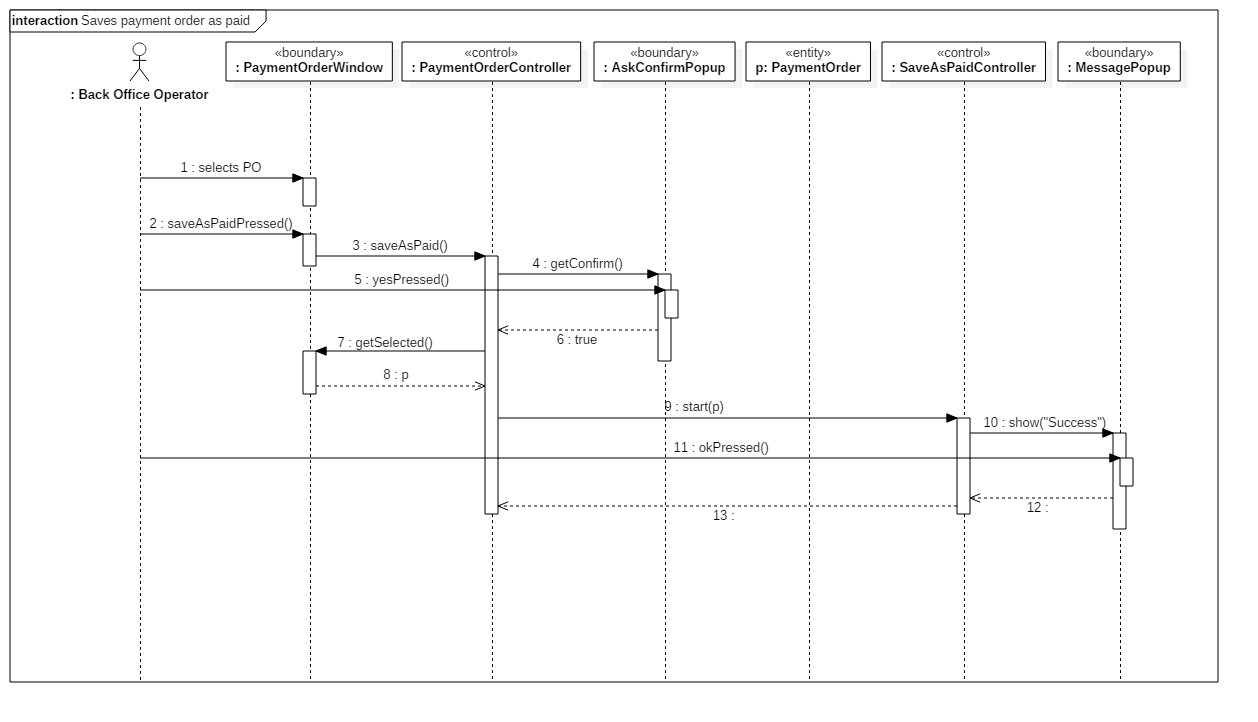
### Save payment order as not pertinent



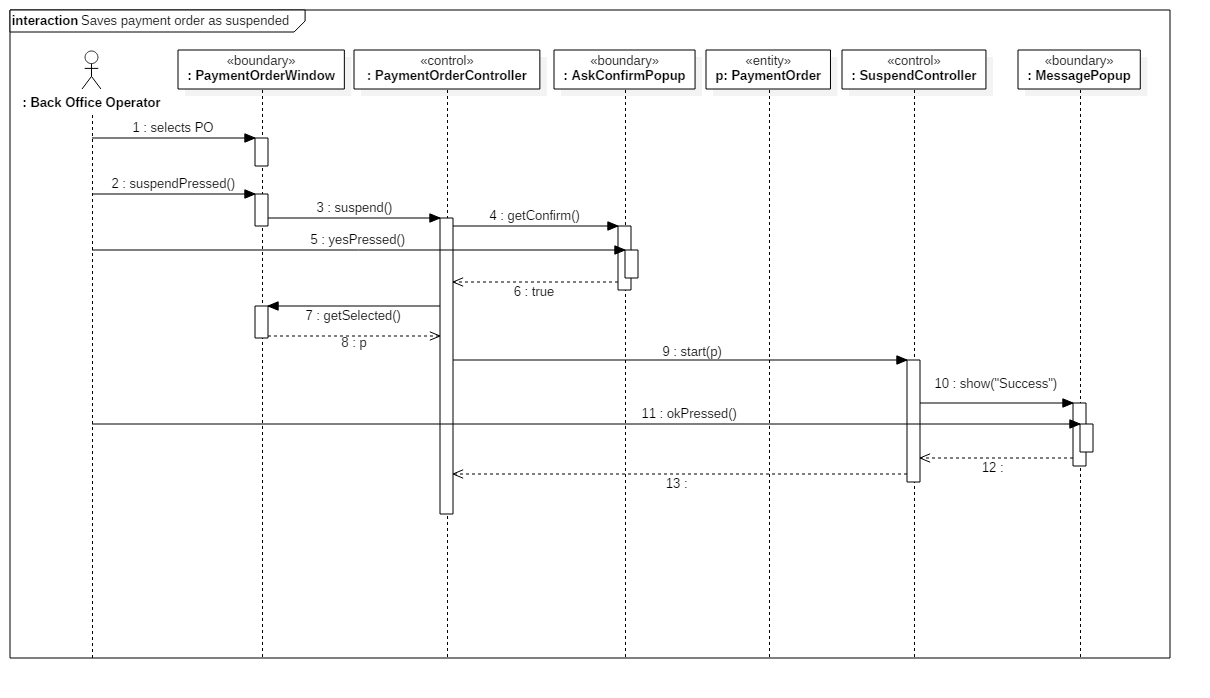
### Reissue payment order



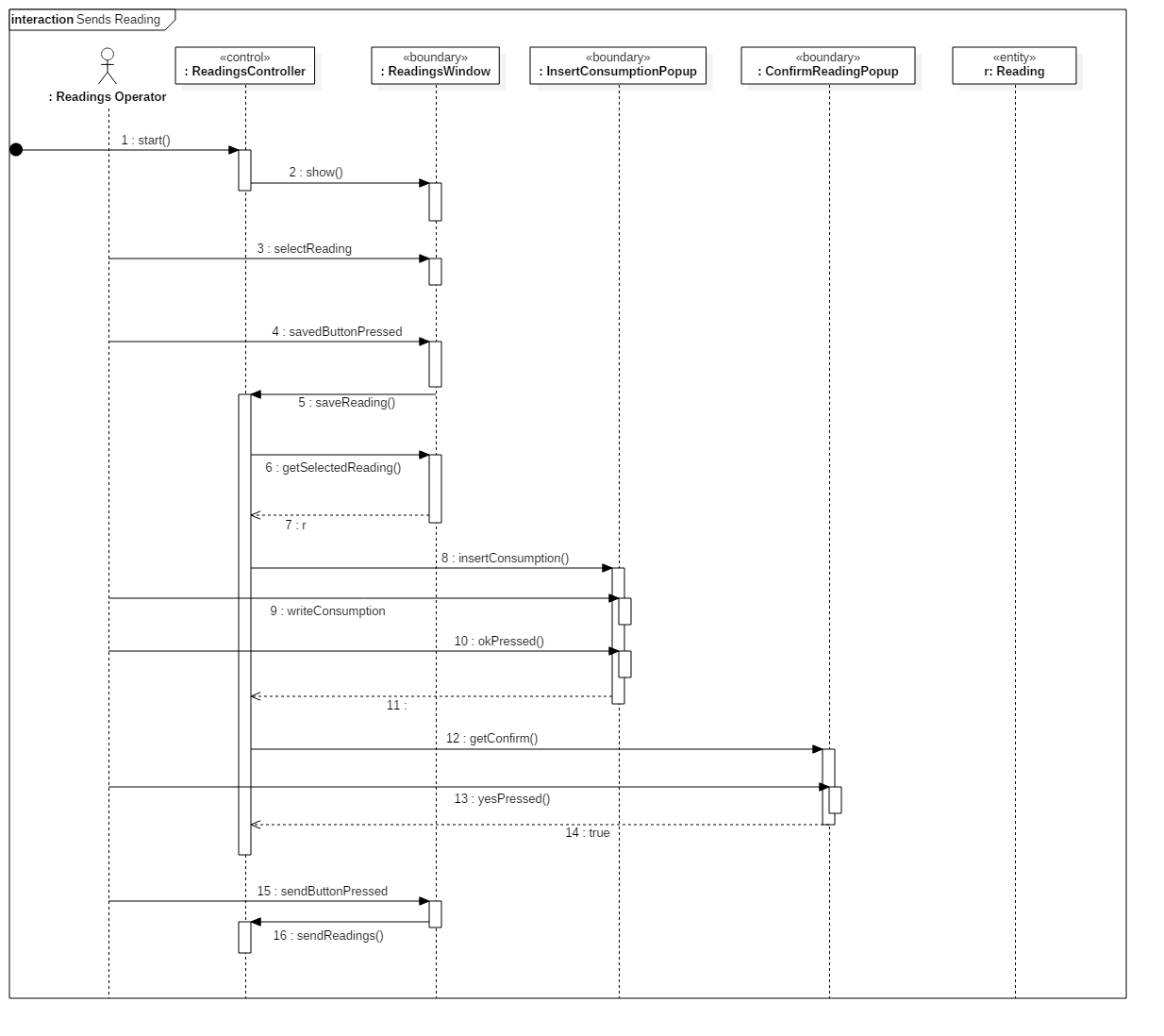
### Save payment order as paid



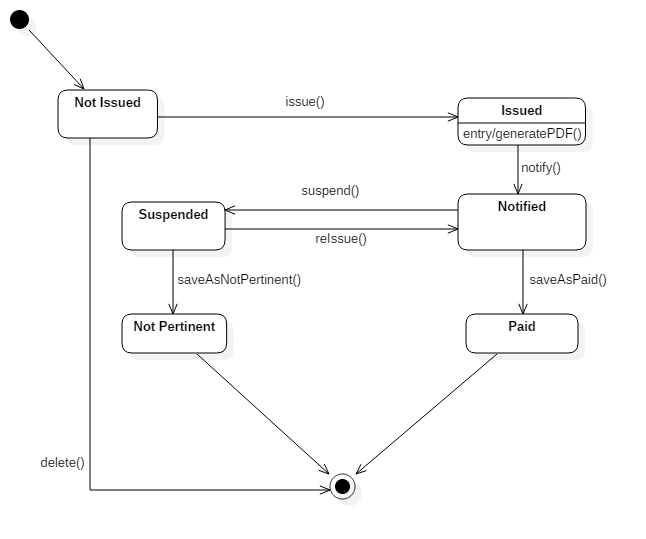
### Save payment order as suspended



### Send readings



## State chart diagram payment order



## Glossary (A-Z)

**Back Office Operator** User of desktop application, his goal is to handle the entire payment order management system.

**Issue** Render a payment order effective. After issuing a payment order, it must be paid by the debtor.

**Not pertinent** If the debtor won the legal case, the payment order would be archived. The debtor won’t have to pay the payment order anymore.

**Payment order** Document which requires the debtor to pay within a defined deadline.

**Protocol number** Every issued payment order is labeled by an index, a protocol number, which is considered a unique identifier.

**Suspend** A payment order can be suspended in presence of a legal case.

**Reading** Reading of water consumption reported on the meter.

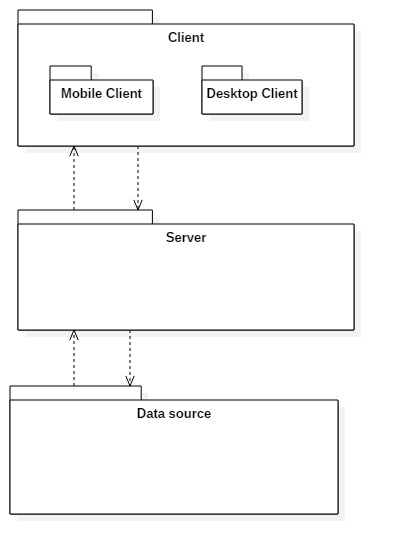
**Reading Operator** User of mobile application.

# Design documentation

## System design

### Software architecture

The system division is based on the Three-Tier Architecture, which divides the system into three modules: client, server, and data source.  
In this way most of the business logic is located in a central server, the code is more scalable and maintainable and the client is independent of the data source.



**Client**:  
Consists on the application which directly interacts with the users.   
This module is divided into two parts: client for readings operator, which is the mobile app that lets the users read the consumptions from the assigned meters, and a desktop application for back office operators.

**Server**:  
Represents the application's logical component, which provides authentication tokens, performs and verifies the correctness of the required operations, for example the status of a payment order.  
It was decided to use only a server to handle application requests, because it is estimated that the number of simultaneous connections is not high.

**Data source** :   
Relational DBMS in which persistent data are saved.

### Client server communication

Client and server communicate using HTTP messages. The server provides a session cookie after a succesful login request and the client must send back the cookie for any further communication. The server listens on port 8081 and replies to client requests with the following response codes:

|  |  |  |
| --- | --- | --- |
| Response code | Reason phrase | Description |
| 200 | OK | The request has been succesfully accomplished. |
| 500 | Internal server error | The request could not be accomplished due to an error encountered in the server. |
| 461 | Wrong id or password | The user has submitted a wrong id or password. |
| 462 | No session | The client has not sent a valid session cookie or the cookie sent has expired. |
| 463 | Missing parameter | The request expected a parameter which has not been sent. |
| 464 | Bad parameter values | One or more parameters presented an unexpected value. |
| 465 | Not practicable action | The requested action could not be applied to the given parameter. |

## Technologies

|  |  |  |
| --- | --- | --- |
| Technology | Version | Usages |
| Java | JDK 1.8 | Client application. |
| Apache Tomcat[1] | 8.5 | Server. |
| Java EE | Java EE 6 | Servlet for server web services. |
| JSON[2] |  | Client and server data exchange. |
| Oracle Database | 11g | Database. |
| Android | SDK 16 – API 4.1 | Mobile appliction for Readings Operators. |
| jUnit | 4.9 | Unit Testing |
| Mockito | 1.9.1 | Unit Testing |

### Libraries

*Gson* (Google JSON) is a Java library that can be used to convert Java objects into their JSON representation. It can be used to retrieve an object from a JSON string.

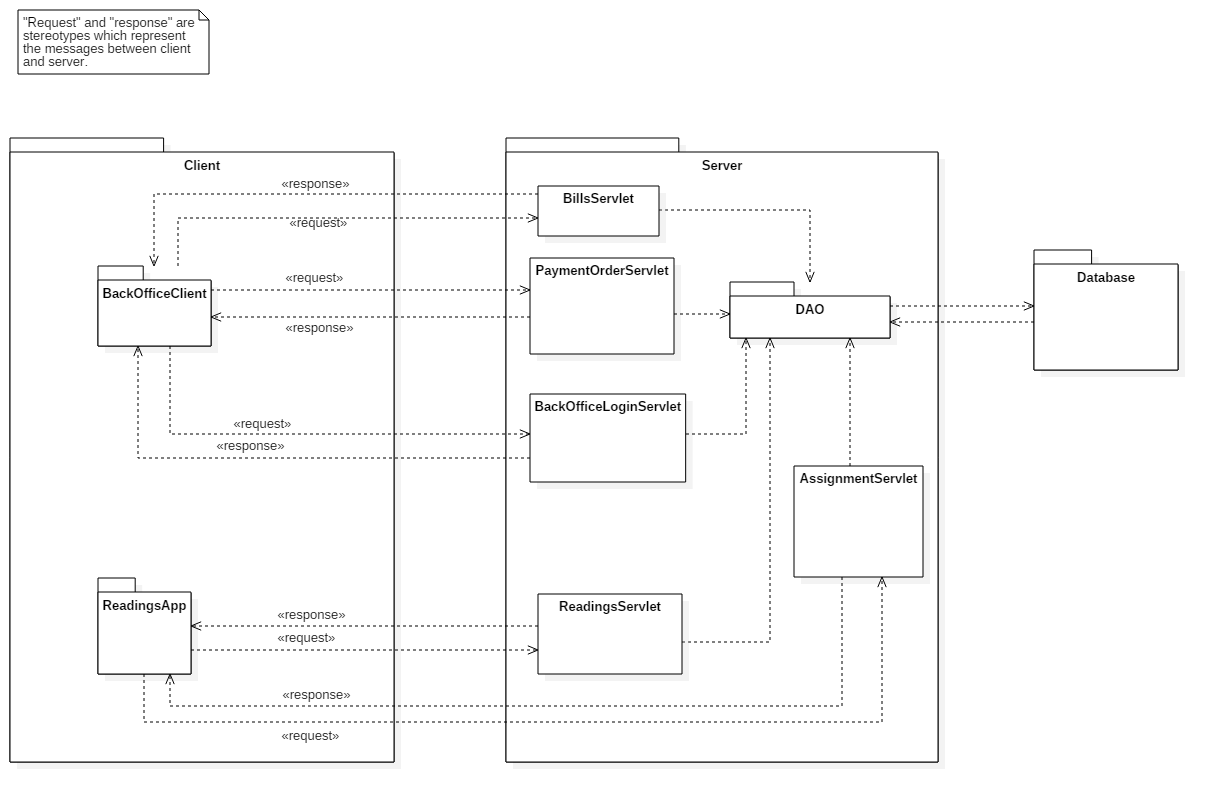
iText is a Java library for creating and manipulating PDF files in Java. It is used to generate the PDF with all the information of payment order after it is issued.

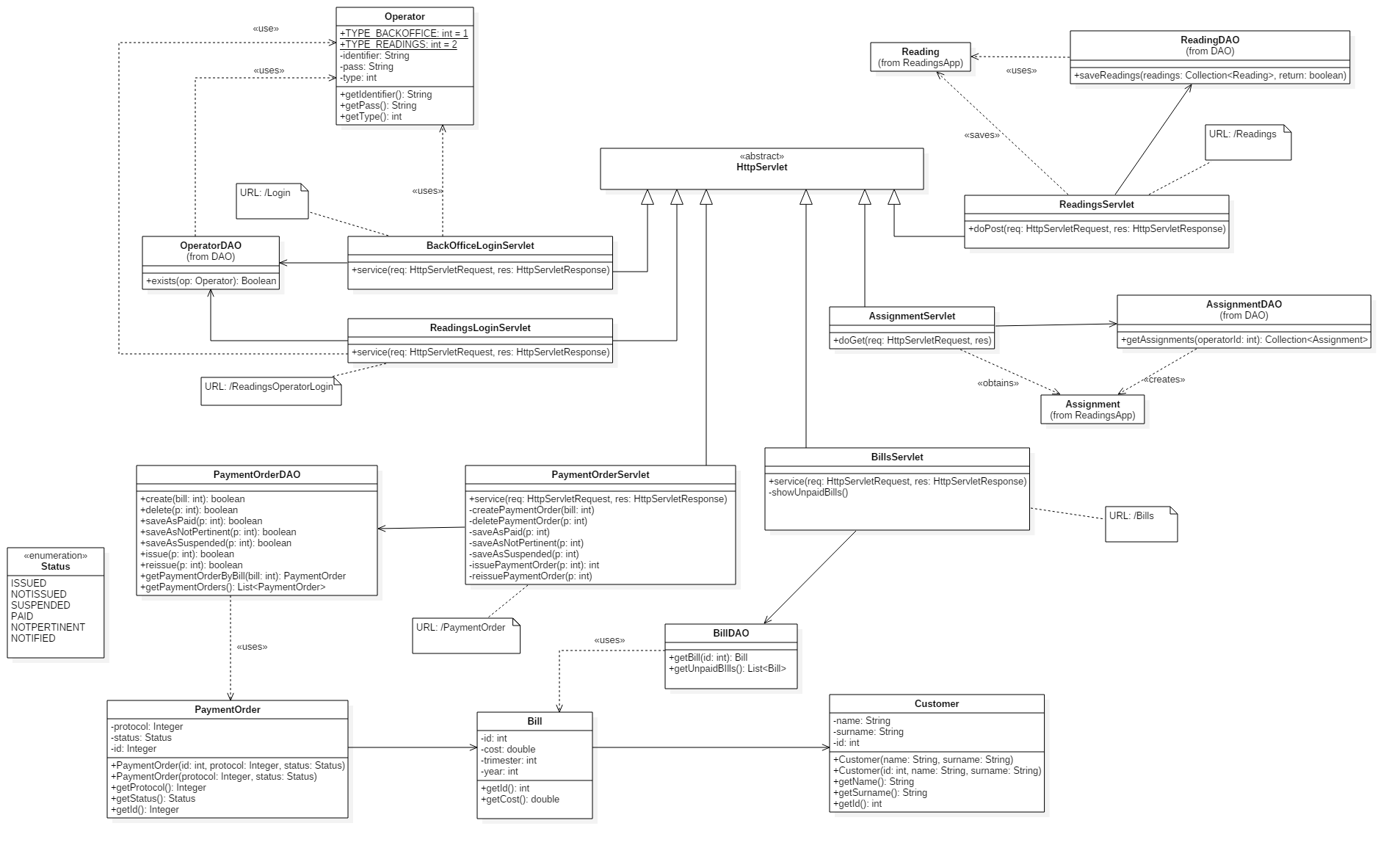
*OJDBC* is a Java library that provides drivers for Oracle Databases.

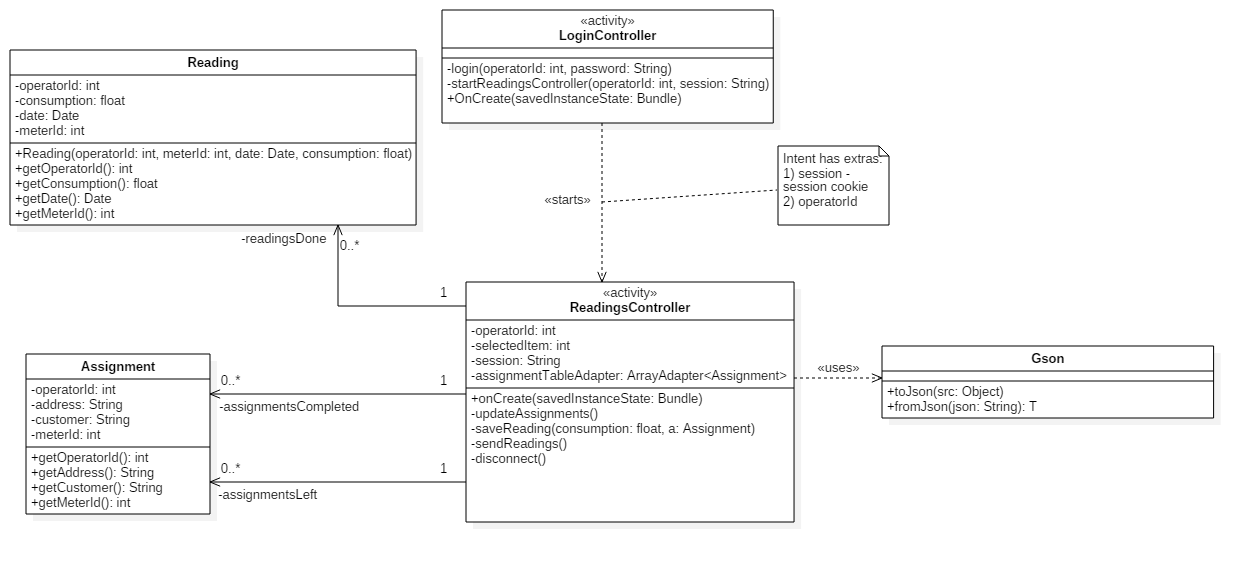
*Mockito* is a Java library that provides an easy way to create stub classes for jUnit testing.

## Object design

### Class diagrams







### Sequence diagrams

## CRC cards

### Desktop application

### Mobile application

|  |  |  |
| --- | --- | --- |
| **Class Name** | Assignment | |
| **Superclass** | Object | |
| **Subclasses** |  | |
| **Responsabilities** | | **Collaborators** |
| Represent the task of performing a reading of a meter. | |  |

|  |  |  |
| --- | --- | --- |
| **Class Name** | AssignmentServlet | |
| **Superclass** | HttpServlet | |
| **Subclasses** |  | |
| **Responsabilities** | | **Collaborators** |
| Provide an operator with the list of his assignments. | | Assignment, AssignmentDAO, Gson |

|  |  |  |
| --- | --- | --- |
| **Class Name** | LoginController | |
| **Superclass** | AppCompatActivity | |
| **Subclasses** |  | |
| **Responsabilities** | | **Collaborators** |
| Let the operator login from the mobile application. | |  |

|  |  |  |
| --- | --- | --- |
| **Class Name** | Reading | |
| **Superclass** | Object | |
| **Subclasses** |  | |
| **Responsabilities** | | **Collaborators** |
| Represent the reading of the water consumption reported on a meter. | |  |

|  |  |  |
| --- | --- | --- |
| **Class Name** | ReadingsController | |
| **Superclass** | AppCompatActivity | |
| **Subclasses** |  | |
| **Responsabilities** | | **Collaborators** |
| Let the user save the reading consumption reported on a meter. | | Assignment, Reading, Gson |
| Update user’s list of assignments. | | Assignment, Gson |
| Send the readings done to a remote server. | | Reading, Gson |

|  |  |  |
| --- | --- | --- |
| **Class Name** | ReadingsServlet | |
| **Superclass** | HttpServlet | |
| **Subclasses** |  | |
| **Responsabilities** | | **Collaborators** |
| Save into the database the readings performed by a Readings Operator. | | Reading, ReadingDAO, Gson |