

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

К защите допустить:

Заведующая кафедрой ПОИТ

_____ Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему

**ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ СИНТЕЗА, ХРАНЕНИЯ И
РАСПРОСТРАНЕНИЯ АУДИОКНИГ КНИГ
НА БАЗЕ SPRING FRAMEWORK**

БГУИР ДП 1-40 01 01 01 029 ПЗ

Студент	В.В. Гринчик
Руководитель	Н.П. Можей
Консультанты: <i>от кафедры ПОИТ по экономической части</i>	Н.П. Можей А.А. Горюшкин
Нормоконтролер	А. В. Манцевич
Рецензент	

Минск 2021

РЕФЕРАТ

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ СИНТЕЗА, ХРАНЕНИЯ И РАСПРОСТРАНЕНИЯ АУДИОКНИГ КНИГ НА БАЗЕ SPRING FRAMEWORK: дипломный проект / В. В. Гринчик. – Минск : БГУИР, 2021, – п. з. – 92 с., чертежей (плакатов) – 6 л. формата А1.

Объектом проектирования является веб-приложение создания и проведения опросов.

Целью данного дипломного проекта является создание приложения для кроссбраузерного создания и проведения опросов. Актуальность данной работы обусловлена современными тенденциями развития информационного общества, благодаря которым появилась возможность создавать и проводить опросы быстрее, качественнее и дешевле. Пользователю лишь необходимо заполнить необходимые данные и отправить опрос на сервер. После этого он станет доступен пользователям приложения по всему миру в указанное автором опроса время.

Проведён анализ существующих на рынке аналогичных программных средств, рассмотрены и учтены достоинства и недостатки данных программных продуктов. На основе проведённого анализа разработаны и полностью описаны спецификации функциональных требований к приложению.

Базируясь на функциональных требованиях, разработана архитектура приложения.

Разработаны тесты для проверки соответствия функциональным требованиям и корректности работы приложения.

Приведено технико-экономическое обоснование эффективности разработки и использования приложения.

Дипломная работа прошла проверку в системе «Антиплагиат». Уникальность данного проекта составляет 95,98 %.

Министерство образования Республики Беларусь

Учреждение образования

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ
И РАДИОЭЛЕКТРОНИКИ**

Факультет КС и С Кафедра ПОИТ
Специальность 1-40 01 01 Специализация 01

УТВЕРЖДАЮ

Н.В.Лапицкая

« » 20 г.

ЗАДАНИЕ

по дипломному проекту студента

Гринчика Всеволода Владимировича

(фамилия, имя, отчество)

1. Тема проекта: **Веб-приложение для синтеза, хранения и распространения аудио-книг на базе Spring Framework**

утверждена приказом по университету от « 06 » апреля 2021 г. № 765-с

2. Срок сдачи студентом законченной работы 01 июня 2021 года

3. Исходные данные к проекту Тип операционной системы – ОС Ubuntu 18.04;
Языки программирования – Java, JavaScript;

Перечень выполняемых функций: регистрация, авторизация, отображение списка аудиокниг, отображение подробной информации об аудиокниге, синтез аудиокниги, фильтрация списка аудиокниг, добавление новой аудиокниги.

Назначение разработки: хранение и распространение аудио версий литературных произведений в сети интернет, а также их синтез программными методами.

4. Содержание пояснительной записки (перечень подлежащих разработке вопросов)

Введение

1 Анализ литературных источников, прототипов и формирование требований к проектируемому приложению

2 Моделирование предметной области и разработка функциональных требований

3 Проектирование приложения

4 Создание приложения

5 Тестирование приложения

6 Руководство по установке и использованию

7 Техничко-экономическое обоснование разработки веб-приложения для синтеза, хранения и распространения аудиокниг

Заключение

Список использованных источников

Приложение А. Исходный код программы

5. Перечень графического материала (с точным указанием наименования) и обозначения вида и типа материала)

Диаграмма развертывания веб-приложения. Плакат - формат A1, лист 1.

Диаграмма UML веб-приложения. Плакат - формат A1, лист 1.

База данных веб-приложения. Плакат - формат A1, лист 1.

Контроллер аудиокниг. Схема алгоритма - формат A1, лист 1.

Движение данных при регистрации. Схема данных - формат A1, лист 1.

Приложение для синтеза, хранения и распространения аудиокниг. Схема программы - формат A1, лист 1.

6. Содержание задания по технико-экономическому обоснованию

Технико-экономическое обоснование разработки веб-приложения для синтеза, хранения и распространения аудиокниг

Задание выдал _____ / А.А. Горюшкин /

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов дипломного проекта (работы)	Объём этапа в %	Срок выполнения этапа	Примечание
Анализ предметной области, разработка технического задания	15-20	23.03–01.04	
Разработка функциональных требований, проектирование архитектуры программы	15-20	02.04–08.04	
Разработка схемы программы, алгоритмов, схемы данных	15-20	09.04–15.04	
Разработка программного средства	15-20	16.04–29.04	
Тестирование и отладка	10	30.04–13.05	
Оформление пояснительной записки и графического материала	20	14.05–31.05	

Дата выдачи задания 22 марта 2021 г. Руководитель _____ /Н.П. Можей/

Задание принял к исполнению _____ / В.В. Гринчик /

СОДЕРЖАНИЕ

Введение.....	7
1 Анализ литературных источников, прототипов и формирование требований к проектируемому приложению.	8
1.1 Анализ литературных источников.....	8
1.2 Аналоги, их недостатки и достоинства	11
1.3 Цели и задачи дипломного проекта. Формирование требований к приложению	17
2 Моделирование предметной области и разработка функциональных требований.....	23
2.1 Функциональная модель программного средства.....	23
2.2 Спецификация функциональных требований	27
3 Проектирование приложения	37
3.1 Разработка архитектуры приложения	37
3.2 Разработка даталогической и физической моделей базы данных ...	40
3.3 Разработка алгоритма приложения и алгоритмов отдельных модулей.....	43
4 Создание приложения.....	48
5 Тестирование, проверка работоспособности и анализ полученных результатов	56
6 Руководство по установке и использованию	67
7 Техничко-экономическое обоснование	74
7.1 Краткая характеристика приложения.....	74
7.2 Расчёт затрат на разработку приложения	74
7.3 Оценка эффекта от использования приложения	77
7.4 Расчёт показателей эффективности инвестиций в разработку приложения	78
7.5 Вывод	78
Заключение	79
Список использованных источников	80
Приложение А	81

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Internet – всемирная система объединённых компьютерных сетей для хранения и передачи информации.

Никнейм – псевдоним, используемый пользователем в сети Internet.

ОС – Операционная Система – комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

СУБД – Система Управления Базами Данных – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Пагинация – в веб-приложениях под пагинацией понимают постраничный вывод с показом ограниченной части информации на одной веб-странице и возможностью переключения между страницами.

ПО – Программное Обеспечение.

MVC – Model-View-Controller (Модель-Представление-Контроллер) – архитектурный паттерн, подразумевающий разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента.

Кроссбраузерность – это идентичное отображение веб-страниц в большинстве браузеров.

Триггер – логическое выражение, запускающее определённое событие при переходе в истинное значение.

API – Application Programming Interface – описание процедур и функций, с помощью которых одна компьютерная программа может взаимодействовать с другой программой.

DTO – Data Transfer Object – объекты, служащие для абстрагирования модели от внешнего мира, так как зачастую внешнему миру не нужны все данные, которые хранятся в модели.

Маппинг – определение соответствия данных между потенциально различными семантиками одного объекта или разных объектов. Термин понимается очень широко от отображения одной последовательности элементов на другую последовательность до банальной конвертации объектов.

ВВЕДЕНИЕ

Аудиокни́га (от лат. audio «слушать») — озвученное литературное произведение. Аудиокниги могут быть как развлекательными, так и просветительскими или образовательными. Сюда можно отнести аудиолитературу для инвалидов, слепых и людей с нарушенным зрением, начитанные сказки для детей младшего возраста, аудиокурсы иностранных языков, и тому подобное. В современном мире аудиокниги обретают всё большую популярность. Это обусловлено многими факторами.

К таким факторам можно отнести удобство прослушивания аудиокниг в ситуациях, не требующих серьёзной концентрации внимания и в то же время по различным причинам не позволяющих читать обычные книги. Например, поездки в общественном транспорте, приготовление пищи, уборка, спортивные тренировки и тому подобное. В подобных ситуациях удобство аудиокниг в первую очередь связано с тем, что они не занимают руки, не требуют использовать зрение, и даже слух продолжает воспринимать информацию, не связанную с аудиокнигой.

Ещё одним важным фактором является то, что прослушивания аудиокниг позволяет снять часть нагрузки со зрительных органов, а это, в свою очередь, положительно сказывается на их здоровье. Особенно важен данный фактор для работников сферы информационных технологий, которые много времени проводят, смотря в монитор компьютера, чем серьёзно напрягают зрение.

Также необходимо обратить внимание на то, что аудиокниги позволяют получить доступ к текстовой информации инвалидам и людям с нарушенным зрением. Что оказывает положительное влияние на их социально-психологическую адаптацию в обществе, и улучшает качество жизни.

На данный момент создание аудиокниги длительный и довольно дорогой процесс. И в связи с этим количество ежегодно выпускаемых аудиокниг значительно меньше количества печатных изданий. На текущий момент у данной проблемы есть два решения: заказать профессиональную озвучку книги по высокой цене или установить на свое устройство специальную программу для озвучивания книги в реальном времени. У каждого из решений есть свои недостатки. Разрабатываемый проект должен стать альтернативным решением, позволяющим не только создавать, но и хранить аудиокниги, а также распространять их в сети интернет.

Целью данного дипломного проекта является создание веб-приложения для синтеза, хранения и распространения аудиокниг. Актуальность темы обусловлена ростом популярности аудиокниг и современными тенденциями развития информационного общества, в связи с которыми возникла необходимость в дешевом и быстром способе создания, хранения и распространения аудиокниг.

1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ, ПРОТОТИПОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРИЛОЖЕНИЮ.

Итоговый успех разработки и реализации программного проекта во многом определяется на этапе подготовки, во время которой необходимо как можно больше и тщательней определить нюансы и особенности проекта.

Первое предварительное условие, которое нужно выполнить перед конструированием, – ясное формулирование проблемы, которую система должна решать. Общая цель подготовки – снижение риска. Адекватное планирование позволяет исключить главные аспекты риска на самых ранних стадиях работы, чтобы основную часть проекта можно было выполнить максимально эффективно.

Главный факторы риска в создании ПО – неудачная выработка требований. Требования подробно описывают, что должна делать программная система. Внимание к требованиям помогает свести к минимуму изменения системы после начала разработки [1].

1.1 Анализ литературных источников

1.1.1 История и тенденции роста популярности аудиокниг

История появления и развития аудиокниг насчитывает много лет. Аудиокниги вызвали огромный интерес у людей во всем мире. Термин «аудиокнига» впервые появился в Германии в 1954 году. Это был год основания немецкой аудиобиблиотеки для слепых. В том же году «Немецкий граммофон» попытался записать инсценировку «Фауста». Эта запись, которая была выпущена в количестве 250 000 экземпляров, имела большой успех и считается началом основания и развития аудиокниги.

С 1963 года изобретение магнитофона дало возможность распространять и продавать записи на кассетах. Первыми слушателями кассет были слепые и с плохим зрением люди, которые благодаря особому качеству аудиокниг имели возможность пользоваться ими в любое время и в любом месте. Постепенно у аудиокниг появляется все больше и больше поклонников: любители литературы, домохозяйки, учащиеся школ и студенты, путешественники и т. д.

В настоящее время популярность аудиокниг не угасла. А появление интернета открыло еще больше возможностей по сбыту аудиокниг, так как формат MP-3 позволяет это сделать без всяких проблем, экономя расходы на их производство, хранение, упаковку и пересылку.

Изменения информационной и общественной среды также вносят свой вклад в растущую в последнее время популярность аудиокниг. К таким изменениям можно отнести:

- визуальное перенапряжение ежедневным потоком информации (дорожные знаки, рекламные щиты, витрины и т. д.);
- работа за компьютером вызывающая соответствующее напряжение глаз;
- ежедневный поток информации из печатных изданий (газеты, журналы, специальная литература и т. д.);
- деятельность, связанная с общими мыслительными процессами и оставляющая место для дополнительной информации (вождение автомобиля, перемещение в общественном транспорте).

Из всего вышесказанного можно сделать вывод, что аудиокниги пользуются большой популярностью благодаря своей развлекательной функции, особенно среди людей, чья деятельность связана со зрительным напряжением. Аудиокниги помогают расслабиться, их можно слушать с закрытыми глазами, а также, благодаря гибкости их использования, в любом месте и в любое время [2].

На основе проведенного анализа тенденций роста популярности аудиокниг, можно сделать вывод как о текущем высоком спросе на аудиокниги, так и о том, что спрос будет продолжать расти. Что позволяет рассчитывать на высокую популярность проектируемого приложения, и как следствие получение значительной прибыли от разработки.

1.1.2 Особенности проектирования программных систем хранения и распространения аудиокниг

По своей технологии и организации программная система хранения и распространения аудиокниг наиболее близка к электронной библиотеке.

Первые шаги по созданию электронных библиотек (ЭБ) были сделаны за рубежом в начале 1980-х гг. В 1992 г. на конференции Национального научного фонда США было введено в оборот понятие «цифровая библиотека» в современном контексте.

В общем случае при проектировании ЭБ следует рассматривать два класса требований, которые можно назвать пользовательскими и общесистемными. Пользовательские требования определяют содержание фонда, его структуру, систему метаданных и функциональные возможности ЭБ. Общесистемные требования определяют общую структуру ЭБ, технологию функционирования ЭБ в рамках действующей организации с учетом ее задач и специфики, взаимодействия с другими организациями, порядок ее использования и администрирования.

Все информационное пространство ЭБ, доступное пользователю, должно быть представлено в виде совокупности самостоятельных объектов. В качестве таковых во многих случаях выступают электронные документы. Электронные объекты в общем случае могут представлять собой текстовые произведения, изображения, аудиофайлы, базы данных или их фрагменты,

словарные статьи, подписи под рисунками, отдельные имена и т. д. Организация информационного пространства как совокупности объектов и однозначная идентификация последних необходимы для обеспечения эффективной навигации и выполнения некоторых видов информационных поисков. Инструментом описания и идентификации выступают метаданные, в том числе библиографические записи, поскольку основную часть фонда будут составлять обычные документы.

Для представления документов в ЭБ могут использоваться разные форматы, в том числе:

- формат PDF;
- форматы DOC (DOCX), TXT;
- аудиоформаты, например, MP3;
- гипертекстовый язык разметки HTML;
- расширенный язык разметки текста XML.

Выбор одного или нескольких форматов для хранения определяется в рамках концепции ЭБ с учетом пользовательских и общесистемных требований [3].

Таким образом разрабатываемое в рамках данного дипломного проекта приложение может быть определено как электронная библиотека, хранящая документы в аудиоформате. А принципы, изложенные ранее, могут применяться при разработке технического задания к приложению.

1.1.3 Модели работы технологии синтеза речи

По сути любые системы синтеза аудиокниг основаны на технологии синтеза речи.

Все существующие в настоящее время методы синтеза человеческой речи основаны на использовании двух моделей — модели компилятивного синтеза и формантно-голосовой модели.

Модель компилятивного синтеза предполагает синтез речи путем конкатенации (составления) записанных образцов отдельных звуков, произнесенных диктором. При использовании этой модели составляется база данных звуковых фрагментов, из которых в дальнейшем будет синтезироваться речь [4]. Преимуществом данной модели является простота реализации. Недостатком требовательность к образцам звуков.

Формантно-голосовая модель основана на моделировании речевого тракта человека. При форматном методе, моделируется результат физиологических процессов образования речи: акустические характеристики речевой волны [5]. Эта модель может быть реализована с применением нейронных сетей и допускает самообучение. К сожалению, ввиду сложности точного моделирования особенностей речевого тракта, а также учета интонационной модуляции речи формантно-голосовая модель обладает относительно низкой точностью синтезируемых звуков речи [4].

При синтезе аудиокниг несомненным преимуществом пользуется компилятивная модель синтеза, поскольку она дает более выразительно звучание при условии наличия качественных образцов отдельных звуков, и при этом требует меньше вычислительных ресурсов нежели формантно-голосовая.

1.2 Аналоги, их недостатки и достоинства

Веб-приложение для синтеза, хранения и распространения аудиокниг – это программное средство, хранящее аудио версии литературных произведений, и способное их синтезировать на основе печатного текста. С его помощью можно озвучивать книги различными голосами, а также распространять созданные аудиокниги среди интернет-сообщества. Кроме того, веб-приложение позволяет скачивать хранимые и распространяемые книги на устройство. Для использования приложения не требуется устанавливать дополнительно программное обеспечение на устройство, достаточно любого браузера.

Также важным достоинством является возможность сохранения аудиокниг на сервере приложения, с привязкой к аккаунту пользователя. Очень удобным является наличие функционала, позволяющего проводить поиск и фильтрацию распространяемых на сайте аудиокниг.

Так как полных аналогов приложения для синтеза, хранения и распространения аудиокниг не существует, в данном разделе будут рассмотрены два вида частичных аналогов: приложения для хранения и распространения аудиокниг и приложения для озвучивания текста.

Одним из самых популярных приложений хранения и распространения аудиокниг, является «Audio-knigki.com» (рисунок 1.1) [6].

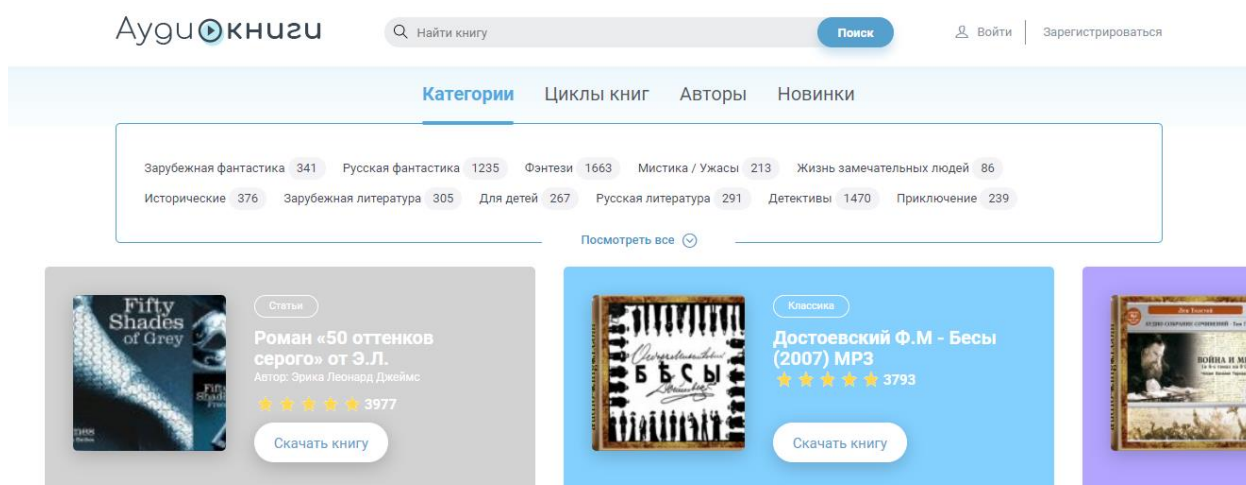


Рисунок 1.1 – Приложение для хранения и распространения аудиокниг «Audio-knigki.com»

Это бесплатное приложение, в котором можно прослушивать и скачи-

вать различные аудиокниги. Здесь можно найти литературу на любой вкус: детективы, нон-фикшн, фантастика, сказки, любовные романы, аудиоспектакли и другое. Есть возможность выбора аудиокниг по жанру, автору или циклам.

Для зарегистрированных пользователей имеется возможно добавить на сайт собственную версию аудиокниги. Или заказать добавление книги у администрации сайта.

В приложении предусмотрены возможность оценивания аудиокниг и добавления комментариев. Так же имеется возможность сортировки аудиокниг по рейтингу и времени добавления.

У данного приложения можно выделить следующие преимущества:

- возможность прослушать отрывок аудиокниги прямо на сайте;
- возможность добавлять на сайт собственные аудиокниги;
- возможность сортировки и фильтрации книг;
- возможность заказать добавление книги у администрации.

Недостатки:

- нельзя загрузить аудиофайл напрямую, на сайте размещены только ссылки на скачивание со сторонних ресурсов;
- часть книг доступна только на платной основе.

В целом, «Audio-knigki.com» – это одно из лучших приложений для хранения и распространения аудиокниг.

Ещё одним популярным приложением для хранения и распространения аудиокниг является «Baza-Knig» (рисунок 1.2) [7].

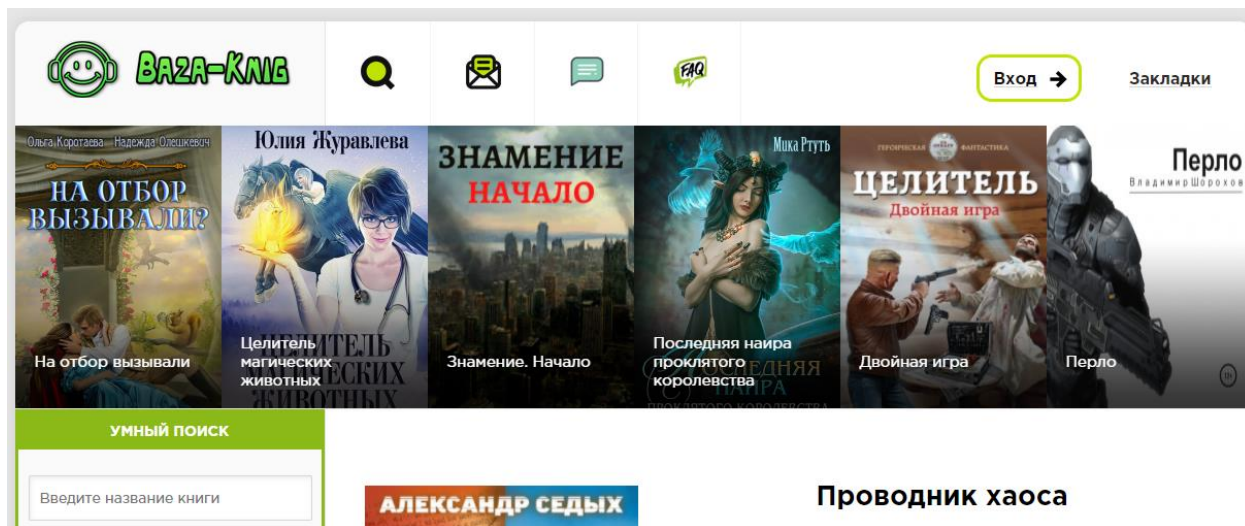


Рисунок 1.2 – Приложение для хранения и распространения аудиокниг «Baza-Knig»

В приложении на данный момент доступно более 20 000 аудиокниг. Классика, фантастика, юмористические и детские книги – все это и многое другое в mp3 формате и бесплатно. Кроме книг разных жанров на сайте есть

аудиоспектакли и курсы по изучению иностранных языков. Найти интересующую литературу можно через поиск или по категориям. Для скачивания аудиокниг не требуется регистрация.

Также имеется возможность прослушивать аудиокниги прямо на сайте. Зарегистрированный пользователь может сохранить понравившуюся аудиокнигу в закладки. В приложении присутствует гибкая система оценивания аудиокниг на основе нескольких параметров.

Преимущества:

- возможность полностью прослушать аудиокнигу без скачивания;
- гибкая система оценок;
- возможность скачивания книги прямо с сайта;
- все книги абсолютно бесплатны;
- возможность сортировки и фильтрации книг.

Недостатки:

- нельзя добавить на сайт собственную версию аудиокниги;
- отсутствует возможность заказать добавление книги у администрации.

«Baza-Knig» несомненно одно из лучших для хранения и распространения аудиокниг в плане количества книг, так и удобства использования. Для тех, кто хочет не только скачать аудиокнигу, но и прослушивать произведения прямо на сайте это прекрасный выбор.

Также стоит обратить внимание на такое приложение как «Au-books» (рисунок 1.3) [8].

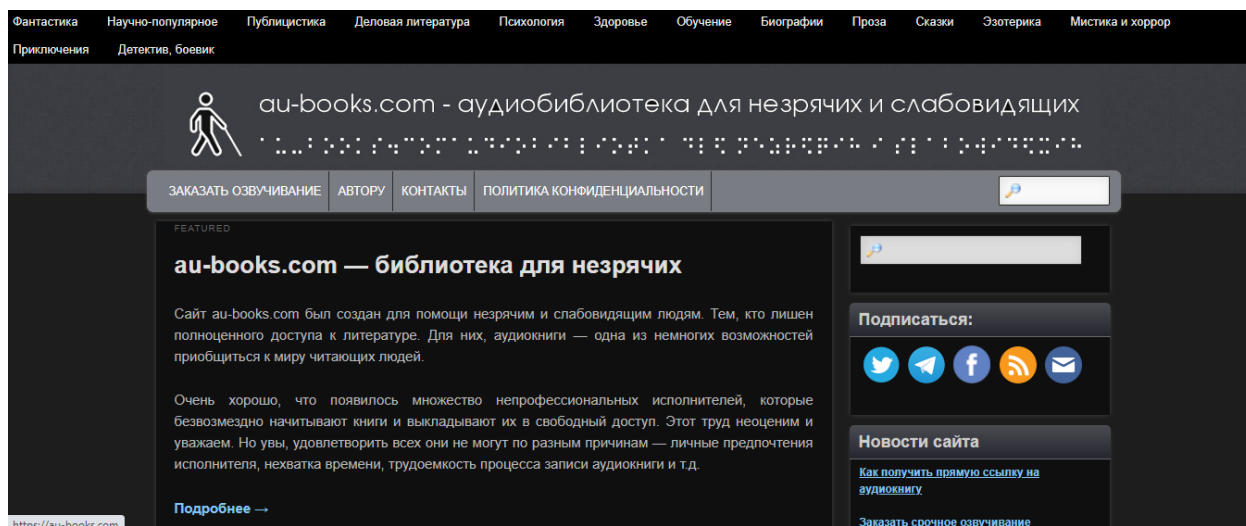


Рисунок 1.3 – Приложение для хранения и распространения аудиокниг «Au-books»

Этот сервис предназначен в первую очередь для незрячих и слабовидящих людей. Для них аудиокниги — это возможность приобщиться к миру читающих людей. Сегодня на сайте есть множество книг: фантастика, научно-популярная, детская, приключение, биографии, обучение и многое

другое. Найти интересующую литературу можно через поиск или по разделам.

Скачивание книг абсолютно бесплатно и не требует регистрации. Имеется возможность заказать синтез аудиокниги, с использованием одного из 7 голосов, представленных на сайте. Синтез книги возможен на русском и украинском языках. Среднее время озвучивания одной книги – сутки.

На данный момент на сайте размещено около 100 000 книг.

Преимущества:

- возможность заказать озвучивание любой книги;
- огромное количество аудиокниг, размещенных на сайте;
- возможность сортировки и фильтрации книг;
- все книги абсолютно бесплатны.

Недостатки:

- нельзя загрузить аудиофайл напрямую, на сайте размещены только ссылки на скачивание со сторонних ресурсов;
- нельзя добавить на сайт собственную версию аудиокниги;
- непонятный интерфейс.

«Au-books» обладает наибольшим объёмом фонда аудиокниг среди аналогов, и в целом является одним из лучших приложений для хранения и распространения аудиокниг. Однако некоторые проблемы вызывает пользовательский интерфейс, в частности непонятен принцип регистрации на сайте.

И так, рассмотрим приложения для озвучивания текста. Одним из наиболее доступных среди них является «VoxWorker» (рисунок 1.4) [9].

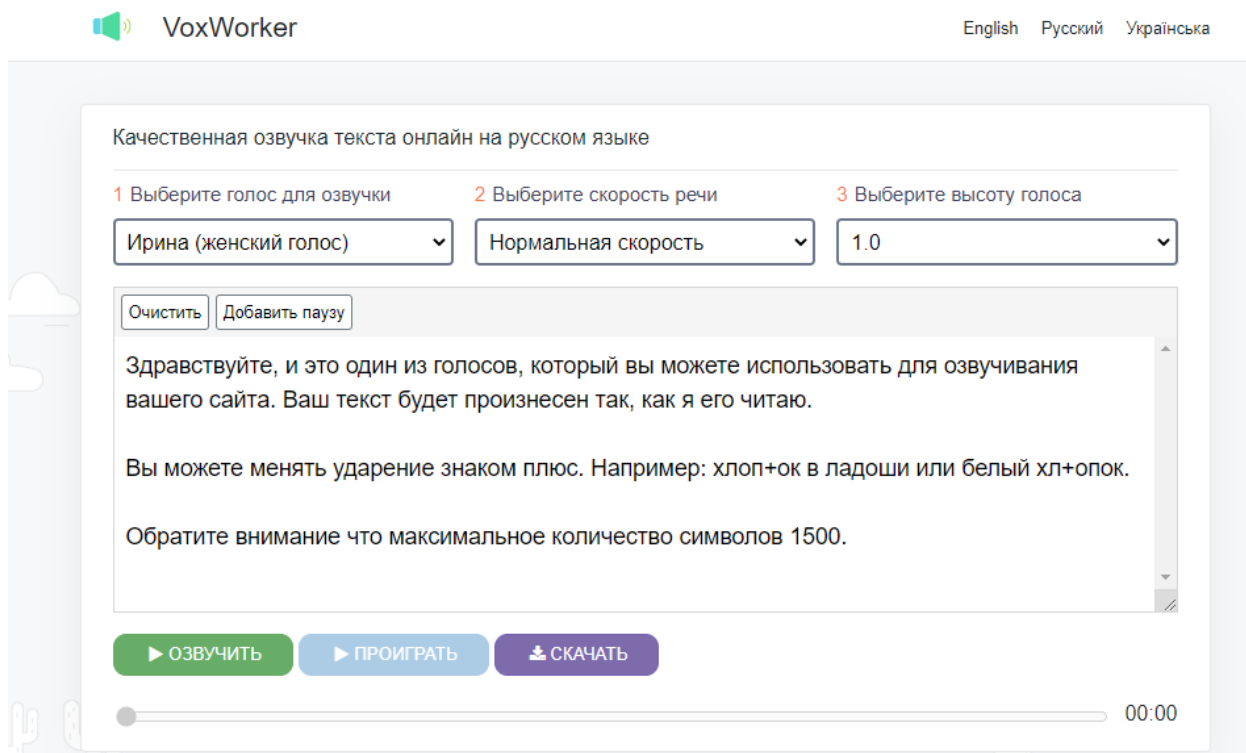


Рисунок 1.4 – Приложения для озвучивания текста «VoxWorker»

«VoxWorker» – это онлайн сервис для озвучки текста, который может переводить текст в аудиозапись. Сайт адаптирован для любых гаджетов, поэтому удобно работать и с мобильной версией. Функционал приложения подразумевает озвучивание небольших фрагментов текста (до 1500 символов).

Для синтеза речи можно выбрать мужской или женский голоса с разным тембром или акцентом. Можно изменять скорость и высоту речи. Озвучивать можно тексты на русском и английском языках.

Результат озвучки можно скачать как файл формата mp3, самого популярного формата для аудиозаписей. Или прослушать прямо в приложении. Сервис не сохраняет тексты для озвучивания. Все голосовые файлы удаляются с сервера через один час.

Преимущества:

- 11 различных голосов для озвучивания;
- возможность настраивать скорость речи и высоту голоса;
- возможность скачать полученный аудио файл;
- возможность прослушать результат прямо на сайте.

Недостатки:

- малый объём озвучиваемого текста;
- нельзя сохранить результат на сервере приложения.

В целом, «VoxWorker» удобный синтезатор речи с минималистичным дизайном и широким функционалом. Однако объем текста, озвучиваемого приложением, за один раз, очень мал.

Так же стоит обратить внимание на такое приложение как «UniTools» (рисунок 1.5) [10].

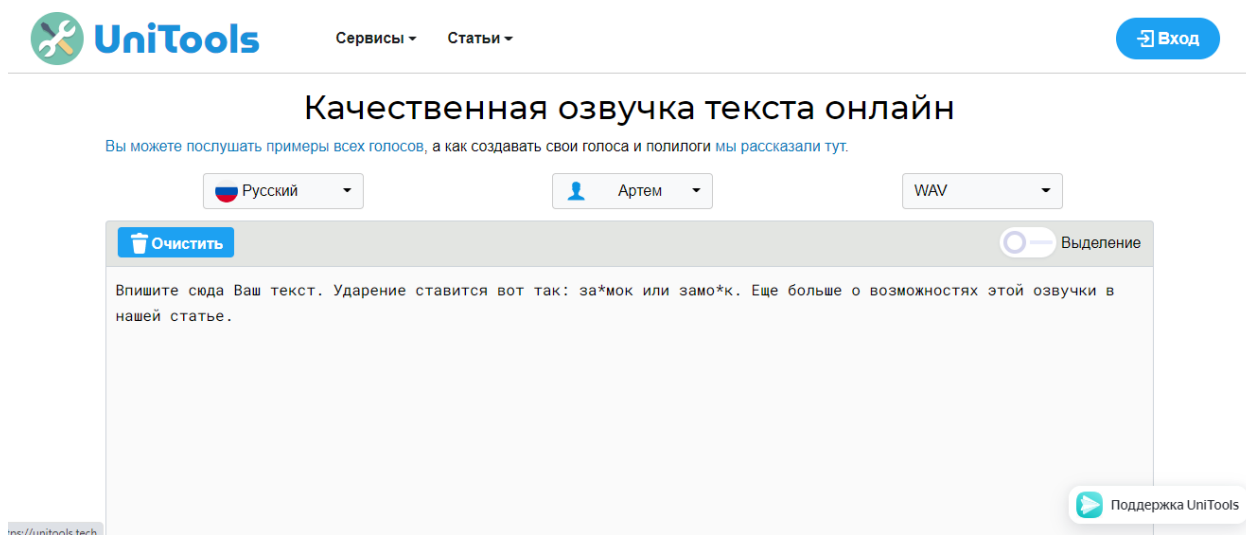


Рисунок 1.5 – Приложения для озвучивания текста «UniTools»

Пожалуй, «Unitools» является одним из наиболее интересных проектов для озвучки. Приложение интуитивно понятно и не вызывает затруднений при использовании даже у начинающих.

Использовать сервис удобно. Озвучка осуществляется голосами, которые легко можно спутать со звучанием речи живого человека. Имеется возможность сохранить результат озвучивания в форматы MP3 и WAV.

Система предусматривает работу в бесплатной и платной версии. Первая отлично подходит для начинающих, а также для тех, кому нужно перевести в аудио формат небольшой фрагмент текста. В день возможно воспроизведение до 750 символов на безвозмездной основе.

Использовать платные возможности сервиса могут зарегистрированные пользователи после пополнения баланса. Стоимость озвучки весьма демократична. Перевод текста в звук обычным голосом обойдется в 1 рубль за 1000 символов. Цена на премиум голоса – 3 рубля за 1000 символов.

Пополнение баланса осуществляется удобным способом и занимает пару минут. Количество доступных для озвучивания символов определяется только балансом на счете.

Преимущества:

- более сорока голосов для озвучивания на семи различных языках;
- возможность скачать полученный аудио файл;
- возможность прослушать результат прямо на сайте;

Недостатки:

- объем озвучиваемого текста в бесплатной версии;
- нельзя сохранить результат на сервере приложения.

«Unitools» хорошее приложение для озвучивания текста с удобным интерфейсом и большим количеством голосов. Однако, при бесплатном использовании, объема озвучиваемого текста хватит лишь на очень короткие тексты.

Ещё одним прекрасным приложением для озвучивания текста является «Zvukogram» (рисунок 1.6) [11].

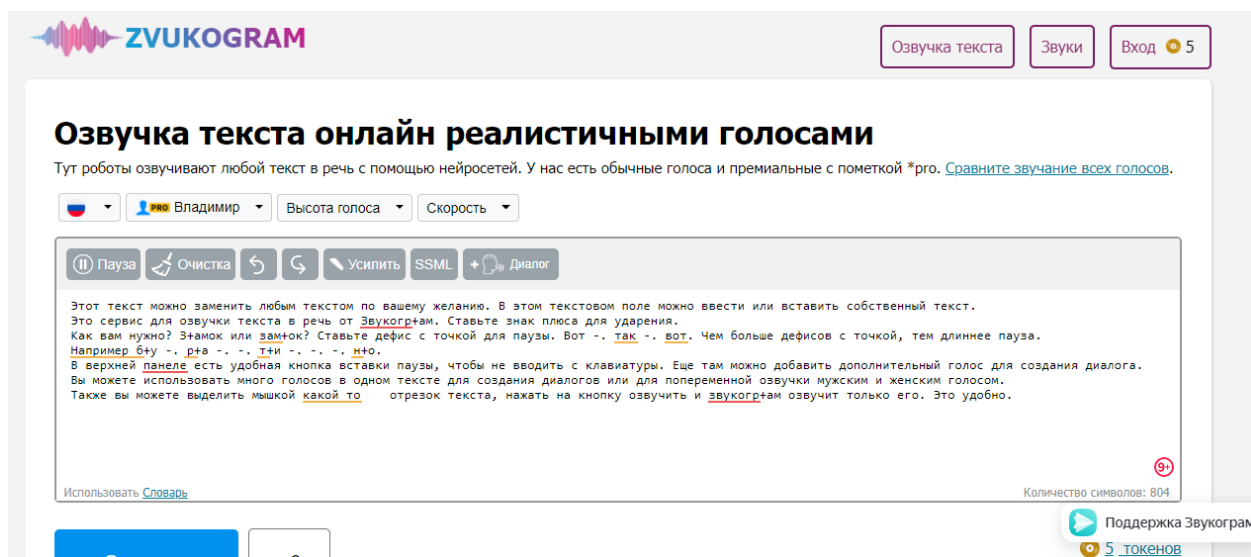


Рисунок 1.6 – Приложения для озвучивания текста «Zvukogram»

«Zvukogram» – это платный сервис, с возможностью бесплатного озвучивания небольшого пробного теста. Интересной особенностью данного веб-приложения является то, что в нем используется сразу две модели синтеза речи, компилятивная и форматно-голосовая. На сайте доступно около 150 голосов, на 11 языках. В сервисе нет ограничений. Можно сразу озвучить большую статью и все это будет в одном файле.

Изначально каждому пользователю доступно озвучивание 5000 символов простым голосом. При регистрации на счет пользователя зачисляется еще 10000 символов. Один символ озвученный премиум голосом равен 5 простым символам. Стоимость одной тысячи символов один рубль.

В отличие от любых других приложений для озвучивания текста, «Zvukogram» позволяет озвучивать текст сразу несколькими голосами, в том числе на разных языках. Это позволяет озвучивать диалоги, и обучающие тексты, содержащие несколько языков.

При озвучивании приложение позволяет добавлять в текст паузы и различные звуковые эффекты.

Преимущества:

- более 150 голосов для озвучивания на 11 различных языках;
- возможность скачать полученный аудио файл в нескольких форматах;
- возможность прослушать результат прямо на сайте;
- возможность при озвучивании построить диалог используя несколько голосов одновременно;
- добавление в тексты различных звуковых эффектов.

Недостатки:

- это платное приложение и объем бесплатно озвучиваемого текста ограничен;
- нельзя сохранить результат на сервере приложения.

Для тех, кто готов платить за озвучивание текста, «Zvukogram» несомненно является лучшим из существующих приложений для озвучивания текста. Этот сервис предоставляет чрезвычайно широкий функционал для профессионального озвучивания текста. А возможность добавления звуковых эффектов и использование нескольких голосов при озвучке, выгодно выделяет его среди аналогов.

1.3 Цели и задачи дипломного проекта. Формирование требований к приложению

1.3.1 Назначение разработки

Подводя итоги всего сказанного ранее, можно определить основные пользовательские требования и сформировать техническое задание для приложения, разрабатываемого в данном проекте.

Данное приложение предназначено для хранения и распространения аудиокниг в сети интернет, а также их синтеза на основе текста книги.

Основными целями создания данного приложения являются:

- 1) предоставление возможности удалённого хранения аудиокниг;
- 2) распространение аудиокниг в сети интернет;
- 3) предоставление возможности синтеза аудиокниг по текстовому файлу;
- 4) повышение доступности литературных произведений для людей с ограниченными возможностями.

К особенностям разрабатываемого программного средства, в сравнении с уже существующими аналогами можно отнести следующее:

- совмещение функционала приложения для хранения и распространения и приложения для синтеза речи;
- возможность распространить собственную версию аудиокниги среди пользователей сети Internet;
- возможность хранить аудиокниги в профиле пользователя.

1.3.2 Состав выполняемых функций

1.3.2.1 Синтез аудиокниги

При синтезе аудиокниги должна присутствовать возможность настройки голоса и темпа речи. Также, должна иметься возможность скачать синтезированную аудиокнигу на устройство пользователя.

1.3.2.2 Регистрация

При прохождении регистрации требуется удостовериться в корректности и уникальности введенного пользователем никнейма. В случае, если пользователь с таким никнеймом уже существует, должно отобразиться сообщение об ошибке. После окончания регистрации пользователь должен быть перенаправлен на авторизацию.

1.3.2.3 Авторизация

При прохождении авторизации требуется удостовериться в корректности введенного пользователем никнейма, а также в существовании пользователя с таким никнеймом и введенным паролем. После окончания авторизации должен быть отображен список распространяемых аудиокниг.

1.3.2.4 Добавление аудиокниги

Форма для добавления аудиокниги должна содержать следующие поля:

- название книги;
- авторы книги;
- жанры;
- год издания;
- краткое описание;
- поле для выбора файла обложки;
- поле выбора файла аудиокниги.

При добавлении аудиокниги требуется проверить корректность данных введенных пользователем. В случае, если имеются некорректные данные, должно отобразиться сообщение об ошибке. В зависимости от статуса пользователя, добавленная аудиокнига должна отобразиться в списке аудиокниг, хранимых пользователем или списке распространяемых аудиокниг.

1.3.2.5 Отображение списка аудиокниг, хранимых пользователем

В списке аудиокниг, хранимых пользователем, должна отображаться краткая информация о каждой хранимой книге.

Краткая информация об аудиокниге включает в себя:

- название книги;
- наименования авторов книги;
- изображение обложки;
- жанры аудиокниги;
- длительность аудиокниги;
- рейтинг аудиокниги;
- краткое описание аудиокниги.

При отображении должна присутствовать возможность сортировки и фильтрации списка аудиокниг. Кроме того, должна присутствовать возможность просмотреть и отредактировать подробную информацию об аудиокниге, а также возможность удалить аудиокнигу. И в довершение всего, должна присутствовать возможность отправить запрос на внесение аудиокниги в список распространяемых.

1.3.2.6 Отправка запроса на внесение аудиокниги в список распространяемых

Запрос должен содержать полную информацию о книге, а также никнейм пользователя и дату отправки запроса.

1.3.2.7 Отображение списка распространяемых аудиокниг

Список распространяемых аудиокниг должен быть доступен для любого пользователя сети Internet. В списке должна отображаться краткая информация о каждой распространяемой аудиокниге. Также, любому пользователю, должна быть доступна возможность просмотреть подробную информацию о любой аудиокниге из списка.

Кроме того, для пользователя с правами администратора, должны присутствовать возможность отредактировать информацию об аудиокниге, и возможность удалить аудиокнигу.

1.3.2.8 Сортировка списка аудиокниг

Сортировка может осуществляться по рейтингу, названию, году издания или дате добавления аудиокниги. По окончании сортировки, порядок элементов списка должен быть изменён согласно выбранному критерию.

1.3.2.9 Фильтрация списка аудиокниг

По окончании фильтрации должны отображаться только те, элементы списка, которые соответствуют выбранному критерию. Критериями для фильтрации могут выступать:

- название аудиокниги;
- жанр;
- наименование автора книги;
- год издания.

1.3.2.10 Отображение подробной информации об аудиокниге

При просмотре подробной информации об аудиокниге должна присутствовать возможность скачать аудиокнигу, а также возможность её оценить.

Подробная информация об аудиокниге содержит:

- название книги;
- изображение обложки;
- наименование авторов книги;
- жанры аудиокниги;
- год издания;
- краткое описание;
- рейтинг аудиокниги;
- длительность аудиокниги;
- дату добавления;
- никнейм пользователя, добавившего аудиокнигу.

1.3.2.11 Редактирование информации об аудиокниге

При редактировании информации об аудиокниге все поля и значения должны быть корректно загружены и отображены. До окончания редактирования должна иметься возможность отменить внесение изменений. После окончания редактирования новая версия информации об аудиокниге должна быть сохранена и отображена.

1.3.2.12 Удаление аудиокниги

Перед удалением пользователь должен подтвердить необходимость выполнения данной функции. По окончании удаления должен быть отображен обновлённый список аудиокниг.

1.3.2.13 Редактирование профиля пользователя

При редактировании профиля пользователя все поля и значения должны быть корректно загружены и отображены. Для окончания редактирования требуется получить подтверждение пользователя. После окончания редактирования новая версия профиля должна быть сохранена и отображена.

1.3.2.14 Скачивание аудиокниги

Перед началом скачивания должен быть отображен формат и размер скачиваемого файла, после чего пользователь должен подтвердить желание скачать файл.

1.3.2.15 Оценивание аудиокниги

По окончании оценивания, рейтинг аудиокниги должен быть пересчитан и сохранен в базе данных. После чего отображаемое значение рейтинга должно быть обновлено.

1.3.2.16 Просмотр профиля

В профиле пользователя должны отображаться никнейм и email. Также должна присутствовать возможность отредактировать профиль, и возможность отобразить список хранимых книг.

1.3.2.17 Выход из профиля

В приложении должна иметься возможность выйти из профиля в любой момент времени. После выхода из профиля, должен быть отображен список распространяемых книг.

1.3.2.18 Отображение списка запросов на внесение аудиокниги в список распространяемых

Элементы списка запросов должны содержать подробную информацию о добавляемой аудиокниге, никнейм отправителя и время отправления запроса.

1.3.2.19 Подтверждение внесения аудиокниги в список распространяемых

После подтверждения внесения аудиокниги в список распространяемых, запрос на осуществление внесения должен быть удален.

1.3.3 Требования к входным данным

Входные данные для приложения должны быть представлены в виде вводимого пользователем с клавиатуры текста, текстовых файлов соответствующего формата и опций, предоставляемых пользовательским интерфейсом приложения. В приложении должны быть реализованы проверки входных данных на корректность, и в случае их некорректности, пользователь должен получать соответствующее уведомление с просьбой ввести данные необходимого формата.

1.3.4 Требования к выходным данным

Выходные данные должны быть представлены в виде аудиофайлов соответствующего формата, файловых архивов, содержащих аудиофайлы, а

также посредством отображения информации при помощи различных элементов реализованного и доступного пользовательского интерфейса.

1.3.5 Требования к надежности

Для обеспечения надежности приложения требуется обеспечить бесперебойное питание технического средства, и своевременные проверки оборудования на наличие вирусных программ.

Время восстановления после отказа, вызванного сбоем электропитания технических средств, или не фатальным сбоем операционной системы, не должно превышать 60-ти минут при условии соблюдения условий эксплуатации технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, или фатальным сбоем операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Веб-приложение не должно непредвиденно прерывать свою работу. Отказы приложения вследствие некорректных действий пользователя при взаимодействии с приложением через веб-интерфейс недопустимы.

1.3.6 Технические требования

Архитектура всей системы должна отвечать следующим требованиям:

- 1) централизованная база данных;
- 2) организация доступа к компонентам системы через внешний канал связи (Internet);
- 3) разделение бизнес логики, обработки и представления данных;
- 4) безопасность;
- 5) надёжность.

Для обеспечения работы системы требуются технические средства для размещения базы данных и серверной части системы. Должна быть обеспечена круглосуточная работа приложения.

Требования к техническому обеспечению серверной части системы:

- 1) процессор Intel Core i5 с тактовой частотой 2 ГГц или более мощный;
- 2) оперативная память в объеме 4Гбайт или более;
- 3) свободное место на жестком диске в объеме не менее 20 ГБ;
- 4) постоянное подключение к сети Internet.

Требования к программному обеспечению серверной части:

- 1) ОС Ubuntu версии 18.04 или выше;
- 2) СУБД MySQL версии 8.0 или выше.

Требования к техническому и программному обеспечению устройства клиента:

- 1) стабильное подключение к сети Internet;
- 2) браузер с поддержкой HTML5 и JavaScript.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Функциональная модель программного средства

Построение приложения подразумевает проектирование его функциональной модели. Функциональная модель приложения представлена в виде диаграмм вариантов использования и информационной модели предметной области.

2.1.1 Варианты использования приложения

Варианты использования данного приложения различаются в зависимости от статуса пользователя. А статус, в свою очередь, зависит от того была ли пройдена авторизация, и от роли авторизованного пользователя. Все возможные статусы отображены на рисунке 2.1.

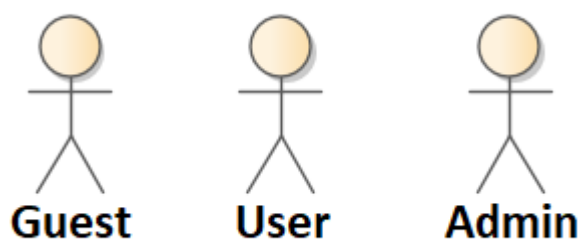


Рисунок 2.1 – Все возможные статусы пользователя

Чтобы пользователи могли зарегистрироваться и авторизоваться, необходимо предусмотреть наличие в базе данных специальных сущностей, предназначенных для хранения информации о пользователе. Диаграмма сущностей, участвующих в процессах регистрации и авторизации пользователя, отображена на рисунке 2.2.

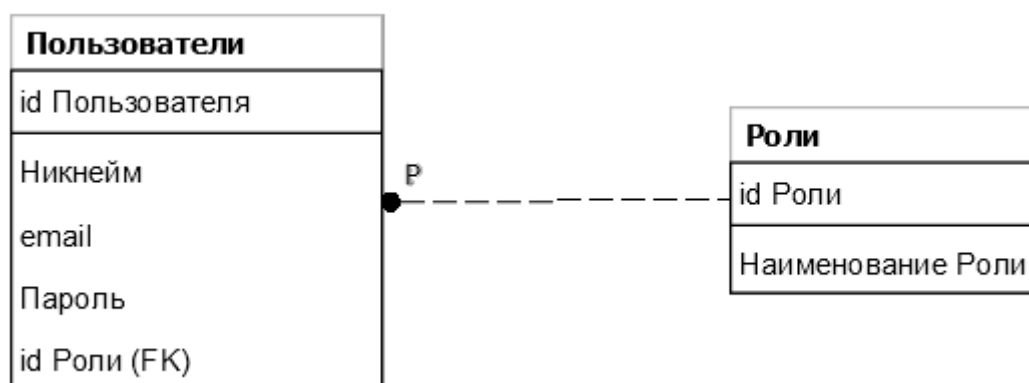


Рисунок 2.2 – Диаграмма сущностей хранящих информацию о пользователе

На данной диаграмме отображены две сущности. Сущность Пользователи представляет собой каждого пользователя, зарегистрировавшегося в приложении, а сущность Роли представляет все возможные роли пользователя.

Любой пользователь, использующий приложение, изначально имеет статус «Guest». Такой пользователь не имеет доступа к синтезу аудиокниг, не может оценивать аудиокниги и не способен добавить аудиокнигу в приложение. Для смены статуса требуется пройти авторизацию. Все действия, которые может выполнять пользователь со статусом «Guest» указаны на рисунке 2.3.



Рисунок 2.3 – Диаграмма вариантов использования приложения пользователем со статусом «Guest»

После прохождения авторизации статус пользователя меняется на «User» или «Admin», это зависит от роли пользователя хранимой в базе данных. Все действия, которые может выполнять пользователь с ролью «User» указаны на рисунке 2.4.

В приложении должна быть предусмотрена административная часть. К ней имеют доступ только те пользователи, которые обладают статусом «Admin». Функциональность доступная пользователям со статусом «Admin» представлена на рисунке 2.5.

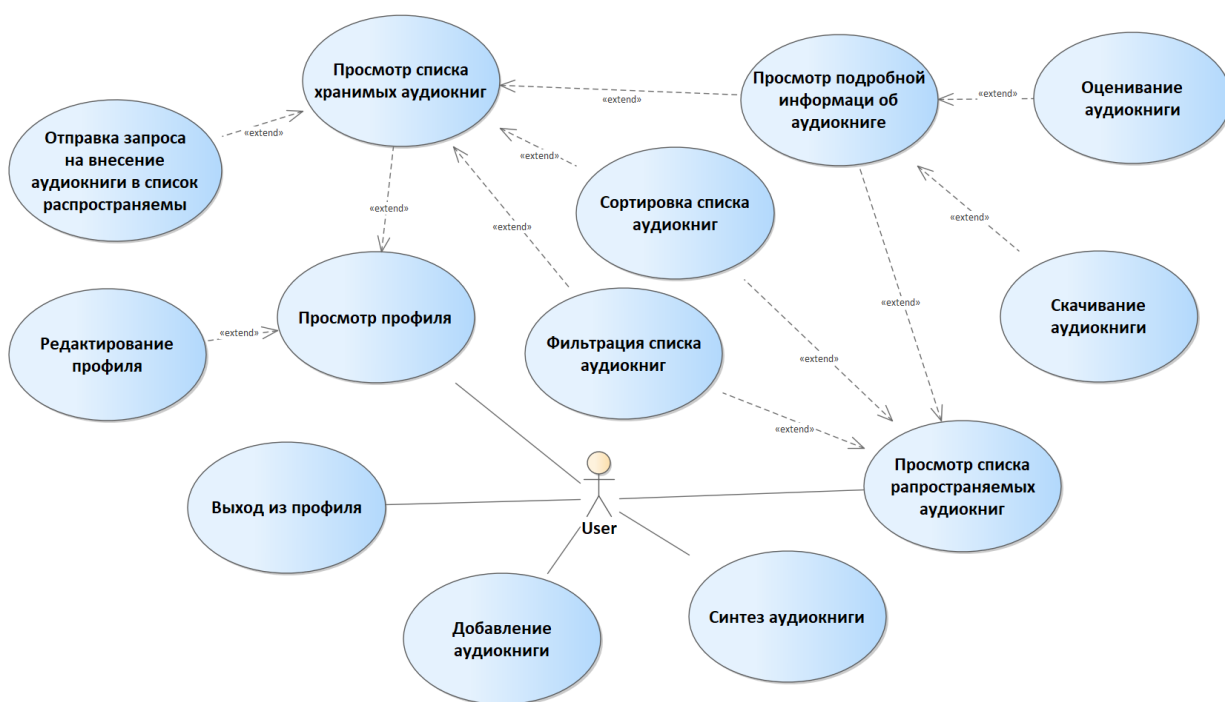


Рисунок 2.4 – Диаграмма вариантов использования приложения пользователем со статусом «User»

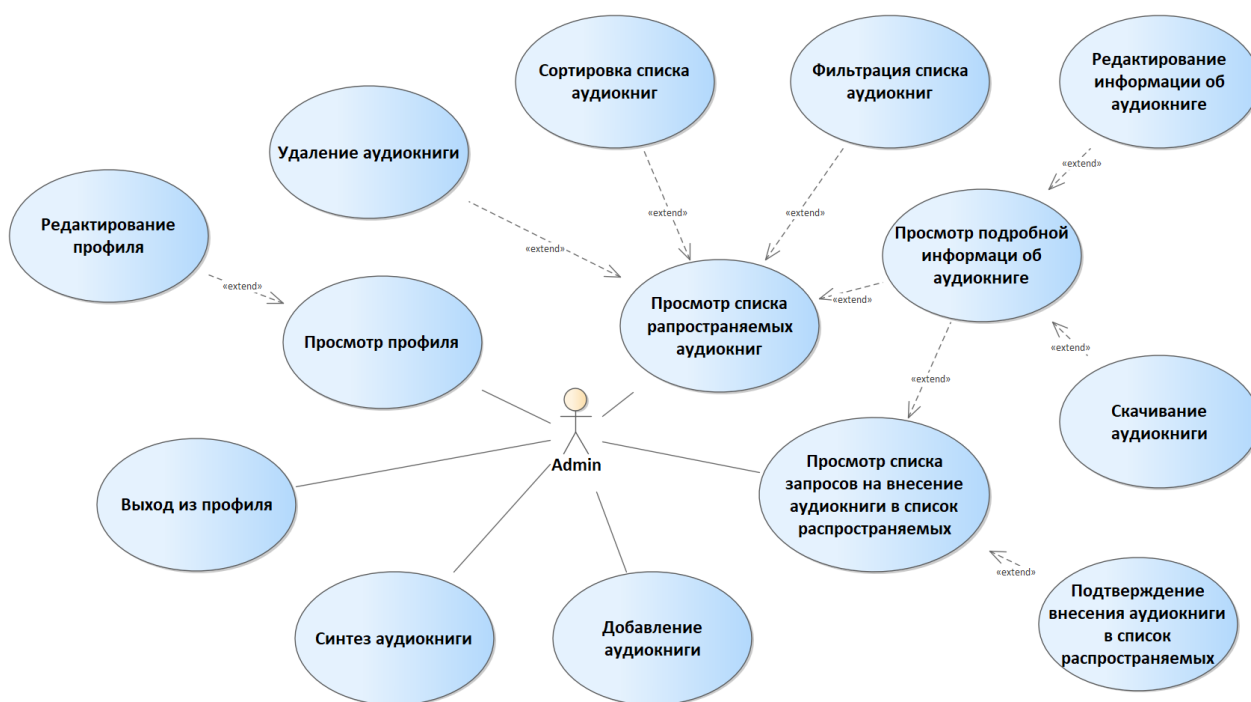


Рисунок 2.5 – Диаграмма вариантов использования приложения пользователем со статусом «Admin»

2.1.2 Разработка инфологической модели базы данных

Исходя из необходимости использования в проектируемом приложении базы данных, разработаем ее инфологическую модель. Для создания данной модели возьмем за основу предметную область проекта.

Предметная область разрабатываемого программного средства включает в себя следующие сущности и их атрибуты:

1) пользователь:

- уникальный идентификатор;
- никнейм;
- хешированный пароль;
- адрес электронной почты;
- идентификатор роли пользователя;

2) роль пользователя:

- уникальный идентификатор;
- наименование роли;

3) аудиокнига:

- уникальный идентификатор;
- идентификатор пользователя, добавившего аудиокнигу;
- является ли книга распространяемой;
- название книги;
- путь к файлу содержащему изображение обложки;
- год издания;
- краткое описание;
- дата добавления;
- идентификатор файла аудиокниги;
- рейтинг аудиокниги;

4) аудиокнига, хранимая пользователем:

- идентификатор пользователя;
- идентификатор аудиокниги;

5) файл аудиокниги:

- уникальный идентификатор;
- расширение файла;
- путь к файлу содержащему аудиокнигу;

6) жанр:

- уникальный идентификатор;
- наименование жанра;

7) автор:

- уникальный идентификатор;
- наименование автора;

8) жанр аудиокниги:

- идентификатор жанра;
- идентификатор аудиокниги;

9) автор аудиокниги:

- идентификатор автора;
- идентификатор аудиокниги;

10) оценка:

- идентификатор пользователя;
- идентификатор книги;
- значение;

11) запрос на внесение аудиокниги в список распространяемых:

- идентификатор пользователя;
- идентификатор аудиокниги;
- дата и время отправления запроса.

2.2 Спецификация функциональных требований

С учетом требований, определенных в подразделе 1.3, представим детализацию функций проектируемого ПС.

Для детализации функций рассмотрим основные требования, предъявляемые к каждой функции программного средства как с точки зрения внутренней организации системы, так и с точки зрения взаимодействия системы с пользователем.

2.2.1 Синтез аудиокниги

Функция синтеза аудиокниги должна быть реализована с учетом следующих требований:

- 1) процесс синтеза аудиокниги может быть инициирован пользователем системы со статусом «User» или «Admin»;
- 2) должна присутствовать возможность настройки параметров работы синтезатора речи, а именно скорости произношения и высоты интонации;
- 3) должна присутствовать возможность выбора голоса, используемого при синтезе, из списка доступных;
- 4) должна присутствовать возможность синтезировать и прослушать пробный текст с настройками синтезатора, установленными пользователем;
- 5) для синтеза аудиокниги пользователь должен предоставить текстовый файл в формате txt или fb2, размером до 900 мегабайт;
- 6) результатом синтеза является один аудиофайл в формате mp3, или архив в формате zip, содержащий несколько mp3 файлов;
- 7) должна присутствовать возможность скачать синтезированную аудиокнигу на устройство пользователя.

2.2.2 Регистрация

Функция регистрации должна быть реализована с учетом следующих требований:

- 1) процесс регистрации инициируется пользователем системы со статусом «Guest»;
- 2) для регистрации пользователь должен предоставить уникальный никнейм, а также дважды ввести пароль;
- 3) корректность введенных данных должна быть проверена при помощи

встроенных инструментов разработки;

4) корректным считается никнейм состоящий из латинских символов и цифр, а также символа «_», длиной не более 15 символов;

5) корректным паролем считается последовательность, состоящая из латинских символов и цифр, а также символа «_», длиной от 10 до 20 символов;

6) в случае некорректности введенных данных пользователь должен увидеть сообщение об этом с предложением попробовать еще раз;

7) необходимо удостовериться в уникальность введенного никнейма;

8) в случае если никнейм не является уникальным пользователь должен увидеть сообщение об этом с предложением изменить никнейм;

9) по окончании регистрации пользователь должен быть перенаправлен на авторизацию.

2.2.3 Авторизация

Функция регистрации должна быть реализована с учетом следующих требований:

1) процесс авторизации инициируется пользователем системы со статусом «Guest»;

2) для прохождения авторизации пользователь должен ввести свой никнейм и пароль;

3) корректность введенных данных должна быть проверена при помощи встроенных инструментов разработки;

4) корректным считается никнейм состоящий из латинских символов и цифр, а также символа «_», длиной не более 20 символов;

5) корректным паролем считается последовательность, состоящая из латинских символов и цифр, а также символа «_», длиной от 10 до 20 символов.

6) в случае некорректности введенных данных пользователь должен увидеть сообщение об этом с предложением попробовать еще раз;

7) необходимо удостовериться в существовании пользователя, с никнейм и паролем, соответствующим введенным;

8) в случае, если пользователь с никнеймом и паролем, соответствующим введенным, не существует, должно отобразиться сообщение некорректности введенных данных;

9) по окончании авторизации пользователя должен быть отображен список распространяемых аудиокниг;

10) по окончании авторизации статус пользователя должен быть изменен на «User» или «Admin», в зависимости от ассоциированной с его профилем роли.

2.2.4 Добавление аудиокниги

Функция добавления аудиокниги должна быть реализована с учетом следующих требований:

1) процесс добавления аудиокниги может быть инициирован пользователем системы со статусом «User» или «Admin»;

2) для добавления аудиокниги должен предоставить следующую информацию об аудиокниге:

- название книги;
- наименования авторов книги;
- жанры книги;
- год издания;
- краткое описание;

3) должна присутствовать возможность загрузить обложку аудиокниги, обложкой является файл изображения в формате png;

4) для добавления аудиокниги пользователь должен предоставить файл содержащий аудиокнигу, это может быть mp3 файл или архив в формате zip;

5) название книги представляет собой последовательность длиной от 3 до 40 символов состоящую из букв кириллического и латинского алфавитов, а также знаков препинания и пробельных символов;

6) должна присутствовать возможность ассоциировать с книгу с одним или несколькими авторами;

7) наименование автора, это последовательность длиной от 3 до 40 символов состоящая из букв кириллического алфавита, знаков препинания и пробельных символов;

8) если в базе данных приложения отсутствует автор с наименованием, соответствующим введённому пользователем, тогда в базу данных должен быть добавлен новый автор с соответствующим наименованием;

9) должна присутствовать возможность ассоциировать книгу с одним или несколькими жанрами, выбираемыми из списка доступных на сайте;

10) год издания представляет собой число в промежутке 1800 до 2100;

11) краткое описание представляет собой последовательность длиной от 0 до 2000 символов состоящую из букв кириллического алфавита, знаков препинания и пробельных символов;

12) данные, введенные пользователем, должны проверяться на корректность, если имеются некорректные данные, должно отобразиться сообщение об ошибке;

13) аудиокнига, добавленная пользователем со статусом «User» должна отобразиться в списке аудиокниг, хранимых пользователем;

14) аудиокнига, добавленная пользователем со статусом «Admin» должна отобразиться в списке распространяемых аудиокниг.

2.2.5 Отображение списка аудиокниг, хранимых пользователем

Функция отображения списка аудиокниг, хранимых пользователем должна быть реализована с учетом следующих требований:

1) доступом к работе со списком хранимых книг должны обладать только пользователи со статусом «User»;

2) в списке аудиокниг, хранимых пользователем, должна отображаться краткая информация о каждой хранимой аудиокниге;

- 3) краткая информация о хранимой аудиокниге включает в себя:
 - название книги;
 - от 1 до 3 наименований авторов книги;
 - изображение обложки;
 - от 1 до 3 жанров аудиокниги;
 - длительность аудиокниги;
 - рейтинг аудиокниги;
 - краткое описание аудиокниги;
- 4) рейтинг аудиокниги представляет собой десятичную дробь с округлением до сотых, и значением в промежутке от 1 до 10;
- 5) должна быть реализована пагинация списка с отображением:
 - текущей страницы;
 - первой страницы пагинации;
 - последней страниц пагинации;
- 6) на одной странице пагинации должно отображаться не более 10 книг;
- 7) по нажатию на название аудиокниги или изображение обложки должна быть отображена подробная информация о выбранной книге;
- 8) для каждого элемента списка должна присутствовать возможность инициировать редактирование информации о книге;
- 9) для каждого элемента списка должна присутствовать возможность инициировать удаление аудиокниги;
- 10) для тех аудиокниг, которые не являются распространяемыми, должна присутствовать возможность отправить запрос на внесение книги в список распространяемых;
- 11) должна присутствовать возможность фильтрации списка;
- 12) должна присутствовать возможность сортировки списка.

2.2.6 Отправка запроса на внесение аудиокниги в список распространяемых

Функция отправки запроса на внесение аудиокниги в список распространяемых должна быть реализована с учетом следующих требований:

- 1) возможность инициировать отправку запроса должна иметься только у пользователей со статусом «User»;
- 2) запрос должен содержать дату и время отправления, а также информацию о книге и никнейм пользователя;
- 3) пользователь должен подтвердить желание отправить запрос;
- 4) после подтверждения отправки, запрос должен быть сохранен в базе данных.

2.2.7 Отображение списка распространяемых аудиокниг

Функция отображения списка распространяемых аудиокниг должна быть реализована с учетом следующих требований:

1) просмотр списка распространяемых аудиокниг, должен быть доступен любому пользователю независимо от статуса;

2) в списке распространяемых аудиокниг должна отображаться краткая информация о каждой распространяемой аудиокниге;

3) краткая информация о распространяемой аудиокниге включает в себя:

- название книги;
- от 1 до 3 наименований авторов книги;
- изображение обложки;
- от 1 до 3 жанров аудиокниги;
- длительность аудиокниги;
- рейтинг аудиокниги;
- краткое описание аудиокниги;
- никнейм пользователя, добавившего книгу;

4) рейтинг аудиокниги представляет собой десятичную дробь с округлением до сотых, и значением в промежутке от 1 до 10;

5) должна быть реализована пагинация списка с отображением:

- текущей страницы;
- первой страницы пагинации;
- последней страниц пагинации;

6) на одной странице пагинации должно отображаться не более 10 книг;

7) по нажатию на название аудиокниги или изображение обложки должна быть отображена подробная информация о выбранной книге;

8) для каждого элемента списка пользователю со статусом «Admin» должна быть доступна возможность инициировать редактирование информации о книге;

9) для каждого элемента списка пользователю со статусом «Admin» должна быть доступна возможность инициировать удаление аудиокниги из списка распространяемых.

10) должна присутствовать возможность фильтрации списка;

11) должна присутствовать возможность сортировки списка.

2.2.8 Сортировка списка аудиокниг

Функция сортировки списка аудиокниг должна быть реализована с учетом следующих требований:

1) сортировка должна выполняться на основе одного из следующих критериев:

- название книги;
- рейтинг;
- год издания;
- дате добавления;

2) должна присутствовать возможность сортировки как по возрастанию, так и по убыванию выбранного критерия;

3) при нажатии на критерий, сортировка по которому выполнена на данный момент, должна происходить сортировка в направлении обратном текущему направлению сортировки;

4) по окончании сортировки должен быть отображён список, порядок элементов которого изменен согласно параметрам сортировки.

2.2.9 Фильтрация списка аудиокниг

Функция фильтрации списка аудиокниг должна быть реализована с учетом следующих требований:

1) фильтрация должна выполняться на основе введенных или выбранных пользователем критериев;

2) критериями для фильтрации могут выступать:

- название книги;
- жанр;
- год издания;
- наименование автора;

3) фильтрация по введенному пользователем названию книги должна выполняться на основе частичного или полного соответствия названию аудиокниги, представленной в списке, без учёта регистра;

4) фильтрация по жанру должна выполняться на основе наличия у книги жанра, который пользователь выбрал из списка доступных в приложении;

5) фильтрация по наименованию должна выполняться на основе частичного или полного соответствия одному из наименований авторов, ассоциированных с аудиокнигой, представленной в списке, без учета регистра;

6) фильтрация по году должна выполняться на основе совпадения года издания книги, с годом введенным пользователем;

7) по окончании сортировки должен быть отображён список, все элементы которого соответствуют выбранным критериям.

2.2.10 Отображение подробной информации об аудиокниге

Данная функция должна быть реализована с учётом следующих требований:

1) подробная информация об аудиокниге включает в себя:

- название книги;
- наименования авторов книги;
- изображение обложки;
- все жанры аудиокниги;
- год издания;
- краткое описание;
- рейтинг аудиокниги;
- длительность аудиокниги;
- дату добавления;
- никнейм пользователя, добавившего аудиокнигу;

2) дата добавления должна быть представлена в формате «dd/mm/yyyy», где «dd» - двухзначный день месяца, «mm» - двухзначный номер месяца, «yyyy» - четырёхзначный номер года;

3) для пользователя со статусом «User» должна присутствовать возможность оценить книгу;

4) должна присутствовать возможность скачать аудиокнигу.

2.2.11 Редактирование информации об аудиокниге

Данная функция должна быть реализована с учётом следующих требований:

1) процесс редактирования информации об аудиокниге может быть инициирован либо пользователем, добавившим аудиокнигу, либо пользователем со статусом «Admin»;

2) при редактировании информации об аудиокниге все поля и значения должны быть корректно загружены и отображены;

3) информация об аудиокниге доступная для редактирования:

- название книги;
- наименования авторов книги;
- изображение обложки;
- жанры аудиокниги;
- год издания;
- краткое описание.

4) должна присутствовать возможность загрузить новую обложку аудиокниги, обложкой является файл изображения в формате png;

5) название книги представляет собой последовательность длиной от 3 до 40 символов состоящую из букв кириллического и латинского алфавитов, а также знаков препинания и пробельных символов;

6) должна присутствовать возможность ассоциировать с книгу с одним или несколькими новыми авторами, а также удалить уже существующие ассоциации;

7) наименование автора, это последовательность длиной от 3 до 40 символов состоящая из букв кириллического алфавита, знаков препинания и пробельных символов;

8) если в базе данных приложения отсутствует автор с наименованием, соответствующим введённому пользователем, тогда в базу данных должен быть добавлен новый автор с соответствующим наименованием;

9) должна присутствовать возможность ассоциировать книгу с одним или несколькими жанрами, выбираемыми из списка доступных на сайте, а также удалить уже существующие ассоциации;

10) год издания представляет собой число в промежутке 1800 до 2100;

11) краткое описание представляет собой последовательность длиной от 0 до 2000 символов состоящую из букв кириллического алфавита, знаков препинания и пробельных символов;

12) данные, введенные пользователем, должны проверяться на корректность, если имеются некорректные данные, должно отобразиться сообщение об ошибке;

13) до окончания редактирования должна иметься возможность отменить внесение изменений;

14) по окончании редактирования новая версия информации об аудиокниге должна быть сохранена в базу данных;

15) по окончании редактирования должна быть отображена обновленная версия подробной информации об аудиокниге.

2.2.12 Удаление аудиокниги

Данная функция должна быть реализована с учётом следующих требований:

1) процесс удаления аудиокниги может быть инициирован либо пользователем, добавившим аудиокнигу, либо пользователем со статусом «Admin»;

2) перед удалением пользователь должен подтвердить необходимость выполнения данной функции;

3) по окончании удаления должен быть отображен обновлённый список аудиокниг.

2.2.13 Редактирование профиля пользователя

Данная функция должна быть реализована с учётом следующих требований:

1) процесс редактирования профиля может быть инициирован только тем пользователем, которому принадлежит профиль;

2) при редактировании профиля пользователя все поля и значения должны быть корректно загружены и отображены;

3) информация о пользователе доступная для редактирования:

- никнейм;
- email;
- пароль;

4) данные, введенные пользователем, должны проверяться на корректность.

5) корректным считается никнейм состоящий из латинских символов и цифр, а также символа «_», длиной не более 15 символов;

6) корректным паролем считается последовательность, состоящая из латинских символов и цифр, а также символа «_», длиной от 10 до 20 символов;

7) корректным значением поля email является последовательность длиной до 64 символов, состоящая из букв латинского алфавита и цифр с добавлением в конец символа «@» и доменного имени (пример: mail.ru);

8) поле email является необязательным для заполнения;

9) в случае некорректности введенных данных пользователь должен увидеть сообщение об этом;

- 10) необходимо удостовериться в уникальность введенного никнейма;
- 11) в случае если никнейм не является уникальным пользователь должен увидеть сообщение об этом с предложением изменить никнейм;
- 12) до окончания редактирования должна иметься возможность отменить внесение изменений;
- 13) для подтверждения редактирования пользователь должен ввести старый пароль, и нажать на кнопку подтверждения;
- 14) по окончании редактирования новая версия профиля должна быть сохранена в базу данных;
- 15) по окончании редактирования должна быть отображена обновленная версия профиля пользователя.

2.2.14 Скачивание аудиокниги

Данная функция должна быть реализована с учётом следующих требований:

- 1) перед началом скачивания должен быть отображен формат и размер скачиваемого файла;
- 2) после ознакомления с размер и форматом файла пользователь должен подтвердить желание скачать файл;
- 3) файлом аудиокниги является аудиофайл в формате mp3, или архив формата zip, содержащий несколько файлов;
- 4) максимальный размер файла, доступного для скачивания, составляет 900 мегабайт.

2.2.15 Оценивание аудиокниги

Данная функция должна быть реализована с учётом следующих требований:

- 1) оценивать книги может только пользователь со статусом «User»;
- 2) по окончании оценивания, рейтинг аудиокниги должен быть пересчитан и сохранен в базе данных, после чего отображаемое значение рейтинга должно быть обновлено.

2.2.16 Просмотр профиля

Данная функция должна быть реализована с учётом следующих требований:

- 1) в профиле пользователя должны отображаться никнейм и email;
- 2) должна присутствовать возможность отредактировать профиль;
- 3) должна присутствовать возможность отобразить список аудиокниг, хранимых пользователем.

2.2.17 Выход из профиля

Данная функция должна быть реализована с учётом следующих требований:

- 1) выход из профиля может быть инициирован пользователями со статусом «User» или «Admin»;
- 2) возможность покинуть профиль должна присутствовать в любой момент времени, независимо от функции выполняемой приложением;
- 3) после выхода из профиля, статус пользователя должен смениться на «Guest»;
- 4) после выхода из профиля, должен быть отображен список распространяемых книг.

2.2.18 Отображение списка запросов на внесение аудиокниги в список распространяемых

Данная функция должна быть реализована с учётом следующих требований:

- 1) доступ к работе со списком запросов имеют только те пользователи, которые обладают статусом «Admin»;
- 2) элементы списка запросов должны содержать название аудиокниги, никнейм отправителя и время отправления запроса;
- 3) должна быть реализована пагинация списка с отображением:
 - текущей страницы;
 - первой страницы пагинации;
 - последней страниц пагинации;
- 4) на одной странице пагинации должно отображаться не более 50 запросов;
- 5) при нажатии на название книги должна быть отображена подробная информация об указанной аудиокниге;
- 6) должна присутствовать возможность подтвердить внесение аудиокниги в список распространяемых.

2.2.19 Подтверждение внесения аудиокниги в список распространяемых

Данная функция должна быть реализована с учётом следующих требований:

- 1) доступ к работе к данной функции имеют только те пользователи, которые обладают статусом «Admin»;
- 2) после подтверждения внесения аудиокниги в список распространяемых, статус аудиокниги в базе данных должен быть изменен;
- 3) при нажатии на название книги должна быть отображена подробная информация об указанной аудиокниге;
- 4) после подтверждения внесения аудиокниги в список распространяемых, запрос на осуществление внесения должен быть удален, а отображение списка запросов обновлено.

3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

3.1 Разработка архитектуры приложения

Для написания любого проекта очень важно, чтобы приложение не только правильно и быстро работало, но и было правильно организовано. Сложность, как правило, растет гораздо быстрее размеров программы. И если не позаботиться об этом заранее, то вскоре наступит момент, когда контролировать приложение станет невозможно.

Правильно подобранная архитектура экономит очень много сил, времени и ресурсов. Хорошая архитектура – это, прежде всего, выгодная архитектура, делающая процесс разработки и сопровождения программы более простым и эффективным. Приложение с правильно подобранной архитектурой легче расширять и изменять, а также тестировать, отлаживать и описывать.

Поэтому, как только были сформулированы основные функциональные требования проектируемого приложения, требуется разработать архитектуру приложения.

Качественная архитектура должна обладать следующими свойствами:

1) функциональность. Данное свойство представляет собой возможность реализации функциональных требований, указанных ранее, на основе данной архитектуры;

2) гибкость системы. Это свойство означает, что качественная архитектура должна обеспечить возможность быстрого и качественного внесения изменений в приложения, а также минимизировать количество ошибок, вызываемых изменениями приложения;

3) возможность независимого изменения. Это свойство заключается в том, что изменения, вносимые в работу одного из компонентов приложения, не должны затрагивать деятельность иных компонентов

4) удобство построения. Данное свойство говорит о том, что архитектура должна обеспечить возможность построения приложения таким образом, чтобы набор компонентов приложения мог реализовываться и тестироваться независимо друг от друга;

5) расширяемость. Возможность добавлять в систему новые сущности и функции, не нарушая ее основной структуры;

6) сопротивление энтропии – поддержание упорядоченности системы за счёт принятия, ограничения и изоляции последствий изменений;

7) модульность. Данное свойство определяет возможность приложения делиться на рабочие задания (модули), и особенно модули, которые могут разрабатываться независимо друг от друга, при этом легко и точно дополняя возможности друг друга;

8) безопасность – качественная архитектура предоставляет приложению возможность управления ограничением доступа к своим данным;

9) многоразовость компонентов – возможность повторного использования модулей системы.

Поскольку в качестве основной платформы функционирования приложения был выбран веб, основной архитектурой приложения будет являться клиент-серверный подход, реализующий концепцию «клиент-сервер» с двумя независимыми друг от друга точками входа, совмещенный с архитектурным пятерном MVC (рисунок 3.1).

Клиент-серверная архитектура – вид архитектуры приложений, при которой нагрузка распределена между двумя объектами: сервером и клиентом. Связь в паре клиент-сервер осуществляется посредством сетевых протоколов, каждый в этой паре является по сути программным обеспечением. К достоинствам такого подхода относят:

- невысокие технические и системные требования к машинам-клиентам, поскольку все вычисления производятся на сервере;
- безопасность хранения данных, т.к. базы данных расположены на сервере, который как правило защищен гораздо лучше, чем машины-клиенты;
- отсутствие необходимости дублирования исходного программного кода на машине-клиенте;
- возможность использования нескольких физических компьютеров, объединенных в одну сеть, в качестве сервера;
- быстрое обновление ПО на машинах-клиентах.

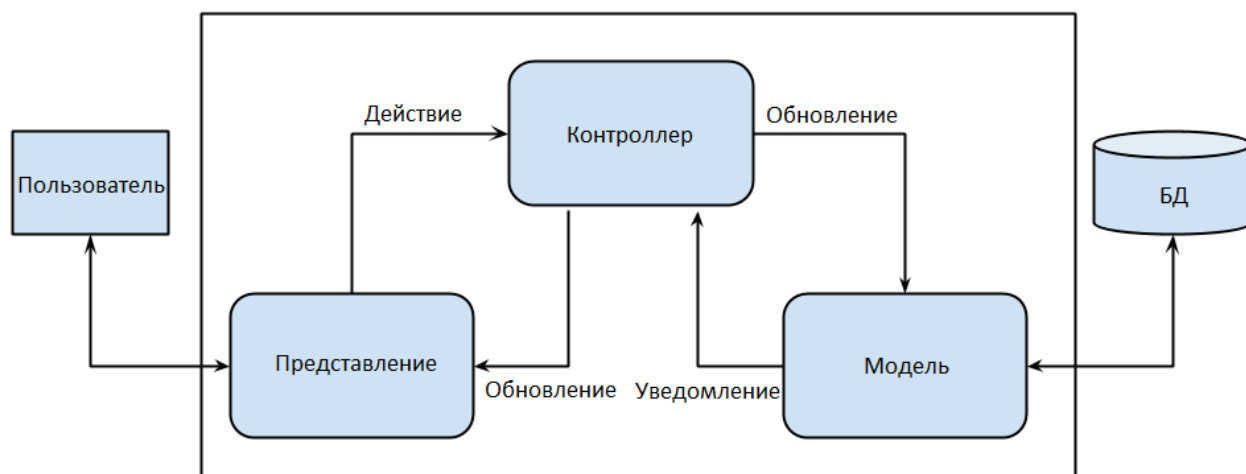


Рисунок 3.1 – Схема работы MVC

MVC – это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер. Изначально это был всего лишь шаблон проектирования, но в течение периода времени данный подход эволюционировал до архитектурного шаблона. Этот архитектурный паттерн является одним из самых популярных в среде веб-приложений. Приложения, основанные на дан-

ной архитектуре, легко сопровождать и дорабатывать, они гибкие и легко масштабируемые.

Модель, в контексте паттерна MVC, представляет данные и методы работы с ними. В веб-приложениях, модель, как правило, отвечает за работу с базами данных и файловой системой. По сути, модель просто предоставляет данные предметной области представлению и реагирует на команды контроллера, изменяя свое состояние.

За отображение данных, которые отдаёт модель, несёт ответственность представление. Зачастую представление описывается как шаблон, заполняемый данными. Оно позволяет менять внешний вид отображаемой информации, но не оказывает влияния на саму информацию. В веб-приложениях представление зачастую реализуется в виде HTML-страницы, но иногда может быть представлено данными в формате JSON или XML.

Модель и представление связываются контроллером. Контроллер получает запросы от клиента, анализирует его параметры и для выполнения операций над данными запроса обращается к модели. От модели поступают уже скомпонованные объекты. Потом они отправляются в представление, которое передаёт сформированную страницу контроллеру, а он, в свою очередь, отправляет её клиенту [12].

Дополнительными плюсам данной архитектуры также являются:

- стандартизация кодирования;
- лёгкость обнаружения и исправления ошибок;
- быстрое вхождение в проект новых разработчиков;
- лёгкое изменение способа хранения сущностей.

Использование шаблона MVC значительно облегчает разработку приложения с клиент-серверной архитектурой, поскольку сразу создает хорошо обозначенное разграничение между основными субъектами данного подхода: клиентом и сервером. К тому же MVC позволяет качественно и быстро отладить и протестировать приложение.

Архитектуру MVC на языке Java поддерживает такой фреймворк как Spring MVC. Он представляет собой веб-службу, которая может взаимодействовать с различными приложениями. При этом приложение может быть, как веб-приложением на базе Spring, так и мобильным или обычным десктопным приложением.

Все запросы, которые обрабатываются Spring MVC приложением, проходят путь, показанный на рисунке 3.2:

- 1) клиентское приложение создаёт запрос к серверу;
- 2) HTTP-сервер загружает конфигурацию, создаёт экземпляр приложения;
- 3) с помощью объекта для управления запросами определяется контроллер;
- 4) для обработки запроса приложение создаёт экземпляр контроллера;
- 5) контроллер находит нужное действие и выполняет фильтры для действия;
- 6) действие не выполняется при неудачном выполнении любого фильтра;
- 7) действие выполняется при успешном выполнении всех фильтров;

- 8) действие загружает модель данных;
- 9) действие возвращает данные контроллеру;
- 10) контроллер сериализует данные и помещает их в тело ответа;
- 11) приложение формирует и отправляет ответ на запрос клиентского приложения.

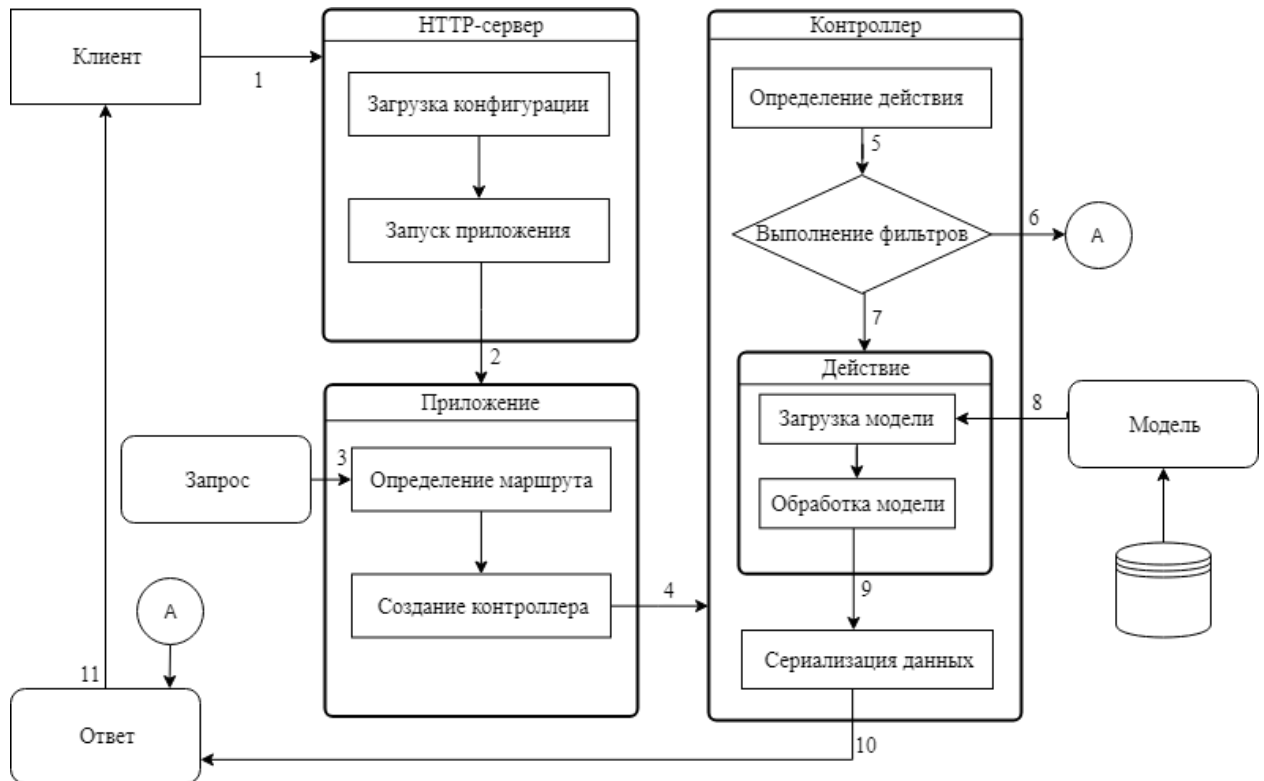


Рисунок 3.2 – Диаграмма обработки запроса серверной частью приложения

Таким образом для разработки серверной части данного приложения очень прекрасно подходит паттерн MVC совмещённой с клиент-серверной архитектурой. Объединение MVC паттерна с клиент-серверной архитектурой, позволяет значительно снизить нагрузку на сервер приложения, перенеся её часть на клиентские аппараты. Учитывая, что сейчас мобильные гаджеты и персональные компьютеры становятся всё мощнее перенос части нагрузки на них не вызывает ухудшения работы разрабатываемого приложения.

3.2 Разработка даталогической и физической моделей базы данных

На даталогическом уровне модель предметной области представляется в привязке к конкретной модели данных и описывает способ организации данных безотносительно их физического размещения в конкретной СУБД. В приложении, описываемом данным дипломным проектом, будет использована реляционная модель данных.

Реляционная модель данных представляет собой интуитивно понятный,

наглядный табличный способ представления данных, где каждая строка, содержащаяся в таблице, представляет собой запись с уникальным идентификатором, который именуется ключом. Столбцы таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута, что дает возможность легко устанавливать взаимосвязь между элементами данных [13].

В реляционной модели данных логическая структура (таблицы, представления и индексы) отличается от физической структуры хранения. Благодаря этому создаётся возможность управления физической системой хранения, не меняя данных, которые содержатся в логической структуре. Например, изменение имени файла базы данных не повлияет на хранящиеся в нем таблицы.

Модель базы данных даталогического уровня, представлена в таблицах 3.1-3.11.

Таблица 3.1 – Даталогическая структура сущности «User»

Атрибут	Значение атрибута	Тип атрибута
Id	Уникальный идентификатор	Числовой
Nickname	Уникальное имя пользователя	Текстовый
Email	Электронная почта	Текстовый
Password	Хэш значение пароля	Текстовый
RoleId	Идентификатор роли пользователя	Числовой

Таблица 3.2 – Даталогическая структура сущности «Role»

Атрибут	Значение атрибута	Тип атрибута
Id	Уникальный идентификатор	Числовой
Title	Наименование роли	Текстовый

Таблица 3.3 – Даталогическая структура сущности «User_Audiobook»

Атрибут	Значение атрибута	Тип атрибута
UserId	Идентификатор пользователя	Числовой
AudiobooksId	Идентификатор аудиокниги	Числовой

Таблица 3.4 – Даталогическая структура сущности «Audiobook_File»

Атрибут	Значение атрибута	Тип атрибута
Id	Уникальный идентификатор	Числовой
Extension	Расширение файла	Текстовый
File_Path	Путь к файлу содержащему аудиокнигу	Текстовый

Таблица 3.5 – Даталогическая структура сущности «Genre»

Атрибут	Значение атрибута	Тип атрибута
Id	Уникальный идентификатор	Числовой
Title	Наименование жанра	Текстовый

Таблица 3.6 – Даталогическая структура сущности «Author»

Атрибут	Значение атрибута	Тип атрибута
Id	Уникальный идентификатор	Числовой
Author_Name	Наименование автора	Текстовый

Таблица 3.7 – Даталогическая структура сущности «Audiobook»

Атрибут	Значение атрибута	Тип атрибута
Id	Уникальный идентификатор	Числовой
UserId	Идентификатор пользователя, добавившего аудиокнигу	Числовой
Distributed	Является ли книга распространяемой	Логический
Title	Название книги	Текстовый
Picture_Path	Путь к файлу содержащему изображение обложки	Текстовый
Publication_Year	Год издания	Числовой
Description	Краткое описание	Текстовый
Add_Date	Дата добавления	Дата/Время
Audiobook_FileId	Идентификатор файла аудиокниги	Числовой
Rating	Рейтинг аудиокниги	Числовой

Таблица 3.8 – Даталогическая структура сущности «Audiobook_Genre»

Атрибут	Значение атрибута	Тип атрибута
GenreId	Идентификатор жанра	Числовой
AudiobookId	Идентификатор аудиокниги	Числовой

Таблица 3.9 – Даталогическая структура сущности «Audiobook_Author»

Атрибут	Значение атрибута	Тип атрибута
AuthorId	Идентификатор автора	Числовой
AudiobookId	Идентификатор аудиокниги	Числовой

Таблица 3.10 – Даталогическая структура сущности «Rating»

Атрибут	Значение атрибута	Тип атрибута
UserId	Идентификатор пользователя	Числовой
AudiobookId	Идентификатор книги	Числовой
Value	Значение оценки	Числовой

Таблица 3.11 – Даталогическая структура сущности «Query»

Атрибут	Значение атрибута	Тип атрибута
UserId	Идентификатор пользователя	Числовой
AudiobookId	Идентификатор аудиокниги	Числовой
Send_DateTime	Дата и время отправления запроса	Дата/Время

3.3 Разработка алгоритма приложения и алгоритмов отдельных модулей

Хранить и оценивать аудиокниги может только зарегистрированный пользователь, поэтому функция регистрации очень важна. На рисунке 3.3 представлен алгоритм регистрации пользователя. Для того чтобы зарегистрировать пользователя необходимо заполнить обязательные поля формы и нажать на кнопку «Зарегистрироваться».

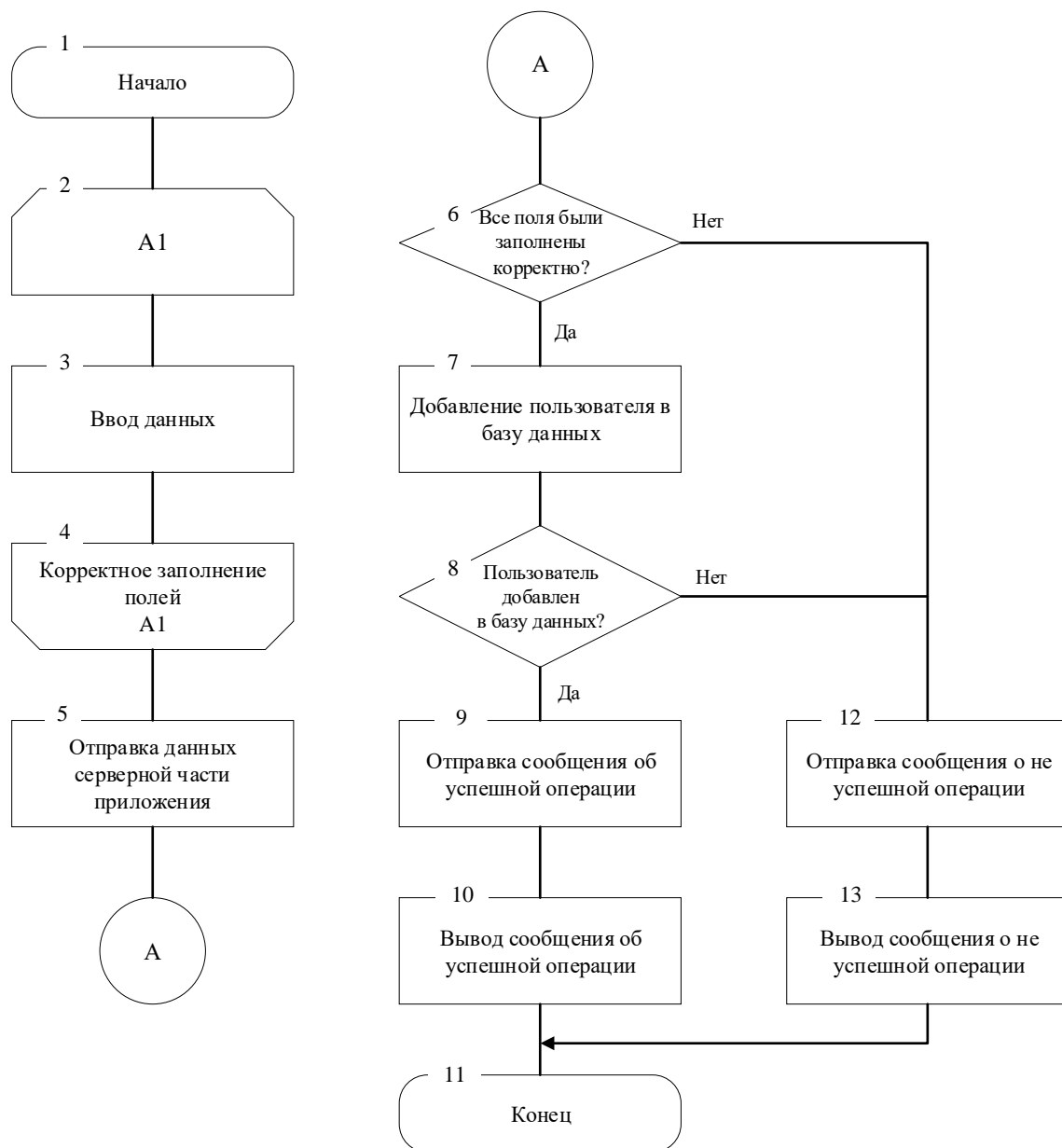


Рисунок 3.3 – Алгоритм регистрации пользователя

При успешной регистрации пользователь будет перенаправлен на страницу авторизации. Сразу после успешной регистрации пользователь может авторизоваться на сайте. Для авторизации пользователя требуется заполнить требуемые поля формы и нажать на кнопку «Войти» (рисунок 3.4).

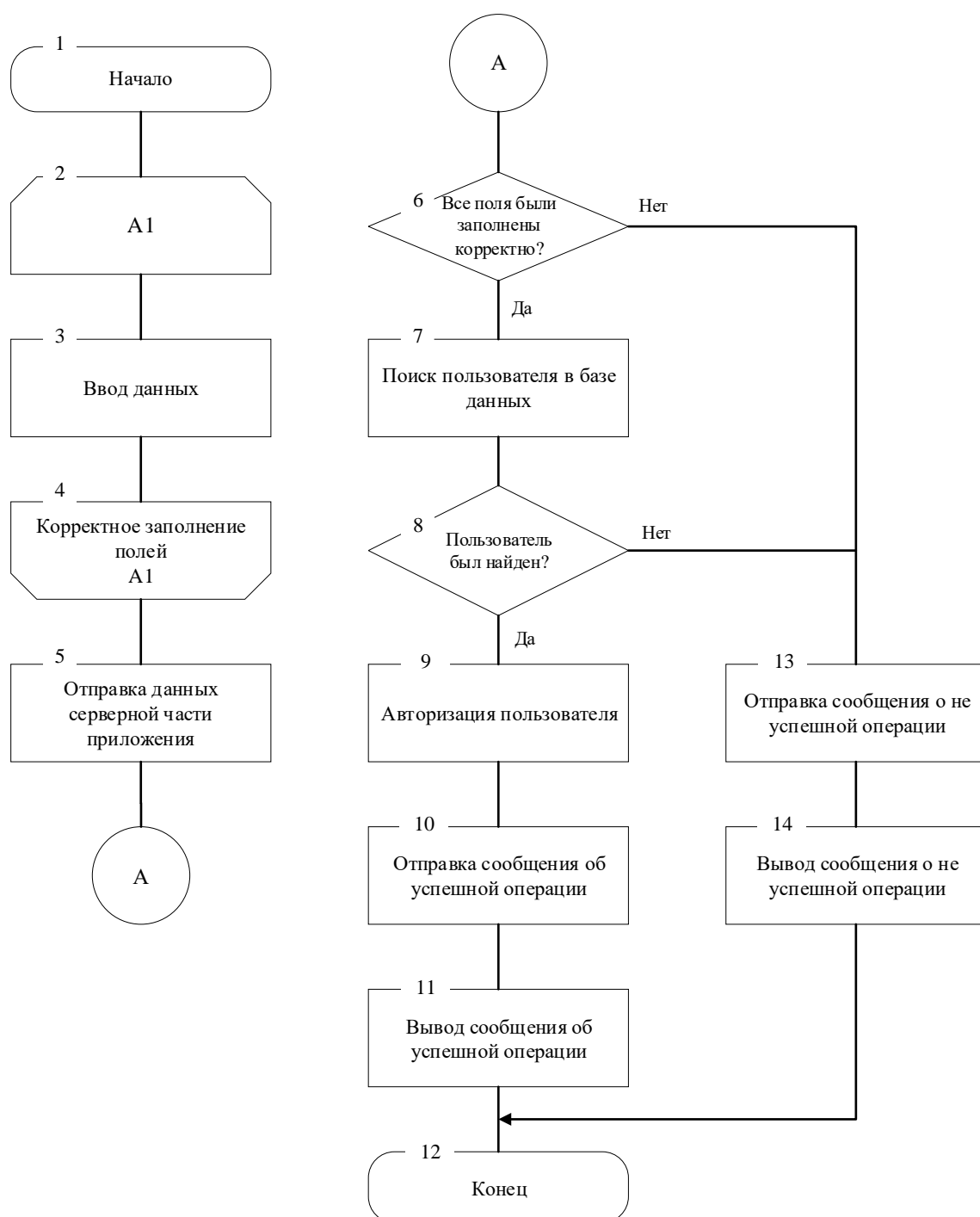


Рисунок 3.4 – Алгоритм авторизации пользователя

В случае успешной авторизации на сайте, пользователь сможет начать использование приложения согласно своей роли.

Пользователь может изменять данные своего профиля. Алгоритм обновления профиля пользователя приведён на рисунке 3.5. Чтобы обновить данные, пользователь должен изменить значения полей и нажать на кнопку «Сохранить».

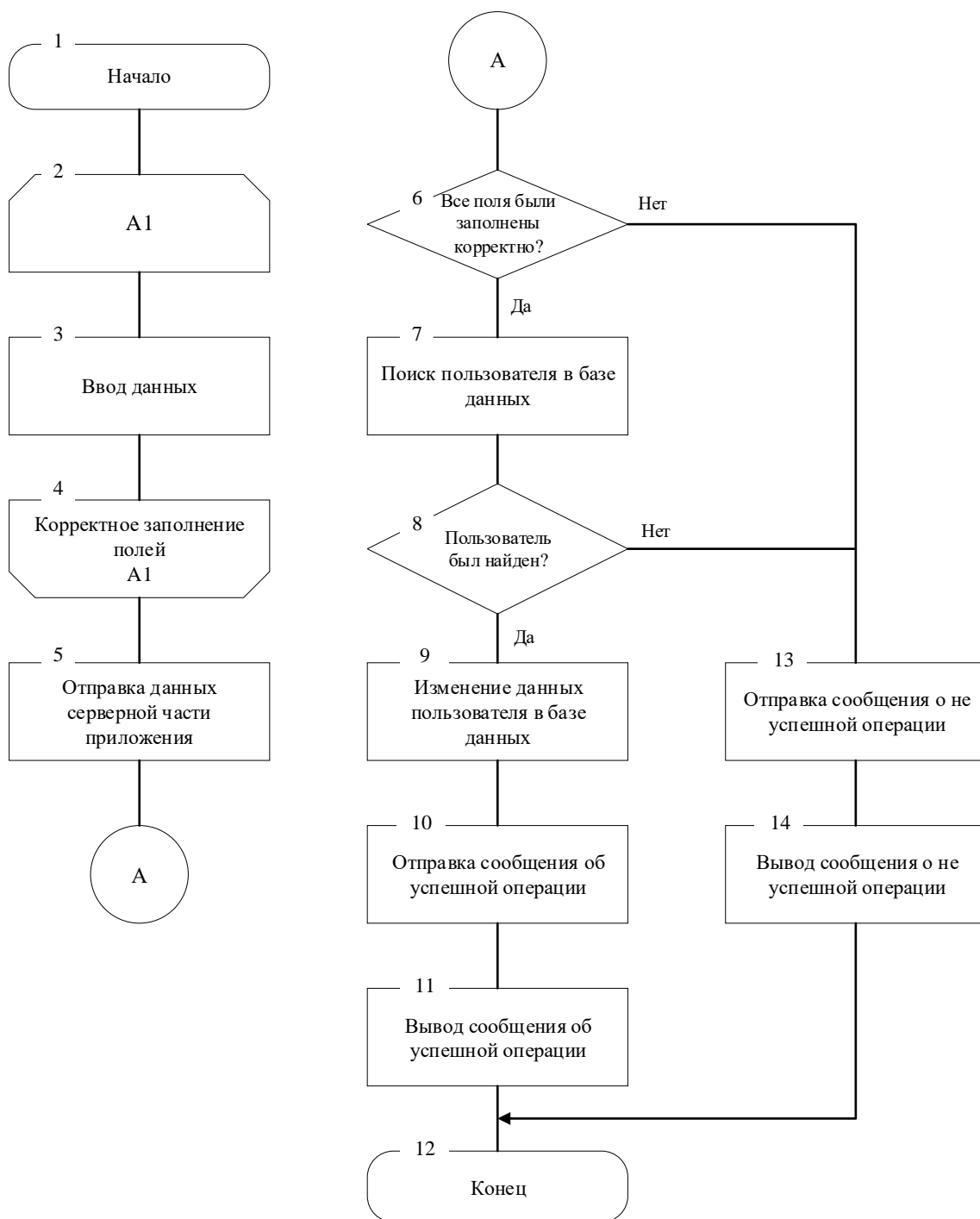


Рисунок 3.5 – Алгоритм обновления профиля пользователя

В случае удачного обновления личных данных появится сообщение об их успешном изменении. Если введённые данные будут не корректны появится сообщение об ошибке.

Пользователь может удалить хранимую аудиокнигу. Для этого в списке хранимых аудиокниг напротив нужного элемента следует нажать кнопку «Удалить». Алгоритм удаления аудиокниги приведён на рисунке 3.6.

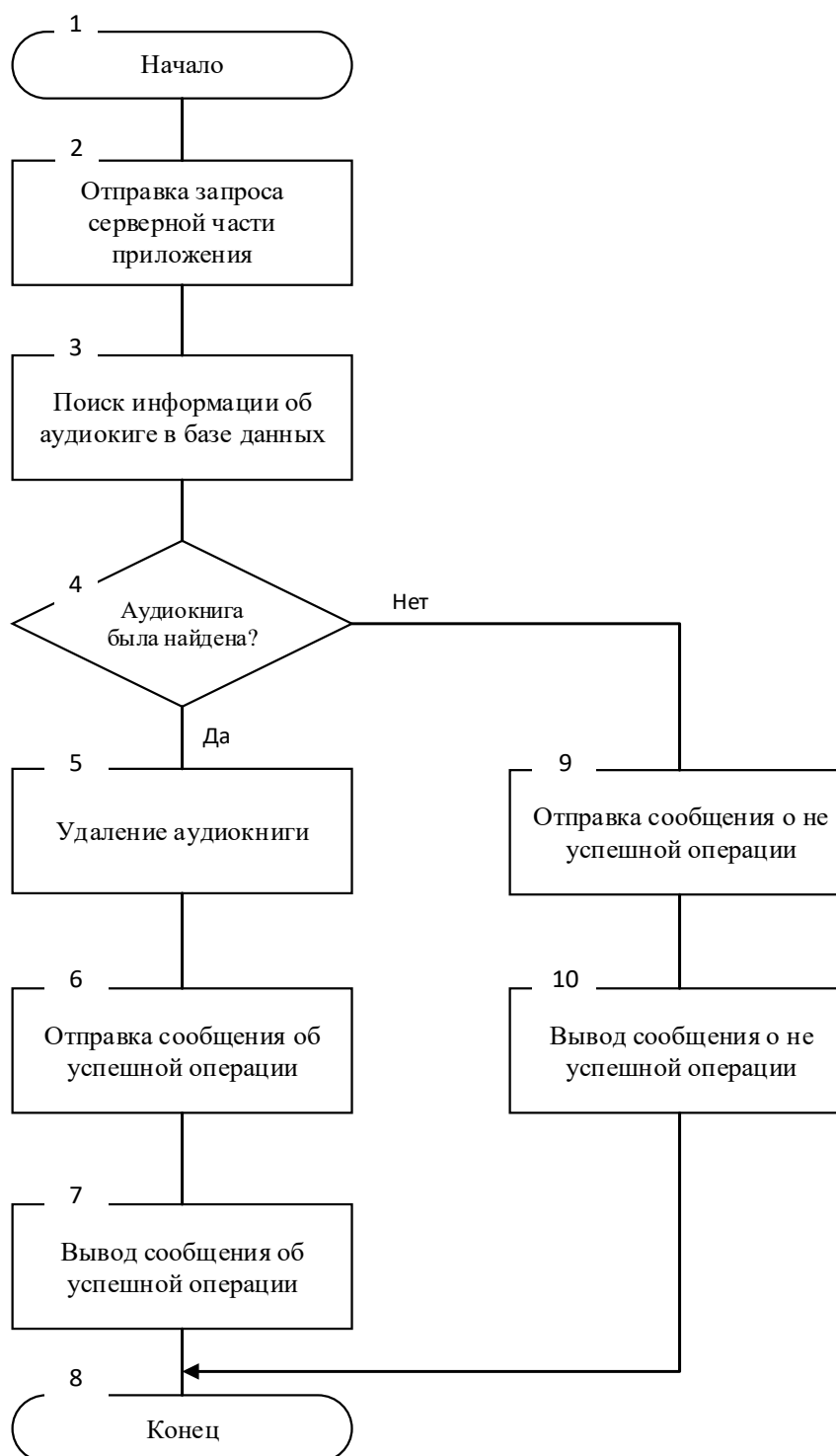


Рисунок 3.6 – Алгоритм удаления аудиокниги

Если удаление прошло успешно, то информация об аудиокниге удалится из базы данных, а файл содержащий аудиокнигу будет уничтожен, после чего

будет выведено сообщение об успешном удалении аудиокниги.

Итак, выше были приведены некоторые алгоритмы разрабатываемого приложения. Согласно этим алгоритмам будет реализована данная функциональность.

4 СОЗДАНИЕ ПРИЛОЖЕНИЯ

При проектировании программного средства особое внимание уделяется стандартным компонентам WebApi 2, существующим для разработки.

Контроллеры – часть MVC архитектуры содержащаяся в WebApi 2. Однако, учитывая, что WebApi 2 разрабатывался с целью создания API для приложений, контроллеры, которые применяются в нём являются несколько ограниченными.

Базовым классом для них служит ApiController. Все контроллеры, унаследованные от него, в качестве результата не могут возвращать представления, компоновки и т. д., а только модель. Модель будет сериализована в JSON, XML или другое представление данных в зависимости от заголовков HTTP запроса.

Так же методы данного вида контроллеров выбираются на основании HTTP метода пришедшего запроса. Это позволяет логически разделить методы на группы для упрощения понимания кода. Контроллеры в WebApi 2 принято создавать по подходу CRUD (Create-Read-Update-Delete). В соответствии с ним методы контроллера делятся на 4 группы:

- добавление новых данных в приложение (Create);
- извлечение данных из приложения (Read);
- изменение существующих данных (Update);
- удаление данных из приложения (Delete).

За каждую группу методов отвечает свой HTTP метод, используемый для запроса действия. За добавление данных отвечает HTTP метод Push, извлечение – Get, изменение – Put, удаление – Delete. На рисунке 4.1 приведён пример контроллера, отвечающего за работу с опросами. В контроллерах указываются атрибуты для указания ролей пользователей, имеющих к ним доступ. Ограничения можно наложить как на весь контроллер, так и на отдельный его метод.

Хранить логику обработки запросов в контроллере было бы громоздко и неудобно. С целью избегания этого приложение имеет 3 слоя:

- веб;
- бизнес логика;
- доступ к данным.

Каждый слой является отдельно компилируемой библиотекой, зависящей от других слоёв только интерфейсами и легко может быть заменён без перекомпиляции других слоёв при условии выполнения интерфейсов.

Контроллеры хранятся в веб слое и обращаются к сервисам, которые хранятся в слое бизнес логики. Сервисы содержат в себе всю логику изменения данных, пришедших в запросе. Для выполнения своей задачи сервисы могут обращаться к другим сервисам. На рисунке 4.2 приведён пример сервиса, отвечающего за работу с пользователями.


```

public class SurveysController : ApiController
{
    private readonly ISurveyService _surveyService;

    public SurveysController(ISurveyService surveyService)
    {
        _surveyService = surveyService;
    }

    public async Task<IServiceResponse> GetSurvey(int id, int version)
    {
        return await _surveyService.GetByIdAndVersionAsync(id, version);
    }

    public async Task<IServiceResponse> GetSurveys([FromUri] string userEmail = "")
    {
        if (userEmail.Length == 0) return await _surveyService.GetAllAsync();
        return await _surveyService.GetAllAsync(userEmail);
    }

    public async Task<IServiceResponse> AddSurvey([EditSurvey]string survey)
    {
        var surv = EditSurveyDto(survey);
        return await _surveyService.CreateAsync(surv);
    }

    public async Task<IServiceResponse> EditSurvey([EditSurvey]string survey)
    {
        var surv = EditSurveyDto(survey);
        return await _surveyService.UpdateAsync(surv);
    }

    public async Task<IServiceResponse> RemoveSurvey(int id)
    {
        return await _surveyService.DeleteByIdAsync(id);
    }
}

```

Рисунок 4.1 – Пример контроллера

```

public class UserService : IUserService
{
    public async Task<IServiceResponse<EditUserDto>> CreateAsync(EditUserDto userDto)
    {
        var us = _mapper.Map<EditUserDto, User>(userDto);
        us.Role = await _roleManager.FindByNameAsync(default(RoleTypes).ToString());
        var result = await _userManager.CreateAsync(us, us.Password);
        return !result.Succeeded ?
            ServiceResponse.CreateUnsuccessful<EditUserDto>(ServiceResponseCode.UserAlreadyExists) :
            ServiceResponse.CreateSuccessful(_mapper.Map<User, EditUserDto>(us));
    }

    public async Task<IServiceResponse> UpdateAsync(EditUserDto user)
    {
        var us = await _userManager.FindByIdAsync(user.Id);
        if (us == null)
        {
            return ServiceResponse.CreateUnsuccessful<object>(ServiceResponseCode.UserNotFoundById);
        }
        await _userManager.UpdateAsync(_mapper.Map(user, us));
        return ServiceResponse.CreateSuccessful();
    }

    public async Task<IServiceResponse> DeleteByIdAsync(int id)
    {
        var us = await _uow.Users.GetByIdAsync(id, u => u.Role);
        if (us == null)
        {
            return ServiceResponse.CreateUnsuccessful<object>(ServiceResponseCode.UserNotFoundById);
        }
        _uow.Users.Delete(us);
        return ServiceResponse.CreateSuccessful();
    }

    public async Task<IServiceResponse<ReadUserDto>> GetByIdAsync(int id)
    {
        var user = await _uow.Users.GetByIdAsync(id, u => u.Role);
        return user == null ?
            ServiceResponse.CreateUnsuccessful<ReadUserDto>(ServiceResponseCode.UserNotFoundById) :
            ServiceResponse.CreateSuccessful(_mapper.Map<User, ReadUserDto>(user));
    }
}

```

Рисунок 4.2 – Пример сервиса

Для повышения независимости слоёв друг от друга используется принцип инверсии зависимостей. Согласно нему компоненты не знают друг о друге ничего кроме интерфейсов. Это позволяет в любой момент подменять компоненты другими с таким же интерфейсом. Он является частью большего принципа написания кода, распространяющегося не только на веб-приложения – SOLID. Этот принцип является очень важным при написании качественного и поддерживаемого кода. Каждая буква в названии отвечает за отдельный принцип, несущий свои правила написания кода:

- принцип одной ответственности (single responsibility);
- принцип открытости/закрытости (open-closed);
- принцип подстановки Барбары Лисков (Liskov substitution);
- принцип разделения интерфейса (interface segregation);
- принцип инверсии зависимостей (dependency inversion).

На рисунке 4.3 приведён пример класса, отвечающего за настройку зависимостей для приложения.

```
public class ServiceModule : Module
{
    protected override void Load(ContainerBuilder builder)
    {
        builder.RegisterType<UserService>()
            .As<IUserService>()
            .InstancePerRequest();

        builder.RegisterType<ResetPasswordService>()
            .As<IResetPasswordService>()
            .InstancePerRequest();

        builder.RegisterType<ApplicationUserManager>()
            .AsSelf()
            .InstancePerRequest();

        builder.RegisterType<ApplicationRoleManager>()
            .AsSelf()
            .InstancePerRequest();
    }
}
```

Рисунок 4.3 – Пример настройки зависимостей

После применения всех преобразований к данным сервисы обращаются к репозиториям для внесения изменений в базу данных. В приложении реализован репозиторий для выполнения базовых операций с любой моделью. При необходимости выполнения нестандартных действий создаётся дополнительный репозиторий, наследуемый от базового, и в нём реализуются все необходимые операции. На рисунке 4.4 приведён пример базового репозитория.

```

public class Repository<TEntity> : IRepository<TEntity>
where TEntity : class, IEntity
{
    public virtual TEntity Create(TEntity entity)
    {
        return _dbSet.Add(entity);
    }

    public virtual async Task<TEntity> GetByIdAsync(int id, params Expression<Func<TEntity, object>>[] includes)
    {
        return await includes.Aggregate(_dbSet.Where(e => e.Id == id), (cur, include) => cur.Include(include))
            .SingleOrDefaultAsync();
    }

    public virtual async Task<IReadOnlyCollection<TEntity>> GetAllAsync(params Expression<Func<TEntity, object>>[] includes)
    {
        return await includes.Aggregate<
            Expression<Func<TEntity, object>>,
            IQueryable<TEntity>
        >(_dbSet, (cur, include) => cur.Include(include)).ToListAsync();
    }

    public virtual void Update(TEntity entity)
    {
        if (!_dbSet.Local.Contains(entity)) Context.Entry(entity).State = EntityState.Modified;
    }

    public virtual void Delete(TEntity entity)
    {
        if (!_dbSet.Local.Contains(entity)) _dbSet.Attach(entity);
        _dbSet.Remove(entity);
    }

    public void DeleteRange(IReadOnlyCollection<TEntity> entities)
    {
        foreach (var entity in entities)
        {
            if (!_dbSet.Local.Contains(entity)) _dbSet.Attach(entity);
        }
        _dbSet.RemoveRange(entities);
    }
}

```

Рисунок 4.4 – Пример базового репозитория

Репозитории обращаются к базе данных за данными, которые в приложении хранятся в моделях – второй части архитектуры MVC. Модели содержат только данные. На рисунке 4.5 приведён пример модели.

```

public class User: IUser<int>, IEntity
{
    public int Id { get; set; }

    public string UserName { get; set; }

    public string Email { get; set; }

    public string Password { get; set; }

    public int RoleId { get; set; }

    public Role Role { get; set; }

    public ICollection<Survey> Surveys { get; set; }

    public ICollection<SurveyTemplate> SurveyTemplates { get; set; }

    public ICollection<SurveyResponse> SurveyResponses { get; set; }
}

```

Рисунок 4.5 – Пример модели

Так как зачастую для выполнения запроса нет необходимости возвращать все данные из модели, в приложении используются промежуточные модели, называемые DTO. Они содержат только необходимые для выполнения запроса данные. Для каждого запроса может создаваться отдельная промежуточная модель или использоваться одна для нескольких, если необходимые данные совпадают.

Так как именно промежуточные модели используются в контроллерах, то именно в них указываются валидационные атрибуты для проверки пришедшей модели на серверной части приложения. На рисунке 4.6 приведён пример промежуточной модели.

```
public class UserDto
{
    [Required]
    [MinLength(3, ErrorMessage = "Name should be at least 3 characters")]
    public string UserName { get; set; }

    [Required]
    [EmailAddress(ErrorMessage = "Invalid email")]
    public string Email { get; set; }
}
```

Рисунок 4.6 – Пример промежуточной модели

Для преобразования моделей в промежуточные модели и обратно в приложении используется библиотека для маппинга. На рисунке 4.7 приведён пример класса для настройки преобразования моделей.

```
public class MapperInitializerModule : Module
{
    protected override void Load(ContainerBuilder builder)
    {
        builder.Register(c => new MapperConfiguration(cfg =>
        {
            cfg.AddProfiles(GetType().Assembly);
            cfg.CreateMap<User, ReadUserDto>().ReverseMap();
            cfg.CreateMap<User, EditUserDto>().ReverseMap();
            cfg.CreateMap<User, UserDto>().ReverseMap();
            cfg.CreateMap<Role, RoleDto>().ReverseMap();
            cfg.CreateMap<Survey, SurveyDto>().ReverseMap();
            cfg.CreateMap<TemplatePage, PageDto>().ReverseMap();
            cfg.CreateMap<SurveyResponse, SurveyResponseDto>().ReverseMap();
            cfg.CreateMap<SurveyTemplate, TemplateDto>().ReverseMap();
            cfg.CreateMap<QuestionVariant, QuestionVariantDto>().ReverseMap();
            cfg.CreateMap<SurveySettings, SurveySettingsDto>().ReverseMap();
            cfg.CreateMap<QuestionType, QuestionTypeDto>().ReverseMap();
            cfg.CreateMap<LocalizableString, LocalizableStringDto>().ReverseMap();
        })).AsSelf().SingleInstance();

        builder
            .Register(c => c.Resolve<MapperConfiguration>().CreateMapper(c.Resolve))
            .As<IMapper>().SingleInstance();
    }
}
```

Рисунок 4.7 – Пример настройки преобразований моделей

Для хранения данных в приложении используется база данных. Для удобной работы с ней используется Entity Framework. Он позволяет работать с базой данных как будто сущности в ней – это объекты, к которым можно обратиться напрямую из кода приложения.

При построении базы данных был использован подход code first. Согласно данному подходу, база данных создаётся по моделям, описанным в приложении. Для корректного создания написаны конфигурационные классы для каждой модели. На рисунке 4.8 приведён пример конфигурации базы данных для модели пользователя.

```
internal class UserConfiguration : EntityTypeConfiguration<User>
{
    public UserConfiguration()
    {
        Property(u => u.UserName).IsRequired();
        Property(u => u.Email).IsRequired();
        Property(u => u.Password).IsRequired();
        HasRequired(u => u.Role)
            .WithMany(u => u.Users)
            .HasForeignKey(u => u.RoleId);

        Property(u => u.Email)
            .HasColumnType("VARCHAR")
            .HasMaxLength(50);

        Property(u => u.Email)
            .HasColumnAnnotation(
                IndexAnnotation.AnnotationName,
                new IndexAnnotation(new IndexAttribute("IX_Email", 1) { IsUnique = true }));
    }
}
```

Рисунок 4.8 – Пример конфигурации базы данных для модели

При изменении модели нет необходимости создавать базу данных заново, теряя все накопленные данные. Entity Framework позволяет создавать миграции для корректного изменения базы данных. На рисунке 4.9 приведён пример миграционного файла. С помощью таких файлов при необходимости можно вернуться к любой версии базы данных.

```
public partial class UsageStat : DbMigration
{
    public override void Up()
    {
        AddColumn("dbo.QuestionVariant", "UsageStat", c => c.Double(nullable: false));
        AlterColumn("dbo.QuestionAnswer", "Value", c => c.String());
        AlterColumn("dbo.Roles", "Name", c => c.String(nullable: false));
    }

    public override void Down()
    {
        AlterColumn("dbo.Roles", "Name", c => c.String());
        AlterColumn("dbo.QuestionAnswer", "Value", c => c.String(nullable: false));
        DropColumn("dbo.QuestionVariant", "UsageStat");
    }
}
```

Рисунок 4.9 – Пример миграционного файла

Последняя часть архитектуры MVC – представление – реализована клиентской частью приложения. В ней используются такие библиотеки как React и Redux.

React – это декларативная, эффективная и гибкая библиотека JavaScript для создания пользовательских интерфейсов (UI). Она позволяет создавать сложные UI из небольших и изолированных частей кода, называемых «компонентами». Компонент – часть интерфейса с определённым функционалом. Они могут использовать внутри себя другие компоненты.

В React предусмотрено три вида компонент:

- компонент-класс;
- компонент-функция;
- компонент-строка.

Компонент-строка – это самый простой вид компонент. В виде обычной строки передаётся HTML или JSX код компонента. Этот вид компонент используется крайне редко из-за малой функциональности.

Компонент-класс – это самый часто используемый вид компонент. Такие компоненты могут регулировать свой жизненный цикл, иметь состояние и содержать логику обработки поступивших данных. Обязательное условие таких компонент – наличие метода `render`, который возвращает `jsx` объект или строку, содержащую `jsx` код. В таблице 4.1 описаны методы жизненного цикла компонент.

Таблица 4.1 Методы жизненного цикла компонент

Имя метода	Описание
<code>constructor</code>	Конструктор для начальной инициализации компонента.
<code>componentWillMount</code>	Вызывается перед рендерингом компонента.
<code>render</code>	Рендеринг компонента.
<code>componentDidMount</code>	Вызывается после рендеринга компонента.
<code>shouldComponentUpdate</code>	Вызывается каждый раз при обновлении объекта <code>props</code> или <code>state</code> .
<code>componentWillUpdate</code>	Вызывается перед обновлением компонента.
<code>componentDidUpdate</code>	Вызывается сразу после обновления компонента.
<code>componentWillUnmount</code>	Вызывается перед удалением компонента.

Компонент-функция – нечто среднее между строкой и классом. Он может содержать в себе минимальную логику, но у него нету состояния и доступа к методам жизненного цикла. На рисунке 4.10 приведён пример компонента-функции, отвечающего за отображение навигационных ссылок в приложении.

```

import React from 'react';
import { Link } from 'react-router-dom';
import { Button } from 'react-bootstrap';

import Routes from '../routes';

import './Header.scss';

const HeaderLink = () => (
  <Button className="navbar__item-wrapper">
    <Link className="navbar__item react-bootstrap-link"
      to={Routes.Login.path}>{Routes.Login.text}</Link>
  </Button>
);

export default HeaderLink;

```

Рисунок 4.10 – Пример компонента-функции

Хоть компоненты-классы могут хранить состояние, состояние всего приложения хранить в компоненте самого верхнего уровня и передавать его во вложенные компоненты неудобно. В связи с этим была придумана библиотека Redux. Эта библиотека отвечает за хранение состояния всего приложения и его изменение.

Принцип, на котором построена данная библиотека, весьма прост. Есть хранилище состояния – store. Оно является единственным источником правды для всего приложения, т. е. все необходимые ему данные хранятся именно в нём. Есть действия, которые вызывает приложение, когда хочет изменить своё состояние. И есть редьюсеры, которые обрабатывают действия, вызванные приложением.

Поток данных имеет следующий вид:

- 1) приложение вызывает действие для смены состояния;
- 2) действие попадает в Redux, который вызывает редьюсер;
- 3) корневой редьюсер может комбинировать результаты нескольких редьюсеров в один целостный результат;
- 4) Redux сохраняет конечное состояние приложения, возвращённое корневым редьюсером.

Редьюсеры это чистые функции, реагирующие на действия и меняющие хранилище определённым образом. Они не должны мутировать предыдущее состояние, а создавать новое на основе старого с изменениями, пришедшими в действии и возвращать его в хранилище.

5 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Тестирование – это обязательный этап разработки приложения. На данном этапе определяется, соответствует ли приложение заявленным требованиям. Также проверяется функциональность спроектированного приложения. Если тест оказался отрицательным, то нужно исправить выявленную ошибку и снова повторить тест.

В таблице 5.1 будет приведено некоторое число разработанных тестов, подтверждающее работоспособность спроектированного приложения.

Таблица 5.1 – Результаты тестирования

Название	Описание	Ожидаемый результат	Полученный результат
Просмотр страницы авторизации.	Нажать на кнопку с подписью «Log In».	Отображение страницы авторизации.	Отобразилась страница авторизации.
Авторизация пользователя.	1. Заполнить поля для ввода данных: email – строка вида *@*.*, пароль – минимум восемь символов. 2. Нажать на кнопку «Войти».	1. Валидация введенных данных. 2. Поиск пользователя в базе данных. Если пользователь найден, то он авторизуется. Вывод сообщения об ошибке в случае неудачи.	1. Данные валидируются. 2. Поиск пользователя в базе данных. Авторизация пользователя. Вывод сообщения об ошибке в случае неудачи.
Просмотр страницы регистрации.	На странице авторизации нажать на кнопку «Signup».	Отображение страницы регистрации.	Отобразилась страница регистрации.
Регистрация пользователя.	1. Заполнить поля для ввода данных: имя – минимум пять символов, email – строка вида *@*.*, пароль – минимум восемь символов, повтор пароля – идентичный паролю. 2. Нажать на кнопку «Sign Up».	1. Валидация введенных данных. 2. Пользователь регистрируется. Вывод сообщения об ошибке в случае неудачи.	1. Данные валидируются. 2. Пользователь зарегистрировался. Выводится сообщение об ошибке в случае неудачи.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Выход из учётной записи.	1. Нажать на кнопку с именем пользователя. 2. Нажать на кнопку с надписью «Logout».	1. Появление панели с выбором действий. 2. Выход из учётной записи.	1. Появилась панель с выбором действий. 2. Пользователь вышел из учётной записи.
Просмотр страницы создания опроса.	1. Авторизоваться как пользователь с ролью Admin или VIP. 2. В левом меню нажать кнопку «New survey»	1. Отображение левой панели с функциями администратора. 2. Отображение страницы создания опроса.	1. Отобразилась левая панель с функциями администратора. 2. Отобразилась страница создания опроса.
Изменение основного языка опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «General» из выпадающего списка «Default language» выбрать один из доступных языков. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение основного языка опроса.	1. Появилось окно настроек опроса. 3. Основной язык опроса изменился.
Изменение видимости номеров страниц.	1. Нажать кнопку «Survey settings». 2. В разделе «General» установить/снять галочку напротив «Show page numbers». 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Возле заголовков страниц присутствуют/отсутствуют номера страниц.	1. Появилось окно настроек опроса. 3. Возле заголовков страниц появились/пропали номера страниц.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Изменение заголовка опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «General» в поле «Title» ввести заголовок. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Валидация введенных данных. Изменение заголовка опроса.	1. Появилось окно настроек опроса. 3. Данные валидируются. Заголовок опроса изменяется.
Изменение текста кнопки «Далее» для основного языка.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» в поле «Page next button text» ввести текст. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение текста кнопки «Далее» при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменился текст кнопки «Далее» при тестировании прохождения опроса.
Изменение текста кнопки «Назад» для основного языка.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» в поле «Page previous button text» ввести текст. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение текста кнопки «Назад» при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменился текст кнопки «Назад» при тестировании прохождения опроса.
Изменение текста кнопки «Старт» для основного языка.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» в поле «Start button text» ввести текст. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение текста кнопки «Старт» при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменился текст кнопки «Старт» при тестировании прохождения опроса.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Изменение текста кнопки «Отправить» для основного языка.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» в поле «Complete button text» ввести текст. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение текста кнопки «Отправить» при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменился текст кнопки «Отправить» при тестировании прохождения опроса.
Изменение позиции кнопок навигации.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» в выпадающем списке «Show navigation buttons» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение позиции кнопок навигации при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменилась позиция кнопок навигации при тестировании прохождения опроса.
Изменение видимости кнопки «Назад».	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» установить/снять галочку напротив «Show previous button». 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Присутствует/отсутствует кнопка «Назад».	1. Появилось окно настроек опроса. 3. Появилась/пропала кнопка «Назад».

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Включение/отключение стартовой страницы.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» установить/снять галочку напротив «The first page in the survey is a started page». 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Первая страница опроса является стартовой.	1. Появилось окно настроек опроса. 3. Первая страница опроса стала стартовой.
Включение/отключение показа страницы завершения опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» установить/снять галочку напротив «Show the completed page at the end». 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. После прохождения опроса показывается/пропускается страница завершения опроса.	1. Появилось окно настроек опроса. 3. После прохождения опроса вывелась/пропустилась страница завершения опроса.
Включение/отключение автоперехода на следующую страницу при заполнении текущей.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» установить/снять галочку напротив «On answering all questions, go to the next page automatically». 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. После заполнения всех вопросов текущей страницы совершается/ не совершается автоматический переход на следующую.	1. Появилось окно настроек опроса. 3. После заполнения всех вопросов текущей страницы совершился/ не совершился автоматический переход на следующую.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Изменение позиции шкалы прогресса.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» в выпадающем списке «Show progress bar» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение позиции шкалы прогресса при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменилась позиция шкалы прогресса при тестировании прохождения опроса.
Включение одностраничного режима у опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Navigation» установить/снять галочку напротив «Show all elements on one page». 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. При тестировании прохождения опроса все вопросы показываются на одной странице.	1. Появилось окно настроек опроса. 3. При тестировании прохождения опроса все вопросы показали на одной странице.
Изменение позиции заголовка вопроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Question» в выпадающем списке «Question title location» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение позиции заголовка вопроса при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменилась позиция заголовка вопроса при тестировании прохождения опроса.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Изменение обозначения обязательности вопроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Question» в поле «The question required symbol» ввести текст. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение обозначения обязательности вопроса.	1. Появилось окно настроек опроса. 3. Изменилось обозначение обязательности вопроса.
Изменение отображения номера вопросов.	1. Нажать кнопку «Survey settings». 2. В разделе «Question» в выпадающем списке «Show question numbers» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Номера вопросов отображаются в соответствии с выбранной настройкой.	1. Появилось окно настроек опроса. 3. Отображение номеров вопросов изменилось в соответствии с выбранной настройкой.
Изменение позиции сообщения с ошибкой.	1. Нажать кнопку «Survey settings». 2. В разделе «Question» в выпадающем списке «Question error location» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение позиции сообщения с ошибкой при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменилась позиция сообщения с ошибкой при тестировании прохождения опроса.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Изменение порядка размещения вопросов на странице.	1. Нажать кнопку «Survey settings». 2. В разделе «Question» в выпадающем списке «Elements order on the page» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение порядка размещения вопросов на странице при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменился порядок размещения вопросов на странице при тестировании прохождения опроса.
Изменение максимального времени на заполнение страницы опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Timer/Quiz» в поле «Maximum time to finish a page in the survey» ввести время в секундах. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение максимального времени на заполнение страницы при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменилось максимальное время на заполнение страницы при тестировании прохождения опроса.
Изменение максимального времени для ответа на опрос.	1. Нажать кнопку «Survey settings». 2. В разделе «Timer/Quiz» в поле «Maximum time to finish the survey» ввести время в секундах. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение максимального времени для ответа на опрос при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменилось максимальное время для ответа на опрос при тестировании прохождения опроса.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Изменение HTML кода, показываемого по окончании опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Completed Html» задать HTML код. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение HTML кода, показываемого по окончании опроса при тестировании прохождения опроса.	1. Появилось окно настроек опроса. 3. Изменился HTML код, показываемый по окончании опроса при тестировании прохождения опроса.
Изменение позиции таймера.	1. Нажать кнопку «Survey settings». 2. В разделе «Timer/Quiz» в выпадающем списке «Show timer panel» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение позиции таймера в соответствии с выбранным вариантом.	1. Появилось окно настроек опроса. 3. Изменилась позиция таймера в соответствии с выбранным вариантом.
Изменение отображаемого таймера.	1. Нажать кнопку «Survey settings». 2. В разделе «Timer/Quiz» в выпадающем списке «Show timer panel mode» выбрать один из предложенных вариантов. 3. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Изменение отображаемого таймера при прохождении опроса.	1. Появилось окно настроек опроса. 3. Изменился отображаемый таймер при прохождении опроса.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Создание триггеров опроса.	1. Нажать кнопку «Survey settings». 2. В разделе «Triggers» выбрать необходимый вид триггера. 3. Заполнить все необходимые поля. 4. Нажать кнопку «Apply» или «OK».	1. Появление окна настроек опроса. 3. Триггер работает в соответствии со своим типом.	1. Появилось окно настроек опроса. 3. Триггер сработал в соответствии со своим типом.
Добавление вопроса.	Нажать на нужный вопрос или потянуть его в нужное место в опросе.	Вопрос располагается в нужном месте в опросе.	Вопрос расположился в нужном месте в опросе.
Сохранение опроса.	1. Добавить минимум 1 вопрос, заполнить заголовок опроса и страницы. 2. Нажать на кнопку «Save survey».	2. Валидация опроса. Сохранение опроса на сервере. Вывод сообщения об успешном сохранении/ошибке.	2. Опрос валидируется. Опрос сохраняется на сервере. Выводится сообщение об успешном сохранении/ошибке.
Сохранение опроса как шаблон.	1. Добавить минимум 1 вопрос, заполнить заголовок опроса и страницы. 2. Нажать на кнопку «Save as template».	2. Валидация шаблона. Сохранение шаблона на сервере. Вывод сообщения об успешном сохранении/ошибке.	2. Шаблон валидируется. Шаблон сохраняется на сервере. Выводится сообщение об успешном сохранении/ошибке.

Продолжение таблицы 5.1

Название	Описание	Ожидаемый результат	Полученный результат
Прохождение опроса.	<ol style="list-style-type: none"> 1. В левом меню нажать на кнопку «Surveys list». 2. В списке опросов нажать на заголовок нужного опроса. 3. Заполнить ответы на все необходимые вопросы опроса. 4. Нажать на кнопку завершения опроса. 	<ol style="list-style-type: none"> 1. Появление окна с опросами. 2. Появление окна прохождения опроса. 3. Валидация введенных данных. 4. Сохранение ответа на опрос на сервере. Изменение статистики ответов на опрос. 	<ol style="list-style-type: none"> 1. Появилось окно с опросами. 2. Появилось окно прохождения опроса. 3. Введенные данные валидируются. 4. Ответ на опрос сохранен на сервере. Ответы учтены в статистике ответов на опрос.
Редактирование профиля	<ol style="list-style-type: none"> 1. В правом верхнем углу нажать на кнопку с именем пользователя. 2. В появившемся окне выбрать «My profile». 3. Нажать на кнопку «Edit». 4. Ввести имя пользователя – минимум пять символов. 5. Нажать на кнопку «Save». 	<ol style="list-style-type: none"> 1. Появление меню с функциями по работе с аккаунтом пользователя. 2. Появление окна с личными данными пользователя. 3. Переход окна в режим редактирования. 4. Имя валидируется. 5. Изменение данных пользователя. 	<ol style="list-style-type: none"> 1. Появилось меню с функциями по работе с аккаунтом пользователя. 2. Появилось окно с личными данными пользователя. 3. Окно перешло в режим редактирования. 4. Имя валидируется. 5. Данные пользователя изменились.

6 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

Для того, чтобы использовать приложение, необходимо открыть любой браузер с поддержкой JavaScript и в строку поиска ввести адрес <http://techartsurvey.000webhostapp.com> (рисунок 6.1).

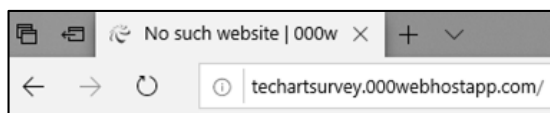


Рисунок 6.1 – Адрес приложения

После перехода по ссылке откроется главная страница (рисунок 6.2).



Рисунок 6.2 – Главная страница приложения

Перед тем, как использовать возможности приложения, требуется зарегистрироваться на сайте (рисунок 6.3).

A screenshot of a 'Sign Up' form. The form is titled 'Sign Up' at the top. It contains four input fields: 'Name', 'Email', 'Password', and 'Confirm password'. Each field has a small label to its left. At the bottom right of the form is a 'Sign Up' button.

Рисунок 6.3 – Форма регистрации

После успешной регистрации пользователь сможет авторизоваться в приложении (рисунок 6.4).

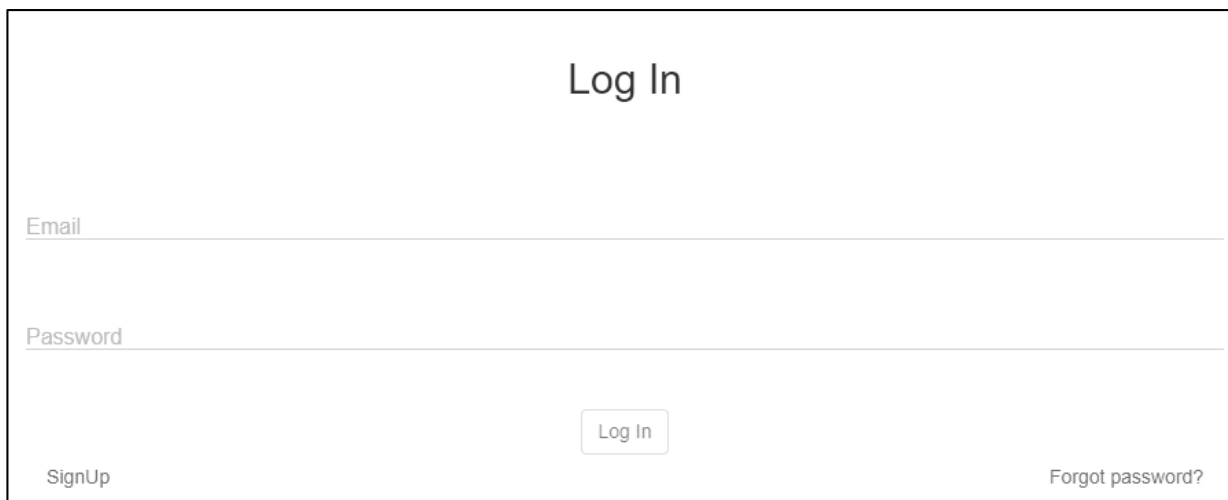
The image shows a login form titled "Log In". It features two input fields: "Email" and "Password". Below the "Password" field is a "Log In" button. At the bottom left, there is a "SignUp" link, and at the bottom right, there is a "Forgot password?" link.

Рисунок 6.4 – Форма авторизации

После авторизации пользователь перенаправляется на главную страницу. После авторизации на ней появится панель с инструментами с доступным пользователю функционалом (рисунок 6.5).

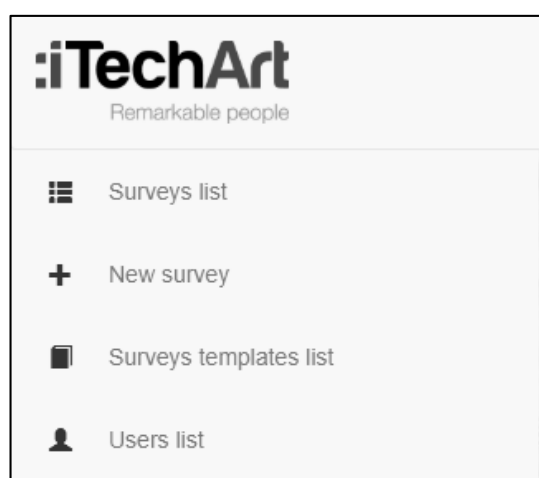


Рисунок 6.5 – Панель с инструментами для роли администратор

Чтобы выбрать опрос для прохождения, нужно нажать на кнопку «Survey list» на панели инструментов. После этого пользователь попадёт на страницу со списком опросов (рисунок 6.6).

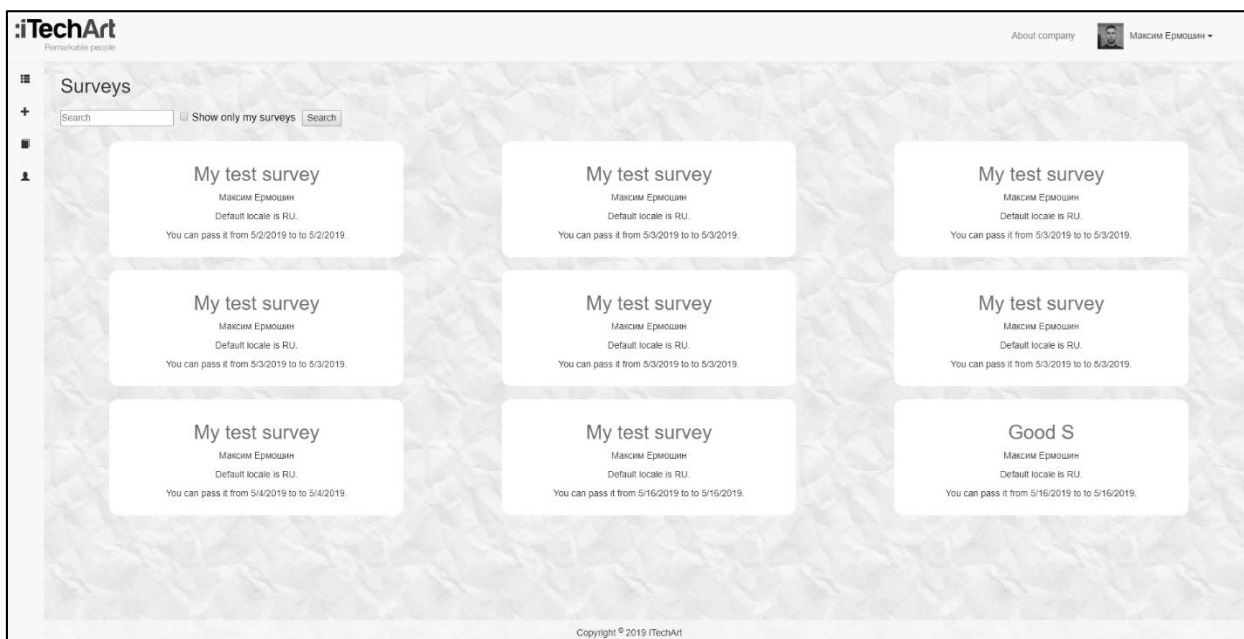


Рисунок 6.6 – Страница со списком опросов

Для прохождения опроса необходимо нажать на заголовок опроса. После этого пользователь попадёт на страницу опроса для прохождения (рисунок 6.7)

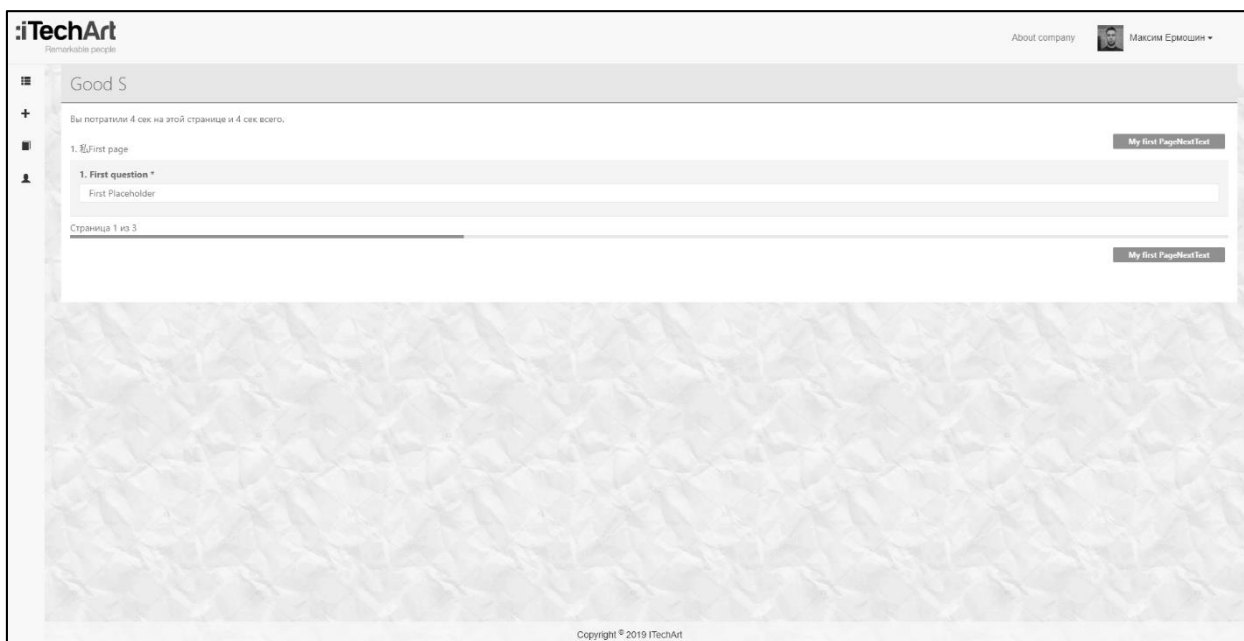


Рисунок 6.7 – Страница опроса

Для создания опроса необходимо обладать ролью администратора и нажать кнопку «New survey» на панели инструментов. После этого пользователь попадёт в редактор опросов на вкладку редактирования опроса «Survey Designer» (рисунок 6.8).

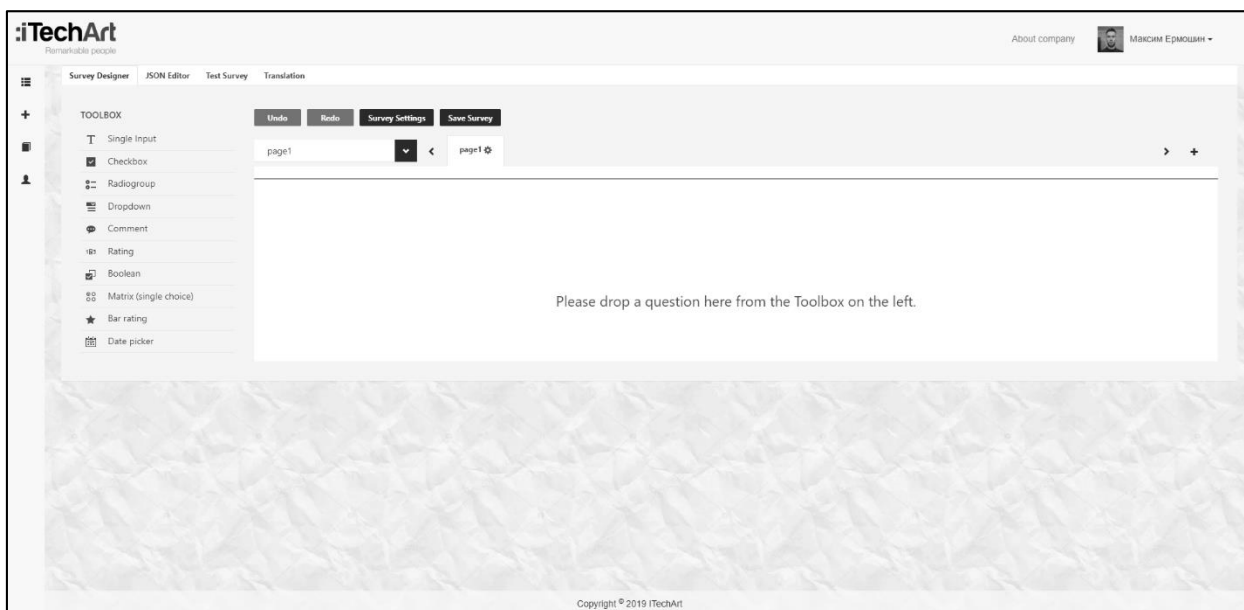


Рисунок 6.8 – Редактор опросов

Для редактирования настроек опроса нужно нажать кнопку «Survey Settings», заполнить необходимые поля и нажать кнопку «Apply» или «OK» (рисунок 6.9).

Рисунок 6.9 – Окно настроек опроса

Для добавления страницы необходимо нажать кнопку «+» на панели страниц. Для редактирования настроек страницы нужно сделать активной нужную страницу, навести на неё курсор мыши и нажать кнопку «Edit». Появится окно редактирования настроек страницы. Далее необходимо заполнить требуемые данные и нажать кнопку «Apply» или «OK» (рисунок 6.10).

Рисунок 6.10 – Окно настроек страницы

Для добавления вопроса в опрос необходимо нажать на него на панели с вопросами, тогда он поместится в конец текущей страницы, либо перетянуть его в нужное место на текущей странице (рисунок 6.11). В результате вопрос добавится к опросу.

Рисунок 6.11 – Добавление вопроса

Для редактирования вопроса нужно сделать его активным кликнув по нему и нажать кнопку «Edit». Откроется окно с настройками вопроса. Далее необходимо заполнить требуемые данные и нажать кнопку «Apply» или «OK» (рисунок 6.12).

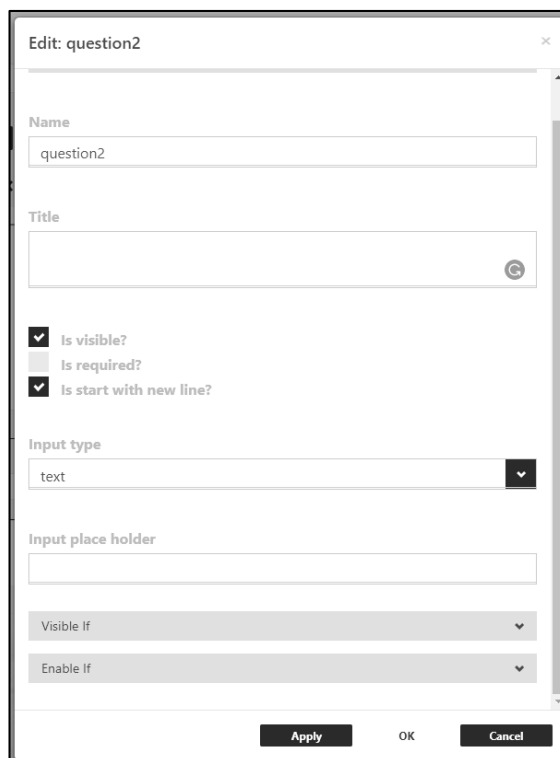


Рисунок 6.12 – Окно настроек вопроса

Для тестирования опроса нужно перейти на вкладку «Test Survey» редактора (рисунок 6.13). На ней отображается текущее состояние опроса. Опрос можно пройти (результаты прохождения опроса нигде сохранены не будут) для проверки корректности вопросов, триггеров и т. д.

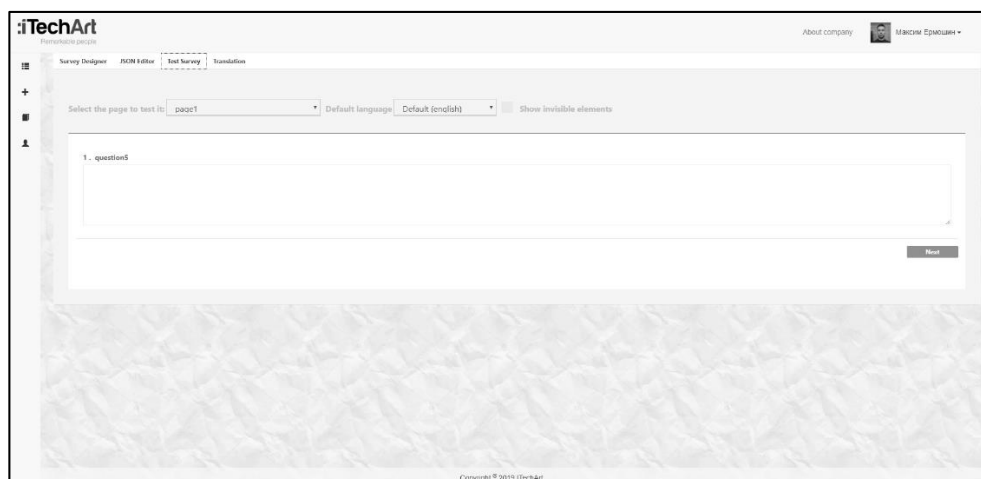


Рисунок 6.13 – Тестирование опроса

Для перевода опроса на различные языки необходимо перейти на вкладку «Translation» редактора (рисунок 6.14). В ней необходимо выбрать языки для перевода и заполнить необходимые поля. Все изменения сохраняются автоматически.

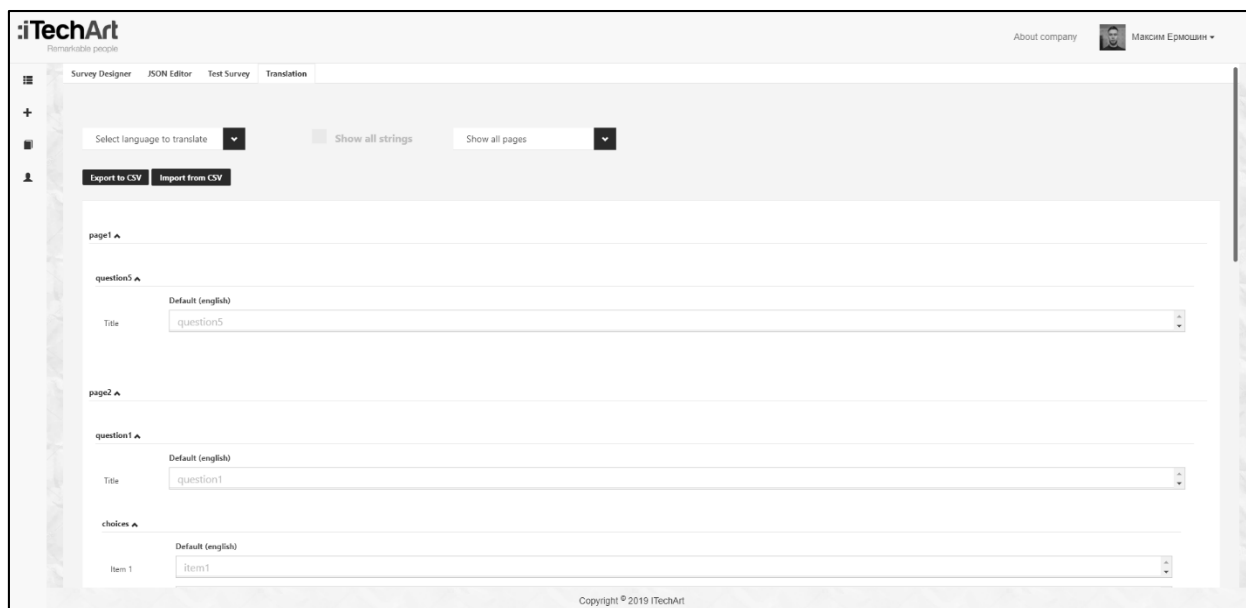


Рисунок 6.14 – Перевод опроса

Для сохранения опроса необходимо нажать кнопку «Save Survey» на вкладке редактирования опроса. Для сохранения опроса как шаблон необходимо нажать кнопку «Save as Template».

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

7.1 Краткая характеристика приложения

Веб-приложение создания и проведения опросов позволяет быстро создавать и проводить опросы. Пользователю предоставляется набор базовых шаблонов опросов. Ссылку на созданный опрос можно послать группе лиц, отобранных для прохождения. Так же пройти опрос сможет любой пользователь, который зарегистрирован в приложении.

Основная цель разрабатываемого приложения – это упрощение создания и проведения социальных опросов, включающая в себя снижение трудоёмкости и стоимости их создания и проведения. Так же требуется обеспечить простоту создания опросов, чтобы это мог сделать любой пользователь, который не обладает знаниями в этой области.

Основными функциями приложения являются:

- регистрация и авторизация пользователя;
- создание опроса;
- создание шаблонов опроса;
- использование шаблонов для создания опроса;
- редактирование прошлых версий опроса;
- прохождение опроса;
- просмотр создателем опроса статистики ответов на вопрос.

Приложение будет размещено в сети Internet. Разработчик получит экономический эффект в виде прироста прибыли от сотрудничества с различными компаниями или сервисами по размещению рекламы, которые захотят разместить объявление на сайте, при условии, что администратор одобрит соответствующие предложения от компании.

Также пользователи смогут приобрести премиум-пакет, который содержит расширенную функциональность приложения. Он будет включать возможность пользоваться премиум-шаблонами, которые не доступны обычному пользователю, а также проводить одновременно больше своих опросов, чем простой пользователь.

7.2 Расчёт затрат на разработку приложения

7.2.1 Расчёт основной заработной платы участников команды осуществляется по формуле:

$$З_o = K_{\text{пр}} \cdot \sum_{i=1}^n З_{\text{чи}} \cdot t_i, \quad (7.1)$$

где n – количество исполнителей, занятых разработкой конкретного программного продукта;

$K_{пр}$ – коэффициент премий, 1,5;

$З_{чi}$ – часовая тарифная ставка i -го исполнителя, руб.;

t_i – трудоёмкость работ, выполняемых i -м исполнителем, ч.

По данным ресурса dev.by в первом квартале 2019 года величина средней часовой ставки для .Net-разработчика с опытом работы 1-3 года – 7 рублей в час, для HTML/CSS/JS-разработчика с опытом работы 1-3 года – 7 рублей в час.

Исходя из приведённых выше данных, рассчитаем основную заработную плату исполнителей. Полученные данные приведены в таблице 7.1.

Таблица 7.1 – Расчёт заработной платы разработчиков

Участник команды	Месячная заработная плата, руб.	Часовая тарифная ставка руб.	Трудоёмкость, часов	Основная заработная плата, руб.
.Net-разработчик	1200	7	120	840
HTML/ CSS/ JS разработчик	1200	7	168	1176
Итого			288	2016
Премия, 50%				1008
Итого затраты на основную заработную плату разработчиков				3024

7.2.2 Затраты на дополнительную заработную плату команды разработчиков включает выплаты, предусмотренные законодательством о труде, и определяется по формуле:

$$З_d = \frac{З_o \cdot Н_d}{100}, \quad (7.2)$$

где $З_d$ – дополнительная заработная плата исполнителей, руб.;

$Н_d$ – норматив дополнительной заработной платы равный 10 %.

Подставив значения в формулу (7.2), получаем:

$$З_d = \frac{3024 \cdot 10}{100} = 302,4 \text{ руб.}$$

7.2.3 Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

$$З_{сз} = \frac{(З_о + З_д) \cdot Н_{сз}}{100}, \quad (7.3)$$

где $Н_{сз}$ – норматив отчислений в фонд социальной защиты населения и на обязательное страхование, 35 %.

Размер отчислений в фонд социальной защиты населения и на обязательное страхование составит:

$$З_{сз} = \frac{(3024 + 302,4) \cdot 35}{100} = 1164,24 \text{ руб.}$$

7.2.4 Расходы по статье «Прочие затраты» ($П_з$) для веб-ориентированного приложения включают затраты на приобретение лицензионного программного обеспечения, необходимого для разработки ПС, оплату потребляемой электроэнергии, оплату аренды рабочего помещения, оборудование рабочих мест и определяются по формуле:

$$П_з = \frac{З_о \cdot Н_{пз}}{100}, \quad (7.4)$$

где $Н_{пз}$ – норматив прочих затрат, 100 %.

Подставив данные в формулу (7.4), получаем:

$$П_з = \frac{3024 \cdot 100}{100} = 3024 \text{ руб.}$$

7.2.5 Полная сумма затрат на разработку приложения находится путём суммирования всех рассчитанных статей затрат. Все расчёты приведены в таблице 7.2.

Таблица 7.2 – Затраты на разработку приложения

Статья затрат	Сумма, руб.
Основная заработная плата команды разработчиков	3024
Дополнительная заработная плата команды разработчиков	302,4
Отчисления на социальные нужды	1164,24
Прочие затраты	3024
Общие затраты на разработку	7514,64

7.3 Оценка эффекта от использования приложения

Экономический эффект представляет собой прибыль, полученную от соглашений с компаниями, проявившими желание опубликовать объявление и получившими одобрение от администратора. Таким образом планируется заключить не менее 7 соглашений. Так как соглашение заключается на месяц и в среднем будут заключены соглашения на 4 месяца, по итогу получатся 28 месячных соглашений. Средний уровень цен на размещение объявлений на подобных сайтах составляет 300 руб. Так как предприятие является резидентом ПВТ оно освобождено от уплаты налога на прибыль.

$$\Pi_{\text{ч}} = \text{Ц} \cdot N - \text{НДС} - \text{З}_p, \quad (7.5)$$

где Ц – цена одного соглашения, руб.;

З_p – сумма расходов на разработку и реализацию, руб.;

N – количество соглашений;

НДС – сумма налога на добавленную стоимость, руб.

НДС рассчитывается по формуле:

$$\text{НДС} = \frac{\text{Ц} \cdot N \cdot \text{Н}_{\text{дс}}}{100\% + \text{Н}_{\text{дс}}}, \quad (7.6)$$

где $\text{Н}_{\text{дс}}$ – ставка налога на добавленную стоимость согласно действующему законодательству, 20 %.

Также экономический эффект включает в себя прибыль, полученную от реализации премиум-пакетов, которые содержат более расширенную функциональность разрабатываемого приложения. Планируется продать не менее 10 пакетов сроком на 3 месяца. Итого 30 месячных продаж. Средний уровень цен на покупку премиум-пакета на подобных сайтах составляет 300 руб.

Исходя из данных сумма налога на добавленную стоимость составит:

$$\text{НДС} = \frac{(300 \cdot 28 + 300 \cdot 30) \cdot 20}{100 + 20} = 2900 \text{ руб.}$$

Таким образом можно подсчитать чистую прибыль разработчиков:

$$\Pi_{\text{ч}} = 300 \cdot 28 + 300 \cdot 30 - 2900 - 7514,64 = 5785,36 \text{ руб.}$$

Для оценки эффективности затрат на разработку приложения необходимо рассчитать уровень рентабельности затрат по следующей формуле:

$$P = \frac{\Pi_{\text{ч}}}{\text{З}_p} \cdot 100 \%. \quad (7.7)$$

Подставив значения в формулу (7.7), получаем:

$$P = \frac{5785,36}{7514,64} \cdot 100\% = 77 \, \%.$$

Проект будет экономически эффективным, если рентабельность затрат на разработку приложения будет не меньше средней процентной ставки по банковским депозитным вкладам. Средняя процентная ставка по банковским вкладам для юридических лиц за март 2019 года составила 8,73 %. Исходя из этих данных можно сделать вывод, что проект рентабелен, так как рентабельность затрат на разработку приложения составила 77 %.

7.4 Расчёт показателей эффективности инвестиций в разработку приложения

Экономическая целесообразность инвестирования в разработку данного веб-ориентированного приложения можно отобразить через рентабельность инвестиций, которая вычисляется по формуле (7.8).

Чтобы рассчитать эффективность инвестиций в разработку приложения, необходимо сравнить размер инвестиций в разработку приложения, и получаемый годовой экономический эффект.

Для расчёта рентабельности инвестиций воспользуемся следующей формулой:

$$P_{\text{и}} = \frac{\Pi_{\text{ч}}}{З_{\text{р}}} \cdot 100 \, \%. \quad (7.8)$$

Таким образом рентабельность инвестиций составит:

$$P_{\text{и}} = \frac{5785,36}{7514,64} \cdot 100\% = 77 \, \%.$$

Так как рентабельность предприятия превышает средний процент по долгосрочным вкладам в банках (8,73 %) можно сделать вывод, что инвестиции будут прибыльнее, чем банковский вклад.

7.5 Вывод

Таким образом, полученные результаты технико-экономического обоснования «Веб-приложения создания и проведения опросов с использованием технологий React, Redux, WebApi 2» свидетельствуют об эффективности разработки и внедрения в эксплуатацию данного веб-ориентированного приложения. Окупаемость произойдёт в течение одного года. Общая сумма затрат составила 7514,64 руб., рентабельность инвестиций 77 %.

ЗАКЛЮЧЕНИЕ

В ходе работы над дипломным проектом была проанализирована литература, связанная с созданием и проведением онлайн-опросов, а также проведено исследование для выявления существующих аналогов, чтобы выделить их достоинства и недостатки, которые необходимо было устранить в разрабатываемом приложении. По результатам исследования были сформулированы задачи на дипломное проектирование.

Во время работы проведён этап моделирования приложения, в котором были сформулированы функциональные требования к приложению и составлены полные спецификации к ним.

На основе функциональных требований произведено проектирование приложения. Оно включало в себя разработку архитектуры приложения, разработку базы данных и разработку алгоритмов отдельных функций приложения.

Были изучены необходимые библиотеки, фреймворки, шаблоны и кодстайлы, требующиеся для создания приложения. Данное приложение учитывает возможность того, что с ним будут работать очень много пользователей одновременно, что скажется на стабильности работы при пиковой нагрузке.

Также для обеспечения стабильной работы приложения разработаны тестовые случаи, покрывающие всю его функциональность. Все тесты успешно выполняются, что свидетельствует о корректном исполнении функций приложения.

На завершающем этапе подробно описана методика использования приложения, позволяющая в короткие сроки освоить работу с ним.

Также был рассчитан экономический эффект от внедрения приложения. В результате расчётов было установлено, что приложение является экономически выгодным, т. к. окупается за приемлемые сроки.

Таким образом, итогом дипломного проектирования стало web-приложение, которое помогает быстро и дёшево проводить онлайн опросы для пользователей по всему миру, помогая собирать необходимые данные и быстро реагировать на мнение пользователей, пользующихся тем или иным продуктом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Макконнелл, С. Совершенный код. Мастер-класс / Пер. с англ. / С. Макконнелл. – М.: Издательско-торговый дом «Русская редакция», 2010. – 896 с.
- [2] Антопольский А. Б., Правовые и технологические проблемы создания и функционирования электронных библиотек. / А. Б. Антопольский, Т. С. Маркарова, Е. А. Данилина – М.: ИНИЦ «Патент», 2008. — 192 с. — ISBN 978-5-89513-119-0.
- [3] Баранова Л. Т., Что такое «аудиокнига» и история ее развития/ Л. Т. Баранова // Актуальные проблемы гуманитарных и естественных наук. – 2015. – №3. – С. 118–121. — ISSN 2073-0071
- [4] Фролов А., Синтез и распознавание речи. Современные решения [Электронный ресурс] / Фролов А., Фролов Г. – Электрон. журн. – 2003. – Режим доступа: <http://www.frolov-lib.ru/>.
- [5] Лобанов Б. М., Компьютерный синтез и клонирование речи. / Б. М. Лобанов, Л. И. Цирульник – Минск : Издательский дом «Белорусская Наука», 2008. — 316 с.
- [6] Audio-knigki.com [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://audio-knigki.com/>.
- [7] Baza-Knig [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://baza-knig.ru/>.
- [8] Au-books [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://au-books.com/>.
- [9] VoxWorker [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://voxworker.com/ru>.
- [10] UniTools [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://unitools.tech/voice>.
- [11] Zvukogram [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://zvukogram.com/>.
- [12] Рогочев С., Обобщенный Model-View-Controller [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://rsdn.org/article/patterns/generic-mvc.xml>.
- [13] Структура реляционных баз, данных [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.oracle.com/ru/database/what-is-a-relational-database/>

ПРИЛОЖЕНИЕ А

(обязательное)

Текст программы

SurveyService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Threading.Tasks;
using AutoMapper;
using JetBrains.Annotations;
using RedTeam.Logger;
using
RedTeam.TechArtSurvey.Foundation.Interfaces.ServiceResponses;
using RedTeam.TechArtSurvey.Repositories.Interfaces;
using RedTeam.TechArtSurvey.DomainModel.Entities.Surveys;
using RedTeam.TechArtSurvey.Foundation.Dto.SurveysDto;
using RedTeam.TechArtSurvey.Foundation.Interfaces;
using RedTeam.TechArtSurvey.Foundation.Responses;
using RedTeam.Common.EnvironmentInfo;
using RedTeam.Common.Validator;

namespace RedTeam.TechArtSurvey.Foundation.Services
{
    [UsedImplicitly]
    public class SurveyService : ISurveyService
    {
        private readonly ITechArtSurveyUnitOfWork _uow;
        private readonly IMapper _mapper;
        private readonly IEnvironmentInfoService
_environmentInfoService;
        private readonly IValidatorFactory _validatorFactory;

        public SurveyService(ITechArtSurveyUnitOfWork uow,
IMapper mapper,
IEnvironmentInfoService environmentInfoService,
IValidatorFactory validatorFactory)
        {
            _uow = uow;
            _mapper = mapper;
            _environmentInfoService = environmentInfoService;
            _validatorFactory = validatorFactory;
        }

        public async Task<IServiceResponse<SurveyDto>>
CreateAsync(EditSurveyDto surveyDto)
        {

```

```

        LoggerContext.Logger.Info($"Create Survey
'{surveyDto.Title.Default}'");

        var survey = await
PrepareSurvey(_mapper.Map<EditSurveyDto,
SurveyOnlyVersion>(surveyDto).ToSurvey());

        var version = survey.Versions.First();

        if(!version.Pages.All(
            page =>
            {
                var res = page.Questions.All(
                    question =>
                    {
                        var result = _validatorFactory.
                            GetValidator(question.Type.Type).
                            ValidateDefaultValue(question.Default);
                        return result;
                    });
                return res;
            }
        )
        )
        {
            return
ServiceResponse.CreateUnsuccessful<SurveyDto>(ServiceResponseCod
e.DefaultValueIsWrong);
        }

        version.CreatedDate =
_environmentInfoService.CurrentUtcDateTime;
        version.StartDate =
_environmentInfoService.CurrentUtcDateTime;
        version.EndDate =
_environmentInfoService.CurrentUtcDateTime;
        version.Number = 1;

        _uow.Surveys.Create(survey);
        await _uow.SaveChangesAsync();

        return
ServiceResponse.CreateSuccessful(_mapper.Map<Survey,
SurveyDto>(survey));
    }

    public async Task<IServiceResponse>
UpdateAsync(EditSurveyDto survey)
    {
        LoggerContext.Logger.Info($"Update Survey with id =
{survey.Id}");
    }

```

```

        var surv = await
_uow.Surveys.GetByIdAsync(survey.Id, s => s.Versions);
        if (surv == null)
        {
            return
ServiceResponse.CreateUnsuccessful<object>(ServiceResponseCode.S
urveyNotFoundById);
        }

        var su = await
PrepareSurvey(_mapper.Map<EditSurveyDto, Survey>(survey));
        var version = su.Versions.First();

        if (version.Pages.Any(
            page => !page.Questions.Any(
                question => _validatorFactory.
                    GetValidator(question.Type.Type).
                        ValidateDefaultValue(question.Default))
            )
        )
        {
            return
ServiceResponse.CreateUnsuccessful<SurveyDto>(ServiceResponseCod
e.DefaultValueIsWrong);
        }

        version.Number = surv.Versions.Count + 1;
        version.CreatedDate =
_environmentInfoService.CurrentUtcDateTime;
        await _uow.Surveys.UpdateVersionAsync(survey.Id,
version);
        await _uow.SaveAsync();

        return ServiceResponse.CreateSuccessful();
    }

    public async Task<IServiceResponse> DeleteByIdAsync(int
id)
    {
        LoggerContext.Logger.Info($"Delete Survey with id =
{id}");

        var includes = new Expression<Func<Survey,
object>>[]
        {
            s => s.Author,
            s => s.Versions,
            s => s.Versions.Select(v => v.Responses),
            s => s.Versions.Select(v => v.Responses.Select(r
=> r.Answers)),
            s => s.Versions.Select(v => v.Pages),
            s => s.Versions.Select(v => v.Pages.Select(p =>

```

```

p.Questions)),
    s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.Type))),
    s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.Choices)))
    };

    var surv = await _uow.Surveys.GetByIdAsync(id,
includes);
    if (surv == null)
    {
        return
ServiceResponse.CreateUnsuccessful<object>(ServiceResponseCode.S
urveyNotFoundById);
    }

    var versions = surv.Versions.ToArray();
    var pages = versions.SelectMany(sv =>
sv.Pages).ToArray();
    var questions = pages.SelectMany(sp =>
sp.Questions).ToArray();
    var variants = questions.SelectMany(q =>
q.Choices).ToArray();
    var responses = versions.SelectMany(sv =>
sv.Responses).ToArray();
    var answers = responses.SelectMany(sr =>
sr.Answers).ToArray();

    _uow.GetRepository<QuestionVariant>().DeleteRange(variants);
    _uow.GetRepository<QuestionAnswer>().DeleteRange(answers);
    _uow.GetRepository<Question>().DeleteRange(questions);
    _uow.GetRepository<SurveyPage>().DeleteRange(pages);
    _uow.GetRepository<SurveyResponse>().DeleteRange(responses);
    _uow.GetRepository<SurveyVersion>().DeleteRange(versions);
    _uow.GetRepository<Survey>().Delete(surv);

    await _uow.SaveAsync();

    return ServiceResponse.CreateSuccessful();
}

public async Task<IServiceResponse<EditSurveyDto>>
GetByIdAsync(int id)
{
    LoggerContext.Logger.Info($"Get Survey with id =
{id}");

```

```

        var surv = await _uow.Surveys.GetByIdAsync(id);
        return surv == null
            ?
            ServiceResponse.CreateUnsuccessful<EditSurveyDto>(ServiceResponseCode.SurveyNotFoundById)
            :
            ServiceResponse.CreateSuccessful(_mapper.Map<Survey,
            EditSurveyDto>(surv));
    }

    public async Task<IServiceResponse<EditSurveyDto>>
    GetByIdAndVersionAsync(int id, int version)
    {
        LoggerContext.Logger.Info($"Get Survey with id =
        {id} and version = {version}");

        var includes = new Expression<Func<Survey,
object>>[]
        {
            s => s.Versions,
            s => s.Versions.Select(v => v.CompletedHtml),
            s => s.Versions.Select(v => v.CompleteText),
            s => s.Versions.Select(v => v.PageNextText),
            s => s.Versions.Select(v => v.PagePrevText),
            s => s.Versions.Select(v => v.StartSurveyText),
            s => s.Versions.Select(v => v.Title),
            s => s.Versions.Select(v => v.Triggers),
            s => s.Versions.Select(v => v.Pages),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Title)),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions)),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.Placeholder))),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.OptionsCaption))),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.MaxRateDescription))),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.MinRateDescription))),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.MatrixRows))),
            s => s.Versions.Select(v =>
            v.Pages.Select(p => p.Questions.Select(q =>
q.MatrixRows.Select(mr => mr.Text))),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.MatrixCols))),
            s => s.Versions.Select(v =>
            v.Pages.Select(p => p.Questions.Select(q =>
q.MatrixCols.Select(mc => mc.Text))),
            s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.MinRateDescription))),

```

```

        s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.Type))),
        s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.Title))),
        s => s.Versions.Select(v => v.Pages.Select(p =>
p.Questions.Select(q => q.Choices))),
        s => s.Versions.Select(v =>
        v.Pages.Select(p => p.Questions.Select(q =>
q.Choices.Select(qv => qv.Text)))),
        s => s.Author
    };

    var surv = await
_uow.Surveys.GetSurveyByIdAndVersionAsync(id, version,
includes);

    if (surv == null)
    {
        return
ServiceResponse.CreateUnsuccessful<EditSurveyDto>(ServiceRespons
eCode.SurveyNotFoundById);
    }

    var su = SurveyOnlyVersion.FromSurveyByVersion(surv,
version);

    return su.Version == null
    ?
ServiceResponse.CreateUnsuccessful<EditSurveyDto>(ServiceRespons
eCode.SurveyNotFoundByVersion)
    :
ServiceResponse.CreateSuccessful(_mapper.Map<SurveyOnlyVersion,
EditSurveyDto>(su));
}

public async
Task<IServiceResponse<IReadOnlyCollection<SurveyDto>>>
GetAllAsync()
{
    LoggerContext.Logger.Info("Get all Surveys");

    var includes = new Expression<Func<Survey,
object>>[]
    {
        s => s.Versions,
        s => s.Versions.Select(v => v.Title),
        s => s.Author
    };

    var surveys = await
_uow.Surveys.GetAllAsync(includes);

```

```

        return
        ServiceResponse.CreateSuccessful(_mapper.Map<IReadOnlyCollection<Survey>, IReadOnlyCollection<SurveyDto>>(surveys));
    }

    public async
    Task<IServiceResponse<IReadOnlyCollection<SurveyDto>>>
    GetAllAsync(string userEmail)
    {
        LoggerContext.Logger.Info($"Get all Surveys by
author {userEmail}");

        var includes = new Expression<Func<Survey,
object>>[]
        {
            s => s.Versions,
            s => s.Versions.Select(v => v.Title),
            s => s.Author
        };

        var surveys = await
        _uow.Surveys.GetAllByEmailAsync(userEmail, includes);

        return
        ServiceResponse.CreateSuccessful(_mapper.Map<IReadOnlyCollection<Survey>, IReadOnlyCollection<SurveyDto>>(surveys));
    }

    private async Task<Survey> PrepareSurvey(Survey survey)
    {
        var user = await
        _uow.Users.GetUserByEmailAsync(survey.Author.Email);
        survey.Author = user ?? throw new
        NullReferenceException(nameof(survey.Author));

        foreach (var version in survey.Versions)
        {
            foreach (var page in version.Pages)
            {
                foreach (var question in page.Questions)
                {
                    if (!Enum.TryParse(question.Type.Name, out
QuestionTypes qt))
                    {
                        throw new
NullReferenceException(nameof(question.Type));
                    }
                    var questionType = await
_uow.QuestionTypes.FindByTypeAsync(qt) ??
                    throw new
NullReferenceException(nameof(question.Type));
                    question.Type = questionType;
                }
            }
        }
    }

```

```

        var empty = new LocalizableString {Default =
""};
        if (question.MinRateDescription == null)
question.MinRateDescription = empty;
        if (question.MaxRateDescription == null)
question.MaxRateDescription = empty;
        if (question.OptionsCaption == null)
question.OptionsCaption = empty;
        if (question.Placeholder == null)
question.Placeholder = empty;
    }
}
}

return survey;
}
}
}

```

SurveyEditor.jsx

```

import React, { Component } from 'react';
import { connect } from 'react-redux';
import PropTypes from 'prop-types';
import ReactRouterPropTypes from 'react-router-prop-types';
import { pushSurveyRequest, getSurveyRequest } from
'./actions';
import { EditorUtils } from './editorUtils';
import * as SurveyJSEditor from 'surveyjs-editor';
import * as SurveyKo from 'survey-knockout';
import $ from 'jquery';

import 'survey-react/survey.css';

import 'surveyjs-editor/surveyeditor.css';
import './SurveyEditor.scss';

import 'jquery-ui/themes/base/all.css';
import 'select2/dist/css/select2.css';

import 'jquery-bar-rating/dist/themes/css-stars.css';
import 'jquery-bar-rating/dist/themes/fontawesome-
stars.css';

import 'jquery-ui/ui/widgets/datepicker.js';
import 'select2/dist/js/select2.js';
import 'jquery-bar-rating';

import 'icheck/skins/square/blue.css';

import * as widgets from 'surveyjs-widgets';

```



```

widgets.jquerybarrating(SurveyKo, $);
widgets.jqueryuidatepicker(SurveyKo, $);

class SurveyEditor extends Component {
  editor;

  componentWillMount() {
    const {surveyId, version} = this.props.match.params;
    if (surveyId && version) {
      this.props.getSurveyRequest(surveyId, version);
    } else {
      //error
    }
  }

  componentDidMount() {
    let editorOptions = {
      generateValidJSON: true,
      showJSONEditorTab: true,
      showTestSurveyTab: true,
      showEmbeddedSurveyTab: false,
      showTranslationTab: true,
      showPropertyGrid: false,
      questionTypes: ['text', 'checkbox', 'radiogroup',
'dropdown', 'boolean', 'comment', 'rating', 'matrix'],
      isAutoSave: false,
      isRTL: true,
      showPagesToolbox: true,
      useTabsInElementEditor: false,
      showState: false,
    };

    var mainColor = '#32292a';
    var mainHoverColor = '#ff0000';
    var textColor = '#4a4a4a';
    var defaultThemeColorsEditor = SurveyJSEditor
      .StylesManager
      .ThemeColors['default'];
    defaultThemeColorsEditor['$primary-color'] = mainColor;
    defaultThemeColorsEditor['$secondary-color'] =
mainColor;
    defaultThemeColorsEditor['$primary-hover-color'] =
mainHoverColor;
    defaultThemeColorsEditor['$primary-text-color'] =
textColor;
    defaultThemeColorsEditor['$selection-border-color'] =
mainColor;
    SurveyJSEditor.StylesManager.applyTheme();

    this.editor = new SurveyJSEditor.SurveyEditor(
      'surveyEditorContainer',

```

```

        editorOptions,
    );
    this.editor.saveSurveyFunc = this.saveMySurvey;
    this.editor.showErrorOnFailedSave = true;

    this.editor.onPropertyValidationCustomError.add((e, opt)
=> {
        console.log(e, '\n', opt);
        switch(opt.propertyName) {
        case 'title':
            if(opt.value === '') opt.error = 'Title can\'t be
empty';
            break;
        default:
            break;
        }
    });

    this.editor
        .onCanShowProperty
        .add(function (sender, options) {
            if(options.property.name == 'choicesByUrl')
options.canShow = false;
            if (options.obj.getType() == 'survey') {
                options.canShow = [
                    'title',
                    'locale',
                    'showPageNumbers',
                    'pagePrevText',
                    'pageNextText',
                    'completeText',
                    'startSurveyText',
                    'showNavigationButtons',
                    'showPrevButton',
                    'firstPageIsStarted',
                    'showCompletedPage',
                    'completedHtml',
                    'maxTimeToFinish',
                    'maxTimeToFinishPage',
                    'showTimerPanel',
                    'showTimerPanelMode',
                    'goNextPageAutomatic',
                    'showProgressBar',
                    'isSinglePage',
                    'questionTitleLocation',
                    'requiredText',
                    'showQuestionNumbers',
                    'questionErrorLocation',
                    'questionsOrder',
                    'triggers',
                ].includes(options.property.name);
            }
        });
    }

```

```

    });

    $(' .svd_commercial_container').remove();
  }

  render() {
    if(this.editor) this.editor.text =
JSON.stringify(this.prepareAfterLoad(this.props.survey));
    return <div id="surveyEditorContainer" />;
  }

  saveMySurvey = (no, doSaveCallback) => {
    console.log(new SurveyKo.SurveyError('Title can\'t be
empty'));
    this.editor.survey.pages[0].elements[0].errors.push(new
SurveyKo.SurveyError('Title can\'t be empty'));
    doSaveCallback(no, false);
    let survey = JSON.parse(this.editor.text);
    survey.author = {
      userName: this.props.userName,
      email: this.props.email,
    };
    let s = this.prepareSurveyToSend(survey,
this.editor.survey);
    this.props.pushSurveyRequest(JSON.stringify(s));
  };

  prepareAfterLoad = EditorUtils.prepareAfterLoad;

  prepareSurveyToSend = EditorUtils.prepareSurveyToSend;
}

const mapStateToProps = (state) => ({
  userName : state.auth.userInfo.userName,
  email : state.auth.userInfo.email,
  survey : state.surveys.survey,
});

const mapDispatchToProps = ({pushSurveyRequest,
getSurveyRequest});

SurveyEditor.propTypes = {
  match : ReactRouterPropTypes.match.isRequired,
  userName : PropTypes.string.isRequired,
  email : PropTypes.string.isRequired,
  survey : PropTypes.object.isRequired,
  pushSurveyRequest : PropTypes.func.isRequired,
  getSurveyRequest : PropTypes.func.isRequired,
};

export default connect(mapStateToProps,
mapDispatchToProps)(SurveyEditor);

```