

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 СОЗДАНИЯ И ПРОВЕДЕНИЯ ОПРОСОВ С
ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ REACT, REDUX, WEBAPI 2**

БГУИР ДП 1-40 01 01 01 033 ПЗ

Студент	М.А. Ермошин
---------	--------------

Руководитель	С.В. Болтак
--------------	-------------

Консультанты: <i>от кафедры ПОИТ</i> <i>по экономической части</i>	С.В. Болтак Т.Л. Слюсарь
--	-----------------------------

Нормоконтролер	С.В. Болтак
----------------	-------------

Рецензент	
-----------	--

Минск 2019

РЕФЕРАТ

ВЕБ-ПРИЛОЖЕНИЕ СОЗДАНИЯ И ПРОВЕДЕНИЯ ОПРОСОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ REACT, REDUX, WEBAPI 2: дипломный проект / М. А. Ермошин. – Минск : БГУИР, 2019, – 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 %.

СОДЕРЖАНИЕ

Введение.....	7
1 Анализ приложений создания и проведения опросов.....	8
1.1 Анализ литературных источников.....	8
1.2 Прототипы, их недостатки и достоинства	9
1.3 Формирование технического задания приложения	20
2 Моделирование предметной области и разработка функциональных требований.....	23
2.1 Описание функциональности приложения.....	23
2.2 Спецификация функциональных требований	26
3 Проектирование приложения	39
3.1 Разработка архитектуры приложения	39
3.2 Разработка физической модели базы данных.....	42
3.3 Разработка алгоритма приложения и алгоритмов отдельных модулей.....	44
4 Создание приложения.....	49
5 Тестирование, проверка работоспособности и анализ полученных результатов	57
6 Руководство по установке и использованию	68
7 Техничко-экономическое обоснование	75
7.1 Краткая характеристика приложения.....	75
7.2 Расчёт затрат на разработку приложения	75
7.3 Оценка эффекта от использования приложения	78
7.4 Расчёт показателей эффективности инвестиций в разработку приложения	79
7.5 Вывод	79
Заключение	80
Список использованных источников	81
Приложение А (обязательное) Текст программы.....	82

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

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

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.

Мэппинг – 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

В современном мире многим компаниям для того, чтобы оставаться конкурентоспособными необходимо собирать отзывы своих клиентов и быстро реагировать на их изменения. Для этого есть много способов. К ним относятся эксперимент, наблюдение, изучение различных документов. Но один из лучших способов сбора этой информации – проведение опроса.

Опрос – это чаще всего небольшой перечень вопросов, способных дать необходимую информацию. Для улучшения качества результатов вопросы могут разбиваться по темам, могут вводиться вопросы-фильтры для отсеивания анкет, заполненных ненадлежащим образом.

В большинстве случаев либо из-за дальности расположения, либо из-за нехватки времени респондентов бумажные опросы проводить не представляется возможным. На выручку приходят онлайн-сервисы для проведения опросов. Большинство людей имеют доступ к сети и ежедневно просматривают огромное множество сайтов. При наличии заинтересованности в развитии продукта, которым они пользуются, пройти опрос за 5-10 минут им не составляет труда.

Целью данного дипломного проекта является создание приложения для создания и проведения онлайн-опросов. Актуальность темы обусловлена современными тенденциями развития информационного общества, благодаря которым появилась возможность создавать и проводить опросы быстрее, качественнее и дешевле.

После составления приложение сохранит созданный опрос на сервере. Так же в приложении можно будет создавать шаблоны опросов для ещё большего ускорения работы. После создания пользователю нужно будет лишь установить временные рамки проведения опроса, в течение которых он станет доступен пользователям по всему миру и, при необходимости, разослать ссылки на опрос отобранным респондентам.

1 АНАЛИЗ ПРИЛОЖЕНИЙ СОЗДАНИЯ И ПРОВЕДЕНИЯ ОПРОСОВ

1.1 Анализ литературных источников

По своей технологии и организации онлайн-опрос наиболее близок к одной из самых старых схем массовых опросов - почтовому анкетированию, методу, имеющему более чем столетнюю историю. Самые яркие годы - это период с 1916 до начала 1930-х годов, когда американский журнал «The Literary Digest», высылая миллионы опросных бюллетеней своим подписчикам, успешно предсказал итоги трёх президентских выборов. Особенно внушительным был успех 1932 года, когда журнал предсказал победу Франклина Рузвельта с 59,85% голосов, а он набрал - 59,14% [1].

В наши дни глобальная сеть получила широкое распространение по всему миру. Она позволяет доставлять информацию на большие расстояния в рекордно короткие сроки, особенно по сравнению с традиционной бумажной почтой. Это и сыграло огромную роль в переходе с бумажных опросов к онлайн-опросам.

Вот некоторые преимущества онлайн-опросов:

- экономия ресурсов (денег, трудозатрат и времени);
- широта охвата (преодоление расстояний и границ, доступ к различным сообществам и социальным группам);
- оставляют возможность выбора удобного места и времени для участия и могут быть завершены в любое удобное для респондента время;
- релевантность (самостоятельность) коммуникации, т. е. более низкий уровень влияния интервьюера на респондента, возможность давать более развёрнутые ответы;
- широта охвата предметных полей (возможность изучать закрытые для публичного обсуждения и деликатные темы);
- возможность автоматической проверки анкет.

Главный недостаток онлайн-опросов связан с проблемой обеспечения репрезентативности выборки. Во-первых, проблема охвата, то есть неспособность используемой выборочной процедуры охватить реальную генеральную совокупность (т. е. задать известную ненулевую вероятность попадания в выборку для каждой единицы совокупности). Во-вторых, это отсутствие основы выборки. Эта проблема может быть успешно решена в исследованиях организаций с широкими сетевыми базами, а также при построении выборки по результатам оффлайн опроса [2].

Так же во многих случаях большой проблемой является анонимность респондентов, так как при наличии заинтересованности можно большим количеством повторных прохождений понизить достоверность результатов опросов. Однако в большинстве случаев эту проблему можно решить без особых усилий.

1.2 Прототипы, их недостатки и достоинства

Приложение создания и проведения опросов – это система из набора инструментов, которая позволяет создавать онлайн-опросы без каких-либо специальных знаний. С её помощью можно выбрать готовый шаблон опроса или создать новый опрос выбирая различные типы вопросов. Для размещения опроса в сети не понадобится годами изучать языки программирования для создания различных страниц так как приложение предоставляет функционал проведения опроса.

Также важным достоинством является наличие готовых шаблонов для использования, что может значительно ускорить создание опроса. Очень удобным является наличие функционала, позволяющего сделать опрос более динамическим, т. е. скрывать или показывать некоторые вопросы или страницы в зависимости от предыдущих ответов респондента, делать простые вычисления на основе введённых данных и т. д.

Одним из самых популярных приложений создания и проведения опросов, является Google Forms (рисунок 1.1) [3].

The image shows a screenshot of a Google Form titled 'Контактная информация' (Contact Information). The form is displayed on a mobile device interface, with a green header bar at the top containing navigation icons and an 'ОТПРАВИТЬ' (SEND) button. The form itself has a white background and is divided into two tabs: 'ВОПРОСЫ' (QUESTIONS) and 'ОТВЕТЫ' (ANSWERS). The 'ВОПРОСЫ' tab is active, showing a series of input fields. The first field is 'Имя *' (Name *), followed by 'Адрес электронной почты *' (Email address *), 'Адрес *' (Address *), 'Номер телефона' (Phone number), and 'Комментарии' (Comments). Each field has a label, a requiredness indicator (red asterisk), and a placeholder text like 'Краткий ответ' (Short answer) or 'Развернутый ответ' (Long answer). A vertical toolbar on the right side of the form contains icons for adding new questions, sections, and other elements. The overall layout is clean and user-friendly, typical of Google's design language.

Рисунок 1.1 – Приложение создания и проведения опросов Google Forms

Это бесплатное приложение, в котором можно создавать разнообразные опросы от обычного голосования до письменных заданий. Создавать и редактировать анкеты можно как с ПК, так и с мобильных устройств. Для мобильной версии требуется установка специального приложения.

В приложении предусмотрены различные шаблоны оформления, возможность загрузки своего логотипа и фото, а также возможность назначить количество баллов за ответы для проведения тестирования.

На выбор предлагается несколько типов вопросов:

- одиночный и множественный выборы;
- текстовые поля;
- дата и время;
- усложнённые сетки ввода.

Допускается вставка видео с YouTube или картинки. Можно создать многостраничный опрос, тогда участник будет переходить на другие страницы в зависимости от своих ответов. Приложение позволяет использовать дополнительные плагины. Дизайн выполнен в минималистичном стиле и приятен глазу.

У данного приложения можно выделить следующие преимущества:

- неограниченное количество анкет;
- выгрузка ответов в таблицу Google;
- различные темы оформления.

Недостатки:

- для прохождения обязательно нужно пройти аутентификацию;
- создатель может настроить сбор email адресов респондентов без их согласия.

Итак, Google Forms – это одно из лучших приложений создания и проведения опросов различных типов.

Ещё одним прекрасным приложением создания и проведения опросов является Typeform (рисунок 1.2) [4].

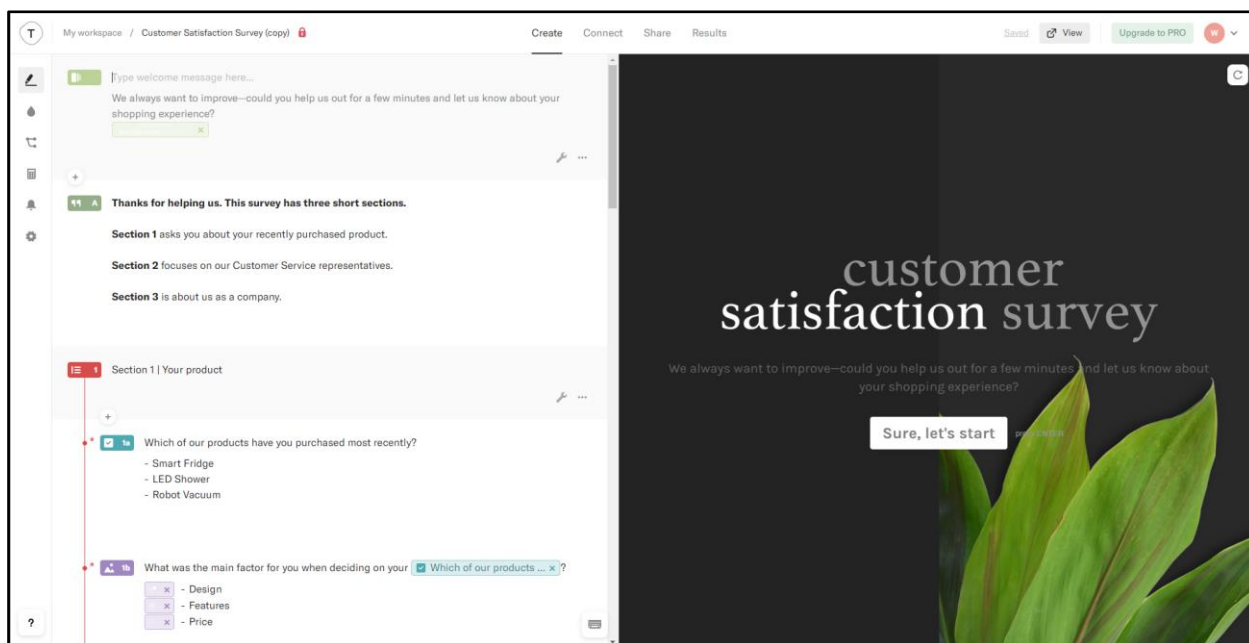


Рисунок 1.2 – Приложение создания и проведения опросов Typeform

При работе с этим приложением невозможно не обратить внимание на то, что разработчики очень хорошо поработали над дизайном создаваемых

опросов. Платформа отличается возможностью создавать стильные формы: доступно множество вариантов дизайна. Это универсальный инструмент для сбора нужной информации, который позволяет создавать опросы и вставлять их на свой сайт. Предусмотрена адаптивная версия для смартфонов. Разработчики смешали автоматические формы с творчеством и получили комфортный сервис для организации анкет.

Приложение предлагает работу в бесплатном тарифе, но с существенными ограничениями. Доступна возможность создания собственных шаблонов за счёт встроенного редактора. Для безопасности данных вся информация защищена SSL-шифрованием. Экспорт данных осуществляется в формате .xls.

Преимущества:

- загрузка файлов;
- настраиваемые уведомления;
- бесплатный доступ к API;
- создание своих тем и шаблонов.

Недостатки:

- практически все возможности доступны только по подписке;
- бесплатный тариф даёт возможность создавать опросы максимум на 10 полей и 100 ответов в месяц.

Tureform несомненно одно из лучших приложений создания и проведения опросов в плане дизайна как опросов, так и редактора для их создания. Для тех, кто хочет не только собрать нужную информацию, но и произвести впечатление на респондентов это определённо лучший выбор.

Также стоит обратить внимание на такое приложение как SurveyMonkey (рисунок 1.3) [5].

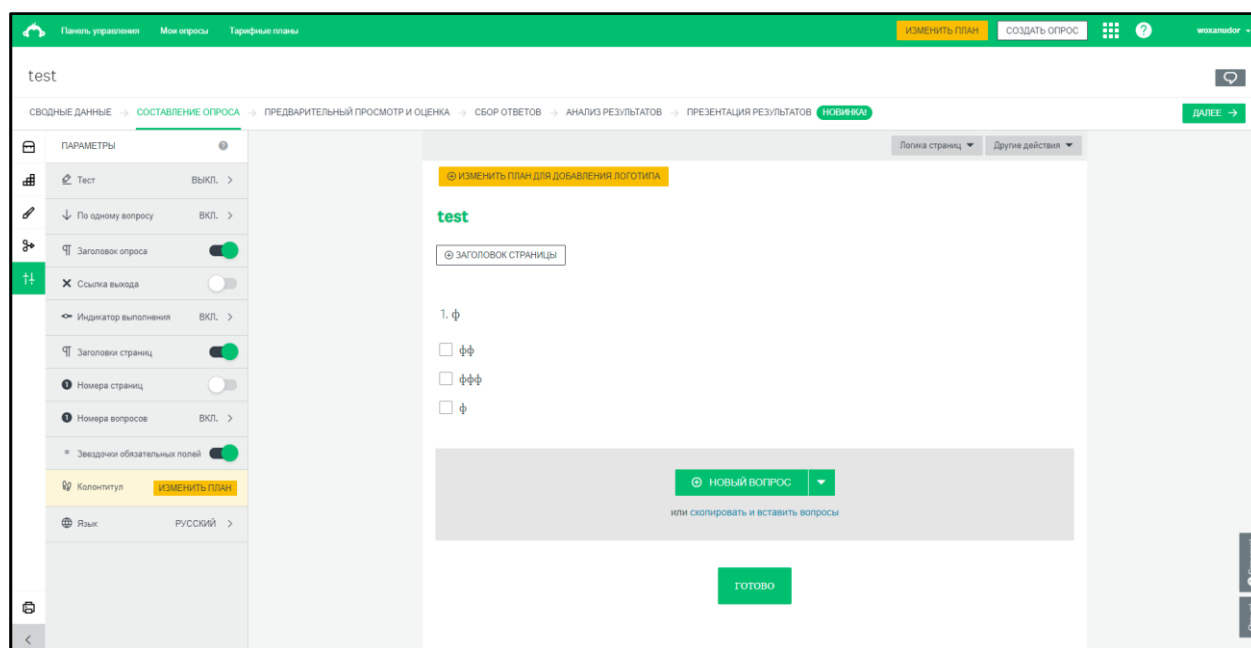


Рисунок 1.3 – Приложение создания и проведения опросов SurveyMonkey

Этот сервис предназначен для создания быстрых онлайн-опросов. В бесплатном тарифе предусмотрено до 10 вопросов для 1 анкеты и работа со 100 респондентами. Подходит для крупных компаний, поскольку программа гарантирует высокий уровень контроля и безопасность во время управления вашими данными. Может использоваться для организации совещания, опроса покупателей, анкетирования участников мероприятия. Доступно также мобильное приложение.

Вы можете создать опрос и прямо во время урока или совещания смотреть на статистику полученных ответов. Сервис позволяет вставлять ссылку на анкету в сайт или делиться ею в социальных сетях. В платформе доступны графики с подробными отчётами. Сервис включает в себя инструменты для совместной работы, создание тестов с выставлением оценки, возможности для брендинга анкеты. Полученные ответы можно экспортировать в .pdf, .xls, .csv, .ppt.

Преимущества:

- аналитика опросов;
- настраиваемый URL-адрес;
- переадресация после завершения и страница благодарностей;
- рассылка опросов диспетчером;
- повышенная безопасность (SSL);
- фильтры и перекрёстные таблицы ответов по критериям;
- возможности для настройки брендинга;
- А/В-тестирование;
- проведение опроса одновременно на нескольких языках (при наличии платной подписки)
- просмотр данных во время анкетирования.

Недостатки:

- отсутствует возможность вставлять медиафайлы в вопрос;
- достаточно сложный интерфейс;
- бесплатный тариф даёт возможность создавать опросы максимум на 10 вопросов, ограничивает возможности анализа данных и максимальное количество ответов на опрос.

SurveyMonkey обладает хорошим функционалом и, если купить подписку и привыкнуть к интерфейсу, даёт возможность создавать отличные опросы с настраиваемой логикой переходов.

Следующее приложение выделяется среди остальных своей функциональностью. Online Test Pad (рисунок 1.4) [6] позволяет создавать не только опросы или тесты, а так же кроссворды и логические игры.

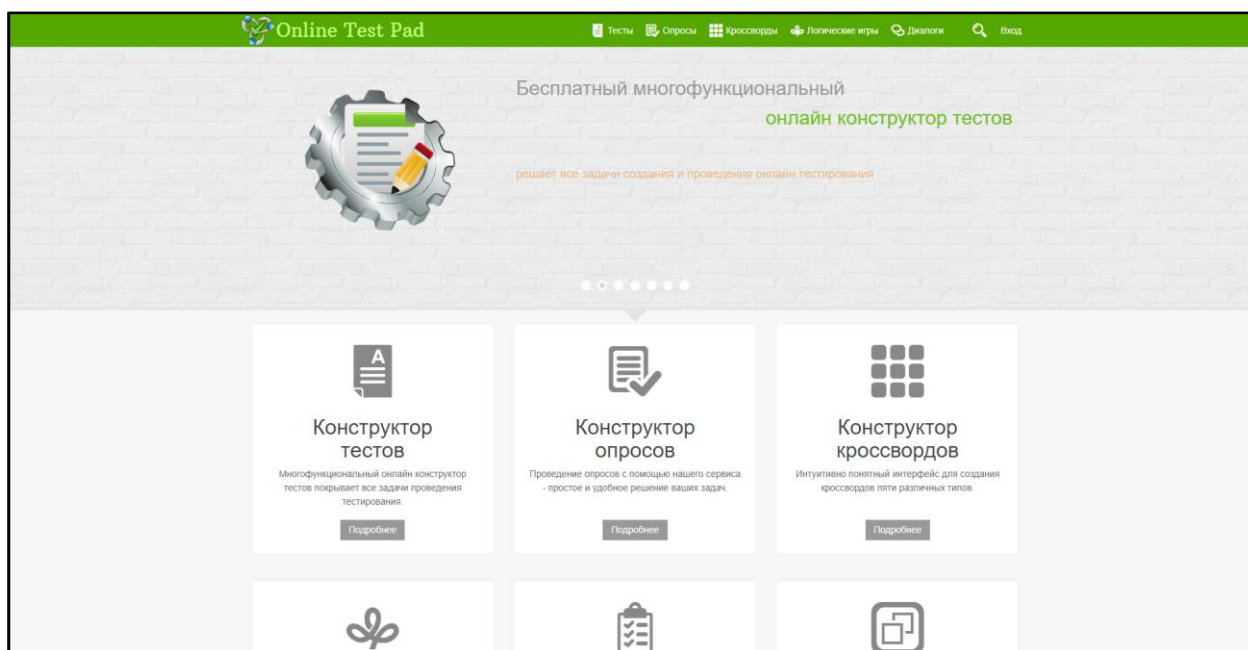


Рисунок 1.4 – Приложение создания и проведения опросов Online Test Pad

Сайт адаптирован для любых гаджетов, поэтому удобно работать и с мобильной версией. Софт предусматривает создание небольшого сайта, специально для тестирования своей аудитории. Редактирование анкет происходит за счёт удобного и простого конструктора.

Сервис предлагает на выбор 10 типов вопросов:

- один выбор;
- текстовый;
- выбор из списка;
- и др.

Удобный инструмент статистики позволяет анализировать каждый результат в формате .excel. Чтобы предоставить доступ к опросу, можно поделиться ссылкой, сделать виджет для сайта или опубликовать в социальных сетях. Предлагаются расширенные настройки для опроса: можно установить постраничное отображение каждого вопроса. Предварительный просмотр позволяет оценить интерфейс созданной анкеты. Кроме того, можно добавлять изображения и формулы. Статистика ответов предоставляется в виде графики или таблицы.

Преимущества:

- удобный конструктор для создания опросов;
- ограничения прохождения по IP;
- статистика в виде графиков или таблицы;
- дополнительные настройки для показа опроса.

Недостатки:

- нет возможности одновременно проводить опрос на разных языках.

Online Test Pad удобный конструктор опросов с минималистичным дизайном и широким функционалом.

Так же стоит обратить внимание на такое приложение как Anketolog.ru (рисунок 1.5) [7].

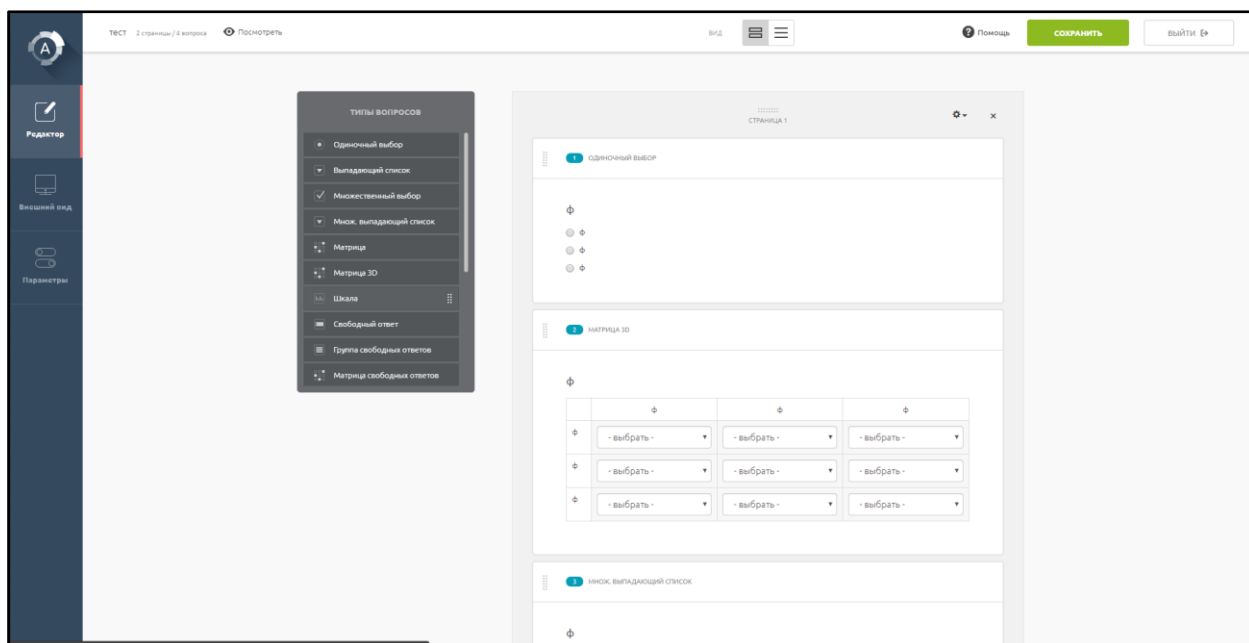


Рисунок 1.5 – Приложение создания и проведения опросов Anketolog.ru

Платформа для создания опросов с помощью расширенного конструктора, сбора ответов и последующего их экспорта в различных форматах. Сайт адаптируется под любые гаджеты: удобно как редактировать, так и отвечать на вопросы. Предусмотрена панель респондентов для компаний, у которых нет базы для опроса. Это очень удобно, если нужно провести исследование по новому товару.

Для создания анкеты доступно на выбор 14 типов вопросов. Чтобы посмотреть на результаты определенной целевой группы, можно применить фильтр для ответов. Предлагаются возможности для брендинга опросов: добавление логотипа, фона и изображений вашей компании. Экспортировать данные можно в формате .pdf, .word, .excel, .spss.

Стоит отметить, что при первом посещении редактора опросов пользователь проходит краткое обучение по интерфейсу, что значительно упрощает освоение работы с ним. Так же в настройках опроса можно задать приветственное сообщение, сообщение для успешного завершения опроса и сообщение для случая, когда анкета дисквалифицирована из-за недостаточной достоверности ответов.

Преимущества:

- удобные и красивые отчёт и аналитика;
- отличный конструктор анкет с обучением;
- фильтрация полученных данных во время прохождения и просмотра статистики ответов;
- расширенные настройки опроса.

Недостатки:

- нет возможности одновременно проводить опрос на разных языках;
- у бесплатного тарифа ограничение - 3 анкеты на 10 вопросов.

Anketolog.ru хорошее приложение для создания опросов с удобным конструктором и возможностью посмотреть, как будет выглядеть опрос на устройствах различных размеров.

Ещё одним прекрасным приложением создания и проведения опросов является Surveygizmo (рисунок 1.6) [8].

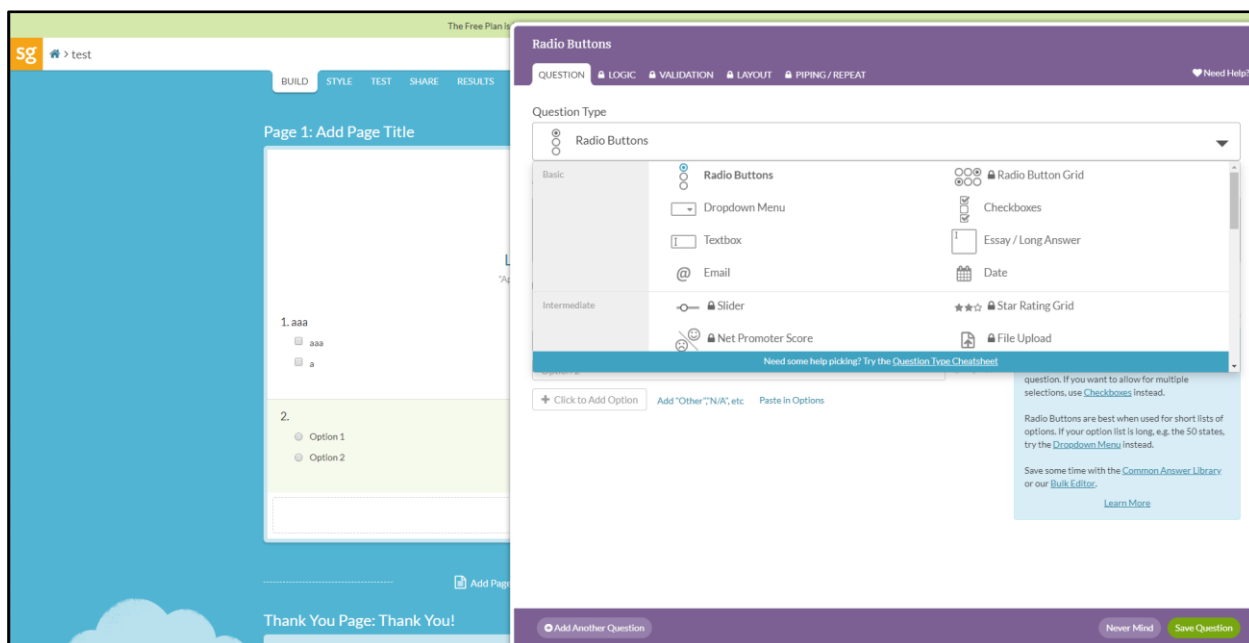


Рисунок 1.6 – Приложение создания и проведения опросов Surveygizmo

Функциональная платформа для создания опросов с помощью конструктора и шаблонов. Доступна возможность получить детализацию данных, которые можно будет отправить коллегам или клиентам. Сервис предусматривает управление одной формой несколькими пользователями, при этом можно выбрать разные роли и права для каждого.

Сервис поддерживает интеграцию с MailChimp и ExactTarget. После окончания редактирования анкеты будет доступен html-код для вставки на ваш сайт или в приложение. Можно пригласить респондентов по электронной почте. Платформа поддерживает несколько языков. Пользователь получает как стандартные отчёты, так и расширенные, например, с кросс-вкладками или сравнением.

Преимущества:

- более 28 типов вопросов;
- уведомления по электронной почте;
- формы для платежей;
- возможность одновременно проводить опрос на разных языках при наличии платной подписки;

- интуитивно понятный дизайн;
- работа в команде.

Недостатки:

- нельзя добавить медиафайлы в опрос;
- практически весь функционал доступен только по подписке.

Surveygizmo отличный конструктор опросов с интересным дизайном и интерфейсом. Однако при бесплатном пользовании функционала хватит лишь на самые простенькие опросы.

Так же стоит обратить внимание на такое приложение как Testograf (рисунок 1.7) [9].

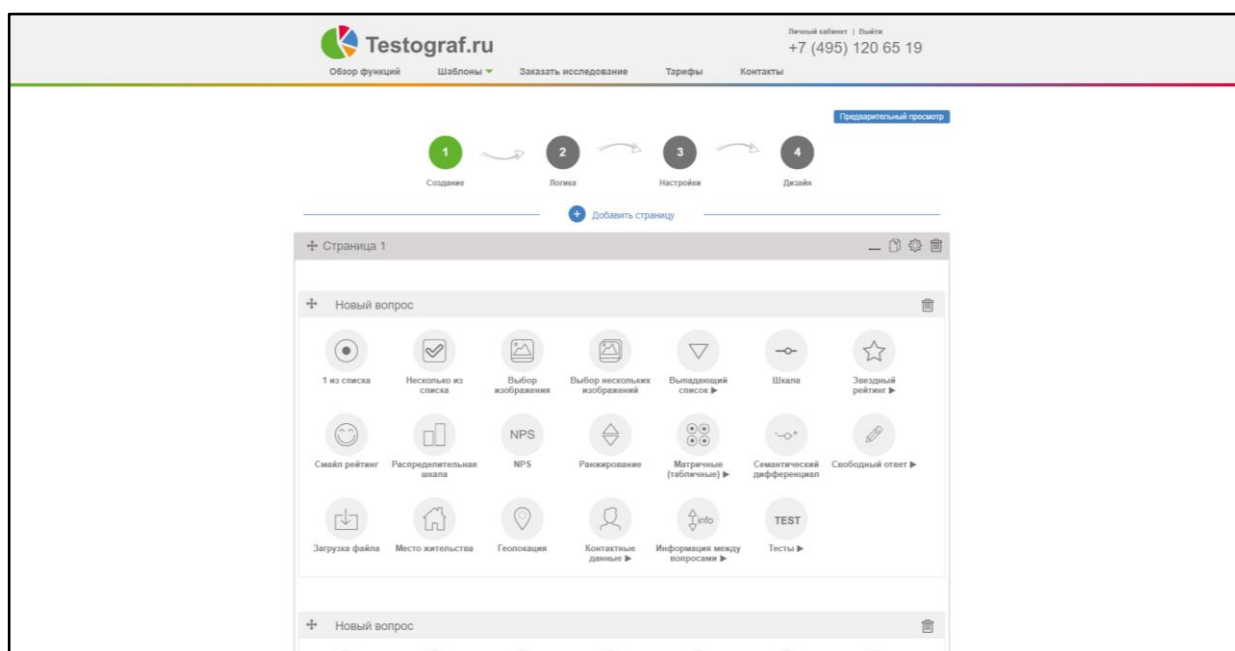


Рисунок 1.7 – Приложение создания и проведения опросов Testograf

Testograf - бесплатный веб-сервис, позволяющий создать форму, опрос или анкету без ограничений по количеству вопросов и опрашиваемых респондентов. Функции системы бесплатны, поэтому она подойдёт небольшим организациям, стартапам, фрилансерам, начинающим специалистам или для личного пользования. Профессионалы смогут приобрести тариф PRO, в котором предусмотрены заключение договора, право собственности на результаты и дополнительные улучшенные опции софта. Программа заинтересует маркетологов, преподавателей, специалистов в областях консалтинга, рекламы и социальных проектов.

Благодаря функционалу Testograf можно проводить исследование рынка, опросы целевой аудитории и сотрудников, создавать формы обратной связи для посетителей сайтов. Кроме того, можно заказать исследование «под ключ» на сайте сервиса. Софт предусматривает библиотеку шаблонов, возможности копирования форм и доступ к некоторым из них по паролю.

На платном тарифе доступны:

- функция таймера;
- брендинг;
- подключение геолокации и галочки-согласия;
- переименование кнопок и оповещений;
- устранение логотипа сервиса.

Вопросы можно снабдить подсказками и комментариями. Платформа предоставляет статистику тестов с проверкой правильности заполнения; администратор может подключить уведомления о новых заполненных опросах на email, а также выгрузить их в форматах .CSV, .XLS, .XLSX, .PDF и .DOC либо архивом .ZIP.

Преимущества:

- удобный конструктор опросов;
- большое количество типов опросов;
- фильтрация анкет на стадии прохождения опроса.

Недостатки:

- нет возможности одновременно проводить опрос на разных языках;
- сохранение ответов и доступ к результатам доступны только после оплаты лицензии.

Testograf отличный сервис по созданию опросов, однако совершенно непригодный для использования при использовании бесплатной лицензии.

Ещё одним прекрасным приложением создания и проведения опросов является Survio (рисунок 1.8) [10].

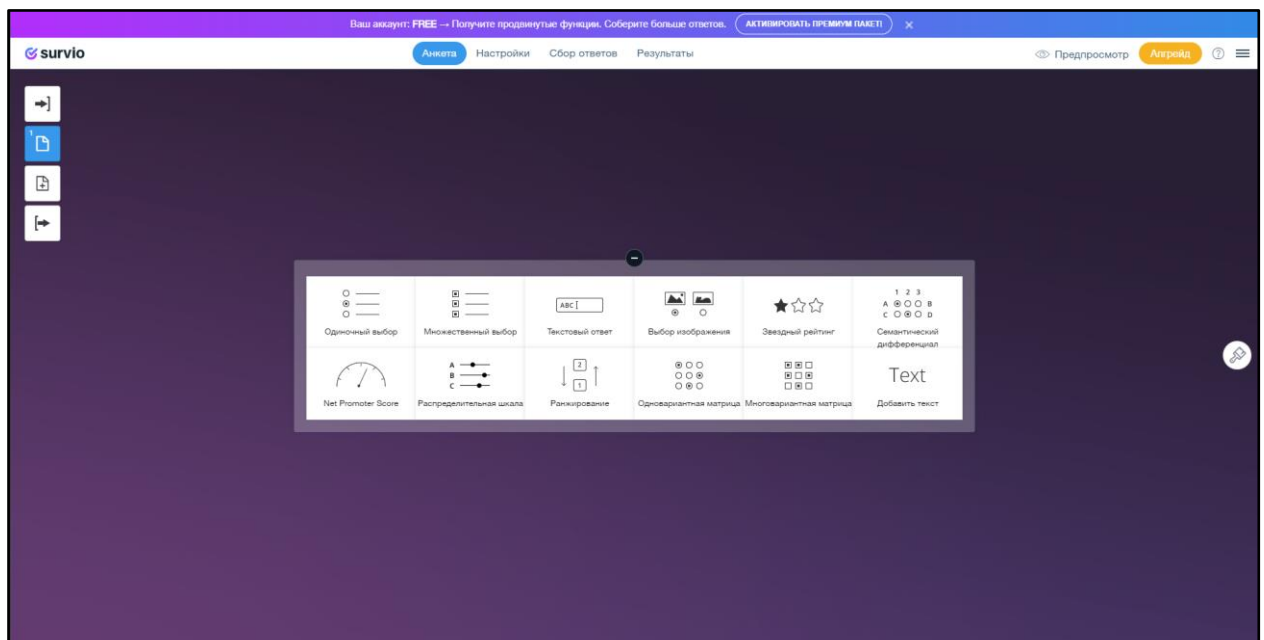


Рисунок 1.8 – Приложение создания и проведения опросов Survio

Это чешский стартап, официально запущенный в 2012 году. На сегодня на его базе уже создано более 1,5 млн. анкет. Предлагает более 100 готовых шаблонов для быстрого создания форм. Поделиться готовой анкетой можно с помощью ссылки или через почту. Предусмотрены бесплатный и платный тарифы. Последний включает в себя дополнительные функции.

Инструменты Survio предлагают создавать 17 типов вопросов. Готовый дизайн адаптивен для мобильных устройств, за счёт чего удобно заполнять анкеты не только с ПК, но и с гаджетов. В настройках доступно копирование уже готовых опросов, подсказки, нумерация, экспорт, брендинг.

Преимущества:

- создание приглашений;
- готовые шаблоны;
- импорт переписки из Gmail, Yahoo;
- брендинг.

Недостатки:

- нет возможности одновременно проводить опрос на разных языках;
- отчётные таблицы и графики оставляют желать лучшего.

Также стоит обратить внимание на такое приложение как Simpoll (рисунок 1.9) [5].

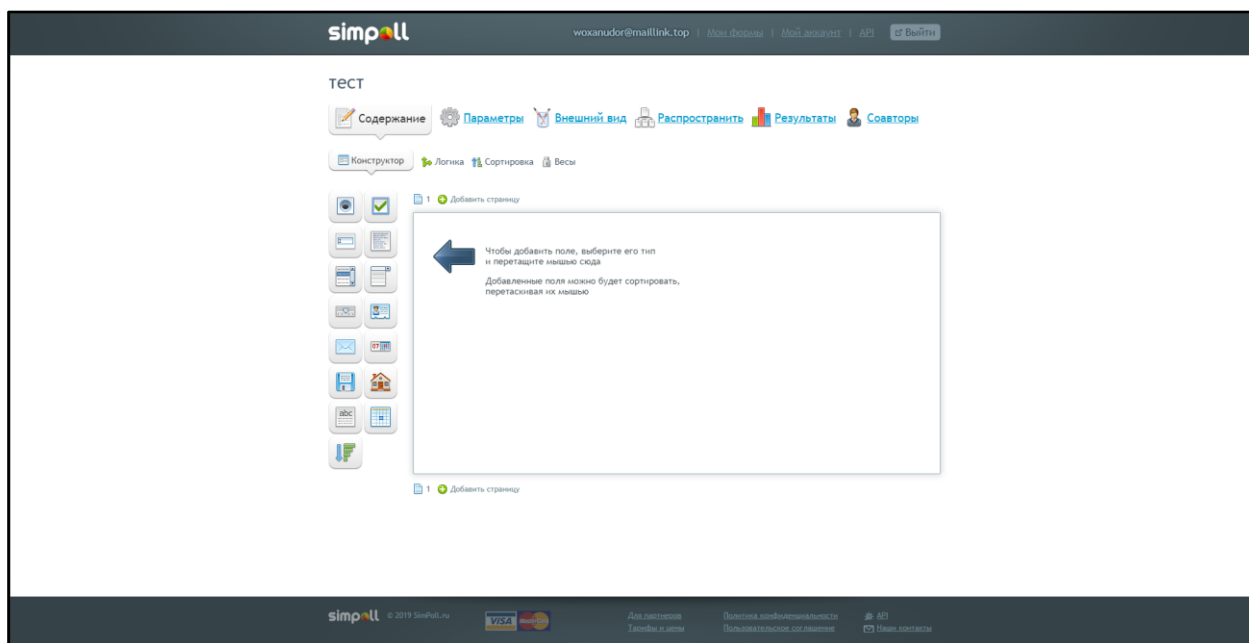


Рисунок 1.9 – Приложение создания и проведения опросов Simpoll

Онлайн-конструктор для опросов и форм обратной связи. Предусматривает возможности для создания анкеты, голосования и тестирования. После редактирования своей формы вы получаете html-код, который можно залить на свой сайт или опубликовать опросы в соцсетях, пригласить пользователей по прямой ссылке.

Можно создавать опрос с развилкой: в таком случае, респондент будет попадать на разные страницы. Доступна возможность устанавливать ограничения на заполнения форм по дате или количеству ответчиков. Чтобы обезопасить себя от любителей повторного прохождения, предусмотрена защита по IP и Cookies. Для каждого элемента можно подобрать свой цвет. Так, опрос будет гармонично смотреться на вашем сайте. Можно добавлять комментарии и оценку ответов, интегрировать календарь. Сервис позволяет добавлять видео и фото в анкеты. Расширенную оценку ответов получаем за счёт фильтрации данных, например, по возрасту или географии.

Преимущества:

- переадресация анкеты;
- брендинг;
- фильтры полученных данных.

Недостатки:

- нет возможности одновременно проводить опрос на разных языках;
- интерфейс сложен в освоении.

Simpoll хороший и функциональный конструктор опросов, но интерфейс достаточно сложный в освоении, что может помешать пользователям использовать данный сервис.

Приложение Opros.by (рисунок 1.10) [12] выделяется среди своих конкурентов тем, что за прохождение опроса там можно получить денежное вознаграждение.

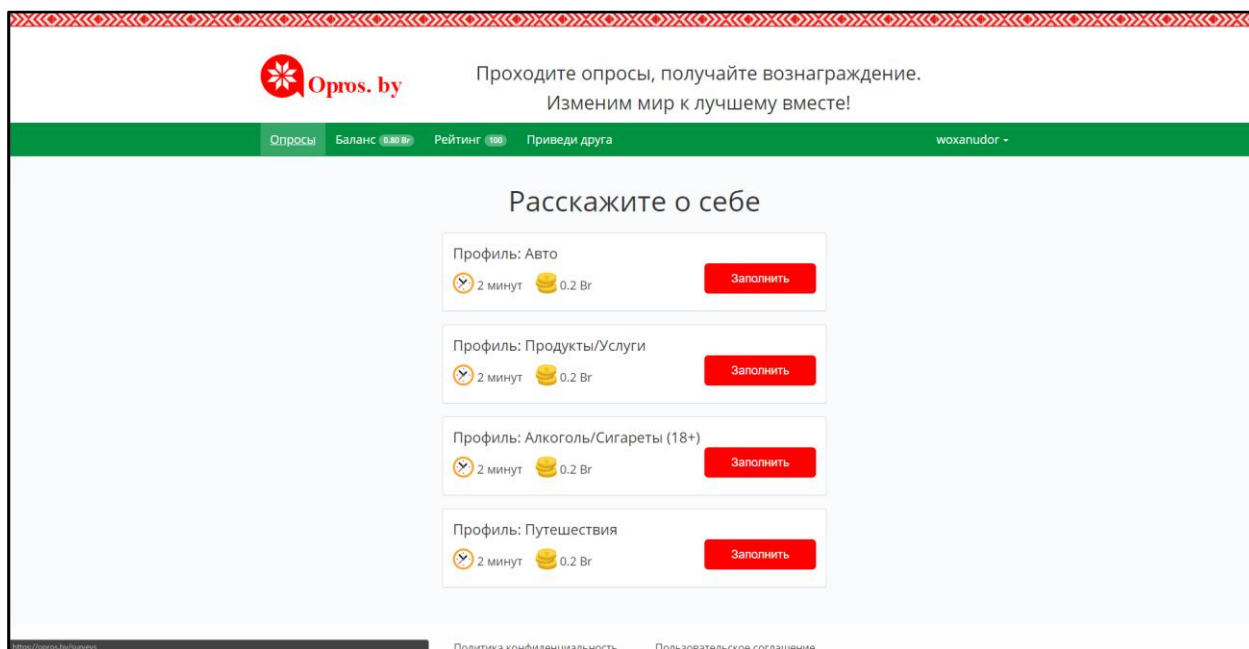


Рисунок 1.10 – Приложение проведения опросов Opros.by

«Opros.by» — это полезный проект для тех, кто хочет поделиться своим опытом использования различных товаров, оценить целесообразность новых

услуг, проверить эффективность рекламной кампании и многое другое. После регистрации пользователи могут проходить предложенные им опросы, получая за это вознаграждение. Однако в приложении отсутствует возможность создания опросов пользователями. Выяснить каким образом можно разместить в этом приложении опрос не представилось возможным.

Преимущества:

- вознаграждение за прохождение опроса;
- понятный и минималистичный дизайн.

Недостатки:

- нет возможности разместить свой опрос.

Opros.by однозначно выделяется среди всех рассмотренных аналогов тем, что за прохождение опросов пользователя начисляется вознаграждение. Однако невозможность разместить свой опрос делает невозможным его использование для сбора данных всем, кроме разработчиков приложения.

1.3 Формирование технического задания приложения

Подводя итоги всего сказанного, можно сформировать техническое задание для приложения, разрабатываемого в данном проекте.

Данное приложение предназначено для упрощения создания и проведения онлайн-опросов.

Основными целями создания данного приложения являются:

- 1) создание онлайн-опросов без каких-либо специальных технических знаний;
- 2) снизить трудоёмкость и сократить время создания онлайн-опросов;
- 3) достижение кроссбраузерности созданных онлайн-опросов.

1.3.1 Создание онлайн-опроса

При создании онлайн-опроса опрос должен появиться в разделе «Мои опросы» в приложении и быть доступным для прохождения в указанные создателем опроса сроки.

Опрос должен состоять из следующих типов вопросов:

- развёрнутый текстовый;
- краткий текстовый;
- множественный выбор;
- одиночный выбор;
- выбор из списка;
- рейтинг;
- логический;
- матрица с одиночным выбором;
- выбор даты.

Опрос должен иметь следующие настройки:

- заголовок;
- язык по умолчанию;

- показать номера страниц;
- текст кнопки «Далее» для основного языка;
- текст кнопки «Назад» для основного языка;
- текст кнопки «Старт» для основного языка;
- текст кнопки «Отправить» для основного языка;
- позиция кнопок навигации;
- показать кнопку «Назад»;
- сделать первую страницу стартовой;
- показать страницу завершения опроса;
- включить автопереход на следующую страницу при заполнении текущей;
- позиция шкалы прогресса;
- одностраничный опрос;
- позиция заголовка вопроса;
- текст обязательности вопроса;
- показать номера вопросов;
- позиция сообщения с ошибкой;
- порядок размещения вопросов на странице;
- html код, показываемый по окончании опроса;
- максимальное время для ответа на опрос;
- максимальное время на заполнение страницы опроса;
- позиция таймера;
- отображаемый таймер;
- триггеры опроса.

Пользователь должен иметь возможность проводить опрос одновременно на нескольких из следующих языков:

- русский;
- английский;
- болгарский;
- японский;
- испанский;
- французский;
- китайский упрощённый;
- китайский традиционный;
- и другие.

1.3.2 Прохождение онлайн-опроса

При прохождении онлайн опроса все вопросы должны быть корректно отображены с учётом всех настроек видимости и условиями, установленными создателем опроса. Должны корректно примениться все настройки опроса, установленные создателем опроса. При отправке опроса данные должны быть корректно сохранены на сервере и учитываться в статистике опроса.

1.3.3 Редактирование ранее созданного онлайн-опроса

При редактировании опроса все вопросы и настройки должны быть корректно загружены и отображены. После редактирования опросы должны быть сохранены новая версия опроса. Старая версия опроса так же должна остаться в системе для сохранения статистики.

1.3.4 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) использование шаблона проектирования MVC – разделение бизнес логики, обработки и представления данных;
- 4) безопасность;
- 5) надёжность.

Требования к техническому обеспечению серверной части учитываются хостингом, на котором будет размещено разрабатываемое приложение. Должна быть обеспечена круглосуточная работа приложения.

Требования к техническому обеспечению клиентской части системы:

- 1) Pentium 2000 МГц или более быстродействующий процессор;
- 2) оперативная память 1Гбайт или более;
- 3) SVGA монитор или монитор с более высоким разрешением;
- 4) устройства для ввода данных, такие как мышь и клавиатура.

Требования к программному обеспечению серверной части:

- 1) .Net Framework 4.5 или выше;
- 2) MSSQL сервер;
- 3) SMTP для работы с почтой.

Все что нужно иметь пользователю – это современный браузер, который поддерживает HTML5.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Описание функциональности приложения

Построение приложения подразумевает проектирование его функциональности. Функциональность данного приложения зависит от роли пользователя.

Данное приложение предусматривает использование базы данных, которая необходима для регистрации пользователей. Регистрация осуществляется для того, чтобы пользователь мог создавать опросы и иметь возможность отвечать на них.

Чтобы пользователи могли регистрироваться необходимы специальные таблицы в базе данных, куда будут заноситься данные о зарегистрированном пользователе. Для полноты понимания процесса регистрации пользователей создадим диаграмму таблиц (рисунок 2.1).

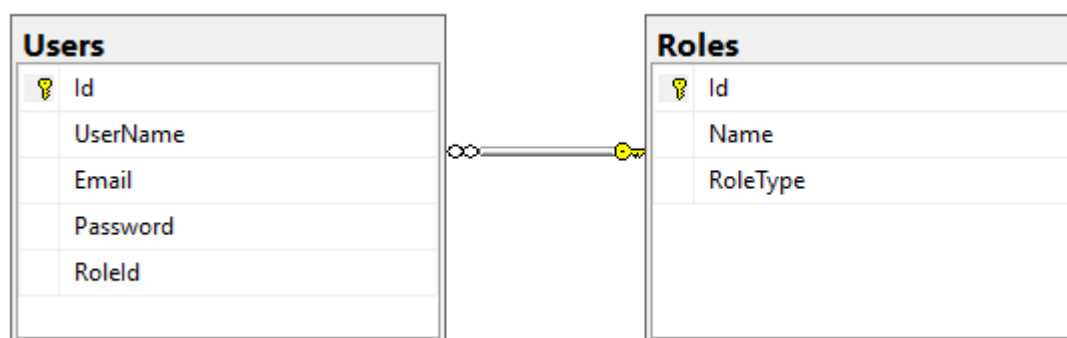


Рисунок 2.1 – Диаграмма таблиц регистрации пользователя

На данной диаграмме видны две сущности. Сущность Пользователи представляет собой каждого пользователя, а сущность Роли представляет все возможные роли пользователя (рисунок 2.2).

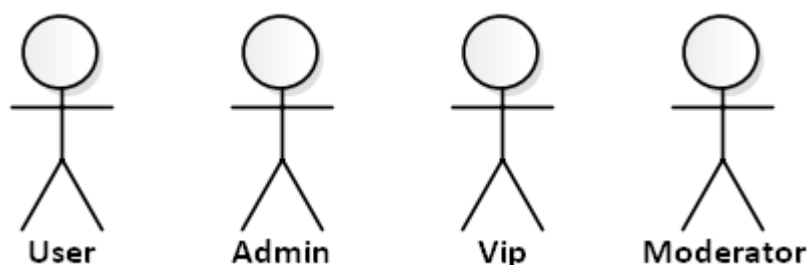


Рисунок 2.2 – Все возможные роли пользователя

Когда пользователь зарегистрировался, он имеет роль «User». Такой пользователь не имеет доступа к конструктору онлайн-опросов и может их лишь проходить. Чтобы пользоваться визуальным конструктором онлайн-опросов пользователь должен иметь роль «Admin» или «Vip». Пользователь «User» может выполнять всё то, что указано на рисунке 2.3.

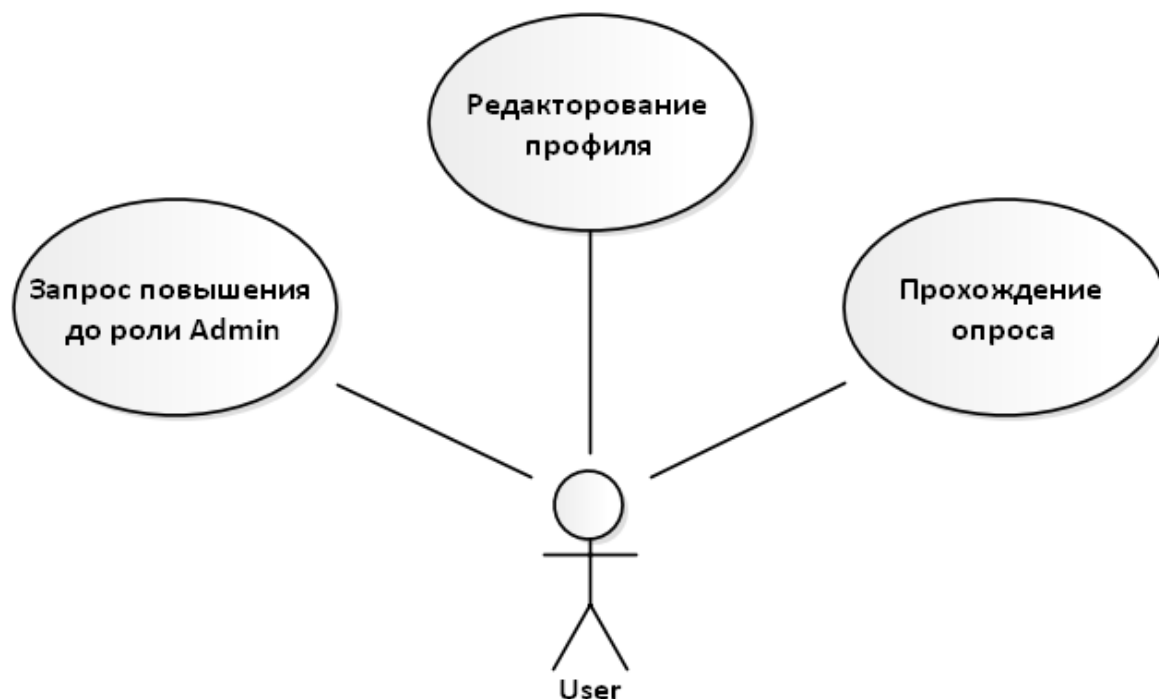


Рисунок 2.3 – Диаграмма вариантов использования приложения в роли «User»

После рассмотрения заявки и повышения до роли «Admin» пользователь получает дополнительные права. Все действия, которые может выполнять пользователь с ролью «Admin» указаны на рисунке 2.4.

Роль «Vip» становится доступной после покупки премиум-пакета. Она позволяет использовать все возможности роли «Admin», а также дополнительные возможности.

На сайте должна быть предусмотрена административная часть. К ней имеют доступ пользователи с ролью «Moderator». Модератору доступна функциональность, которая представлена на рисунке 2.5.

Основными компонентами визуального конструктора онлайн-опросов являются вопросы. Для данного приложения предусмотрены шаблоны следующих элементов

- краткий тестовый вопрос;
- развёрнутый текстовый вопрос;
- одиночный выбор;
- множественный выбор;
- рейтинг;
- выбор из списка;



Рисунок 2.4 – Диаграмма вариантов использования приложения в роли «Admin»

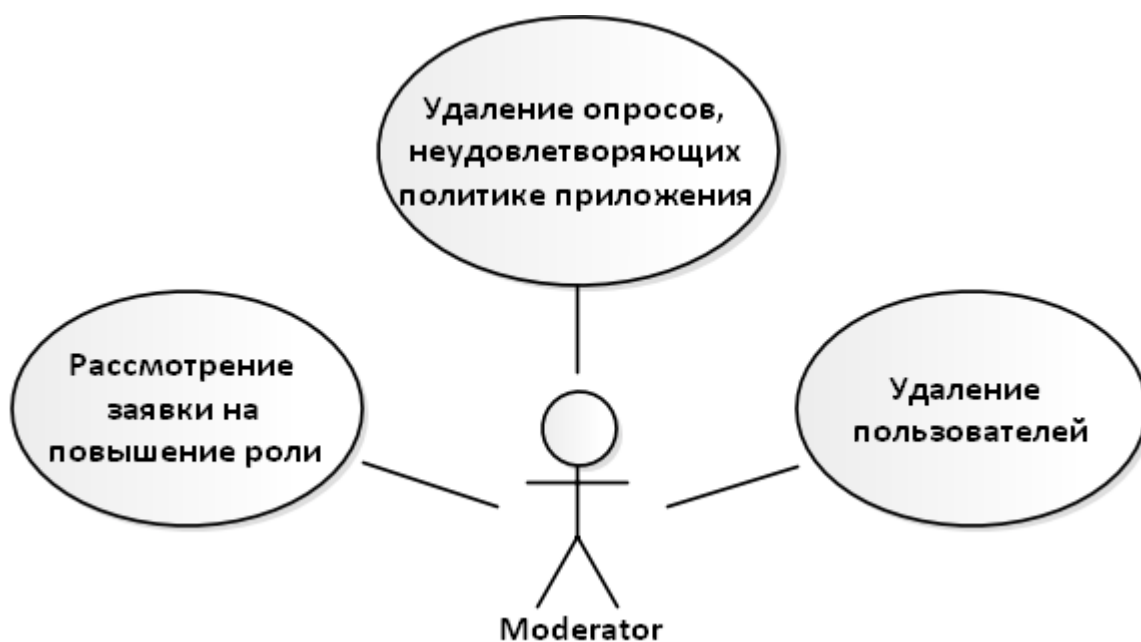


Рисунок 2.5 – Диаграмма вариантов использования приложения в роли «Moderator»

- логический;
- дата;
- матрица одиночных выборов.

В базе данных предусматривается общая таблица, которая будет хранить следующие поля:

- уникальный идентификатор;
- тип;
- имя
- заголовок
- условие отображения
- условие активности;
- обязательность.

Так же предусмотрены таблицы, хранящие дополнительную уникальную информацию для каждого типа вопроса. Для большего понимания приведена схема существующих в данном приложении сущностей шаблонов элементов (рисунок 2.6).

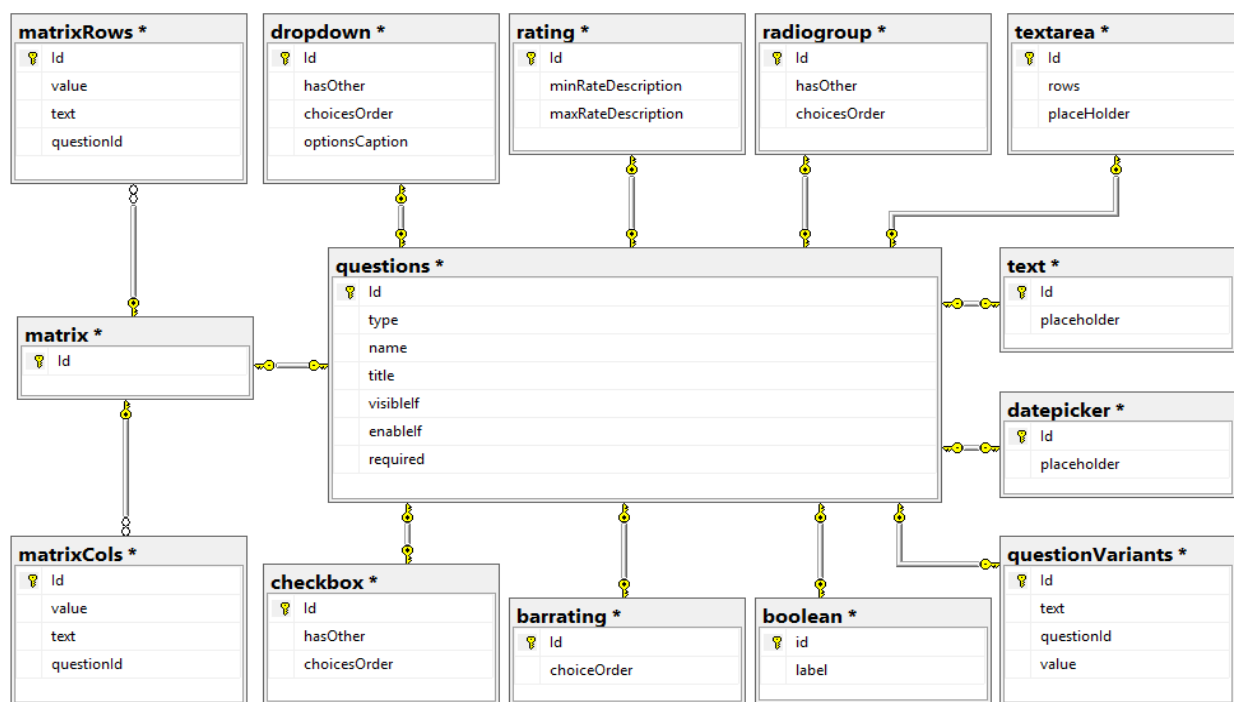


Рисунок 2.6 – Сущности шаблонов добавляемых вопросов

2.2 Спецификация функциональных требований

Основной задачей приложения является создание опросов. В состав данной задачи включается задача добавления различных типов вопросов и их настройка. Рассмотрим добавление каждого из типов вопросов подробнее.

2.2.1 Краткий текстовый вопрос

Краткий текстовый вопрос представляет из себя поле с одной строкой для ввода ответа. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Подтип вопроса. Пользователь должен иметь возможность выбрать один из следующих подтипов вопроса: цвет, текст, дата, дата со временем, время, неделя, месяц, email, число, пароль.

При выборе определённого типа поле должно соответствующим образом валидироваться при прохождении опроса.

7 Текст-заместитель. Пользователь должен иметь возможность задать текст, который появляется в поле, когда оно не заполнено, если подтип вопроса позволяет это.

8 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

9 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.2 Развёрнутый текстовый вопрос

Развёрнутый текстовый вопрос представляет из себя поле с несколькими строками для ввода ответа. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Количество строк. Пользователь должен иметь возможность задать цифру, которая будет отвечать за то, сколько строк будут одновременно видимы в поле. Ответ на вопрос может занимать как больше строк, так и меньше.

7 Текст-заместитель. Пользователь должен иметь возможность задать текст, который появляется в поле, когда оно не заполнено, если подтип вопроса позволяет это.

8 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

9 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.3 Вопрос с одиночным выбором

Вопрос с одиночным выбором представляет из себя несколько радиокнопок, помеченных надписями. В процессе заполнения анкеты в данном

типе вопроса может быть выбран только один из предложенных вариантов. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Порядок расположения вариантов ответов. Пользователь должен иметь возможность задать один из следующих порядков расположения вариантов ответа: по убыванию, по возрастанию, случайный, в порядке добавления. Варианты ответа должны быть отсортированы в зависимости от выбранного порядка.

7 Количество колонок. Пользователь должен иметь возможность задать количество колонок, на которые будут разбиваться варианты ответов. Количество вариантов ответов в каждом столбце должно отличаться максимум на единицу.

8 Варианты ответов. Пользователь должен иметь возможность задать любое количество вариантов ответов на вопрос. Должна быть предусмотрена возможность сделать это двумя способами: с помощью специальной формы и быстрым заполнением (пользователь в текстовом поле указывает варианты ответов каждый с новой строки). Пользователь должен иметь возможность для каждого варианта задать условия видимости и условия активности.

9 Вариант ответа «Другое». Пользователь должен иметь возможность разместить в вопросе вариант ответа «Другое». После выбора этого пункта при прохождении должно появиться текстовое поле, куда респондент впишет собственный вариант ответа.

10 Текст-заместитель для поля «Другое». Пользователь должен иметь возможность задать текст, который появляется в поле «Другое», когда оно не заполнено.

11 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

12 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.4 Вопрос с множественным выбором

Вопрос с множественным выбором представляет из себя несколько флажков, помеченных надписями. В процессе заполнения анкеты в данном типе вопроса может быть выбрано несколько из предложенных вариантов. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Порядок расположения вариантов ответов. Пользователь должен иметь возможность задать один из следующих порядков расположения вариантов ответа: по убыванию, по возрастанию, случайный, в порядке добавления. варианты ответа должны быть отсортированы в зависимости от выбранного порядка.

7 Количество колонок. Пользователь должен иметь возможность задать количество колонок, на которые будут разбиваться варианты ответов. Количество вариантов ответов в каждом столбце должно отличаться максимум на единицу.

8 Варианты ответов. Пользователь должен иметь возможность задать любое количество вариантов ответов на вопрос. Должна быть предусмотрена возможность сделать это двумя способами: с помощью специальной формы и быстрым заполнением (пользователь в текстовом поле указывает варианты ответов каждый с новой строки). Пользователь должен иметь возможность для каждого варианта задать условия видимости и условия активности.

9 Вариант ответа «Другое». Пользователь должен иметь возможность разместить в вопросе вариант ответа «Другое». После выбора этого пункта при прохождении должно появиться текстовое поле, куда респондент впишет собственный вариант ответа.

10 Текст-заместитель для поля «Другое». Пользователь должен иметь возможность задать текст, который появляется в поле «Другое», когда оно не заполнено.

11 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

12 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.5 Рейтинг в виде звёзд

Рейтинг в виде звёзд представляет из себя несколько звёзд. В процессе заполнения анкеты в данном типе вопроса может быть выбрано от одной до максимального количества звёзд. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Порядок расположения вариантов ответов. Пользователь должен иметь возможность задать один из следующих порядков расположения вариантов ответа: по убыванию, по возрастанию, случайный, в порядке добавления. варианты ответа должны быть отсортированы в зависимости от выбранного порядка.

7 Варианты ответов. Пользователь должен иметь возможность задать любое количество вариантов ответов на вопрос. Пользователь должен иметь возможность для каждого варианта задать условия видимости и условия активности.

8 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

9 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.6 Рейтинг в виде значений

Рейтинг в виде значений представляет из себя несколько значений, расположенных друг за другом. В процессе заполнения анкеты в данном типе вопроса может быть выбрано одно из предложенных значений. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Описание минимальной и максимальной границ рейтинга. Пользователь должен иметь возможность задать тексты, которые описывают минимальную и максимальную границы рейтинга. Эти тексты должны располагаться с левой и правой стороны от рейтинга соответственно.

7 Варианты ответов. Пользователь должен иметь возможность задать любое количество вариантов ответов на вопрос.

8 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

9 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.7 Выбор из списка

Выбор из списка представляет из себя поле, в котором отображается список вариантов ответов. В процессе заполнения анкеты в данном типе вопроса может быть выбран только один из предложенных вариантов. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Порядок расположения вариантов ответов. Пользователь должен иметь возможность задать один из следующих порядков расположения вариантов ответа: по убыванию, по возрастанию, случайный, в порядке добавления. варианты ответа должны быть отсортированы в зависимости от выбранного порядка.

7 Текст-заместитель. Пользователь должен иметь возможность задать текст, который появляется в поле, когда оно не заполнено.

8 Варианты ответов. Пользователь должен иметь возможность задать любое количество вариантов ответов на вопрос. Должна быть предусмотрена возможность сделать это двумя способами: с помощью специальной формы и быстрым заполнением (пользователь в текстовом поле указывает варианты

ответов каждый с новой строки). Пользователь должен иметь возможность для каждого варианта задать условия видимости и условия активности.

9 Вариант ответа «Другое». Пользователь должен иметь возможность разместить в вопросе вариант ответа «Другое». После выбора этого пункта при прохождении должно появиться текстовое поле, куда респондент впишет собственный вариант ответа.

10 Текст-заместитель для поля «Другое». Пользователь должен иметь возможность задать текст, который появляется в поле «Другое», когда оно не заполнено.

11 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

12 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.8 Логический вопрос

Логический вопрос представляет из себя флажок, который может находиться в трёх состояниях: ложь, истина, не выбрано. В процессе заполнения анкеты в данном типе вопроса может быть выбрано только одно из состояний. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Метка. Пользователь должен иметь возможность задать один текст, который будет располагаться рядом с флажком при прохождении опроса.

7 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

8 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.9 Выбор даты

Выбор даты представляет из себя поле с одной строкой для ввода даты. При выделении поля должен появиться календарь, для более удобного выбора даты. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность пометить вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Текст-заместитель. Пользователь должен иметь возможность задать текст, который появляется в поле, когда оно не заполнено.

7 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

8 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.10 Матрица одиночных выборов

Матрица одиночных выборов представляет из себя несколько радио-кнопок, размещённых на пересечениях помеченных строк и столбцов. В процессе заполнения анкеты в данном типе вопроса может быть выбран только один из предложенных вариантов в каждой из строк. В настройках этого вопроса можно задать следующие параметры:

1 Короткое имя вопроса. Это текст, который служит для идентификации вопроса в создаваемом опросе. С его помощью возможна настройка логики видимости других вопросов или страниц, а также триггеров опроса.

2 Заголовок вопроса. В данное поле необходимо вписать текст задаваемого вопроса. Пользователь должен иметь возможность задать перевод заголовка на другие языки.

3 Видимость вопроса. Пользователь должен иметь возможность делать вопросы невидимыми для пользователя. Если флажок не установлен напротив этого пункта, то вопрос не должен отображаться при прохождении опроса.

4 Обязательность вопроса. Пользователь должен иметь возможность помечать вопросы обязательными. Если флажок установлен напротив этого пункта, то при прохождении опроса респонденты будут обязаны ответить на этот вопрос, иначе они не смогут перейти на следующую страницу либо завершить заполнение анкеты.

5 Начать вопрос с новой строки. Пользователь должен иметь возможность разместить на одной строке несколько вопросов. Если флажок установлен напротив этого пункта, то вопрос, если он не первый на странице, должен быть расположен на той же строке, что и предыдущий вопрос.

6 Пометки колонок. Пользователь должен иметь возможность задать любое количество колонок. Должна быть предусмотрена возможность сделать это двумя способами: с помощью специальной формы и быстрым заполнением (пользователь в текстовом поле указывает варианты ответов каждый с новой строки). Пользователь должен иметь возможность для каждого варианта задать условия видимости.

8 Пометки строк. Пользователь должен иметь возможность задать любое количество строк. Должна быть предусмотрена возможность сделать это двумя способами: с помощью специальной формы и быстрым заполнением (пользователь в текстовом поле указывает варианты ответов каждый с новой строки). Пользователь должен иметь возможность для каждого варианта задать условия видимости.

9 Условия видимости вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет отображаться при прохождении опроса. Если логическое выражение принимает истинное значение, то вопрос должен отобразиться на странице. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

10 Условия активности вопроса. Пользователь должен иметь возможность задать условия, при которых вопрос будет доступен для заполнения при прохождении опроса. Если логическое выражение принимает истинное

значение, то вопрос должен быть доступен для заполнения. Если логическое выражение синтаксически неверное - оно принимает значение ложь.

2.2.11 Настройка опроса

При создании опроса пользователь должен иметь возможность задать следующие настройки опроса:

- 1 Заголовок. Это текст, который содержит название опроса.
- 2 Основной язык. Пользователь должен иметь возможность выбрать основной язык для проведения опроса, и перевести опрос на другие доступные в приложении языки.
- 3 Показать номера страниц. Пользователь должен иметь возможность отключить показ номеров страниц, оставив при этом только их заголовки.
- 4 Текст кнопок «Далее», «Назад», «Старт» и «Отправить» для основного языка. Пользователь должен иметь возможность задать текст кнопок «Далее», «Назад», «Старт» и «Отправить» для основного языка, и перевести их на другие доступные в приложении языки.
- 5 Позиция кнопок навигации. Пользователь должен иметь возможность задать позицию кнопок навигации одним из следующих значений: сверху, снизу, сверху и снизу вместе, не отображать.
- 6 Показать кнопку «Назад». Пользователь должен иметь возможность настроить показ кнопки «Назад» для включения или отключения возврата респондентов к предыдущим страницам во время прохождения опроса.
- 7 Сделать первую страницу стартовой. Пользователь должен иметь возможность сделать первую страницу стартовой. И разместить на ней любую информацию. Ответы на вопросы с такой страницы не попадут в статистику ответов.
- 8 Показать страницу завершения опроса. Пользователь должен иметь возможность настроить отображение html кода, который будет отображаться в конце прохождения опроса.
- 9 Включить автопереход на следующую страницу при заполнении текущей. Пользователь должен иметь возможность включить автопереход на следующую страницу, когда респондент ответил на все вопросы текущей страницы.
- 10 Позиция шкалы прогресса. Пользователь должен иметь возможность задать положение шкалы прогресса заполнения анкеты в одно из следующих значений: сверху, снизу, сверху и снизу вместе, не отображать.
- 11 Одностраничный опрос. Пользователь должен иметь возможность сделать свой опрос одностраничным. Если флажок у этого пункта установлен, то в конструкторе должна пропасть панель управления страницами.
- 12 Позиция заголовка вопроса. Пользователь должен иметь возможность задать позицию заголовка вопроса в одно из следующих значений: сверху, снизу, слева.
- 13 Текст обязательности вопроса. Пользователь должен иметь возможность задать текст, который будет означать, что вопрос обязателен для заполнения.

14 Показать номера вопросов. Пользователь должен иметь возможность отключить показ номеров вопросов, оставив при этом только их заголовки.

15 Позиция сообщения с ошибкой. Пользователь должен иметь возможность задать позицию сообщения о некорректном заполнении вопроса в одно из следующих значений: сверху, снизу.

16 Порядок размещения вопросов на странице. Пользователь должен иметь возможность выбрать порядок вопросов на странице. Предлагаемые варианты: исходный (порядок, который был при отправке опроса на сервер), случайный.

17 Html код, показываемый по окончании опроса. Пользователь должен иметь возможность задать html код, который будет отображаться в конце опроса, при включённом параметре его отображения.

18 Максимальное время для ответа на опрос. Пользователь должен иметь возможность задать максимальное время в секундах для ответа на весь опрос, чтобы ограничить время ответа респондента.

19 Максимальное время на заполнение страницы опроса. Пользователь должен иметь возможность задать максимальное время в секундах для ответа на все вопросы на странице, чтобы ограничить время ответа респондента.

20 Позиция таймера. Пользователь должен иметь возможность задать положение таймера в одно из следующих значений: сверху, снизу, не отображать.

21 Отображаемый таймер. Пользователь должен иметь возможность выбрать какой таймер отображать респонденту: таймер для страницы, таймер для опроса, оба таймера.

22 Триггеры опроса. Пользователь должен иметь возможность настроить триггеры опроса для создания более динамического опроса.

23 Перевод опроса. Пользователь должен иметь возможность перевести все вопросы и надписи на несколько языков. Респонденты смогут выбирать, на каком языке они будут проходить опрос.

3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

3.1 Разработка архитектуры приложения

Как только были сформулированы основные функциональные требования проектируемого приложения, требуется разработать его архитектуру. Это важное проектное решение, которое обеспечивает набор свойств.

Свойствами разрабатываемой архитектуры являются:

1 **Функциональность.** Данное свойство представляет собой функциональность, предоставляемую пользователям, исходя из требований к приложению.

2 **Гибкость.** Это свойство предоставляет подходящие механизмы для решения разнообразных задач с относительно небольшим объёмом выразительных средств.

3 **Возможность независимого изменения.** Это свойство означает, что приложение имеет изолированные элементы, которые сводят к минимуму количество мест внесения изменений при модификации.

4 **Удобство построения.** Данное свойство управляет правильным и логичным процессом построения приложения, когда набор компонентов приложения может реализовываться и тестироваться независимо друг от друга.

5 **Адаптация к росту.** Это свойство означает, что приложение сможет приспособиться к возможному росту.

6 **Сопротивление энтропии** – поддерживает порядок 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 дополнять возможности друг друга.

8 **Безопасность** – управляет ограничением доступа к своим данным.

Все вышеперечисленные свойства может обеспечить архитектура MVC (рисунок 3.1). 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 данной архитектуре легко сопровождать и дорабатывать, они гибкие и легко масштабируемые. Так же в последнее время популярным становится объединять MVC архитектуру с клиент-серверной, т. е. когда есть разделение на клиентскую часть приложения и серверную. Данный подход 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 должен вызвать ухудшения работы разрабатываемого приложения.

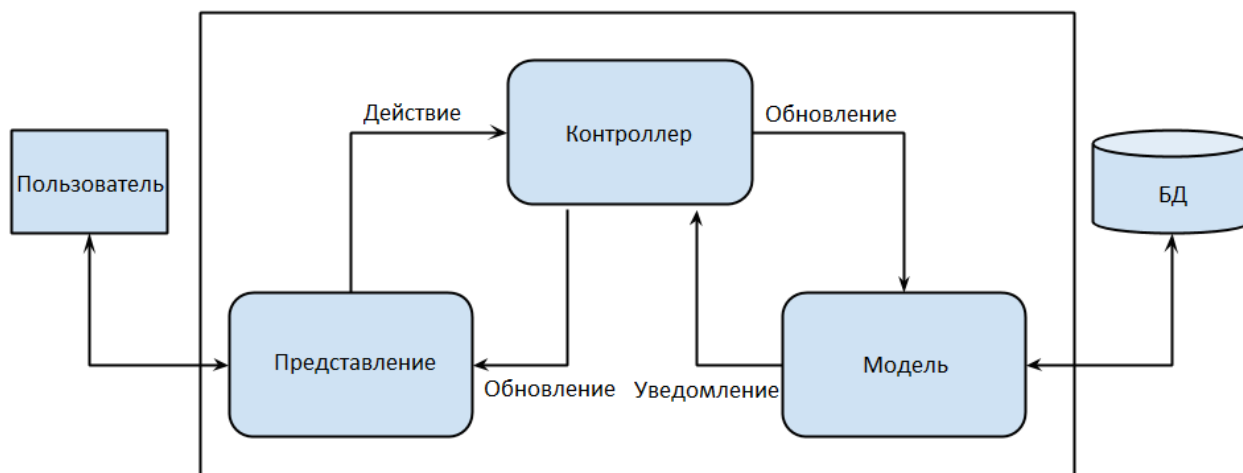


Рисунок 3.1 – Схема работы MVC

Модель, выбранной архитектуры, представляет данные и методы работы с ними. Например, запросы в базу данных или проверки на корректность введенных данных. Таким образом, модель просто представляет доступ к данным и управляет ими.

За отображение данных, которые отдаёт контроллер, несёт ответственность представление. Понятие шаблона тесно с ним связано. Оно позволяет менять внешний вид показываемой информации. В веб-приложениях представление зачастую реализуется в виде HTML-страницы, но иногда может быть представлено данными в виде 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.

Дополнительными плюсам данной архитектуры также являются:

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

Архитектуру MVC на платформе .Net поддерживает такой фреймворк как WebApi 2. Он представляет собой веб-службу, которая может взаимодействовать с различными приложениями. При этом приложение может быть веб-приложением ASP.NET, либо может быть мобильным или обычным десктопным приложением.

Существенным отличием от других фреймворков платформы является то, что используется специальный вид контроллеров - ApiController,

который обладает двумя характеристиками:

- методы действий возвращают объекты моделей;
- методы действий выбираются на основе HTTP-метода, используемого в запросе.

Объекты моделей, возвращаемые методом действия контроллера API, кодируются в формате JSON или XML и отправляются клиенту. Контроллеры API предназначены для доставки данных, поэтому они не поддерживают представления, компоновки или любые другие средства.

Все запросы, которые обрабатываются WebApi 2 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.

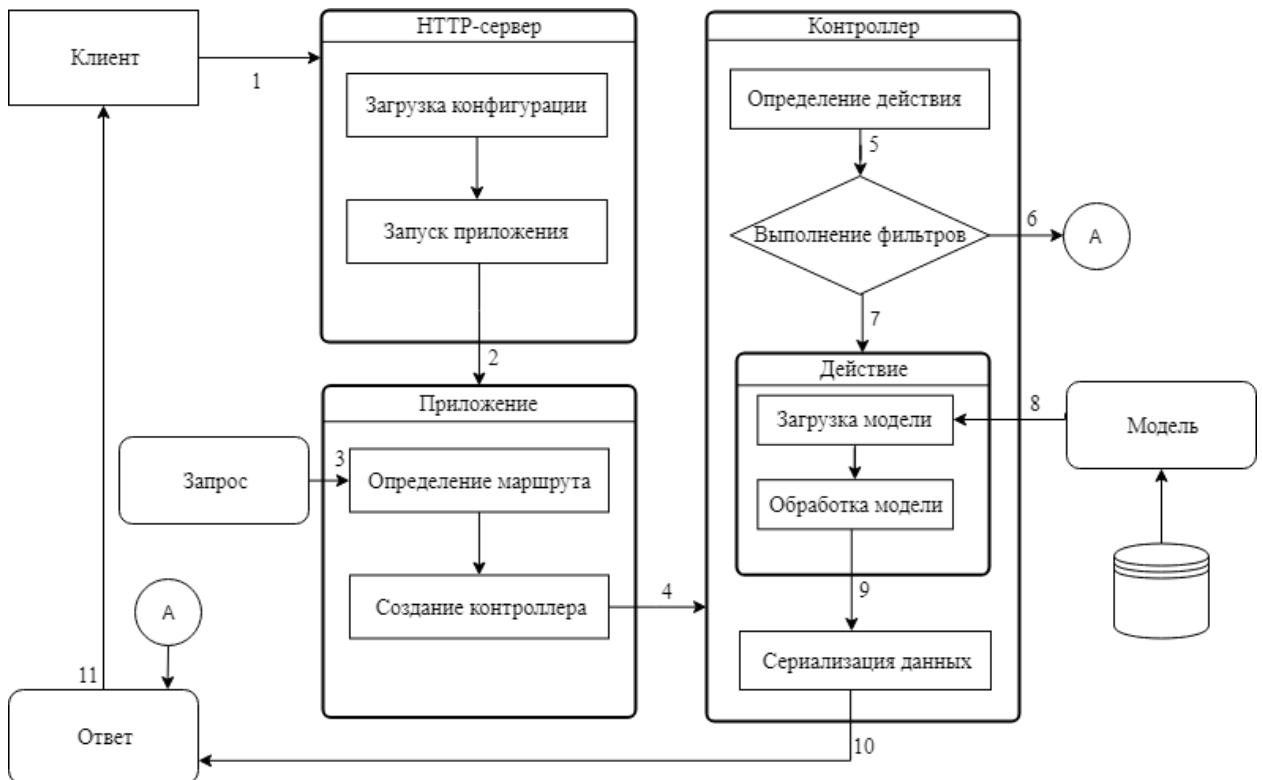


Рисунок 3.2 – Диаграмма обработки запроса серверной частью приложения

Таким образом для разработки серверной части данного приложения очень хорошо подходит WebApi 2 и архитектура MVC совмещённая с клиент-серверной архитектурой. Для разработки будет использоваться шаблон с тремя слоями, так как он облегчает разработку приложения и повышает независимость модулей друг от друга.

3.2 Разработка физической модели базы данных

Сущность – это объект, сведения о котором нужно сохранить в базе данных.

Атрибут – это свойство сущности в предметной области. Его наименование уникально, для конкретной сущности.

Проектируемая база данных имеет сущности и атрибуты. Они представлены в таблице 3.1.

Таблица 3.1 – Сущности и атрибуты базы данных приложения

Сущность	Атрибуты	Значение атрибутов
Users	Id	Уникальный идентификатор
	UserName	Имя
	Email	Электронная почта
	Password	Хэш значение пароля
	RoleId	Уникальный идентификатор роли
Roles	Id	Уникальный идентификатор
	Name	Имя роли
	RoleType	Тип роли
Survey	Id	Уникальный идентификатор
	CreatedDate	Дата создания
	AuthorId	Уникальный идентификатор автора
SurveyVersion	Id	Уникальный идентификатор
	Version	Номер версии
	Title	Заголовок
	CreatedDate	Дата изменения
	SurveyId	Уникальный идентификатор опроса
Page	Id	Уникальный идентификатор
	Title	Заголовок
	Name	Номер
	Visible	Видимость страницы
	VisibleIf	Условие видимости страницы
	QuestionsOrder	Порядок вопросов на странице
SurveyPage	Id	Уникальный идентификатор
	SurveyVersionId	Уникальный идентификатор версии опроса

Продолжение таблицы 3.1

Сущность	Атрибуты	Значение атрибутов
Question	Id	Уникальный идентификатор
	Title	Заголовок вопроса
	Name	Имя вопроса
	IsRequired	Обязательность вопроса
	PageId	Уникальный идентификатор страницы
	QuestionTypeId	Уникальный идентификатор типа вопроса
	EnableIf	Условие активности
	VisibleIf	Условие видимости
	Visible	Видимость вопроса
	StartWithNewLine	Начало с новой строки
QuestionType	Id	Уникальный идентификатор
	Name	Имя типа вопроса
	Type	Тип типа опроса
Datepicker	Id	Уникальный идентификатор
	Placeholder	Текст-заполнитель
Text	Id	Уникальный идентификатор
	Placeholder	Текст-заполнитель
Textarea	Id	Уникальный идентификатор
	Rows	Количество строк
	Placeholder	Текст-заполнитель
Radiogroup	Id	Уникальный идентификатор
	HasOther	Наличие поля «Другое»
	ChoicesOrder	Порядок вариантов ответов
Rating	Id	Уникальный идентификатор
	MinRateDescription	Описание нижней границы
	MaxRateDescription	Описание верхней границы
Dropdown	Id	Уникальный идентификатор
	HasOther	Наличие поля «Другое»
	ChoisesOrder	Порядок вариантов ответов
	OptionsCaption	Текст-заполнитель
Matrix	Id	Уникальный идентификатор
MatrixRow	Id	Уникальный идентификатор
	Text	Заголовок строки
	Value	Значение строки
	MatrixId	Уникальный идентификатор матрицы
MatrxCol	Id	Уникальный идентификатор
	Text	Заголовок колонки
	Value	Значение колонки
	MatrixId	Уникальный идентификатор матрицы

Продолжение таблицы 3.1

Сущность	Атрибуты	Значение атрибутов
Checkbox	Id	Уникальный идентификатор
	HasOther	Наличие поля «Другое»
	ChoicesOrder	Порядок вариантов ответов
Barrating	Id	Уникальный идентификатор
	ChoicesOrder	Порядок вариантов ответов
Boolean	Id	Уникальный идентификатор
	Label	Метка флажка
QuestionVariant	QuestionId	Уникальный идентификатор вопроса
	Value	Значение варианта
	Text	Текст варианта
SurveyTemplate	Id	Уникальный идентификатор
	Title	Заголовок шаблона
	Dedcription	Описание шаблона
TemplatePage	Pageid	Уникальный идентификатор страницы
	SurveyTemplateId	Уникальный идентификатор шаблона опроса
SurveyResponse	Id	Уникальный идентификатор
	PassedDate	Дата прохождения
	SurveyVersionId	Уникальный идентификатор версии опроса
	UsersId	Уникальный идентификатор респондента
QuestionAnswer	Id	Уникальный идентификатор
	Value	Выбранное значение
	SurveyResponseId	Уникальный идентификатор ответа на опрос

Для взаимодействия приложения с базой данных используется Entity Framework. Это объектно-ориентированный фреймворк для доступа и манипулирования данными, хранящимися в базах данных.

3.3 Разработка алгоритма приложения и алгоритмов отдельных модулей

Создавать онлайн-опросы 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».

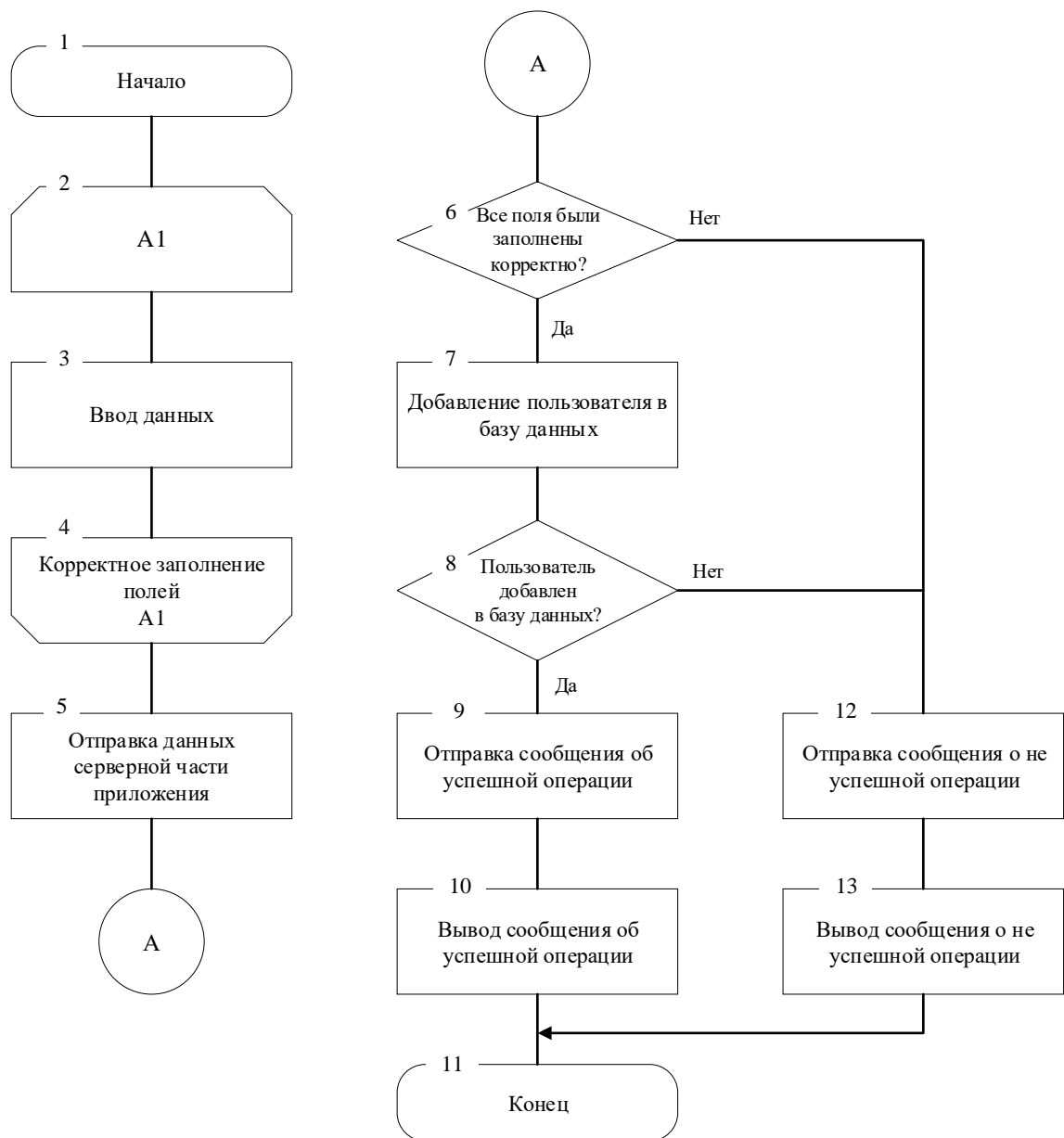


Рисунок 3.3 – Алгоритм регистрации пользователя

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).

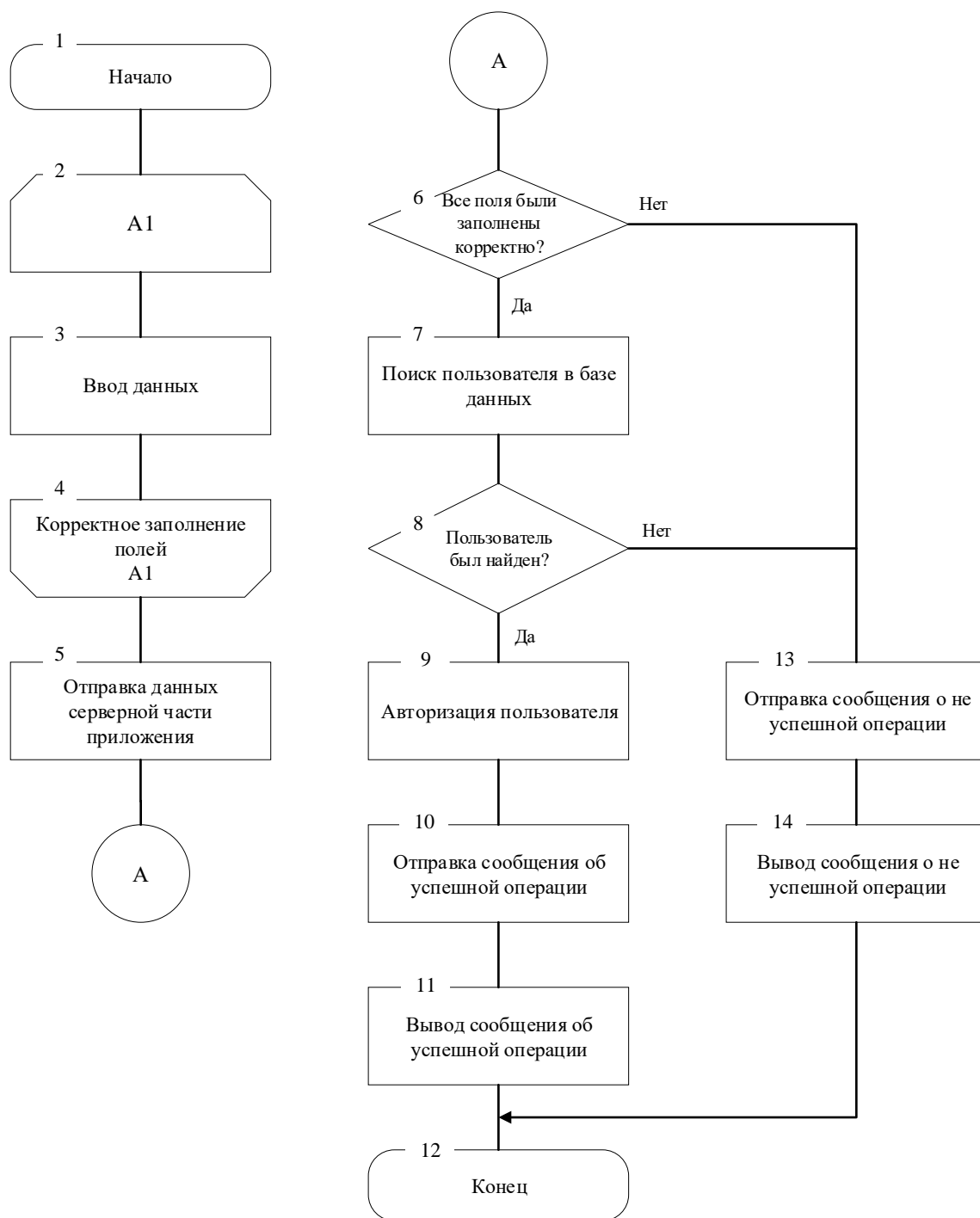


Рисунок 3.4 – Алгоритм авторизации пользователя

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

Пользователь может изменять личные данные. Алгоритм изменения данных пользователя приведён на рисунке 3.5. Чтобы обновить данные пользователь должен изменит данные в полях и нажать на кнопку «Сохранить».

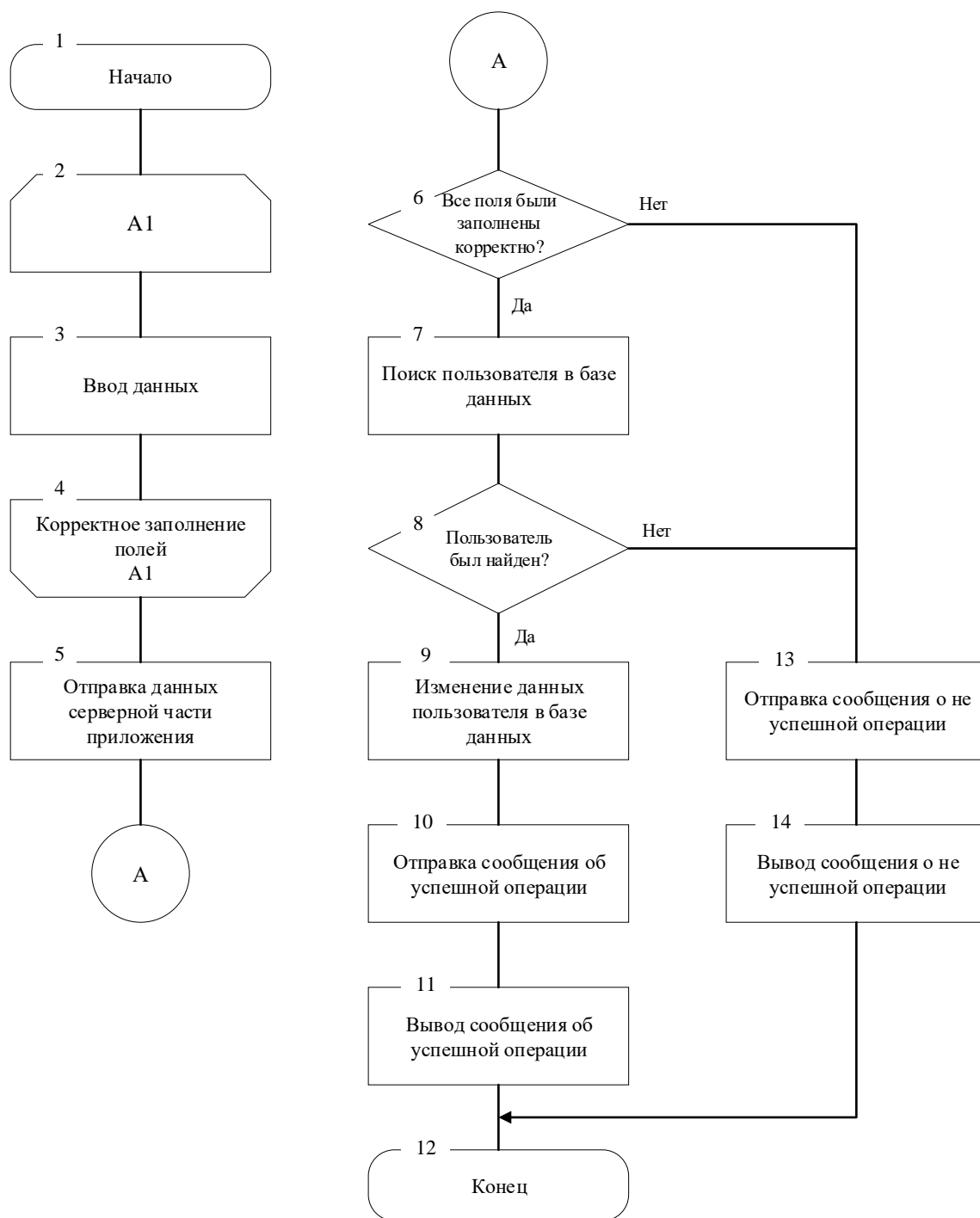


Рисунок 3.5 – Алгоритм обновления профиля пользователя

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

Модератор может удалять пользователей. Для этого в административной части сайта напротив нужного пользователя следует нажать кнопку «Удалить». Алгоритм удаления пользователя приведён на рисунке 3.6.

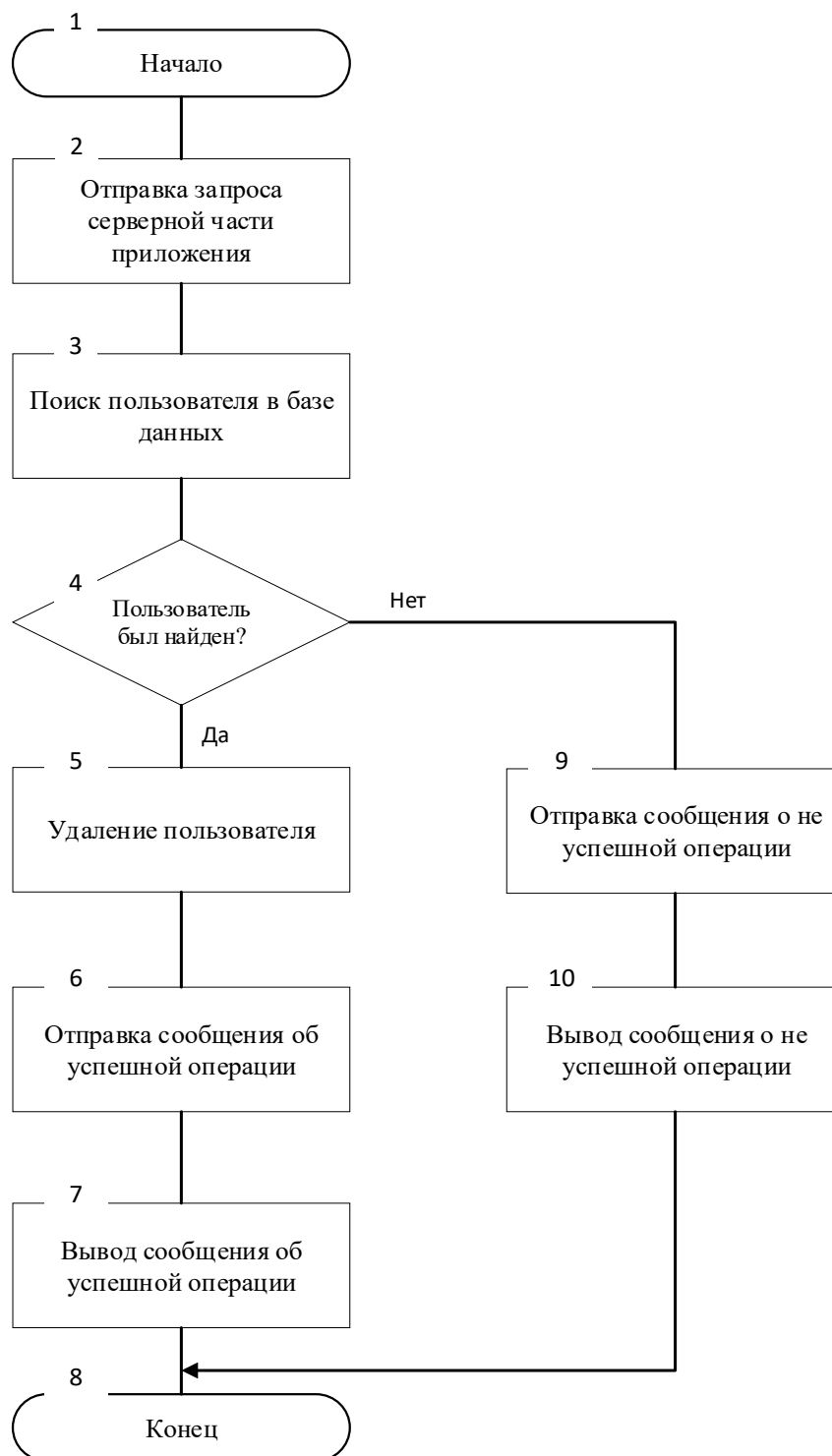


Рисунок 3.6 – Алгоритм удаления пользователя

Если удаление прошло успешно, то пользователь удалится из базы данных и будет выведено сообщение об успешном удалении пользователя.

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.	Отобразилось страница авторизации.
Авторизация пользователя.	1. Заполнить поля для ввода данных: email – строка вида *@*.*, пароль – минимум восемь символов. 2. Нажать на кнопку «Войти».	1. Валидация введённых 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.	Отобразилось 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 введённых 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. Появится панель 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 одностороннего 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 сохранён 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).

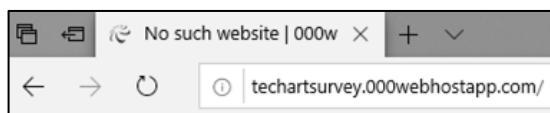


Рисунок 6.1 – Адрес приложения

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

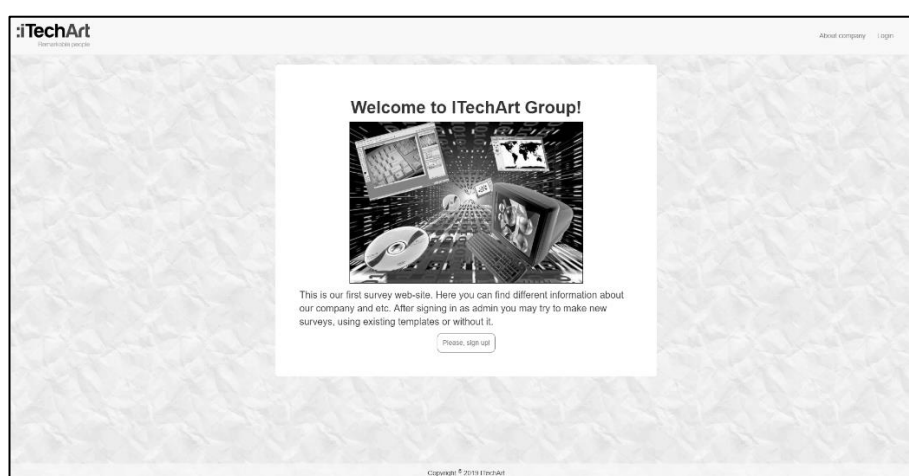


Рисунок 6.2 – Главная страница приложения

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

A screenshot of a 'Sign Up' registration 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).

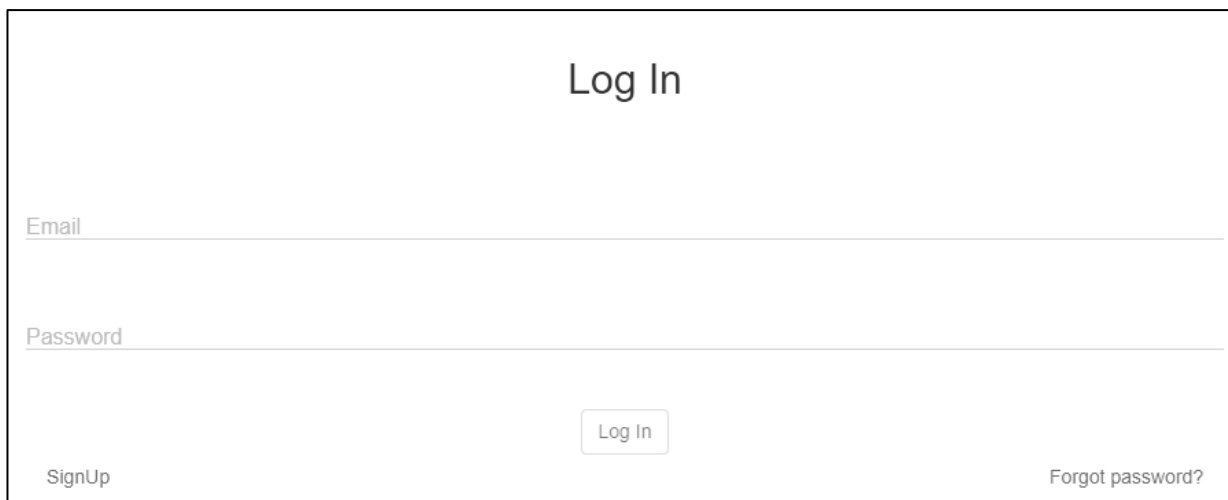
The image shows a login form titled "Log In". It contains two input fields: "Email" and "Password". Below the "Password" field is a "Log In" button. At the bottom left is a "SignUp" link, and at the bottom right is a "Forgot password?" link.

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).

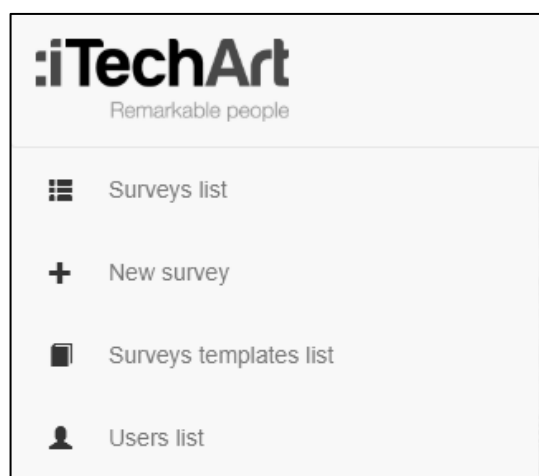


Рисунок 6.5 – Панель с инструментами для роли администратор

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).

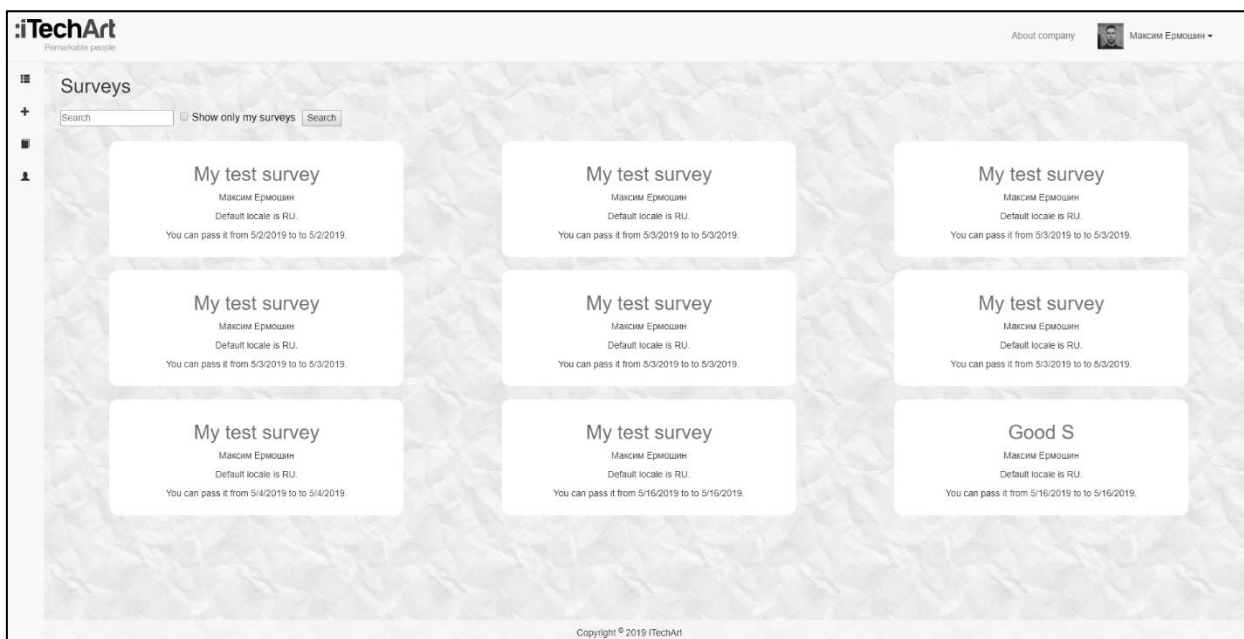


Рисунок 6.6 – Страница со списком опросов

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

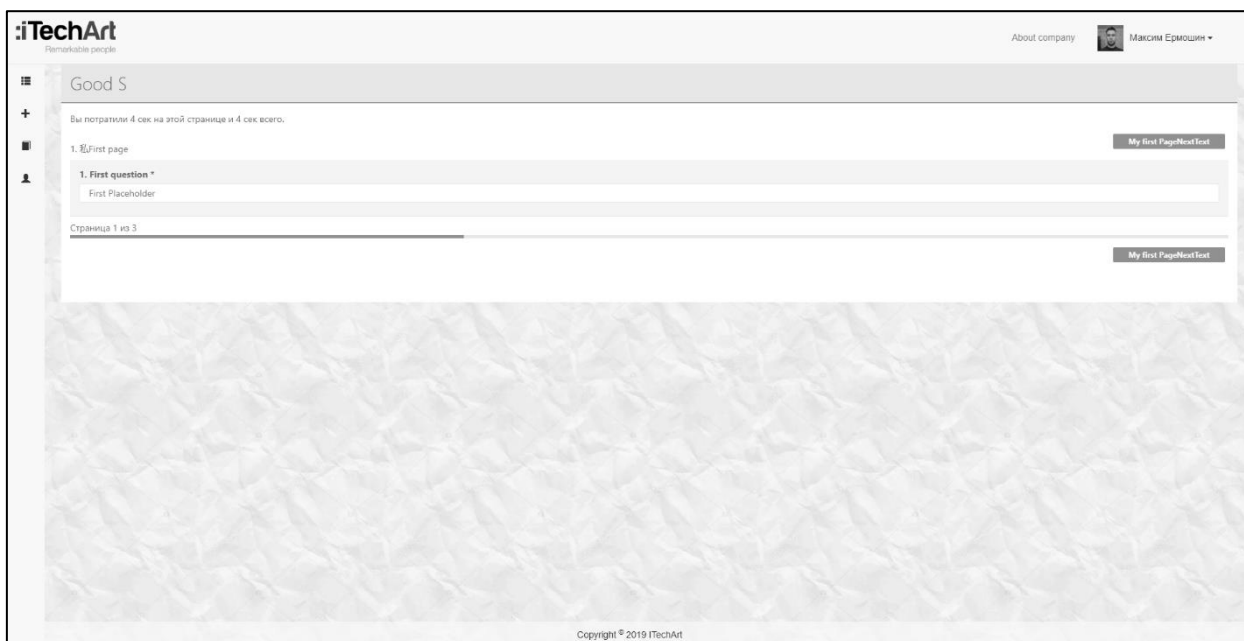


Рисунок 6.7 – Страница опроса

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).

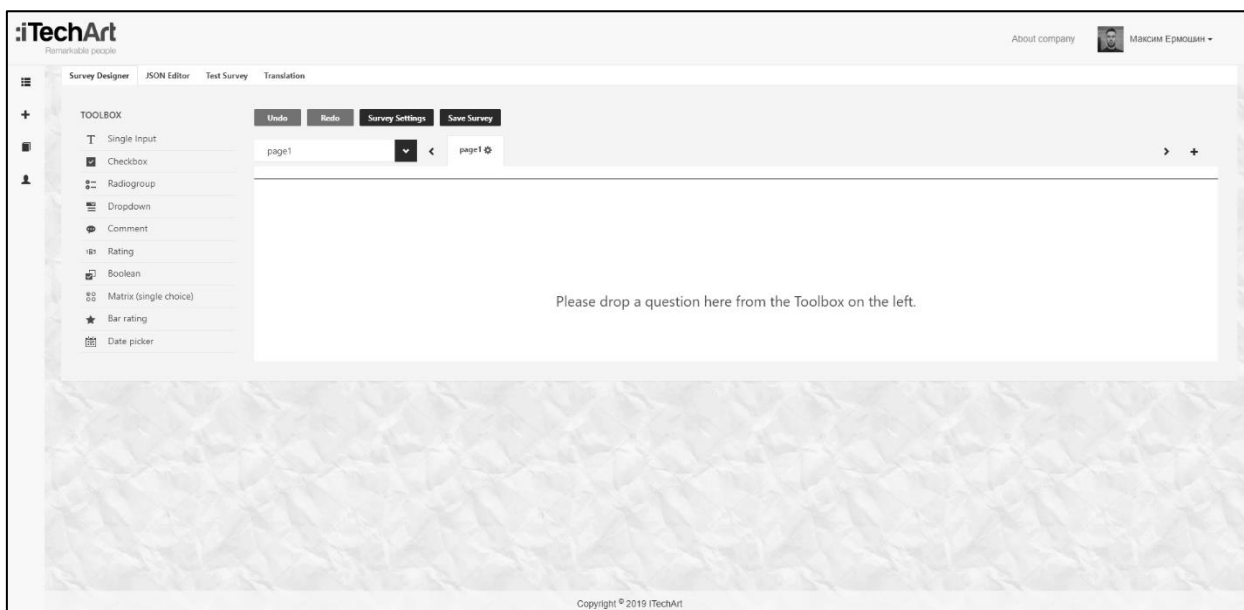


Рисунок 6.8 – Редактор опросов

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).

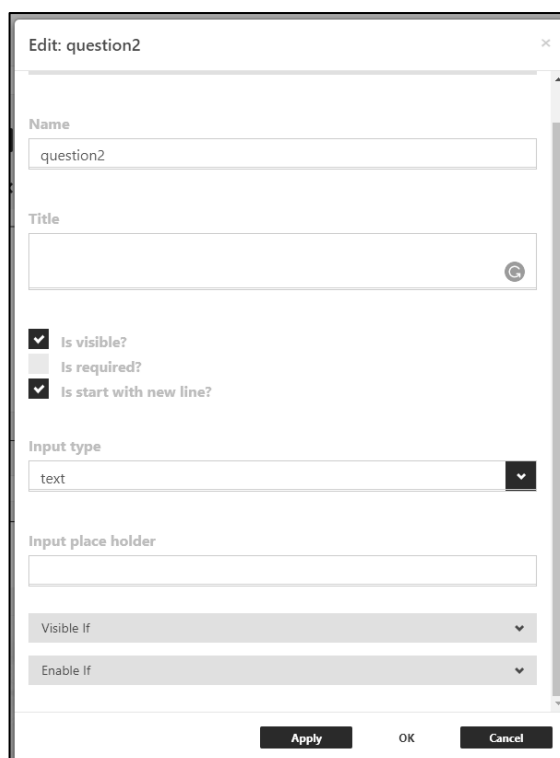


Рисунок 6.12 – Окно настроек вопроса

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.

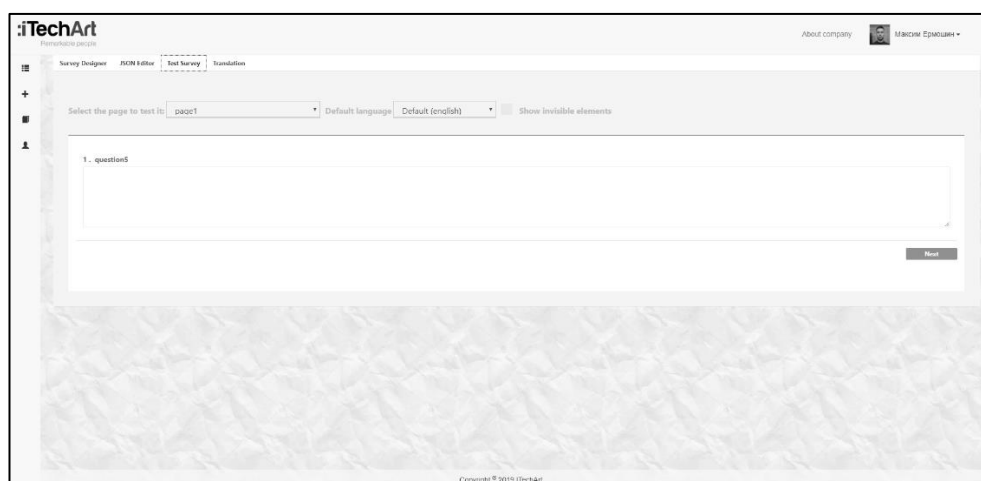


Рисунок 6.13 – Тестирование опроса

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.

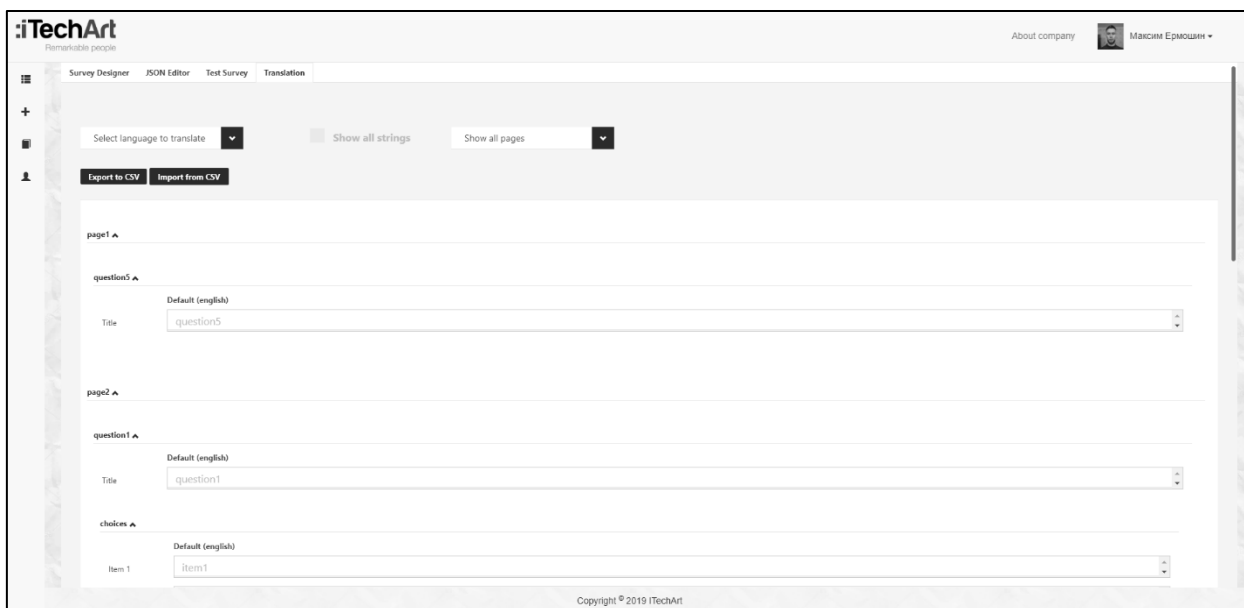


Рисунок 6.14 – Перевод опроса

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:

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

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] История онлайн исследований [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://bit.ly/2Z289rq>.
- [2] Девятко И.Ф, Онлайн исследования и методология социальных наук: новые горизонты, новые (и не столь новые) трудности/ Онлайн-исследования в России 2.0. М.: РИЦ «Северо-Восток», 2010. — С.17-31. — ISBN 978-5-9901939-1-8 <https://bit.ly/2IcKrn7>.
- [3] Google Forms [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.google.com/intl/ru_ua/forms/about/.
- [4] Typeform [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.typeform.com/>.
- [5] Survey Monkey [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.surveymonkey.com/>.
- [6] Online Test Pad [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://onlinetestpad.com/ru>.
- [7] Anketolog.ru [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://anketolog.ru/>.
- [8] Survey Gizmo [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://app.surveygizmo.com/login/v1>.
- [9] Testograf.ru [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.testograf.ru/ru/>.
- [10] Survio [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.survio.com/ru/>.
- [11] Simpoll [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://simpoll.ru/>.
- [12] Opros.by [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://opros.by/>.

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.SaveChangesAsync();

    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);

```