Отчет по проекту “Promeets”

Сервис для организации и проведения удаленных аудио и видео конференций

Зацепин В.В.

ФКН ВГУ

Воронеж 2016

Содержание

[Введение 1](#_Toc466911526)

[1. Постановка задачи 2](#_Toc466911527)

[2. анализ задачи 3](#_Toc466911528)

[2.1. Основные объекты системы 3](#_Toc466911529)

[2.2. Существующие аналоги 5](#_Toc466911530)

[2.3. Средства реализации 6](#_Toc466911531)

[2.4. Технические требования к клиентской среде исполнения 6](#_Toc466911532)

[2.5. Система оповещений 7](#_Toc466911533)

[3. Реализация 9](#_Toc466911534)

[3.1. Структура базы данных 9](#_Toc466911535)

[3.2. Реализация архитектуры сервера 9](#_Toc466911536)

[3.3. Реализация архитектуры клиента. 11](#_Toc466911537)

[3.4. Система оповещения 12](#_Toc466911538)

[3.5. Размещение 14](#_Toc466911539)

[3.6. Тестирование и использование 14](#_Toc466911540)

[Заключение 18](#_Toc466911541)

Введение

В современном мире вопрос удаленной коммуникации между людьми играет большую роль. Возможность связаться с другими людьми, находящимся в разных точках мира, позволяет быстро решать актуальные вопросы и принимать своевременные решения во всех сферах жизни. Кроме того, в большинстве случаев, наибольшую роль играет предмет разговора и его визуализация. Поэтому наличие удобного сервиса, позволяющего показывать другим участникам различную текстовую и графическую информацию во время удаленной видео конференции, является современным средством решения многих задач. На сегодняшний день существует много сервисов, позволяющий проводить удаленные встречи, но все они имеют те или иные недостатки: необходимость устанавливать дополнительное ПО, отсутствие возможности передавать информацию отличную от аудио и видео в реальном времени и т.д. Поэтому вопрос создание подобного современного и удобного сервиса является и по сей день открытым.

1. Постановка задачи

Целью проекта является предоставление конечному пользователю удобного и современного сервиса для организации и проведения встреч через Интернет средствами аудио и видео связи. Сферы применения:

* Организация и проведение удаленных собраний внутри компании или между компаниями
* Организация и проведение различных образовательных курсов, лекций
* Организация и проведение удаленного обучения на всех уровнях образования (школа, университет)
* Организация различных мероприятий
* И т.д.

Необходимо реализовать следующие требования:

* Организация системы в виде социальной сети с максимально упрощенной схемой коммуникации между людьми.
* Визуализация предмета встречи, возможность наглядного изложения материала.
* Видео (аудио) конференция.
* Удобная организация встреч и наличие личного календаря для каждого пользователя.

1. анализ задачи
   1. Основные объекты системы

Для реализации системы на всех уровнях: от базы данных до клиентского приложения необходимо определить основные объекты системы, их свойства и требования к ним.

* Пользователи
  + Доступ к объектам и возможностям системы имеют только зарегистрированные пользователи
  + Наличие личного календаря
  + Наличие личного профиля с возможность редактирования. Профиль включает такие элементы, как:
    - Имя и фамилия
    - Адрес электронной почты
    - Контактный номер телефона
    - Компания
    - Должность в компании
    - Контактный адрес
* Группы
  + Создаются пользователями
  + Внутри группы для всех ее участников могут быть доступны основные элементы группы:
    - Файлы
    - Информация о проведенных встречах
    - Сообщения
    - Информация о предстоящих встречах (расписание)
  + Внутри группы пользователь может:
    - Добавить, просмотреть или скачать файлы
    - Создать новую встречу
    - Участвовать в удаленных аудио или видео встречах
    - Обмениваться сообщениями с другими участниками группы
    - Редактировать расписание встреч
* Встречи
  + Существуют только в рамках групп
  + Создаются пользователями
  + Проводятся в виде удаленной конференции средствами аудио и видео связи
  + Основные элементы:
    - Название
    - Дата и время
    - Цели
    - Участники
    - Заметки
    - Карточки. Основной элемент встречи посредством которого участники обмениваются различной информацией. Выражается в виде элемента интерфейса, схожим по внешнему виду с стикером или любым иным объектом, предоставляющий общий доступ к графической информации (объявление, рекламный плакат и т.д.). Служит для отображения любой графической информации: изображения, текст, ссылки на любые другие файлы.
  + Информация о встрече, ее основных элементах и итогах сохраняются в группе с возможностью просмотра и редактирования

Определив основные объекты системы возможно построение диаграммы вариантов использования приложения.

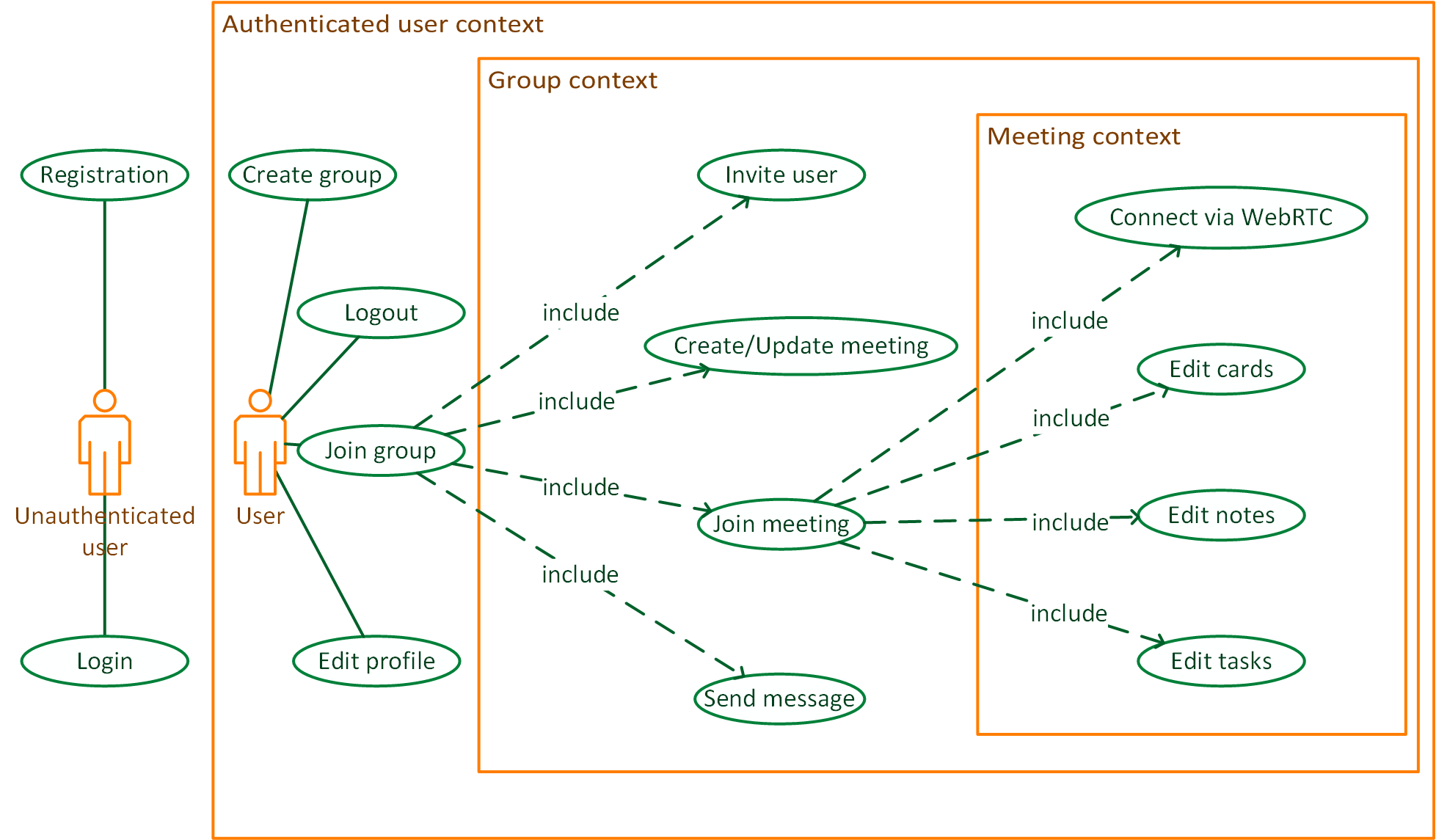


Рисунок 2.1.1 – Диаграмма вариантов использования

Существует две роли: не аутентифицированный пользователь и аутентифицированный пользователь. Регистрация и процесс аутентификации являются необходимыми для доступа к системе. После того, как пользователь войдет в систему возможно создание группы или присоединение к ней, в случае наличия приглашения от другого пользователя. В контексте группы возможно приглашение других пользователей, создание или редактирование встреч, отправка сообщений и присоединение ко встрече. Из контекста группы возможен переход на контекст встречи, в котором пользователь может принять участие в аудио-видео конференции и редактировать основные ее элементы: карточки, записки и заметки.

* 1. Существующие аналоги

Основными аналогами рассматриваемой системы являются такие сервисы как «WebEx», «Webinar.ru», «GoToMeeting». Они позволяют проводить заранее запланированные веб-конференции с участием географически удалённых сотрудников, обмениваться документами, презентациями, выполнять демонстрацию экрана. Но у них также есть свои недостатки. Основным из них является плата за использование сервиса. Также для работы некоторых из них (webEx) требуются установка плагинов или дополнительных приложений. А еще не все из перечисленных продуктов имеют удобные встроенные механизмы для непосредственной организации встреч и отправки приглашений всем участникам. Наш же сервис прост и интуитивно понятен, не обременен избыточным функционалом, усложняющим пользовательский интерфейс, и позволяет быстро и легко организовать и провести встречу для небольшой группы разработчиков.

* 1. Средства реализации
* Серверная часть
  + Java
  + REST
  + Spring Framework
  + PostgresSQL (СУБД)
  + WebRTC + WebSockets

Для реализации аудио и видео связи предполагается использования технологии WebRTC, позволяющей устанавливать P2P (Peer To Peer) соединение по полносвязной топологии между клиентами для обмена медиа потоками при помощи браузера и сигнального сервера. Последний является необходимым объектом для первоначального обмена данными между клиентами до установления прямого соединения между ними и реализуется в контексте основного сервера в видео контроллера, обрабатывающий запросы, отправленные клиентом по технологии WebSockets.

* Клиентская часть
  + HTML + CSS
  + AngularJS Framework

Обмен данными между клиентом и сервером осуществляется при помощи REST API.

* 1. Технические требования к клиентской среде исполнения
* Браузер с поддержкой технологии HTML5
* Для аудио и видео связи – поддержка браузером WebRTC (Chrome, Mozilla Firefox, Opera)
  1. Система оповещений

Система оповещений необходима для задания определенного уровня интерактивности и используется для оповещения пользователя о:

* Новом приглашении в группу
* Новом сообщении
* Новой встречи или обновлении встречи (изменены элементы встречи, дата, время, название)
* В случае нахождении пользователя на странице встречи, элементы, которые изменяются: при новой карточке, новой записке или новой задаче

В общем случае система оповещений позволяет синхронизировать данные без перезагрузки страницы.

Общий алгоритм работы системы оповещений представлен на рисунках 2.5.1. и 2.5.2. Один пользователь меняет данные, после чего клиентское приложение отправляет HTTP запрос. Сервер в свою очередь асинхронно отправляет всех подходящим пользователям сообщение об изменении сущности и возвращает HTTP ответ.

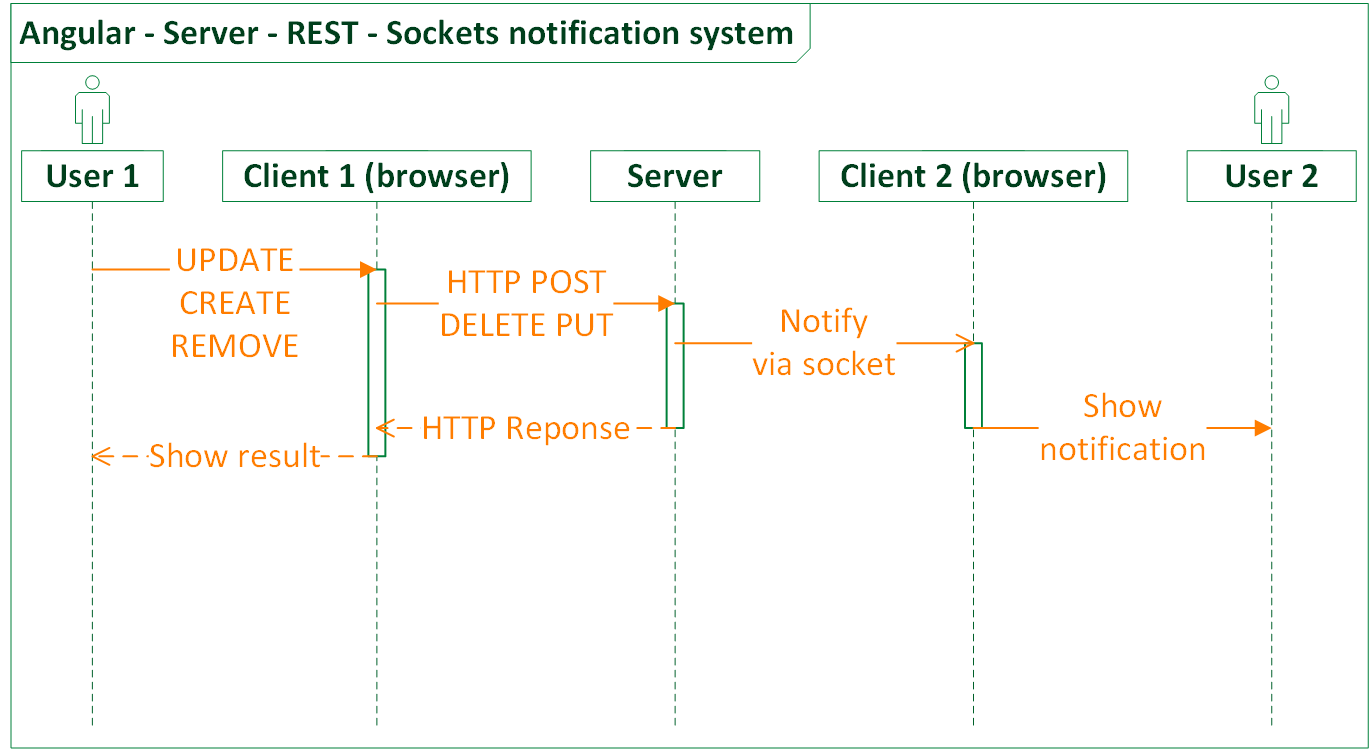


Рисунок 2.5.1. – Диаграмма последовательности для системы оповещений.

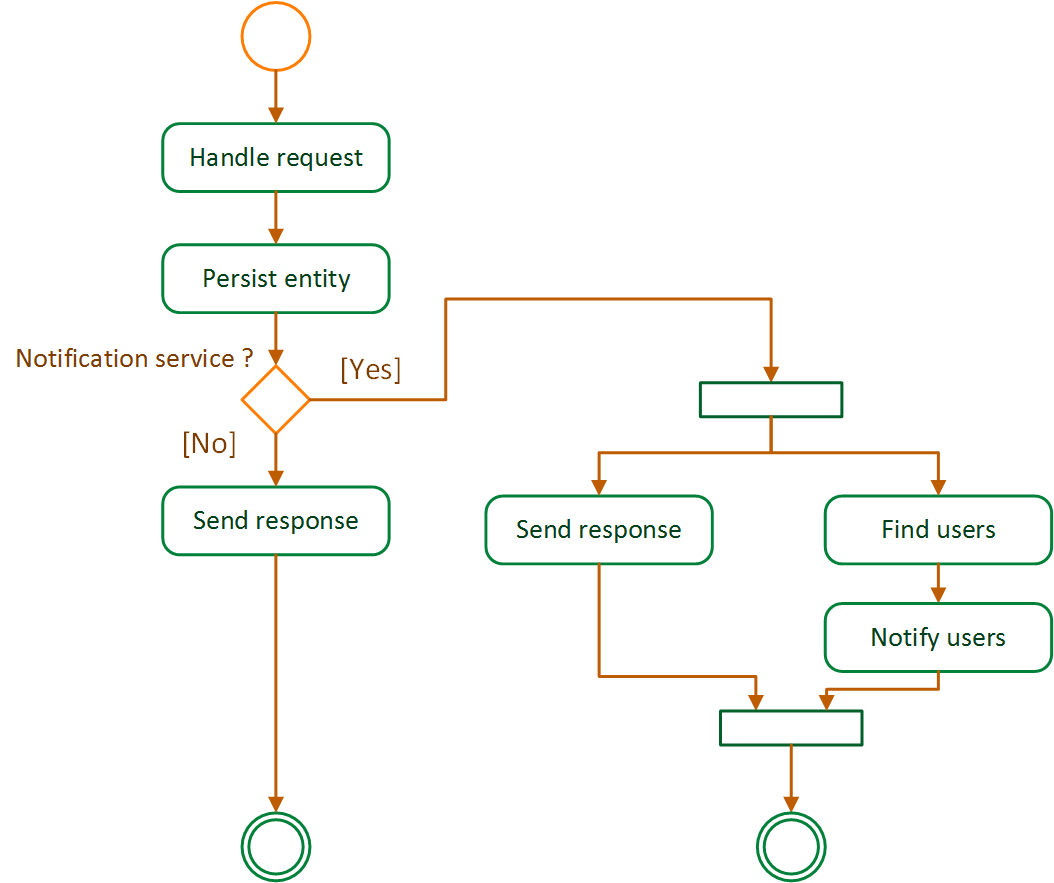


Рисунок 2.5.2. - Диаграмма деятельности алгоритма системы оповещений.

1. Реализация

В результате анализа было принято решение о разработке системы, описание которой приводится в данной главе.

* 1. Структура базы данных

Структура базы данных приведена на рисунке 3.1.1. Данная структура отображает лишь отношения между основными сущностями, что является необходимым для построения базы данных. Атрибуты сущностей не являются критическим аспектом.

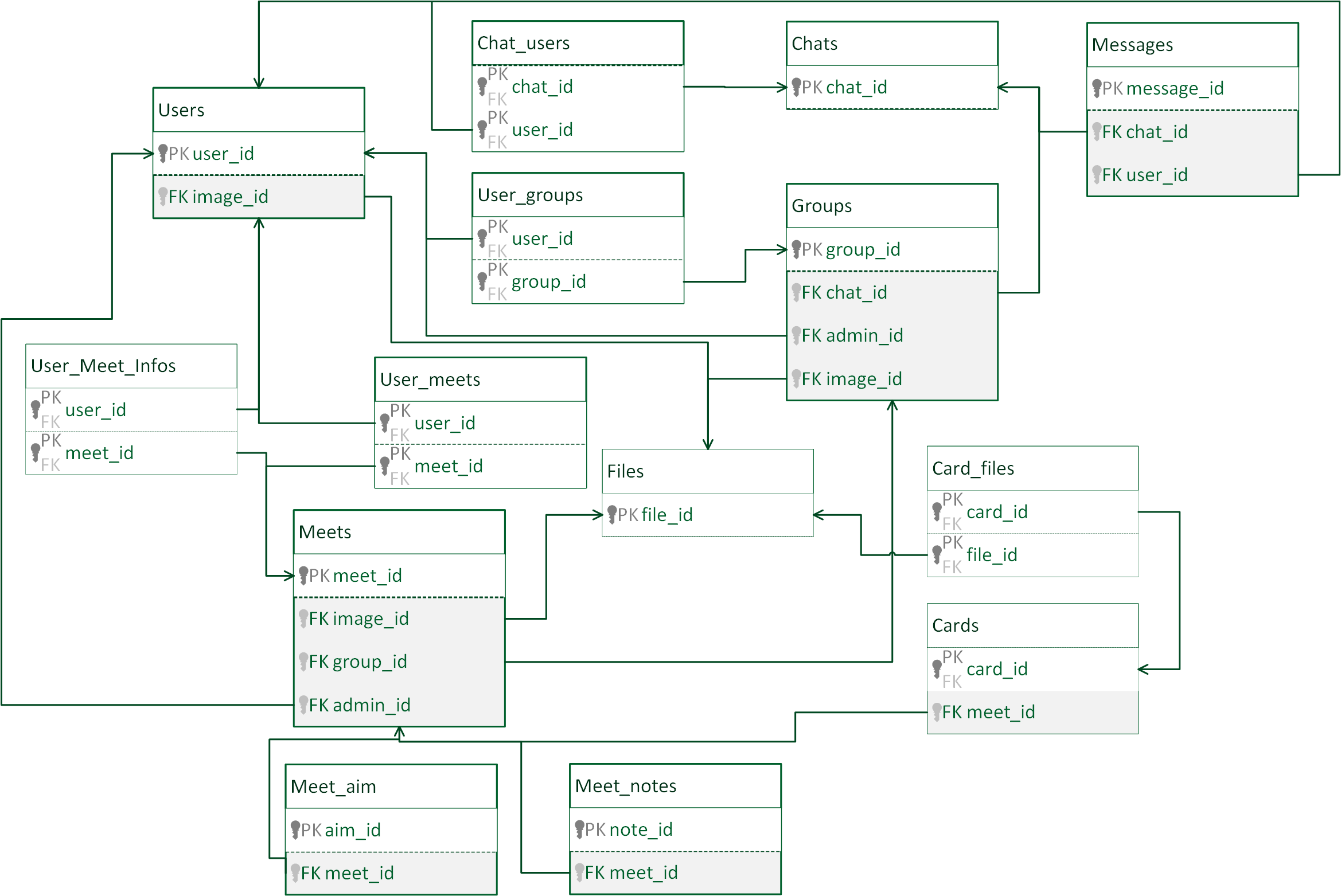


Рисунок 3.3.1. – Схема базы данных

* 1. Реализация архитектуры сервера

Все классы разрабатываемого приложения могут быть разделены на несколько функциональных слоев:

* Слой контроллера
* Слой сервиса
* Слой доступа к данным

Данная структура построена на основе взаимодействия классов не напрямую, а через интерфейсы, что необходимо для создания функциональной независимости взаимодействия между модулями от их реализации.

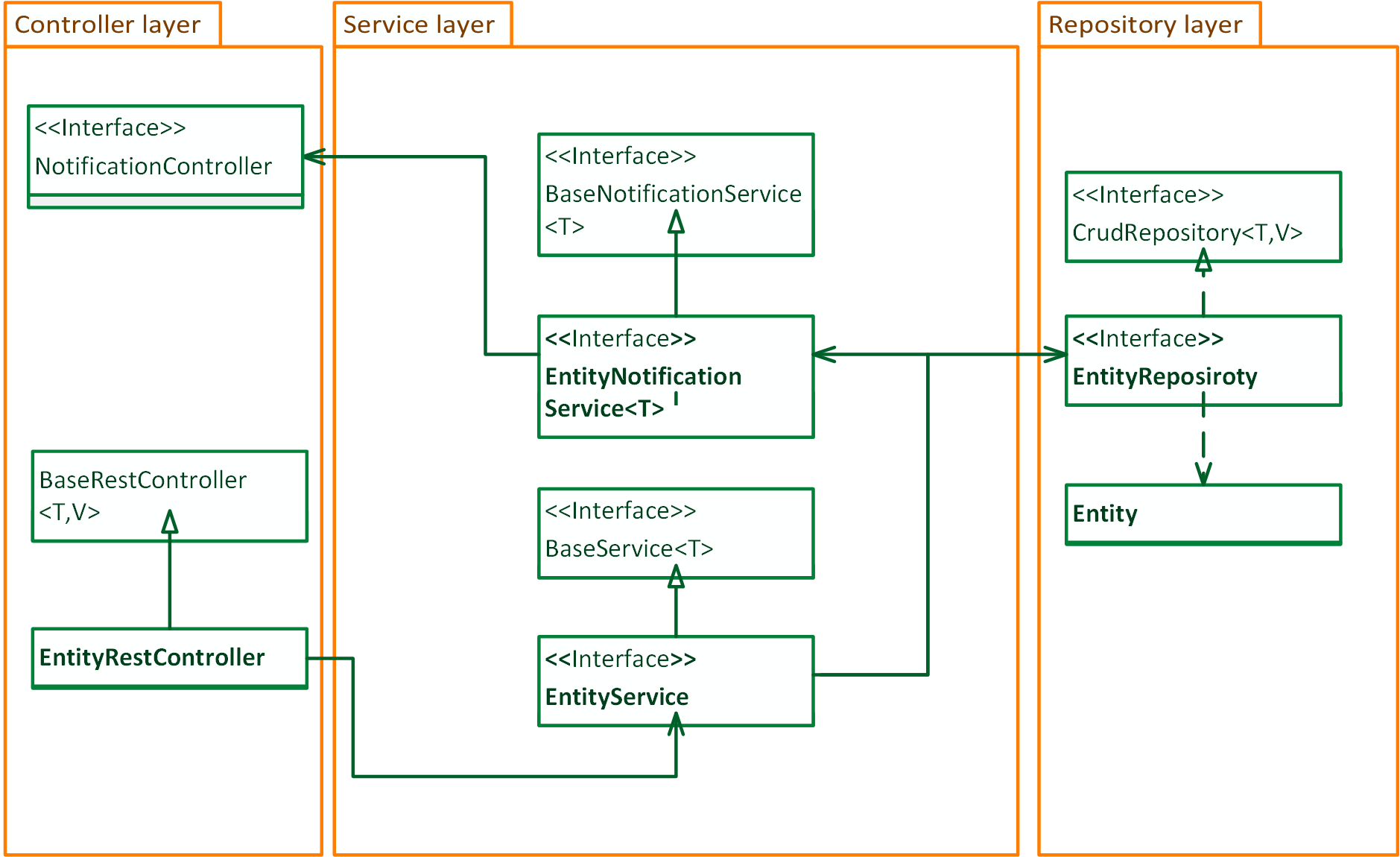


Рисунок 3.2.1. – Общая структура классов

Слой контроллера отвечает за обработку запросов пользователя и реализует REST API.

Слой сервиса необходим для реализации бизнес логики всего приложения.

Слой доступа к данным (repository) необходим для обеспечения работы с базой данных.

На рисунке 3.2.2. представлена часть диаграммы отражающую способ получения пользователя по его Id используя REST API путь «/api/users/{id}». Контроллер обрабатывает GET запрос, поступающий на сервер – метод getUserById. После чего сервис производит работу с слоем доступа к базе данных и получает пользователя используя параметр Id и возвращает значение контроллеру, который в свою очередь возвращает сущность клиенту в формате JSON.

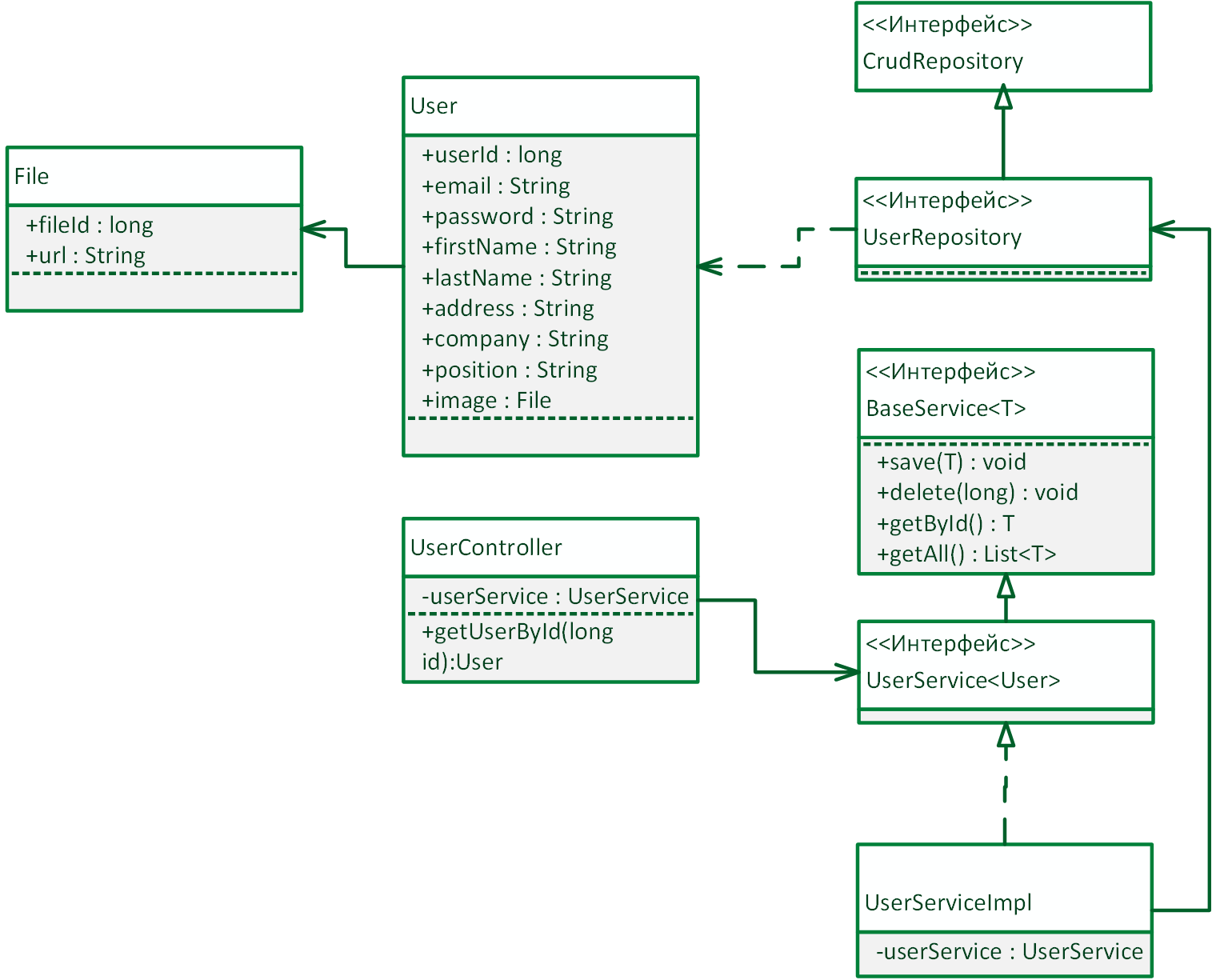


Рисунок 3.2.2. – Диаграмма классов, связанных с сущностью User

* 1. Реализация архитектуры клиента.

Клиентская часть приложения построена на основе HTML5 и Фреймворка AngularJS по архитектуре SPA (Single Page Application). Общий вид структуры объектов показан на рисунке 3.3.1:

* View - html страничка.
* UI-Router – объект для реализации механизма разрешения состояний и определения состояний по адресу странички.
* Directory – шаблон отдельного элемента View. Например, общий заголовок всех страничек сайта или список объектов.
* Service – подобно объектам на сервисном уровне архитектуры сервера, реализует бизнес логику на стороне клиента.
* Controller - осуществляет взаимодействие между View и Service.

Для примера разберем работу клиентского приложения в случае попадания на страничку профиля: «/profile/{id}», где {id} – это Id пользователя. После ввода данного адреса, при загрузке страницы, UI Router определяет кукую именно страничку запросил пользователь и сообщает UserService о том, что необходимо загрузить пользователя с указанным в адресе Id. После этого UserService отправляет запрос на сервер по пути: «/api/users/{id}» и сохраняет ответ у себя в области видимости. Как только страничка пользователя (View) была полностью загружена и обработана браузером начинает работать Controller, который получает уже загруженного пользователя из UserController и отображает его на View в специальной директиве UserViewProfile.

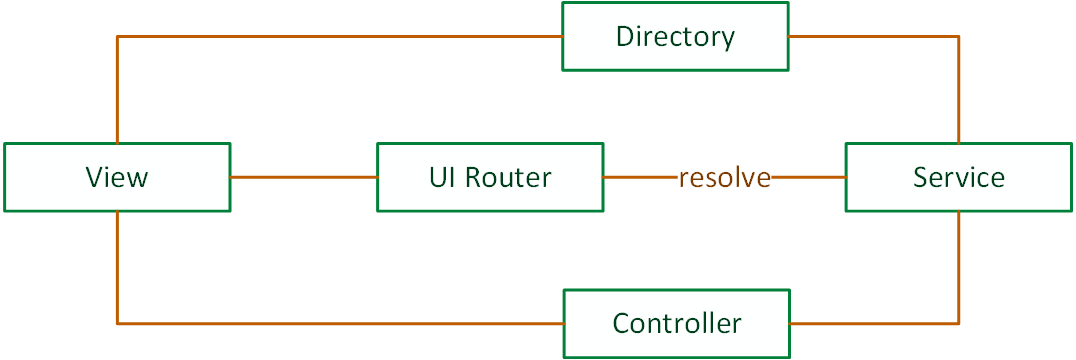


Рисунок 3.3.1. – схема взаимодействия основных элементов клиентской части приложения.

* 1. Система оповещения

Архитектурно на стороне сервера это выполнено в следующем виде (Рисунок 3.4.1.): сервис сущности имеет ассоциацию с сервисом оповещений, который определяет пользователей, которых нужно оповестить и совершает это через Контроллер Оповещений.

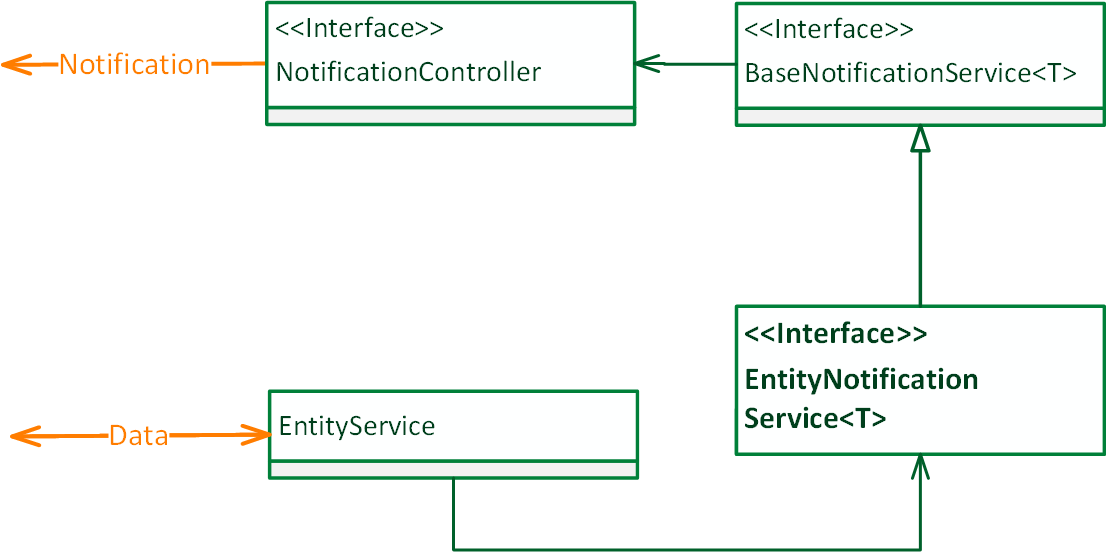


Рисунок 3.4.1. – Серверная часть архитектуры системы оповещений.

На клиентской части система реализована на основе возможностей и особенностей Фреймворка AngularJS: каждая страничка сайта является некоторым абстрактным состоянием для каждого из которого создается свой контроллер со своей областью видимости Scope (Рисунок 3.4.2.). Причем эти состояния образуют иерархию и соответственно области видимости так же находятся в иерархической структуре. UserScope является базовой для всех страничек пользователя, в нем происходит установка соединения с сервером. Далее при помощи механизма общения между контроллерами, предусмотренным AngularJS, при поступлении оповещения происходит широковещательная рассылка на все нижележащие области видимости, которые обрабатывают их по логике приложения.

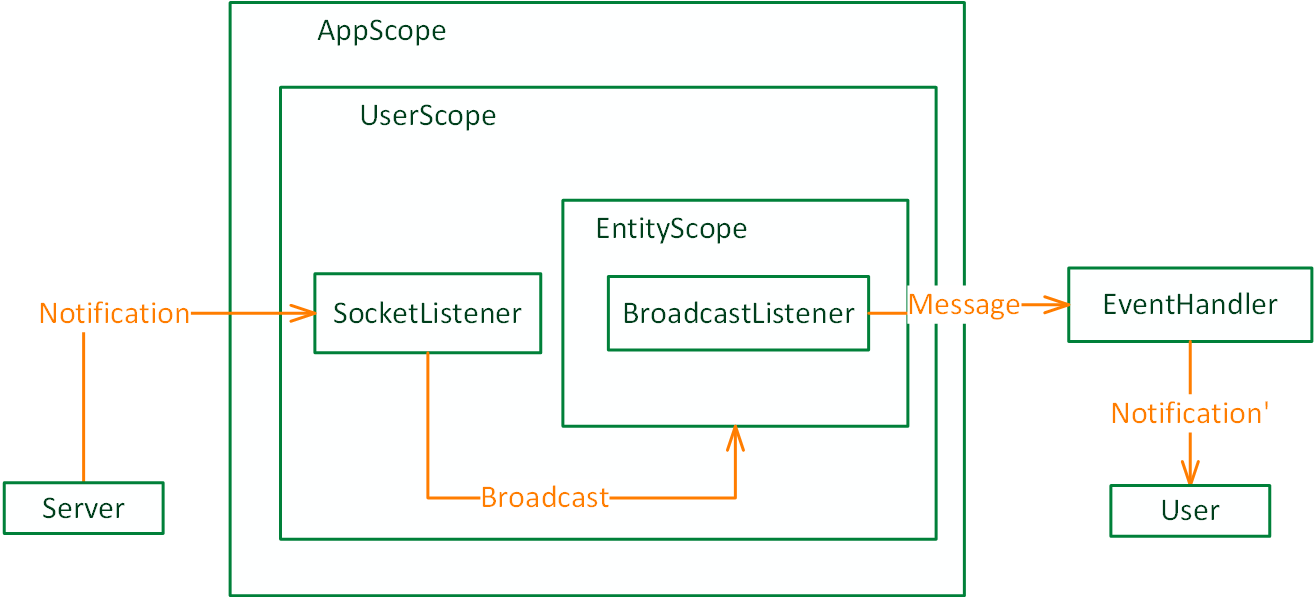


Рисунок 3.4.2. – Клиентская часть архитектуры системы оповещений.

* 1. Размещение

Диаграмма размещения представлена на рисунке 3.5.1.

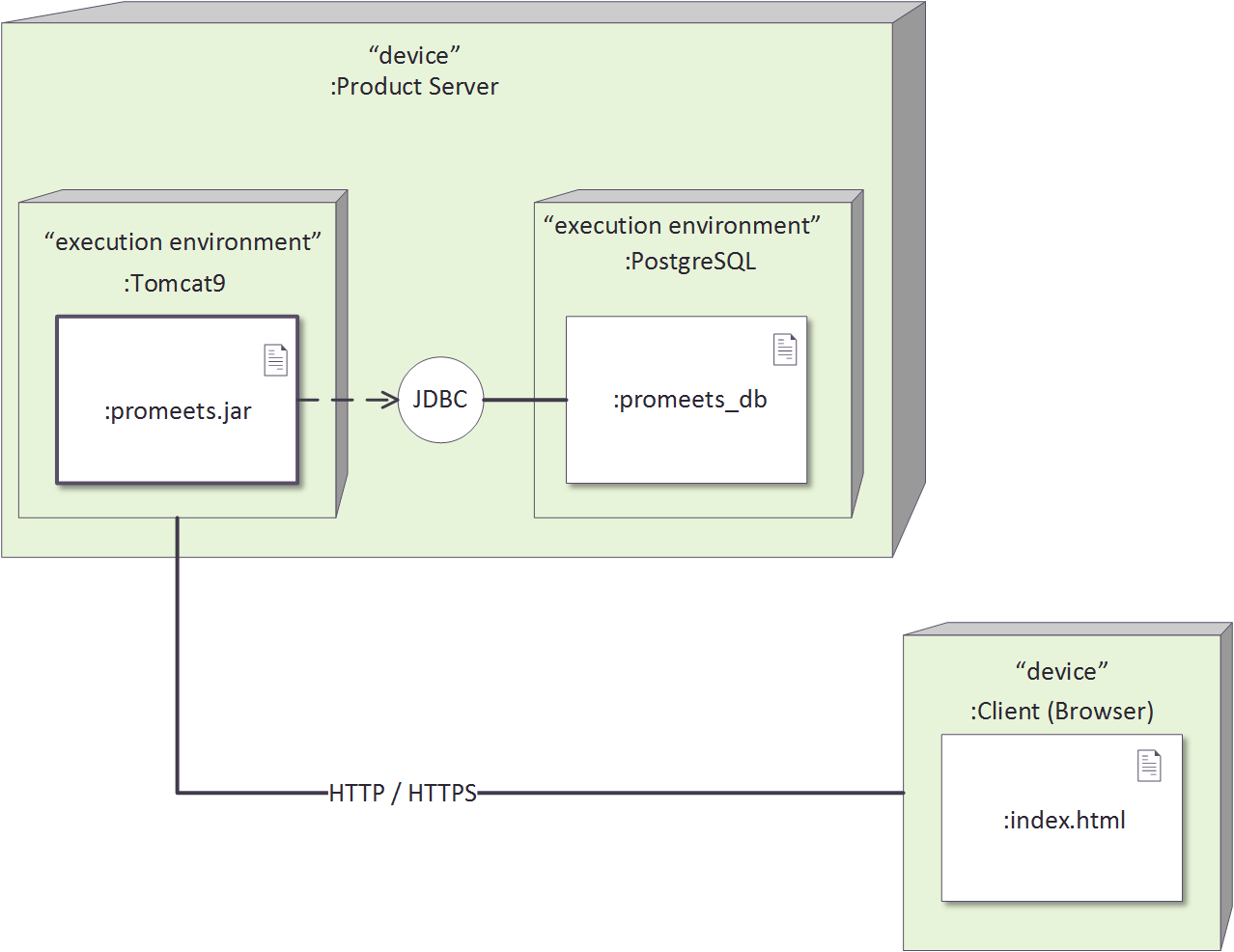


Рисунок 3.5.1. – Диаграмма размещения.

* 1. Тестирование и использование

Тестирование производилось в три этапа:

* + - Автоматическое тестирование бизнес логики на стороне сервера
    - Тестирование REST API и клиентской бизнес логики
    - Тестирование всего продукта

На рисунках 3.7.1. – 3.7.6. изображены основные странички приложения.

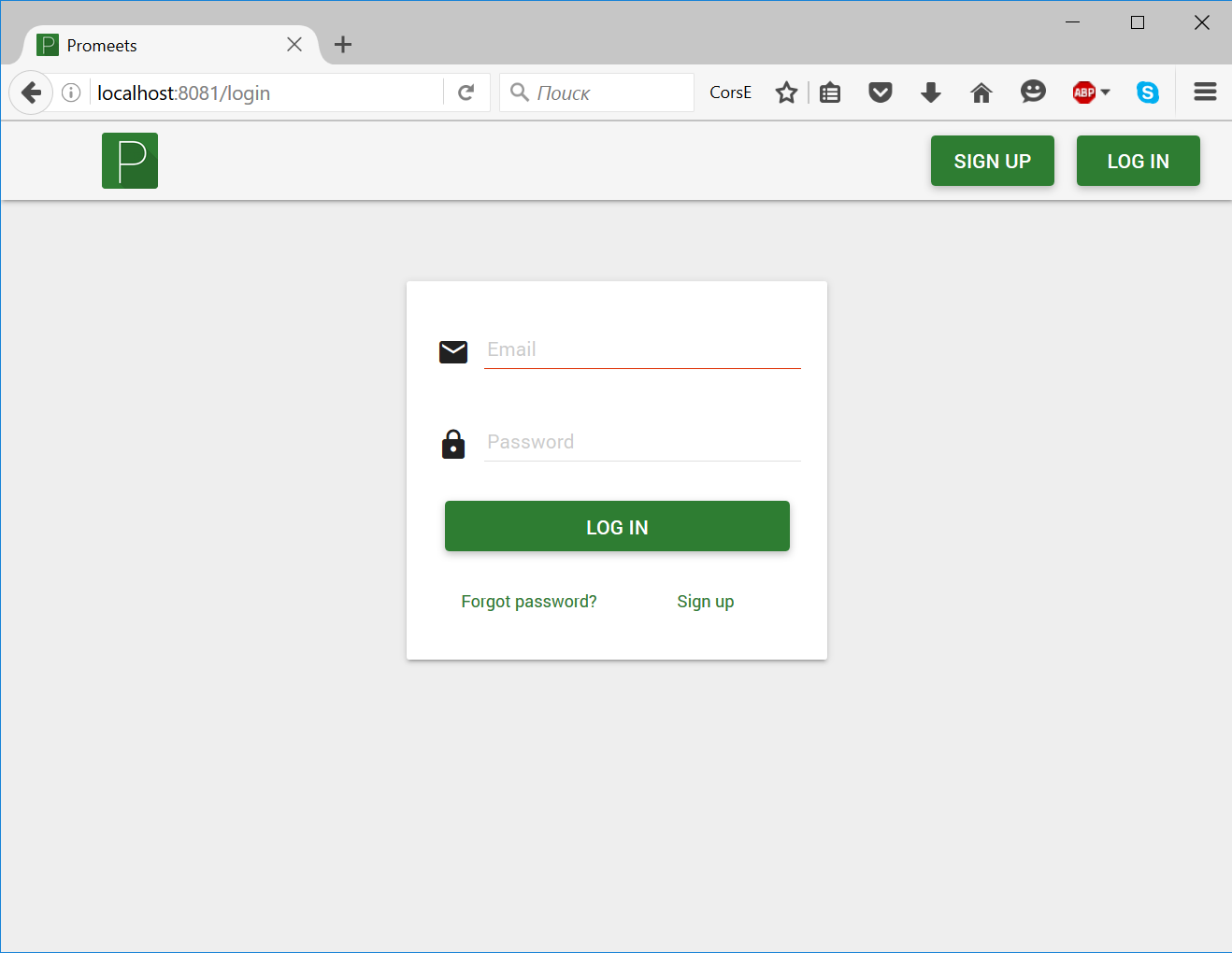


Рисунок 3.7.1. - Форма аутентификации.

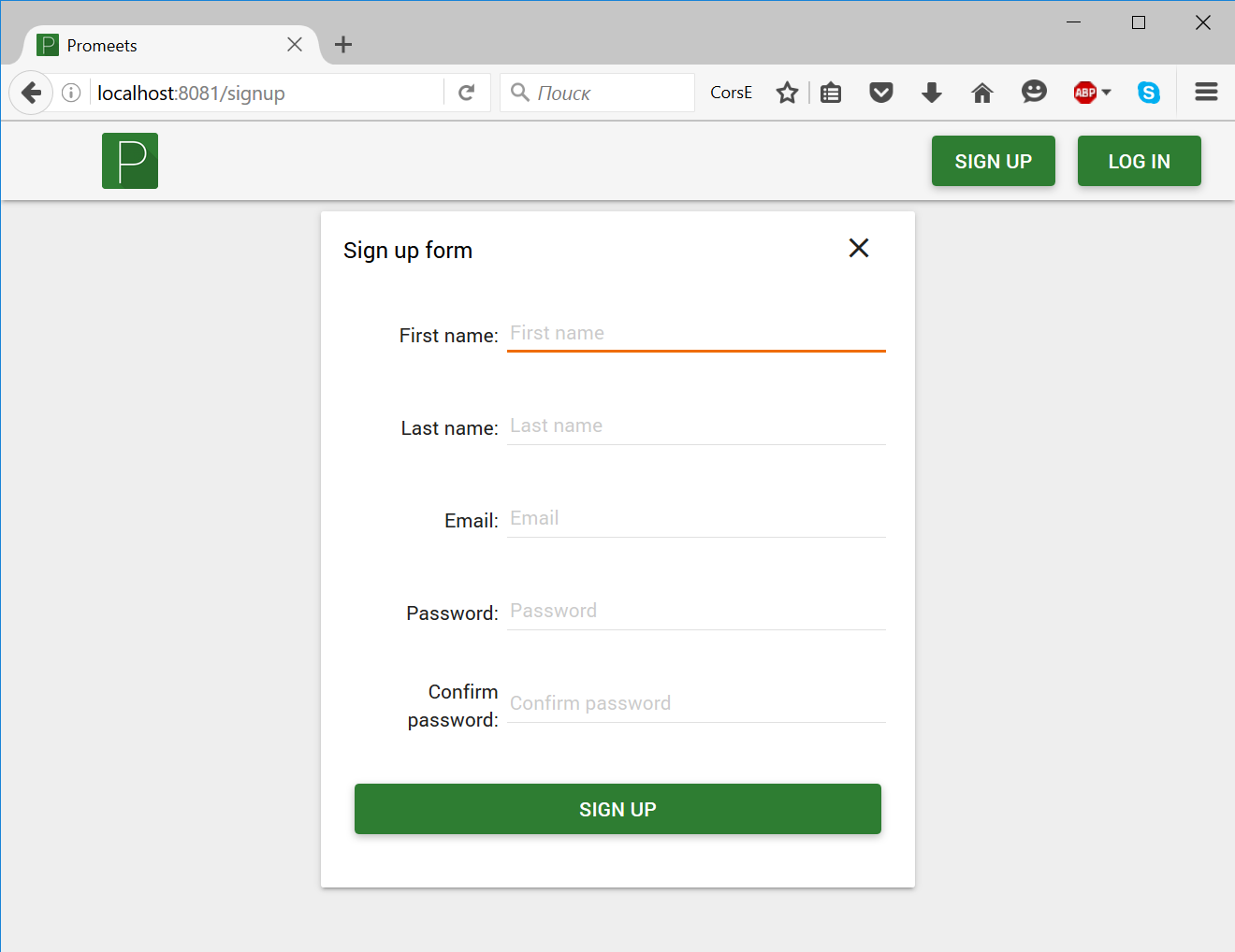


Рисунок 3.7.2 - Форма регистрации.

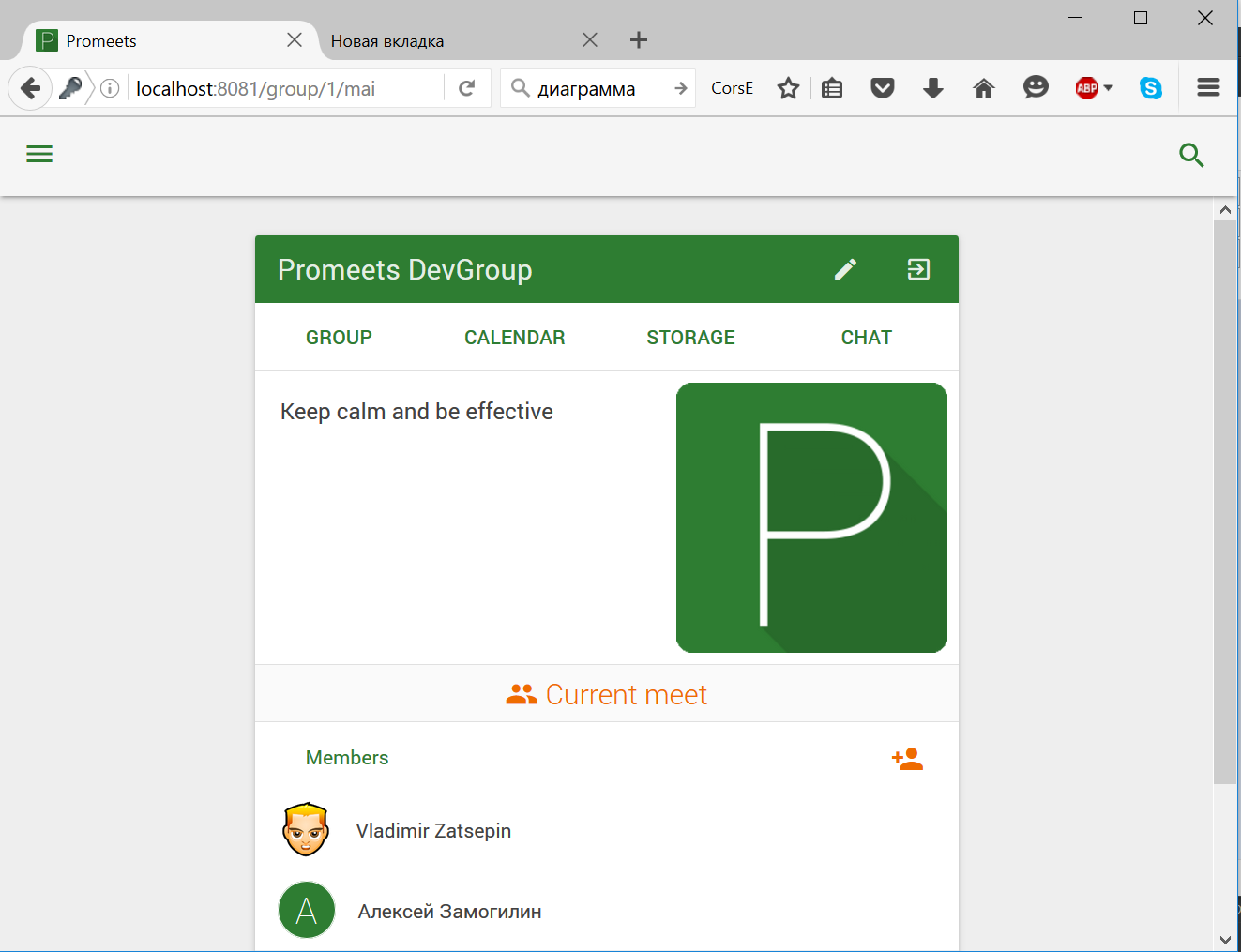


Рисунок 3.7.3. - Страница группы.

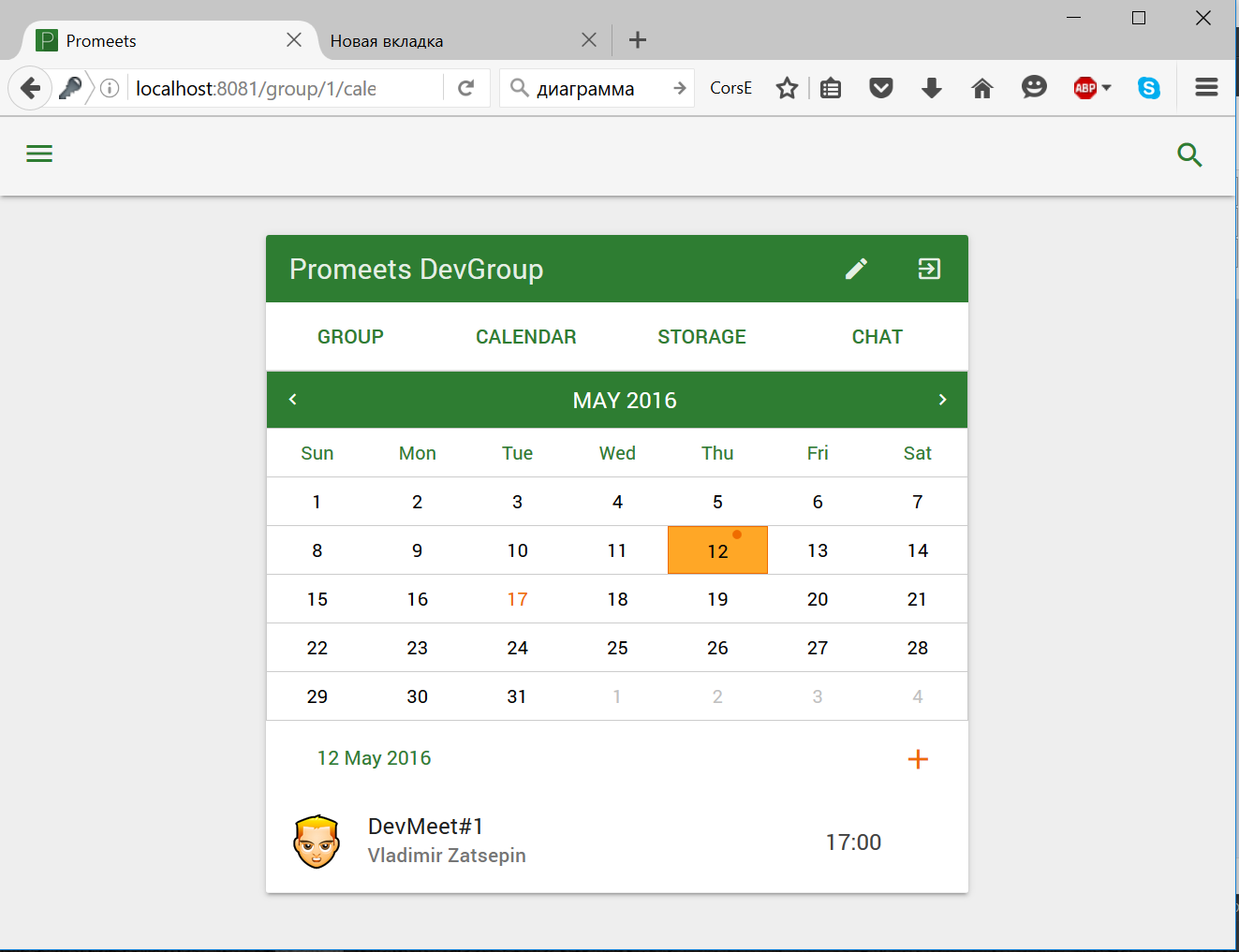


Рисунок 3.7.4. - Страница календаря в группе.

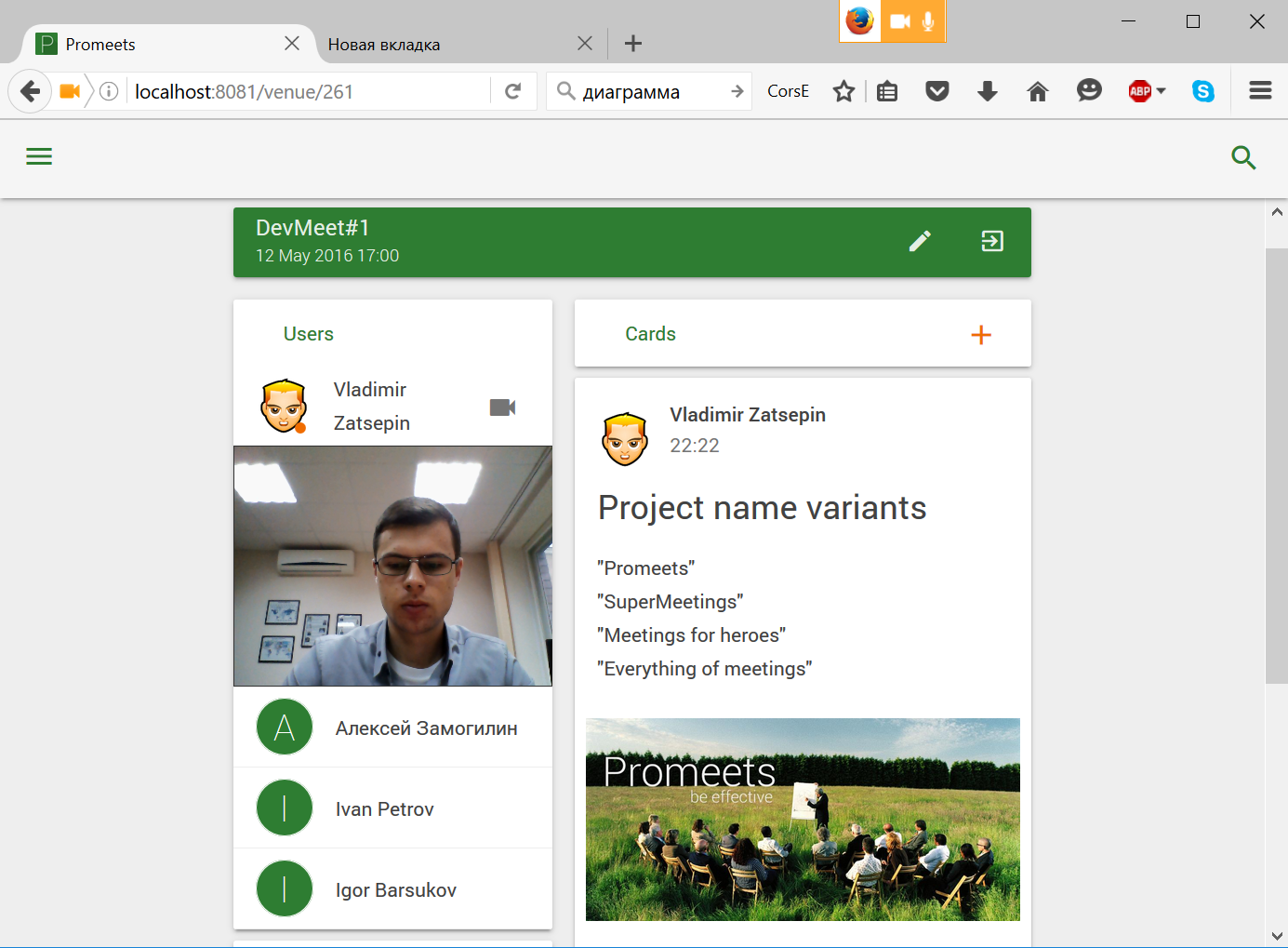


Рисунок 3.7.5. - Страница встречи группы (1).

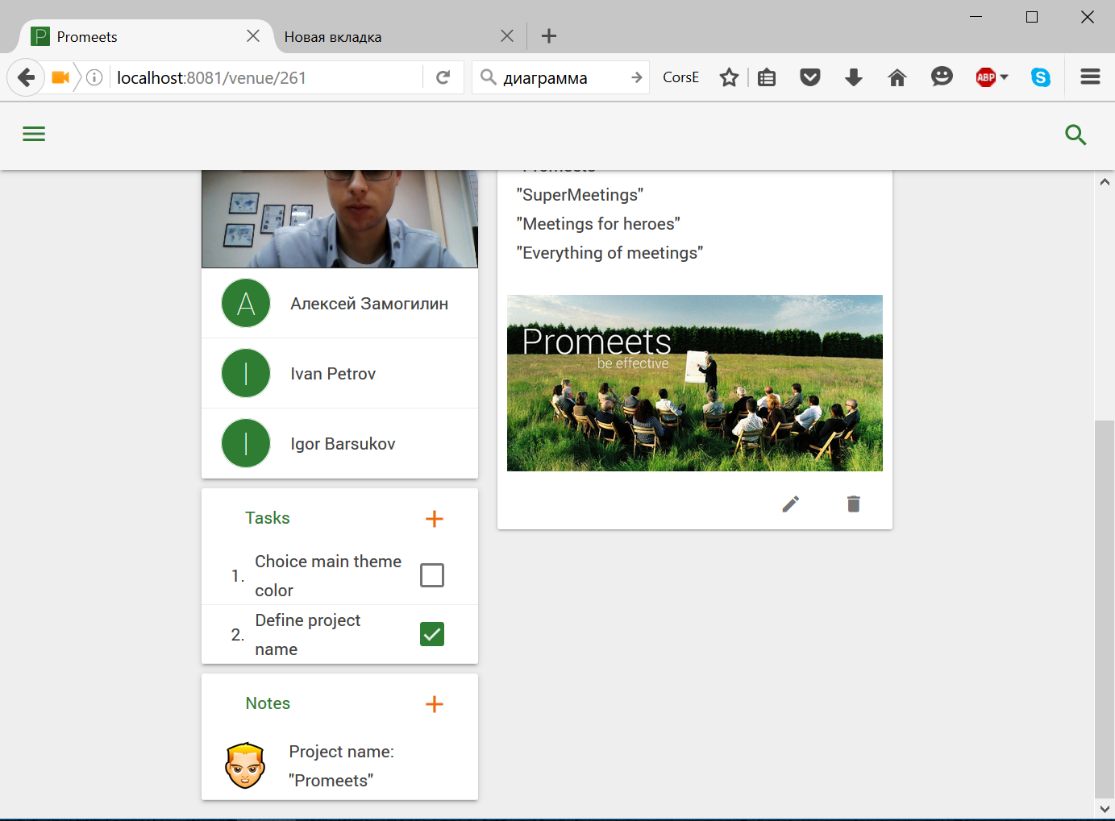


Рисунок 3.7.6. - Страница встречи группы (2).

Заключение

В рамках данной работы был разработан сервис для организации и проведения удаленных аудио-видео конференций. В ходе реализации были решены следующие задачи:

* Проанализирован рынок сервисов, предоставляющих услуги по реализации удаленных аудио-видео сервисов.
* Реализовано приложение, которое позволяет:
  + Создавать группы и приглашать туда других пользователей;
  + В рамках группы организовывать встречи;
  + В рамках встречи участвовать в аудио-видео конференции, а также обмениваться различными файлами.
* В рамках приложения реализована интерактивная система оповещений.