Art Collection and Discovery

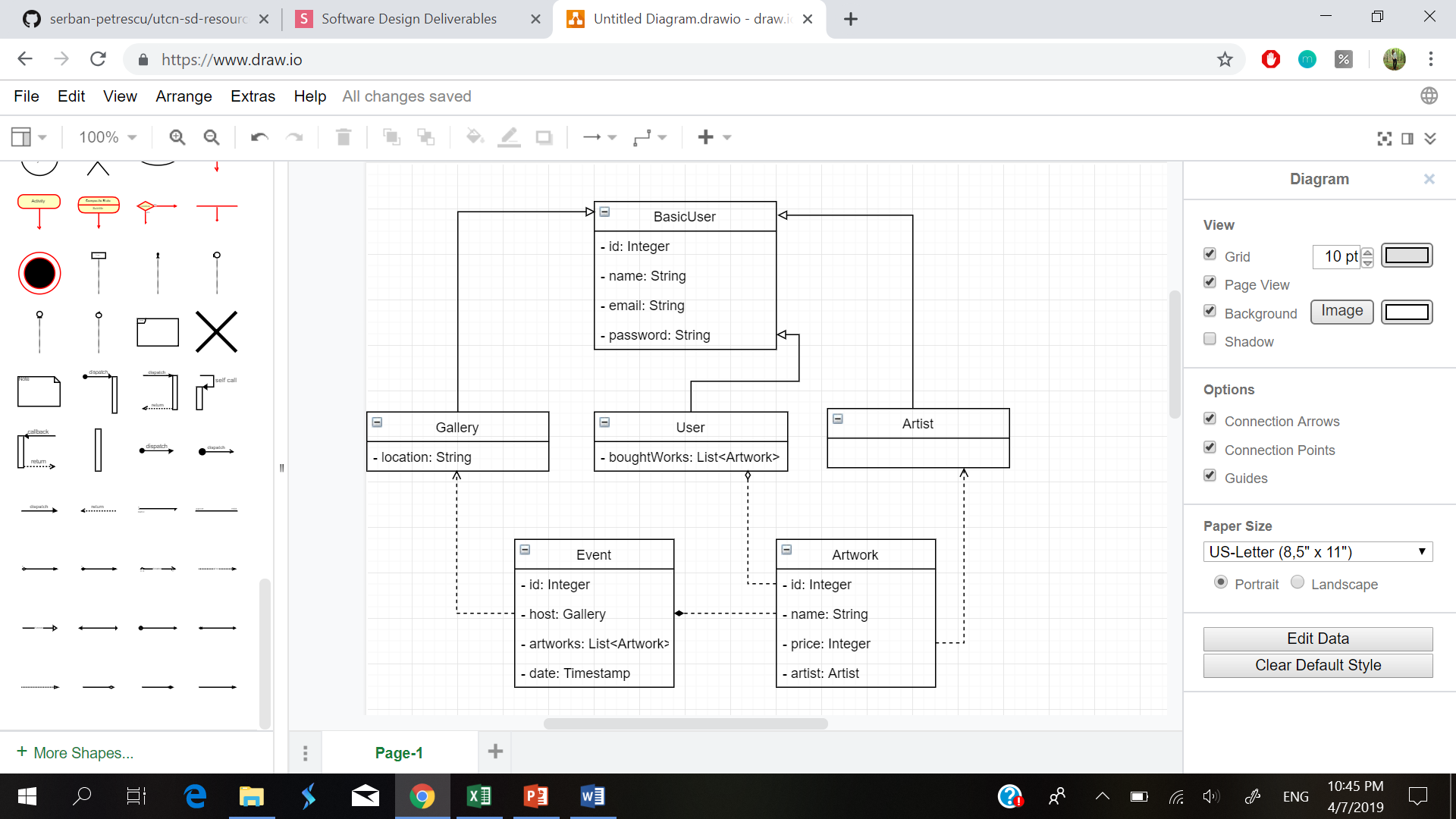
Analysis and Design Document

# Elaboration – Iteration 1.1

# Domain Model

Since the app is designed for artists, galleries, as well as art enthusiasts, the following entities must be present:

* Artist
* Artwork
* User
* Gallery
* Event

Since the artist, the buyer, and the gallery all have names and credentials, they can inherit those features from a basic user, and add their specific fields (and later have specific methods associated to each one of them).

# Architectural Design

## Conceptual Architecture

The architecture is going to be multi-layered. It is a simple architecture, which we have worked with before, easy to understand and use, and it offers a clear separation based on roles: each layer handles different types of functionalities. The presentation layer contains the UI components, while the model includes the aforementioned classes. The service layer handles all operations that can be performed on and by the entities in the model. The DAOs, as their name suggests, are objects which access and retrieve the data, one object corresponding to each entity. Last but not least, there is a database, designed in MySQL, which will contain the tables with all the data needed by the application.

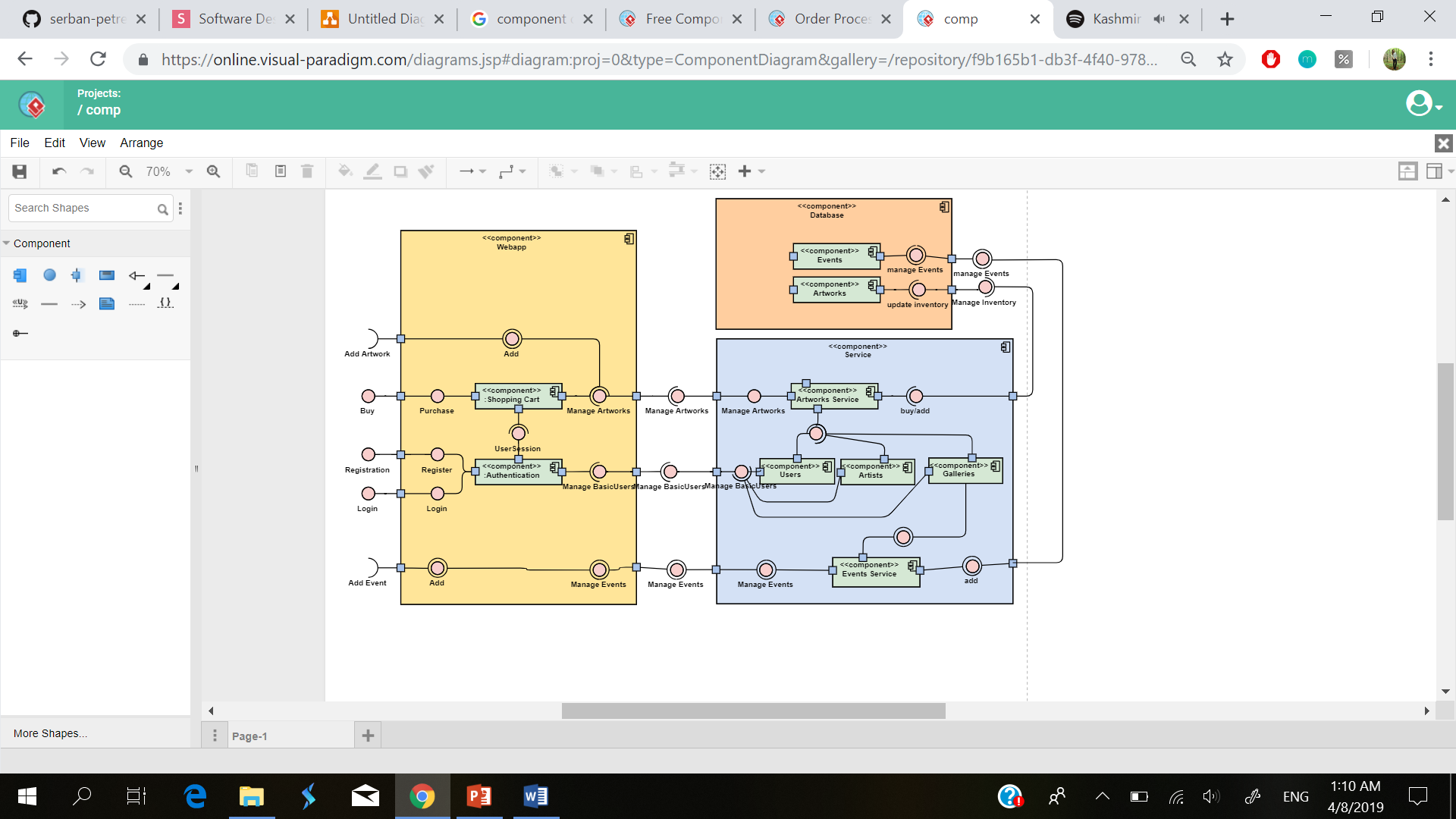
## Package Design

## 

## Component and Deployment Diagrams

# Deployment Diagram:

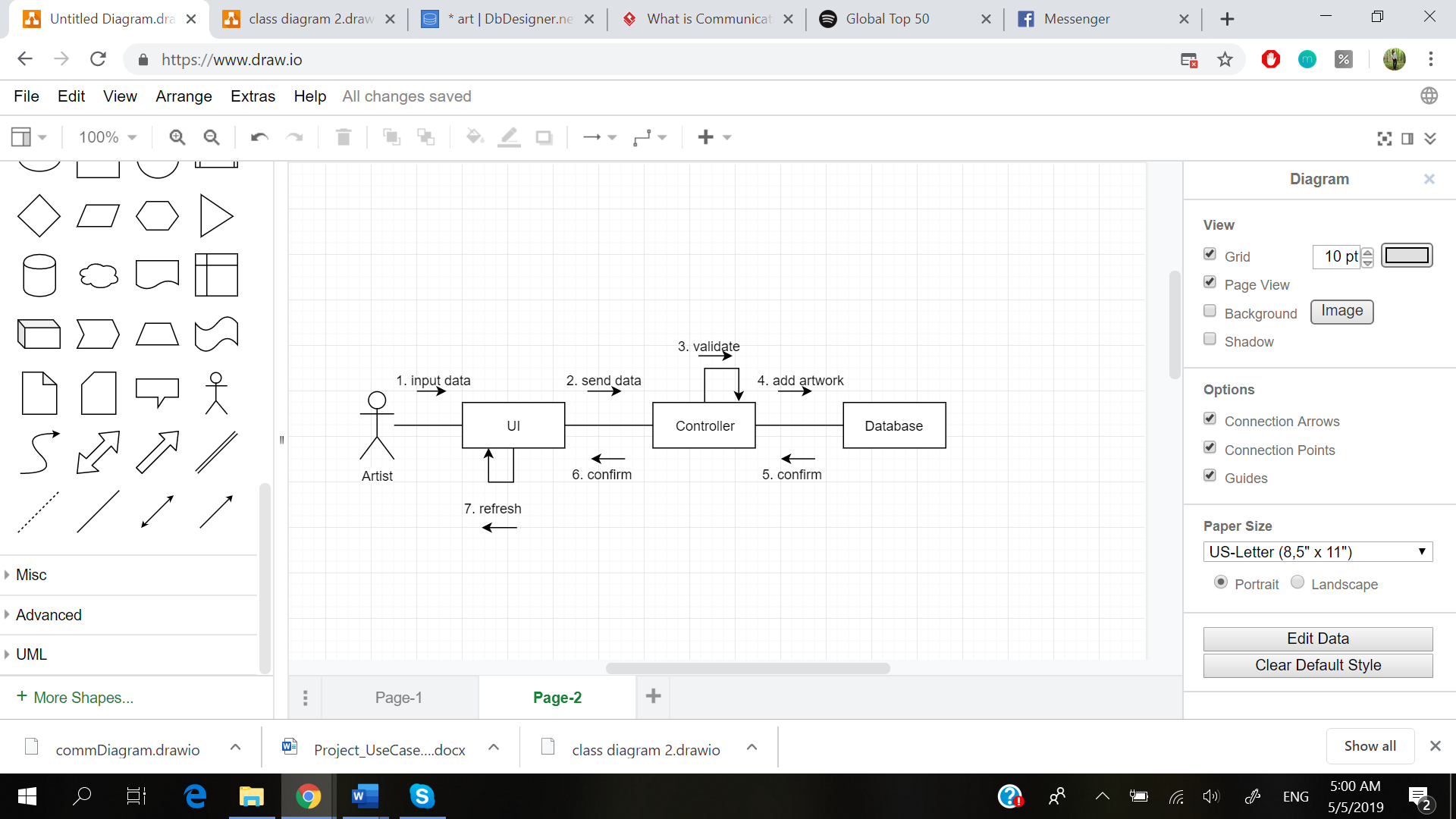
# 

 Component Diagram:

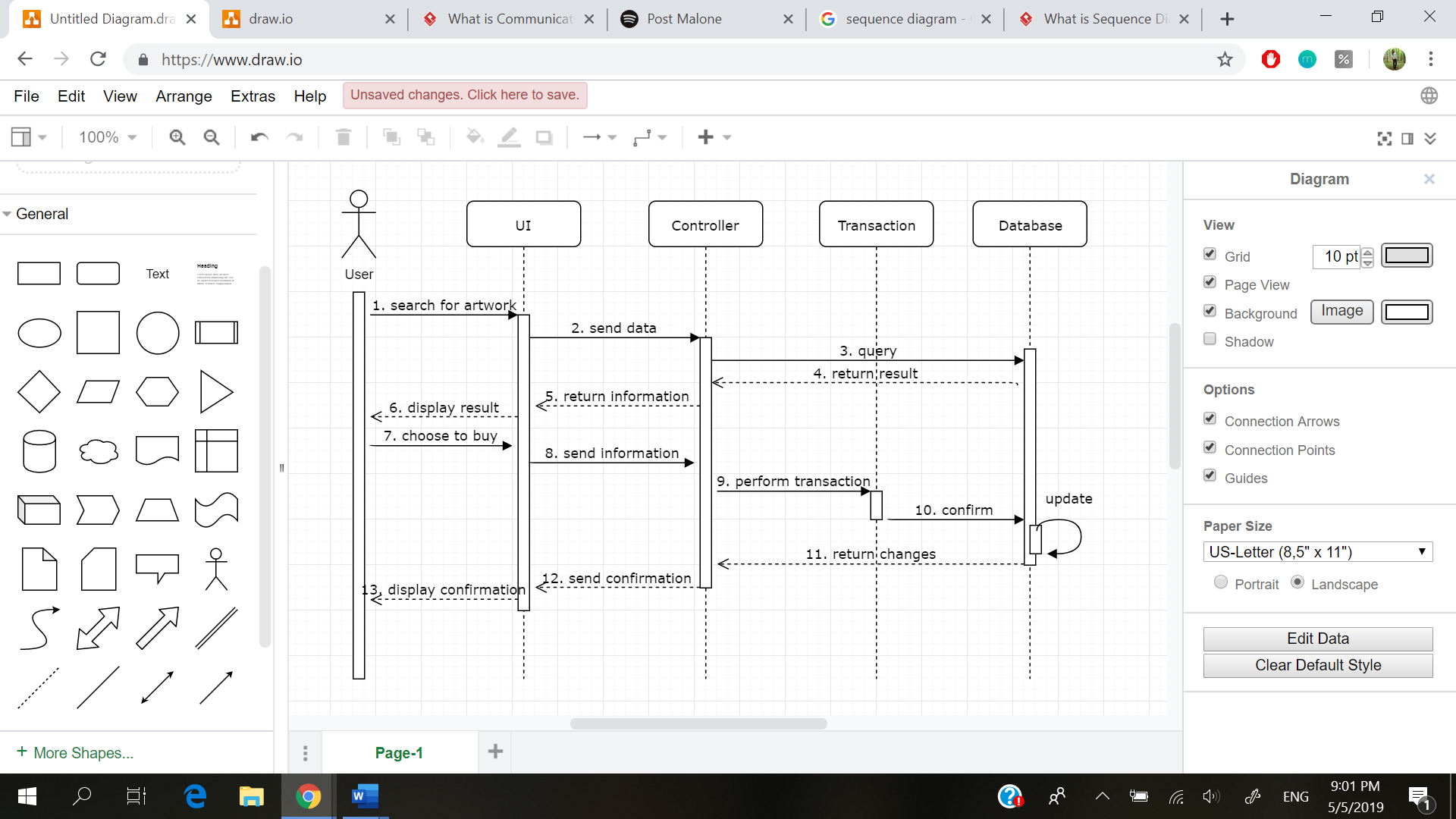
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior



Communication Diagram - Post an Artwork Scenario

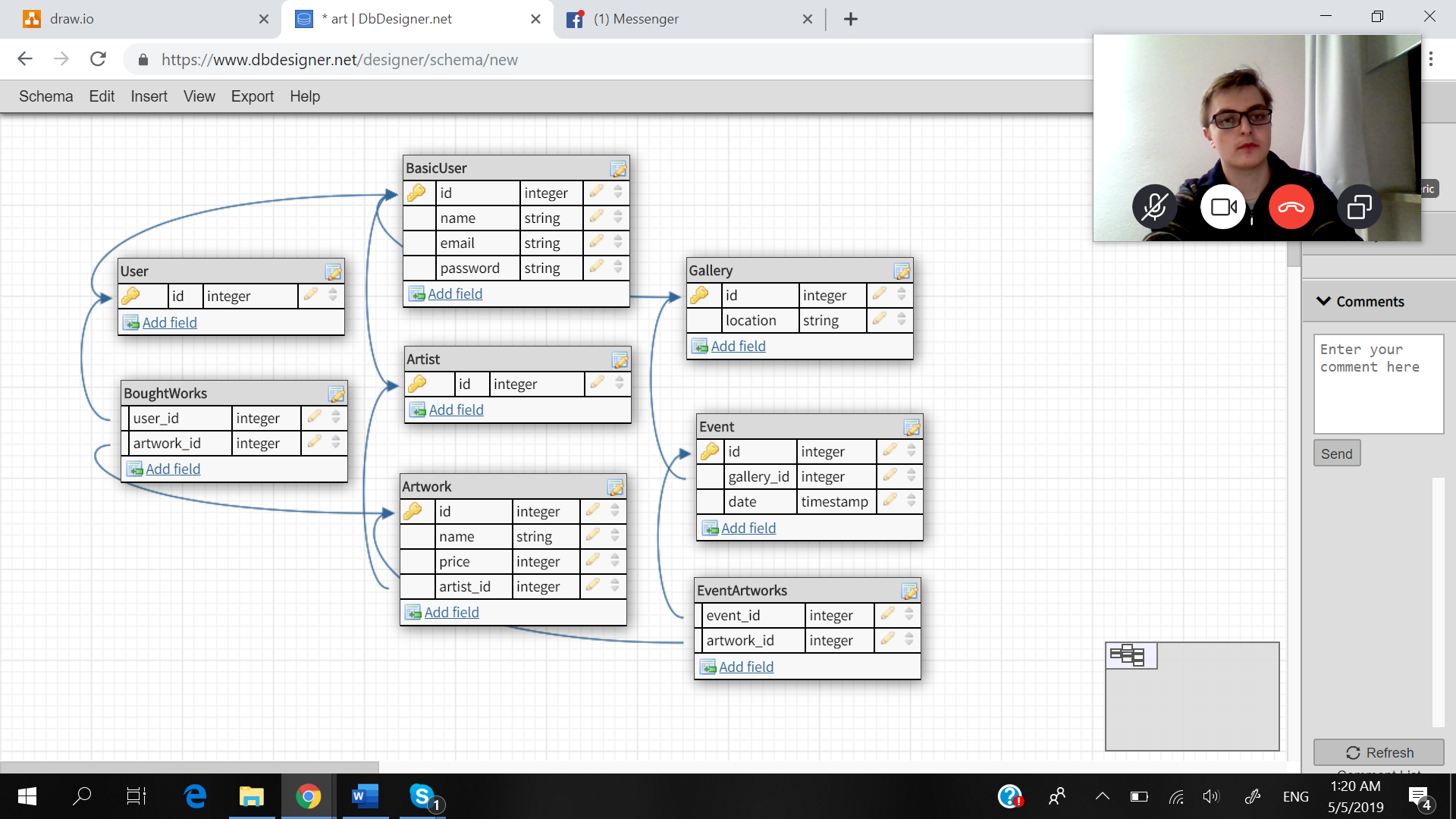


Sequence Diagram - Buy an Artwork Scenario

## Class Design

The most appropriate design pattern for this application is, in my opinion, the Observer pattern. This is due to the fact that an artist lists an artwork with the intent to sell it, and would want to be notified when this happens. Therefore, the artwork will extend the Observable class, and will notify the observer (the artist) whenever an artwork is sold.

# Data Model

*.*

# Test Strategy

The first step in testing the application consists of developer tests. These are tests performed “by hand”, to ensure that the added functionalities perform as desired. The main scenarios will be tested: adding an artwork as an artist, buying an artwork, creating an event or posting an article as a gallery. Also, the case in which the person performing the action changes their mind and cancels the operation will be performed as well, to make sure that this option is also available, that no residual information gets stored, and that they can return to their previous pages. Additional to this type of testing, unit tests can be introduced, ensuring that the basic functions responsible with the system’s functionalities behave as they should. What is more, end to end tests can be performed (especially since at the moment of testing the application we probably will have had a little bit of experience with them as well).

# 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]*