МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**Разработка веб-приложения «NetMate»**

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

ЮУрГУ – 02.03.02.2024.308-007.КР

|  |  |
| --- | --- |
| Нормоконтролер,ст. преподаватель кафедры СП  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.С. Силкина  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г. | Научный руководитель:  ст. преподаватель кафедры СП  \_\_\_\_\_\_\_\_\_\_ Н.С. Силкина  Автор работы:  студент группы КЭ-301  \_\_\_\_\_\_\_\_\_\_ Д.А. Иванов  Работа защищена  с оценкой: \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Челябинск 2024МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_Л.Б. Соколинский

10.02.2024

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы КЭ-301 Иванову Дмитрию Александровичу,

обучающемуся по направлению 02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы**Разработка веб-приложения «NetMate»
2. **Срок сдачи студентом законченной работы:** 31.05.2024 г.
3. **Исходные данные к работе**
4. React – A JavaScript library for building user interfaces. [Электронный ресурс] URL: <https://legacy.reactjs.org> (дата обращения 19.02.2024 г.)
5. React Documentations. [Электронный ресурс] URL: <https://react.dev> (дата обращения 19.02.2024 г.)
6. Руководство по CSS. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Web/CSS/Reference (дата обращения: 19.02.2024 г.).
7. Руководство JavaScript. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 19.02.2024 г.).
8. **Перечень подлежащих разработке вопросов**
9. Провести анализ аналогичных приложений;
10. Спроектировать веб-приложение;
11. Реализовать прототип веб-приложения;
12. Провести тестирование реализованного прототипа.
13. **Дата выдачи задания:** 9 февраля 2024 г.

Научный руководитель

ст. преподаватель кафедры СП, Н.С. Силкина

Задание принял к исполнению Д.А. Иванов

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc169739665)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc169739666)

[1.1. Обзор аналогичных проектов 7](#_Toc169739667)

[1.2. Анализ существующих решений для реализации проекта 10](#_Toc169739668)

[2. ТРЕБОВАНИЯ К СИСТЕМЕ 13](#_Toc169739669)

[3. ПРОЕКТИРОВАНИЕ 14](#_Toc169739670)

[3.1. Диаграмма вариантов использования 14](#_Toc169739671)

[3.2. Проектирование интерфейса 15](#_Toc169739672)

[3.3. Диаграмма компонентов 15](#_Toc169739673)

[3.4. Проектирование базы данных 16](#_Toc169739674)

[4. РЕАЛИЗАЦИЯ 18](#_Toc169739675)

[4.1. Создание проекта 18](#_Toc169739676)

[4.2. Страница авторизации 19](#_Toc169739677)

[4.3. База данных 21](#_Toc169739678)

[4.4. Главная страница 21](#_Toc169739679)

[4.5. Страница трансляции 25](#_Toc169739680)

[4.6. Страница настроек 28](#_Toc169739681)

[5. ТЕСТИРОВАНИЕ 30](#_Toc169739682)

[ЗАКЛЮЧЕНИЕ 32](#_Toc169739683)

[ЛИТЕРАТУРА 33](#_Toc169739684)

[ПРИЛОЖЕНИЯ 36](#_Toc169739685)

[Приложение А. Спецификация вариантов использоания 36](#_Toc169739686)

[Приложение Б. Листинги компонентов 39](#_Toc169739687)

ВВЕДЕНИЕ

**Актуальность**

В современном мире интернет играет ключевую роль в повседневной жизни людей, обеспечивая доступ к информации, развлечениям и социальным взаимодействиям. Одной из наиболее быстрорастущих сфер является индустрия видеотрансляций. Стриминг применяется в разных сферах, позволяя проводить некоторые мероприятия в удаленном формате.

В рамках данной работы была выбрана сфера развлекательных трансляций. Подобные платформы, например Twitch, привлекают миллионы пользователей, предоставляя им возможность как смотреть, так и создавать собственный контент.

Актуальность разработки нового веб-приложения для трансляции видео и просмотра трансляций обусловлена ростом популярности стриминга, широкими возможностями для монетизации контента, большим охватом аудитории за счет разнообразия контента. [25]

**Цель и задачи работы**

Целью проекта является разработка веб-приложения для потоковой передачи видео и общения пользователей.

Для реализации поставленной цели были определены следующие задачи:

1. Провести анализ предметной области;
2. Разработать требования к веб-положению;
3. Спроектировать веб-приложение;
4. Реализовать веб-приложение для потоковой передачи видео и общения пользователей в групповом чате;
5. Провести тестирование веб-приложения.

**Структура и содержание работы**

Работа состоит из введения, пяти глав, заключения и списка литера-туры. Объем работы составляет 43 страниц, объем списка литературы – 29 источников.

В первой главе описывается анализ предметной области – обзор аналогичных проектов, а также используемые инструменты для разработки.

Вторая глава посвящена функциональным и нефункциональным требованиям.

В третьей главе рассмотрен процесс проектирования системы. Приведена диаграмма вариантов использования и диаграмма развертывания. А также спроектированы макеты страниц веб-приложения и база данных.

В четвертой главе приведена реализация веб-приложения.

В пятой главе представлены результаты тестирования разработанного веб-приложения

В приложениях содержится спецификация вариантов использования разрабатываемого программного обеспечения, которая описывает процесс взаимодействия пользователя с системой.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Обзор аналогичных проектов

В течение последнего десятилетия все популярнее становятся сервисы для публикации и просмотра медиаконтента. В том числе и для потокового вещания или стриминга. Наиболее часто стриминг предполагает прямое общение, трансляции видеоигр и спортивных состязаний. Именно его стали называть «медиа будущего». [12]

**Twitch**

Twitch – видеостриминговый сервис, специализирующийся на тематике компьютерных игр. Но, помимо этого, Twitch проводит трансляции и видео на другие тематик, например, музыкальные выступления, искусство, турниры по различным дисциплинам. Видео на платформе Twitch можно просматривать как в реальном времени, так и по запросу. Twitch принадлежит Twitch Interactive – дочерней компании Amazon. Сервис был создан в 2011 году. [8]

На рисунке 1 представлен скриншот страницы трансляции с площадки Twitch.

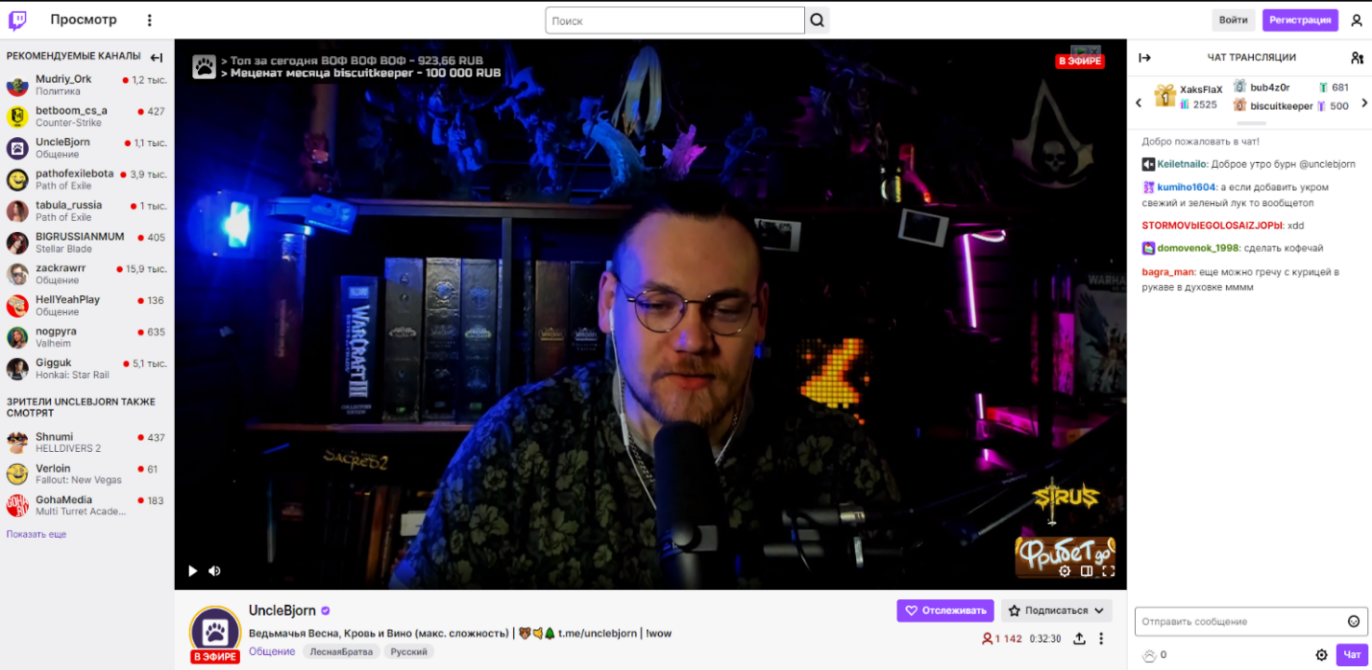
****

Рисунок – скриншот Twitch

Twitch предлагает множество инструментов для взаимодействия между стримером и аудиторией. Это групповой чат, денежные поощрения, подписки, опросы, рейды и многое другое. Его легко начать использовать как стримеру, так и зрителю. Twitch первым предложил систему подписок, которая позволяет зрителям материально поддерживать своих любимых стримеров, при этом получая уникальные возможности для общения.

Twitch ведет модерацию контента, благодаря чему на платформе нет вредного и опасного для общества контента. Однако в последнее время люди начали находить лазейки для трансляции откровенного контента, при этом модерация платформы ничего с этим не делает. [9]

Также стоит отметить, что Twitch дает возможность составить расписание трансляций и отправляет уведомления о начале стрима. Еще Twtich позволяет зрителям делать клипы – короткие вырезки интересных или смешных моментов со стрима.

Из основных минусов можно отметить большую конкуренцию и несправедливую, по мнению большинства зрителей и стримеров модерацию контента на платформе.

**VK Play Live**

VK Play Live – стриминговая платформа, принадлежащая российская инвестиционная технологическая корпорация VK. Была создана в 2022 году, входит в состав крупного российского игрового сервиса VK Play. Создавалась как аналог Twitch.

VK Play Live, как и Twitch, в первую очередь ориентирован на видеоигры и игроков. Однако имеет значительно меньшую аудиторию, чем Twitch. В основном имеет те же возможности что и конкурент, но при этом имеет пониженную комиссию. Основной отличительной чертой можно назвать то, что он ориентирован на Россию и внедрен в игровой сервис VK Play. [10]

Скриншот с площадки VK Play Live представленная на рисунке 2.

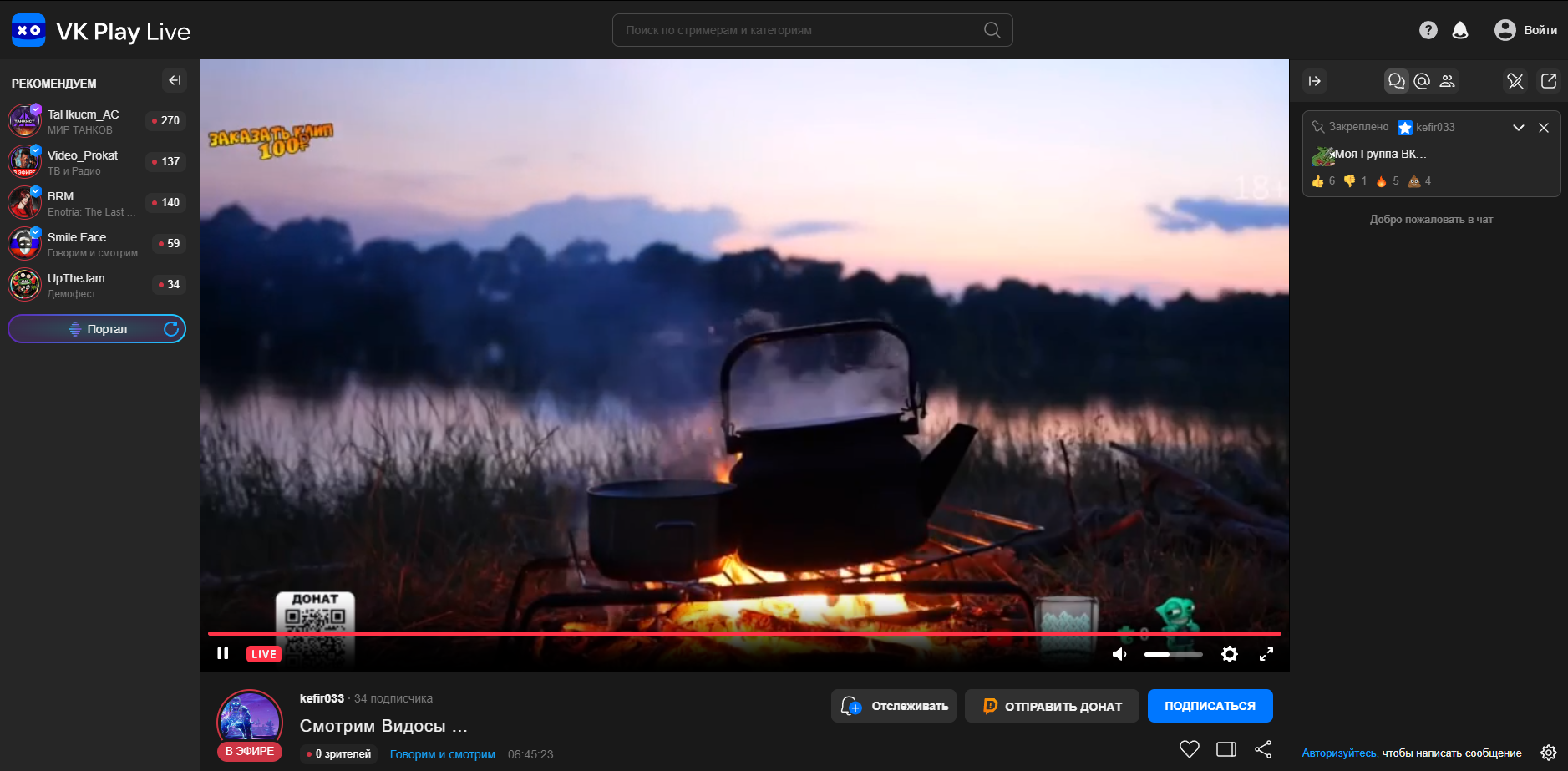


Рисунок 2 – VK Play Live

**Kick**

Kick – видеостриминговый сервис, основанный одноименной компанией в 2022 году при поддержке сайта Stake.com как конкурент Twitch, с акцентом на более свободную модерацию и более высокую долю доходов для стримеров. На рисунке 3 представлен скриншот с площадки Kick.

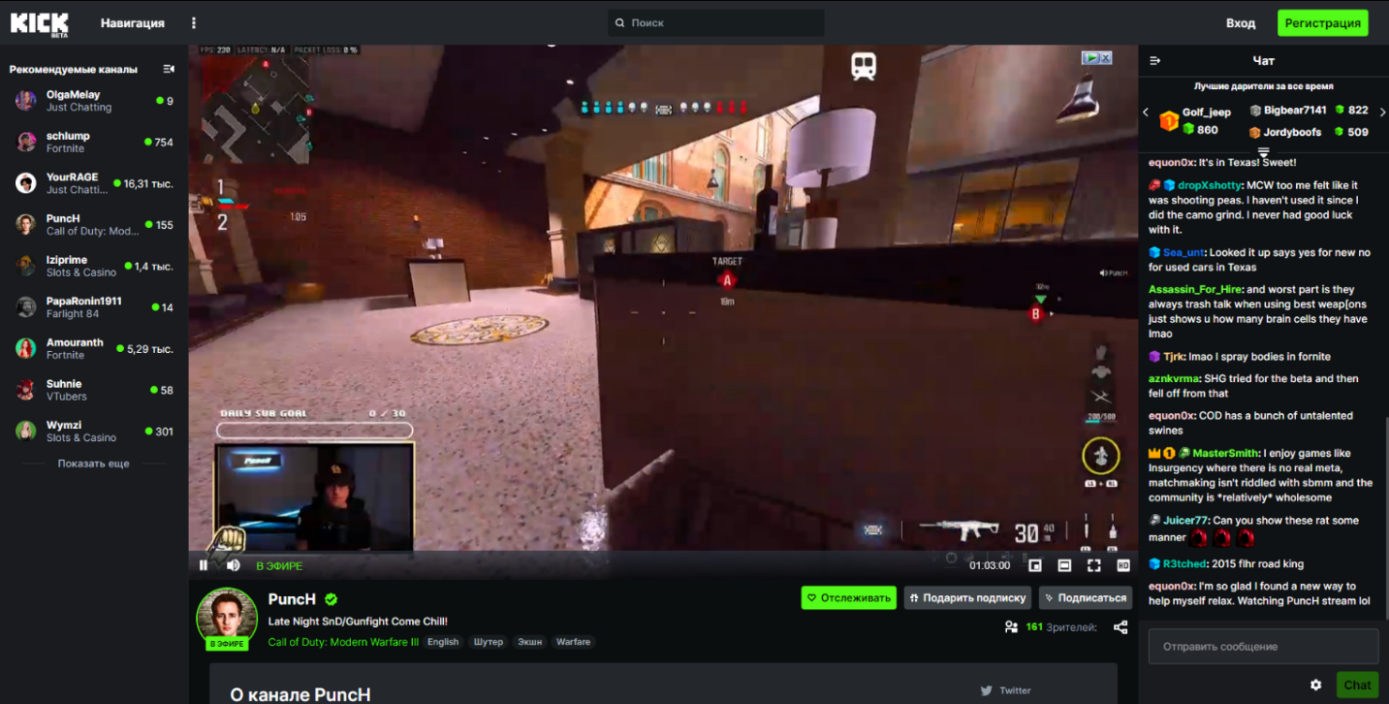
****

Рисунок – Kick

Предлагает тот же функционал что и Twitch, но имеет и некоторые преимущества. К примеру, Kick не показывает рекламу во время трансляций. Kick предоставляет большую свободу создателям контента, а также самую выгодную для стримеров систему монетизации (95%/5%). [11]

Таким образом, можно выделить набор обязательных функций, которые следует реализовать при разработке подобного веб-приложения:

* Предоставлять возможность авторизации на сервисе;
* Предоставлять возможность проведения прямых трансляций на сервисе;
* Предоставлять возможность поиска каналов и трансляций;
* Предоставлять возможность переписке в общем чате трансляции;
* Предоставлять возможность отслеживать интересуемые каналы.
  1. Анализ существующих решений для реализации проекта

На сегодняшний день существует немало систем для создания веб-приложений. Рассмотрим наиболее популярные технологии.

При создании интерфейса (frontend), разработчики обычно используют, такие языки программирования как HTML, CSS и JavaScript, которые позволяют создавать различные элементы интерфейса, определять их внешний вид и поведение. [7]

При разработке также используются наборы уже написанного кода, и специальных инструментов, предоставляющих готовые функции и структуры, которые разработчики могут использовать для упрощения определенных задач разработки. Такие наборы инструментов называют библиотеками и фреймворками. Вот некоторые популярные из них [5]:

* React.js: Библиотека от Facebook для создания пользовательских интерфейсов. Очень популярна за счет своей гибкости и мощного экосистема. [2]
* Vue.js: Прогрессивный фреймворк для построения пользовательских интерфейсов. Легкий в изучении и интеграции в проекты, что делает его привлекательным для стартапов и малых проектов. [13]
* Angular: Платформа и фреймворк для создания SPA (одностраничных приложений), разработанный Google. Предоставляет мощные инструменты для разработки. [14]
* Tailwind: CSS-фреймворк, ориентированный на полезность, содержит множество классов, которые можно комбинировать для создания любого дизайна прямо в вашей разметке. [4]
* Bootstrap: библиотека готовых компонентов для создания веб-страниц. [15]

При выполнении работы я выбрал библиотеку React для JS, потому что она достаточно популярна и уже прошла проверку временем, над ее поддержкой работает крупная компания. Также эту библиотеку мы проходили на курсе веб-программирования. Помимо этого, я выбрал Tailwind для изменения внешнего вида веб-страницы.

Для бэкенда, то есть для той части, которая отвечает за логику приложения на сервере, также существуют свои языки программирования и фреймворки. На сегодняшний день самыми популярными являются следующие решения [5]:

* Python: универсальный язык программирования, хорошо подходит для веб-разработки. [21]
* JavaScript: с помощью Node.js JavaScript можно использовать и на сервере. [16]
* PHP: язык программирования, изначально созданный для веб-разработки. [17]
* Java: популярный язык программирования, подходит для создания сложных веб-приложений. [18]
* Django: фреймворк Python, известный своей простотой и удобством использования. [19]
* Next.js: полнофункциональный фреймворк React для создания веб-приложений. [3]
* Spring: фреймворк Java, хорошо подходит для создания масштабируемых веб-приложений. [20]

При выполнении работы я выбрал программную платформу Node.js и фреймворк Next.js. Node.js предоставляет хорошую производительность, широкий выбор модулей и библиотек. Next.js в свою очередь предоставляет удобную систему папок, простую интеграцию с API и неплохую производительность.

Еще важным элементом являются системы управления базами данных. СУБД – это программное обеспечение, которое позволяет организовывать и хранить данные, а также управлять ими. [6]

* MySQL: свободная реляционная система управления базами данных. Одна из популярнейших СУБД. [22]
* PostgreSQL: это продвинутая объектно-реляционная система баз данных с открытым исходным кодом. [23]
* MongoDB: NoSQL база данных, хорошо подходит для работы с JSON-документами. [24]

В качестве систему управления базами данных я выбрал MySQL, так как она является одной из популярнейших СУБД, распространяется бесплатно с открытым исходным кодом, и при этом предоставляет оптимальную производительность и стабильность.

Для реализации потоковой трансляции видео будет использоваться LiveKit – набор инструментов с открытым исходным кодом, обеспечивающий масштабируемую многопользовательскую конференц-связь на основе WebRTC. Выбор пришелся на LiveKit поскольку это проект с открытым исходным кодом, а также он предоставляет бесплатный пробный период использования. [26]

1. ТРЕБОВАНИЯ К СИСТЕМЕ

В ходе работы мной были сформулированы следующие функциональные требования, которые обеспечат широкие возможности для пользователей веб-приложения.

* + Пользователи должны иметь возможность создавать учетные записи и входить в систему с использованием уникального логина и пароля.
  + Приложение должно предоставлять возможность поиска каналов и трансляций по названию.
  + Пользователи должны иметь возможность просматривать живые трансляции.
  + Пользователи должны иметь возможность начинать и завершать собственные трансляции с помощью специальных приложений.
  + Приложение должно поддерживать возможность подписки на каналы стримеров.
  + Пользователи должны иметь возможность писать сообщения в чат для общения друг с другом и со стримером во время трансляции.
  + Пользователи, ведущие трансляцию, должны иметь возможность блокировать других пользователей.

Таким же образом были сформулированы следующие нефункциональные требования.

* + Веб-приложение должно быть разработано с использованием HTML, CSS, TypeScript;
  + Веб-приложение должно быть разработано с помощью библиотеки React и фреймворков Next.js, Tailwind CSS;
  + Для реализации потокового вещания будет использоваться инструментарий LiveKit.
  + База данных должна быть реализована с помощью MySQL.

1. ПРОЕКТИРОВАНИЕ
   1. Диаграмма вариантов использования

В процессе определения требований к веб-приложению была сформирована диаграмма вариантов использования, представленная на рисунке 4.

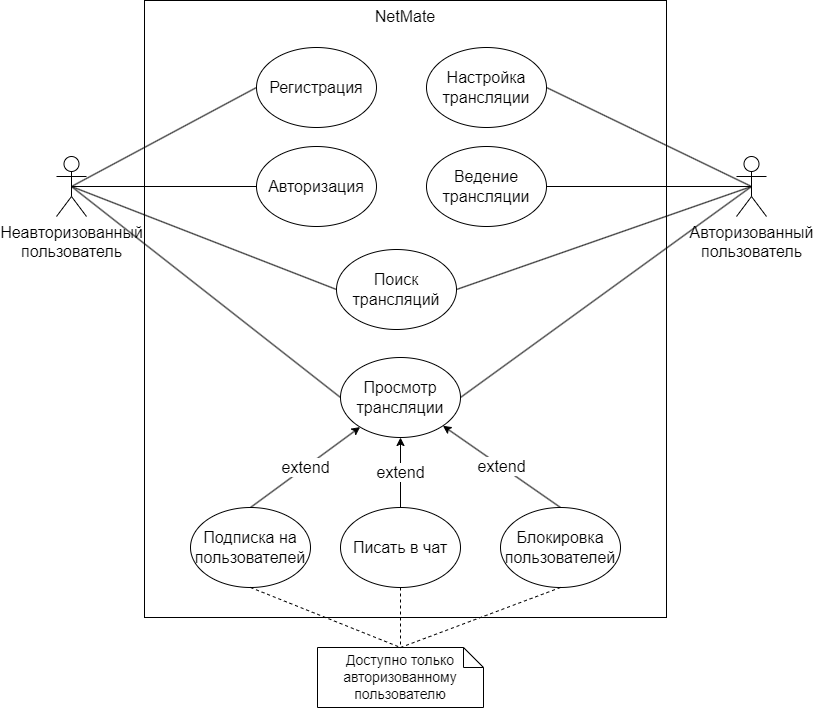


Рисунок 4 – Диаграмма вариантов использования

В данной диаграмме представлены два основных актера: авторизованный пользователь и неавторизованный пользователь.

Авторизованный пользователь – это зарегистрированный пользователь, авторизовавшийся в системе, который может просматривать трансляции и взаимодействовать с контентом.

Неавторизованный пользователь – это любой человек, который посещает платформу без регистрации и входа в систему.

У актеров есть разные варианты использования. Так неавторизованный пользователь имеет возможность для регистрации и авторизации, также он может производить поиск трансляций и просматривать их, но не может общаться в чате.

Авторизованный пользователь в свою очередь не может зарегистрировать новый аккаунт или авторизоваться под другими данными, для этого ему потребуется выйти из своей учетной записи. Но при этом ему доступны все остальные функции платформы. Авторизованный пользователь может искать и смотреть трансляции, самостоятельно проводить трансляции, а также подписываться на других пользователей или блокировать их. Можно настраивать трансляцию изменяя ее название и обложку. Также при настройке трансляции пользователю необходимо сгенерировать и сохранить уникальный ключ трансляции, для того чтобы стримить на платформе.

Спецификацию вариантов использования можно найти в приложении к работе.

* 1. Проектирование интерфейса

У рассмотренных аналогов интерфейсы очень похожи. Поэтому за основу я взял интерфейс платформы Twitch, для того чтобы пользователям, которые использовали такие платформы ранее было легче ориентироваться в приложении.

* 1. Диаграмма компонентов

В процессе определения требований к веб-приложению была сформирована диаграмма компонентов, представленная на рисунке 5.

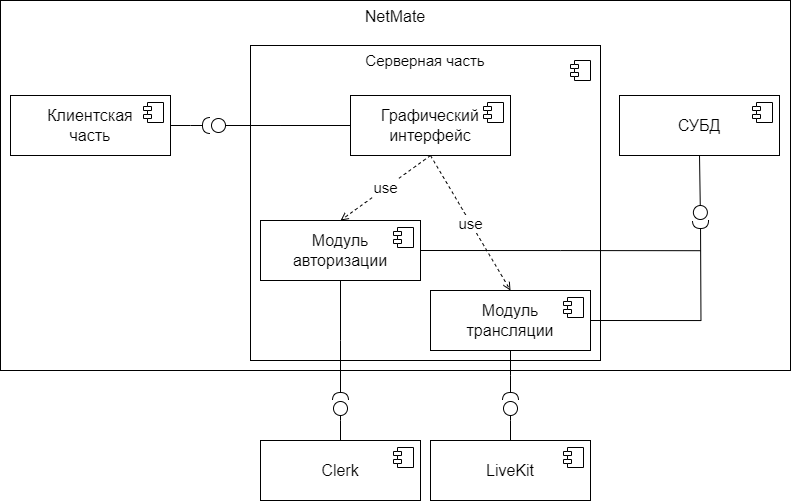


Рисунок 5 – Диаграмма компонентов

Для разработки за основу взята трехуровневая архитектура. Архитектурная модель, предполагающая наличие в веб-приложении трех типов компонентов: клиент, сервер приложения и сервер базы данных.

Помимо этого, использовались API сторонних приложений: Clerk и LiveKit. Сервис Clerk предоставляет удобную систему аутентификации. А LiveKit позволяет организовывать трансляции по протоколу WebRTC.

* 1. Проектирование базы данных

База данных предназначена для хранения, обработки и изменения информации, выстраивания взаимосвязей. В этой работе она необходима для работы с данными пользователей. Структура базы данных представлена на рисунке 6.

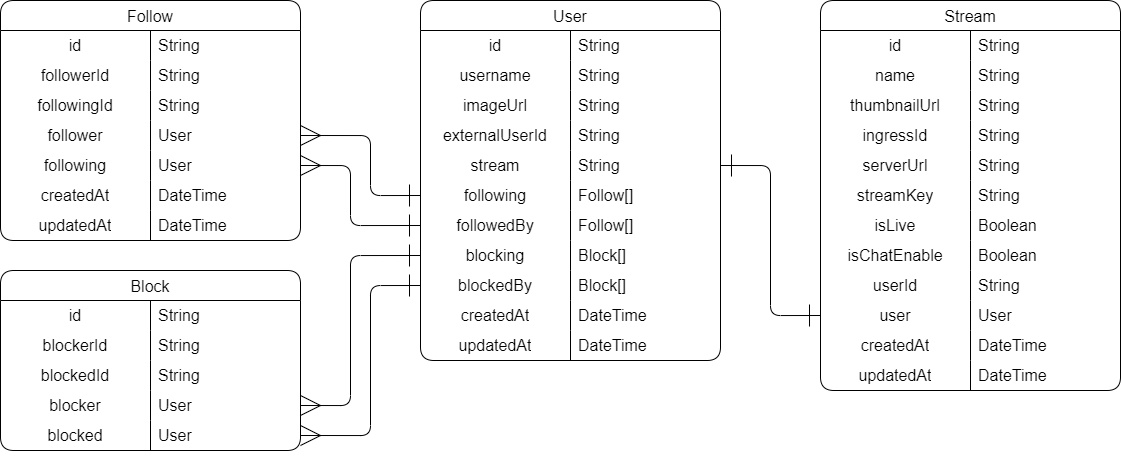


Рисунок 6 – Схема базы данных

Для работы с базой данных выбрана СУБД MySQL. Это реляционная система управления базами данных с открытым исходным кодом. Реляционная модель довольно удобная и производительная, также она позволяет легко масштабировать данные без нарушения структуры базы.

1. РЕАЛИЗАЦИЯ
   1. Создание проекта

Для начала работы необходимо скачать и установить Node.js. Далее, чтобы создать новый проект на основе Next.js нужно прописать в консоли «npx create-next-app». Прямо в консоли можно произвести первоначальные настройки проекта, после чего автоматически загрузятся все требуемые пакеты.

Одной из основных функций Next.js является маршрутизация на основе файловой системы. Как это работает наглядно показано на рисунке 7 и на рисунке 8.

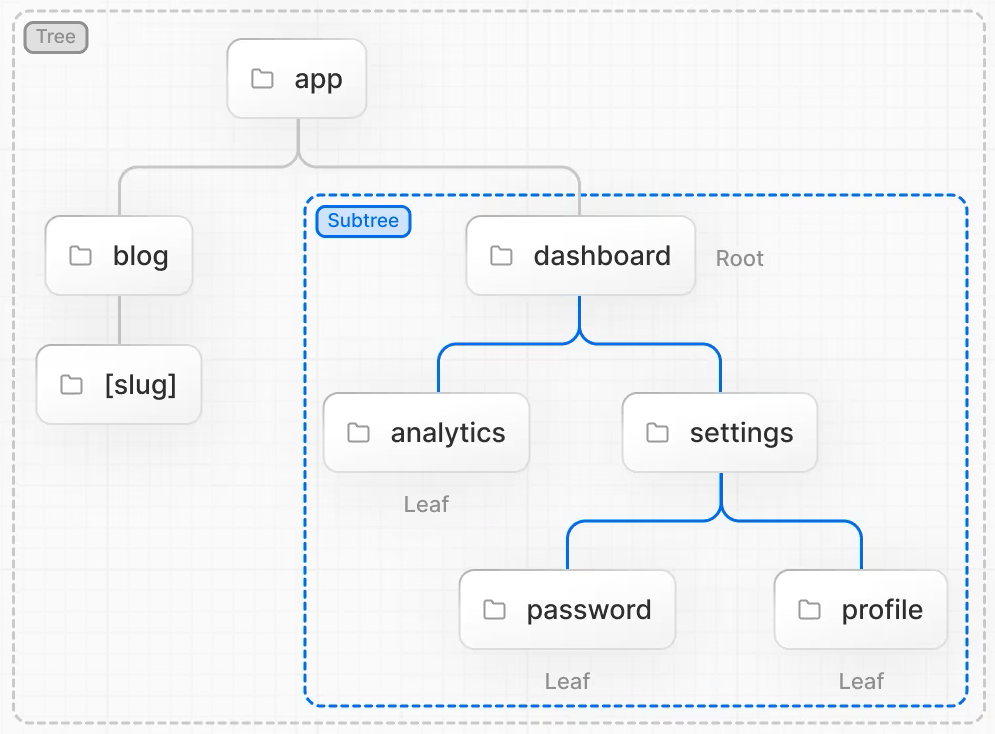


Рисунок 7 – Структура файловой системы

Рисунок 7 взят с официального сайта Next.js [3]. На нем изображен пример файловой структуры приложения.

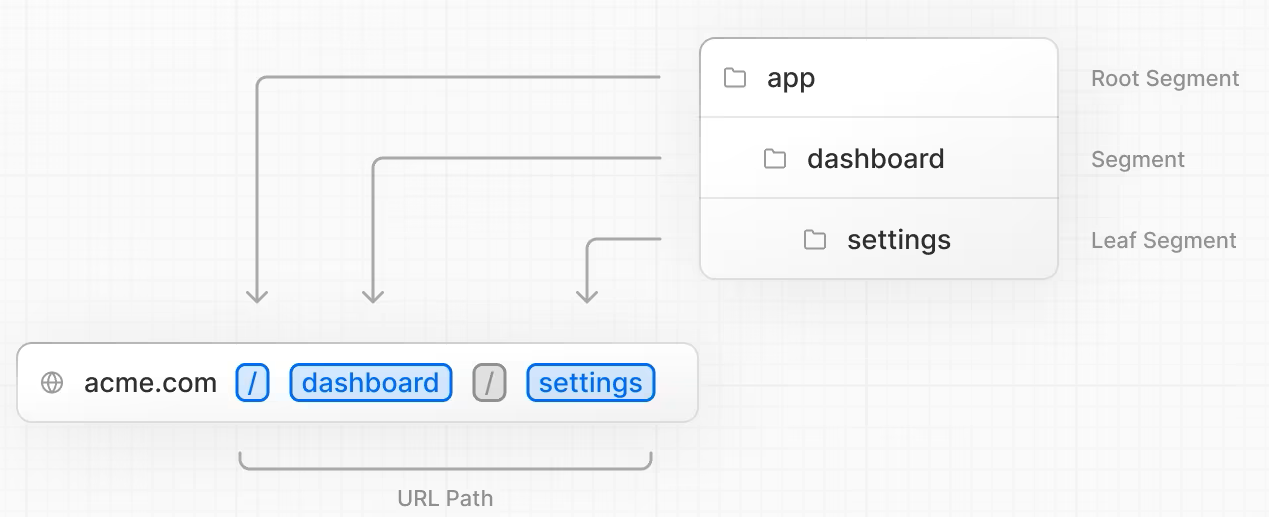


Рисунок 8 – Адрес страницы

Рисунок 8 также взят с официального сайта Next.js [3]. На нем изображена адресная строка. Таким образом адрес «acme.com/dashboard/settings» ведет к странице «settings». Чтобы страница отображалась у клиента, необходимо чтобы в соответствующей папке на сервере был файл «page.tsx». В этом файле в свою очередь должна быть функция, вызываемая по умолчанию. Эта функция должна возвращать HTML код, который будет отображаться у пользователя.

* 1. Страница авторизации

В первую очередь нужно создать страницы авторизации и регистрации. В папке «app» необходимо создать папку «(auth)». В Next.js папка в скобках не будет отображаться в адресе страницы. Внутри будут еще 2 папки «sign-in» и «sign-up» для авторизации и регистрации соответственно. Также в папке «(auth)» будет файл «layout.tsx», отвечающий за общие элементы внешнего вида страниц авторизации и регистрации. Данные из этого файла не будут заново загружаться при переходе между этими страницами. На рисунке 9 представлен скриншот страницы авторизации в приложении.

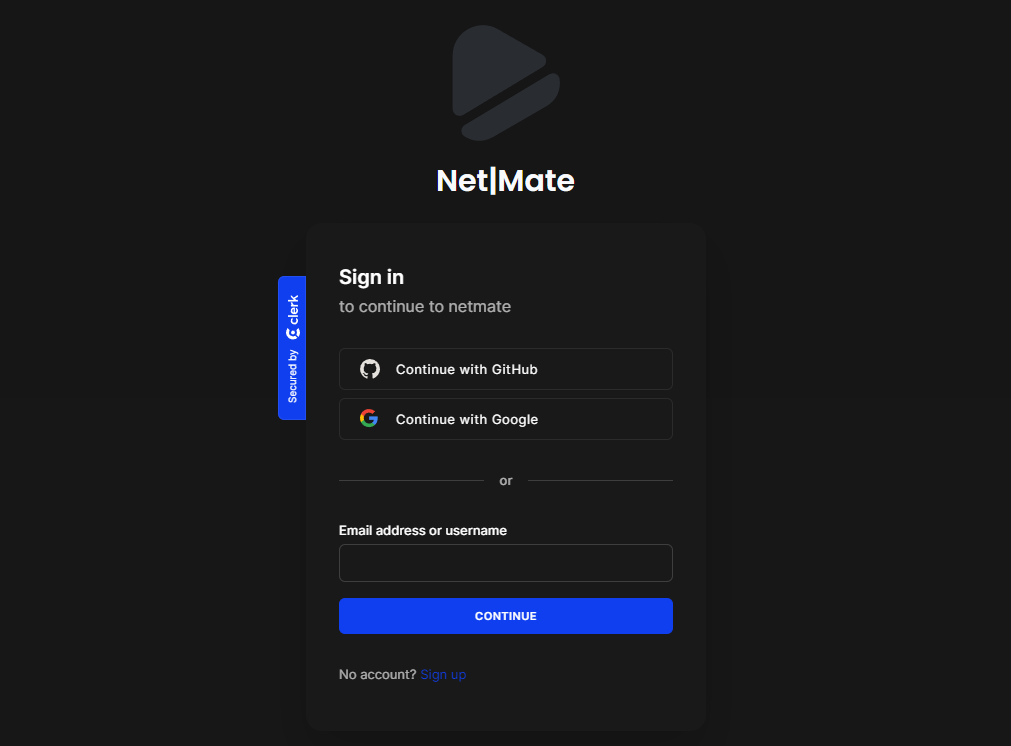


Рисунок 9 – Страница авторизации

Логика аутентификации реализована с помощью сервиса Clerk. Когда пользователь проходит аутентификацию сервис Clerk сообщает об этом серверу моего приложения.

Clerk оправляет запросы на сервер приложения как только происходит какое-нибудь событие. Всего может быть 3 события: создание, изменение или удаление аккаунта. И для каждого из них необходимо прописать свою логику работы с базой данных. На листинге 1 показан пример обработки запроса для создания нового аккаунта.

Листинг 1 – обработка запроса для создания аккаунта

if (eventType === "user.created") {

await db.user.create({

data: {

externalUserId: payload.data.id,

username: payload.data.username,

imageUrl: payload.data.image\_url,

stream: {

create: {

name: `Стрим пользователя ${payload.data.username}`,

}

}

}

})

}

* 1. База данных

Для работы с базой данных использовалась реляционная СУБД MySQL. Сама база данных располагается локально на том же компьютере, где и сервер приложения.

Для связи MySQL и Node.js использовалась ORM Prisma. Prisma – инструмент, позволяющий работать с базой данных с помощью JavaScript или TypeScript без использования SQL (хотя такая возможность имеется).

Сама база данных расширялась постепенно в ходе разработки приложения. Сначала необходимо создать базу данных, для этого необходимо запустить MySQL от имени администратора командой «mysql -u root». Далее необходимо создать новую базу данных командой «CREATE DATABASE netmatedb;», netmatedb – название базы данных.

Теперь необходимо подключить созданную базу данных к ORM Prisma. Для этого в файле «.env» необходимо прописать строчку «DATABASE\_URL="mysql://root:root@localhost:3333/netmatedb"». После этого в файле «schema.prisma» нужно указать источник данных. Это показано на листинге 2.

Листинг 2 – источник данных Prisma

datasource db {

provider = "mysql"

url = env("DATABASE\_URL")

relationMode = "prisma"

}

После этого Prisma имеет доступ к созданной базе данных. Так что дальнейшую работу с базой данных можно производить с помощью Prisma. А обращаться к данным можно непосредственно в коде на языке TypeScript.

* 1. Главная страница

На главной странице располагаются все трансляции в виде сетки. Карточка трансляции представляет из себя прямоугольник с обложкой, которую может задать пользователь. Под обложкой написано название трансляции, аватар и имя пользователя. Если трансляция активна, то аватар пользователя будет обведен красным цветом.

Скриншот главной страницы показан на рисунке 10.

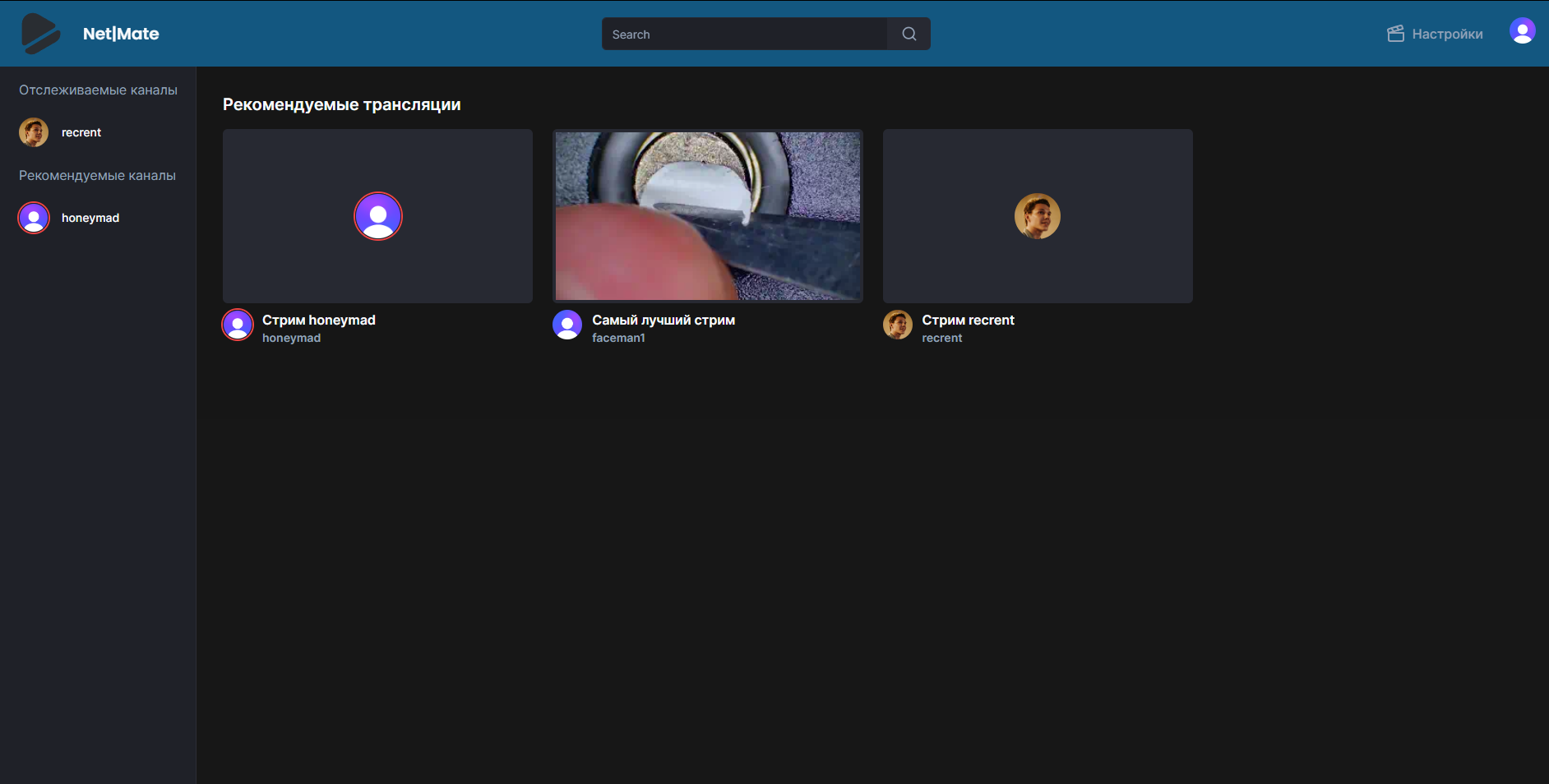


Рисунок 10 – Главная страница

На листинге 3 представлена основная структура главной страницы. Сначала идёт заголовок в теге «h2», далее идёт проверка на наличие трансляций. Если трансляций нет, то выводится сообщение об этом. В ином случае выводится карточки всех существующих трансляций. Карточка трансляции представлена компонентом «StreamCard». Код этого компонента можно увидеть на листинге 1 в приложении Б.

Листинг 3 – Главная страница

export const Streams = async () => {

const streams = await getStreams()

return (

<div>

<h2 className="text-xl font-semibold mb-4">

Рекомендуемые трансляции

</h2>

{streams.length === 0 && (

<div className="text-muted-foreground ">

Здесь пока пусто

</div>

)}

<div className="grid grid-cols-4 gap-6">

{streams.map((stream) => (

<StreamCard

key={stream.id}

stream={stream}

/>

))}

</div>

</div>

)}

Слева на экране расположено боковое меню со списком активных и отслеживаемых пользователей. На листинге 4 представлен код бокового меню. Здесь используются методы «getRecommended» и «getFollowedUsers» для получения списка рекомендованных активных стримеров. Эти списки выводятся в боковом меню через компоненты «Recommended» и «Following»

В качестве примера приведены компоненты «getRecommended» и «Recommended» в листинге 2 и листинге 3 в приложении Б.

Листинг 4 – Боковое меню

export const Sidebar = async () => {

const recommended = await getRecommended()

const following = await getFollowedUsers()

return (

<div className="fixed left-0 flex flex-col w-60 h-full bg-background border-r border-[#2D2E35] z-50">

<div className="space-y-4 pt-4">

<Following data={following}/>

<Recommended data={recommended}/>

</div>

</div>

)

}

Сверху расположено еще одно меню. На нем есть логотип и название приложения, поисковая строка, а также кнопки для настройки трансляции и аккаунта. Если пользователь еще не авторизован, то на месте кнопок для настройки будет кнопка для авторизации. Верхнее меню представлено компонентом «Navbar». Код этого компонента представлен на листинге 5.

Листинг 5 – Верхнее меню

export const Navbar = () => {

return (

<nav className="fixed top-0 w-full h-20 z-[49] bg-[#135780] px-2 lg:px-4 flex justify-between items-center shadow-sm">

<Logo />

<Search />

<Actions />

</nav>

)

}

С помощью поисковой строки в верхнем меню можно искать трансляции. Поиск происходит сразу и по названию трансляции, и по имени пользователя. Все трансляции, подходящие под запрос, отображаются в сетке так же, как и на главной странице.

На Рисунке 11 представлена диаграмма последовательности, отображающая процесс поиска стримов.

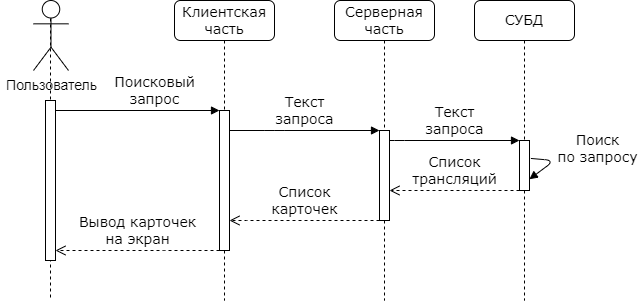


Рисунок 11 – Поиск трансляций

На листинге 6 представлен метод, возвращающий список подходящих по запросу стримов.

Листинг 6 – Поиск

export const getSearchResults = async (term?: string) => {

let userId

try {

const self = await getSelf()

userId = self.id

} catch {

userId = null

}

let streams = []

streams = await db.stream.findMany({

where: {

OR: [

{

name: {

contains: term

}

},

{

user: {

username:{

contains: term

}

}

}

]

},

include: {

user: true

},

orderBy: [

{

isLive: 'desc'

},

{

updatedAt: 'desc'

}

]

})

return streams

}

* 1. Страница трансляции

При нажатии на карточку трансляции или кнопку пользователя в боковом меню открывается страница стрима. Скриншот этой страницы представлен на рисунке 12. Боковое и верхнее меню остаются на своих местах. В центре располагается форма для трансляции.

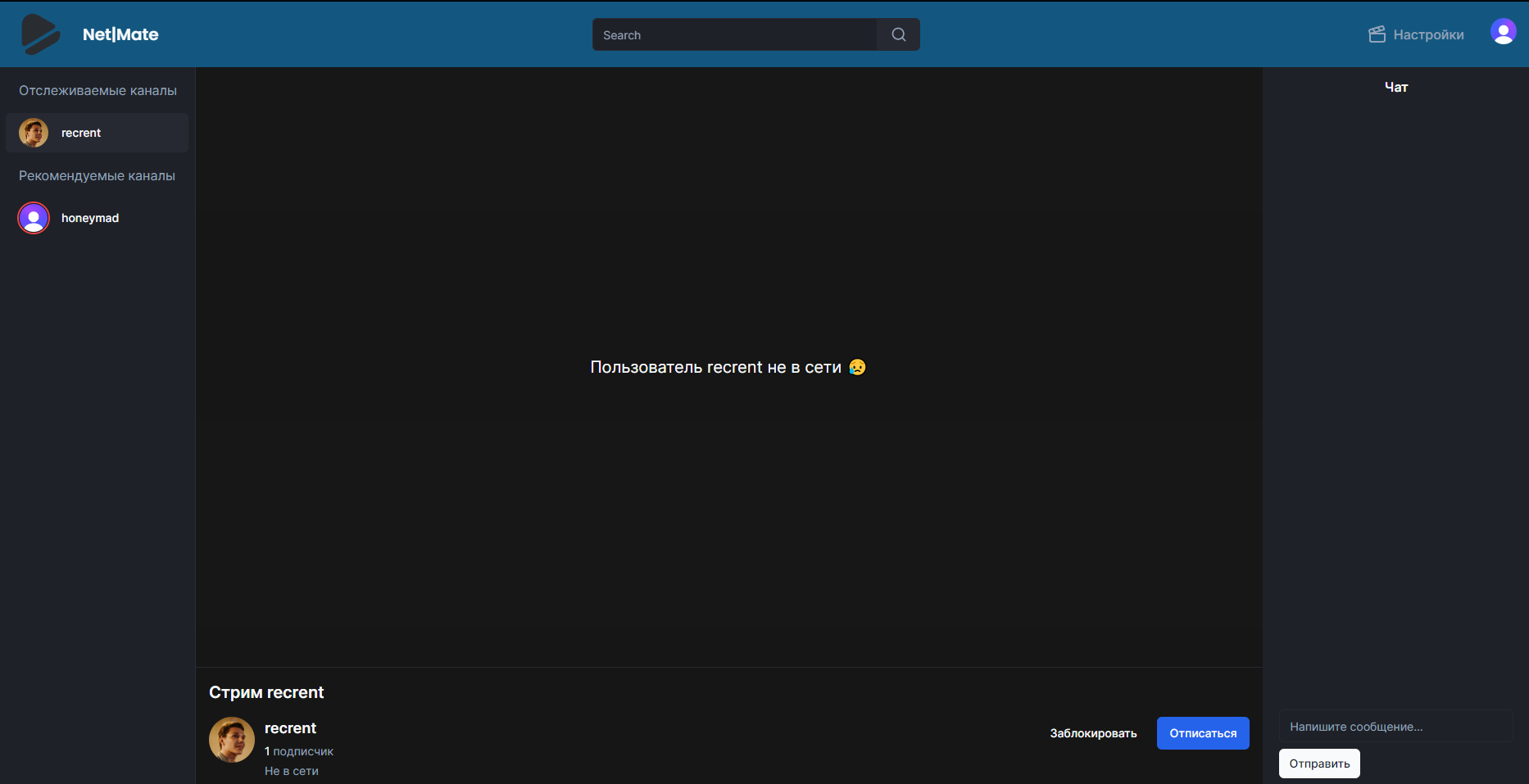


Рисунок 12 – Страница трансляции

Если стрим запущен, то в этой форме появляется браузерный медиапроигрыватель. В нем можно ставить стрим на паузу, менять громкость и разворачивать трансляцию во весь экран. Так как это трансляция в прямом эфире перематывать ее во времени нельзя. Если же стрим не запущен, то по середине формы появляется сообщение об этом.

Справа от формы для трансляции находится общий чат для всех пользователей, которые сейчас находятся на этой странице. История сообщений хранится у клиента до момента перезагрузки страницы стрима.

Потоковая трансляция видео, а также функционал чата предоставляется сервисом LiveKit. На листинге 7 представлен компонент «StreamPlayer». Этот компонент отвечает непосредственно за видеотрансляцию, поле с информацией о трансляции и её авторе, чат трансляции.

Листинг 7 – Компонент трансляции

interface StreamPlayerProps {

user: User & {

stream: Stream,

\_count: { followedBy: number }

}

stream: Stream

isFollowing: boolean

isBlocked: boolean

}

export const StreamPlayer = ({

user,

stream,

isFollowing,

isBlocked

}: StreamPlayerProps) => {

const { token, name, id } = useViewerToken(user.id)

if (!token || !name || !id) return

return (

<div>

<LiveKitRoom

token={token}

serverUrl={process.env.NEXT\_PUBLIC\_LIVEKIT\_WS\_URL}

className="grid grid-cols-5"

>

<div className="space-y-4 col-span-4 hidden-scrollbar pb-10">

<Video

hostName={user.username}

hostId={user.id}

/>

<Header

hostName={user.username}

hostId={user.id}

viewerId={id}

imageUrl={user.imageUrl}

name={stream.name}

isFollowing={isFollowing}

followedByCount={user.\_count.followedBy}

isBlocked={isBlocked}

/>

<InfoCard

hostId={user.id}

viewerId={id}

name={stream.name}

thumbnailUrl={stream.thumbnailUrl}

/>

</div>

<div>

<Chat

viewerName={name}

viewerId={id}

hostName={user.username}

hostId={user.id}

isChatEnabled={stream.isChatEnabled}

/>

</div>

</LiveKitRoom>

</div>

)

}

Под формой находится информация о трансляции и ее авторе. То есть название стрима, аватар и имя пользователя, количество зрителей. А также кнопки для подписки на этого пользователя и для блокировки этого пользователя. При подписке или блокировке пользователя, кнопки меняются на противоположные, для отписки и разблокировки соответственно. Если неавторизованный пользователь нажмёт кнопку для подписки или блокировки, то он будет перенаправлен на страницу авторизации.

Заблокированный пользователь не может находиться на странице трансляции пользователя, который его заблокировал.

На рисунке 13 изображена диаграмма деятельности, отображающая процесс подписки одного пользователя на другого пользователя.

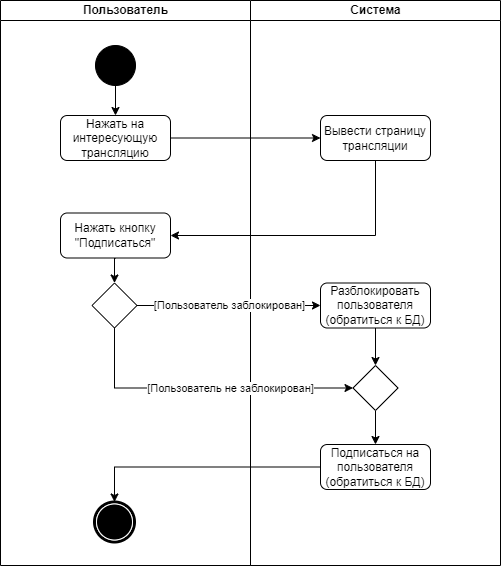


Рисунок 13 – Подписка на пользователя

На листинге 4 в приложении Б показан компонент «Actions». Этот компонент отвечает за кнопки для подписки и блокировки.

* 1. Страница настроек

У авторизованного пользователя в верхнем меню есть кнопка «Настройки». При нажатии этой кнопки пользователь попадёт на страницу настройки трансляции. На этой странице в боковом меню слева есть 3 пункта: «Стрим», «Ключ» и «Чат».

Пункт «Стрим» позволяет увидеть свою трансляцию и изменить информацию о ней.

Пункт «Ключ» отвечает за специальную информацию для проведения трансляции. Это ключ и сервер трансляции, которые расположены в соответствующих полях на странице. Эти данные нужны для специальных приложений (например OBS), которые позволяют вести трансляцию настроенных сцен. Если пользователь впервые хочет провести трансляцию, то для начала ему потребуется сгенерировать ключ и сервер трансляции. Для этого есть соответствующая кнопка на странице. На рисунке 14 изображен скриншот страницы настройки ключа трансляции.

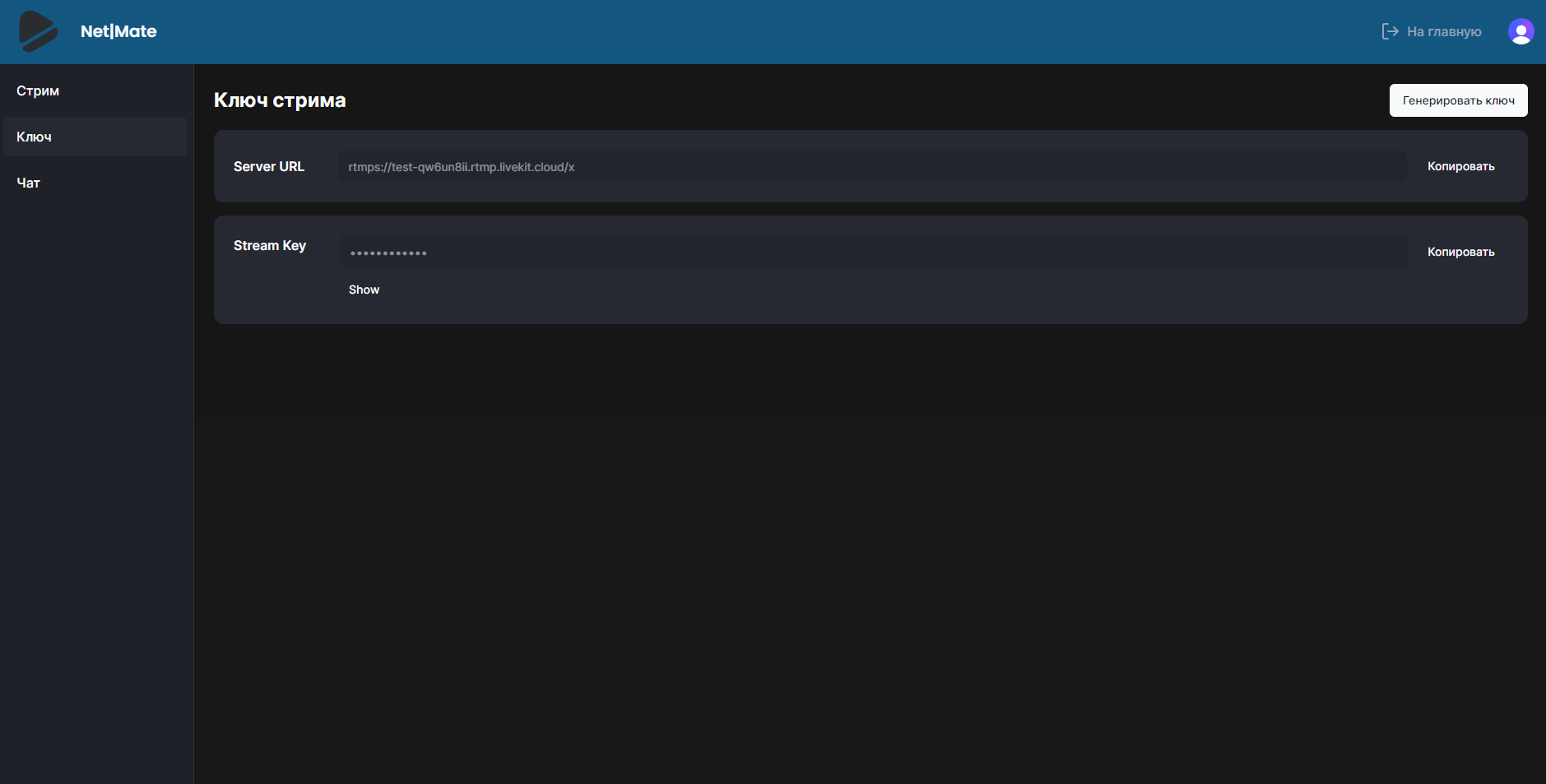


Рисунок 14 – Страница настройки ключа

Пункт «Чат» отвечает за настройки чата у трансляции. Там есть лишь одна настройка для включения или отключения чата. Скриншот этой настройки изображён на рисунке 15.

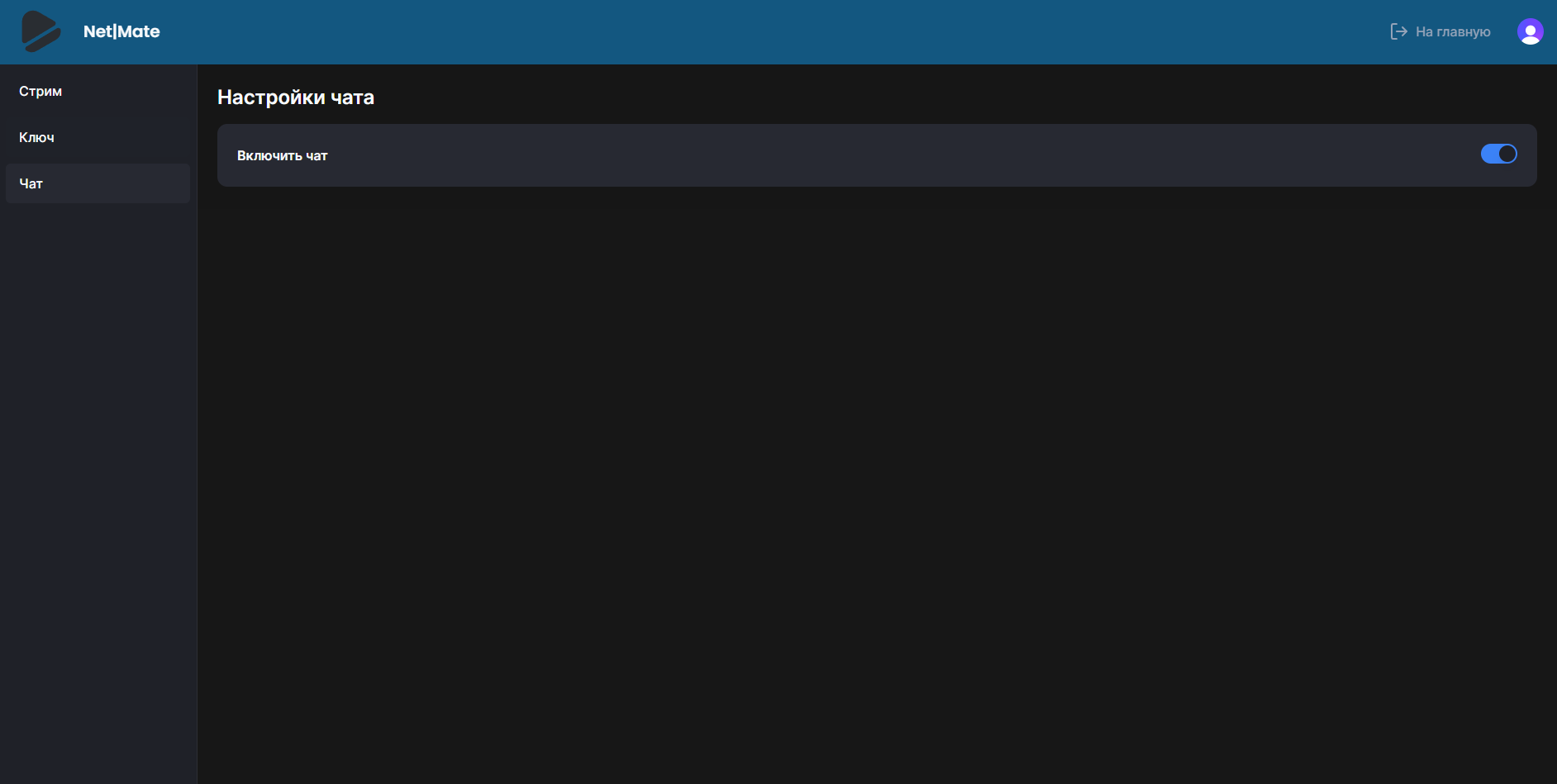


Рисунок 15 – страница настройки чата

1. ТЕСТИРОВАНИЕ

Для проверки системы использовалось функциональное тестирование. Это один из самых популярных видов тестирования, который проверяет соответствие функциональности продукта тому, как он был задуман [27]. В таблице 1 представлен протокол функционального тестирования.

Таблица 1 – Функциональное тестирование

| **№** | **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
| --- | --- | --- | --- | --- |
|  | Проверка регистрация | 1. На главной странице нажать кнопку «Login» 2. На странице авторизации нажать кнопку «Sign-up». 3. Заполнить форму регистрации. 4. Нажать на кнопку «Готово». | В базе данных появляется информация о новом пользователе. | Да |
|  | Проверка входа в систему | 1. На главной странице нажать кнопку «Login» 2. На странице аутентификации ввести зарегистрированный адрес электронной почты и верный пароль. 3. Нажать на кнопку «Войти». | Пользователю становится доступны новые функции. | Да |
|  | Проверка поиска | 1. Ввести запрос в поисковую строку на сайте. 2. Нажать кнопку «Поиск» | Отображается список трансляций, удовлетворяющих запросу | Да |
|  | Проверка подписки на канал | 1. На главной странице выбрать интересующую трансляцию или канал. 2. Нажать кнопку «Подписаться». | Этот канал появляется в списке отслеживаемых. Появляется новая запись в БД | Да |
|  | Проверка блокировки пользователя | 1. На главной странице выбрать нужную трансляцию или канал. 2. Нажать кнопку «Заблокировать». | Заблокированный пользователь больше не может смотреть ваши трансляции. Появляется новая запись в БД | Да |
|  | Проверка генерации ключа трансляции | 1. На главной странице нажать кнопку «Настройки» 2. В пункте «Ключ» нажать кнопку «Генерировать». | В полях «Stream Key» и «Server URL» появляются новые значения. | Да |

Окончание таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | 1. **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
|  | Проверка запуска трансляции | 1. Открыть специальное приложение (например OBS). 2. Ввести сгенерированные «Stream Key» и «Server URL». 3. Запустить трансляцию | На странице пользователя появляется транслируемая из OBS сцена. | Да |
|  | Проверка чата | 1. Запустить трансляцию. 2. Перейти на страницу пользователя. 3. Ввести сообщение в поле чата 4. Нажать кнопку «Отправить». | В чате появляется новое сообщение. | Да |
|  | Проверка функции отключения чата | 1. На главной странице нажать кнопку «Настройки». 2. Перейти в пункт «Чат». 3. Выключить чат с помощью переключателя «Включить чат» 4. Запустить трансляцию | На странице пользователя поле ввода сообщения в чат блокируется. | Да |

Все тесты были пройдены успешно.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы было разработано веб-приложение для потоковой передачи видео и общения пользователей.

При этом были решены следующие задачи:

1. Проведен анализ предметной области;
2. Разработаны требования к веб-положению;
3. Спроектировано веб-приложение;
4. Реализовано веб-приложение для потоковой передачи видео и общения пользователей в групповом чате;
5. Провести тестирование веб-приложения.

ЛИТЕРАТУРА

1. SimilarWeb – сервиса для анализа сайтов. [Электронный ресурс] URL: https://www.similarweb.com (дата обращения: 3.03.2024).
2. React – библиотека для JavaScript [Электронный ресурс] URL: https://react.dev (дата обращения: 01.03.2024).
3. Next.js – фреймворк для React. [Электронный ресурс] URL: https://nextjs.org (дата обращения: 01.04.2024).
4. Tailwind CSS – CSS-фреймворк. [Электронный ресурс] URL: https://tailwindcss.com (дата обращения: 01.04.2024).
5. Statista – Портал статистики. [Электронный ресурс] URL: <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/> (дата обращения: 20.04.2024).
6. DB-Engines – База знаний о СУБД. [Электронный ресурс] URL: <https://db-engines.com/en/ranking> (дата обращения: 1.05.2024).
7. Айтилогия – Онлайн-школа. [Электронный ресурс] URL: <https://itlogia.ru/article/html_css_javascript_i_php_chto_takoe_i_dlya_chego> (дата обращения: 1.05.2024).
8. Twitch – Видеостриминговый сервис. [Электронный ресурс] URL: https://www.twitch.tv (дата обращения: 5.04.2024).
9. 3DNews – российское онлайн-издание, посвященное цифровым технологиям. [Электронный ресурс] URL: https://3dnews.ru/1097451/o-vremena-o-nravy-twitch-smyagchil-pravila-pokaza-seksualnogo-kontenta (дата обращения: 5.04.2024).
10. RBC – медиахолдинг. [Электронный ресурс] URL: <https://www.rbc.ru-/life/news/6308acd09a7947f6f1995ea4> (дата обращения: 5.04.2024).
11. Internet Matters – медиа-портал. [Электронный ресурс] URL: https://www.internetmatters.org/ru/hub/news-blogs/what-is-kick-streaming/ (дата обращения: 5.04.2024).
12. Onlypult – платформа для работы с социальными медиа. [Электронный ресурс] URL: https://onlypult.com/ru/blog/media-buduschego-7-sposobov-ispolzovat-striming (дата обращения: 5.04.2024).
13. Vue.js – JavaScript-фреймворк. [Электронный ресурс] URL: https://vuejs.org (дата обращения: 5.04.2024).
14. Angular – JavaScript-фреймворк. [Электронный ресурс] URL: https://angular.io (дата обращения: 5.04.2024).
15. Bootstrap – библиотека для работы c HTML, CSS и JS. [Электронный ресурс] URL: https://getbootstrap.com (дата обращения: 5.04.2024).
16. Node.js – среда выполнения JavaScript. [Электронный ресурс] URL: https://nodejs.org/en (дата обращения: 5.04.2024).
17. Php – язык программирования. [Электронный ресурс] URL: https://www.php.net (дата обращения: 5.04.2024).
18. Java – язык программирования. [Электронный ресурс] URL: https://vuejs.org (дата обращения: 5.04.2024).
19. Django – Python-фреймворк. [Электронный ресурс] URL: https://www.djangoproject.com (дата обращения: 5.04.2024).
20. Spring – Java-фреймворк. [Электронный ресурс] URL: https://spring.io (дата обращения: 5.04.2024).
21. Python – язык программирования. [Электронный ресурс] URL: https://www.python.org (дата обращения: 5.04.2024).
22. MySQL – система управления базами данных. [Электронный ресурс] URL: https://www.mysql.com (дата обращения: 5.04.2024).
23. PostgreSQL – система управления базами данных. [Электронный ресурс] URL: https://www.postgresql.org (дата обращения: 5.04.2024).
24. MongoDB – система управления базами данных. [Электронный ресурс] URL: https://www.mongodb.com (дата обращения: 5.04.2024).
25. Business of Apps. [Электронный ресурс] URL: https://www.mongo-db.com (дата обращения: 16.05.2024).
26. LiveKit. [Электронный ресурс] URL: https://livekit.io (дата обращения: 18.05.2024).
27. Яндекс Практикум. [Электронный ресурс] URL: https://practicum.yandex.ru/blog/funkcionalnoe-testirovanie-po/ (дата обращения: 2.06.2024).
28. Розенберг, Д. Применение объектного моделирования с использованием UML и анализ прецедентов
29. Вейцман, В. М. Проектирование информационных систем: учебное пособие / В. М. Вейцман. — Санкт-Петербург, 2019. — 316 с.

ПРИЛОЖЕНИЯ

Приложение А. Спецификация вариантов использоания

Таблица 1 – Спецификация варианта использования «Авторизация»

|  |
| --- |
| ***UseCase:* Авторизация** |
| *ID:* 1 |
| *Аннотация:* Авторизация пользователя |
| *Главные актеры:* Неавторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* Пользователь не авторизован |
| *Основной поток:*   1. Пользователь нажимает кнопку «Login» 2. Система выводит страницу авторизации 3. Пользователь вводит данные для авторизации 4. Система проверяет данные и выдает соответствующие права пользователю |
| *Постусловия:* Пользователь авторизован |
| *Альтернативные потоки:* 2 |

Таблица 2 – Спецификация варианта использования «Регистрация»

|  |
| --- |
| ***UseCase:* Регистрация** |
| *ID:* 2 |
| *Аннотация:* Регистрация пользователя |
| *Главные актеры:* Неавторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* Пользователь не зарегистрирован в системе |
| *Основной поток:*   1. Пользователь нажимает кнопку «Login» 2. Система выводит страницу авторизации 3. Пользователь нажимает кнопку «Sign-Up» 4. Система выводит страницу регистрации 5. Пользователь вводит данные для регистрации 6. Система создает аккаунт пользователя |
| *Постусловия:* Пользователь зарегистрирован |
| *Альтернативные потоки:* нет |

Таблица 3 – Спецификация варианта использования «Ведение трансляции»

|  |
| --- |
| ***UseCase:* Ведение трансляции** |
| *ID:* 3 |
| *Аннотация:* Пользователь запускает и ведет трансляцию |
| *Главные актеры:* Авторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* нет |
| *Основной поток:*   1. Пользователь нажимает кнопку «Настройки» 2. Система выводит страницу настроек трансляции 3. Пользователь переходит в пункт «Ключ» 4. Пользователь копирует уникальный ключ из соответствующего поля на странице 5. Если уникального ключа нет, то перейти в альтернативный поток 6. Пользователь запускает трансляцию через специальное приложение на компьютере используя уникальный ключ |
| *Постусловия:* Пользователь ведет трансляцию |
| *Альтернативные потоки:* 4 |

Таблица 4 – Спецификация варианта использования «Настройка трансляции»

|  |
| --- |
| ***UseCase:* Настройка трансляции** |
| *ID:* 4 |
| *Аннотация:* Пользователь производит настройку трансляции |
| *Главные актеры:* Авторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* нет |
| *Основной поток:*   1. Пользователь нажимает кнопку «Настройки» 2. Система выводит страницу настроек трансляции 3. Пользователь может перейти в один из пунктов «Стрим», «Ключ», «Чат» 4. В пункте «Стрим» можно сменить название и обложку стрима 5. Система изменить информацию о стриме в базе данных 6. В пункте «Ключ» можно сгенерировать уникальный ключ трансляции 7. Система сгенерирует уникальный ключ трансляции и выведет его в соответствующие поле на странице 8. В пункте «Чат» можно включить или выключить чат у трансляции 9. Система установит доступ к чату в соответствии с настройками |
| *Постусловия:* Произведена настройка трансляции |
| *Альтернативные потоки:* нет |

Таблица 5 – Спецификация варианта использования «Поиск трансляции»

|  |
| --- |
| ***UseCase:* Поиск трансляции** |
| *ID:* 5 |
| *Аннотация:* Пользователь производит поиск трансляции по названию |
| *Главные актеры:* Авторизованный пользователь, Неавторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* нет |
| *Основной поток:*   1. Пользователь вводит поисковый запрос и нажимает кнопку поиска 2. Система выдает перечень подходящих трансляций |
| *Постусловия:* Пользователь нашел перечень подходящих трансляций |
| *Альтернативные потоки:* нет |

Таблица 6 – Спецификация варианта использования «Просмотр трансляции»

|  |
| --- |
| ***UseCase:* Просмотр трансляции** |
| *ID:* 6 |
| *Аннотация:* Пользователь открывает трансляцию другого пользователя |
| *Главные актеры:* Авторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* нет |
| *Основной поток:*   1. Пользователь нажимает на карточку интересующего стрима 2. Система выводит страницу выбранной трансляции 3. Система загружает веб-проигрыватель |
| *Постусловия:* Пользователь находится на странице трансляции и смотрит стрим. |
| *Альтернативные потоки:* нет |

Таблица 7 – Спецификация варианта использования «Подписка на пользователя»

|  |
| --- |
| ***UseCase:* Подписка на пользователя** |
| *ID:* 7 |
| *Аннотация:* Пользователь подписывается на другого пользователя |
| *Главные актеры:* Авторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* Пользователь находится на странице трансляции |
| *Основной поток:*   1. Пользователь нажимает на кнопку «Подписаться» 2. Система создаёт новое отношение между двумя пользователями в базе данных. 3. Система выводит список отслеживаемых пользователей в боковом меню |
| *Постусловия:* Пользователь подписан на другого пользователя |
| *Альтернативные потоки:* нет |

Таблица 8 – Спецификация варианта использования «Блокировка пользователя»

|  |
| --- |
| ***UseCase:* Блокировка пользователя** |
| *ID:* 8 |
| *Аннотация:* Пользователь блокирует другого пользователя |
| *Главные актеры:* Авторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* Пользователь находится на странице трансляции |
| *Основной поток:*   1. Пользователь нажимает на кнопку «Заблокировать» 2. Система создаёт новое отношение между двумя пользователями в базе данных. 3. Система выводит не позволяет заблокированному пользователю перейти на вашу трансляцию |
| *Постусловия:* Один пользователь заблокирован другим пользователем |
| *Альтернативные потоки:* нет |

Таблица 9 – Спецификация варианта использования «Писать в чат»

|  |
| --- |
| ***UseCase:* Писать в чат** |
| *ID:* 8 |
| *Аннотация:* Пользователь пишет сообщение в групповой чат |
| *Главные актеры:* Авторизованный пользователь |
| *Второстепенные актеры:* нет |
| *Предусловия:* Пользователь находится на странице запущенной трансляции |
| *Основной поток:*   1. Пользователь пишет сообщение в форму и отправляет 2. Отправленное сообщение отображается в чате и становится видно всем зрителям трансляции |
| *Постусловия:* Пользователь отправил сообщение в чат |
| *Альтернативные потоки:* нет |

Приложение Б. Листинги компонентов

Листинг 1 – Компонент «StreamCard»

interface StreamCardProps {

stream: Stream & { user: User }

}

export const StreamCard = ({

stream

}: StreamCardProps) => {

return (

<div>

<Link href={`/${stream.user.username}`}>

<div className="h-fill w-full space-y-2">

<div className="group aspect-video relative rounded-md border-4 hover:border-indigo-500">

{stream.thumbnailUrl ? (

<Image

src={stream.thumbnailUrl}

alt={stream.name}

fill

className="object-cover "

/>

) : (

<div className="bg-muted flex flex-col items-center justify-center h-full w-full gap-y-4">

<UserAvatar

size="lg"

username={stream.user.username}

imageUrl={stream.user.imageUrl}

isLive={stream.isLive}

/>

</div>

)}

</div>

<div className="flex gap-x-3">

<UserAvatar

username={stream.user.username}

imageUrl={stream.user.imageUrl}

isLive={stream.isLive}

/>

<div className="flex flex-col overflow-hidden font-semibold">

<p>

{stream.name}

</p>

<p className="text-sm text-muted-foreground">

{stream.user.username}

</p>

</div>

</div>

</div>

</Link>

</div>

)

}

Листинг 2 – Компонент « getRecommended »

export const getRecommended = async () => {

let userId

try {

const self = await getSelf()

userId = self.id

} catch {

userId = null

}

let users = []

if (userId) {

users = await db.user.findMany({

where: {

AND: [

{

NOT: {

id: userId

}

},

{

NOT: {

followedBy: {

some: {

followerId: userId

}

}

}

},

{

NOT: {

blockedBy: {

some: {

blockerId: userId

}

}

}

},

{

stream: {

isLive: true

}

}

],

},

include: {

stream: true

},

orderBy: {

createdAt: "desc"

}

})

} else {

users = await db.user.findMany({

where: {

stream: {

isLive: true

}

},

include: {

stream: true

},

orderBy: {

createdAt: "desc"

}

})

}

return users

}

Листинг 3 – Компонент «Recommended»

interface RecommendedProps {

data: (User & {stream: Stream})[]

}

export const Recommended = ({

data

}: RecommendedProps) => {

return (

<div>

<div className="pl-6 mb-4">

{!!data.length &&

<p className="text-muted-foreground">

Рекомендуемые каналы

</p>

}

</div>

<ul className="space-y-2 px-2">

{data.map((user) => (

<UserItem

key={user.id}

username={user.username}

imageUrl={user.imageUrl}

isLive={user.stream?.isLive}

/>

))}

</ul>

</div>

)

}

Листинг 4 – Компонент «Actions»

interface ActionsProps {

isFollowing: boolean

isBlocked: boolean

hostId: string

isHost: boolean

}

export const Actions = ({

isFollowing,

isBlocked,

hostId,

isHost

}: ActionsProps) => {

const [isPending, startTransition] = useTransition()

const { userId } = useAuth()

const route = useRouter()

const handleFollow = () => {

startTransition(() => {

onFollow(hostId)

.then((data) => toast.success(`Вы подписались на ${data.following.username}`))

.catch(() => toast.error("Что-то пошло не так"))

})

}

const handleUnfollow = () => {

startTransition(() => {

onUnfollow(hostId)

.then((data) => toast.success(`Вы отписались от ${data.following.username}`))

.catch(() => toast.error("Что-то пошло не так"))

})

}

const toggleFollow = () => {

if (!userId) {

return route.push("/sign-in")

}

if (isHost) return

if (isFollowing) {

handleUnfollow()

} else {

handleFollow()

}

}

const handleBlock = () => {

startTransition(() => {

if (isFollowing) {

onUnfollow(hostId)

.catch(() => toast.error("Что-то пошло не так"))

}

onBlock(hostId)

.then((data) => toast.success(`Вы заблокировали ${data.blocked.username}`))

.catch(() => toast.error("Что-то пошло не так"))

})

}

const handleUnblock = () => {

startTransition(() => {

onUnblock(hostId)

.then((data) => toast.success(`Вы разблокировали ${data.blocked.username}`))

.catch(() => toast.error("Что-то пошло не так"))

})

}

const toggleBlock = () => {

if (!userId) {

return route.push("/sign-in")

}

if (isHost) return

if (isBlocked) {

handleUnblock()

} else {

handleBlock()

}

}

return (

<div className="flex gap-2">

<Button onClick={toggleBlock} variant="ghost" disabled={isPending || isHost}>

{isBlocked ? "Разблокировать" : "Заблокировать"}

</Button>

<Button onClick={toggleFollow} className="w-auto" variant="primary" disabled={isPending || isHost}>

{isFollowing ? "Отписаться" : "Подписаться"}

</Button>

</div>

)

}