Bilkent University

Department of Computer Engineering

CS319 Term Project

Project short-name: BilBoard

Section 3

Group 3a

Design Report Iteration 1

Project Group Member Names:

İlke Doğan	(21702215)
Hacı Çakın	(21802641)
Metehan Saçakçı	(21802788)
Muhammet Abdullah Koç	(21802832)
Aslı Dinç	(21802527)

Instructor: Eray Tüzün

28 November 2021

Table of Contents

1 Introduction **4**
- 1.1 Purpose of the system 4
- 1.2 Design goals 4
  - 1.2.1 Usability 4
  - 1.2.2 Rapid Development 4
  - 1.2.3 Functionality 4

2 Proposed System **5**
- 2.1 Subsystem Decomposition 5
- 2.2 Hardware/Software Mapping 6
- 2.3 Persistent Data Management 6
- 2.4 Access Control and Security 7
- 2.5 Boundary Conditions 7
  - 2.5.1 Initialization 7
  - 2.5.2 Termination 7
  - 2.5.3 Failure 7

3 Low-Level Design **8**
- 3.1 Object Design Trade-offs 8
- 3.2 Final Object Design 9
- 3.3 Packages 12
  - 3.3.1 Packages Introduced by Developers 12
    - 3.3.1.1 controller 12
    - 3.3.1.2 entity 12
    - 3.3.1.3 repository 12
    - 3.3.1.4 requestModel 12
    - 3.3.1.5 responseModel 12
    - 3.3.1.6 security 12
    - 3.3.1.7 service 12
    - 3.3.1.8 serviceImplementation 12
    - 3.3.1.9 dto 12
    - 3.3.1.10 util 12
    - 3.3.1.11 exception 13
    - 3.3.1.12 resources 13
  - 3.3.2 External Library Packages 13
    - 3.3.2.1 springframework.data.jpa 13
    - 3.3.2.2 springframework.security 13
    - 3.3.2.3 io.jsonwebtoken 13
    - 3.3.2.4 springframework.mysql 13
    - 3.3.2.5 javax.mail 13
    - 3.3.2.6 org.modelmappeer 13
- 3.4 Class Interfaces 13

3.4.1 User Interface Management Layer 13
3.4.2 Web Server Layer 16
3.4.3 Data Management Layer 17

4. Glossary & references 20

1 Introduction

1.1 Purpose of the system

This application provides the users with the opportunity to access and communicate with all the clubs established within the framework of the school and enables them to follow the developments organized by the clubs in an interactive, instantaneous manner. Besides being an application where the activities, projects, and participants of the clubs are kept, it is aimed and offered to provide ease of use to all actors in the club network by offering specific pages for each actor of the application. This application has adopted the principle of appealing to all actors with unique features such as users' access to events and related information, access to club management's accounts via the website application, event status arrangements, and event authorisation and editing page that the student office directly manages.

1.2 Design goals

This application offers a user-friendly, reliable, and secure interface with functionality, maintainability, and scalability to appeal to all kinds of users.

1.2.1 Usability

It is a priority to accommodate a user-friendly interface so as to provide multi-dimensional ease of operation and intelligibility to various users. Easy-to-read font, annotated button/text field service, a comfortable transition between tabs, and straightforward access by everyone are offered. Being a single-page application will ensure that users are responsive when browsing the application.

1.2.2 Rapid Development

It is planned to develop the project within three months. This process should be managed meticulously and the task distribution should be given appropriately. Additionally, rapid prototyping of the product, which focuses on the feedback of the target user group and adopts agile software, is aimed.

1.2.3 Functionality

It is intended to be practised for numerous user types. Users can perform certain functional actions within the application. The application has an extensible feature in itself and more functionality can be added as needed in the later stages of the project. Restriction states that customize specific tasks of more than one user and that can limit access to confidential information contribute to the functionality.

2 Proposed System

2.1 Subsystem Decomposition

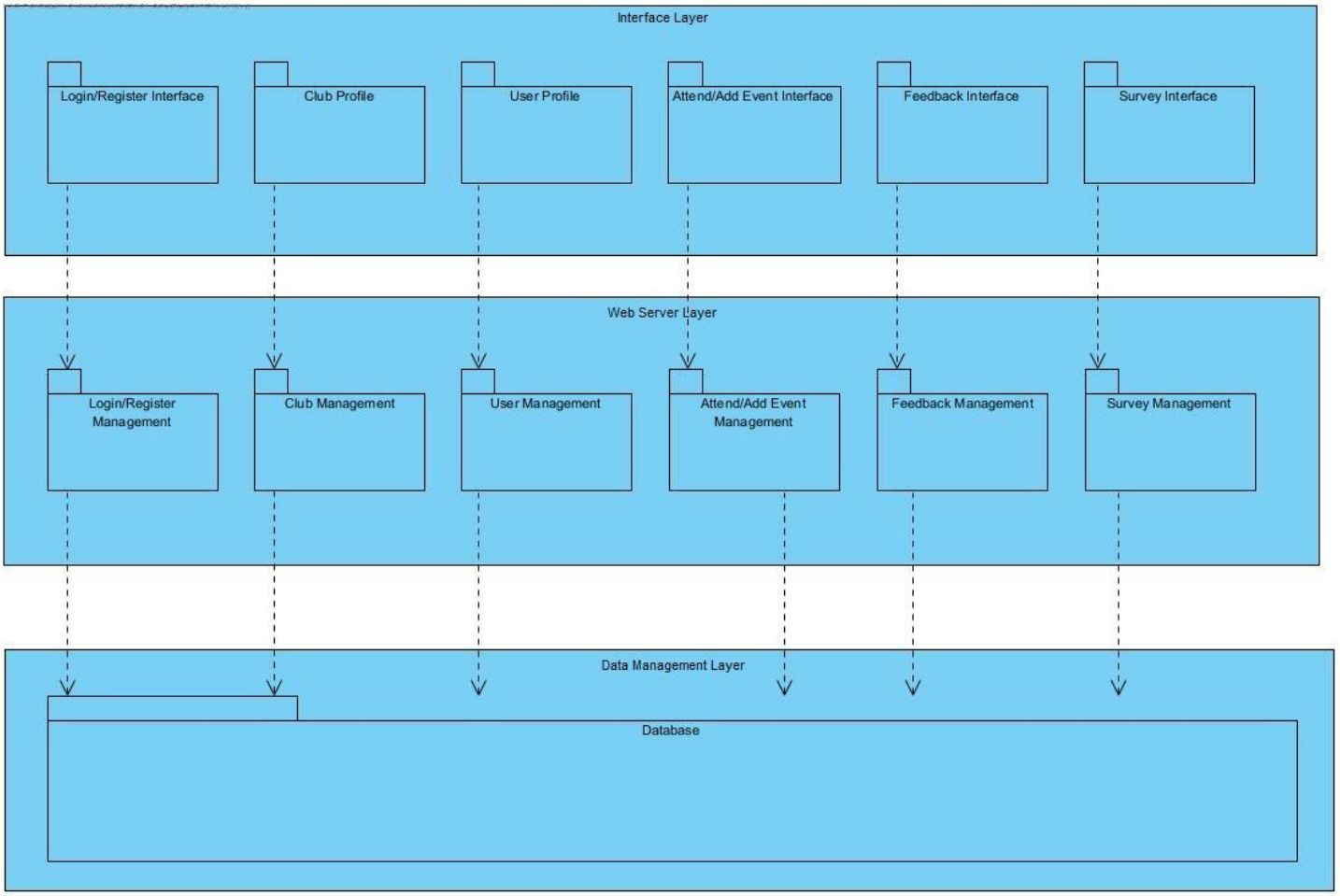

Figure 1: Subsystem Decomposition

To create an area to maintain our project, we decided to use three-layer architecture.

First of all, the UI Interface Layer is designed as a boundary area that contains our web application pages. It should be seen that every user will interact with our

application using these pages. The main reason for this approach is to make our application much more usable.

Secondly, the Web Server Layer is designed to carry out all of the backend operations. The diagram also shows that there are different systems to manage different operations in our application.

Thirdly, the Data Management Layer is designed to communicate with the database to store all necessary data.

2.2 Hardware/Software Mapping

Since we are developing a project to run on the web, our project does not require any special hardware components. However, it should be underlined that the system where our project will be run should handle standard web browsers. Also, we are expecting to run our project on Google Chrome, Mozilla Firefox, Internet Explorer, Opera and Safari with the latest updated version. We are mainly aiming that our project should work perfectly on personal computers. That is why standard components of computers such as a mouse, keyboard, monitor are essential in order to use our application.

2.3 Persistent Data Management

Since our application is a web application, we decided to use cloud options, not local ones. At this point, we had two options that we could prefer. One of them was SQL(relational databases), the other one was NoSQL(not relational databases). Because none of us knows Graph based databases, we haven’t considered it. Relational databases are suitable when the system object has lots of many-to-many relationships. However, NoSQL databases are a better option when we do not have lots of queries and relationships. In NoSQL, lots of the properties of the objects are embedded in a single entity object. Unlike this, SQL has rows and objects construct relations via their ids. Performing more queries makes SQL a better option for us. Moreover, because our data is structured, SQL takes advantage. If our schemas or

models were not exact, NoSQL would be a good alternative. As a technology, we chose to use MySQL because members of our team are familiar with it.

2.4 Access Control and Security

To achieve security within the Bilkent system, Bilkent offers the option to register users only in Bilkent by accepting Bilkent emails. For the sake of security, the passwords of users will be encrypted. All the users' requests which will be handled on the client-side will be authorized by using JSON Web Token(JWT)[2]. Each JWT will be created when the user logs in and will be valid for 1 hour. Also, whole requests which are not originated from the front-end will be responded with on CORS error [3]. All the passwords must be at least 8 characters.

2.5 Boundary Conditions

2.5.1 Initialization

Because our application is designed as a web application, it will be run on a web server at any time, except the server shuts down the application. That is why if a user with an internet connection is found on the application’s database, he/she will be allowed to the main screen of the program. It is also allowed that users can join the application from different devices.

2.5.2 Termination

Our application will be terminated in the case that any subsystem crashes. The main reason for this approach is to prevent any problematic operation from happening when a subsystem is down. Also, before the termination process starts all of the last saved data will be saved to the database to prevent data loss.

2.5.3 Failure

Paas products of AWS(open form) particularly [4], Elastic Beanstalk [5] and RDS(Relational Database System) [6] can be used to track and record failures. Also, some products can restart the entire environment.

There is also the possibility that failure might not be resolved with restart operation. In this case, the developer will be notified. Our application service layer has an HTTP response implementation for bugs that come from problematic operations.

3 Low-Level Design

3.1 Object Design Trade-offs

Functionality v. Usability: Our application can be used by students, student club board members, administrative assistants, and instructors. Also, many functions are unique for these user types. This makes the application functional. On the other hand, the application and interface are easy to use. We tried to keep a balance between functionality and usability. However, because the application focuses on students' use, we focused more on usability.

Rapid development v. Functionality: Because of time limitations on the development of the application, rapid development is one of the main concerns in our application. This time constraint causes the application to be less functional.

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.

3.2 Final Object Design

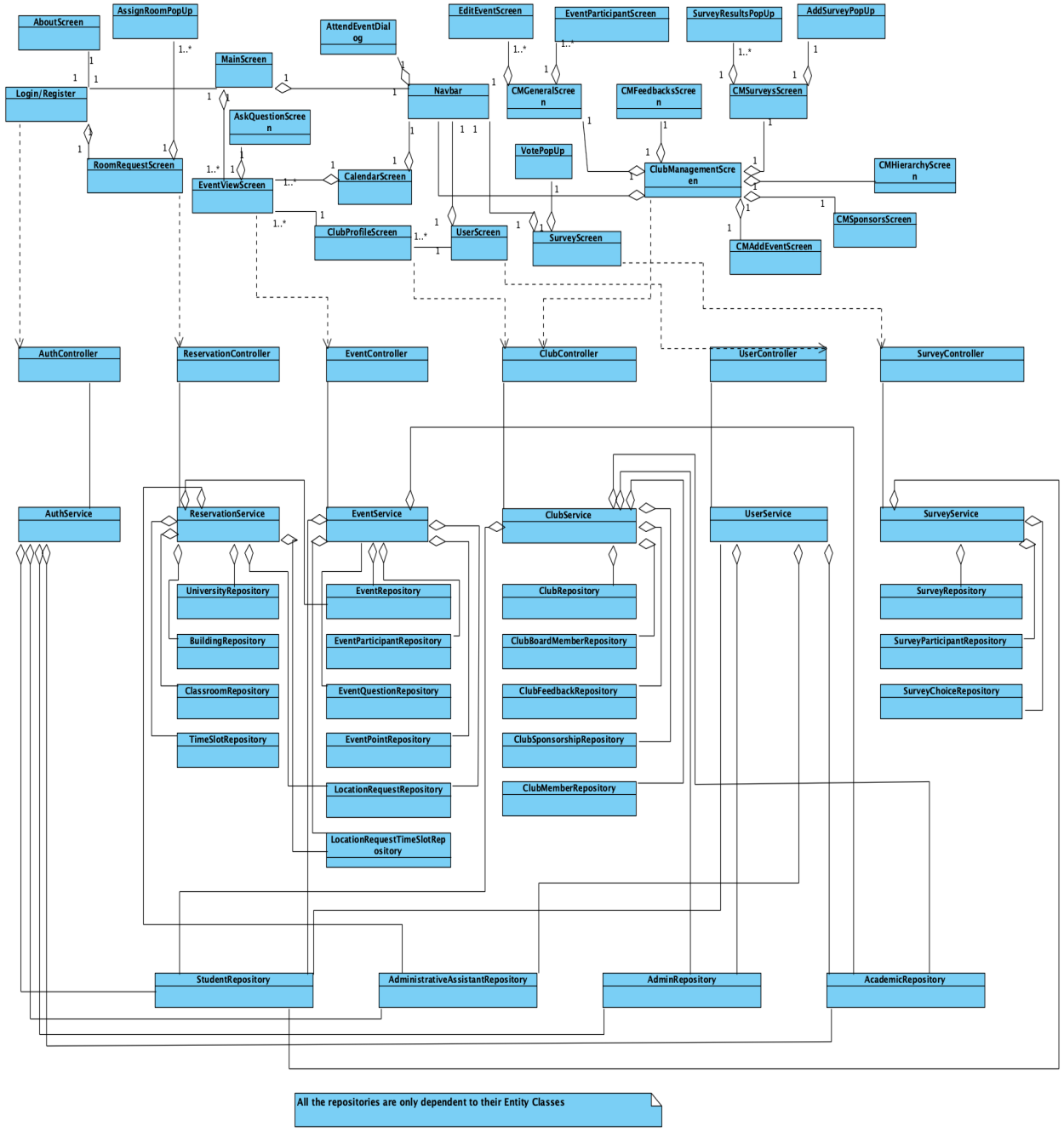

Figure 2: Final Object Design

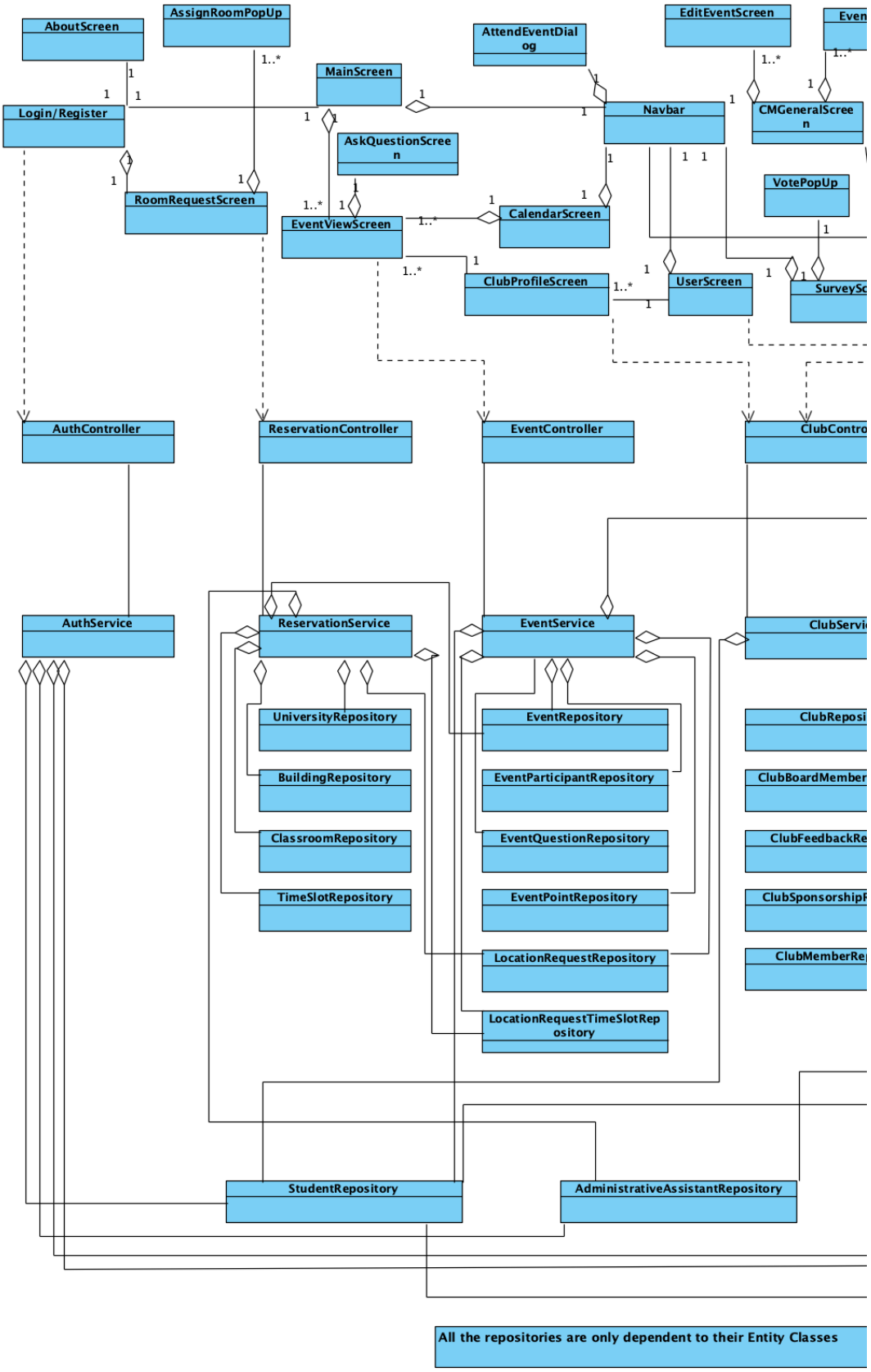

Figure 3: Left Part of the Final Object Design

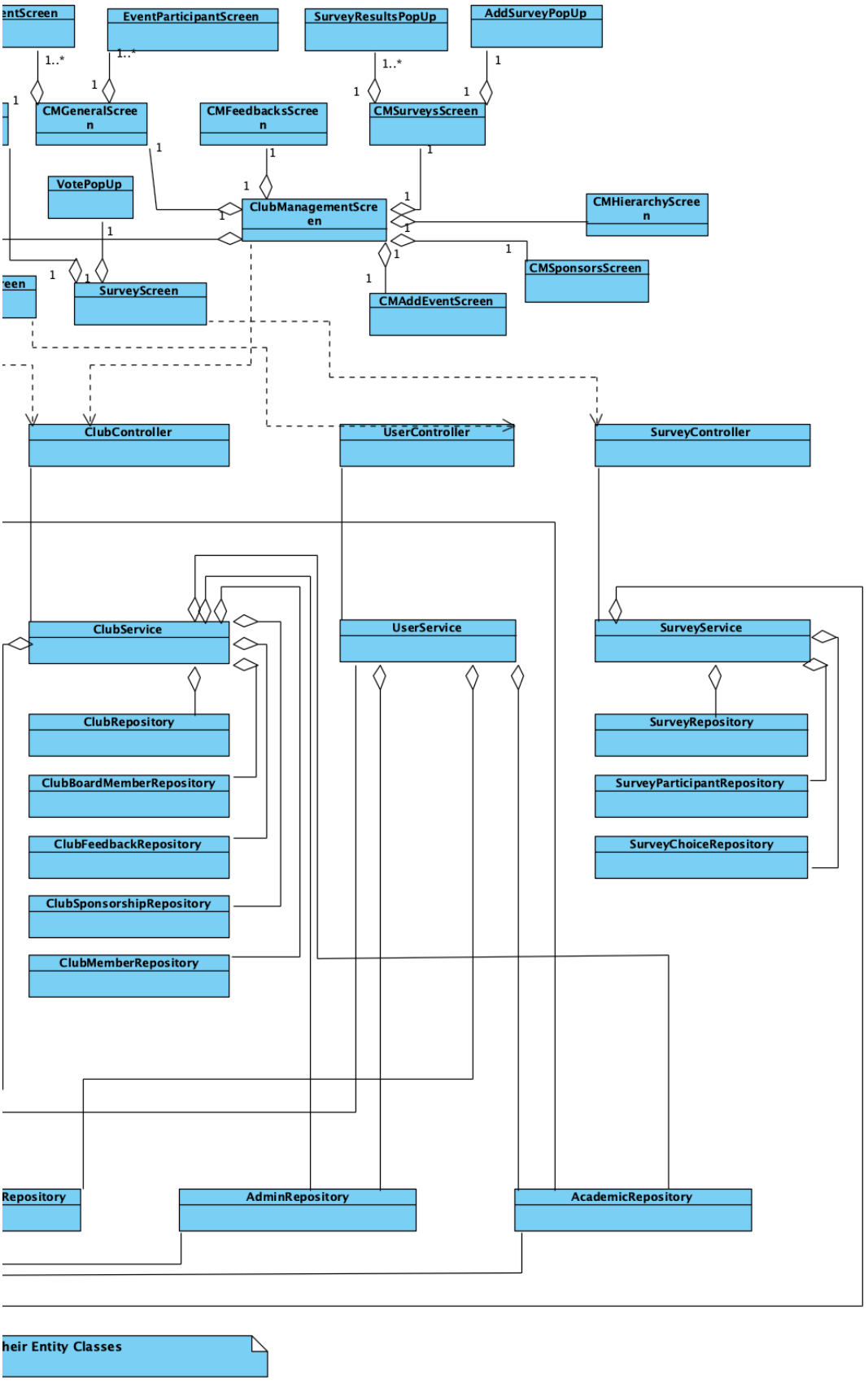

Figure 4: Right Part of the Final Object Design

3.3 Packages

3.3.1 Packages Introduced by Developers

3.3.1.1 controller

This package has all the controller classes that handle requests coming from the clients.

3.3.1.2 entity

The entity package has all the entity classes that are the base of the database.

3.3.1.3 repository

This package has all the repository interfaces that perform queries and communicate with tables.

3.3.1.4 requestModel

This package has all models that request bodies have.

3.3.1.5 responseModel

This package has all models that response data is constructed.

3.3.1.6 security

This package has security-related classes that perform access limitation, authorization, CORS policy etc.

3.3.1.7 service

This package has all service interfaces that help controllers to handle requests.

3.3.1.8 serviceImplementation

This package has all service implementations that have implementations of all functions in its interfaces.

3.3.1.9 dto

This package has all dto objects that are helping communicate between layers.

3.3.1.10 util

This package has classes that help other classes by utilizing some functionalities.

3.3.1.11 exception

This package has common exceptions that can occur in runtime.

3.3.1.12 resources

This package has environment variables that are used in the program.

3.3.2 External Library Packages

3.3.2.1 springframework.data.jpa

This library makes it easy to implement JPA based repositories and queries. This module deals with enhanced support for entity data access layers.

3.3.2.2 springframework.security

This module provides the implemented spring functionalities like authentication, session trace, defining public accessible endpoints or password encoding and decoding.

3.3.2.3 io.jsonwebtoken

This library helps to construct, encode and decode JWT to trace user authorization.

3.3.2.4 springframework.mysql

This implemented library helps to construct a connection with the MySQL database.

3.3.2.5 javax.mail

This package provides IMAP, SMTP objects and functions to handle email traffic.

3.3.2.6 org.modelmappeer

This package helps to make object transactions between layers

3.4 Class Interfaces

3.4.1 User Interface Management Layer

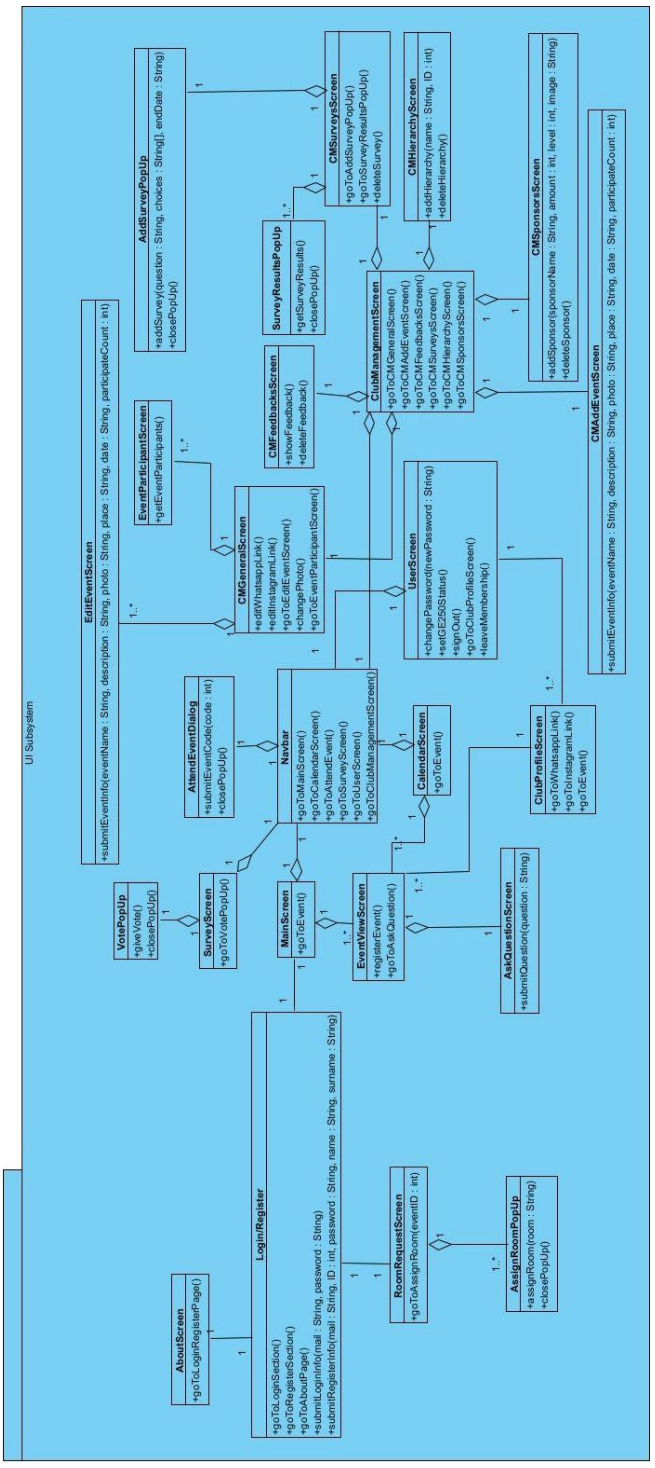

Figure 5: User Interface Management Layer Composition

This layer is a bridge between the user and bottom layers. In this layer, all classes are JavaScript files with the React library. The classes in this layer communicate with Web Server Layer's classes. Classes in this layer use components that can easily be modified. The diagram is divided into following pages for easy reading.

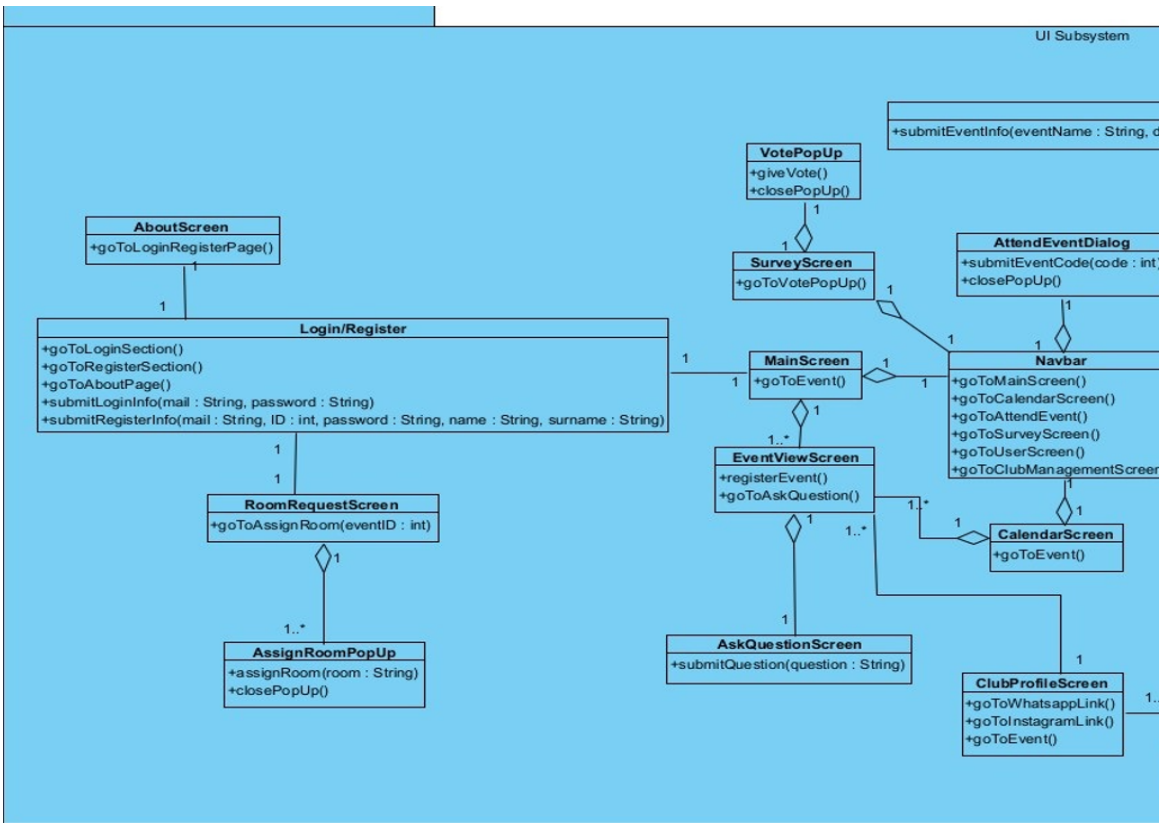

Figure 6: Left Part of the User Interface Management Layer Composition

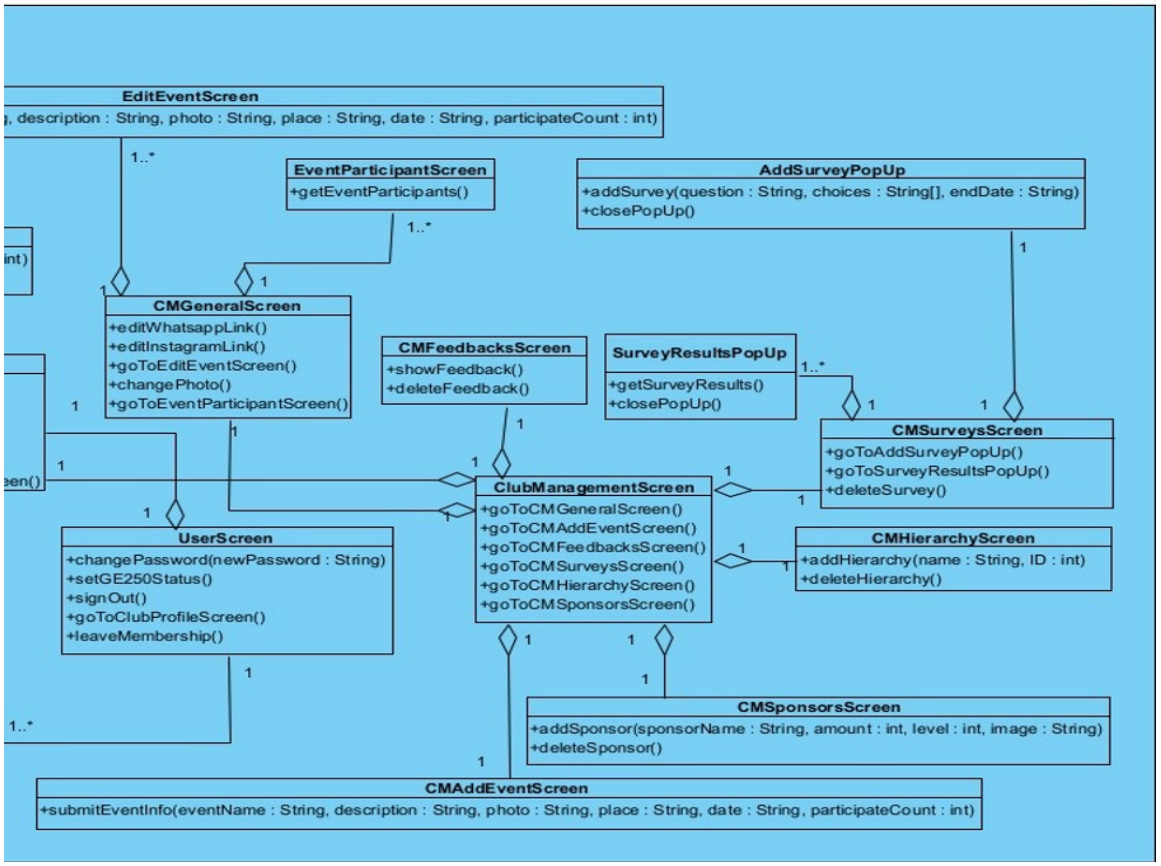

Figure 7: Right Part of the User Interface Management Layer Composition

3.4.2 Web Server Layer

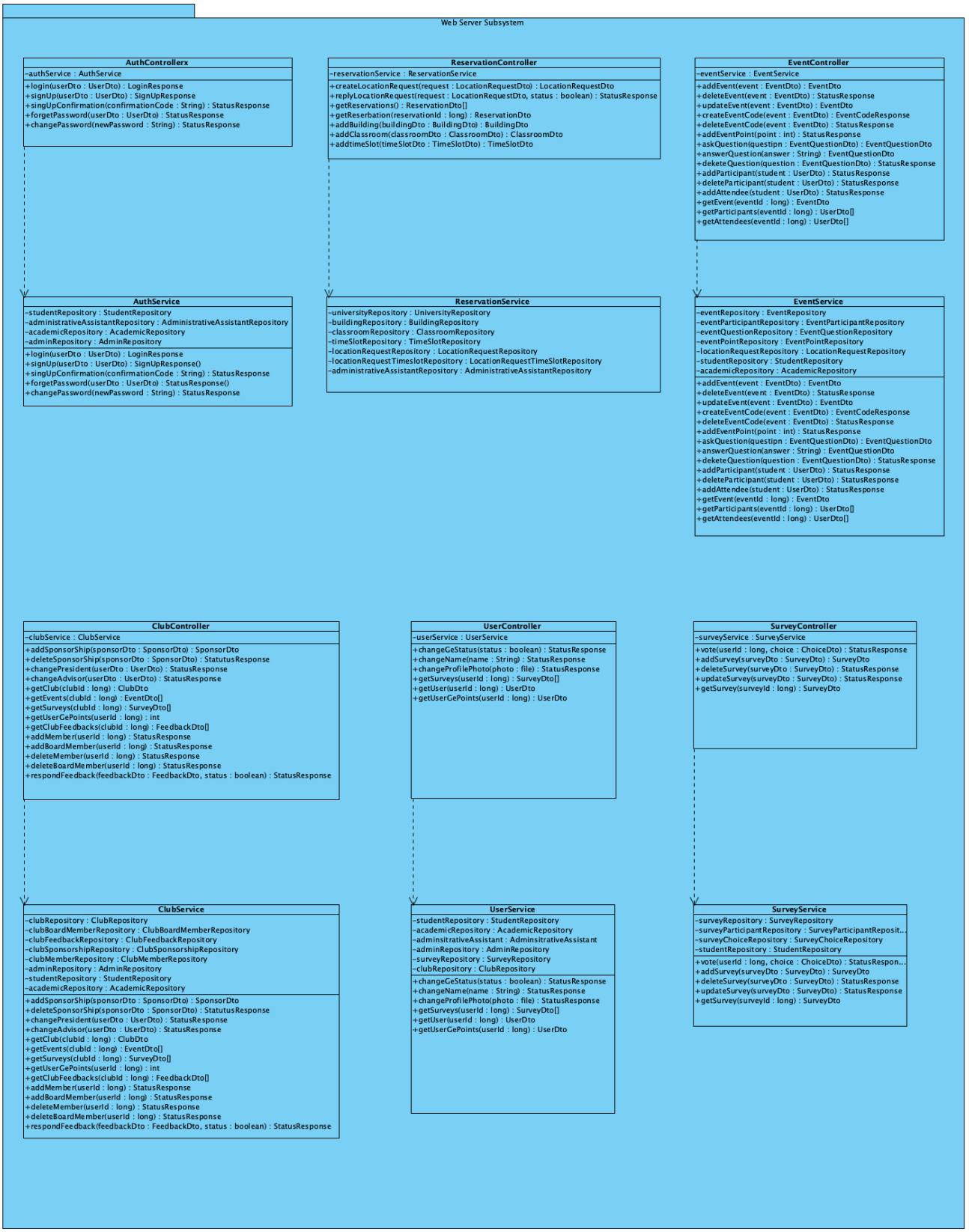

Figure 8: Web Server Layer Composition

This layer handles the request coming from the client and returns required responses. It constructs the communication between user interface and data management layer.

3.4.3 Data Management Layer

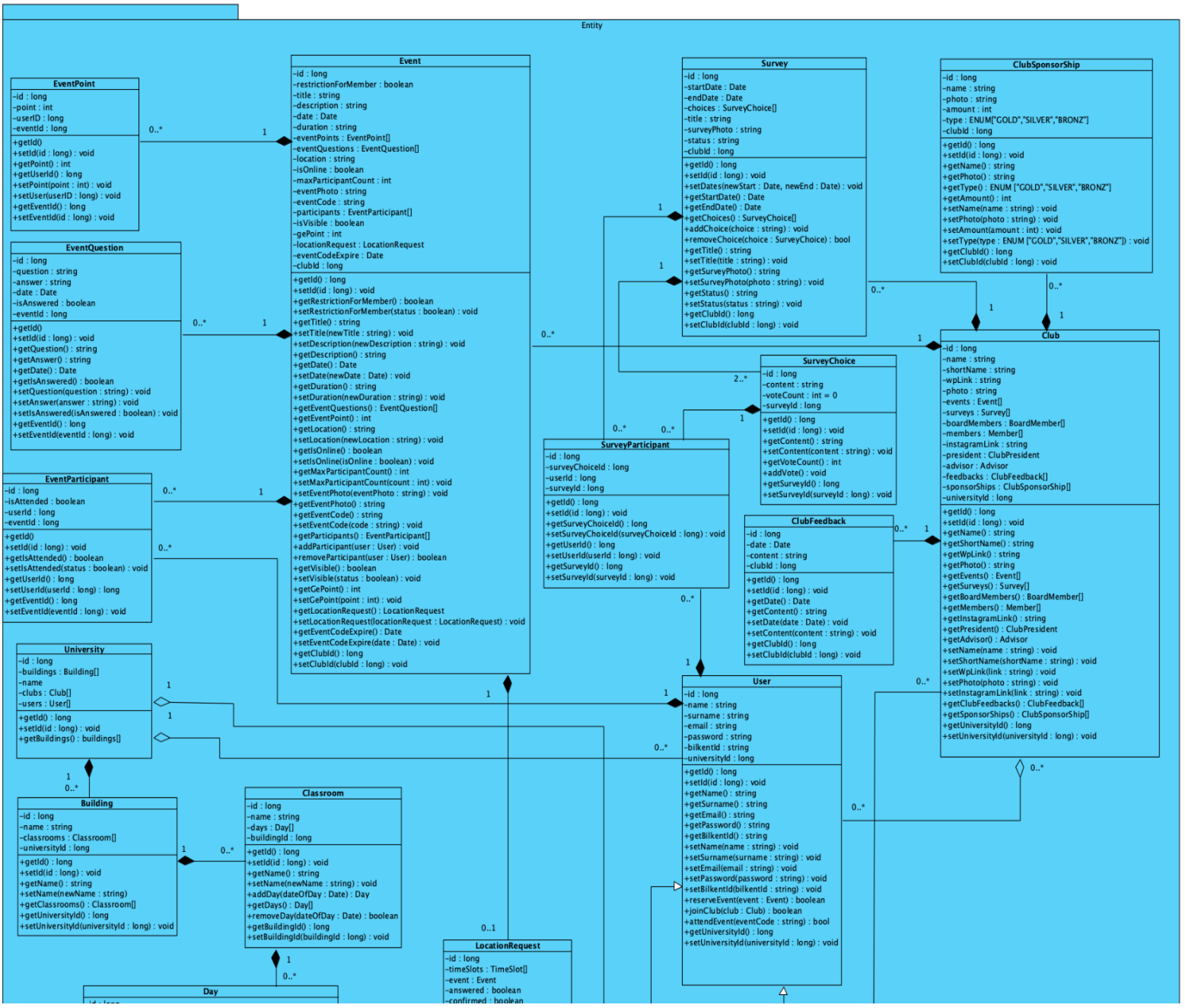

Figure 9: Upper Part of the Entity of the Data Management Layer Composition

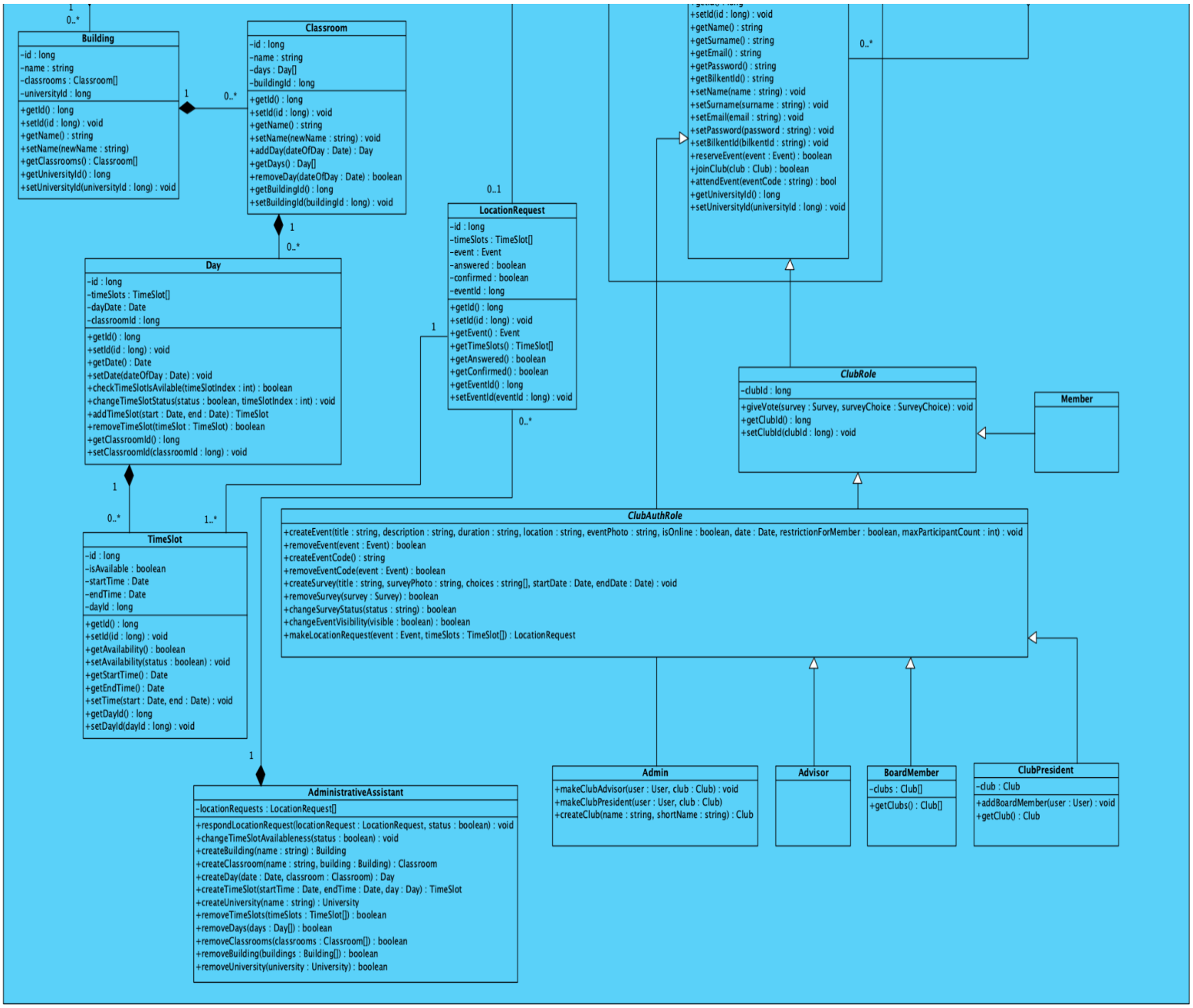

Figure 10: Bottom Part of the Entity of the Data Management Layer Composition

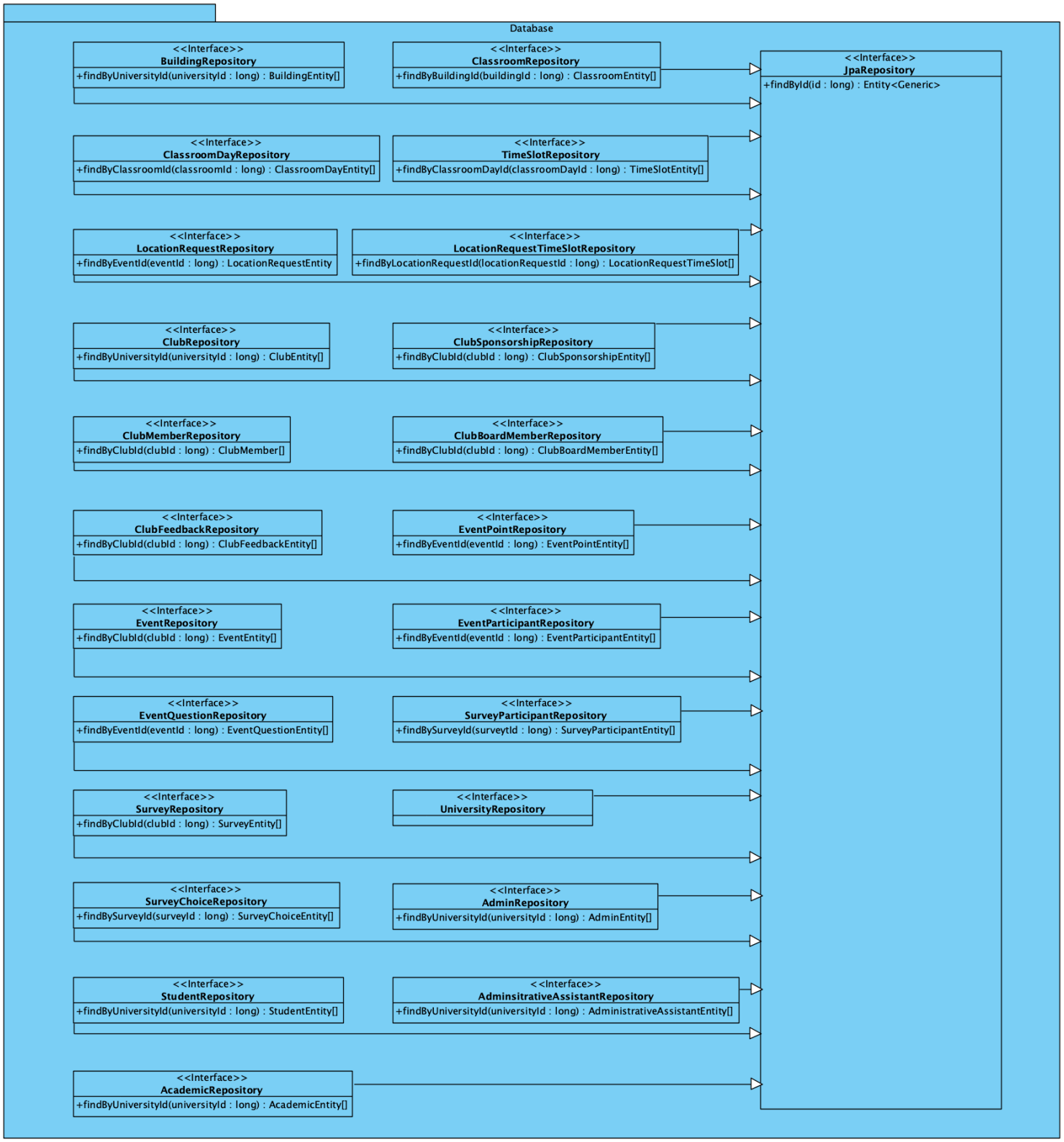

Figure 11: Repository Part of the Data Management Layer Composition

4. Glossary & references

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.
[2] "Introduction to JSON Web Tokens". [Online]. Available: https://jwt.io/introduction. [Accessed: 29 October 2021].

[3] "Cross-Origin Resource Sharing (CORS)". [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Accessed: 29 October 2021].

[4] K. Yadav, "What is AWS and What can you do with it". [Online]. Available: https://medium.com/@kunalyadav/what-is-aws-and-what-can-you-do-with-it-395b585b03c. [Accessed: 29 October 2021].

[5] "AWS Elastic Beanstalk". [Online]. Available: https://aws.amazon.com/elasticbeanstalk/?nc1=h_ls. [Accessed: 29 October 2021].

[6] B. Lutkevich , "Amazon RDS (Relational Database Service)". [Online]. Available: https://searchaws.techtarget.com/definition/Amazon-Relational-Database-Service-RDS. [Accessed: 29 October 2021].