**Министерство науки И ВЫСШЕГО образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М.Ф. Решетнева»**

Институт Информатики и телекоммуникаций

Направление 09.03.04 «Программная инженерия»

ПРОФИЛЬ Разработка программно-информационных систем

Кафедра Информатики и вычислительной техники

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Вид ВКР: **бакалаврская работа**

|  |
| --- |
| **Разработка веб-сервиса** |
| **для обучения звукорежиссуре** |

Обучающийся И.А. Терехин

инициалы и фамилия

Руководитель М.В. Дамов

инициалы и фамилия

Ответственный

за нормоконтроль А.В. Гуменникова

инициалы и фамилия

**Допускается к защите**

Заведующий кафедрой ИВТ М.Н. Фаворская

инициалы и фамилия

« » 2024 г.

Красноярск 2024

**Министерство науки И ВЫСШЕГО образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

УТВЕРЖДАЮ

Заведующий кафедрой ИВТ

М.Н. Фаворская

подпись инициалы, фамилия

«» 2024 г.

**ЗАДАНИЕ**

**на выпускную квалификационную работу (ВКР)**

**в форме бакалаврской работы**

Обучающийся Терехин Иван Алексеевич

Фамилия Имя Отчество

Группа БПИ20-02 Направление 09.03.04

«Программная инженерия»

Тема ВКР Разработка веб-сервиса для обучения звукорежиссуре

утверждена приказом по университету от « 15 » марта 2024 г. № 747д

Руководитель ВКР М.В. Дамов, к.т.н., доцент кафедры ИВТ

Инициалы, Фамилия, ученая степень, ученое звание, должность и место работы

Исходные данные для ВКР Теория и практика создания веб-сервисов по обучению звукорежиссуре, существующие системы по обучению звукорежиссуре

Перечень разделов ВКР Анализ особенностей работы сервисов по обучению звукорежиссуре, обзор применяемых технологий и методов, проектирование и реализация сервиса, разработка руководств программиста и пользователя, тестирование продукта

Перечень графического материала (с указанием обязательных чертежей при необходимости) Цель и задачи – 1 л., предметная область – 1 л., обзор аналогов – 2 л., диаграмма прецедентов – 1 л., структура разработанного сервиса – 3 л., варианты использования сервиса – 1 л., заключение – 1 л.

Срок сдачи обучающимся первого варианта ВКР 21.05.2024

Срок сдачи обучающимся окончательного варианта ВКР 10.06.2024

Руководитель ВКР М.В. Дамов

Задание принял к исполнению И.А. Терехин

« 08 » апреля 2024 г.

**АННОТАЦИЯ**

к бакалаврской работе

**«Разработка веб-сервиса для обучения звукорежиссуре»**

Терехин Иван Алексеевич

Ключевые слова: модульная система, звукорежиссура, *React*, *API*, микро сервисная архитектура.

В выпускной квалификационной работе рассматривается создание веб-сервиса по обучению звукорежиссуре. Разработанный сервис имеет возможность работать с аудиофайлами, что позволяет каждому желающему использовать аудиоматериалы для обучения. В работе рассматриваются принципы создания подобных систем, описывается архитектура и интерфейс взаимодействия модулей с системой.

Работа включает: 30 страниц, 2 таблицы, 23 рисунка, 4 листинга. Использованных источников – 17.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc169013373)

[1 Анализ особенностей работы сервисов по обучению звукорежиссуре 7](#_Toc169013374)

[1.1 Обзор сервисов по обучению звукорежиссуре 7](#_Toc169013375)

[1.2 Обзор функциональных особенностей сервисов по обучению звукорежиссуре 11](#_Toc169013376)

[1.3 Выводы по главе 11](#_Toc169013377)

[2 ОБЗОР ПРИМЕНЯЕМЫХ ТЕХНОЛОГИЙ, МЕТОДОВ И АЛГОРИТМОВ 12](#_Toc169013378)

[2.1 Технологии для реализации *backend*-части 12](#_Toc169013379)

[2.2 Технологии для реализации *REST API* 13](#_Toc169013380)

[2.3 Технологии для реализации *fronted*-части 14](#_Toc169013381)

[2.4 Обзор методов и алгоритмов 15](#_Toc169013382)

[2.5 Проектирование базы данных 16](#_Toc169013383)

[2.6 Выводы по главе 17](#_Toc169013384)

[3 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 18](#_Toc169013385)

[3.1 Структура программного продукта 18](#_Toc169013386)

[3.2 Реализация модульной системы 19](#_Toc169013387)

[3.3 Руководство программиста 20](#_Toc169013388)

[3.4 Краткое руководство пользователя 21](#_Toc169013389)

[3.5 Тестирование программного продукта 27](#_Toc169013390)

[3.6 Выводы по главе 28](#_Toc169013391)

[ЗАКЛЮЧЕНИЕ 29](#_Toc169013392)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc169013393)

[ПРИЛОЖЕНИЕ А 32](#_Toc169013394)

# ВВЕДЕНИЕ

*Актуальность*. В современном мире, где все больше групп и частных лиц связывают свою жизнь с творческой сферой, а именно написанием музыки или игрой на музыкальных инструментах, все больше требуется обучение людей звукорежиссуре и принципам написания своих песен или аранжировок на чьи-то композиции. Существующие на рынке решения (как очные занятия в музыкальных школах или кружках, так и курсы обучения звукорежиссуре) не всегда удовлетворяют потребности потребителей, что открывает возможности для создания новых приложений, предоставляющих более удобный и эффективный способ обучения звукорежиссуре и написания своих композиций.

Разработка веб-сервиса для обучения звукорежиссуре имеет большой потенциал для успешного запуска на рынок. Кроме того, веб-сервис для обучения звукозаписи и написанию собственных композиций позволит упростить процесс обучения и сделать его более удобным, что повысит эффективность работы и снизит вероятность ошибок.

Следовательно, разработка нового веб-сервиса для обучения принципам звукорежиссуры и написанию собственных композиций является актуальной задачей, такой веб-сервис позволит пользователям удобно и эффективно обучаться принципам звукозаписи и написанию собственных композиций.

*Цель и задачи.* Цельюбакалаврской работы является обучение пользователей основам звукорежиссуры.

Для достижения поставленной цели необходимо решить следующие задачи:

* осуществить анализ и обзор существующих приложений и сайтов по обучению звукозаписи и написанию собственных композиций;
* выбрать средство для реализации программы;
* выполнить концептуальное проектирование базы данных;
* осуществить логическое проектирование базы данных;
* осуществить физическое проектирование базы данных;
* разработать структуру программы;
* разработать механизм защиты;
* осуществить программную реализацию;
* разработать отчеты и графики;
* сформировать руководства программиста и пользователя;
* провести тестирование разработанного программного продукта.

*Структура работы.* Пояснительная записка к бакалаврской работе состоит из введения, трех глав, заключения и списка использованных источников из 17 наименований. Изложена на 30 страницах и содержит 23 рисунка и 2 таблицы.

В первой главе бакалаврской работы приводится обзор существующих программных продуктов (ПП) по обучению звукорежиссуре. Разобраны основные особенности ПП по обучению звукорежиссуре. Сформулированы требования к разрабатываемой системе.

Во второй главе описан процесс выбора технологий, необходимых для реализации сервиса. Произведен обзор методов и алгоритмов реализации сервиса по обучению звукорежиссуре.

В третьей главе описан процесс разработки структуры программного продукта. Также в этой главе приведены руководства программиста и пользователя. Отражены результаты тестирования разработанного веб-сервиса.

# 1 Анализ особенностей работы сервисов по обучению звукорежиссуре

Целью выпускной квалификационной работы является создание сервиса по обучению звукорежиссуре. Такие сервисы обычно называют онлайн курсами. Онлайн курсы, как и музыкальные школы, сводят межу собой учеников и преподавателей.

Онлайн-курсы сильно упрощают процесс поиска преподавателя и информации для обучения. Они предоставляют возможность в одном месте просматривать уроки, обучаться, тестировать свои знания и работать с мультимедийными файлами.

Работникам данные сервисы также упрощают жизнь. Им не нужно постоянно искать клиентов для занятий, ездить в другой конец города до ученика, если занятия выездные. Нет необходимости регистрироваться на локальных площадках с досками объявлений.

В связи с тем, что данные сервисы столь удобны и имеют большую целевую аудиторию, их создано достаточно много. Далее будет рассмотрены наиболее популярные из них.

## 1.1 Обзор сервисов по обучению звукорежиссуре

Для определения необходимой функциональности сервиса, необходимо изучить сайты и приложения, которые решают схожую задачу. Также необходимо выявить сильные и слабые стороны конкурентов, чтобы впитать все лучшее в разрабатываемый сервис.

Аналогов разрабатываемого продукта множество, но большая часть сервисов копируют один-в-один своих старших собратьев. Среди самых крупных и интересных сервисов можно выявить следующие: *SkillBox*, Школа профессий, *Netology.*

### Сайт «SkillBox»

Среди сервисов для обучения разным навыкам и профессиям лидирует [9] *SkillBox.*

Однако, не смотря на большую популярность, данный сервис имеет ряд проблем со стороны качества информации:

для доступа к курсу и обучению по нему, потенциальному клиенту нужно заплатить большую сумму [18]: «Чтобы приобрести курс обучения по направлению «Введение в звукорежиссуру» нужно оплатить доступ и выходит это на сумму 87.970 рублей»;

также пользователи данного сервиса отмечают, что информационная база либо не актуальна, либо не полная: «Информация сильно сжата. Редкая связь с преподавателями. В бесплатных источниках информации по этой теме предоставляется не меньше. Пример: *ZuubofM*.».

Как итог, данный сервис хоть и имеет большую популярность среди российских пользователей, но он не лишен существенных минусов: очень дорогое обучение, не всегда обширная база информации у курсов обучения. Список навыков, приобретённых после прохождения курса и интерфейс сервиса предоставлен на рисунке 1.1.

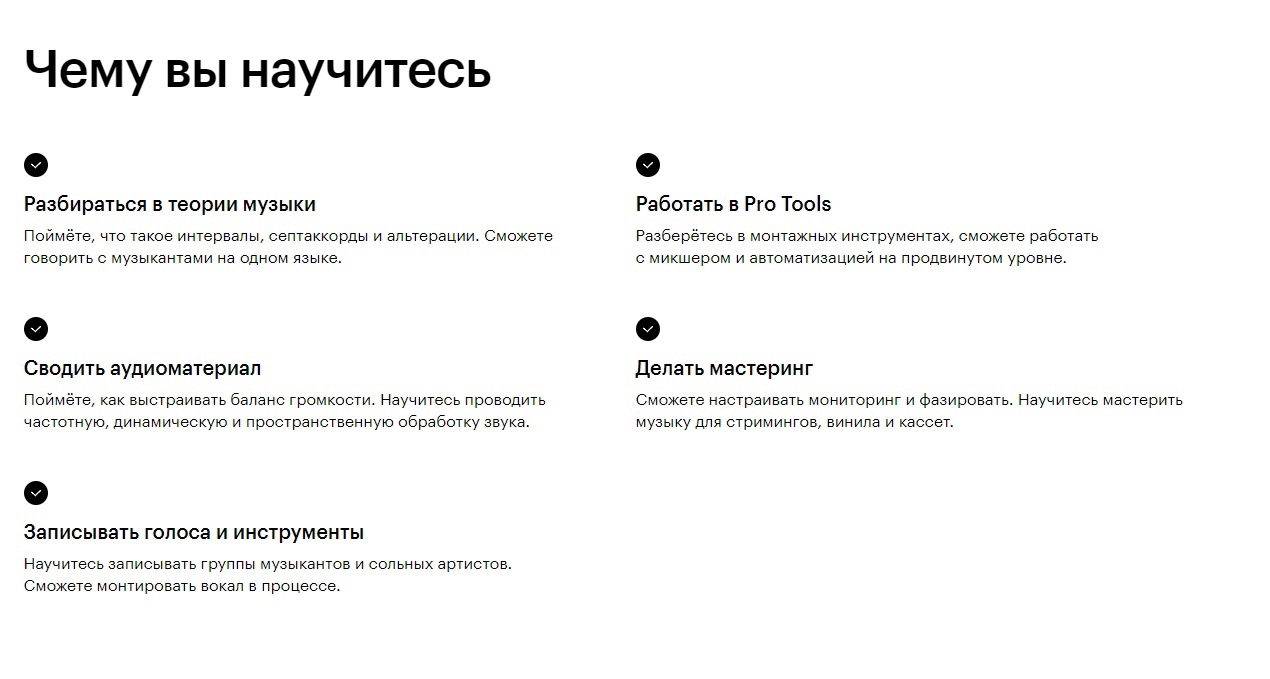


Рисунок 1.1 – Интерфейс *SkillBox*

### Сервис «Школа профессий»

Школа профессий является популярной программой обучения. Сервис направлен на повышение обоснованности профессионального самоопределения обучающихся с целью обеспечения профессиональной мобильности и профессионального развития будущих кадров.

Для доступа к курсу и обучению по нему, потенциальному клиенту нужно заплатить большую сумму. Чтобы приобрести курс обучения по направлению «Саунд Дизайн» нужно оплатить доступ и выходит это на сумму 28.800 рублей. Из дополнительных минусов данного сервиса – это отсутствие онлайн чата для общения между пользователями и учителями на прямую.

Однако, несмотря на описанные минусы, сервисом пользоваться крайне приятно из-за весьма удачного дизайна (Рисунок 1.2).

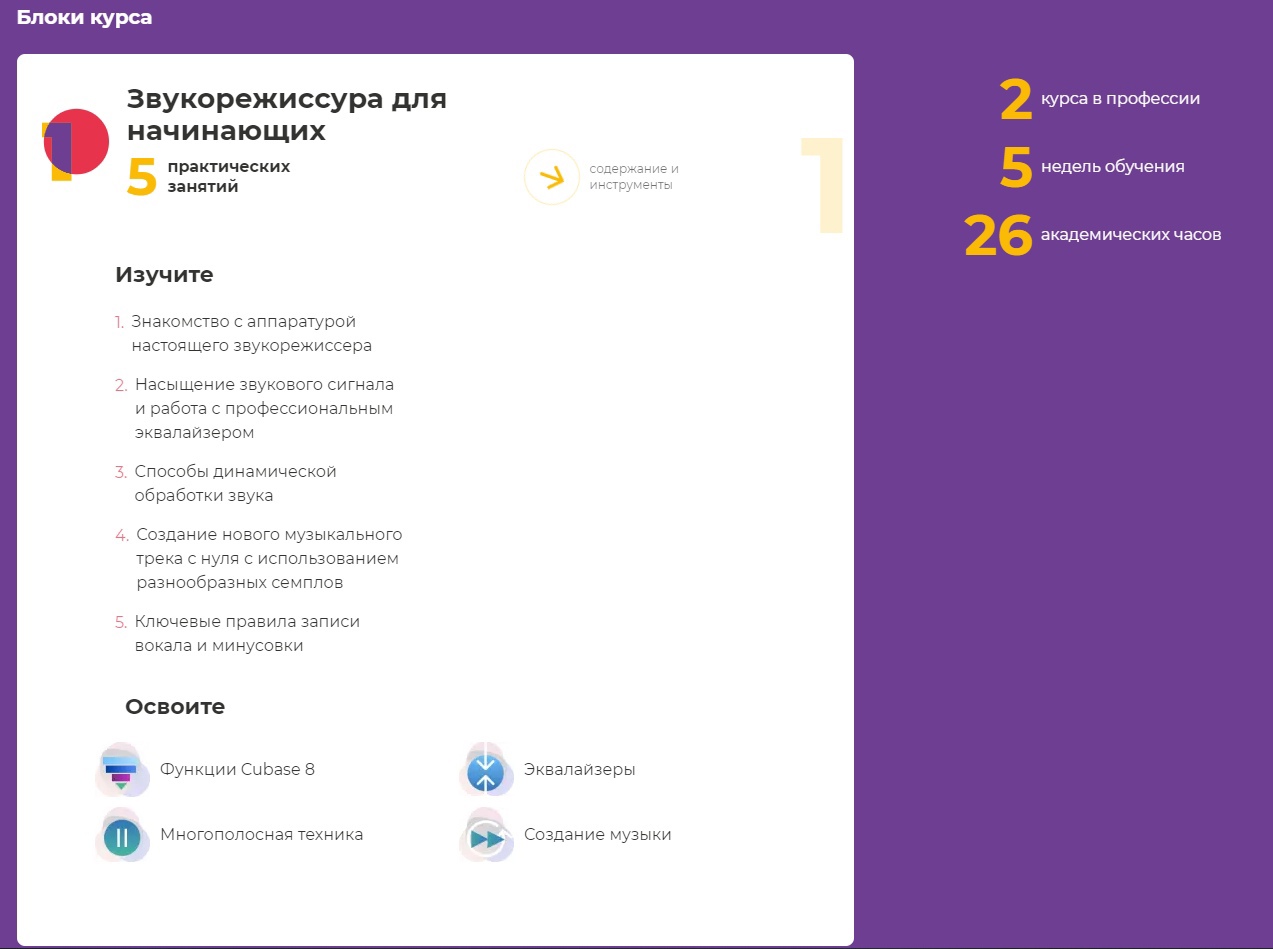


Рисунок 1.2 – Интерфейс и блоки курса *Школа профессий*

### Сайт «Netology»

*Netology* [17] – это онлайн-сервис, образовательная онлайн-платформа. Как заявляют создатели этого сервиса, Нетология – лучшая образовательная платформа по доле выпускников, довольных обучением. Сервис обещает вернуть деньги за обучение, если обучение не подойдёт, так же существует возможность оплаты обучения в рассрочку и частями.

Для доступа к курсу и обучению по нему, потенциальному клиенту нужно заплатить большую сумму: «Чтобы приобрести курс обучения по направлению «Саунд Дизайн» нужно оплатить доступ и выходит это на сумму 135.560 рублей»; пожалуй, это самый дорогой сервис по обучению звукорежиссуре из вышеперечисленных.

Однако у данного сервиса хорошие отзывы о предоставляемой информационной базе. Пользователи отмечают, что информации много и она чётко структурирована.

Как итог, данный сервис имеет свои преимущества в виде информационной базы и структурированности. Программа обучения большая и долгая, что является существенным плюсом (Рисунок 1.3). Однако также имеет и недостаток в дороговизне обучения относительно других аналогов.

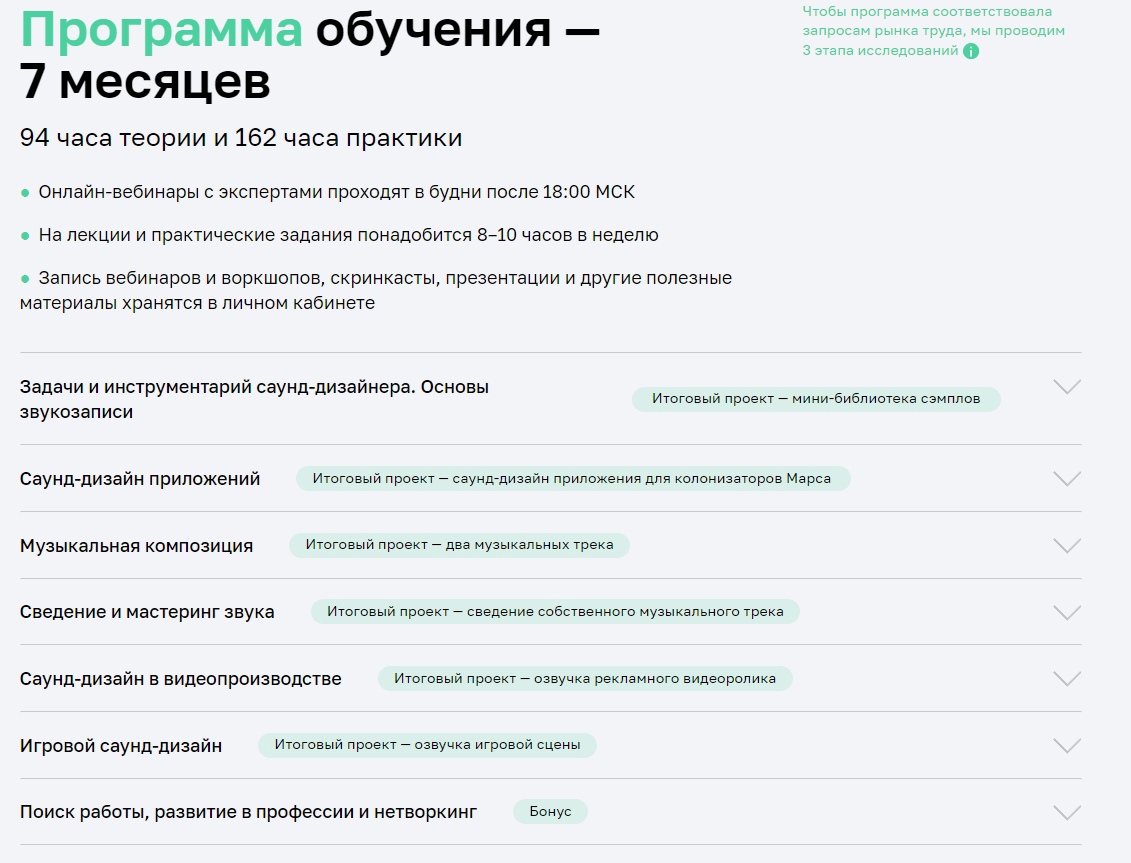


Рисунок 1.3 – Интерфейс и программа обучения *Netology*

### Результат сравнения сервисов

В таблице 1.1 приведен результат сравнительного анализа сервисов по обучению звукорежиссуре.

Таблица 1.1 – Сравнительный анализ сервисов по обучению звукорежиссуре

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | *SkillBox* | Школа профессий | *Netology* |
| Аудио/ видеоматериалы по обучению | Присутствует | Присутствует | Присутствует |
| Общение с другими пользователями | Присутствует | Отсутствует | Присутствует |
| Наличие тестов об изученном материале | Присутствует | Присутствует | Присутствует |
| Наличие личного кабинета | Присутствует | Присутствует | Присутствует |
| Возможность работы с аудиофайлами | Частично присутствует | Отсутствует | Частично присутствует |
| Добавление выбранных статей себе в личный кабинет | Отсутствует | Отсутствует | Присутствует |

На основе проведенного анализа можно сделать вывод, что при разработке стоит сделать упор на обширную информационную базу. Также необходимо проработать тесты на успешность усвоенного материала.

Так как характеристики данных сервисов очень схожи, то было решено реализовать весь функционал из сравнения в собственном веб-сервисе.

## 1.2 Обзор функциональных особенностей сервисов по обучению звукорежиссуре

Сервисы по обучению звукорежиссуре в большинстве своем представляют огромную базу статей и видеоуроков по музыкальной теории. Такие обучающие материалы в систему могут как добавляться вручную, так и агрегировать из различных источников.

Пользователь подобной системы должен иметь возможность каким-либо образом получать информацию о музыкальной теории, устройстве микрофонов, сведение, мастеринге и т.д. В большинстве сервисов это делается посредством различных статей и видеоуроков, находящихся на собственных базах данных.

Также большинство сервисов имеют различные тесты знаний, где пользователям нужно выбрать 1 верный ответ на вопрос из 3 или 4 вариантов. Такие тесты помогают понять, насколько хорошо или плохо был изучен материал на этом обучающем сервисе или же как уровень знаний пользователь имеет до прохождения материала.

Помимо этого, большим плюсом является возможность создания собственного аккаунта. Так, зарегистрировав себе аккаунт на данном сервисе, пользователь может добавлять отдельные материалы из курса себе в личных кабинет, делать заметки и при возможности менять личную информацию о себе.

Таким образом, разрабатываемый сервис должен иметь возможность добавления интересующих пользователя курсов себе в личный кабинет. Позволять управлять этими курсами, а именно добавлять и убирать их из личного кабинета. А также сервис должен предоставлять возможность онлайн общения в чате с другими пользователями и тестирования своих знаний, с помощью итогового теста по теме «звукорежиссура».

## 1.3 Выводы по главе

В первой главе была рассмотрена предметная область, проанализированы существующие аналоги разрабатываемого веб-сервиса по обучению звукорежиссуре, а также его функциональные особенности. Опираясь на данные результаты, будет проводиться дальнейшая разработка и анализ технологий, применяемых в проекте.

# 2 ОБЗОР ПРИМЕНЯЕМЫХ ТЕХНОЛОГИЙ, МЕТОДОВ И АЛГОРИТМОВ

Разрабатываемый программный продукт будет состоять из 3 частей:

*backend*-часть сервиса;

*frontend*-часть сервиса;

*REST API* для подключения модулей.

## 2.1 Технологии для реализации *backend*-части

Данная часть приложения отвечает за всю бизнес-логику сервиса, она манипулирует всеми данными и маршрутизирует запросы между модулями.

Для реализации серверной части существуют следующие варианты:

*Django* [12] – свободный фреймворк для веб-приложений на языке *Python*, использующий шаблон проектирования *MVC*. Сайт на *Django* строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других;

*Flask* [14] – [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для создания веб-приложений на языке программирования [*Python*](https://ru.wikipedia.org/wiki/Python), использующий набор инструментов *Werkzeug*, а также шаблонизатор *Jinja*2. Относится к категории так называемых микро фреймворков – минималистичных каркасов веб-приложений, сознательно предоставляющих лишь самые базовые возможности;

*Node.js* [16] – это среда выполнения *JavaScript*-программ/программная библиотека, построенная на движке *Chrome V*8. Раньше на *JS* можно было делать только интерактивные сайты, так как это специализированный браузерный язык программирования. С программной платформой *Node.js* появилась возможность создавать «безбраузерные» приложения для компьютера. Это значит, что при необходимости на *JavaScript* теперь можно делать то же, что на *Python* и других скриптовых языках программирования общего назначения.

Учитывая, что *Node.js*, с минимальными усилиями со стороны программиста, позволяет быстро разработать надежную и легко поддерживаемую серверную часть, эта среда выполнения была выбрана в качестве основной для разработки. Документацию *Node.js* можно увидеть на рисунке 2.1.

Для обеспечения сохранности данных будут применяться алгоритмы хеширование *SHA*256 для паролей. Хеширование паролей необходимо для того, чтобы в случае утечки базы данных злоумышленникам было сложнее разгадать пароль. Помимо технологии *bcrypt* к хэшу пароля добавляет соль, чтобы усложнить подбор пароля по хэшам.

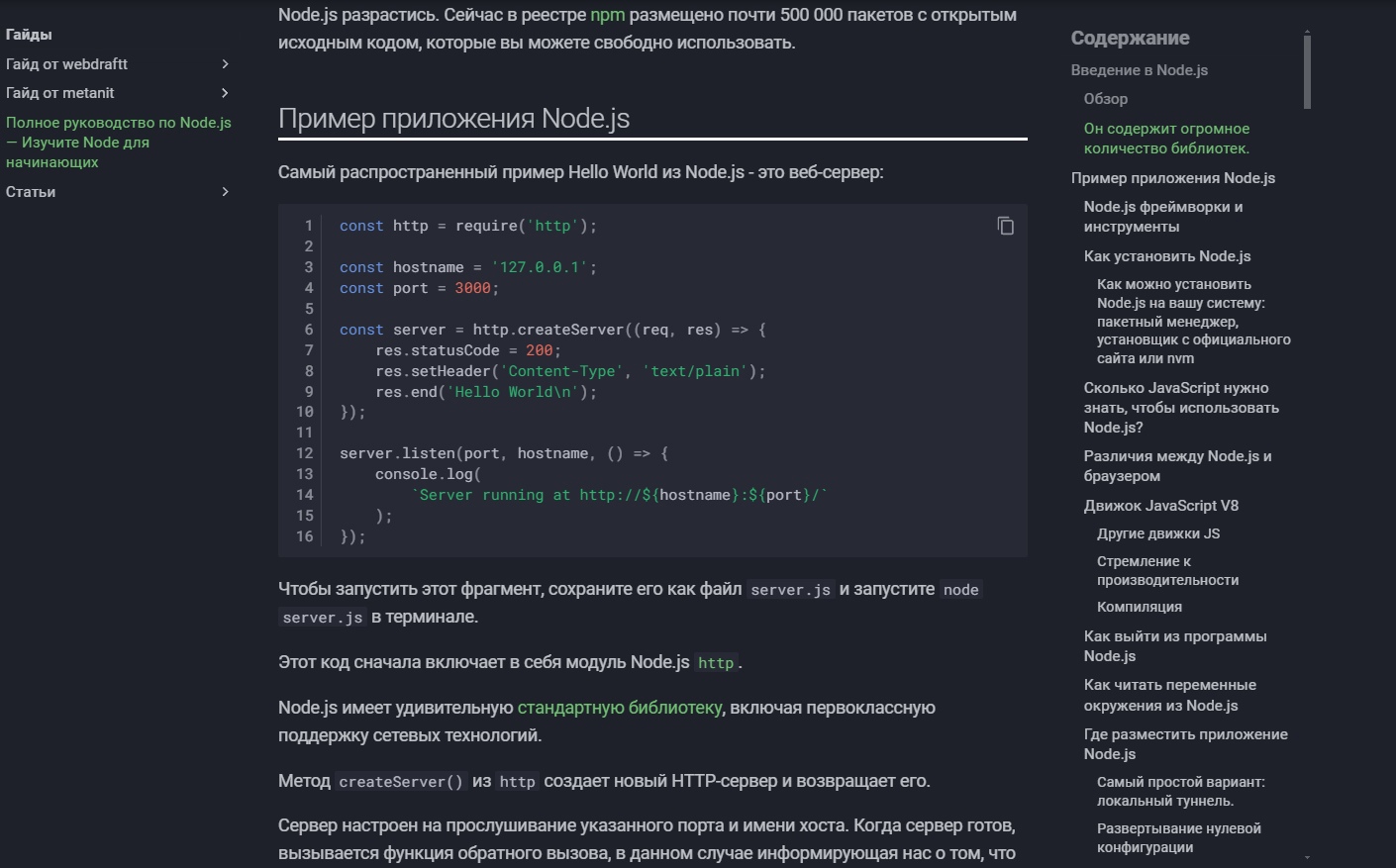


Рисунок 2.1 – Интерфейс документации *Node*.*js*

В качестве основной базы данных был выбран *MongoDB*, т.к. эта СУБД хорошо справляется с нагрузками и является неким стандартном при разработке приложений с использованием технологии *MERN*.

## 2.2 Технологии для реализации *REST API*

Для того, чтобы доступ к данным был не только в рамках *backend* части сервиса, необходимо сделать доступ через *REST API*. Для реализации можно напрямую подключаться в БД, используя один из следующих инструментов:

– *Spring Boot* [8] – *Java*-фреймворк, предоставляющий удобные функции для построения веб-приложений и *REST API.* Может выдерживать высокую нагрузку на относительно слабых серверах;

– *FastAPI* [14] – веб-фреймворк, написанный на языке программирования *Python*. Из плюсов можно отметить, что он позволяет создавать документацию к *API*, по ходу написания кода, что сильно экономит время. А также он имеет реализованную логику для авторизации запросов.

Если использовать один из вышеперечисленных фреймворков, то придется заново создавать слой логики для взаимодействия с базой данных на языке отличном от *JavaScript*. Поэтому лучше пойти другим путем и разработать *REST API* на основе *Express* [10]. Это позволит переиспользовать логику взаимодействия с моделями базы данных, а также логику авторизации запросов. Отправку запросов (*post/get*) через приложения для тестирования *CRUD* можно увидеть на рисунке 2.2 в приложении *Insomnia*.

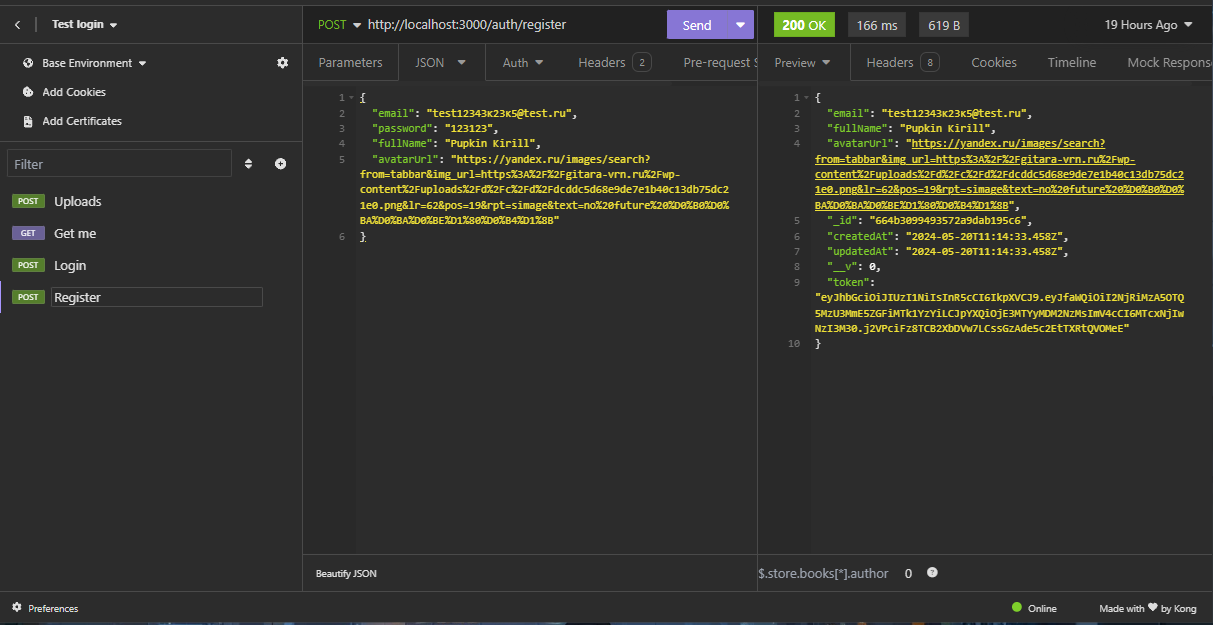


Рисунок 2.2 – Интерфейс *Insomnia*

Предполагается, что для обеспечения безопасного хранения данных будет использоваться база данных *MongoDB*, доступ к которой будет только через *REST API* для авторизированных запросов.

## 2.3 Технологии для реализации *fronted*-части

Для разработки *fronted*-части существует большое количество фреймворков. Некоторые из них позволяют создавать исключительно сайты, на основе других без особого труда можно создавать кроссплатформенные приложения:

– *React* [11] – *JavaScript*-фреймворк с открытым исходным от компании *Facebook.* Компонентой подход, используемый в данном фреймворке, значительно ускоряет разработку;

*– Angular* [7] – чуть более старший собрат *React*, основанный компанией *Google*. Отличается от конкурентов большей плавностью и большей производительностью. Однако, в отличие от предыдущего фреймворка, он является более сложным в освоении из-за необходимости знать *TypeScript*;

– *Vue* [14] – пожалуй, один из самых легких *JS*-фреймворков. Имеет большое количество компонентов и прост в освоении.

Учитывая, что в *React*, использует компонентный подход и часто используется при разработки сервисов с технологией *MERN* (*mongo/express/react/node.js*), этот фреймворк был выбран в качество основного для разработки.

*React* предоставляет *DOM*, который позволяет эффективно обновлять и рендерить пользовательские интерфейсы, управлять состоянием и жизненным циклом компонентов, использовать виртуальный *DOM* для повышения производительности, легко интегрироваться с другими библиотеками и фреймворками, а также создавать многоразовые и тестируемые компоненты.

## 2.4 Обзор методов и алгоритмов

Для более точного понимания будущих работ, необходимо определить методы и алгоритмы для следующих частей разрабатываемого сервиса:

– взаимодействие клиентской и серверной частей приложения;

– алгоритм хэширования паролей;

– алгоритм *JWT*.

Определив все вышеперечисленные пункты, можно будет приступать к разработке данного сервиса.

### Взаимодействие клиентской и серверной частей приложения

В качестве основного метода разработки была выбрана клиент серверная архитектура, идеально подходящее для реализации выбранной темы. Схематичное описание клиент серверной архитектуры представлено на рисунке 2.3.

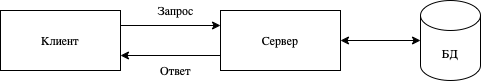


Рисунок 2.3 – Схематичное представление клиент-серверной архитектуры

На клиент-серверной архитектуре построены все сайты и интернет-сервисы [7]. Также ее используют десктоп-программы, которые передают данные по интернету.

Клиент работает с пользователем. Он нужен для того, чтобы превращать байты программного кода в понятный и простой интерфейс, потому что пользователь непрограммист и не понимает языки программирования.

Сервер нужен для того, чтобы обрабатывать запросы от множества клиентов. В нашем случае, на получение сконструированного сайта. Сервер, получая запросы, обращается к базе данных и, следуя бизнес-логике, формирует ответы на эти запросы.

При таком подходе к разработке практически отсутствует дублирование кода, так как логика находится в одном месте – на сервере. Кроме того, он безопаснее, потому что со стороны пользователя нет доступа к приватным данным.

Для достижения еще большей безопасности, на сервере есть промежуточные обработчики запросов, которые проверяют отправляемый в запросе токен пользователя, а также приватный *API key*. Если что-то из этого не является верным, тогда пользователь не сможет получить доступ к данным.

Соединение с сервером будет происходить по протоколу *HTTPS*, которые имеет встроенный метод шифрования передаваемых данных.

### Алгоритм хэширования паролей

*Bcrypt* – адаптивная криптографическая хеш-функция формирования ключа, используемая для защищенного хранения паролей.

Использование *bcrypt* для хеширования паролей перед их сохранением в базе данных. Алгоритм хэширования паролей является ключевым компонентом безопасности пользовательских данных. Его основная задача заключается в том, чтобы защитить пароли пользователей от несанкционированного доступа. При помощи этого алгоритма пароль пользователя преобразуется в некий уникальный хэш, который затем хранится в базе данных. При попытке входа в систему пароль вводится и сравнивается с хэшем, хранящимся в базе. Даже если злоумышленник получит доступ к хранимым хэшам, ему будет крайне сложно восстановить оригинальные пароли.

### Алгоритм JWT

Алгоритм *JWT* (*JSON Web Token*) используется для безопасной передачи информации между сторонами в формате *JSON*. Он позволяет создавать токены, содержащие утверждения о пользователе, которые могут использоваться для авторизации и аутентификации. *JWT* имеет несколько преимуществ, включая компактность, надежность и возможность проверки подлинности и целостности данных. Кроме того, *JWT* поддерживает механизмы шифрования, что обеспечивает дополнительный уровень безопасности при передаче информации.

## 2.5 Проектирование базы данных

База данных является одной из важнейших частей системы. От правильности ее проектирования зависит и быстродействие всей системы, и сохранность данных.

Для начала необходимо выделить основные сущности и связи между ними. Такую концептуальную структуру можно отобразить при помощи *ER*-диаграммы. Она представлена на рисунке 2.4.

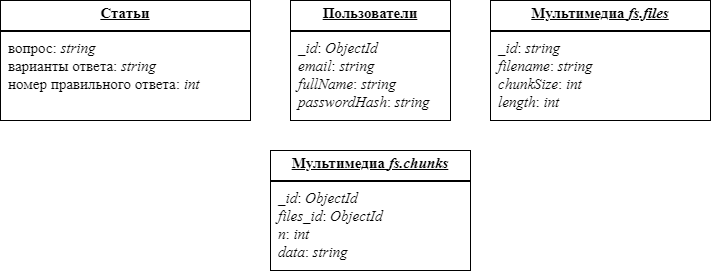


Рисунок 2.4 – *ER*-диаграмма базы данных

В данной диаграмме представлены все таблицы, необходимые для функционирования сервиса. Так как *MongoDB* считается *NoSQL* базой данных, то таблицы не связаны между собой и хранят в себе json объекты.

## 2.6 Выводы по главе

Во второй главе был приведен обзор технологий для разработки *backend,* *frontend* частей приложения и *RESP API* для взаимодействия модулей. После этого был выбран технологический стек для разработки данного сервиса. Кроме того, были рассмотрены методы и алгоритмы реализации сервиса по обучению звукорежиссуре, описан способ взаимодействия модулей с сервисом.

# 3 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

## 3.1 Структура программного продукта

Разработанный программный продукт представляет из себя сервис обучения звукорежиссуре, который включает в себя:

­– *backend*, реализованная с помощью *Node.js*;

– *REST API,* реализованное с помощью *Express*;

– *frontend*, реализованный на *React*;

– набор модулей для авторизации и загрузки мультимедийных файлов.

Схема взаимодействия частей сервиса предоставлена на рисунке 3.1.

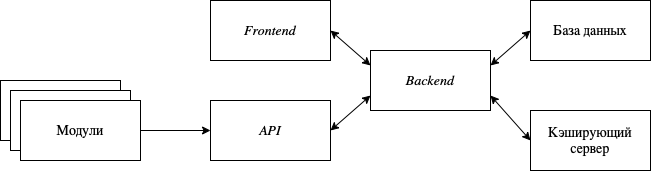


Рисунок 3.1 – Схема взаимодействия частей сервиса

Далее будут рассмотрены все модули программного продукта, а также их структура.

### Разработка frontend-части

*React* приложения используют структуру *DOM* [11]. Подобную структуру использует и клиентская часть.

Пользователь взаимодействует с сайтом, создаваемым посредством компонентного подхода. Данные из компонентов передаются в формы, страницы. Через импорт компонентов мы можем избежать переиспользования одинаковых частей веб-сервиса, кода в целом. *React* с подключенным *Router* и его функциями *Link* позволяет сделать веб-сервис *SPA* (*single page application*), в котором при переходе по вкладкам, другим страницам сайта, браузер не будет заново загружать (рендерить) страницу браузера, а будет динамически, практически моментально менять страницу на ту, куда перейдёт пользователь.

### Разработка серверной части

На рисунке 3.2 приведена схематичная структура *backend*-части сервиса. Структура описывает, как именно происходит обращение к данным при получении запроса от клиентской части.

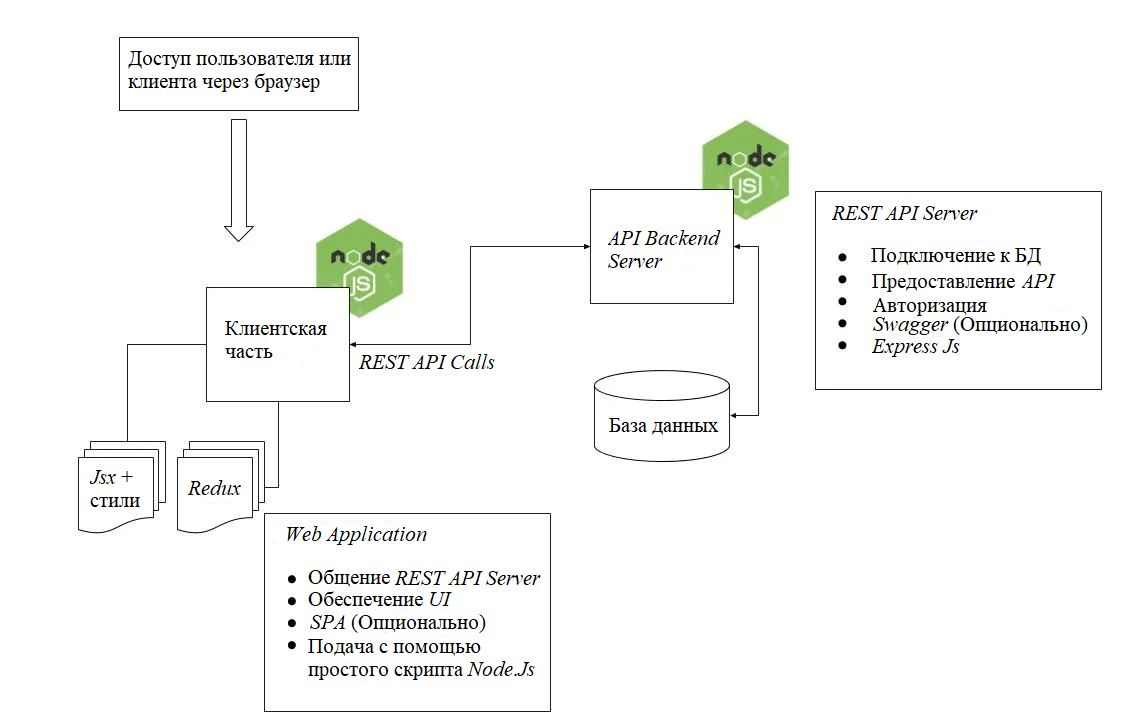


Рисунок 3.2 – Схема структуры *backend-*части

Как видно, приложение получает данные из *frontend* части. Далее *Rest API Calls* связывает клиентскую часть с *API* серверной части. *API Backend Server* подключена напрямую к БД. Клиентская часть веб-сервиса содержит *Jsx* файлы, включающие в себя *Html* вместе с *Javascript* кодом. *Redux* используется для предсказуемого и поддерживаемого управления глобальным состоянием*.* Так для написания логики регистрации и авторизации в веб-сервисе был использован *Redux* для управления глобальным состоянием авторизации*.*

## 3.2 Реализация модульной системы

Как уже обозначалось выше, в сервисе реализована система модулей для авторизации и загрузки мультимедийных файлов. Модули общаются с основным сервисом посредством *REST API*. Несмотря на то, что существует 3 вида модулей, решающих разные задачи, интерфейс взаимодействия у всех модулей одинаковый.

### Авторизация запросов в системе

Для того, чтобы система знала от какого пользователя, поступают запросы, в *API* сервиса была реализована авторизация. Авторизация нужна абсолютно для всех запросов и представляет из себя дополнительный заголовок с токеном пользователя.

Для получения токена пользователя, необходимо отправить *POST*-запрос на адрес /*authi*/*login*/. В теле запроса необходимо передать *JSON* с логином и паролем пользователя (Листинг 3.1).

Листинг 3.1 – Получение токена пользователя

{

"username": "w0rng",

"password": "1"

}

В ответ на запрос сервер отправит токен пользователя (Листинг 3.2), который необходимо будет отправлять во всех запросах.

Листинг 3.2 – Ответ от сервера при авторизации

{

"key": "06b2f0350a1758272ac4f3b67726d491c9d2f25d",

"created": "2022-05-19T14:49:02.290244+07:00"

}

Если по какой-то причине необходимо сбросить токен пользователя, например, чтоб злоумышленники не могли его использовать в случае утечки данных, пользователь может его сбросить. Для этого реализован метод автоматического обновления токена раз в час, так пользователю не придётся думать об этом, а сервис сделает это за него. Также, при выходе из аккаунта в локальном хранилище в браузере токен удаляется.

#### Загрузка мультимедийных файлов

Для загрузки мультимедийных файлов на сервер, таких как аудиодорожки и фотографии, необходимо отправить *POST*-запрос на адрес /*upload*/. В теле запроса необходимо передать название файла и сам файл.

## 3.3 Руководство программиста

Весь проект написан на языке *JavaScript* в интегрированной среде разработки *Visual Studio* 2022. В качестве менеджера зависимостей был использован *npm*.

Самый удобный вариант разработки – через *Visual Studio* 2022.

Для этого необходимо проделать следующие шаги:

1. Установить *MongoDB Compass*.
2. Отредактировать файл *index.js* и *db.js*, добавив в него параметры для подключения к базе данных.
3. Установить пакетным менеджер *npm/yarn/pnpm* и зависимости проекта командой *npm install (extensions)* && *pnpm install* (*extensions*).
4. Проверить наличие *script* на запуск программного продукта*.*

После проделанных действий можно запустить проект командой *npm run dev из папки* \*frontend*\*musicEd* и *npm run start*:*dev* для серверной части приложения из папки \*backend*\. При редактировании кода через любой редактор приложение будет автоматически обновляться.

Если все сделано правильно, при открытии в браузере страницы *localhost* будет показана главная страница проекта.

Для того, чтобы редактировать проект, необходимо перейти в папку проекта и в папку *src*. Для изменения логики приложения и набора компонентов вы можете редактировать файлы с расширением .*js* и.*jsx*, каждый из которых отвечает за свой конкретный компонент. Для редактирования страниц сайта необходимо изменять файлы .*jsx* в каталоге \*frontend*\*musicEd*\*src*\*pages*.

Системные требования к составу и параметрам технических средств:

процессор *Intel* *Core I*3/ *AMD Ryzen* 3или лучше;

оперативная память не меньше 8 Гб;

не менее 30 Гб свободного места на жестком диске;

постоянный канал связи с глобальной сетью «Интернет» со скоростью соединения не менее 5 Мб/с.

Требования к информационной и программной совместимости:

операционная система *Windows* 8/10, *MacOS* или *Linux*;

браузер *Google Chrome* 96.0.4664 и выше*, Mozilla Firefox* 91 и выше*, Safari* 13 и выше или другие.

## 3.4 Краткое руководство пользователя

Когда пользователь заходит на сайт впервые, ему предлагается войти в существующую учетную запись или зарегистрироваться (Рисунок 3.3).

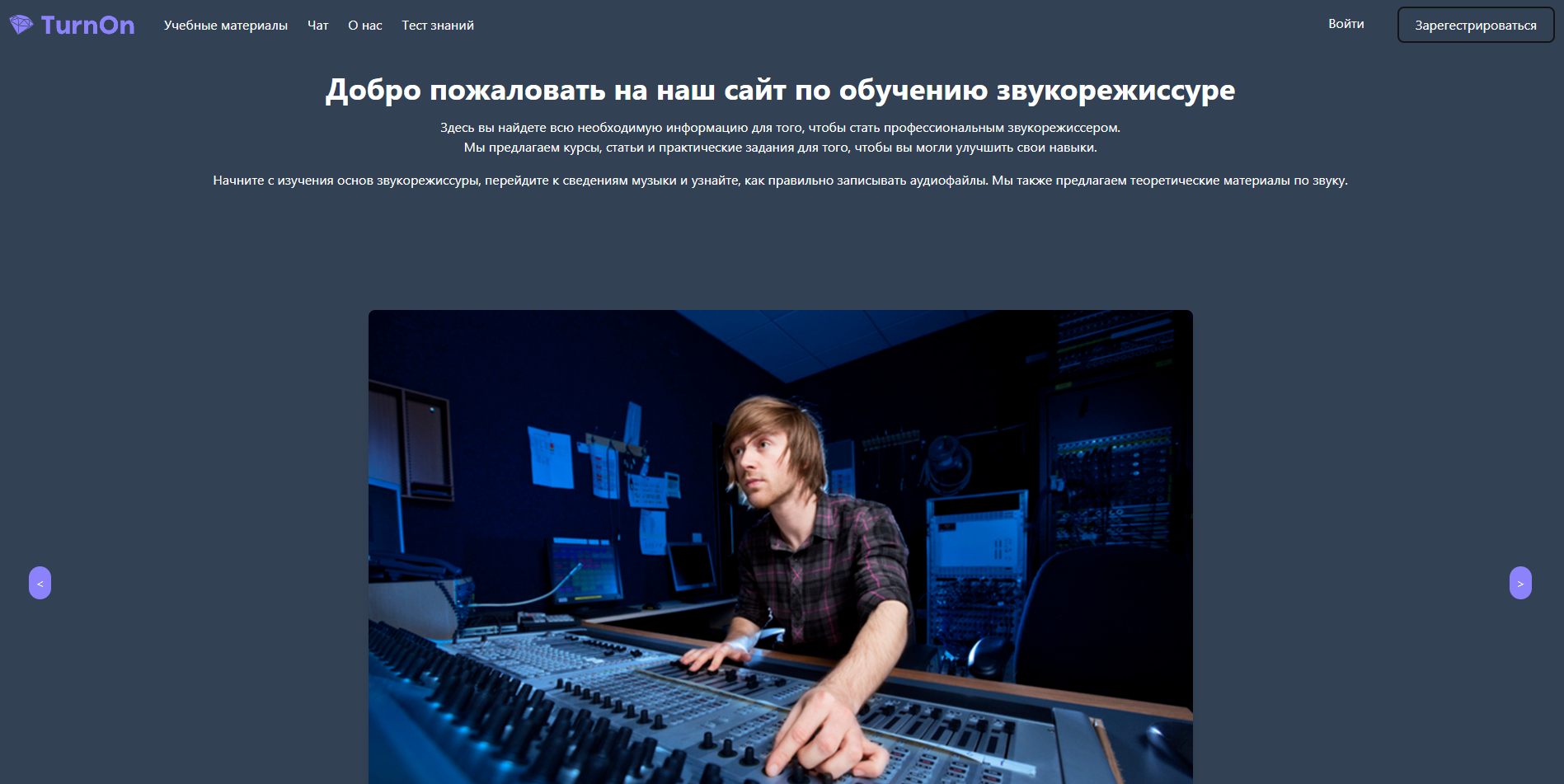


Рисунок 3.3 – Главная страница сервиса

При нажатии на кнопку «Зарегистрироваться», пользователю предлагается заполнить не сложную форму регистрации (Рисунок 3.4).

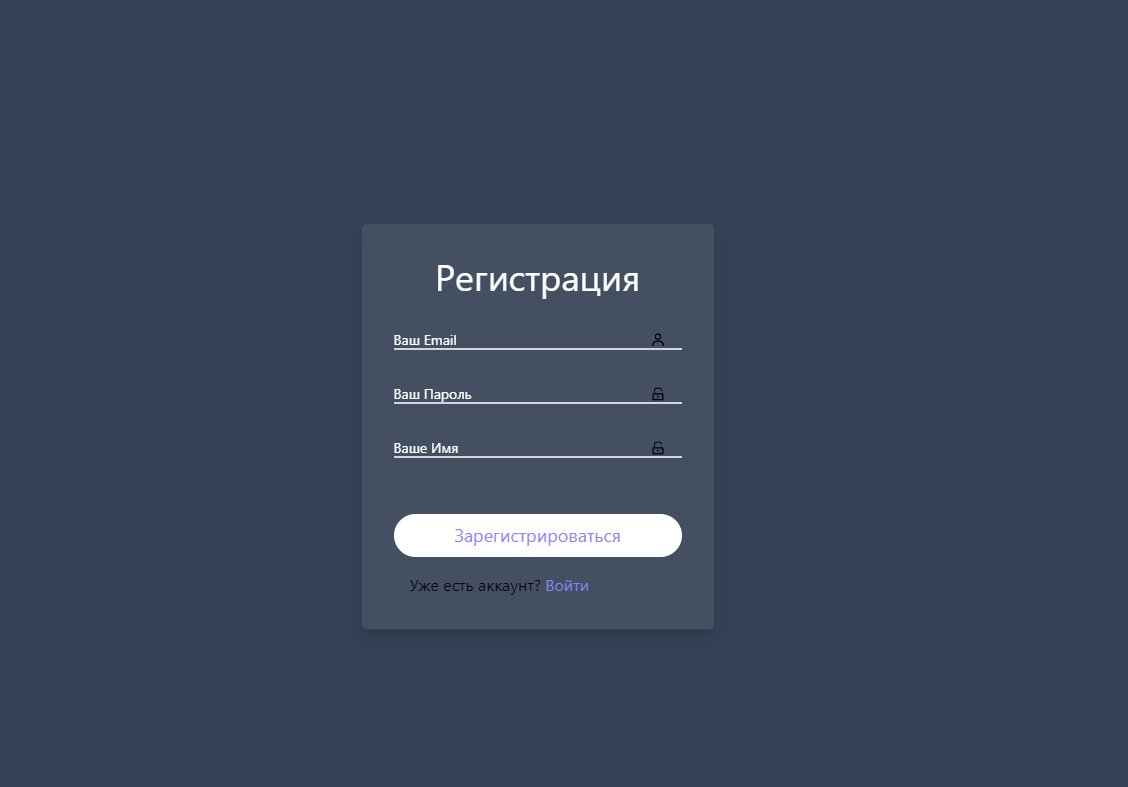


Рисунок 3.4 – Форма регистрации в системе

При нажатии на кнопку входа необходимо ввести логин и пароль от существующей учетной записи (Рисунок 3.5).

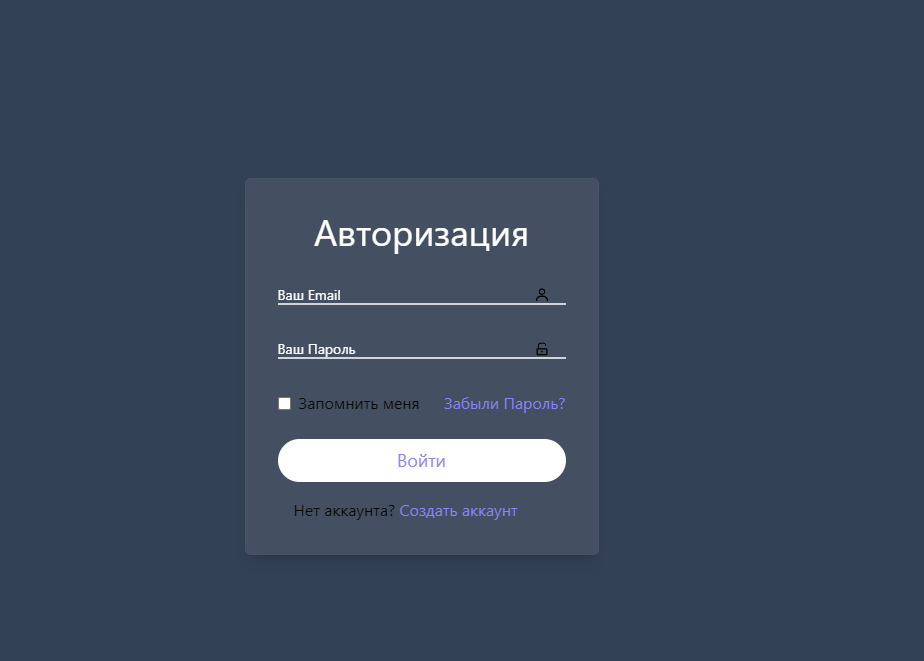


Рисунок 3.5 – Форма авторизации в сервисе

После авторизации или регистрации пользователю открываются новые пункты меню «Мой профиль» и «Выйти из аккаунта». Пункты представлены на рисунке 3.6.



Рисунок 3.6 – Пункты меню для авторизированного пользователя

Пункты меню для всех пользователей в виде учебных материалов, чата, информации о сервисе и тесте знаний представлены на рисунке 3.7.



Рисунок 3.7 – Пункты меню для всех пользователей

Нажимая на вкладку «Учебные материалы», пользователь попадает на список всех обучающих материалов (Рисунок 3.8).

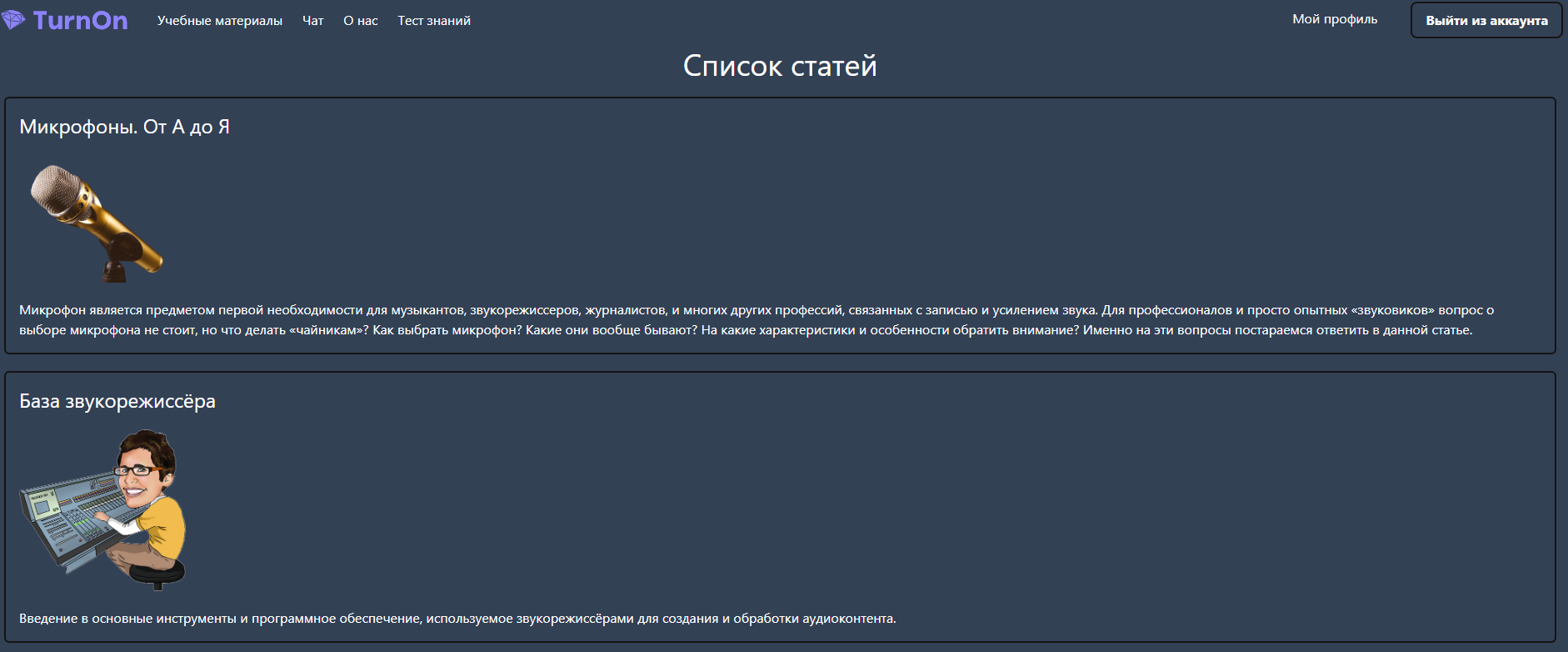


Рисунок 3.8 – Список статей

Нажимая на вкладку «Чат», пользователь попадает на страницу выбора комнаты и выбора имени, которое он хочет, чтоб отображалось в чате (Рисунок 3.9).

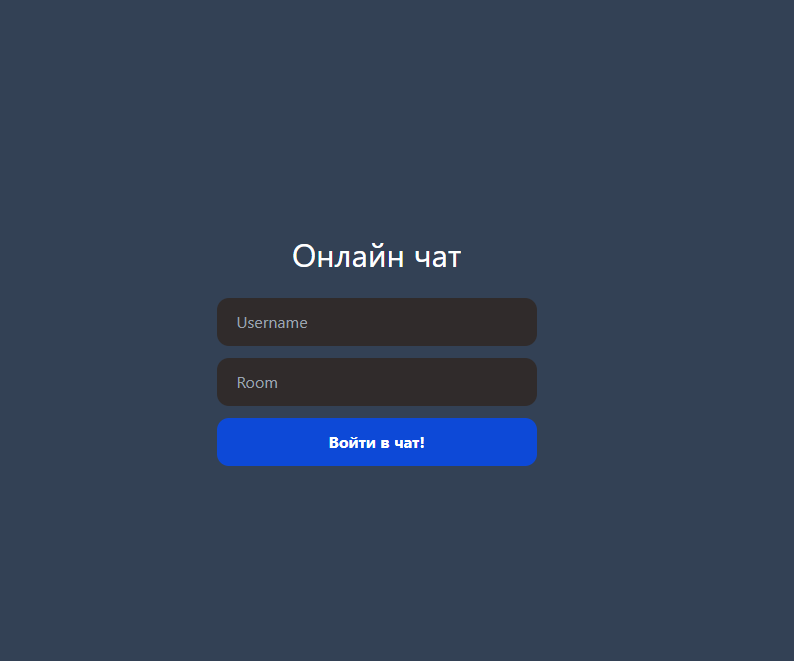


Рисунок 3.9 – Раздел подключения к чату

После ввода нужной информации в поле имени пользователя и номера комнаты, пользователь попадает в сам чат, где может видеть приветственное сообщение, количество всех пользователей в этой комнате и полноценный чат для общения с другими участниками комнаты (Рисунок 3.10).

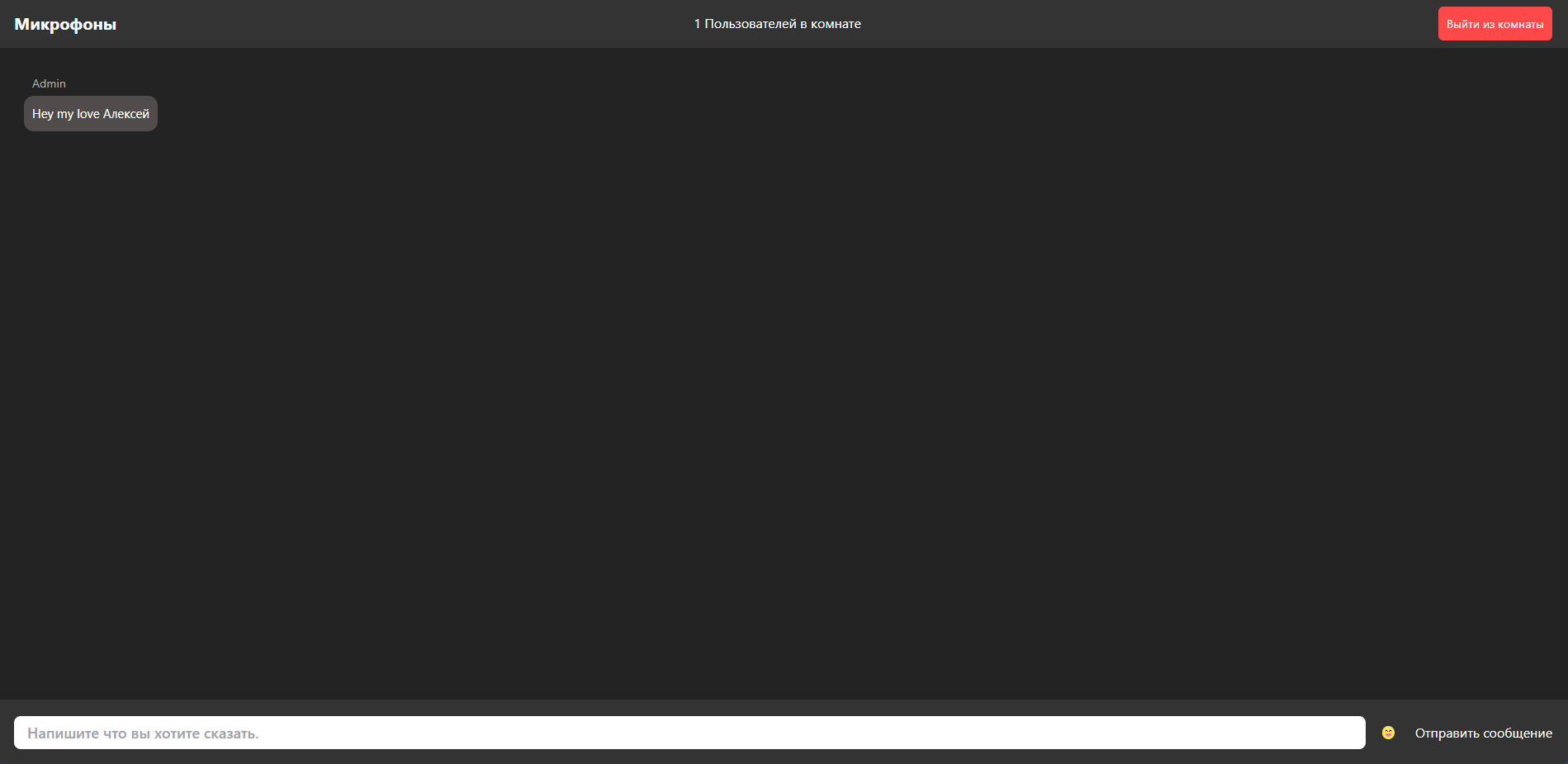


Рисунок 3.10 – Комната онлайн-чата

После завершения общения в чате пользователь может нажать кнопку «Выйти из комнаты», после чего покинет данную комнату и вернётся на главную страницу.

Если пользователь хочет выйти из аккаунта, то достаточно сверху на навигационной части сервиса нажать «Выйти из аккаунта» (Рисунок 3.11), после чего пользователь вернётся на главную страницу и будет не авторизован.



Рисунок 3.11 – Кнопка выхода из аккаунта

При переходе в «Мой профиль», будучи авторизированным пользователем, пользователь увидит информацию о своей почте и имени, а также избранные учебные материалы. Учебные материалы «Микрофоны. От А до Я» и «Всё о сведении музыки» добавлены сразу в список избранного (Рисунок 3.12).

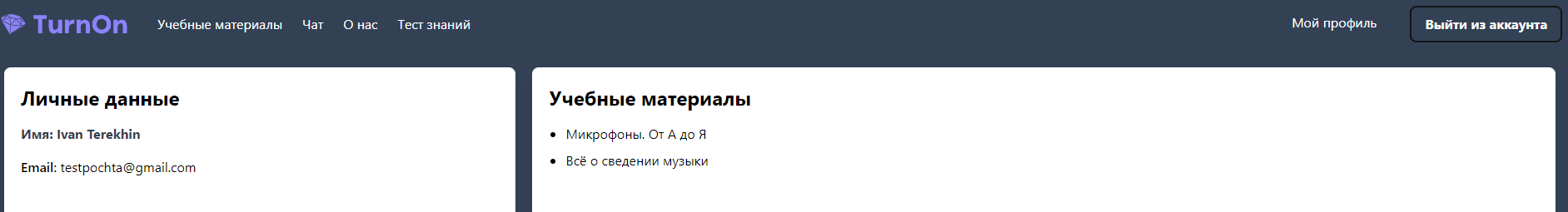


Рисунок 3.12 – Личный кабинет

При переходе на вкладку «Тест знаний» пользователь попадает к тесту на знание материала по теме «звукорежиссура». На этой вкладке находится 17 вопросов, где нужно выбрать 1 правильных ответ из 4 предложенных вариантов (Рисунок 3.13). Вопросы охватывают темы из материалов, представленных во вкладке «Учебные материалы».

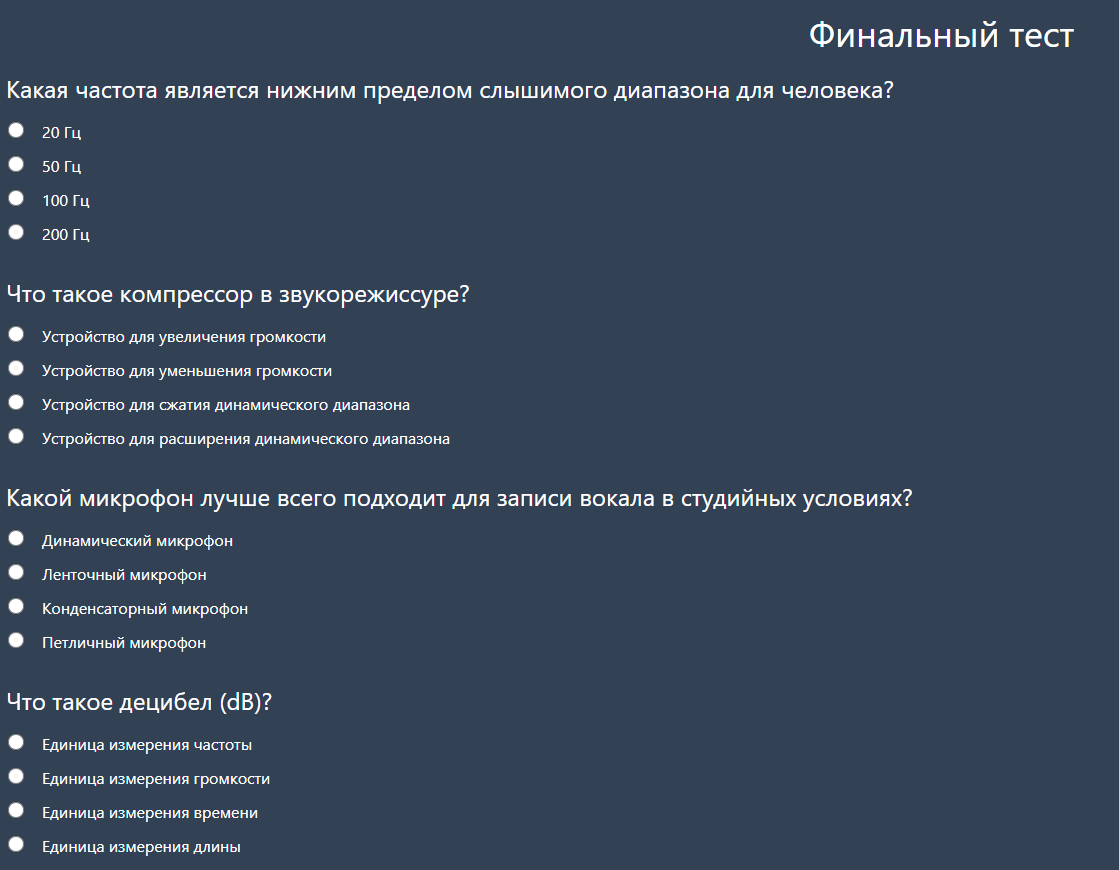


Рисунок 3.13 – Финальный тест знаний

После завершения теста пользователь может отправить свои ответы нажав на кнопку «Отправить» (Рисунок 3.14).

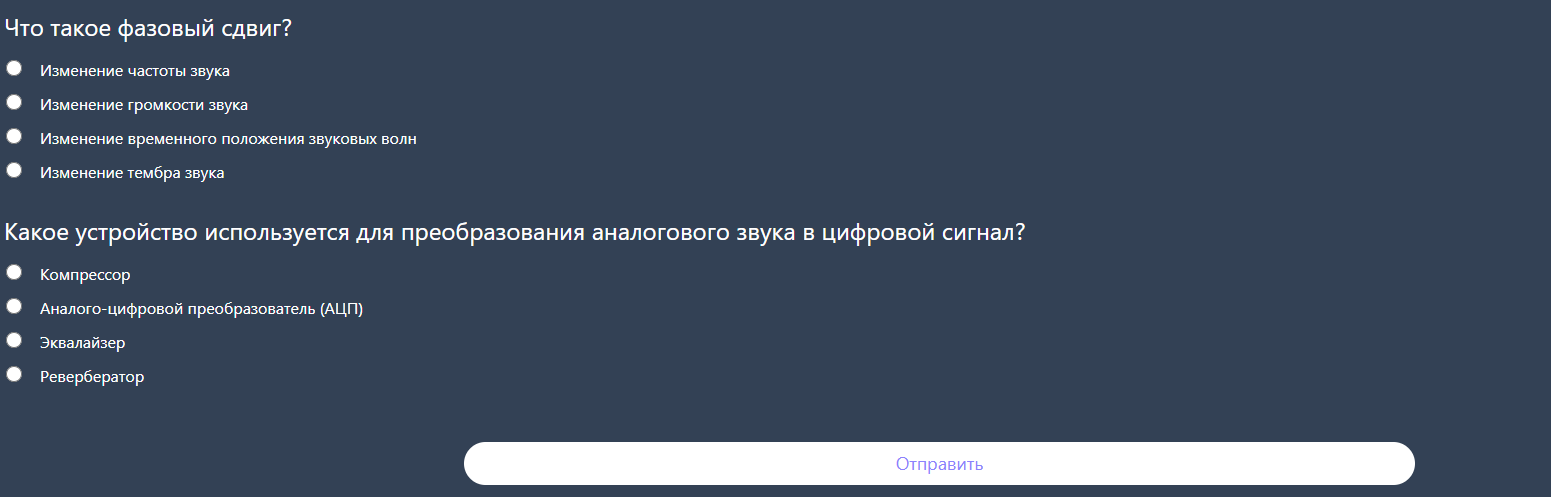


Рисунок 3.14 – Отправка ответов

После отправки ответов на вопросы и получения результатов, под кнопкой «Отправить» можно увидеть результат тестирования (Рисунок 3.15).

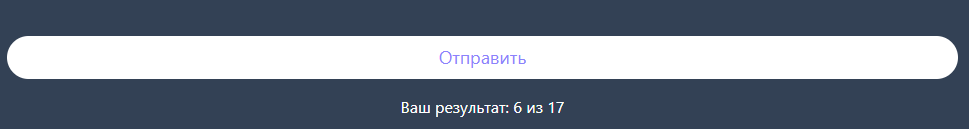


Рисунок 3.15 – Результат тестирования

При переходе на вкладку «Аудио тесты» пользователь попадает к аудио тесту музыкального слуха. На этой вкладке находится 6 вопросов с аудиодорожками, где нужно выбрать 1 правильный ответ из 3-4 предложенных вариантов (Рисунок 3.16). C помощью кнопки проигрывания музыкальных дорожек пользователь может ознакомиться с музыкальным фрагментом и сделать на поле ниже.

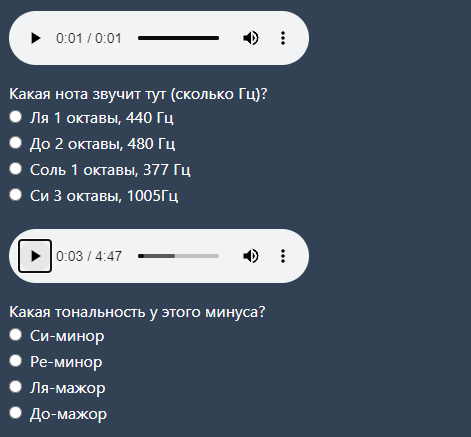


Рисунок 3.16 – Аудио тест

Выше был рассмотрен весь процесс работы с сервисом.

## 3.5 Тестирование программного продукта

Для успешного функционирования сервиса необходимо протестировать каждый этап работы с ним, в котором пользователь влияет на используемые данные. Поэтому нужно найти случаи вводимых данных, при которых разработанный программный продукт работает некорректно. Для этого было решено использовать метод «черного ящика».

Для тестирования были выбраны следующие методики:

*Unit*-тестирование;

дымное тестирование.

### Unit-тестирование

Данный вид тестирование идеально подходит для проверки каких-то частей логики, например, проверка функции получения вакансии для пользователя.

Для написания подобных тестов существует множество библиотек, в том числе и для *Python*. Однако, самых популярных 2: *unittest* и *pytest*. Было решено использовалась для написания тестов библиотеку *unittest*.

В разработанной системе *unit*-тесты были написаны для следующего функционала:

получение списка активных модулей;

получение списка статей из БД;

отправка мультимедийных файлов на сервер.

Работоспособность данного функционала является жизненно необходимой для сервиса, поэтому он был проверен точечно. Пример тестов в листинге 3.3.

Листинг 3.3 – Проверка получения статей

describe('GET /api/articles', () => {

it('should get all articles', (done) => {

chai.request(app)

.get('/api/articles')

.end((err, res) => {

expect(res).to.have.status(200);

expect(res.body).to.be.an('array');

expect(res.body.length).to.be.equal(1);

done();

});

});

### Дымное тестирование

Данный вид тестирования необходим, чтобы быстро проверить правильность работы приложения. Если приложение не проходит данный вид тестирования, значит оно абсолютно неработоспособно.

Все тесты данного типа можно свести к простому запуску приложения в ожидании его падения. Однако для Ошибка с номером 500 означает, что произошел сбой в работе веб приложения, соответственно если она отсутствует, значит все работает корректно.

Листинг 3.4 – Проверка получения активных модулей

const chai = require('chai');

const chaiHttp = require('chai-http');

const app = require('../app');

const expect = chai.expect;

chai.use(chaiHttp);

describe('Smoke Tests', () => {

it('should return 200 for the home page', (done) => {

chai.request(app)

.get('/')

.end((err, res) => {

expect(res).to.have.status(200);

done();

});

});

it('should return 404 for an unknown route', (done) => {

chai.request(app)

.get('/unknown-route')

.end((err, res) => {

expect(res).to.have.status(404);

done();

});

});

it('should be able to register and login a user', (done) => {

const user = { name: 'Test User', email: 'test@example.com', password: 'testpassword' };

chai.request(app)

.post('/api/register')

.send(user)

.end((err, res) => {

expect(res).to.have.status(201);

chai.request(app)

.post('/api/login')

.send({ email: user.email, password: user.password })

.end((err, res) => {

expect(res).to.have.status(200);

expect(res.body).to.have.property('token');

done();

});

});

});

В процессе тестирования ошибок обнаружено не было.

## 3.6 Выводы по главе

В третьей главе данной бакалаврской работы была описана структура разработанного сервиса.

Основываясь на этом, была проведена реализация модульной системы сервиса. Описаны принципы авторизации и отправки мультимедийных файлов.

Для дальнейшей разработки, а также использования сервиса, были написаны руководства программиста и пользователя.

Кроме того, было проведено тестирование разработанного сервиса, в ходе которого не было установлено явных нарушений в его работе.

# ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы была разработана система по обучению звукорежиссуре.

В процессе разработки был проведен анализ существующих аналогов данного продукта. На основе результатов обзора таких проектов, как *SkillBox*, Школа профессий, *Netology* по нескольким показателям были сформированы требования к разрабатываемому приложению и выявлены оптимальные функциональные возможности для работы с ним.

Был проведен обзор технологий, использование которых наиболее оптимально при разработке данного программного продукта и были выбраны наиболее подходящие для решения поставленных задач. Кроме того, был проведен обзор методов и алгоритмов, применяемые при разработке. В том числе, были изучены и применены методы для защиты данных, хранящихся в базе данных. Благодаря этому была определена архитектура проекта, а также пути для его реализации.

В качестве используемых технологий были выбраны:

*React* – для создания веб-интерфейса;

*Node.js* – для создания бизнес логики;

*MongoDB* – в качестве основной базы данных.

В ходе рабочего тестирования недостатков, касающихся требований реализованных функций, выявлено не было. На заключительном этапе было проведено тестирование, которое показало, что разработанный программный продукт показывает корректную работу на всех этапах работы.

Было написано руководство программиста, в котором были описаны процессы запуска проекта и способы его редактирования. Также сформировано руководство пользователя, в котором полностью описан процесс работы с разработанной системой.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Саммерфилд, М. Изучаем *Python* / М. Саммерфилд. – СПб.: Символ Плюс, 2009. – 848 с. – Текст: непосредственный.
2. Уилсон, Д. Р. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL / Д. Р. Уилсон. – СПб.: БХВ-Петербург, 2018. – 432 с. – Текст: непосредственный.
3. Флэнаган, Д. JavaScript. Подробное руководство, 6-е издание / Д. Флэнаган. – Перевод А. Киселева. – СПб: Символ Плюс, 2012. – 1080 с. – Текст: непосредственный.
4. Меерзон, Б.Я. Акустические основы звукорежиссуры. – URL: https://www.globalaudio.ru/sadm\_files/Meerzon20B\_YA\_20Akusticheskie\_osnovue\_zvukorezhiss (дата обращения: 17.04.2024). – Текст: электронный.
5. Клиент серверная архитектура: сайт. – URL: <https://habr.com/ru/post/495698/> (дата обращения: 20.04.2024). – Текст: электронный.
6. Построение микросервисной архитектуры: сайт. – URL: <https://habr.com/ru/post/441310/> (дата обращения: 25.04.2024). – Текст: электронный.
7. Angular: сайт. – URL: <https://angular.io> (дата обращения: 07.04.2024). – Текст: электронный.
8. FastAPI: сайт. – URL: <https://fastapi.tiangolo.com> (дата обращения: 04.04.2024). – Текст: электронный.
9. Flask documentation: сайт. – URL: <https://flask.palletsprojects.com/en/2.1.x/> (дата обращения: 03.04.2024). – Текст: электронный.
10. ItBlack - Средства шифрования: сайт. – URL: https://it-black.ru/sredstva-shifrovanija-v-python/ (дата обращения: 04.04.2024). – Текст: электронный.
11. React: сайт. – URL: <https://ru.reactjs.org> (дата обращения: 07.04.2024). – Текст: электронный.
12. The web framework Django: сайт. – URL: <https://www.djangoproject.com> (дата обращения: 01.04.2024). – Текст: электронный.
13. VueJS: сайт. – URL: <https://vuejs.org> (дата обращения: 07.04.2024). – Текст: электронный.
14. Flask documentation: сайт. – URL: <https://flask.palletsprojects.com/en/2.1.x/> (дата обращения: 03.04.2024). – Текст: электронный.
15. The web framework Django: сайт. – URL: <https://www.djangoproject.com> (дата обращения: 01.04.2024). – Текст: электронный.
16. Node.js: сайт. – URL: https://nodejsdev.ru/guides/freecodecamp (дата обращения: 01.04.2024). – Текст: электронный.
17. Netology: сайт. – URL: https://netology.ru/programs/sound-design (дата обращения: 01.04.2024). – Текст: электронный.
18. Skillbox: сайт. – URL: https://skillbox.ru/course/profession-sound-engineer/ (дата обращения: 01.04.2024). – Текст: электронный.

**ПРИЛОЖЕНИЯ**

ПРИЛОЖЕНИЕ А

Техническое задание на разработку

*Введение*

В современном мире, где все больше групп и частных лиц связывают свою жизнь с творческой сферой, а именно написанием музыки или игрой на музыкальных инструментах, все больше требуется обучение людей звукорежиссуре и принципам написания своих песен или аранжировок на чьи-то композиции. Существующие на рынке решения (как очные занятия в музыкальных школах или кружках, так и курсы обучения звукорежиссуре) не всегда удовлетворяют потребности потребителей, что открывает возможности для создания новых приложений, предоставляющих более удобный и эффективный способ обучения звукорежиссуре и написания своих композиций.

Разработка веб-сервиса для обучения звукорежиссуре имеет большой потенциал для успешного запуска на рынок. Кроме того, веб-сервис для обучения звукозаписи и написанию собственных композиций позволит упростить процесс обучения и сделать его более удобным, что повысит эффективность работы и снизит вероятность ошибок.

Следовательно, разработка нового веб-сервиса для обучения принципам звукорежиссуры и написанию собственных композиций является актуальной задачей, такой веб-сервис позволит пользователям удобно и эффективно обучаться принципам звукозаписи и написанию собственных композиций.

*Общие положения*

Полное наименование: Веб-сервис по обучению звукорежиссуре.

Краткое наименование: *TurnOn*.

Вид разработки: *web*-приложение.

Разработчик: Терехин Иван Алексеевич, студент группы БПИ20-02.

*Основание для разработки*

Разработка программного продукта ведется на основании учебного плана СибГУ им. М.Ф. Решетнева по направлению подготовки 09.03.04 Программная инженерия, направленность «Разработка программно-информационных систем».

*Плановые сроки начала и окончания выполнения работы*

Плановый срок начала работ – 08.04.2024

Плановый срок окончания работ – 21.05.2024

*Назначение разработки*

Разрабатываемый программный продукт предназначен для обучения пользователей принципам звукорежиссуры.

*Требования к программе*

Требования к функциональным характеристикам

Сервис должен:

позволять пользователям регистрироваться и авторизироваться; иметь возможность добавления учебных материалов себе в личный кабинет;

позволять пользователям общаться в онлайн чате с другими пользователями сервиса, а также с преподавателями и менторами;

позволять пользователям просматривать ленту обучающих статей;

позволять пользователям проходить итоговый тест полученных знаний в сфере звукорежиссуры.

Ввод данных должен выполняться в программе посредством специальных блоков для ввода текста или выбора необходимого варианта. Вывод информации должен выполняться в виде блоков сайта.

Требования к надежности

В программе должен быть обеспечен контроль за личными данными пользователя. Возникновение каких-либо внутренних ошибок не должно приводить к утере данных. Некорректная работа сторонних модулей не должна приводить к сбоям в работе системы. Программа должна быть в правильной форме реагировать на действия пользователей.

Для защиты от несанкционированного использования должна быть предусмотрена авторизация, а данные должны передаваться безопасным путём. Кроме того, доступ к базе данных должен быть возможным только с локальных *ip*-адресов.

*Условия эксплуатации*

Требования к составу и параметрам технических средств

процессор *Intel* *Core I*3/ *AMD Ryzen* 3или лучше;

оперативная память не меньше 8 Гб;

не менее 30 Гб свободного места на жестком диске;

* постоянный канал связи с глобальной сетью «Интернет» со скоростью соединения не менее 512 Кб/с.

*Требования к информационной и программной совместимости*

* операционная система *Windows* 8/10, *MacOS* или *Linux*;

браузер *Google Chrome* 96.0.4664 и выше*, Mozilla Firefox* 91 и выше*, Safari* 13 и выше или другие.

*Состав и содержание работ по выполнению бакалаврской работы*

Таблица А.1 – Календарный план-график выполнения стадий и этапов разработки

|  |  |
| --- | --- |
| Наименование работ | Сроки  выполнения |
| Анализ предметной области | 15.04.2024 |
| Обзор существующих веб-сервисов для обучения звукорежиссуре | 18.04.2024 |
| Обзор технологий | 21.04.2024 |

Продолжение таблицы А.1

|  |  |
| --- | --- |
| Наименование работ | Сроки  выполнения |
| Обзор методов и алгоритмов | 23.04.2024 |
| Разработка серверной части | 25.04.2024 |
| Разработка модульной системы | 28.04.2024 |
| Разработка *fronted* части | 03.05.2024 |
| Предоставление ВКР руководителю и ее доработка в соответствии с полученными замечаниями | 18.05.2024 |
| Предоставление ВКР на нормоконтроль и ее доработка в соответствии с полученными замечаниями | 19.05.2024 |
| Подготовка презентации и доклада для представления на предзащите | 20.05.2024 |
| Представление ВКР на предзащите и ее доработка по результатам предзащиты | 21.05.2024 |
| Предоставление на кафедру всех необходимых для защиты материалов | 10.06.2024 |