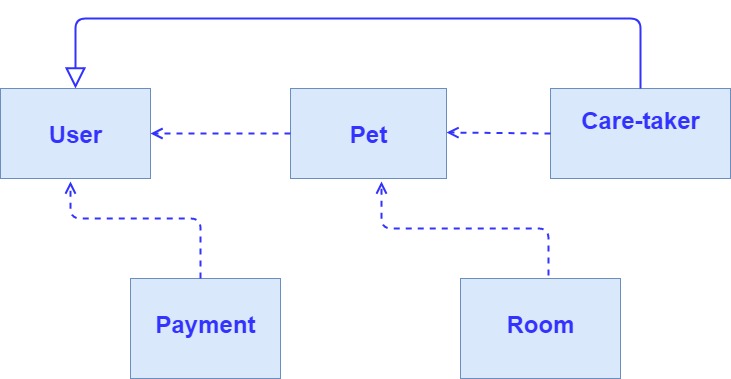
Pet Hotel

Analysis and Design Document

# Elaboration – Iteration 1.1

# Domain Model

The domain model will consist of 5 main classes: User(there will be 2 types of users), Pet, Payment, Room and Care-taker(extension of the User class). The care-taker will have some common attributes and methods with the user, but what will be different is that he will also have a list of pets that he’s in charge of.



# Architectural Design

## Conceptual Architecture

The **architecture** that I will be using for this web application is the **Layered Architecture**, consisting of the following layers:

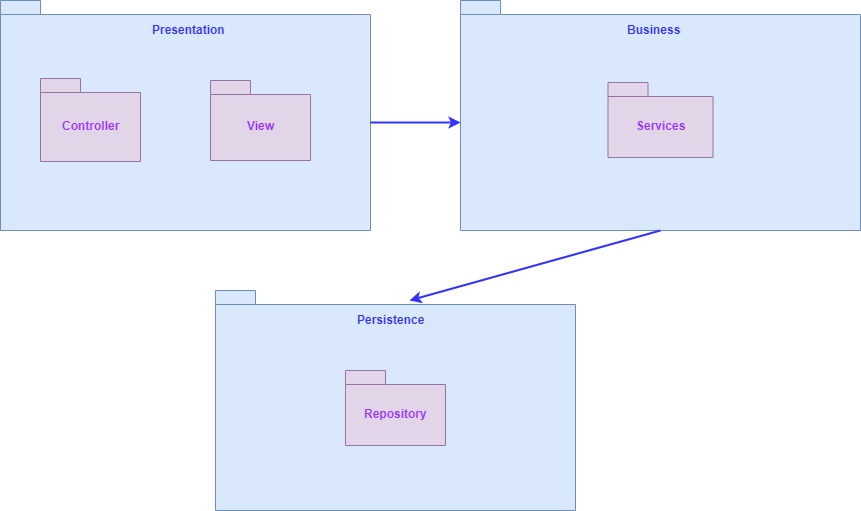
* **Presentation layer** – contains classes responsible for the User Interface
* **Business Layer** – contains rules that determine how data can be created, stored and changed
* **Persistence Layer –** deals with persisting(storing and retrieving) data from a database
* **Database Layer –** provides access to data stored in database

I have chosen this architecture because the web application that I am going to develop can be distributed into layers as described above. In this way, the dependency between classes is at minimum level, which means changes are easily done in a class without needing too much extra work in other classes.

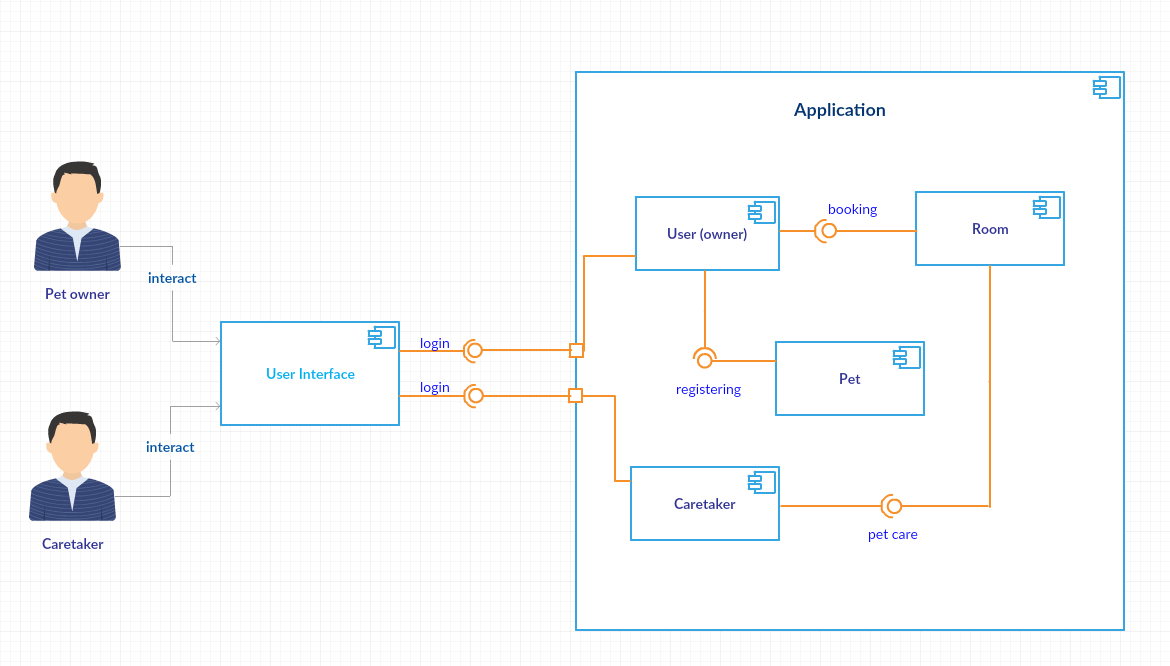
The **design pattern** that I am going to use is the MVC design pattern: Model, View and Controller. The main advantages for using this pattern are:

* Rapid application development
* Modification does not affect the entire model
* Ability to provide multiple levels

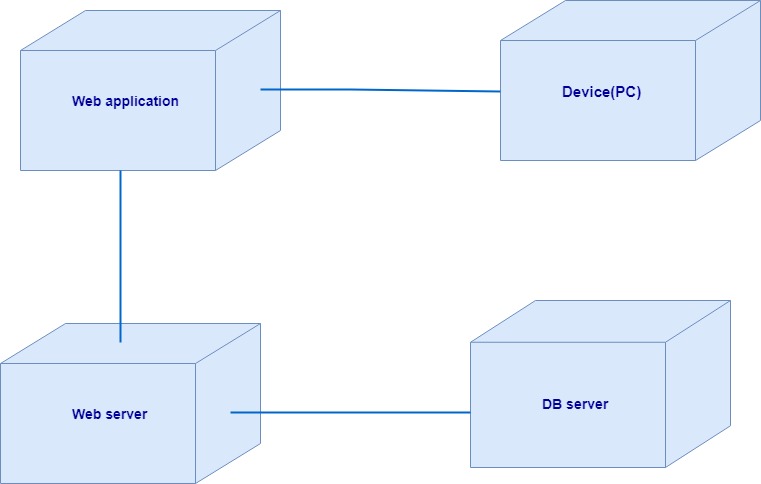
## Package Design



## Component and Deployment Diagrams

Component diagram:

Deployment diagram:

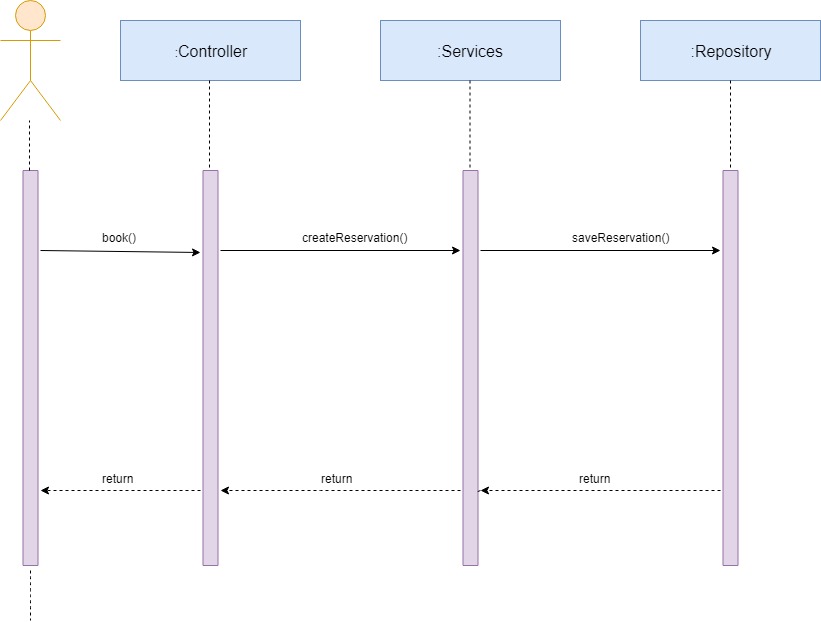


# Elaboration – Iteration 1.2

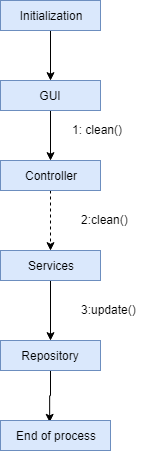
# Design Model

## Dynamic Behavior

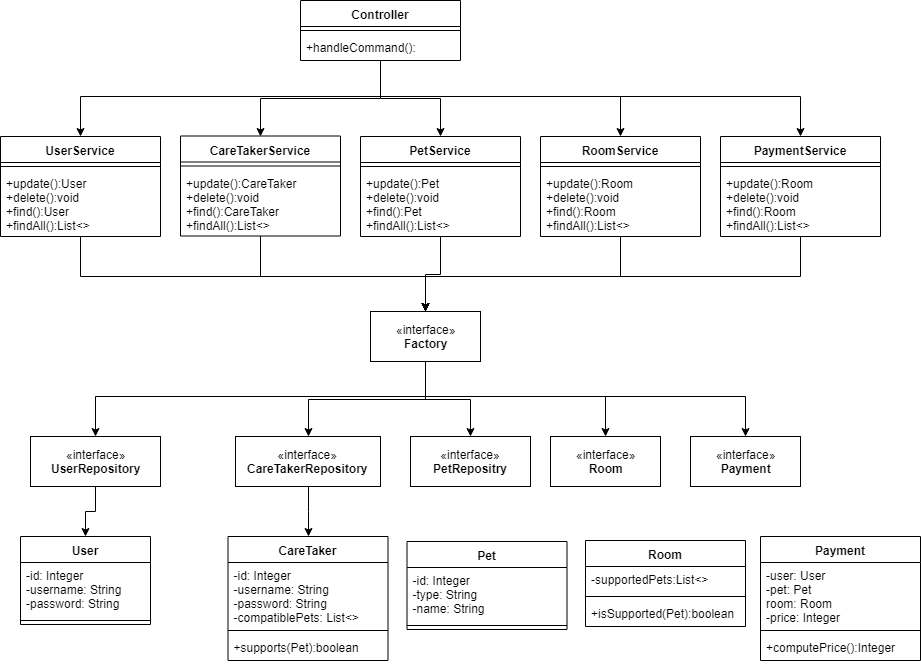
Sequence Diagram - Make Reservation Scenario :



Communication Diagram – Clean Room Scenario:

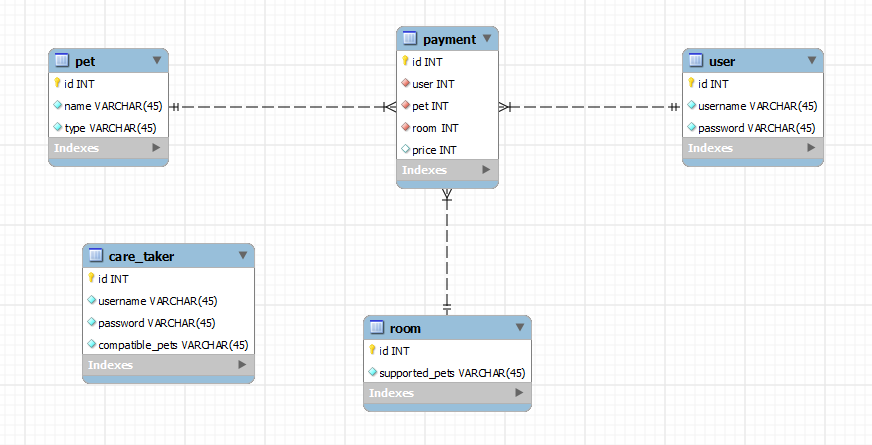


## Class Design



I chose to implement the Abstract Factory design pattern because it allows the project to follow the SOLID principles more closely. Also, it allows a more decoupled and more testable design.

# Data Model



# Test Strategy

**Unit tests** – tests done for the service layer (making a reservation, cleaning a room, registering, paying)

**GUI tests** – tests done for the graphical user interface (checking the screens with the controls like menus, buttons, icons and bars: menu bar, dialog boxes)

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

*[Present future improvements for the system]*