Tourism Agency Application

Analysis and Design Document

Student:Ilas Alexandra

**Group:30235**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Use JAVA/C# API to design and implement an application for the agents of a tourism agency. The application should have two types of users (a regular user represented by the regular tourism agent and an administrator user) which have to provide a username and a password in order to use the application.

The regular user can perform the following operations:

* Add/update/view client information (name, identity card number, personal numerical code, address, etc.).
* Add/update/view/delete a holiday reservation for a client (destination, hotel name, number of persons who are going on holiday, details about each member going on holiday, total price, final payment date)
* Accept partial payments from a client before final payment date
* View all the clients who missed the final payment deadline and have the possibility to cancel their holiday

The administrator user can perform the following operations:

* CRUD on agents’ information.
* Generate reports for a particular period containing the activities performed by an agent.

# Functional Requirements

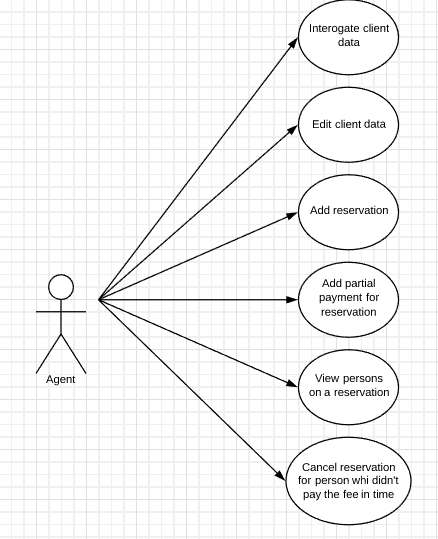
The application is crossed platform as it is implemented in Java.

# Non-functional Requirements

The application can be used by anyone at any time, given that it is running

2. Use-Case Model

**Use case diagram for agent:**



**Use-Case descriptions:**

Use case: Interrogate client data

Primary actor: Agent

Main success scenario: When asking the system for the clients for the specified agent the system returns a table with all his/her clients.

Extensions: Failure can occur when :

* the application crashes

Use case: Edit client data

Primary actor: Agent

Main success scenario: When asking the system for specified field from the clients table to edit the system allows to edit the certain field and update the data base

Extensions: Failure can occur when :

* the agent tries to edit fields that cannot be edited (such as: client PNC or id)
* the application crashes or triggers an exception

Use case: Add reservation

Primary actor: Agent

Main success scenario: When asking the system to add a reservation for a given client and the system allows to insert the new reservation into the data base and shows it in the reservation’s table

Extensions: Failure can occur when :

* the agent does not add the correct data for the reservation shown on the interface
* the application crashes or triggers an exception

Use case: Add partial payment for reservation

Primary actor: Agent

Main success scenario: When asking the system to add a partial payment for a selected reservation , the system updates the total payment of the reservation and shows the updated value in the reservation’s table

Extensions: Failure can occur when :

* the agent does not select a reservation from the table
* the agent does not add a valid sum(sum<=0 or a different type than int)
* the application crashes or triggers an exception

Use case: View persons of a reservation

Primary actor: Agent

Main success scenario: When asking the system to view persons that are part of the selected reservation , the system shows these persons in a table

Extensions: Failure can occur when :

* the agent does not select a reservation from the table
* the table doesn’t show the correct persons for that reservation

Use case: Cancel reservation for person who didn’t pay the fee in time

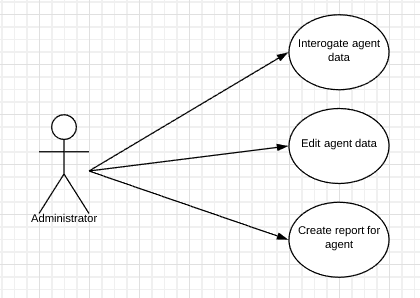
Primary actor: Agent

Main success scenario: When asking the system to view persons that are part of the selected reservation but didn’t pay the fee in time, the system shows these persons in a table

Extensions: Failure can occur when :

* the agent does not select a reservation from the table or the table containing the persons

**Use case diagram for administrator:**

****

**Use-Case descriptions:**

Use case: Interrogate agent data

Primary actor: Administrator

Main success scenario: When asking the system for all the agents registered in the application (contained in the data base) the system shows all of these in a table

Extensions: Failure can occur when :

* the application crashes

Use case: Edit agent data

Primary actor: Administrator

Main success scenario: When asking the system for specified field from the clients table to edit the system allows to edit the certain field and update the data base or when it comes to Delete, the application deletes the agent from the data base and removes the agent from the agent’s table

Extensions: Failure can occur when :

* the administrator tries to edit fields that cannot be edited (such as: client PNC or id)
* the administrator does not add the correct agent id when trying to delete
* the application crashes or triggers an exception

Use case: Create report for agent

Primary actor: Administrator

Main success scenario: When asking the application to create a report for a given agent id, the application creates a new .txt file with the agent’s activity from a certain period of time.

Extensions: Failure can occur when :

* the agent does not add a valid agent id
* the application crashes or triggers an exception

3. System Architectural Design

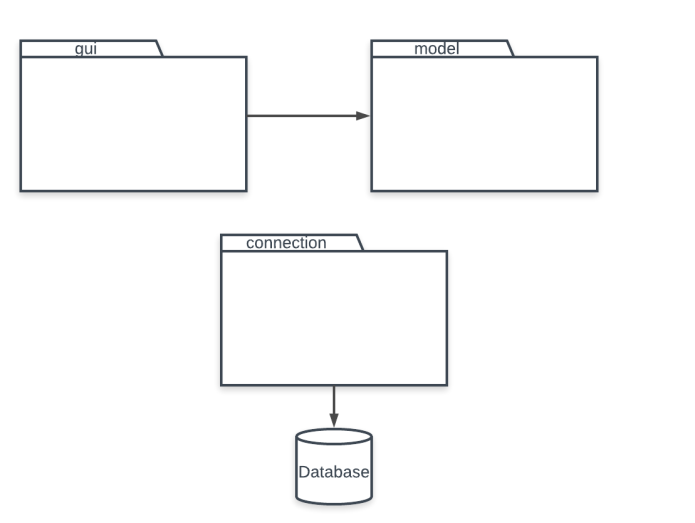
**3.1 Architectural Pattern Description**

The used architectural patterns are: Layered architecture and Active Record.

* Layered architecture: methods that access the data base , business logic methods and gui are separated

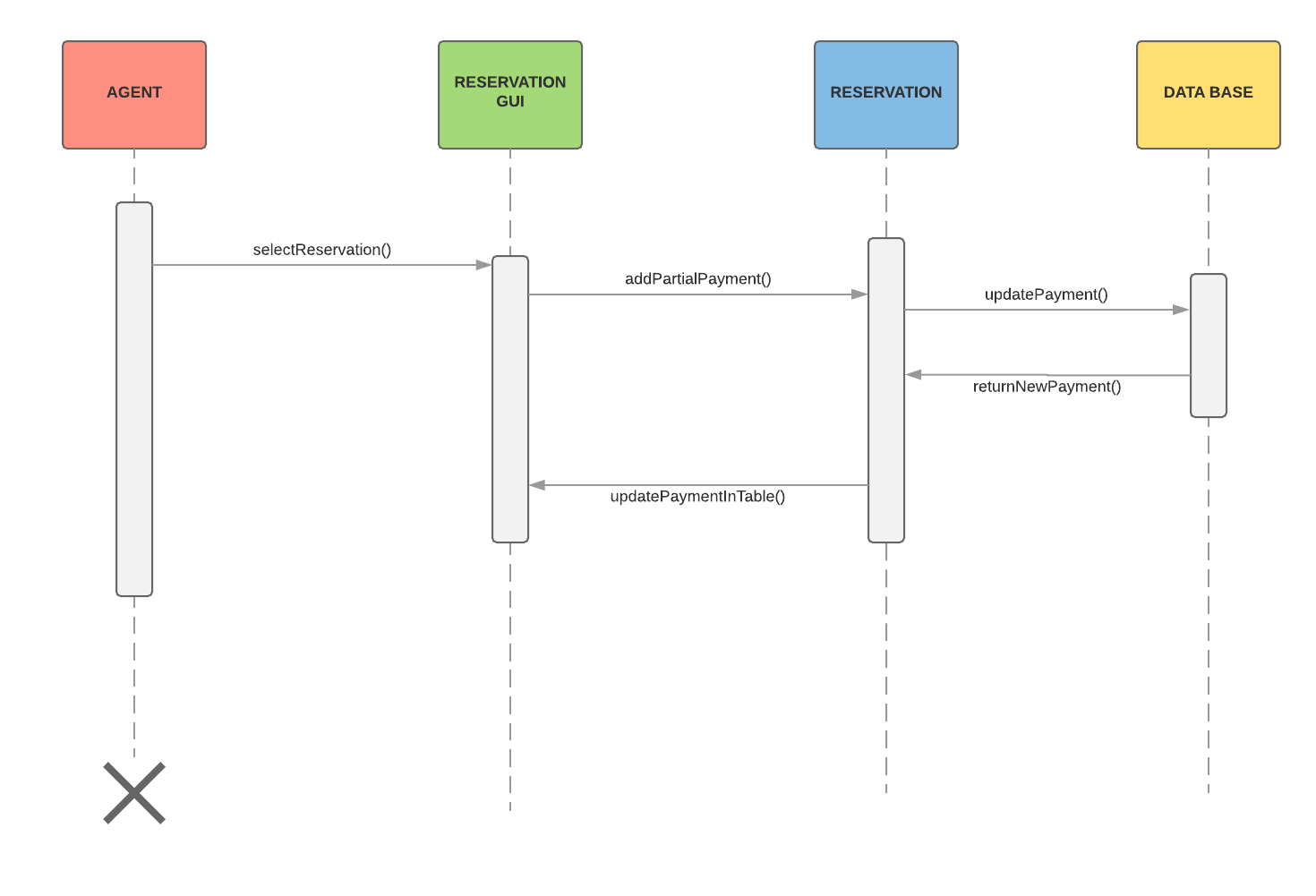
**3.2 Diagrams**

**Packages diagram:**



4. UML Sequence Diagram

**Adding a partial payment for a reservation:**



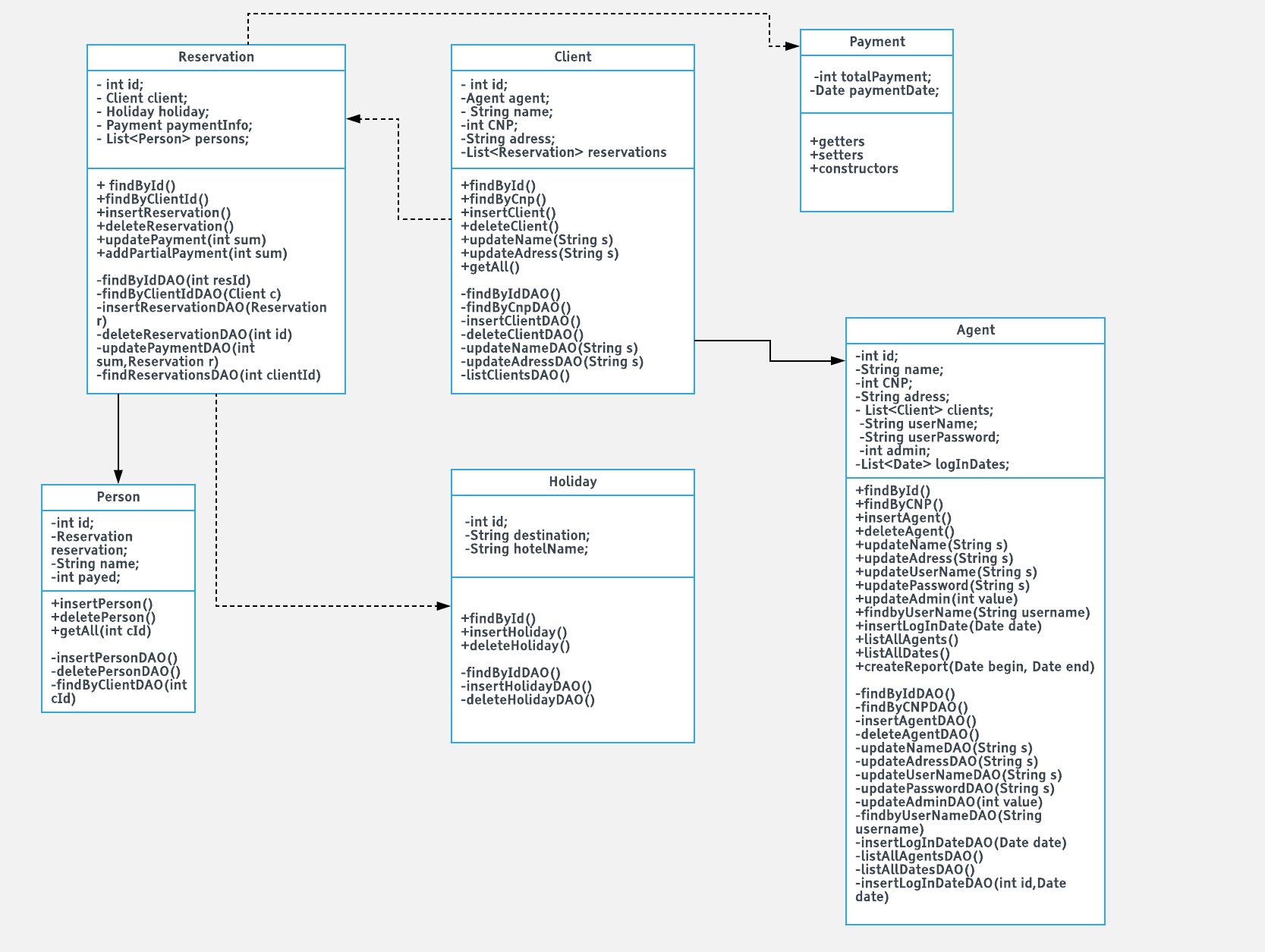
5. Class Design

**5.1 Design Patterns Description**

* Active record: data base accessing methods and business logic methods are in the same class, thus combining two different architectural patterns: Row Data Gateaway and Domain Model.

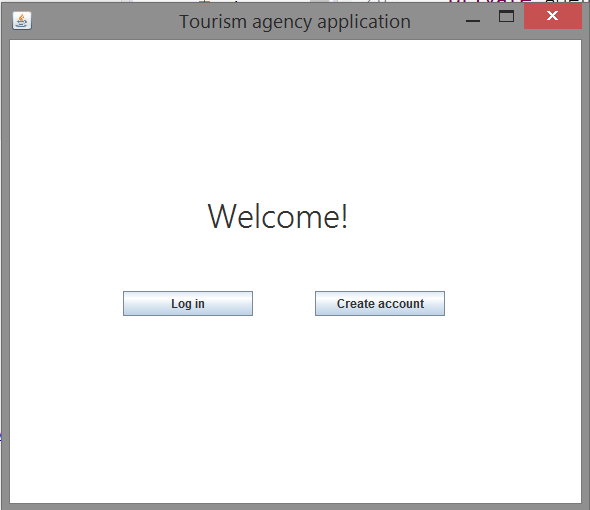
**5.2 UML Class Diagram**

With this class diagram I showcase the use of the Active Record pattern. All methods regarding a certain class model is in one class. The Layered architecture comes in as from exterior classes, the methods that access the data base cannot be called.

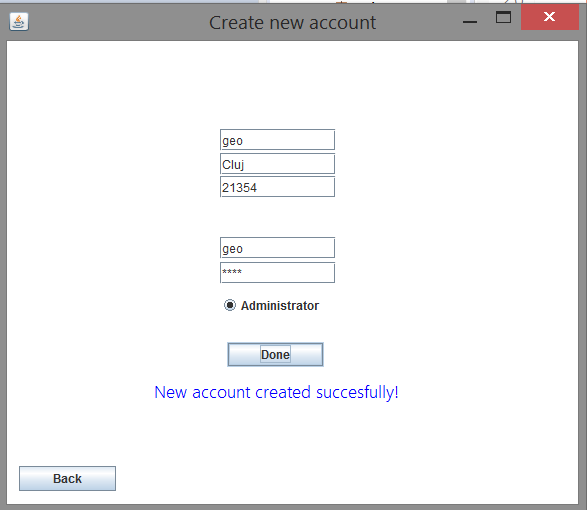


6. System Testing

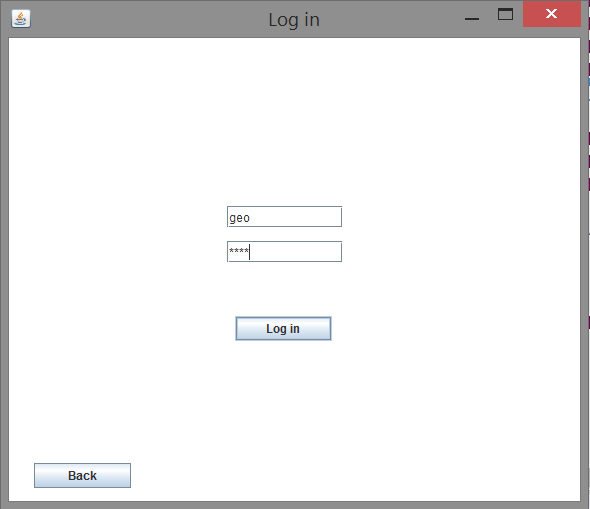
**6.1. Begin**



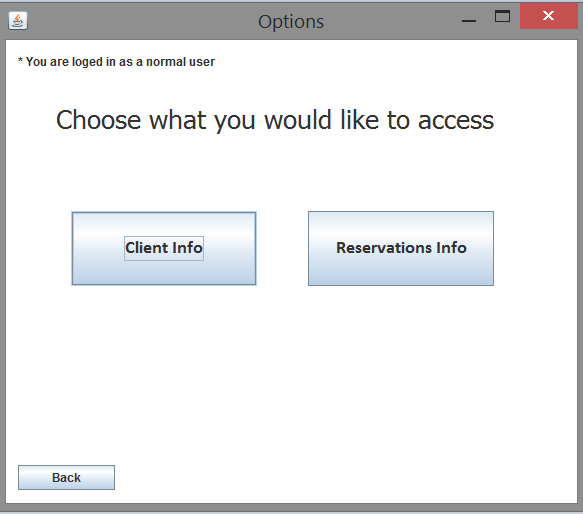
**6.2. Creating account**



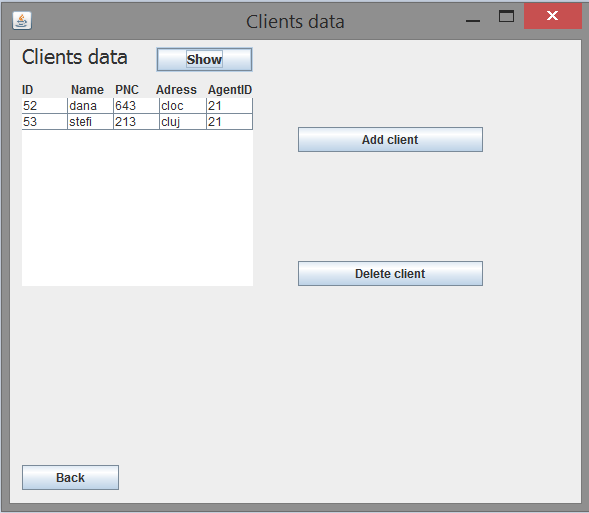
**6.3. Log in**

****

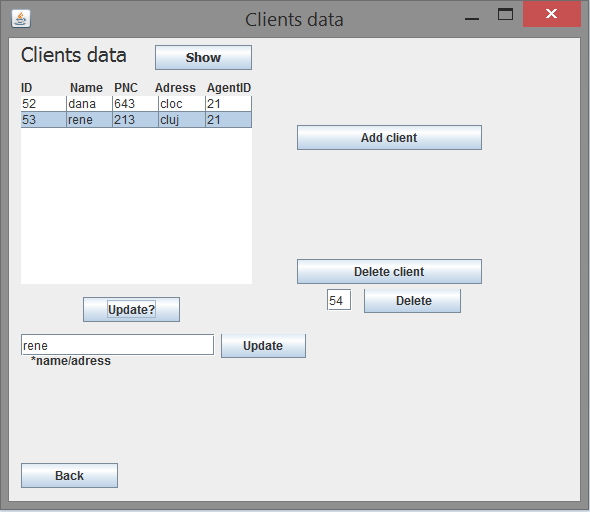
**6.4. Choosing an option as a normal user**

****

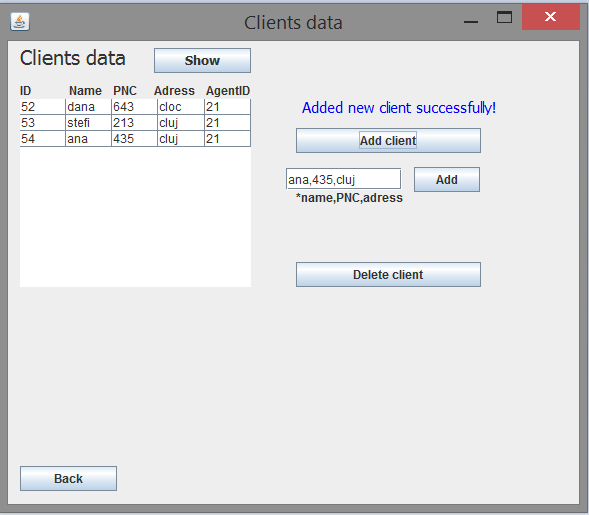
**6.5. Showing client data**

****

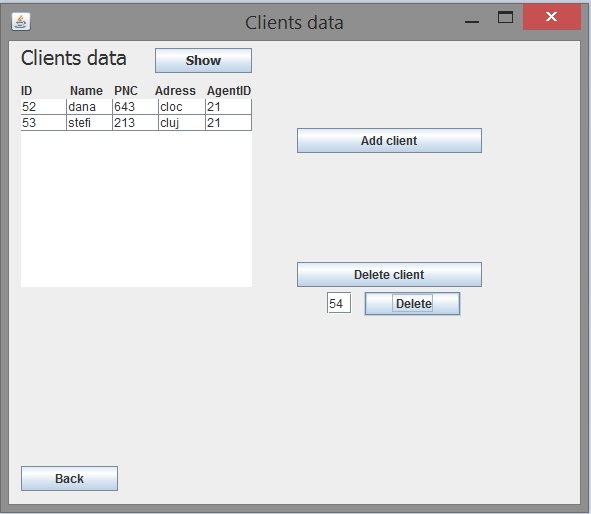
**6.6. Updating field**

****

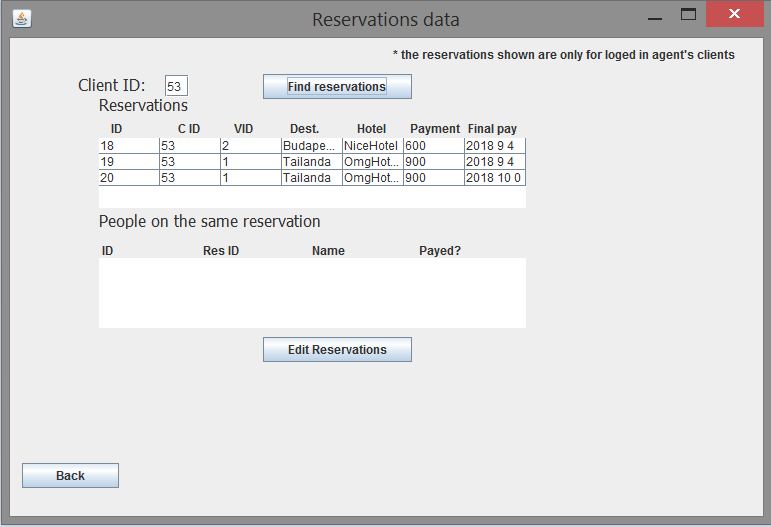
**6.7. Adding client**

****

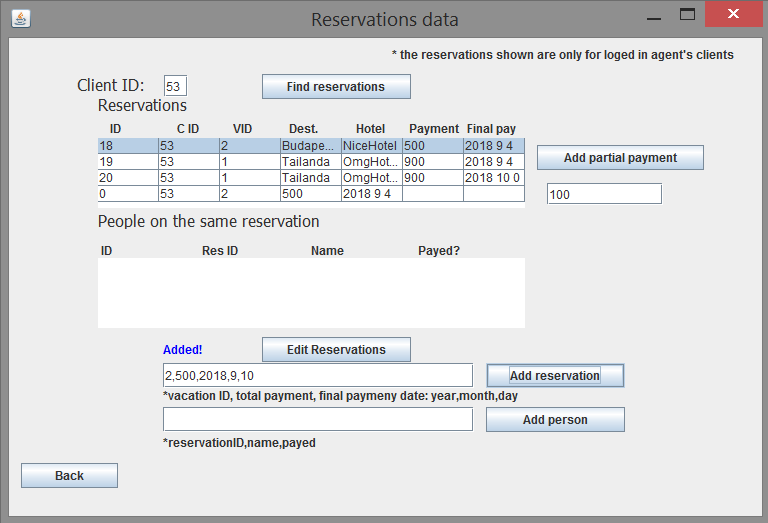
**6.8. Deleting client**

****

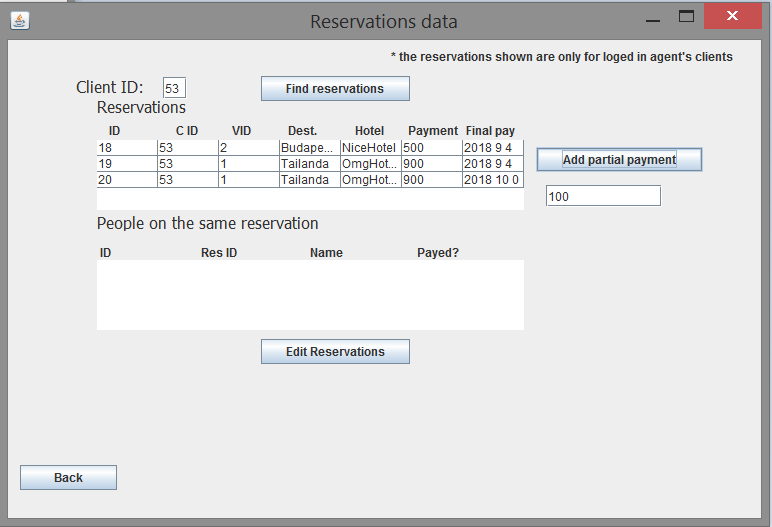
**6.9. Seeing reservations for a client id**

****

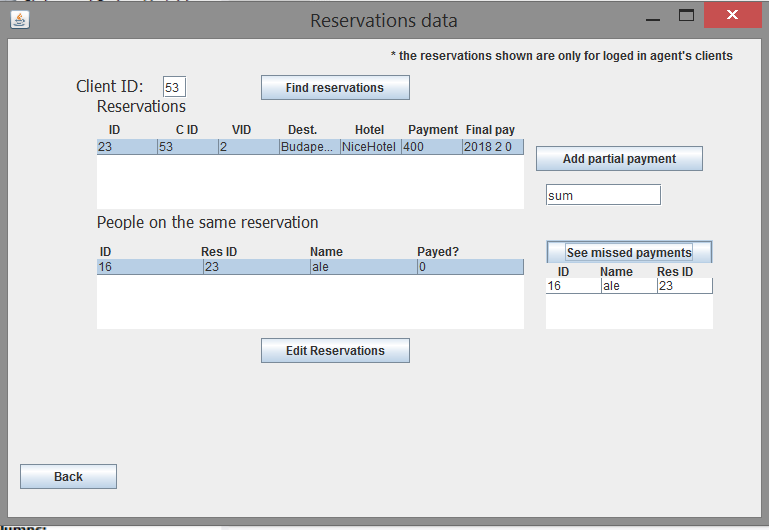
**6.10. Adding a reservation**

****

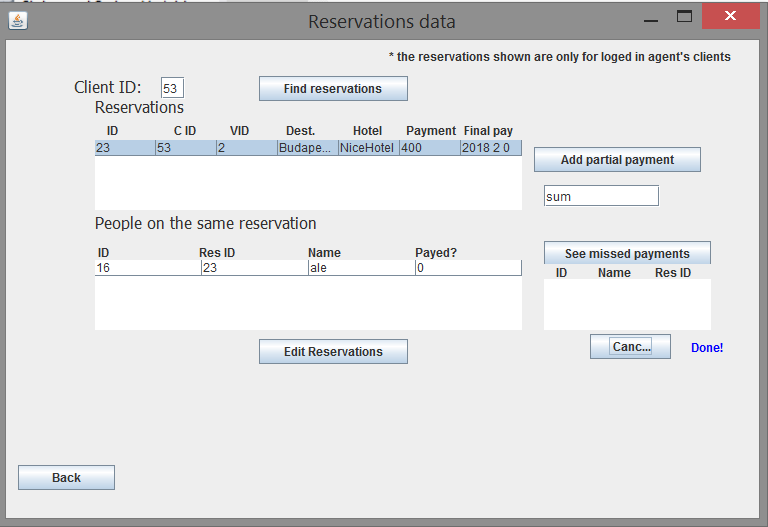
**6.11. Adding partial payment**

****

**6.12. Seeing if a person missed the payment date**

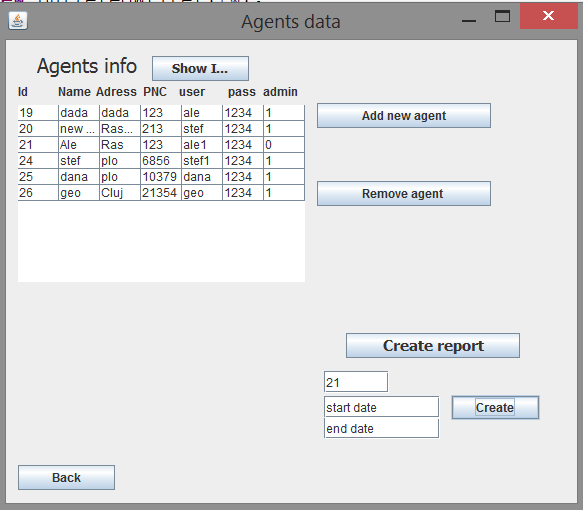
****

**6.13. Removing that person**

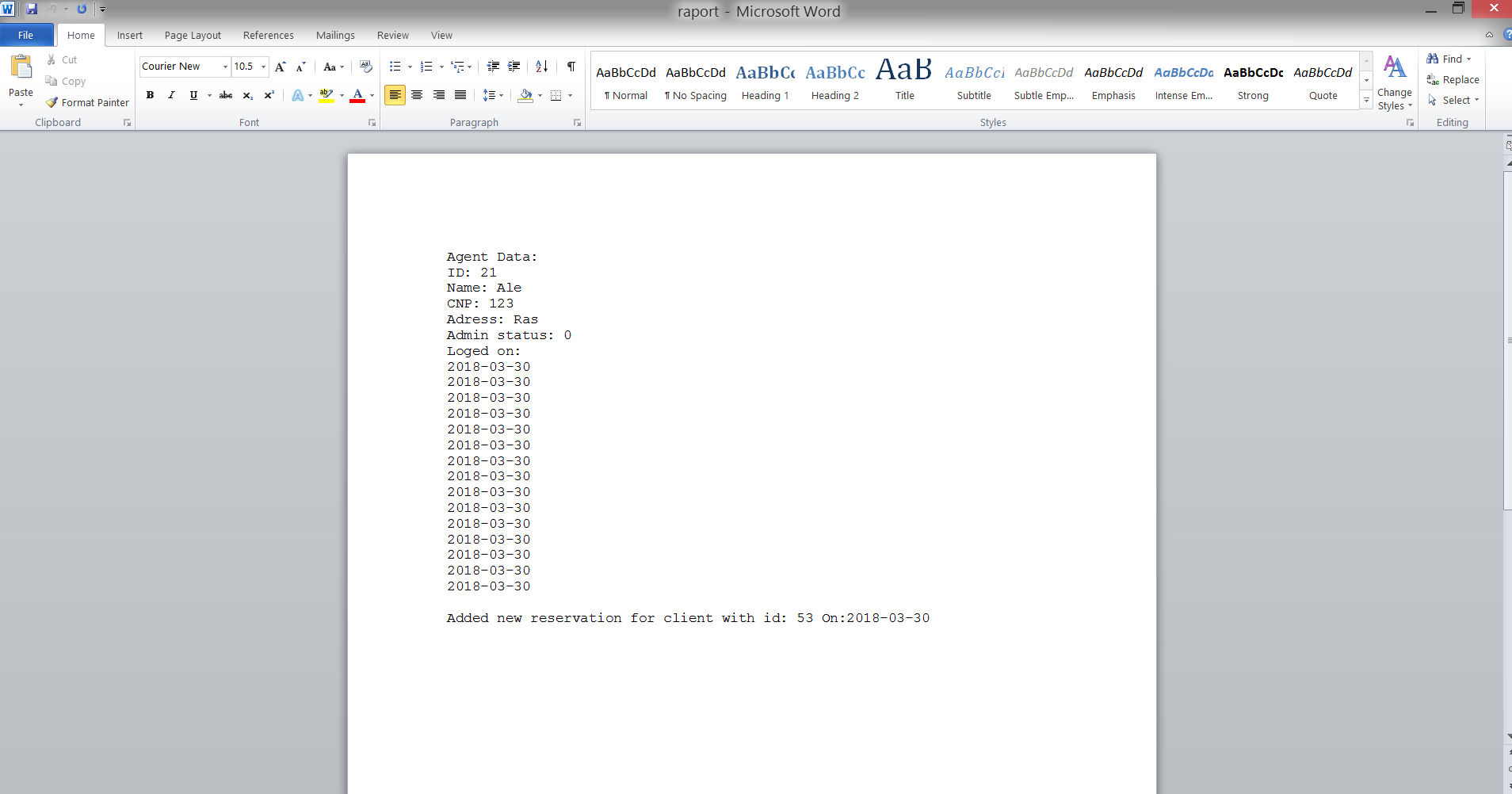


**6.14. Agent data**

# Agent data is handled the same as client data. Except for the fact that an administrator can create a report for a given agent id.



The report:



7. Bibliography

* <https://stackoverflow.com/>
* <https://www.lucidchart.com>
* <https://sourceforge.net/projects/itext/?source=typ_redirect>
* <http://richard.jp.leguen.ca/tutoring/>
* https://www.youtube.com/