Bilkent University

Department of Computer Engineering

CS319 Term Project

Section 3

Group 3A

Bilasmus

Design Report

Group Members:

Alp Afyonluoğlu (22003229)
Emre Karataş (22001641)
Erkin Aydın (22002956)
Parsa Keihan (22001422)
Selin Bahar Gündoğar (22001514)

Instructor: Eray Tüzün
Teaching Assistant: Mert Kara

Contents

Design Report

Bilasmus

1. Introduction

1.1. Purpose of the system

Bilasmus is a web-based project to help both Bilkent University students and faculty members during Erasmus and exchange programs. Bilasmus is used after the approval of students. The program's primary goal is to minimize paperwork as much as possible, and ease communication between students and faculty members. Bilasmus has different user types, such as Erasmus/Exchange coordinators, outgoing/incoming students, admin, department secretary, and the international student office. Thus, essential processes such as the course approval period and its paperwork are done in the Bilasmus system. The system also provides and reflects any update on each period of the Erasmus/Exchange process. The website also provides an in-system messaging platform and a To-Do list for each user as a reminder.

1.2. Design goals

The application offers a sustainable, user-friendly, and functional user interface. As there are users of different kinds, Bilasmus will address to users with all users, even those with low knowledge of computer and technology organization. Also, it is essential to note that requirements from the analysis report form the design goals and purposes of the system; therefore, the system should be secure and maintainable as well. It should also have the most efficient performance for the best experience. Though, the most important two non-functional requirements are usability and reliability in our project.

1.2.1. Usability/User-friendliness

Bilasmus is a program that addresses a variety of actors: outgoing students, incoming students, Faculty Board Executive Committee representative, coordinators, instructors, International Student Office, admin, and department secretary. Thus, not everyone may be familiar with the newest technologies or may want to spend so much time on the application. Therefore, the program is planned to be designed as simple and usable as possible with the most user-friendly interface possible. This is to say that the features used in the system, such as buttons, sections, upload/download fields, toolbars, etc., should be easy-to-use. Each section of the application will have text sections for those unfamiliar, and there is a help button to let the user contact the responsible if there is a problem or question. Every clickable section (such as buttons) is appropriately designed, and each text field or input section is adequately aligned. The color design of the program is also helpful for the users as they are matched with the general concept of design. For example, delete buttons are colored red, and download buttons are colored green. In general, the program will be designed to provide multi-dimensional ease of operation and intelligibility to various users. This section's important features are an easy-to-read font theme and size, aligned text/button/alert fields, and a useful tab for transitions.

1.2.2. Security/Reliability

The system will be as secure as possible as it contains essential documents of Erasmus/Exchange students and actors' personal details. Details such as ID, password, courses, transcripts, etc. will be shared with other actors accordingly if needed. Each actor will see the required information provided for them according to their roll in in the application. This means a student cannot access some other student's transcript. Moreover, the system should ensure that no data will be deleted during any crash. Each completed assignment in the system should be saved and kept as a record in the system as long as needed based on the university's rules.

1.2.3. Sustainability/Maintainability

The system should be supported for all up-to-date web browsers. The locations of every feature in a page, such as buttons, alerts, and text fields should be sustained in all devices. The database should be updated every 30 minutes, and the overall control should be maintained by admins if needed. Also, the system will be designed in a way that anyone as Bilasmus' new developer could build up on it easily. To make the code of the program understandable, comments and reports will be used; therefore, new features can be added faster and easier. As we apply OOP, it ensures that adding new elements or modifying the existing ones will not be a difficulty.

1.2.4. Performance

Overall, the system should be as sufficient as it can be. This means there should not be any crash, data loss, and user dissatisfaction performance wise. This follows that users should run the program fluently and have the best experience provided by the system. Therefore, the number of the users using the system at the same time should not cause a problem in the performance of the system, and anyone with internet access and sustainable web browser should be able to use the system.

2. High-level software architecture

2.1. Subsystem decomposition

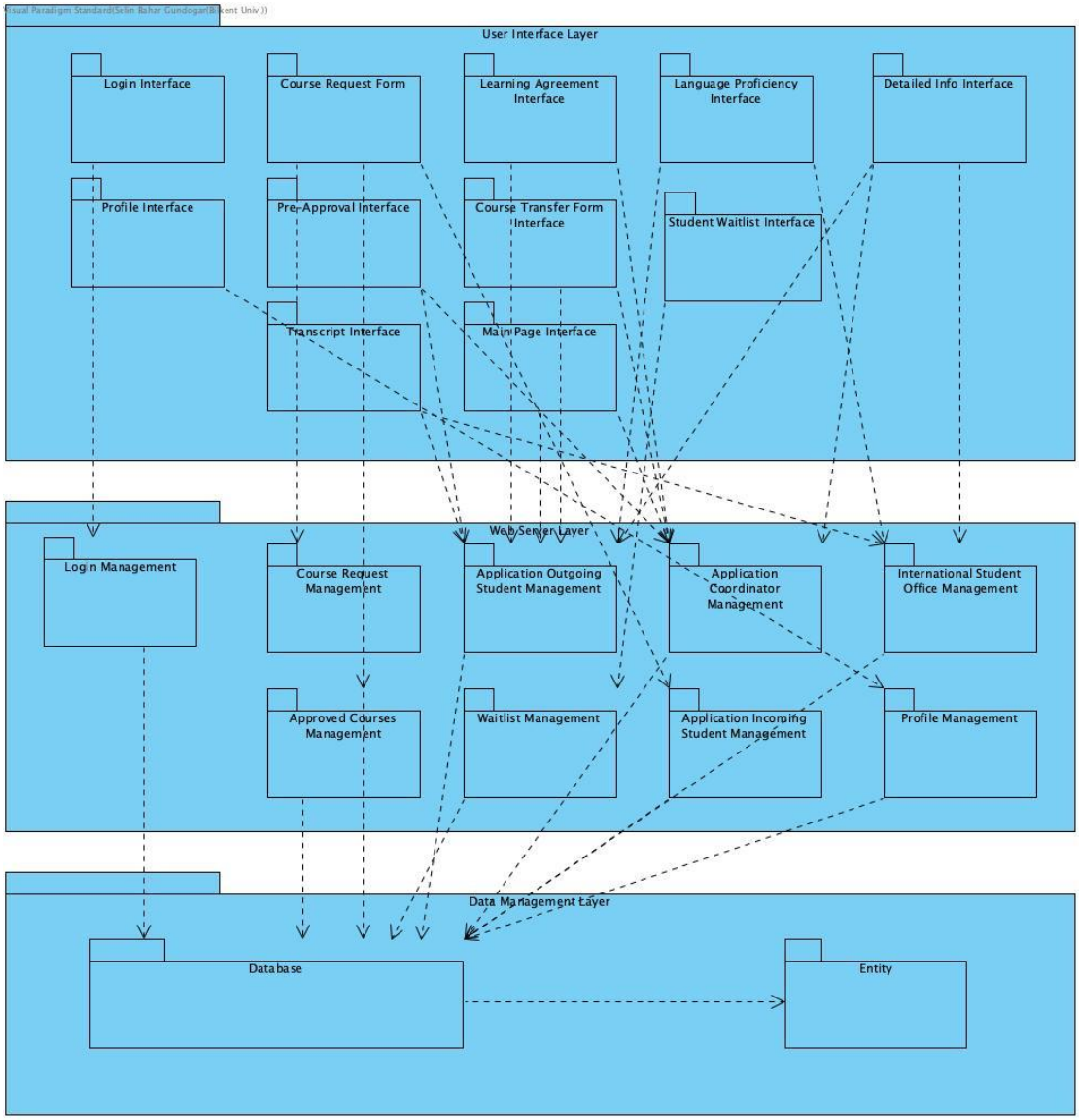

Figure 2.1: 3 Layer Architecture

A three-layer architecture was used in our planning. The User Interface Layer represents the boundary level and is responsible for guiding and showing the user the visuals of the website and takes requests from users and sends it to the controller layer.

The Middle section is the Web Server Layer which serves as the backend of the code and represents the controller section. It is connected to both the boundary and the entity part of the architecture.

The Data Management Layer contains the database and its repositories contains all information about the users and their files. It represents the Entity layer and is connected to controller.

2.2. Hardware/software mapping

This is a web project. Hence, there are no strict hardware requirements, any device able to handle modern web browsers should be sufficient. Our program should be able to run on stable versions of Google Chrome and Chromium-based browsers, Mozilla Firefox, and Safari. Our project should work on any operating system running these browsers. Typical computer components, such as at least one monitor, keyboard, and mouse, are required to use the web application.

2.3. Persistent data management

There were two options for databases: relational databases(SQL) and non-relational databases(NoSQL). There can be, of course, other types of databases, but since we have never heard of them, they were never an option in this limited time. We knew about non-relational databases, and some worked with non-relational databases. However, since non-relational databases are not dynamic, we thought they might create specific problems. That's why we eliminated non-relational databases. We chose PostgreSQL out of all relational databases, as it was the one we were most familiar with.

2.4. Access control and security

Security and reliability of the data is guaranteed in the software. The data obtained from software's database are correct and dependent on the software.Also, it should be noted that the data is kept safe by various precautions provided by PostgreSQL.

All of the users in the system are able to login and sign out from the application. Depending on their role, they can see reports submitted to them, the reports that they need to submit, notifications, messages from other users. Furthermore, every user may reach the help section in the navigation bar to tell them how to use the application.

Student users are divided up to two different roles, incoming student and outgoing student. Incoming students coming from abroad universities to Bilkent University submit a form of the courses they wish to take at Bilkent and their request is sent to the coordinator for confirmation. Coordinator evaluates the list and sends it to the department secretary. Based on the coordinator's approval, the incoming students will be registered to their courses by the department secretary. On the other hand, outgoing students submit their course form, pre-approval form, learning agreement, and their language proficiency results before going to Erasmus/Exchange. The coordinator is responsible for checking all of these submitted documents except for the language proficiency result. For the course form, the instructor may also access the course if the course is a mandatory course that has not been approved before. Learning agreement forms are approved or rejected by the faculty executive committee board after it has been approved by the coordinator.

Apart from evaluating students' forms, the coordinator also uploads a course transfer form to the faculty executive committee board. Coordinator also receives the transcripts from the international student office and receives Erasmus student ranking excel sheet from the admin.

Admin staff may add new users to the program. If it is needed, they can change information regarding users. In some cases (if a student cancels his/her placement or for other valid reasons) admin staff may delete users from the system. Admin also sends the list of student raking for Erasmus to the coordinator.

Department secretary receives the list of courses that the incoming students wish to take after the courses have been approved by the coordinator.

International student office is responsible for announcing language proficiency exams taking place, processing language proficiency results, and sending the transcripts to coordinators.

Faculty executive committee board is responsible for approving or rejecting learning agreement forms and course-transfer forms.

2.5. Boundary conditions

2.5.1. Initialization

Bilasmus only requires an internet connection to use the website. As the users cannot sign up to the application, the admin user must register every user that will be using the application to the system, in which the system automatically sends a url link through email to registered users of the application. By clicking on the link, users will decide on their passwords on the application and be able to use the application.

Additionally, all data that will be kept about Bilasmus will be stored in the database, and the related data will be fetched for the specific user.

2.5.2. Termination

Any resulting system crash will automatically lead to the termination of the application as the system must be working fully right to save any data changes the user may make. The user will be notified if a system crash does occur and will be prompted to login to the application again. The system will not abruptly shut down but instead users will be notified of it. Furthermore, if the admin will be making any maintenance to the application, then the users will be notified priorly on the application a day before the maintenance occurs.

2.5.3. Failure

Google Cloud Platform App Engine server will be used as the domain to handle the unintended failures of the system. On the other hand, the intended failures of the system that are caused by the developing situations are handled by returning different HTTP codes properly. In these cases, the GUI pop-ups a notification to describe the situation and failure to the user.

3. Low-level design

3.1. Object design trade-offs

3.1.1. Functionality v. Usability

Our project is supposed to be used by everyone who is involved in the Erasmus project, ranging from incoming and outgoing students to the academic staff

such as coordinators, instructors, faculty board committee. This makes the application functional. On the other hand, the application and interface are easy to use. It has been tried to keep a balance between functionality and usability. However, because the application focuses on students' and staffs' use, we focused more on usability.

3.1.2. Rapid development v. Functionality

Because of time limitations on the development of the application, rapid development is one of the main concerns for the application. This time constraint causes the application to be less functional.

3.1.3. Security v. Usability

When a user logs in to the application, unless the user signs out, the session remains active for one hour. This increases usability but security decreases as there is no need to log in again in one hour.Also, it should be noted that new passwords are sent by e-mail to the user. This increases security of the user.

3.2. Final object design

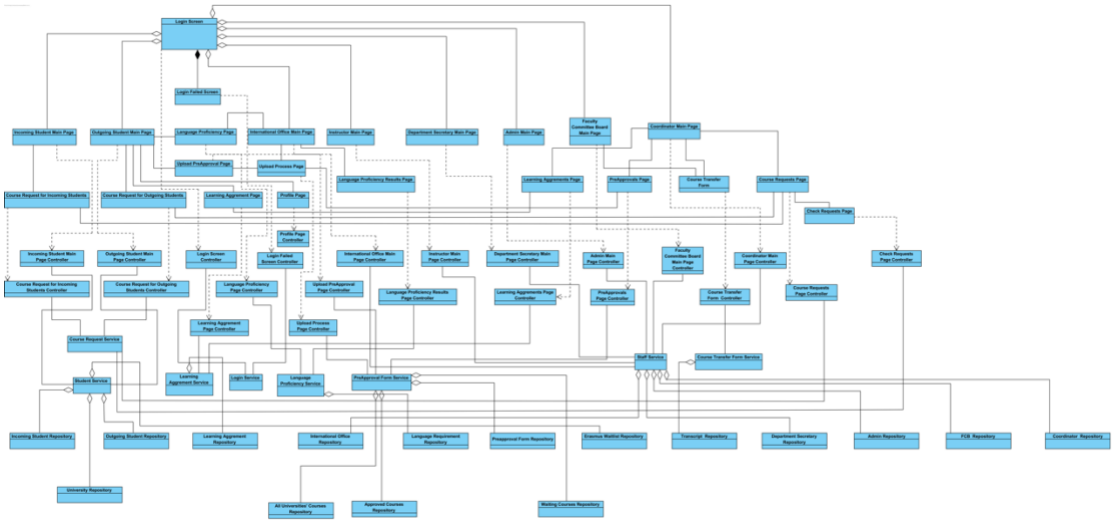

Figure 3.2: Final Object design

Figure 3.2 represents the final object design of the Bilasmus application. For readability, the diagram split into three different parts.

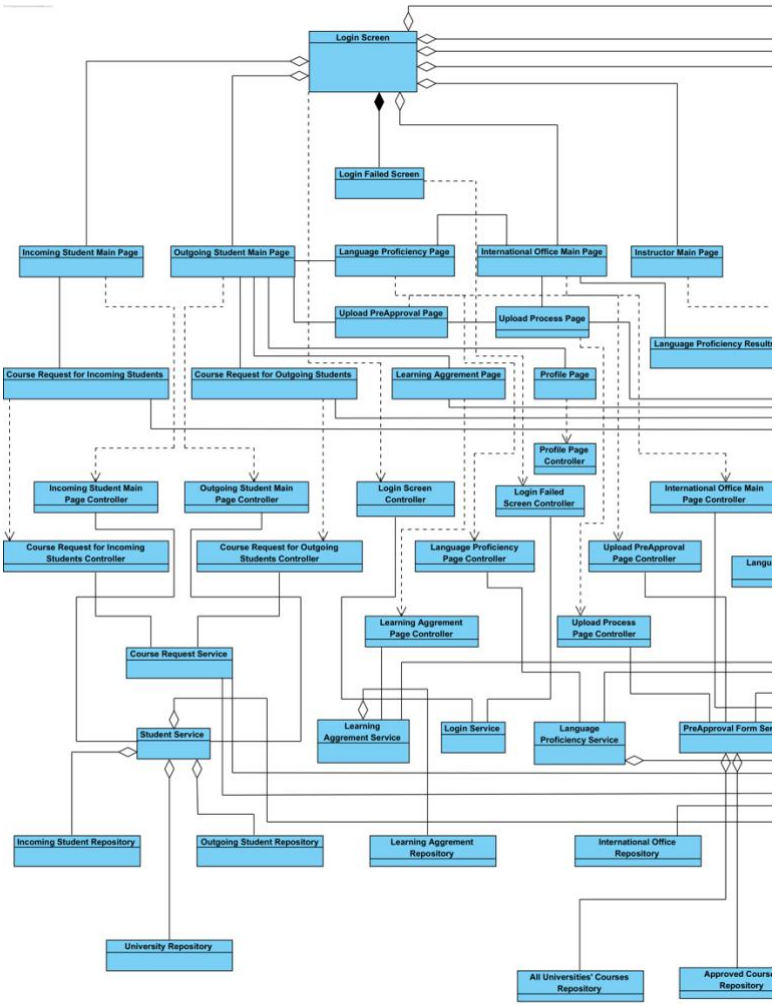

Figure 3.2: Left side of the Final Object Design

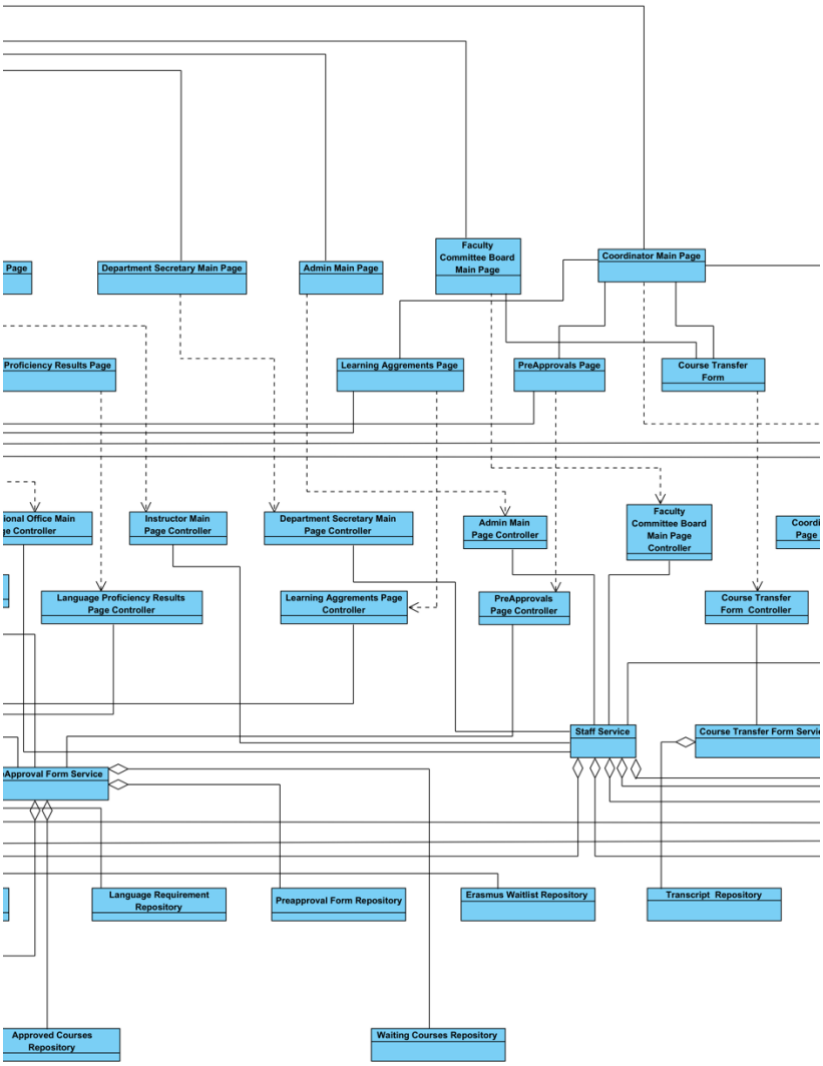

Figure 3.2: Middle side of the Final Object Design

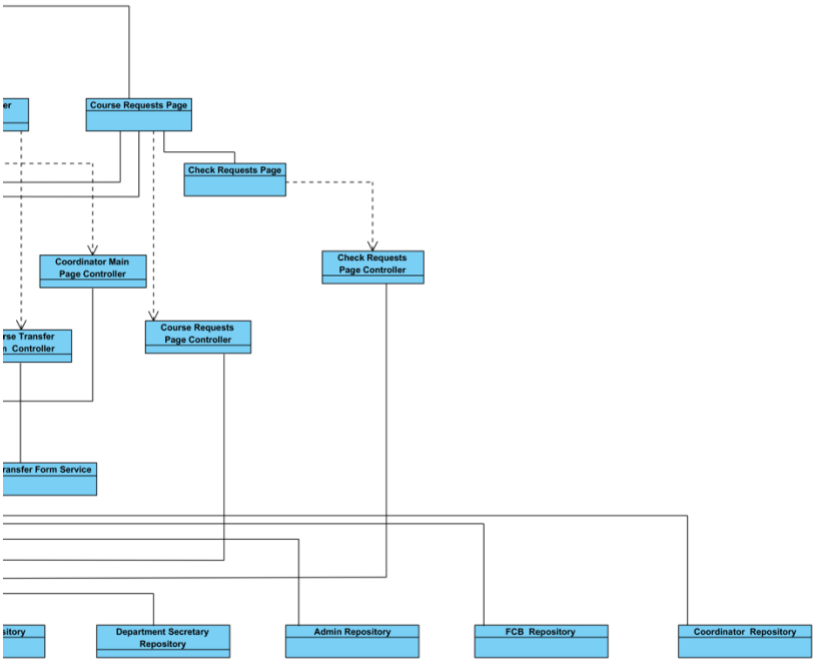

Figure 3.2: Right side of the Final Object Design

3.3. Layers

3.3.1. User Interface Management Layer

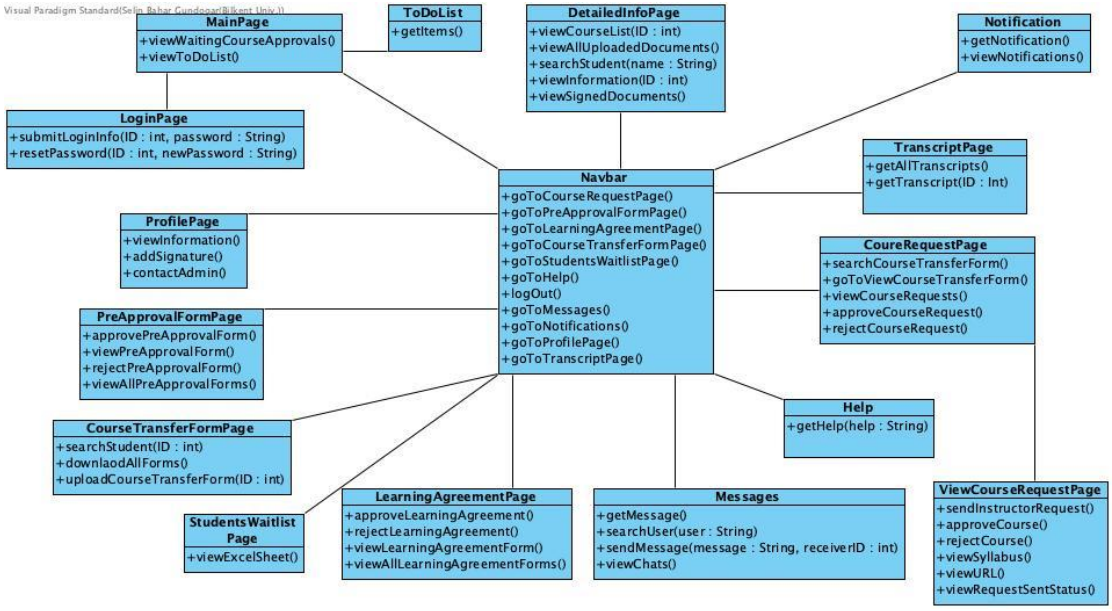

Figure 3.3.1: User Interface Management Layer

The User Interface Management Layer shows the connection between the user interface layers of the most complicated user, the coordinator. As the coordinator includes maximum functionality of the application, it was preffered to be showcased.

3.3.2. Web Server Management Layer

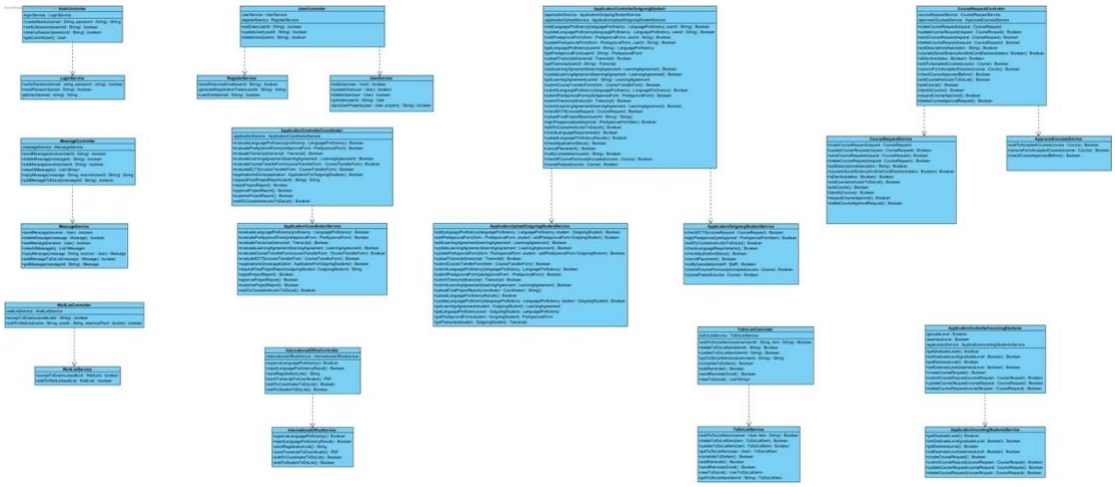

Figure 3.3.2: Web Server Management Layer

This represents the web server management layer of our project. It links the User Interface Layer and the data management layer and is responsible for all requests to get connected to each other between the frontend and the backend.

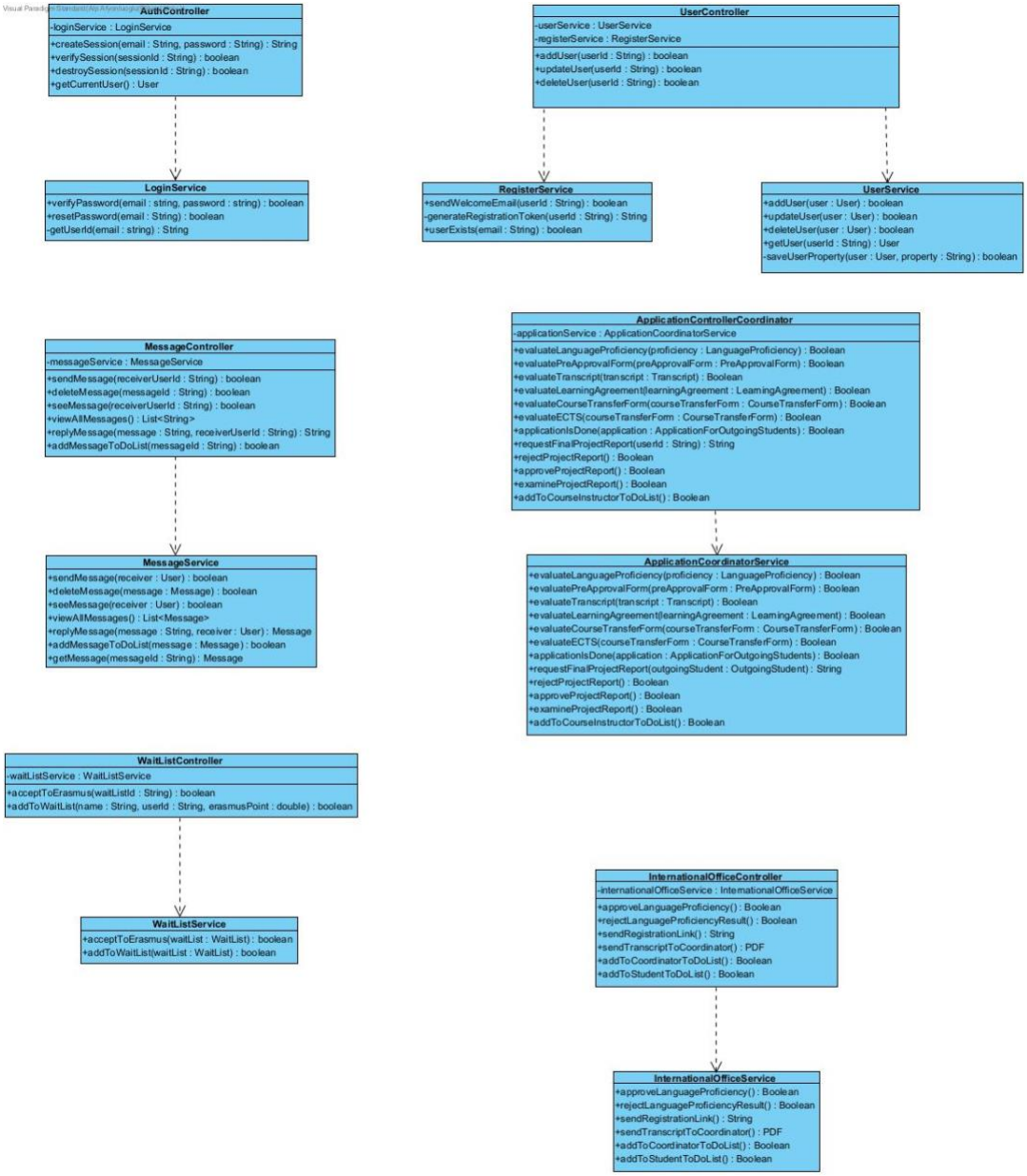

Figure 3.3.2: Left side of Web Server Management Layer

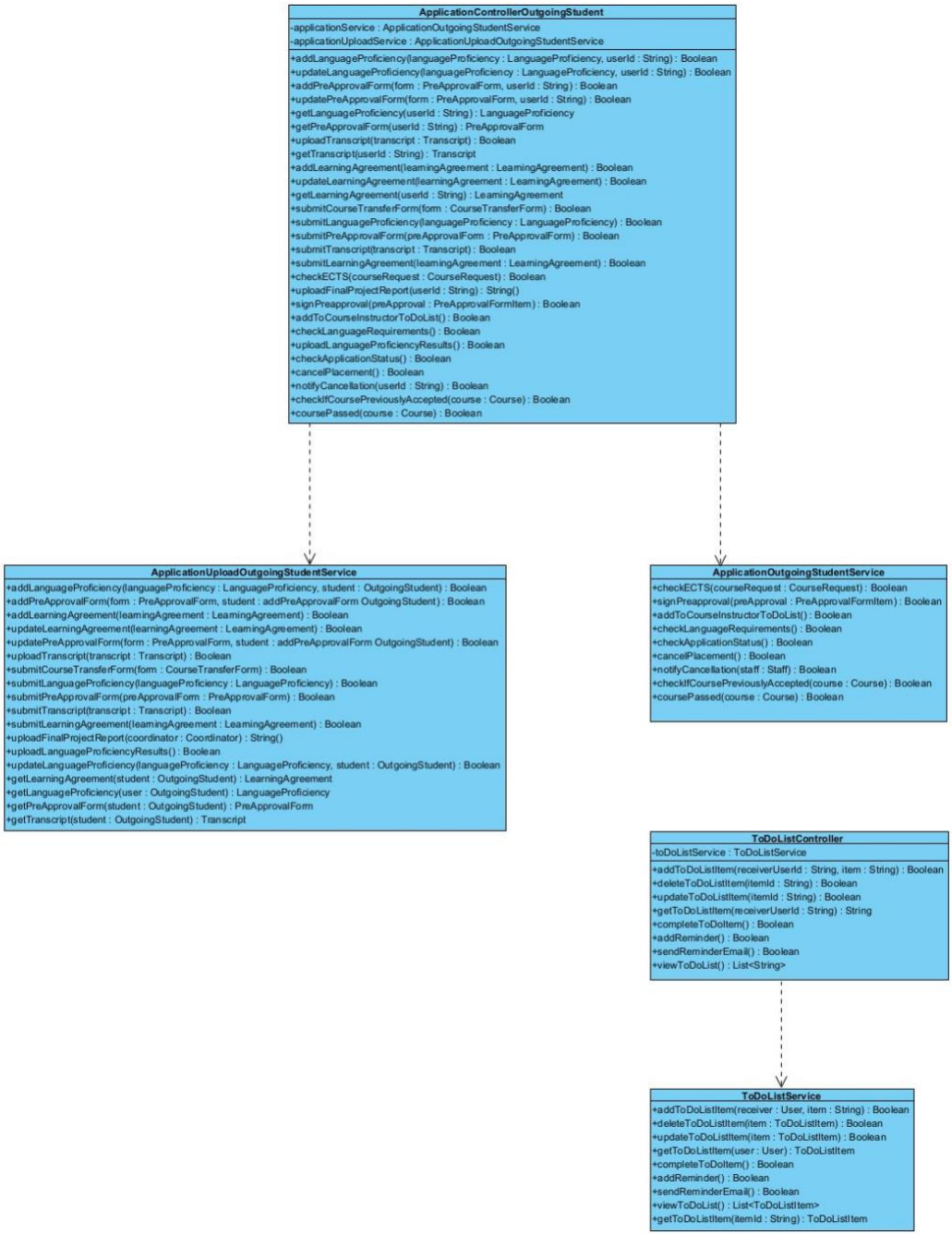

Figure 3.3.2: Middle of Web Server Management Layer

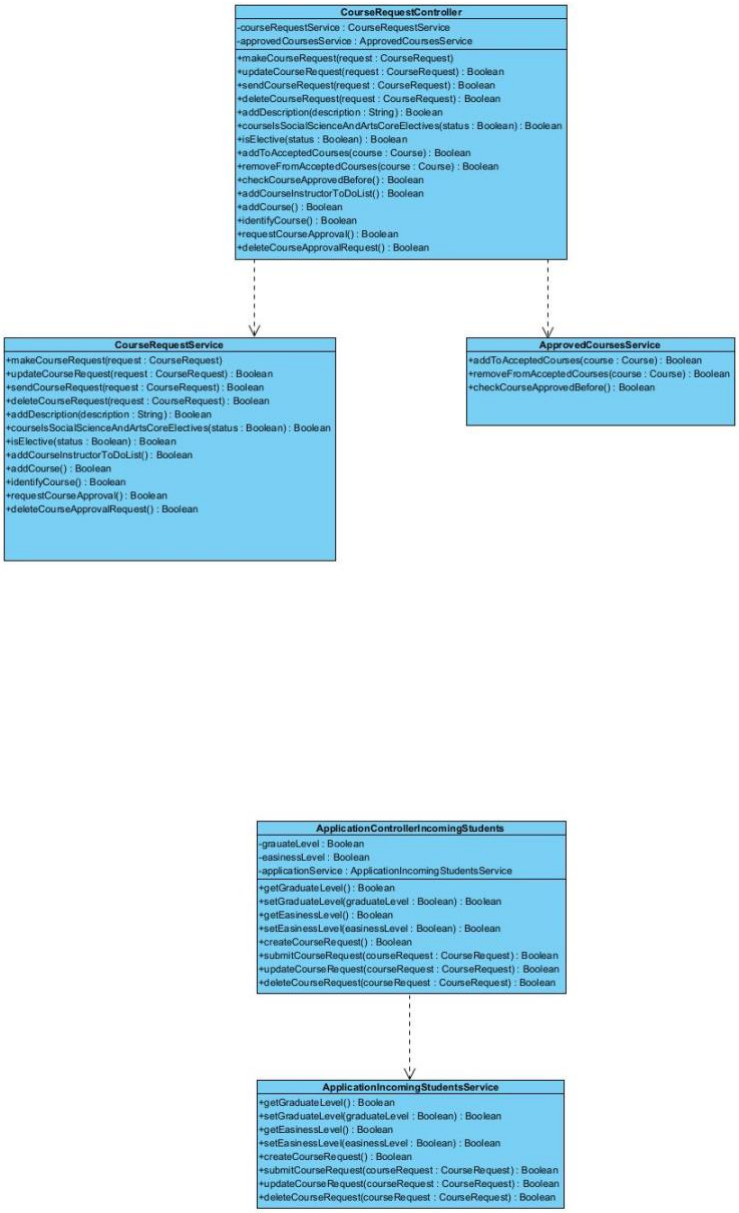

Figure 3.3.2: Right side of Web Server Management Layer

3.3.3. Data Management Layer

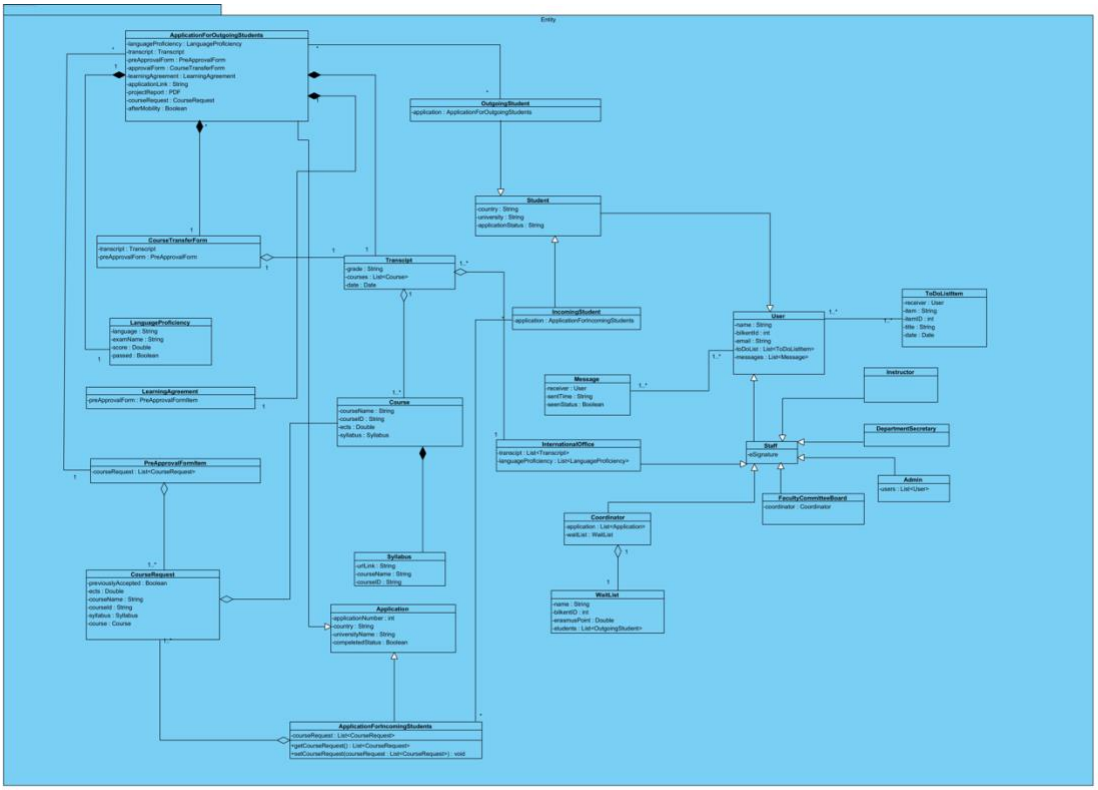

Figure 3.3.3: Entity Classes

In Figure 3.3.3, in addition to what have been shown in the figure, each class have empty and parametrized constructors and getter and setter methods of attributes. Note that operations are not shown in the figure for simplicity.

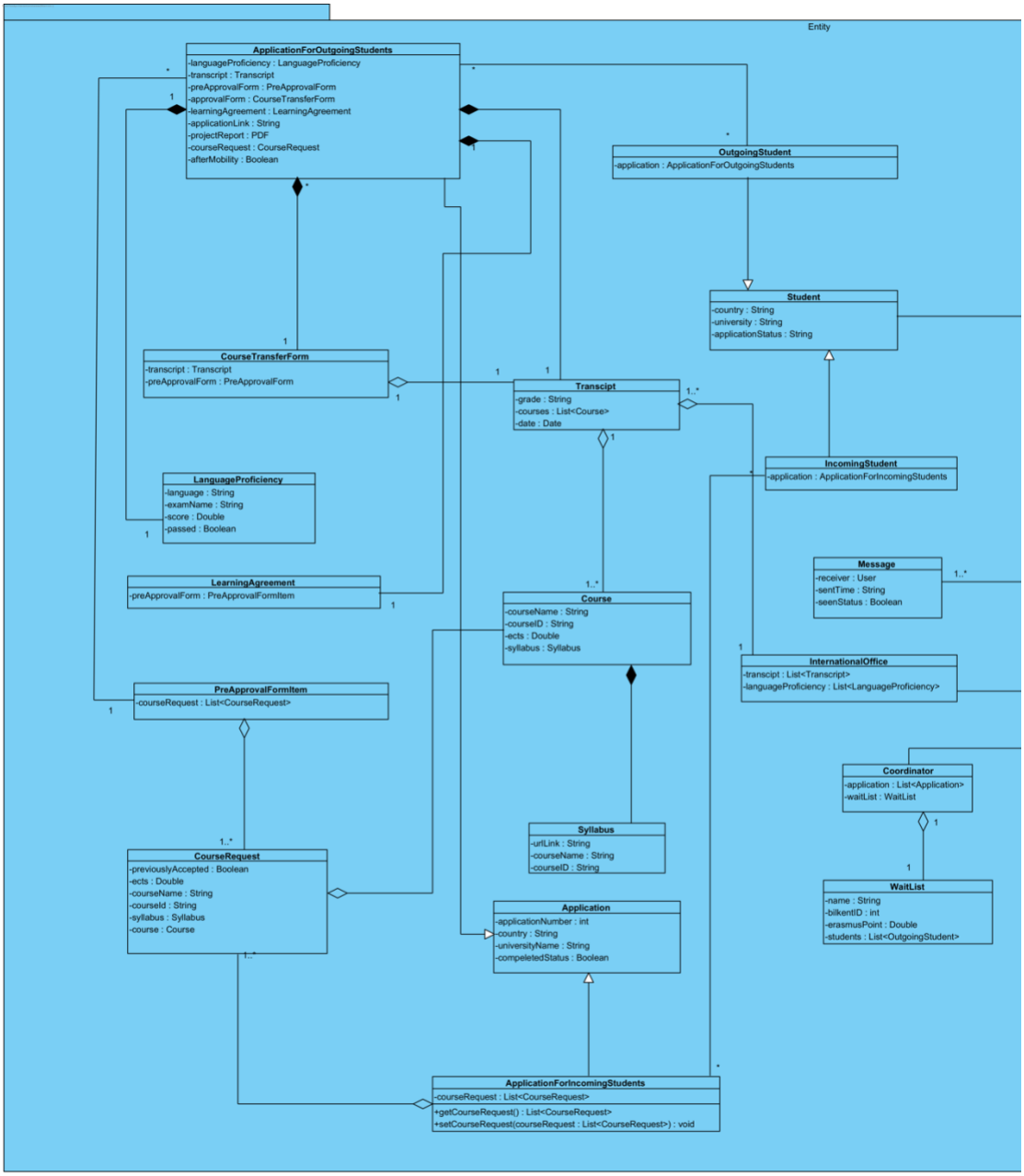

Figure 3.3.3: Left side of Entity Classes

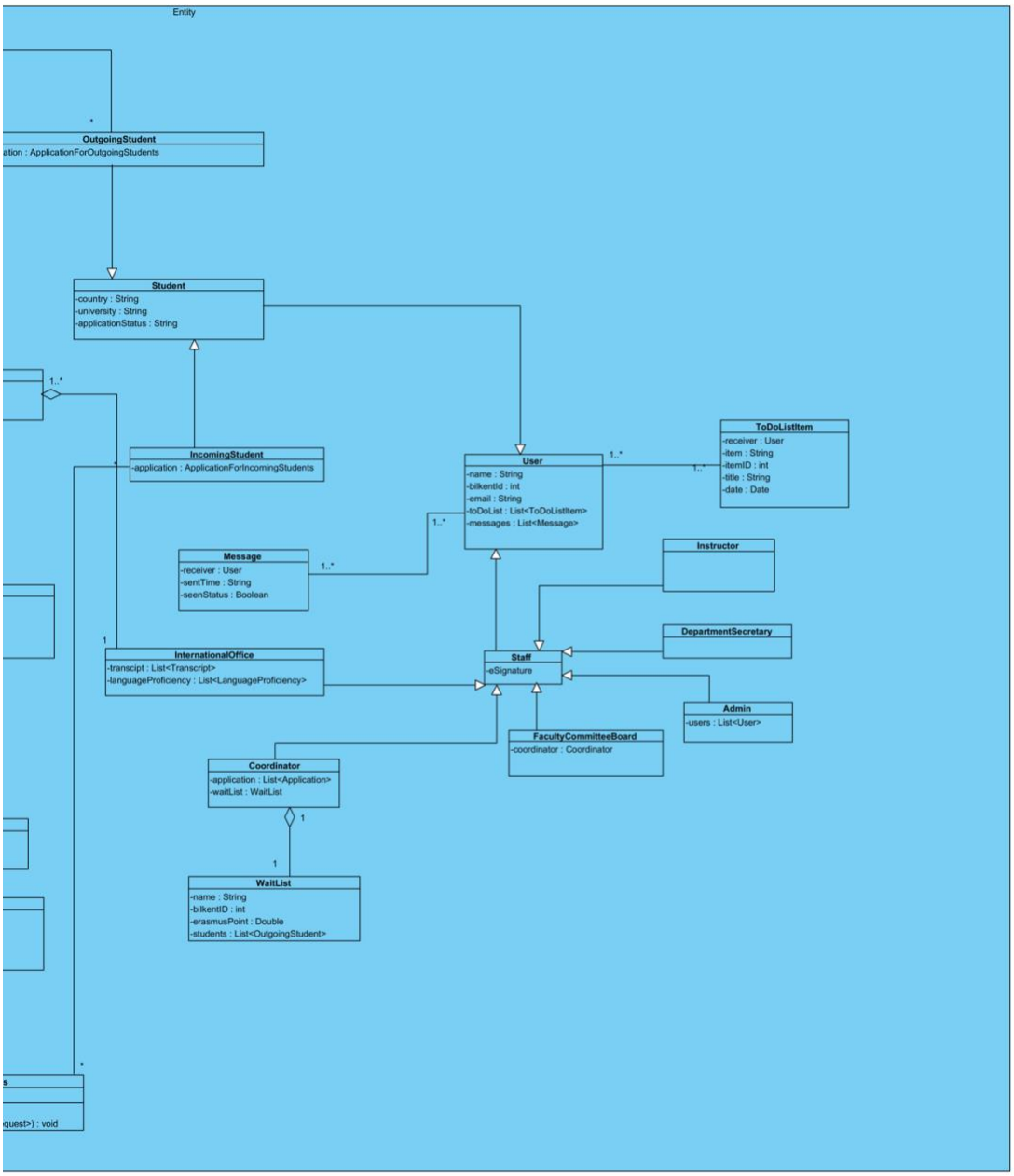

Figure 3.3.3: Right side of Entity Classes

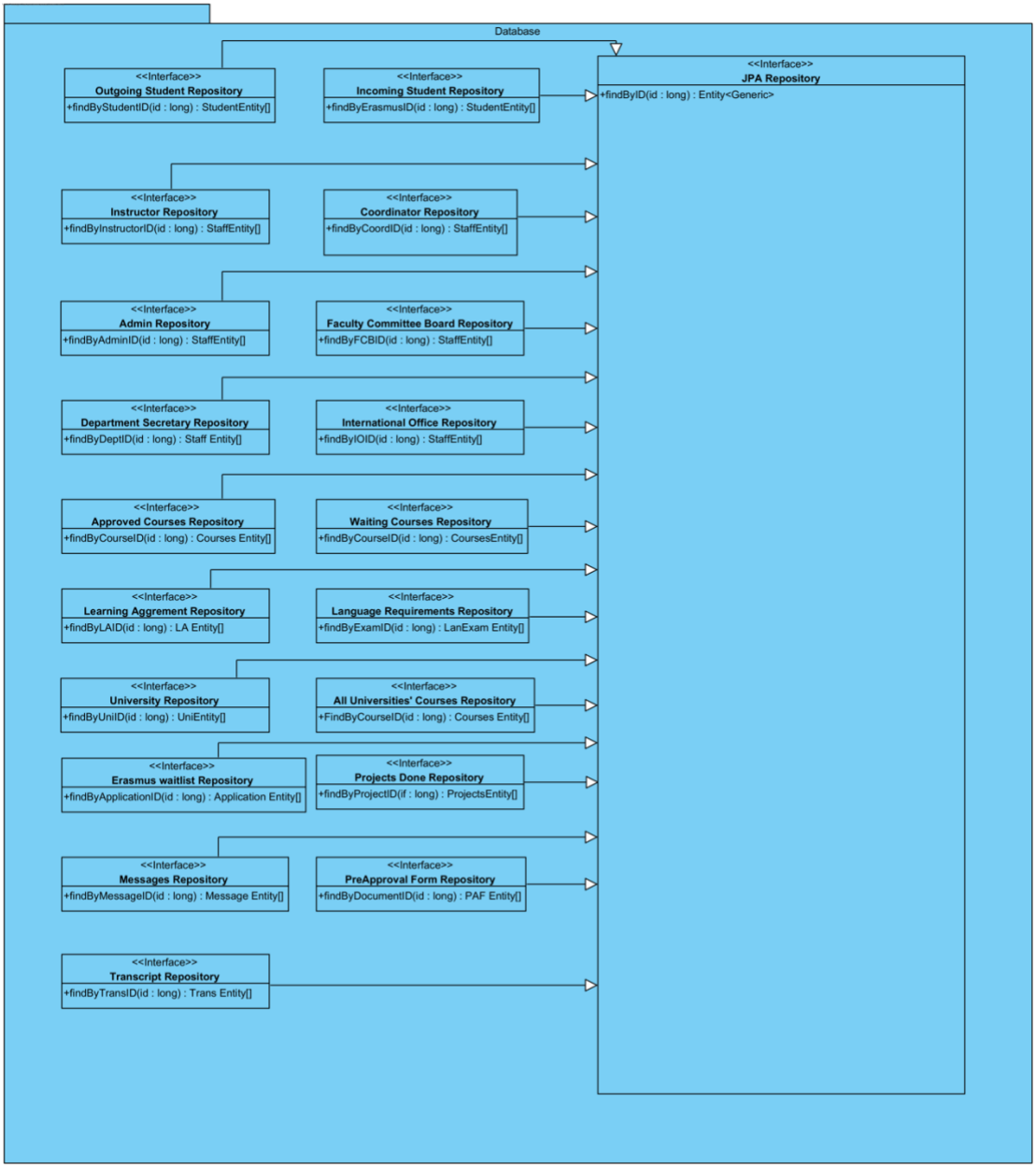

Figure 3.3.3: Data Management Layer

This layer represents the Data Management Layer and it consists of the database. This layer handles all requests and changes of the database by inheriting the JPA Repository in its repository interfaces.

3.4. External Packages

3.4.1. Frontend

3.4.1.1. Vue-pdf

This package allows pdf files to be downloaded and uploaded on the website.

3.4.1.2. Vue-signature

This package allows users to sign documents with an e-signature.

3.4.1.3. Vue-upload-image

This package allows users to upload images on the website to update profile photos.

3.4.2. Backend

3.4.2.1. express

Express is a web application framework built to be used with Node JS.

3.4.2.2. express-subdomain

This package allows creating and managing subdomains and related routing.

3.4.2.3. http-errors

This package allows creating and handling error cases.

3.4.2.4. express-session

This package creates sessions once a user logs in, and stores session-related data separately for each user.

3.4.2.5. node-postgres

This package will be used to communicate with the postgresql database.

3.4.2.6. express-fileupload

This package will be used to handle the server-side of a file-upload task and save the files that are sent via a form.

3.4.2.7. pbkdf2-password

This package will be used to calculate hash values of passwords.

3.4.2.8. nodemailer

This package will be used to send automated emails.

3.5. Internal Packages

3.5.1. routes

The routes package contains router files that map API endpoints with controller functions.

3.5.2. public

This package contains separate sub-packages/sub-folders for static resources, such as css files and image/document resource files.

3.5.3. controllers

This package contains controller classes that are called by routers or other controllers. Files in this package call service classes when required, and are used to determine the structural logic of how the classes work.

3.5.4. services

This package contains service classes that handle the async tasks done in the background. Classes of this package are called by controllers when a task, that is generally long and takes much time, is chosen to be passed to the background services.

3.5.5. models

This package contains model classes that determine data that need to be stored for a class type, in a dedicated database table.

3.5.6. database

This package contains database-related classes that are used when querying database entries to insert, select, update, or delete data.

4. Glossary & references

[1] https://app.erasmus.bilkent.edu.tr/
[2] https://w3.bilkent.edu.tr/www/degisim-programlari/