МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
им. ГУМАРБЕКА ДАУКЕЕВА»
Кафедра IT – инжиниринг

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой PhD, доцент Досжанова А.А
_____ «____» _____2020 г.

# ДИПЛОМНЫЙ ПРОЕКТ

На тему: «Разработка информационной системы логистической компании с применением современных Web-технологий»
Специальность 5В070300 – Информационные системы
Выполнил Исмадулла Ш. Н. Группа ИС-16-2
Научный руководитель к.т.н., доцент Балгабаева Л.Ш.

Консультанты:
по экономической части: к.э.н., профессор Габелашвили К.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
по безопасности жизнедеятельности: к.т.н доцент Приходько Н.Г
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
по программному обеспечению: ст.преп. Майкотов М.Н
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
Нормоконтролер: ст.преп. Абсатарова Б.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
Рецензент:_____
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ им. ГУМАРБЕКА ДАУКЕЕВА»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070300 – Информационные системы

**ЗАДАНИЕ**
на выполнение дипломного проекта

Студенту Исмадулла Шыңғысхану Нурланұлы

Тема проекта: Разработка информационной системы логистической компании с применением современных Web-технологий

Утверждена приказом по университету № 147 от «11» ноября 2019 г.

Срок сдачи законченного проекта «___» _____ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): постановленные задачи, информационная система «Delivery».

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

а) определение основных целей и требований к ИС, разработка технического задания проекта;

б) проведение анализа существующих систем;

б) разработка информационного обеспечения ИС;

в) разработка интерфейса и программная реализация системы;

г) расчет экономических показателей;

д) расчет показателей по обеспечению безопасности жизнедеятельности.

Основная рекомендуемая литература:

1 PHP 7. В подлиннике Игорь Симдянов, Дмитрий Котеров 2016.

2 Vue.js в действии. - СПб.: Питер, 2019. (Серия «Библиотека программиста»).

3 Effective TypeScript: 62 Specific Ways to Improve Your TypeScript 2019.

4 PHP: объекты, шаблоны и методики программирования 2-е издание Мэтт Зандстра 2015.

5 Официальная документация Zend https://framework.zend.com/learn

6 Официальная документация Vue JS https://vuejs.org/v2/guide/

Консультация по проекту с указанием относящихся к ним разделов проекта

| Раздел | Консультант | Сроки | Подпись |
|---|---|---|---|
| Экономическая часть | Габелашвили К.Р | 02.03.2020 – 10.04.2020 | |
| Охрана труда и БЖД | Приходько Н.Г | 02.03.2020 – 13.04.2020 | |
| Нормоконтроль | Абсатарова Б.Р | 13.05.2020 – 18.05.2020 | |
| Программное обеспечение | Майкотов М.Н | 14.05.2020 – 15.05.2020 | |

ГРАФИК
подготовки дипломной работы (проекта)

| Наименования разделов, перечень разрабатываемых вопросов | Сроки представления научному руководителю | Примечания |
|---|---|---|
| Обзор литературы и анализ существующих систем | 13.01.2020 – 13.02.2020 | |
| Разработка информационного обеспечения системы | 13.01.2020 – 10.03.220 | |
| Разработка программного обеспечения | 13.01.2020 – 30.04.2020 | |
| Экономическая часть | 02.03.2020 – 10.04.2020 | |
| Охрана труда и БЖД | 02.03.2020 – 13.04.2020 | |

Дата выдачи задания «____» _____ 2020 г.

Заведующий кафедрой _____ А.А.Досжанова
(подпись)

Научный руководитель проекта _____ Л.Ш.Балгабаева
(подпись)

Задание принял к исполнению студент_____ Ш.Н.Исмадулла
(подпись)

## Аннотация

Данный дипломный проект предназначен для разработки информационный системы логистической компании ТОО «Asia Freight» с использованием современных Web-технологий. Для реализации проекта использовались современные веб-фреймворки, такие как Vue JS, Zend Framework и библиотека Vuetify для построения пользовательского интерфейса.

Цель данного дипломного проекта заключается в том, чтобы разработать программный продукт, который будет улучшать и автоматизировать бизнес процессы логистической компании.

В двух последних главах рассматриваются вопросы безопасности жизнедеятельности, приводится технико-экономическое обоснование и рассчитывается цена разработки проекта.

## Аңдатпа

Бұл дипломдық жоба заманауи веб-технологияларды қолдана отырып, ЖШС «Asia Freight» логистикалық компаниясы үшін ақпараттық жүйені құруға арналған. Жобаны жүзеге асыру үшін Vue JS, Zend Framework және Vuetify кітапханасы сияқты заманауи веб-құрылымдар пайдаланылды.

Дипломдық жобаның мақсаты логистикалық компанияның бизнес-процестерін жақсартатын және автоматтандыратын бағдарламалық өнімді әзірлеу болып табылады.

Соңғы екі тарауда өмір қауіпсіздігі мәселелері қарастырылған, техникалық-экономикалық негіздеме берілген және жобаны жасау құны есептелінген.

## Annotation

This diploma project is designed to develop an information system for the logistics company Asia Freight LLP using modern Web technologies. To implement the project, modern web frameworks were used, such as Vue JS, Zend Framework and the Vuetify library for building the user interface.

The purpose of this graduation project is to develop a software product that will improve and automate the business processes of a logistics company.

The last two chapters address life safety issues, provide a feasibility study and calculate the cost of developing a project.

## Содержание

## Введение

Данный программный продукт создается для компании ТОО «Asia Freight». ТОО «Asia Freight» была основана 29 апреля 2002 года начав свою деятельность в качестве агента по грузовым перевозкам авиакомпании «Иртыш АВИА». За относительно небольшой период из маленькой фирмы они выросли в крупную транспортную компанию, одного из лидеров отечественного рынка грузовых авиаперевозок. Клиентами компании являются более 4500 организаций и частных лиц. На данный момент штат компании насчитывает более 200 человек, офисы продаж и представительства есть во всех крупных городах Республики Казахстан, а также в Москве.

Asia Freight доставляет грузы благодаря транспортно-экспедиторской службе и специализированным офисам со складскими помещениями в городской черте. А также компания для своих клиентов определяет оптимальный способ и маршрут доставки. Компания перевозит тяжеловесные и негабаритные грузы, скоропортящиеся и требующие специальные условия перевозки. С помощью веб-программы компании клиенты могут оперативно контролировать движение груза. Компания Asia Freight являеnся агентом авиакомпаний AIR ASTANA, SCAT, KLM CARGO, Lufthansa, ТрансАэро и так далее.

Главной целью данной работы является разработка информационной системы логистической компании, соответсвующая всем современным требованиям в веб-разработке. Основным функционалом которого является представление информации о заявках, которые создаются менеджером в системе. Возможность добавление пользователей, групп и ролей для управления данной системой. Кроме того, в информационной системе будет доступна возможность добавление водителей, а также возможность закрепления их заявкам. Водителям доступна принятие заявки на забор или довоз груза по городу. В системе должно быть осуществлено ролевое разделение (менеджер, водитель, администратор).

Основными задачами дипломного проекта являются:

- определение основных целей и требований к ИС (описание постановки задачи и разработка технического задания);
- разработка информационного обеспечения системы (спроектировать функциональную структуру системы, разработать бизнес-модели ИС, описать информационное обеспечение, обосновать выбор СУБД);
- разработка программного обеспечения (описать фреймворк Vue JS, описать возможности и преимущества фреймворка Zend, построить макеты пользовательского интерфейса, соответствующую всем современным требованиям, описать структуру программного обеспечения, обосновать выбор инструментального ПО, описать клиентскую и серверную части, произвести тестирование и отладку ПО с целью выявления ошибок и их дальнейшего предотвращения).

# 1 Теоретическая часть

## 1.1 Постановка задачи

Основными задачами дипломного проекта являются:
- провести сравнительный анализ существующих систем;
- определение основных целей и требований к ИС;
- разработка технического задания;
- разработка информационного обеспечения системы;
- спроектировать функциональную структуру системы;
- разработать бизнес-модели ИС;
- описать информационное обеспечение;
- обосновать выбор СУБД;
- построить макеты пользовательского интерфейса;
- описать фреймворк Vue JS;
- описать структуру программного обеспечения;
- обосновать выбор инструментального ПО;
- описать клиентскую и серверную части;
- произвести тестирование и отладку ПО с целью выявления ошибок и их дальнейшего предотвращения.

## 1.2 Обзор литературы

### 1.2.1 Книга «Vue.js в действии»

Для разработки клиентской части системы нас необходима книга про Vue JS. Очень полезным материалом является книга «Vue.js в действии», авторами которого являются Хэнчеп Эрик, Листуон Бенджамин.

Данная книга разбита на три части. Часть 1 знакомит с Vue.js. В главах 1 и 2 приводится пример создания первого приложение, а также узнаем, что такое экземпляр Vue и как он относится к нашему коду [2].

В части 2, а именно в главах 3-9, мы более подробно рассмотрим концепции представления (View) и модели-представления (ViewModel), а также самые «сочные». аспекты Vue.js. Можно сказать, что часть 1 - всего лишь затравка, а основной курс начинается с части II. Мы начинаем узнавать о сложностях, связанных с созданием программ на Vue.js. Далее идет начало изучения реактивной модели.

Также присутствует рассмотрение формы, поля ввода и привязку информации с помощью мощных директив, встроенных в Vue.js, после чего перейдем к условным выражениям и циклам.

Чрезвычайно важны главы 6 и 7. В них вы научитесь разбивать приложение Vue.js на несколько логических частей, используя компоненты, а также познакомитесь с инструментами сборки, которые понадобятся для построения программ.

Помимо прочего, в главе 7 затрагивается тема маршрутизации. До этого для навигации по приложению будем применять простые условные выражения. Благодаря маршрутизации мы сможем правильно переходить между компонентами и передавать информацию между маршрутами.

В главе 8 мы познакомимся с гибкой анимацией и переходами, которые выполняются в Vue.js. Эти замечательные возможности, встроенные в язык, заслуживают внимания. В главе 9 мы научимся расширять Vue с помощью примесей и пользовательских директив без дублирования кода.

Часть 3 посвящена моделированию данных, работе с API и тестированию. В главах 10 и 11 мы подробно рассмотрим систему управления состоянием в Vue под названием Vuex, а затем попробуем пообщаться с серверной стороной. Мы также познакомимся с Nuxt.js - фреймворком для серверной отрисовки [2].

### 1.2.2 Книга «PHP 7»

Серверная сторона написана полность на фреймфорке Zend Framework, а сам фреймворк написан на языке программирования PHP. Чтобы написание серверной части было правильным и структуированным необходимо иметь под рукой справочник. В качестве такого справочника была выбрана книга «PHP 7» авторами которого являются Котеров Д.В и Симдянов В.И.

В данной книге рассмотрены основы языка PHP и его рабочего окружения в Windows, Mac OS X и Linux.

Отражены радикальные изменения в языке PHP, произошедшие с момента выхода предыдущего издания: трейты, пространство имен, анонимные функции, замыкания, элементы строгой типизации, генераторы, встроенный Web-сервер и многие другие возможности. Приведено описание синтаксиса PHP 7, а также функций для работы с массивами, файлами, СУБД MySQL, memcached, регулярными выражениями, графическими примитивами, почтой, сессиями и т. д. Особое внимание уделено рабочему окружению: сборке PHP-FPM и Web-сервера nginx, СУБД MySQL, протоколу SSH, виртуальным машинам VirtualBox и менеджеру виртуальных машин Vagrant. Рассмотрены современные подходы к Web-разработке, система контроля версий Git, GitHub и другие бесплатные Git-хостинги, новая система распространения программных библиотек и их разработки, сборка Web-приложений менеджером Composer, стандарты PSR и другие инструменты и приемы работы современного PHP-сообщества [1].

В третьем издании добавлены 24 новые главы, остальные главы обновлены или переработаны.

## 1.3 Анализ существующих систем

Высокие требования к качеству конечного продукта промышленного производства, непрерывное удорожание топливно-сырьевых ресурсов, большая доля в себестоимости продукции затраченных на этапе

производства энергетических ресурсов определяют необходимость внедрения новых технологий, как в непосредственном производстве, так и в решении организационных вопросов, затрагивающих всю инфраструктуру предприятия. Промышленное производство характеризуется перемещением больших объемов тяжелых и специфичных по физическим и химическим характеристикам грузов, разнообразием операций и используемых технических средств. Ряд производств, например, металлургические, отличают высокие требования к температурному режиму, определенные требования накладывает непрерывность технологического процесса. К особенностям металлургических и машиностроительных предприятий также относится детализация производственной логистики на межцеховую и цеховую (внутри цеховую), что в свою очередь накладывает свои требования, как к организации снабжения, так и к планированию производства. В результате формируется существенная зависимость времени производства и стоимости продукции от транспортных, складских затрат и других издержек, связанных с логистикой.

Логистические задачи возникают в рамках изменения производственных мощностей и при оптимизации, повышении эффективности организации существующего производства (рисунок 1. 1).


Рисунок 1.1 - Цели, определяющие постановку логистических задач, и пути их достижения

### 1.3.1 Информационные системы логистики

Если учитывать области решаемых задач выделяют виды логистики: закупочную, транспортную, сбытовую (распределительную), производственную, складскую. Кроме того, в свете высоких требований к качеству выделяют сервисное обслуживание, информационную логистику. Помимо рассмотрения отдельных областей логистики как самостоятельных на различных этапах, необходимо их интегрировать в общей системе управления.

Для сравнительного анализа информационных систем в сфере логистики выбраны разработки следующих программных продуктов:

- комплекс программных решений IBM для управления цепочками поставок, который включает в себя:

1) блок планирования цепочки поставок - включает модули оптимизации сети, проектирования и стратегического планирования; планирования маршрутов; оптимизацию запасов; управление принятием ответственных бизнес-решений в новых отраслях;

2) блок поддержания функционирования цепочки поставок - включает модули автоматизации входящих и исходящих транспортных процессов; оптимизацию бизнес-процессов в сетях дистрибуции; оптимизацию процессов входящих и исходящих поставок на основе сквозной наблюдаемости; распространение B2B-озможностей на каждого поставщика:

- Roadnet Transportation Suite - пакет программных продуктов, направленных на оптимизацию транспортной логистики в сфере торговли;

- iSolutions-Логистика - система для расширенного управления складом на базе Microsoft Dynamics AX;

- программы для логистики компании «Первый БИТ» - транспортная и складская логистика;

- E-SKLAD - программное обеспечения для решения задач складской логистики;

- Solvo - автоматизация складских комплексов, автоматизация портов и контейнерных терминалов, управление цепочками поставок;

- DNA evolutions - on-line сервисы для различных оптимизационных задач транспортного планирования: JOpt.NET, JOpt.SDK, JOpt.ASP, Jopt.J2EE;

- JDA - программное решение направлено на бизнес-трансформацию системы поставок. Объектом планирования является цепочка поставок на основе управления спросом, включающая точки продаж, промежуточные склады, распределительные центры, производственные объекты, поставщиков;

- Axapta Retail - система, предназначенная для автоматизации управления на предприятиях крупного и среднего бизнеса, относящаяся к системам класса ERP II;

- Epicor - комплекс отраслевых ERP-систем на основе сервисно-ориентированной архитектуры и веб-сервисов.

Список систем ERP можно продолжить, однако данные системы реализуют в основном логистические функции, связанные с автоматизацией заявок на закупки и продажи, учитывая задачи данного исследования достаточно рассмотреть несколько выбранных примеров.

### 1.3.2 Сравнительный анализ информационных систем логистики

Приведем результаты сравнительного анализа ИС по следующим критериям:

- обеспечение видов логистики (таблица 1.1);
- обеспечение функциональных уровней логистики в рамках одного предприятия, организации (таблица 1.2).

В таблицах 1.1 и 1.2 отмечены виды/уровни, о наличии которых явно свидетельствует описание соответствующих программных продуктов.

Таблица 1.1 - Обеспечение информационными системами различных видов логистики

| Виды логистики | Информационные системы | | | | | | |
|---|---|---|---|---|---|---|---|
| | IBM | Roadnet Transportation Suite | iSolutions-Логистика | DNA evolutions | JDA | Axapta Retail | Epicor |
| Закупочная | + | | | | + | + | + |
| Производственная | ± | | | | | | + |
| Распределительная | | | | | + | ± | ± |
| Складская | + | | + | | | + | + |
| Транспортная | + | + | | + | + | | |
| Сервисное обслуживание | ± | + | | | | | + |

Таблица 1.2 - Обеспечение ИС уровней логистики

| Уровень системы | Информационные системы | | | | | | |
|---|---|---|---|---|---|---|---|
| | IBM | Roadnet Transportation Suite | iSolutions-Логистика | Axapta Retail / Microsoft Dynamics AX | DNA evolutions | JDA | Epicor |
| Оперативный | + | + | + | | | + | ± |
| Диспетчерский | + | + | + | | + | + | + |
| Плановый | + | | | + | | + | + |

Наибольшее отражение в ИС имеют решения задач закупочной, складской (при этом из выборки в данное сравнение включены не все

системы складской логистики), транспортной. Менее всего обеспечены функции производственной и распределительной логистики.

Из вышеприведенных таблиц сравнения ИС по видам логистики и уровням системы видно, что масштабные системы, содержащие несколько разноплановых программных модулей, обеспечивают большинство видов логистики, как правило, на уровне прогнозирования, поиска комплексных эффективных решений для организации в целом, определения стратегии развития. Специализированные программные продукты складской и транспортной логистики ориентированы на обеспечение оперативного и диспетчерского уровня.

В таблицах 1.3 и 1.4 представлены результаты сравнительного анализа возможностей ИС для обеспечения наиболее распространенных видов логистики - транспортной и складской.

Таблица 1.3 - Сравнение программного обеспечения в сфере транспортной логистики

| № п/п | Критерий оценки | Информационные системы | | |
|---|---|---|---|---|
| | | Roadnet Transportation Suie | ИС «Первый БИТ» | DNA evolutions |
| 1 | Разбиение территории на зоны обслуживания | ДА | ДА | ДА |
| 2 | Возможность выбора параметров балансировки разбиения территории | ДА | | |
| 3 | Анализ сценариев | ДА | | |
| 4 | Автоматическая перестройка территорий | ДА | | |
| 5 | Создание планов на разные ситуации | ДА | | |
| 6 | Построение маршрутов, планирование (критерии: набор заказов для развозки, наличие транспортных средств, балансировка использования ресурсов, оговоренное время доставки, минимизация маршрутов и рабочего дня) | ДА | ДА (для одного транспортно-го средства) | ДА |

*Продолжение таблицы 1.3*

| | | | | |
|---|---|---|---|---|
| 7 | Формирование оптимальных схем загрузки товара в транспортное средство (формирование оптимальных вариантов загрузки / разгрузки ТС, уменьшение времени загрузки, сокращение процента повреждения груза при погрузке / разгрузке) | ДА (ориентирован на компании-дистрибьют-оры напитков) | | |
| 8 | Контроль передвижения транспортных средств и персонала (с помощью GPS) | ДА | ДА | |
| 9 | Учет отклонений от заданного маршрута | ДА | | ДА |
| 10 | Контроль выполнения заказов | ДА | ДА | |
| 11 | Учет расхода ГСМ | | ДА | |
| 12 | On-line взаимодействие водителя с логистом | | ДА | |
| 13 | Сбор статистических данных о передвижении, времени, проводимом в пункте назначения | ДА | | |
| 14 | WEB-отчетность, удаленное оформление документов | ДА | | ДА |
| 15 | Мультимодальные перевозки (различные виды транспорта) | | ДА | |

Таблица 1.4 - Сравнение программного обеспечения в сфере складской логистики

| № п/п | Критерий оценки | Информационные системы | | | |
|---|---|---|---|---|---|
| | | «Первый БИТ» | E-SKLAD | Isolutions-Логистика | Solvo |
| 1 | Формирование правил размещения товара на складе | ДА | | ДА | ДА |

*Продолжение таблицы 1.4*

| | | | | | |
|---|---|---|---|---|---|
| 2 | Учет серий и сроков годности при размещении | ДА | | ДА | ДА |
| 3 | Контроль качества товара | ДА | ДА | ДА | ДА |
| 4 | Оптимизация складских запасов за счёт перераспределения товара | ДА | | | ДА |
| 5 | Оптимизация использования складских площадей | ДА | ДА (оптимизация изначального размещения товара на складе) | ДА | ДА (учет массо-габаритных характеристик упаковок товара) |
| 6 | Получение актуальной информации об остатках товара на складе в разрезе адресов хранения | ДА | ДА | ДА | ДА |
| 7 | Оптимизация маршрутов отбора товара по различным критериям (срок годности, партия, зона хранения и т.п.) | | | ДА | ДА |
| 8 | Проведение инвентаризации без остановки работы склада | ДА | | ДА | ДА |

Как указывалось выше, блок внутрипроизводственной (включая цеховую и межцеховую) логистики слабо представлен в ИС. Вместе с тем он имеет особое значение для промышленного производства. Например, решение задач календарного пла-нирования со стальным литьем и прокаткой определяет эффективность производства резки стали.

Организация внутренних поставок оптимизирует временные затраты на основные операции транспортировки, хранения и обеспечивает непрерывность производственного процесса, синхронизирует производственный цикл и сокращает производство бракованных изделий.

Организация внутрипроизводственной логистики позволяет оптимизировать временные затраты на основные, транспортные и складские операции, обеспечить непрерывность производственного процесса, синхронизировать производственные циклы, сократить выпуск бракованной продукции. Все эти комбинации обеспечивают гибкость производства и позволяют адаптировать работу к текущим и рыночным потребностям. В то же время специфика производственного процесса в каждой технологической сети определяет причину низкой распространенности связанных информационных систем.

Таким образом, логистические и организационные задачи управления предприятием отличаются большой разноплановостью. Сравнение программных операций и возможностей в области управления логистикой предприятия показывает следующее.

Большинство разработок сосредоточено на автоматизированных процессах (через создание баз данных и постоянное обновление данных). Решения на основе автоматизации сокращают время, упрощают обработку данных, защищают хранилища и позволяют нам находить нужную информацию в любое время.

Автоматизация является необходимым условием для перехода к следующему этапу управления бизнес-процессами оптимизации на основе использования определенного метода. Использование методов оптимизации позволяет гибко проектировать в различных сценариях развития, не только в текущей ситуации, но и в способности принимать правильные решения в меняющихся обстоятельствах.

Специфика и спектр логистических задач крупных предприятий (например, металлургических) требует использования модульной системы, состоящей из следующих элементов:

1) конструктора моделей;

2) хранилища данных, содержащего как модельные данные, так и первичные данные предприятия о процессах, агрегатах, единицах продукции, транспортных средствах и так далее;

3) оптимизатора, реализующего мультиагентное имитационное моделирование (имитационное моделирования с целью построения адекватных моделей технологических и логистических процессов, агентный подход - для формализации эвристик на элементах модели процесса). Исходя из анализа проведенного выше, было принято решение разработать информационную систему на современных веб-технологиях. Ведь современные проблемы, требуют современных решений.

## 2 Разработка информационного обеспечения системы

### 2.1 Проектирование функциональной структуры

Организационная структура предприятия показана на рисунке 2.1.

Исходя из организационной структуры предприятия выделяются следующие подсистемы информационной системы:

1) подсистема «Заявки». Основные функции данной подсистемы:

- создание заявки, в которой отражается такая информация, как номер заявки, статус, город отправки, город доставки;
- сортировка заявок по временному отрезку, т.е. указываются начальная и конечная даты и система отображает заявки в данном диапазоне;
- фильтрация результата поиска по дате создания и дате обновления;
- отображение заявок, в которых указывается такие данные, как забор из, доставка в, получатель, телефон получателя, количество мест, вес, номер заказа, статус, дата заявки, дата доставки, доставка до;
- отображение заявок-заборов, в них указываются следующие данные: забор из, отправитель, телефон отправителя, количество мест, вес, номер заказа, статус, дата заявки, забрать до;
- отображение заявок-довозов, отображается следующая информация: доставка в, получатель, телефон получателя, количество мест, вес, номер заказа, статус, дата заявки, дата доставки, доставка до;
- возможность смены статуса заявки, всего 4 статуса: сформирован, принят на склад отправления AsiaFreight, передан в регион, доставлено;
- отображение количества заявок, посылок и их веса;
- загрузка выданных заявок в формате Excel.

2) подсистема «Пользователи». Основные функции:

- хранение и вывод информации о количестве пользователей системы;
- добавление пользователя, при добавлении указываются такие данные - как: email, телефон, пароль, роль, город.
- обновление данных пользователя;
- вывод списком и хранения данных о пользователях: номер в списке, email, телефон, статус, роль, город, действие.

3) подсистема «Водители». Данная подсистема реализует те же функции, что и подсистема «Пользователи», но конкретно для пользователя driver. Данная подсистема создается для удобства выполнения статистических работ менеджера.

4) подсистема «Статистика». В данной подсистеме отображаются аналитические сведения в виде графиков (по дням недели, по дням месяца, по месяцам), на которых отображены такие данные как заявок в пути, доставленных в срок, просроченные, просроченные с комментариями. Осуществлен фильтр по временному диапазону.

5) подсистема «Счета». Счета необходимы для оперирования данными в бухгалтерии. Имеется поиск по номеру заказа или же по временному отрезку, отображаются количество посылок и заявок, итог в тенге. В списках указана

такая информация как номер заказа, город доставки, количество мест, сумма за доставку, дата заявки, дата доставки. Также имеется возможность выгрузки данных в Excel.

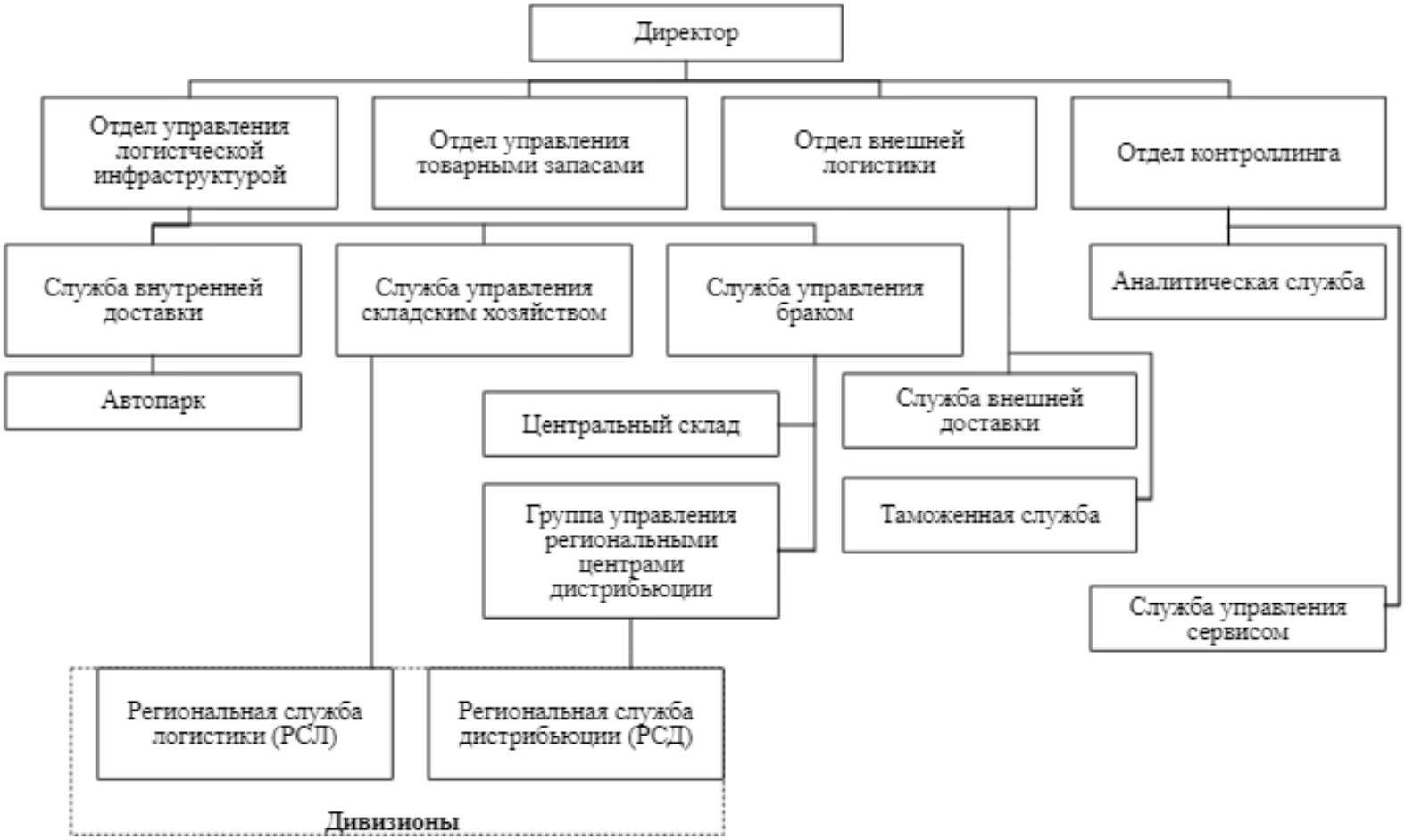



Рисунок 2.1 - Организационная структура компании Asia Freight

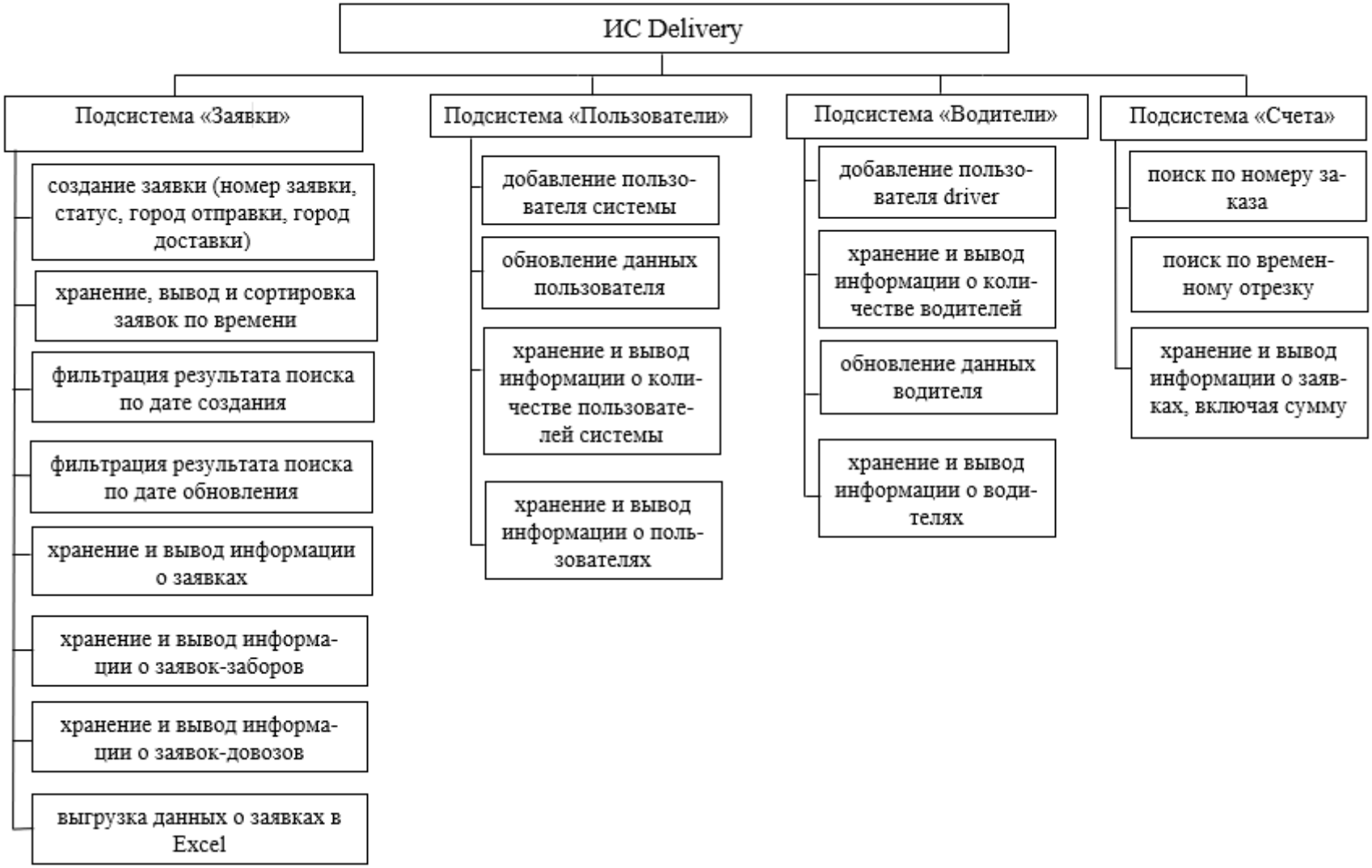



Рисунок 2.2 - Функциональная структура подсистем «Заявки», «Пользователи», «Водители» и «Счета»

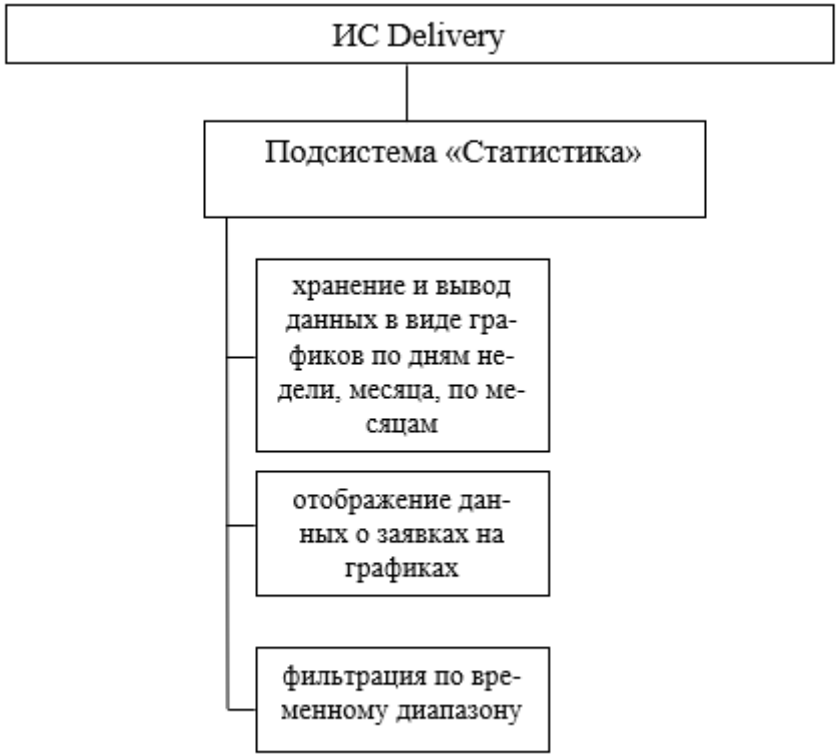



Рисунок 2.3 - Функциональная структура подсистемы «Статистика»

## 2.2 Разработка бизнес модели ИС

### 2.2.1 Функциональная и информационная модели

Пользователь входит на сайт вводя логин и пароль, в зависимости от его роли, если он является менеджером, то он может ввести соответствующие данные о заявке (номер заявки, статус, город отправки, город доставки, начальная дата, конечная дата). Далее менеджер инициализирует событие нажатия кнопки «Обновить». Заполненная форма отправляется веб-серверу. Сервер баз данных обрабатывает данные и возвращает менеджеру список заявок, соответствующий входным данным. Процесс просмотра заявок пользователем показан на рисунке 2.4.

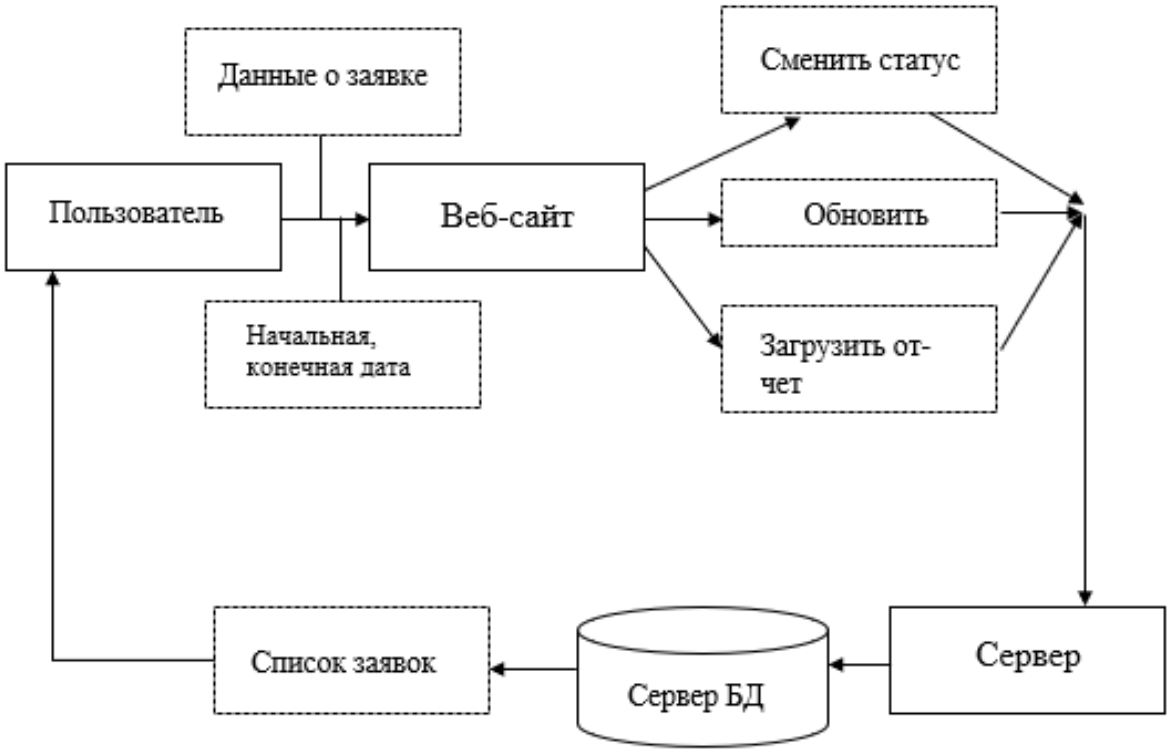



Рисунок 2.4 - Просмотр заявок

Менеджер входит в подсистему «Заявки», инициализирует событие нажатия на кнопку «Создать заявку». Открывается форма для заполнения данных о заявке. После заполнения всех полей менеджер отправляет данные, нажав на кнопку «Добавить». Сервер проверяет легитимность данных, после чего производит DML операцию INSERT для добавления данных о новой заявке. Процесс показан на рисунке 2.5.

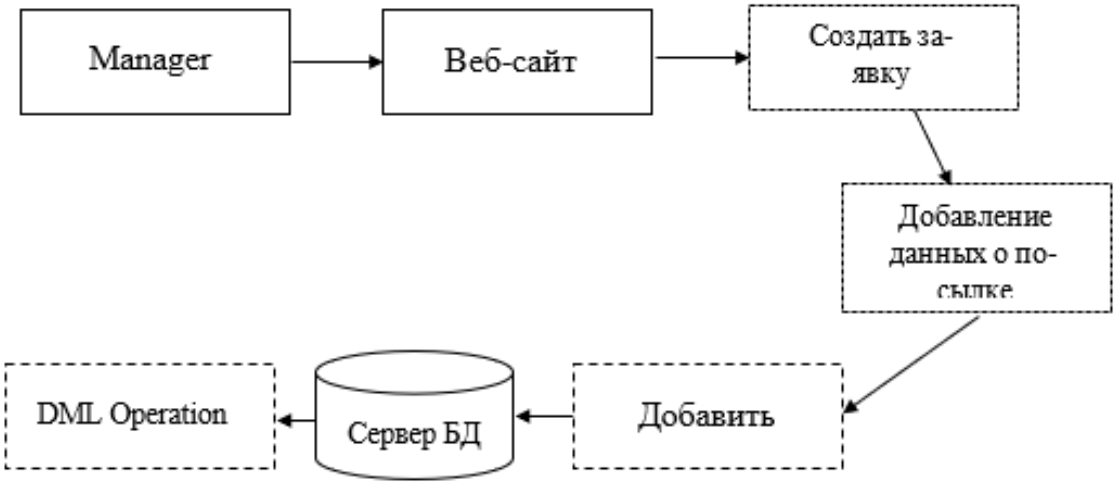



Рисунок 2.5 - Добавление заявки

Добавление пользователя относится к подсистеме «Пользователи». Менеджер входи в подсистему, на веб-странице отображается список пользователей. Также на данной странице имеется функция добавления пользователей. Менеджер инициализирует событие по нажатию кнопки «Добавить пользователя». Он заполняет форму с необходимыми данными (электронная почта, телефон, пароль, роль, город). Далее жмет кнопку «Добавить», данные отправляются серверу БД, где производится DML операция INSERT. Весь процесс показан на рисунке 2.6. Добавление водителя так же происходит по данному принципу. Менеджер входи в подсистему «Водители», на веб-странице отображается список водителей. Также на данной странице имеется функция добавления водителей. Менеджер инициализирует событие по нажатию кнопки «Добавить водителя». Он заполняет форму с необходимыми данными (электронная почта, телефон, пароль, роль, город водителя). Далее жмет кнопку «Добавить», данные отправляются серверу БД, где производится DML операция INSERT. Весь процесс показан на рисунке 2.7.

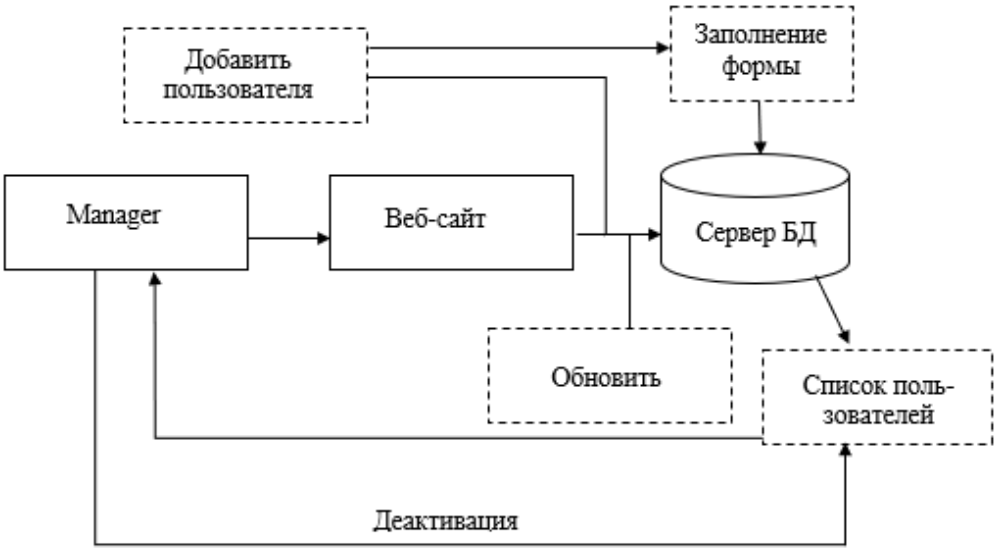



Рисунок 2.6 - Добавление пользователя

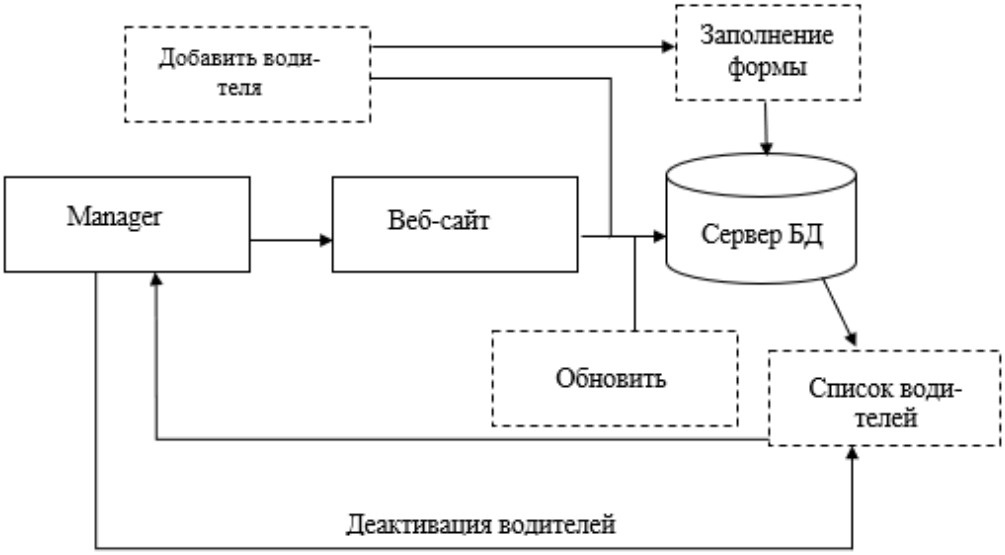



Рисунок 2.7 - Добавление водителя

Подсистема «Статистика» служит для отображения статистических данных в виде графиков. Менеджер отправляет начальную и конечную даты, сервер возвращает все заявки, которые попадают в данный диапазон. На странице имеются три графика. Первый сортирует заявки по дням недели, второй – по числам месяца, третий – по месяцам. Весь процесс показан на рисунке 2.8.

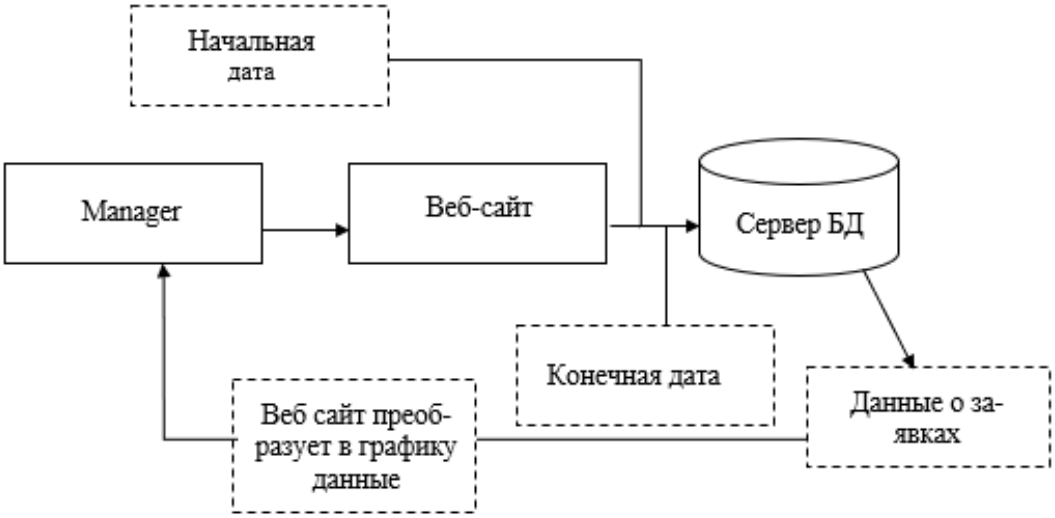



Рисунок 2.8 - Подсистема «Статистика»

## 2.3 Описание информационного обеспечения

### 2.3.1 Диаграмма UML

Диаграмма UML – это диаграмма, основанная на UML (унифицированном языке моделирования), целью которого является визуальное представление системы, визуализация происходит вместе с основными участниками, действиями, ролями, классами и артефактами. Данные диаграммы помогают лучше понимать, поддерживать, изменять и документировать информацию о системе. Основными диаграммами, которые описывают систему, являются диаграммы вариантов использования, диаграммы последовательной и кооперативные диаграммы. Говоря элементарно, UML – это модернизированный подход к программному обеспечению для моделирования и документирования. Де-факто, это одна из самых популярных методик по моделированию бизнес-процессов.

Существуют две наиболее широкие категории – это диаграмма поведенческого UML и диаграмма структурного UML. Они охватывают все другие типы. Исходя из названия, некоторые диаграммы UML пытаются проанализировать и изобразить структуру системы или процесса, в то время как другие описывают поведение системы, ее участников (акторов) и компонентов ее построения. Различные типы разбиты следующим образом[15]:

- диаграмма классов;
- диаграмма компонентов;
- диаграмма композитной/составной структуры;
- диаграмма развёртывания;
- диаграмма объектов;
- диаграмма пакетов;
- диаграмма деятельности;

- диаграмма автомата;
- диаграмма вариантов использования;
- диаграммы коммуникации и последовательности;
- диаграмма обзора взаимодействия;
- диаграмма синхронизации.

### 2.3.2 Построение диаграммы вариантов использования

Краеугольным камнем системы являются функциональные требования, которые система выполняет. Диаграммы прецедентов используются для анализа требований к системе высокого уровня. Эти требования выражены в разных вариантах использования. Необходимо отметить три основные компоненты этой UML-диаграммы[16]:

- функциональные требования – представлены как варианты использования; глагол, описывающий действие;
- акторы, взаимодействующие с системой; актор может быть человеком, организацией или внутренним или внешним приложением;
- отношения между акторами и вариантами использования представлены прямыми стрелками.

В приведенных ниже рисунках показаны диаграммы UML вариантов использования для системы, описанной в техническом задании проекта. В данном случае у нас есть студент, преподаватель и сотрудник. На рисунке 2.9 изображена UML-диаграмма вариантов использования для актора «Менеджер», на рисунке 2.10 – для актора «Водитель», на рисунке 2.11 – для актора «Администратор».

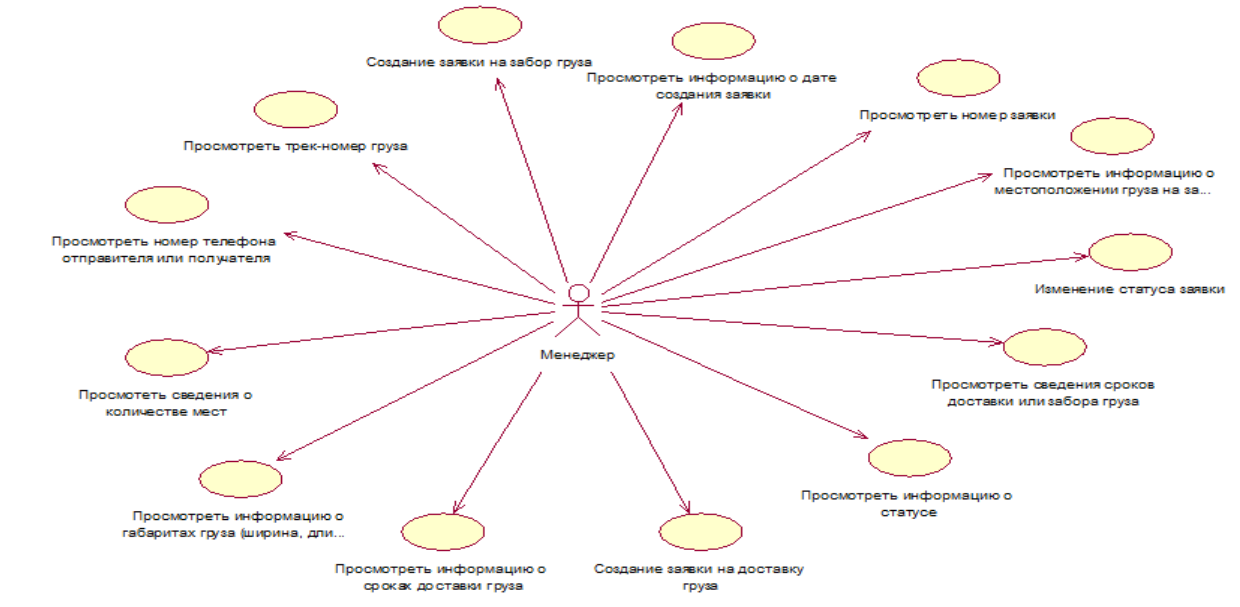



Рисунок 2.9 - UML-диаграмма вариантов использования для актора «Менеджер»

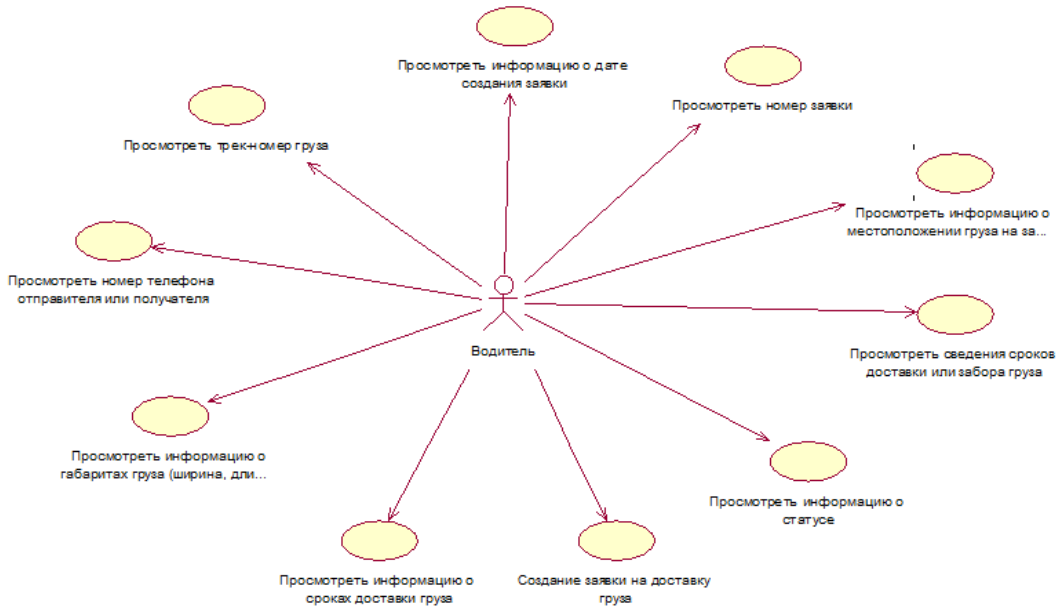



Рисунок 2.10 - UML-диаграмма вариантов использования для актора «Водитель»

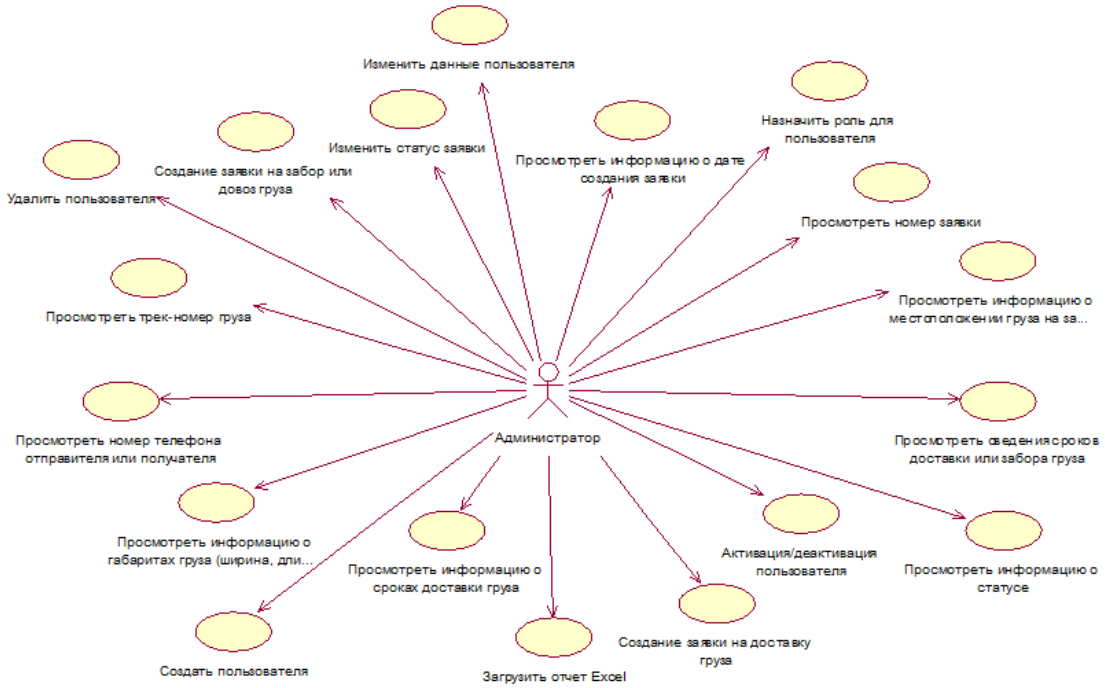



Рисунок 2.11 - UML-диаграмма вариантов использования для актора «Администратор»

Рассмотрим UML-диаграммы вариантов использования конкретно для подсистем основной системы с точки зрения отдельных акторов. На рисунке 2.12 изображена UML-диаграмма вариантов использования для подсистемы «Заявки» актора «Менеджер», на рисунке 2.13 – для «Водитель».

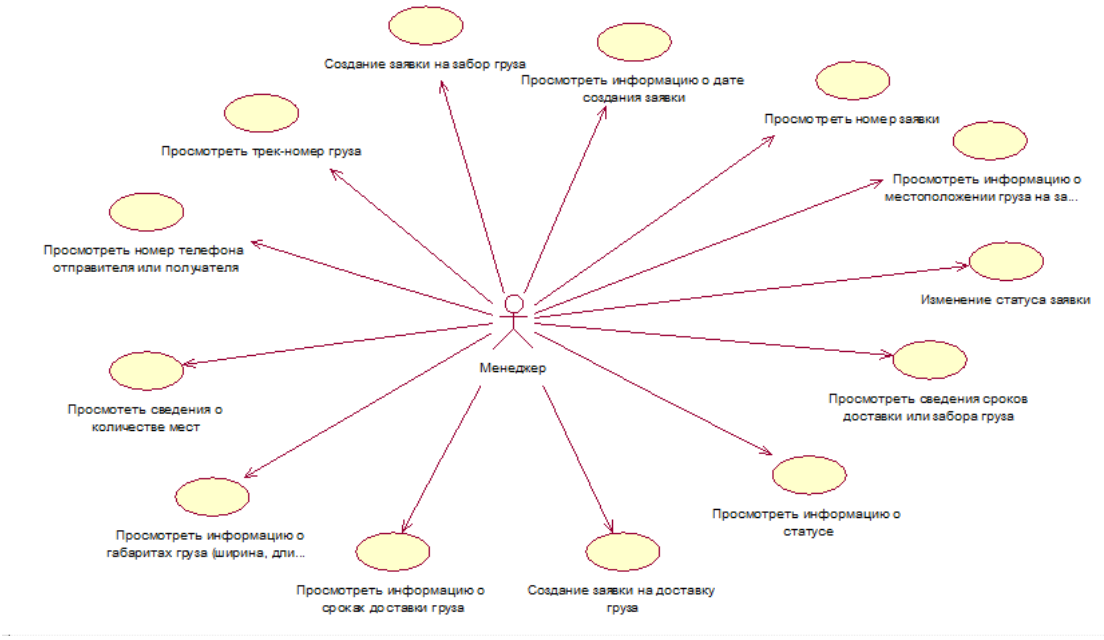



Рисунок 2.12 - UML-диаграмма вариантов использования для подсистемы «Заявки» актора «Менеджер»

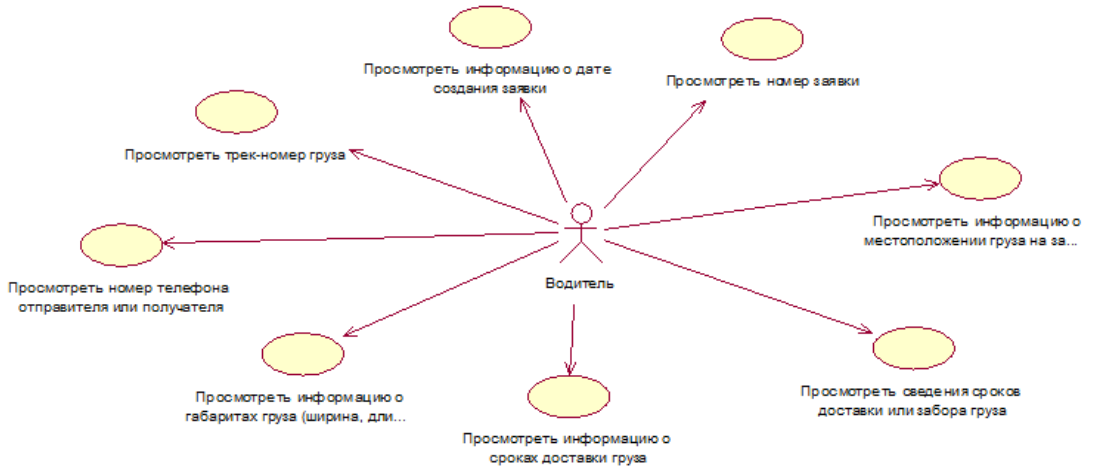



Рисунок 2.13 - UML-диаграмма вариантов использования для подсистемы «Заявки» актора «Водитель»

В круглых контейнерах выражаются действия, которые выполняют акторы. Такими действиями являются: просмотреть сведения о дате создания или изменения заявки, просмотреть сведений о типе заявки, просмотреть информацию о статусе заявки, просмотреть контактные данные клиента сделавший заявку, просмотреть информацию о весе посылки, просмотреть информацию о габаритах груза, просмотреть сроки доставки или забора груза, просмотреть информацию о водителях, к которым прикреплены заявки, создание заявки, добавление типов продукта, просмотреть информацию о пользователях, активация или деактивация пользователей, просмотреть роли пользователей, изменение пользовательских настроек и т.д. Как можно

заметить, UML-диаграммы сценариев использования хороши для демонстрации динамического поведения между участниками внутри системы, упрощая представление системы и не отражая детали реализации.

### 2.3.3 Обоснование выбора СУБД

При выборе системы управления базами данных крайне важно учитывать множество аспектов, каждый из которых накладывает определенные ограничения на выбор продукта. Это могут быть как и технические показатели, так и экономические соображения, учет рынка, а также определенные культурологические факторы, связанные с базой данных.

Исторически сложилось, что MySQL имеет репутацию чрезвычайно быстрой базы данных для тяжелых нагрузок на чтение, иногда за счет параллелизма при смешивании с операциями записи.

PostgreSQL, также известный как Postgres, объявляет себя «самой продвинутой реляционной базой данных с открытым исходным кодом в мире». Он был построен, чтобы быть многофункциональным, расширяемым и соответствующим стандартам. В прошлом производительность Postgres была более сбалансированной - чтение обычно происходило медленнее, чем в MySQL, но оно могло более эффективно записывать большие объемы данных и лучше обрабатывать параллелизм.

Различия в производительности между MySQL и Postgres были в значительной степени стерты в последних версиях. MySQL по-прежнему очень быстро читает данные, но только при использовании старого движка MyISAM. При использовании InnoDB (который допускает транзакции, ключевые ограничения и другие важные функции) различия незначительны (если они вообще существуют). Эти функции крайне важны для приложений масштаба предприятия или потребителя, поэтому использование старого механизма невозможно. С другой стороны, MySQL также был оптимизирован, чтобы уменьшить разрыв, когда дело доходит до тяжелых данных записи.

При выборе между MySQL и PostgreSQL производительность не должна быть фактором для большинства обычных приложений - в любом случае она будет достаточно хорошей, даже если вы учитываете ожидаемый будущий рост. Обе платформы идеально подходят для репликации, и многие облачные провайдеры предлагают управляемые масштабируемые версии любой базы данных. Поэтому стоит рассмотреть другие преимущества Postgres по сравнению с MySQL, прежде чем вы начнете свой следующий проект с настройкой базы данных по умолчанию.

Postgres – это объектно-реляционная база данных, а MySQL - чисто реляционная база данных. Это означает, что Postgres включает такие функции, как наследование таблиц и перегрузка функций, которые могут быть важны для определенных приложений. Postgres также более тесно придерживается стандартов SQL.

Postgres обрабатывает параллелизм лучше, чем MySQL по нескольким причинам:

Postgres реализует Multiversion Concurrency Control (MVCC) без блокировок чтения. Postgres поддерживает параллельные планы запросов, которые могут использовать несколько процессоров / ядер. Postgres может создавать индексы неблокирующим способом (с помощью CREATE INDEX CONCURRENTLYсинтаксиса) и может создавать частичные индексы (например, если у вас есть модель с мягким удалением, вы можете создать индекс, который игнорирует записи, помеченные как удаленные) Postgres известен защитой целостности данных на уровне транзакций. Это делает его менее уязвимым для повреждения данных.

Установка Postgres по умолчанию обычно работает лучше, чем установка MySQL по умолчанию (но вы можете настроить MySQL для компенсации). MySQL имеет несколько странных настроек по умолчанию (например, для кодировки символов и сопоставления).

Postgres очень расширяемый. Он поддерживает ряд расширенных типов данных, недоступных в MySQL (геометрические / ГИС, типы сетевых адресов, JSONB, которые можно индексировать, собственные UUID, временные метки с учетом часовых поясов). Если этого недостаточно, вы также можете добавить свои собственные типы данных, операторы и типы индексов.

Postgres является действительно открытым исходным кодом и управляемым сообществом, в то время как у MySQL были некоторые проблемы с лицензированием. Он был запущен как продукт компании (с бесплатной и платной версией), и приобретение Oracle MySQL AB в 2010 году привело к некоторым опасениям среди разработчиков относительно его будущего статуса с открытым исходным кодом. Тем не менее, существует несколько форков с открытым исходным кодом оригинальной MySQL (MariaDB, Percona и т. Д.), Поэтому в настоящий момент это не считается огромным риском.

Несмотря на все эти преимущества, у Postgres есть некоторые небольшие недостатки, которые следует учитывать.

Postgres по-прежнему менее популярен, чем MySQL (несмотря на то, что он догонял в последние годы), поэтому доступно меньшее количество сторонних инструментов или разработчиков / администраторов баз данных.

Postgres разветвляет новый процесс для каждого нового клиентского соединения, который выделяет нетривиальный объем памяти (около 10 МБ).

Postgres построен с учетом расширяемости, соответствия стандартам, масштабируемости и целостности данных - иногда за счет скорости. Поэтому для простых рабочих процессов, требующих большого количества чтения, Postgres может оказаться более плохим выбором, чем MySQL.

Таким образом, PostgreSQL это не только мощная система управления базами данных, позволяющая обеспечить деятельность организации, но и платформа разработки приложений, требующих использования реляционной СУБД.

Выбор, кроме вышеперечисленных аргументов, учитывался также из того факта, что заказчик использует СУБД PostgreSQL.

### 2.3.4 Описание структур таблиц

База данных информационной системы состоит из 13 таблиц:
- USER (ПОЛЬЗОВАТЕЛЬ);
- CLIENT (КЛИЕНТ);
- ADDRESS (АДРЕС);
- PHONE (ТЕЛЕФОН);
- DRIVER (ВОДИТЕЛЬ);
- PRODUCT_TYPE (ТИП_ПРОДУКТА);
- PACKAGE (ПОСЫЛКА);
- REQUEST (ЗАЯВКА);
- PICKUP (ЗАБОР ГРУЗА);
- DELIVERY (ДОВОЗ_ГРУЗА);
- PAYMENT (СЧЕТ);
- PAYMENT_DETAIL (ДЕТАЛЬ_ОПЛАТЫ);
- PERIOD (ПЕРИОД).

Таблица USER (ПОЛЬЗОВАТЕЛЬ) предназначена для хранения информации об идентификаторе пользователя, его имя пользователя (логин), email, номер телефона, сведения о роли пользователя, а также хеш-пароль для входа в систему. Ее структура приведена в таблице 2.1.

Таблица 2.1 - Структура таблицы USER (ПОЛЬЗОВАТЕЛЬ)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | USER_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор пользователя |
| 2 | EMAIL | VARCHAR2 | 120 | Хранит сведения о email-а пользователя |
| 3 | USERNAME | VARCHAR2 | 120 | Хранит сведения об имени пользователя |
| 4 | PHONE | VARCHAR2 | 120 | Хранит сведения о номере телефона |
| 5 | PASSWORD_HASH | VARCHAR2 | 100 | Хранит информацию о пароле пользователя (хеш-пароль) |
| 6 | STATUS | VARCHAR2 | 25 | Хранит информацию о статусах пользователя |
| 7 | ROLE | VARCHAR2 | 100 | Хранит информацию о ролях пользователя |
| 8 | CREATED_AT | TIMESTAMP | | Хранит информацию о дате создания пользователя с точностью до секунды |

*Продолжение таблицы 2.1*

| | | | | |
|---|---|---|---|---|
| 9 | GROUP_COLUMN | VARCHAR2 | 100 | Хранит сведения о группе, к которой относится пользователь, является также внешним ключом из таблицы GROUP |
| 10 | CITY_UUID | VARCHAR2 | 75 | Хранит сведения об идентификаторе города, к которой относится пользователь, является также внешним ключом из таблицы CITY |

Таблица CLIENT (КЛИЕНТ) предназначена для хранения сведений о клиенте, идентификаторе клиента, имени и фамилии клиента. Ее структура приведена в таблице 2.2.

Таблица 2.2 - Структура таблицы CLIENT (КЛИЕНТ)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | CLIENT_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор клиента |
| 2 | FIRST_NAME | VARCHAR2 | 200 | Хранит сведения об имени клиента |
| 3 | LAST_NAME | VARCHAR2 | 200 | Хранит информацию об фамилии клиента |
| 4 | CREATED_AT | TIMESTAMP | | Хранит информацию о дате создания клиента с точностью до секунды |

Таблица ADDRESS (АДРЕС) предназначена для хранения информации об адресах клиентов, идентификаторе города, улицы, широты, долготы, идентификаторе клиента и информацию о zipcode-е. Ее структура приведена в таблице 2.3.

Таблица 2.3 - Структура таблицы ADDRESS (АДРЕС)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | ADDRESS_UUID | VARCHAR2 | 50 | Является первичным ключом и хранит идентификатор адреса |
| 2 | ADDRESS_NAME | VARCHAR2 | 500 | Хранит информацию об адресе, улице и номере дома |

*Продолжение таблицы 2.3*

| | | | | |
|---|---|---|---|---|
| 3 | ZIPCODE | VARCHAR2 | 50 | Хранит сведения о номере груза |
| 4 | LATITUDE | VARCHAR2 | | Хранит сведения о широте, данные сведения нужны для отрисовки местоположения на карте |
| 5 | LONGITUDE | VARCHAR2 | | Хранит сведения о долготе, данные сведения нужны для отрисовки местоположения на карте |
| 6 | CLIENT_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе клиент, к которому относится адрес, является также внешним ключом из таблицы CLIENT |
| 7 | CITY_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе города, к которой относится клиент, является также внешним ключом из таблицы CITY |

Таблица PHONES (НОМЕРА) предназначена для хранения информации о номерах клиента, идентификаторе клиента к которому они относятся. Ее структура приведена в таблице 2.4.

Таблица 2. 4 - Таблица PHONE (НОМЕР)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | PHONE_UUID | VARCHAR2 | 50 | Является первичным ключом и хранит идентификатор номера телефона |
| 2 | NUMBER | INTEGER | 20 | Хранит информацию о фамилии |
| 3 | CLIENT_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе клиент, к которому относится адрес, является также внешним ключом из таблицы CLIENT |

Таблица DRIVER (ВОДИТЕЛЬ) предназначена для хранения сведений о водителях, идентификаторе водителя, имени и фамилии водителя. Ее структура приведена в таблице 2.5.

Таблица 2.5 - Структура таблицы DRIVER (ВОДИТЕЛЬ)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | DRIVER _UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор водителя |
| 2 | FIRST_NAME | VARCHAR2 | 200 | Хранит сведения об имени водителя |
| 3 | LAST_NAME | VARCHAR2 | 200 | Хранит информацию об фамилии водителя |
| 4 | CREATED_AT | TIMESTAMP | | Хранит информацию о дате создания водителя с точностью до секунды |

Таблица PRODUCT_TYPE (ТИП_ПРОДУКТА) предназначена для хранения информации о типах продукта, идентификатор и наименование типа продукта. Ее структура приведена в таблице 2.6.

Таблица 2.6 - PRODUCT_TYPE (ТИП_ПРОДУКТА)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | PRODUCT_TYPE_UUID | VARCHAR2 | 50 | Является первичным ключом и хранит идентификатор типа продукта |
| 2 | NAME | VARCHAR2 | 200 | Хранит информацию о полном наименовании типа продукта |

Таблица REQUEST (ЗАЯВКА) предназначена для хранения сведений о заявках, типах заявок, идентификаторе заявки, такие данные как полное имя, номер телефона, адрес получателя или отправителя. Ее структура приведена в таблице 2.7.

Таблица 2.7 - Структура таблицы REQUEST (ЗАЯВКА)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | REQUEST_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор заявки |
| 2 | TRACK_NUMBER | VARCHAR2 | 20 | Хранит сведения о трек-номере груза, относящейся к данной заявке |

*Продолжение таблицы 2.7*

| | | | | |
|---|---|---|---|---|
| 3 | CREATED_AT | TIMESTAMP | | Хранит информацию о дате создания заявки с точностью до секунды |
| 4 | UPDATED_AT | TIMESTAMP | | Хранит информацию о дате изменения заявки с точностью до секунды |
| 5 | CREATED_USER_UUID | VARCHAR2 | | Хранит информацию об идентификаторе пользователя, который создал заявку |
| 6 | NOTE | VARCHAR2 | | Хранит информацию комментария, если груз не был получен или доставлен по каким-либо причинам |
| 7 | GROUP_COLUMN | VARCHAR2 | 50 | Хранит сведения о группе, к которой относится пользователь, создавший заявку. Является также внешним ключом из таблицы GROUP |
| 8 | CLIENT_MESSAGE_ID | VARCHAR2 | 50 | Хранит сведения об идентификаторе сообщения, которое отправляется клиенту после успешного забора или довоза груза |
| 9 | STATUS | VARCHAR2 | 50 | Хранит сведения статусе груза, этапы во время забора или довоза груза |

Таблица PICKUP (ЗАБОР ГРУЗА) предназначена для хранения сведений об отправителе груза, то есть вся необходимая информация для водителей, которые осуществляют забор груза. Ее структура приведена в таблице 2. 8.

Таблица 2.8 - Структура таблицы PICKUP (ЗАБОР)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | PICKUP_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор заявки на забор |

*Продолжение таблицы 2.8*

| | | | | |
|---|---|---|---|---|
| 2 | SHIPPER_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе отправителя, является внешним ключом из таблицы CLIENT |
| 3 | ADDRESS | VARCHAR2 | 500 | Хранит сведения о местоположении, где необходимо забрать груз водителем, является внешним ключом из таблицы ADDRESS |
| 4 | PHONE | VARCHAR2 | 50 | Хранит сведения о номере телефона отправителя груза, является внешним ключом из таблицы PHONE |
| 5 | PLANNED_PICKUP_AT | TIMESTAMP | | Хранит информацию о запланированном времени забора груза |
| 6 | PICKED_UP_AT | TIMESTAMP | | Хранит информацию о времени получения груза на забор водителем с точностью до секунды |
| 7 | CITY_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе города, к которой относится клиент, является также внешним ключом из таблицы CITY |
| 8 | DRIVER_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе водителя, к которому относится данная заявка, является также внешним ключом из таблицы DRIVER. Значение по умолчанию NULL |

*Продолжение таблицы 2.8*

| | | | | |
|---|---|---|---|---|
| 9 | REQUEST_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе заявки, к которому относится данная заявка на забор. Является также внешним ключом из таблицы REQUEST |
| 10 | ZIPCODE | VARCHAR2 | 50 | Хранит сведения о номере груза |

Таблица DELIVERY (ДОВОЗ_ГРУЗА) предназначена для хранения сведений о заявках на довоз груза, то есть вся необходимая информация для водителей, которые осуществляют довоз груза. Ее структура приведена в таблице 2.9.

Таблица 2.9 - Структура таблицы DELIVERY (ДОВОЗ_ГРУЗА)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | DELIVERY_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор заявки на довоз |
| 2 | ADDRESS | VARCHAR2 | 60 | Хранит сведения о местоположении, где необходимо забрать груз водителем, является внешним ключом из таблицы ADDRESS |
| 3 | CONSIGNEE_UUID | VARCHAR2 | 100 | Является внешним ключом из таблицы CLIENT и хранит идентификатор клиента |

*Продолжение таблицы 2.9*

| | | | | |
|---|---|---|---|---|
| 4 | PHONE | VARCHAR2 | 100 | Хранит сведения о номере телефона отправителя груза, является внешним ключом из таблицы PHONE |
| 5 | PLANNED_DELIVERY_AT | TIMESTAMP | | Хранит информацию о запланированном времени довоза груза |
| 6 | DELIEVRED_AT | TIMESTAMP | | Хранит информацию о времени получения груза на довоз водителем с точностью до секунды |
| 7 | CITY_UUID | VARCHAR2 | 100 | Хранит сведения об идентификаторе города, к которой относится клиент, является также внешним ключом из таблицы CITY |
| 8 | DRIVER_UUID | VARCHAR2 | 100 | Хранит сведения об идентификаторе водителя, к которому относится данная заявка, является также внешним ключом из таблицы DRIVER. Значение по умолчанию NULL |
| 9 | CONSIGNEE_FULL_NAME | VARCHAR2 | 100 | Хранит сведения о полном имени клиента (получателя), который сделал заявку на довоз груза |

*Продолжение таблицы 2.9*

| | | | | |
|---|---|---|---|---|
| 11 | DELIVERY_UNTIL | TIMESTAMP | | Хранит сведения о сроке в промежутке которого необходимо доставить груз |
| 12 | SIGN_IMAGE_ID | VARCHAR2 | 100 | Хранит идентификатор занятия штрих-кода груза. Является внешним ключом из таблицы SIGN-IMAGE |

Таблица PAYMENT (СЧЕТ) предназначена для хранения сведений о счетах, идентификаторе счета, статус оплаты заявки. Ее структура приведена в таблице 2.10.

Таблица 2.10 - Структура таблицы PAYMENT (СЧЕТ)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | PAYMENT_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор цикла |
| 2 | AMOUNT | INTEGER | | Хранит информацию о количестве посылок |
| 3 | REQUEST_UUID | VARCHAR2 | 50 | Хранит сведения об идентификаторе заявки, к которому относится данная заявка на забор. Является также внешним ключом из таблицы REQUEST |
| 4 | STATUS | VARCHAR2 | 50 | Хранит сведения статусе груза, этапы во время забора или довоза груза |
| 5 | CREATED_AT | TIMESTAMP | | Хранит информацию о дате создания счета с точностью до секунды |

Таблица PACKAGE (ПОСЫЛКА) предназначена для хранения сведений о посылке, трек-номер посылки, габариты груза и идентификатор типа продукта. Ее структура приведена в таблице 2.11.

Таблица 2.11 - Структура таблицы PACKAGE (ПОСЫЛКА)

| № | Наименование поля | Тип поля | Длина | Назначение поля |
|---|---|---|---|---|
| 1 | PACKAGE_UUID | VARCHAR2 | 50 | Является первичным ключом, а также хранит идентификатор посылки |
| 2 | TRACK_NUMBER | VARCHAR2 | 20 | Хранит сведения о трек-номере груза |
| 3 | WIDTH | VARCHAR2 | 20 | Хранит сведения о ширина груза (см.) |
| 4 | HEIGHT | VARCHAR2 | 20 | Хранит сведения о высоте груза (см.) |
| 5 | LENGTH | VARCHAR2 | 20 | Хранит сведения о длине груза (см.) |
| 6 | WEIGHT | VARCHAR2 | 20 | Хранит сведения о весе груза (кг.) |
| 7 | PRODUCT_TYPE_UUID | VARCHAR2 | 20 | Хранит сведения об идентификаторе заявки, к которому относится данная заявка на забор. Является также внешним ключом из таблицы REQUEST |

# 3 Разработка программного обеспечения

## 3.1 Построение пользовательского интерфейса системы.

Перед тем как создать какую-либо страницу для нашей системы, необходимо определить расположение компонентов Vue. Ведь в этом и весь смысл Vue JS, возможность работать компонентным подходом.

### 3.1.1 Макет главной страницы

Макет должен состоять их четырех основных блоков:
Main container (основной контейнер);
- Toolbar (шапка сайта);
- Drawer (панель навигации);
- Content block (блок для контента).

Main container представляет один большой контейнер в котором будут находиться дочерние блоки и компоненты. Например, панель навигации для приложения, блок для основного контента.

Navbar это шапка сайта. Она расположена на одном уровне что и панель навигации и блок для основного контента.

Drawer это блок для панели навигации, она играет немаловажную роль в системе. На ней расположены все доступные страницы. Обычно панель навигации располагается по левому краю страницы и должна иметь свойство сокращать свою ширину.

Content block это блок для основного контента и данный блок занимает самое большое пространство в макете системы.

На рисунке 3.1 показано, как из этих компонентов образуется интерфейс страницы «Заявки».

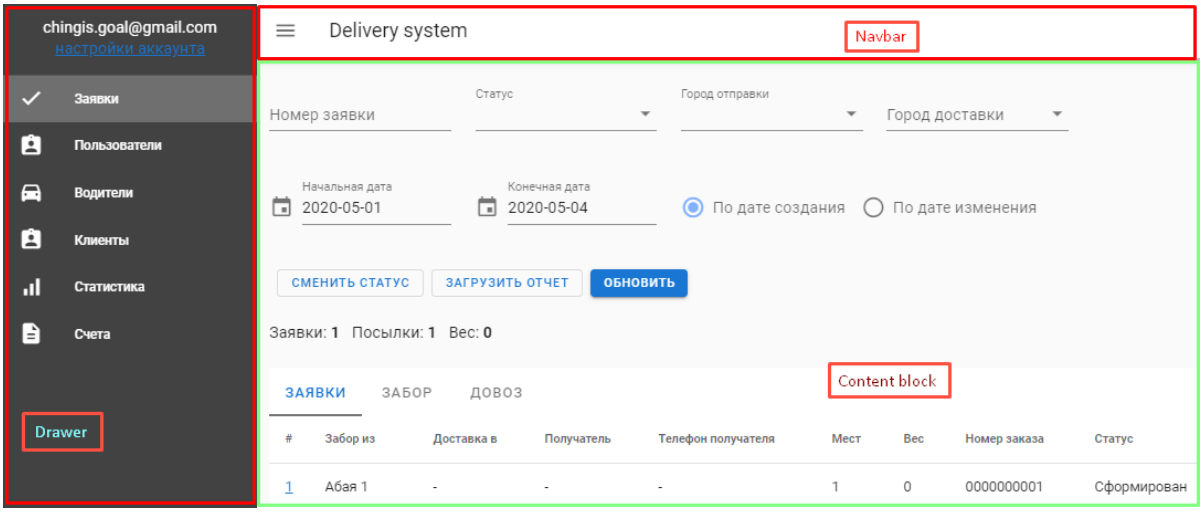



Рисунок 3.1 - Запланированное расположение компонентов

Далее определяются вышеназванные виджеты в файле DashboardLayout.vue. Для определения компонентов во Vue (DashboardLayout.vue) был написан следующий модуль:

```
<template>
 <v-app wrap>
  <dashboard-layout-drawer></dashboard-layout-drawer>
  <dashboard-layout-toolbar ></dashboard-layout-toolbar>
    <v-content>
      <router-view></router-view>
    </v-content>
    <the-loader></the-loader>
    <the-snackbar></the-snackbar>
  </v-app>
</template>

<script lang='ts'>
@Component({
  components: {
    DashboardLayoutDrawer,
    DashboardLayoutToolbar,
    TheSnackbar,
    TheLoader,
  },
})
export default class DashboardLayout extends Vue {}
</script>
```

Данный модуль выполняет функцию позиционирования виджет-объектов на экране. Далее строится иерархия представлений для макета MainActivity, которая показана на рисунке 3.2.

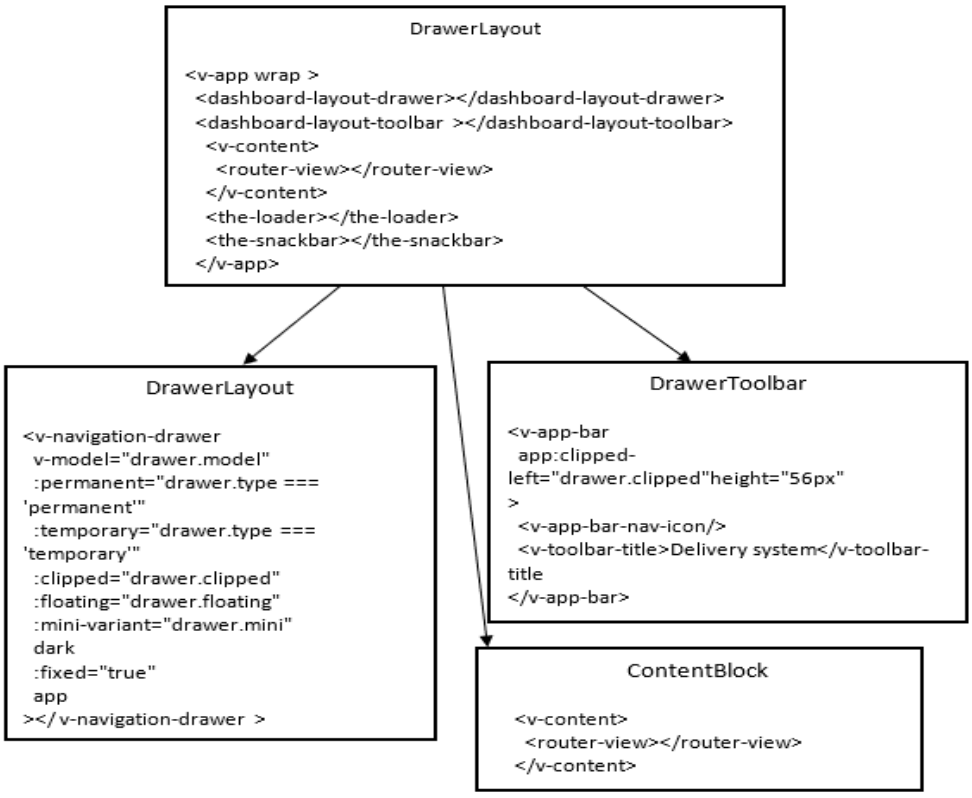



Рисунок 3.2 - Иерархия представлений макета

Ниже показаны основные атрибуты, которые используются в вышеперечисленных компонентах:

- `v-model="drawer.model"`. Используется для для двунаправленного связывания данных с элементами форм input, textarea и select. Способ обновления элемента выбирается автоматически в зависимости от типа элемента;
- `:permanent="drawer.type === 'permanent'"`. Принимает *boolean* тип, панель навигации остаётся видимой независимо от размера экрана;
- `:temporary="drawer.type === 'temporary'"`. Является атрибутом компонента Navigation Drawer. Принимает *boolean* тип, временная панель находиться над своим приложением и использует холст (наложение), чтобы затемнить фон [7];
- `:clipped="drawer.clipped""`. Является атрибутом компонента Navigation Drawer. Принимает *boolean* тип, обрезанный ящик находится под панелью инструментов приложения;
- `:floating="drawer.floating"`. Является атрибутом компонента Navigation Drawer. Принимает *boolean* тип, плавающая панель не имеет видимого контейнера (без границ);
- `:mini-variant="drawer.mini"`. Является атрибутом компонента Navigation Drawer. Принимает *boolean* тип, ширина панели навигации, также принимает модификатор *.sync *. При этом панель снова откроется при нажатии [7];

### 3.2 Описание структуры программного обеспечения

Структура программного обеспечения информационной системы состоит из системного ПО, инструментального ПО и прикладного ПО. Для запуска информационной системы необходимо системное программное обеспечение, включающее в себя операционную систему Windows 7 и выше. Инструментальное программное обеспечение включает в себя IDE JetBrains WebStorm для языка программирования TypeScript, IDE JetBrains PhpStorm 2019 для языка программирования PHP 7.3, СУБД PostgreSQL 10.7, языка программирования TypeScript и серверного языка программирования PHP 7.3. Схематично структура представлена на рисунке 3.3.


Рисунок 3.3 - Структура ПО

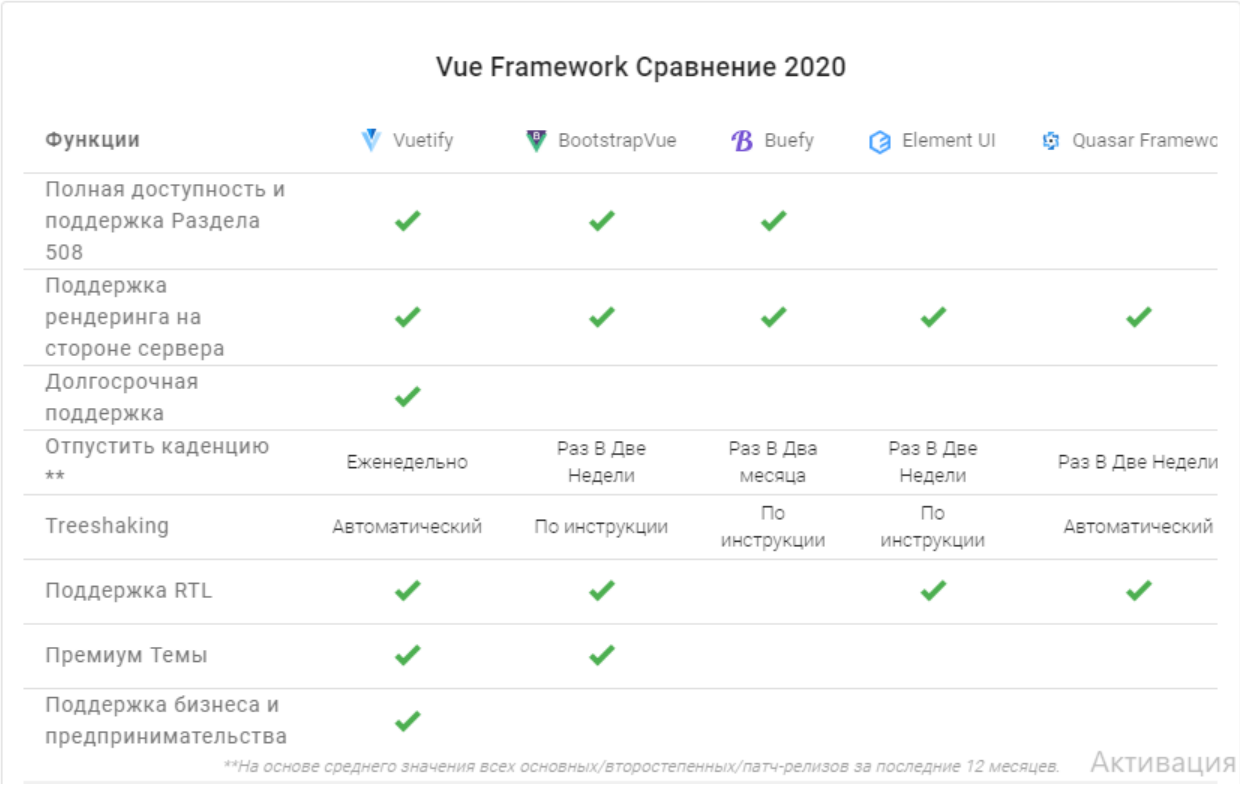
## 3.3 Обоснование выбора частей ПО

### 3.3.1 Vuetify

Интерфейс системы полностью строится на библиотеке Vuetify для Vue.js.

Vuetify является библиотекой №1 для Vue.js и активно разрабатывается с 2016 года. Цель проекта - предоставить пользователям все, что необходимо для создания богатых и интересных веб-приложений, используя спецификацию Material Design .Это достигается благодаря постоянному циклу обновлений, долгосрочной поддержке предыдущих версий, отзывчивому участию сообщества, обширной экосистеме ресурсов и приверженности качественным компонентам.

С момента своего первого релиза в 2014 году, Vue.js вырос до одного из самых популярных JavaScript фреймворков в мире. Одной из причин такой популярности является обширное использование компонентов, которые позволяют разработчикам создавать лаконичные модули, которые можно использовать повторно в своих приложениях. Библиотеки UI – это коллекции таких компонентов, которые реализуют определенные рекомендации по стилю и предоставляют необходимые инструменты для построения многофункциональных веб-приложений. Ниже приведен краткий список лучших UI библиотек для Vue в 2019 году (рис.3.4) [7]:



Vue Framework Сравнение 2020

| Функции | Vuetify | BootstrapVue | Buefy | Element UI | Quasar Framewc |
|---|---|---|---|---|---|
| Полная доступность и поддержка Раздела 508 | ✓ | ✓ | ✓ | | |
| Поддержка рендеринга на стороне сервера | ✓ | ✓ | ✓ | ✓ | ✓ |
| Долгосрочная поддержка | ✓ | | | | |
| Отпустить каденцию ** | Еженедельно | Раз В Две Недели | Раз В Два месяца | Раз В Две Недели | Раз В Две Недели |
| Treeshaking | Автоматический | По инструкции | По инструкции | По инструкции | Автоматический |
| Поддержка RTL | ✓ | ✓ | | ✓ | ✓ |
| Премиум Темы | ✓ | ✓ | | | |
| Поддержка бизнеса и предпринимательства | ✓ | | | | |

***На основе среднего значения всех основных/второстепенных/патч-релизов за последние 12 месяцев.*



Рисунок 3.4 - Сравнение лучших UI библиотек за 2019 год

### 3.3.2 Vue js

Когда JavaScript-фреймворки начали поддерживать асинхронные методы программирования, исчезла необходимость в обновлении всей неб-страницы при каждом запросе. Частичное обновление представления позволяет сайтам и приложениям быстрее реагировать, но в определенной степени приводит к дублированию кода. Логика нредставлсния часто повторяет бюнес-логику [2].

#### 3.3.2.1 Шаблон проектирования MVVM

Шаблон MVVM (Model - View - ViewModel - «модель - представление - модель представления») – это следующий этап в развитии MVC. Его отличительными чертами являются модель представления (ViewModel, VM) и связывание данных. MVVM обеспечивает основу для построения клиентских приложений, которые делают взаимодействие с пользователем и обратную связь более отзывчивыми, избегая при этом затратного дублирования кода в рамках всей архитектуры. Этот шаблон упрощает также модульное тестирование. Но имейте в виду, что для простых пользовательских интерфейсов он может оказаться избыточным.

MVVM позволяет создавать веб-приложения, которые мгновенно реагируют на действия пользователя и позволяют свободно переходить от одной задачи к другой. Модель представления тоже способна играть

несколько ролей(рис.3.5) [2]. Такая консолидация ответственности имеет одно фундаментальное последствие для представлений приложения: при изменении данных внутри VM автоматически обновляются все представления, связанные с этой моделью. Механизм связывания делает данные доступными и гарантирует, что любые изменения будут отражены в представлении.

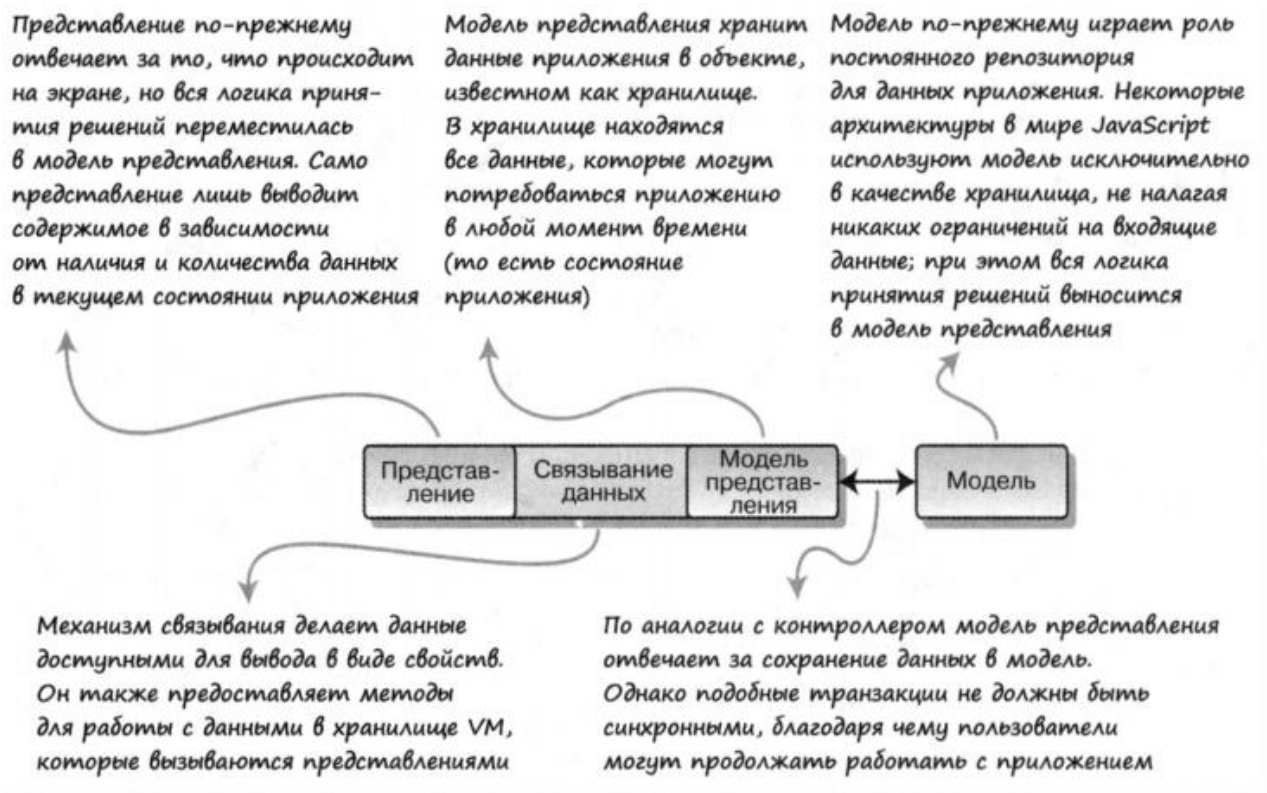



Рисунок 3.5 - Шаблоны проектирования MVVM

### 3.3.2.2 MVVM и реактивность сила Vue js

Библиотеку Vue иногда называют прогрессивным фреймворком. В общем смысле это означает, что для выполнения простых задач ее можно встраивать в существующие веб-страницы, а еще она способна становиться основой для построения крупномасштабных веб-проектов. Какой бы способ применения вы ни выбрали, любое Vue-приложение содержит как минимум один экземпляр Vue. Самые простые программы ограничиваются единственным экземпляром, который связывает заданную разметку и данные, хранящие в модели представления (рис.3.6) [2].

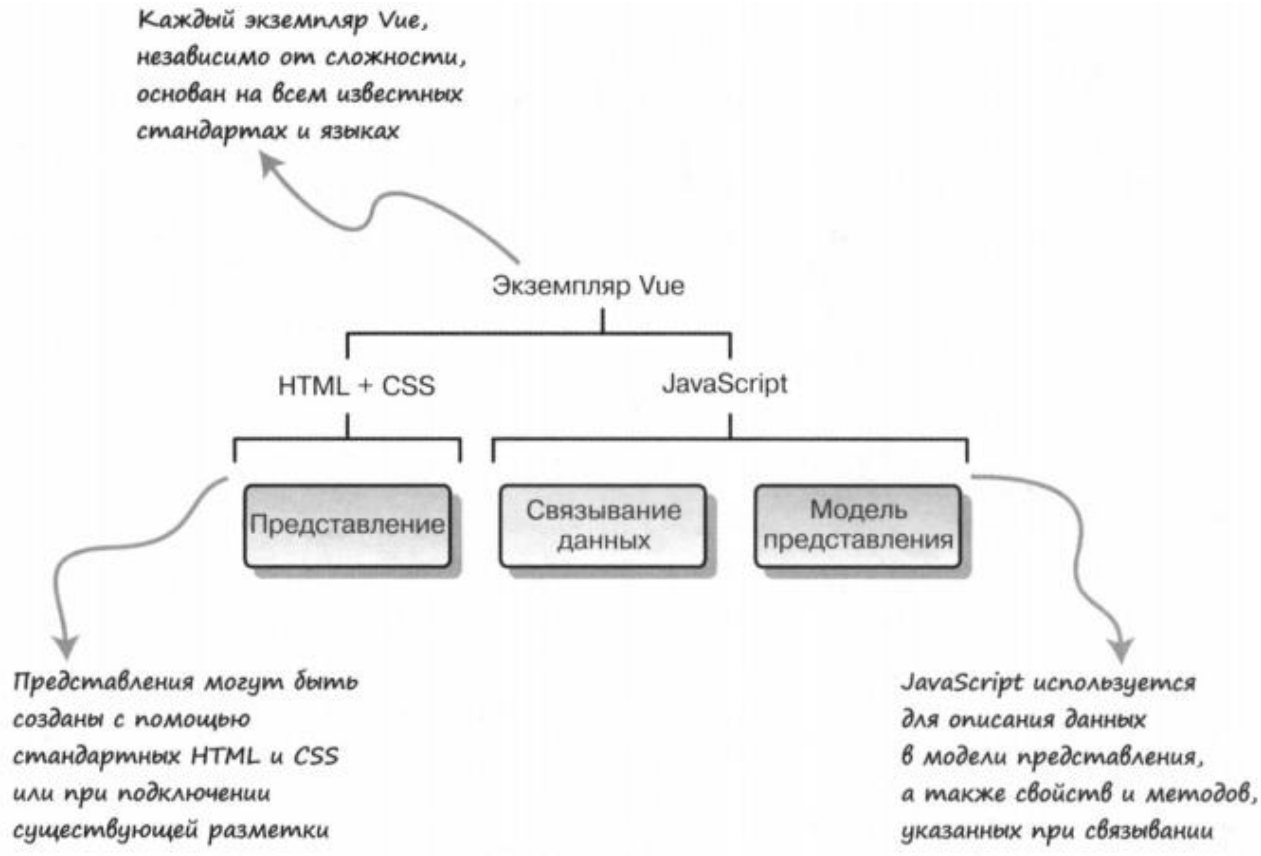



Рисунок 3.6 - Экземпляр Vue связывает разметку с данными в модели представления

Отдельно взятый экземпляр Vue полностью основывается на веб-технологии и существует исключительно в рамках браузера. В сущности, это означает, что для обновления выводимых данных, выполнения бизнес-логики или любой другой задачи, входящей в область ответственности представления или модели представления, не нужно полагаться на генерацию страниц на сервере. Самая поразительная особенность реактивного подхода по сравнению с клиентской архитектурой MVC - то, что на протяжении всего сеанса работы пользователь очень редко сталкивается с перезагрузкой страницы в браузере, если вообще сталкивается. Представление, модель представления и связывание данных реализованы в HTML и JavaScript, поэтому задачи, которые делегируются модели, могут быть асинхронными и не блокировать взаимодействие с пользователем. Когда модель возвращает новые данные, механизм связывания, встроенный в Vue, вносит в представление необходимые изменения [2].

Главная задача Vue - это обеспечение взаимодействия с гюльзователем путем установления и поддержания связи между представлениями, которые мы создаем, и данными в модели представления. Vue обеспечивает надежный фундамент для любого реактивного веб-приложения .

### 3.3.3 TypeScript

TypeScript – это строго типизированный объектно-ориентированный компилируемый язык. Он был разработан *Андерсом Хейлсбергом* (дизайнером C #) в Microsoft. TypeScript – это и язык, и набор инструментов. TypeScript – это типизированный расширенный набор JavaScript, скомпилированный в JavaScript. Другими словами, TypeScript – это JavaScript плюс некоторые дополнительные функции. Возможности TypeScript визуально показаны на рисунке 3.7.

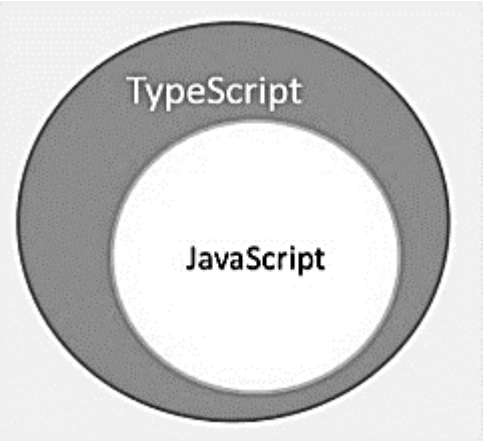



Рисунок 3.7 - Возможности TypeScript

***Особенности TypeScript.***

*TypeScript это просто JavaScript*. TypeScript начинается с JavaScript и заканчивается JavaScript. Typescript принимает основные строительные блоки вашей программы из JavaScript. Следовательно, вам нужно знать только JavaScript, чтобы использовать TypeScript. Весь код TypeScript преобразуется в его эквивалент JavaScript для целей исполнения.

*TypeScript поддерживает другие библиотеки JS*. Скомпилированный TypeScript можно использовать из любого кода JavaScript. Сгенерированный TypeScript JavaScript может повторно использовать все существующие платформы JavaScript, инструменты и библиотеки.

*JavaScript это TypeScript*. Это означает, что любой допустимый файл .js может быть переименован в .ts и скомпилирован с другими файлами TypeScript.

*TypeScript является переносимым*. TypeScript переносим через браузеры, устройства и операционные системы. Он может работать в любой среде, в которой работает JavaScript. В отличие от своих аналогов, TypeScript не требует выделенной виртуальной машины или определенной среды выполнения для выполнения [3].

***TypeScript и ECMAScript.***

Спецификация ECMAScript – это стандартизированная спецификация языка сценариев. Опубликовано шесть изданий ECMA-262. Версия 6 стандарта имеет кодовое название «Гармония». TypeScript соответствует спецификации ECMAScript6. Разбор языка TypeScript показан на рисунке 3.8.


Рисунок 3.8 - Состав языка TypeScript

TypeScript использует базовые языковые функции из спецификации ECMAScript5, то есть официальную спецификацию для JavaScript. Функции языка TypeScript, такие как модули и ориентация на основе классов, соответствуют спецификации EcmaScript 6. Кроме того, TypeScript также включает в себя такие функции, как обобщения и аннотации типов, которые не являются частью спецификации EcmaScript6 [3].

Возникает вопрос, почему именно стал основным языком программирования для разработки клиентской части системы. TypeScript превосходит другие аналоги, такие как языки программирования CoffeeScript и Dart, в том смысле, что TypeScript является расширенным JavaScript. Напротив, такие языки, как Dart, CoffeeScript, сами по себе являются новыми языками и требуют специфичной для языка среды выполнения.

***Преимущества TypeScript включают:***

*Компиляция* – JavaScript – это интерпретируемый язык. Следовательно, это должно быть выполнено, чтобы проверить, что это действительно. Это означает, что вы пишете все коды, просто чтобы не найти выходных данных, на случай, если возникнет ошиб-ка. Следовательно, вам нужно часами пытаться найти ошибки в ко-де. Транспортер TypeScript предоставляет возможность проверки оши-бок. TypeScript скомпилирует код и сгенерирует ошибки компиляции, если обнаружит какие-то синтаксические ошибки. Это помогает выде-лить ошибки перед запуском скрипта.

*Строгая статическая типизация* – JavaScript не является строго типизированным. TypeScript поставляется с дополнительной системой статической типизации и вывода типов через TLS (служба языка TypeScript). Тип переменной, объявленной без типа, может определяться TLS на основе ее значения.

TypeScript *поддерживает определения типов* для существующих библиотек JavaScript. Файл определения TypeScript (с расширением .d.ts ) содержит определение для внешних библиотек JavaScript. Следовательно, код TypeScript может содержать эти библиотеки.

TypeScript *поддерживает* такие концепции *объектно-ориентированного программирования*, как классы, интерфейсы, наследование и так далее.

### 3.3.4 Hypertext Preprocessor (PHP)

Серверная часть информационной системы написан на языке программирования PHP версии 7.3. PHP – это язык сценариев общего назначения, который обычно используется в веб-разработке на стороне сервера. Созданный в 1994 году Расмусом Лердорфом, PHP начал свою жизнь как набор сценариев, известных как «Инструменты для персональных домашних страниц». В 1995 году Расмус расширил свой набор для добавления функциональности и выпустил исходный код для общественности.

Языки сценариев (семейство языков программирования, включая PHP, а также такие языки, как JavaScript и Ruby) представляют собой подмножество языков программирования, используемых для автоматизации процессов, которые в противном случае должны выполняться пошагово в коде сайта при каждом их возникновении.

В отличие от JavaScript, который на стороне клиента, PHP на стороне сервера. Когда браузер запрашивает информацию с сервера, сервер выполняет код и отправляет результат клиенту. Обычно вывод представляет собой файл HTML, но он также может включать CSS и JavaScript. Браузер использует эту информацию для создания веб-страницы.

PHP включает в себя такие вещи, как диалоговые окна, открывающиеся на экране в ответ на действия пользователя, чат-боты, отвечающие на определенное поведение пользователя соответствующими сообщениями, или анимацию, которая происходит, когда пользователь прокручивает страницу выше определенной точки - любые динамические функции веб-сайта, которые должны происходят на экране без необходимости перезагрузки сайта пользователем. Языки сценариев, такие как PHP, отличаются от языков разметки, таких как HTML и CSS, в том смысле, что в то время как HTML и CSS определяют макет и внешний вид веб-страниц, языки сценариев сообщают статической веб-странице (созданной с использованием HTML и CSS) «выполнять» определенные действия.

PHP предназначен для большого количества полезных задач. Эти задачи помогают разработчикам добавлять динамический контент и улучшать взаимодействие с пользователем:

- PHP может использоваться для взаимодействия с файлами на сервере (создание, открытие, чтение, запись);
- PHP может использоваться для отправки и получения файлов cookie;
- PHP может использоваться для доступа и изменения данных, хранящихся в базе данных;
- PHP может быть использован для создания динамического контента;
- PHP может использоваться для создания зон только для членов на вашем сайте и управления аутентификацией;
- PHP может использоваться для генерации документов Excel и PDF.

PHP анализируется и выполняется быстро, поэтому по сравнению с загрузкой статических файлов производительность снижается незначительно. Поскольку PHP встроен в HTML, он также быстро развивается; это позволяет

легко добавлять PHP к существующим веб-сайтам без необходимости переписывать все.

Также PHP работает на многих платформах, совместим с современными браузерами и поддерживает широкий спектр баз данных.

PHP-скрипт может автоматически отображать три ваших последних записи в блоге на главной странице вашего сайта. В этом случае сами сообщения хранятся на сервере сайта и вызываются, когда они занимают один из трех последних опубликованных слотов. Это позволяет избежать как предварительной загрузки сообщений на вашем сайте, так и необходимости администратора сайта загружать и обновлять сообщения при публикации новых историй. PHP-скрипты могут также включать условные операторы (if / else / endif), которые предписывают вашему сайту изменять его отображение и добавлять контент с вашего веб-сервера по мере необходимости. Это может включать такие действия, как указание, что если администратор сайта загрузит ссылку на видео в поле «x», сайт загрузит видео со своего сервера и отобразит его для пользователя. Далее сценарий может указать, что если администратор не загрузит ссылку, то на странице будет отображаться изображение «y» по умолчанию. Серверные действия PHP предоставляют веб-сайту совершенно новый уровень динамических возможностей (помимо статических функций, предлагаемых HTML и CSS, и даже динамического содержимого на стороне клиента, которое стало возможным благодаря JavaScript).

Любая веб-страница PHP сначала взаимодействует с переводчиком после запроса, который я отправил из браузера. Этот интерпретатор, в свою очередь, взаимодействует с файловыми системами, базами данных и почтовыми серверами. Для настройки PHP используется файл php.ini. Интерпретатор использует этот INI-файл для проверки настроек. PHP интерпретирует операторы процессов, заключенные в <?php?>. Каждое утверждение заканчивается точкой с запятой, которая указывает интерпретатору, что это конец утверждения.

Разрешения в PHP очень похожи на UNIX. Каждый файл имеет три типа прав доступа – чтение, запись и выполнение. Разрешения могут быть изменены с помощью режима изменения или команды CHMOD. За CHMOD следует 3-значный номер, указывающий тип разрешения. CHMOD 777 предназначен для чтения, записи и выполнения.

### 3.3.5 Zend Framework

При разработке серверной части за основу брался шаблон проектирования на фреймворке Zend. Zend framework – программный каркас написанный на языке PHP, который применяется для разработки интернет-проектов. В основу положен принцип MVC (распределяет интерфейс приложения и его логику), также имеет много библиотек и компонентов, которые позволяют интегрироваться с посторонними проектами, к примеру,

YouTube. Начиная с версии 1.6 включает JavaScript-фреймворк Dojo. А в 2012 году вышла вторая полноценная версия Zend [5].

Особенности Zend framework:

- написан полностью на PHP 5 и E_STRICT-совместим;
- части проекта мало зависят друг от друга;
- поддерживаются много видов СУБД;
- поддерживаются почтовые протоколы mbox, Maildir, POP3 и IMAP4;
- наличие гибкой системы кэширования, которая поддерживает различные типы (в памяти или в файловой системе).

Zend framework следует использовать по нескольким причинам. Прежде всего, это максимальное соответствие стандартам, которое предоставляет фреймворк и возможность многократно использовать код. Ведь часто возникают задания, где необходимо проделывать однотипные процедуры. При этом нет необходимости подстраиваться под логику новой задачи, можно использовать существующий код, который был написан для предыдущей подобной задачи. Кроме этого, фреймворк имеет открытый исходный код и большое сообщество разработчиков.

### 3.4 Клиентская часть

На этапе проектирования интерфейса информационной системы были созданы пользовательские компоненты Vue, теперь необходимо написать код для динамической работы этих компонентов. Код, который представлен в листинге 1 необходим для функционирования авторизации. Данный код использует основные конструкции, представленные в таблице 3.1.

Таблица 3.1 - Основные конструкции класса LoginPage

| № | Конструкция | Пояснения |
|---|---|---|
| 1 | `public form: AuthForm = {`<br>`  identifier: '', password: ''`<br>`};` | Объект *form*, хранящий идентификатор и пароль пользователя для входа в систему, который построен по интерфейсу *AuthForm.* |
| 2 | `public showPassword: boolean = false;` | Переменная *showPassword,* который имеет тип *Boolean.* |

*Продолжение таблицы 3. 1*

| | | |
|---|---|---|
| 3 | private async login() {<br>try{<br>const success = await this.authRequest(this.form);<br>if (success instanceof Success) {<br>this.$store.commit('app/showMessage', ['success', 'Успешный вход в систему']);<br>this.$router.push('/');<br>}<br>}<br>catch (error) {<br>alert('Неправильный логин или пароль.'); | Метод *login*, который отправляет запрос для входа в систему. В зависимости от ответа сервера данный метод оповещает пользователя успешным входом или наоборот. |

Код, представленный в листинге 2 необходим для извлечения вывода информации о заявках с сервера. Данный код использует основные конструкции, представленные в таблице 3.2.

Таблица 3.2 - Основные конструкции класса RequestList

| № | Конструкция | Пояснения |
|---|---|---|
| 1 | public async fetchDeliveryCities() {<br>***requestService***.fetchDeliveryCities()<br>.then((response: any) => {<br>if (response.data.length > 0) {<br>this.deliveryCities.push('');<br>for (const city of response.data) {<br>this.deliveryCities.push(city.name);<br>}<br>}<br>})<br>.catch((error: any) => {<br>this.$store.commit('app/showMessage', ['error', error.response.data.message]);<br>});<br>} | Метод *fetchDeliveryCities*, который отправляет GET запрос для получения массива всех городов. |
| 2 | public acceptRequestFromDriver() {<br>this.trackNumber = '';<br>***requestService***.acceptRequestFromDriver(this.pickupOrDeliveryUuid, this.currentUserUuid, this.isPickup)<br>.then((response: any) => {<br>this.acceptedRequestDialog = false; | Метод *acceptRequestFromDriver*, который отправляет POST запрос с данными водителя для закрепления водителя к заявке. |

*Продолжение таблицы 3. 2*

| | | |
|---|---|---|
| 3 | `public async downloadExcelReport() { const search: Search = { trackNumber: this.trackNumber, status: this.status, pickupCity: this.pickupCity, deliveryCity: this.deliveryCity, startDate: this.startDate, endDate: this.endDate, byUpdatedAt: this.byUpdatedAt, }; try { const result = await requestService.downloadReport(search); // TODO: save file saveAs(result.data, 'Отчет о заявках.xlsx'); }` | Метод *downloadExcelReport*, который отправляет POST запрос по объекту *search* для скачивания отчет в формате Excel. |
| 4 | `public async getRequestData() { let pickupAddress = ' - '; let deliveryAddress = ' - '; let deliveryFullName = ' - '; let deliveryPhone = ' - '; let deliveredAt = ' - '; let deliveryUntil = ' - '; requestService.selectRequestByUuid(this.requestUuid) .then((response: any) => { this.requestDataLoading = false; if (response.data.pickup) { pickupAddress = response.data.pickup.city; } { if (this.isPickup) { this.pickupOrDeliveryUuid = response.data.pickup.uuid; } else { this.pickupOrDeliveryUuid = response.data.delivery.uuid; .catch((error) => { this.$store.commit('app/showMessage', ['error', error]);` | Метод *getRequestData*, который отправляет запрос для получения данных о заявках, а также где идет проверка на значения некоторых полей *null*, если значение *null*, то вместо *null* ставится «–», а также инициализация объекта *request*, который построен на интерфейсе *RequestData*. |

*Продолжение таблицы 3. 2*

| | | |
|---|---|---|
| 5 | get deliveryTableData() {<br>const tableData: DeliveryRequestTableRowData[] = [];<br>for (const request of this.requests) {<br>let deliveredAtVal = ' - ';<br>let driver = null;<br>if (request.delivery) {<br>if (this.currentUserCity === request.delivery.city \|\| this.currentUserRole !== 'driver') {<br>if (request.delivery.deliveredAt !== null) {.fullName,<br>phone: request.delivery.phone,<br>requestPackagesAmount: request.amountPackages,<br>requestTotalWeight: request.packages.reduce((a, b) => a + (b.weight \|\| 0), 0); | Computed свойство *deliveryTableData*, которое преобразует данные о заявках типа довоз, для удобного отображения в таблице. |
| 6 | get pickupTableData() {<br>const tableData: PickupRequestTableRowData[] = [];<br>for (const request of this.requests) {<br>let driver = null;<br>if (request.pickup !== null) {<br>if (this.currentUserCity === request.pickup.city \|\| this.currentUserRole !== 'driver') {<br>if (request.pickup.driver !== null) {<br>driver = request.pickup.driver.fullName;<br>driver,<br><br>}<br>}<br>return tableData;<br>} | Computed свойство *pickupTableData*, которое преобразует данные о заявках типа забор, для удобного отображения в таблице. |

*Продолжение таблицы 3. 2*

| | | |
|---|---|---|
| 7 | public async addComment() {<br><br>***requestService***.addNote(this.requestUuid, this.note)<br>.then((response: any) => {<br><br>this.$store.commit('app/showMessage', ['success', 'Ваш комментарий успешно добавлен']);<br>this.deleteCommentBtn = true;<br>this.fetchRequestList();<br>})<br>.catch((error) => {<br><br>this.$store.commit('app/showMessage', ['error', error.response.data.message]);<br>});<br>} | Метод *addComment*, который отправляет запрос для добавления комментария к заявке. Это необходимо в том случае, когда получатель или отправитель по каким-либо обстоятельствам не передает или не получает посылку (например, получатель не оказался дома). |

Код в листинге 3 позволяет пользователю с ролью «Менеджер» создавать заявки. Данный класс содержит следующие основные конструкции, представленные в таблице 3.3.

Таблица 3.3 - Основные конструкции класса RequestAddPage

| № | Конструкция | Пояснения |
|---|---|---|
| 1 | public fetchTypes(): void {<br>***requestService***.fetchPackageTypes()<br>.then((response) => {<br>this.types = response.data;<br>});<br>} | Метод *fetchTypes*, который отправляет GET запрос для получения массива всех типов посылок. |
| 2 | public addRequest() {<br>if ((this.$refs.form as Vue & { validate: () => boolean }).validate()) {<br>this.saveButtonLoader = true;<br>this.$store.commit('loader', true);<br>if (this.$store.state.pickupFormShowed \|\| this.$store.state.deliveryFormShowed) {<br><br>***requestService***.addRequest(this.packages)<br>.then((response) => {<br>const uuid = response.data.uuid; | Метод *addRequest*, который отправляет POST запрос с необходимыми данными для добавления заявки. |

Код, представленный в листинге 4 необходим для извлечения вывода информации о клиентах с сервера. Данный код использует основные конструкции, представленные в таблице 3.4.

Таблица 3.4 - Основные конструкции класса ClientList

| № | Конструкция | Пояснения |
|---|---|---|
| 1 | public fetchClients(): void {<br>***clientService***.fetchClients()<br>.then((response) => {<br>this.clients = response.data.clients;<br>})<br>.catch((error) => {<br>this.errorText = error.response.data.message;<br>});<br>} | Метод *fetchClients*, который отправляет GET запрос для получения списка всех клиентов. |
| 2 | get transformedClients() {<br>const transformedClients: TransformedClient[] = [];<br>for (const client of this.clients) {<br>let clientPhones: string = '';<br>let clientAddresses: string = '';<br><br>if (client.phones && client.phones.length > 0) {<br>clientPhones = client.phones.map((p) => p.number).join(', ');<br>}<br>if (client.addresses && client.addresses.length > 0) {<br>clientAddresses = client.addresses.map((a) => a.city.name + ', ' + a.name).join('; ');<br>}<br>const filteredClient: TransformedClient = {<br>uuid: client.uuid,<br>fullName: client.firstName + ' ' + client.lastName,<br>phones: clientPhones,<br>addresses: clientAddresses,<br>};<br><br>transformedClients.push(filteredClient);<br>}<br>return transformedClients;<br>} | Computed свойство *transformedClients*, которое преобразует полученные данные клиентов под объект *transformedClients*, который в свою очередь построен по интерфейсу *TransformedClient* для удобного отображения данных в таблице. |

Код, представленный в листинге 5 необходим для добавления клиентов в базу данных. Данный код использует основные конструкции, представленные в таблице 3.5.

Таблица 3.5 - Основные конструкции класса NewClient

| № | Конструкция | Пояснения |
|---|---|---|
| 1 | public addClient() {<br><br>***clientService***.addClient(this.clientForm)<br>.then((response) => {<br>if (response.status === 201) {<br>const uuid = response.data.uuid;<br>if (!(this.clientPhones.length === 0 \|\| this.clientAddresses.length === 0)) {<br>this.allSaved = true;<br>}<br>if (this.clientPhones.length > 0) {<br>this.savePhone(uuid);<br>}<br>if (this.errors.length === 0) {<br>if (this.clientAddresses.length > 0) {<br>this.saveAddress(uuid);<br>}<br>}<br>}<br>})<br>.then(() => {<br>if (this.clientPhones.length === 0 && this.clientAddresses.length === 0) {<br>this.saveButtonLoader = false;<br>this.$store.commit('loader', false);<br><br>this.$store.commit('app/showMessage', ['success', 'Успешно сохранено']);<br>}<br>});<br>} | Метод *addClient*, который отправляет POST запрос с данными клиента для добавления клиентов в базу данных. |

*Продолжение таблицы 3. 5*

| | | |
|---|---|---|
| 2 | public savePhone(uuid: string) {<br>  ***clientService***.savePhone(uuid, this.clientPhones)<br>  .then(() => {<br>  if ((this.clientPhones.length === 0 \|\| this.allSaved) && (this.clientAddresses.length === 0 \|\| this.allSaved) && (this.errors.length === 0)) {<br>  this.$router.push({name: 'client-list'});<br>  } else if (this.clientAddresses.length === 0) {<br>  this.saveButtonLoader = false;<br>  this.$router.push({name: 'client-list'});<br>  }<br>  })<br>  .catch((error) => {<br>  this.errors.push(error.response.data.message);<br>  this.$store.commit('app/showMessage', ['error', error.response.data.message]);<br>  this.saveButtonLoader = false;<br>  this.$store.commit('loader', false);<br>  ***clientService***.deleteClient(uuid);<br>  });<br>} | Метод *savePhone*, который отправляет POST запрос с массивом номеров телефонов для определенного клиента. После процесса добавления самого клиента за ним идет процесс закрепления номеров телефонов. |
| 3 | public saveAddress(uuid: string) {<br>  ***clientService***.saveAddress(uuid, this.clientAddresses)<br>  .then(() => {<br>  if ((this.clientPhones.length === 0) && (this.errors.length === 0)) {<br>  (this.$refs.form as Vue & { reset: () => boolean }).reset();<br>  this.$router.push({name: 'client-list'});<br>  }<br>  })<br>  .catch((error) => {<br>  this.errors.push(error.response.data.message);<br>  });<br>} | Метод *saveAddress*, который отправляет POST запрос с массивом адресов для определенного клиента. После процесса добавления самого клиента за ним идет процесс закрепления адреса. |

### 3.5 Тестирование и отладка программного обеспечения

Для входа в информационную систему необходимо авторизоваться. Авторизоваться можно по номеру телефона или по e-mail. Страница авторизации показана на рисунке 3.9.

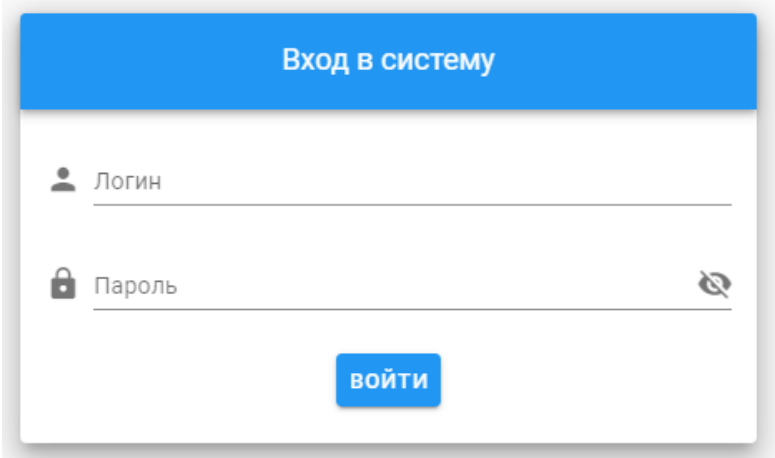



Рисунок 3.9 - Форма авторизации для входа в систему

После успешного входа в систему пользователь получает оповещение. Компонент оповещения показан на рисунке 3.10.

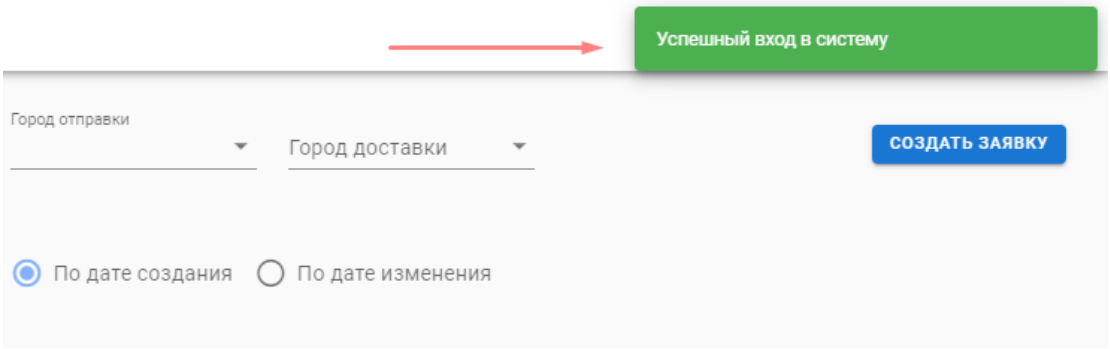



Рисунок 3.10 - Компонент Snackbar для оповещения пользователя

Немаловажную роль в системе играет панель навигации. В разделе описания пользовательского интерфейса она представляла из себя компонент Drawer. Панель навигации показан на рисунке 3.11.

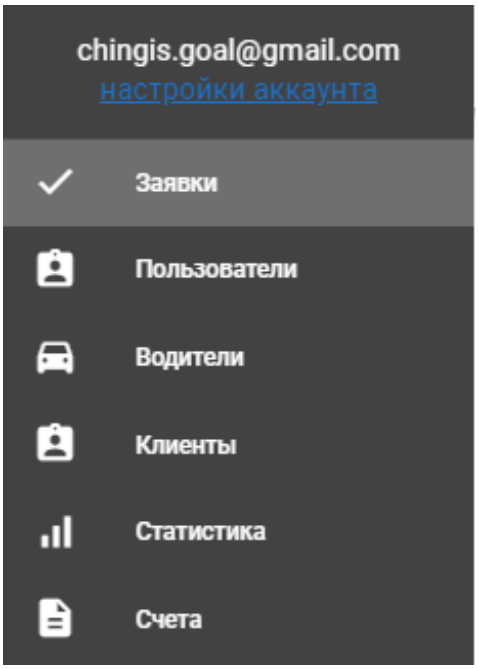



Рисунок 3.11 - Панель навигации

Страница «Заявки» показана на рисунке 3.12. Сверху страницы расположен фильтр и элементы сортировки. Есть возможность осуществить поиск по номеру заявки, статусу заявки, городу отправки, городу доставки. А также можно отсортировать список заявок заполнив поля (начальная дата, конечная дата). Чуть ниже расположены кнопки: «Сменить статус», «Загрузить отчет», «Обновить». С помощью кнопки «Сменить статус» можно сменить статус заявок на один из следующих статусов: Сформирован, Принят на склад отправления Asia Freight, Передан в регион, Доставлено. Кнопка «Загрузить отчет» скачивает отчет о заявках в формате Excel. Кнопка «Обновить» обновляет страницу, то есть после изменения любых характеристик фильтра или сортировки пользователь нажимает данную кнопку, отправляется запрос уже по установленным характеристикам и сервер возвращает ответ, который отображается в таблице (рис.3.13).

Все заявки разделены логически на две вкладки: заявки на забор и довоз. Вкладка забор показана на рисунке 6. Вкладка довоз показана на рисунке 3.14.

В списке заявок можно заметить, что нумерация заявок сделана в виде ссылок. Если кликнуть по ссылке заявки, откроется модальное окно, которое занимает 50% высоты от всего окна (рис. 3.15). Данное модальное окно показывает информацию о заявке. Можно оставить комментарий к заявке, этот процесс необходим в том случае, когда груз не был доставлен по тем или причинам. Например, получатель не оказался на месте в момент получения груза. Вы можете также заметить, что заявки, у которых есть комментарий в таблице визуально отличаются от заявок, у которых отсутствует комментарий. А именно тем, что заявки с комментариями помечены знаком «*» (рис.3.16).

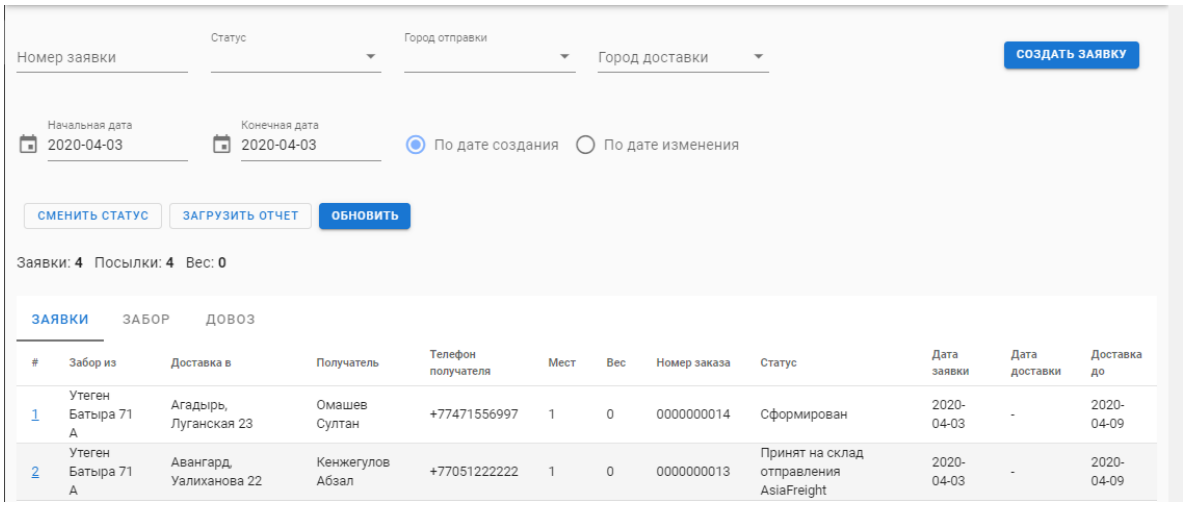



Рисунок 3.12 - Страница «Заявки»

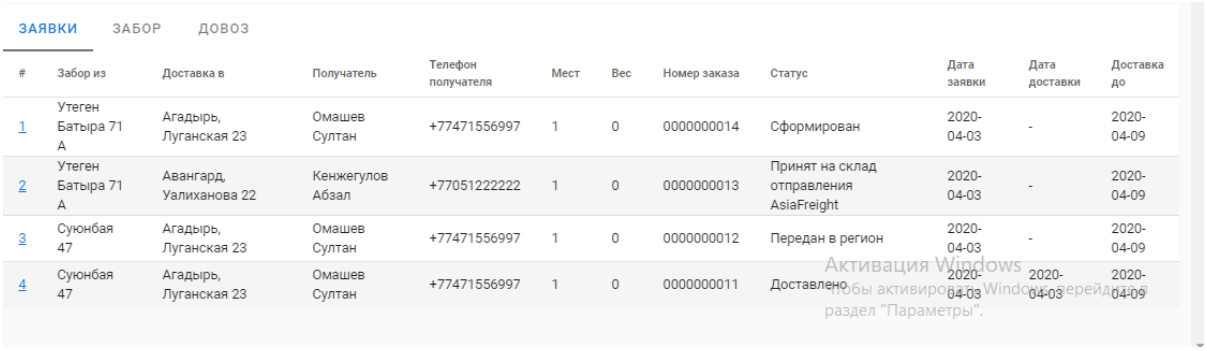



Рисунок 3.13 - Таблица всех заявок

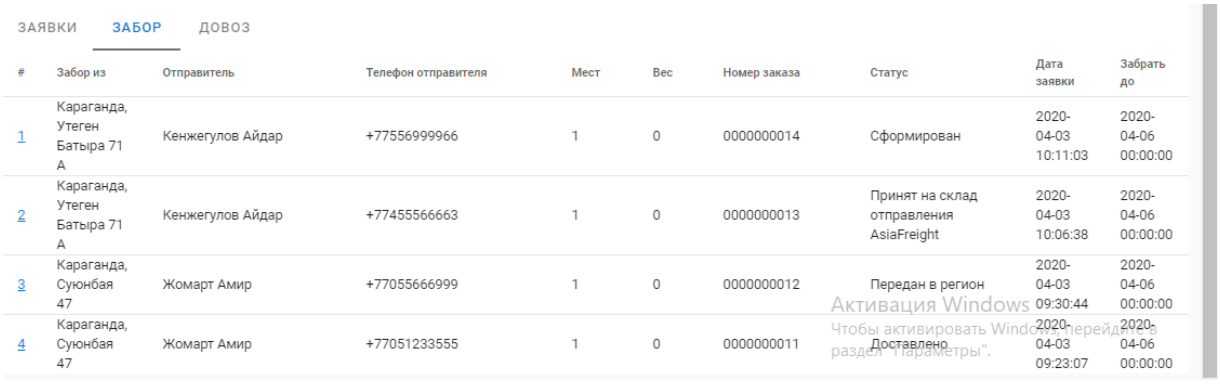



Рисунок 3.14 - Таблица заявок на забор

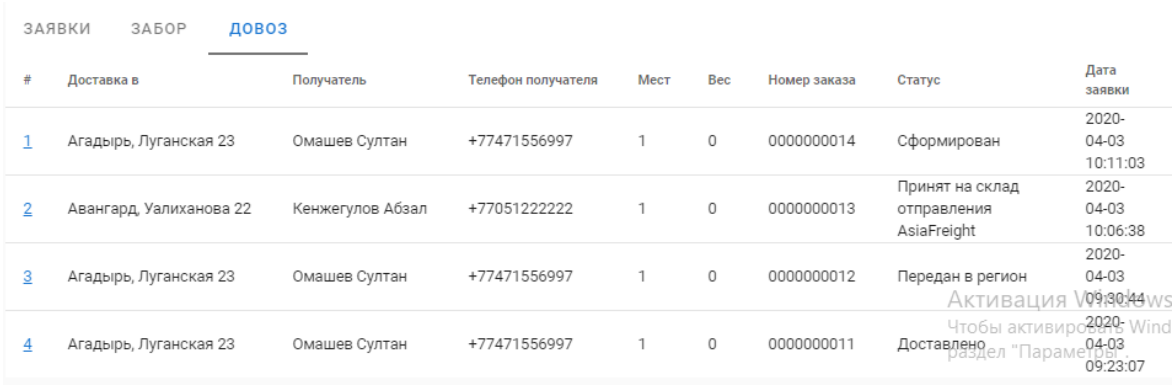



Рисунок 3.15 - Таблица заявок на довоз

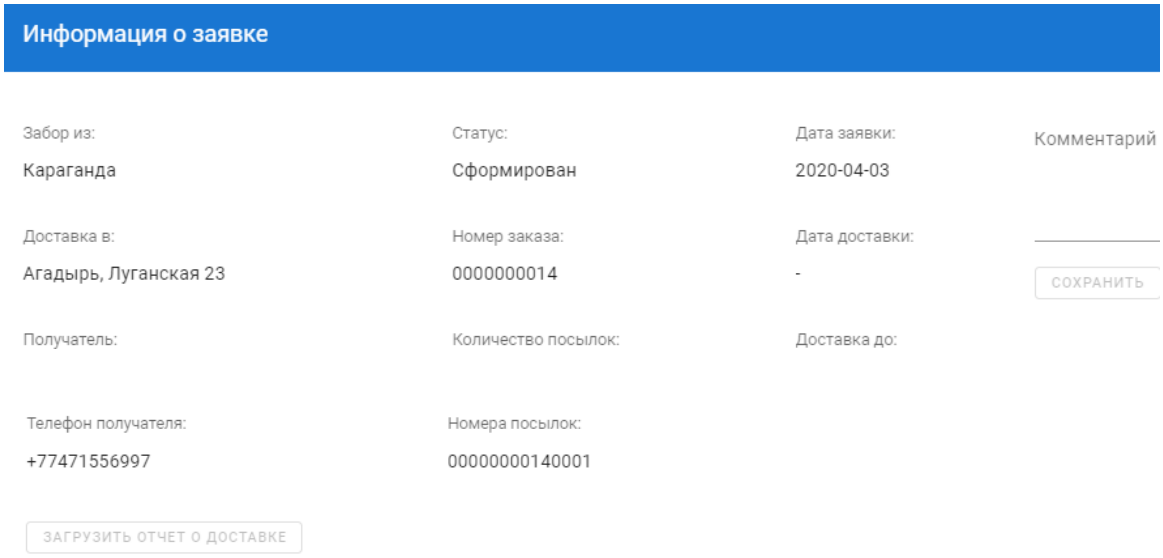



Рисунок 3.16 - Окно информации о заявке

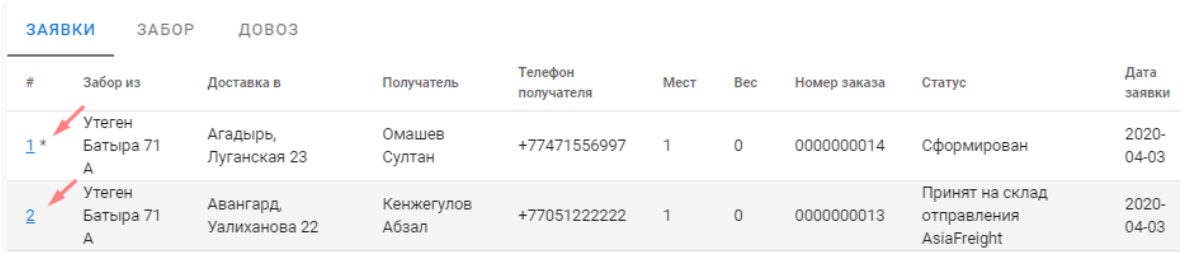



Рисунок 3.17 - Ссылка для просмотра информации о заявке

Страница «Новая заявка» показана на рисунке 3.18. Чтобы добавить новую заявку, для начала необходимо тип посылки. В случае отсутствия типа посылки нужно кликнуть на значок «+», в открывшемся модальном окне добавить новый тип посылки (рис. 3.19). После выбора типа посылки нужно выбрать тип услуги (забор или довоз груза). На рисунке 3.20 показано, что выбран тип забор. Чтобы заполнить данные клиента, необходимо ввести

первые три буквы клиента, а остальные данные, то есть номер телефона и адрес заполняются автоматически благодаря директиве v-model во VueJS.

Все данные для создания заявки заполнены, но может быть такое что некоторые данные (номер телефона, адрес) клиента устарели. Для таких случаев предусмотрена возможность добавление этих данных в процессе создания заявки на забор для данного клиента (рис. 3.21). Для окончательного завершения процесса создания заявки нужно нажать на кнопку «Добавить», пользователь получает оповещения об успешном завершении процесса и происходит перенаправления на страницу «Заявки».

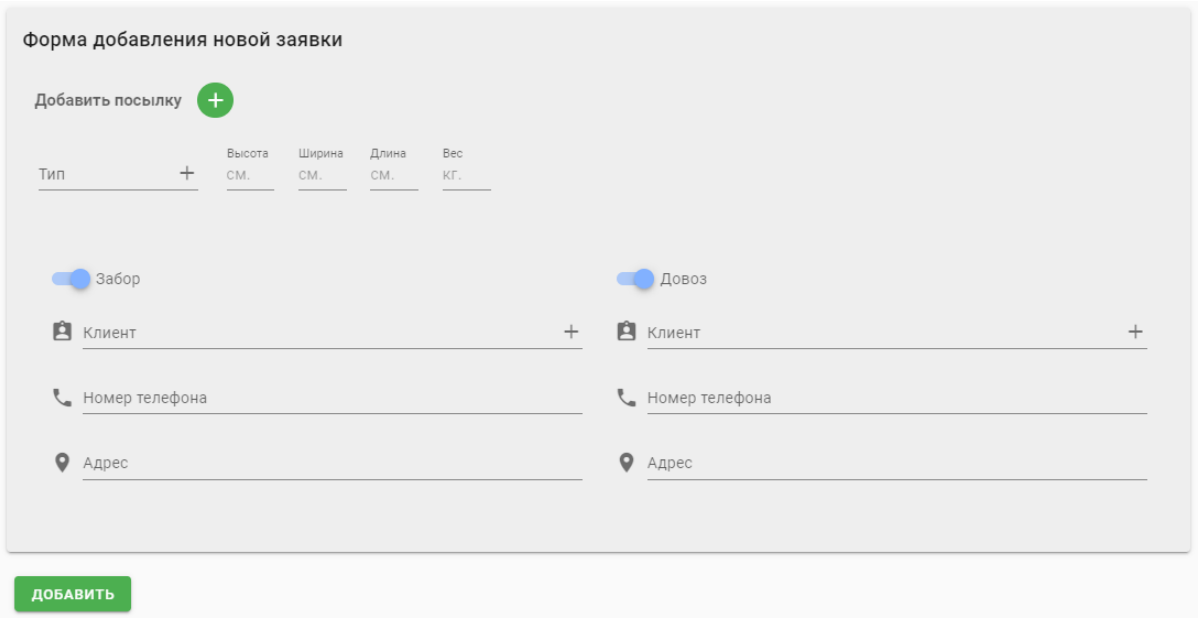



Рисунок 3.18 - Форма добавления новой заявки

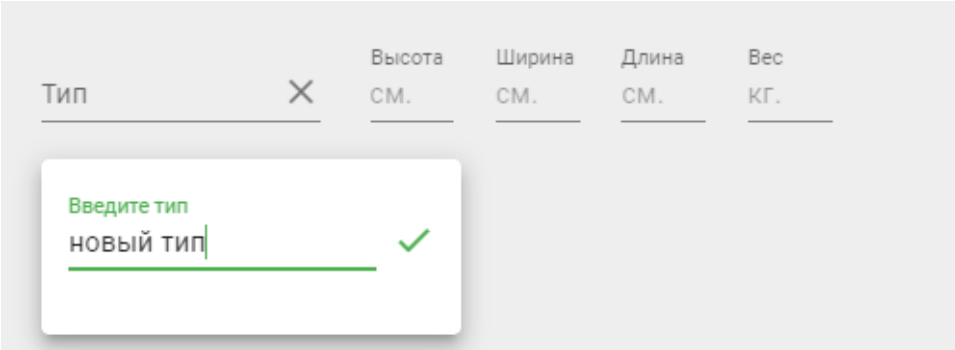



Рисунок 3.19 - Форма добавления нового типа

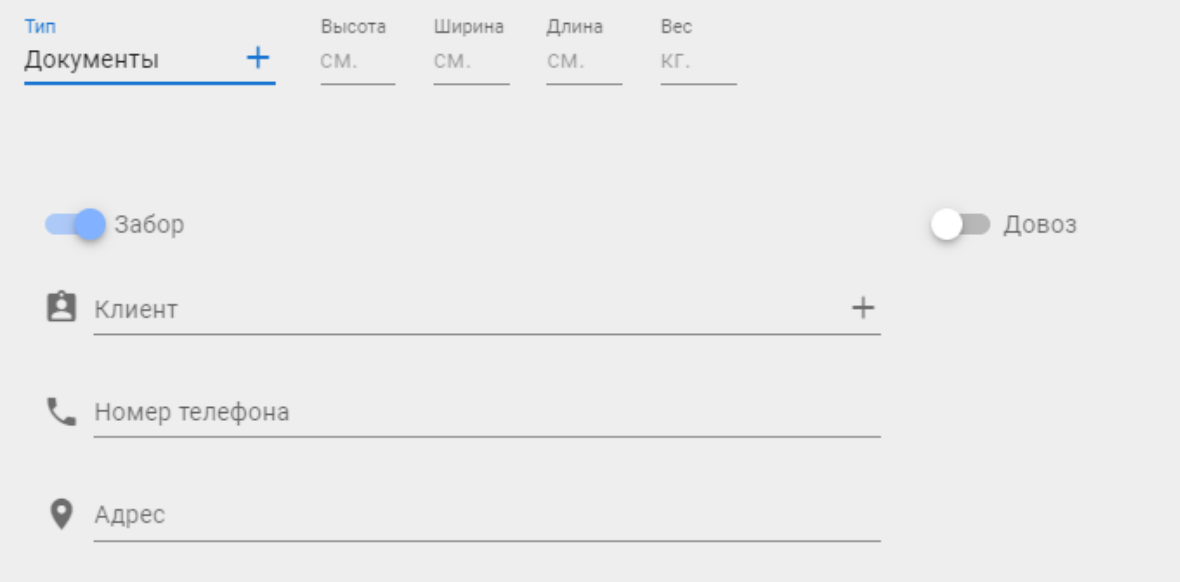



Рисунок 3.20 - Добавлен новый тип «Документы»

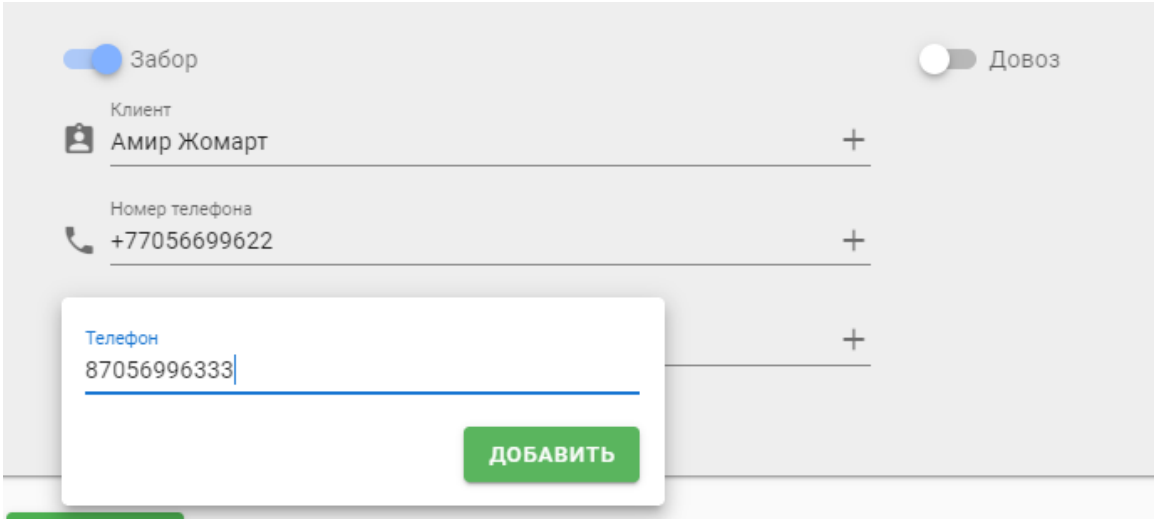



Рисунок 3.21 - Форма добавления номера телефона для клиента

Страница «Пользователи» показана на рисунке 3.22. Сверху страницы расположены две кнопки: «Добавить пользователя», «Обновить». Если кликнуть на кнопку «Добавить пользователя», то откроется модальное окно, корректно заполнив которое можно добавить пользователя данной информационной системы (рис. 3.23).

В таблице пользователей email пользователя имеет вид ссылки, кликнув по нему можно открыть модальное окно, которое показывает информацию о пользователе (рис. 3.24). А также в данном окне есть возможность изменять данные пользователя, активировать, деактивировать. К сведению неактивные пользователи не имеют возможности войти в систему. Процесс активации показан на рисунке 3.25.

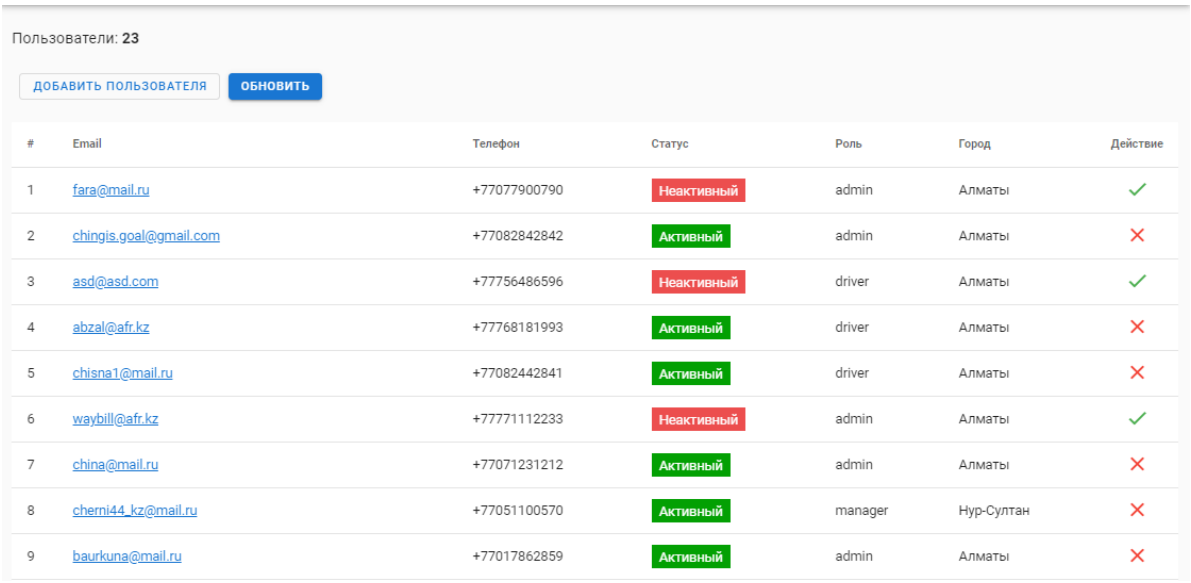



Рисунок 3.22 - Страница «Пользователи»

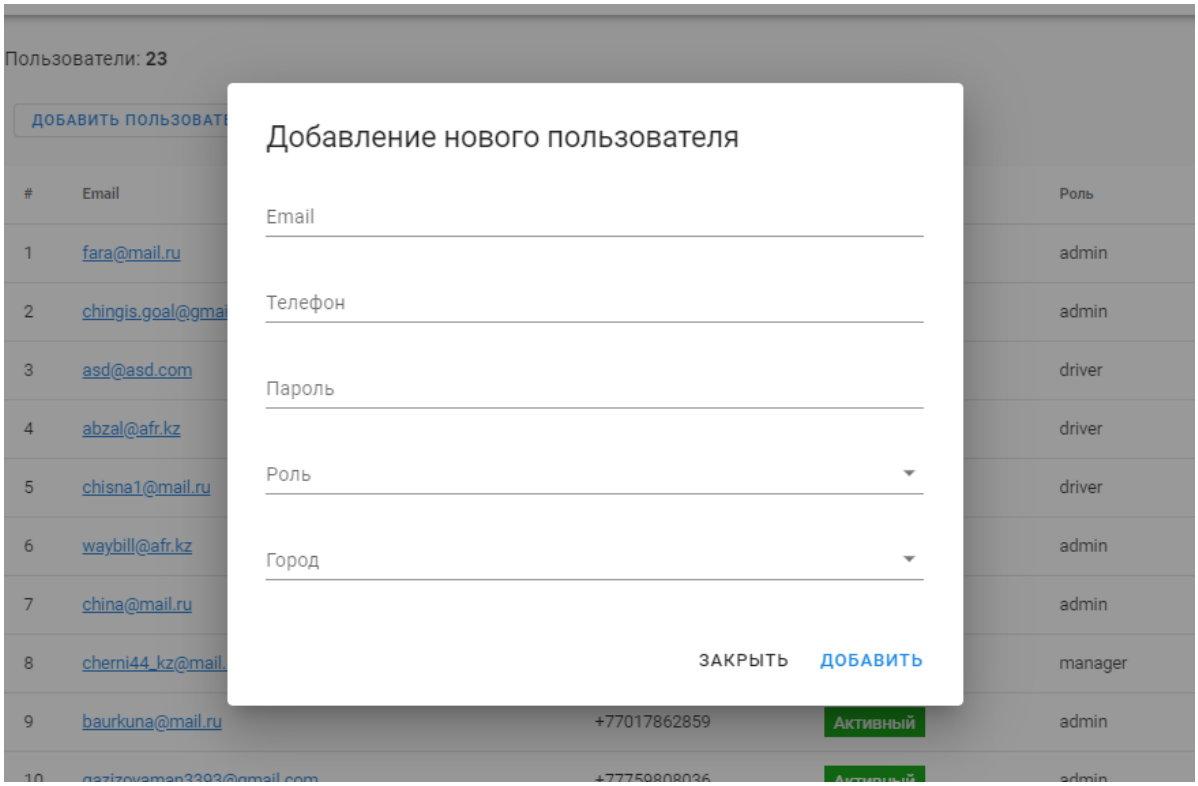



Рисунок 3.23 - Форма добавления нового пользователя

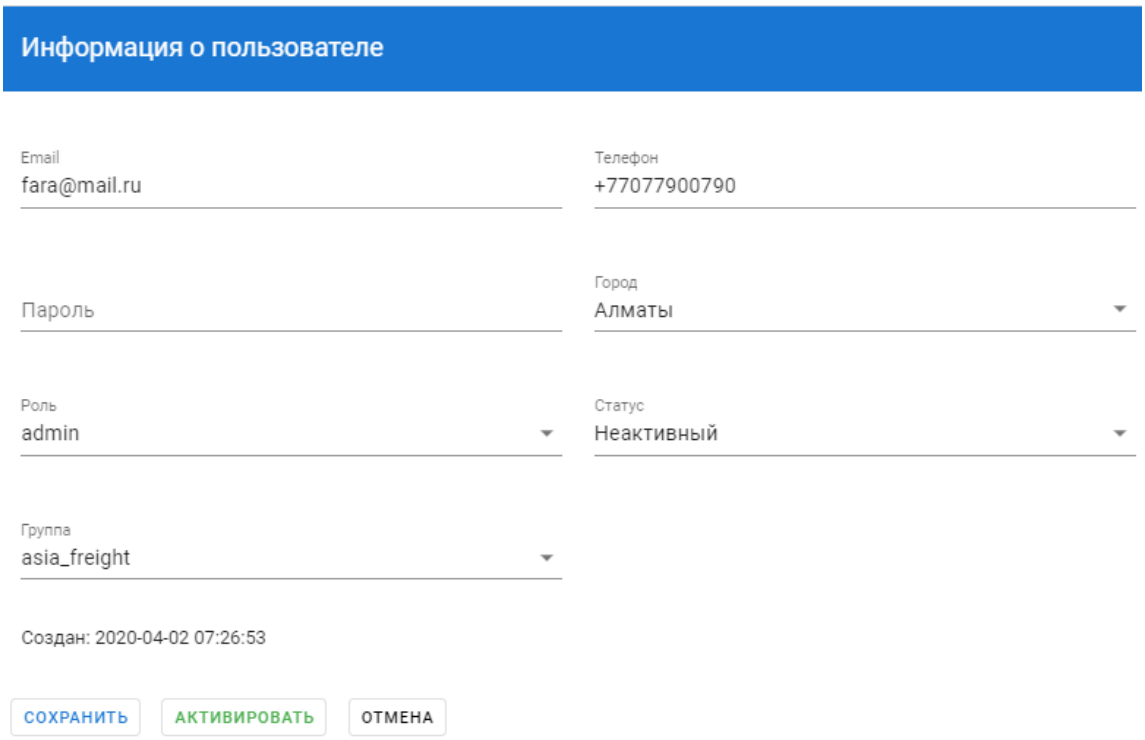



Рисунок 3.24 - Окно информации о пользователе

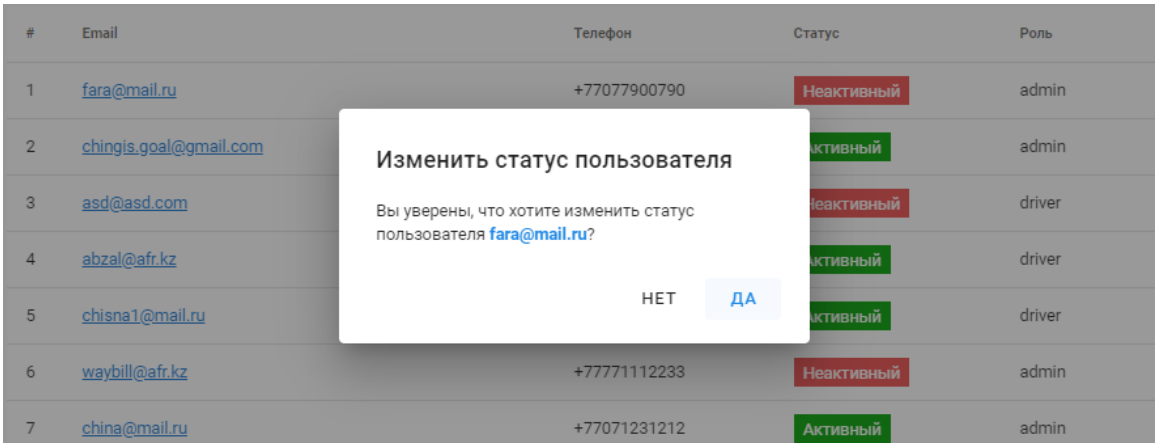



Рисунок 3.25 - Изменение статуса пользователя

Страница «Водители» визуально очень похожа на страницу «Пользователи» и показана на рисунке 3.26. Сверху страницы расположены две кнопки: «Добавить водителя», «Обновить». Если кликнуть на кнопку «Добавить водителя», то откроется модальное окно, корректно заполнив которое можно добавить водителя, у которых есть право выбрать заявку для дальнейшей реализации услуги (рис. 3.27).

В таблице водителей поле «Водитель» имеет вид ссылки, кликнув по нему можно открыть модальное окно, которое показывает информацию о водителе (рис. 3.28). А также в данном окне есть возможность изменять данные водителя, активировать, деактивировать. К сведению неактивные

водителей не имеют возможности войти в систему. Процесс активации показан на рисунке 3.29.

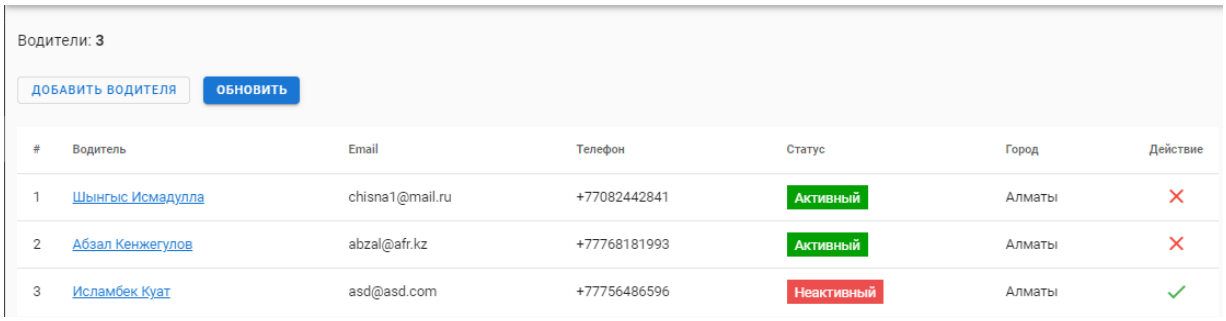



Рисунок 3.26 - Страница «Водители»

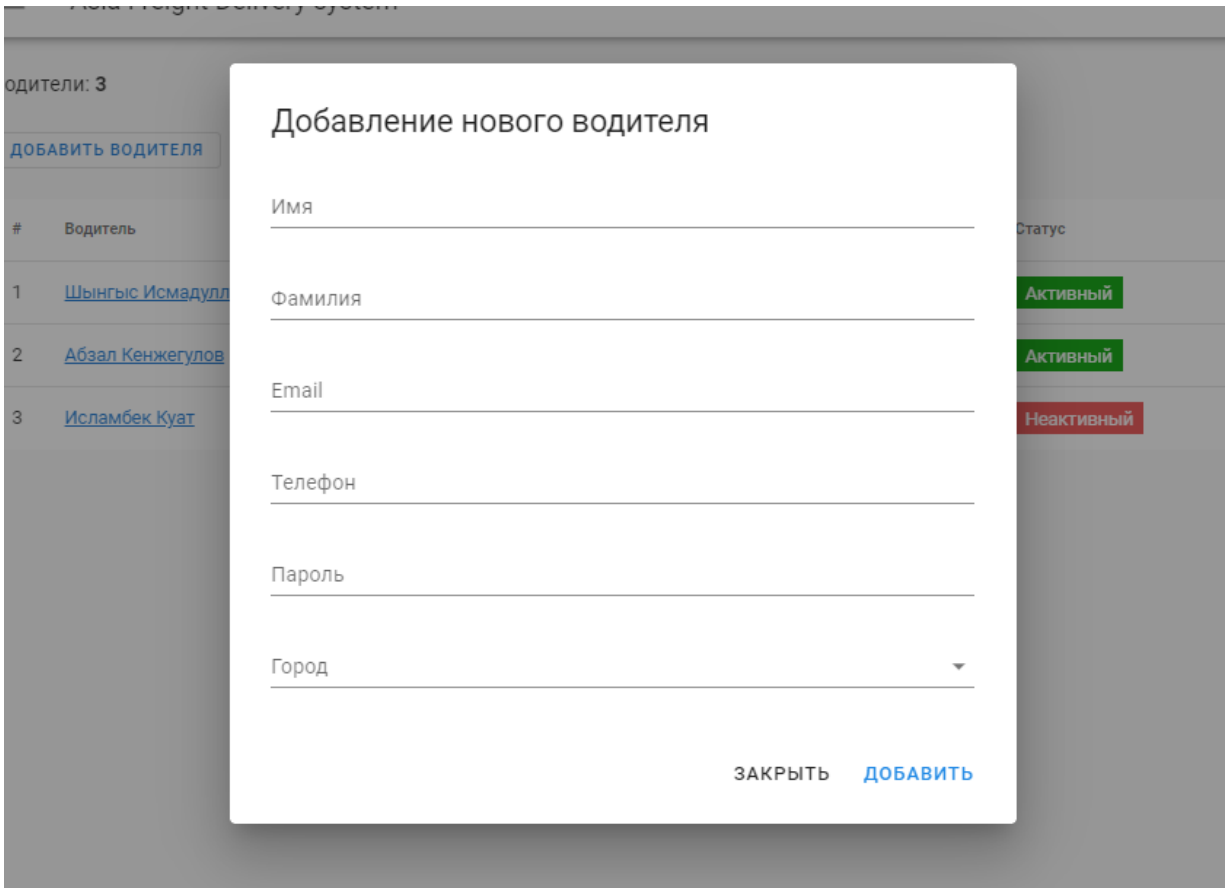



Рисунок 3.27 - Форма добавления нового водителя

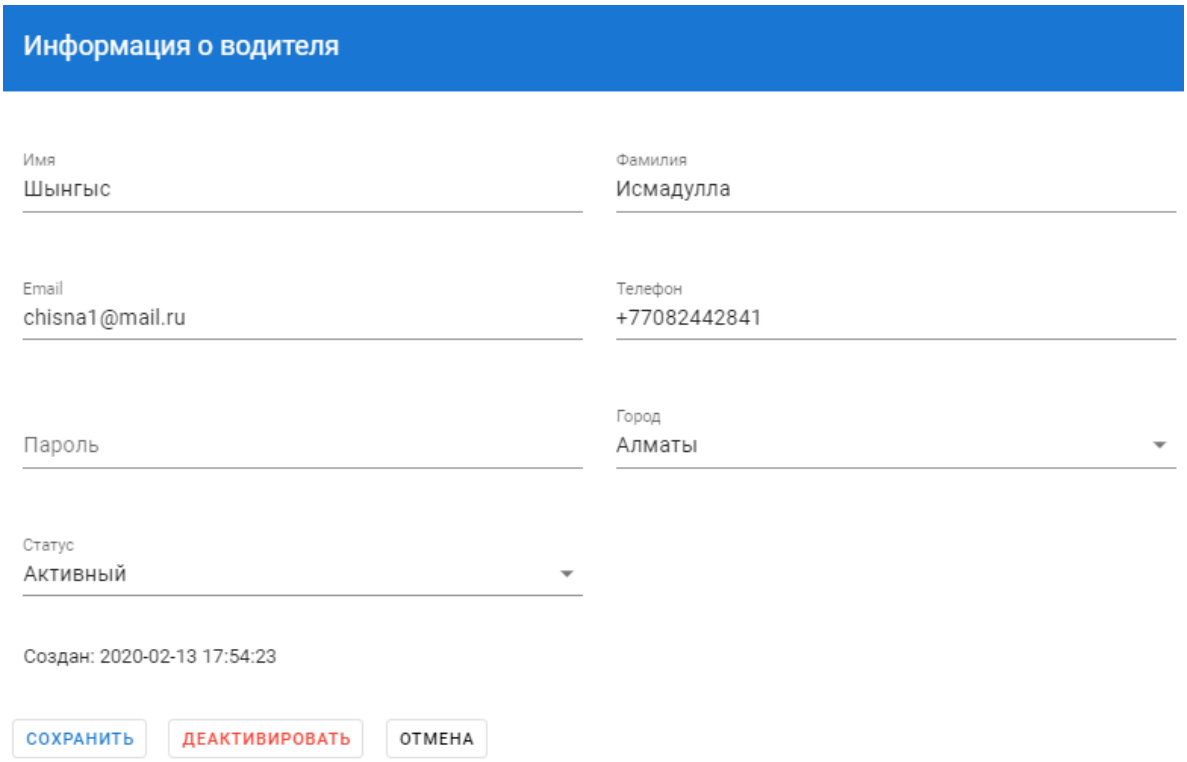



Рисунок 3.28 - Окно информации о водителе

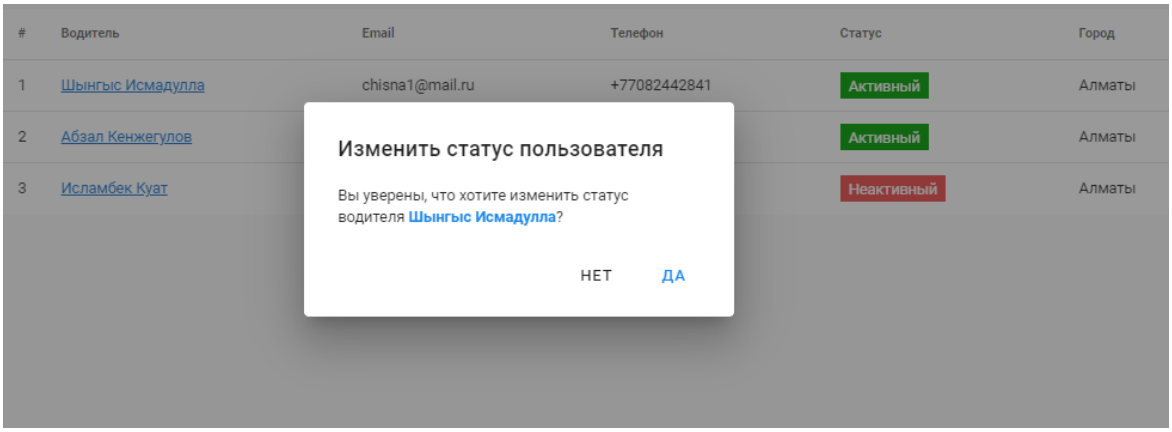


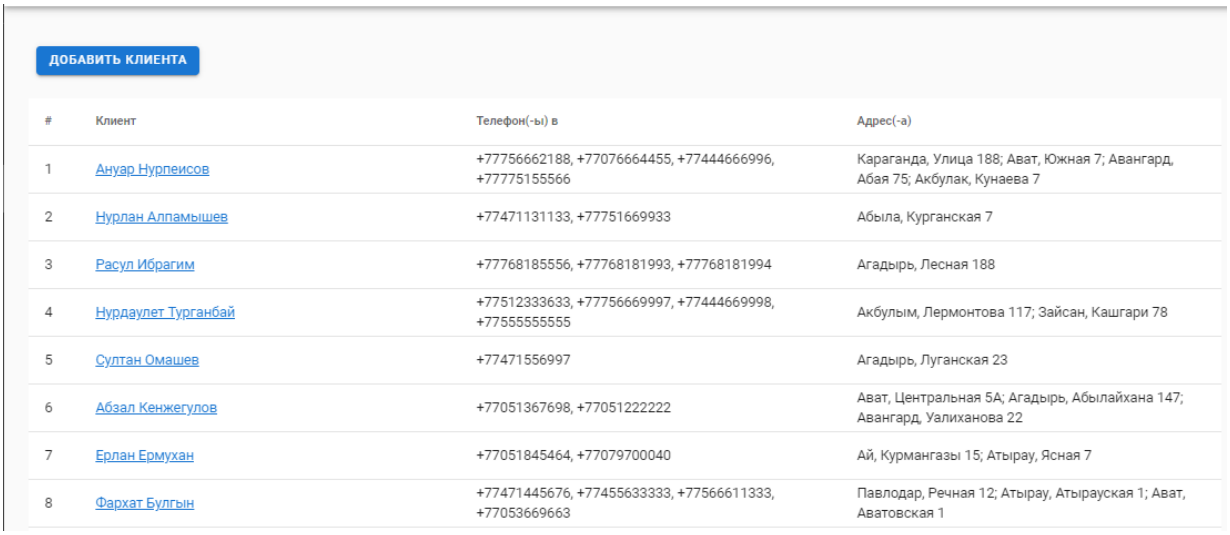

Рисунок 3.29 - Изменение статуса водителя

Страница «Клиенты» показана на рисунке 3.30. На странице расположена кнопка «Добавить клиента» и таблица клиентов. В таблице находятся все основные сведения клиентов:

- фамилия и имя;
- номер телефона;
- адрес (город, улица и номер дома).

Поле «Клиент» имеет вид ссылки, перейдя по ней можно отредактировать сведения определенного клиента (рис. 3.31).

Если кликнуть на кнопку «Добавить клиента», то произойдет перенаправления на страницу добавления клиента, где есть форма для создания нового клиента (рис. 3.32).




| # | Клиент | Телефон(-ы) в | Адрес(-а) |
|---|---|---|---|
| 1 | Ануар Нурпеисов | +77756662188, +77076664455, +77444666996, +77775155566 | Караганда, Улица 188; Ават, Южная 7; Авангард, Абая 75; Акбулак, Кунаева 7 |
| 2 | Нурлан Алпамышев | +77471131133, +77751669933 | Абыла, Курганская 7 |
| 3 | Расул Ибрагим | +77768185556, +77768181993, +77768181994 | Агадырь, Лесная 188 |
| 4 | Нурдаулет Турганбай | +77512333633, +77756669997, +77444669998, +77555555555 | Акбулым, Лермонтова 117; Зайсан, Кашгари 78 |
| 5 | Султан Омашев | +77471556997 | Агадырь, Луганская 23 |
| 6 | Абзал Кенжегулов | +77051367698, +77051222222 | Ават, Центральная 5А; Агадырь, Абылайхана 147; Авангард, Уалиханова 22 |
| 7 | Ерлан Ермухан | +77051845464, +77079700040 | Ай, Курмангазы 15; Атырау, Ясная 7 |
| 8 | Фархат Булгын | +77471445676, +77455633333, +77566611333, +77053669663 | Павлодар, Речная 12; Атырау, Атырауская 1; Ават, Аватовская 1 |

Рисунок 3.30 - Страница «Клиенты»

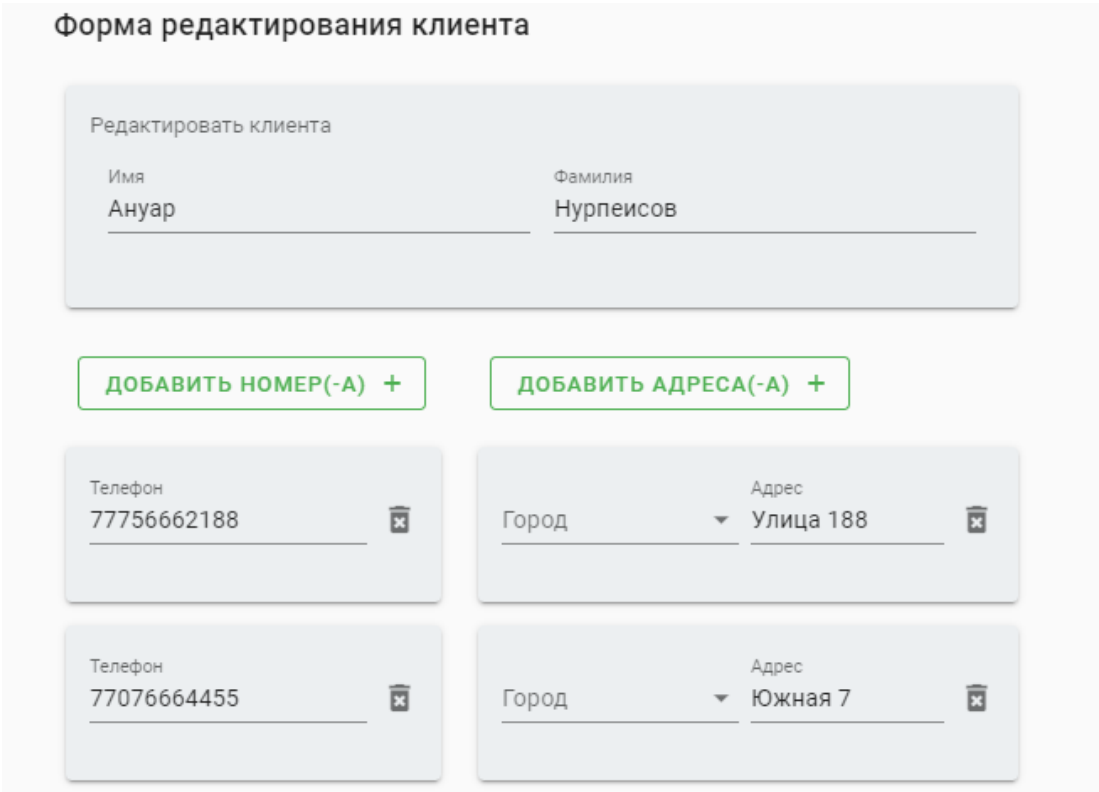



Рисунок 3.31 - Форма редактирования клиента

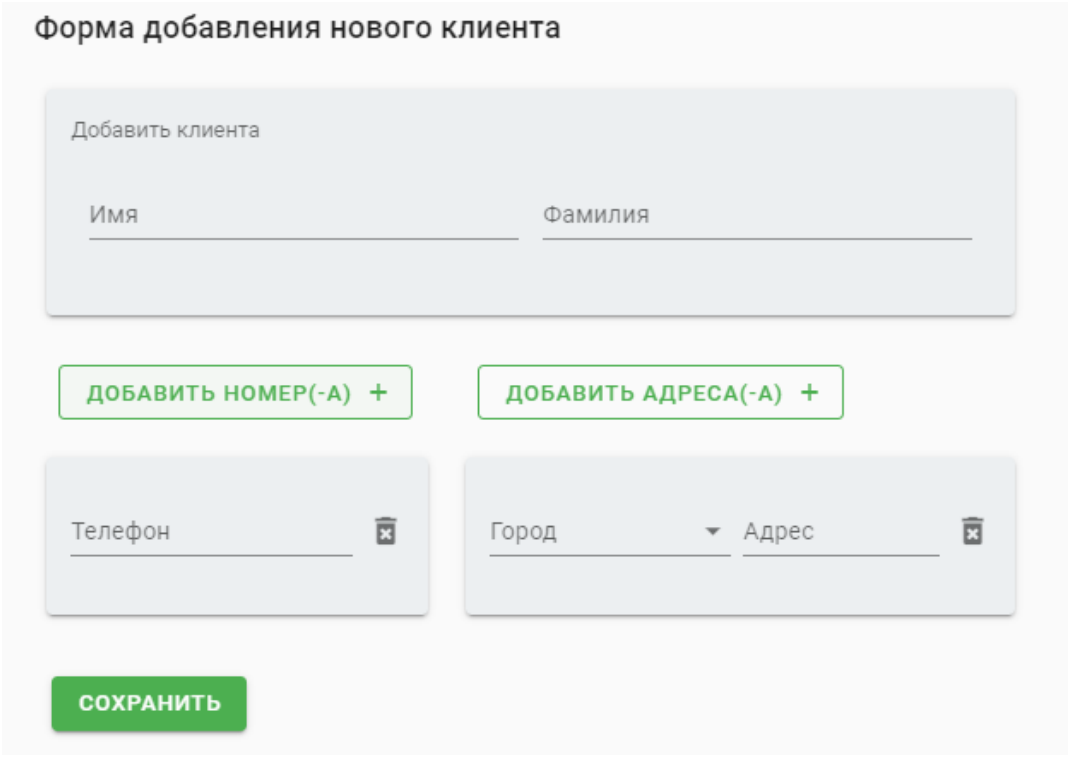



Рисунок 3.32 - Форма добавления клиента

Страница «Статистика» состоит из четырех видов графиков для вывода статистики по заявкам. В первом графике выводятся статистика по дням, который показан на рисунке 3.33. Во втором графике статистика за месяц, который показан на рисунке 3.34. В третьем графике за год, который показан на рисунке 3.35. Чуть ниже данных графиков расположен компонент Pie, в котором показывается в виде «Пирога» за определенный промежуток времени, который указывает менеджер (рис.3.36).

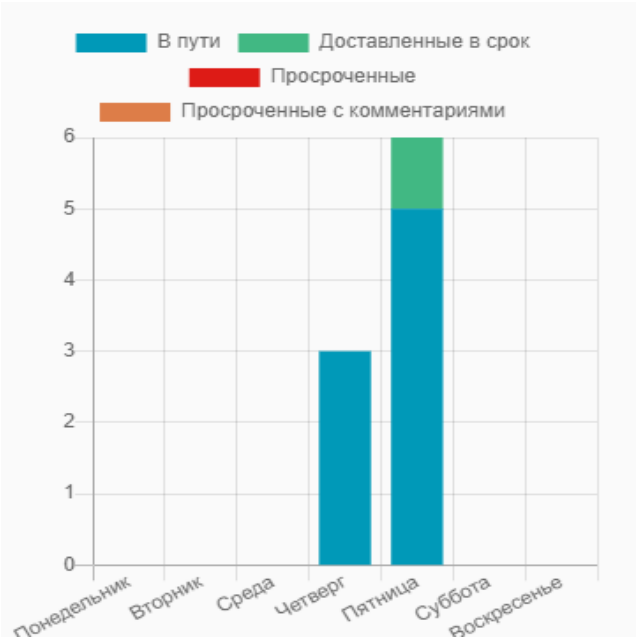


Рисунок 3.33 - График статистики по дням

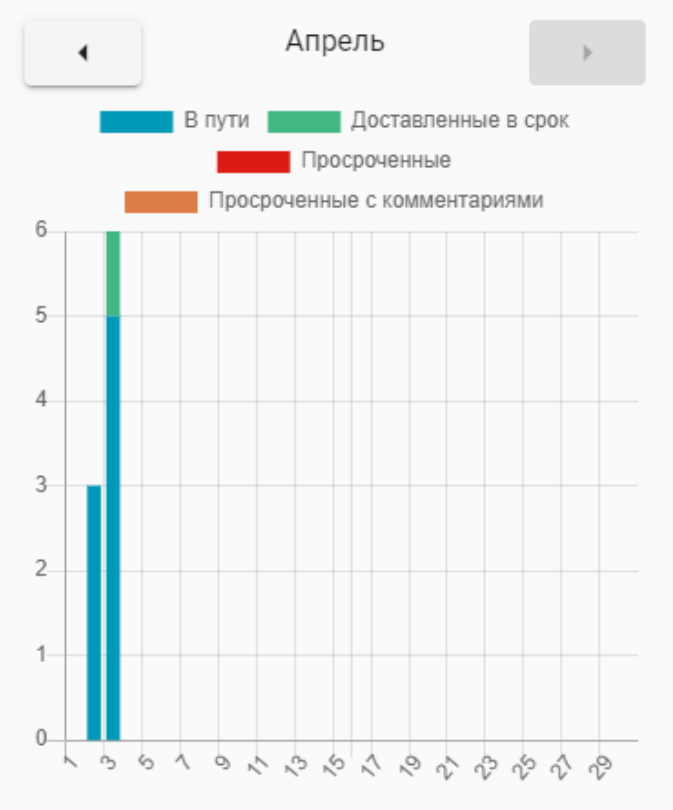



Рисунок 3.34 - График статистики по месяцам

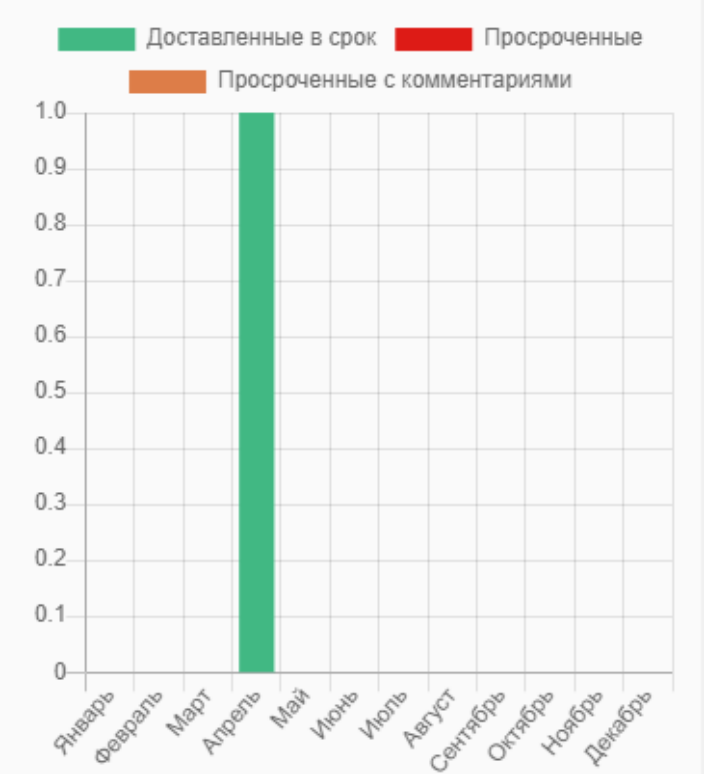



Рисунок 3.35 - График статистики за год

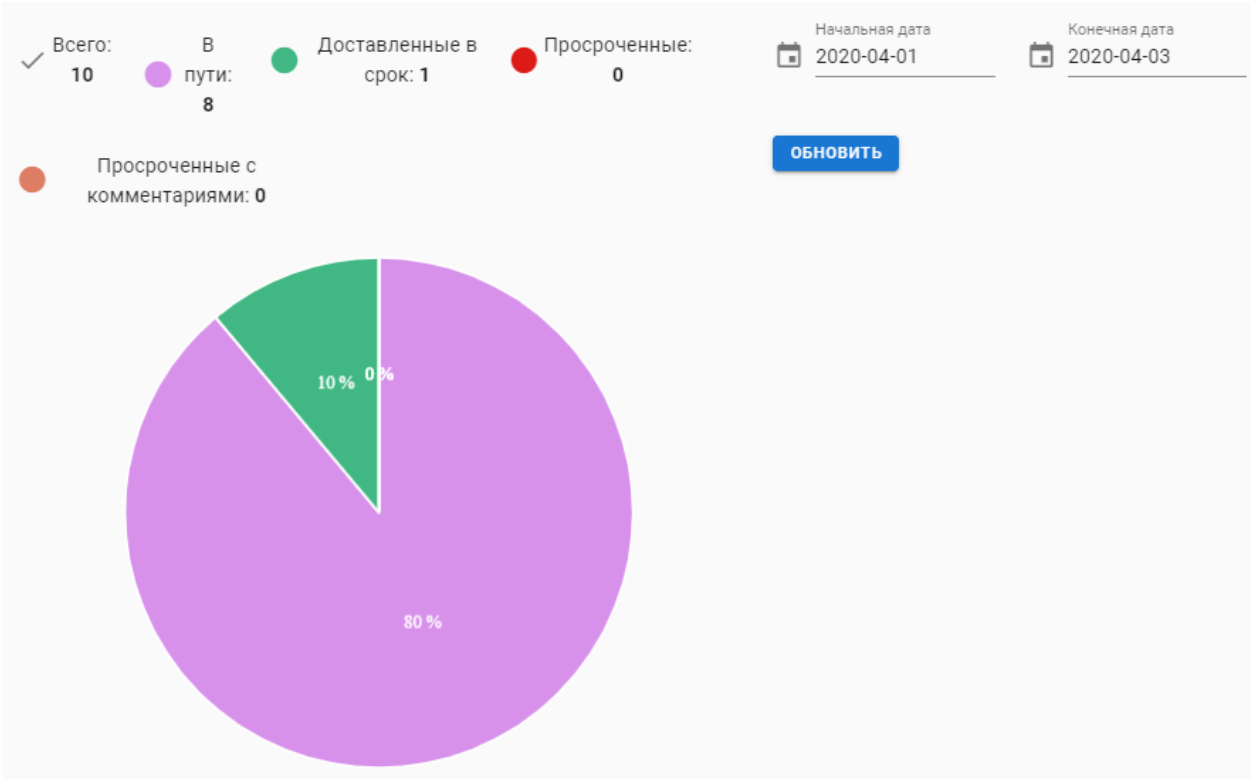



Рисунок 3.36 - Компонент «Pie»

## 4 Экономическое обоснование проекта

Целью дипломного проектирования является разработка информационной системы логистической компании ТОО «Asia Freight», основным функционалом которого является представление информации о заявках, которые создаются менеджером в системе. Возможность добавление пользователей, групп и ролей для управления данной системой. Кроме того, в информационной системе будет доступна возможность добавление водителей, а также возможность закрепления их заявкам. Водителям доступна принятие заявки на забор или довоз груза по городу. Имеется информация о счетах и тарификации, также стоимость каждой заявки. Основная цель программного продукта снизить издержки и увеличить количество заказчиков. Основной задачей экономического раздела является расчет трудоемкости, себестоимости проекта (конкретно разрабатываемых подсистем) и определение экономической эффективности.

### 4. 1 Расчет трудоемкости разработки информационной системы для логистической компании ТОО «Asia Freight»

Базовый показатель для определения составляющих затрат труда вычисляется по формуле:

$$Q = q \times c \qquad (4.1)$$

где $Q$ – число строк;

$q$ – коэффициент, учитывающее условное число команд в зависимости от типа задачи;

с – коэффициент, учитывающий новизну и сложность программы.

Для того чтобы решить базовый показатель необходимо, выбрать коэффициенты $c$ и $q$ в таблице 4.1 и 4.2.

Таблица 4.1 - Значения коэффициента q

| Тип задачи | Пределы изменений коэффициента |
|---|---|
| Задачи учета | от 1400 до 1500 |
| Задачи оперативного управления | от 1500 до 1700 |
| Задача планирования | от 3000 до 3500 |
| Многовариантные задачи | от 4500 до 5000 |
| Комплексные задачи | от 5000 до 5500 |

Таблица 4.2 - Коэффициенты расчета трудоемкости

| Язык программирования | Группа сложности | Степень новизны | | | | Коэффициент B |
|---|---|---|---|---|---|---|
| | | А | Б | В | Г | |
| Высокого уровня | 1 | 1,38 | 1,26 | 1,15 | 0,69 | 1,2 |
| | 2 | 1,30 | 1,19 | 1,08 | 0,65 | 1,35 |
| | 3 | 1,20 | 1,10 | 1,00 | 0,60 | 1,5 |

| Язык программирования | Группа сложности | Степень новизны | | | | Коэффициент В |
|---|---|---|---|---|---|---|
| | | А | Б | В | Г | |
| Низкого уровня | 1 | 1,58 | 1,45 | 1,32 | 0,79 | 1,2 |
| | 2 | 1,49 | 1,37 | 1,24 | 0,74 | 1,35 |
| | 3 | 1,38 | 1,26 | 1,15 | 0,69 | 1,5 |

Из перечисленных выше значений из таблицы выбираем q = 4500, с =1,19 и В = 1,35.

Подставив коэффициенты в уравнение (4.1), рассчитаем число команд:

$Q = 4500 * 1{,}19 = 5355$ (строк кода).

Трудоемкость стадий определяется с учетом сложности, новизны, степени использования в разработке стандартных модулей ПО и удельного веса трудоемкости каждой стадии в общей трудоёмкости ПО [11]:

$$T_{yi} = T_{\text{н}} * d_{cmi} * K_c * K_m * K_{\text{н}},$$

(4.2)

где $T_{yi}$ – уточненная трудоемкость разработки ПО на $i-$ й стадии;

$T_{\text{н}}$ – нормативная трудоёмкость;

$d_{cmi}$ – удельный вес трудоемкости $i-$ й стадии разработки ПО в общей трудоемкости разработки ПО;

$K_c$ – коэффициент, учитывающий сложность ПО, вводится на всех стадиях;

$K_m$ – коэффициент, учитывающий степень использования стандартных модулей ПО, вводится только на стадии рабочего проекта;

$K_{\text{н}}$ – коэффициент, учитывающий степень новизны ПО, вводится на всех стадиях.

Нормативная трудоёмкость $T_{\text{н}}$ определяем по таблице Б.1 приложения Б [11]:

$T_{\text{н}}$ = 182 часа.

Коэффициент сложности рассчитывается по формуле [11]

$$K_c = 1 + \sum_{i=0}^{n} K_i, \quad (4.3)$$

где $K_i$ – коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;

$n$ – количество характеристик.

$$K_c = 1 + \frac{0{,}06}{1} = 1{,}06.$$

Коэффициент использования сторонних модулей берется из таблицы 2 [11] и составляет $K_m = 0{,}9$ используется только 20% стороннего ПО.

Коэффициент новизны, определяется по таблице 3 [11] и равен $K_н = 1{,}0$, так как программный продукт не имеет аналогов для разрабатываемой компании. Значения коэффициентов удельных весов трудоемкости стадий берутся из таблицы 4 [11], согласно степени новизны.

Таблица 4.3 - Стадии разработки программного продукта

| № | Название | $d_{cmi}$ – удельный вес трудоемкости |
|---|---|---|
| 1 | Техническое задание | 0,11 |
| 2 | Техническое проектирование (без эскизного проектирования) | 0,2 |
| 3 | Рабочее проектирование (программирование) | 0,55 |
| 4 | Ввод в эксплуатация | 0,14 |

Трудоемкость стадий ПО рассчитывается по следующим формулам [11]

$T_{уз} = T_н * d_з * K_c * K_н = 182 * 1{,}06 * 0{,}11 * 1{,}0 = 21{,}22$ часов;
$T_{ут} = T_н * d_m * K_c * K_н = 182 * 1{,}06 * 0{,}2 * 1{,}0 = 38{,}58$ часов;
$T_{ур} = T_н * d_р * K_c * K_н * K_m = 182 * 1{,}06 * 0{,}55 * 1{,}0 * 0{,}9 = 95{,}5$ часов;
$T_{ув} = T_н * d_в * K_c * K_н = 182 * 1{,}06 * 0{,}14 * 1{,}0 = 27$ часов.

Общая трудоемкость будет равна

$T_y = 21{,}22 + 38{,}58 + 95{,}5 + 27 = 182{,}3$ часа $\approx 8$ дней.

### 4. 2 Расчет затрат на разработку информационных технологий

Расчет полных затрат на разработку проектного решения осуществляется по формуле [11]:

$$С_{ni} = ЗП + СН + СО + ВОСМС + М + ПО + Э + А + НК + П + Н, \quad (4.4)$$

где ЗП – общий фонд оплаты труда разработчиков, тенге;

СН – отчисления по социальному налогу, тенге;

$СО$ – социальное отчисление, тенге;

ВОСМС – обязательное медицинское страхование, тенге;

М – затраты на материалы, тенге;

ПО – затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;

Э – затраты на электричество, тенге;

А – амортизационные отчисления, тенге;

НК – затраты на научные командировки, тенге;

П – прочие затраты, тенге;

Н – накладные расходы, тенге.

Размер фонда оплаты труда разработчиков рассчитывается по формуле [11]:

$$ЗП = Зп\mathit{О} + ЗпД, \quad (4.5)$$

где $Зп\mathit{О}$ – основная заработная плата, тенге;

ЗпД – дополнительная заработная плата, тенге.

Основная заработная плата рассчитывается по следующей формуле [11]

$$Зп\mathit{О} = T_y * СП/(t_{ср} * 8), \quad (4.6)$$

где $T_y$– суммарные затраты труда;

$t_{ср}$ – среднее число дней в месяце, равно 22 дню, умножается на количество часов в рабочем дне – 8;

СП – средняя заработная плата.

Средняя месячная зарплата в РК на 01.01.2020 г. составляет 203900 тенге, тогда основная заработная плата составит:

$$Зп\mathit{О} = 182{,}3 * \frac{203\,900}{22 * 8} = 212\,000 \text{ тенге.}$$

Дополнительная заработная плата составляет 10% от основной и рассчитывается по формуле [11]:

$$ЗпД = Зп\mathit{О} * 10\%, \quad (4.7)$$

Тогда,

$$ЗпД = 212\,000 * 10\% = 21\,200 \text{ тенге;}$$

$$ЗП = 212\,000 + 21\,200 = 233\,200 \text{ тенге.}$$

Социальный налог составляет (9,5%) (ст. 358 п.1 НК РК) от дохода работника, и рассчитывается по формуле [11]:

$$\text{СН} = (\text{ЗП} - \text{П}O - \text{В}O\text{СМС}) * 9{,}5\% - \text{С}O, \quad (4.8)$$

где ПО – пенсионные отчисления, которые составляют 10% от ЗП и социальным налогом не облагаются:

$$\text{П}O = \text{ЗП} * 10\%. \quad (4.9)$$

Тогда,

$$\text{П}O = 233\,200 * 10\% = 23\,320 \text{ тенге};$$

$$\text{СН} = (233\,200 - 23\,320 - 4664) * 9{,}5\% - 7345 = 12150 \text{ тенге}.$$

Социальное отчисление составляет (3,5%) (ст. 358 п.1 НК РК) от дохода работника, и рассчитывается по формуле:

$$\text{С}O = (\text{ЗП} - \text{П}O) * 3{,}5\%, \quad (4.10)$$

Тогда,

$$\text{С}O = (233\,200 - 23\,320) * 3{,}5\% = 7\,345 \text{ тенге}.$$

ВОСМС составляет 2% от ЗП, тогда

$$\text{В}O\text{СМС} = 233\,200 * 2\% = 4\,664 \text{ тенге}.$$

Итого выплаты налогов составят:

$$\text{Нал}o\text{ги} = \text{С}O + \text{СН} + \text{В}O\text{СМС} = 12150 + 7345 + 4664 = 241159 \text{ тг},$$

$$\text{чт}o\ \text{со}\text{ставляет } 10{,}46\%.$$

Величина затрат на материалы определяется по формуле [11]:

$$\text{М} = \frac{\text{З}_{o\text{сн}} * \text{Н}_{\text{мз}}}{100}\%, \quad (4.11)$$

где $\text{Н}_{\text{мз}}$ – норма расхода материалов от основной заработной платы (5%).

Следовательно,

$$\text{М} = 212\,000 * \frac{5}{100} = 10\,600 \text{ тенге}.$$

Амортизационные отчисления производятся по установленным нормам амортизации, выражаются в процентах к балансовой стоимости оборудования и рассчитываются по формуле

$$A = \frac{\text{С}_{\text{обор}} \times \text{На} \times N}{100 \times 12 \times t}, \quad (4.12)$$

где $\text{Н}_\text{а}$– норма амортизации (25%);

$С_{обор}$ – первоначальная стоимость оборудования ($С_{обор}$ = 200 тыс. тенге стоимость ноутбука, на котором создаётся программный продукт);

N – время использования персонального компьютера;

t – количество рабочих дней в месяце (t = 22 дней).

$$N = 182{,}3/8 \approx 23 \text{ дня.}$$

Расчитывая амортизационные отчисления согласно формуле (4.12), получим результат:

$$А_{н} = 200\,000 * 25 * \frac{23}{100*12*22} = 4356 \text{ тенге (для ноутбука)}$$

$$А_{ПО} = 85\,400 * 33{,}3 * \frac{23}{100*12*22} = 2477 \text{ тенге (phpStorm PRO)}$$

$$A = 4356 + 2477 = 6833 \text{ тенге}$$

Затраты на электроэнергию вычисляется по формуле:

$$СЭЭ = М * kз * Т * СкВт/ч, \quad (4.13)$$

где $M$ – мощность ЭВМ (0,2 кВт);

$k_з$ – коэффициент загрузки (0,8);

$С_{кВт-ч}$ – стоимость 1 кВт-час электроэнергии (16,53 тенге);

Т – время работы компьютера, час.

Следовательно, затраты на электроэнергию составят:

$$СЭЭ = 0{,}2 * 0{,}8 * 183{,}2 * 16{,}53 = 485 \text{ тенге.}$$

Расходы по статье «Спецоборудование» ($ПО$) включает в себя следующие затраты, представленные в таблице 4.4.

Таблица 4.4 - Программные продукты

| Наименование продукта | Стоимость, тенге |
|---|---|
| Visual Studio Code | 0 |
| PhpStorm PRO | 85 400 |
| Ноутбук ASUS X541 | 200 000 |

Расходы по статье «Научные командировки» (НК) на конкретное ПО определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате [11]:

$$НК = ЗпО * Н_{рнк}/100 \quad (4.14)$$

где $Н_{рнк}$ – норматив расходов на командировки в целом по организации в (%), равен 30%.

Тогда,

$$НК = 212\ 000 * \frac{30}{100} = 63\ 600 \text{ тенге.}$$

Расходы по статье «Прочие затраты» (П) на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате [11]:

$$П = ЗпО * Н_{пз}/100 \quad (4.15)$$

где $Н_{пз}$ – норматив прочих затрат в целом по организации в (%), равен 20%.

Тогда,

$$П = 212\ 000 * \frac{20}{100} = 42\ 400 \text{ тенге.}$$

Затраты по статье «Накладные расходы» (Н), рассчитывается по следующей формуле [11]:

$$Н = ЗпО * Н_{нр}/100 \quad (4.16)$$

где Н – накладные расходы на конкретную ПО (тыс. тенге);

$Н_{нр}$ – норматив накладных расходов в целом по организации в (%), равен 70%.

Следовательно,

$$Н = 212\ 000 * \frac{70}{100} = 148\ 400 \text{ тенге.}$$

Результаты выполненных расчетов запишем в таблицу 4.5. На рисунке 4.1 предоставлена диаграмма затрат.

Таблица 4.5 - Затраты на разработку информационных технологий

| Затраты на разработку | Значение, тенге | В процентах от общей суммы |
|---|---|---|
| Фонд оплаты труда | 233 200 | 37,91 |
| Социальный налог | 12 150 | 1,97 |
| Социальное отчисление | 7 345 | 1,19 |

| | | |
|---|---|---|
| Материалы | 10 600 | 1,72 |
| Спецоборудование | 85 400 | 13,88 |
| Расход на электричество | 485 | 0,08 |
| ВОСМС | 4 664 | 0,77 |
| Научные командировки | 63 600 | 10,35 |
| Прочие затраты | 42 400 | 6,89 |
| Накладные расходы | 148 400 | 24,13 |
| Амортизационные отчисления | 6 833 | 1,11 |
| Итого: | 615 077 | 100 |

Продукт создается по заказу от логистической компании.

Первоначальная цена рассчитывается через рентабельность разрабатываемого проекта. Рентабельность равна 50%, тогда:

$$C_п = C \times \left(1 + \frac{Р}{100}\right), \quad (4.17)$$

где Р – рентабельность.

$C_0 = 615\ 077 * \left(1 + \frac{50}{100}\right) = 922\ 616$ тенге.

Цена готовой продукции рассчитывается по формуле:

$$C_Р = C_0 + НДС, \quad (4.18)$$

где НДС – 12% от начальной цены готового продукта.
Рассчитаем НДС использовав формулу (4.18)

$$НДС = 922\ 616 * 0{,}12 = 110\ 714 \text{ тенге.}$$

Следовательно, конечная итоговая цена программного продукта составит:

$$C_Р = 922\ 616 + 110\ 714 = 1\ 033\ 330 \text{ тенге.}$$

### 4.3 Сравнительный анализ эксплуатационных затрат до и после внедрения

Статьи затрат после внедрения программного обеспечения:

– основная и дополнительная заработная плата 3 менеджеров с отчисляемыми налогами;

– расходуемые материалы;

– износ ПЭВМ;

– накладные расходы.

Статьи затрат до внедрения:

– основная и дополнительная заработная плата 3 операторов, 1 бухгалтера и 3 менеджеров;

– расходуемые материалы;

– накладные расходы.

***Расчет затрат после внедрения ПО***

Таблица заработной платы и отчисляемых налогов представлена в таблице 4.6.

Таблица 4.6 - Затраты на заработную плату

| Сотрудник | Кол-во, чел | Заработная плата в месяц, тг | Заработная плата в год, тг | Социальное отчисление | Социальный налог | ВОСМС |
|---|---|---|---|---|---|---|
| Менеджер | 3 | 150 000 | 1 800 000 | 153 900 | 56 700 | 36 000 |
| Итого | 3 | 450 000 | 5 400 000 | 461 700 | 170 100 | 108 000 |
| К оплате | 6 139 800 | | | | | |

Износ ПЭВМ рассчитывается исходя из 25 % амортизационных отчислений за год. Занесем данные в таблицу 4.7

Таблица 4.7 - Амортизационные отчисления

| Оборудование | Количество | Стоимость, тенге | Отчисления |
|---|---|---|---|
| Ноутбук | 3 | 600 000 | 150 000 |
| МФУ | 1 | 40 000 | 10 000 |
| Итого | 4 | 640 000 | 160 000 |

Затраты на расходуемые материалы занесем в таблицу 4.8

Таблица 4.8 - Затраты на расходуемые материалы

| Материал | Количество | Стоимость, тенге |
|---|---|---|
| Бумага | 6 | 6 000 |
| Краска черная | 1 | 2 990 |
| Краска желтая | 1 | 2 990 |
| Краска красная | 1 | 2 990 |
| Краска синяя | 1 | 2 990 |
| Итого | 10 | 17 960 |

Общие накладные расходы рассчитываются по формуле:

$$З_{Н.Р.} = 6\,139\,800 * 0{,}2 = 1\,227\,960 \text{ тенге.}$$

Итого, общие эксплуатационные расходы после внедрения программного продукта будут равны,

$$З = 6\,139\,800 + 160\,000 + 17\,960 + 1\,227\,960 = 7\,545\,720 \text{ тенге.}$$

***Расчет затрат до внедрения ПО***

Таблица заработной платы и отчисляемых налогов представлена в таблице 4.9.

Таблица 4.9 - Затраты на заработную плату

| Сотрудник | Кол-во, чел | Заработная плата в месяц одного, тг | Заработная плата в год, тг | Социальное отчисление | Социальный налог | ВОСМС |
|---|---|---|---|---|---|---|
| Менеджер | 3 | 120 000 | 4 320 000 | 369 360 | 136 080 | 84 600 |
| Оператор | 3 | 100 000 | 3 600 000 | 307 800 | 113 400 | 72 000 |
| Бухгалтер | 1 | 150 000 | 1 800 000 | 153 900 | 56 700 | 36 000 |
| Итого | 7 | 370 000 | 9 720 000 | 831 060 | 306 180 | 192 600 |
| К оплате | 11 049 840 | | | | | |

Износ ПЭВМ рассчитывается исходя из 25 % амортизационных отчислений за год. Занесем данные в таблицу 4.10

Таблица 4.10 - Амортизационные отчисления

| Оборудование | Количество | Стоимость, тенге | Отчисления |
|---|---|---|---|
| Ноутбук | 7 | 1 400 000 | 350 000 |
| МФУ | 2 | 80 000 | 20 000 |
| Итого | 4 | 1 480 000 | 370 000 |

Затраты на расходуемые материалы занесем в таблицу 4.11

Таблица 4.11 - Затраты на расходуемые материалы

| Материал | Количество | Стоимость, тенге |
|---|---|---|
| Бумага | 12 | 12 000 |
| Краска черная | 2 | 5 980 |
| Краска желтая | 2 | 5 980 |
| Краска красная | 2 | 5 980 |
| Краска синяя | 2 | 5 980 |
| Итого | 10 | 35 920 |

Общие накладные расходы рассчитываются по формуле:

$$З_{Н.Р.} = 11\,049\,840 * 0{,}2 = 2\,209\,968 \text{ тенге.}$$

Итого, общие эксплуатационные расходы до внедрения программного продукта будут равны,

$$З = 11\,049\,840 + 370\,000 + 35\,920 + 2\,209\,968 = 13\,665\,728 \text{ тенге.}$$

Для наглядности сведем все данные в аналитическую таблицу 4.12.

Таблица 4.12 - Годовые эксплуатационные затраты

| Статьи | До внедрения ПП | После внедрения ПП |
|---|---|---|
| Годовая заработная плата | 11 049 840 | 6 139 800 |
| Расходуемые материалы | 35 920 | 17 960 |
| Износ ПЭВМ | 370 000 | 160 000 |
| Накладные расходы | 2 209 968 | 1 227 960 |
| Всего | 13 665 728 | 7 545 720 |

Годовая экономия денежных средств будет составлять:

$$13\,665\,728 - 7\,545\,720 = 6\,120\,008 \text{ тенге.}$$

Необходимо определить экономический эффект от внедрения программного продукта.

Технико-экономические показатели позволяют определить целесообразность проведения разработки и ее внедрения, а также оценить реальную выгоду, как для разработчика системы, так и для ее пользователя.

Необходимо определить срок окупаемости программного продукта в месяцах по формуле:

$$П_{окуп} = 12 * \frac{З_{разработка}}{Э_{годовая}} \quad (4.19)$$

где $П_{окуп.}$ – период окупаемости в месяцах;

$З_{разр.}$ – затраты на разработку программы;

$Э_{годовая}$ – годовая экономия.

$$П_{окуп.} = 12 \cdot \frac{1\,033\,330}{6\,120\,008} = 12 \cdot 0{,}1688 = 2{,}03 \text{ мес.}$$

Зная годовую экономию и годовые эксплуатационные затраты с внедрением программного продукта можно определить коэффициент эффективности ($К_{эф.}$) по формуле:

Расчетный коэффициент экономической эффективности капитальных вложений составляет:

$$Е_р = \frac{Э_{уг}}{К}$$

$$E_р = \frac{6\,120\,008}{1\,033\,330} * 100\% = 592\% \text{ или } 5{,}92 \text{ тенге.}$$

Коэффициент эффективности показывает сколько экономии в тенге приходится на каждый тенге затрат. Следовательно, на каждый тенге затрат приходится 5,92 тенге экономии.

Сравнивая данные показатели видно, что внедрение программного продукта отлично отразиться на прибыльности предприятия, поскольку внедрение способствует уменьшению кадров, что в свою очередь уменьшит выплаты заработной платы и налогов. Кроме того, уменьшится количество оборудования и амортизационных отчислений вместе с потребляемой энергией. Программный продукт сделает работу более автоматизированной, тем самым повлечет за собой уменьшение материальных расходов. Кроме того, данная система увеличит количество заказчиков за оперативное выполнение работ.

## 5 Охрана труда и безопасность жизнедеятельности

Разрабатываемый программный продукт представляет собой информационную систему, которую можно использовать в браузере. Вся работа по разработке продукта проводилась в помещении без использования, какого-либо сложного оборудования. Все что необходимо для разработки это персональный компьютер и организованное рабочее место. Однако программист может быть подвержен многим неблагоприятным факторам при работе, основные из которых это недостаточность освещенности, шум от работающей техники, плохое кондиционирование, подверженность электромагнитному полю и электрическому току, неправильно оборудованное рабочее место.

Данная система может использоваться как рядовыми пользователями, так и IT-специалистами. Все манипуляции проводятся на персональном компьютере и не требуют больших навыков пользования компьютером. Целевая направленность данного продукта – внедрение в информационную систему компании для упрощения и повышение скорости поиска и устранения неисправностей.

### 5.1 Анализ условий труда в офисном помещении

Кабинет имеет следующие параметры:
Размеры: высота – 4 м, длина 12 м, ширина – 5 м и площадь 60 м$^2$;
1 окно размером 9 м на 2,5 м и площадью световых проемов 22,5 м$^2$;
Схема помещения приведена на рисунке 5.1

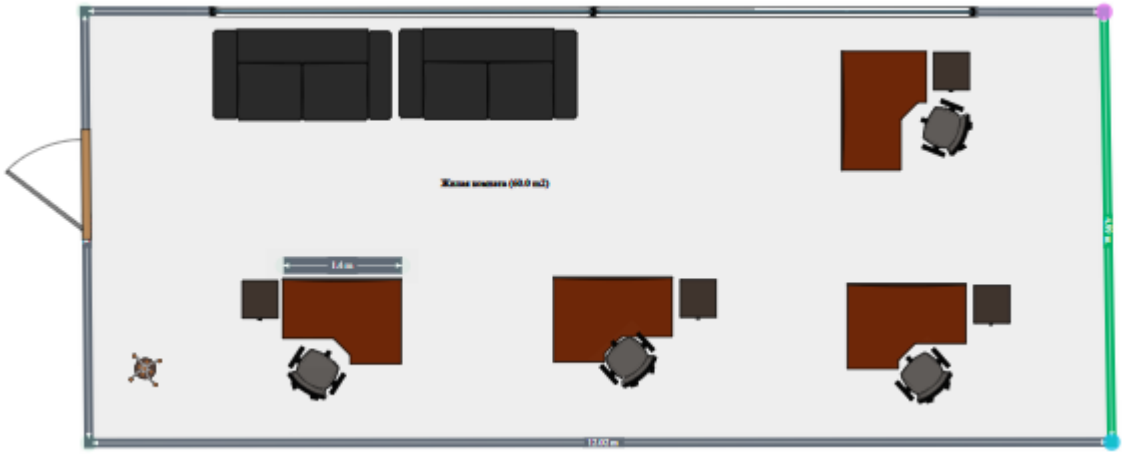


Рисунок 5.1 - Схема рабочего места

Работа программиста может быть определена как легкая 1а (работа, производимая сидя, не требующая сильного физического напряжения). Для данной категории работ по уровню энергозатратности оптимальные величины показателей микроклимата на рабочем месте рабочих помещений приведены в таблице 5.1.

Таблица 5.1 - Оптимальные величины показателей микроклимата

| Период года | Температура воздуха, °С | Температура поверхностей, °С | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
|---|---|---|---|---|
| Холодный | 21-23 | 20-25 | 40-60 | 0,1 |
| Теплый | 23-26 | 21-27 | 40-60 | 0,1 |

Микроклимат в помещении поддерживается с помощью кондиционера Elenberg CSH-09OB. Согласно характеристикам установленный в кабинете кондиционер, рассчитанный на площадь обслуживания до 40 м2, сможет поддерживать температуру и влажность, удовлетворяющую нормам микроклимата для помещений. Кабинет оборудован системой центрального отопления в холодное время года что так же удовлетворяет соблюдению микроклимата.

Правильно спроектированное и выполненное освещение помещений, оказывает положительное воздействие на работников, способствует повышению эффективности и безопасности труда, снижает утомление и минимизирует травматизм, сохраняет высокую работоспособность. В дневное время в помещении освещенность поддерживается естественным освещением, которое обеспечивают 2 окна. При недостаточности освещенности используется искусственное освещение с помощью четырех люминесцентных ламп. Это позволяет создать достаточную освещенность рабочего места. Рабочие столы расположены так, чтобы не отсвечивать на мониторе проекции естественного освещения, а именно лучей солнца которые создают нежелательную нагрузку на органы зрения.

Рабочие места соответствуют санитарно-эпидемиологическим требованиям. Площадь одного рабочее место на базе плоских дискретных экранов при любом расположении составляет не менее 4 $м^2$. Расстояние между рабочими столами с мониторами между боковыми поверхностями мониторов составляет не менее 1,2 м. Монитор находится от глаз на расстоянии 600 - 700 миллиметров, но не ближе 500 мм [14]. Схема размещения рабочих мест приведена на рисунке 5.2.

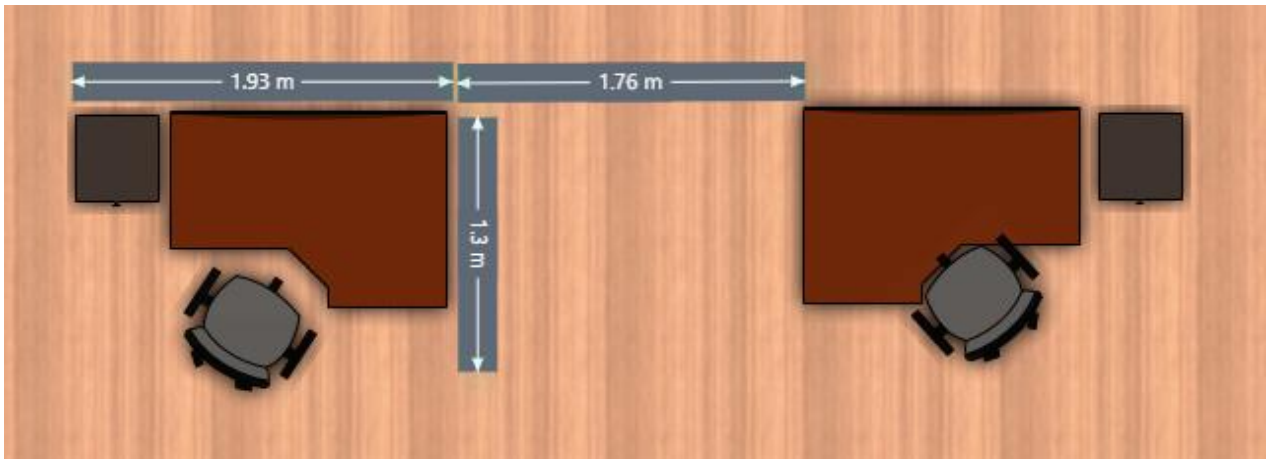



Рисунок 5.2 - Схема размещения рабочих мест

Как видно в приведенной выше схемой, площадь одного рабочего места составляет 14 м$^2$, что удовлетворяет утвержденным стандартам. Размещение рабочих мест в кабинете верное.

Рабочее место при выполнении работ сидя организуется в соответствии с ГОСТ 12.2.032-78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования»[14].

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, должно быть расположено в зоне легкой досягаемости рабочего пространства.

К требованиям эргономичного рабочего места относится следующее:

- площадь рабочего места;
- высота стола;
- положение кресла и размеры пространства для ног;
- возможность регулировки рабочего места и положения кресла;
- расстояние и углы обзора от монитора;
- расположение элементов рабочего пространства с соблюдением общих средних антропометрических показателей работника.

В кабинете имеются рекомендуемые размеры рабочего стола для средне статического взрослого человека:

- ширина стола – 1300 мм;
- глубина стола – 900 мм;
- высота – 750 мм.

Все конструкции рабочего места должны быть обеспечено оптимальное положение работающего, которое достигается регулированием высоты рабочей поверхности, сиденья, пространства для ног и т.д.

Кресло работников регулируется по высоте. Высота сиденья варьируется от 400 до 430 см. Спинка так же может изменять угол наклона. Угол наклона спинки кресла варьируется от 90 до 110 градусов. Это позволяет работнику произвести настройку под необходимые параметры и обеспечить максимальное удобство при работе.

### 5.2.1 Разработка системы естественного, искусственного освещения офиса

#### 5.2.1.1 Условие задачи и исходные данные

Рассчитать необходимую площадь окон для создания нормируемой естественной освещенности в производственном помещении.

В соответствии данным из таблицы 5.2:

- составить схему размещения светильников;
- рассчитать освещенность на рабочем месте точечным методом;
- привести нормативные значения освещенности согласно СниП РК 2.04-05.2002. Естественное и искусственное освещение. Общие требования[13];
- произвести реконструкцию системы освещения производственного помещения методом коэффициента использования;
- привести планируемую схему расположения светильников;
- сделать выводы.

Таблица 5.2 – Исходные данные для расчета освещения

| Тип помещения | Параметры помещения | | | hок, м | hнок, м | Разряд зрит. Работ |
|---|---|---|---|---|---|---|
| | L, м | B, м | H, м | | | |
| Офис | 12 | 5 | 4 | 2,5 | 1 | IV, в |

| Световой пояс | Нзд, М | Расст. До ближ. Здаиия Р, м | Коэффициенты отражения | | |
|---|---|---|---|---|---|
| | | | $\rho_{пот}$ | $\rho_{ст}$ | $\rho_{пол}$ |
| Алматинская область | 13 | 9 | 70 | 50 | 30 |

### 5.2.1.2 Расчет естественного освещения

Помещения с постоянным пребыванием людей должны иметь, как правило, естественное освещение.

Расчет естественного освещения заключается в определении площади световых проемов.

Общая площадь окон определяется по формуле для бокового освещения:

$$100\frac{S_o}{S_{\ n}} = \frac{e_н K_з \eta_o}{\tau_o r_1} K_{зд} \rightarrow S_o = \frac{S_n \cdot e_н K_з \eta_o K_{зд}}{100 \tau_o r}$$

где:

$S_o$ – площадь световых проемов при боковом освещении, м$^2$;

$S_n$ – площадь пола помещения, м$^2$;

$$S_n = L \cdot B = 12 \cdot 5 = 60 \text{ м}^2$$

$e_н$ – нормируемое значение КЕО;

$K_з$ –коэффициент запаса, принимают по таблице 3.11[12]. $K_з$=1,2;

$\eta_o$ – световая характеристика окон, принимают по таблице 3.2 [12];

По условию ширина помещения (В = 5м), следовательно, мы можем применить одностороннее освещение, то есть (l = 5 м):

L/l=12/5=2,4 м

hрасч = hок + hнок – hрп = 2,5 + 1 – 0,5 = 3 м

где hрп = 1 м – высота рабочей поверхности

l/ hрасч = 5/3 = 1.67 м

Следовательно, $\eta_o$=17,8 (по методу линейной интерполяции);

$\tau_o$ – общий коэффициент светопропускания, определяют по формуле:

$\tau_o = \tau_1 \tau_2 \tau_3 \tau_4$,

где $\tau_1$ – коэффициент светопропускания материала, $\tau_1$=0,8 (для стекла оконного листового двойного);

$\tau_2$ – коэффициент, учитывающий потери света в переплетах светопроема $\tau_2$=0,6 (переплеты деревянные двойные раздельные);

$\tau_3$ – коэффициент, учитывающий потери света в несущих конструкциях $\tau_3$=0,8 (железобетонные и деревянные формы и арки);

$\tau_4$ – коэффициент, учитывающий потери света в солнцезащитных устройствах, $\tau_4$=1 (убирающиеся регулируемые жалюзи и шторы);

$\tau_o=\tau_1\tau_2\tau_3\tau_4=0,8*0,6*0,8*1=0,384$

Расстояние от расчетной точки до наружной стены составит В = 5 м.

Отношение расстояния расчетной точки от наружной стены к глубине помещения:

$$В/l=5/5=1 \text{ м}$$

Отношение глубины помещения к высоте от уровня условной рабочей поверхности верха окна:

$$l/ h_{расч}=5/3=1,67 \text{ м}$$

$$\frac{P_{пот} + P_{ст} + P_{пол}}{3} = \frac{70 + 50 + 30}{3} = 50\ \%$$

$$L/l = 12/5 = 2,4 \text{ м}$$

$$r_1 = 2,4$$

$r_1$ – коэффициент, учитывающий повышение КЕО при боковом освещении, благодаря свету, отраженному от поверхности помещения и подстилающего слоя, примыкающего к зданию;

$К_{зд}$ – коэффициент, учитывающий затемнение окон противостоящими зданиями, принимают по таблице 3.8 [12]:

$$P/Н_{зд} = 9/13 = 0,7$$

$К_{зд}$ = 1,58 (по методу линейной интерполяции)

Нормированные значения КЕО $e_н$ для зданий, располагаемых в различных районах, следует определять по формуле:

$$e_N = e_н \cdot m_N = 1,5 \cdot 0,9 = 1,35$$

где $N$ – номер группы обеспеченности естественным светом по таблице 3.1[12];

$e_н$ – значения КЕО по таблице 3.12 для IV(в) разряда зрит. Работ при естественном боковом освещении;

$m_N$ – коэффициент светового климата, табл.3.1 [12];

Подставим все значения в расчетную формулу:

$$S_o = \frac{S_п \cdot e_н K_з \eta_o K_{зд}}{100 \tau_o r}$$

$$S_o = \frac{60 \cdot 1{,}35 \cdot 1{,}2 \cdot 17{,}8 \cdot 1{,}58}{100 \cdot 0{,}384 \cdot 2{,}4} = 29{,}66203 м^2$$

$$l_{ok} = \frac{S_o}{h_{ok}} = 29{,}66203/2{,}5 = 8{,}86481 м.$$

Таким образом, площадь светового проема составит 29,66203 м$^2$ (9,86481 х 2,5) (см. рисунок 5.3)

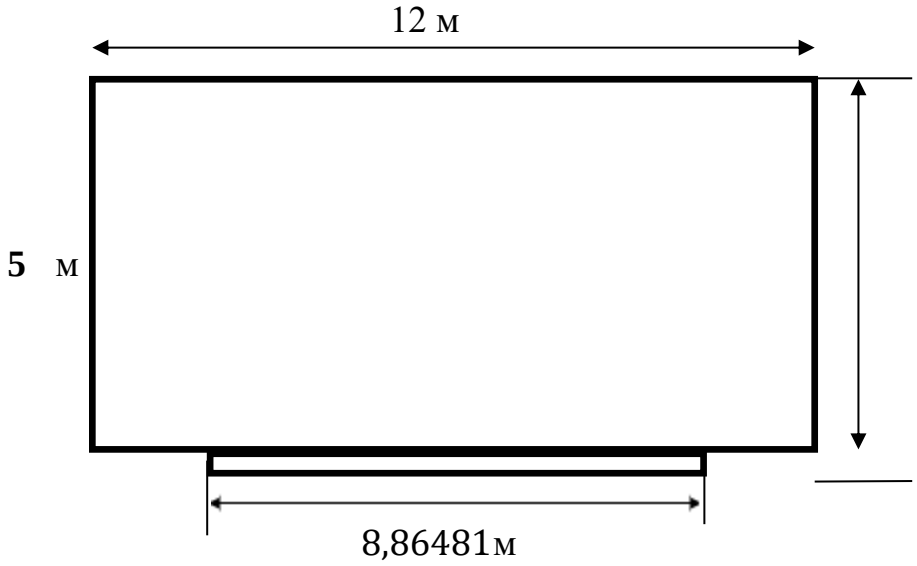



Рисунок 5.3 – Схема помещения при естественном освещении

**Вывод:** в данном задании была рассчитана площадь боковых световых проемов учебной лаборатории, необходимой для создания нормируемой освещенности на рабочих местах для разряда зрительной работы IV(в). Для помещения размерами 12 х 5 х 4 м$^3$ значение для площади одного светового проема получилось равным 29,66203 м$^2$, где высота окна $h_{ок}$ = 2,5 м, а длина окна lок = 8,86481 м.

### 5.2.2 Расчет искусственного освещения

Для расчета искусственного освещения используют один из трех методов: по коэффициенту использования светового потока, точечный и метод удельной мощности.

При расчете общего равномерного освещения основным является метод использования светового потока, создаваемого источником света, и с учетом отражения от стен, потолка, пола.

Расчет освещения начинают с выбора типа светильника, который принимается в зависимости от условий среды и класса помещений по взрывопожароопасности.

### 5.2.2.1 Расчет освещенности точечным методом

Разряд зрительной работы IV (в), поэтому нормируемая освещенность (по таблице 5.1) $E_н$ = 200 лк (при системе общего освещения).

Определение расчетной высоты подвеса:

$$h_{расч} = H_{помещения} - H_{свеса} - H_{р.п.},$$

где $H_{свеса}$= 0,3 – высота свеса ламп, м;

$H_{р.п.}$= 0,5 – расстояние рабочей поверхности над полом, м;

$H_{помещения} = 4$ – высота помещения, м.

$$h_{расч} = 4 - 0{,}3 - 0{,}5 = 3{,}2 \text{ м}$$

Определение наивыгоднейшего расстояния между светильниками (Z): Расстояния между соседними светильниками L определяется как L= $\lambda * h$. Рекомендуемое значение $\lambda$ равное 0,5-1,5 для равномерного освещения.

La= 0,9375 * 3,2 = 3 м

la= 0,9375 * 3,2/2 = 1,5 м

$L_b$= 0,9375 * 3,2 = 3 м

$l_b$= 0,9375 * 3,2/3 = 1 м.

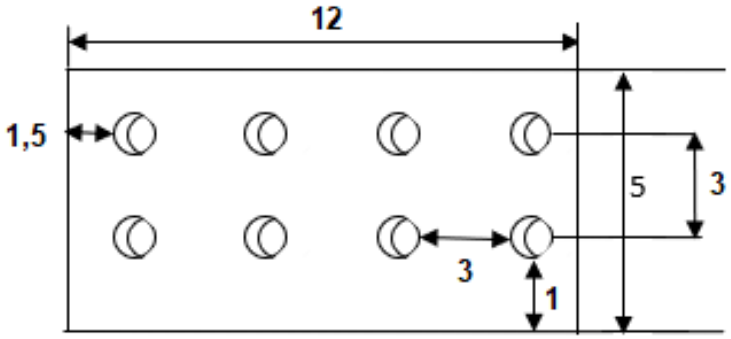



Рисунок 5.4 – Расчетная схема точечного метода

Исходя из габаритов помещения, используем 2 ряда по 4 светильников в каждом. Расстояние между светильниками 3 метра, между рядами 3 метра, от стены до ряда 1,5 метра, от боковых стен 1 метр.

Выберем контрольную точку А (рисунок 5.5). Из (1)→

$$\rightarrow \qquad E = \frac{\Phi \cdot \mu \cdot \sum_{i=1}^{4} E_y}{1000 \cdot K_3}$$

Величина условной освещенности зависит от светораспределения светильника и геометрических размеров: расстояния от точки до проекции освещающего ее светильника ($d = \sqrt{b^2 + c^2}$) и высоты расположения светильника над освещаемой поверхностью (h) (рисунок 3).

$$E_y = \frac{I_\alpha \cdot \cos^3 \alpha}{h^2}$$

где $I_\alpha$ - сила света, зависит от типа источника и угла падения света.

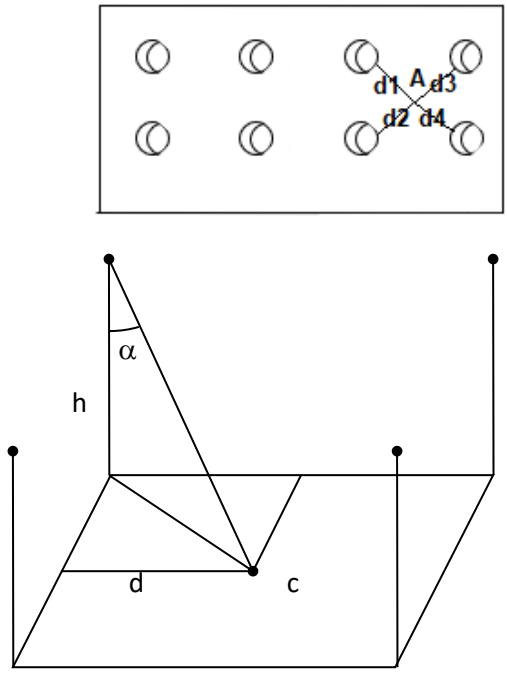



Рисунок 5.5 - Выбор точки

1,2,3,4-я лампы d =2,12 м;

$tg\alpha_{1,2,3,4} = \frac{d_{1.1.2.3.4}}{h_{расч}} = \frac{2,12}{3.2} = 0 \rightarrow \alpha_{1,2,3,4} = 33,52°$;

$cos^3 \alpha_1 = 0,579$

$E_y = \frac{300 \cdot 0,579}{3,2^2} = 16,96$ лк;

$E_{yобщ} = \frac{300 \cdot 0,579}{3,2^2} * 4 = 67,85$ лк

Тип лампы - LINER/S DR LED 1500 W 4000K, мощность 33 Вт, световой поток 2850 лм.

$$E_г = \frac{2850*1,2*67,85}{1000*1,1} = 210,95 \text{ лк}$$

Вывод: Освещенность на рабочем месте считается достаточной, если Ег>Ен; Ен = 200 лк (таблица 5.1). В данном случае условие выполняется, т.е. Ег= 210,95 лк > 200 лк.

## Заключение

В результате данного дипломного проекта были выполнены поставленные в введении цели и задачи. Были проанализированы существующие логистические системы. В ходе работы было спроектировано информационное обеспечение программного обеспечения: выбрана СУБД для хранения данных, построена физическая и логическая модели базы данных, на основе которого были созданы и описаны структуры таблиц БД, описаны связи между ними.

В ходе проектирования программного обеспечения была описана структура ПО, обоснован выбор инструментально ПО (IDE WebStorm, СУБД PostgreSQL, Zend Framework, язык программирования PHP 7), разработаны прототипы интерфейса и написаны основные классы и методы для нормального функционирования системы.

## Список литературы


1 PHP 7. В подлиннике Игорь Симдянов, Дмитрий Котеров 2016.

2 Vue.js в действии. - СПб.: Питер, 2019. (Серия «Библиотека программиста»).

3 Effective TypeScript: 62 Specific Ways to Improve Your TypeScript 2019.

4 PHP: объекты, шаблоны и методики программирования 2-е издание Мэтт Зандстра 2015.

5 Официальная документация Zend https://framework.zend.com/learn

6 Официальная документация Vue JS https://vuejs.org/v2/guide/

7 Официальная документация Vuetify https://vuetifyjs.com/en/getting-started/quick-start/

8 Статья «Преимущества языка программирования PHP» на сайте http://www.php.su/php/?opport

9 Статья «Обзор возможностей популярных PHP фреймворков – zend, codeignater, phalcon и laravel» на сайте https://hyperhost.ua/info/obzor-vozmozhnostey-populyarnyih-php-freym

10 Статья «PostgreSQL» на сайте https://ru.opensuse.org/Postgresql

11 Г.Ш.Боканова. Методические указания по выполнению экономической части дипломных работ для студентов специальностей 5В070400 – «Вычислительная техника и программное обеспечение», 5В070300 – «Информационные системы», 5В060200 – «Информатика». Алматы, АУЭС – 2020год.

12 Абдимуратов Ж.С., Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. Бакалавриат - Алматы: АИЭС, 2009. - 20 с.

13 СНиП РК 2.04-05-2002. Естественное и искусственное освещение. Общие требования. – М.: Стройиздат, 2002.

14 СНиП II - 4 - 79. Естественное и искусственное освещение. Нормы проектирования.-М.: Стройиздат, 1980.

15 Фаулер М. UML. Основы. 3-е издание. Символ-Плюс, 2005.

16 Буч Г., Рамбо Д., Якобсон А. Язык UML. Руководство пользователя. Второе издание. ДМК, 2006.

## Приложение А
## Техническое задание

1 Общие сведения

1.1 Наименование системы

1.1.1 Полное наименование системы

Полное наименование: Информационная система логистической ТОО «Asia Freight»

1.1.2 Краткое наименование системы

Краткое наименование: Delivery System

1.1.3 Основания для проведения работ

Работа выполняется на основании договора от 1.08.2019 между руководством ТОО «Asia Freight» и Исмадулла Ш. Н.

1.1.4 Наименование организаций Заказчика и Разработчика

1.1.4.1 Заказчик

Заказчик: ТОО «Asia Freight»
Адрес фактический: г. Алматы, Закарпатская, 158
Телефон / Факс: +7 (727) 393 11 10, +7 (727) 393 28 88

1.1.4.2 Разработчик

Разработчик: Исмадулла Ш.Н
Адрес фактический: : г. Алматы
Телефон / Факс: +7 (708) 284 28 42

1.2 Плановые сроки начала и окончания работы

Работа проводится в соответствии графику, представленному в таблице 1.

Таблица 1 - Плановые сроки работы

| № | Наименование работы | Начало работы | Конец работы |
|---|---|---|---|
| 1 | Анализ существующих систем для логистической компании | 17.02.2020 | 28.02.2020 |
| 2 | Проектирование функциональной структуры ИС | 2.03.2020 | 13.03.2020 |
| 3 | Разработка бизнес-модели | 16.03.2020 | 27.03.2020 |

*Продолжение таблицы 1*

| | | | |
|---|---|---|---|
| 4 | Описание структуры программного обеспечения | 30.03.2020 | 10.04.2020 |
| 5 | Разработка пользовательского интерфейса системы | 13.04.2020 | 24.04.2020 |
| 6 | Тестирование и отладка кода ИС | 27.04.2020 | 08.05.2020 |

1. 2.1 Порядок оформления и предъявления заказчику результатов работ

Работы по созданию данной системы предоставляются Разработчиком согласно календарному плану проекта поэтапно.

Результаты работ предъявляются заказчику в виде документов – отчетов, которые содержат скриншоты части работ, далее все необходимые файлы загружаются на тестовый сервер компании для дальнейшей тестировки и демонстрации системы.

1.3 Назначение и цели создания (развития) системы

1.3.1 Цели создания системы

Разработать информационную систему согласно современным стандартам интерфейсов «Material Design» и с помощью современных веб-технологий.

1. 4 Характеристика объектов автоматизации

Объектом автоматизации является логистическая компания ТОО «Asia Freight», а точнее процессы, происходящие в нем. Основным функционалом которого является представление информации о заявках, которые создаются менеджером в системе. Возможность добавление пользователей, групп и ролей для управления данной системой. Кроме того, в информационной системе будет доступна возможность добавление водителей, а также возможность закрепления их заявкам. Водителям доступна принятие заявки на забор или довоз груза по городу. В системе должно быть осуществлено ролевое разделение (менеджер, водитель, администратор).

1.5 Требования к системе

1.5.1 Требования к системе в целом

Взаимодействия производятся по протоколу HTTPS.

В системе предлагается выделить следующие функциональные подсистемы:

- подсистема «Клиенты» (СУБД PostgreSQL);
- подсистема «Заявки» (СУБД PostgreSQL);

- подсистема «Пользователи» (СУБД PostgreSQL);
- подсистема «Водители» (СУБД PostgreSQL).

В *режиме функционирования* система должна обеспечивать:

- работу пользователей в режиме – 24 часа в день, 7 дней в неделю (24x7);
- выполнение своих функций: сбор, обработка и загрузка данных; хранение данных.

Необходимо обеспечить защиту пользовательских данных.

1.5.2 Требования к функциям, выполняемым системой

С точки зрения пользователя «Менеджер»:

1 Добавление заявок на доставку или забор груза

2 Просмотр списка всех заявок

Отражается информация о заявках, то есть сведения о посылке, клиенте, который сделал заявку, сведения о местоположении груза и куда необходимо доставить.

3 Просмотр списка пользователей

Данная страница будет включать в себя таблицу и форму для добавления нового пользователя. В таблице хранятся данные о пользователях системы, здесь же есть возможность изменить доступы для входа в систему.

4 Просмотр списка водителей

Данная страница будет включать в себя таблицу и форму для добавления нового водителя. В таблице хранятся данные о водителях компании , здесь же есть возможность изменить доступы для входа в систему, а также активация и деактивация водителя.

Требования к интерфейсу приложения:

Страница Заявки:

- таблица заявок на забор;
- таблица заявок на довоз;
- таблица всех заявок;
- форма добаления новой заявки.

С точки зрения пользователя «Водитель»

1 Просмотр заявок

Отражается информация о заявках, то есть сведения о посылке, клиенте, который сделал заявку, сведения о местоположении груза и куда необходимо
доставить. Для водителя отображаются только те заявки, которые относятся городу водителя.

2 Выбор заявки

Водитель выбирает заявку и после заявка выбранная водителем появляется на странице «Мои заявки».

1.5.3 Требования к ПО

Клиентская часть информационной системы разработана в среде программирования «WebStorm IDEA 2020.1» на языке программирования TypeScript.

Серверная часть пишется на языке программирования PHP версии 7.2. В качестве системы управления базами данных выступает СУБД PostgreSQL версии 10.5.

1.6 Состав и содержание работ по созданию системы

Работы по созданию системы выполняются в три этапа: проектирование, разработка интерфейса системы, разработка технического проекта.

Подробные сведения о этапах разработки приведены в пункте 1.2.6 Плановые сроки начала и окончания работы.

1.7 Источник разработки

Настоящее Техническое Задание разработано на основе следующих документов и информационных материалов:

- договор от 1.08.2019 между руководством от 1.08.2019 между руководством ТОО «Asia Freight» и Исмадулла Ш. Н.

## Приложение Б
## Листинг программы

### Листинг 1. Класс LoginPage:

```
import { Component, Vue } from 'vue-property-decorator';
import { AuthForm } from '../service/authService';
import { Action } from 'vuex-class';
import { Success } from '@/modules/auth/store/actions';
const namespace: string = 'auth';

@Component
export default class LoginPage extends Vue {
  @Action('authRequest', { namespace }) public authRequest: any;

  public form: AuthForm = { identifier: '', password: '' } ;
  public showPassword: boolean = false;

  private async login() {
    try {
      const success = await this.authRequest(this.form);
      if (success instanceof Success) {
        this.$store.commit('app/showMessage', ['success', 'Успешный вход в систему']);
        this.$router.push('/');
      }
    } catch (error) {
      // TODO: update ui with error
      alert('Неправильный логин или пароль.');
    }
  }
}
```

### Листинг 2. Класс RequestList:

```
import {Component, Vue} from 'vue-property-decorator';
import {Action} from 'vuex-class';
import {namespace} from '@/modules/request/store';
import {Request} from '@/network/data/request';
import {requestService, Search} from '@/modules/request/service/requestService';
import moment from 'moment';
import {saveAs} from 'file-saver';
import {AxiosResponse} from 'axios';
import {
  DeliveryRequestTableRowData,
  RequestTableRowData,
} from '@/modules/request/data/requestTableData';
import {RequestData} from '@/modules/request/data/requestData';

@Component
export default class RequestList extends Vue {
  @Action('requestList', {namespace}) public requestList: any;
  @Action('downloadReport', {namespace}) public downloadReport: any;

  public searchStatuses: string[] = [
    'Сформирован',
    'Принят на склад отправления AsiaFreight',
    'Передан в регион',
```

```
  'Доставлено',
];
public pickupCities: string[] = [
  '',
  'Алматы',
  'Нур-Султан',
];
public statuses: string[] = [
  'Сформирован',
  'Принят на склад отправления AsiaFreight',
  'Передан в регион',
  'Доставлено',
];
public selectedRequests: string[] = [];
public requestUuid: string = '';
public note: string = '';
public commentBtnDisabled: boolean = true;
public deleteCommentBtn: boolean = true;
public trackNumber: string = '';
public status: string = '';
public byUpdatedAt: boolean = false;
public pickupCity: string = '';
public deliveryCity: string = '';
public startDate: string = moment().format('YYYY-MM-DD');
public endDate: string = moment().format('YYYY-MM-DD');
public async fetchRequestList() {
  this.selectedRequests = [];
  this.requests = [];
  this.loading = true;
  const search: Search = {
    trackNumber: this.trackNumber,
    status: this.status,
    pickupCity: this.pickupCity,
    deliveryCity: this.deliveryCity,
    startDate: this.startDate,
    endDate: this.endDate,
    byUpdatedAt: this.byUpdatedAt,
  };
  try {
    const result = await this.requestList(search);
    this.requests = result.data.requests;
    this.loading = false;
  } catch (error) {
    this.$store.commit('app/showMessage', ['error', 'Заявки не найдены']);
    this.loading = false;
  }
}
public async fetchDeliveryCities() {
  requestService.fetchDeliveryCities()
    .then((response: any) => {
      console.log(response);
      if (response.data.length > 0) {
        this.deliveryCities.push('');
        for (const city of response.data) {
          this.deliveryCities.push(city.name);
```

```
        }
      })
      .catch((error: any) => {
        this.$store.commit('app/showMessage', ['error', error.response.data.message]);
      });
  }

  public selectRequest(event: any, uuid: string) {
    if (event.target.tagName === 'A') {
      return false;
    }
    const index = this.selectedRequests.indexOf(uuid);
    if (index === -1) {
      this.selectedRequests.push(uuid);
    } else {
      this.selectedRequests.splice(index, 1);
    }
  }

  public isRequestSelected(uuid: string) {
    return this.selectedRequests.indexOf(uuid) !== -1;
  }

  public isDelaying(deliveryUntil: string, status: string) {
    if (status === 'Доставлено') {
      return false;
    }
    return moment(deliveryUntil).add(-1, 'days') < moment();
  }

  public async downloadExcelReport() {
    const search: Search = {
      trackNumber: this.trackNumber,
      status: this.status,
      pickupCity: this.pickupCity,
      deliveryCity: this.deliveryCity,
      startDate: this.startDate,
      endDate: this.endDate,
      byUpdatedAt: this.byUpdatedAt,
    };
    try {
      const result = await requestService.downloadReport(search);
      // TODO: save file
      saveAs(result.data, 'Отчет о заявках.xlsx');
    } catch (error) {
      this.$store.commit('app/showMessage', ['error', error.toString()]);
    }
  }

  public async downloadPdfReport() {
    try {
      const result = await
requestService.downloadReportPdf(this.requestData.trackNumber);
      // TODO: save file
      saveAs(result.data, this.requestData.trackNumber + '.pdf');
```

```
    } catch (error) {      this.$store.commit('app/showMessage', ['error',
error.toString()]);
    }
  public clearChangeStatusDialog() {
    this.changeStatusTo = '';
    this.statusDate = '';
    this.changeStatusDialog = false;
  }

  public async addComment() {
    requestService.addNote(this.requestUuid, this.note)
      .then((response: any) => {
        this.$store.commit('app/showMessage', ['success', 'Ваш комментарий успешно
добавлен']);
        this.deleteCommentBtn = true;
        this.fetchRequestList();
      })
      .catch((error) => {
        this.$store.commit('app/showMessage', ['error', error.response.data.message]);
      });
  }

  public async deleteComment() {
    requestService.deleteNote(this.requestUuid)
      .then((response: any) => {
        this.$store.commit('app/showMessage', ['warning', 'Комментарий удален']);
        this.deleteCommentBtn = false;
        this.note = '';
        this.commentBtnDisabled = true;
        this.fetchRequestList();
      })
      .catch((error) => {
        this.$store.commit('app/showMessage', ['error', error]);
      });
  }

  get pickupTableData() {
    const tableData: PickupRequestTableRowData[] = [];
    for (const request of this.requests) {
      let driver = null;
      if (request.pickup !== null) {
        if (this.currentUserCity === request.pickup.city || this.currentUserRole !==
'driver') {
          if (request.pickup.driver !== null) {
            driver = request.pickup.driver.fullName;
          }
          const tableRow: PickupRequestTableRowData = {
            requestUuid: request.uuid,
            address: request.pickup.city + ', ' + request.pickup.address,
            fullName: request.pickup.shipper,
            phone: request.pickup.phone,
            requestPackagesAmount: request.amountPackages,
            requestTotalWeight: request.packages.reduce((a, b) => a + (b.weight || 0),
0),
            requestTrackNumber: request.trackNumber,
            requestStatus: request.status,
            requestCreatedAt: request.createdAt,
```

```
            plannedPickupAt: request.pickup.plannedPickupAt,
            driver,
            requestNote: request.note,
          };
          tableData.push(tableRow);
        }
      }
    }
    return tableData;
  }

  get totalPackages() {
    let amount: number = 0;
    if (this.requests.length > 0) {
      amount = this.requests.map((request: Request) =>
request.amountPackages).reduce((a, count) => a + count);
    }
    return amount;
  }

  get isSelectedRequestsContainsNotDelivered() {
    return this.requestData.status !== 'Доставлено';
  }

  get group() {
    return this.$store.state.auth.jwtData.group;
  }

  get currentUserRole() {
    return this.$store.state.auth.jwtData.role;
  }

  get currentUserCity() {
    return this.$store.state.auth.jwtData.user.city;
  }

  get currentUserUuid() {
    if (this.currentUserRole === 'driver') {
      return this.$store.state.auth.jwtData.user.uuid;
    }
    return this.$store.state.auth.jwtData.user;
  }

  public mounted() {
    this.fetchDeliveryCities();
    this.fetchRequestList();
  }

}
```

Листинг 3. Класс RequestAddPage:

```
import {Component, Vue} from 'vue-property-decorator';
import DeliveryForm from './DeliveryForm.vue';
import PickupForm from './PickupForm.vue';
import {
  City,
```

```
  Client,
  DeliveryDataForm, Packages,
  PickupDataForm,
  RequestDataForm,
  Type,
} from '@/modules/request/data/requestData';
import {Package} from '@/network/data/package';
import {Pickup} from '@/network/data/pickup';
import {Delivery} from '@/network/data/delivery';
import {requestService} from '@/modules/request/service/requestService';
import validator_rules from '@/modules/request/views/validate';
import {Address} from '@/modules/client/data/clientData';

@Component({
  components: {
    DeliveryForm,
    PickupForm,
  },
})
export default class RequestAddPage extends Vue {

  public rules = validator_rules;
  public saveButtonLoader: boolean = false;
  public menu: boolean = false;
  public clients: Client[] = [];
  public errors: string[] = [];
  public cities: City[] = [];
  public types: Type[] = [];
  public packages: Package[] = [];

  public packageData = {} as Package;

  public pickupForm: PickupDataForm = {
    pickup: {} as Pickup,
  };

  public deliveryForm: DeliveryDataForm = {
    delivery: {} as Delivery,
  };

  public type: Type = {} as Type;

  public addPickupForm($event: Pickup) {
    this.pickupForm.pickup = $event;
  }

  public addDeliveryForm($event: Delivery) {
    this.deliveryForm.delivery = $event;
  }

  public showTypeField() {
    this.menu = !this.menu;
  }

  public addPackageField() {
    this.packageData = {} as Package;
    this.packages.push(this.packageData);
```

```
  }

  public removePackageField(uuid: string, index: number) {
    if (typeof uuid === 'undefined') {
      this.packages.splice(index, 1);
    }
  }


  public addType() {
    this.$store.commit('loader', true);
    requestService.addType({name: this.type.name})
      .then((response: any) => {
        if (response.status === 201) {
          this.$store.commit('loader', false);
          this.type.name = '';
          this.fetchTypes();
          this.packageData.productType = response.data.uuid;
          this.menu = !this.menu;
          this.$store.commit('app/showMessage', ['success', 'Тип посылки успешно
добавлен']);
        }
      })
      .catch((error: any) => {
        this.$store.commit('app/showMessage', ['error', error.response.data.message]);
        this.$store.commit('loader', false);
      });
  }

  public addRequest() {
    if ((this.$refs.form as Vue & { validate: () => boolean }).validate()) {
      this.saveButtonLoader = true;
      this.$store.commit('loader', true);
      if (this.$store.state.pickupFormShowed || this.$store.state.deliveryFormShowed)
{
        requestService.addRequest(this.packages)
          .then((response) => {
            const uuid = response.data.uuid;
            if (response.status === 201) {
              this.$store.commit('loader', false);
              this.saveButtonLoader = false;
              if (this.$store.state.pickupFormShowed) {
                requestService.addPickupForm(uuid, this.pickupForm.pickup)
                  .then((res: any) => {
                    if (res.status === 201) {
                      this.$store.commit('loader', false);
                      this.saveButtonLoader = false;
                      this.$router.push({name: 'request'});
                      this.packageData = {} as Package;
                      this.pickupForm.pickup = {} as Pickup;
                      if (!this.$store.state.deliveryFormShowed) {
                        this.$store.commit('app/showMessage', ['success', 'Заявка на
забор добавлена']);
                      }
                    }
                  })
                  .catch((error: any) => {
```

```
                    this.$store.commit('loader', false);
                    this.saveButtonLoader = false;
                    this.$store.commit('app/showMessage', ['error',
error.response.data.message]);
                  });
              }
              if (this.$store.state.deliveryFormShowed) {
                requestService.addDeliveryForm(uuid, this.deliveryForm.delivery)
                  .then((res) => {
                    if (res.status === 201) {
                      this.$store.commit('loader', false);
                      this.saveButtonLoader = false;
                      this.$router.push({name: 'request'});
                      this.packageData = {} as Package;
                      this.deliveryForm.delivery = {} as Delivery;
                      if (!this.$store.state.pickupFormShowed) {
                        this.$store.commit('app/showMessage', ['success', 'Заявка на
довоз добавлена']);
                      }
                    }
                  })
                  .catch((error) => {
                    this.$store.commit('loader', false);
                    this.saveButtonLoader = false;
                    this.$store.commit('app/showMessage', ['error',
error.response.data.message]);
                  });
              }
            }
          })
          .then(() => {
            if (this.$store.state.deliveryFormShowed &&
this.$store.state.pickupFormShowed) {
              this.$store.commit('app/showMessage', ['success', 'Заявка успешно
добавлена']);
            }
          })
          .catch((error) => {
            this.$store.commit('loader', false);
            this.saveButtonLoader = false;
            this.$store.commit('app/showMessage', ['error',
error.response.data.message]);
          });
      } else {
        this.$store.commit('loader', false);
        this.saveButtonLoader = false;
        this.$store.commit('app/showMessage', ['error', 'Выберите забор или довоз']);
      }
    }
  }
}
```

Листинг 4. Класс ClientList:

```
import {Component, Vue} from 'vue-property-decorator';
import {Client, TransformedClient} from '@/modules/client/data/clientData';
import {clientService} from '@/modules/client/service/clientService';
```

```
@Component
export default class ClientList extends Vue {
  public clients: Client[] = [];
  public errorText: string = '';


  get transformedClients() {
    const transformedClients: TransformedClient[] = [];
    for (const client of this.clients) {
      let clientPhones: string = '';
      let clientAddresses: string = '';

      if (client.phones && client.phones.length > 0) {
        clientPhones = client.phones.map((p) => p.number).join(', ');
      }
      if (client.addresses && client.addresses.length > 0) {
        clientAddresses = client.addresses.map((a) => a.city.name + ', ' +
a.name).join('; ');
      }
      const filteredClient: TransformedClient = {
        uuid: client.uuid,
        fullName: client.firstName + ' ' + client.lastName,
        phones: clientPhones,
        addresses: clientAddresses,
      };
      transformedClients.push(filteredClient);
    }
    return transformedClients;
  }

  public fetchClients(): void {
    clientService.fetchClients()
      .then((response) => {
        this.clients = response.data.clients;
      })
      .catch((error) => {
        this.errorText = error.response.data.message;
      });
  }

  public mounted() {
    this.fetchClients();
  }
}
```

## Листинг 5. Класс NewClient:

```
import {Component, Vue} from 'vue-property-decorator';
import {Address, City, NewClientData, Phone} from '@/modules/client/data/clientData';
import {clientService} from '@/modules/client/service/clientService';
@Component
export default class NewClient extends Vue {
  public phoneRemoveModal: boolean = false;
  public addressRemoveModal: boolean = false;
  public saveButtonLoader: boolean = false;
```

```
  public allSaved: boolean = false;
  public errors: string[] = [];
  public clientForm: NewClientData = {
    firstName: '',
    lastName: '',
  } as NewClientData;

  public clientPhones: Phone[] = [];
  public clientAddresses: Address[] = [];
  public phones: Phone = {} as Phone;
  public addresses: Address = {} as Address;


  public addPhoneField() {
    this.phones = {} as Phone;
    this.clientPhones.push(this.phones);
  }

  public addAddressField() {
    this.addresses = {} as Address;
    this.clientAddresses.push(this.addresses);
  }

  public deletePhoneField(uuid: string, index: number) {
    if (typeof uuid === 'undefined') {
      this.clientPhones.splice(index, 1);
    } else {
      this.phoneRemoveModal = true;
    }
  }

  public deleteAddressField(uuid: string, index: number) {
    if (typeof uuid === 'undefined') {
      this.clientAddresses.splice(index, 1);
    } else {
      this.addressRemoveModal = true;
    }
  }

  public savePhone(uuid: string) {
    clientService.savePhone(uuid, this.clientPhones)
      .then(() => {
        if ((this.clientPhones.length === 0 || this.allSaved) &&
(this.clientAddresses.length === 0 || this.allSaved)
          && (this.errors.length === 0)) {
          this.$router.push({name: 'client-list'});
        } else if (this.clientAddresses.length === 0) {
          this.saveButtonLoader = false;
          this.$router.push({name: 'client-list'});
        }
      })
      .catch((error) => {
        this.errors.push(error.response.data.message);
        this.$store.commit('app/showMessage', ['error', error.response.data.message]);
        this.saveButtonLoader = false;
        this.$store.commit('loader', false);
        clientService.deleteClient(uuid);
```

```
      });
  }

  public saveAddress(uuid: string) {
    clientService.saveAddress(uuid, this.clientAddresses)
      .then(() => {
        if ((this.clientPhones.length === 0) && (this.errors.length === 0)) {
          (this.$refs.form as Vue & { reset: () => boolean }).reset();
          this.$router.push({name: 'client-list'});
        }
      })
      .catch((error) => {
        this.errors.push(error.response.data.message);
      });
  }

  public addClient() {
      clientService.addClient(this.clientForm)
        .then((response) => {
          if (response.status === 201) {
            const uuid = response.data.uuid;
            if (!(this.clientPhones.length === 0 || this.clientAddresses.length ===
0)) {
              this.allSaved = true;
            }
            if (this.clientPhones.length > 0) {
              this.savePhone(uuid);
            }
            if (this.errors.length === 0) {
              if (this.clientAddresses.length > 0) {
                this.saveAddress(uuid);
              }
            }
          }
        })
        .then(() => {
          if (this.clientPhones.length === 0 && this.clientAddresses.length === 0) {
            this.saveButtonLoader = false;
            this.$store.commit('loader', false);
            this.$store.commit('app/showMessage', ['success', 'Успешно сохранено']);
          }
        });

  }

  public updateError() {
    this.errors = [];
    this.saveButtonLoader = false;
    this.$store.commit('loader', false);
  }

  public mounted() {
    this.phones = {} as Phone;
    this.clientPhones.push(this.phones);
    this.addresses = {} as Address;
    this.clientAddresses.push(this.addresses);
```

**Приложение В**
**Акт внедрения**

Утверждаю
Руководитель ИТ-отдела
ТОО «Asia Freight»
Фомин Г. В.
«08» мая 2020г.

**АКТ ВНЕДРЕНИЯ**

Настоящий акт составлен о том, что программный продукт, полученный в результате выпускной работы студента НАО АУЭС имени Гумарбека Даукеева гр. ИС-16-2 очной формы обучения Исмадулла Ш. Н. на тему «*Разработка информационной системы логистической компании с применением современных Web-технологий*» внедрен в ТОО «Asia Freight» и используютсяв компании. Использование результата выпускной работы Исмадулла Ш.Н. обеспечивает доступ к информации о заявках на доставку груза, а также дает возможность водителям компании приминимать заявки в системе.

Руководитель ИТ-отдела
ТОО «Asia Freight»


Фомин Г. В.