Art Gallery Management

Analysis and Design Document

Student:Pop Laura-Maria

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <05/APR/2017> | <1.0> | - | Pop Laura-Maria |
| <04/MAY/2017> | <1.1> | Sequence Diagram, DB Diagram | Pop Laura-Maria |
|  |  |  |  |
|  |  |  |  |

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

*AGM will be an online Art Gallery Management application. It will have two types of user:*

1. *Regular user*
2. *Admin*

*Through a Data Base, the application will keep information about:*

1. *Artists:*
   1. *Name*
   2. *Birthplace*
   3. *Age*
   4. *Style of art*
2. *Art Work:*
   1. *Artist*
   2. *Year it was made*
   3. *Unique title*
   4. *Type of art*
   5. *Price*
3. *Type of Art:*
   1. *It can be various (like renaissance or work of the 19th century, etc.)*
4. *Customers:*
   1. *Name*
   2. *Address*
   3. *Email*
   4. *Phone*
   5. *Maybe a short description for their profile, but it is not obligatory*
   6. *Likes of arts*
   7. *Total amount they spent on Gallery*
5. *Administrators:*
   1. *Name*
   2. *Address*
   3. *Email*
   4. *Phone*
   5. *Etc.*

*The regular user can perform the following operations:*

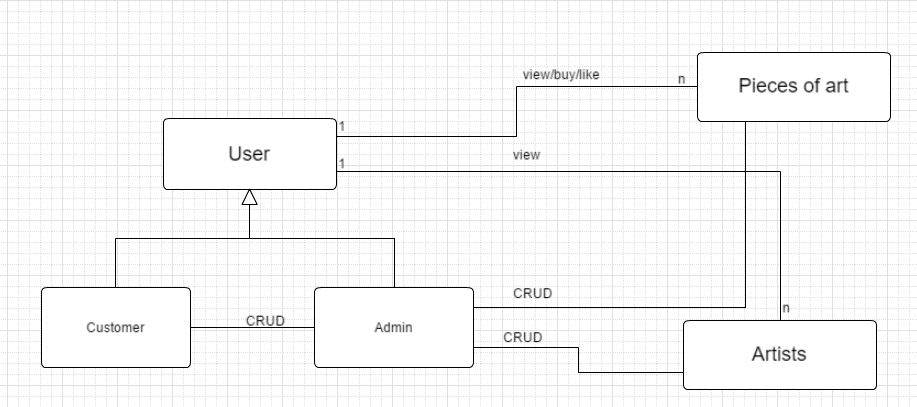
1. *To like a piece of art*
2. *View art*
3. *Buy art*
4. *Register on the site*
5. *Log-in/Log-out*

*The Admin can perform the following operations:*

1. *All operation that a customer can do except Register*
2. *CRUD on artists*
3. *CRUD on customers*
4. *CRUD on pieces of art*
5. *Generate reports*

# Elaboration – Iteration 1.1

# Domain Model

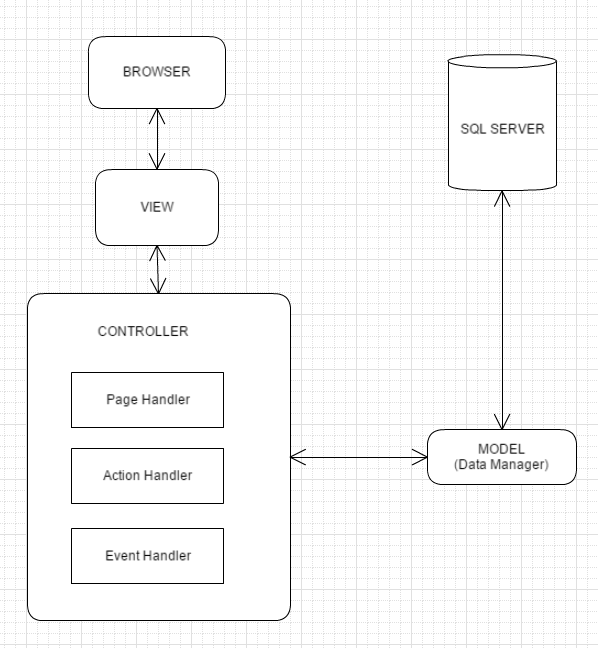
**

# Architectural Design

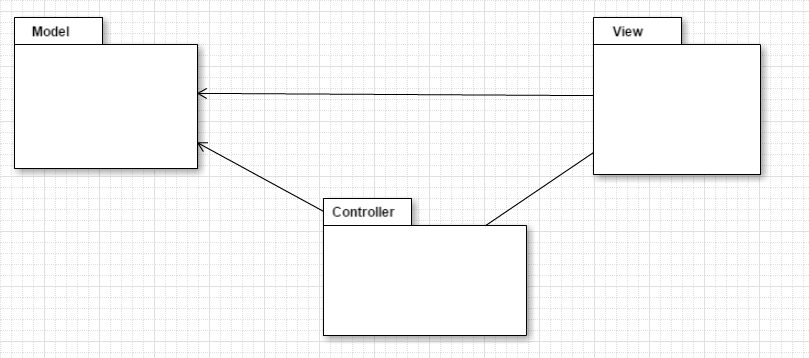
## Conceptual Architecture

*I will use a MVC (Model-View-Controller) because is perfect considering that the application will be online. A lot of frameworks work with this MVC pattern.*

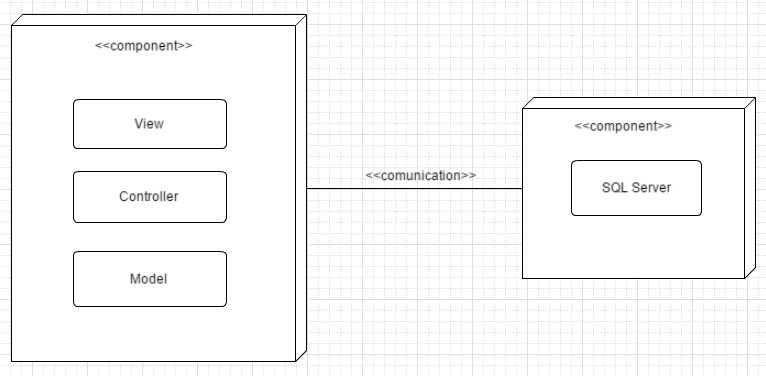
*It will be easy to implement a MVC pattern and it will be useful because the application operations had to be very clear.*

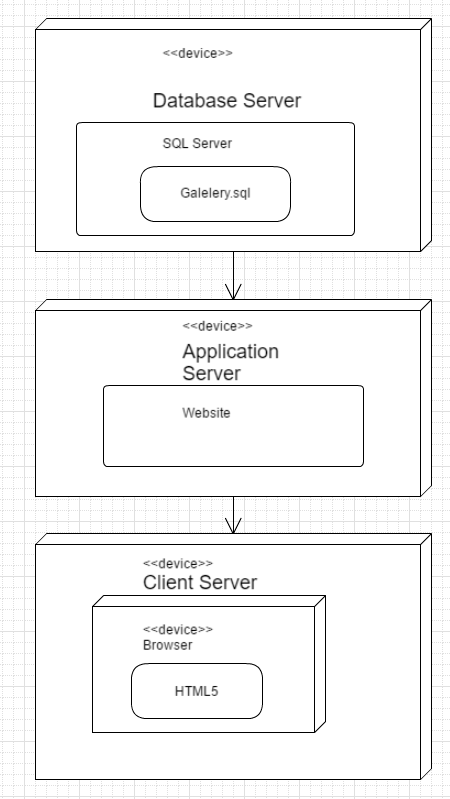
**

## Package Design

**

## Component and Deployment Diagrams

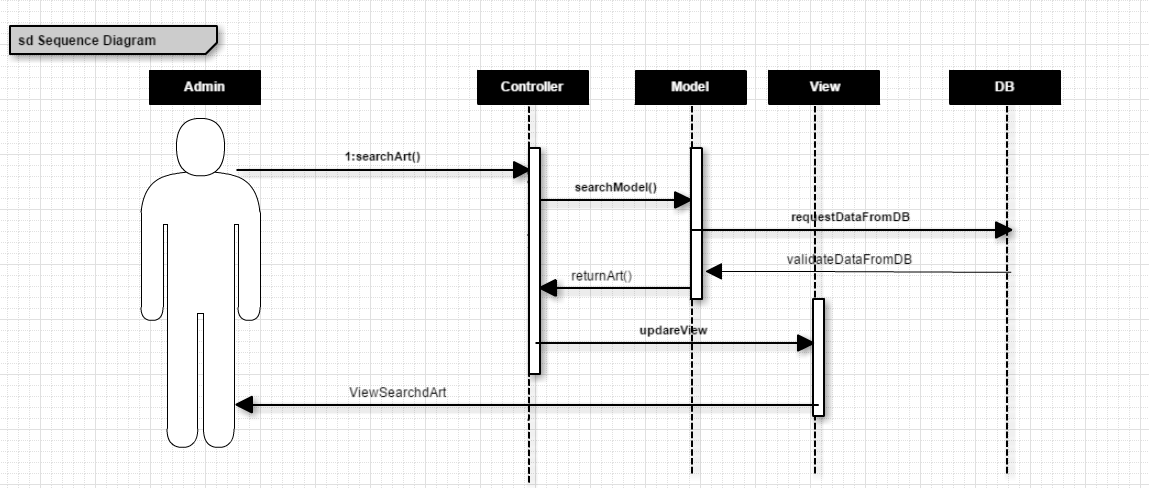




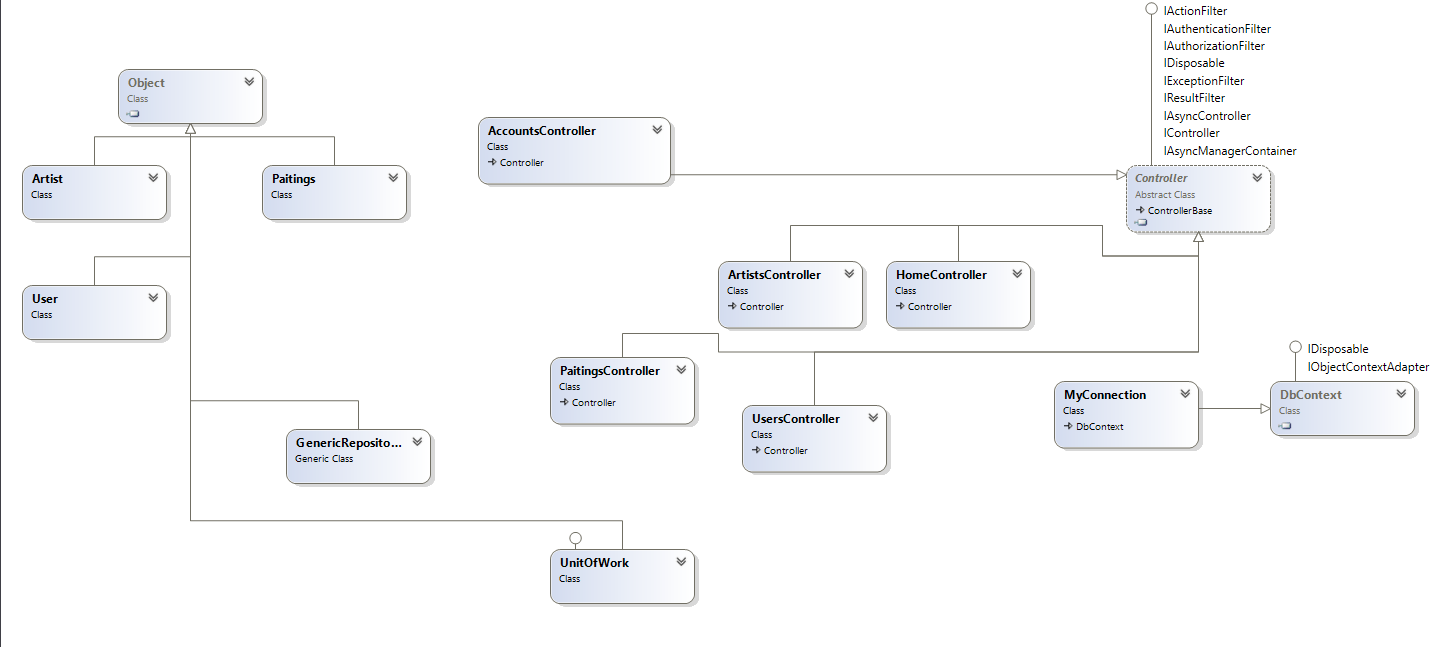
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior



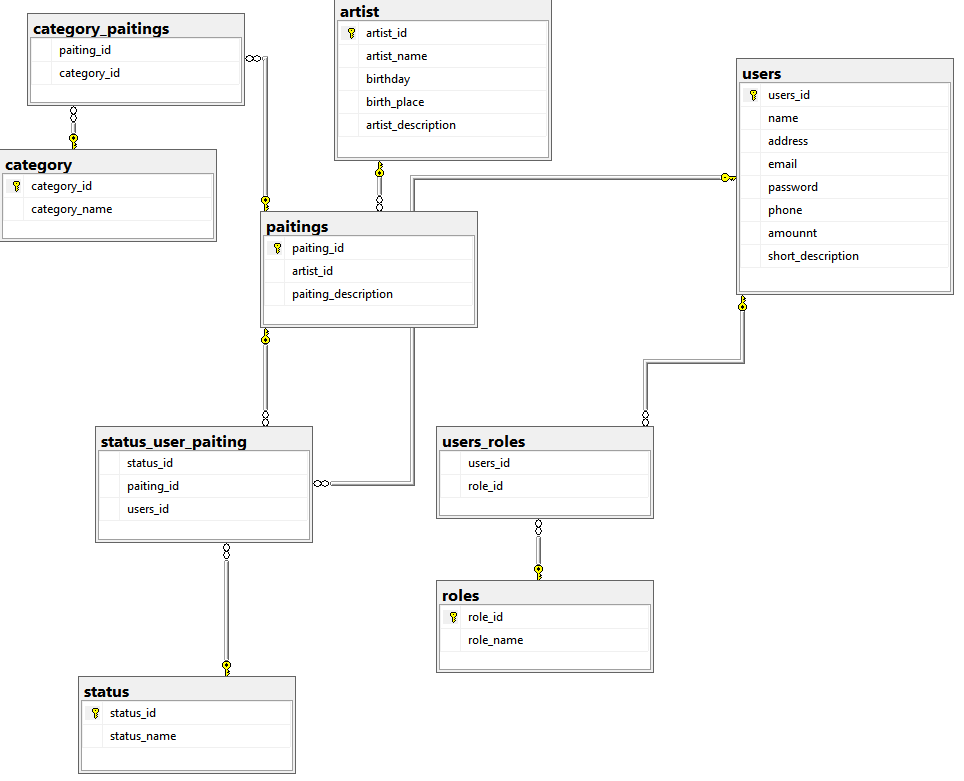
## Class Design

**

I used a Repository pattern because it creates an abstraction layer between the data access layer and the business logic layer and it helps to isolate the application from changes in the data store.

Also it prevents duplicated code.

# Data Model

**

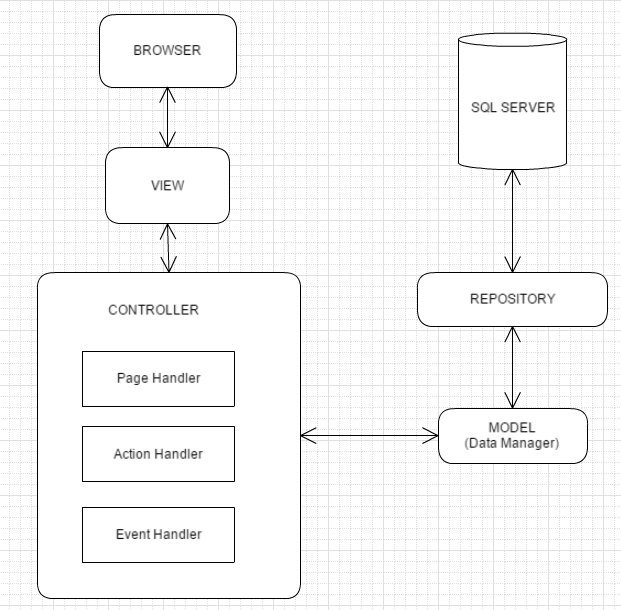
# Unit Testing

For the testing, I used manual testing. I was testing every URL path that was a possible danger for the program. For example I tested if a regular user can access admin panel, for example.

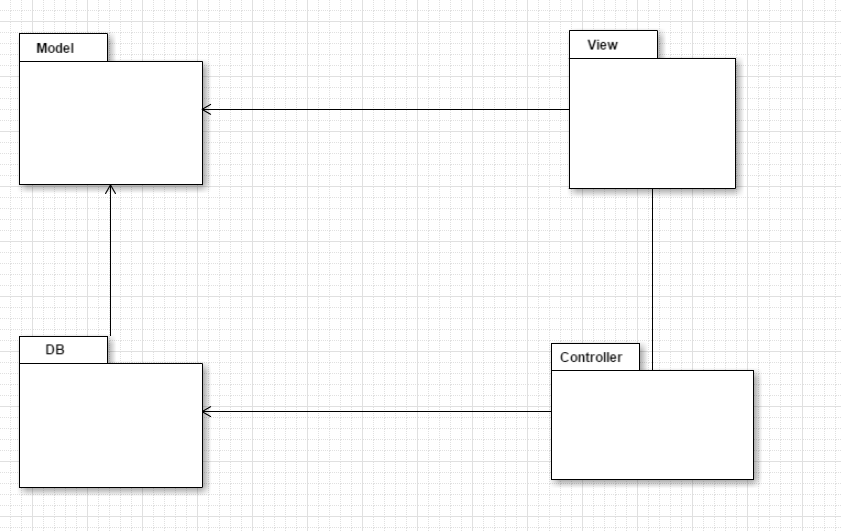
# Elaboration – Iteration 2

# Architectural Design Refinement

Package diagram changed a little bit because I created a new package for Repository pattern and database sets.



And package diagram:

******

# Construction and Transition

# System Testing

I was testing it like before with manual testing and make sure that the path can’t be access by anyone.

# Future improvements

This project is not finished so a first improvement will be exactly this. Another improvement will be email service. For example, if a user makes a new account, hi will be notify via email. Also, checking password can be added and maybe an option for user to personalize his page (like a theme) and add new items to user profile, like photos.

# Bibliography

[1] <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>

[2] <https://www.youtube.com/watch?v=Uq0y8oxnx-8&t=1234s>

[3] <https://www.youtube.com/watch?v=rtXpYpZdOzM&t=1373s>

[4] <https://www.youtube.com/watch?v=E7Voso411Vs&t=1505s>

[5] <https://mva.microsoft.com/en-us/training-courses/introduction-to-asp-net-mvc-8322?l=nKZwZ8Zy_3504984382>

[6] <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started>

[7] <https://www.w3schools.com/html/>

[8] <https://www.w3schools.com/css/css_intro.asp>

[9] <https://www.w3schools.com/js/default.asp>

[10] <https://www.tutorialspoint.com/ms_sql_server/index.htm>