SINU Application

Analysis and Design Document

Student: Pantea Paul Alexandru

**Group: 30233**

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

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Design and implement a Java application for the management of students in the CS Department at UTCN. The application should have two types of users(student and teacher/administrator user) which have to provide a username and a password in order to use the application.

# Functional Requirements

The regular user can perform the following operation:

* Add/Update/View client information ( name, identity card number, personal numerical code, address, etc )
* Create/Update/Delete/View student profile ( account information: identification number, group, enrolments, grades).
* Process class enrolment ( enroll, exams, grades ).

The administrator user can perform the following operations:

* Create/Read/Update/Delete on students information
* Generate reports for a particular period containing the activities performed by a student

# Non-functional Requirements

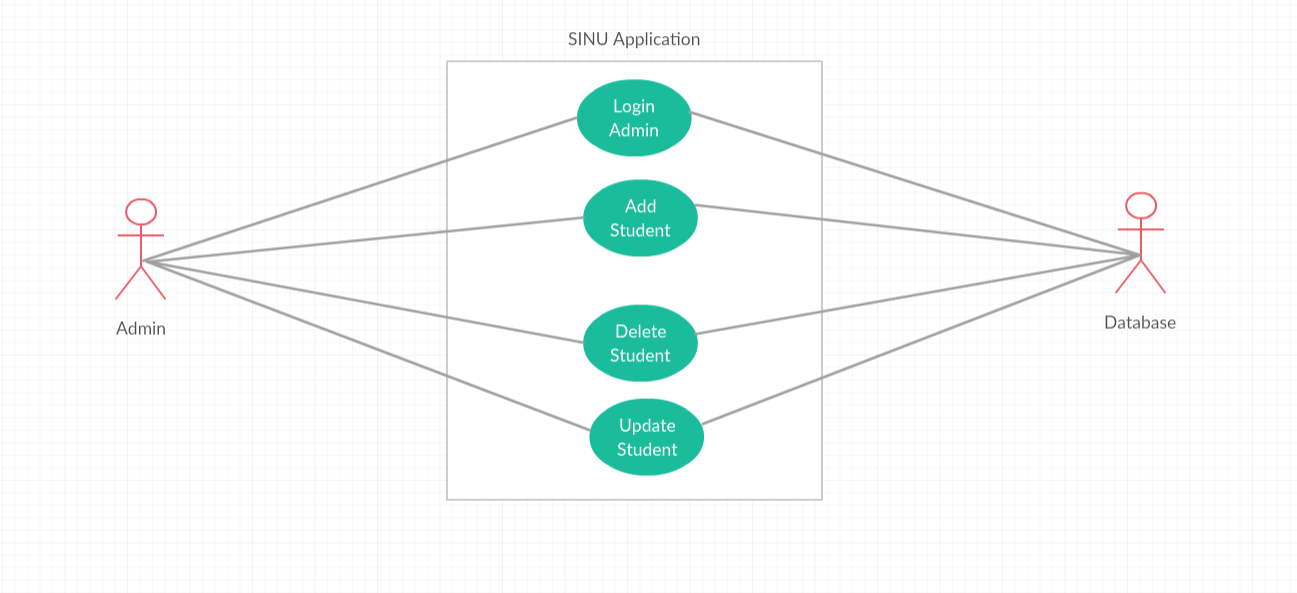
Application Constraints :

* The data will be stored in a relational database ( Postgres/MySql/MsSql). Use the Layers architectural pattern to organize your application. Create the Data Access Layer using SQL statements in the way you find most suitable for the application
* All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database

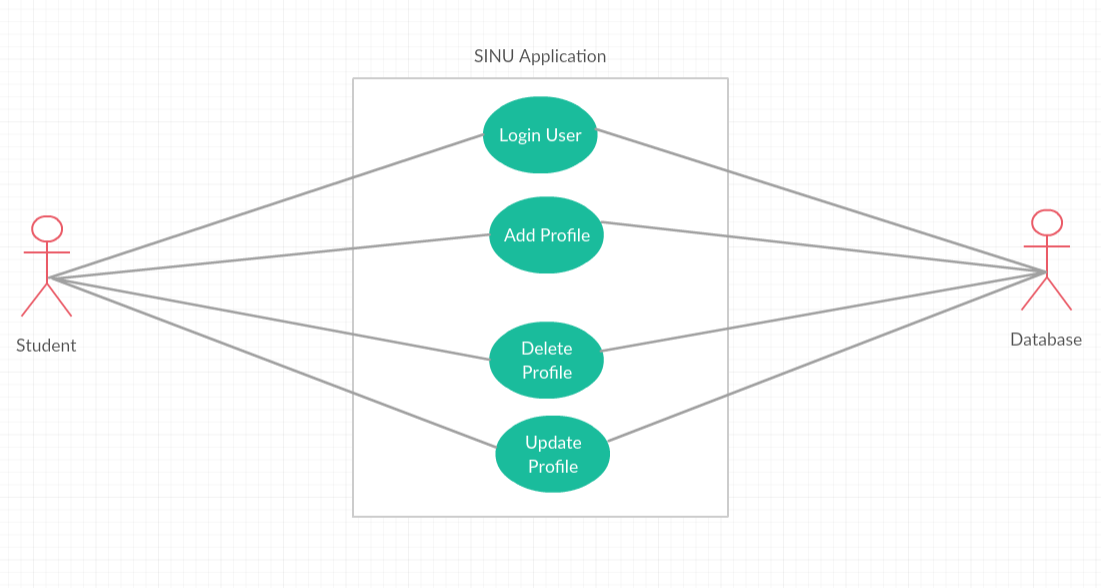
2. Use-Case Model

A Use Case Diagram is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It is represented using ellipse. Actos is any external entity that makes use of the system being modeled .

Use Case Diagram for Admin:



Use Case Diagram for Student:



The actors are Admin, Student and Database (the use cases are the activities performed by actors). Actions are Login Admin/Student . , Add Student/Profile, Delete Student/Profile, Update Student/Profile. Database stores the details and returns the information .

3. System Architectural Design

**3.1 Architectural Pattern Description**

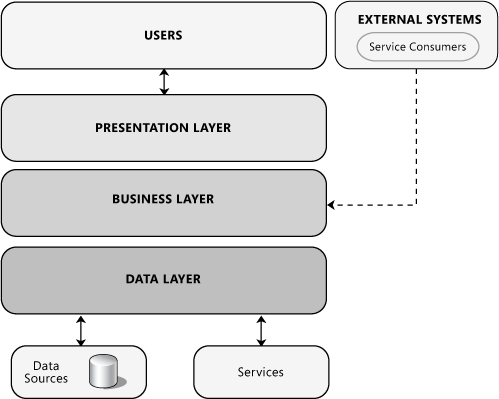
Architectural Pattern used in this application are : Layered pattern and N-tier Architecture.

Layered pattern can be used to structure programs that can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer. The most commonly found 3 layers of a general information system are as follows:

* Presentaion Layer
* Business Layer
* DataBase Layer

The idea is that the user initiates a piece of code in the Presentation Layer by performing some action ( e.g. clicking a button) .Then we go into the Business Layer and finally, stores everything in the Database Layer. So higher layers are dependent upon and make calls to the lower layers. Layer Pattern structure :

**3.2 Diagrams**



These Layers may be located on the same physical tier, or may be located on separate tiers. If they are located on separate tiers, or separated by physical boundaries, your design must accommodate this.

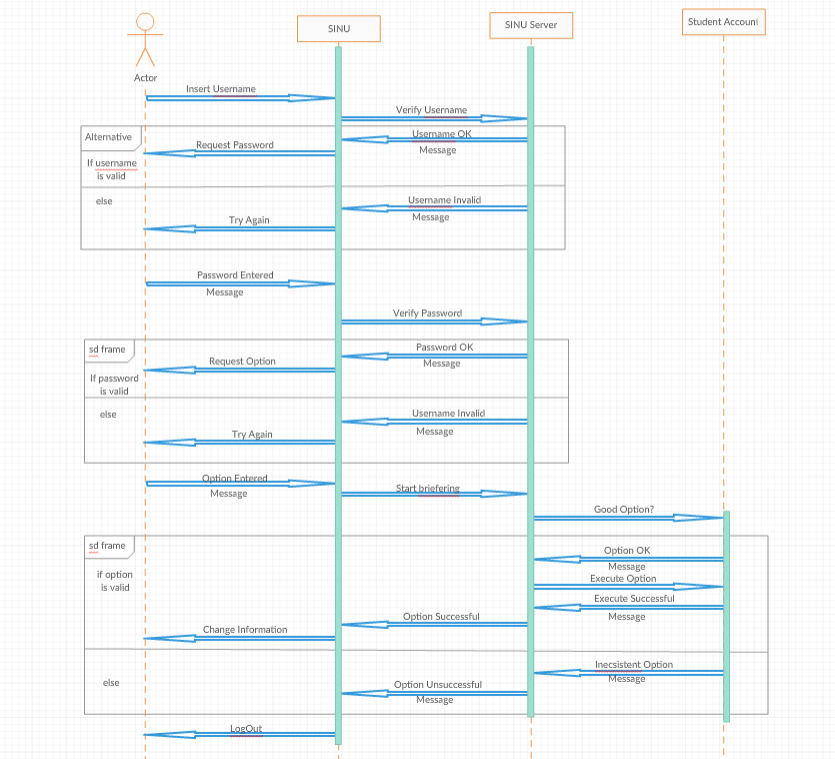
* Presentation Layer- This layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.
* Business Layer- his layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use.
* Data Layer- This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

4. UML Sequence Diagrams

A sequence diagram in Unified Modeling Language ( UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. There are two dimensions.

* Vertical dimension-represent time
* Horizontal dimension-represent different objects

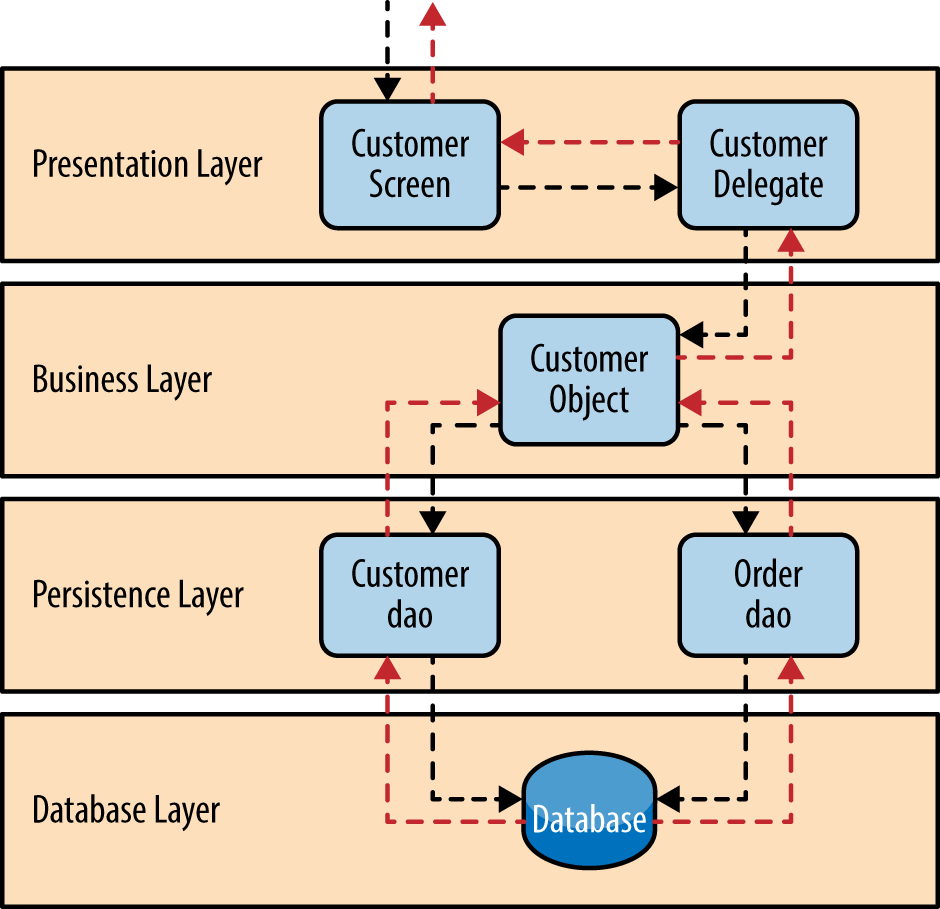
Example of UML Sequence Diagram ( for student) :



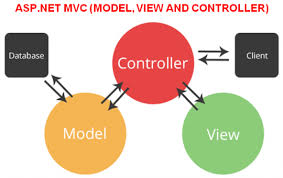
5. Class Design

**5.1 Design Patterns Description**

# [Software Architecture Patterns-Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn’t need to know or worry about how to get customer data; it only needs to display that information on a screen in particular format. Similarly, the business layer doesn’t need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data (e.g., calculate values or aggregate data), and pass that information up to the presentation layer.](https://learning.oreilly.com/library/view/software-architecture-patterns/9781491971437/)

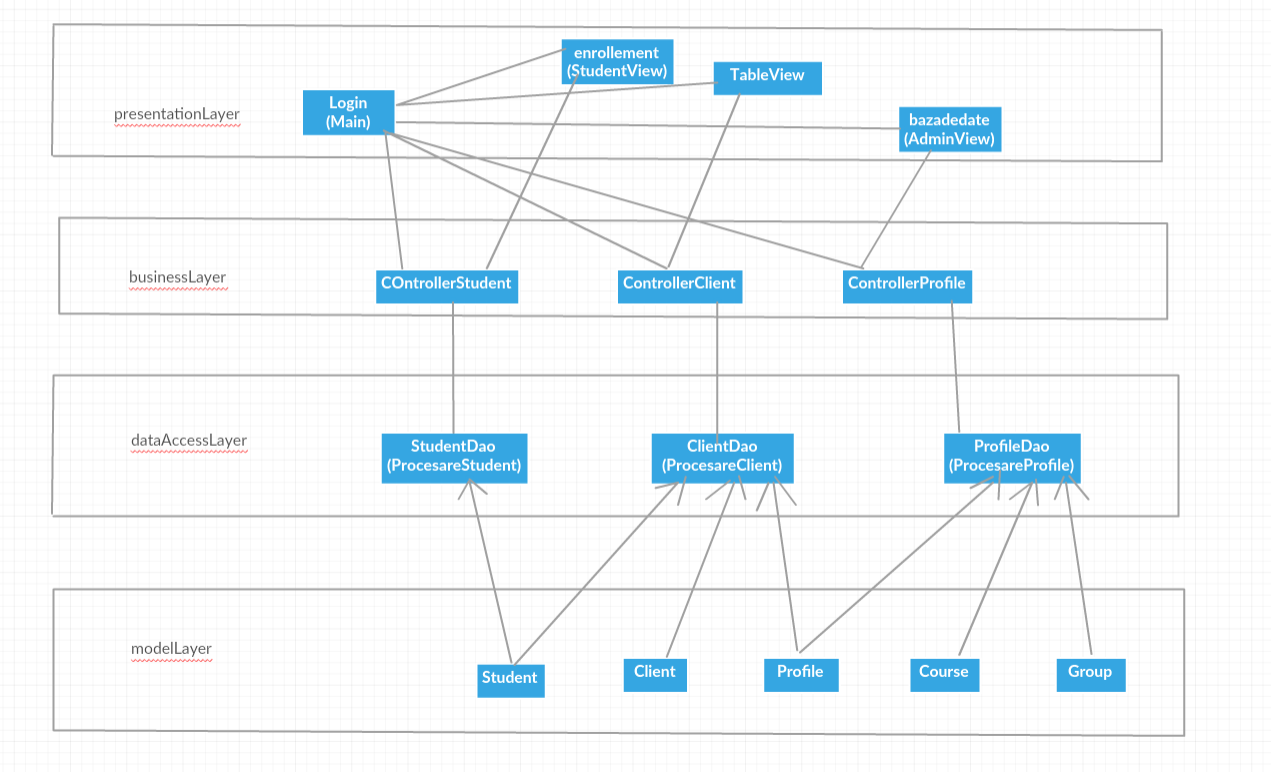


Model-View-Controller is a programming design pattern which was created to solve the design problem of a rather frequent application: Displaying data to the user (and possibly handling input from the user). The basic idea of this pattern is to divide the program (or module) into three main modules: The Model, the View and the Controller. In its purest form, the MVC pattern works like this:



**5.2 UML Class Diagram**

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, and the relationships between the classes. It is represented using a rectangle with three compartments. Top compartments have the class name, middle compartment the attributes and the bottom compartment with operations.



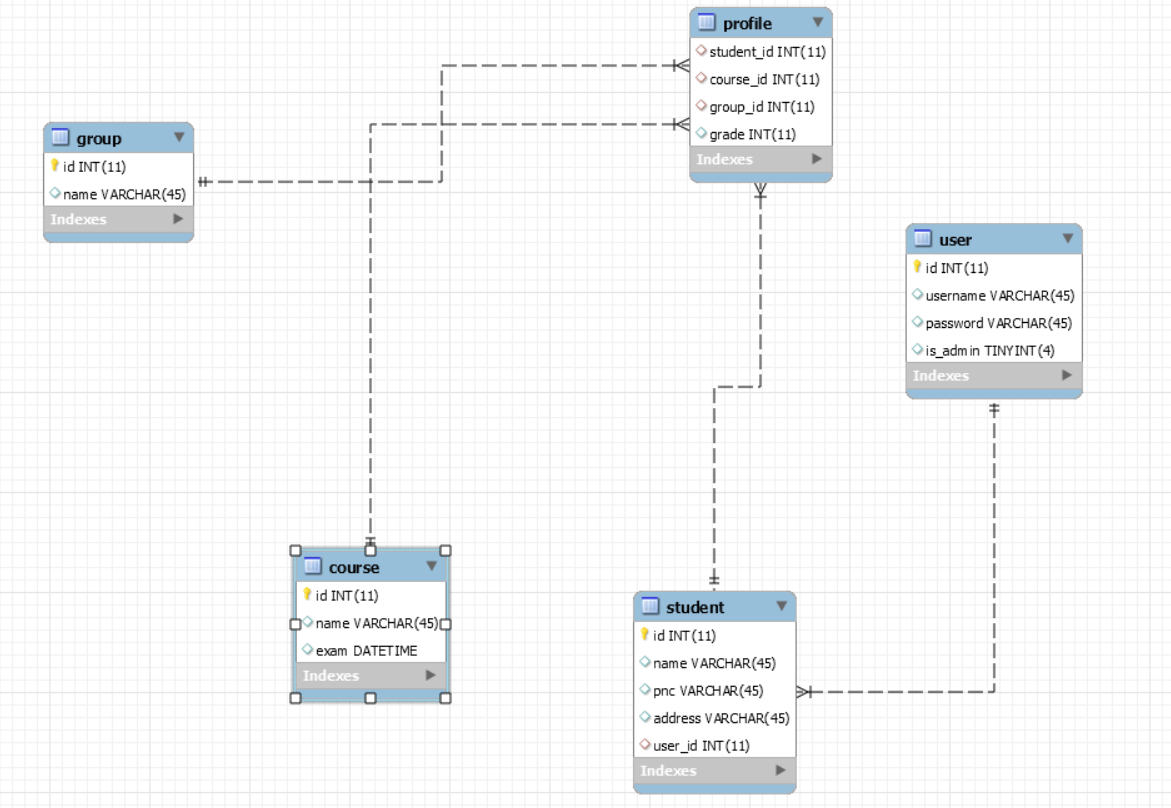
6. Data Model

Data modeling is the process of creating a data model for the data to be stored in a Database. This data model is a conceptual representation of

* Data objects
* The associations between different data objects
* The rules

Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. Data Models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data. Example of Data Model:

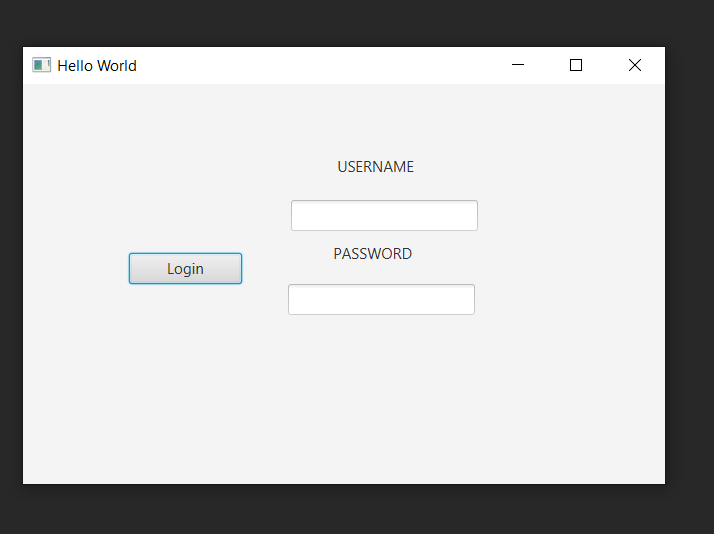
The Database used in this Application is PostgresSQL ; is a general-purpose object-relational database management system. It allows you to add custom functions developed using different programming languages such as C/C++, Java, etc.



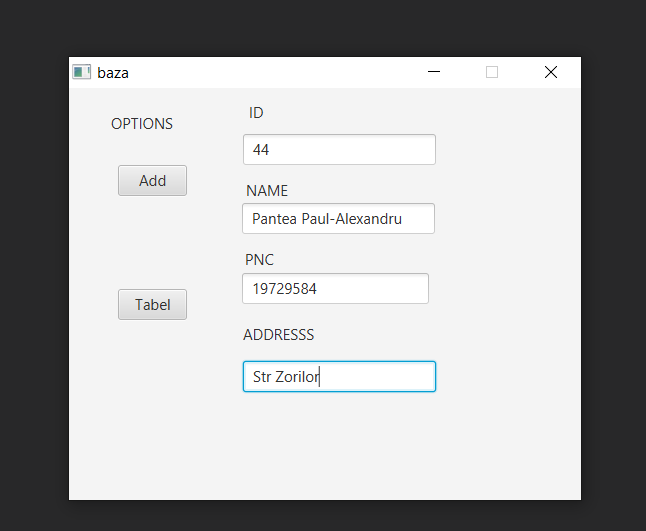
7. System Testing

We will use testing to verify that the functionality is implemented for both student and admin:

* Login Menu



* Adding Student Menu



8. Bibliography

* <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>
* <https://stackify.com/n-tier-architecture/>
* <https://www.academia.edu/9385119/UML_diagram_for_University_Information_System?fbclid=IwAR0oixpm0W-Phz4DRwtx3cshFUmwf59e1539skcBbL-gAlBDKj0o7meihfc>
* <https://www.journaldev.com/1827/java-design-patterns-example-tutorial#creational-patterns>
* <https://www.youtube.com/watch?v=zid-MVo7M-E>
* <https://www.youtube.com/watch?v=pCK6prSq8aw>
* <https://www.guru99.com/data-modelling-conceptual-logical.html>
* <https://opentextbc.ca/dbdesign01/chapter/chapter-5-data-modelling/>
* <https://sourcemaking.com/design_patterns>
* <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109%28v%3dpandp.10%29>
* <https://www.geeksforgeeks.org/mvc-design-pattern/>