**Часть 1:**

1. Описание проекта

/\*Проект “форум по программированию” направлен на создание платформы, где пользователи могут обмениваться знаниями, сотрудничать над проектами, задавать и отвечать на вопросы, связанные с программированием.\*/  
  
Проект направлен на создание форума по программированию, где пользователи могут обмениваться знаниями, сотрудничать над проектами, задавать и отвечать на вопросы, связанные с программированием. Пользователи должны будут зарегистрироваться и пройти аутентификацию для взаимодействия с приложением. Они могут создавать и управлять своими проектами, участвовать в обсуждениях. Форум будет иметь систему репутации, которая служит для поощрения пользователей к активному участию. Пользователи также могут иметь роли модераторов или администраторов для поддержания позитивной атмосферы в сообществе и улучшения общего качества контента. Проекты будут иметь теги которые помогают классифицировать и организовывать контент. У проекта может быть несколько тегов, которые могут характеризовать стек, на котором был он реализован, его тематику и т.д.. Они помогают пользователям легко искать и фильтровать контент, относящийся к определенным темам. Форум будет иметь систему поиска и фильтрации, которая поможет пользователям находить контент на основе названия проекта, тега или ключевого слова. Также форум будет иметь функции создания и управления проектами, позволяющие пользователям создавать новые проекты и связывать их с соответствующими тегами проекта. Пользователи могут задавать вопросы по проекту, а другие пользователи могут давать ответы на эти вопросы - по сути создавать топик обсуждения. Вопросы привязываются к соответствующему проекту. Может быть реализована система уведомлений для оповещения пользователей о новой активности, связанной с их проектами или вопросами, которые они задали или на которые ответили. Наконец, ответу может быть присвоен рейтинг, который поможет другим оценить полезность ответа на конкретный вопрос.

1. Типовые ИТ процессы:

* Сбор (производится с помощью форм библиотеки “sweetalert2”, а также вручную написанных HTML форм. Примеры: создание обсуждения, проекта, комментария)
* Передача (Frontend взаимодействует с Backend API посредством HTTP запросов)
* Обработка (производится с помощью сервисов NestJs, а также библиотеки “class-validator”)
* Хранение (данные сохраняются в сущности PostgreSQL, через обращение к репозиториям внутри сервисов, посредством TypeORM маппинга)
* Предоставление (Все данные отрисовываются в браузере с использованием HTML, CSS и JS, страницы рендерятся на сервере с помощью HandleBars шаблонизации)

1. Ключевые информационные объекты, формируемые и обрабатываемые в рамках проекта

/\*

* Аккаунт пользователя, включающий никнейм, полное имя, e-mail, пароль, аватарку.
* Проект, включающий название, описание, ссылку на гитхаб и тэг.
* Данные об обсуждениях, комментариях и репутации

\*/

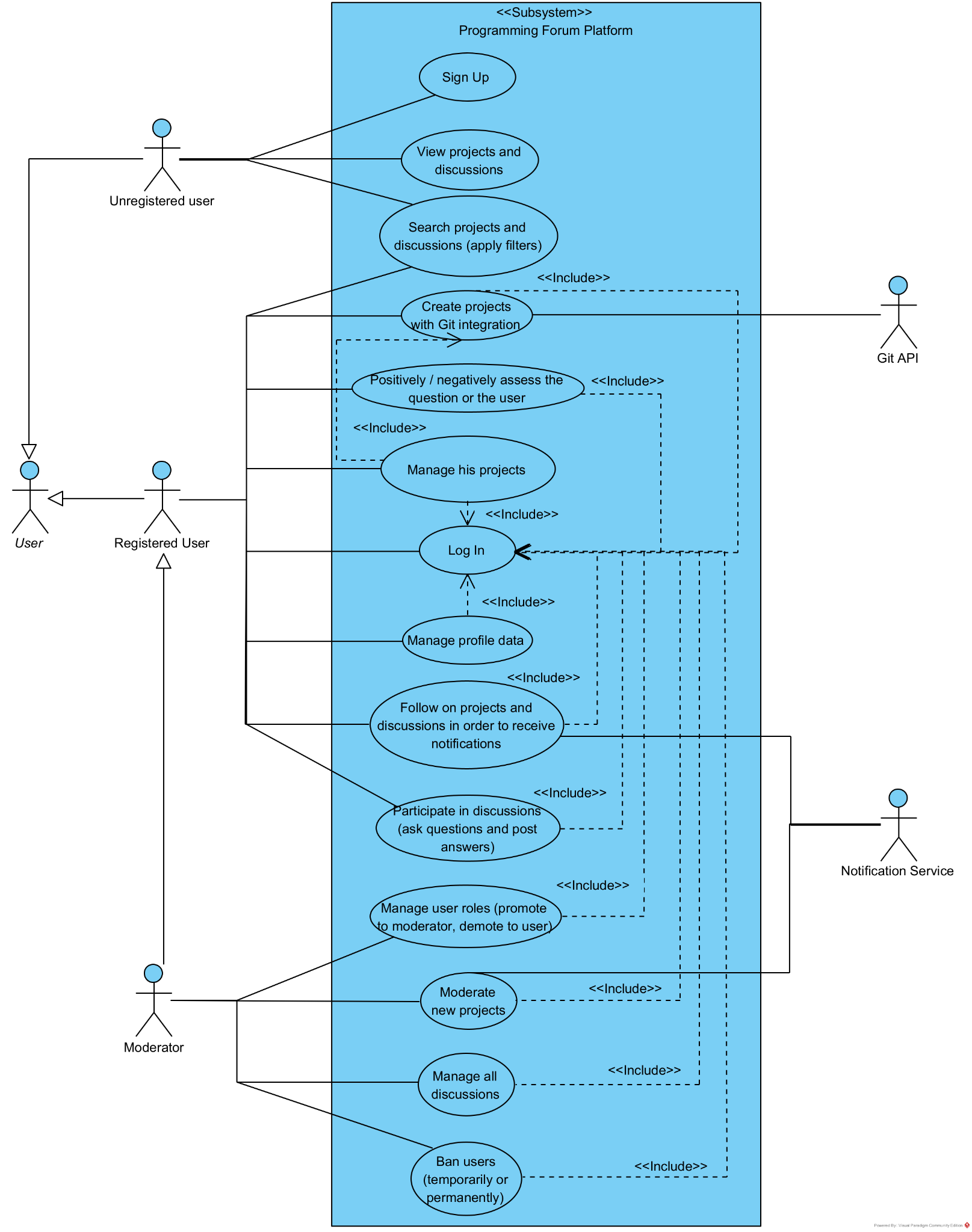
* Пользователь - лицо, зарегистрировавшее учетную запись на платформе и имеющее доступ к полному спектру функциональных возможностей, таких как создание и управление проектами, участие в обсуждениях, создание и ответы на вопросы, оценка комментариев и пользователей.
* Модератор - пользователь, которому предоставлены специальные привилегии для модерирования платформы и обеспечения качества контента. Они могут иметь дополнительные функции, такие как возможность редактировать или удалять контент, предупреждать или блокировать пользователей, нарушающих правила сообщества.
* Незарегистрированный пользователь - человек, который не вошел в систему и имеет ограниченный доступ к платформе, например, может просматривать контент, но не может создавать и управлять проектами или участвовать в обсуждениях.
* Проект - набор информации, описывающий проект по программированию, включая его название, описание, участников, используемые инструменты и другие детали, которые автор проекта считает важными.
* Метка проекта - метка или ключевое слово, которое присваивается проекту для его классификации на основе предметной области, технологического стека или других соответствующих атрибутов.
* Тема обсуждения - набор сообщений, относящихся к определенному проекту или теме, где пользователи могут получать ответы на вопросы или обсуждать какие-либо темы.
* Комментарий - сообщение, сделанное пользователем в ответ на вопрос, содержащее решение, объяснение или дополнительную информацию.
* Репутации - оценка, которая вознаграждает пользователей за положительный вклад, например, за предоставление полезных ответов или отзывов.
* Роль пользователя - уровень полномочий, присвоенный пользователю, который определяет его разрешения и обязанности на платформе. На данный момент включает в себя 2 уровня доступа – обычный пользователь и модератор.
* Уведомлений - отправляются пользователю, когда кто-то отвечает на один из его вопросов или комментариев, или когда проект, за которым он следит, обновляется.

1. Функциональные требования

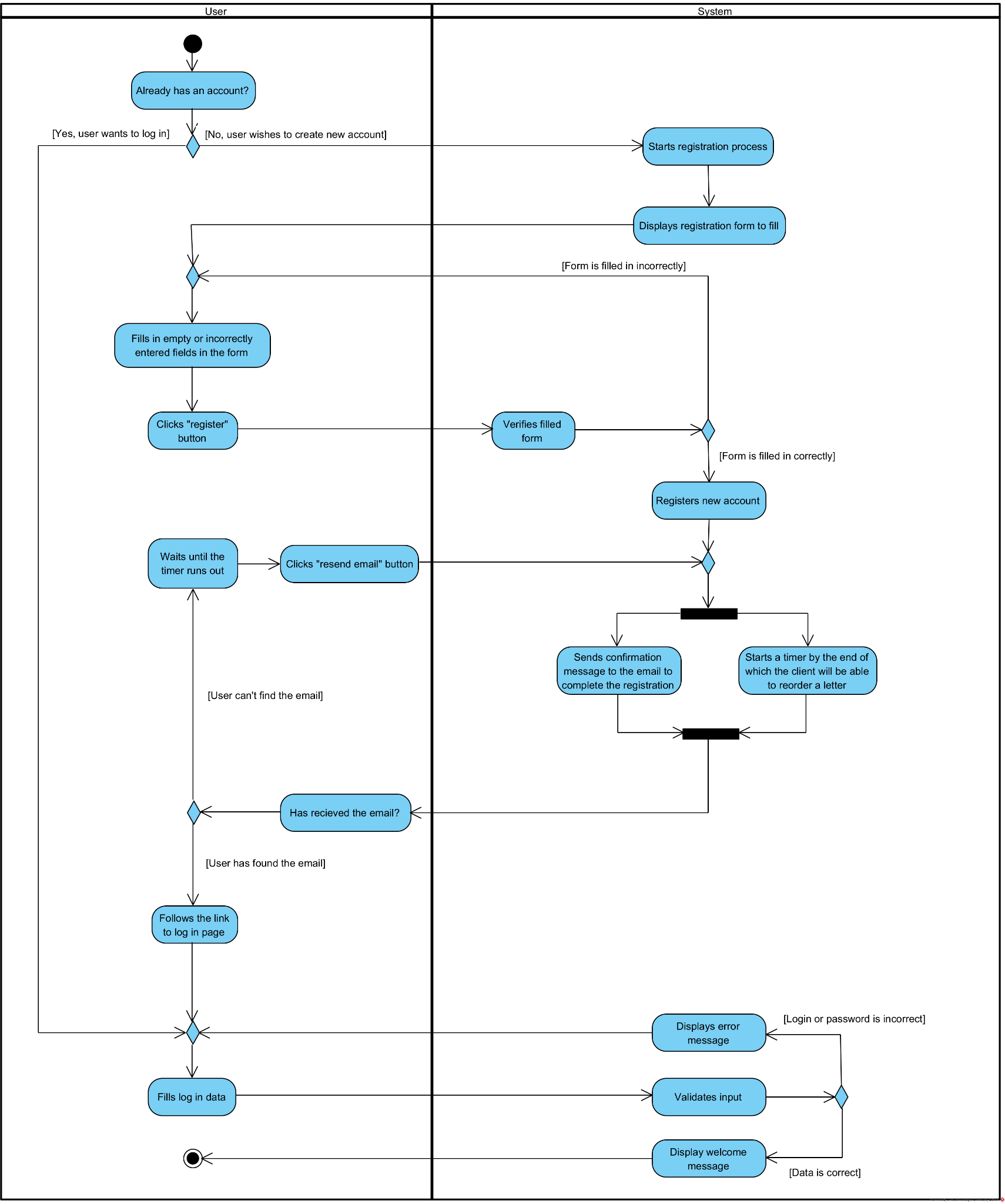
* **Аутентификация пользователей**  
  Пользователи должны иметь возможность зарегистрировать учетную запись на платформе и пройти аутентификацию для доступа к полному спектру функциональных возможностей.
* **Создание и управление проектами**   
  Пользователи должны иметь возможность создавать новые проекты и управлять содержанием и разрешениями для своих собственных проектов.
* **Дискуссионные темы**   
  Пользователи должны иметь возможность создавать и участвовать в дискуссионных темах, связанных с конкретными проектами.
* **Система вопросов и ответов**  
  Пользователи должны иметь возможность задавать вопросы, связанные с программированием, и отвечать на них, привязываясь к соответствующему проекту или теме.
* **Система репутации**  
  Платформа должна иметь систему репутации, которая вознаграждает пользователей за положительный вклад, например, за предоставление полезных ответов или отзывов. Эта система может включать такие функции, как повышение или понижение рейтинга ответов, присвоение значков или баллов за определенные достижения или выделение пользователей с высокой репутацией.
* **Роли пользователей**   
  Платформа должна иметь различные роли пользователей с разными уровнями полномочий, например, модераторы, чтобы обеспечить хорошее содержание контента и сообщества.
* **Теги и фильтрация**  
  Проекты и обсуждения должны быть организованы и доступны для поиска на основе тегов, которые могут характеризовать стек, тематику или другие аспекты контента.
* **Уведомления**  
  Пользователи должны иметь возможность получать уведомления о новой активности, связанной с их проектами или вопросами, которые они задали или на которые ответили.
* **Интеграция с Git**  
  Платформа должна иметь возможность отслеживать состояние репозитория и своевременно обновлять информацию о проекте.
* **Функциональность поиска**   
  Платформа должна иметь систему поиска и фильтрации, которая позволяет пользователям легко находить релевантный контент на основе названия проекта, тега или ключевого слова.
* **Профиль пользователя и настройки**  
  Пользователи должны иметь возможность просматривать и управлять информацией своего профиля.

1. Нефункциональные требования

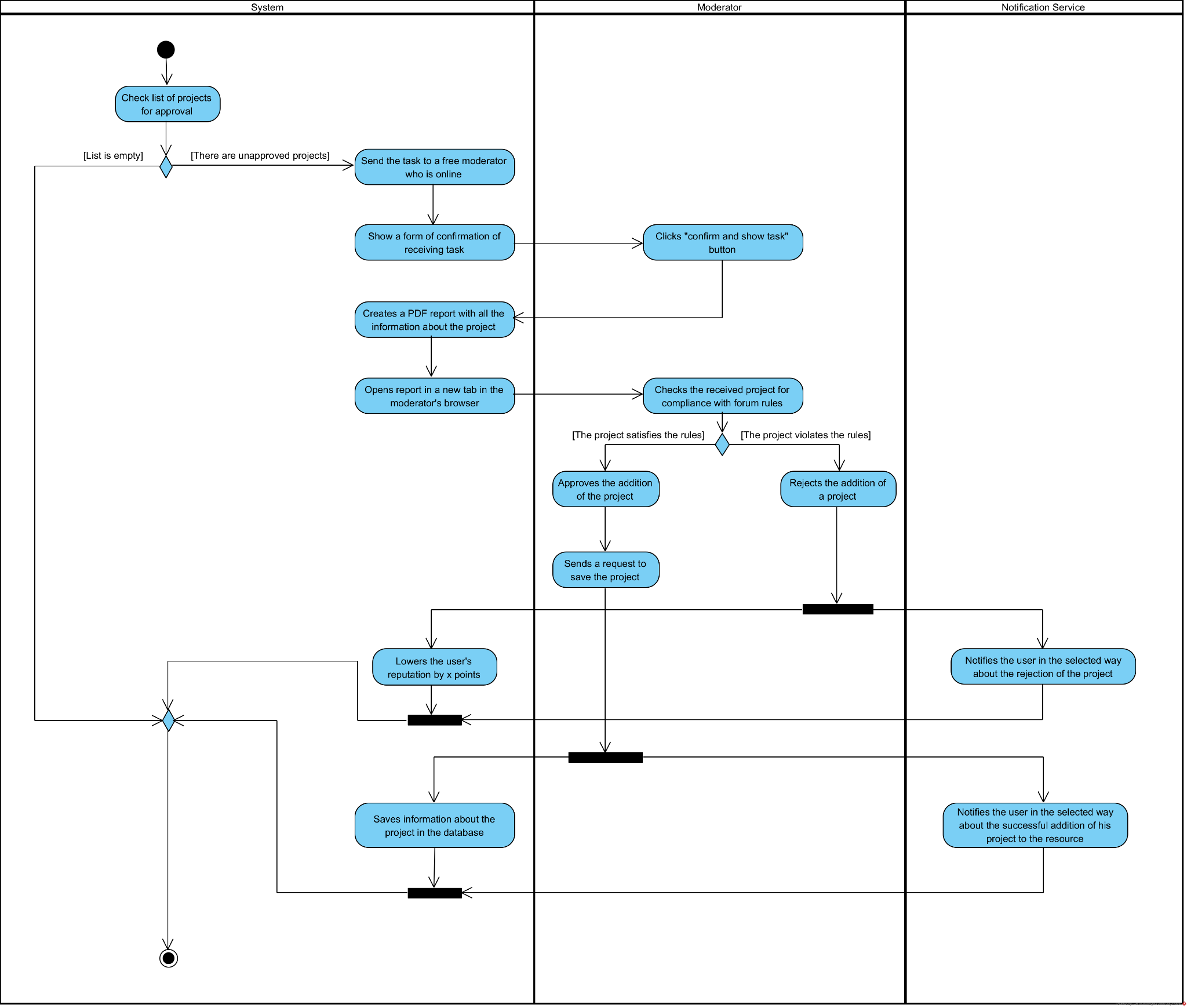
* **Доступность**  
  Платформа должна быть отзывчивой и хорошо работать на различных устройствах и размерах экрана, то есть на основных платформах (web, mobile).
* **Язык**   
  Платформа должна поддерживать переключение между двумя языками (русский, english)
* **Безопасность**   
  Платформа должна иметь соответствующие меры безопасности для защиты пользовательских данных и предотвращения несанкционированного доступа и атак.

1. Use-Case Diagram  
   
2. Activity Diagram

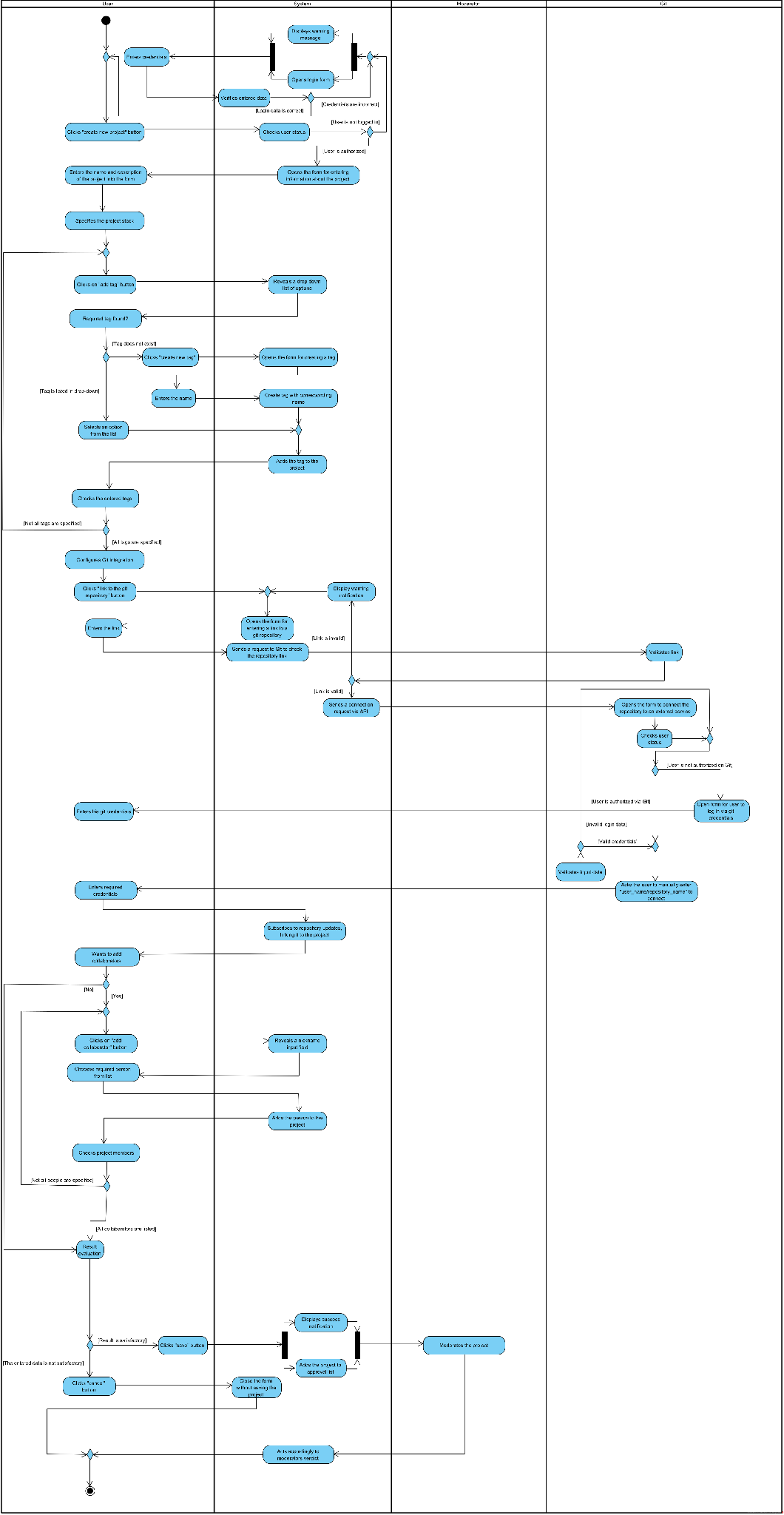
Процесс “авторизация пользователя”



Процесс “модерация созданных проектов”

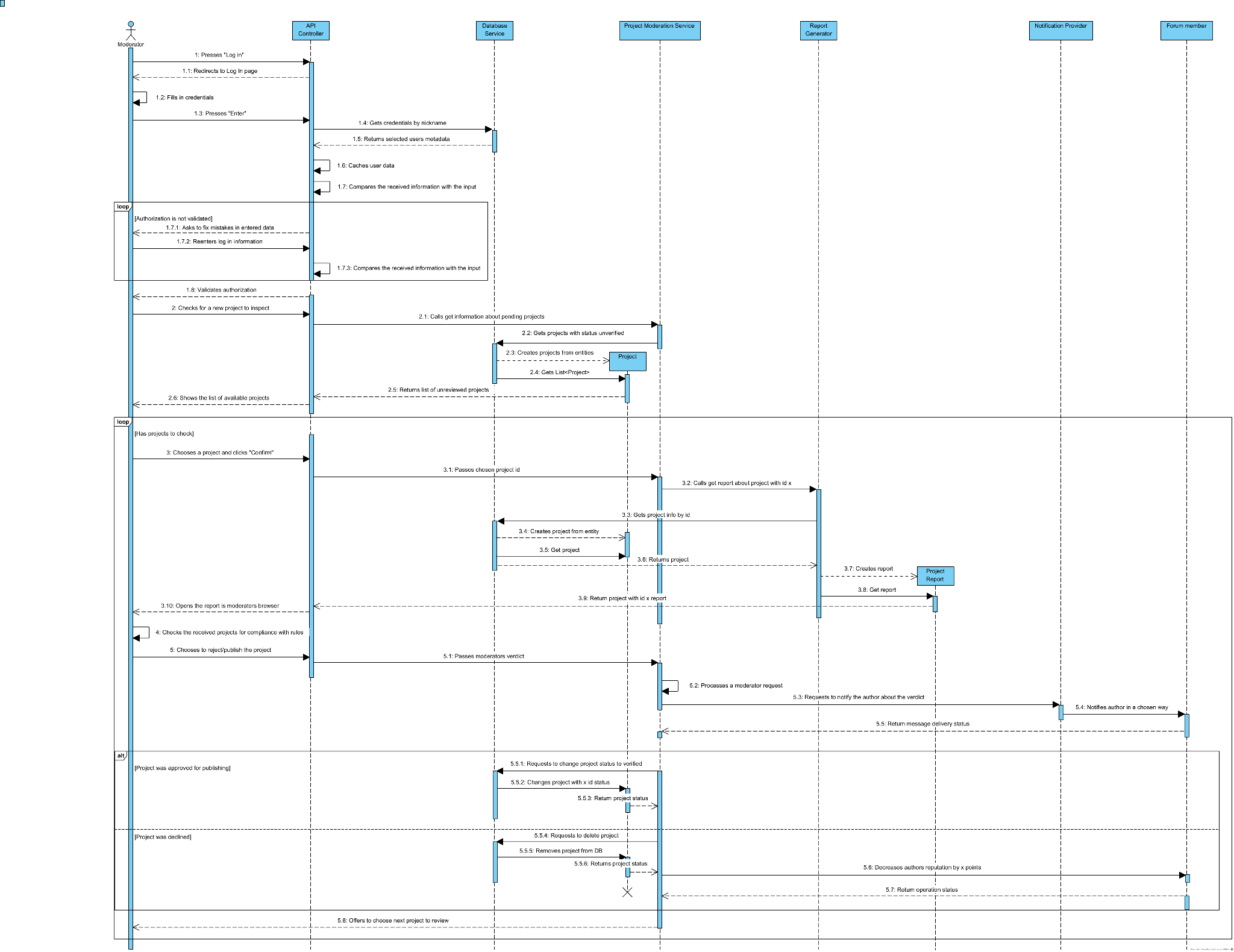


Процесс “создание проекта”

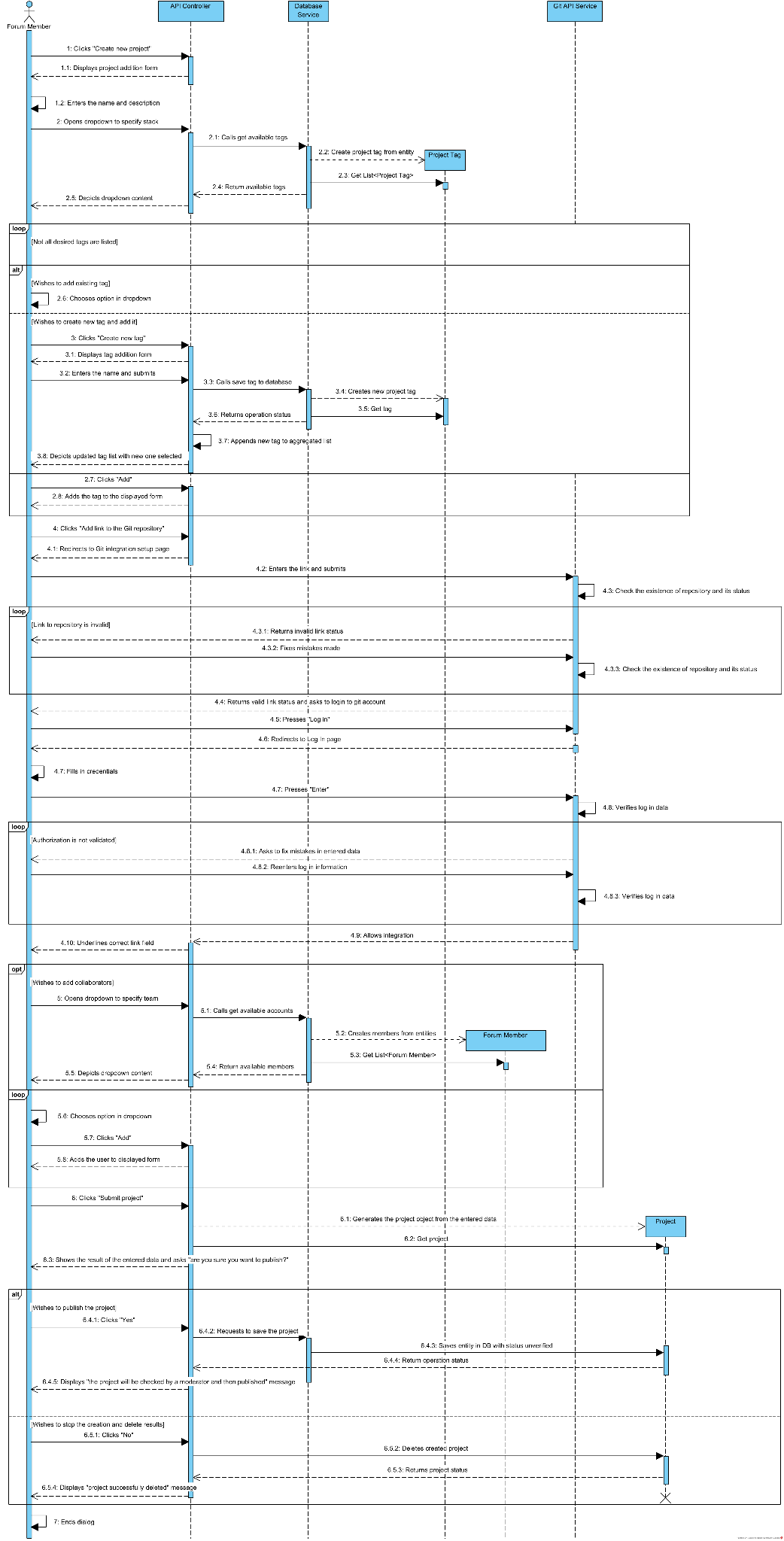


1. Sequence Diagram

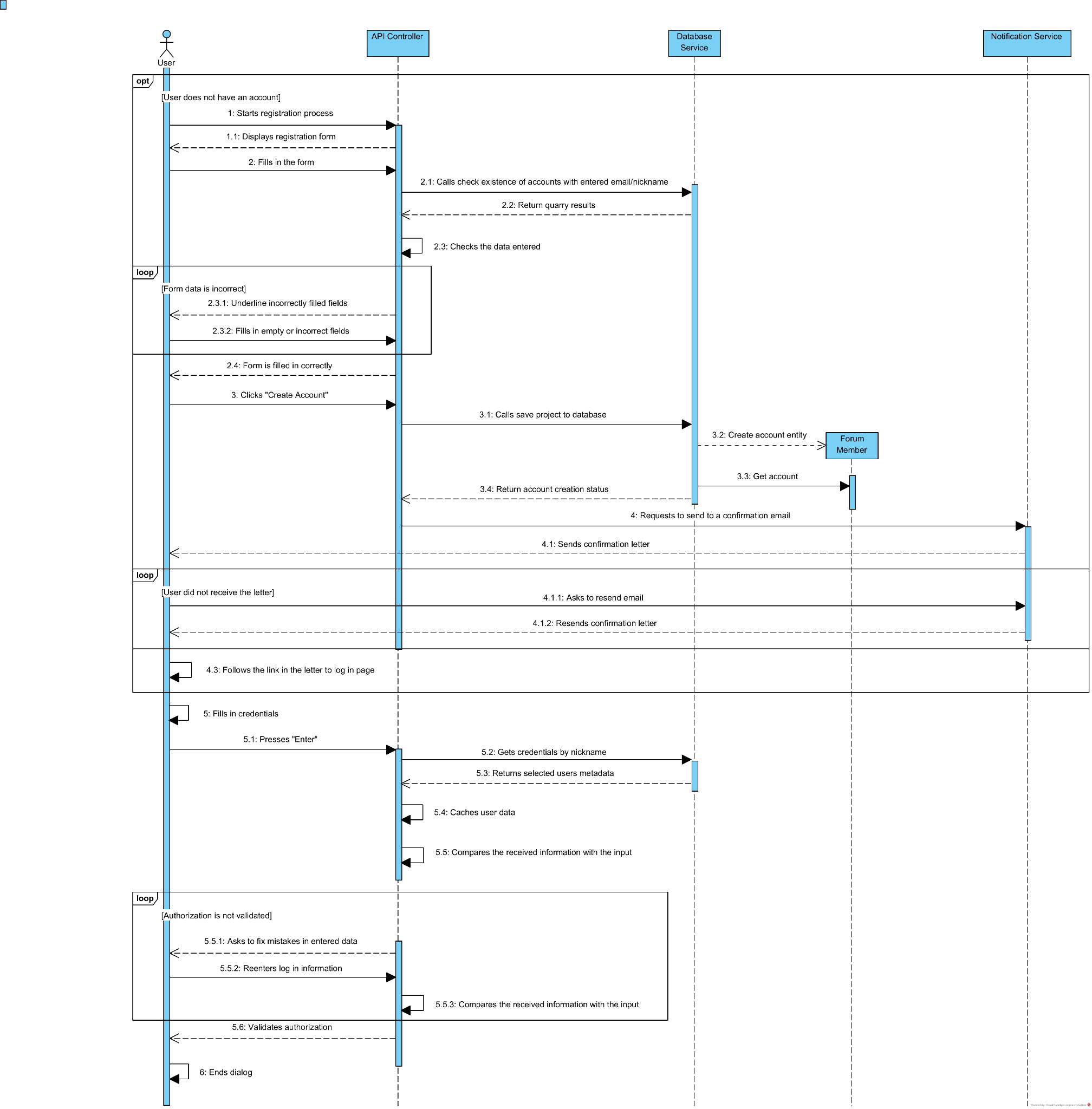
Процесс “модерация проекта”

****

Процесс “создание проекта”

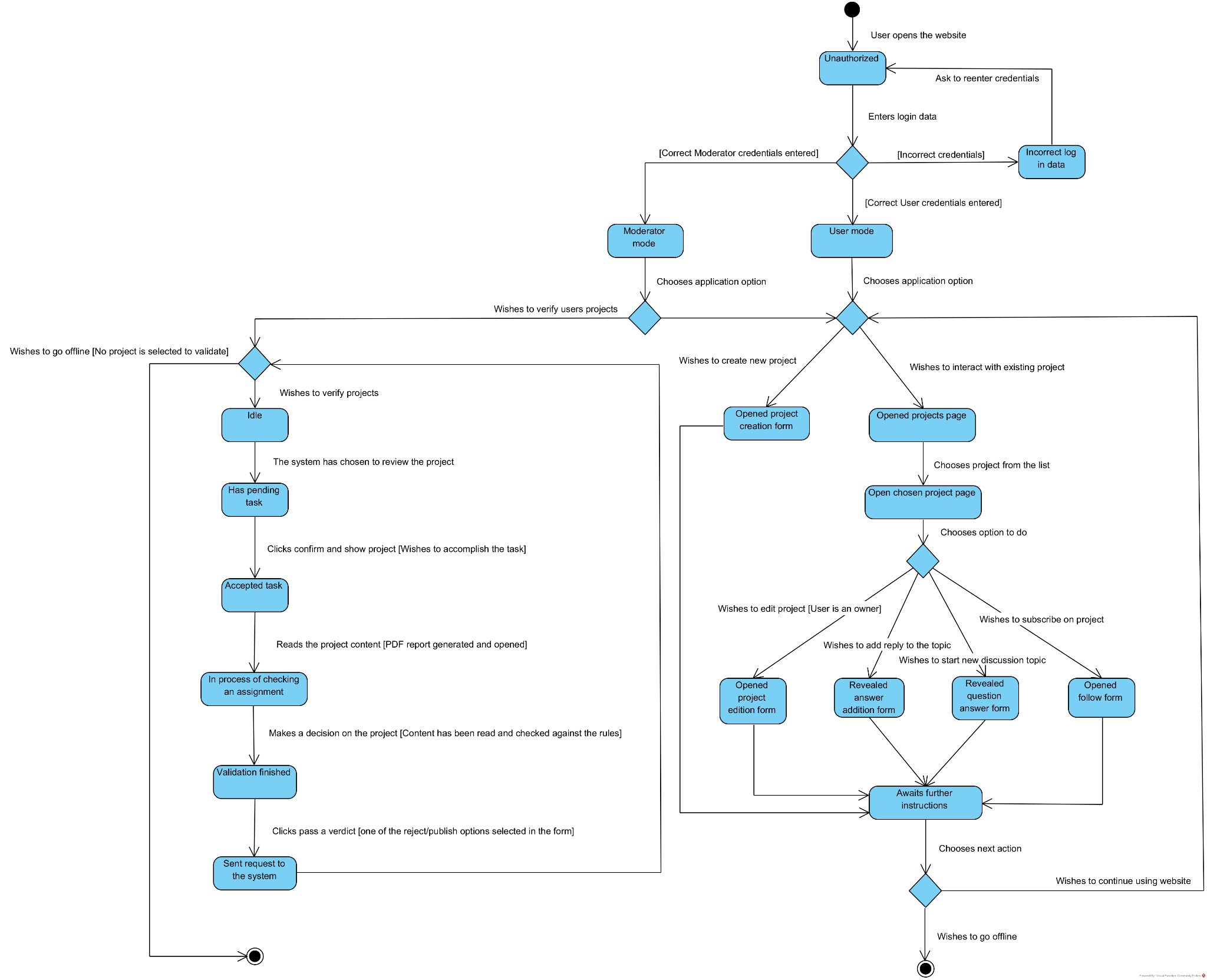
****

Процесс “авторизация пользователя”

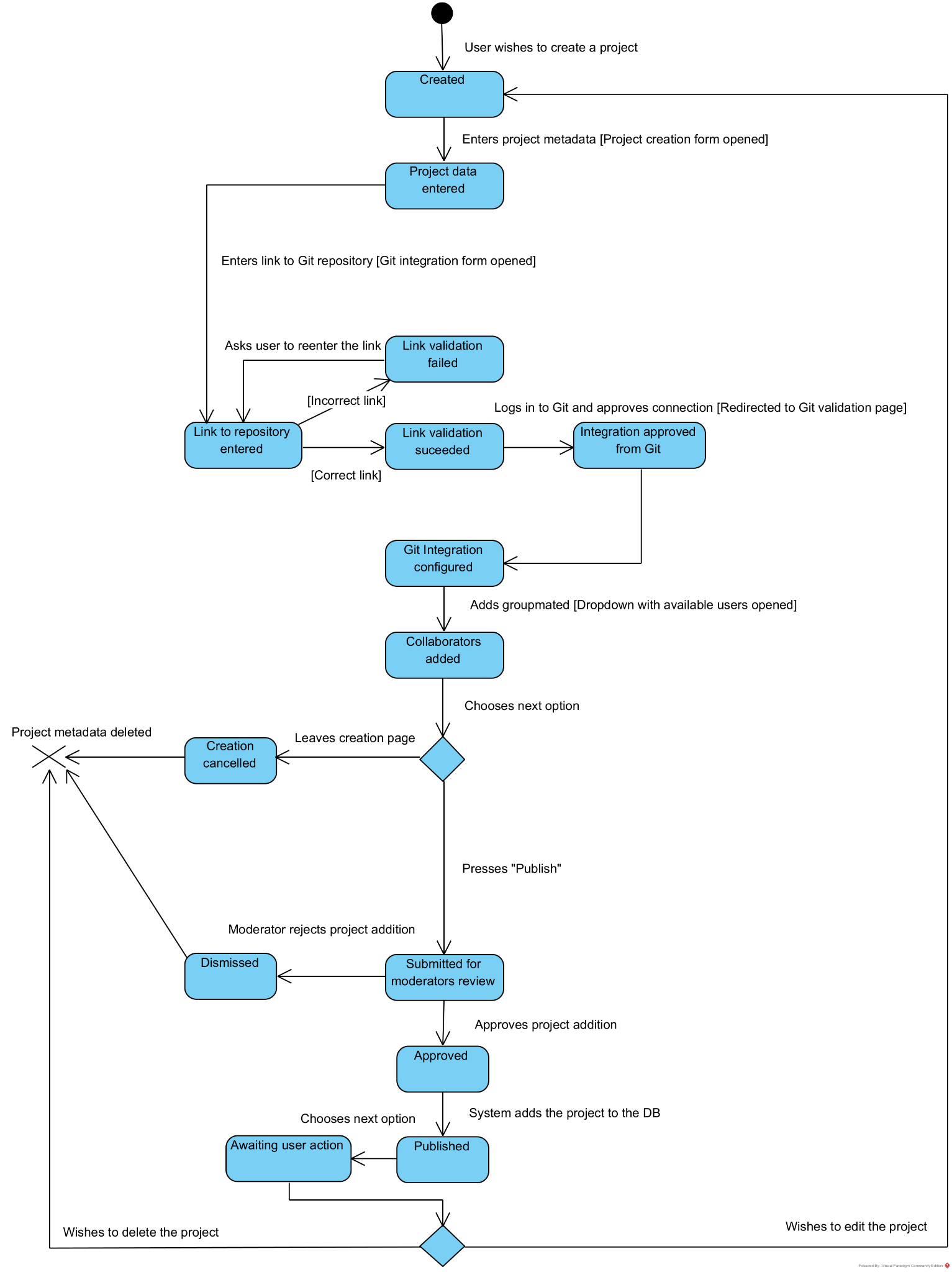


1. Диаграммы состояний

Класс Forum Member

****

Класс Project

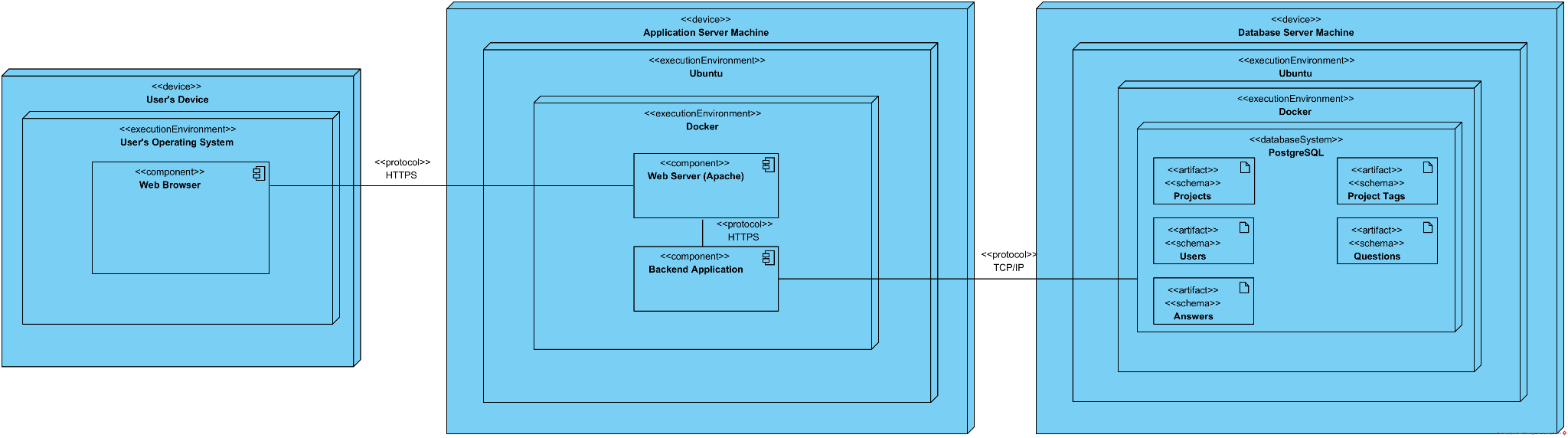
****

**Часть 2:**

1. Компоненты архитектуры:

* Web приложение

Разработано с использованием NestJS. Совместимо со всеми современными браузерами.

* Облачный сервер
* render.com
* Диаграмма размещения  
  
* Библиотеки и технологии
* NestJs, TypeScript
* HandleBars
* PostgreSQL
* TypeORM
* HTML, CSS, JS
* Sweetalert2
* Axios
* Socket.io
* Auth0
* Модули
* Auth

Реализует получение данных пользователя из cookies входящего запроса и создание сущности пользователя в базе данных. Также описывает методы для получения текущего авторизованного пользователя.

* Discussion

Описывает методы для создания обсуждений, комментариев и оценок, их редактирования и удаления, а также пагинации по страницам в дискуссии.

* Project

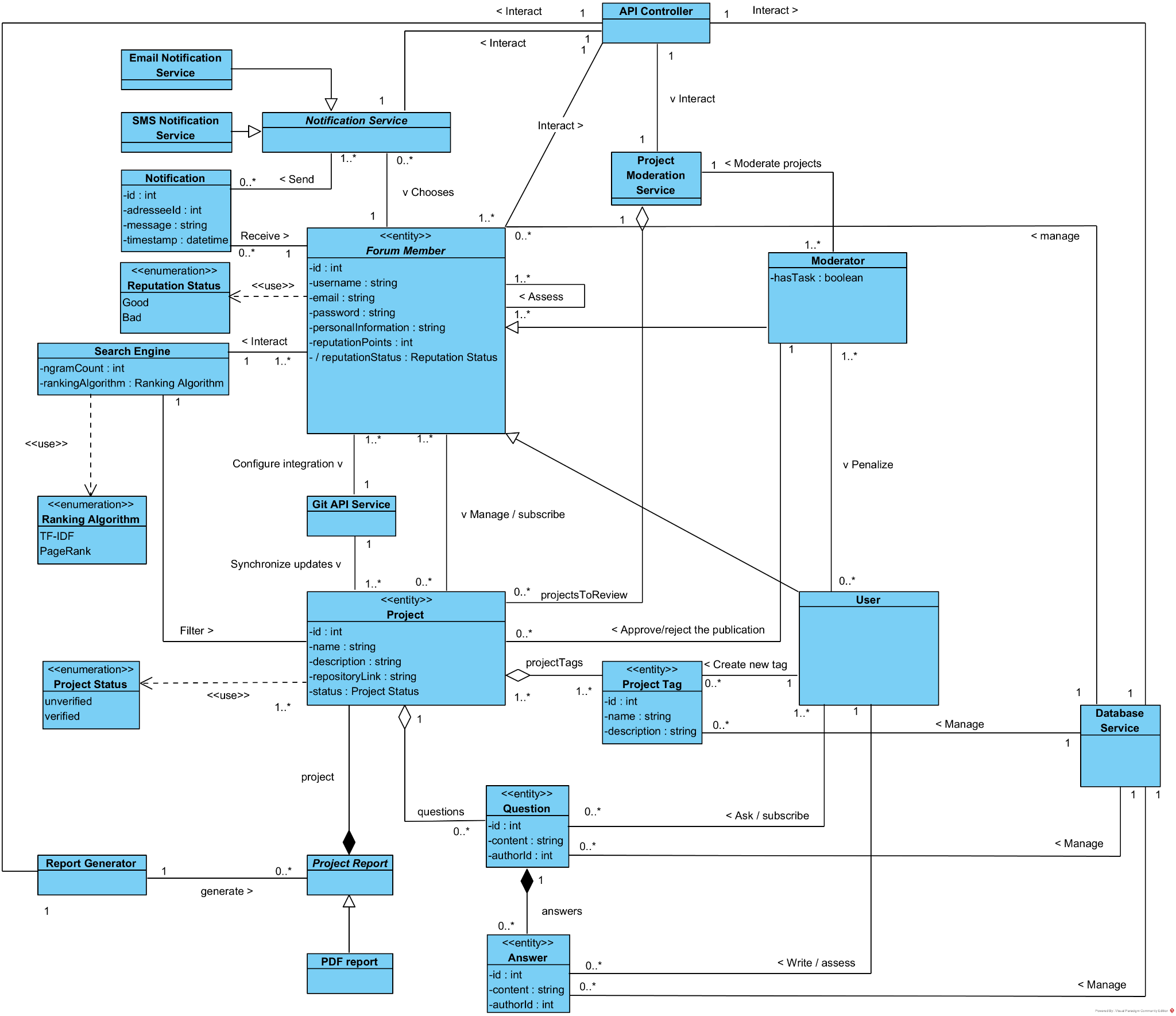
Описывает методы для создания проектов, тегов, их редактирования и удаления, а также пагинации по страницам в дискуссии.

* User

Описывает методы для создания пользователей, их редактирования и удаления, а также пагинации по страницам в дискуссии.

* Архитектура данных

Диаграмма классов

****

* Организация хранения данных
* Участник форума: ID, имя пользователя, пароль, электронная почта, личная информация, репутация.
* Пользователь и модератор: наследуются от участника форума.
* Проект: ID, название, описание, ссылка на гит-репозиторий.
* Тег проекта: ID, имя, описание.
* Вопрос: ID, содержание, ID автора.
* Ответ: ID, содержание, ID автора.
* Уведомление: ID, ID адресата, сообщение, метка времени.
* Статус репутации (enum): хороший, плохой.

**Часть 3:**

1. Сопоставление функциональной и программной архитектуры

* Обработка данных и передача их на сервер  
  Функциональный компонент - render.com  
  Программные компоненты - JavaScript методы на frontend части проекта + библиотека axios js, API Controller (UserController, ProjectController, ProjectTagController, TopicController, CommentController, AssessmentController)
* Получение данных с сервера и отображение их на сайте  
  Функциональный компонент - Programming Forum  
  Программные компоненты - Шаблонизатор HandleBars, AppController
* Создание собственного проекта и тегов к нему  
  Функциональный компонент - Project Entity, ProjectTag Entity  
  Программные компоненты - ProjectService, ProjectTagService
* Создание дискуссии и комментариев  
  Функциональный компонент - Topic Entity, Comment Entity  
  Программные компоненты - TopicService, CommentService
* Оценка комментариев  
  Функциональный компонент - Assessment Entity, Comment Entity  
  Программные компоненты - AssessmentService, CommentService

1. Сопоставление информационной архитектуры и архитектуры данных

* Информационный объект - “Пользователь”  
  Структура данных: User  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами проектов, обсуждений, комментариев и оценок.
* Информационный объект - “Проект”  
  Структура данных: Project  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами пользователей, обсуждений, и тегов.
* Информационный объект - “Тег проекта”  
  Структура данных: ProjectTag  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицей проектов.
* Информационный объект - “Обсуждение”  
  Структура данных: Topic  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами проектов и оценок.
* Информационный объект - “Комментарий”  
  Структура данных: Comment  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами оценок, пользователей и обсуждений.
* Информационный объект - “Оценка”  
  Структура данных: Assessment  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами пользователей и комментариев.

1. Обеспечение целостности и непротиворечивости данных

Мы обеспечиваем целостность и непротиворечивость данных при помощи встроенных средств в СУБД PostgreSQL. Отношения между сущностями в программе и базе данных организованы с помощью TypeORM. Кроме этого, в структурах данных указаны уникальные первичные ключи - id для всех сущностей, и уникальные поля для сущностей:

* User: nickname, email
* ProjectTag: name
* Project: name
* Topic: name

Текст выступления:

1. Добрый день, уважаемые коллеги. Наше задание на данную работу заключалось в анализе и моделировании средства автоматизации ИТ-процессов.

Работу выполнили: …

1. Начнём с описания проекта:  
   Он представляет из себя форум по программированию для обмена знаниями, совместной работы над проектами и ответов на вопросы. Регистрация и аутентификация обязательны. Также есть система репутации, роли модераторов и администраторов, теги для классификации контента и система поиска и фильтрации. Пользователи могут создавать проекты, задавать и отвечать на вопросы, которые связаны с проектом. Есть возможность уведомлений и рейтингов ответов на вопросы.
2. Перейдём к рассмотрению типовых ИТ процессов:

* **Сбор**

производится с помощью форм библиотеки “sweetalert2”, а также вручную написанных HTML форм. Например, создание обсуждения, проекта, комментария

* **Передача**

Frontend взаимодействует с Backend API посредством HTTP запросов

* **Обработка**

Производится с помощью сервисов NestJs, а также библиотеки “class-validator”

* **Хранение**

Данные сохраняются в сущности PostgreSQL, через обращение к репозиториям внутри сервисов, посредством TypeORM маппинга

* **Предоставление**

Все данные отрисовываются в браузере с использованием HTML, CSS и JS, страницы рендерятся на сервере с помощью HandleBars шаблонизации

1. Перейдём к рассмотрению ключевых информационных объектов:

* **Пользователь**

Лицо, зарегистрировавшее учетную запись на платформе и имеющее доступ к полному спектру функциональных возможностей, таких как создание и управление проектами, участие в обсуждениях, создание и ответы на вопросы, оценка комментариев и пользователей.

* **Модератор**

Пользователь, которому предоставлены специальные привилегии для модерирования платформы и обеспечения качества контента. Они могут иметь дополнительные функции, такие как возможность редактировать или удалять контент, предупреждать или блокировать пользователей, нарушающих правила сообщества.

* **Незарегистрированный пользователь**

Человек, который не вошел в систему и имеет ограниченный доступ к платформе, например, может просматривать контент, но не может создавать и управлять проектами или участвовать в обсуждениях.

* **Проект**

Набор информации, описывающий проект по программированию, включая его название, описание, участников, используемые инструменты и другие детали, которые автор проекта считает важными.

* **Тэг проекта**

Метка или ключевое слово, которое присваивается проекту для его классификации на основе предметной области, технологического стека или других соответствующих атрибутов.

1. Рассмотрим ещё несколько ключевых информационных объектов:

* **Тема обсуждения**

Набор сообщений, относящихся к определенному проекту или теме, где пользователи могут получать ответы на вопросы или обсуждать какие-либо темы.

* **Комментарий**

Сообщение, сделанное пользователем в ответ на вопрос, содержащее решение, объяснение или дополнительную информацию.

* **Репутации**

Оценка, которая вознаграждает пользователей за положительный вклад, например, за предоставление полезных ответов или отзывов.

* **Роль пользователя**

Уровень полномочий, присвоенный пользователю, который определяет его разрешения и обязанности на платформе. На данный момент включает в себя 2 уровня доступа – обычный пользователь и модератор.

* **Уведомления**

Отправляются пользователю, когда кто-то отвечает на один из его вопросов или комментариев, или когда проект, за которым он следит, обновляется.

1. Рассмотрим теперь функциональные требования:

* **Аутентификация пользователей**  
  Пользователи должны иметь возможность зарегистрировать учетную запись на платформе и пройти аутентификацию для доступа к полному спектру функциональных возможностей.
* **Создание и управление проектами**   
  Пользователи должны иметь возможность создавать новые проекты и управлять содержанием и разрешениями для своих собственных проектов.
* **Дискуссионные темы**   
  Пользователи должны иметь возможность создавать и участвовать в дискуссионных темах, связанных с конкретными проектами.
* **Система вопросов и ответов**  
  Пользователи должны иметь возможность задавать вопросы, связанные с программированием, и отвечать на них, привязываясь к соответствующему проекту или теме.
* **Система репутации**  
  Платформа должна иметь систему репутации, которая вознаграждает пользователей за положительный вклад, например, за предоставление полезных ответов или отзывов. Эта система может включать такие функции, как повышение или понижение рейтинга ответов, присвоение значков или баллов за определенные достижения или выделение пользователей с высокой репутацией.
* **Роли пользователей**   
  Платформа должна иметь различные роли пользователей с разными уровнями полномочий, например, модераторы, чтобы обеспечить хорошее содержание контента и сообщества.
* **Теги и фильтрация**  
  Проекты и обсуждения должны быть организованы и доступны для поиска на основе тегов, которые могут характеризовать стек, тематику или другие аспекты контента.
* **Уведомления**  
  Пользователи должны иметь возможность получать уведомления о новой активности, связанной с их проектами или вопросами, которые они задали или на которые ответили.
* **Интеграция с Git**  
  Платформа должна иметь возможность отслеживать состояние репозитория и своевременно обновлять информацию о проекте.
* **Функциональность поиска**   
  Платформа должна иметь систему поиска и фильтрации, которая позволяет пользователям легко находить релевантный контент на основе названия проекта, тега или ключевого слова.
* **Профиль пользователя и настройки**  
  Пользователи должны иметь возможность просматривать и управлять информацией своего профиля.

1. Перейдём к нефункциональным требованиям:

* **Доступность**  
  Платформа должна быть отзывчивой и хорошо работать на различных устройствах и размерах экрана, то есть на основных платформах (web, mobile).
* **Язык**   
  Платформа должна поддерживать переключение между двумя языками (русский, english)
* **Безопасность**   
  Платформа должна иметь соответствующие меры безопасности для защиты пользовательских данных и предотвращения несанкционированного доступа и атак.

1. Теперь рассмотрим Use Case диаграмму
2. Перейдём к рассмотрению Activity диаграмм. На этой диаграмме рассмотрен процесс авторизации пользователя.
3. На этой диаграмме рассмотрен процесс создания проекта
4. На этой диаграмме рассмотрен процесс модерации проекта
5. Перейдём к Sequence диаграммам. На этой диаграмме рассмотрен процесс авторизации пользователя.
6. На этой диаграмме рассмотрен процесс создания проекта
7. На этой диаграмме рассмотрен процесс модерации проекта
8. Перейдём к State Machine Diagram. На этой диаграмме рассмотрен класс ForumMember
9. На этой диаграмме рассмотрен класс Project
10. Теперь перейдём к рассмотрению Компонентов архитектуры. В качестве компонентов у нас непосредственно Web приложение, которое разработано с использованием NestJS и совместимо со всеми современными браузерами. А также облачный сервер в лице render.com.

Также на этом слайде представлена диаграмма размещения.

1. Теперь рассмотрим используемые библиотеки и технологии:

* **NestJs, TypeScript**
* **HandleBars**
* **PostgreSQL**
* **TypeORM**
* **HTML, CSS, JS**
* **Sweetalert2**
* **Axios**
* **Socket.io**
* **Auth0**

1. Далее рассмотрим модули:

* **Auth**

Реализует получение данных пользователя из cookies входящего запроса и создание сущности пользователя в базе данных. Также описывает методы для получения текущего авторизованного пользователя.

* **Discussion**

Описывает методы для создания обсуждений, комментариев и оценок, их редактирования и удаления, а также пагинации по страницам в дискуссии.

* **Project**

Описывает методы для создания проектов, тегов, их редактирования и удаления, а также пагинации по страницам в дискуссии.

* **User**

Описывает методы для создания пользователей, их редактирования и удаления, а также пагинации по страницам в дискуссии.

1. На этом слайде рассмотрим диаграмму классов. Классы, которые хранятся в базе данных помечены Entity.
2. Теперь рассмотрим как организовано хранение данных.

* **Участник форума:**

ID, имя пользователя, пароль, электронная почта, личная информация, репутация.

* **Пользователь и модератор:**

наследуются от участника форума.

* **Проект:**

ID, название, описание, ссылка на гит-репозиторий.

* **Тег проекта**

ID, имя, описание.

* **Вопрос**

ID, содержание, ID автора.

* **Ответ**

ID, содержание, ID автора.

* **Уведомление**

ID, ID адресата, сообщение, метка времени.

* **Статус репутации (enum)**

хороший, плохой.

1. Теперь сопоставим функциональную и программную архитектуру:

* **Обработка данных и передача их на сервер**  
  Функциональный компонент - render.com  
  Программные компоненты - JavaScript методы на frontend части проекта + библиотека axios js, API Controller (UserController, ProjectController, ProjectTagController, TopicController, CommentController, AssessmentController)
* **Получение данных с сервера и отображение их на сайте**  
  Функциональный компонент - Programming Forum  
  Программные компоненты - Шаблонизатор HandleBars, AppController
* **Создание собственного проекта и тегов к нему**  
  Функциональный компонент - Project Entity, ProjectTag Entity  
  Программные компоненты - ProjectService, ProjectTagService
* **Создание дискуссии и комментариев**  
  Функциональный компонент - Topic Entity, Comment Entity  
  Программные компоненты - TopicService, CommentService
* **Оценка комментариев**  
  Функциональный компонент - Assessment Entity, Comment Entity  
  Программные компоненты - AssessmentService, CommentService

1. Теперь сопоставим информационную архитектуру и архитектуру данных:

* **Информационный объект - “Пользователь”**  
  Структура данных: User  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами проектов, обсуждений, комментариев и оценок.
* **Информационный объект - “Проект”**  
  Структура данных: Project  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами пользователей, обсуждений, и тегов.
* **Информационный объект - “Тег проекта”**  
  Структура данных: ProjectTag  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицей проектов.
* **Информационный объект - “Обсуждение”**  
  Структура данных: Topic  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами проектов и оценок.
* **Информационный объект - “Комментарий”**  
  Структура данных: Comment  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами оценок, пользователей и обсуждений.
* **Информационный объект - “Оценка”**  
  Структура данных: Assessment  
  Хранение: таблица в базе данных PostgreSQL, связанная с таблицами пользователей и комментариев.

1. Наконец разберём как обеспечивается целостность и непротиворечивость данных.

Мы обеспечиваем целостность и непротиворечивость данных при помощи встроенных средств в СУБД PostgreSQL. Отношения между сущностями в программе и базе данных организованы с помощью TypeORM. Кроме этого, в структурах данных указаны уникальные первичные ключи - id для всех сущностей, и уникальные поля для сущностей:

* + User: nickname, email
  + ProjectTag: name
  + Project: name
  + Topic: name

Вопросы:

1. Зачем комментарию нужна репутация?  
   В обсуждении комментарии отфильтрованы по убываю репутации для получения наиболее релевантных ответов. Также при достижении очень негативной репутации, комментарий автоматически будет удалён системой.
2. Почему у вас таблица topic не связана с user?  
   В данном проекте не планировалось ассоциировать пользователя и обсуждение. Топик привязан к проекту и уничтожается при его удалении. У пользователя же есть комментарии, которые он может создать.
3. Почему используется СУБД PostgreSQL?  
   У неё есть ряд преимуществ перед другими СУБД: свободный доступ, отсутствие необходимости дополнительных инструментов для установки, поддержка разных форматов данных, позволяет работать с большими объёмами данных.
4. Почему в БД картинки хранятся в формате string?  
   На фронте, при приёме картинки из формы, она преобразовывается в base64 и сохраняется в БД. При необходимости её отрисовки, происходит аналогичный процесс. Наверное, можно было сделать лучше, сохраняя её в двоичном виде с помощью Buffer, это помогло бы оптимизации.
5. Вы упоминали библиотеку sweetalert2, расскажите, что она из себя представляет?  
   Данная библиотека представляет из себя замену стандартных popup js форм. Она предоставляет приятный дизайн, и, что немаловажно, работает без сторонних зависимостей, что удобно в данном случае.
6. На 19 слайде у вас указано, все модули включены в AppModule, происходит ли у вас общение между самими модулями?  
   Нет, модули не связаны, потому что мы в них напрямую обращаемся к репозиториям, так что в их связи нет необходимости.
7. Почему вы выбрали библиотеку Auth0?  
   Auth0 - это популярная платформа идентификации, которая позволяет разработчикам интегрировать различные методы аутентификации.
8. Среди библиотек была указана “Socket.io”, но не рассказано для чего она используется, можете уточнить?  
   Мы интегрировали эту библиотеку для создания уведомлений для пользователей. Когда создается
9. В какой последовательности вы делали uml диаграммы? До проекта или после, и почему?  
   В целом, мы создали все диаграммы до начала разработки проекта. Но, при написании кода, поняли, что некоторая логика и сущности нуждаются в доработке, так что в реализации уже присутствуют вещи, соответствующие новым требованиям. В целом, мы считаем, что стоит рисовать uml диаграммы до старта проекта для обеспечения понимания между всеми членами команды и удобной разработки, но, возможно, есть смысл и после, чтобы избежать подобных оказий и написать хорошую документацию.
10. Как вы считаете, являются ли uml диаграммы необходимыми для проекта?  
    В целом, при создании небольших Pet проектов в одиночку или с небольшой командой, можно обойтись и без диаграмм, чтобы обеспечить небольшой time to market. Но при решении более сложных задач заблаговременное планирование и моделирование значительно упрощают программирование. Кроме того, вносить изменения в диаграммы классов легче, чем в исходный код.
11. У вас есть сущности проекта и его тегов, которые связаны как ManyToMany, как это реализовано в приложении?  
    TypeORM реализует данный вид связи между таблицами Project и ProjectTag отношения ManyToMany с помощью дополнительной таблицы Project\_ProjectTag, в которой содержатся id этих сущностей.