College Portal
Analysis and Design Document

Student: Raul-Mihai Acu

Group: 30432

Revision History

Date	Version	Description	Author
21/03/2018	1.0	Vision, Use-Case Model, Glossary, Supplementary Specification	Acu Raul-Mihai
05/04/2018	2.0	Domain Model, Architectural Design, Component diagram	Acu Raul-Mihai
25/04/2018	3.0	Design Model, Data model	Acu Raul-Mihai
30/05/2018	4.0	Final	Acu Raul-Mihai

Table of Contents

I. Project Specification

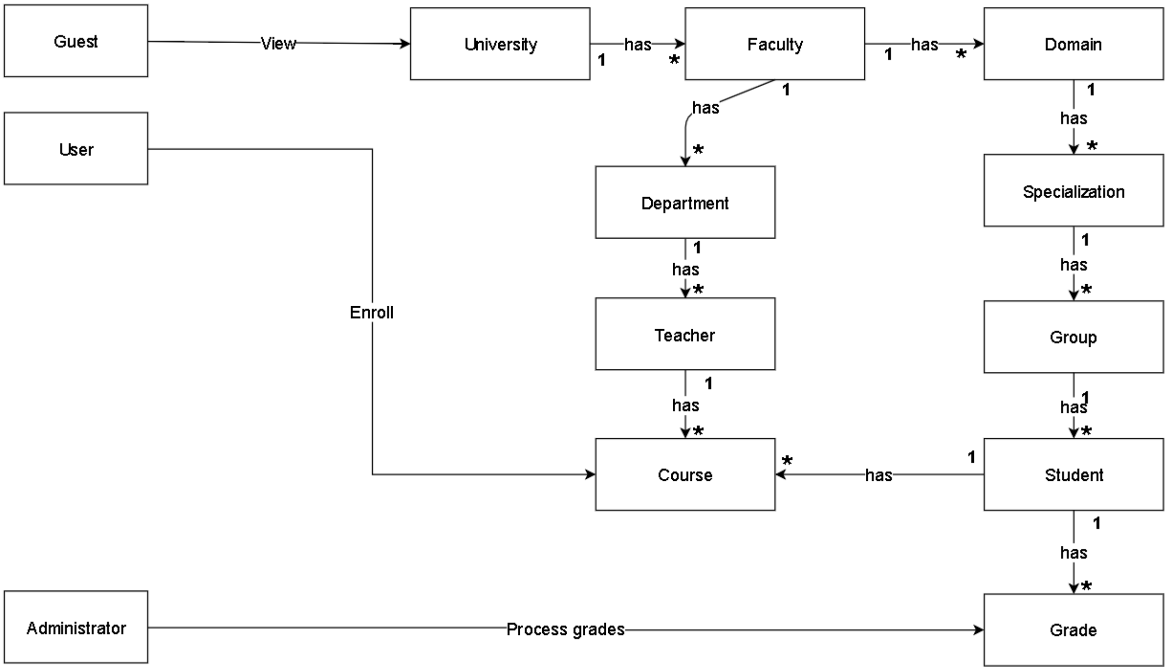
The College Portal application will provide means of communication between students and teachers, announcements, information regarding activities or events related to their college, and also access to their profiles, grades or payments.

II. Elaboration – Iteration 1.1

1. Domain Model

A domain model is a system of abstractions that describes selected aspects of a sphere of knowledge, influence or activity . The model can then be used to solve problems related to that domain. The domain model is a representation of meaningful real-world concepts pertinent to the domain that need to be modelled in software. The concepts include the data involved in the business and rules the business uses in relation to that data.

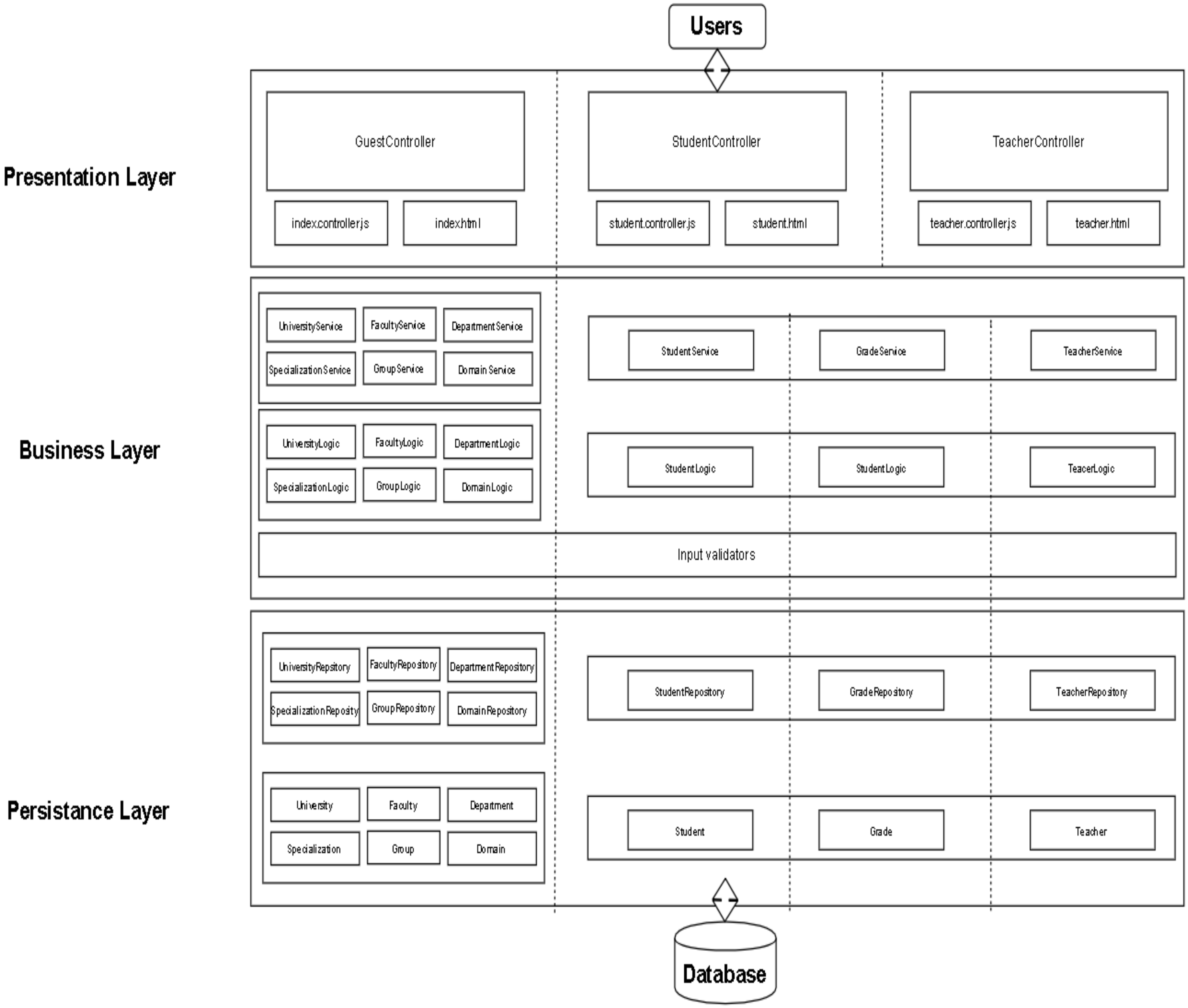
A domain model generally uses the vocabulary of the domain, thus allowing a representation of the model to be communicated to non-technical stakeholders.



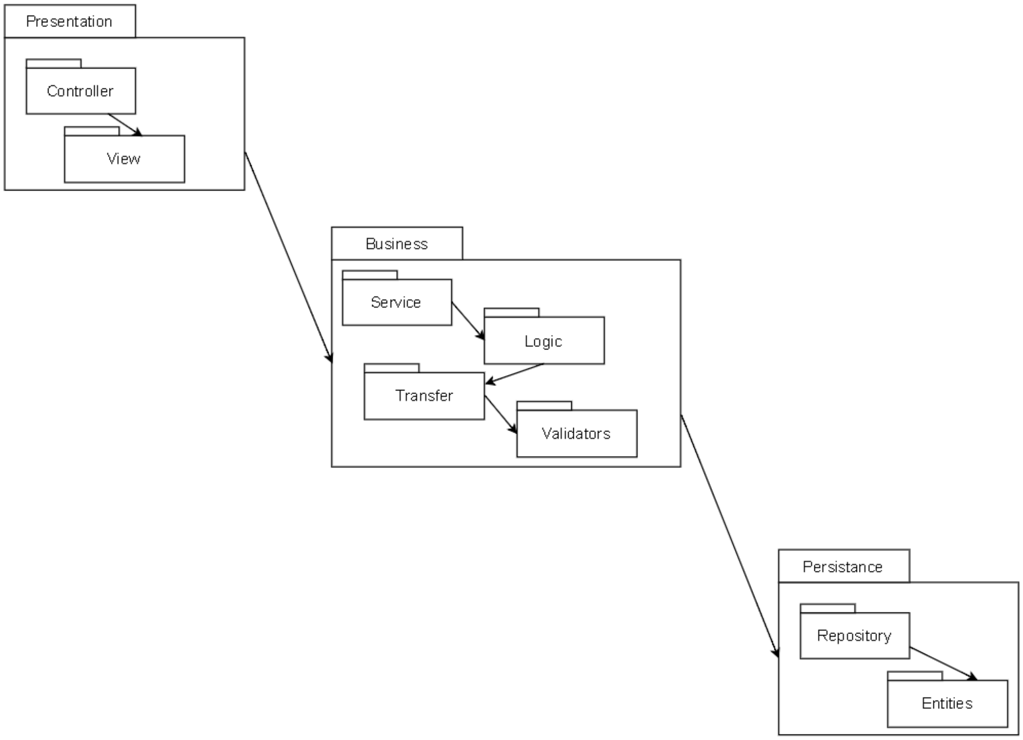
2. Architectural Design

2.1 Conceptual Architecture

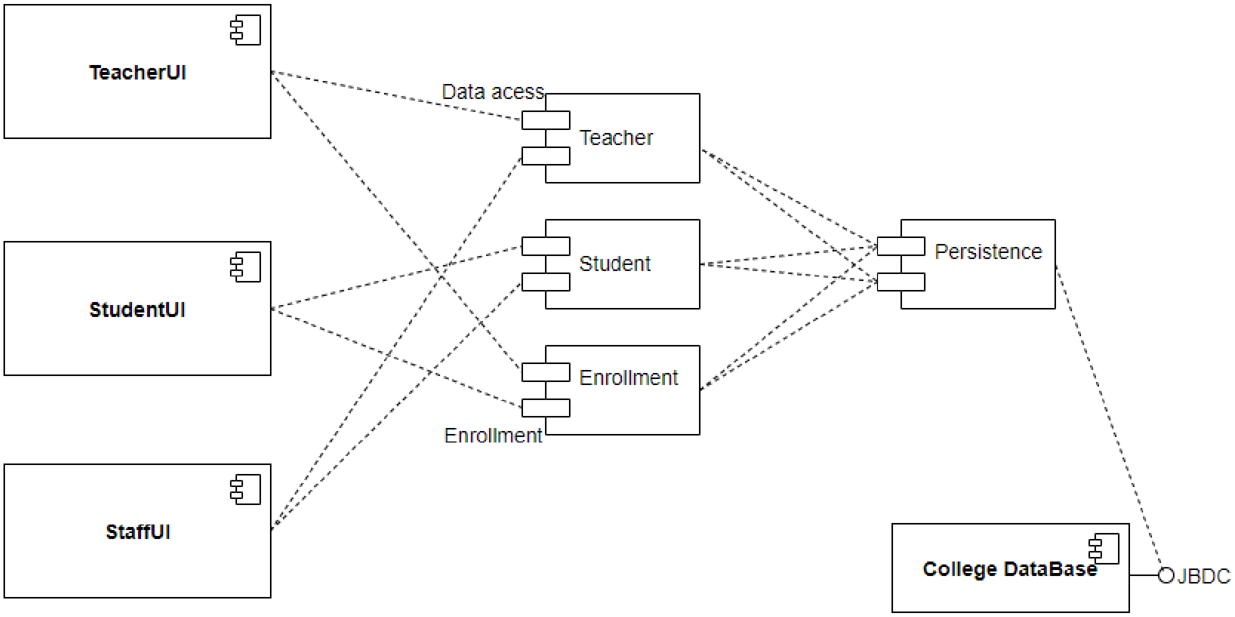
Conceptual architecture is a form of architecture that utilizes conceptualism, characterized by an introduction of ideas or concepts from outside of architecture often as a means of expanding the discipline of architecture. This produces an essentially different kind of building than one produced by the widely held 'architect as a master-builder' model, in which craft and construction are the guiding principles. The finished building as product is less important in conceptual architecture, than the ideas guiding them, ideas represented primarily by texts, diagrams, or art installations.



2.2 Package Design



2.3 Component and Deployment Diagrams

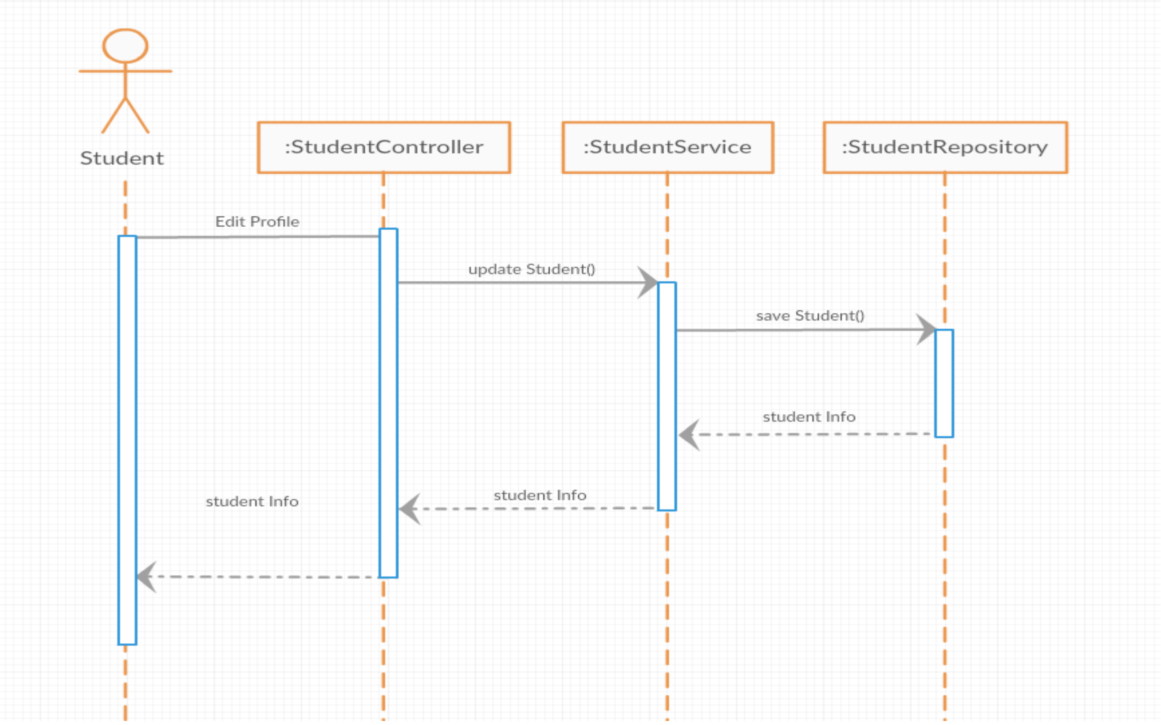


III. Elaboration – Iteration 1.2

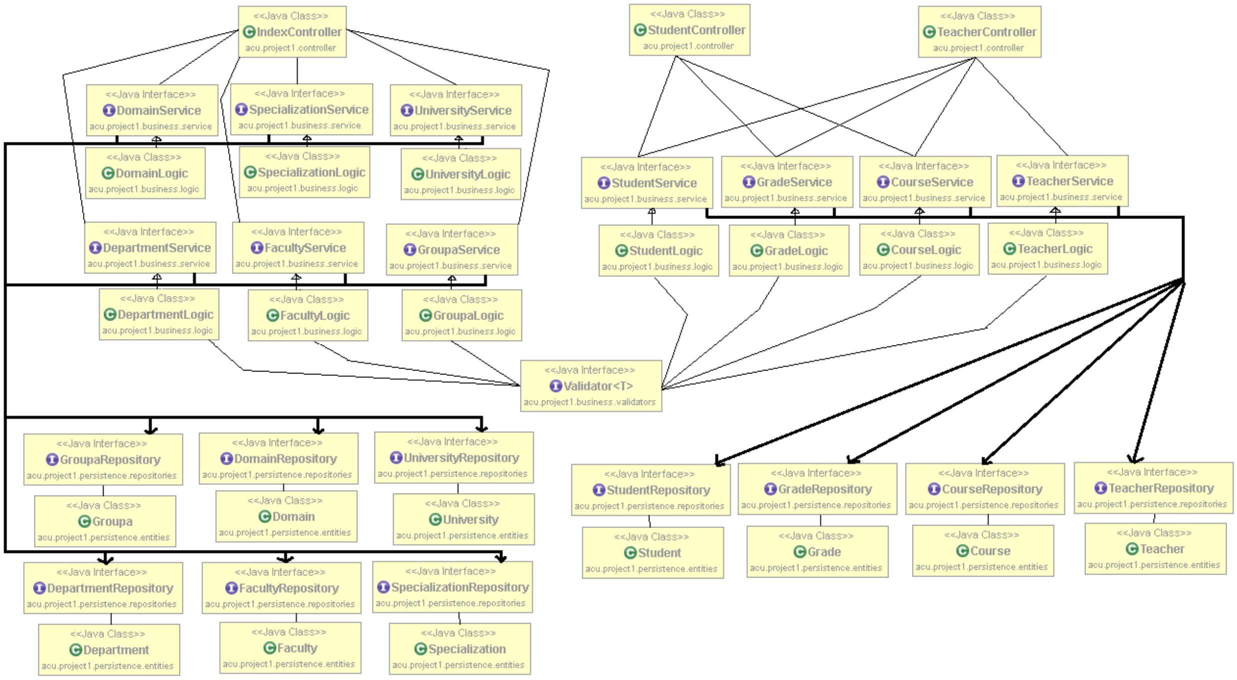
1. Design Model

1.1 Dynamic Behavior

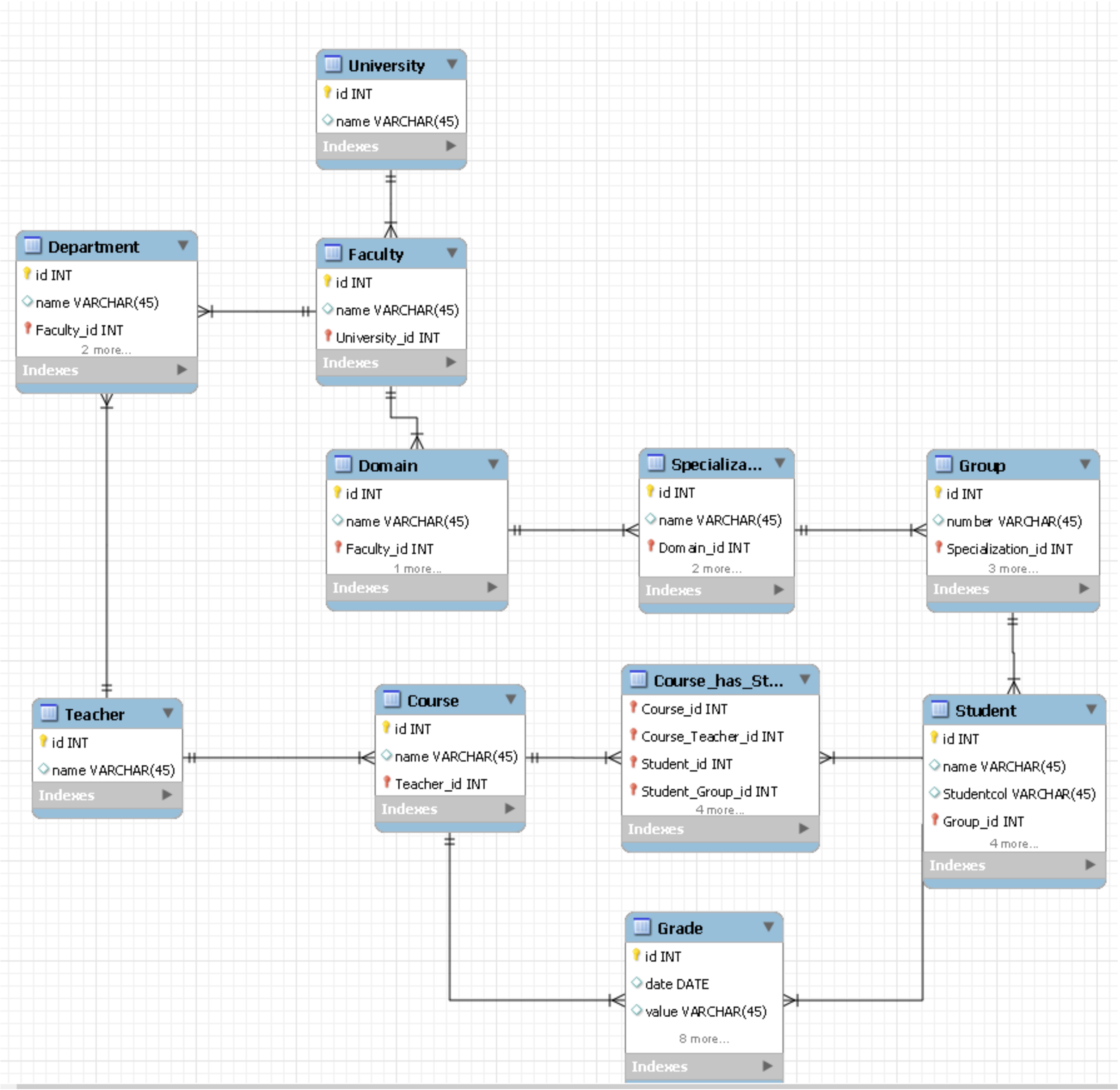
Sequence Diagram



1.2 Class Design


<<Java Class>> IndexController acu.project1.controller
<<Java Interface>> DomainService acu.project1.business.service
<<Java Interface>> SpecializationService acu.project1.business.service
<<Java Interface>> UniversityService acu.project1.business.service
<<Java Class>> DomainLogic acu.project1.business.logic
<<Java Class>> SpecializationLogic acu.project1.business.logic
<<Java Class>> UniversityLogic acu.project1.business.logic
<<Java Interface>> DepartmentService acu.project1.business.service
<<Java Interface>> FacultyService acu.project1.business.service
<<Java Interface>> GroupaService acu.project1.business.service
<<Java Class>> DepartmentLogic acu.project1.business.logic
<<Java Class>> FacultyLogic acu.project1.business.logic
<<Java Class>> GroupaLogic acu.project1.business.logic
<<Java Interface>> Validator<T> acu.project1.business.validators
<<Java Interface>> GroupaRepository acu.project1.persistence.repositories
<<Java Interface>> DomainRepository acu.project1.persistence.repositories
<<Java Interface>> UniversityRepository acu.project1.persistence.repositories
<<Java Class>> Groupa acu.project1.persistence.entities
<<Java Class>> Domain acu.project1.persistence.entities
<<Java Class>> University acu.project1.persistence.entities
<<Java Interface>> DepartmentRepository acu.project1.persistence.repositories
<<Java Interface>> FacultyRepository acu.project1.persistence.repositories
<<Java Interface>> SpecializationRepository acu.project1.persistence.repositories
<<Java Class>> Department acu.project1.persistence.entities
<<Java Class>> Faculty acu.project1.persistence.entities
<<Java Class>> Specialization acu.project1.persistence.entities
<<Java Class>> StudentController acu.project1.controller
<<Java Class>> TeacherController acu.project1.controller
<<Java Interface>> StudentService acu.project1.business.service
<<Java Interface>> GradeService acu.project1.business.service
<<Java Interface>> CourseService acu.project1.business.service
<<Java Interface>> TeacherService acu.project1.business.service
<<Java Class>> StudentLogic acu.project1.business.logic
<<Java Class>> GradeLogic acu.project1.business.logic
<<Java Class>> CourseLogic acu.project1.business.logic
<<Java Class>> TeacherLogic acu.project1.business.logic
<<Java Interface>> StudentRepository acu.project1.persistence.repositories
<<Java Interface>> GradeRepository acu.project1.persistence.repositories
<<Java Interface>> CourseRepository acu.project1.persistence.repositories
<<Java Interface>> TeacherRepository acu.project1.persistence.repositories
<<Java Class>> Student acu.project1.persistence.entities
<<Java Class>> Grade acu.project1.persistence.entities
<<Java Class>> Course acu.project1.persistence.entities
<<Java Class>> Teacher acu.project1.persistence.entities

2. Data Model



3. System Testing

• **Unit testing** is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

• **Graphical user interface testing** is the process of testing a product's graphical user interface to ensure it meets its specifications. This is normally done through the use of a variety of test cases. To generate a set of test cases, test designers attempt to cover all the functionality of the system and fully exercise the GUI itself.

• **Usability testing** is a technique used in user-centered interaction design to evaluate a product by testing it on users. Setting up a usability test involves carefully creating a scenario, or realistic situation, wherein the person performs a list of tasks using the product being tested while observers watch and take notes (dynamic verification).

IV. Bibliography

https://msdn.microsoft.com/en-us/library/ee658109.aspx
https://www.oreilly.com/ideas/software-architecture-patterns/page/2/layered-architecture
https://en.wikipedia.org/wiki/System_testing