МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра *Систем сбора и обработки данных*
(полное название кафедры)

Утверждаю

Зав. кафедрой *М.А. Бакаев*

(подпись, инициалы, фамилия)

«___» _____ 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Меркулова Никиты Максимовича
(фамилия, имя, отчество студента – автора работы)

Разработка системы удаленного консультирования в сфере онкологических заболеваний
(тема работы)

Факультет автоматики и вычислительной техники
(полное название факультета)

Направление подготовки *12.03.04 Биотехнические системы и технологии*
(код и наименование направления подготовки бакалавра)

Руководитель от НГТУ

Квашнина Елена Анатольевна
(фамилия, имя, отчество)

Ст. преподаватель
(ученая степень, ученое звание)

(подпись, дата)

Автор выпускной квалификационной работы

Меркулов Никита Максимович
(фамилия, имя, отчество)

АВТФ, АО-71
(факультет, группа)

(подпись, дата)

Новосибирск 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра *Систем сбора и обработки данных*
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой *М.А. Бакаев*
(фамилия, имя, отчество)

(подпись, дата)

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

студенту *Меркулову Никите Максимовичу*
(фамилия, имя, отчество)

Направление подготовки *12.03.04 Биотехнические системы и технологии*
(код и наименование направления подготовки бакалавра)

текст

Факультет автоматики и вычислительной техники
(полное название факультета)

Тема *Разработка системы удаленного консультирования в сфере онкологических заболеваний*
(полное название темы выпускной квалификационной работы бакалавра)

Исходные данные (или цель работы) *Целью работы является разработка прототипа веб-приложения для отправки сообщений и файлов между врачами-онкологами*

Структурные части работы *Введение*
Глава 1. Обзор существующих аналогичных систем
Глава 2. Анализ инструментов для реализации решения
Глава 3. Выбор конкретных средств
Глава 4. Проектирование системы
Глава 5. Разработка серверной части веб-приложения
Глава 6. Разработка клиентской части веб-приложения

Заключение

Список использованных источников

Приложение

Задание согласовано и принято к исполнению.

Руководитель от НГТУ	**Студент**
Квашнина Елена Анатольевна	*Меркулов Никита Максимович*
(фамилия, имя, отчество)	(фамилия, имя, отчество)
Ст. преподаватель	*АВТФ, АО-71*
(ученая степень, ученое звание)	(факультет, группа)
(подпись, дата)	(подпись, дата)

Тема утверждена приказом по НГТУ № *451/2* от «*11*» *февраля* 2021 г.

ВКР сдана в ГЭК № _____, тема сверена с данными приказа

(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

(фамилия, имя, отчество секретаря государственной экзаменационной комиссии по защите ВКР)

Реферат

ВКР 89 с., 41 рис., 44 источника.

Ключевые слова: веб-приложение, JavaScript, GraphQL, Express.js, React, база данных.

Объектом исследования является разработка приложений с функцией дистанционного взаимодействия медицинских специалистов. Предметом исследования является разработка клиент-серверного веб-приложения на языке программирования JavaScript для дистанционного взаимодействия медицинского персонала онкологических отделений.

Целью выпускной квалификационной работы является разработка прототипа веб-приложения для отправки сообщений и файлов между врачами-онкологам.

Первая глава посвящена обзору существующих аналогичных систем, приводится таблица и краткий вывод.

Во второй главе произведен анализ инструментов и технологий, используемых при разработке веб-приложений.

В третьей главе приводятся программные пакеты, используемые при разработке прототипа веб-приложения.

Четвертая глава содержит подробное описание спроектированной базы данных и схемы объектов для организации прикладного программного интерфейса.

Пятая и шестая главы посвящены разработке серверной и клиентской частей веб-приложения соответственно. Приводится описание элементов интерфейса.

В конечном итоге был реализован прототип веб-приложения, предназначенный для использования медицинскими работниками – онкологами. Направления развития приложения включают: добавление средств видеосвязи, развитие API для интеграции с региональным порталом записи.

Оглавление

Введение

В систематическом обзоре применения мессенджеров в клинической медицине [2] отмечается высокий интерес коллективов медицинских работников, команд врачей к мессенджерам «WhatsApp» и «Viber», как к средствам организации эффективного взаимодействия и, как к инструментам обучения, контроля, поддержки между специалистами разных уровней квалификации. Также отмечается проблема интеграции указанных выше приложений с медицинскими информационными системами (МИС) для передачи данных телеконсультаций или их печати.

На активное использование мессенджеров в различных сферах медицины влияет качество интернет соединения между пользователями и ЦОД (Центры Обработки Данных), на которых размещается серверная часть названных ранее приложений для коммуникации. Далеко не каждый населенный пункт и располагаемые в нем медицинские учреждения России обеспечены высокоскоростными точками доступа в сеть Интернет. Также существует вероятность отключения российского сегмента от мирового Интернета, притом по некоторым оценкам — высокая [1][5].

Разрабатываемое веб-приложение предлагается как альтернативное и доступное решение при недоступности популярных мессенджеров, т.к.:

- построено с использованием современных веб-технологий (в том числе языка производительных запросов между сервером и клиентом GraphQL) с открытым исходным кодом, что позволит эффективно интегрировать его в существующие информационные сети учреждений и наименее затратно поддерживать на протяжении многих лет [25];
- размещается в локальных сетях или в ЦОД на территории РФ, что решает проблему недоступности мессенджеров или ограниченного доступа к ним.

Одним из приоритетных направлений развития в рамках национального проекта «Здравоохранение» является федеральный проект «Борьба с онкологическими заболеваниями». Достижение заявленных в рамках

федерального проекта «Борьба с онкологическими заболеваниями» показателей невозможно без приведения функциональных возможностей онкологической службы в соответствие современным требованиям, а также решению существующих проблем методами информационных технологий. Важнейшим обеспечивающим процессом при оказании медицинской помощи по профилю «онкология» является ИТ-поддержка соответствующих медицинских процессов, которая в современных условиях оказывает существенное влияние на их эффективность. Поэтому одним из аспектов достижения заявленных в рамках федерального проекта показателей, является внедрение централизованной Подсистемы «Онкология» [6]. Одним из требований Подсистемы «Онкология» является наличие функций дистанционного взаимодействия в режимах врач-врач и врач-пациент, а также обеспечение преемственности оказания медицинской помощи между амбулаторным и стационарным этапами лечения.

В соответствии с Методическими рекомендациями Минздрава России к построению Единых государственных информационных систем в сфере здравоохранения в субъектах Российской Федерации предъявляется следующее требование: [7]: «В части вопросов об организации уведомлений, рекомендуется обеспечивать функциональные возможности для использования любых способов оперативного оповещения, которые обеспечат соответствующее эффективное информирование участников процесса при организации оказания медицинской помощи с применением телемедицинских технологий с использованием Системы ТМК, например, SMS, e-mail, иные способы. При этом необходимо обеспечить возможность фиксировать информацию об их получении и прочтении адресатом с фиксацией даты и времени указанных действий».

Цель работы: разработка прототипа веб-приложения для отправки сообщений и файлов между врачами-онкологами.

Для выполнения цели работы были поставлены следующие задачи:

1. Проанализировать существующие решения в данной области.
2. Выбрать методы и средства реализации и обосновать выбор.
3. Спроектировать и реализовать прототип веб-приложения.

4. Оценить результаты работы и перспективы дальнейшей доработки.

Глава 1. ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГИЧНЫХ СИСТЕМ

Сравнение характеристик аналогичных систем представлено в Таблице 1.

1.1 TeamSpirit IM

Корпоративный мобильный мессенджер с видеоконференциями, совместимый с сетевой инфраструктурой предприятия, размещаемый на серверах организации. Для внедрения мессенджера требуется оставить заявку на сайте teamspirit.im, предоставить банковские данные, т.к. после трёх месяцев использования заканчивается пробный период пользования программным кодом.

Достоинства:

- Возможность развертывания в локальной сети;
- Наличие встроенных средств видеосвязи и видеоконференций;
- Разработка российской компании.

Недостатки:

- Только конкретные браузеры, используемые в качестве клиентской стороны мессенджера, могут использоваться без установки дополнительного программного обеспечения [8];
- Исходный код модулей видеосвязи и видеоконференций не передается клиентам;
- Исходный код модулей отправки сообщений и обмена файлами передается клиентам в аренду и требует периодической оплаты. Он не находится в открытом доступе [9].

1.2 WhatsApp

Разработанное как альтернатива SMS и изначально приложение для отправки сообщений по Интернету. На данный момент используется для передачи файлов, видеозвонков, аудиозвонков. Общение организуется либо в

виде диалога, либо в качестве чата, аудио- или видеозвонка группы пользователей.

Достоинства:

- Возможность проведения видеозвонков;
- Возможность отправки файлов;
- Наличие приложений для разных систем;
- Основные функции доступны в веб-приложении, не требующем установки на устройства. Веб-приложение адаптировано для персональных компьютеров и мобильных устройств.

Недостатки:

- Использование собственных ресурсов для размещения программного обеспечения не предусмотрено;
- Не предусмотрена возможность интеграции приложения с МИС для передачи данных консультаций.

1.3 Slack

Главной особенностью Slack является деление на каналы. Это позволяет эффективно организовывать взаимодействие пользователей и обмен файлами в зависимости от назначения каждого канала.

Достоинства:

- Широкие возможности для интеграции с различными облачными приложениями, например, Google Drive, Microsoft Office 365;
- Основные функции доступны в веб-приложении, не требующем установки на устройства. Веб-приложение адаптировано для персональных компьютеров и мобильных устройств.

Недостатки:

- Использование собственных ресурсов для размещения программного обеспечения не предусмотрено;

- Не предусмотрена возможность интеграции приложения с МИС для передачи данных консультаций.

Таблица 1 — Сравнение характеристик аналогичных систем.

	Открытое программное обеспечение	Размещение в собственных информационных сетях	Возможность совершения видеозвонков
TeamSpirit IM	Исходный код предоставляется клиентам частично за плату. Ограничено его распространение.	Да	Да
WhatsApp	Нет	Нет	Да
Slack	Нет	Нет	Да

Таким образом, анализ существующих решений показывает, что присутствующие на рынке системы распространяются без открытого исходного кода и возможности размещения в локальных сетях. Одна из систем (TeamSpirit IM) имеет возможность размещения в локальных сетях, и компания-разработчик предоставляет исходный код программы в распоряжение клиентам, но не содержит его в открытом доступе.

Глава 2. АНАЛИЗ ИНСТРУМЕНТОВ ДЛЯ РЕАЛИЗАЦИИ РЕШЕНИЯ

2.1 Анализ стилей API

Веб-приложения — это интерактивные компьютерные приложения, доступные для использования посредством сети интернет, позволяющие пользователям передавать, получать и манипулировать данными. Такие программы обычно имеют тесную связь с удалённым сервером, обмениваясь с ним множеством запросов [4].

Большинство веб-приложений спроектированы по архитектуре клиент-сервер. При такой реализации некоторые задания или сетевая нагрузка распределяются между поставщиками услуг, именуемыми серверами, и заказчиками услуг, именуемыми клиентами. В сущности, клиент и сервер являются реализациями программного обеспечения [3]. Любые программные продукты обмениваются данными и функциональными возможности через машиночитаемые интерфейсы — API (программные интерфейсы приложений). Для веб-приложений существует отдельная категория — веб-API, которые в основном доставляют запросы от веб-приложений и ответы от серверов с использованием протокола прикладного уровня передачи данных HTTP (протокола передачи гипертекста) [27].

Перечислим ниже основные спецификации или протоколы для реализации API.

- Удалённый вызов процедур (RPC) — класс технологий, позволяющих программам вызывать функции или процедуры в другом адресном пространств Один из двух способов реализовать удаленный вызов процедуры — это SOAP.

- Простой протокол доступа к объектам (SOAP). API-интерфейсы, соответствующие принципам SOAP, позволяют обмениваться сообщениями XML между системами через HTTP или простой

протокол передачи почты (SMTP). SOAP в основном используется в корпоративном сегменте для обеспечения высокой безопасности передаваемых данных [10].

- REST представляет собой набор архитектурных правил и ограничений, учитываемых при проектировании распределённой гипермедиа-системы. REST-системы реализуются с применением стандартов: URL, HTTP, JSON. А также поддерживают обмен сообщениями в различных форматах, таких как: обычный текст, HTML, YAML, XML и JSON. SOAP же только допускает расширяемый язык разметки (XML). Реализации REST API в составе RESTful сервисов [28] характеризуются проблемой чрезмерных или недостаточных данных. Ответы REST могут содержать либо избыточное, либо недостаточное количество сведений. Образуется необходимость в отправке дополнительных запросов [11].

- GraphQL — это язык запросов для API, позволяющий клиенту определить выборку данных, которые ему нужны, и предоставляющий функции агрегатора данных из разных источников. Поэтому разработчик может использовать один вызов API для запроса всех необходимых данных [29]. Снижение числа запросов также позволяет использовать менее производительные сервера для размещения веб-приложений.

Так как самым распространенным устройством среди пользователей разрабатываемого веб-приложения является смартфон [43], то оптимальным выбором инструмента реализации схемы данных для API станет GraphQL. В работах исследователей [30][31] также отмечается, что при совершении запросов определенной выборки данных с GraphQL API объем передаваемых по сети данных был вплоть до 70% меньшим, чем при использовании REST.

2.2 Анализ веб-фреймворков

Веб-фреймворк — это программная база, используемая при написании веб-приложений. Он определяет структуру, задаёт правила и предоставляет необходимый набор инструментов для разработки клиент-серверного приложения.

Бэкенд-фреймворки — это фреймворки веб-разработки, написанные для работы на сервере. Бэкенд-фреймворки включают в себя инструментарий для решения многих задач: маршрутизации URL-адресов, взаимодействия с базами данных, авторизации пользователей и др.

Перечислим популярные бэкенд-фреймворки и их особенности, а также отличительные свойства языков программирования, на которых они написаны [12]:

- Django — Python. Django — кросс-платформенный проект с встроенными инструментами взаимодействия с различными базами данных. Инструментарий используется при моделировании базовых классов, позволяющих репрезентировать тем самым объектно-реляционное отображение или «виртуальные объектные базы данных» [13]. Применение Python позволяет писать емкий и понятный код, но экосистема пакетов значительно уступает в размерах той, что используется в JavaScript. Также отмечается проблема интеграции серверной части приложений, написанной на Python, и клиентской — на JavaScript [14].

- Laravel — PHP. Laravel предоставляет многие инструменты полезные на начальных стадиях разработки: Artisan — консольное приложение для миграции баз данных; встроенный сервер Homestead, включающий виртуальный образ с настроенной средой PHP-разработки; PHPUnit — фреймворк, предназначенный для тестирования. Основной недостаток PHP определяется в его характере взаимодействия с вводом-выводом, а именно в блокирующем режиме [15]. Эта особенность приводит к простаиванию ресурсов сервера при приостановлении потока выполнения.

- Express.js и требуемая для его работы среда исполнения Node.js — JavaScript. Express.js — это минималистичный бэкенд-фреймворк для приложений Node.js, встраиваемый в проектируемые системы и предоставляющий обширный набор служебных методов HTTP и промежуточных обработчиков. Сильной стороной всех JavaScript фреймворков является именно Node.js. Разработанные на данной платформе приложения показывают высокую производительность и отзывчивость системы ввода-вывода, благодаря событийно-ориентированной модели и неблокирующую ввод-вывод архитектуре [16]. Система распространения и менеджмента программных пакетов NPM дает доступ к многим инструментам и проектам, решающим широкий спектр задач, что ускоряет и удешевляет разработку. Одновременное использование JavaScript в серверной и клиентской частях, наличие особенного синтаксиса для простой работы с асинхронными функциями, значительно снижают временные затраты при проектировании систем.

Последний факт является определяющим в выборе языка программирования в разрабатываемом веб-приложении. Была выбрана связка «Node.js + Express.js», так как она является самой распространенной и хорошо задокументированной, не перегружена дополнительным функционалом.

Основное назначение фронтенд-фреймворков — визуальное отображение данных, предоставление интерфейса передачи информации между пользователем и программно-аппаратной частью веб-приложения. Фронтенд-фреймворки работают на стороне клиента, чаще всего в браузере. Они облегчают и ускоряют разработку пользовательских интерфейсов.

Перечислим популярные фронтенд-фреймворки. Отметим, что все они поддерживают язык JavaScript:

- Angular. Существует две версии Angular. Разработка первой версии в данный момент ограничена долгосрочной поддержкой, что означает прекращение активной разработки и выпуск обновлений только для решения проблем с безопасностью [26]. Активно разрабатываемая версия

Angular используется для реализации крупномасштабных проектов командами опытных разработчиков. Компоненты Angular могут быть добавлены к проектам, использующим иные технологии, например, React. Развитие Angular осуществляется при поддержке компании Google. Использование этого фреймворка требует знакомства с его структурными особенностями: Pipes, Components, Injectables и другими. В то же время React и Vue.js имеют только «Component».

- Vue.js. Vue.js используется при написании одностраничных приложений, либо для создания комплексных веб-интерфейсов приложений. Также имеется возможность добавления небольших интерактивные элементов в существующие инфраструктуры веб-приложений без нарушения целостности и работоспособности проектов. Vue.js — довольно молодой фреймворк, имеющий не столь высокий интерес со стороны сообщества и, следовательно, предоставляющий меньшее число учебных проектов для ознакомления.

- React — не является фреймворком, но библиотекой. Однако React повсеместно используется в веб-разработке и эффективно решает те же задачи, что и фронтенд-фреймворки. Библиотека React известна как эффективный инструмент для написания сложных, динамических пользовательских интерфейсов. Изменения в элементах интерфейса (перерисовка) реализуется модификацией объектной модели документа (DOM), притом в React используется виртуальная DOM, что повышает производительность и отзывчивость веб-приложений. Нужно отметить, что через менеджера пакетов NPM разработчику доступно множество вспомогательных библиотек, дополняющих функционал React. Для множества проблем в экосистеме React можно найти несколько вариантов решений. Остановимся в выборе на этом фронтенд-фреймворке для разрабатываемого веб-приложения, т.к. он наиболее многофункционален и испытан в проектах наибольшего числа разработчиков [17].

2.3 Анализ систем БД

Выбранный ранее язык запросов для API GraphQL получать данные практически из любых разновидностей баз данных.

2.3.1 MongoDB

MongoDB — документоориентированная система управления базами данных (СУБД), разработанная с акцентом на удобство разработки и масштабирование. Запись в этой базе данных представляет собой документ — структуру данных, состоящую из пар «поле-значение» [32].

Пользователи отмечают [18], что существует проблема в администрировании крупных баз Mongo, а именно необходимость использования наиболее времязатратного встроенного механизма создания резервных копий, по сравнению с MySQL и PostgreSQL.

2.3.2 MySQL

MySQL — реляционная СУБД, используемая в больших и малых проектах. На аппаратном уровне поддерживается использование нескольких потоков процессора, а также мультипроцессорных систем. Высокая производительность обеспечивается системой выделения памяти, распределенной по потокам.

2.3.3 PostgreSQL

PostgreSQL относится к объектно-реляционным базам данных. Такие системы имеют возможность добавления многих пользовательских объектов: операции, индексы, типы данных и других, что наделяет их гибкостью и надежностью.

MySQL и PostgreSQL обладают важным преимуществом перед MongoDB — они поддерживают технологию объектно-реляционного отображения (ORM), реализуемую с помощью библиотеки Sequilize. Эта ORM-библиотека,

распространяемая на платформе Node.js, сопоставляет таблицы из базы данных и их отношения с классами. С применением Sequelize не требуется использовать SQL-запросы, а можно манипулировать данными, как обычными объектами [19].

Выделяется преимущество PostgreSQL перед MySQL — максимальный размер строки в PostgreSQL ограничен 400 гигабайт [33], против 64 килобайт в MySQL [34]. Обе эти системы имеют равные показатели надежности и производительности. В разрабатываемом проекте была выбрана СУБД PostgreSQL так как, благодаря поддержке сложных структур и вариативности встроенных и пользовательских типов данных, она должна обеспечить больший набор возможностей в долгосрочной перспективе развития веб-приложения.

Таким образом, был проведен анализ языков программирования и существующих веб-фреймворков. Была выбрана программная экосистема JavaScript, инструмент для построения веб-API GraphQL, СУБД PostgreSQL.

Глава 3. ВЫБОР КОНКРЕТНЫХ СРЕДСТВ

3.1 NPM-пакеты для серверной части приложения

3.1.1 GraphQL.js

Пакет GraphQL.js содержит JavaScript версию GraphQL — языка запросов для API, созданный компанией Facebook. GraphQL.js предоставляет две важные возможности: построение схемы типов объектов и обслуживание запросов, в соответствии с построенной схемой. Язык построения схем — Schema Definition Language (SDL).

Схема определяет иерархию типов объектов с полями, которые должны быть возвращены в соответствии с запросом. Поле типа — единица данных, принадлежащая типу объекта схемы. Значения некоторых полей типа могут являться другими типами, тоже определенными в схеме (вложенность типов в полях не ограничивается), либо скалярными типами данных: Int (знаковое 32-битное целое число), Float (знаковое число с плавающей запятой двойной точности), String (последовательность символов UTF-8), Boolean (булев тип), Id (уникальный идентификатор, сериализируемый как String). Также, тип значения поля может быть промаркирован как «ненулевой» символом «!». Тем самым гарантируется, что это поле никогда не вернет нулевое значение [20].

Например, пусть есть тип «книга», который имеет поля: «название», «автор», «издатель», «год_издания», тогда существует некоторый набор столбцов (полей) из базы данных веб-приложения, содержащий соответствующие именам полей данные. Притом, полям «название», «год_издания» будет соответствовать по одному столбцу, а полям «автор» и «издатель» по множеству, так как те являются типами объектов.

Схема также определяет конкретные запросы, мутации, подписки для пользовательской иерархии типов. Запросы — операции получения данных. Мутации — операции изменения данных. Подписки — операции получения

данных длительного действия, возвращающие обновленные значения при определенных событиях на сервере.

3.1.2 Express

Express — самый популярный веб-фреймворк на платформе Node и основа для других фреймворков. В спектр возможностей, реализуемых с Express, входит: установка базовых параметров веб-приложений, таких как номер порта подключения или выбор расположения шаблонов для отрисовки ответа; написание обработчиков запросов с разными HTTP-глаголами (GET, POST и другие) для разных URL-путей; добавление в линии обработки запросов дополнительных программных приложений («middleware»).

Пусть и Express достаточно минималистичен в своей структуре, сообщество программистов разработало достаточно «middleware» для решения практически любых задач веб-разработки: обработка «cookies», сохранение данных сессий пользователей, авторизация пользователей и другие [21].

3.1.3 Apollo-server-express

Apollo Server — это библиотека, интегрирующая схему GraphQL и HTTP сервер на платформе Node. Apollo Server может использоваться в качестве:

- изолированного сервера GraphQL;
- дополнительного «middleware» в линии обработки запросов;
- шлюза для интегрированного графа данных.

3.1.4 Babel-cli

Babel — это набор инструментальных средств, используемый для конвертации (транспиляции) кода, написанного на современных версиях языка JavaScript, в тот, что совместим с устаревшими браузерами или средами исполнения. Одно из средств — babel-cli — используется для исполнения команд компиляции файлов в командной строке [35].

3.1.5 Body-parser

Body-parser — это Express «middleware», используемое для чтения данных HTTP POST и хранения их в виде объекта JavaScript, доступного для других функций. HTTP-метод POST служит для отправки информации на сервер и обычно отправляется через элемент страницы — форму отправки данных [36].

3.1.6 Nodemon

Nodemon — инструмент, который ускоряет разработку приложений Node, перезапуская активное приложение, при изменении файлов проекта.

3.1.7 Pg-hstore

Этот пакет реализует пользовательский тип данных hstore в базе PostgreSQL. Структура типа данных hstore — пара ключ-значение.

3.1.8 Sequelize

Sequelize — инструмент реализации технологии объектно-реляционного отображения (ORM) на платформе Node, совместимый с PostgreSQL, MySQL и другими СУБД. Sequelize поддерживает в полной мере: транзакции, связи, шаблоны проектирования загрузки данных (Eager/Lazy loading) и другие операции [37].

3.2 NPM-пакеты для клиентской части приложения

3.2.1 React

React — библиотека для построения пользовательских интерфейсов. Главная цель этой библиотеки — предотвратить серьезные программные ошибки при построении интерфейсов, предоставляя для использования «компоненты» — фрагменты кода, описывающие логику работы отдельных

элементов интерфейса. React также абстрагирует механизмы отрисовки элементов для упрощения процесса разработки интерфейсов.

Код «компонентов» пишется на JavaScript с особенным дополнением синтаксиса — JSX. JSX имеет общие признаки с языками разметки, но после компиляции JSX-выражения переводятся в вызов обычной JavaScript-функции [38].

3.2.2 React Apollo

React Apollo — библиотека для получения данных с сервера GraphQL и построения на их основе и React сложных и динамических пользовательских интерфейсов. React Apollo позволяет выполнять операции: получения, изменения, кэширования данных React-приложений, автоматически обновляя элементы пользовательского интерфейса [39].

3.2.3 React-dom

В React используется независимая от браузера DOM-система (Document Object Model — стандарт получения, изменения, добавления, удаления HTML элементов), предоставляющая улучшенную совместимость веб-приложения с разными браузерами. Библиотека React-dom позволяет использовать методы для работы с DOM [22].

3.2.4 MobX

MobX — библиотека управления состояниями, следующая принципам функционального реактивного программирования, основная функция которой — предотвращение неявных состояний компонентов React [40].

3.2.5 Semantic-ui-react

Semantic-ui-react — реализация фронтенд-фреймворка Semantic UI для React, используемая для тематического оформления интерфейса веб-приложений [41].

3.2.6 Styled-components

Styled Components — это библиотека React для написания и управления фрагментов CSS (Cascading Style Sheets, язык описания внешнего вида веб-документов). Styled Components относится к технологиям «CSS-in-JS», следовательно CSS-код допускается размещать в файлах Javascript [23].

3.2.7 Jwt-decode

Jwt-decode — браузерная библиотека, основная функция которой — декодирование JWT токенов (JSON Web Tokens), стандартизированных средств аутентификации [42].

Глава 4. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Проектирование концептуальных схем связей таблиц базы данных, схемы GraphQL является определяющим моментом при построении архитектуры веб-приложения. Ошибки, заложенные на этом этапе, в дальнейшем ведут к серьезной работе по их устранению, так как для этого может потребоваться внесение изменений во все программные модули веб-приложения.

4.1 Проектирование базы данных PostgreSQL

Исходя из анализа предметной области (различных мессенджеров) были определены основные сущности для реализации реляционной базы данных: пользователи, сообщения, каналы, команды.

В соответствии с сущностями были реализованы таблицы [24]:

- channels
- members
- teams
- channel_member
- messages
- users

4.1.1 Таблица Channels

Эта таблица (Рис. 1) содержит записи о каналах, созданных пользователями. Имеет следующие атрибуты:

- Id — идентификатор.
    - Тип данных — serial.
    - Поле обязательное — да. Имеется в виду, что поле не может хранить значение равное NULL.
    - Поле ключевое — первичный ключ.
- Name — имя канала.

    - Тип данных — varchar.
    - Поле обязательное — нет. Имеется в виду, что имя канала может не содержать ни одного символа.
    - Поле ключевое — нет.
- Public — публичность канала.
    - Тип данных — bool.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Created_at — временная отметка о создании записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Updated_at — временная отметка о обновлении записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Team_id — идентификатор записи о команде.
    - Тип данных — int4.
    - Поле обязательное — нет. Первичный ключ из таблицы Teams уже является обязательным.
    - Поле ключевое — вторичный ключ.

Также эта таблица имеет следующие связи с другими таблицами:

- Связь «один ко многим» с таблицей Teams.
    - Первичный ключ — поле Id таблицы Teams.
    - Вторичный ключ — поле Team_id таблицы Channels.
- Связь «один ко многим» с таблицей Messages.
    - Первичный ключ — поле Id таблицы Channels.
    - Вторичный ключ — поле Channel_id таблицы Messages.
- Связь «один ко многим» с таблицей Channel_member.

- Первичный ключ — поле Id таблицы Channels.
- Вторичный ключ — поле Channel_id таблицы Channel_member.

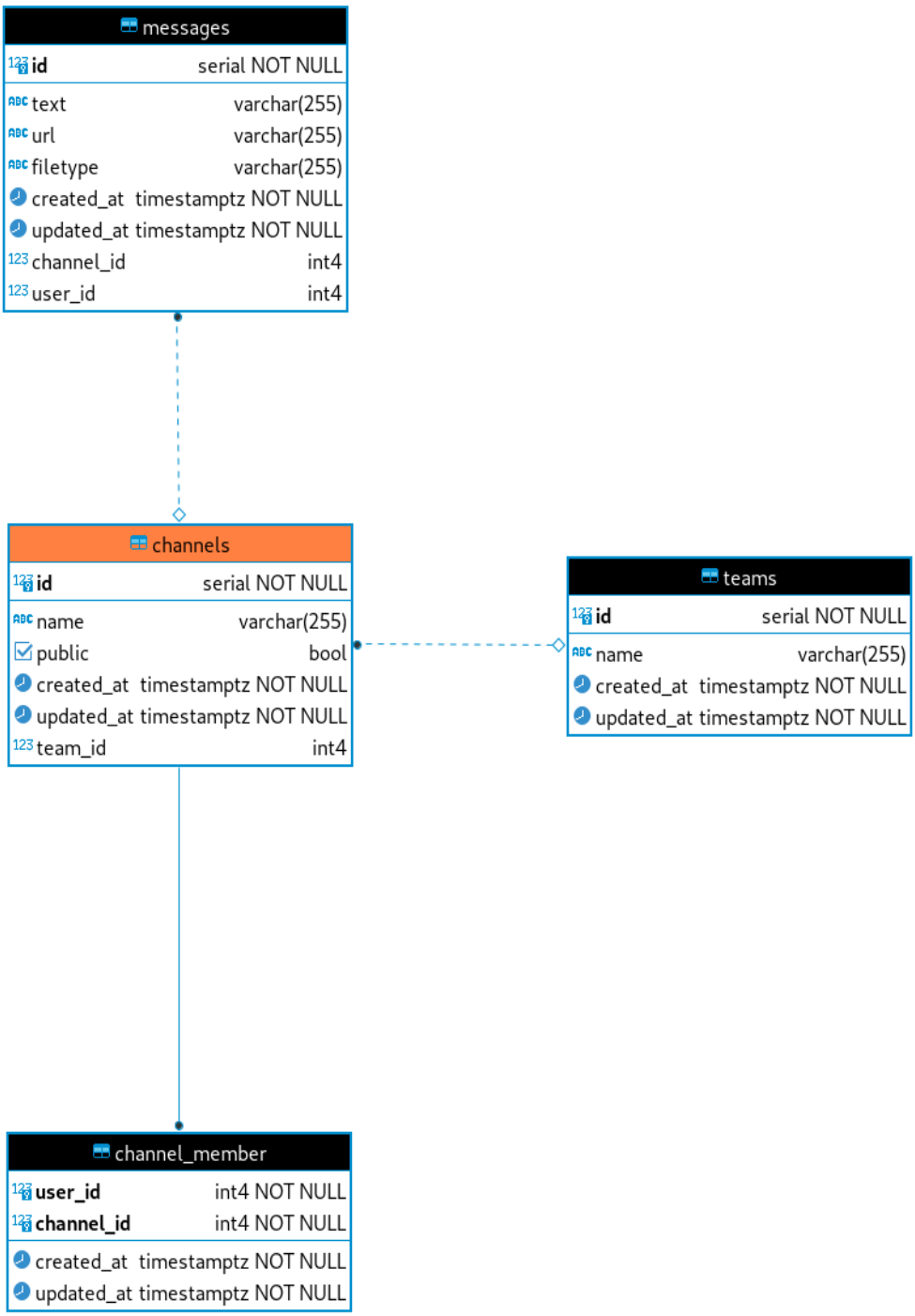

Рисунок 1 — концептуальная схема связей таблицы Channels. Рисунок выполнен автором.

4.1.2 Таблица Members

Эта таблица (Рис. 2) содержит записи о принадлежности пользователей к разным командам. Имеет следующие атрибуты:

- User_id — идентификатор записи о пользователе.
    - Тип данных — int4.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Team_id — идентификатор записи о команде.
    - Тип данных — int4.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Admin — наличие прав администратора у пользователя в конкретной команде.
    - Тип данных — bool.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Created_at — временная отметка о создании записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Updated_at — временная отметка о обновлении записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.

Также эта таблица является промежуточной в связи «многие ко многим» таблиц Teams и Users:

- Первичный ключ — поле Id таблицы Teams.
- Первичный ключ — поле Id таблицы Users.
- Первичный ключ — комбинация полей Team_id и User_id таблицы Members.

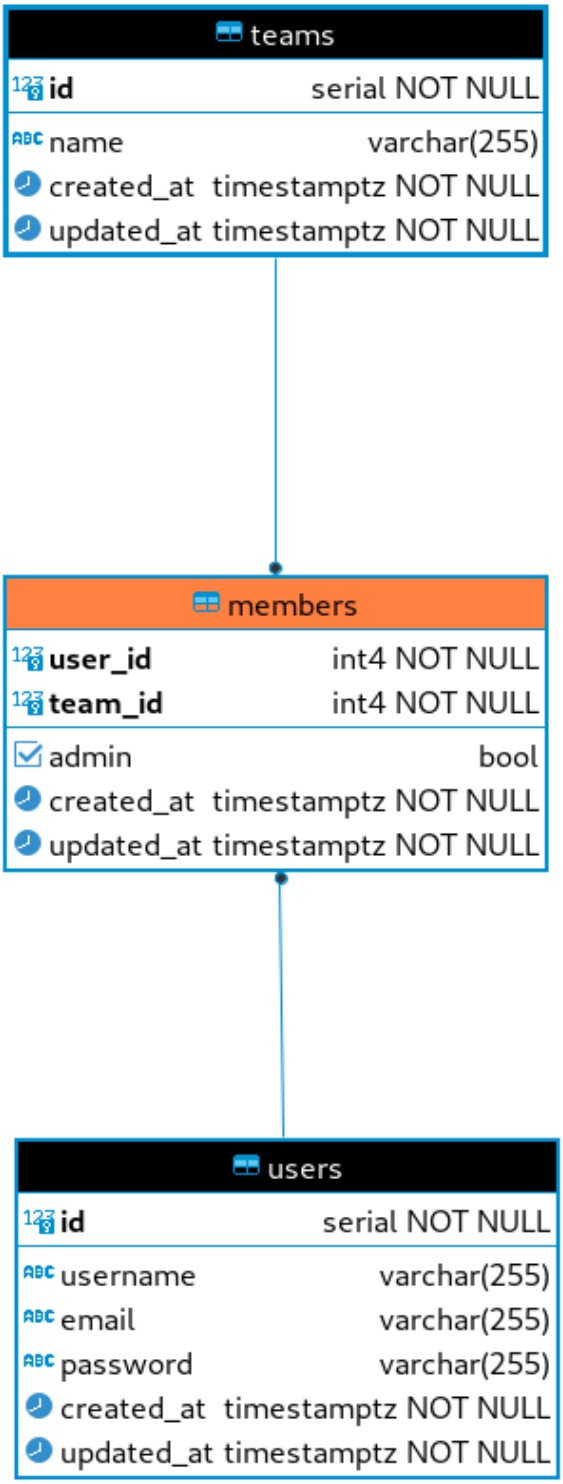

Рисунок 2 — концептуальная схема связей таблицы Members. Рисунок выполнен автором.

4.1.3 Таблица Channel_member

Эта таблица (Рис. 3) содержит записи о принадлежности пользователей к разным каналам. Имеет следующие атрибуты:

- User_id — идентификатор записи о пользователе.
    - Тип данных — int4.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Channel_id — идентификатор.
    - Тип данных — int4.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Created_at — временная отметка о создании записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Updated_at — временная отметка о обновлении записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.

Также эта таблица является промежуточной в связи «многие ко многим» таблиц Channels и Users:

- Первичный ключ — поле Id таблицы Channels.
- Первичный ключ — поле Id таблицы Users.
- Первичный ключ — комбинация полей Channel_id и User_id таблицы Channel_member.

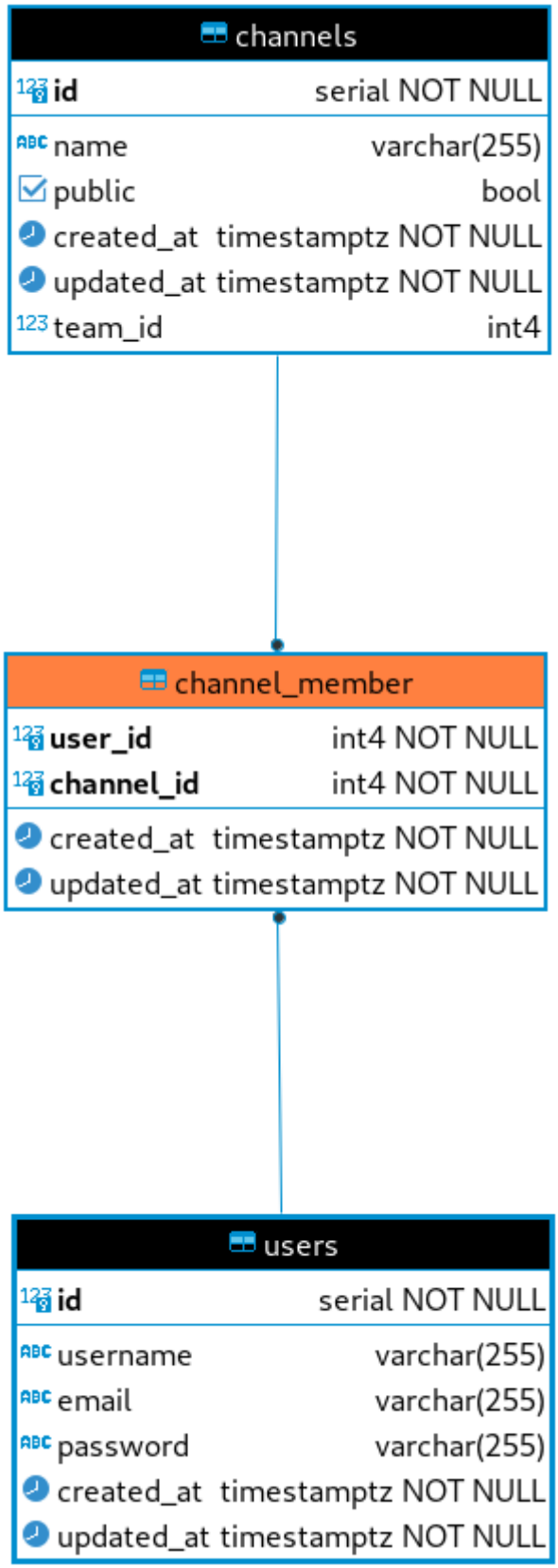

Рисунок 3 — концептуальная схема связей таблицы Channel_member.

Рисунок выполнен автором.

4.1.4 Таблица Messages

Эта таблица (Рис. 4) содержит записи о сообщениях, созданных пользователями. Имеет следующие атрибуты:

- Id — идентификатор.
    - Тип данных — serial.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Text — текст сообщения.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Url — указатель местонахождения ресурса.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Filetype — тип передаваемого медиа в сообщении.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Created_at — временная отметка о создании записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Updated_at — временная отметка о обновлении записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- User_id — идентификатор записи о пользователе.
    - Тип данных — int4.

- Поле обязательное — нет.
- Поле ключевое — вторичный ключ.

Также эта таблица имеет следующие связи с другими таблицами:

- Связь «один ко многим» с таблицей Users.
  - Первичный ключ — поле Id таблицы Users.
  - Вторичный ключ — поле User_id таблицы Messages.
- Связь «один ко многим» с таблицей Channels.
  - Первичный ключ — поле Id таблицы Channels.
  - Вторичный ключ — поле Channel_id таблицы Messages.

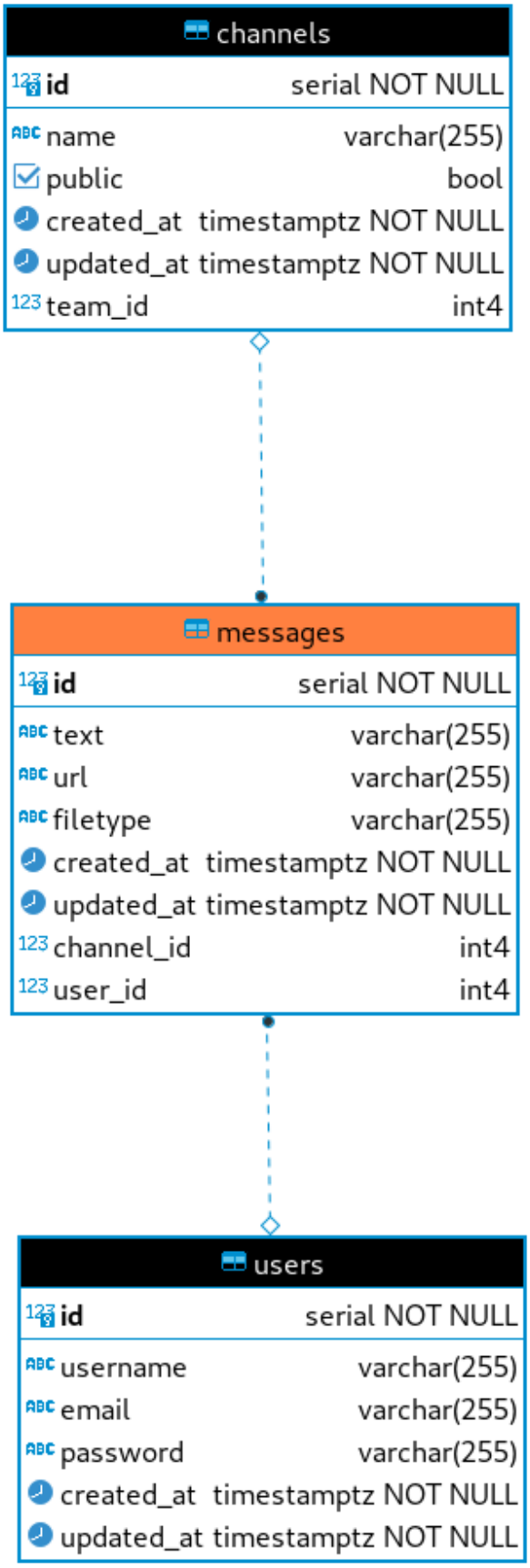

Рисунок 4 — концептуальная схема связей таблицы Messages. Рисунок выполнен автором.

4.1.5 Таблица Teams

Эта таблица (Рис. 5) содержит записи о командах, созданных пользователями. Имеет следующие атрибуты:

- Id — идентификатор.
    - Тип данных — serial.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Name — имя команды.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Created_at — временная отметка о создании записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Updated_at — временная отметка о обновлении записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.

  Также эта таблица имеет следующие связи с другими таблицами:

    - Связь «один ко многим» с таблицей Members.
        - Первичный ключ — поле Id таблицы Teams.
        - Вторичный ключ — поле Team_id таблицы Members.
    - Связь «один ко многим» с таблицей Channels.
        - Первичный ключ — поле Id таблицы Teams.
        - Вторичный ключ — поле Team_id таблицы Channels.

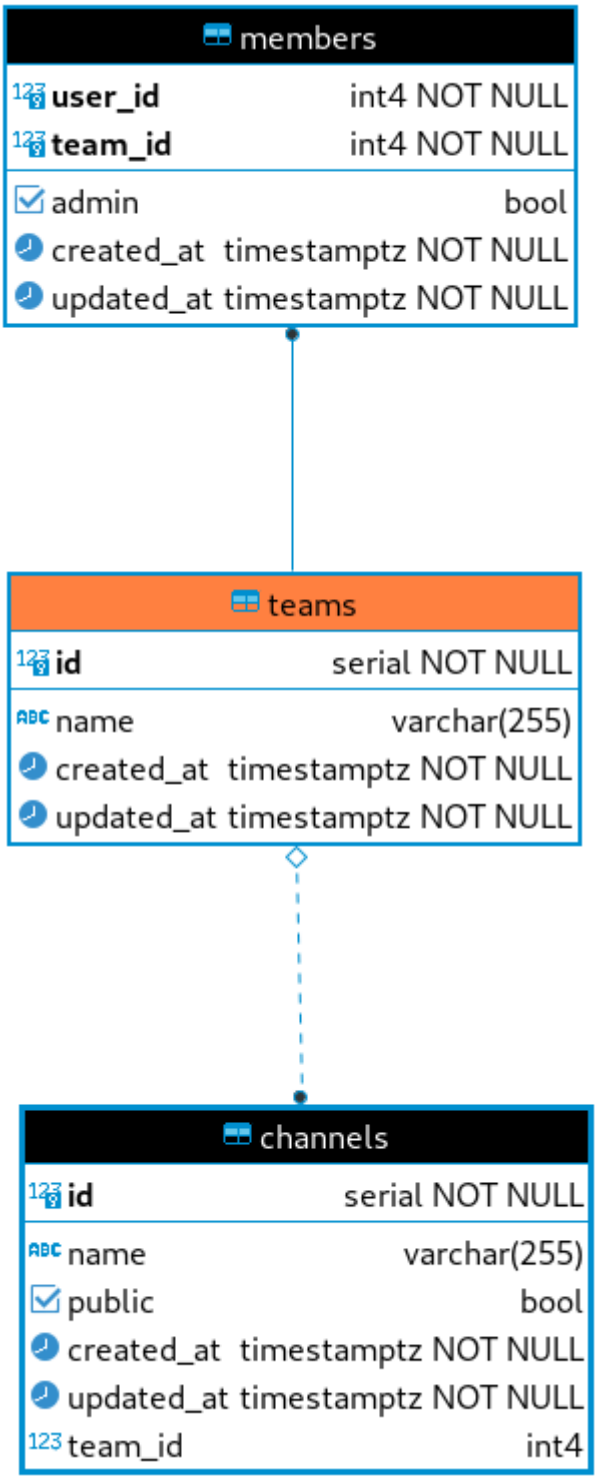

Рисунок 5 — концептуальная схема связей таблицы Teams. Рисунок выполнен автором.

4.1.6 Таблица Users

Эта таблица (Рис. 6) содержит записи о данных пользователей. Имеет следующие атрибуты:

- Id — идентификатор.
    - Тип данных — serial.
    - Поле обязательное — да.
    - Поле ключевое — первичный ключ.
- Username — никнейм пользователя.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Email — адрес электронной почты пользователя.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Password — данные о пароле пользователя.
    - Тип данных — varchar.
    - Поле обязательное — нет.
    - Поле ключевое — нет.
- Created_at — временная отметка о создании записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.
- Updated_at — временная отметка о обновлении записи.
    - Тип данных — timestamptz.
    - Поле обязательное — да.
    - Поле ключевое — нет.

  Также эта таблица имеет следующие связи с другими таблицами:

    - Связь «один ко многим» с таблицей Messages.
        - Первичный ключ — поле Id таблицы Users.
        - Вторичный ключ — поле User_id таблицы Messages.
    - Связь «один ко многим» с таблицей Members.
        - Первичный ключ — поле Id таблицы Users.

- Вторичный ключ — поле User_id таблицы Members.

- Связь «один ко многим» с таблицей Channel_member.
  - Первичный ключ — поле Id таблицы Users.
  - Вторичный ключ — поле User_id таблицы Channel_member.

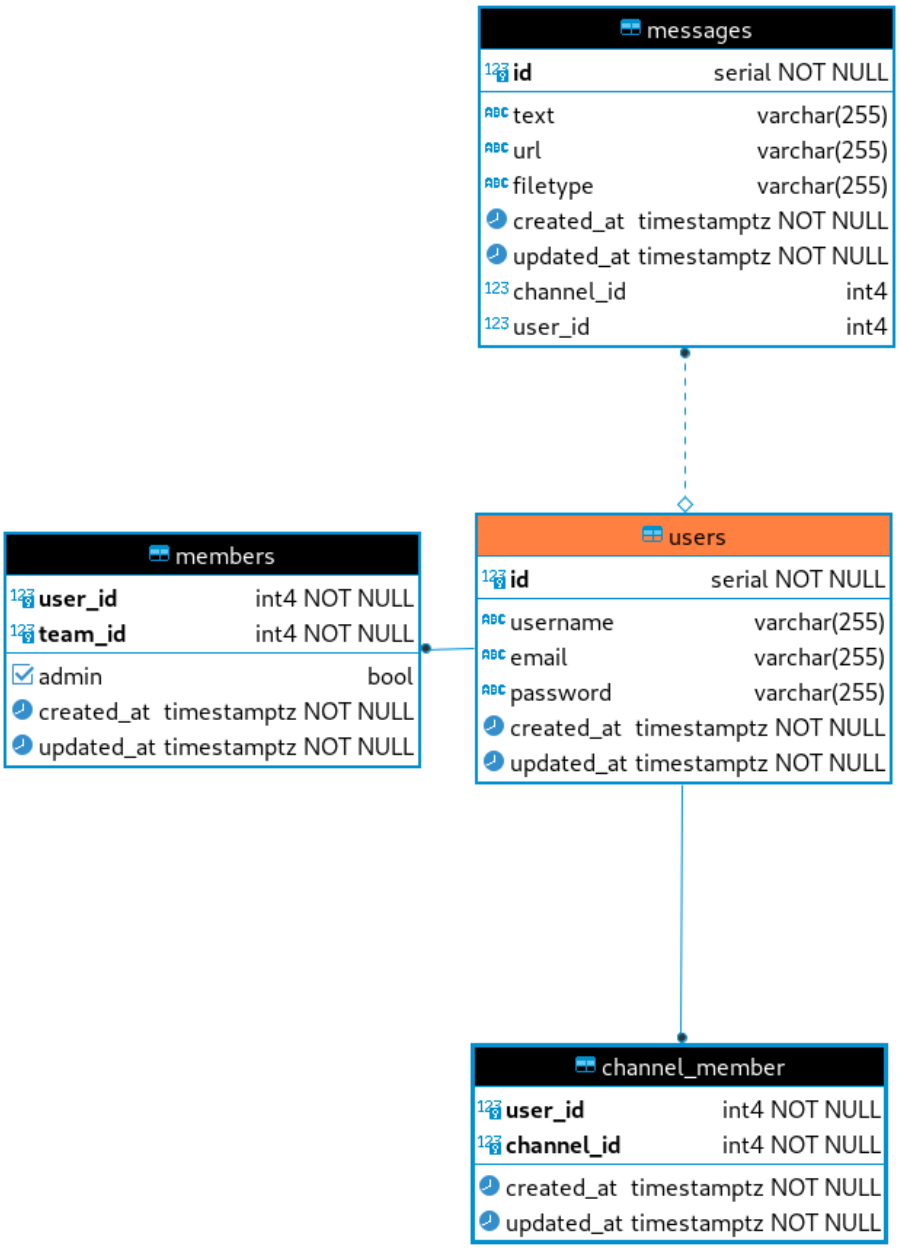

Рисунок 6 — концептуальная схема связей таблицы Users. Рисунок выполнен автором.

4.2 Проектирование схемы GraphQL

Исходя из спроектированной реляционной базы данных была спроектирована схема GraphQL с конкретными типами объектов, запросами, мутациями и подписками.

4.2.1 Проектирование типов объектов GraphQL

4.2.1.1 Channel

Этот тип (Рис. 7) декларирует структуру данных в различных запросах и мутациях, направленных к записям базы данных, содержащих информацию об объектах каналов (отправители сообщений в каналах, содержание отправленных сообщений).

Поля этого типа:

- id
    - Описание — поле содержит идентификатор канала.
    - Тип значения поля — Int.
        - Скалярный тип — да. Скалярный означает, что тип является частью GraphQL и не определяется разработчиком.
    - Поле обязательное — да. Имеется в виду, должно ли поле содержать значение.
- name
    - Описание — поле содержит имя канала.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — да.
- public
    - Описание — поле содержит информацию о публичности канала.
    - Тип значения поля — Boolean.
        - Скалярный тип — да.
    - Поле обязательное — да.

- messages
    - Описание — поле содержит массив элементов Message, каждый из которых состоит из выборки полей этого типа.
    - Тип значения поля — Message.
        - Скалярный тип — нет.
    - Поле обязательное — да.
- users
    - Описание — поле содержит массив элементов типа User, каждый из которых состоит из выборки полей этого типа.
    - Тип значения поля — User.
        - Скалярный тип — нет.
    - Поле обязательное — да.

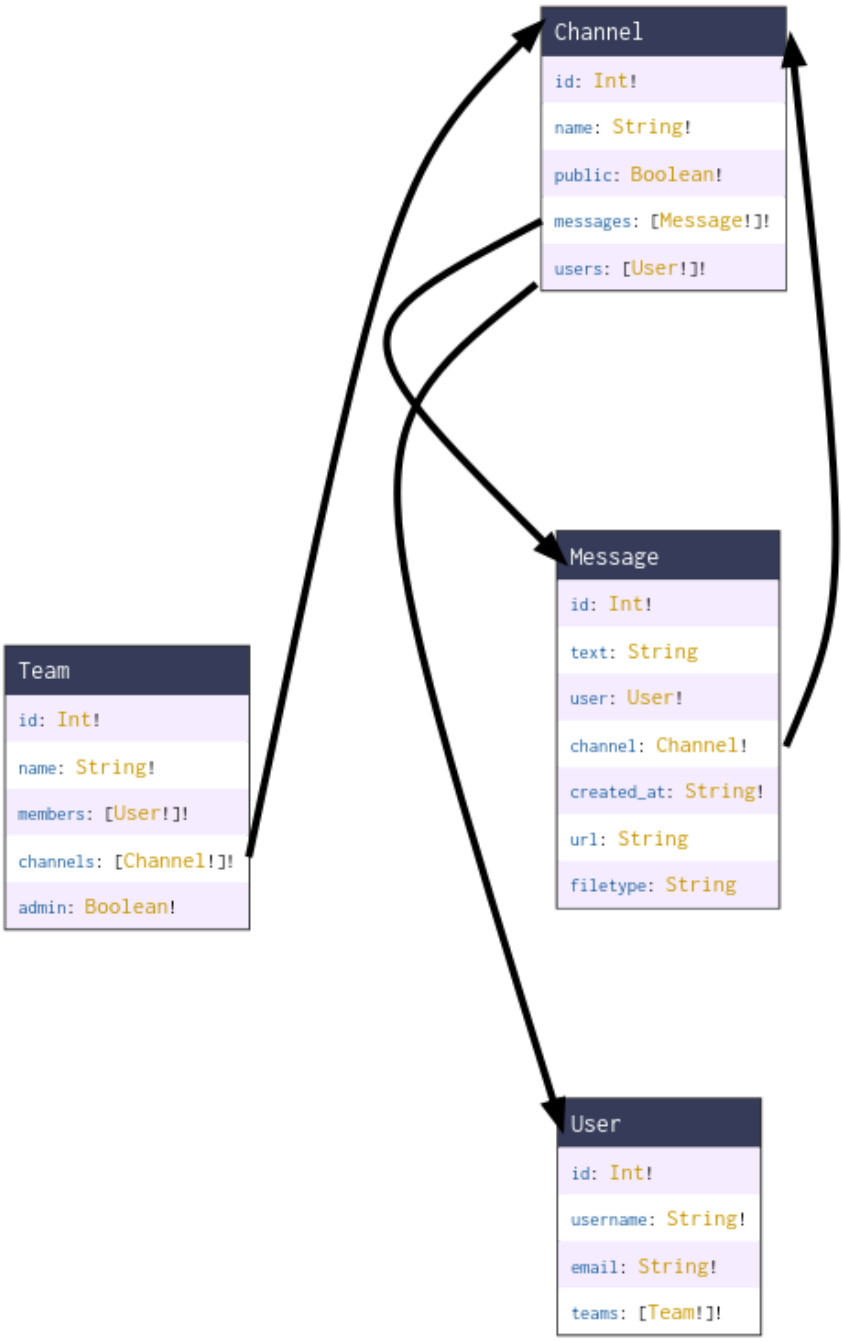

Рисунок 7 — концептуальная схема связей типа Channel с другими типами и объектами GraphQL. Рисунок выполнен автором.

4.2.1.2 Message

Этот тип (Рис. 8) декларирует структуру данных в различных запросах и мутациях, запрашивающих или передающих информацию об отправленных пользователями сообщениях и файлах.

Поля этого типа:

- id
    - Описание — поле содержит идентификатор сообщения.
    - Тип значения поля — Int.
        - Скалярный тип — да. Скалярный означает, что тип является частью GraphQL и не определяется разработчиком.
    - Поле обязательное — да. Имеется в виду, должно ли поле содержать значение.
- text
    - Описание — поле содержит текст сообщения.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — нет.
- user
    - Описание — поле содержит тип User.
    - Тип значения поля — User.
        - Скалярный тип — да.
    - Поле обязательное — да.
- channel
    - Описание — поле содержит тип Channel.
    - Тип значения поля — Channel.
        - Скалярный тип — нет.
    - Поле обязательное — да.
- created_at
    - Описание — поле содержит временную отметку.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — да.
- url
    - Описание — поле содержит указатель местонахождения ресурса.
    - Тип значения поля — String.

        - Скалярный тип — да.
    - Поле обязательное — нет.
- filetype
    - Описание — поле содержит тип передаваемого медиа в сообщении.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — нет.

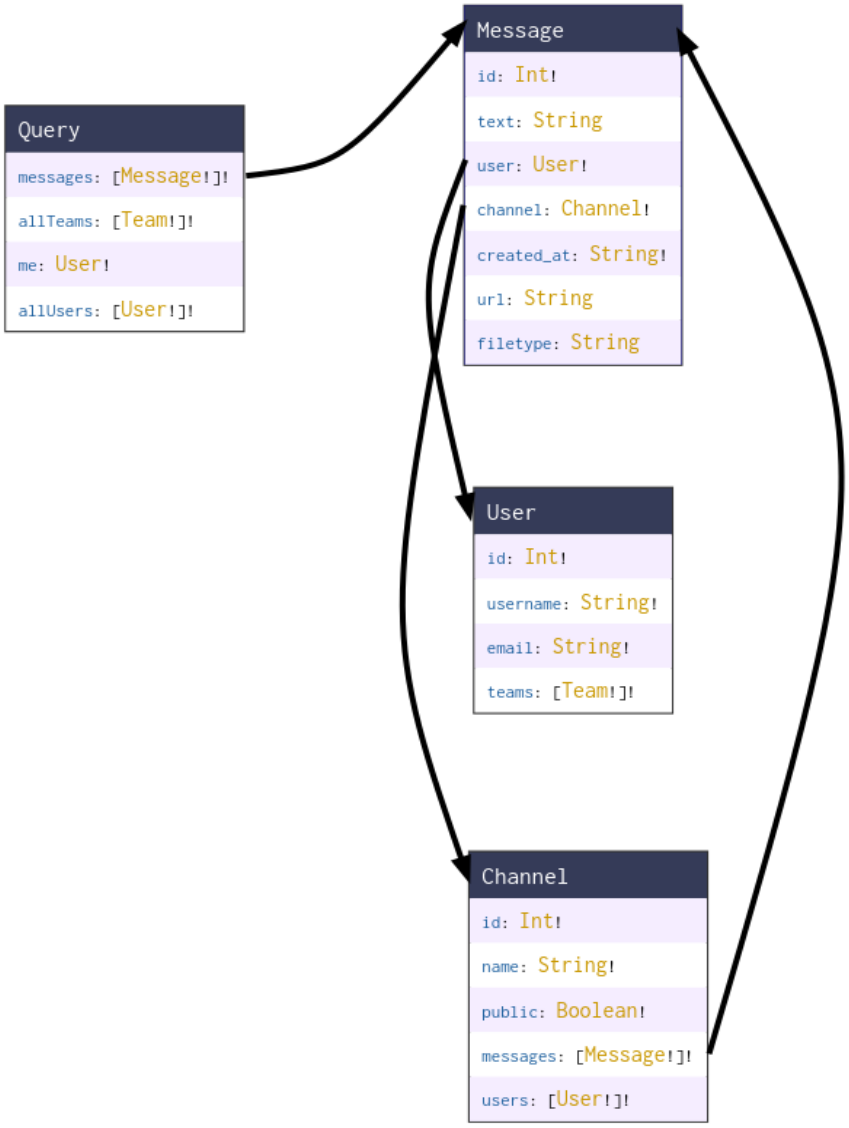

Рисунок 8 — концептуальная схема связей типа Message с другими типами и объектами GraphQL. Рисунок выполнен автором.

4.2.1.3 User

Этот тип (Рис. 9) декларирует структуру данных в различных запросах и мутациях, реализующих авторизацию и аутентификацию пользователей.

Поля этого типа:

- id
    - Описание — поле содержит идентификатор канала.
    - Тип значения поля — Int.
        - Скалярный тип — да. Скалярный означает, что тип является частью GraphQL и не определяется разработчиком.
    - Поле обязательное — да. Имеется в виду, должно ли поле содержать значение.
- username
    - Описание — поле содержит никнейм пользователя.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — да.
- email
    - Описание — поле содержит адрес почты пользователя.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — да.
- teams
    - Описание — поле содержит массив элементов Team, каждый из которых состоит из выборки полей этого типа.
    - Тип значения поля — Team.
        - Скалярный тип — нет.
    - Поле обязательное — да.

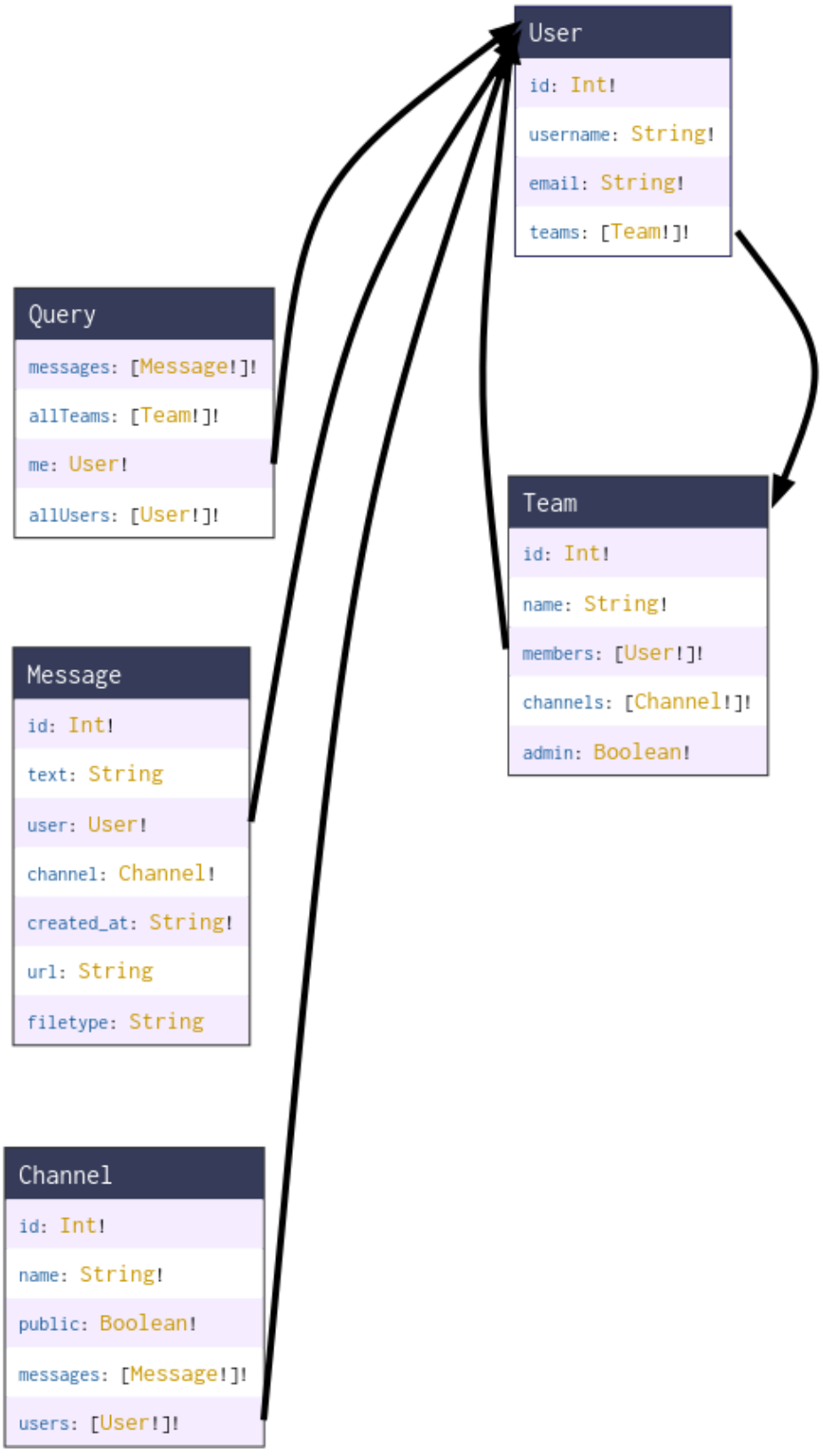

Рисунок 9 — концептуальная схема связей типа User с другими типами и объектами GraphQL. Рисунок выполнен автором.

4.2.1.4 Team

Этот тип (Рис. 10) декларирует структуру данных в различных запросах и мутациях, направленных к записям базы данных, содержащих информацию об объектах команд (каналы конкретной команды, члены команд и их права).

Поля этого типа:

- id
    - Описание — поле содержит идентификатор команды.
    - Тип значения поля — Int.
        - Скалярный тип — да. Скалярный означает, что тип является частью GraphQL и не определяется разработчиком.
    - Поле обязательное — да. Имеется в виду, должно ли поле содержать значение.
- name
    - Описание — поле содержит имя команды.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — да.
- members
    - Описание — поле содержит массив элементов User, каждый из которых состоит из выборки полей этого типа.
    - Тип значения поля — User.
        - Скалярный тип — нет.
    - Поле обязательное — да.
- channels
    - Описание — поле содержит массив элементов Channel, каждый из которых состоит из выборки полей этого типа.
    - Тип значения поля — Channel.
        - Скалярный тип — нет.
    - Поле обязательное — да.

- admin
    - Описание — поле определяет наличие прав администратора у пользователя в конкретной команде.
    - Тип значения поля — Boolean.
        - Скалярный тип — да.
    - Поле обязательное — да.

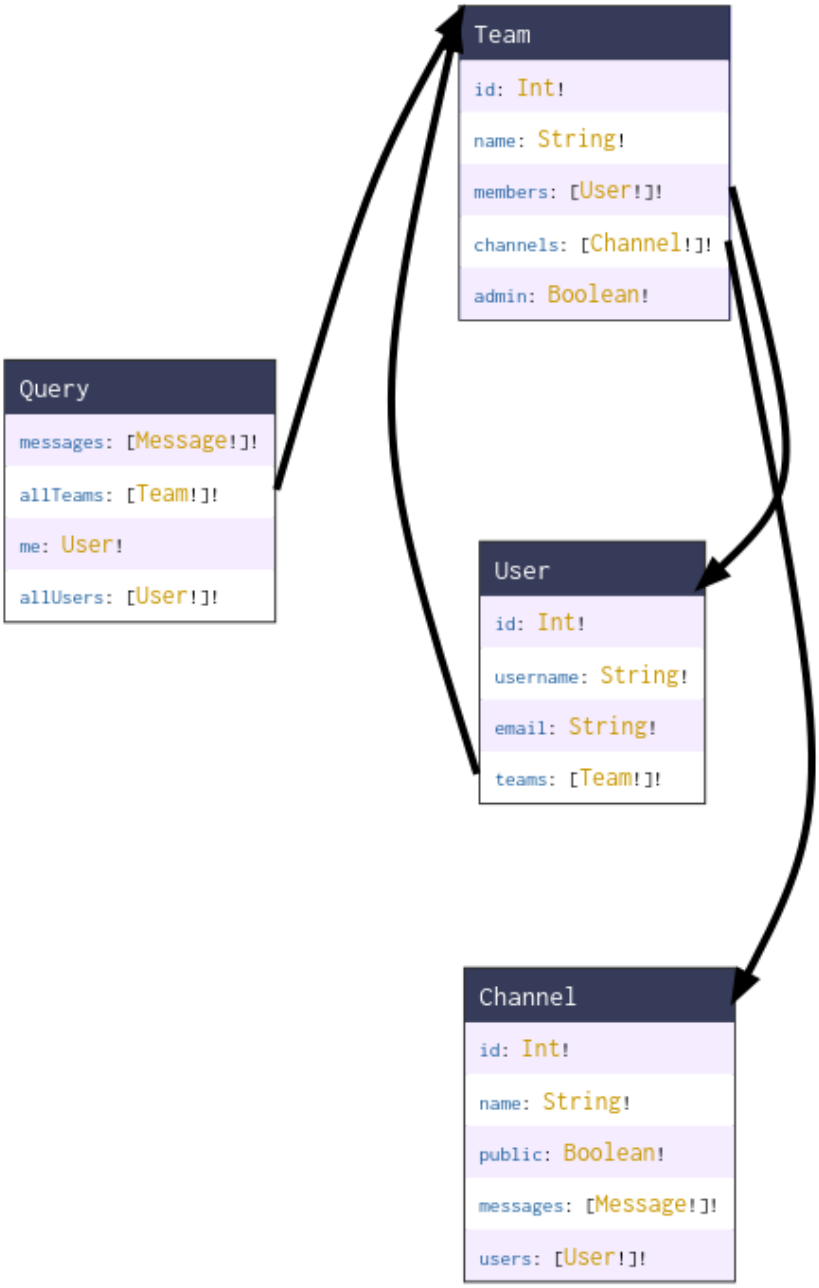

Рисунок 10 — концептуальная схема связей типа Team с другими типами и объектами GraphQL. Рисунок выполнен автором.

4.2.1.5 Error

Этот тип (Рис. 11) декларирует структуру данных возвращаемых сервером ошибок при реализации различных запросов и мутаций.

Поля этого типа:

- path
    - Описание — поле содержит маршрут, по которому перенаправляется запрос в момент возникновения ошибки.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — да.
- message
    - Описание — поле содержит текст сообщения, характеризующего ошибку.
    - Тип значения поля — String.
        - Скалярный тип — да.
    - Поле обязательное — нет.

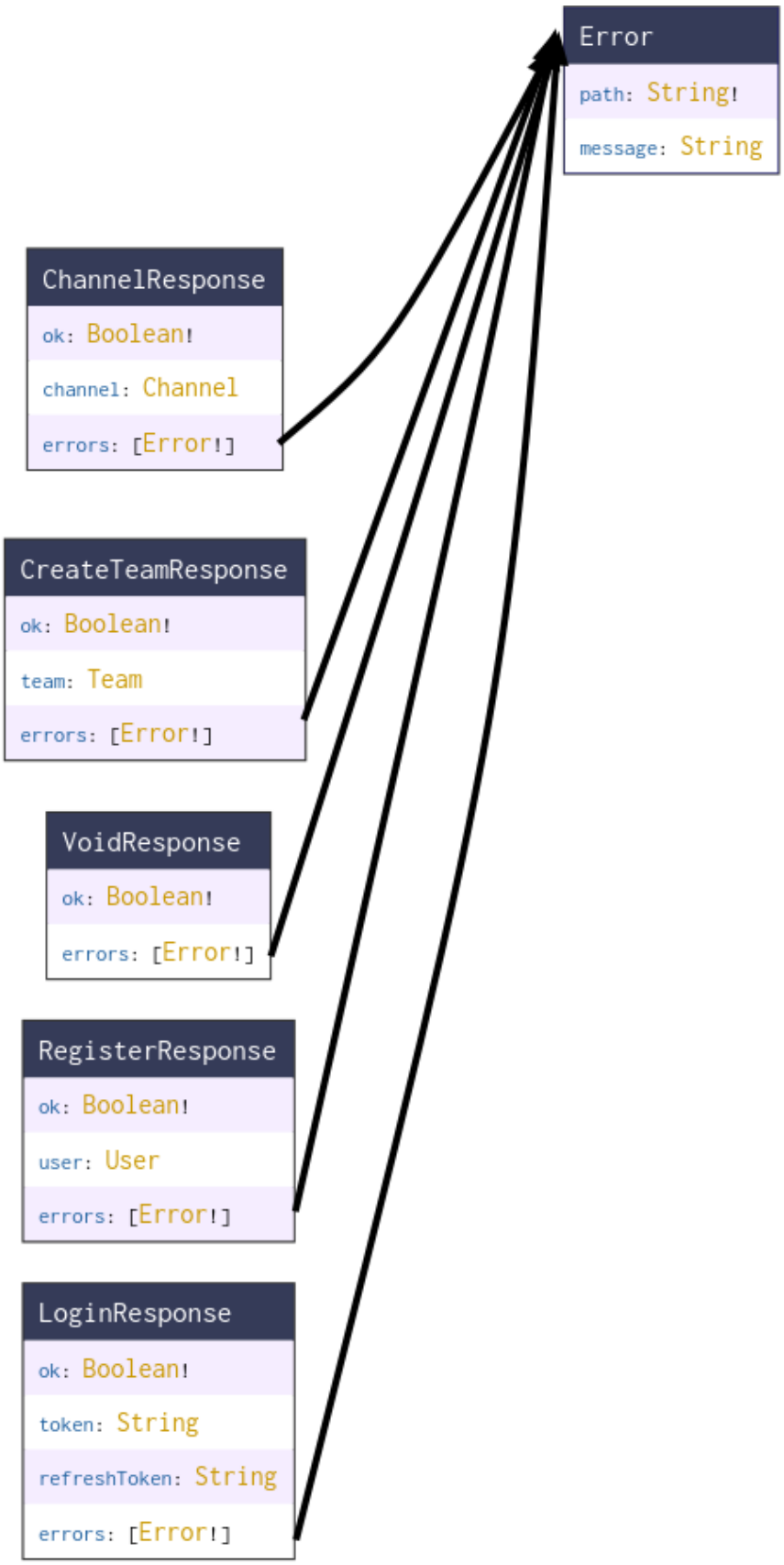

Рисунок 11 — концептуальная схема связей типа Error с другими типами и объектами GraphQL. Рисунок выполнен автором.

4.2.1.6 File

Этот тип декларирует структуру входных данных в различных запросах и мутациях, запрашивающих или передающих информацию об отправленных пользователями файлах.

Поля этого типа:

- type
  - Описание — поле содержит расширение посылаемого медиа.
  - Тип значения поля — String.
    - Скалярный тип — да.
  - Поле обязательное — да.
- path
  - Описание — поле содержит указатель местонахождения ресурса.
  - Тип значения поля — String.
    - Скалярный тип — да.
  - Поле обязательное — да.

4.2.2 Проектирование мутаций

4.2.2.1 CreateChannel

Эта мутация (Рис. 12) передает данные для добавления записи о новом канале в базу данных.

- Аргументы мутации:
  - teamId
    - Описание — поле содержит идентификатор команды, в которую добавляется новый канал.
    - Тип значения поля — Int.
      - Скалярный тип — да. Скалярный означает, что тип является частью GraphQL и не определяется разработчиком.

        - Поле обязательное — да. Имеется в виду, должно ли поле содержать значение при отправке мутации.
    - name
        - Описание — поле содержит имя нового канала.
        - Тип значения поля — String.
            - Скалярный тип — да.
        - Поле обязательное — да.
    - public
        - Описание — поле содержит информацию о публичности канала.
        - Тип значения поля — String.
            - Скалярный тип — да.
        - Поле обязательное — нет.
- Возвращаемый мутацией тип — ChannelResponse. Этот тип позволяет определить успешность мутации и получить ошибки при необходимости. Тип содержит следующие поля:
    - ok
        - Описание — поле содержит код ответа об успешном статусе http-запроса.
        - Тип значения поля — Boolean.
            - Скалярный тип — да.
        - Поле обязательное — да.
    - channel
        - Описание — поле содержит выборку полей типа Channel.
        - Тип значения поля — Channel.
            - Скалярный тип — нет.
        - Поле обязательное — нет.
    - errors
        - Описание — поле содержит массив элементов типа Error.
        - Тип значения поля — Error.

- Скалярный тип — нет.

■ Поле обязательное — да.

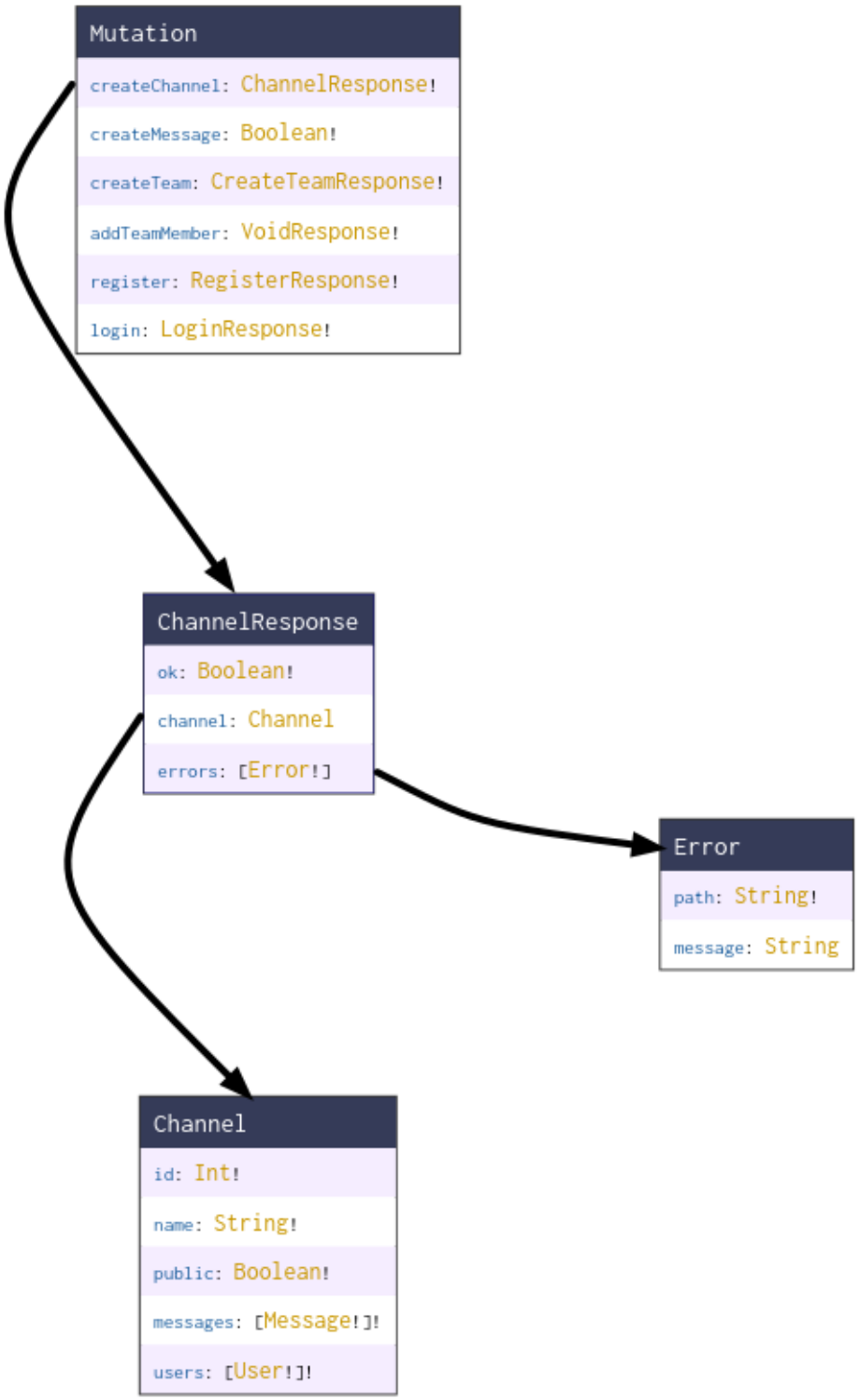

Рисунок 12 — концептуальная схема полей мутации createChannel.

Рисунок выполнен автором.

4.2.2.2 CreateMessage

Эта мутация (Рис. 13) передает данные для добавления записи о новом сообщении в базу данных.

- Аргументы мутации:
    - channelId
        - Описание — поле содержит идентификатор канала, в который отправляется сообщение.
        - Тип значения поля — Int.
            - Скалярный тип — да.
        - Поле обязательное — да.
    - text
        - Описание — поле содержит текст сообщения.
        - Тип значения поля — String.
            - Скалярный тип — да.
        - Поле обязательное — нет.
    - file
        - Описание — поле содержит ссылку на посылаемый файл.
        - Тип значения поля — File.
            - Скалярный тип — нет.
        - Поле обязательное — нет.
- Возвращаемый мутацией тип — Boolean.

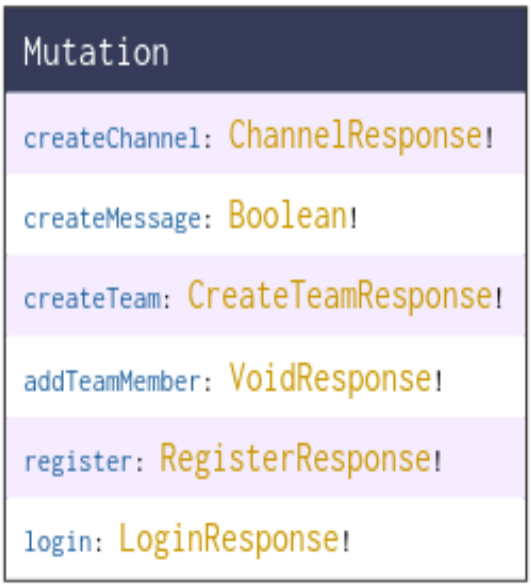

Рисунок 13 — концептуальная схема полей мутации createMessage.

Рисунок выполнен автором.

4.2.2.3 CreateTeam

Эта мутация (Рис. 14) передает данные для добавления записи о новой команде в базу данных.

- Аргументы мутации:
  - name
    - Описание — поле содержит имя новой команды.
    - Тип значения поля — String.
      - Скалярный тип — да.
    - Поле обязательное — да.
- Возвращаемый мутацией тип — CreateTeamResponse. Этот тип позволяет определить успешность мутации и получить ошибки при необходимости. Тип содержит следующие поля:
  - ok
    - Описание — поле содержит код ответа об успешном статусе http-запроса.
    - Тип значения поля — Boolean.
      - Скалярный тип — да.

        - Поле обязательное — да.
    - team
        - Описание — поле содержит выборку полей типа Team.
        - Тип значения поля — Team.
            - Скалярный тип — нет.
        - Поле обязательное — нет.
    - errors
        - Описание — поле содержит массив элементов типа Error.
        - Тип значения поля — Error.
            - Скалярный тип — нет.
        - Поле обязательное — да.

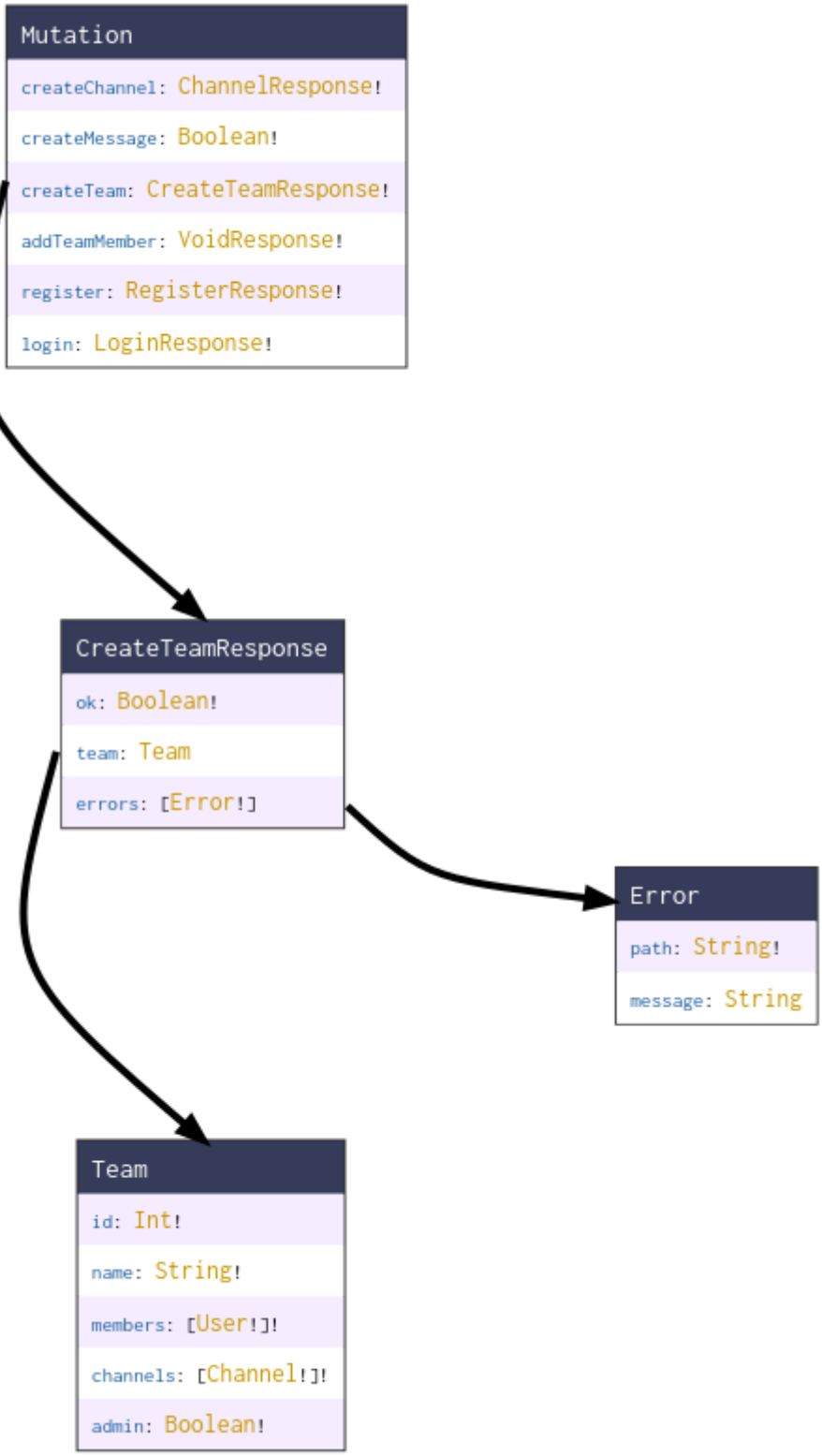

Рисунок 14 — концептуальная схема полей мутации createTeam. Рисунок выполнен автором.

4.2.2.4 AddTeamMember

Эта мутация (Рис. 15) передает данные для добавления записи о новом члене команды в базу данных.

- Аргументы мутации:
  - email
    - Описание — поле содержит адрес почты нового члена команды.
    - Тип значения поля — String.
      - Скалярный тип — да.
    - Поле обязательное — да.
  - teamId
    - Описание — поле содержит идентификатор команды, в которую добавляется пользователь.
    - Тип значения поля — Int.
      - Скалярный тип — да.
    - Поле обязательное — да.

- Возвращаемый мутацией тип — VoidResponse. Этот тип позволяет определить успешность мутации и получить ошибки при необходимости. Тип содержит следующие поля:
  - ok
    - Описание — поле содержит код ответа об успешном статусе http-запроса.
    - Тип значения поля — Boolean.
      - Скалярный тип — да.
    - Поле обязательное — да.
  - errors
    - Описание — поле содержит массив элементов типа Error.
    - Тип значения поля — Error.

- Скалярный тип — нет.

■ Поле обязательное — да.

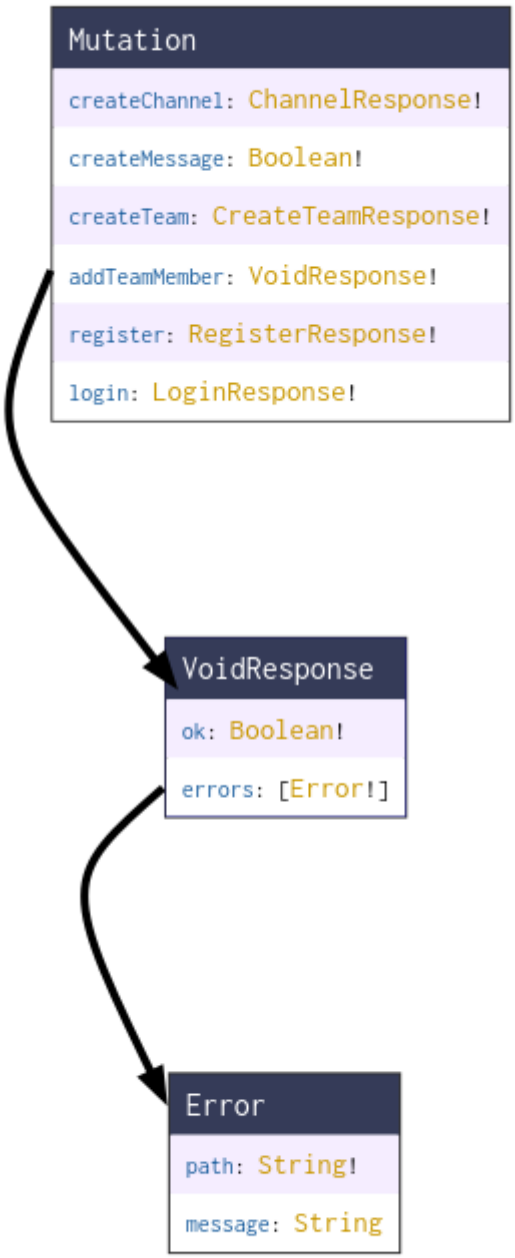

Рисунок 15 — концептуальная схема полей мутации addTeamMember.

Рисунок выполнен автором.

4.2.2.5 **Register**

Эта мутация (Рис. 16) передает данные для добавления записи о новом пользователе веб-приложения в базу данных.

- Аргументы мутации:
  - username
    - Описание — поле содержит никнейм пользователя.
    - Тип значения поля — String.
      - Скалярный тип — да.
    - Поле обязательное — да.
  - email
    - Описание — поле содержит адрес почты.
    - Тип значения поля — String.
      - Скалярный тип — да.
    - Поле обязательное — да.
  - password
    - Описание — поле содержит пароль пользователя.
    - Тип значения поля — String.
      - Скалярный тип — да.
    - Поле обязательное — да.
- Возвращаемый мутацией тип — RegisterResponse. Этот тип позволяет определить успешность мутации и получить ошибки при необходимости. Тип содержит следующие поля:
  - ok
    - Описание — поле содержит код ответа об успешном статусе http-запроса.
    - Тип значения поля — Boolean.
      - Скалярный тип — да.
    - Поле обязательное — да.
  - user

- Описание — поле содержит выборку полей типа User.
- Тип значения поля — User.
  - Скалярный тип — нет.
- Поле обязательное — нет.

- errors
  - Описание — поле содержит массив элементов типа Error.
  - Тип значения поля — Error.
    - Скалярный тип — нет.
  - Поле обязательное — да.

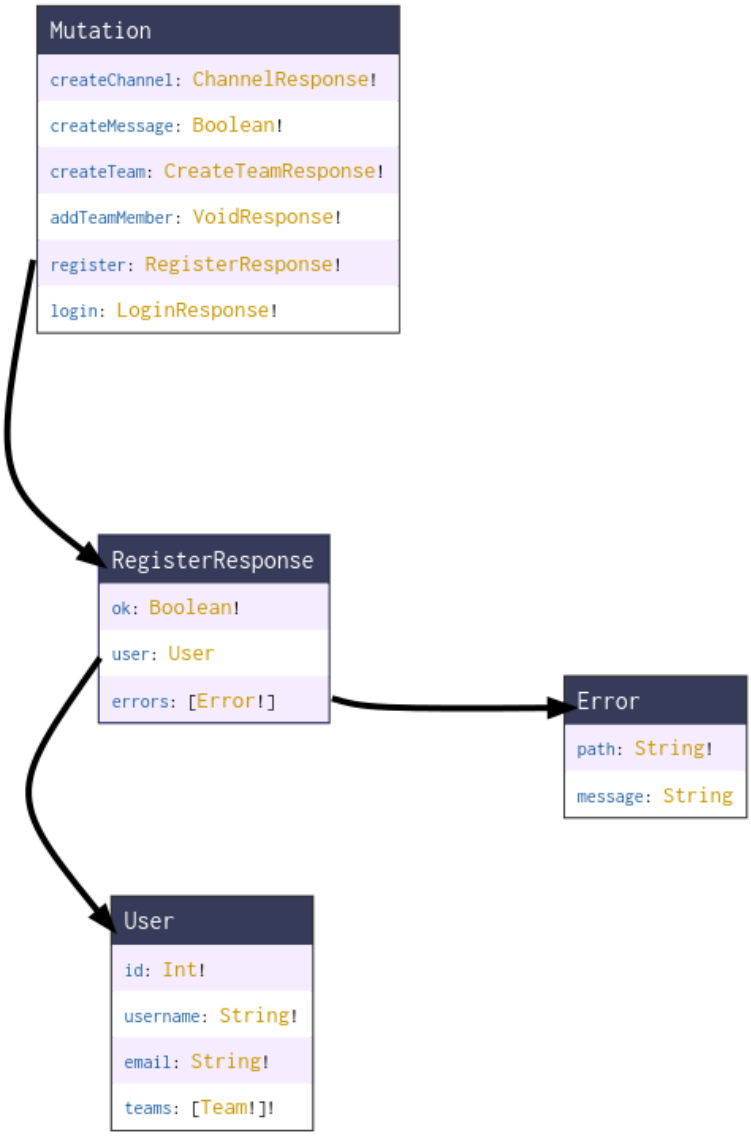

Рисунок 16 — концептуальная схема полей мутации register. Рисунок выполнен автором.

4.2.2.6 Login

Эта мутация (Рис. 17) передает данные для авторизации пользователя в веб-приложении в базу данных.

- Аргументы мутации:
    - email
        - Описание — поле содержит адрес почты.
        - Тип значения поля — String.
            - Скалярный тип — да.
        - Поле обязательное — да.
    - password
        - Описание — поле содержит пароль пользователя.
        - Тип значения поля — String.
            - Скалярный тип — да.
        - Поле обязательное — да.

- Возвращаемый мутацией тип — LoginResponse. Этот тип позволяет определить успешность мутации и получить ошибки при необходимости. Тип содержит следующие поля:
    - ok
        - Описание — поле содержит код ответа об успешном статусе http-запроса.
        - Тип значения поля — Boolean.
            - Скалярный тип — да.
        - Поле обязательное — да.
    - token
        - Описание — поле содержит авторизационный токен.
        - Тип значения поля — String.
            - Скалярный тип — да.
        - Поле обязательное — нет.

- refreshToken
  - Описание — поле содержит токен обмена [44].
  - Тип значения поля — String.
    - Скалярный тип — да.
  - Поле обязательное — нет.
- errors
  - Описание — поле содержит массив элементов типа Error.
  - Тип значения поля — Error.
    - Скалярный тип — нет.
  - Поле обязательное — да.

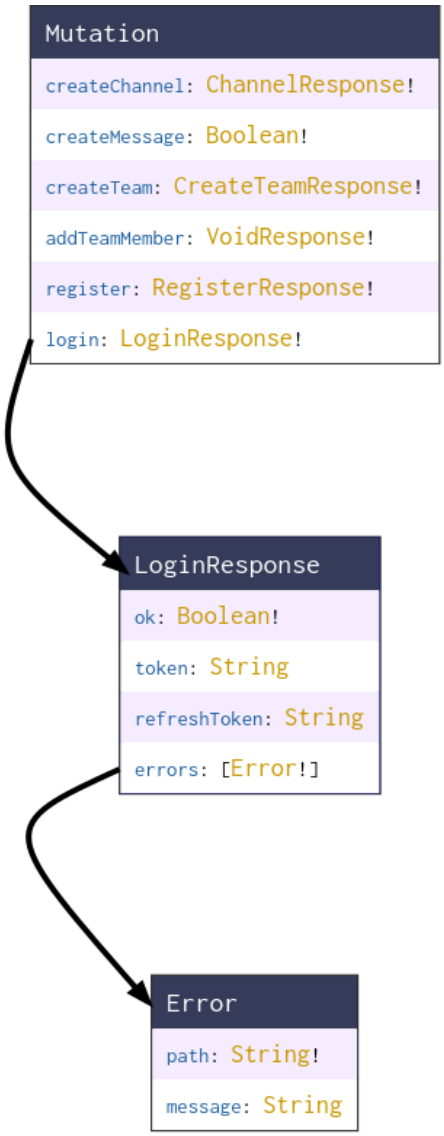

Рисунок 17 — концептуальная схема полей мутации login. Рисунок выполнен автором.

4.2.3 Проектирование запросов

4.2.3.1 Messages

Этот запрос (Рис. 18) используется для передачи данных об отправленных сообщениях в конкретном канале из базы данных.

- Аргументы запроса:
  - channelId
    - Описание — поле содержит идентификатор канала, в котором были отправлены сообщения.
    - Тип значения поля — Int.
      - Скалярный тип — да.
    - Поле обязательное — да.
- Возвращаемый запросом тип — непустой массив элементов типа Message.

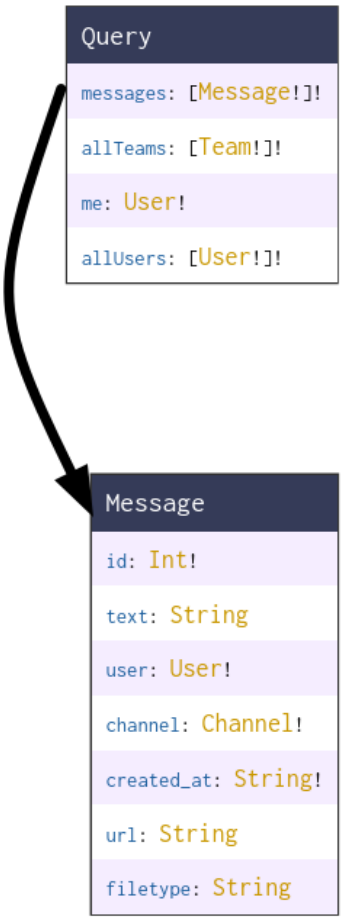

Рисунок 18 — концептуальная схема полей запроса messages. Рисунок выполнен автором.

4.2.3.2 AllTeams

Этот запрос (Рис. 19) используется для передачи данных об созданных пользователями командах из базы данных.

- Возвращаемый запросом тип — непустой массив элементов типа Team.

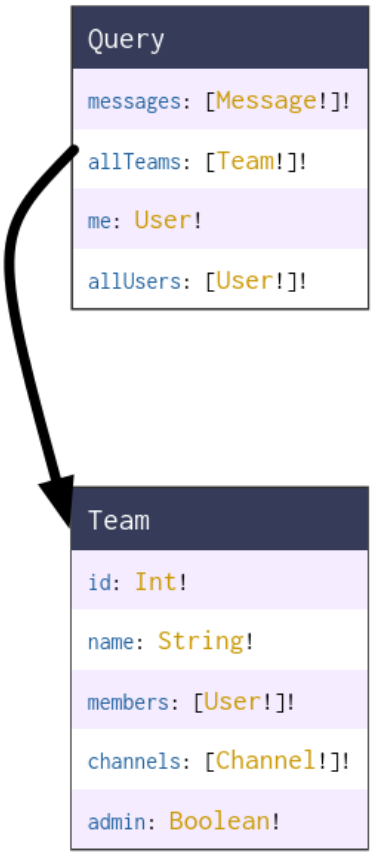

Рисунок 19 — концептуальная схема полей запроса allTeams. Рисунок выполнен автором.

4.2.3.3 Me

Этот запрос (Рис. 20) используется для передачи данных об активном пользователе базы данных.

- Возвращаемый запросом тип — ненулевая выборка полей типа User.

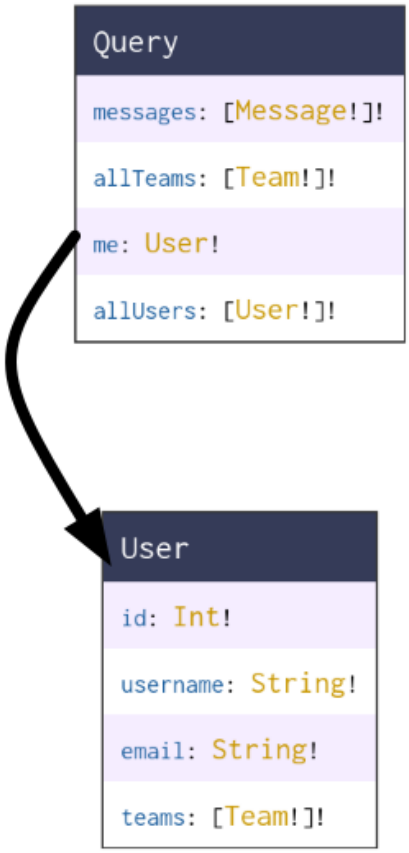

Рисунок 20 — концептуальная схема полей запроса me. Рисунок выполнен автором.

4.2.3.4 AllUsers

Этот запрос (Рис. 21) используется для передачи данных об зарегистрированных в системе пользователях из базы данных.

- Возвращаемый запросом тип — непустой массив элементов типа User.

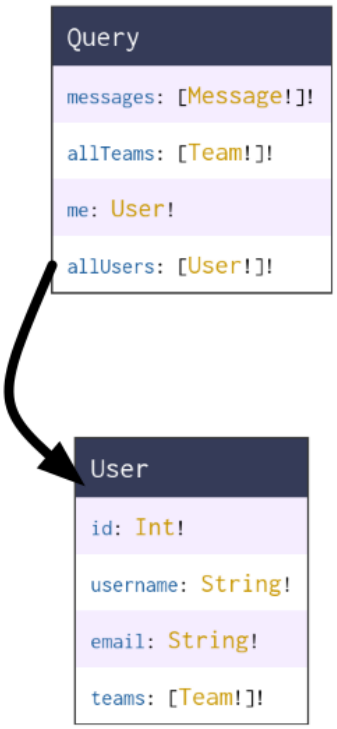

Рисунок 21 — концептуальная схема полей запроса allUsers. Рисунок выполнен автором.

4.2.4 Проектирование подписок

4.2.4.1 NewChannelMessage

Эта подписка (Рис. 22) используется для долговременной передачи данных об новых сообщениях из базы данных.

- Возвращаемый подпиской тип — ненулевая выборка полей типа Message.

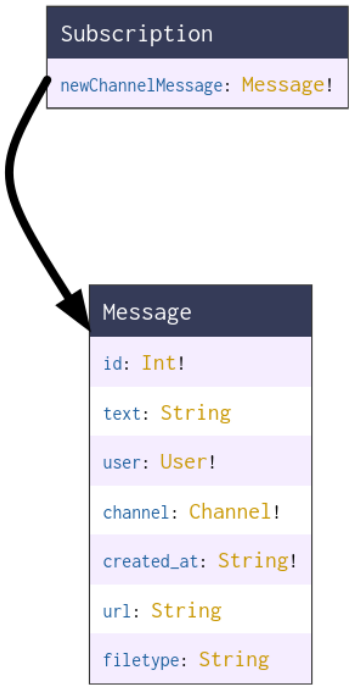

Рисунок 22 — концептуальная схема полей подписки newChannelMessage. Рисунок выполнен автором.

Глава 5. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ

Работа сервера определяется по методам, описанным в файле запуска «index.js», располагающемся в директории серверной части проекта веб-приложения. Файл «index.js» используется для решения двух задач. Первая — подключение фреймворка «Express.js», библиотек: «Sequelize», «Apollo-server-express» и других; подключение дополнительных программных приложений к линии обработки запросов «middleware». Вторая — запуск сервера.

В разработанном проекте задействованы следующие «middleware» для: конфигурирования CORS (Cross-origin resource sharing), аутентификации пользователей, трансфера файлов, подключения схемы GraphQL.

Рассмотрим механизм подключения дополнительных программных приложений к линии обработки запросов на примере промежуточного обработчика схемы GraphQL.

В соответствии с спроектированной ранее схемой типов, в субдиректорию «schema» серверной части проекта были добавлены файлы, содержащие реализацию типов и определенных для них мутаций, запросов на языке SDL. Пример такого файла представлен ниже (Рис. 23).

```
export default `

  type Channel {
    id: Int!
    name: String!
    public: Boolean!
    messages: [Message!]!
    users: [User!]!
  }

  type ChannelResponse {
    ok: Boolean!
    channel: Channel
    errors: [Error!]
  }

  type Mutation {
    createChannel(teamId: Int!, name: String!,
      public: Boolean=false): ChannelResponse!
  }
`;
```

Рисунок 23 — листинг кода файла channel.js.

В соответствии с спроектированной ранее базой данных были разработаны модели Sequelize, с разделением на отдельные JavaScript файлы в субдиректории «models». Пример модели представлен ниже (Рис. 24).

```
export default (sequelize, DataTypes) => {
  const Message = sequelize.define('message', {
    text: DataTypes.STRING,
    url: DataTypes.STRING,
    filetype: DataTypes.STRING,
  });

  Message.associate = (models) => {
    // 1:M
    Message.belongsTo(models.Channel, {
      foreignKey: {
        name: 'channelId',
        field: 'channel_id',
      },
    });
    Message.belongsTo(models.User, {
      foreignKey: {
        name: 'userId',
        field: 'user_id',
      },
    });
  };

  return Message;
};
```

Рисунок 24 — листинг кода файла «message.js». Sequelize-модель сущности сообщение.

Далее, требовалось объединить модели и подключить к базе данных PostgreSQL, а также экспортировать их, как переменную «models». (Рис. 25).

```
const sequelize = new Sequelize('slack2', 'postgres', 'postgres', {
  dialect: 'postgres',
  operatorsAliases: Sequelize.Op,
  define: {
    underscored: true,
  },
});

const models = {
  User: sequelize.import('./user'),
  Channel: sequelize.import('./channel'),
  Message: sequelize.import('./message'),
  Team: sequelize.import('./team'),
  Member: sequelize.import('./member'),
};

Object.keys(models).forEach((modelName) => {
  if ('associate' in models[modelName]) {
    models[modelName].associate(models);
  }
});

models.sequelize = sequelize;
models.Sequelize = Sequelize;

export default models;
```

Рисунок 25 — обработка Sequelize-моделей.

После разработки моделей Sequelize и схемы GraphQL были реализованы распознаватели — инструкции по выполнению операций GraphQL. Распознаватели также были разделены на отдельные JavaScript файлы в субдиректории «resolvers». В примере ниже (Рис. 26) с помощью методов объектов Sequelize реализуется добавление записей в базу данных при обработке мутации «createChannel».

```
Mutation: {
  createChannel: requiresAuth.createResolver(
  async (parent, args, { models, user }) => {
    try {
      const member = await models.Member.findOne(
        { where: { teamId: args.teamId, userId: user.id } },
        { raw: true },
      );
      if (!member.admin) {
        return {
          ok: false,
          errors: [
            {
              path: 'name',
              message: 'Только создатель команды может добавлять каналы',
            },
          ],
        };
      }
```

Рисунок 26 — фрагмент распознавателя для мутации «createChannel».

Далее, требовалось объединить типы GraphQL и распознаватели. Для этого использовались функции из библиотеки «merge-graphql-schemas» и была получена переменная «schema» (Рис. 27).

```
const typeDefs = mergeTypes(fileLoader(path.join(__dirname, './schema')));

const resolvers = mergeResolvers(fileLoader(path.join(__dirname, './resolvers')));

const schema = makeExecutableSchema({
  typeDefs,
  resolvers,
});
```

Рисунок 27 — обработка файлов реализованной схемы GraphQL.

В конечном итоге, в Express-метод «use» передавалась функция «graphqlExpress», импортированная из библиотеки «apollo-server-express», вместе с полученными ранее аргументами «models», «schema» и другими.

Глава 6. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ

В директории клиентской части веб-приложения был создан только один HTML-документ, используемый в качестве точки входа для обеспечения работы компонентов и всех разработанных экранов веб-приложения. Реализация и описание приводится далее.

Файл «index.js» является основным, в него импортируются файлы, в которых перечислены методы, определяющие перерисовку React-компонентов. Для маршрутизации веб-приложения в этом файле используются методы, определенные в библиотеке «react-dom». Основная логика маршрутизации заключается в проверке авторизации пользователя. При ее отсутствии пользователь перенаправляется на «Экран логина», иначе на «Главный экран», элементы которого позволяют перенаправление по каждому из маршрутов, описанных в файлах из субдиректории «routes». Пример такого файла показан ниже (Рис. 28)

```
return (
  <AppLayout>
    <Sidebar
      teams={teams.map(t => ({
        id: t.id,
        letter: t.name.charAt(0).toUpperCase(),
      }))}
      team={team}
      username={username}
    />
    {channel && <Header channelName={channel.name} />}
    {channel && <MessageContainer channelId={channel.id} />}
    {channel && (
      <SendMessage
      channelId={channel.id}
        placeholder={channel.name}
        onSubmit={async (text) => {
          await mutate({ variables: { text, channelId: channel.id } });
        }}
      />
    )}
  </AppLayout>
);
```

Рисунок 28 — фрагмент файла-маршрута «ViewTeam.js».

Устройство элементов экранов и их обработчиков построено с переиспользованием компонентов React, доступных для импорта из субдиректорий «components» и «containers». Ниже показан пример одного из реализованных компонентов React (Рис. 29).

```
import React from 'react';
import { Form, Input, Button, Modal } from 'semantic-ui-react';
import { withFormik } from 'formik';
import gql from 'graphql-tag';
import { compose, graphql } from 'react-apollo';
import findIndex from 'lodash/findIndex';

import { meQuery } from '../graphql/team';
const customStyles = {
    marginTop: '50px',
  };

const AddChannelModal = ({
  open,
  onClose,
  values,
  handleChange,
  handleBlur,
  handleSubmit,
  isSubmitting,
}) => (
  <Modal open={open} onClose={onClose}  style={customStyles}>
    <Modal.Header>Добавить канал</Modal.Header>
    <Modal.Content>
      <Form>
        <Form.Field>
          <Input
            value={values.name}
            onChange={handleChange}
            onBlur={handleBlur}
            name="name"
            fluid
            placeholder="Имя канала"
          />
        </Form.Field>
        <Form.Group widths="equal">
          <Button disabled={isSubmitting} fluid onClick={onClose}>
            Отмена
          </Button>
          <Button disabled={isSubmitting} onClick={handleSubmit} fluid>
            Добавить
          </Button>
        </Form.Group>
      </Form>
    </Modal.Content>
  </Modal>
);
```

Рисунок 29 — фрагмент файла-компонента «AddChannelModal.js».

6.1 Главный экран

Пользователи, зарегистрированные в системе и прошедшие аутентификацию, имеют доступ к главному экрану (Рис. 30) веб-приложения. Функционально он разделен на три части:

- Крайняя левая — содержит кнопки для переключения между каналами пользователя. Кнопка с символом «+» служит для открытия экрана создания команды, в котором пользователь вводит имя новой команды.
- Средняя — содержит некоторый список строк. Первая — наименование активной команды. Следующая строка содержит никнейм пользователя. После слова «Каналы» располагаются наименования-ссылки каналов текущей команды. Символ «⊕» служит для открытия модального окна добавления канала. Последняя строка «+ Добавить пользователя» служит для открытия модального окна добавления пользователя в команду.
- Крайняя правая — содержит: заголовок с наименованием активного канала; текстовые сообщения и файлы с временной отметкой об отправке и никнеймом отправителя; поле ввода сообщений и кнопку отправки файлов.

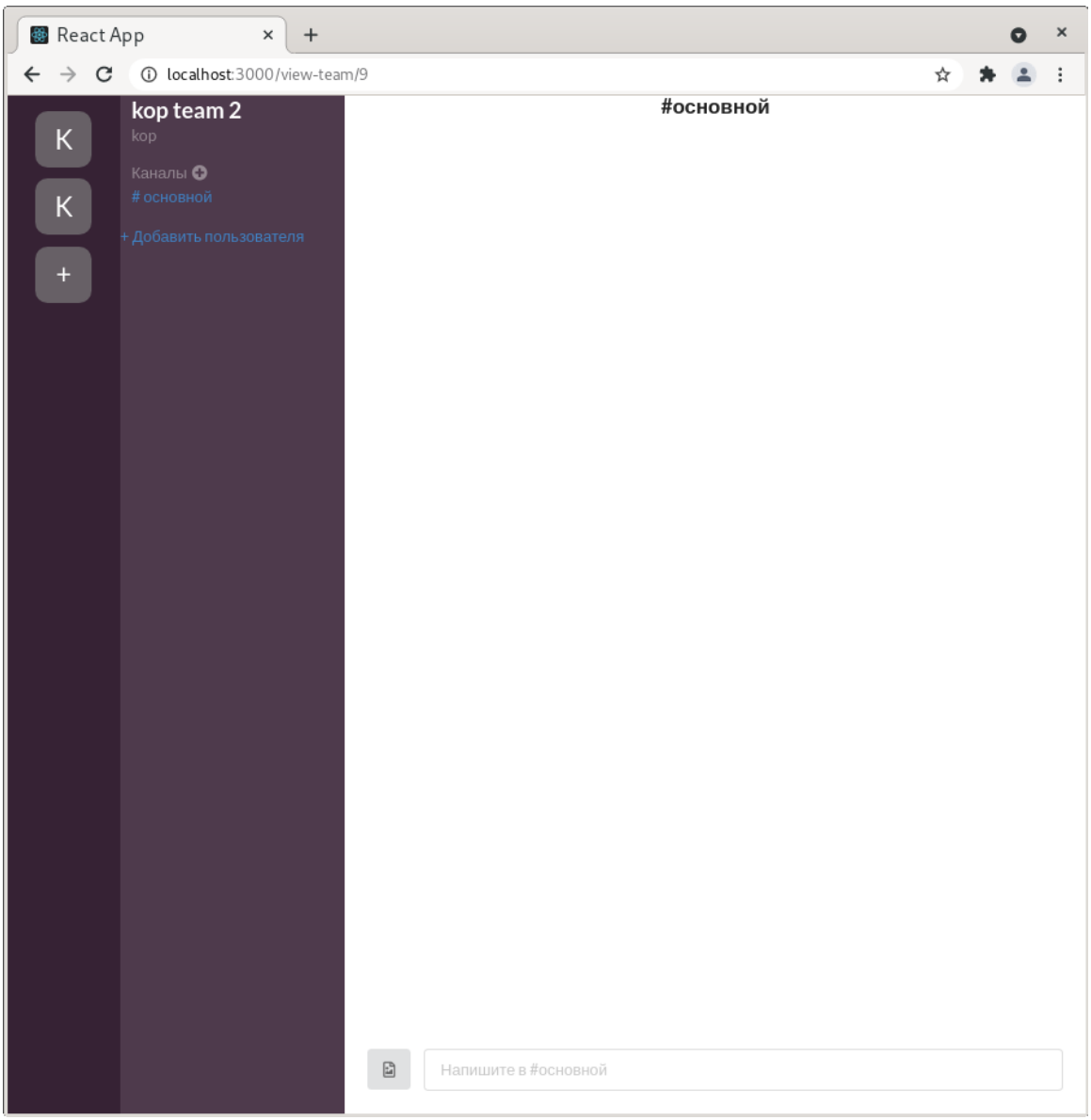

Рисунок 30 — вид главного экрана веб-приложения. Рисунок выполнен автором.

6.2 Экран регистрации

Экран регистрации (Рис. 31) служит для отправки серверу информации о новом пользователе. Он содержит три поля ввода текстовой информации, выдающих сообщения об ошибках ввода данных при необходимости (Рис. 32).

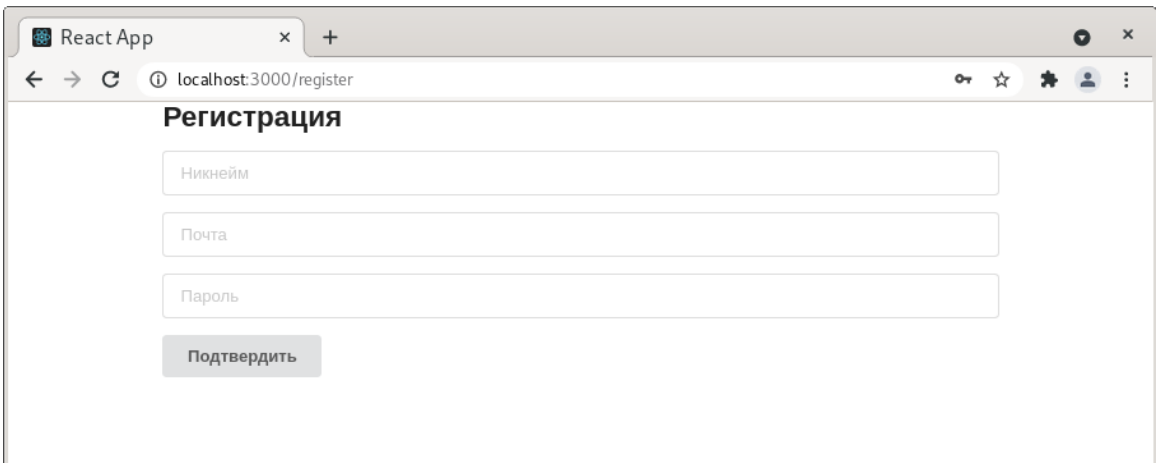

Рисунок 31 — вид экрана регистрации веб-приложения. Рисунок выполнен автором.

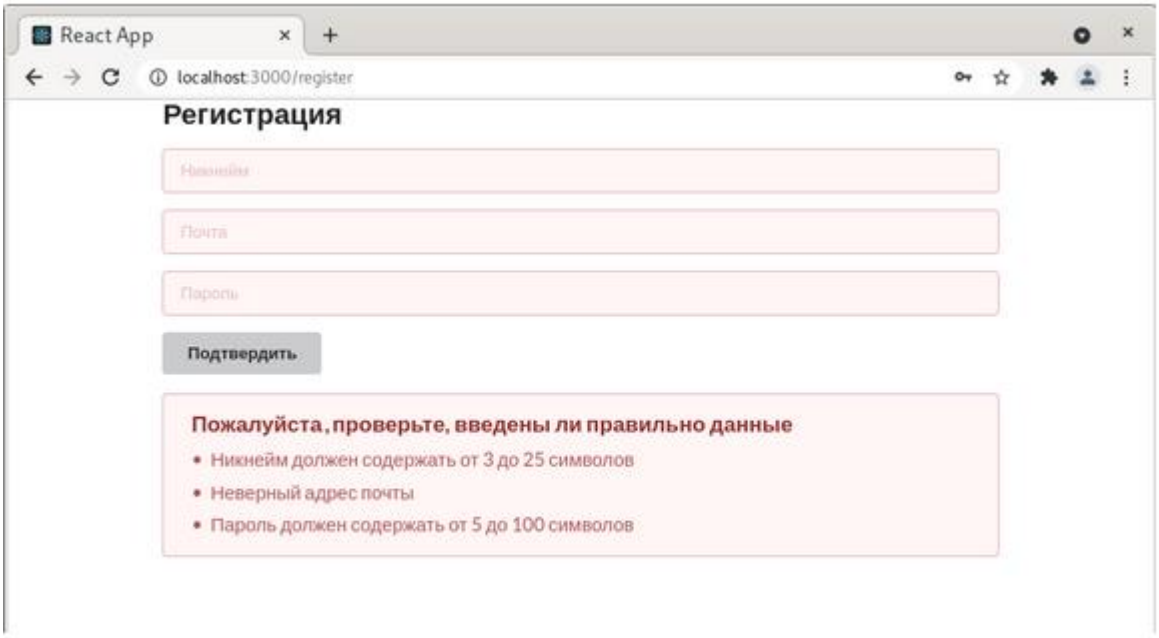

Рисунок 32 — вид выдачи информации об ошибках ввода на экране регистрации веб-приложения. Рисунок выполнен автором.

6.3 Экран логина

Экран логина (Рис. 33) служит для отправки серверу информации о данных существующего пользователя. Он содержит два поля ввода текстовой информации, выдающих сообщения об ошибках ввода данных при

необходимости (Рис. 34). Над полями ввода располагается ссылка для перехода на экран регистрации.

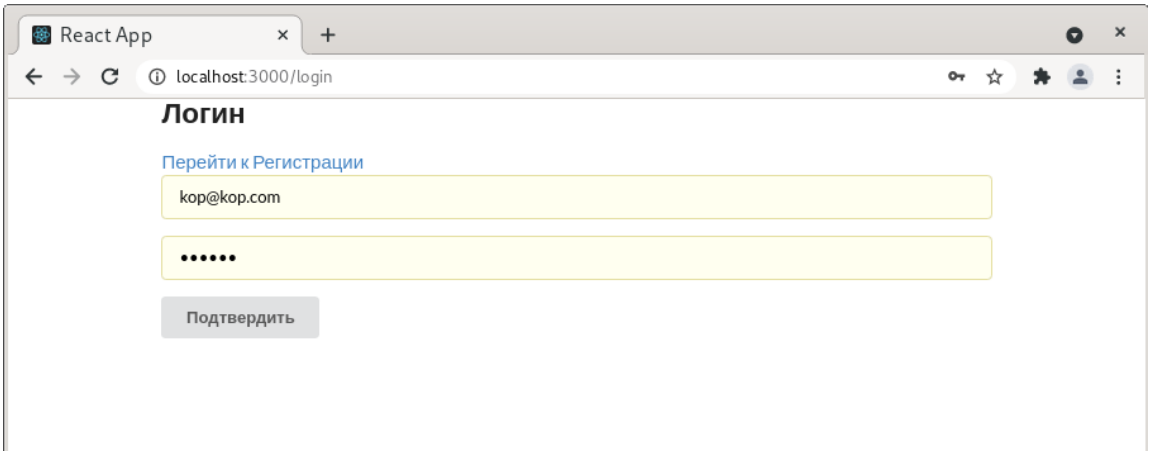

Рисунок 33 — вид экрана логина веб-приложения. Рисунок выполнен автором.

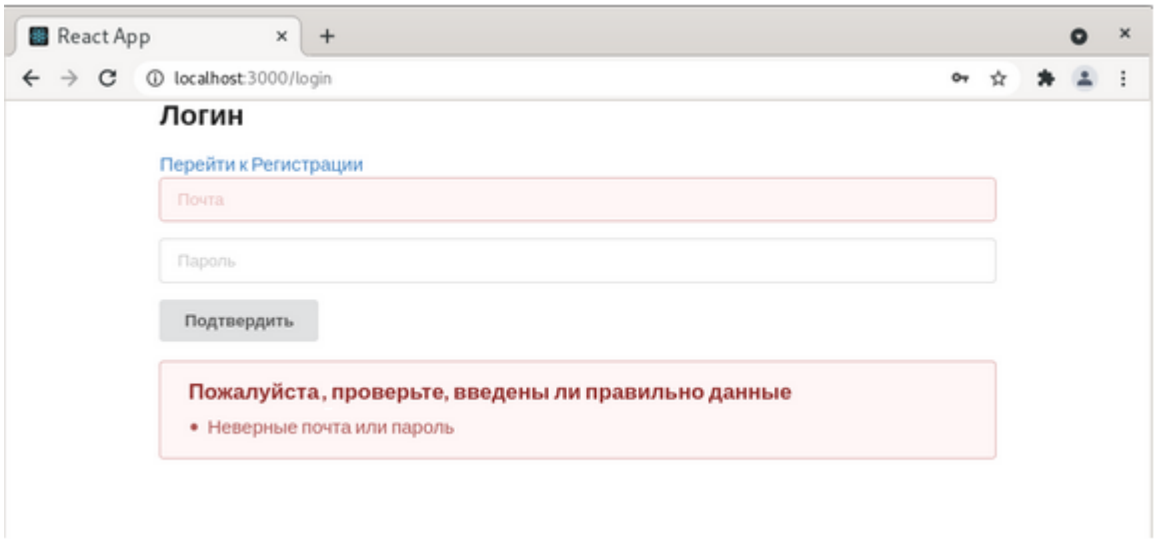

Рисунок 34 — вид выдачи информации об ошибках ввода на экране логина веб-приложения. Рисунок выполнен автором.

6.4 Экран создания команды

Экран создания команды (Рис. 35) служит для отправки серверу информации о данных новой команды. Он содержит одно поля ввода текстовой информации, выдающее сообщения об ошибках ввода данных при необходимости (Рис. 36).

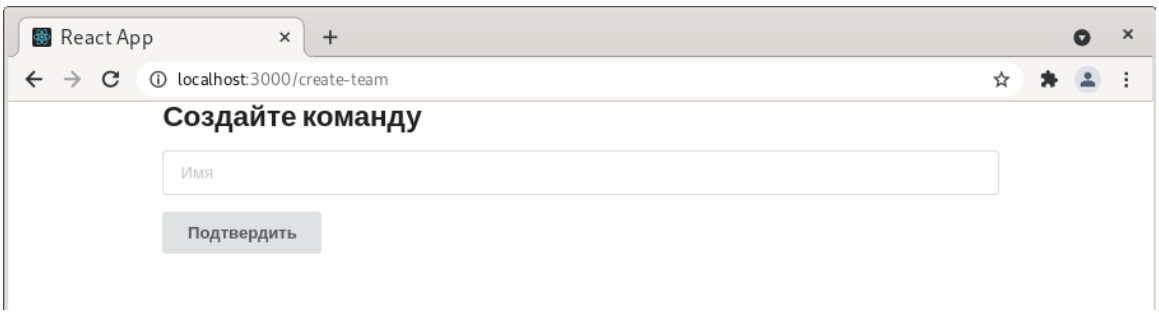

Рисунок 35 — вид экрана создания команды веб-приложения. Рисунок выполнен автором.

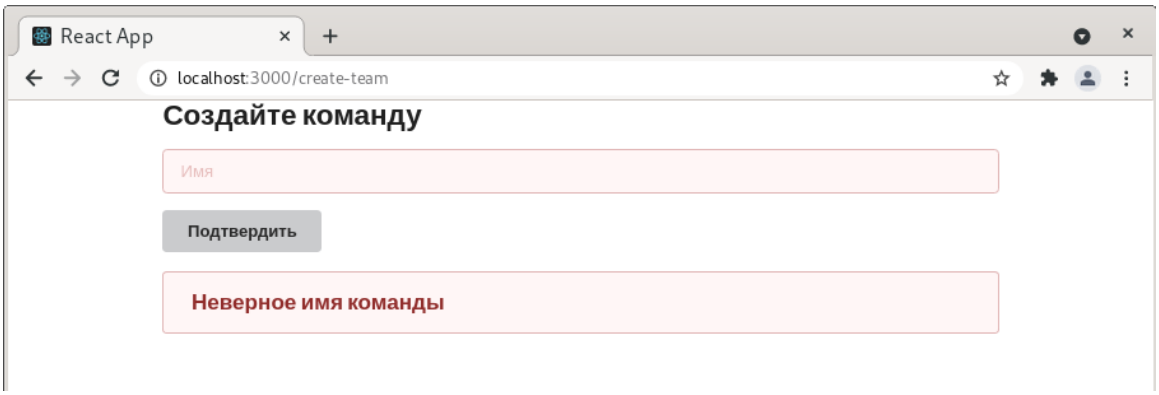

Рисунок 36 — вид выдачи информации об ошибках ввода на экране создания команды веб-приложения. Рисунок выполнен автором.

6.5 Модальное окно добавления канала

Под модальным подразумевается, что пока открыто данное окно, тело главного экрана не будет обрабатывать касания и клики пользователя. Это окно (Рис. 37) служит для отправки серверу информации о данных нового канала. Оно содержит одно поля ввода текстовой информации.

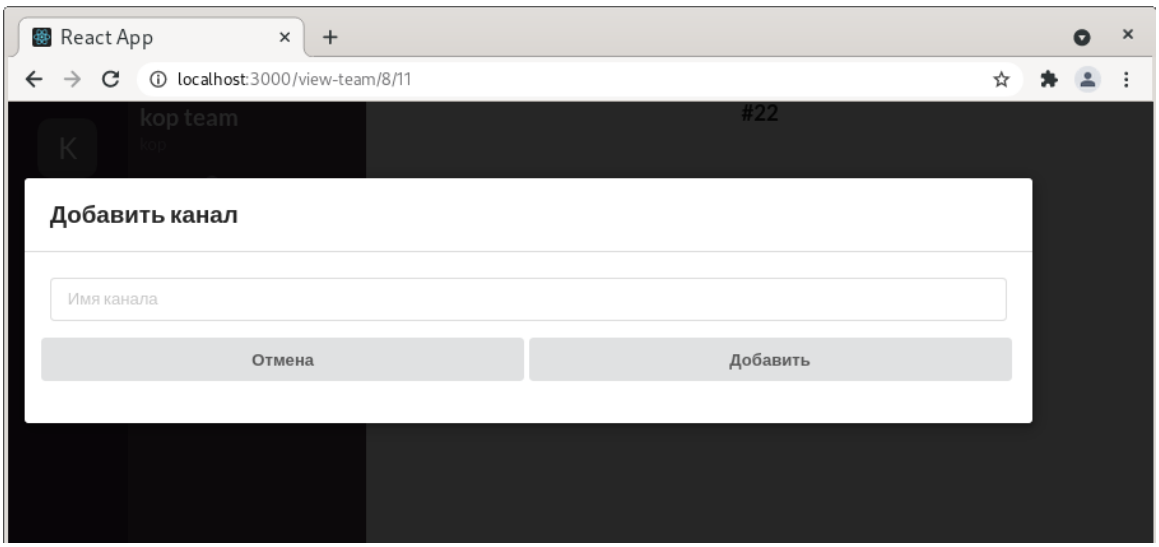

Рисунок 37 — вид модального окна добавления канала веб-приложения. Рисунок выполнен автором.

6.6 Модальное окно добавления пользователя

Это окно (Рис. 38) служит для отправки серверу информации о данных нового пользователя в команде. Оно содержит одно поля ввода текстовой информации, выдающее сообщения об ошибках ввода данных при необходимости (Рис. 39).

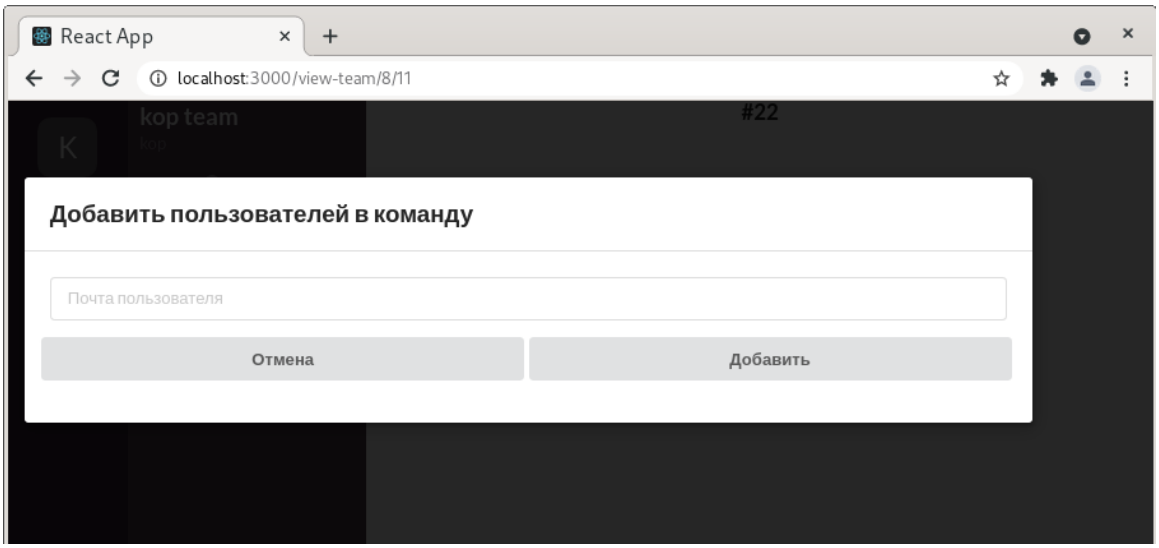

Рисунок 38 — вид модального окна добавления пользователя в команду веб-приложения. Рисунок выполнен автором.

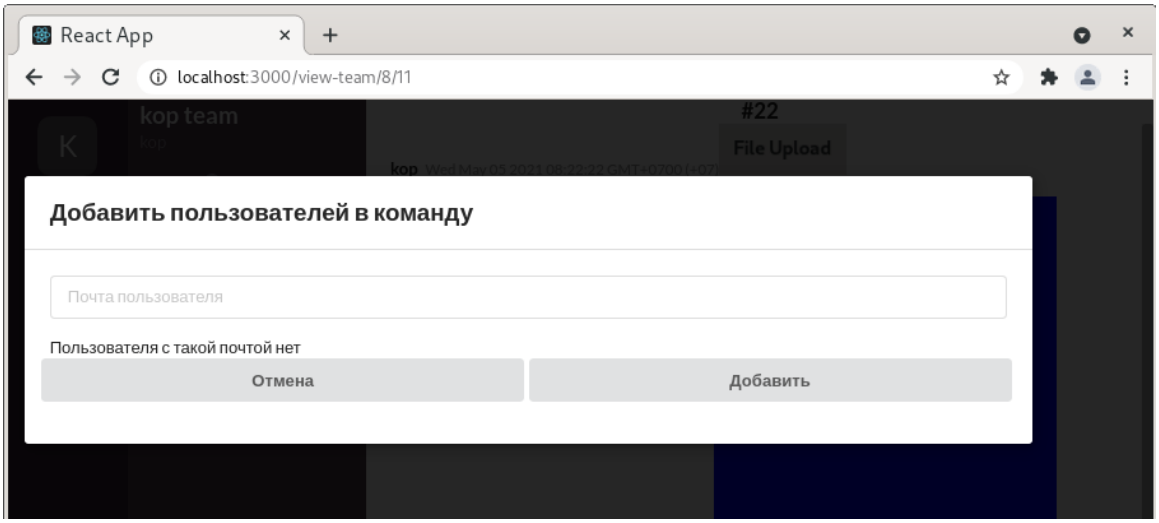

Рисунок 39 — вид выдачи информации об ошибках ввода в модальном окне добавления пользователя в команду. Рисунок выполнен автором.

6.7 Отправка файлов

Для отправки файлов требуется нажать кнопку отправки файлов, расположенную рядом с полем ввода сообщений. После нажатия появится системное диалоговое окно выбора файла для отправки (Рис. 40). После этого отправленный файл будет отображен в блоке с остальными сообщениями (Рис. 41).

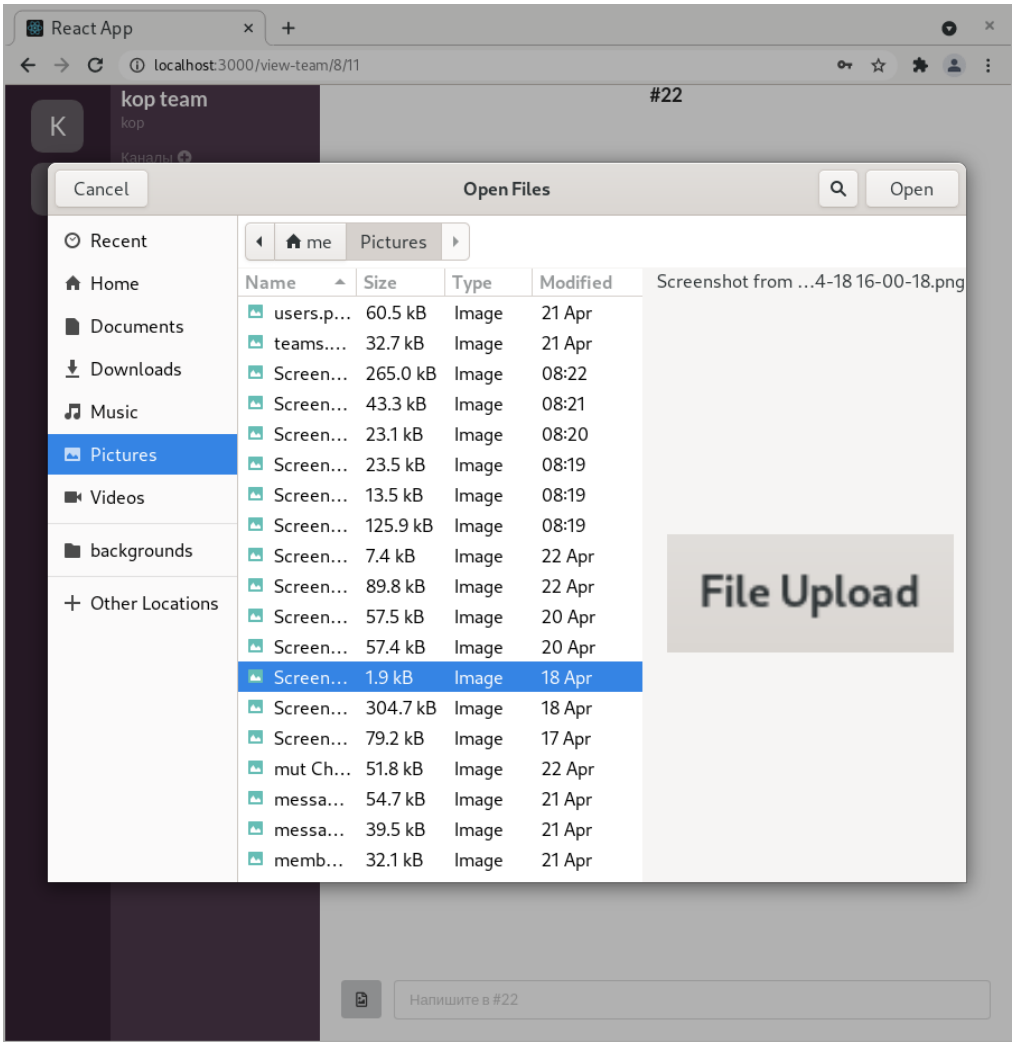

Рисунок 40 — вид системного диалогового окна для выбора файла.

Рисунок выполнен автором.

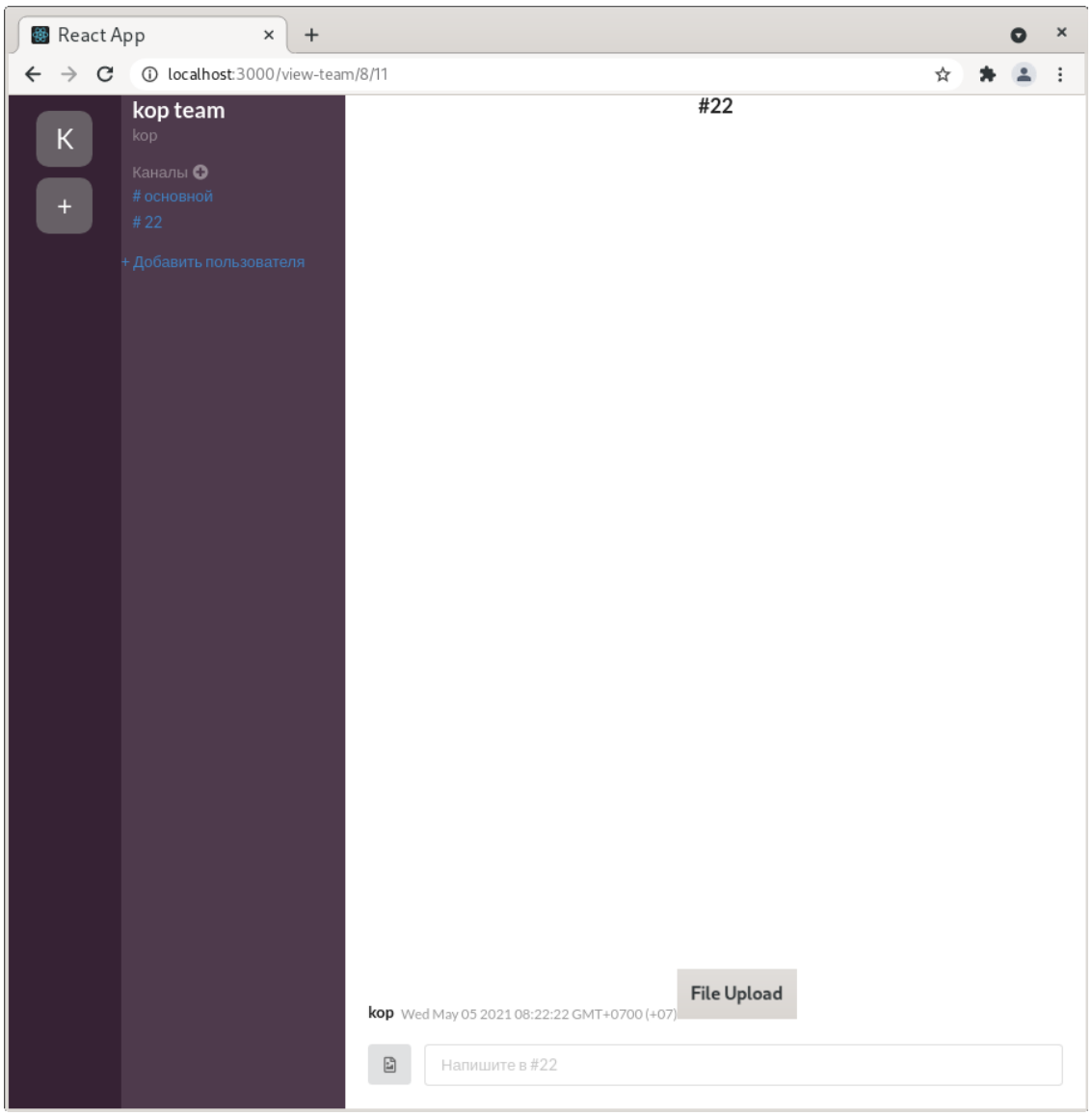

Рисунок 41 — вид главного экрана веб-приложения после отправки файла. Рисунок выполнен автором.

Заключение

В ходе анализа существующих решений в области приложений для коммуникации, используемых медицинскими специалистами, были выявлены следующие тенденции:

- Две из трех популярных аналогичных систем (WhatsApp, Slack) распространяются без открытого исходного кода и возможности размещения в локальных сетях. Одна из систем (TeamSpirit IM) имеет возможность размещения в локальных сетях, и компания-разработчик предоставляет исходный код программы в распоряжение клиентам, но не содержит его в открытом доступе.
- Веб-версии систем WhatsApp, Slack обеспечивают функции отправки сообщений и видеосвязи без необходимости установки программного обеспечения. Веб-версия системы TeamSpirit IM требует установки дополнительных программных пакетов.
- Все рассмотренные системы в процессе своего развития были дополнены программными модулями, обеспечивающими функцию видеозвонков.

В ходе анализа языков программирования и веб-фреймворков была выбрана программная экосистема JavaScript по таким признакам, как: простой синтаксис для работы с асинхронными функциями, возможность написания серверной и клиентской частей приложения на одном языке программирования, наличие системы распространения и менеджмента программных пакетов NPM, предоставляющей доступ к многим инструментам.

Для связи клиентской и серверной частей приложения в ходе анализа веб-API была выбрана JavaScript-реализация языка запросов GraphQL — программная платформа Apollo. Запросы GraphQL являются более производительными и эффективными, в сравнении с REST-запросами и позволяют получать данные из многих сервисов и ресурсов, что позволит с

малыми временными затратами интегрировать веб-приложение в МИС медицинских организаций.

Была спроектирована база данных и на ее основе схема GraphQL.

Был реализован прототип веб-приложения: имеющий возможность размещения в локальных сетях; не требующий установки дополнительного ПО на устройства пользователей; доступный и с мобильных устройств, и с ПК.

Прототип веб-приложения предназначен для использования: сестринским составом и главами паллиативных, диспансерных отделений для решения организационных и административных вопросов; врачами лучевой диагностики, специалистами-онкологами разных уровней квалификации для обмена опытом, консультаций. Все коммуникационные процессы (отправка файлов, обмен сообщениями) происходят между объединенными в команды пользователями в определенных каналах. Пользовательские каналы представляют отдельные рабочие пространства, обеспечивают структуризацию информации.

Разработанный прототип предполагается размещать в защищенной сети, в которой расположена медицинская информационная система Новосибирской области, и встраивать в графический интерфейс МИС, а также использовать построенную схему для организации прикладного программного интерфейса взаимодействия.

Направлениями развития разрабатываемого веб-приложения являются:

- добавление программных модулей, обеспечивающих функцию видеозвонков;
- развитие API для интеграции с региональным порталом записи на прием к врачу для использования в режиме врач-пациент.

Список использованных источников

1. Федеральный закон от 1 мая 2019 г. № 90-ФЗ «О внесении изменений в Федеральный закон “О связи” и Федеральный закон “Об информации, информационных технологиях и о защите информации”», URL: http://www.consultant.ru/

2. Владзимирский А.В. Систематический обзор применения мессенджеров «WhatsApp®» и «Viber®» в клинической медицине [Электронный ресурс] / А.В. Владзимирский — Журнал телемедицины и электронного здравоохранения. 2017. №1 – Режим доступа: https://cyberleninka.ru/article/n/sistematicheskiy-obzor-primeneniya-messendzherov-whatsapp-i-viber-v-klinicheskoy-meditsine (дата обращения: 08.05.2021).

3. Горюнова М.П. Архитектурные стили в разработке web-приложений и область их применения [Электронный ресурс] / М.П. Горюнова — Проблемы Науки. 2017. №12 (94). – Режим доступа: https://cyberleninka.ru/article/n/arhitekturnye-stili-v-razrabotke-web-prilozheniy-i-oblast-ih-primeneniya (дата обращения: 09.05.2021).

4. Колисниченко Д.Н. PHP и MySQL. Разработка Web-приложений. 6-е изд./ Д.Н. Колисниченко. – БХВ-Петербург, 2017. – 640 с.

5. «Россию могут отключить от мирового интернета»: Клименко в интервью RT. [Электронный ресурс] – Сетевое издание rt.com – Режим доступа: https://russian.rt.com/russia/article/346157-intervyu-klimenko-internet // (дата обращения 08.05.2021).

6. Методические рекомендации по обеспечению функциональных возможностей централизованной системы (подсистемы) «Организация оказания медицинской помощи больным онкологическими заболеваниями государственной информационной системы в сфере здравоохранения субъекта Российской Федерации [Электронный ресурс] – ФГБУ ЦНИИОИЗ – Режим доступа: https://portal.egisz.rosminzdrav.ru/files/%D0%9C%D0%A0_%D0%9E%D0%BD%

D0%BA%D0%BE_%D0%B4%D0%BB%D1%8F%20%D0%BF%D1%83%D0%B1%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8%20(1).pdf (дата обращения 08.05.2021).

7. Методические рекомендации Минздрава России к построению Единых государственных информационных систем в сфере здравоохранения в субъектах Российской Федерации [Электронный ресурс] – ФГБУ ЦНИИОИЗ – Режим доступа: https://portal.egisz.rosminzdrav.ru/files/%D0%9C%D0%A0%20%D0%A2%D0%B5%D0%BB%D0%B5%D0%BC%D0%B5%D0%B4%D0%B8%D1%86%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B5%20%D0%BA%D0%BE%D0%BD%D1%81%D1%83%D0%BB%D1%8C%D1%82%D0%B0%D1%86%D0%B8%D0%B8_v002_2021_03_22.pdf (дата обращения 08.05.2021).

8. Team Spirit Instant Messenger [Электронный ресурс] – VideoMost.com – Режим доступа: https://www.videomost.com/teamspirit-korporativnyj-messendzher (дата обращения 08.05.2021).

9. Тарифы VideoMost [Электронный ресурс] – VideoMost.com – Режим доступа: https://www.videomost.com/tarifs (дата обращения 08.05.2021).

10. Rational Software Architect Standard Edition. Документация [Электронный ресурс] – IBM Corporation – Режим доступа: https://www.ibm.com/docs/ru/rsas/7.5.0?topic=standards-soap (дата обращения 08.05.2021).

11. Сравнение архитектурных стилей API: SOAP vs REST vs GraphQL vs RPC [Электронный ресурс] – nuancesprog.ru – Режим доступа: https://nuancesprog.ru/p/11310/ (дата обращения 08.05.2021).

12. Выбираем лучший бэкенд-фреймворк 2021 года [Электронный ресурс] – Блог компании RUVDS.com – Режим доступа: https://habr.com/ru/company/ruvds/blog/519478/ (дата обращения 08.05.2021).

13. Django ORM для начинающих [Электронный ресурс] – Медиа habr.com – Режим доступа: https://habr.com/ru/post/503526/ (дата обращения 08.05.2021).

14. Django или Laravel [Электронный ресурс] – Блог IT-компании Wezom – Режим доступа: https://wezom.com.ua/blog/django-ili-laravel (дата обращения 08.05.2021).

15. Асинхронный PHP [Электронный ресурс] – Медиа habr.com – Режим доступа: https://habr.com/ru/post/451916/ (дата обращения 08.05.2021).

16. Запускаем асинхронные запросы в NodeJS параллельно [Электронный ресурс] – Блог the-evening-code.com – Режим доступа: https://the-evening-code.com/posts/handling-asynchronous-operations-in-parallel-in-nodejs (дата обращения 08.05.2021).

17. Что лучше выбрать в 2020 году — React или Vue? [Электронный ресурс] – Блог компании RUVDS.com – Режим доступа: https://habr.com/ru/company/ruvds/blog/470413/ (дата обращения 08.05.2021).

18. 14 вещей, которые я хотел бы знать перед началом работы с MongoDB [Электронный ресурс] – Блог компании OTUS – Режим доступа: https://habr.com/ru/company/otus/blog/520412/ (дата обращения 08.05.2021).

19. Sequelize [Электронный ресурс] – Сайт о программировании METANIT.COM – Режим доступа: https://metanit.com/web/nodejs/9.1.php (дата обращения 08.05.2021).

20. Руководство по GraphQL [Электронный ресурс] – Блог компании reg.ru – Режим доступа: https://www.reg.ru/support/vps-servery/oblachnie-serveri-vps/ustanovka-programmnogo-obespechenija/graphql (дата обращения 08.05.2021).

21. Express/Node introduction [Электронный ресурс] – Сайт веб-документации MDN – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs/Introduction (дата обращения 08.05.2021).

22. ReactDOM [Электронный ресурс] – Документация React – Режим доступа: https://ru.reactjs.org/docs/react-dom.html (дата обращения 08.05.2021).

23. Знакомство с Styled components [Электронный ресурс] – Блог компании Productivity Inside – Режим доступа:

https://habr.com/ru/company/productivity_inside/blog/321804/ (дата обращения 08.05.2021).

24. Как определить связи между таблицами в базе данных Access [Электронный ресурс] – Документация Microsoft – Режим доступа: https://docs.microsoft.com/ru-ru/office/troubleshoot/access/define-table-relationships (дата обращения 08.05.2021).

25. 8 advantages of using open source in the enterprise [Электронный ресурс] / Lee Congdon – Red Hat, Inc. – Режим доступа: https://enterprisersproject.com/article/2015/1/top-advantages-open-source-offers-over-proprietary-solutions (дата обращения 08.05.2021).

26. Support status of the significant versions of AngularJS. [Электронный ресурс] – Документация AngularJS – Режим доступа: https://docs.angularjs.org/misc/version-support-status (дата обращения 08.05.2021).

27. What is API: Definition, Types, Specifications, Documentation [Электронный ресурс] – AltexSoft – Режим доступа: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/?utm_source=MediumCom&utm_medium=referral&utm_campaign=shared (дата обращения 08.05.2021).

28. Introduction to REST API - RESTful Web Services [Электронный ресурс] – springboottutorial.com – Режим доступа: https://www.springboottutorial.com/introduction-to-rest-api (дата обращения 08.05.2021).

29. Introduction to GraphQL [Электронный ресурс] – GraphQL Foundation – Режим доступа: https://graphql.org/learn/ (дата обращения 08.05.2021).

30. How fast GraphQL is compared to REST APIs [Электронный ресурс] – Haaga-Helia University – Режим доступа: https://www.theseus.fi/bitstream/handle/10024/340318/Thesis_Camille_Oggier.pdf?sequence=2&isAllowed=y (дата обращения 08.05.2021).

31. Performance analysis of Web Services. Comparison between RESTful & GraphQL web services. [Электронный ресурс] – diva-portal.org – Режим доступа: https://www.diva-portal.org/smash/get/diva2:1107850/FULLTEXT01.pdf (дата обращения 08.05.2021).

32. Introduction to MongoDB [Электронный ресурс] – Документация mongoDB – Режим доступа: https://docs.mongodb.com/manual/introduction/ (дата обращения 08.05.2021).

33. What is the maximum size for a row, a table, and a database? [Электронный ресурс] – Документация PostgreSQL – Режим доступа: https://wiki.postgresql.org/wiki/FAQ#What_is_the_maximum_size_for_a_row.2C_a_table.2C_and_a_database.3F (дата обращения 08.05.2021).

34. Row Size Limits [Электронный ресурс] – Документация MySQL – Режим доступа: https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.7/en/column-count-limit.html#row-size-limits (дата обращения 08.05.2021).

35. What is Babel? [Электронный ресурс] – Документация Babel – Режим доступа: https://babeljs.io/docs/en/ (дата обращения 08.05.2021).

36. Body-parser [Электронный ресурс] – Описание пакета – Режим доступа: https://www.npmjs.com/package/body-parser (дата обращения 08.05.2021).

37. Sequelize API reference [Электронный ресурс] – Документация Sequelize – Режим доступа: https://sequelize.org/master/ (дата обращения 08.05.2021).

38. Introducing JSX [Электронный ресурс] – Документация React – Режим доступа: https://reactjs.org/docs/introducing-jsx.html (дата обращения 08.05.2021).

39. Introduction to Apollo Client [Электронный ресурс] – Документация Apollo API – Режим доступа: https://www.apollographql.com/docs/react/ (дата обращения 08.05.2021).

40. Ten minute introduction to MobX and React [Электронный ресурс] – Документация MobX – Режим доступа: https://mobx.js.org/getting-started (дата обращения 08.05.2021).

41. The official Semantic-UI-React integration. [Электронный ресурс] – Документация Semantic-UI – Режим доступа: https://react.semantic-ui.com/ (дата обращения 08.05.2021).

42. Jwt-decode [Электронный ресурс] – Описание пакета – Режим доступа: https://www.npmjs.com/package/jwt-decode (дата обращения 08.05.2021).

43. Ozdalga E. The Smartphone in Medicine: A Review of Current and Potential Use among Physicians and Students [Электронный ресурс] / E. Ozdalga — Journal of Medical Internet Research. 2012. 14(5):e128 – Режим доступа: https://www.researchgate.net/profile/Errol-Ozdalga/publication/231223676_The_Smartphone_in_Medicine_A_Review_of_Current_and_Potential_Use_Among_Physicians_and_Students/links/53f46ed70cf2fceacc6e73fa/The-Smartphone-in-Medicine-A-Review-of-Current-and-Potential-Use-Among-Physicians-and-Students.pdf (дата обращения: 08.05.2021).

44. OAuth 2.0 Token Exchange [Электронный ресурс] – Спецификация Security Token Service – Режим доступа: https://datatracker.ietf.org/doc/html/rfc8693 (дата обращения 08.05.2021).

Приложение

Исходные коды разработанного прототипа веб-приложения представлены на цифровом носителе (CD-диск прилагается).