# **РЕФЕРАТ**

Выпускная квалификационная работа по теме «Корпоративная социальная сеть с элементами контроля рабочего процесса» содержит 52 страницы, 1 список сокращений, 1 приложение. При выполнении исследовано 11 источников. Использовано 26 рисунков.

Цель работы: создание сайта корпоративной социальной сети, который поможет организациям с различными областями деятельности, обеспечить взаимодействие сотрудников, а также предоставит им удобный универсальный инструмент для управления рабочим процессом.

Объект работы: веб-приложение для организаций c возможностями взаимодействия между сотрудниками и управления рабочим процессом.

В результате выполнения ВКР были выполнены следующие задачи:

* анализ существующих решений в предметной области;
* выбор программных средств разработки веб-приложения;
* проектирование модели базы данных;
* реализация клиентской и серверной частей веб-приложения;
* тестирование веб-приложения;
* анализ полученных результатов работы.

В выпускную квалификационную работу входит введение, 3 главы и заключение.

Во введении ставится цель, и выполняется ее декомпозиция на задачи.

В первой главе проводится анализ предметной области, рассматриваются существующие аналоги, формулируются основные требования к веб-приложению и выбираются инструменты разработки.

Во второй главе разрабатываются: архитектура серверной части и клиента, структура базы данных, описываются функции отдельных модулей.

В третьей главе описана реализация основных элементов веб-приложения. Составлена инструкция разработчика.

В заключении подводятся итоги по выполненной работе.

**СОДЕРЖАНИЕ**

[РЕФЕРАТ 1](#_Toc69874010)

[СОДЕРЖАНИЕ 2](#_Toc69874011)

[ВВЕДЕНИЕ 4](#_Toc69874012)

[1 Анализ задания на ВКР 6](#_Toc69874013)

[1.1 Анализ предметной области 6](#_Toc69874014)

[1.1.1 Корпоративная социальная сеть 6](#_Toc69874015)

[1.1.2 Анализ существующих аналогов 6](#_Toc69874016)

[1.1.2.1 Социальная сеть SBIS 7](#_Toc69874017)

[1.1.2.2 Социальная сеть Workplace 9](#_Toc69874018)

[1.1.2.3 Социальная сеть Битрикс24 10](#_Toc69874019)

[1.1.2.4 Вывод по анализу аналогов 11](#_Toc69874020)

[1.2 Разработка спецификации требований 12](#_Toc69874021)

[1.2.1 Функциональные требования к веб-приложению 13](#_Toc69874022)

[1.3 Выбор средств разработки 17](#_Toc69874023)

[1.3.1 TypeScriprt 18](#_Toc69874024)

[1.3.2 Node.js 19](#_Toc69874025)

[1.3.3 NestJS 19](#_Toc69874026)

[1.3.4 MongoDB 20](#_Toc69874027)

[1.3.5 Angular 21](#_Toc69874028)

[1.3.6 Обмен сообщениями в реальном времени 22](#_Toc69874029)

[1.4 Вывод по разделу 23](#_Toc69874030)

[2 Проектирование и программная реализация веб-приложения 24](#_Toc69874031)

[2.1 Алгоритм взаимодействия сервера с клиентом 24](#_Toc69874032)

[2.2 Разработка архитектуры базы данных 26](#_Toc69874033)

[2.3 Разработка архитектуры серверной части веб-приложения 26](#_Toc69874034)

[2.3.1 Модуль для регистрации и аутентификации 27](#_Toc69874035)

[2.3.2 Модуль для работы с сущностью пользователя 29](#_Toc69874036)

[2.3.3 Модуль для работы с сущностью организации 29](#_Toc69874037)

[2.3.4 Модуль для работы с сущностью сообщения из чата 30](#_Toc69874038)

[2.3.5 Модуль для работы с сущностью чата 31](#_Toc69874039)

[2.3.6 Модуль для работы с сущностью группы 32](#_Toc69874040)

[2.3.7 Модуль для работы с сущностью информационного поста в группе 33](#_Toc69874041)

[2.3.8 Модуль для работы с сущностью команды 34](#_Toc69874042)

[2.3.9 Модуль для работы с сущностью задачи на kanban-доске 35](#_Toc69874043)

[2.4 Выводы по главе 36](#_Toc69874044)

[3.1.1 Страницы аутентификации и регистрации 36](#_Toc69874045)

[3.1.2 Личная страница сотрудника 37](#_Toc69874046)

[3.1.3 Страница со списком организаций 38](#_Toc69874047)

[3.1.4 Страница с списком чатов 39](#_Toc69874048)

[3.1.5 Страница с чатом 39](#_Toc69874049)

[3.1.6 Страница со списком сотрудников 40](#_Toc69874050)

[3.1.4 Страница команды и kanban-доски 41](#_Toc69874051)

[3.2 Инструкция разработчика 43](#_Toc69874052)

[Теперь при запросе к серверу можно попасть на разработанный сайт. 45](#_Toc69874053)

[ЗАКЛЮЧЕНИЕ 46](#_Toc69874054)

[СПИСОК СОКРАЩЕНИЙ 47](#_Toc69874055)

[Список использованных источников 48](#_Toc69874056)

[ПРИЛОЖЕНИЕ А 49](#_Toc69874057)

**ВВЕДЕНИЕ**

В настоящее время существует множество различных социальных сетей, в которых люди могут развлекаться, знакомиться и общаться, но использование их для работы не всегда является удобным, однако большинство именно так и делает. Согласитесь, что многие сотрудники компаний хотели бы явно отделить рабочую деятельность от своих личных страниц в социальных сетях. Это значит, что сотрудникам было бы намного удобнее будь под рукой так называемая корпоративная социальная сеть. И подобные вещи уже существуют, однако немногие из них могут предложить в дополнение к этому какой-либо удобный инструмент для управления рабочим процессом.

Актуальностью выбранной темы является потребность современных организаций в удобном инструменте, который даст их сотрудникам способ быстрого и привычного для многих современных людей обмена сообщениями, а также будет содержать в себе универсальное средство для управления рабочим процессом.

В качестве универсального средства для управления рабочим процессом в проекте предполагается использовать kanban-доску, которая может идеально подойти к большинству задач касающихся управления рабочим процессом, начиная от контроля работы курьеров и доставщиков пиццы до управления командой разработчиков программного обеспечения.

Целью данной выпускной квалификационной работы является создание сайта корпоративной социальной сети, который поможет организациям с различными областями деятельности обеспечить взаимодействие сотрудников, а также предоставит им удобный универсальный инструмент для управления рабочим процессом. Для достижения данной цели в работе решаются следующие задачи:

* анализ существующих решений в предметной области;
* выбор программных средств разработки веб-приложения;
* проектирование модели базы данных;
* программная реализация клиентской и серверной частей веб-приложения;
* тестирование веб-приложения;
* анализ полученных результатов работы.

# **1 Анализ задания на ВКР**

# **Анализ предметной области**

# **1.1.1 Корпоративная социальная сеть**

В наше время все так или иначе понимают, что такое социальная сеть, однако в рамках данной работы стоит дать этому точное определение. Социальная сеть — это онлайн-платформа, которая используется для общения, знакомств, создания социальных отношений между людьми, которые имеют схожие интересы или офлайн-связи, а также для развлечения (музыка, фильмы и т.д.) и работы.

Первая социальная сеть появилась в 1995 году и на сегодняшний день они так разрослись, что почти каждый человек зарегистрирован хотя бы в одной социальной сети.

Существуют разные подвиды социальных сетей, но в данной работе разрабатывается корпоративная социальная сеть, а значит стоит сказать пару слов об этом подвиде. Такая сеть отличается от других тем, что служит для организации внутрикорпоративного общения и взаимодействия.

Корпоративная социальная сеть по сути очень похожа на обычную, но с расширенными возможностями, полезными функциями для компаний. Основная идея такой сети – обеспечить лёгкое и быстрое общение между сотрудниками, облегчить контакты между подчинёнными и руководителем.

Дополнительными возможностями такой сети могут выступать различные функции контроля рабочего процесса.

# **1.1.2 Анализ существующих аналогов**

На сегодняшний день уже существует несколько аналогов разрабатываемого сайта, поэтому требуется рассмотреть существующие решения, выделить их преимущества и недостатки. Основной фокус будет составляться на том, какие инструменты есть на сайте для контроля рабочего процесса.

# **1.1.2.1 Социальная сеть SBIS**

Первым аналогом разрабатываемого веб-приложения является корпоративная сеть SBIS [1]. Данный сайт предоставляет множество способов обмена информации между сотрудниками, как посредством сообщений, так и по видеосвязи.

На сайте есть:

* возможность поручения задач и отслеживание хода их выполнения;
* личная страница сотрудника с возможностью публикации на ней новостей;
* новостная лента и обсуждения, репосты, лайки и комментарии;
* обмен сообщениями и документами;
* рабочие группы и мероприятия;
* видеосовещания и вебинары;
* конкурсы и опросы.

На рисунке 1 представлена часть возможностей SBIS и интерфейс. На рисунке 2 приведена личная страница сотрудника.

Для контроля рабочего процесса SBIS предлагает использовать рабочие группы. В них можно планировать события, создавать обсуждения, хранить документы и файлы различных типов. После событий можно подводить итоги и выдавать задачи сотрудникам.

К недостатку можно отнести то, что данная социальная сеть является платной.

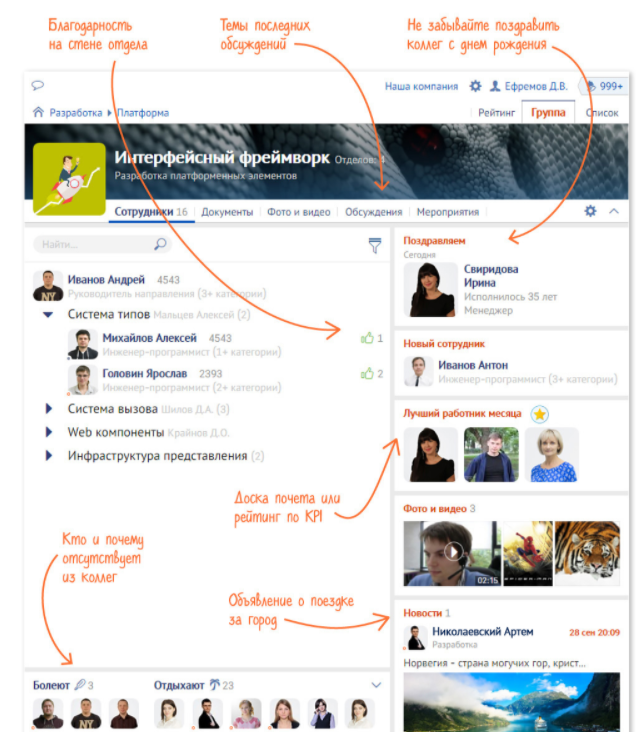


Рисунок 1 – Возможности SBIS



Рисунок 2 – Личная страница в SBIS

# **1.1.2.2 Социальная сеть Workplace**

Workplace [2] – это корпоративная платформа, разработанная Facebook, Inc., которая включает использование групп, обмен мгновенными сообщениями и новостную ленту. Разработкой руководит Жюльен Кодорниу. На рисунке 3 представлена страница новостной ленты компании на сайте Workplace.

По сути своей Workplace почти ничем не отличается от стандартного Facebook, кроме того, что он является отдельным для каждой компании.

В качестве элементов контроля рабочего процесса здесь можно считать создание событий. Однако группы здесь не очень подходят для контроля рабочего процесса как это сделано в SBIS. В группах нет наглядного и удобного инструмента поручения задач и просмотром хода их выполнения, а также к минусу можно отнести то, что проект является платным.

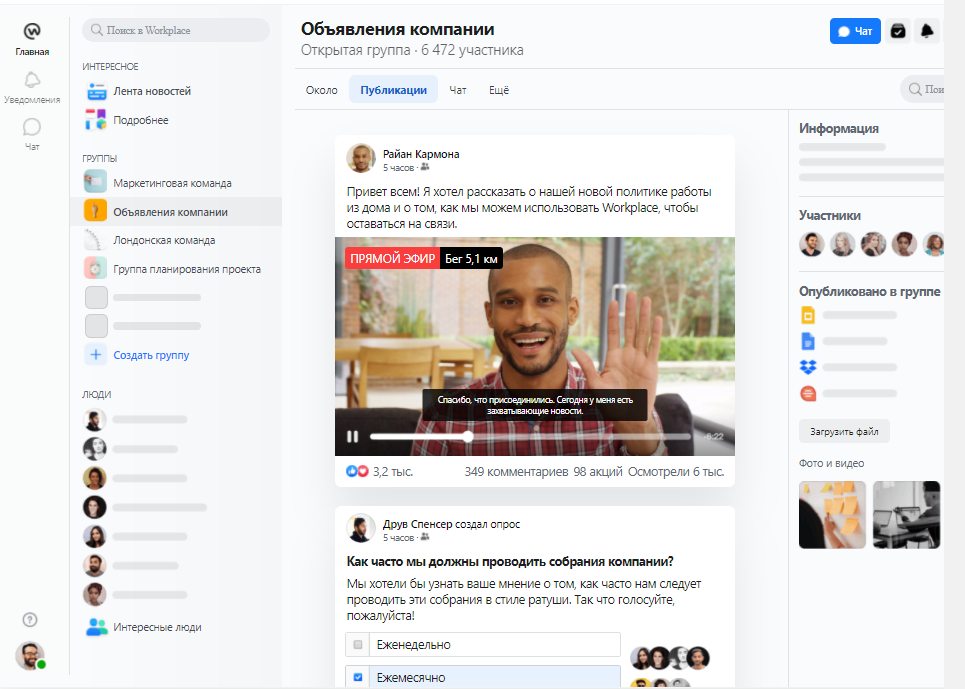


Рисунок 3 – сайт Workplace

# **1.1.2.3 Социальная сеть Битрикс24**

Еще одним аналогом разрабатываемой социальной сети является «Битрикс24» [3]. Это очень крупный проект, который появился в 2012 году. Он был разработан российской компанией ООО «1С-Битрикс» и на данный момент им активно пользуются большие компании по всему миру. У данной социальной сети есть как облачная версия, так и версия, которая может быть установлена на собственный сервер.

«Битрикс24» включает в себя следующий функционал:

* CRM;
* менеджер задач;
* чаты;
* конструктор сайтов;
* конструктор интернет-магазинов;
* облачное хранилище и онлайн-документы;
* календарь;
* бизнес-процессы;
* учёт рабочего времени;
* сквозную аналитику;
* аудио/видеозвонки и видеоконференции в HD;
* генератор документов и отчётов.

Как видно из списка у данного проекта уже имеется огромнейший функционал, который может быть полезен организациям, а также он уже имеет такой инструмент как kanban-доска.

К главному недостатку можно отнести то, что «Битрикс24», как и другие аналогичные проекты является платным. Однако в нем могут работать компании до 25 человек.

# **1.1.2.4 Вывод по анализу аналогов**

У уже существующих корпоративных социальных сетей имеется множество разных функций. Многие из них схожи. Однако рассмотренным сайтам не хватает более удобного инструмента по выдаче задач и их контролю, когда сотрудники работают в командах. Таким инструментом может выступать kanban-доска. Хотя у «Битрикс24» уже есть подобный функционал, но он как и другие аналоги является платным. Из того что они платные, следует недостаток в виде того, что один пользователь под одной учетной записью не может состоять в нескольких организациях.

Общий недостаток рассмотренных аналогов состоит в отсутствии закрытых групп, которые не должны отражаться в списке всех групп компании. Такой подход предоставляет больше личной свободы сотрудникам от контроля руководства.

**1.1.2 Kanban-доска**

Прежде чем говорить о том, что такое kanban-доска стоит ввести определение kanban.

Kanban — это метод для определения, управления и совершенствования сервисов, поставляющих результаты умственного труда, такие как экспертные и креативные услуги, а также разработку физических или программных продуктов.

В качестве сервисов выступают один и более человек.

Kanban-метод базируется на визуализации нематериальной деятельности, умственного труда, в целях правильной работы сервисов, достижения баланса между работой, которую необходимо выполнить и возможностями сервисов. С этой целью используются kanban-доска. На доске изображена поточная система, рабочие элементы которой проходят различные этапы обработки в направлении слева направо. На физической доске рабочими элементами являются разноцветные стикеры, на которых пишутся различные задачи которые нужно выполнить.

Она представляет из себя таблицу со множеством колонок, которые являются этапами через которые должна пройти каждая задача прежде чем будет считаться выполненной. Способов построения kanban-доски очень много, но базовые принципы у них одинаковые. Самый простой вариант kanban-доски (рисунок 4) состоит из трех колонок:

* что нужно сделать;
* задачи, над которыми ведется работа;
* выполненные задачи.

Такой принцип выполнения задач позволяет контролировать рабочий процесс и оптимизировать его, ведя статистику.



Рисунок 4 – Макет kanban-доски

Изначально задачи добавляются в первой колонке и нигде больше, далее задача на каждом этапе должна закрепляться за каким-то сотрудником или целой командой, которые работают над ней и передают ее на следующий этап.

Обычно kanban-доска реализуется в физическом виде, но это не всегда удобно.

# **1.2 Разработка спецификации требований**

# **1.2.1 Функциональные требования к веб-приложению**

Функциональные требования выражены в виде диаграммы прецедентов (рисунок 5).

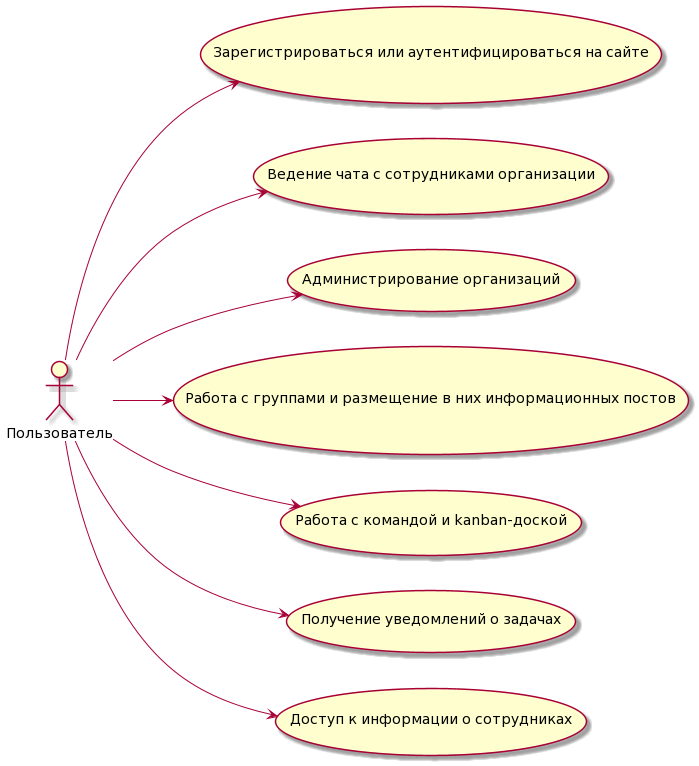


Рисунок 5 – Диаграмма прецедентов

Описание прецедентов:

1. Прецедент «Зарегистрироваться или аутентифицироваться на сайте»;

Так как на сайте у каждого пользователя должна быть личная страница и действия, которые должны быть авторизованы необходимо реализовать страницы с регистрацией и аутентификацией.

Страница для регистрации должна иметь ссылку на страницу для аутентификацию и форму с кнопкой для регистрации. Форма должна включать следующие обязательные поля:

* электронная почта;
* фамилия;
* имя;
* отчество;
* дата рождения;
* логин;
* пароль;
* повторите пароль.

Необязательные поля:

* аватар;
* номер телефона.

После успешной регистрации должен быть осуществлен переход на страницу для входа.

Страница для входа должна включать поля для логина и пароля, а также ссылку на страницу регистрации и кнопку для входа.

1. Прецедент «Ведение чата с сотрудниками организации».

Социальная сеть должна включаться в себе возможность переписки между пользователями, что и обуславливает наличие данного прецедента.

Необходимо разработать страницу со списком чатов, в которых состоит текущий пользователь и возможностью войти в один из чатов или создать новый. При создании нового чата в него можно добавить только сотрудников текущей организации.

При переходе в чат, должна открываться страница со следующими возможностями: написать в чат, просмотреть историю сообщений, получение новых сообщений из чата в реальном времени.

На странице чата должны быть возможности просмотра добавленных в чат сотрудников, добавления и удаления сотрудников из чата.

1. Прецедент администрирования организаций.

Разрабатываемая социальная сеть предназначена для использования организациями, следовательно необходимо создать страницу, в которой у пользователя должны быть следующие возможности:

* просмотр списка организаций, в которых он состоит;
* создание новой организации;
* выбор текущей организации. От выбранной организации должно зависеть то, какие сотрудники, команды и группы будут отображаться на соответствующих страницах;
* у организаций из списка, в которых создателем является текущий пользователь должны присутствовать возможности менять название, аватар и описание, а также функционал по добавлению и удалению сотрудников из организаций. Руководитель организации также должен иметь возможность менять должности у своих сотрудников.

1. Прецедент «Работа с группами и размещение в них информационных постов».

В социальной сети должен присутствовать инструмент, который служит в качестве доски объявлений с возможностью размещения различного контента, поэтому на сайте необходимо реализовать следующие страницы:

* страницы со списком всех открытых и закрытых групп, в которых состоит пользователь. На этой странице также должна присутствовать возможность создать группу, а также просмотреть историю информационных постов;
* страница группы, которая дает возможность разместить контент как с указанием автора, так и без. Если пользователь является создателем организации, он должен иметь функционал по изменению названия, описания и аватара группы, а также, если группа является закрытой – возможность добавления сотрудников организации к которой относится группа.

1. Прецедент «Работа с командой и kanban-доской».

В задание ВКР также входит реализация средства для объединения сотрудников для работы, выдачи им задач и контроля над их выполнением. Для этого необходимо создать следующие страницы:

* страница со списком команд, в которых состоит пользователь. На этой странице также должна присутствовать возможность создать новую команду;
* страница команды с kanba-доской. На этой странице должны быть следующие возможности для работы с командой: просмотр состава команды; если текущий пользователь является создателем команды, он должен иметь функцию удаления сотрудников из команды и добавления сотрудников текущей организации в нее, а также право давать им статус администратора. Функционал kanban-доски для администратора команды должен включать: возможность добавления задач, имеющих крайний срок и без него; возможность редактирования задач и прикрепления сотрудников к ним; возможность изменения статусов kanban-доски и добавление новых; возможность принимать задачи, после чего они должны отправляться в историю задач. Обычный сотрудник должен иметь возможность отвечать на задачи, к которым он был прикреплен, а также смотреть историю задач.

1. Прецедент «Получение уведомлений о задачах».

Так как некоторые задачи на kanban-доске могут иметь крайний срок, необходимо реализовать страницу, которая будет отображать задачи крайний срок, которых подходит к концу.

1. Прецедент «Доступ к информации о сотрудниках».

Для обеспечения коммуникаций между сотрудниками и возможности побольше узнать о коллегах необходимо реализовать следующие страницы:

* страница со списком сотрудников текущей организации. Пользователи, которые в данный момент находятся онлайн должны отображаться особым образом;
* страница личной страницы. На этой странице можно просмотреть информацию о пользователе, в том числе информацию о том, в каких организациях он состоит и находится ли он сейчас онлайн. Если открытая личная страница принадлежит текущему пользователю, то у него должна быть возможность редактировать ее содержимое. Если личная страница принадлежит другому сотруднику – должна присутствовать возможность создать чат с ним.

# **1.3 Выбор средств разработки**

Выбор средств разработки является очень важной частью процесса проектирования. Этот выбор может сыграть решающую роль в том, насколько сложным и долгим будет процесс разработки, поэтому к нему надо отнестись очень ответственно. Существует немало рабочих и проверенных временем веб-стеков.

Под веб-стеком понимается набор программ, которые последовательно обрабатывают поток данных, циркулирующий между клиентом и сервером. Компонентами стека принято считать:

* операционную систему, на которой выполняются все программы на сервере;
* веб-сервер, который принимает запросы и отвечает на них либо в формате JSON, либо в виде страниц на языке HTML;
* систему управления базами данных, которая применяется для создания базы данных, а также внесения, поиска, изменения и удаления информации из базы данных;
* серверный язык программирования, на котором написаны программы на стороне сервера.

Разнообразие видов каждого из данных компонентов очень велико и не все они помогут сочетаться с другими. Из всех стеков я решил использовать обновленный стек MEAN. Он отличается от других тем, что платформа независим, и при разработке как серверной, так и клиентской части использует инструменты в основе, которых лежат одни и те же технологии. А именно сервер node.js и языки Typescript или Javascript.

Стек MEAN расшифровывается как Mongodb, Express, Angular и Node.js. Обновленный стек MEAN подразумевает использование не чистого Node.js, а использование фреймворка NestJS. Для контроля версий разрабатываемого программного обеспечения будет использоваться Git.

# **1.3.1 TypeScriprt**

Стоит начать описывать веб-стек именно с языка, который используется в разработке как на Angular, так и на NestJS.

TypeScript [4] — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript отличается от JavaScript возможностью явного [статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использования кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и ускорить выполнение программ внося в нее шаблон async/await.](https://en.wikipedia.org/wiki/Manifest_typing)

Компилятор TypeScript называется tsc, написан на языке TypeScript и может быть скомпилирован в стандартный JavaScript, и может быть запущен на любом движке JavaScript, например, в браузере. Компилятор идет вместе с сервером сценариев, который может запускать компилятор. Также он доступен в виде пакета для node.js, который использует node.js в качестве сервера.

Является отличным инструментом для написания серверных приложений.

# **1.3.2 Node.js**

Следующий ключевой компонент – это Node.js, выполняющий роль веб-сервера при разработке на Angular и NestJS.

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения [5]. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера. В основе Node.js лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом/выводом.

# **1.3.3 NestJS**

NestJS — это масштабируемая серверная среда JavaScript, созданная на основе TypeScript, которая сохраняет совместимость с JavaScript, что делает её действенным инструментом для создания эффективных и надежных серверных приложений [6]. Он имеет модульную архитектуру, которая обеспечивает структурный шаблон проектирования для разработки Node.js.

Фреймворк NestJS выводит Typescript на совершенно новый уровень, предоставляя современные инструменты бэкенд-разработчику для создания долговечных и высокопроизводительных приложений с использованием компонентов, провайдеров, модулей и других полезных высокоуровневых абстракций.

Уже из коробки он включает в себя Express.js и дает удобный интерфейс для его использования в виде декораторов.

Он создан для облегчения жизни разработчика, предоставляя правильные архитектурные подходы и диктуя свои правила.

Удобно также то, что создатели NestJS вдохновлялись архитектурой Angular, и освоение одного из этих фрэймворков гораздо облегчит понимание и освоения другого. Это сыграло не последнюю роль при выборе веб-стека.

NestJS также позволяет с легкостью интегрировать в свое приложение Swagger для описания и тестирования Rest API.

# **1.3.4 MongoDB**

MongoDB — документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных [7]. Применяется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

Удобна тем, что в сравнении с реляционными базами данных, нет необходимости в строгом формате документа, а также способствует большей скорости разработки.

В качестве ORM будет использоваться библиотека mongoose, которая была специально создана для работы с MongoDB.

ОРМ (объектно-реляционное сопоставление) – это технология программирования, которая позволяет взаимодействовать с базой данных с использованием объектно-ориентированного шаблона. Обычно под ОРМ подразумевается библиотека, которая реализует объектно-реляционное сопоставление. ОРМ сопоставляет записи нашей базы данных с объектами, которые мы определяем. Это позволяет нам писать много многоразовой логики для нашей базы данных, а также упрощает многие сложные запросы и может сократить раздувание нашей базы кода. Большинство ОРМ имеют встроенную защиту от SQL-инъекций [8].

В разрабатываемом проекте, для подключения к базе данных, будет использоваться Docker. Он позволяет использовать уже готовую к использованию БД MongoDB из контейнера, без необходимости устанавливать ее на свой компьютер.

# **1.3.5 Angular**

Angular - это фреймворк, который был написан на языке Typescript и применяемый для создания одностраничных клиентских приложений (SPA-приложение) [9]. Под SPA-приложением понимают веб-приложение, использующее единственный HTML-документ как оболочку для всех веб-страниц и организующее взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript файлы. Angular реализует основной и дополнительный функционал в виде набора библиотек, которые можно импортировать в приложения.

Начиная со 2-й версии, он был полностью переработан, и в своей основе использует компонентную архитектуру. Под компонентной архитектурой подразумевается то, что основными строительными блоками являются компоненты, которые определяют представления. Компоненты организованы в модули, собирающие связанный код в функциональные наборы.

Важной частью архитектуры также являются сервисы, которые предоставляют определенные функции не связанные напрямую с представлениями. Они вводятся в компоненты как зависимости, что позволяет сделать код модульным и многоразовым.

Это только краткое и неполное описание архитектуры, но даже освоение только основ дает разработчику понимание того, как можно создавать большие и легко расширяемые клиентские приложения.

**1.3.6 Обмен сообщениями в реальном времени**

Одной из задач разрабатываемого сайта является обмен сообщениями между сотрудниками в реальном времени, но его не получится корректно реализовать с помощью стандартных протоколов HTTP и HTTPs. Это связано с тем, что браузер и сервер работают по клиент-серверной архитектуре. Это означает, что для получения сообщения от другого пользователя наш браузер должен постоянно посылать запросы серверу с требованием предоставить новые сообщения. И на большинство таких запросов серверу будет нечего ответить, ведь они должны будут посылаться каждую секунду, чтобы имитировать получение сообщение в реальном времени, но люди еще не научились так быстро писать. Таким образом сервер будет сильно нагружен постоянными запросами.

Для решения этой проблемы придумали Web-socket – это протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени [10]. Для установления соединения браузер отправляет запрос серверу о том, что хочет перейти на Web-socket, а сервер в ответ подтверждает переход, и после этого соединение считается установленным. Установив соединения, браузер и сервер могут вести двунаправленный обмен сообщениями, это значит, что браузеру для получения сообщения от другого пользователя просто достаточно слушать сервер.

Из вышесказанного можно сделать вывод, что именно Web-socket соединение должно производиться для реализации в разрабатываемом сайте обмена сообщениями в реальном времени. Для использования WebSocket есть несколько библиотек. Я буду использовать socket.io, так как по сравнению с остальными она является более высокоуровневой [11].

# **1.4 Вывод по разделу**

В результате анализа задания на дипломное проектирование были сформулированы требования к разрабатываемому веб-приложению, было решено разрабатывать сайт с четким разделением front-end и back-end частей; так же был проведен анализ существующих решений. Выделен перечень функций, отсутствующих в аналогичных корпоративных социальных сетях и который необходимо реализовать. Был сформулирован стек используемых технологий, в частности стек MEAN и TypeScript как основного языка разработки.

# **2 Проектирование и программная реализация веб-приложения**

Разрабатываемое веб-приложение имеет 2 части: клиентскую и серверную (рисунок 6). Серверная часть включает в себя БД. Под клиентом здесь понимается SPA-приложение выполненное на Angular.

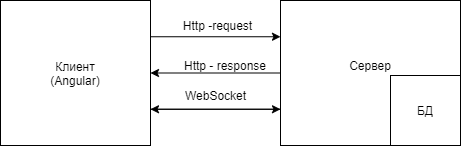


Рисунок 6 – Общая архитектура системы

# **2.1 Алгоритм взаимодействия сервера с клиентом**

Общий алгоритм взаимодействия браузера и сервера представлен на рисунке 7.

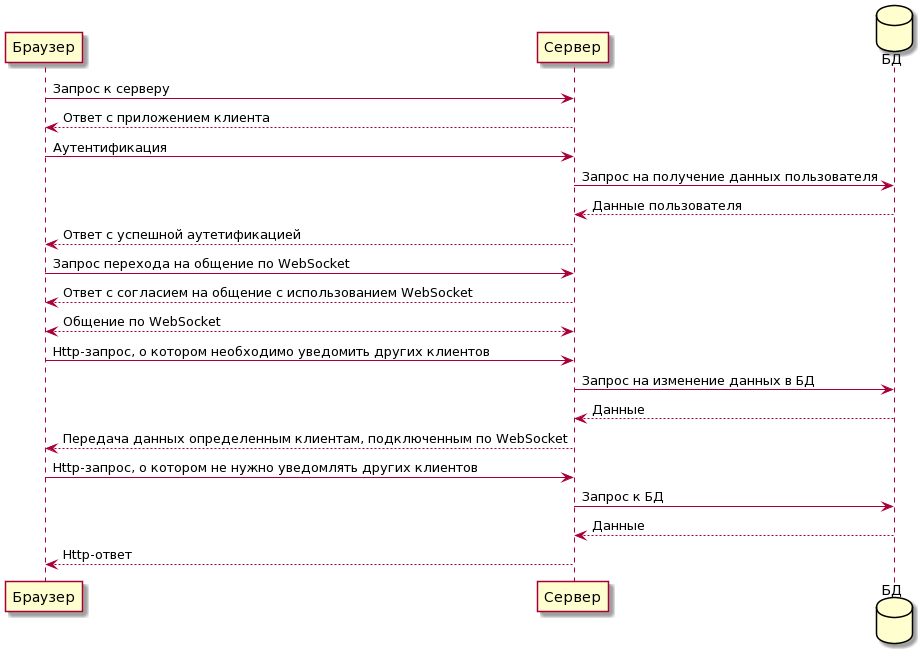


Рисунок 7 – Общая схема взаимодействия браузера, сервера и БД

Сначала браузер должен получить клиентское приложение для работы с интерфейсом сервера. Для этого он обращается к серверу и в ответ получает код SPA-приложения, который выполняется в браузере клиента.

Далее пользователь должен либо зарегистрироваться, либо аутентифицироваться. На рисунке 7 представлен вариант с аутентификацией. При получении запроса сервер запрашивает данные с базы данных и возвращает ответ с результатом аутентификации. В данном случае считаем, что она пройдена успешно. После успешной аутентификации клиентское приложение запрашивает переход на общение по WebSocket.

С этого момента инициатором общения может выступать не только клиент, но и сервер c помощью вызова событий, когда это необходимо, а именно, когда клиент сделал запрос подразумевающий изменение данных, которые необходимо отобразить в реальном времени другим клиентам. В случаях, когда сервер обрабатывает такой запрос, он делает запросы к БД и в случае успеха посылает нужные данные всем заинтересованным клиентам.

Запросы, не подразумевающие изменения в БД, которые необходимо показать другим клиентам, используют простое общение с помощью Http без вовлечения WebSocket.

# **2.2 Разработка архитектуры базы данных**

На рисунке 8 представлена ER-диаграмма базы данных. На диаграмме изображены основные сущности и отношения между ними. Точка перед свойством сущности означает, что оно обязательно.

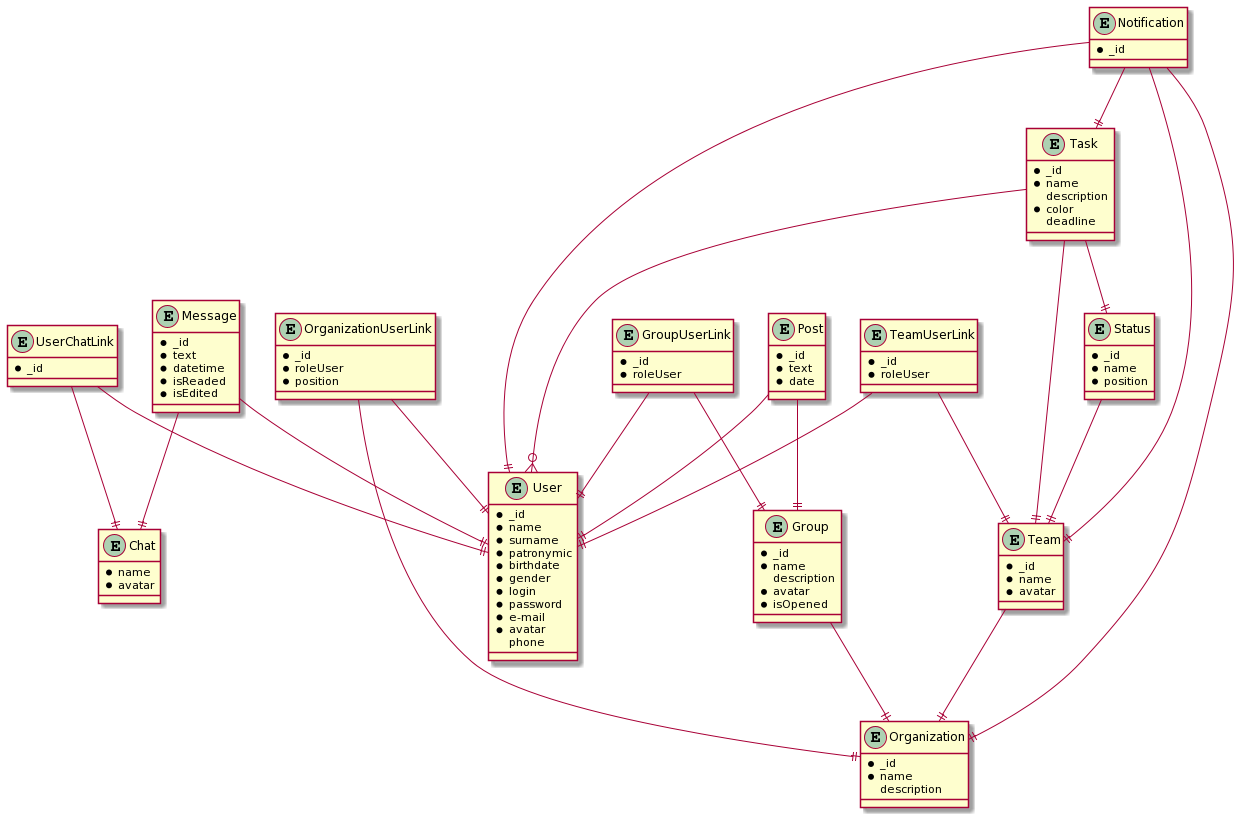


Рисунок 8 – ER-диаграмма БД

# **2.3 Разработка архитектуры серверной части веб-приложения**

При разработке алгоритма работы серверной части была спроектирована схема, представленная на рисунке 9. Каждый модуль будет описан отдельно в следующих подразделах пояснительной записки.

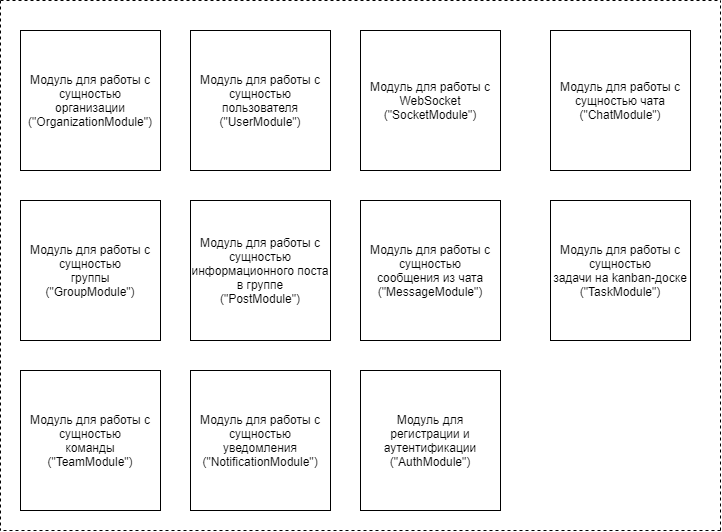


Рисунок 9 – Структурная схема серверной части веб-приложения

# **2.3.1 Модуль для регистрации и аутентификации**

В этом модуле определены конечные точки для регистрации и аутентификации. Во время сохранения данных о пользователе в БД пароль хешируется криптографической функцией bcrypt.

При аутентификации используется jwt-стратегия из библиотеки Passport.js. Passport.js предоставляет middleware для работы с аутентификацией и авторизацией.

JWT – стратегия подразумевает, что после аутентификации пользователю выдается специальный токен доступа. Токен состоит из трех частей: заголовка, полезных данных, подписи. Заголовок и полезные данные являются JSON объектами. В объекте заголовка обязательный ключ только один – алгоритм шифрования. В объект полезных данных помещается логин пользователя, пароль, время создания и время жизни токена. Подпись вычисляется алгоритмом безопасного хеширования HS256 на основе заголовка и полезных данных с добавлением секретного ключа. После вычисления подписи заголовок и полезные данные кодируются алгоритмом [Base64-URL](https://ru.wikipedia.org/wiki/Base64) после чего добавляется подпись и все три части разделяются точной.

Получив токен, клиент для подтверждения личности должен отправлять его в заголовке Authorization Http-запроса. Сервер получив токен расшифровывает его и проверяет его валидность. Если токен действителен, запрос допускается к следующей функции middleware или к конечной точке.

Конечные точки в формате Rest API модуля изображены на рисунке 10.

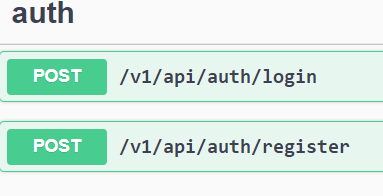


Рисунок 10 – Rest API модуля AuthModule

# **2.3.2 Модуль для работы с сущностью пользователя**

В модуле реализованы следующие функции:

* получение данных о текущем пользователе;
* изменение данных пользователя;
* получение данных пользователя по id.

Rest API модуля представлены на рисунке 11.

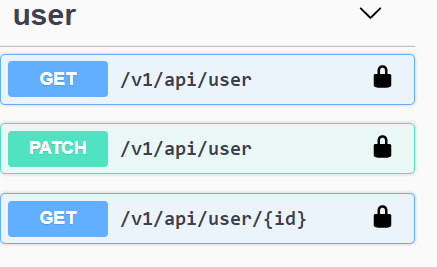


Рисунок 11 – Конечные точки UserModule.

# **2.3.3 Модуль для работы с сущностью организации**

Данный модуль предоставляет Rest API и функции для работы с организациями. В модуле есть следующие функции:

* создание организации;
* удаление организации. При удалении идет проверка, является ли пользователь создателем организации и если это так, то все записи связанные с этой коллекцией в БД удаляются;
* изменение названия, аватара или описания организации. При этом происходит проверка является ли текущий пользователь создателем организации;
* получение данных организации по id;
* покинуть организацию;
* нанять работника;
* получение всех организаций пользователя по его id;
* получение всех пользователей по id организации;
* назначение пользователю новой должности. Для этого нужно иметь роль администратора организации или ее создателя, а пользователь которому назначается должность состоять в организации.

Rest API модуля представлены на рисунке 12.

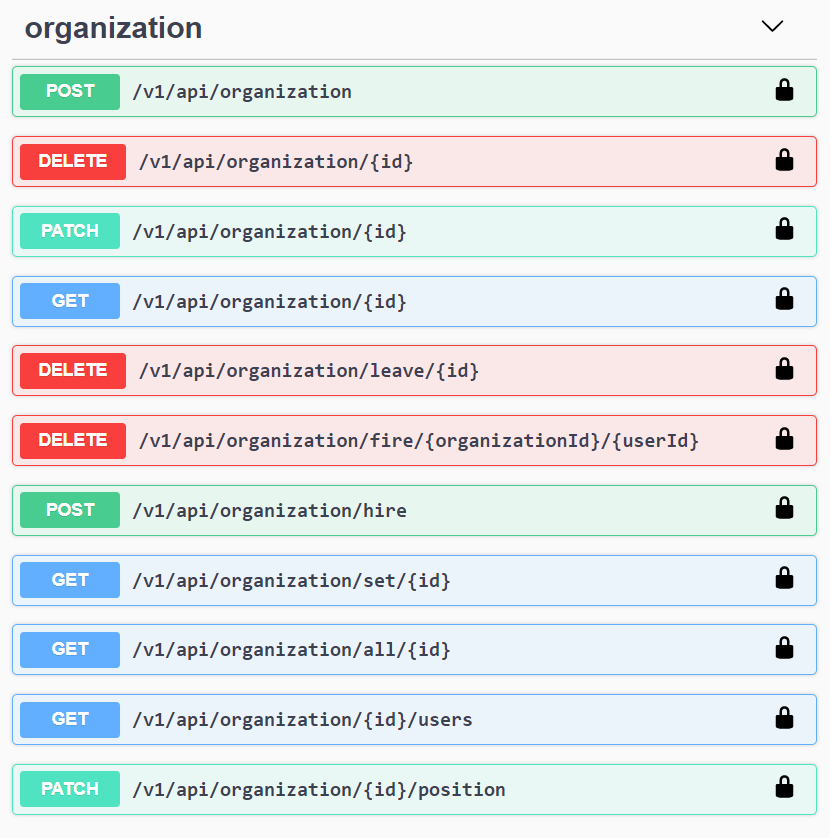


Рисунок 12 – Rest API модуля OrganizationModule.

# **2.3.4 Модуль для работы с сущностью сообщения из чата**

Данный модуль предоставляет Rest API и функции для работы с сообщениями. Также он активно использует сервис из модуля SocketModule. В модуле есть следующие функции:

* создать сообщение в чате. Если сообщение было создано в БД, то оно посылается по WebSocket всем подключенным пользователям, состоящим в этом чате;
* получение всех сообщений из чата;
* удаление сообщения из чата по id. Чтобы выполнить удаление, нужно быть автором этого сообщения. При удалении сообщения вызывается событие в WebSocket, которое призывает удалить это сообщение у всех подключенных пользователей, состоящих в этом чате;
* изменение сообщения по id. Чтобы выполнить изменение нужно быть автором этого сообщения. При удалении сообщения вызывается событие в WebSocket, которое призывает изменить это сообщение у всех подключенных пользователей, состоящих в этом чате.

Rest API модуля представлены на рисунке 13.

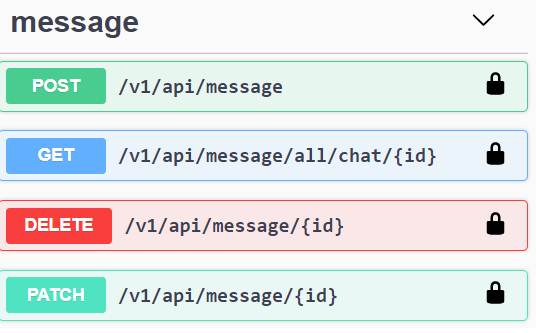


Рисунок 13 – Rest API модуля MessageModule.

# **2.3.5 Модуль для работы с сущностью чата**

Данный модуль предоставляет Rest API и функции для работы с чатами. Также он активно использует сервис из модуля SocketModule. В модуле есть следующие функции:

* создать чат;
* покинуть чат по его id. Если чат покидают все пользователи, то все сообщения и сам чат удаляются из БД;
* добавить пользователя в чат по его id и id группы. При добавлении пользователя в чат подключенного по WebSocket пользователя, ему придет событие о добавлении его в чат;
* получить все чаты текущего пользователя.

Rest API модуля представлены на рисунке 14.

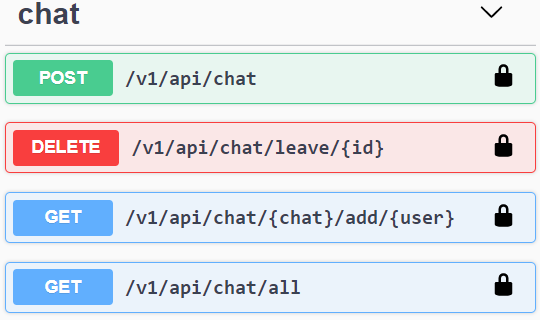


Рисунок 14 – Rest API модуля ChatModule.

# **2.3.6 Модуль для работы с сущностью группы**

Данный модуль предоставляет Rest API и функции для работы с группами. В модуле есть следующие функции:

* создать группу в текущей организации;
* получить все группы текущей организации по ее id;
* удалить группу по id. Для этого нужно быть ее администратором;
* изменить аватар, название или описание группы по id. Для этого нужно быть администратором группы;
* добавить пользователя в закрытую группу. Для этого нужно быть администратором группы, а добавляемый пользователь состоять в текущей организации;
* Удалить пользователя из закрытой группы. Для этого нужно быть администратором группы.

Rest API модуля представлены на рисунке 15.

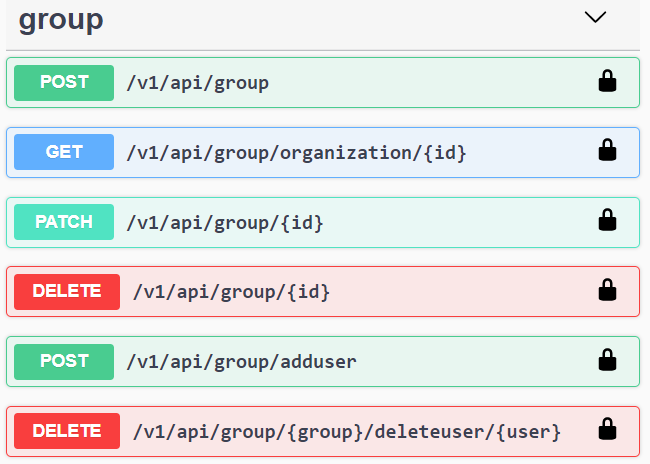


Рисунок 15 – Rest API модуля GroupModule.

# **2.3.7 Модуль для работы с сущностью информационного поста в группе**

Данный модуль предоставляет Rest API и функции для работы с постами в группах. В модуле есть следующие функции:

* создание поста в текущей организации;
* удаление поста из группы по id. Для этого нужно быть ее администратором;
* получение всех постов группы.

Rest API модуля представлены на рисунке 16.

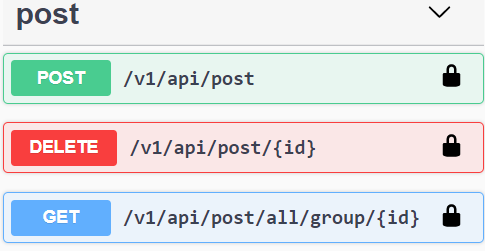


Рисунок 16 – Rest API модуля PostModule.

# **2.3.8 Модуль для работы с сущностью команды**

Данный модуль предоставляет Rest API и функции для работы с командами. В модуле есть следующие функции:

* создание команды в текущей организации;
* получение массива команд организации по ее id;
* изменение названия или аватара команды по ее id. Для этого нужно быть ее администратором;
* удаление команды по id. Для этого нужно быть ее администратором;
* добавление сотрудника организации в команду;
* удаление сотрудника из команды по их id;
* получение статусов, которые есть на доске kanban.

Rest API модуля представлены на рисунке 17.

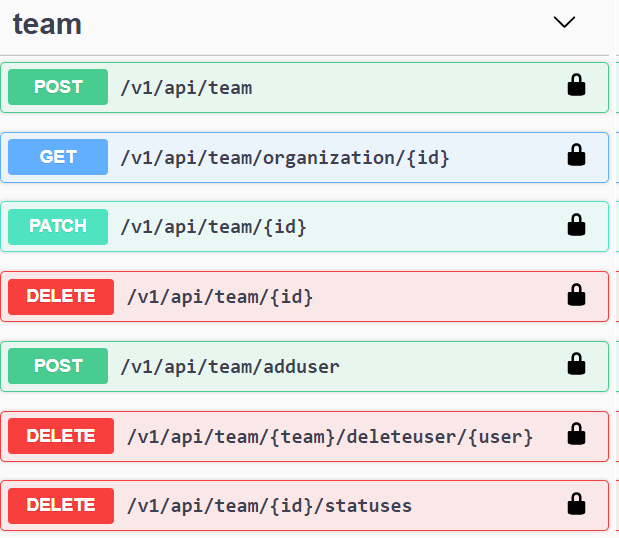


Рисунок 17 – Rest API модуля TeamModule.

# **2.3.9 Модуль для работы с сущностью задачи на kanban-доске**

Данный модуль предоставляет Rest API и функции для работы с задачами. Также он активно использует сервис из модуля SocketModule. В модуле есть следующие функции:

* создание задачи на kanban-доске. После создания, задача передается подключенным сотрудникам по WebSocket;
* получение массива задач команды по ее id;
* перенос задачи на другую стадию в kanban-доске. После переноса, новый статус задачи передается подключенным сотрудникам по WebSocket;
* редактирование названия и описания задачи на kanban-доске. После изменения в БД, задача передается подключенным сотрудникам по WebSocket;
* получение всей информации задачи по ее id;
* оставление ответ на задачу.

Rest API модуля представлены на рисунке 18.

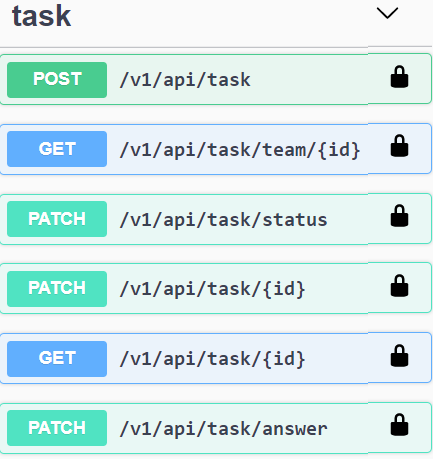


Рисунок 18 – Rest API модуля TaskModule.

# **2.4 Выводы по главе**

В соответствии с заданием, предложена архитектура системы и структура базы данных. Разработаны и задокументированы Rest API. Реализована клиентская часть веб-приложения.

**3 Описание разработанного приложения**

# **3.1.1 Страницы аутентификации и регистрации**

При заходе на сайт у пользователя будет выбор, либо зайти на сайт по логину и паролю, либо перейти на страницу регистрации. На странице регистрации пользователь должен указать некоторые данные о себе и по желании загрузить аватар. Формы со страницы регистрации и аутентификации представлены на рисунке 19.

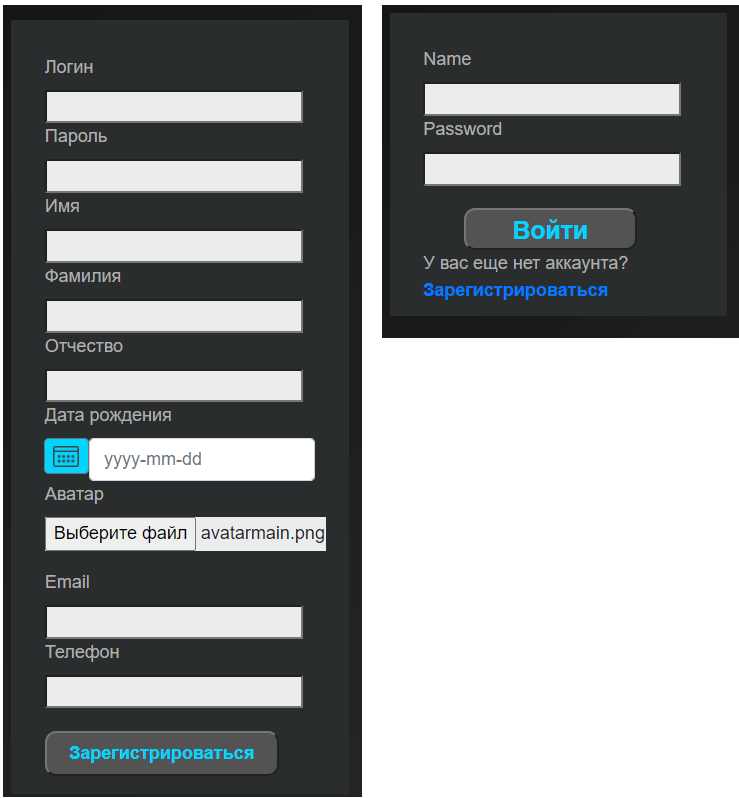


Рисунок 19 – Формы со страниц аутентификации и регистрации

В случае ввода неверных данных в форме регистрации отобразится ошибка с информацией о некорректных данных. Если пользователь ввел верные данные, то он будет перенаправлен на страницу аутентификации. На странице аутентификации пользователь должен ввести логин и пароль, которые были указаны им ранее в форме регистрации, после чего он будет направлен на личную страницу сотрудника.

# **3.1.2 Личная страница сотрудника**

Личная страница сотрудника представляет собой блоки навигации и информации о сотруднике, помимо этого на странице присутствует шапка с логотипом сайта, именем и фамилией текущего пользователя, а также название текущей организации. Шапка будет присутствовать на всех следующих страницах, как и блок навигации. При нажатии по ссылке пользователь будет перенаправлен на соответствующую страницу.

В зависимости от того, является ли пользователь обладателем личной страницы, меняется её функционал, а именно вместо кнопки редактирования будет дана кнопка для создания чата с обладателем страницы. При нажатии на кнопку редактирования, откроется модальное окно с содержанием, которое похоже на форму регистрации. В блоке личной информации можно узнать – онлайн ли сейчас данный сотрудник. Личная страница сотрудника представлена на рисунке 20.

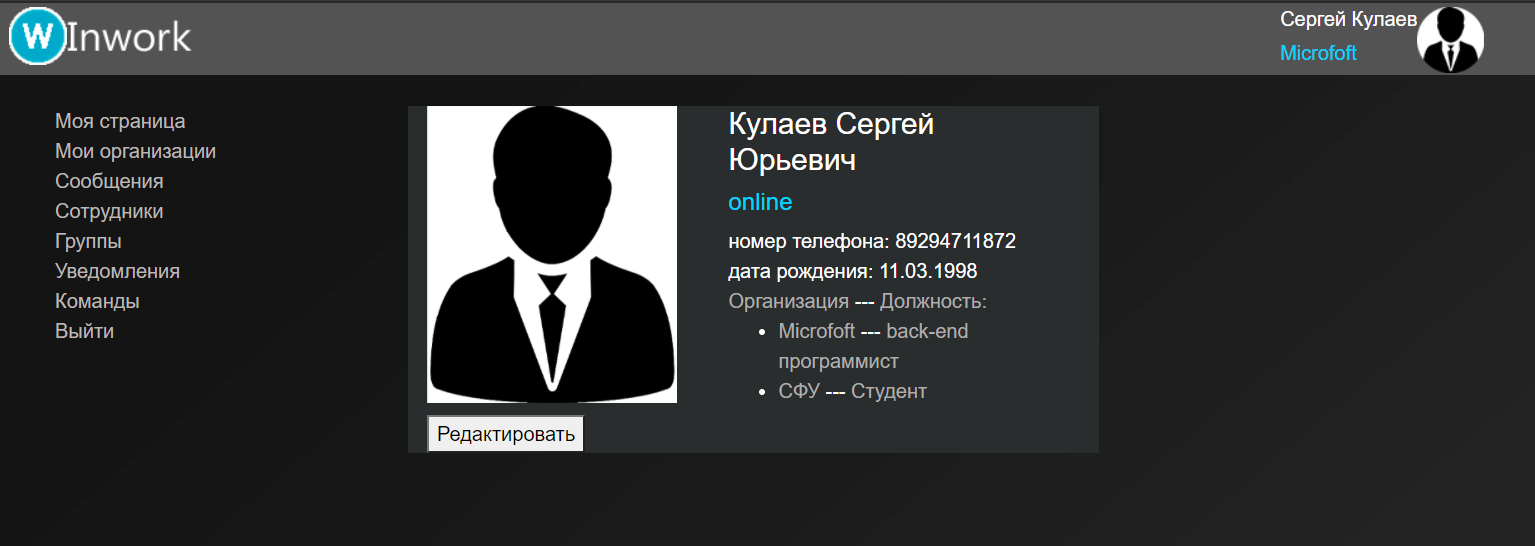


Рисунок 20 – Личная страница пользователя

# **3.1.3 Страница со списком организаций**

На странице со списком организаций отображаются все организации в которых состоит пользователь. Он может выбрать организацию, с которой в данный момент хочет работать нажав на кнопку «Выбрать» у соответствующей организации. В списке организаций можно увидеть ее название, описание, должность и статус пользователя в организации. На странице также есть возможность добавить новую организацию. Страница со списком организаций представлена на рисунке 21.

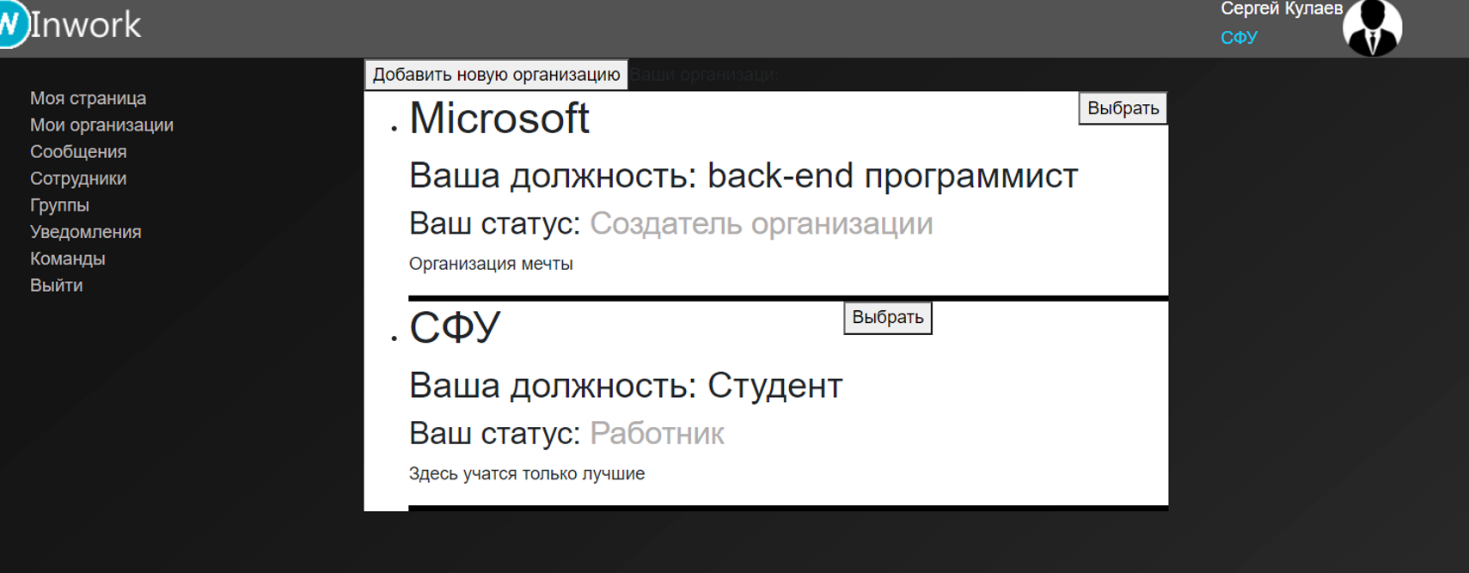


Рисунок 21 – Список организаций

# **3.1.4 Страница с списком чатов**

На рисунке 22 представлена страница со списком чатов. Здесь можно увидеть название чата, последнее сообщение в этом чате, отправителя сообщения и время отправки. Помимо этого, на странице есть кнопка создания чата, в котором нужно указать название чата и, при желании, сразу же добавить сотрудников организации, список которых будет предложен в форме создания чата.

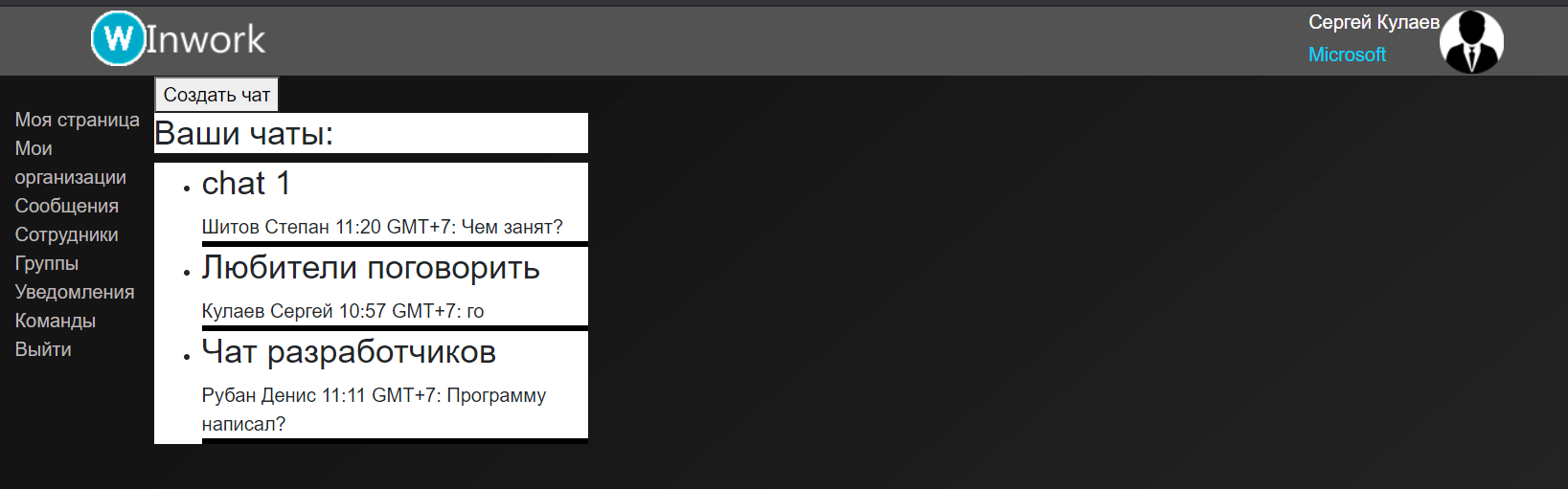


Рисунок 22 – Список чатов

# **3.1.5 Страница с чатом**

Страница чата служит для просмотра и отправки сообщий в чат. При заходе на страницу выбранного чата, в него сразу же будет загружена история сообщений. У каждого сообщения можно увидеть его отправителя, дату и время отправки. Для отправки сообщения необходимо ввести сообщеине в поле и нажать на кнопку отправить. Страница чата предсталена на рисунке 23.

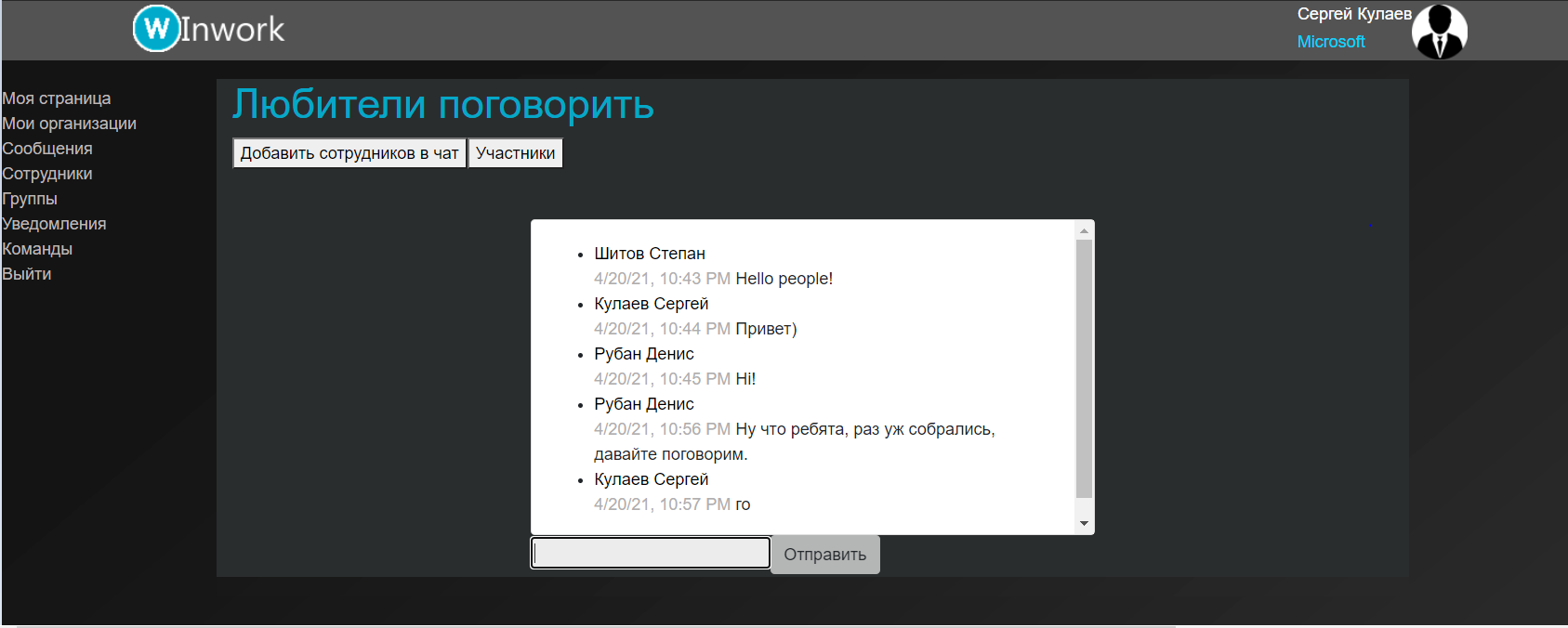


Рисунок 23 – Чат с сотрудниками

Также на странице присутствует возможность добавить сотрудников в чат и/или просмотреть список сотрудников, которые уже состоят в чате.

# **3.1.6 Страница со списком сотрудников**

На данной странице представлены сотрудники текущей организации. В списке можно увидеть следующую информацию о сотрудниках:

* имя и фамилию;
* должность в организации;
* статус в организации;
* информацию о том, находится ли сотрудник сейчас онлайн.

При двойном нажатии по сотруднику откроется его личная страница.

Также у каждого элемента списка есть кнопка позволяющая назначить новую должность.

Для добавления нового сотрудника в организацию необходимо ввести его логин и назначаемую ему должность. Если такой пользователь существует, то он будет добавлен в организацию. Страница сотрудников представлена на рисунке 24.

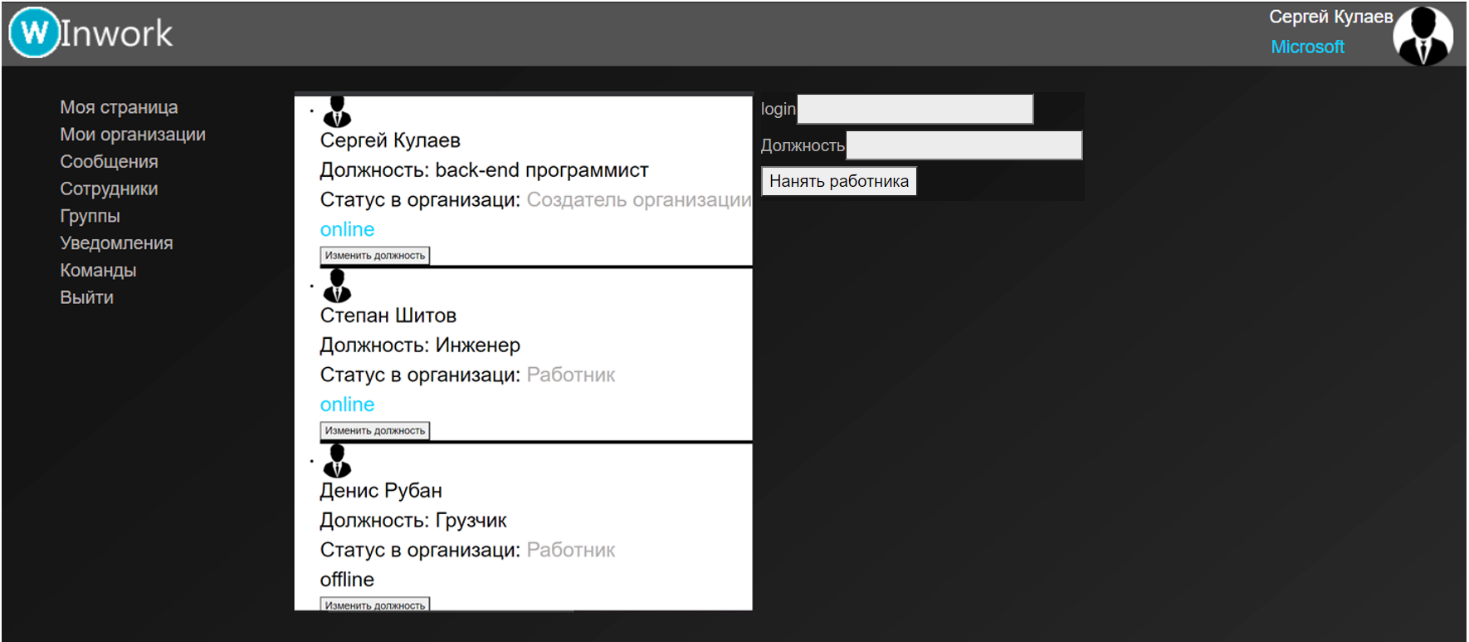


Рисунок 24 – Список сотрудников

# **3.1.4 Страница команды и kanban-доски**

При переходе по ссылке «Команды» откроется страница со списком команд, в которых состоит пользователь. Помимо списка там присутствует кнопка, которая открывает модальное окно, в котором можно указать название команды и выбрать добавляемых сотрудников.

Если нажать дважды по команде из представленного списка, откроется страница команды и kanban-доски. На этой странице можно создавать задачи. Для создания задачи необходимо нажать на нужный статус правой кнопкой мыши, после чего в открывшемся контекстном меню выбрать пункт «Создать задачу». В открывшемся модальном окне можно указать название задачи, описание, имеет ли задание крайний срок, если имеет – указать и его. Каждую задачу можно перемещать по этапам, выбирая нужное действие в контекстном меню, которое открывается нажатием правой кнопки мыши по задаче. Задачу можно также принять и пометить как завершенную, после этого она будет перемещена в историю задач. В истории задач можно просмотреть подробную информацию по каждой завершенной задаче.

На странице существует возможность просмотреть состав команды, нажав на кнопку «Состав команды», после этого откроется страница со списком участников и кнопкой «Добавить участников». Добавить участников можно только из сотрудников организации, к которой принадлежит команда.

На задачи, которые находятся на kanban-доске, можно оставлять ответ только тем сотрудникам, которые были прикреплены к ним. Страница команды и kanban-доски представлена на рисунке 25.



Рисунок 25 – Kanban-доска

Для редактирования или ответа на задачу необходимо дважды нажать на нее, в этом случае откроется модальное окно, в котором можно увидеть подробную информацию о задаче. В этом окне администратор команды может назначить, удалить исполнителей или отредактировать задачу. Модальное окно редактирования задачи представлено на рисунке 26.

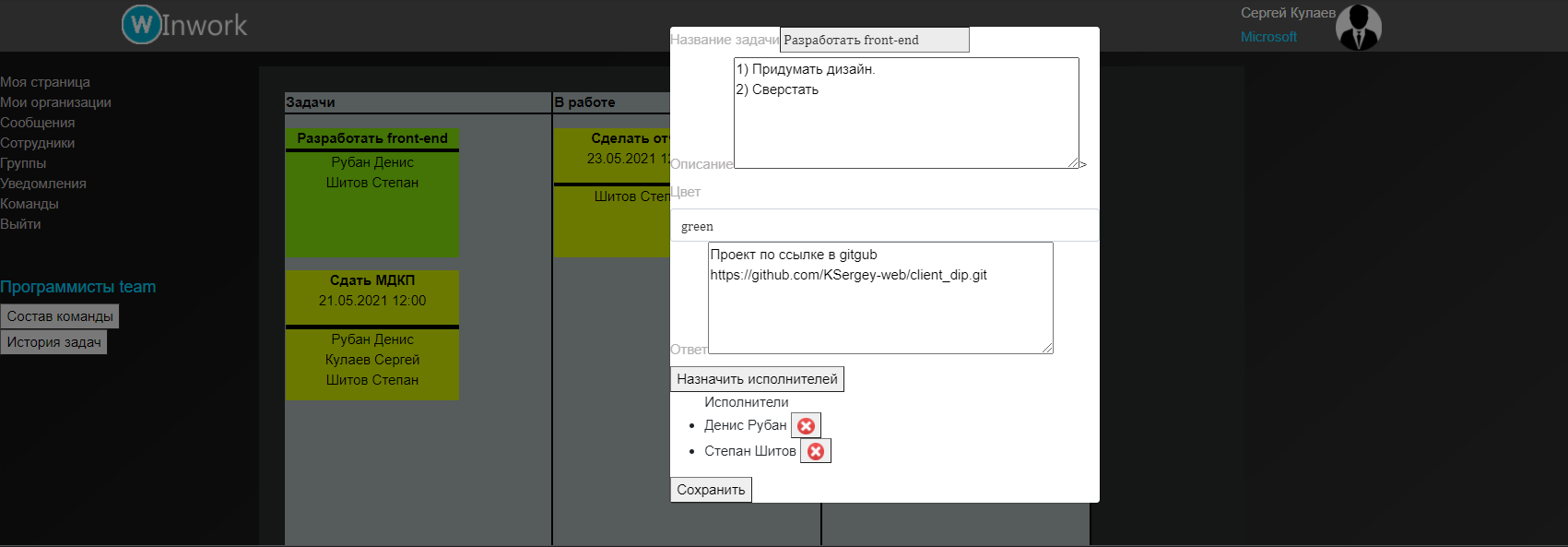


Рисунок 26 – Модальное окно редактирования задачи

# **3.2 Инструкция разработчика**

Для работы, у вас предварительно должны быть установлены: пакетный менеджер npm и node.js.

Создайте папку в которую поместите веб-приложение и перейдите в нее.

mkdir inwork

cd inwork

**Сервер**

Сервер выполнен на платформе Nestjs и для его использования сначала необходимо глобально установить следующий пакет.

npm i -g @nestjs/cli

Для работы с базой данных на сервере требуются docker и docker-compose.

В качестве пакетного менеджера используется yarn.

Клонирование проекта с сервером

git clone https://github.com/KSergey-web/Diploma.git server

cd server

Установить все зависимости

yarn install

Теперь можно начинать работать.

Запустить сервер в dev режиме.

yarn start:dev

В файле .env.development находятся переменные окружения.

Для подключения сервера к базе данных Mongodb соберите мультиконтейнер.

docker-compose up -d

По умолчанию сервер базы данных доступен на порте 27017.

**Документирование**

Для просмотра документации по REST API в SWAGGER запустите сервер

yarn start:dev

Откройте браузер и перейдите по адресу localhost:4000/api. Из этой страницы вы можете делать запросы на сервер.

**Клиент**

Клиент выполнен с использованием Angular для его запуска необходим следующий инструмент.

npm i -g @angular/cli

Клонирование проекта с клиентом

git clone <https://github.com/KSergey-web/client_dip.git> client

cd client

Установка всех зависимостей проекта

npm install

Теперь можно начать работу с клиентом.

Для запуска клиента введите команду.

ng serve

После того как произойдет сборка клиента можно зайти на localhost:4200 и увидеть сайт.

Все routes расположены в файле app-routing.module.ts.

**Развертывание клиента на сервере**

Для развертывания клиента на сервере нужно перейти в директорию с клиентом и выполнить следующую команду.

ng build --prod

SPA-приложение скомпилируется в директорию ./dist. Далее необходимо скопировать полученные файлы на сервер.

xcopy /e ./dist/ ../server/static/

# Теперь при запросе к серверу можно попасть на разработанный сайт.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения выпускной квалификационной работы была изучена предметная область и существующие на данный момент аналоги. После изучения аналогов был сформулирован ряд требований, предъявляемый к разрабатываемому веб-приложению корпоративной социальной сети и выбраны оптимальные средства разработки.

На основе сформированных требований было разработано и протестировано веб-приложение для организаций, c возможностями взаимодействия между сотрудниками и управления рабочим процессом.

Итогом всей разработки является веб-приложение, которое готово к развертыванию**.**

# **СПИСОК СОКРАЩЕНИЙ**

БД – База данных

ВКР – выпускная квалификационная работа

JSON – JavaScript Object Notation

CSS – Cascading Style Sheets

MEAN – стек MongoDB, Express, Angular, NestJS или Node.js

**Список использованных источников**

1. SBIS [Электронный ресурс]: – Режим доступа: https://sbis.ru/help/com
2. Workplace [Электронный ресурс]: – Режим доступа: https://www.workplace.com/
3. Битрикс24 [Электронный ресурс]: – Режим доступа: <https://www.bitrix24.ru/>
4. TypeScript [Электронный ресурс]: – <http://web.spt42.ru/index.php/chto-takoe-typescript>
5. Node.js [Электронный ресурс]: – <https://ru.wikipedia.org/wiki/Node.js>
6. NestJS [Электронный ресурс]: – <https://nestjs.com/>
7. MongoDB [Электронный ресурс]: – <https://ru.wikipedia.org/wiki/MongoDB>
8. What is an ORM [Электронный ресурс]: – <https://www.sololearn.com/learning/1092/3355/7557/1>
9. Angular [Электронный ресурс]: – <https://angular.io/>
10. WebSocket <https://ru.wikipedia.org/wiki/WebSocket>
11. Socket.io [Электронный ресурс]: – https://socket.io/

# **ПРИЛОЖЕНИЕ А**

**Исходный код файла app.module.ts клиента**

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { HttpClientModule, HTTP\_INTERCEPTORS } from '@angular/common/http';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { SigninComponent } from './signin/signin.component';

import { NotFoundComponent } from './not-found/not-found.component';

import { ReactiveFormsModule } from '@angular/forms';

import {JwtModule} from '@auth0/angular-jwt';

import { environment } from 'src/environments/environment';

import { ACCESS\_TOKEN\_KEY } from './signin/services/singin.service';

import { MyPageComponent } from './mypage/mypage.component';

import { AUTH\_API\_URL, INWORK\_API } from './app-injection-tokens';

import { httpInterceptorProviders } from './interceptors';

import { SignUpComponent } from './sign-up/sign-up.component';

import { FormsModule } from '@angular/forms';

import { HeaderComponent } from './header/header.component';

import { InfoAboutUserComponent } from './mypage/components/info-about-user/info-about-user.component';

import { NavigationComponent } from './navigation/navigation.component';

import { MyOrganizationsComponent } from './my-organizations/my-organizations.component';

import { ArrayOrganizationsComponent } from './my-organizations/components/array-organizations/array-organizations.component';

import { OrganizationFormComponent } from './my-organizations/components/organization-form/organization-form.component';

import { EmployeesComponent } from './employees/employees.component';

import { MessagesComponent } from './messages/messages.component';

import { ChatsComponent } from './chats/chats.component';

import { ListWorkersComponent } from './shared/list-workers/list-workers.component';

import { ChatFormComponent } from './chats/chat-form/chat-form.component';

import { AddUsersComponent } from './messages/add-users/add-users.component';

import { UsersInChatComponent } from './messages/users-in-chat/users-in-chat.component';

import { TeamsComponent } from './teams/teams.component';

import { TeamFormComponent } from './teams/components/team-form/team-form.component';

import { TeamComponent } from './team/team.component';

import { KanbanComponent } from './team/components/kanban/kanban.component';

import { ContextMenuComponent } from './team/components/kanban/context-menu/context-menu.component';

import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

import { TaskFormComponent } from './team/components/kanban/task-form/task-form.component';

import { TeamStructureComponent } from './team/components/team-structure/team-structure.component';

import { AddUserFormComponent } from './team/components/team-structure/add-user-form/add-user-form.component';

import { UpdateTaskFormComponent } from './team/components/kanban/update-task-form/update-task-form.component';

import { HistoryTasksComponent } from './team/components/history-tasks/history-tasks.component';

import { NewPositionComponent } from './employees/new-position/new-position.component';

export function tokenGetter(){

  return localStorage.getItem(ACCESS\_TOKEN\_KEY);

}

@NgModule({

  declarations: [

    AppComponent,

    SigninComponent,

    NotFoundComponent,

    MyPageComponent,

    SignUpComponent,

    HeaderComponent,

    InfoAboutUserComponent,

    NavigationComponent,

    MyOrganizationsComponent,

    ArrayOrganizationsComponent,

    OrganizationFormComponent,

    EmployeesComponent,

    MessagesComponent,

    ChatsComponent,

    ListWorkersComponent,

    ChatFormComponent,

    AddUsersComponent,

    UsersInChatComponent,

    TeamsComponent,

    TeamFormComponent,

    TeamComponent,

    KanbanComponent,

    ContextMenuComponent,

    TaskFormComponent,

    TeamStructureComponent,

    AddUserFormComponent,

    UpdateTaskFormComponent,

    HistoryTasksComponent,

    NewPositionComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    HttpClientModule,

    ReactiveFormsModule,

    FormsModule,

    JwtModule.forRoot({config:{

      tokenGetter,

      allowedDomains: environment.tokenWhiteListedDomains

    }}),

    NgbModule,

  ],

  providers: [{

    provide: AUTH\_API\_URL,

    useValue: environment.inWorkApi

  },

  {

    provide: INWORK\_API,

    useValue: environment.inWorkApi

  },

  httpInterceptorProviders

],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Исходный код файла app.module.ts клиента**

import { Module } from '@nestjs/common';

import { AppController } from './app.controller';

import { AppService } from './app.service';

import { configModule } from './configure.root';

import { MongooseModule } from '@nestjs/mongoose';

import { UserController } from './user/user.controller';

import { UserModule } from './user/user.module';

import { AuthController } from './auth/auth.controller';

import { AuthModule } from './auth/auth.module';

import { SharedModule } from './shared/shared.module';

import { OrganizationModule } from './organization/organization.module';

import { SocketModule } from './socket/socket.module';

import { MessageModule } from './message/message.module';

import { ChatModule } from './chat/chat.module';

import { ServeStaticModule } from '@nestjs/serve-static';

import { join } from 'path';

import { GroupModule } from './group/group.module';

import { PostModule } from './post/post.module';

import { TeamModule } from './team/team.module';

import { TaskModule } from './task/task.module';

@Module({

imports: [

configModule,

MongooseModule.forRoot(process.env.MONGODB\_WRITE\_CONNECTION\_STRING),

UserModule,

AuthModule,

SharedModule,

OrganizationModule,

SocketModule,

ChatModule,

MessageModule,

ServeStaticModule.forRoot({

rootPath: join(\_\_dirname, '..', 'static'),

}),

GroupModule,

PostModule,

TeamModule,

TaskModule,

],

controllers: [AppController],

providers: [AppService],

})

export class AppModule {}