YOUR PET EVERYWHERE

Analysis and Design Document

Student: Vadean Andrada Anastasia

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <04/Apr/19> | <1.0.> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

# I want to make an application which monitors your pet. There are so many people which has pets and it so hard to keep in mind everything about them. This application will manage this for them. It will be two types of users: administrator user and regular user.

# The user will be able to make an account for himself and after that he will have to do the same thing for his pats, if he has more than one. Every pet account will store different kinds of information (like when was its last medical appointment or when was its last cut). Depending on each animal breed the application will give information about the pet’s needs. Also, if you don’t have a pet and you wish one, the user will be able to make a profile for himself where he will describe himself and the application will recommend an animal breed.

# Elaboration – Iteration 1.1

# Domain Model

A domain model is generally implemented as an object model within a layer that uses a lower-level layer for persistence and publishes an API higher-level layer to gain access to the data and behavior of the model.

My domain model will contain the classes: User, Pet, Description, Appointment, Staff.

An user will be able to make an account for his pets. He can have one or more accounts, or he can have no one. Maybe he is an animal lover and he had no one, so he can make a Profile and he will receive a breed suggestion. If the user has an account for a pet, there he will receive notification for his pet appointments, or he can go shopping something for his pet.

In the Unified Model Language (UML), a class diagram is used to represent the domain model.

# Architectural Design

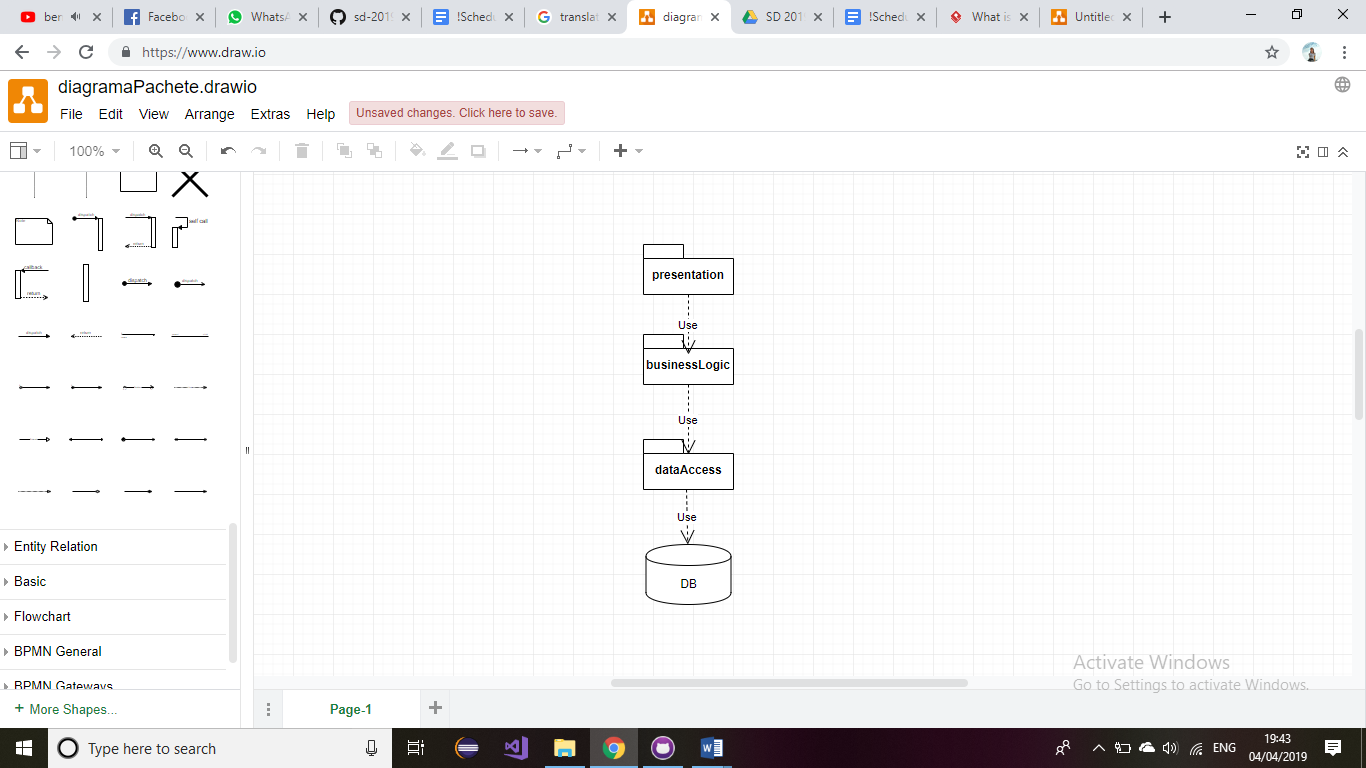
## Conceptual Architecture

An architectural pattern expresses a fundamental structural organization schema for software systems.

The system will be organized using Layers. Firstly, I will create a database, using MySQL Workbench. Then, I will split my work into three packages. In dataAcces package I will have the connection with the database and all the classes that helps to manage this connection. In businessLogic package will be all the “magic”, here I will implement all the operation that will ensure a good functionality for the application. Presentation package will manage the interaction between user and the application.

I want to choose the Layers Pattern for a good organization and an organized structure of the project.

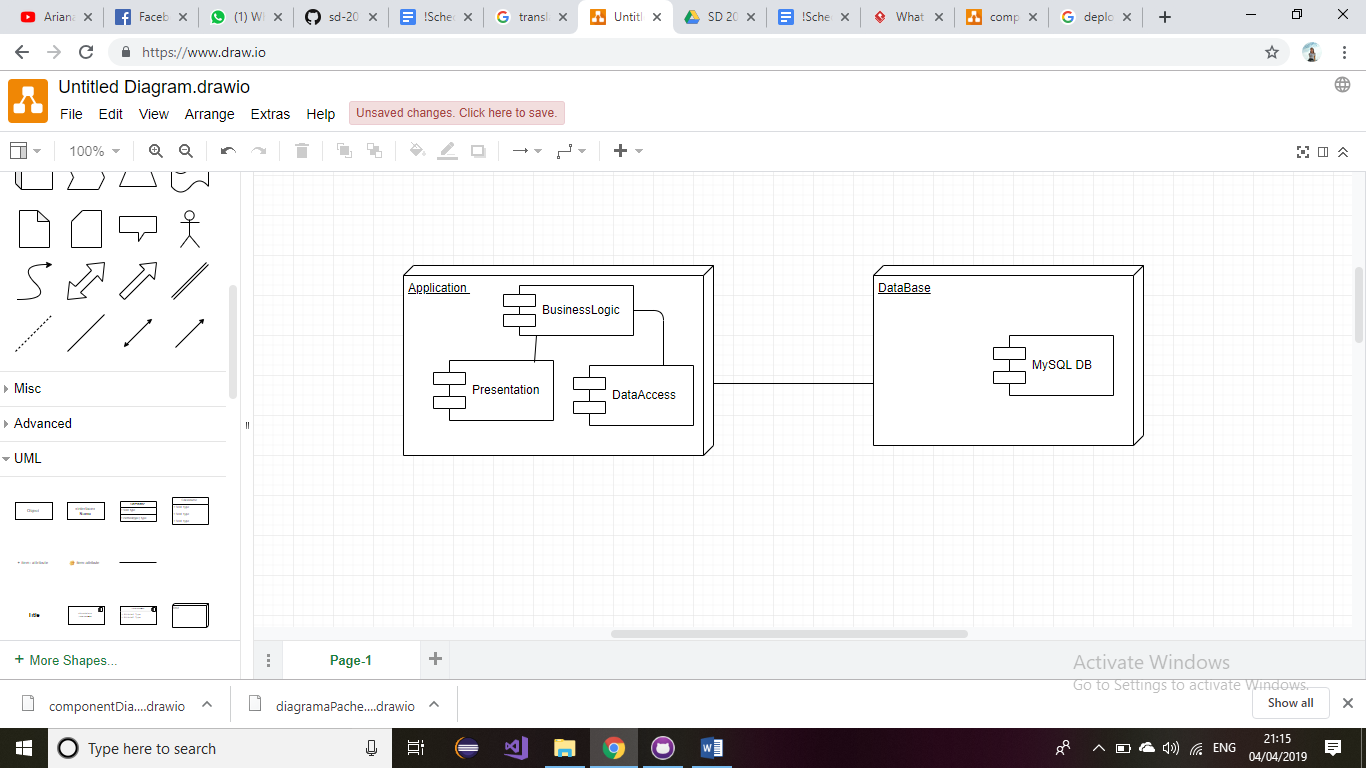
## Package Design



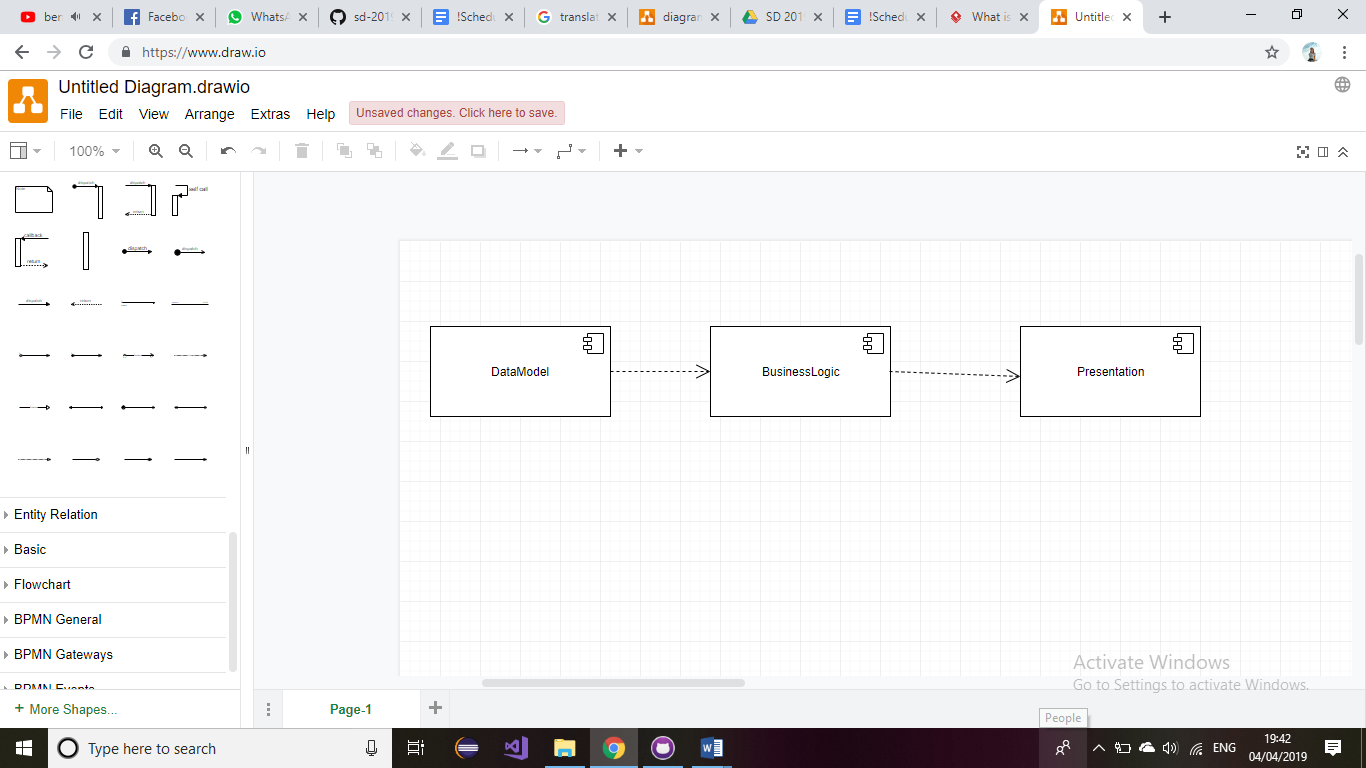
## Component and Deployment Diagrams

*[Create the component and deployment diagrams.]*

Deployment diagram:



Component diagram:



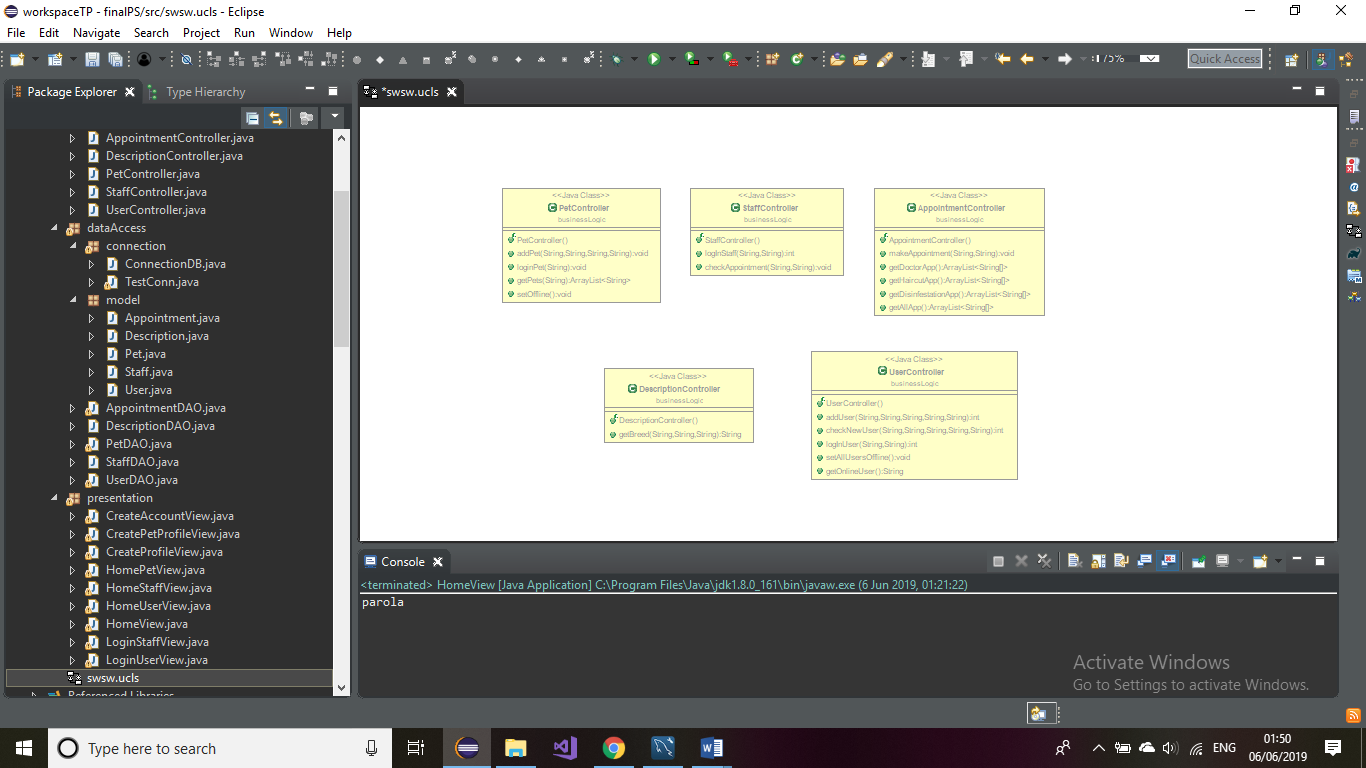
# Elaboration – Iteration 1.2

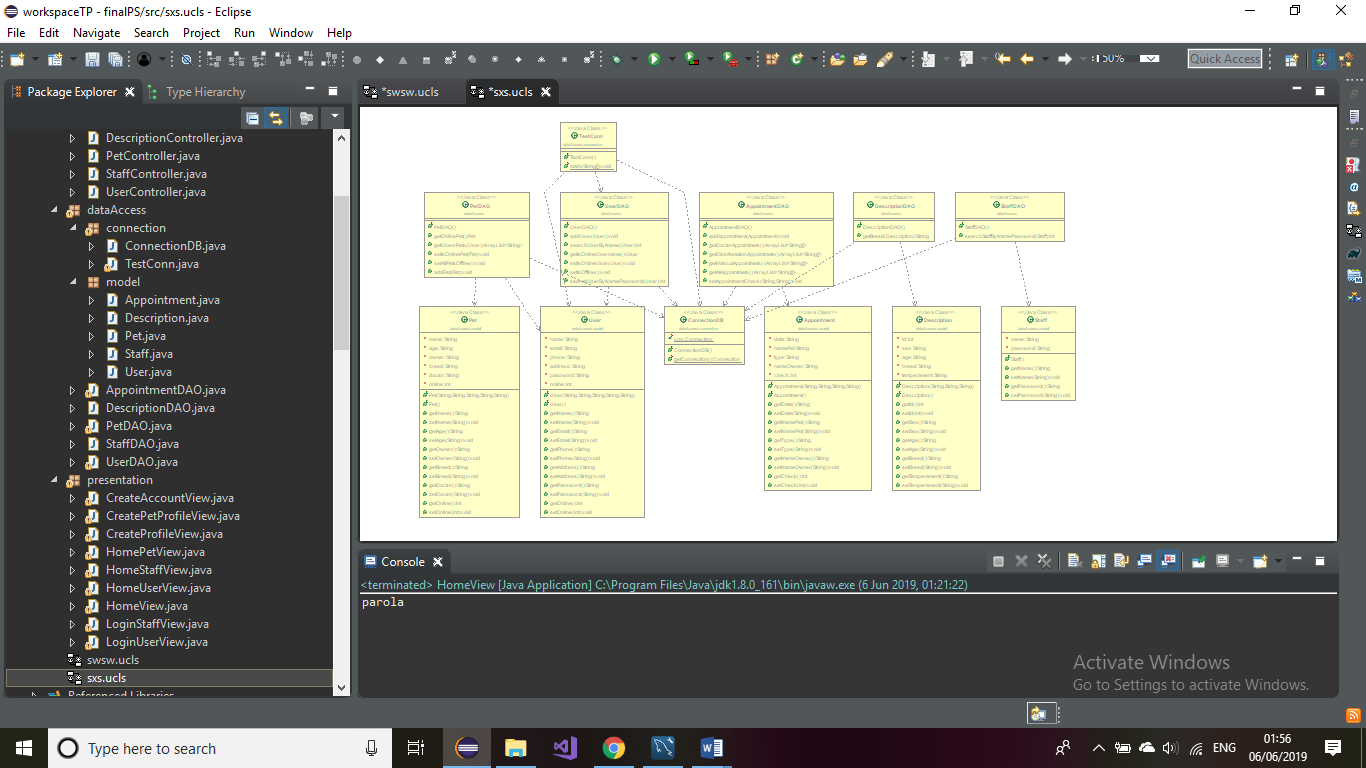
# Design Model

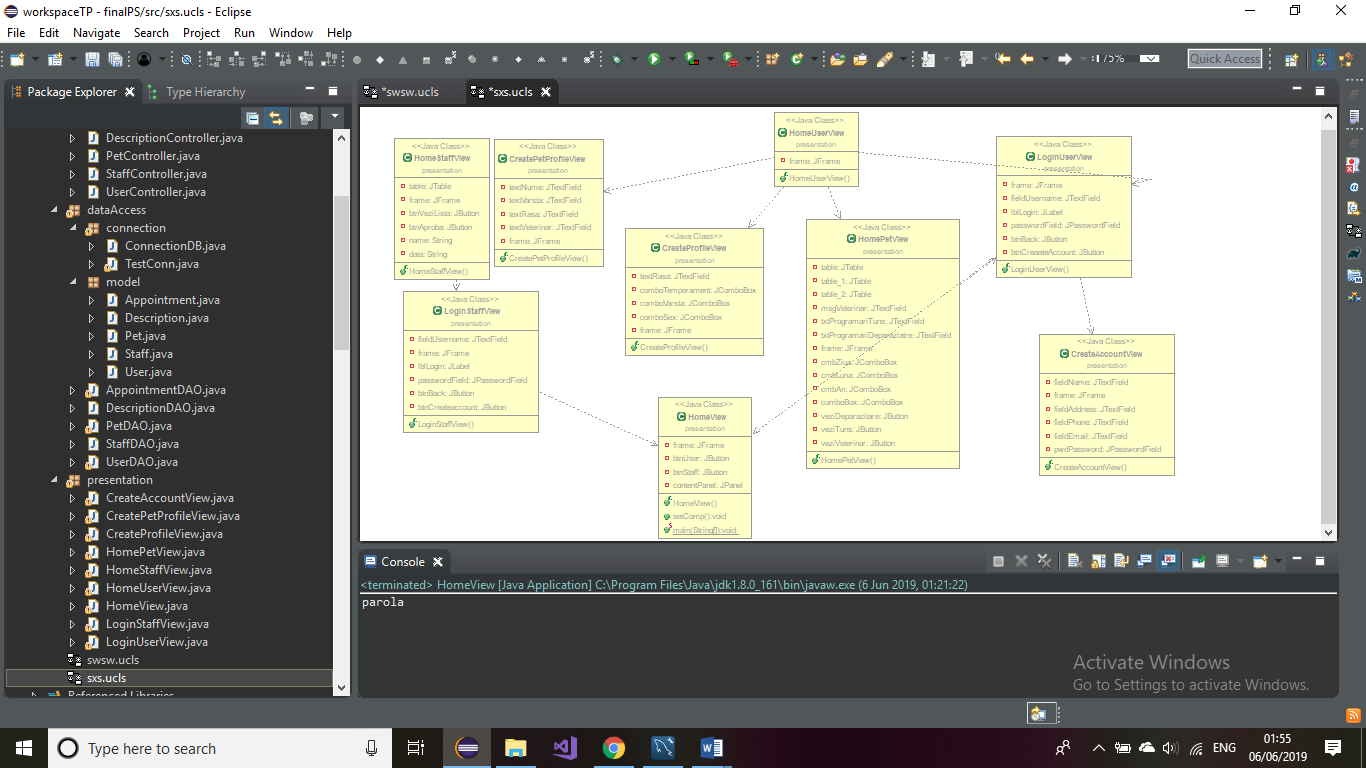
## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

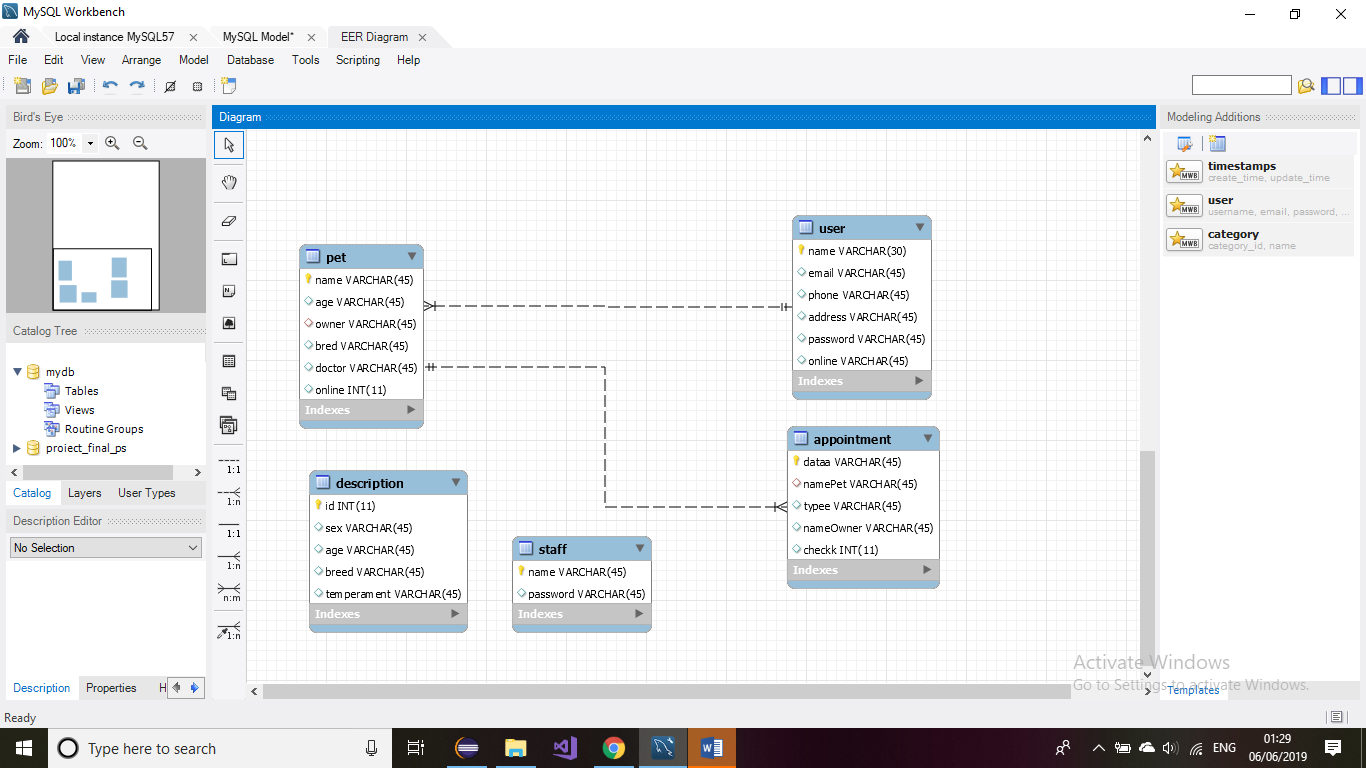
## Class Design







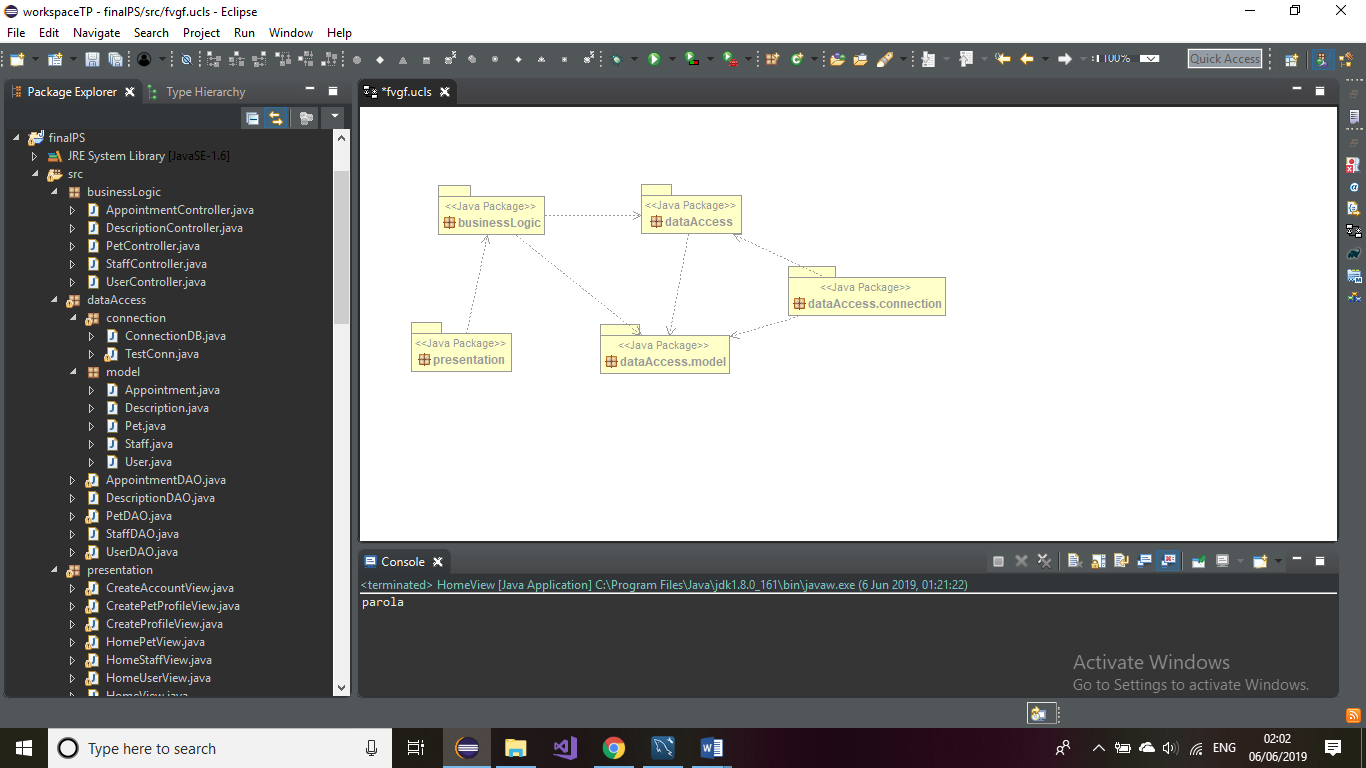
# Data Model



# Unit Testing

# Elaboration – Iteration 2

# Architectural Design Refinement



# Design Model Refinement

# Construction and Transition

# System Testing

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

In this part, the integration testing will contain test case scenarios necessary to verify the good functionality of the program, like the connection with database important to the work of all the functionalities of this application.

# Future improvements

*[Present future improvements for the system]*

It is a large domain, so we can have so many improvements. I will mention some of them:

- a chat window where the pet’s possessor can chat

- a gallery where the user could store the pictures with his pets

- a map, so the user can see if he had friend in his area

- option for make appointments for different pet staff.

# Bibliography