Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«_____» _____ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Разработка автоматизированной системы формирования заявок на отправку
Тема

корреспонденции: модуль клиентской части

Руководитель ВКР	_____	доцент, канд. техн. наук	А. В. Хныкин
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		В. С. Якимов
	подпись, дата		инициалы, фамилия

Красноярск 2021

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«_____» _____ 2021 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Якимов Владислав Сергеевич
фамилия, имя, отчество

Группа КИ17-16Б Направление (специальность) 09.03.04
номер код

Программная инженерия
наименование

Тема выпускной квалификационной работы
Разработка автоматизированной системы формирования заявок на отправку корреспонденции: модуль клиентской части

Утверждена приказом по университету № 5308/с от 19.04.2021

Руководитель ВКР А. В. Хныкин, доцент, канд. техн. наук, ИКИТ СФУ
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР описание предметной области

Перечень разделов ВКР введение, анализ предметной области, планирование и проектирование, разработка и тестирование клиентской части, заключение, список сокращений, список использованных источников

Перечень графического материала Презентация в формате PowerPoint

Руководитель ВКР _____ А. В. Хныкин
подпись инициалы и фамилия

Задание принял к исполнению _____ В. С. Якимов
подпись инициалы и фамилия студента

« ___ » _____ 2021 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка автоматизированной системы формирование заявок на отправку корреспонденции: модуль клиентской части» содержит 48 страниц текстового документа, 20 использованных источников, 34 иллюстрации, 2 таблицы.

ВЕБ-ПРИЛОЖЕНИЕ, АВТОМАТИЗАЦИЯ, КОРРЕСПОНДЕНЦИЯ, КЛИЕНТСКАЯ ЧАСТЬ, УПРАВЛЕНИЕ ПРОЕКТОМ, МЕТОДОЛОГИЯ РАЗРАБОТКИ.

Целью данной выпускной квалификационной работы является реализация клиентской части системы для автоматизации обмена корреспонденцией между отделами организации за счет ускорения заполнения заявки пользователем, проверки, дополнении заявки делопроизводителем и оповещении пользователей.

Для достижения поставленной цели были решены следующие задачи:

- определен технологический стек;
- разработана исходная концепция;
- подготовлено предварительное техническое решение;
- подготовлен предварительный план работ;
- спроектирована архитектура будущего приложения;
- спроектирован интерфейс системы;
- разработана спроектированная система;
- протестирована клиентская часть приложения.

Кроме этого, появился опыт в реализации взаимодействия клиентской и серверной частей системы, а также в организации работы с заказчиком.

Как итог, была разработана клиентская часть автоматизированной системы формирования заявок на отправку корреспонденции с использованием ASP.NET Core. Данная система позволит облегчить и ускорить прием и отправку корреспонденции.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сотрудникам организации при исполнении своих должностных обязанностей и в прочих рабочих процессах часто возникает необходимость отправки и получении деловой корреспонденции (оригиналы документов, договоры, бухгалтерские документы и т. д.). Чтобы отправить корреспонденцию, приходится проделывать множество рутинных действий. Например, необходимо тратить время на хождение по кабинетам, распечатку заявки, самостоятельное отслеживание статуса заявки. Данный процесс отправки корреспонденции зачастую никак не автоматизирован, из-за чего в организациях тратится множество времени на те действия, которые могли бы выполняться в разы быстрее.

Целью данной выпускной квалификационной работы является реализация клиентской части системы для автоматизации обмена корреспонденцией между отделами организации за счет ускорения заполнения заявки пользователем, проверки, дополнении заявки делопроизводителем и оповещении пользователей.

Для достижения цели работы были решены следующие задачи:

- определен технологический стек;
- разработана исходная концепция;
- подготовлено предварительное техническое решение;
- подготовлен предварительный план работ;
- спроектирована архитектура будущего приложения;
- спроектирован интерфейс системы;
- разработана спроектированная система;
- протестирована клиентская часть приложения.

1 Анализ предметной области

1.1 Анализ существующего решения

На данный момент процесс отправки корреспонденции выстроен следующим образом:

1. Отправитель заполняет шаблон заявки на отправку корреспонденции (рисунок 1). Заполняется информация о получателе, отправителе, а также о самом отправлении.

2. Отправитель распечатывает заполненную заявку на отправку корреспонденции и расписывается в ней.

3. Отправитель относит распечатанную заявку и само отправление (документы) делопроизводителям организации.

4. Делопроизводители скрепляют заявку и отправление вместе, а затем передают в курьерскую службу.

Далее опционально:

5. Отправитель через какое-то время пишет делопроизводителям и просит предоставить трек-номер отправления.

6. Отправитель сообщает трек-номер получателю отправления (чтобы получатель мог самостоятельно отследить статус отправления).

Все поля кроме «Примечание» являются обязательными для заполнения. Дата заявки формируется автоматически по макросу, подставляется текущая дата.

После изучения и анализа данного алгоритма был предложен и реализован новый алгоритм:

1. Пользователь заходит в приложение и создает заявку:

1.1. создает новую заявку с нуля (все поля пустые); для заполнения заявки либо выбирает значения «организации» и «получателя» из справочника, либо вписывает их вручную;

1.2. создает новую заявку на основе созданных ранее заявок; в таком случае

все поля дублируются из выбранной заявки и подставляются в новую (кроме даты).

Заявка на экспресс-доставку

Дата заявки	*	15.09.2020
Организация-получатель	*	АО «Рога и Копыта»
ФИО получателя	*	Иванов Иван Иванович
Контактный телефон получателя	*	8 800 555 25 35
Адрес получателя	*	876543, г. Томск, пр. Мира, д. 72
Описание вложения (что отправляем)	*	Дополнительное соглашение
Примечание		
ФИО отправителя	*	Александров Александр Александрович
Наименование структурного подразделения	*	Отдел автоматизации
Подпись отправителя		
Отметка об исполнении заявки		

* поля обязательные для заполнения

Отслеживание отправлений по номеру накладной
http://www.dimex.ws/otslezhivanie-nakladnyh/

Рисунок 1 – Шаблон заявки на отправку корреспонденции

2. Пользователь отправляет заявку в обработку нажатием на кнопку. При этом приложение должно отправлять оповещения делопроизводителям и самому отправителю по эл. почте. Отправленная в работу заявка сохраняется в списке

заявок пользователя.

3. Заявка попадает в реестр заявок делопроизводителей. Делопроизводители заполняют оставшуюся информацию о заявке (вес, количество мест, стоимость и т. д.).

4. После факта передачи отправления в экспресс-службу делопроизводители меняют статус заявки на «Выполнено» и вносят в реестр трек-номер выполненной заявки.

5. Приложение посылает эл. письмо с трек-номером (номер накладной) заявки получателю и отправителю.

1.2 Определение требований к программному продукту

В результате общения с заказчиком, был составлен ряд требований к разрабатываемому продукту.

Во-первых, в приложении подразумеваются разные роли пользователей: пользователь, делопроизводитель, администратор. От роли будут зависеть доступность разных функций и частей приложения.

Для пользователя с ролью «Пользователь»:

- доступен сервис создания заявок;
- доступен сервис просмотра списка заявок;
- создание заявки на основе ранее созданных (дублирование);
- удаление созданных данным пользователем ранее заявок в списке заявок.

Для пользователя с ролью «Делопроизводитель»:

- тот же функционал что и у «Пользователя»;
- доступ к сервису «Реестр заявок» в полном объеме.

Для пользователя с ролью «Администратор»:

- доступ ко всем сервисам приложения;
- доступ к «Панели администратора».

Во-вторых, приложение должно поддерживать работу со справочниками, которые будут использоваться для заполнения заявок на отправку

корреспонденции (далее заявка). В таком справочнике будет три сущности: организации-контрагенты, адреса организации (здания) и сотрудники-получатели.

У одной организации может быть несколько адресов и несколько сотрудников. Сотрудники и адреса между собой не связаны, а связаны только с организацией.

У организации будет только одно поле:

- наименование организации;

У адреса организации будут следующие поля:

- связь с организацией, которой принадлежит здание;
- индекс;
- город;
- адрес (улица, дом и т.д.);

У сотрудника-получателя будут следующие поля:

- связь с организацией, в которую входит сотрудник;
- ФИО;
- контактный телефон;
- адрес электронной почты.

В-третьих, дизайн сайта должен соответствовать следующим требованиям:

- интуитивно понятный интерфейс;
- цветовая палитра соответствующая корпоративному дизайну;
- весь функционал, предоставляемый веб-приложением, должен быть очевидным, то есть назначение каждого элемента и блока на сайте должно быть понятно из названия или внешнего вида;
- удобное навигационное меню;
- пункты меню должны должны быть видны только для соответствующей роли.

1.3 Выбор инструментов разработки

Стек технологий – это набор технологий и инструментов, на основе которых разрабатывается программный продукт.

Из-за того, что данный модуль планируется в дальнейшем интегрировать в веб-системы заказчика, стек технологий был определен заранее. Это язык C# и фреймворк Asp.NET Core.

C# – объектно-ориентированный язык программирования, разработанный корпорацией Microsoft в 1998–2001 годах [1]. C# создавался как язык компонентного программирования, и в этом одно из главных достоинств языка, направленное на возможность повторного использования созданных компонентов. Язык реализует возможности наследования и универсализации и тесно связан с мощной средой Framework .Net. Благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java.

Эффективность кода даже повышается, поскольку исполнительная среда для C# представляет собой компилятор промежуточного языка [2], в то время как виртуальная Java-машина является интерпретатором байт-кода.

Изначально язык C# не являлся кроссплатформенным, однако с появлением платформы NET.Core [3, 4] появилась возможность запуска приложений не только на Windows, но и на Linux и MacOS.

В качестве платформы для разработки веб-приложений наиболее актуальной является Asp.NET Core. Плюсами данной платформы являются:

- несколько возможностей развертывания приложения;
- удобное распространение пакетов приложения;
- расширяемость;
- кроссплатформенность;
- быстрое развитие;

- возможность асинхронной обработки запросов.

Недостатки:

- относительно молодая платформа, некоторые инструменты еще не поддерживаются.

Основной средой программирования для языка C# и платформы Asp.NET Core является Microsoft Visual Studio Community Edition. Данная среда не имеет аналогов по качеству поддержки всех необходимых инструментов для разработки на выбранной платформе. Существует несколько версий данной среды, нами использовалась Microsoft Visual Studio 2019.

Для клиентской части приложения, так как разрабатываемый продукт является веб-приложением, то обязательными элементами технологического стека являются язык гипертекстовой разметки HTML, формальный язык CSS [5], необходимый для описания стилей веб-страниц, и язык программирования JavaScript.

Для удобной работы с JavaScript была выбрана библиотека jQuery, обладающая определенным рядом положительных сторон [6]:

- скорость, приходится писать меньше кода, по сравнению с чистым JavaScript;

- понятность, код и функции JQuery гораздо более удобны для чтения и понимания;

- кроссбраузерность, то что пишется в JQuery почти 100% будет работать во всех современных браузерах.

Также был выбран фреймворк Asp.NET Core MVC [7] для реализации архитектуры MVC. Model-View-Controller (MVC) — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние. Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Для отображения данных для клиента был использован фреймворк DevExtreme by DevExpress [8]. DevExtreme − это коммерческая прикладная среда, разработанная DevExpress. Она основана на HTML5 и JavaScript и может создавать собственные приложения для смартфонов и планшетов (Windows Phone, iOS и Android) и адаптивные веб-приложения для традиционного рабочего стола.

DevExtreme − это набор корпоративных наборов компонентов пользовательского интерфейса для Angular, React, Vue и jQuery. В нем есть все, что нужно для создания адаптивных веб-приложений для сенсорных устройств и традиционных персональных компьютеров: таблицы данных, интерактивные диаграммы, редакторы данных, навигация и многоцелевые виджеты. Эти элементы управления предназначены для того, чтобы отлично выглядеть и обеспечивать мощную функциональность в любом браузере.

1.4 Выводы

Целью данного раздела являлся анализ предметной области, в результате которого был предложен новый алгоритм отправки корреспонденции. Анализ предметной области помог выявить требования к сайту, позиционирующего себя как модуль отправки корреспонденции. Это немаловажный пункт, так как к подобным информационным ресурсам предъявляются особенные требования. Далее предложенный алгоритм был использован при создании веб-приложения, учитывая его ответвления.

Следующий этап – определение функциональных требований к разрабатываемому продукту. Выявленные требования были основаны на проведенном интервью с заказчиком. Разрабатываемый продукт призван отвечать требованиям заказчика.

Заключительным этапом работы стал выбор инструментов разработки.

Этот шаг был очень важным и требовал особого внимания, так как выбор инструментов влияет на множество аспектов: масштабируемость, быстродействие, тестируемость, надежность, безопасность и т.п. В итоге был определен следующий технологический стек:

- языки программирования для клиентской части C# и JavaScript;
- программная платформа для реализации клиентской части Asp.NET Core;
- среда программирования Microsoft Visual Studio 2019;
- фреймворк для JavaScript – jQuery;
- фреймворк для реализации архитектуры MVC – Asp.NET Core MVC;
- фреймворк для отображения данных для клиента DevExtreme by DevExpress.

2 Планирование и проектирование

2.1 Планирование

Важным аспектом управления проектом является грамотное распределение работ процесса разработки программного продукта так, чтобы разработка проекта успешно и завершилась в установленные сроки.

2.1.1 Выбор методологии разработки

Методология разработки ПО описывает подход к разработке программного продукта и методы организации коллективной разработки. Существует большое количество таких подходов, каждый из которых учитывает различные аспекты и предназначен для конкретной ситуации. Для выбора методологии для данного проекта сравним некоторые из них.

Водопадная модель является одной из самых старых, она является традиционной и подразумевает строгое и последовательное выполнение всех этапов [9]. Новый этап должен начинаться только после завершения предыдущего. Такая модель крайне проста в использовании и дедлайны проекта определены заранее. Но данная модель не очень гибкая, она не позволяется вернуться на предыдущие этапы.

В качестве альтернативы воподадному подходу, можно рассмотреть спиральный подход. В спиральном подходе жизненный цикл разработки продукта имеет очень сложную организацию и требует большой внимательности, так как он, в первую очередь, направлен на раннее выявление и уменьшение проектных рисков.

Есть еще итеративный подход, решающий проблемы водопадной методологии [10]. Он дает возможность учитывать изменяющиеся требования, а также использовать опыт повторно, выделяя некоторые типовые решения. Но в случае данного проекта, требования точно определены и необходимость в

использовании данной методологии отсутствует.

Существуют и более гибкие решения, говоря о которых, нельзя не упомянуть об Agile – наборе методов и методологий, которые помогают команде разработчиков эффективнее мыслить, работать и принимать решения [11]. Agile поддерживает открытое планирование, обсуждение дизайна и других задач всей командой.

Одним их популярных Agile подходов является Scrum [12], концепция которого заключается в делении процесса разработки на спринты, то есть итерации, каждая из которых строго ограничена по времени. В конце каждой такой итерации конечному пользователю предоставляется работающая версия ПО с новыми возможностями, для которых определен наибольший приоритет.

Возможности ПО к реализации в очередном спринте определяются в начале спринта на этапе планирования не могут изменяться на всем его протяжении. При этом строго фиксированная небольшая длительность спринта придает процессу разработки предсказуемость и гибкость.

Scrum – это наглядная система разработки, и для достижения наглядности используются Scrum-доски. Такая доска имеет минимум три колонки: «сделать» (to-do), «в процессе» (in progress) и «сделано» (done).

Scrum методология проста в использовании, наглядна, благодаря визуальному отображению информации в формате досок, а также позволяет привнести в разработку большую гибкость и предсказуемость. Из минусов можно отметить отсутствие строгих дедлайнов и сложность применения к долгосрочным проектам. Именно благодаря данным преимуществам была выбрана методология Scrum для управления разработкой веб-приложения.

2.1.2 Команда

Команда, занимающаяся разработкой системы, состоит из 2 человек: Якимова Владислава Сергеевича и Андрианова Владислава Сергеевича

(таблица 2.1). Для обеспечения эффективного использования человеческих ресурсов проекта, за каждым участником команды была закреплена роль и ответственность за соответствующую часть самого проекта (таблица 2.2).

Во время планирования выяснилось, что персоналу недостает опыта, который было решено восполнять в процессе разработки.

Якимову Владиславу Сергеевичу недостает опыта в:

- ASP.NET Core;
- ASP.NET Core MVC;
- DevExtreme by DevExpress;
- управление персоналом.

Андрианову Владиславу Сергеевичу недостает опыта в:

- C#;
- Чистая архитектура;
- Entity Framework Core.

Таблица 2.1 – Состав команды и роли участников

ФИО	Роль
Якимов Владислав Сергеевич	Руководитель проекта, Программист, Технический писатель
Андрианов Владислав Сергеевич	Программист, Тестировщик

Таблица 2.2 – Функции и ответственности участников команды

Роль	Ответственность / функции
Руководитель проекта, Программист, Технический писатель	Управление всеми процессами, постановка задач участникам проекта, ответственность за сроки сдачи задач, руководство участниками проекта, реализация клиентской части проекта, сбор информационных данных с команды для составления отчета, составление рабочей документации проекта.
Программист, Тестировщик	Реализация серверной части проекта, оценка качества разрабатываемого модуля, поиск ошибок и сбоев в функциональности модуля

2.1.3 Управление проектом

Для управления работой над проектом был использован инструментарий, предлагаемый системой управления проектами Trello [13]. Данная система используется для небольших проектов, так как обладает для этого всем необходимым функционалом, а также имеет бесплатный доступ к большинству функций. Более того, она позволяет организовать работу по идеям инструмента Scrum-досок, применяемых в методологии Scrum, выбранной для данного проекта. Trello имеет простой и понятный интерфейс, десктопную, мобильную и онлайн версии.

В общем структуру Trello можно описать следующими тремя элементами:

- доска – рабочий экран, содержащий списки;
- списки, содержащие карточки;
- карточки – формы для описания задач, которым можно задать название,

метку, срок выполнения, ответственных за задачу и подзадачи.

Пример использования сервиса Trello в данном проекте представлен на рисунке 2.

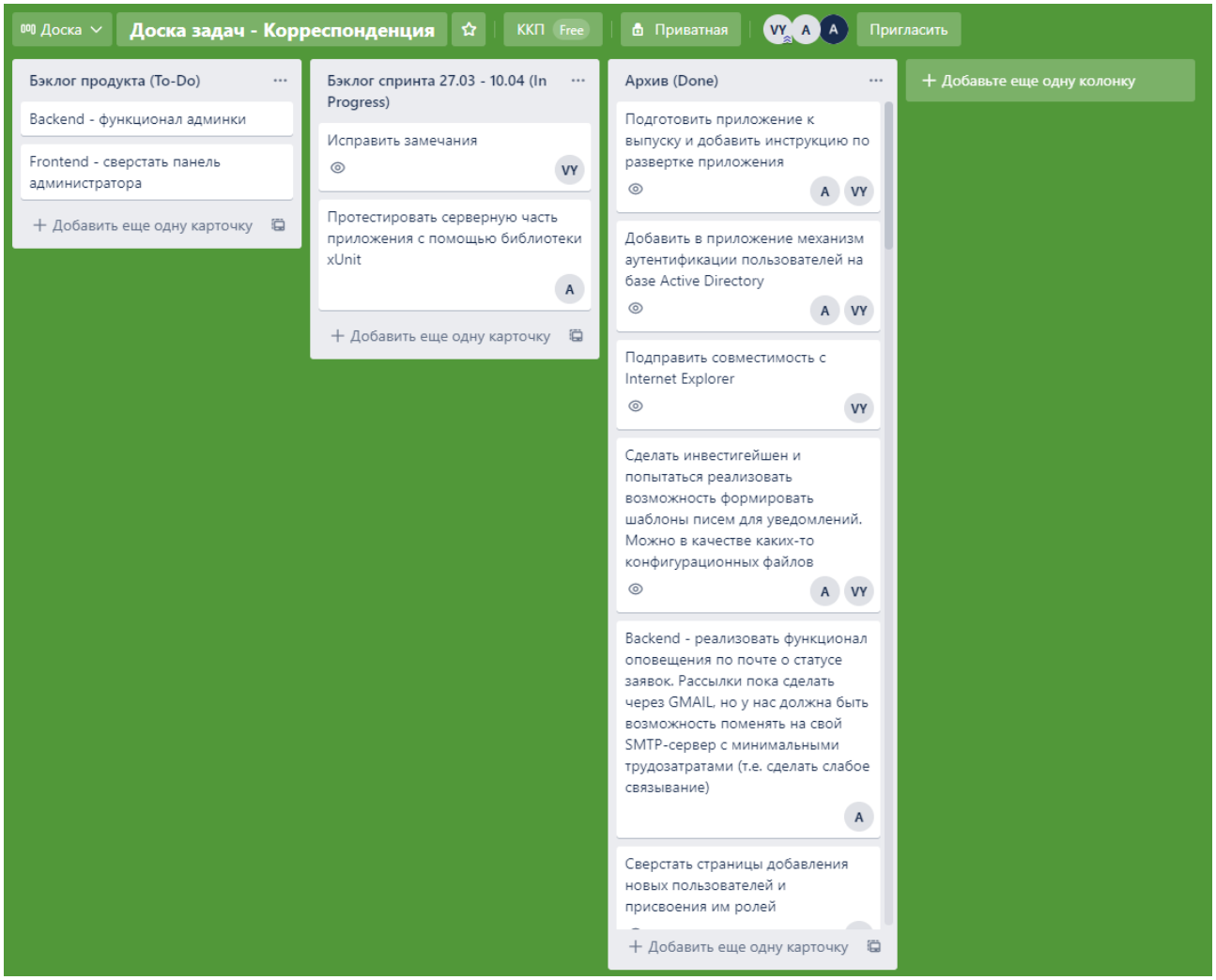


Рисунок 2 – Доска в Trello

2.2 Проектирование

2.2.1 Проектирование архитектуры информационной системы

Архитектура информационной системы – это концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов системы [14, 15]. Говоря о веб-приложениях, архитектура определяет

взаимодействие между приложениями, системами промежуточного программного обеспечения и базами данных, чтобы обеспечить совместную работу нескольких приложений.

При выборе архитектуры для веб-приложения необходимо понимать, что Интернет – это постоянно изменяющаяся и развивающаяся среда. Технологии разработки, архитектуры и другие элементы веб-разработки не всегда имеют строгую терминологию, и понятия могут быть размыты. Таким образом, разработчики понимают технологии по-разному, вследствие чего возникает огромное количество вариаций реализаций веб-приложений, что может запутать неопытных программистов на этапе проектирования.

Классическая клиент-серверная модель. Это двухзвенная архитектура, состоящая из клиента (объект, запрашивающий информацию по сети, например, персональный компьютер) и сервера. Функциональные части приложения, разработанного согласно данной архитектуре, взаимодействуют по схеме «запрос-ответ», где клиент инициирует запросы, а сервер пассивно на них отвечает [16]. Минусом данной модели является так называемый «толстый клиент», когда на клиентской части системы реализуется бизнес-логика. Это возлагает на клиент большую нагрузку, которую можно перенести на сервер. Для решения данной проблемы разработчики начали использовать серверные языки программирования, в результате чего выделился еще один уровень и образовался новый вид модели, описываемый далее.

Трехуровневая (трехзвенная) архитектура состоит из трех компонентов: клиент (уровень представления), сервер веб-приложения (уровень бизнес-логики) и сервер БД (уровень хранилища данных). Помимо устранения проблемы «толстого клиента», данная модель представляет возможность масштабируемости и большую конфигурируемость. Промежуточный уровень выполняет основную бизнес-логику, то есть принимает и обрабатывает запросы пользователя, совершает соответствующие запросы к БД, а затем обрабатывает полученный от сервера БД ответ и отправляет его клиенту. Так как данная модель легко масштабируема, то не возникнет трудности в работе с несколькими

базами данных одновременно.

Для более комплексных и сложных систем существует сервис-ориентированная архитектура (SOA) [17], суть которой заключается в модульном подходе к разработке. Часто данная архитектура используется для разработки программных комплексов, которые представляют из себя набор-веб служб.

Проанализировав вышеперечисленные модели, было принято решение использовать трехуровневую архитектуру.

Данная модель была выбрана по следующим причинам:

- масштабируемость;

- возможность многократного повторного использования общих функций обработки данных;

- конфигурируемость;

- высокая безопасность.

При выборе и проектировании архитектуры преследовалась цель максимально снизить зависимость между уровнями. Для этого необходимо убедиться, что каждый уровень оперирует необходимым только ему набором данных.

В итоге, веб-приложение приложение разбито на три независимые части:

- клиентская часть, разрабатываемая с использованием технологий HTML, CSS, JavaScript, jQuery, C#, Asp.NET Core, Asp.NET Core MVC, DevExtreme;

- серверная часть, разрабатываемая с использованием технологий C#, Asp.NET Core, Depedency Injection, MediatR, CQRS, Repository и библиотека Entity Framework Core;

- сервер БД, в роли которого выступает SQL Server Express.

2.2.2 Проектирование интерфейса веб-приложения

Интерфейс приложения является своеобразным «рычагом» взаимодействия пользователя с системой, соответственно, чем лучше интерфейс,

тем эффективнее взаимодействие. Интерфейс пользователя характеризуется множеством факторов, от выбора цветовой палитры до распределения элементов интерфейса на странице.

При проектировании интерфейса веб-приложения, необходимо было выполнить следующие задачи:

- определить общие принципы и правила, которым должен соответствовать выходной продукт;

- определить принципы, которых необходимо придерживаться при создании элементов веб-страниц;

- определить цветовую палитру.

Выполнение всех вышеперечисленных заданий облегчилось благодаря пожеланиям заказчика. Заказчик захотел, чтобы цветовая палитра сайта была выполнена в корпоративных цветах, которые также использовались в другом продукте заказчика (рисунки 3 и 4).

Кроме этого, заказчик предоставил макет приложения, который он хочет видеть (рисунок 5).

Важно, чтобы пользователь совершал важные и критические действия предварительно подтверждая их. Причем диалоги подтверждения и другие различные важные информационные сообщения должны быть заметны на экране, что можно сделать либо, выводя сообщение в центре экрана, то есть в активной рабочей зоне пользователя, либо делать акцент на размер элемента или его цвет.

Создаваемые элементы интерфейса не должны быть сложными и разноцветными, в данном случае работает принцип «чем проще, тем понятнее». Также все элементы должны быть выполнены в едином стиле.

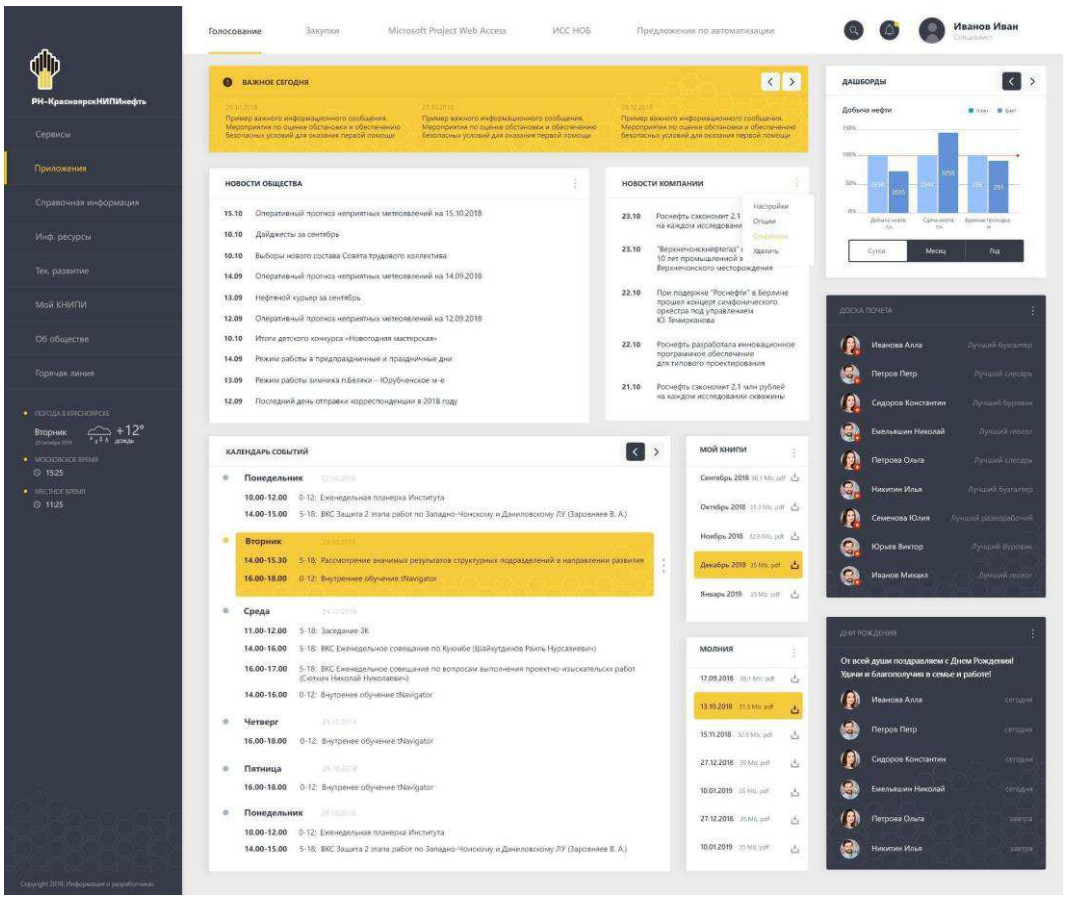


Рисунок 3 – Пример продукта заказчика, dashboard

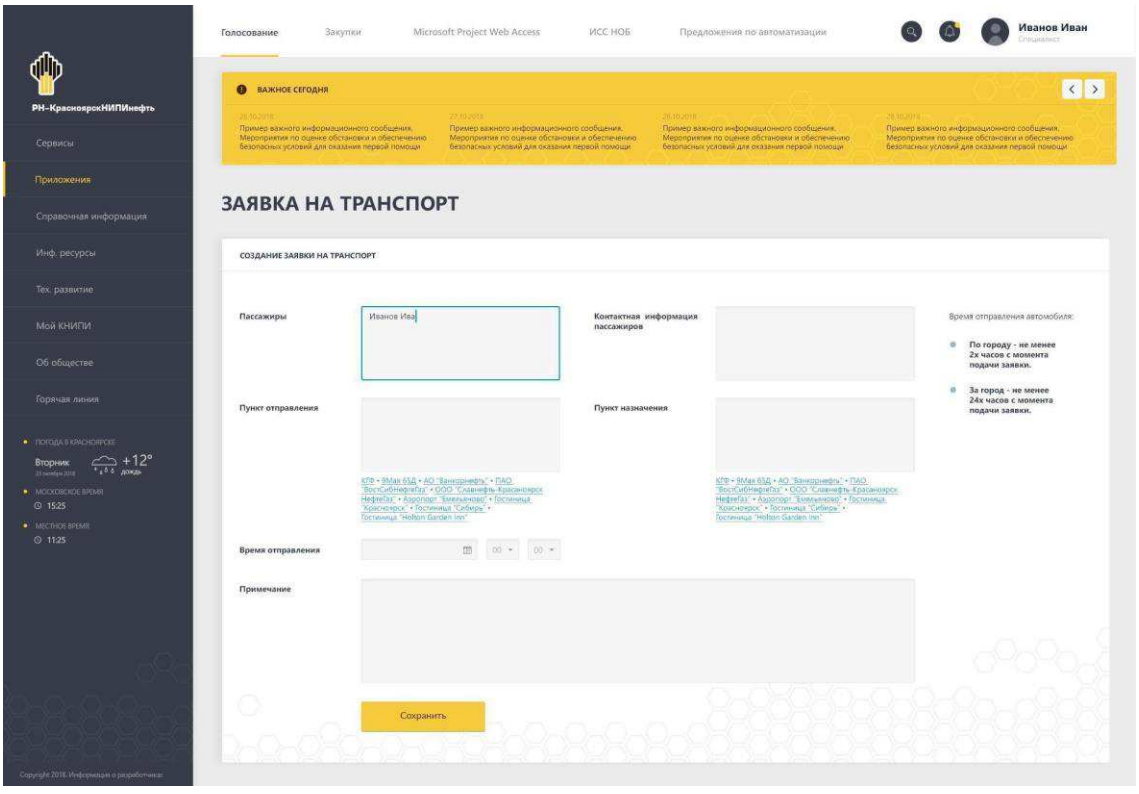


Рисунок 4 – Пример продукта заказчика, заявка на транспорт

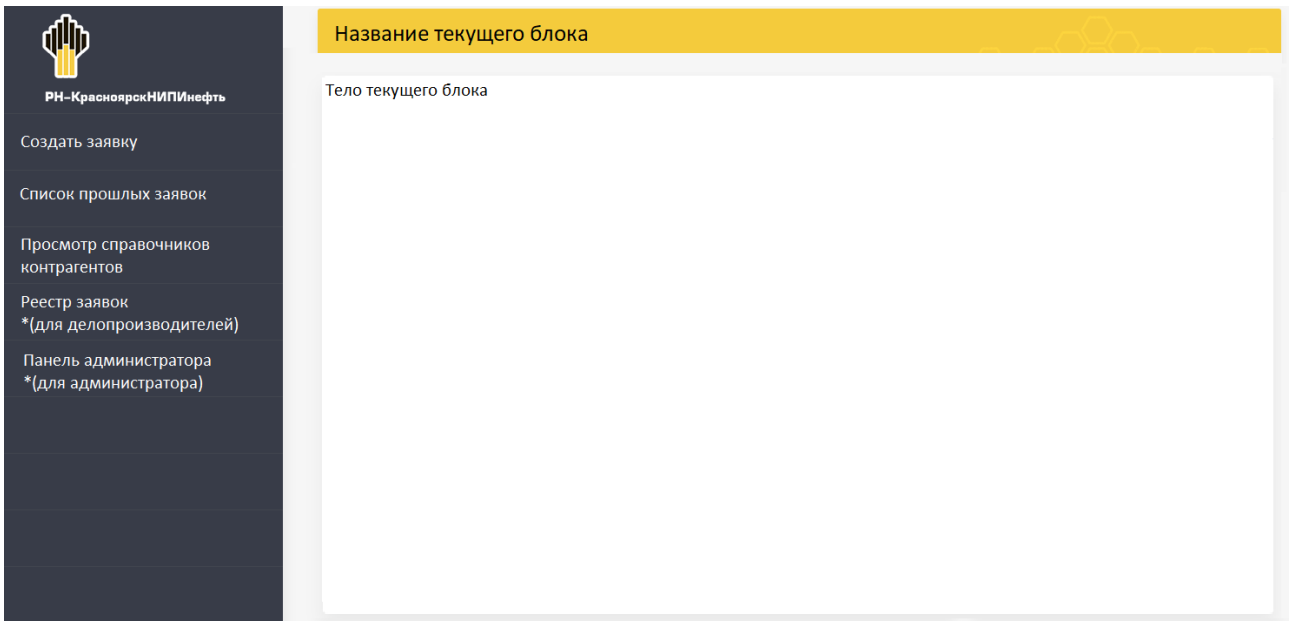


Рисунок 5 – Макет интерфейса приложения

Разрабатываемое веб-приложение служит для автоматизации обмена корреспонденции между отделами организации за счет ускорения заполнения заявки пользователем, проверки, дополнении заявки делопроизводителем и оповещении пользователей. Таким образом, появляется необходимость отделить функционал обычного пользователя сайта и делопроизводителя. Данное разделение происходит с помощью скрытии функционала в зависимости от роли пользователя. Для обычных пользователей, которые желают отправить корреспонденцию, для них доступна только возможность создать заявку и посмотреть текущие и старые заявки. Для делопроизводителей системы открывает возможность редактировать справочники, просматривать любые заявки, работать в реестре заявок.

2.3 Выводы

Была определена методология разработки программного продукта. Для достижения поставленной цели были проанализированы и сравнены такие модели, как водопадная, спиральная, итеративная и Scrum. Так как для данного проекта гибкость и предсказуемость является очень важными свойствами, то в

качестве методологии разработки был выбран подход Scrum, обеспечивающий данные свойства, так как управление проектом происходит с помощью визуализации списка задач. Также были распределены роли в команде, а для удобства применения методологии Scrum и распределения задач в команде был выбран сервис Trello.

В следующем подразделе были спроектированы архитектура информационной системы и пользовательский интерфейс.

Для данной системы с ее характеристиками было принято решение использовать трехзвенную архитектуру «клиент-сервер-база данных». Это не только снизит объем нагрузки на клиентскую часть веб-приложения, но также позволит при необходимости расширить систему.

Последним этапом было проектирование интерфейса. Были установлены требования к его реализации и общие принципы, которым необходимо следовать, чтобы сделать систему максимально удобной и понятной конечному пользователю.

3 Разработка и тестирование клиентской части

Для разработки клиентской части системы была выбрана архитектура MVC. Для ее реализации был использован фреймворк ASP.Net Core MVC. MVC разделяет приложение на три отдельных компонента:

- модель, предоставляющая данные и реагирующая на команды контроллера, изменяя свое состояние;

- представление, отвечающее за отображение данных модели пользователю, и реагирующее на изменения модели;

- контроллер, интерпретирующий действия пользователя, оповещая модель о необходимости изменений.

Архитектура MVC также явно прослеживается в структуре проекта (рисунок 6) по директориям Controllers, Models и Views.

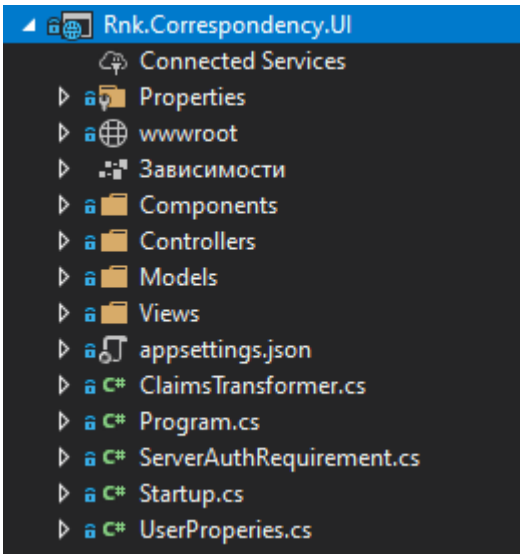


Рисунок 6 – Структура проекта

Данная архитектура предоставляет возможность модификации каждого компонента независимо. Также единая глобальная архитектура системы позволяет легко ориентироваться в программном коде даже в сложных системах.

3.1 Описание функционала

Разработанная система является веб-приложением, которое предназначено для автоматизации обмена корреспонденцией между отделами организации заказчика за счет ускорения заполнения заявки пользователем, проверки, дополнении заявки делопроизводителем.

В системе введена система ролей пользователей. Роли пользователя влияют на доступность некоторых пунктов меню или на доступные ему действия. Недоступные для роли пользователя пункты навигационного меню не отображаются для него. Всего реализовано 3 роли пользователей:

- Пользователь – рядовой пользователь, которому доступен лишь сервис создания заявки, а также сервис просмотра списка заявок и, соответственно, создания заявки на основе ранее созданных (дублирование заявки). Также пользователь может удалять созданные им ранее заявки в списке заявок.

- Делопроизводитель – ему доступны те же функции что и пользователю, но также дополнительно доступен сервис «Реестр заявок» в полном объеме.

- Администратор – ему доступны все сервисы приложения. Единственная роль с доступом к «Панели администратора».

Также система поддерживает работу со справочниками, которые используются для заполнения заявок на отправку корреспонденции. В таком справочнике три сущности: организации-контрагенты, адреса организации (здания) и сотрудники-получатели.

У одной организации может быть несколько адресов и несколько сотрудников. Сотрудники и адреса между собой не связаны, а связаны только с организацией.

У организации есть только одно поле:

- наименование организации.

У адреса организации следующие поля:

- индекс;

- город;

- адрес (улица, дом и т.д.).

У сотрудника-получателя следующие поля:

- ФИО;
- контактный телефон;
- адрес электронной почты.

Кроме этого, в приложении реализован реестр для хранения, обработки и редактирования заявок. Реестр хранит все заявки за все время.

Отправленная пользователем заявка попадает в верхнюю строчку реестра и автоматически заполняет следующие поля на основе полей заявки:

- дата отправления;
- отправитель (ФИО отправителя);
- получатель (Организация-контрагент);
- адрес получателя;
- получатель (ФИО получателя);
- контактный телефон получателя;
- электронная почта получателя;
- описание;
- подразделение.

Остальные поля заполняются делопроизводителем вручную:

- регион;
- количество мест;
- вес;
- вес КС Красноярск;
- номер накладной (Трек-номер).

3.2 Авторизация и аутентификация

После общения с заказчиком, было принято решение для аутентификации пользователя использовать Windows-аутентификацию [18]. Как и следует из названия, этот метод основывается на использовании учетных записей Windows.

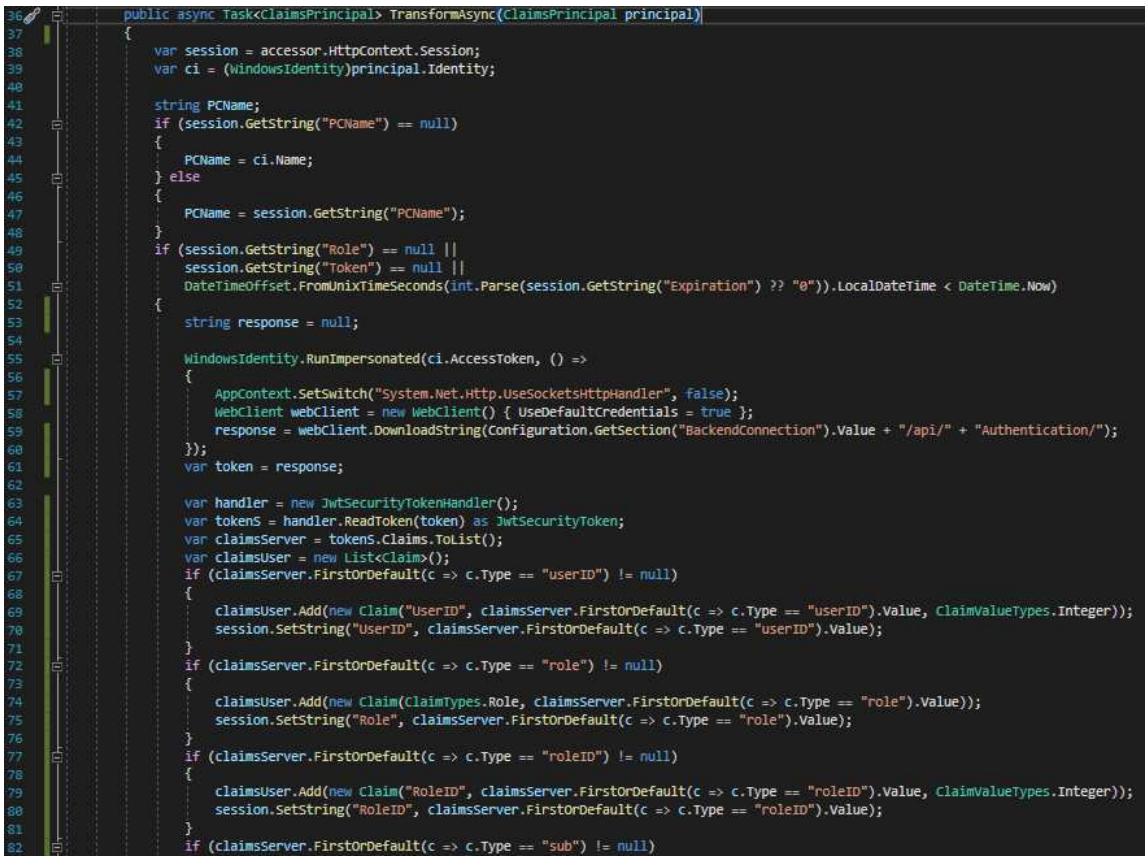
Огромным преимуществом данной аутентификации является то, что от пользователя не требуется вводить какую-либо информацию. При использовании этой модели аутентификации при обращении пользователя к ресурсам веб-приложения вместе с HTTP-запросом посылается и токен безопасности Windows, который и верифицирует пользователя.

Для реализации авторизации и аутентификации был реализован интерфейс IClaimsTransformation, который используется для изменения и добавления объектов Claim, хранящие информацию о пользователе.

В реализованном интерфейсе есть функция TransformAsync (рисунки 7 и 8), являющейся центральной точкой преобразования для изменения объектов Claim.

При аутентификации создаются следующие объекты Claim:

- UserID – уникальный идентификатор пользователя;
- Role – название роли пользователя;
- RoleID – уникальный идентификатор пользователя;
- Name – название учетной записи пользователя;
- Email – электронная почта пользователя;
- Department – отдел пользователя;
- FirstName – имя пользователя;
- LastName – фамилия пользователя;
- Patronymic – отчество пользователя;
- Token – токен пользователя.



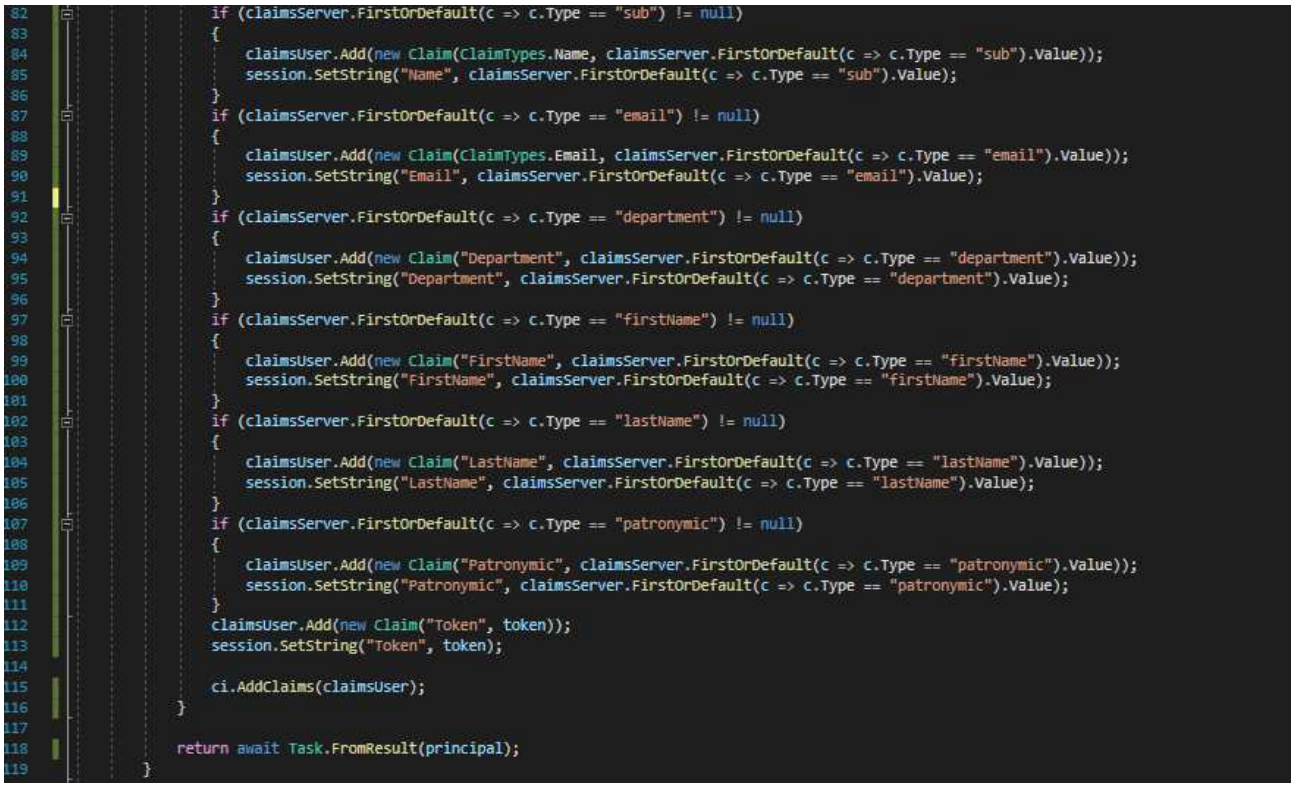
```
public async Task<ClaimsPrincipal> TransformAsync(ClaimsPrincipal principal)
{
    var session = accessor.HttpContext.Session;
    var ci = (WindowsIdentity)principal.Identity;

    string PCName;
    if (session.GetString("PCName") == null)
    {
        PCName = ci.Name;
    } else
    {
        PCName = session.GetString("PCName");
    }
    if (session.GetString("Role") == null ||
        session.GetString("Token") == null ||
        DateTimeOffset.FromUnixTimeSeconds(int.Parse(session.GetString("Expiration") ?? "0")).LocalDateTime < DateTime.Now)
    {
        string response = null;

        WindowsIdentity.RunImpersonated(ci.AccessToken, () =>
        {
            AppContext.SetSwitch("System.Net.Http.UseSocketsHttpHandler", false);
            WebClient webClient = new WebClient() { UseDefaultCredentials = true };
            response = webClient.DownloadString(Configuration.GetSection("BackendConnection").Value + "/api/" + "Authentication/");
        });
        var token = response;

        var handler = new JwtSecurityTokenHandler();
        var tokenS = handler.ReadToken(token) as JwtSecurityToken;
        var claimsServer = tokenS.Claims.ToList();
        var claimsUser = new List<Claim>();
        if (claimsServer.FirstOrDefault(c => c.Type == "userID") != null)
        {
            claimsUser.Add(new Claim("UserID", claimsServer.FirstOrDefault(c => c.Type == "userID").Value, ClaimValueTypes.Integer));
            session.SetString("UserID", claimsServer.FirstOrDefault(c => c.Type == "userID").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "role") != null)
        {
            claimsUser.Add(new Claim(ClaimTypes.Role, claimsServer.FirstOrDefault(c => c.Type == "role").Value));
            session.SetString("Role", claimsServer.FirstOrDefault(c => c.Type == "role").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "roleID") != null)
        {
            claimsUser.Add(new Claim("RoleID", claimsServer.FirstOrDefault(c => c.Type == "roleID").Value, ClaimValueTypes.Integer));
            session.SetString("RoleID", claimsServer.FirstOrDefault(c => c.Type == "roleID").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "sub") != null)
```

Рисунок 7 – Функция TransformAsync, часть 1



```
        if (claimsServer.FirstOrDefault(c => c.Type == "sub") != null)
        {
            claimsUser.Add(new Claim(ClaimTypes.Name, claimsServer.FirstOrDefault(c => c.Type == "sub").Value));
            session.SetString("Name", claimsServer.FirstOrDefault(c => c.Type == "sub").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "email") != null)
        {
            claimsUser.Add(new Claim(ClaimTypes.Email, claimsServer.FirstOrDefault(c => c.Type == "email").Value));
            session.SetString("Email", claimsServer.FirstOrDefault(c => c.Type == "email").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "department") != null)
        {
            claimsUser.Add(new Claim("Department", claimsServer.FirstOrDefault(c => c.Type == "department").Value));
            session.SetString("Department", claimsServer.FirstOrDefault(c => c.Type == "department").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "firstName") != null)
        {
            claimsUser.Add(new Claim("FirstName", claimsServer.FirstOrDefault(c => c.Type == "firstName").Value));
            session.SetString("FirstName", claimsServer.FirstOrDefault(c => c.Type == "firstName").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "lastName") != null)
        {
            claimsUser.Add(new Claim("LastName", claimsServer.FirstOrDefault(c => c.Type == "lastName").Value));
            session.SetString("LastName", claimsServer.FirstOrDefault(c => c.Type == "lastName").Value);
        }
        if (claimsServer.FirstOrDefault(c => c.Type == "patronymic") != null)
        {
            claimsUser.Add(new Claim("Patronymic", claimsServer.FirstOrDefault(c => c.Type == "patronymic").Value));
            session.SetString("Patronymic", claimsServer.FirstOrDefault(c => c.Type == "patronymic").Value);
        }
        claimsUser.Add(new Claim("Token", token));
        session.SetString("Token", token);

        ci.AddClaims(claimsUser);
    }

    return await Task.FromResult(principal);
}
```

Рисунок 8 – Функция TransformAsync, часть 2

Затем данные объекты используются в контроллерах и в представлениях. Например, Claim Role используется в компоненте представления для отображения меню (рисунок 9), пункты которого различаются в зависимости от роли.

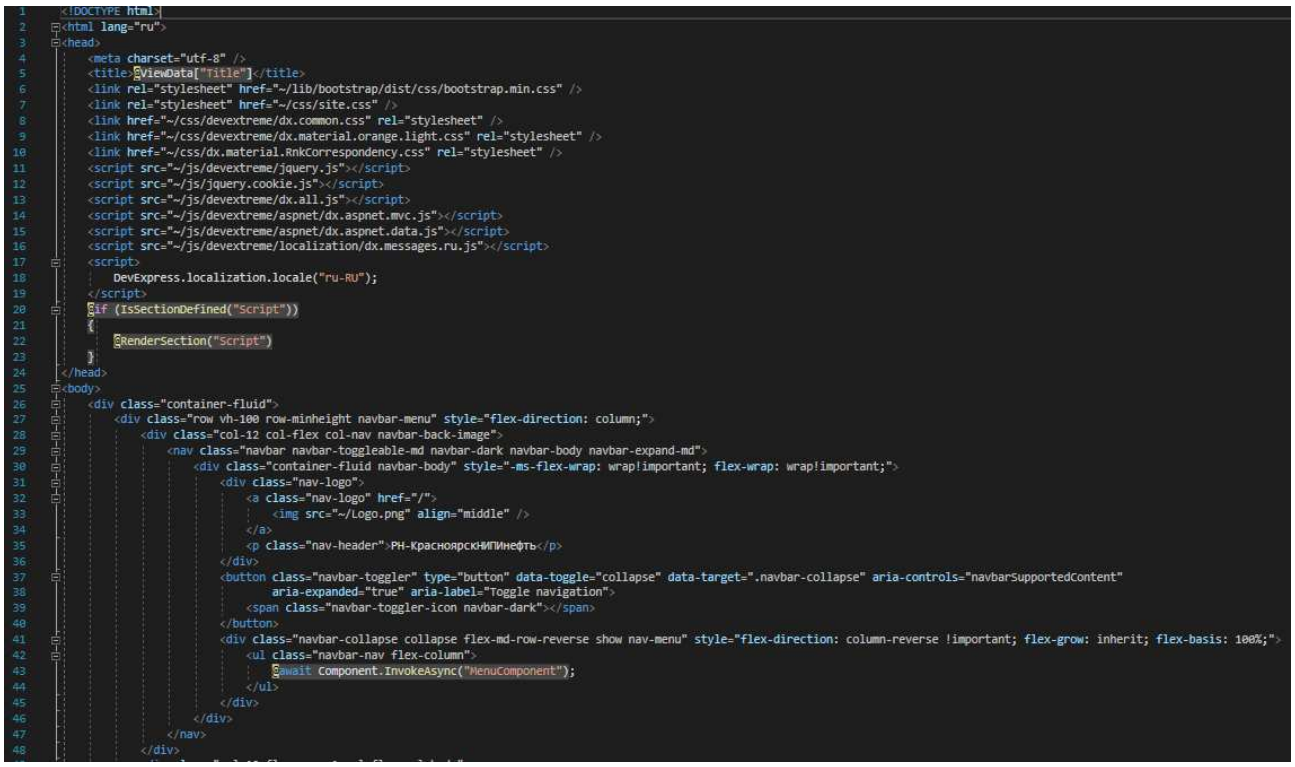
```
<li class="nav-item">
    <a class="nav-link" asp-area="" asp-controller="Apply" asp-action="CreateApply">Создать заявку</a>
</li>
<li class="nav-item">
    <a class="nav-link" asp-area="" asp-controller="Apply" asp-action="ListOfApply">Прошлые заявки</a>
</li>
@if (User.Role() == "Clerk" || User.Role() == "Admin")
{
    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-controller="Directory" asp-action="Index">Справочник контрагентов</a>
    </li>

    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-controller="Register" asp-action="Index">Реестр заявок</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-controller="Register" asp-action="GetRegister">Просмотр заявок для делопроизводителя</a>
    </li>
}
@if (User.Role() == "Admin")
{
    <li class="nav-item">
        <a class="nav-link" asp-area="" asp-controller="Admin" asp-action="Index">Панель администратора</a>
    </li>
}
```

Рисунок 9 – Компонент для отображения меню

3.3 Графический интерфейс

Для реализации макета, предоставленного заказчиком, была создана мастер-страница (рисунки 10, 11) с использование фреймворка Bootstrap и технологии Razor Pages [19]. Каждая страница Razor представляет собой файл с расширением .cshtml и содержит смесь кода html и конструкций C#. С помощью механизма визуализации Razor можно интуитивно легко создавать сложные композиции программного кода, гармонирующего с разметкой HTML. Когда сервер "читает страницу", движок Razor обрабатывает код веб-страницы прежде, чем сервер отправит сгенерированные данные в браузер. Загруженная в браузер веб-страница, порожденная серверным кодом, ничем не отличается от статического контента HTML.



```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="utf-8" />
    <title>@ViewData["Title"]</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
    <link href="~/css/devextreme/dx.common.css" rel="stylesheet" />
    <link href="~/css/devextreme/dx.material.orange.light.css" rel="stylesheet" />
    <link href="~/css/dx.material.RnkCorrespondency.css" rel="stylesheet" />
    <script src="~/js/devextreme/jquery.js"></script>
    <script src="~/js/jquery.cookie.js"></script>
    <script src="~/js/devextreme/dx.all.js"></script>
    <script src="~/js/devextreme/aspnet/dx.aspnet.mvc.js"></script>
    <script src="~/js/devextreme/aspnet/dx.aspnet.data.js"></script>
    <script src="~/js/devextreme/localization/dx.messages.ru.js"></script>
    <script>
        DevExpress.localization.locale("ru-RU");
    </script>
    @if (IsSectionDefined("Script"))
    {
        @RenderSection("Script")
    }
</head>
<body>
    <div class="container-fluid">
        <div class="row vh-100 row-minheight navbar-menu" style="flex-direction: column;">
            <div class="col-12 col-flex col-nav navbar-back-image">
                <nav class="navbar navbar-toggleable-md navbar-dark navbar-body navbar-expand-md">
                    <div class="container-fluid navbar-body" style="-ms-flex-wrap: wrap!important; flex-wrap: wrap!important;">
                        <div class="nav-logo">
                            <a class="nav-logo" href="/">
                                <img src="~/Logo.png" align="middle" />
                            </a>
                            <p class="nav-header">РН-КрасноярскНИПИнефть</p>
                        </div>
                        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                                aria-expanded="true" aria-label="Toggle navigation">
                            <span class="navbar-toggler-icon navbar-dark"></span>
                        </button>
                        <div class="navbar-collapse collapse flex-md-row-reverse show nav-menu" style="flex-direction: column-reverse !important; flex-grow: inherit; flex-basis: 100%;">
                            <ul class="navbar-nav flex-column">
                                @await Component.InvokeAsync("MenuComponent");
                            </ul>
                        </div>
                    </div>
                </nav>
            </div>
```

Рисунок 10 – Мастер-страница, часть 1

```
                        <div class="navbar-collapse collapse flex-md-row-reverse show nav-menu" style="flex-direction: column-reverse !important; flex-grow: inherit; flex-basis: 100%;">
                            <ul class="navbar-nav flex-column">
                                @await Component.InvokeAsync("MenuComponent");
                            </ul>
                        </div>
                    </div>
                </nav>
            </div>
            <div class="col-12 flex-grow-1 col-flex col-body">
                <div>
                    <div class="name-block name-block-back-image">
                        <p class="name-block-p" style="float: left">@ViewData["Title"]</p>
                        <p class="name-block-p" style="float: right">@User.FullName()</p>
                    </div>
                    <div class="body-block dx-viewport">
                        @RenderBody()
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

Рисунок 11 – Мастер-страница, часть 2

Для отображения данных были использованы DevExtreme виджеты. DevExtreme это набор высокопроизводительных виджетов пользовательского интерфейса, разработанные компанией DevExpress. Данные виджеты могут использоваться как в традиционных веб-приложениях, так и в мобильных приложениях нового поколения, кроме того они реализованы для множества технологий, но в данной системе была использована версия под jQuery.

У DevExtreme большой набор виджетов: Data Grid, Pivot Grid, Tree List,

Scheduler, Html Editor, Diagram, Charts, Gantt, Gauges, Navigation Editors, Forms, File Managment, Actions and Lists, Maps, Dialogs and Notifications [20]. Но в данной работе были использованы не все.

Data Grid - это виджет, который отображает данные в виде таблицы. Именно этот виджет был использован больше всего. С помощью данного виджета в этой работе отображаются прошлые заявки (рисунок 12), реестр заявок (рисунок 13), список пользователей (рисунок 14), справочник контрагентов (рисунок 15) и другие.

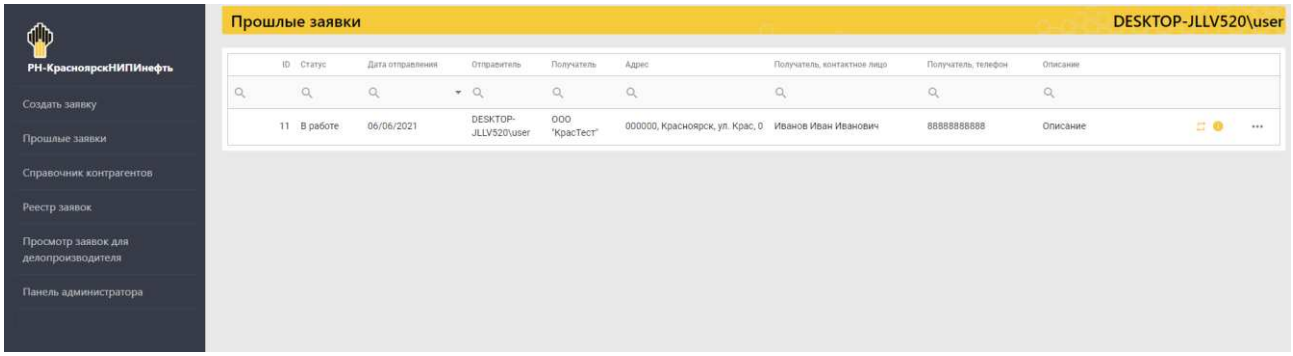


Рисунок 12 – Прошлые заявки

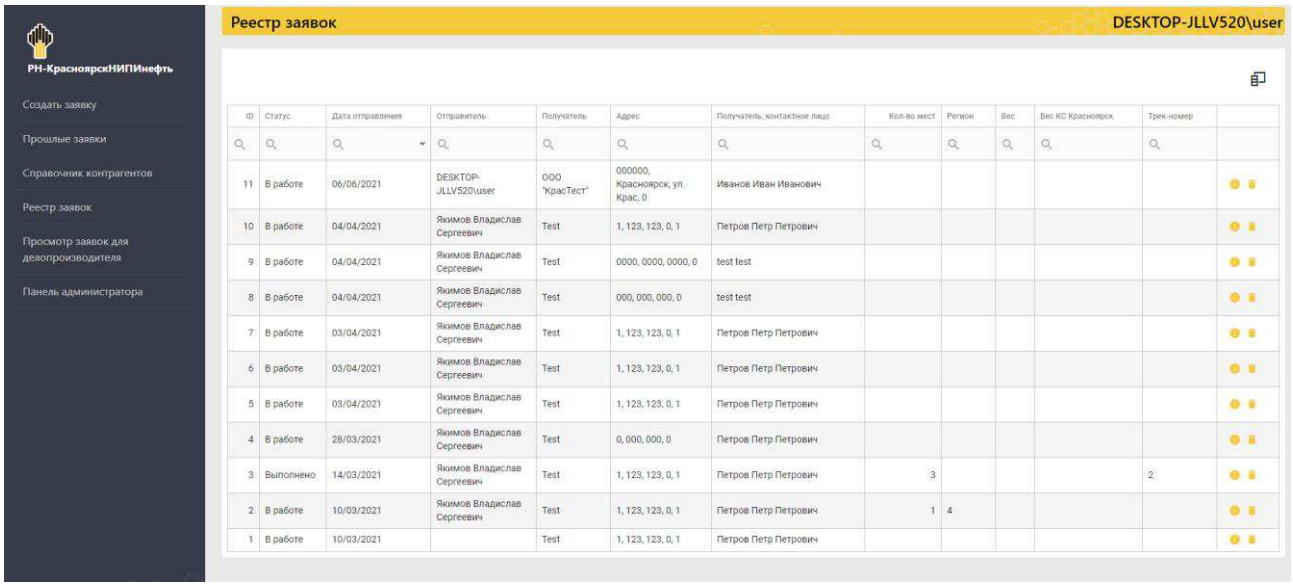


Рисунок 13 – Реестр заявок

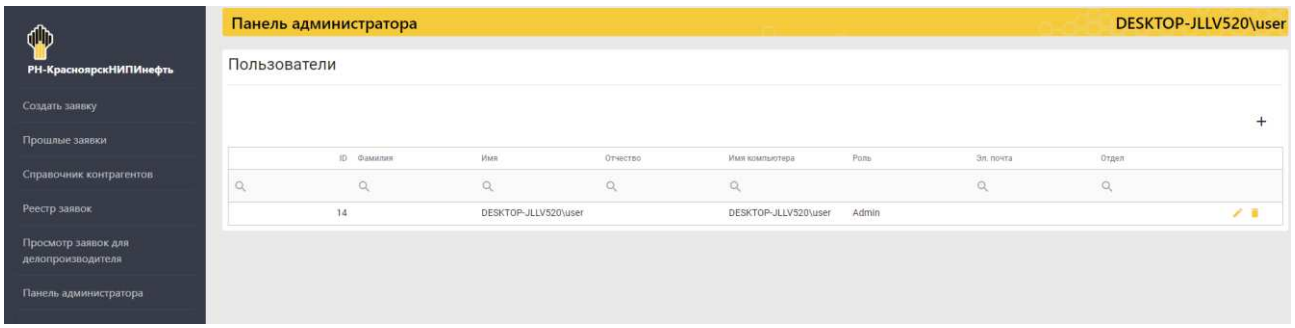


Рисунок 14 – Список пользователей

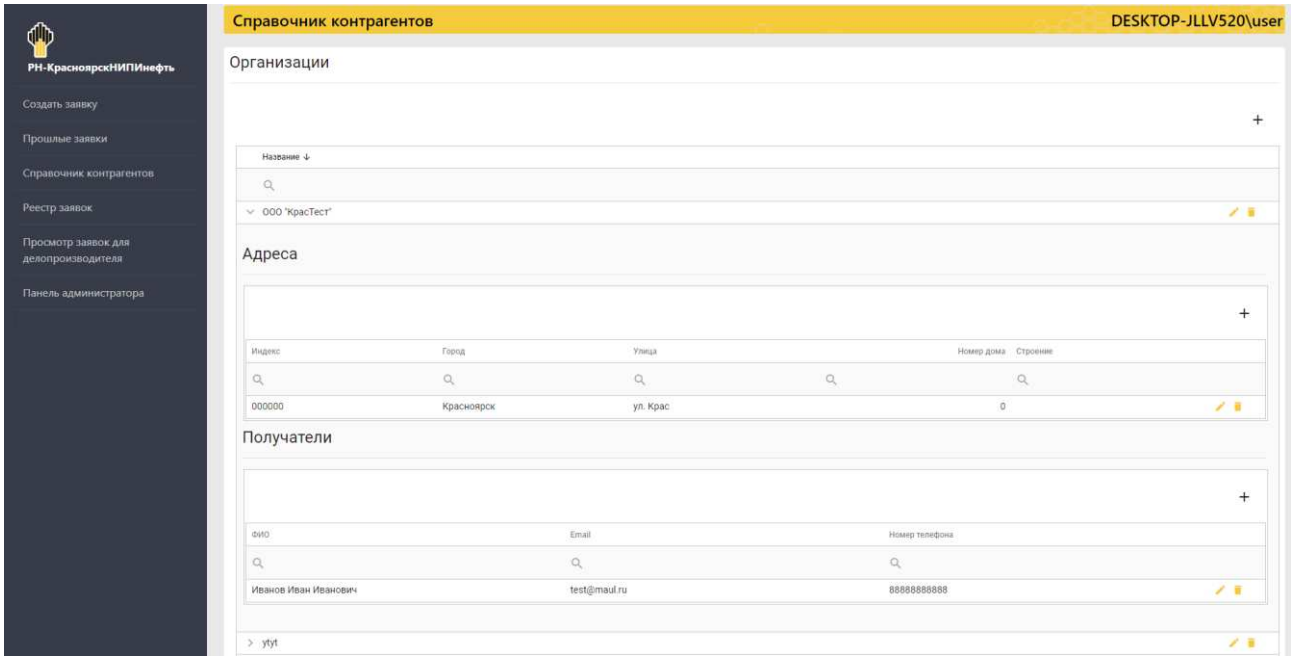
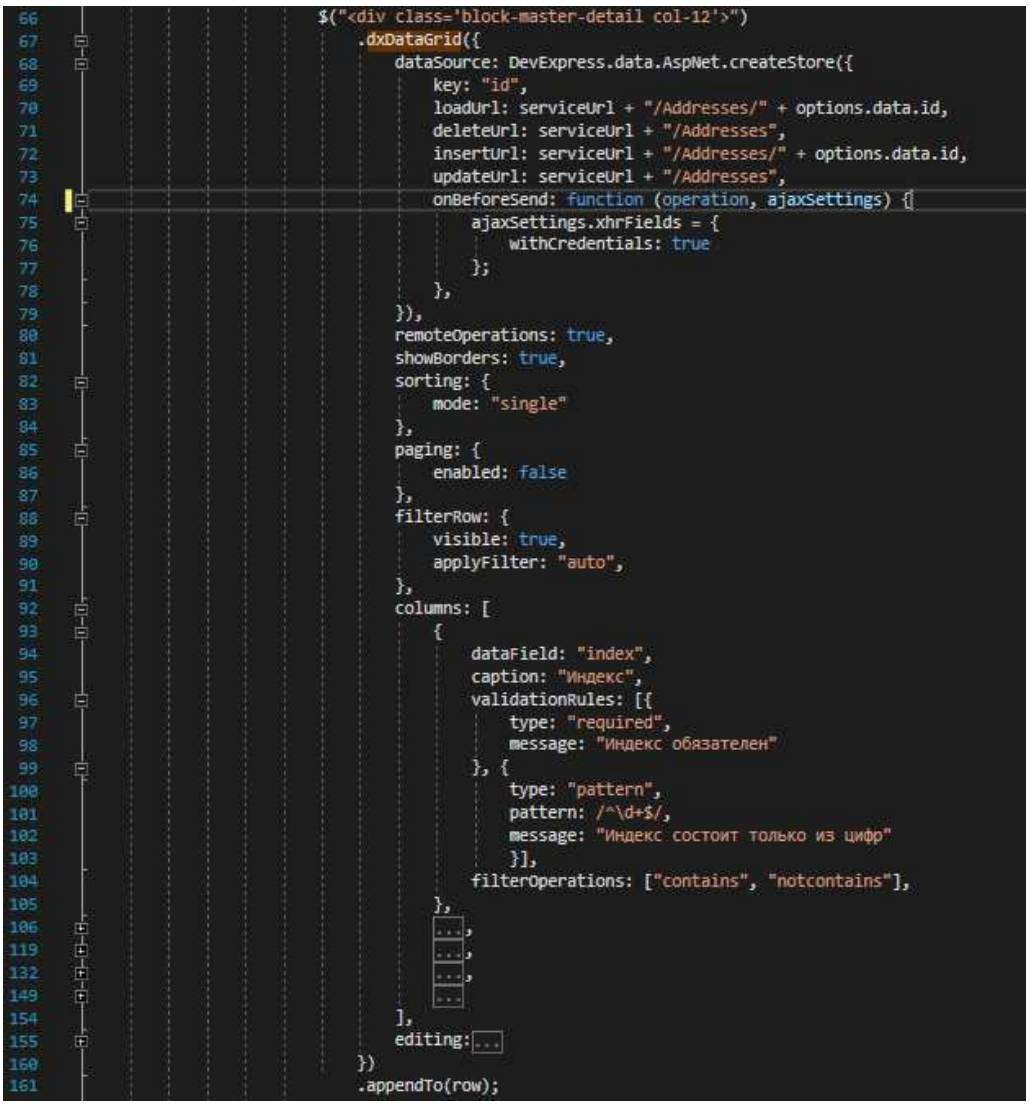


Рисунок 15 – Справочник контрагентов

Для добавления данного виджета при помощи фреймворка jQuery, используется функция dxDataGrid (рисунок 16), принимающая объект с огромным количеством свойств.

Forms - это виджет, представляющий данные в виде набора пар метка-редактор. Данный виджет был использован для создания заявок (рисунок 17).

Editors - это набор виджетов, который состоит из более чем 15 различных компонентов. В данной работе было использовано множество из них, но хочется особо отметить компонент dxSelectBox (рисунки 18, 19), который был использован в создании заявок и который используется для поиска, выбора и создания организаций, получателей и адресов.



```
$("<div class='block-master-detail col-12'>")
    .dxDataGrid({
        dataSource: DevExpress.data.AspNet.createStore({
            key: "id",
            loadUrl: serviceUrl + "/Addresses/" + options.data.id,
            deleteUrl: serviceUrl + "/Addresses",
            insertUrl: serviceUrl + "/Addresses/" + options.data.id,
            updateUrl: serviceUrl + "/Addresses",
            onBeforeSend: function (operation, ajaxSettings) {
                ajaxSettings.xhrFields = {
                    withCredentials: true
                };
            },
        }),
        remoteOperations: true,
        showBorders: true,
        sorting: {
            mode: "single"
        },
        paging: {
            enabled: false
        },
        filterRow: {
            visible: true,
            applyFilter: "auto",
        },
        columns: [
            {
                dataField: "index",
                caption: "Индекс",
                validationRules: [{
                    type: "required",
                    message: "Индекс обязателен"
                }, {
                    type: "pattern",
                    pattern: /^\d+$/,
                    message: "Индекс состоит только из цифр"
                    }],
                filterOperations: ["contains", "notcontains"],
            },
            ...,
            ...,
            ...,
            ...
        ],
        editing:...
    })
    .appendTo(row);
```

Рисунок 16 – Реализация Data Grid при помощи jQuery

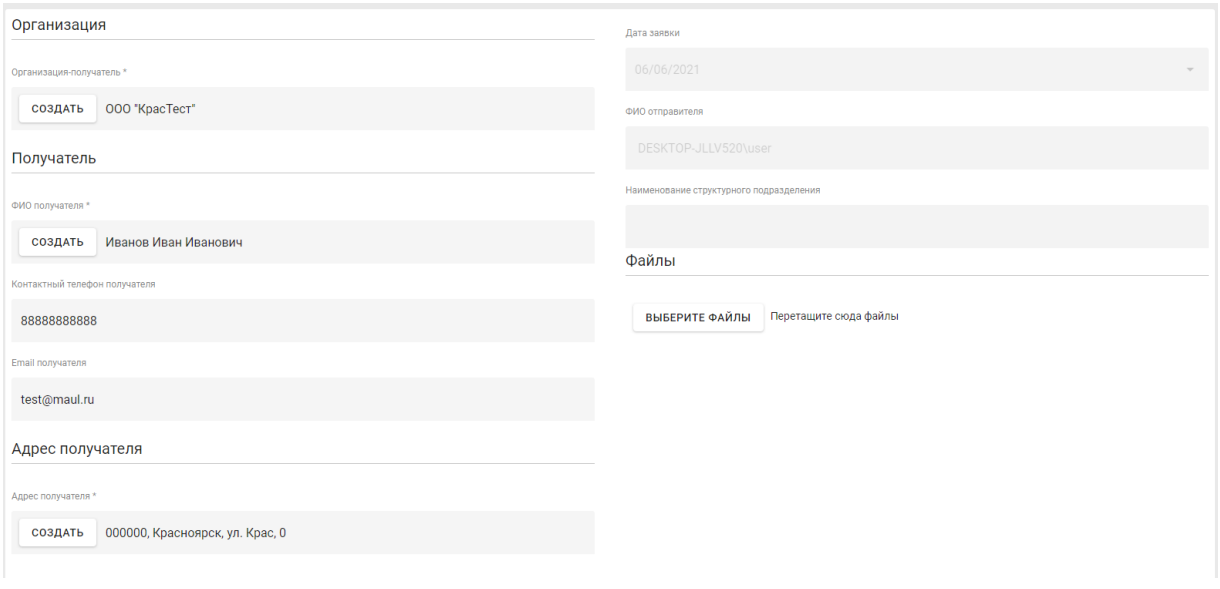


Рисунок 17 – Создание заявки

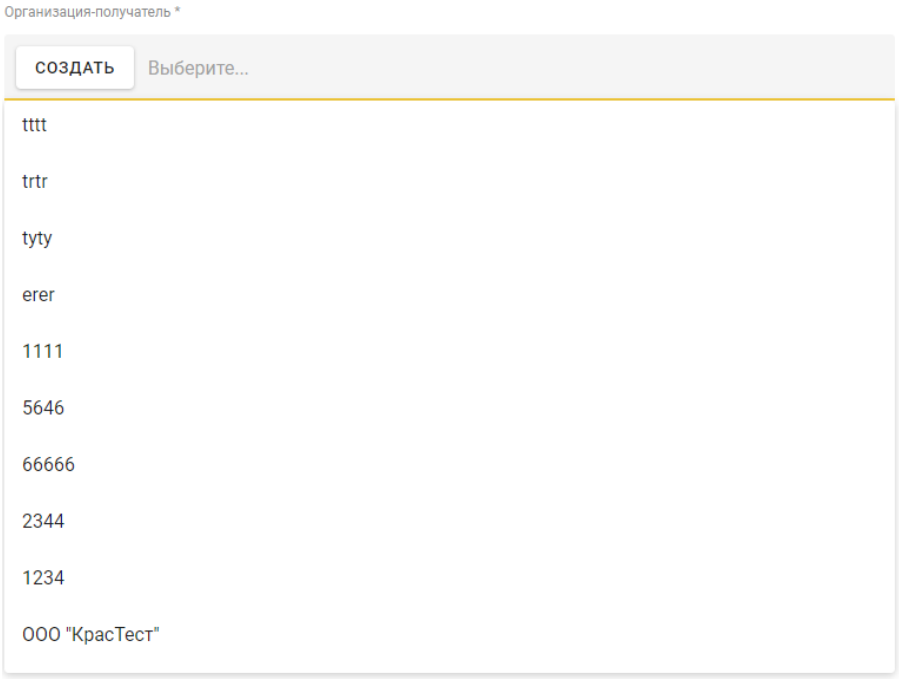


Рисунок 18 – Компонент dxSelectBox

```
{
    itemType: "group",
    caption: "Организация",
    items: [
        {
            dataField: "Organization",
            label: {
                text: "Организация-получатель"
            },
            editorType: "dxSelectBox",
            editorOptions: {
                dataSource: {
                    loadMode: 'raw',
                    load: function () {
                        var a = getOrganizations();
                        return a;
                    }
                },
                searchEnabled: true,
                placeholder: "Выберите...",
                displayExpr: "name",
                @if (User.Role() == "Clerk" || User.Role() == "Admin") {
                <text>buttons: [{
                    name: "create",
                    location: "before",
                    options: {
                        text: "Создать",
                        onClick: function () {
                            showCreateOrganizations();
                        }
                    }
                }],</text>
                }
                onValueChanged: function (e)...,
            },
            validationRules: [{
                type: "required",
                message: "Организация-получатель обязательна"
            }]
        },
    ]
}, {
```

Рисунок 19 – Реализация dxSelectBox

File Management – это также набор виджетов, но состоящий только из двух компонентов. В данной работе был использован компонент File Uploader (рисунки 20 и 21), для прикрепления пользователями файлов к заявке при ее создании.

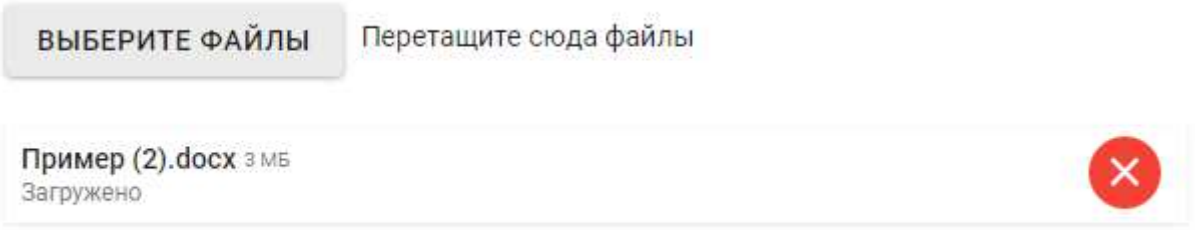


Рисунок 20 – Компонент File Uploader

```
{
    dataField: "Files",
    label: {
        visible: false
    },
    colSpan: 5,
    editorType: "dxFileUploader",
    editorOptions: {
        multiple: true,
        name: "files[]",
        selectButtonText: "Выберите файлы",
        labelText: "Перетащите сюда файлы",
        allowCanceling: true,
        uploadMode: "instantly",
        readyToUploadMessage: "Готов к загрузке",
        onUploaded: function (e)...,
        onContentReady: function (e)...

    }
}
```

Рисунок 20 – Реализация File Uploader

Dialogs – это наборы виджетов, позволяющий реализовать всплывающие окна (рисунок 21). Один из его компонентов, Popup, был использован в создании заявок при создании организации (рисунок 22), получателя (рисунок 23) и адреса (рисунок 24).

```
function showCreateOrganizations() {
    var container = $("<div />");
    var popupOptions = {
        width: "80%",
        height: "80%",
        contentTemplate: function () {
            var result = $('<form action="@Model.BackendConnection/api/Organizations" method="post" />');

            var form = $("<div />").dxForm({
                readOnly: false,
                items: [{
                    itemType: "group",
                    items: [
                        {
                            dataField: "name",
                            label: {
                                text: "Название"
                            },
                            validationRules:...
                        }]
                    }, ...]
                });
                result.append(form);
                result.on...;
                result.dxScrollView...;
                return result;
            },
            showTitle: true,
            title: "Создание организации",
            visible: false,
            dragEnabled: false,
            closeOnOutsideClick: true
        };
        popup = null;

        if (popup) {
            popup.option("contentTemplate", popupOptions.contentTemplate.bind(this));
        } else {
            popup = $("<div />").dxPopup(popupOptions).appendTo("#popup").dxPopup("instance");
        }

        popup.show();
    };
```

Рисунок 21 – Реализация Dialogs

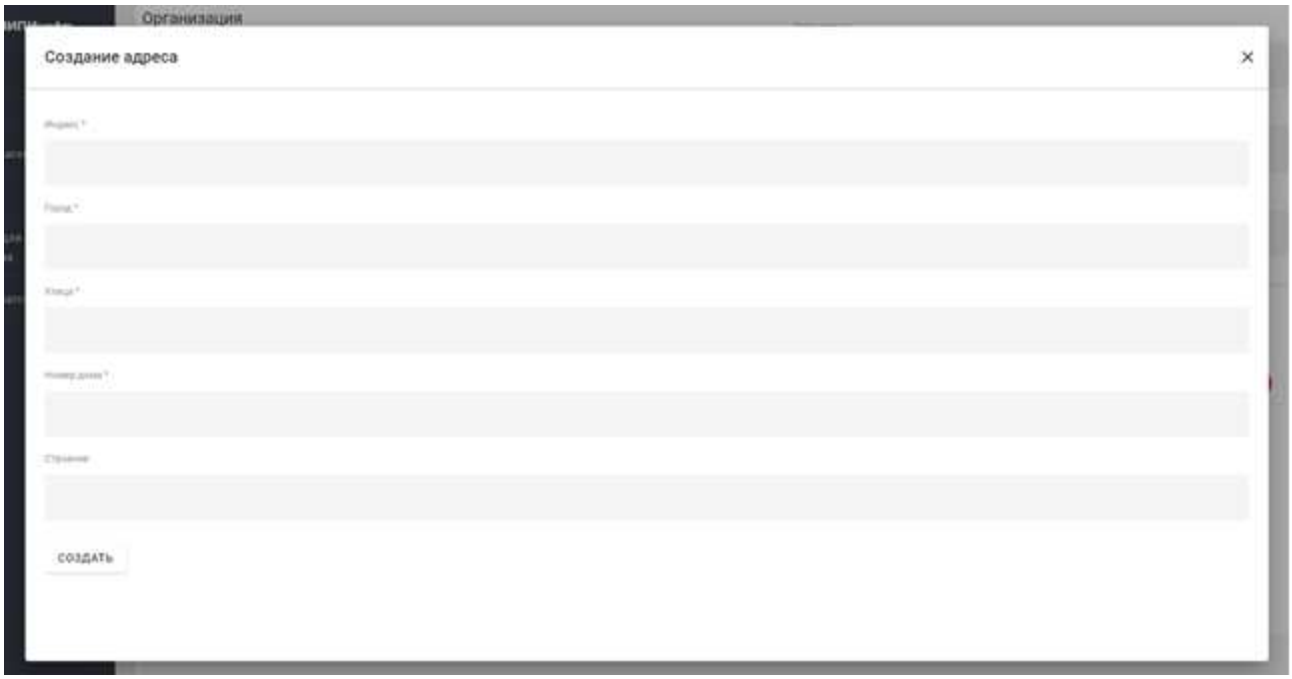


Рисунок 22 – Создание организации при помощи виджета Dialogs

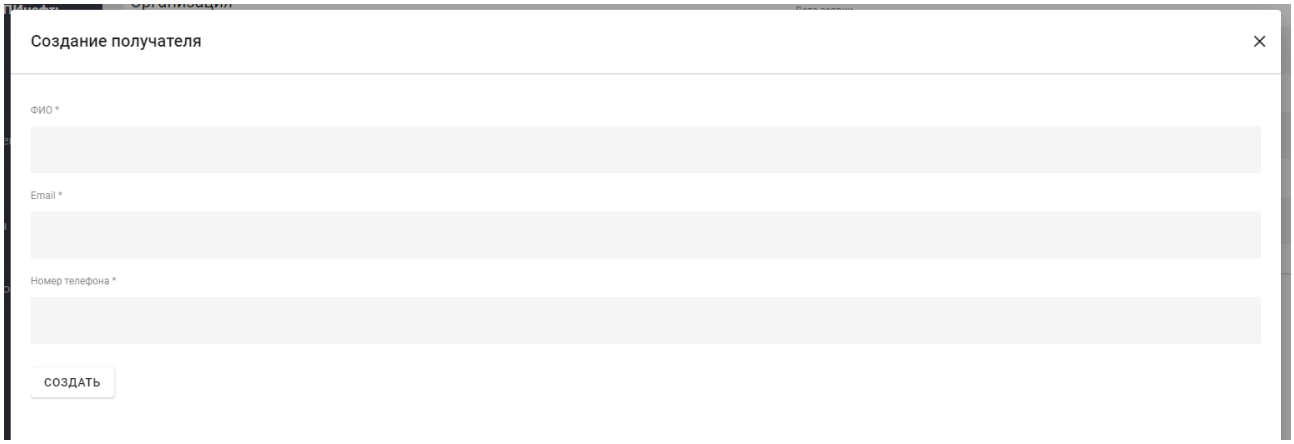


Рисунок 23 – Создание получателя при помощи виджета Dialogs

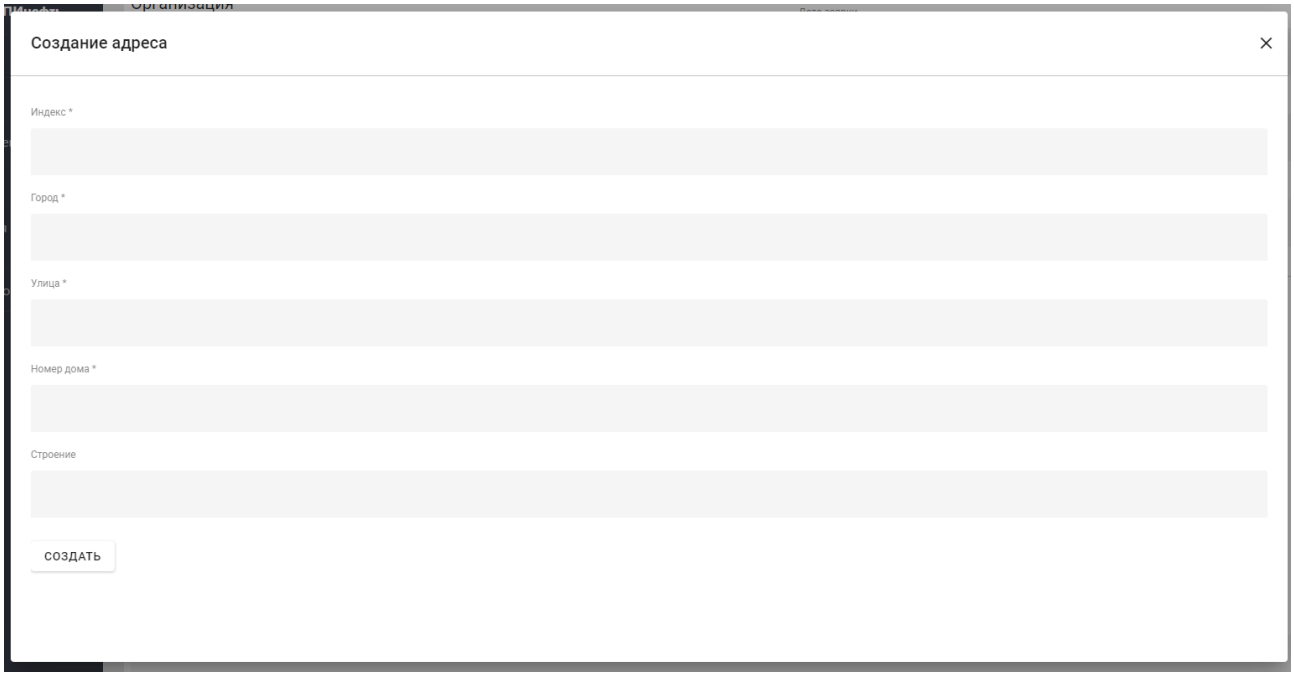


Рисунок 24 – Создание адреса при помощи виджета Dialogs

Из набора Dialogs также использовался компонент Confirm при вводе трек-номера в реестре заявок (рисунок 25). Этот компонент используется для отображения диалогового окна, в котором подтверждают, что пользователь хочет сохранить сделанные им изменения. Ведь после ввода трек-номера будут отправлены электронные письма отправителю и получателю.

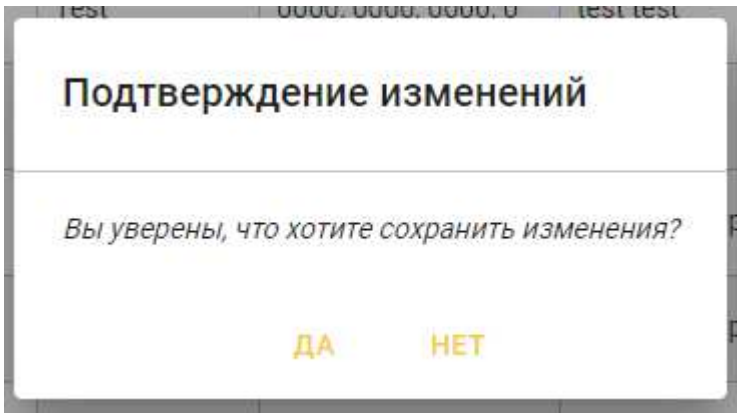


Рисунок 25 – Диалоговое окно при вводе трек-номера

3.4 Тестирование

Было проведено usability-тестирование, с целью определить, удобен ли интерфейс модуля. Для данного тестирования были подключены ресурсы заказчика. Данное тестирование выявило следующие проблемы:

- исправить нелогичность в панели администратора. Сейчас сначала имя, потом фамилия и отчество, нужно изменить порядок на ФИО (рисунок 26);

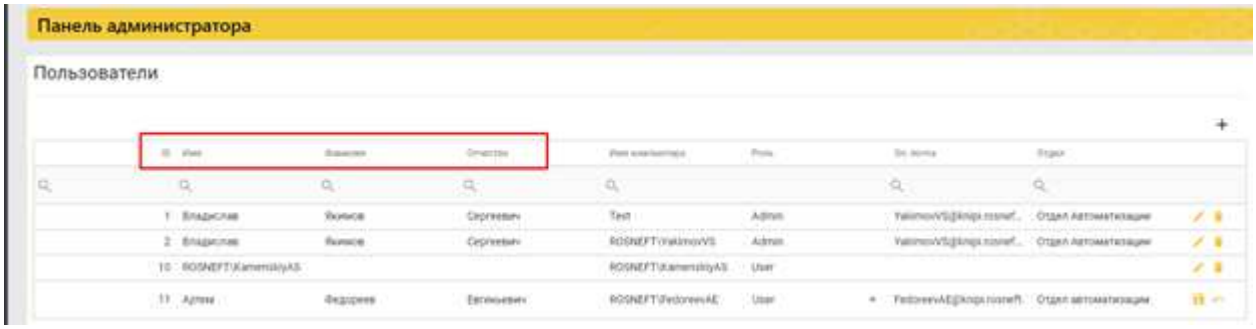


Рисунок 26 – Порядок ФИО

- требуется дать возможность пользователям редактировать поля получателя после его выбора из справочника в создании заявки (рисунок 27);

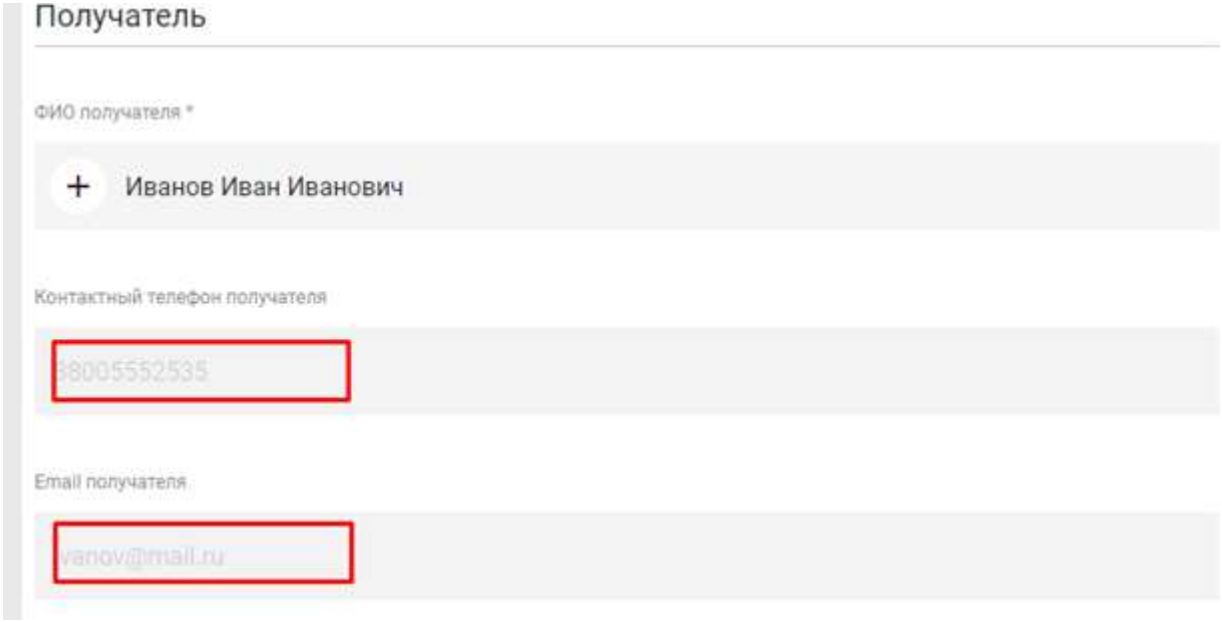


Рисунок 27 – Поля получателя

- дополнить предупреждения при вводе номера дома, тем что нельзя вводить текст (рисунок 28);

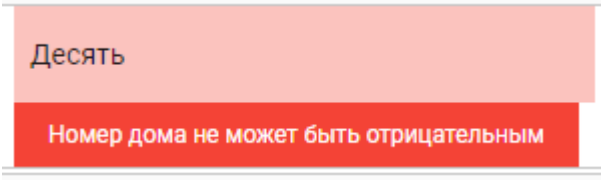


Рисунок 28 – Предупреждение в поле “Номер дома”

- дать возможность пользователю в поле номер телефона вводить не номер телефона, а просто вводить текст, так как бывает, что требуется указать добавочный номер (рисунок 29);

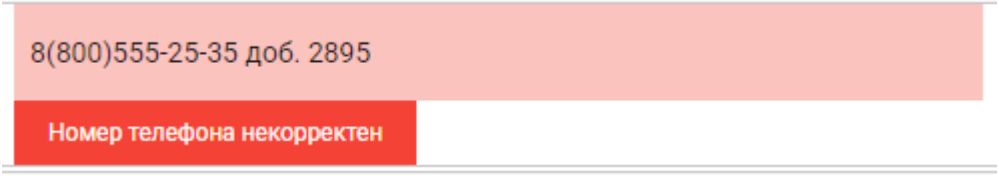


Рисунок 29 – Поле “Номер телефона”

- функционал по раскрыванию подробностей сделать не на нажатие кнопки «...», а на саму строчку заявки (рисунок 30);

Рисунок 30 – Раскрытие подробностей заявки

- убрать лишние фильтры и оставить только самые нужные, чтобы разгрузить интерфейс (рисунок 31);

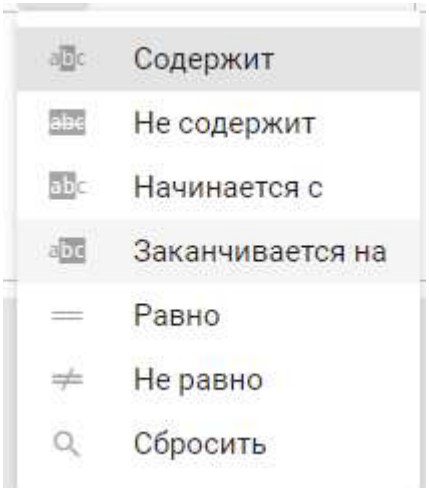


Рисунок 31 – Список фильтров

- в «Реестре заявок» при нажатии на Enter отбрасывает в левую часть страницы и приходится скроллить вправо;

- в «Реестре заявок» при изменении видимых столбцов, панель выбора столбцов частично, а иногда и полностью пропадает в правой части (рисунок 32);

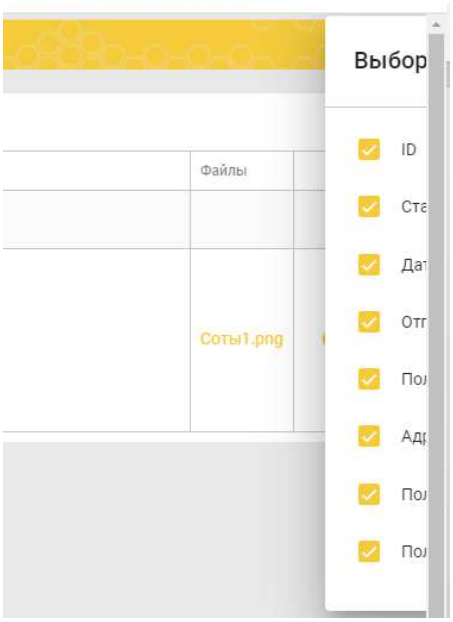


Рисунок 32 – Пропавшая панель выбора

- при создании заявки, когда создается новая запись в справочнике, подставлять созданную запись в соответствующее поле автоматически;

- при прикреплении к заявке нескольких файлов, нет возможности удалить отдельные файлы, есть только возможность удалить все заявки (рисунок 33);

- в «Реестре заявок» сделать по умолчанию вывод вначале новых заявок;

- изменить кнопку «+» во вкладке «Создать заявку» на «Создать», так как это будет более понятнее (рисунок 34).

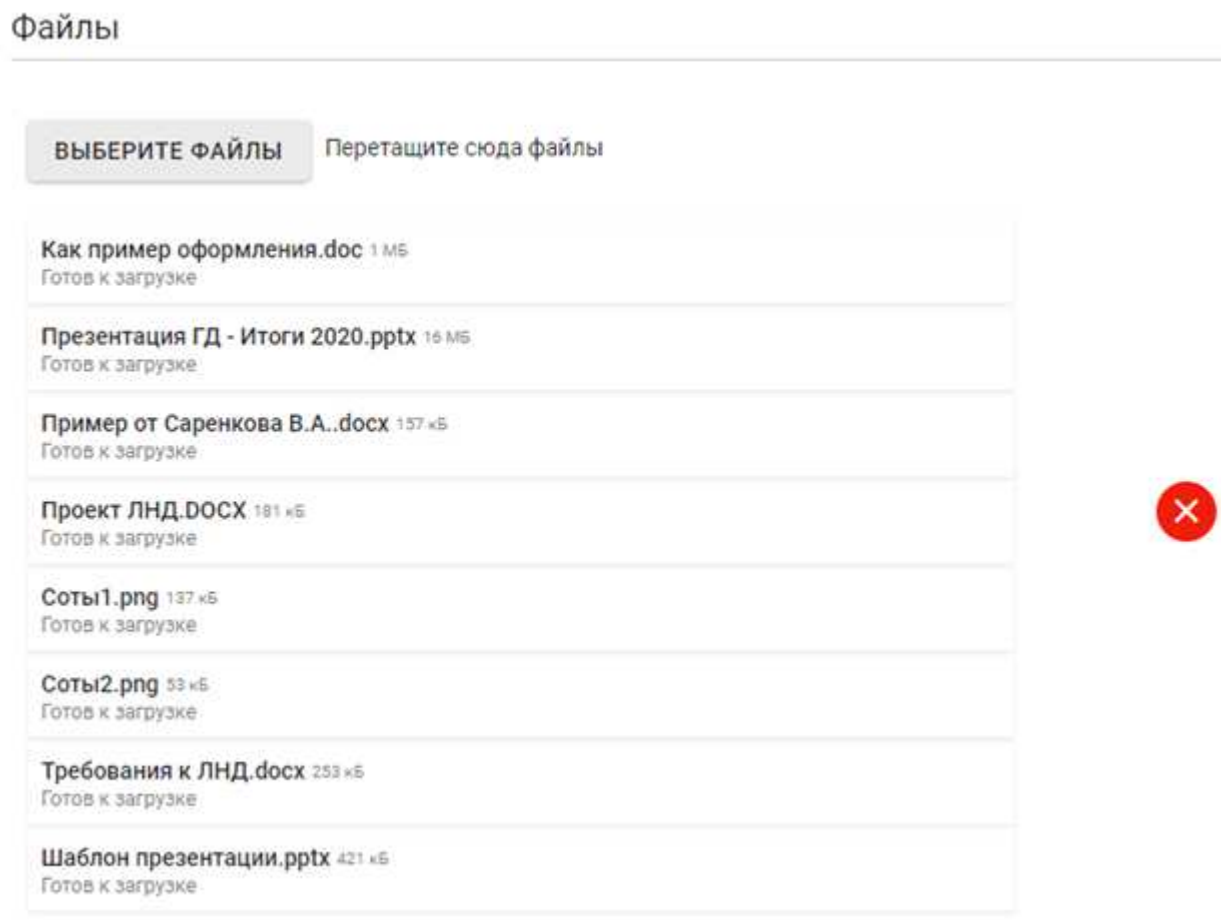


Рисунок 33 – Прикрепление файлов к заявке

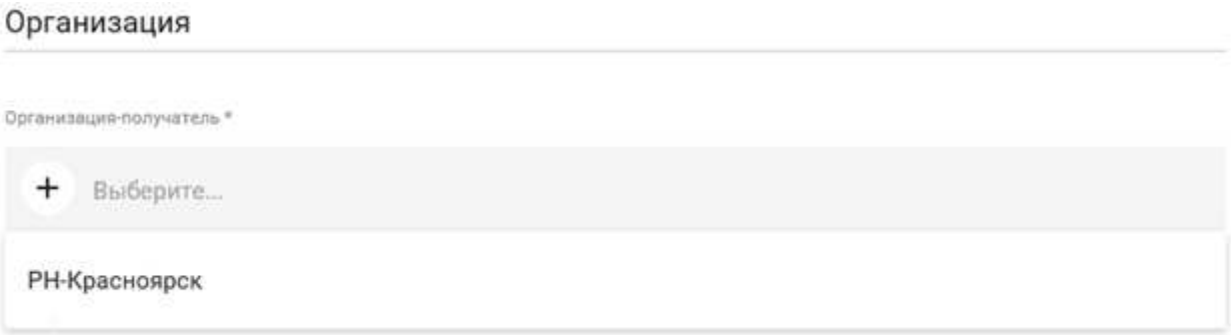


Рисунок 34 – Кнопка для создание новой записи в справочник

3.5 Выводы

В первом подразделе была разобрана архитектура клиентского модуля вместе с его структурой. Также был описан функционал разработанной системы: роли, справочники и заявки.

Следующий подраздел был посвящен механизму аутентификации и

авторизации пользователя, в котором рассказывалось о Windows-аутентификации, а также об объектах Claim. Далее был подробно разобран реализованный графический интерфейс вместе с набором использованных виджетов DevExtreme.

Последним этапом было проведенное usability-тестирование на конечных пользователях. В результате тестирования обнаружились недочеты, впоследствии исправленные.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была достигнута цель и решены все поставленные задачи, а именно:

- определен технологический стек;
- разработана исходная концепция;
- подготовлено предварительное техническое решение;
- подготовлен предварительный план работ;
- спроектирована архитектура будущего приложения;
- спроектирован интерфейс системы;
- разработана спроектированная система;
- протестирована клиентская часть приложения.

Кроме этого, были получены практические навыки: освоен фреймворк DevExtreme и углублены знания в фреймворке jQuery. Также появился опыт в организации рабочего процесса разработки посредством различных сервисов для коммуникации, в реализации взаимодействия клиентской и серверной частей системы, а также в организации работы с заказчиком.

Как итог, была разработана клиентская часть автоматизированной системы формирования заявок на отправку корреспонденции с использованием ASP.NET Core. Данная система позволит облегчить и ускорить прием и отправку корреспонденции.

Разработанная система была передана заказчику и была оценена им достаточно высоко (Приложение А).

СПИСОК СОКРАЩЕНИЙ

БД – база данных;

КС – курьерская служба;

ПО – программное обеспечение;

ФИО – фамилия, имя, отчество;

CQRS – command and query responsibility segregation;

CSS – cascading style sheets;

HTML – hypertext markup language;

HTTP – hyperText transfer protocol;

MVC – Model-View-Controller;

SOA – service-oriented architecture;

SQL – structured query language.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Либерти Д. Язык программирования C# // Программирование на C#. – Санкт-Петербург. – 2003: Символ-Плюс, 2003. – 688 с. – ISBN 5-93286-038-3.

2. Скит, Дж. C# для профессионалов. Тонкости программирования / Дж. Скит. 3-е изд. – СПб.: Питер, 2014. – 602 с.

3. Чамберс, Дж. ASP.NET Core. Разработка приложений / Дж. Чамберс, Д. Пэкетт, С. Тиммс. – Санкт-Петербург : Питер, 2018. – 464 с.

4. Прайс М. C# 7 и платформа .Net core. / Прайс М. 3-е изд. – СПб.: Питер, 2017. – 640 с.

5. Роббинс Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс; [пер. с англ. М. А. Райтман]. – 4-е издание. – М. : Эксмо, 2014. – 528 с.

6. Дакетт Д. Javascript и jQuery. Интерактивная веб-разработка. – Эксмо, 2017. – 640 с.

7. Жизненный цикл ПО. Каскадная модель (Waterfall) - XB Software: [Электронный ресурс]. URL: https://xbsoftware.ru/blog/zhiznennyj-tsykl-po-kaskadnaya-model-waterfall/. (Дата обращение 10.06.2021).

8. Итеративная модель разработки: [Электронный ресурс]. URL: https://web-creator.ru/articles/iterative_development (дата обращения: 10.06.2021).

9. Trello: [Электронный ресурс]. URL: https://trello.com/. (дата обращения: 10.06.2021).

10. Адам Фримен. ASP.NET Core MVC 2 с примерами на C# для профессионалов, 7-е изд.: Пер. с англ. – СПб.: ООО “Диалектика”, 2019. – 1008 с.

11. DevExtreme - JavaScript UI Components for Angular, React, Vue and jQuery by DevExpress: [Электронный ресурс]. URL: https://js.devexpress.com. (дата обращения: 10.06.2021).

12. Грин, Д. Постигая Agile. Ценности, принципы, методологии / Д. Грин, Э. Стеллман. – Манн, Иванов и Фербер. – 2018. – 448 с.

13. Кон, М. Scrum: гибкая разработка ПО / М. Кон. – М. : ООО «И.Д. Вильямс», 2011. – 576 с.

14. Кусов, А. А. Проблемы интеграции корпоративных информационных систем / А. А. Кусов // Управление экономическими системами. – 2011. – № 28. – С. 103.

15. Балдин, А. О понятии «Архитектура системы» / А. В. Балдин, А. Н. Данчул // Электронный научно-технический журнал «Инженерный вестник». – 2012. – № 6.

16. Коржов, В. Многоуровневые системы клиент-сервер / В. Коржов // Открытые системы. – 1997. – Режим доступа: https://www.osp.ru/nets/1997/06/142618.

17. Krafzig, D. Enterprise SOA: Service-oriented Architecture Best Practices / D. Krafzig, K. Banke, D. Slama. – Prentice Hall Professional, 2005. – 382 p.

18. Настройка проверки подлинности Windows в ASP.NET Core | Microsoft Docs [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/security/authentication/windowsauth (дата обращения: 10.06.2021).

19. Введение в Razor Pages в ASP.NET Core | Microsoft Docs [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/razor-pages (дата обращения: 10.06.2021).

20. Overview: DevExtreme - JavaScript UI Components for Angular, React, Vue and jQuery by DevExpress: [Электронный ресурс]. URL: https://js.devexpress.com/Documentation/. (дата обращения: 10.06.2021).

ПРИЛОЖЕНИЕ А

Акт внедрения

АКТ

вводе информационной системы в промышленную эксплуатацию

Мы, нижеподписавшиеся члены приемной комиссии, составили настоящий Акт о том, что:

1. Автоматизированная система формирования заявок на отправку корреспонденции (разработчики: Андрианов Владислав Сергеевич и Якимов Владислав Сергеевич) полностью соответствует Техническому проекту и полностью удовлетворят требованиям Технического задания.
2. Работы по внедрению и подготовке информационной системы к вводу в промышленную эксплуатацию выполнены в полном объеме.

Председатель приемной комиссии:

Начальник отдела автоматизации — А.В. Саренков
«01» июня 2021 г.

Члены приемной комиссии:

Главный специалист — О.В. Ткачук
«01» июня 2021 г.

Ведущий инженер — А.А. Шпилевой
«01» июня 2021 г.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«16» 06 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Разработка автоматизированной системы формирования заявок на отправку
Тема

корреспонденции: модуль клиентской части

Руководитель ВКР _____ 16.06.21 доцент, канд. техн. наук А. В. Хныкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 16.06.21 В. С. Якимов
подпись, дата инициалы, фамилия

Консультант _____ 16.06.21 старший преподаватель А. С. Михалев
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2021