МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой, д.т.н., с.н.с.
_____/ Б.М. Суховилов /
«_____»_____2021 г.

Электронная торговая площадка для купли-продажи автозапчастей

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2021.301/50.ВКР

Руководитель, ст. преподаватель
_____/ С.Ю. Нестеренко /
«_____» _____2021 г.

Автор
студент группы ЭУ – 402
/ Т.Н. Мухамадеев /
«_____» _____2021 г.

Нормоконтролер, доцент
/ Е.А. Конова /
«_____» _____2021 г.

Челябинск 2021

АННОТАЦИЯ

Мухамадеев Т.Н. Разработка электронной торговой площадки для купли-продажи автозапчастей. - Челябинск: ЮУрГУ, ЭУ-402, 2021. - 65с., 19 ил., 22 табл., библиографический список – 4 наим.

Дипломный проект выполнен с целью разработки электронной торговой площадки для купли-продажи автозапчастей.

В дипломном проекте выполнен анализ существующих решений в области электронной торговой площадки для купли-продажи автозапчастей. Произведен сбор информации о современных технологиях разработки сайтов.

Представлен поэтапный процесс разработки структуры веб-ресурса. Дано ознакомительное описание используемых средств разработки в проекте.

Объектом исследования является способы организации торговли в сети Интернет. Предмет исследования – электронная торговая площадка.

Цель работы – разработка электронной торговой площадки для купли-продажи автозапчастей.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Разработанный веб-сайт является посредником между дилерами автозапчастей и покупателями. Это организовывает процесс торговли в Интернет-пространстве.

Цель работы – разработка электронной торговой площадки для купли-продажи автозапчастей.

Задачи работы:

- проанализировать современные технологии в разработке веб-сайтов;
- выполнить обзор существующих решений выбранной области;
- разработать электронную торговую площадку для купли-продажи автозапчастей;
- описать процесс создания проекта поэтапно.

Объект работы –электронной торговой площадки.

Результаты работы рекомендуется использовать при дальнейшей разработке проекта, действующего на территории РФ.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Тема выпускной квалификационной работы

Электронная торговая площадка купли-продажи автозапчастей.

Необходимо разработать полноценную клиент-серверную систему, пользователи которой могут продавать или покупать автозапчасти в зависимости от их роли. Могут быть следующие роли: дилер и оптовик.

Дилер имеет возможность редактировать список своих клиентов - оптовиков, загружать прайс-листы и справочники для дальнейшего использования и редактирования.

Клиент имеет возможность просматривать список доступных для покупки автозапчастей, добавлять их в корзину и оформлять заказы.

Также пользователи обоих ролей имеют возможность ознакомиться с текущими акциями и посмотреть отчеты за прошедшие периоды

Диаграмма, наглядно отображающая структуру сайта, представлена на рисунке 1

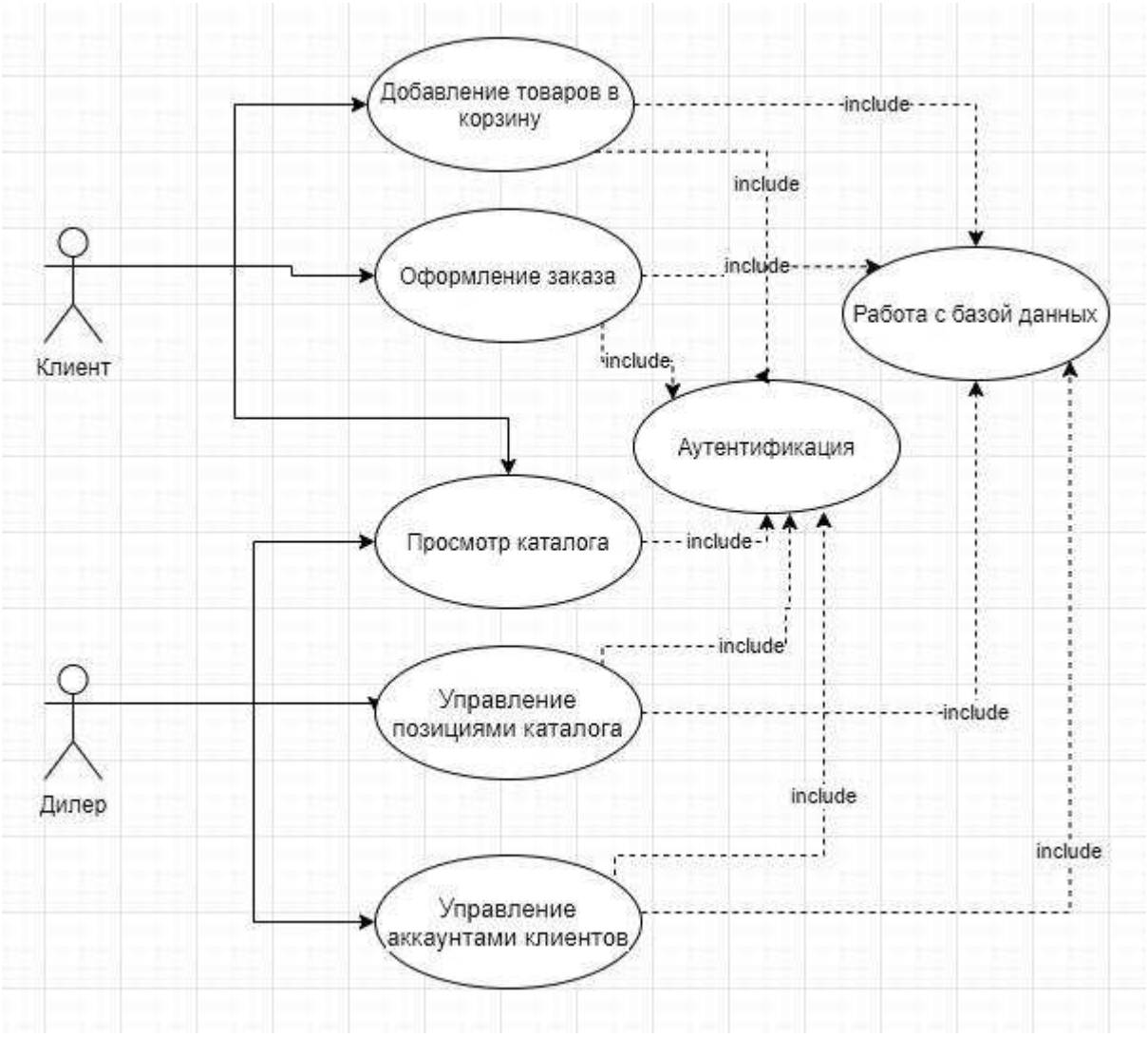

Рисунок 1 – Диаграмма прецедентов «Структура сайта»

1.2 Обзор существующих решений и сравнительный анализ

1.2.1 OSZZ.RU

Электронная торговая площадка автозапчастей и автотоваров, объединяющая поставщиков и покупателей. Это оптовый поставщик автозапчастей для иномарок и логистический оператор. Главная страница сайта представлена на рисунке 2.

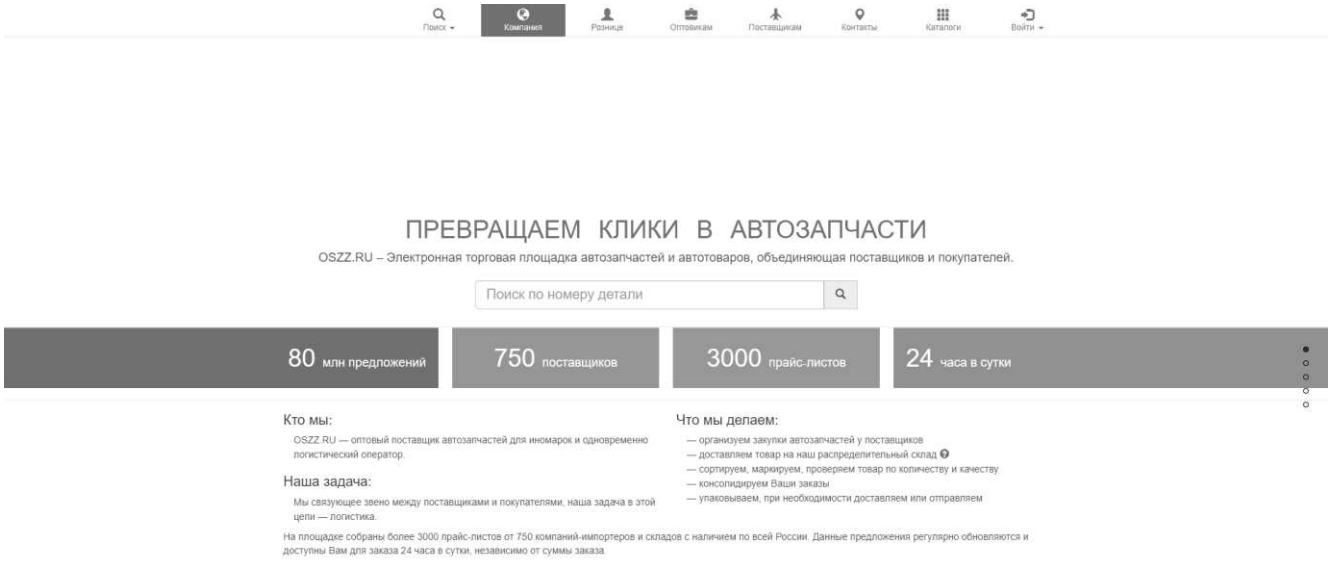

Рисунок 2 –Главная страница OSZZ.RU

На площадке собраны более 3000 прайс-листов от 750 компаний-импортеров и складов с наличием по всей России. Она выполняет следующие функции:

- организация закупки автозапчастей у поставщиков;
- доставка товара на распределительный склад;
- сортировка, маркировка, проверка товара по количеству и качеству;
- консолидация заказов;
- упаковка, при необходимости доставка.

На сайте можно зарегистрироваться в роли оптового покупателя или розничного. Для зарегистрированного розничного покупателя во всех пунктах

выдачи устанавливается единая цена, оформить заказ можно прямо на сайте и отслеживать статус заказа, в то время, как для незарегистрированного розничного покупателя устанавливается цена в каждом пункте выдачи индивидуально, детали можно приобрести только в рабочее время и отслеживать статус заказа можно только по телефону.

Схема работы с оптовиками представлена в таблице 1.

Таблица 1 – Схема работы с оптовиками

Схема работы с оптовиками	Оптовик партнера	Оптовик	Партнер
Торговые и финансовые взаимоотношения	С региональным партнером	С ООО "Авто-Стратегия" (ЦС-Москва)	С ООО "Авто-Стратегия" (ЦС-Москва)
Оформление заказов	на сайте OSZZ.RU	на сайте OSZZ.RU	на сайте OSZZ.RU
Получение товара	Доставка до двери	Самовывоз, доставка до ТК в Москве	Самовывоз, доставка до ТК в Москве
Оплата за товар	При получении в регионе	Предоплата или оплата по факту	Отсрочка
Оплата услуг ТК	Без оплаты	Оптовиком	Региональным клиентом
Ежемесячный оборот	Без ограничений	Без ограничений	От 3млн р. в месяц
Добавление подклиентов на сайте	Нет	Есть	Есть
Подключение складов для ваших подклиентов	Нет	1 прайс-лист	Без ограничений
Добавление пунктов выдачи OSZZ.RU на карту	Нет	Только розничные	Розничные и оптовые
Отображение на карте и в списке партнеров	Нет	Нет	Да
Уровень цен	По договоренности с партнером	Согласно таблице скидок	Цена регионального партнера
Специальные требования	Нет	Нет	Да

Также можно зарегистрироваться как поставщик.

Схема работы с поставщиками выглядит следующим образом:

– регулярная отправка прайс-листа на почту электронной торговой площадки, который автоматически загружается в систему;

– по расписанию система в автоматическом режиме отправляет заказ поставщику;

– поставщик присылает ответ на данный заказ, с информацией об отказах и изменениях;

– ожидание груза и счет-фактуры в электронном виде;

– приходование товара, проверка его количества и качества;

– система автоматически проводит оплату.

Чтобы стать поставщиком, необходимо:

– юридическое лицо;

– сертификаты на товар;

– счет-фактура в электронном виде (xls, txt);

– прайс-лист в электронном виде (xls, txt, веб-сервис);

– конкурентные цены;

– работа с НДС.

1.2.2 Российская Торговая Система 21 века

Система онлайн покупки и продажи запчастей для любой техники. Главная страница сайта представлена на рисунке 3.

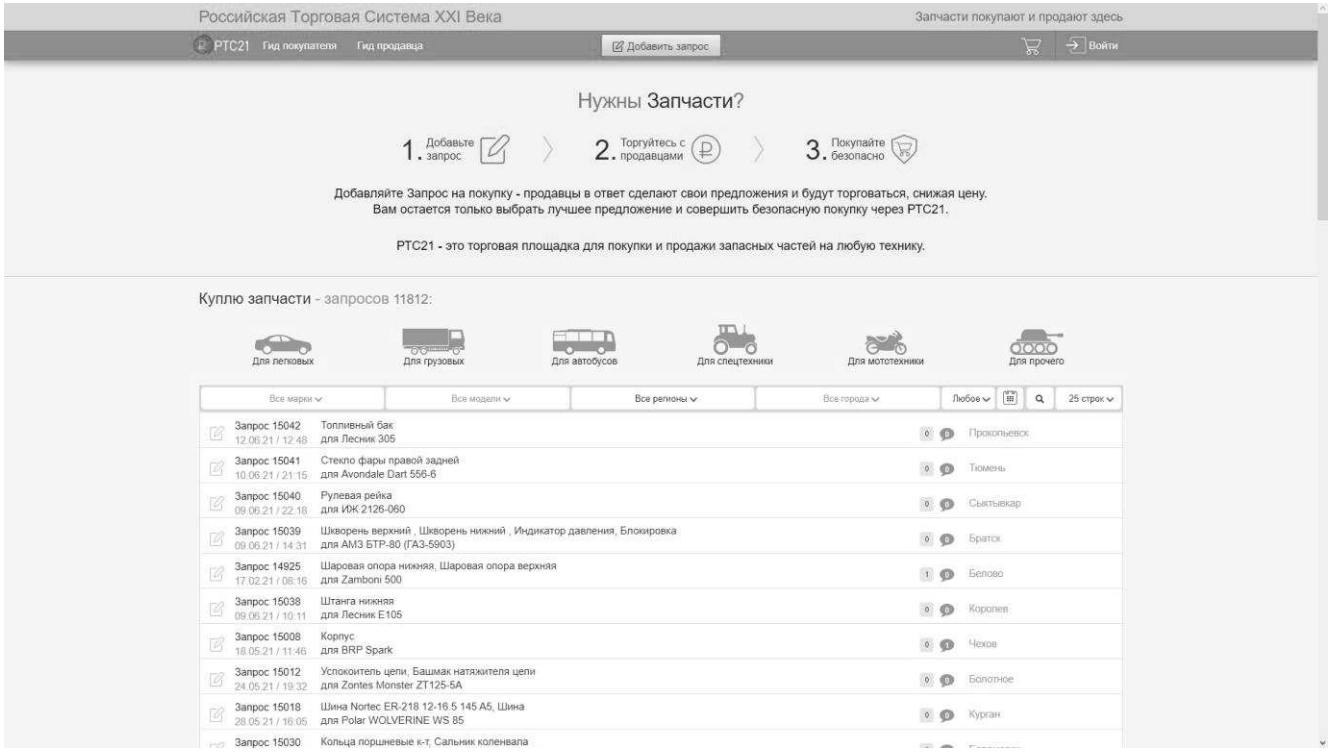

Рисунок 3 – Главная страница Российская Торговая Система 21 века

Зарегистрироваться могут как покупатели, так и продавцы

После регистрации покупатель может оформлять запросы, в которых должен указать описание техники, для которой нужны запчасти и список нужных запчастей.

Поставщиками могут быть как компании, так и юридические лица. Они могут отвечать на запросы покупателей и предлагать свою цену.

Покупатели могут выбирать из предложений наиболее выгодные для себя и заключать сделки, при этом указывая способ доставки и оплачивая выбранное предложение. После заключения сделки покупатель может общаться с продавцом через чат, оставить на него отзыв и отозвать заказ, если продавец не выполняет условия сделки, и покупатель хочет отказаться от сделки и забрать свои деньги.

Продавец получает деньги за отправленный товар если не имеет претензий от покупателя в течение четырех дней после получения товара покупателем.

1.2.3 APR

Торговая площадка по продаже БУ автозапчастей с большим количеством компаний-продавцов. Главная страница сайта представлена на рисунке 4.

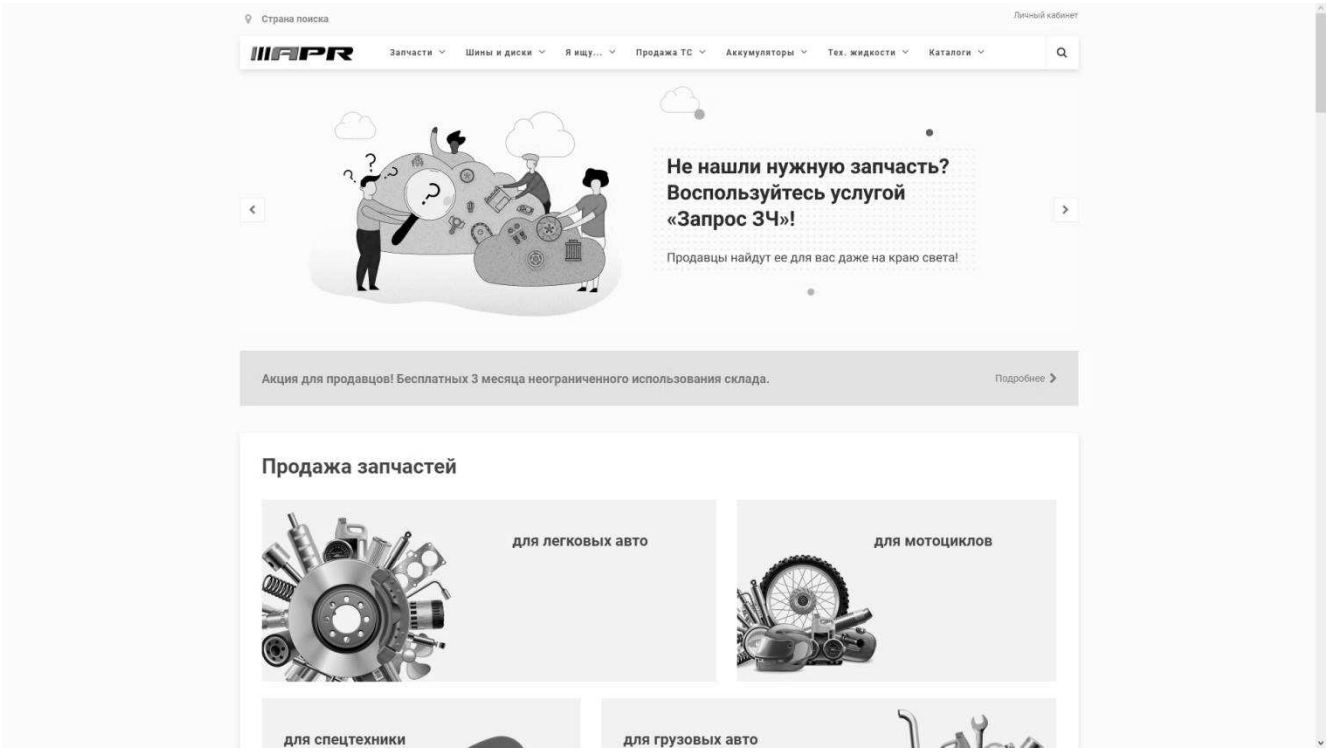

Рисунок 4 - Главная страница ARP

APR также позволяет зарегистрироваться как покупателям, так и продавцам, в том числе частным лицам для купли-продажи автозапчастей и тс.

Данные о результатах сравнительного анализа представлены в таблице 2.

Таблица 2 – Данные о результатах сравнительного анализа

	Регистрация физ. лиц как продавцов	Каталог товаров	Поиск по коду запчасти	Загрузка прайс-листов продавцов	Регистрация розничных покупателей
OSZZ.RU	нет	да	да	да	да
Российская Торговая Система 21 века	да	нет	нет	нет	да
APR	да	да	да	нет	да
Текущий	нет	да	да	да	нет

проект					

1.2.4 Выводы сравнительного анализа

Таким образом, проект является B2B электронной торговой площадкой, в которой продавцами являются зарегистрированные официальные представители дилеров, которые могут сами добавлять себе оптовиков и выставлять свои товары, а также регулировать собственные прайс-листы, в то время как оптовики могут безопасно покупать необходимые товары у проверенных поставщиков.

1.3 Выбор инструментов для реализации проекта

Приложение подразделяется на две части: клиентская и серверная. Серверная часть отвечает за хранение и обработку данных. Сохраняются данные пользователя – его логин, e-mail, пароль, и т.п.

При создании клиента сохраняются такие данные, как наименование пользователя, наименование организации, должность руководителя, а также банковские реквизиты и оптовая группа.

Для решения поставленных задач необходима база данных, сервер и веб-клиент. В качестве БД будет использоваться PostgreSQL — свободная реляционная система управления базами данных. Более подробно, об этом будет рассказано в соответствующем разделе.

Для разработки серверной части приложения используется бесплатный PHP веб-фреймворк Laravel. Данный фреймворк был выпущен под лицензией MIT и является одним из самых популярных фреймворков на протяжении многих лет.

Для разработки веб клиента используется фреймворк «Vue.js», JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Данный фреймворк активно развивается, и имеет структуру, наиболее удобную для восприятия и обучения его использованию начинающих разработчиков (низкий порог вхождения). Он легко интегрируется в проекты с использованием других JavaScript-библиотек.

Выводы по разделу 1

Обзор существующих проектов показал, что тема организации интернет-площадки для купли-продажи автозапчастей актуальна. Можно выделить ряд особенностей, характерных для данного направления:

– дизайн таких ресурсов должен быть максимально простым и удобным;

– наличие системы поиска по товарам и фильтры обязательны;

– приветствуется блок поэтапного создания магазина, лавки в помощь пользователю.

На основе данного материала можно спроектировать основную логику веб-сайта для купли-продажи автозапчастей.

Описаны используемые инструменты.

2 РЕАЛИЗАЦИЯ ПРОЕКТА

2.1 Структура базы данных

База данных загружена на хостинг и используется как API для принятия запросов и отправки ответов.

Схема данных изображена на рисунке 5.

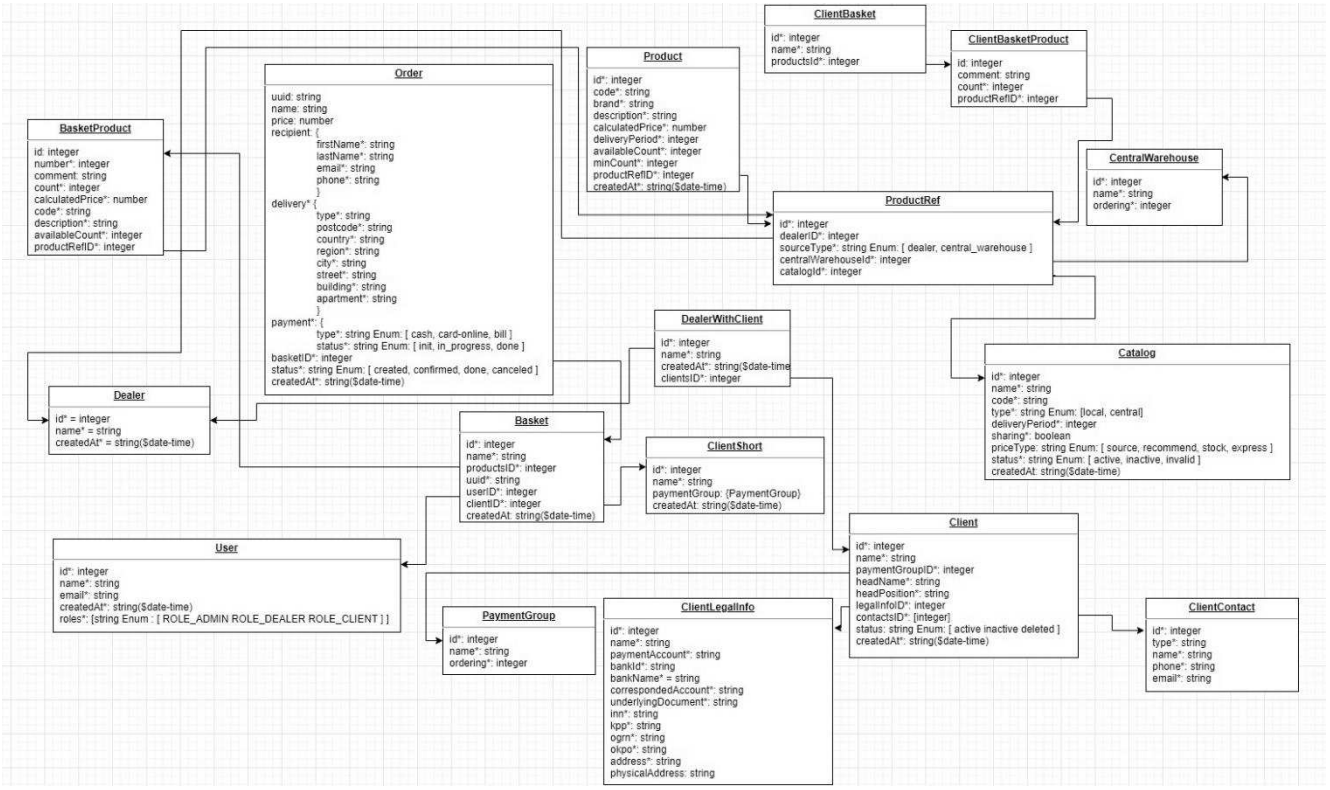

Рисунок 5 – Схема данных

Таблица User – это таблица, которая хранит информацию о клиенте. Подробное описание структуры таблицы дано в таблице 3.

Таблица 3 – Описание таблицы User

Название столбца	Тип	Описание
ID	int	id
name	string	Имя пользователя
email	string	Данные об оптовой группе клиента
createdAt	string($date-time)	Адрес электронной почты пользователя
roles	string	Роль пользователя

Таблица Client – это таблица, которая хранит информацию о клиенте. Подробное описание структуры таблицы дано в таблице 4.

Таблица 4 – Описание таблицы Client

Название столбца	Тип	Описание
ID	int	id
name	string	Имя пользователя
paymentGroupID	int	ID оптовой группы
headName	string	ФИО руководителя
headPosition	string	Должность руководителя
legalInfo	object	Реквизиты клиента
contacts	[int]	Контактная информация клиента
status	string Enum: [active inactive invalid]	Статус клиента

Таблица ClientLegalInfo – это таблица, которая хранит информацию о реквизитах клиента. Подробное описание структуры таблицы дано в таблице 5.

Таблица 5 – Описание таблицы ClientLegalInfo

Название столбца	Тип	Описание
ID	int	id
name	string	Наименование юридического лица
paymentAccount	string	Номер счета
bankId	string	БИК
bankName	string	Наименование банка
correspondedAccount	string	Корреспондентский счет
underlyingDocument	string	Документ-основание
inn	string	ИНН
kpp	string	КПП
ogrn	string	ОГРН
okpo	string	ОКПО
address	string	Юридический адрес
physicalAddress	string	Физический адрес

Таблица ClientContact – это таблица, которая хранит информацию о контактных данных клиента. Подробное описание структуры таблицы дано в таблице 6.

Таблица 6 – описание таблицы ClientContact

Название столбца	Тип	Описание
id	int	Объем двигателя автомобиля
type	string	Тип контакта
name	string	ФИО
phone	string	Телефон
email	string	Электронная почта

Таблица PaymentGroup – это таблица, которая хранит информацию об оптовой группе клиента. Подробное описание структуры таблицы дано в таблице 7.

Таблица 7 – описание таблицы PaymentGroup

Название столбца	Тип	Описание
id	int	
name	string	Наименование группы
ordering	int	Очередность группы

Таблица ProductRef – это таблица, которая хранит информацию об источнике товара. Подробное описание структуры таблицы дано в таблице 8.

Таблица 8 – описание таблицы ProductRef

Название столбца	Тип	Описание
id	int	
dealerID	int	ID дилера
sourcetype	string	Тип источника товара
centralWarehouseId	int	Номер центрального склада
catalogId	int	Номер каталога

Таблица Product – это таблица, которая хранит информацию о товарах. Подробное описание структуры таблицы дано в таблице 9.

Таблица 9 – описание таблицы Product

Название столбца	Тип	Описание
id	int	

окончание таблицы 9

Название столбца	Тип	Описание
code	string	Код товара
brand	string	Наименование бренда
description	string	Описание товара
calculatedPrice	number	Сумма товара
deliveryPeriod	int	Срок доставки
availableCount	int	Товаров в наличии
minCount	int	Минимальное количество товаров в наличии
productRefID	int	ID источника товара
createdAt	String($date-time)	Дата создания товара

Таблица ClientBasketProduct – это таблица, которая хранит информацию о товарах в корзине клиента. Подробное описание структуры таблицы дано в таблице 10.

Таблица 10 – описание таблицы ClientBasketProduct

Название столбца	Тип	Описание
id	int	Ключевое поле
comment	string	Комментарий о товаре
count	int	Количество товара
productRefID	int	ID источника товара

Таблица ClientBasket – это таблица, которая хранит информацию о товарах в корзине клиентов. Подробное описание структуры таблицы дано в таблице 11.

Таблица 11 – описание таблицы ClientBasket

Название столбца	Тип	Описание
id	int	Ключевое поле
name	string	Имя клиента
productsID	int	ID корзины товаров клиента

Таблица BasketProduct – это таблица, которая хранит полную информацию о товарах в корзине клиентов. Подробное описание структуры таблицы дано в таблице 12

Таблица 12 – описание таблицы BasketProduct

Название столбца	Тип	Описание
id	int	Ключевое поле
number	integer	Номер товара
comment	string	Комментарий к товару
count	int	Количество товара
calculatedPrice	number	Сумма товара
code	int	Код товара
description	string	Описание товара
availableCount	int	Товара в наличии
productRefID	int	ID источника товара

Таблица Basket – это таблица, которая хранит информацию о товарах в корзине клиента. Подробное описание структуры таблицы дано в таблице 13.

Таблица 13 – описание таблицы Basket

Название столбца	Тип	Описание
id	int	Ключевое поле
name	string	Наименование корзины
products	object	Данные о товарах в корзине
uuid	string	Номер заказа
userID	int	ID пользователя
clientID	int	ID клиента
createdAt	string($date-time)	Дата создания

Таблица Order – это таблица, которая хранит информацию о товарах в корзине клиента. Подробное описание структуры таблицы дано в таблице 14.

Таблица 14 – описание таблицы Order

Название столбца	Тип	Описание
uuid	int	Ключевое поле
name	string	Наименование корзины

Окончание таблицы 14

Название столбца	Тип	Описание
price	number	Товары в корзине
Recipient.firstName	string	Номер заказа
Recipient.firstName	string	Данные о пользователе
Recipient.email	string	Данные о клиенте
Recipient.phone	string	Дата создания
Delivery.type	string	Тип доставки
Delivery.postcode	string	Почтовый индекс
Delivery.country	string	Страна
Delivery.region	string	Регион
Delivery.city	string	Город
Delivery.street	string	Улица
Delivery.building	string	Дом
Delivery.apartment	string	Квартира
Payment.type	string	Способ оплаты
Payment.status	string	Статус оплаты
basketID	int	Id корзины
status	string	Статус заказа
createdAt	string($date-time)	Дата создания

Таблица Catalog – это таблица, которая хранит информацию о каталогах с товарами. Подробное описание структуры таблицы дано в таблице 15.

Таблица 15 – описание таблицы Catalog

Название столбца	Тип	Описание
id	int	Ключевое поле
name	string	Наименование каталога
code	int	Код каталога
type	string	Тип каталога
deliveryPeriod	int	Срок доставки
sharing	boolean	Раздельно или нет
priceType	string	Тип цены
status	string	Статус
createdAt	string($date-time)	Дата создания

Таблица CentralWarehouse – это таблица, которая хранит информацию о складах. Подробное описание структуры таблицы дано в таблице 16.

Таблица 16 – описание таблицы CentralWarehouse

Название столбца	Тип	Описание
id	int	Ключевое поле
name	string	Наименование
ordering	int	Очередь

2.2 Серверная часть

Взаимодействие между сервером и клиентами происходит посредством REST и SignalR.

Для обработки запросов используется фреймворк PHP Laravel, так как постоянно расширяется. Имеется обширная библиотека документации, множество руководств и видеоуроков. Имеет надежные инструменты для внедрения зависимостей, модульного тестирования, создания очередей, событий и т.д. Laravel оптимизирован для создания профессиональных веб-приложений и готов обрабатывать корпоративные рабочие нагрузки.

Помимо всего прочего, из-за встроенной поддержки быстрых распределенных систем кэширования Laravel легко масштабируется для обработки сотен миллионов запросов в месяц.

2.3 Концепции разработки веб клиента

Для разработки веб клиента используется фреймворк «Vue.js» – JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек.

Веб клиент разработан по принципу Single Page Application – сокращенно SPA. Приложение такого типа появились сравнительно недавно, с началом эры HTML5 и SPA является типичным представителем приложений на HTML5.

Также в проекте активно используются следующие вспомогательные библиотеки Vue.js:

- Vuex;
- Vuelidate.
- Axios

Vuex – паттерн управления состоянием - библиотека для приложений на Vue.js. Он служит централизованным хранилищем данных для всех компонентов приложения с правилами, гарантирующими, что состояние может быть изменено только предсказуемым образом.

Vuelidate – библиотека для Vue.js, используемая для валидации полей.

Axios, предназначенная для выполнения HTTP-запросов, основана на промисах. Она подходит для использования в среде Node.js и в браузерных приложениях. Библиотека поддерживает все современные браузеры, и, в том числе, IE8+.

2.4 Структура проекта.

Проект состоит из основных директорий:

— src/assets – содержит ресурсы, используемые в проекте (статичные рисунки, лого, графики, отображаемые на страницах);

— src/components – содержит компоненты страниц, в которых отображаются такие элементы, как таблица и их функционал;

— src/helpers – содержит файлы – помощники, необходимые для корректной работы библиотеки axios, в которых указывается базовый url сайта, обрабатывается привязывание токена к пользователю после авторизации и проверка на авторизацию пользователя с редиректом на страницу авторизации в случае попытки входа на сайт без авторизации;

— src/layouts – слои, используемые как шаблоны для страниц. EmptyLayout – пустой шаблон, используемый для страницы авторизации и MainLayout – шаблон с модулями header, sider, footer и breadcrumb;

— src/modules – модули шаблона страницы, используемые в MainLayout – header, footer, sider;

— src/router – файл роутинга (маршрутов пользователя);

— src/services/api – содержит файлы с ссылкой на api и запросы к нему;

— src/store – содержит модули хранилища данных, используемых на страницах в таблицах;

— src/utils/errorHandler – объект, содержащий метод обработки ошибок и отображения соответствующего сообщения;

— src/views – содержит представления, отображаемые при переходе на страницу. Включает в себя соответствующие компоненты.

Компоненты, используемые в данном проекте разбиты по папкам с соответствующим представлению наименованием, представленным на рисунке 6.

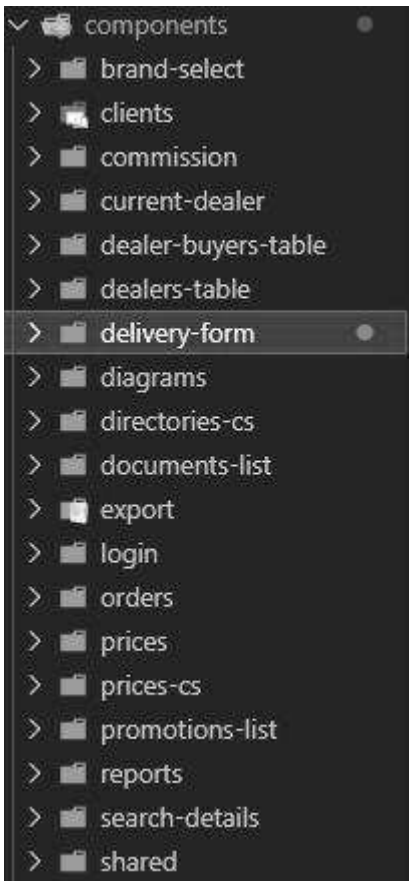

Рисунок 6 – Используемые компоненты проекта

Листинг кода компонента на примере ClientData.vue представлен в приложении А.

Данные, получаемые с API используются для отображения на страницах и разделены на модули, представленные на рисунке 7.

Рисунок 7 – Модули хранилища данных

Листинг кода модулей хранилища на примере модуля clients.js представлен в приложении Б

Запросы к API находятся в папке services и содержатся в файлах с соответствующим данным названиями и припиской Service. Используемые в проекте сервисы представлены на рисунке 8.

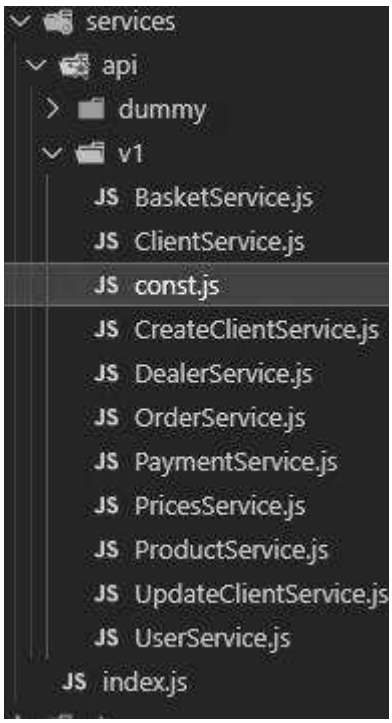

Рисунок 8 – Сервисы с запросами к API

Листинг кода сервиса с запросами на примере ClientService.js представлен в приложении В

2.4.1 Регистрация и аутентификация пользователей в системе

Регистрация пользователя в системе в роли дилера происходит после отправки приглашения на почту представителя. Пользователи в роли клиента могут быть зарегистрированы при создании клиента в кабинете дилера.

Внешний вид формы авторизации представлен на рисунке 9.

Рисунок 9 – Страница авторизации.

При переключении активного поля происходит клиентская валидация полей, например, на обязательное заполнение. Если валидация нашла ошибки, то пользователь получит соответствующие сообщения.

Для того, чтобы при каждом открытии приложения не заставлять пользователя снова и снова вводить логин и пароль, эти данные кэшируются в локальном хранилище браузера и запрашиваются оттуда при открытии сайта, тем самым улучшая User Expirience».

В зависимости от роли пользователя у него есть собственные возможности на сайте. В соответствии с этим меню для каждой роли имеет различные пункты.

Главное меню для дилера представлено на рисунке 11.

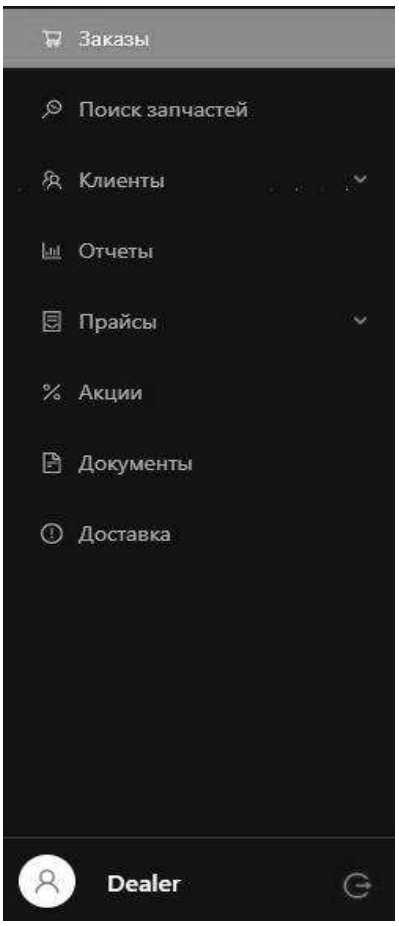

Рисунок 11 – Главное меню дилера

Через главное меню дилер имеет возможность попасть на страницы заказов, поиска запчастей, клиентов, прайсов.

Главное меню для клиента представлено на рисунке 12.

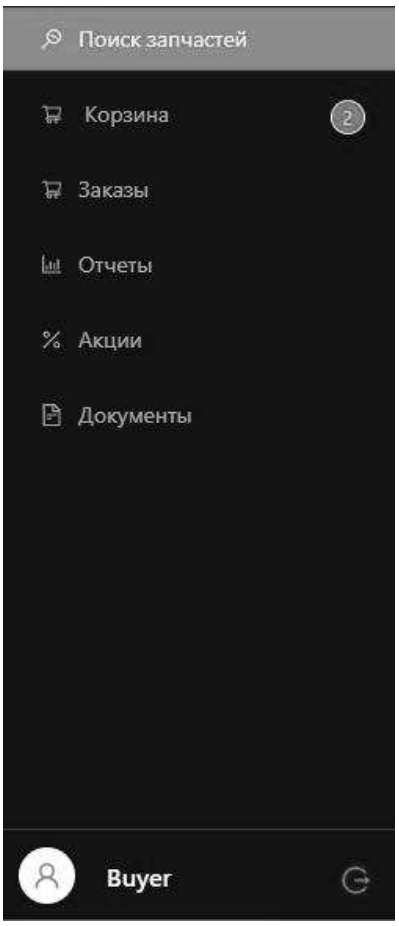

Рисунок 12 – Главное меню клиента

Через главное меню клиент может попасть на страницы поиска запчастей, корзины, и заказов.

2.4.2 Страница клиентов

На странице «Клиенты» дилер может увидеть список клиентов с возможностью фильтра. Внешний вид этого экрана можно наблюдать на рисунке 13.

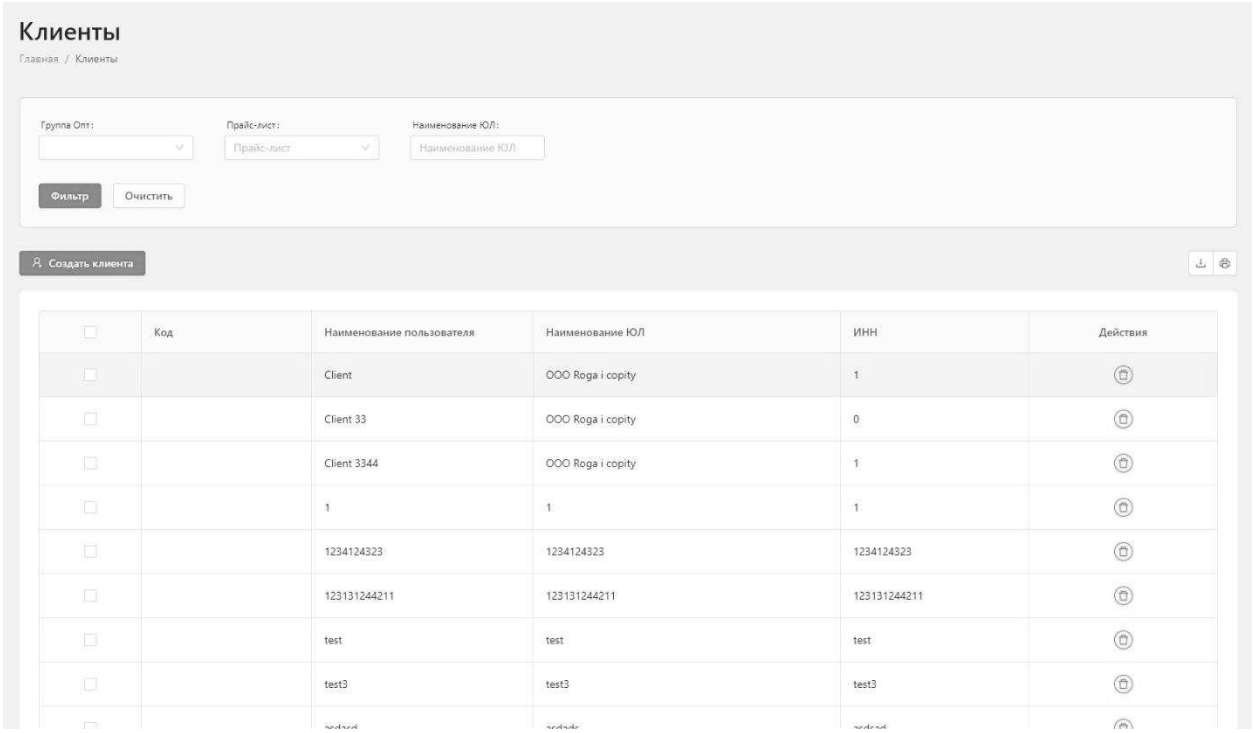

Рисунок 13 – Страница «Список клиентов»

При нажатии на строку клиента в списке пользователь будет перенаправлен на страницу всей информации о клиенте с возможностью редактирования полей

При нажатии на кнопку «Создать клиента» будет произведен переход на страницу «Регистрация нового клиента», на которой заполняются поля учетной записи клиента: логин, пароль, почта; данные клиента: Наименование пользователя, ФИО руководителя, банковские реквизиты и т.д.

Внешний вид формы создания клиента представлен на рисунке 14.

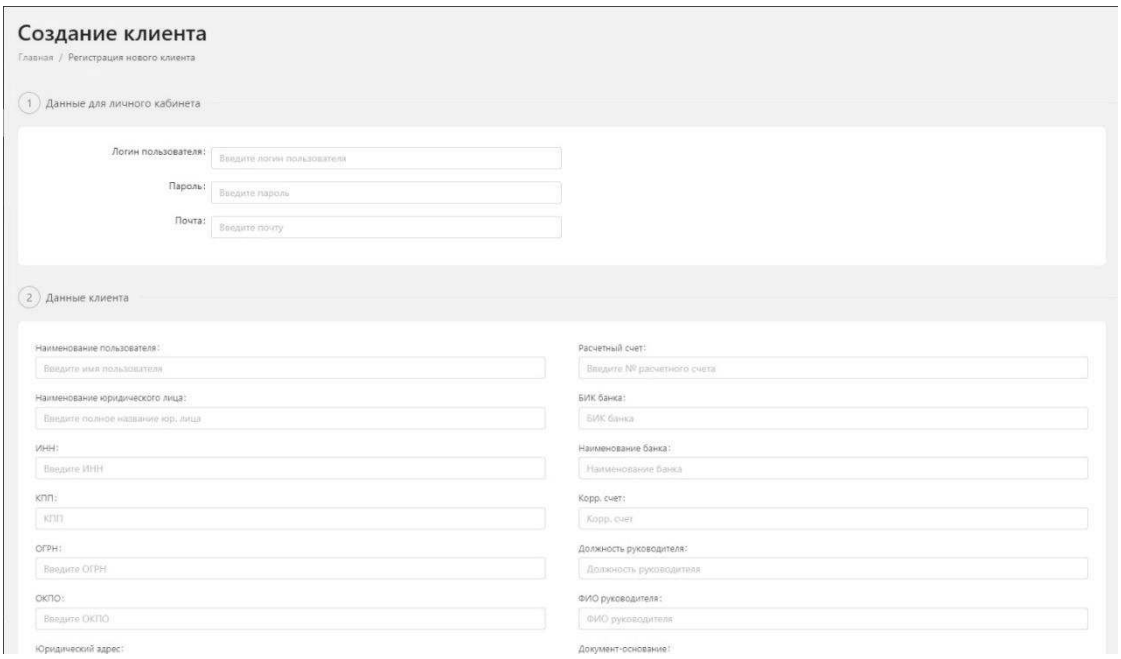

Рисунок 14 – Страница «Создание клиента»

На странице «Создание клиента» вводится логин пользователя, пароль, почта, данные об организации, банковские реквизиты и контактная информация.

2.4.3 Страница прайсов

На странице «Локальные прайс-листы отображаются загруженные дилером прайс-листы с возможностью загрузки новых и фильтрацией списка существующих. Страница «Локальные прайс-листы» показана на рисунке 15.

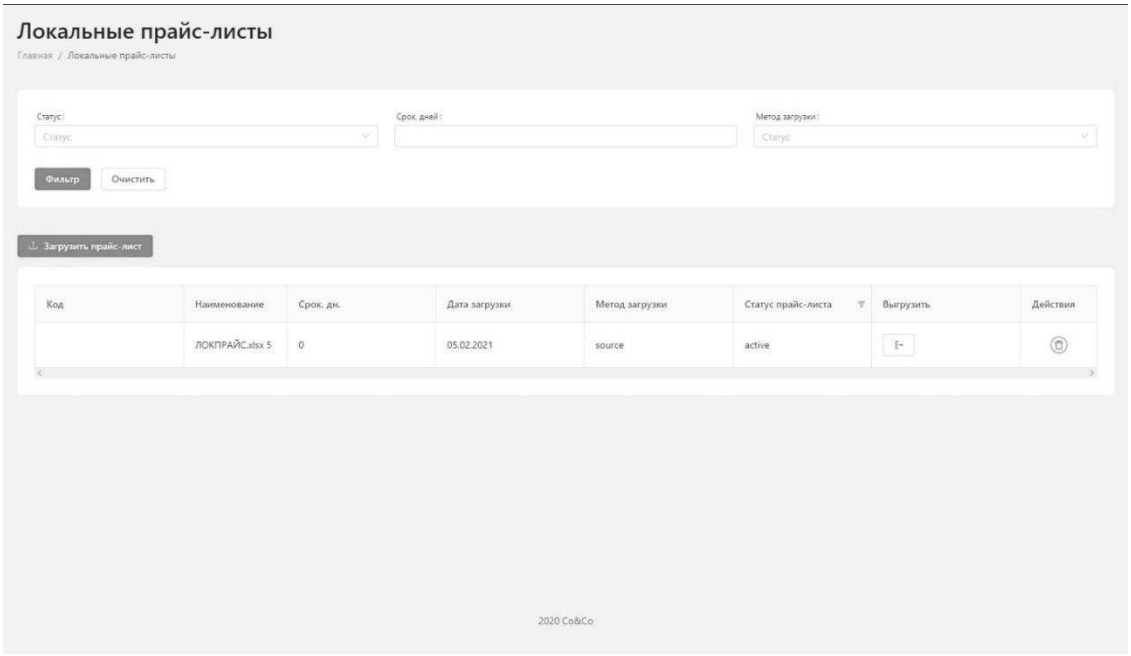

Рисунок 15 – Страница «Локальные прайс-листы»

Прайс-листы загружаются в формате xls, xlsx, в которых указывается список товаров и их данные.

На странице «Наценки» отображаются настройки наценок прайс-листов для определенных групп клиентов, разделенных на группы ОПТ. Внешний вид этой страницы представлен на рисунке 18.

2.4.4 Страница поиска запчастей

Страница «Поиск запчастей» представляет собой каталог запчастей, позиции из которого можно добавить в корзину. Внешний вид страницы «Поиск запчастей» представлен на рисунке 16.

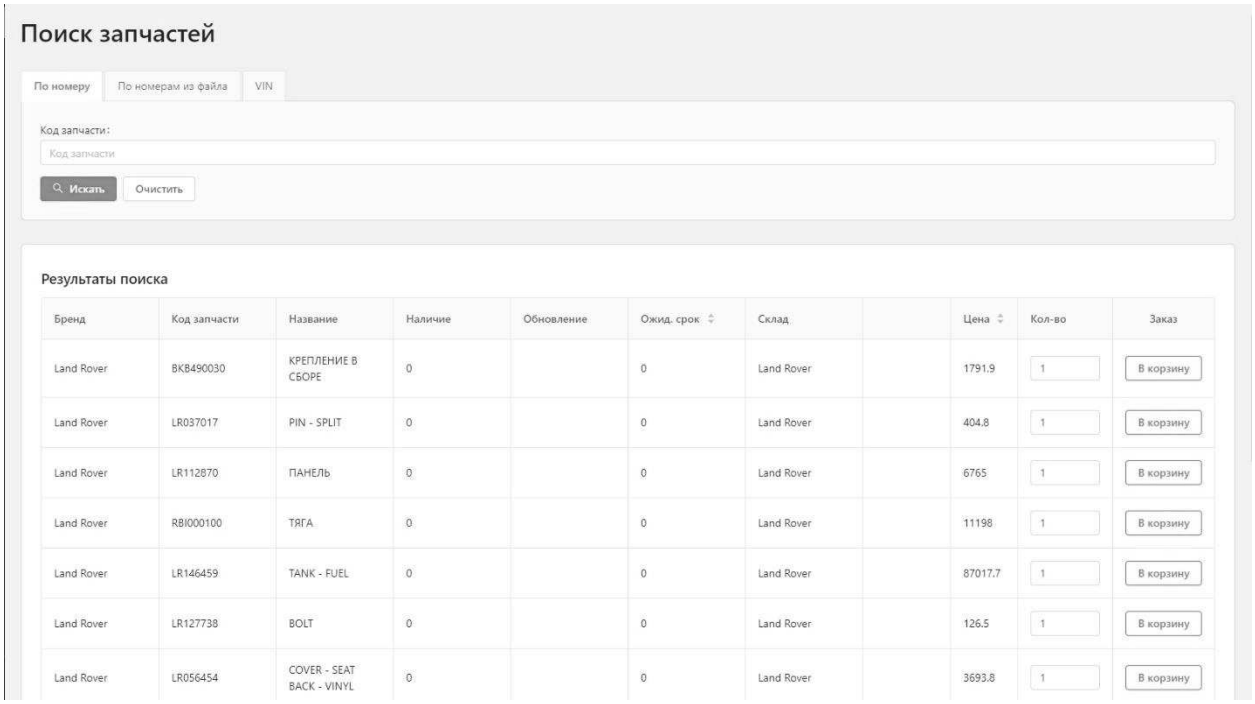

Рисунок 16 – Страница «Поиск запчастей»

На странице есть возможность фильтровать запчасти по коду запчасти и по номерам из файла в формате CVS. В таблице можно указать количество товара и добавить его в корзину.

2.4.5 Страница корзины

После добавления в корзину пользователь может посмотреть товары, которые он хочет заказать. удалить ненужные и при нажатии на кнопку «Оформить»

Внешний вид страницы «Корзина» представлен на рисунке 17.

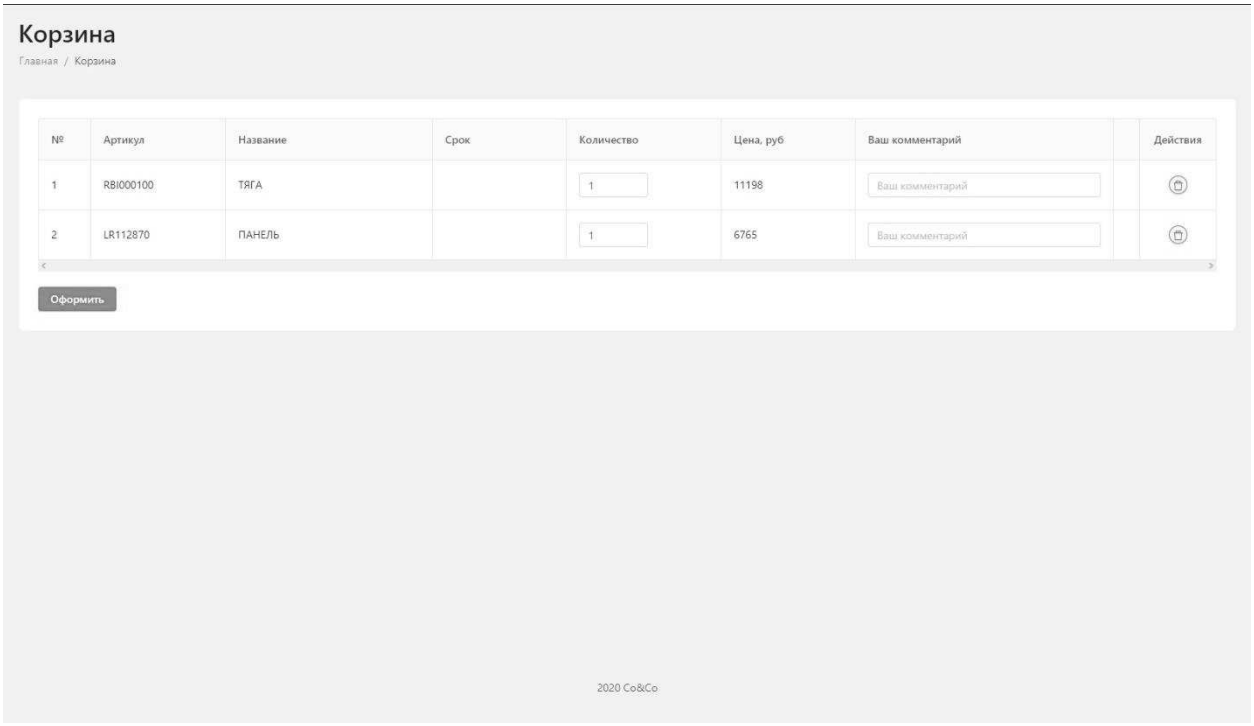

Рисунок 17 – Страница «Корзина»

Пользователь может удалить ненужные товары и при нажатии на кнопку «Оформить» перейти на страницу оформления заказа.

2.4.6 Страница заказов

На странице «Заказы» пользователь может посмотреть список заказов клиентов, узнать номер заказа, клиента, сумму заказа, количество позиций, статус заказа и дату создания заказа.

Внешний вид страницы заказов можно увидеть на рисунке 18.

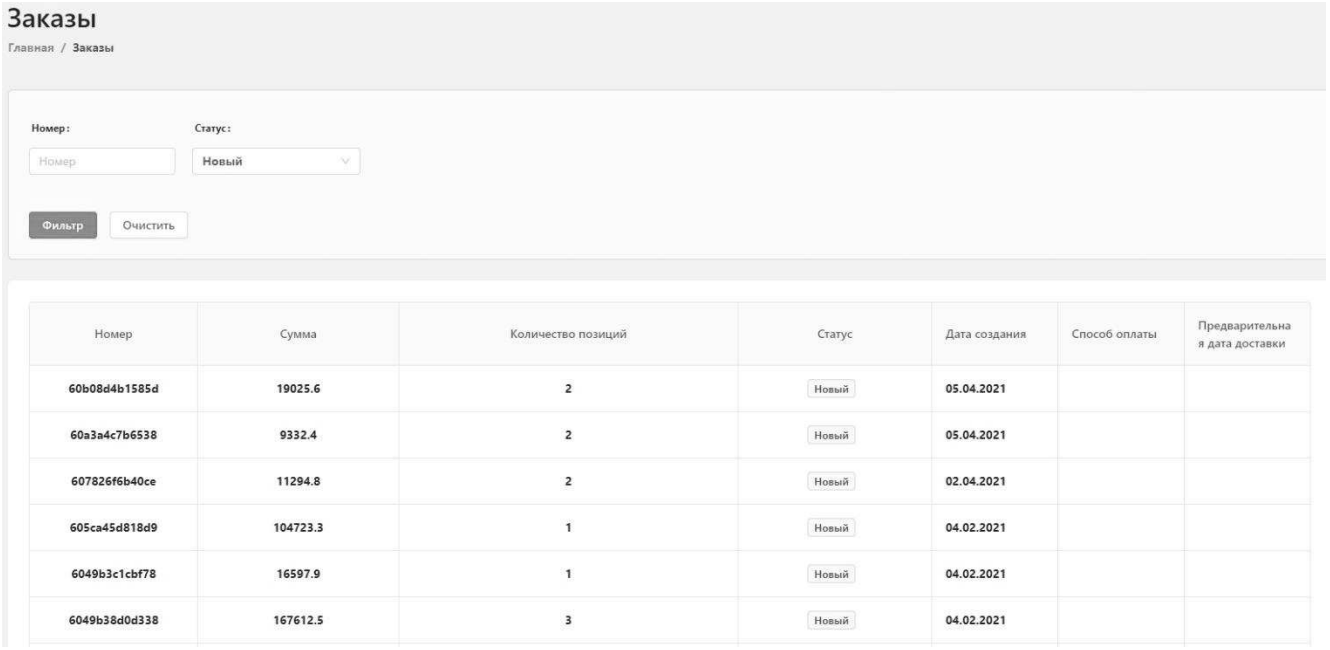

Рисунок 18 – Страница «Заказы»

Пользователи с ролью дилера видят заказы на свои товары. Пользователи с ролью оптовика видят свои заказы. На странице есть возможность отфильтровать список заказов по номеру заказа и статусу. При нажатии на строку заказа в таблице откроется полная информация о заказе.

Выводы по разделу 2

Была составлена схема взаимодействия пользователей сайта с системой, на основании их ролей и распределенных прав. Выполнено проектирование основных страниц веб-ресурса.

3 РАСЧЕТ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ ОТ ВНЕДРЕНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При разработке программного обеспечения необходимы финансовые ресурсы для приобретения необходимого оборудования и программного обеспечения, а также на саму разработку и подготовку к ней. Поэтому необходимо экономически обосновать целесообразность ведения разработки.

Чтобы разрабатываемое программное обеспечение достигло успеха в распространении и продажах, необходимо составить бизнес–план. Бизнес-план содержит показатели, показывающие экономическую и коммерческую эффективность работы.

В таблице 17 произведен расчет единовременных затрат.

Таблица 17 – Расчет единовременных затрат

№	Наименование технического средства и ПО	Тип или модель	Стоимость
1	Операционная система	MS Windows 10 Pro	20 887
2	Системный блок	ACER Aspire XC-895	34 890
3	Монитор	SAMSUNG F24T354FHI	11 990
4	Принтер	HP Laser 107a	6990
5	Клавиатура и мышь	DEXP KM–103BU	899
6	Стол компьютерный		6000
7	Стул		5000
Итого			86 606

Единовременные затраты составляют:

Зк = 20887 + 34890 + 11990 + 6990 + 899 + 6000 + 5000 = 86606 руб.

Затраты на оплату машинного времени

$$Зм = См * Тм, \quad (1)$$

где См =50 руб./час – стоимость одного часа машинного времени;

Тм =160 + 158 час – время использования машины.

Исходя из расчетов по формуле 1 Зм = 50 * 318 = 15900 руб.

Накладные расходы, связанные с проектированием и отладкой ПП, в том числе стоимость используемых материалов описаны в таблице 18.

Таблица 18 – Потребности

Материалы	Потребность	Стоимость одной ед., руб.	Общая стоимость, руб.	Примечания
Бумага	200 листов	1	200	Печать текстов
Расходные материалы для лазерного принтера	1 шт.	997	997	Для печати необходимых данных
Перезаписываемый диск флеш-накопитель	2 шт.	460	920	Резервирование данных и материалов
Использование сети Internet	18 часов	8,8	158,4	Поиск информации и литератур.
Итого			2275,4	

Для выполнения работ необходимо материалов на сумму 2275,4руб.

Оплата за пользование электричеством составляет

Зэ=0,5кВатт*318 часа* 2,35 руб= 373,65 руб.

Текущие затраты (себестоимость) (С) включают затраты на постановку задачи, разработку алгоритмов и программ, а также затраты, связанные с содержанием и эксплуатацией ВТ. Формула расчета себестоимости:

$$С = Зпр + Зф + Змаш + Зн + Зэ, \quad (2)$$

Зпр = 30225,3

В таблице 19 представлены текущие затраты.

Таблица 19 – Текущие затраты

Наименование статей затрат	Сумма
1. Затраты на заработную плату (Зпр)	30225,3
2. Отчисления в фонды с заработной платы труда (ФФОМС, ПФР, ФСС) (Зф = 30,2%*Зпр)	9128
3. Затраты, связанные с использованием машинного времени (Змаш)	15900
5. Накладные расходы (Зн)	2275,4
6. Расходы на электричество при пользовании вычислительной техникой(Зэ)	373,65
Итого затрат (С)	57 902,35

В таблице 20 представлен расчет затрат на разработку программного средства.

Таблица 20 – Расчет затрат на разработку программного средства

Затраты и ожидаемые доходы от внедрения ИС	Период			
Затраты	1 квартал	2 квартал	3 квартал	4 квартал
1. Капитальные затраты	86606			
2. Текущие затраты:	57902,35	57902,35	57902,35	57902,35
–Заработная плата	30225,3	30225,3	30225,3	30225,3
–Отчисления в фонды (ФФОМС, ПФР, ФСС)	9128	9128	9128	9128
–Затраты, связанные с использованием машинного времени	15900	15900	15900	15900
–Накладные расходы	2275,4	2275,4	2275,4	2275,4
–Расходы на электричество при пользовании вычислительной техникой	373,65	373,65	373,65	373,65
Итого затрат:	144508,35	57902,35	57902,35	57902,35

При отсутствии необходимого программного обеспечения на работе по переводу иностранных документов имеет большие временные затраты. Зная

заработную плату за день (в среднем она составляет 300 руб./день) можно подсчитать, что работы с переводом документации обходится предприятию в сумму равную 300*3дн./нед.*4,5нед= 4050 рублей в месяц, (4050*3=12150) за квартал.

За счет введения в эксплуатацию сайта количество клиентов должно увеличиться, при этом, чем дольше будет работать сайт, тем больше клиентов воспользуется программным продуктом, из-за чего увеличится прибыль.

Определение доходов от внедрения в работу программного обеспечения представлено в таблице 21.

Таблица 21 – Доходы

Доходы/Месяцы	1 квартал	2 квартал	3 квартал	4 квартал
1 Снижение затрат на ручную обработку больших объемов информации, руб.	12150	12150	12150	12150
2. Привлечение дополнительных клиентов, руб.	40000	52500	65000	77500
Итого доходов тыс. руб.:	52150	64650	72150	89650

Расчет экономической эффективности проводится по следующим показателям:

- чистый дисконтированный доход (ЧДД) или интегральный эффект;
- индекс доходности (ИД);
- срок окупаемости;

При расчетах этих показателей используется константа а – это норма дисконта, равная приемлемой для инвестора норме дохода на капитал -0,14.

В таблице 22 представлены технико-экономические показатели работы.

Таблица 22 – Технико-экономические показатели работы

Периоды (месяц)	Показатели		$1/(1+a)^t$	Дисконтированные	Годовая экономическая эффективность	ЧДД с нарастающим итогом	
	Доходы	Расходы					
	1	2	3	4=1*3	5=2*3	6=4-5	7
1 квартал 2021	52150	144508,35	0.877192982	45745,61	126761,71	-81016,1	-81016,1
2 квартал 2021	64650	57902,35	0.769467528	50515,54	44553,97	5961,57	-75054,53
3 квартал 2021	77150	57902,35	0.674971516	52074,05	39082,43	12991,62	-62062,91
4 квартал 2021	89650	57902,35	0.592080277	53079,99	34282,83	18797,16	-43265,75
1 квартал 2022	102150	57902,35	0.519368664	53053,5	30072,66	22980,84	-20284,91
2 квартал 2022	114650	57902,35	0.455586548	52232,99	26379,53	25 853,46	5568,55
Итого:	500400	434020,1	3.369298852	306701,68	301133,13		

Индекс доходности представляет собой отношение суммы приведенных эффектов к величине капитальных вложений:

$$ИД = \frac{\sum_{t=1}^{T} Д_t * \frac{1}{(1+\alpha)^t}}{\sum_{t=1}^{T} P_t * \frac{1}{(1+\alpha)^t}} \quad (3)$$

Индекс доходности строится из тех же элементов, что и ЧДД. Если ЧДД положителен, то ИД > 1 и наоборот. Исходя из расчетов по формуле 3 индекс доходности: 1,01

Расчет ЧДД инвестиционного проекта показывает, является ли он эффективным при некоторой заданной норме дисконта.

На рисунке 23 представлен график срока окупаемости.

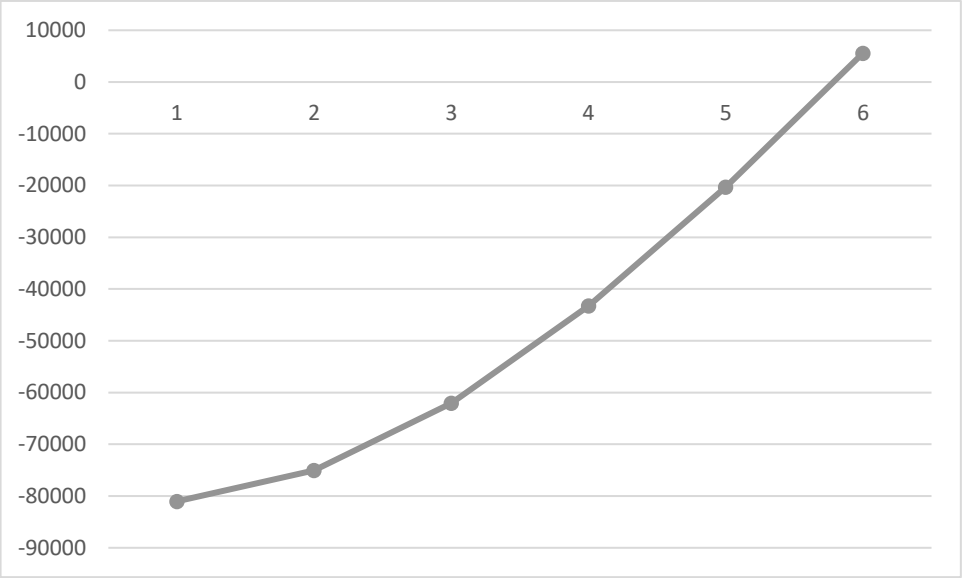

Рисунок 19 – График окупаемости

Затраты на разработку составляют 57 902,35 руб. ежеквартально. Экономический эффект от использования данного программного продукта за расчётный период (1 год и 6 месяцев) составит 5568,55 при этом разработчик покроет свои расходы на создание системы ориентировочно за 1,5 года и затем начнёт получать прибыль.

Выводы по разделу 3

Проведён анализ основных разделов бизнес-плана, осуществлена калькуляция темы с оценкой экономической эффективности реализации программного продукта.

Показатели ЧДД и ИДД положительны. Через 1 год и 6 месяцев разработчик покроет расходы на создание системы и начнёт получать прибыль. Можно сделать вывод о целесообразности и актуальности разработки данного программного продукта.

ЗАКЛЮЧЕНИЕ

В результате проведенной работы были расширены и углублены теоретические знания, полученные при обучении в университете, приобретены практические навыки работы в области разработки веб-приложений. В ходе реализации проекта реализованы следующие функции:

- авторизация;
- управление аккаунтами клиентов дилера;
- оформление заказа;
- загрузка прайс-листов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кингсли, Х.Э. JavaScript в примерах. [Электронный ресурс] : учеб. пособие / Х.Э. Кингсли, Х.К. Кингсли. — Электрон. дан. — М. : ДМК Пресс, 2009. — 272 с. — Режим доступа: http://e.lanbook.com/book/1271 — Загл. с экрана./

2. Марк, Б. CoffeeScript. Второе дыхание JavaScript. [Электронный ресурс] : учеб. пособие — Электрон. дан. — М. : ДМК Пресс, 2012. — 312 с. — Режим доступа: http://e.lanbook.com/book/50573 — Загл. с экрана

3. Штефен, В. Разработка приложений для Windows 8 с помощью HTML5 и JavaScript. Подробное руководство. [Электронный ресурс] : рук. — Электрон. дан. — М. : ДМК Пресс, 2013. — 344 с. — Режим доступа: http://e.lanbook.com/book/58696 — Загл. с экрана.

4. Тиге, Д.К. DHTML и CSS. [Электронный ресурс] : учеб. пособие — Электрон. дан. — М. : ДМК Пресс, 2008. — 558 с. — Режим доступа: http://e.lanbook.com/book/1069 — Загл. с экрана.

ПРИЛОЖЕНИЕ А

ЛистингА.1 – Код компонента на примере компонента ClientData

```
<template>
  <div>
    <a-form :label-col="{}" :wrapper-col="{}" class="clientdata">
      <a-row :gutter="48">
        <a-col
          v-for="v in $v.fields.$each.$iter"
          :key="v.$model.id"
          class="form-item"
          :span="12"
        >
          <a-form-item
            :has-feedback="v.value.minLength"
            :validate-status="v.value.$error ? 'error' : ''"
            :label="v.$model.name"
            :help="v.value.$error && 'Необходимо заполнить поле'"
          >
            <a-input
              v-model.trim="v.value.$model"
              @input="
                e =>
                  setField({
                    name: v.$model.createclientname,
                    value: e.target.value,
                  })
              "
              :placeholder="v.$model.placeholder"
              :value="checkFields(v.$model.createclientname)"
            />
          </a-form-item>
        </a-col>
        <a-col :span="12" class="form-item">
          <PaymentGroupNewClient />
        </a-col>
      </a-row>
    </a-form>
    <div
      v-if="this.$route.name !== 'CreateNewClient' && !isOwner"
      class="buttonsholder"
    >
      <ButtonsHolderUpdateClient />
    </div>
  </div>
</template>

<script>
import { mapMutations, mapGetters } from "vuex"
```

```
import { required } from "vuelidate/lib/validators"

im-
port ButtonsHolderUpdateClient from "@/components/clients/ButtonsHolderUpdateClient"
import PaymentGroupNewClient from "@/components/clients/PaymentGroupNewClient"

export default {
  name: "ClientData",
  components: {
    ButtonsHolderUpdateClient,
    PaymentGroupNewClient,
  },
  computed: mapGetters({
    createForm: "clients/createForm",
    isOwner: "auth/isOwner",
  }),
  data: () => ({
    fields: [
      {
        id: 1,
        name: "Наименование пользователя",
        placeholder: "Введите имя пользователя",
        createclientname: "name",
        value: "",
      },
      {
        id: 2,
        name: "Расчетный счет",
        placeholder: "Введите № расчетного счета",
        createclientname: "legalInfoPaymentAccount",
        value: "",
      },
      {
        id: 3,
        name: "Наименование юридического лица",
        placeholder: "Введите полное название юр. лица",
        createclientname: "legalInfoName",
        value: "",
      },
      {
        id: 4,
        name: "БИК банка",
        placeholder: "БИК банка",
        createclientname: "legalInfoBankId",
        value: "",
      },
      {
        id: 5,
```

```
      name: "ИНН",
      placeholder: "Введите ИНН",
      createclientname: "legalInfoInn",
      value: "",
    },
    {
      id: 6,
      name: "Наименование банка",
      placeholder: "Наименование банка",
      createclientname: "legalInfoBankName",
      value: "",
    },
    {
      id: 7,
      name: "КПП",
      placeholder: "КПП",
      createclientname: "legalInfoKpp",
      value: "",
    },
    {
      id: 8,
      name: "Корр. счет",
      placeholder: "Корр. счет",
      createclientname: "legalInfoCorrespondedAccount",
      value: "",
    },
    {
      id: 9,
      name: "ОГРН",
      placeholder: "Введите ОГРН",
      createclientname: "legalInfoOgrn",
      value: "",
    },
    {
      id: 10,
      name: "Должность руководителя",
      placeholder: "Должность руководителя",
      createclientname: "headPosition",
      value: "",
    },
    {
      id: 11,
      name: "ОКПО",
      placeholder: "Введите ОКПО",
      createclientname: "legalInfoOkpo",
      value: "",
    },
    {
```

```
      id: 12,
      name: "ФИО руководителя",
      placeholder: "ФИО руководителя",
      createclientname: "headName",
      value: "",
    },
    {
      id: 13,
      name: "Юридический адрес",
      placeholder: "Юридический адрес",
      createclientname: "legalInfoAddress",
      value: "",
    },
    {
      id: 14,
      name: "Документ-основание",
      placeholder: "Введите навзание и № документа",
      createclientname: "legalInfoUnderlyingDocument",
      value: "",
    },
    {
      id: 15,
      name: "Фактический адрес",
      placeholder: "Фактический адрес",
      createclientname: "legalInfoPhysicalAddress",
      value: "",
    },
  ],
}),
validations: {
  fields: {
    required,
    $each: {
      value: {
        required,
      },
    },
  },
},
watch: {
  createForm() {
    this.setFormFields()
  },
},
mounted() {
  if (this.$route.name === "CreateNewClient") {
    this.setClearFields()
  } else {
```

```
      this.setFormFields()
    }
  },
  methods: {
    checkClient() {
      this.$v.fields.$touch()
    },
    ...mapMutations({
      setField: "clients/SET_CREATE_FORM_FIELD",
    }),
    setClearFields() {
      this.fields = this.fields.map(item => {
        item.value = ""
        return item
      })
    },
    setFormFields() {
      this.fields = this.fields.map(item => {
        const createFormValue = this.createForm[item.createclientname]
        if (createFormValue) {
          item.value = createFormValue
        }
        return item
      })
    },
    checkFields(e) {
      // TODO: Переделать
      if (e === "name") {
        return this.createForm.name
      } else if (e === "legalInfoPaymentAccount") {
        return this.createForm.legalInfoPaymentAccount
      } else if (e === "legalInfoName") {
        return this.createForm.legalInfoName
      } else if (e === "legalInfoBankId") {
        return this.createForm.legalInfoBankId
      } else if (e === "legalInfoInn") {
        return this.createForm.legalInfoInn
      } else if (e === "legalInfoBankName") {
        return this.createForm.legalInfoBankName
      } else if (e === "legalInfoKpp") {
        return this.createForm.legalInfoKpp
      } else if (e === "legalInfoCorrespondedAccount") {
        return this.createForm.legalInfoCorrespondedAccount
      } else if (e === "legalInfoOgrn") {
        return this.createForm.legalInfoOgrn
      } else if (e === "headPosition") {
        return this.createForm.headPosition
      } else if (e === "legalInfoOkpo") {
```

```
        return this.createForm.legalInfoOkpo
      } else if (e === "headName") {
        return this.createForm.headName
      } else if (e === "legalInfoAddress") {
        return this.createForm.legalInfoAddress
      } else if (e === "legalInfoUnderlyingDocument") {
        return this.createForm.legalInfoUnderlyingDocument
      } else if (e === "legalInfoPhysicalAddress") {
        return this.createForm.legalInfoPhysicalAddress
      }
    },
    handleAcrtionClientData() {
      this.checkClient()
      if (this.$v.$error) {
        window.scrollTo(0, "100%")
      }
    },
  },
}
</script>

<style lang="scss">
.buttonsholder {
  margin-top: 20px;
  display: flex;
  flex-direction: column;
  align-items: left;
  padding: 0 auto;
}
.clientdata {
  label {
    font-size: 13px !important;
    color: rgba(black, 0.7);
  }
}
</style>
```

ПРИЛОЖЕНИЕ Б

Листинг Б.1 – Код модуля хранилища на примере модуля clients.js

```
import ClientService from "@/services/api/v1/ClientService"
const service = new ClientService()
import CreateClientService from "@/services/api/v1/CreateClientService"
const createService = new CreateClientService()

export const state = {
  columns: [
    {
      title: "Код",
      dataIndex: "code",
      key: "code",
      width: 100,
    },
    {
      title: "Наименование пользователя",
      key: "name",
      dataIndex: "name",
      width: 130,
    },
    {
      title: "Наименование ЮЛ",
      dataIndex: "legalInfo.name",
      key: "ulName",
      width: 180,
    },
    {
      title: "ИНН",
      dataIndex: "legalInfo.inn",
      key: "inn",
      width: 110,
    },
    {
      title: "Действия",
      dataIndex: "operation",
      scopedSlots: { customRender: "operation" },
      className: "column-action",
      width: 110,
    },
  ],
  clients: [],
  paymentGroup: null,
  filters: {
    name: null,
    paymentGroup: null,
    inn: null,
  },
```

```
  createFormError: null,
  contactNames: [
    "Взаиморасчеты",
    "Отправка прайс-листов",
    "Руководитель",
    "Заказы",
    "Возвраты",
    "Обновление паролей",
  ],
  createForm: {
    name: "",
    paymentGroup: {
      id: "",
      name: "",
      ordering: "",
    },
    headName: "",
    headPosition: "",
    legalInfo: {
      name: "",
      paymentAccount: "",
      bankId: "",
      bankName: "",
      correspondedAccount: "",
      underlyingDocument: "",
      inn: "",
      kpp: "",
      ogrn: "",
      okpo: "",
      address: "",
      physicalAddress: "",
    },
    contacts: [
      {
        type: "Взаиморасчеты",
        name: "",
        phone: "",
        email: "",
      },
      {
        type: "Отправка прайс-листов",
        name: "",
        phone: "",
        email: "",
      },
      {
        type: "Руководитель",
        name: "",
```

```
        phone: "",
        email: "",
      },
      {
        type: "Заказы",
        name: "",
        phone: "",
        email: "",
      },
      {
        type: "Возвраты",
        name: "",
        phone: "",
        email: "",
      },
      {
        type: "Обновление паролей",
        name: "",
        phone: "",
        email: "",
      },
    ],
  },
}
export const mutations = {
  SET_CLIENTS(state, payload) {
    state.clients = payload
  },
  CLEAR_FILTERS(state) {
    state.filters.paymentGroup = null
    state.filters.name = null
  },
  SET_FILTER_NAME(state, payload) {
    state.filters.name = payload
  },
  SET_FILTER_INN(state, payload) {
    state.filters.inn = payload
  },
  SET_FILTER_PAYMENT_GROUP(state, payload) {
    state.filters.paymentGroup = payload
  },
  SET_CREATE_CLIENT_FORM_ERROR(state, payload) {
    state.createFormError = payload
  },
  CLEAR_CREATE_CLIENT_FORM_ERROR(state) {
    state.createFormError = null
  },
  SET_CREATE_FORM_CONTACT_FIELD(state, payload) {
```

```
    state.createForm.contacts[payload.index][payload.field] = payload.value
  },
  SET_CREATE_FORM_FIELD(state, payload) {
    state.createForm[payload.name] = payload.value
  },
  SET_CREATE_FORM(state, payload) {
    state.createForm = payload
  },
  SET_PAYMENT_GROUPS(state, payload) {
    state.paymentGroup = payload
  },
  DEL_CLIENT_ITEM(state, payload) {
    state.clients = state.clients.filter(el => el.id !== payload)
  },
}

export const actions = {
  async fetchClients({ commit, getters }) {
    const filters = getters.filters
    const response = await service.search(
      filters.paymentGroup,
      filters.name,
      filters.inn
    )
    commit("SET_CLIENTS", response.data.items)
  },

  async createClient({ commit, getters }) {
    const clientFormData = getters.createForm
    const contacts = clientFormData.contacts.filter(item => {
      if (item && (item.name || item.phone || item.email)) {
        return item
      }
    })

    clientFormData.contacts = contacts
    return new Promise((resolve, reject) => {
      commit("CLEAR_CREATE_CLIENT_FORM_ERROR")
      createService
        .createClient(clientFormData)
        .then(resp => resolve(resp))
        .catch(error => {
          reject(error.response)
        })
    })
  },
  async updateClient({ commit, getters }) {
    const clientFormData = JSON.parse(JSON.stringify(getters.createForm))
```

```
    clientFormData.contacts = clientFormData.contacts.filter(item => {
      if (item.name || item.phone || item.email) {
        return item
      }
    })
    return new Promise((resolve, reject) => {
      commit("CLEAR_CREATE_CLIENT_FORM_ERROR")
      service
        .updateClient(clientFormData)
        .then(() => resolve())
        .catch(error => {
          console.log(error)
          reject()
        })
    })
  },
  async deleteClient({ commit }, id) {
    return new Promise((resolve, reject) => {
      commit("CLEAR_CREATE_CLIENT_FORM_ERROR")
      commit("DEL_CLIENT_ITEM", id)
      service
        .delSingleClient(id)
        .then(() => resolve())
        .catch(error => {
          console.log(error)
          reject()
        })
    })
  },
  async bindClientWithUser({ getters }, userId) {
    const { name, headName, headPosition } = getters.createForm
    const neededClient = await getters.clients.filter(el => {
      if (
        el.name === name &&
        el.headName === headName &&
        el.headPosition === headPosition
      ) {
        return el
      }
    })
    const [el] = neededClient
    return new Promise((resolve, reject) => {
      service
        .bindClient(userId, el.id)
        .then(() => resolve())
        .catch(e => {
```

```
          console.log(e)
          reject()
        })
    })
  },

  async getClient({ commit }, clientid) {
    const response = await service.getSingleClient(clientid)
    commit("SET_CREATE_FORM", {
      name: response.data.name,
      headName: response.data.headName,
      headPosition: response.data.headName,
      legalInfoName: response.data.legalInfo.name,
      legalInfoPaymentAccount: response.data.legalInfo.paymentAccount,
      legalInfoBankId: response.data.legalInfo.bankId,
      legalInfoBankName: response.data.legalInfo.bankName,
      legalInfoCorrespondedAccount: response.data.legalInfo.correspondedAccount,
      legalInfoUnderlyingDocument: response.data.legalInfo.underlyingDocument,
      legalInfoInn: response.data.legalInfo.inn,
      legalInfoKpp: response.data.legalInfo.kpp,
      legalInfoOgrn: response.data.legalInfo.ogrn,
      legalInfoOkpo: response.data.legalInfo.okpo,
      legalInfoAddress: response.data.legalInfo.address,
      legalInfoPhysicalAddress: response.data.legalInfo.physicalAddress,
      paymentGroup: response.data.paymentGroup,
      contacts: response.data.contacts,
      id: response.data.id,
      status: response.data.status,
      createdAt: response.data.createdAt,
    })
  },
}
export const getters = {
  columns(state) {
    return state.columns
  },
  clients(state) {
    return state.clients
  },
  filters(state) {
    return state.filters
  },
  createForm(state) {
    return state.createForm
  },
  data(state) {
    return state.createForm.contacts
  },
```

```
  contactNames(state) {
    return state.contactNames
  },
  paymentGroup(state) {
    return state.createForm.paymentGroup
  },
}
```

ПРИЛОЖЕНИЕ В

Листинг В.1 – Код сервиса запросов на примере ClientService.js

```
import axios from "axios"
import { API_PREFIX } from "@/services/api/v1/const"

export default class ClientService {
  async search(paymentGroup, name, inn) {
    const options = {}

    if (paymentGroup) {
      options.paymentGroup = paymentGroup
    }
    if (name) {
      options["legalInfo.name"] = name
    }
    if (inn) {
      options["legalInfo.inn"] = inn
    }

    return axios.get(
      `${API_PREFIX}/client?${new URLSearchParams(options).toString()}`
    )
  }
  async getSingleClient(id) {
    return axios.get(`${API_PREFIX}/client/${id}`)
  }
  async bindClient(userId, clientId) {
    return axios.put(`${API_PREFIX}/client`, {
      userId,
      clientId,
    })
  }
  async delSingleClient(id) {
    return axios.delete(`${API_PREFIX}/client/${id}`)
  }
  async updateClient(createForm) {
    console.log(createForm)
    return axios.patch(`${API_PREFIX}/client/${createForm.id}`, {
      name: createForm.name,
      paymentGroup: createForm.paymentGroup,
      headName: createForm.headName,
      headPosition: createForm.headPosition,
      legalInfo: {
        name: createForm.legalInfoName,
        paymentAccount: createForm.legalInfoPaymentAccount,
        bankId: createForm.legalInfoBankId,
        bankName: createForm.legalInfoBankName,
        correspondedAccount: createForm.legalInfoCorrespondedAccount,
```

```
        underlyingDocument: createForm.legalInfoUnderlyingDocument,
        inn: createForm.legalInfoInn,
        kpp: createForm.legalInfoKpp,
        ogrn: createForm.legalInfoOgrn,
        okpo: createForm.legalInfoOkpo,
        address: createForm.legalInfoAddress,
        physicalAddress: createForm.legalInfoPhysicalAddress,
      },
      contacts: createForm.contacts,
    })
  }
}
```