Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В.Малафей

"\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021

РАЗРАБОТКА САЙТА ОБЩЕСТВА С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «БЕЛГРУЗ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.795011.401

Председатель цикловой комиссии ( Т.Г. Багласова )

Руководитель проекта ( Д.Ф. [Клименко](https://kbp.by/rasp/timetable/view_beta_kbp/?cat=teacher&id=34) )

Консультант по экономической части ( М.А. Григораш )

Консультант по охране труда ( В.С. Кудласевич )

Учащийся ( В.А. Борщевский )

Рецензент ( )

2021

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ДП Т.795011.401 ПЗ

Разраб.

Борщевский В.А.

Провер.

Клименко Д.Ф.

Т. Контр.

*Багласова Т.Г.*

Н. Контр.

Басалыга Л.В.

Утверд.

Багласова Т.Г.

*Разработка сайта Общества*

*с ограниченной ответственностью «БелГруз»*

Лит.

Листов

КБП

77

У

[Введение 4](#_Toc74136867)

[1 Объектно-ориентированный анализ и проектирование приложения 5](#_Toc74136868)

[1.1 Назначение и цели создания Web-приложения 5](#_Toc74136869)

[1.2 Проектирование модели 6](#_Toc74136870)

[2 Проектирование Web-приложения 9](#_Toc74136871)

[2.1 Требования к Web-приложению 9](#_Toc74136872)

[2.2 Структура Web-приложения 9](#_Toc74136873)

[2.3 Проектирование макета web-приложения 10](#_Toc74136874)

[2.4 Программно-технические средства, необходимые для разработки приложения 10](#_Toc74136875)

[2.5 Защита и сохранность данных 11](#_Toc74136876)

[2.6 Организация и ведение информационной базы 11](#_Toc74136877)

[3 Реализация Web-приложения 15](#_Toc74136878)

[3.1 Разработка административной части приложения 15](#_Toc74136879)

[3.2 Разработка клиентской части приложения 15](#_Toc74136880)

[3.3 Описание используемых функций и процедур 15](#_Toc74136881)

[4 Описание Web-приложения 17](#_Toc74136882)

[4.1 Общие сведения 17](#_Toc74136883)

[4.2 Функциональное назначение 17](#_Toc74136884)

[4.3 Описание разделов web-приложения 17](#_Toc74136885)

[5 Методика испытаний 19](#_Toc74136886)

[5.1 Технические требования 19](#_Toc74136887)

[5.2 Функциональное тестирование 19](#_Toc74136888)

[6 Применение 25](#_Toc74136889)

[6.1 Назначение программы 25](#_Toc74136890)

[6.2 Программно-аппаратное обеспечение сервера и клиента 25](#_Toc74136891)

[6.3 Справочная система 25](#_Toc74136892)

[7 Охрана труда и окружающей среды на предприятии 27](#_Toc74136893)

[7.1 Правовые, нормативные, социально-экологические и организационные вопросы](#_Toc74136894)

[охраны труда 27](#_Toc74136894)

[7.2 Обеспечение светотехнических условий рабочего места 29](#_Toc74136895)

[7.3 Пожарная безопасность 33](#_Toc74136896)

[7.4 Охрана окружающей среды 34](#_Toc74136897)

[8 Экономический раздел 35](#_Toc74136898)

[8.1 Технико-экономическое обоснование разработки программного средства 35](#_Toc74136899)

[8.2 Составление плана по разработке программного средства 35](#_Toc74136900)

[8.3 Определение цены программного средства 35](#_Toc74136901)

[8.4 Экономическая эффективность разработки 40](#_Toc74136902)

[Заключение 41](#_Toc74136903)

[Список информационных источников 42](#_Toc74136904)

[Приложение А](#_Toc74136905) [Текст программы 44](#_Toc74136907)

[Приложение Б](#_Toc74136908) [Формы выходных документов 76](#_Toc74136910)

3 лист – это бланк задания

# Введение

Сайт ООО «БелГруз» – это сайт, позволяющий осуществлять выбор, заказ и оплату транспортно-логических услуг онлайн. Иными словами, это логистическая компания, которая предоставляет свои услуги через интернет.

В интернете появились удобные и недорогие поставщики услуг, и их услугами пользуется все больше и больше людей. Это торговые сайты компаний малого и среднего бизнеса.

На сайте логистической компании обычно представлен подробный каталог услуг с ценами, на основе которого пользователь формирует свой заказ. Заказывая услуги на сайте, вы можете воспользоваться ими в определённый момент времени, а оплачивать непосредственно при предоставлении услуги или также через Интернет.

Преимущества сайта транспортно-логических услуг:

* удобство выбора услуг из каталога;
* возможность оформления заявок;
* круглосуточная работа без перерывов.

Целью разработки проекта на тему «Реализация сайта Общества с ограниченной ответственностью «БелГруз» является создание сайта «Chef», которая будет осуществлять ведение базы данных, содержащей информацию об услугах, машинах, водителях, клиентах и их заявок. Для достижения цели преддипломного проекта нужно решить следующие задачи:

* выполнить объектно-ориентированный анализ и проектирование системы, результатом которой будет модель системы;
* определить вычислительную систему, необходимую для создания сайта;
* по модели выполнить проектирование задачи;
* разработать сайт;
* описать созданный сайт;
* выбрать методику испытаний;
* описать процесс тестирования;
* привести примеры области применения.

Отчет по преддипломной практике состоит из шести разделов, содержащих необходимую информацию по организации эксплуатации сайта.

В первом разделе «Объектно-ориентированный анализ и проектирование приложения» описываются назначения и цели Web-приложения, а также проектирование модели.

Второй раздел «Проектирование Web-приложения» включает в себя описание требований к Web-приложению, проектирование макета Web-приложения, программно-технические средства, необходимые для разработки приложения, описание защиты и сохранности данных, организацию и ведение информационной базы.

Третий раздел «Реализация Web-приложения» содержит разработку клиентской части приложения, описание используемых функций и процедур.

Четвертый раздел «Описание Web-приложения» содержит общие сведения, функциональное назначение и описание разделов сайта.

Пятый раздел «Методика испытаний» содержит технические требования и порядок проведения испытаний: функциональное и полное тестирования.

В шестом разделе «Применение» описывается назначение программы и программно-аппаратное обеспечение сервера и клиента.

В седьмом разделе «Охрана труда и окружающей среды» описываются различные вопросы, касающиеся охраны труда и окружающей среды: правовые, нормативные, социально-экономические и организационные вопросы охраны труда, пожарная безопасность и охрана окружающей среды.

В восьмом разделе «Экономический раздел» описываются экономические затраты, затраты на дипломный проект, решаемые экономические проблемы с помощью данного программного продукта.

В заключении проанализировано созданное программное приложение, определена степень соответствия поставленной задачи и выполненной работы.

Приложение А содержит текст программы.

Приложение Б содержит выходные данные.

Графическая часть содержит диаграмму вариантов использования, диаграмму развёртывания, диаграмму деятельности, диаграмму структуры Web-приложения.

1. Объектно-ориентированный анализ и проектирование приложения
2. Назначение и цели создания Web-приложения

Предметной областью данного преддипломного проекта является сайт транспортно-логических услуг.

Сайт ООО «БелГруз» – сайт, предоставляющий услуги посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на оформление услуг, выбрать способ оплаты и воспользоваться ими в определённый момент времени, оплатить заказ.

В услуги сайта включаются:

* взаимодействия с каталогом услуг;
* взаимодействие с заказом.

Каждая услуга обладает следующими характеристиками:

* конкретное название;
* цена без грузчиков;
* цена с грузчиками.

На сайте будет присутствовать одна должность клиент, администратор.

Клиент – лицо (физическое или юридическое), заинтересованное в выполнении исполнителем работ, оказании им услуг или приобретении у продавца какого-либо продукта (в широком смысле). Иногда при этом предполагается оформление заказа, но не обязательно [4].

Администратор – это человек, который управляет деятельностью заведения в целом. Основными функциями администратора интернет-магазина являются: наполнение сайта, загрузка новых товаров, проверка характеристик и обновление каталога; обслуживание клиентов; отслеживание заказов и доставок; проверка оплаты товаров; добавление новостей на сайт [5].

Каталог – обозначает перечень товаров или услуг с их описанием [6].

Каталог включает в себя: список услуг и их описание.

Заказ – поручение на изготовление чего-либо или оказание услуги. Заказ состоит из услуг, заказанные заказчиком [7].

Каждый заказ обладает следующими характеристиками:

* номер заказа;
* адрес начала;
* адрес завершения;
* вес груза;
* высота груза;
* ФИО заказчика;
* электронная почта заказчика;
* телефон заказчика;
* номер машины;
* тариф;
* предоставляемая услуга;
* машина.

Учет заказов на сайте происходит следующим образом:

* заказ оформляет клиент или администратор;
* заказ добавляется в базу заказов;
* заказ исполняется.

Существует две роли − администратор и клиент.

Администратор приложения может:

* добавлять, удалять, просматривать каталог услуг;
* добавлять, удалять, просматривать каталог машин;
* добавлять, удалять, просматривать каталог тарифов;
* добавлять, удалять, просматривать каталог водителей;
* вести учёт заказов;
* производить поиск по названию в любой сущности.

Клиент приложения может:

* просматривать каталог услуг;
* просматривать информацию об определённой услуге;
* просматривать информацию о компании;
* оформлять заказы, с указанием подробной информации.

На основе изученной предметной области приложение должно выполнять следующие функции:

* ведение единой базы данных;
* производить поиск по названиям услуги;
* формирование выходных текстовых документов, которые будут содержать данные обо всех соответствующих сущностях.

Данное приложение является небольшим аналогом сайта «Pegas-transit», который также занимается организацией грузоперевозок. Но из-за того, что все они выполняются по индивидуальному заказу, провести оценку и сравнение невозможно.

Проектирование модели

При проектировании будет организован поэтапный план в виде основных аспектов реализации: подготовки к проектированию и сбор необходимой информации, установка используемого программного средства, создание и подключение библиотек для упрощения реализации, проектирование базовой модели для первичного тестирования, обновление и дополнение существующих компонентов, многоступенчатое тестирование продукта и его оптимизация.

Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных. Нотация ERD была впервые введена П. Ченном и получила дальнейшее развитие в работах Баркера.

Для данного сайта можно выделить следующие сущности: «Заказ», «Машина», «Водитель», «Услуга», «Тариф». Диаграмма «Сущность-связь» представлена на рисунке 1.1.

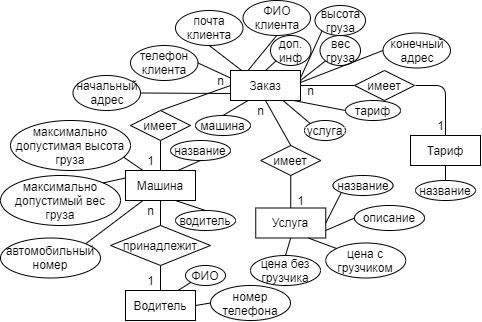


Рисунок 1.1 − Диаграмма «Сущность-связь»

Для сущности «Тариф» атрибутом будет являться: название.

Для сущности «Услуга» атрибутами будут являться:

* название;
* описание;
* цена без грузчика;
* цена с грузчиком.

Для сущности «Водитель» атрибутами будут являться:

* ФИО;
* номер телефона.

Для сущности «Машина» атрибутами будут являться:

* название;
* водитель;
* автомобильный номер;
* максимально допустимый вес груза;
* максимально допустимая высота груза.

Для сущности «Заказ» атрибутами будут являться:

* начальный адрес;
* конечный адрес;
* высота груза;
* вес груза;
* дополнительная информация;
* телефон клиента;
* почта клиента;
* ФИО клиента;
* машина;
* услуга;
* тариф.

Диаграмма вариантов использования в UML − диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Существуют два вида отношений в данной диаграмме. Расширение (англ. Extend) − разновидность отношения зависимости между базовым вариантом использования и его специальным случаем. Включение (англ. Include) − определяет взаимосвязь базового варианта использования с другим вариантом использования, функциональное поведение которого всегда задействуется базовым вариантом использования [8]. На данной диаграмме имеются два «актёр» − администратор и клиент.

Администратор и клиент имеют такие варианты использования как:

* авторизация;
* добавление;
* удаление;
* публикация;
* просмотр записей.

В основном при создании диаграммы рассматривались такие связи как «включение».

Данная диаграмма находится в графической части на листе 1.

Диаграмма деятельности – это технология, позволяющая описывать логику процедур, бизнес-процессы и потоки работ. Ветвления означают, что есть один входной поток и несколько параллельных потоков. Поток представляет собой самый общий вид перехода и задает порядок выполнения операций. Эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Данная диаграмма находится в графической части на листе 2.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации. Диаграмма развертывания завершает процесс объектно-ориентированного анализа и проектирования для конкретной программной системы и ее разработка, как правило, является последним этапом спецификации модели.

Для диаграммы развертывания проектируемой системы представлены все компоненты, существующие на этапе исполнения, а именно:

* база данных PostgreSQL;
* ruby скрипты;
* web-сервер;
* web-браузер;
* html-страницы.

Диаграмма развертывания представлена в графической части на листе 3.

Структура Web-приложения представлена на листе 4 в графической части.

1. Проектирование Web-приложения
2. Требования к Web-приложению

Разрабатываемое проект будет представлять собой сайт, который подразумевает использование по сети Интернет.

Сайт будет находиться на сервере, и доступ к нему будут иметь только администратор. Для каждой группы пользователей предусмотрено разграничение прав доступа к приложению. Администратор сайта может осуществлять все действия доступные на сайте, добавление, изменение и удаление любой информации. Пользователи сайта могут просматривать и осуществлять заказы на своё имя.

Сайт должно быть оформлено в тонах, не раздражающих и не режущих глаза. Ключевым требованием должно быть наличие удобной системы навигации и продуманный дизайн.

Визуальное оформление сайта должно быть спокойным и не отвлекать от работы. Дизайн Web-сайта должен быть современным, но деловым.

Для представления текстовых материалов должен использоваться один из стандартных шрифтов. Размер (кегль) шрифта должен обеспечивать удобство восприятия текста.

Контент сайта будет представлять собой текст, таблицы, изображения и формы.

Сайт должно поддерживать кроссбраузерность, и производить впечатление практичной и удобной информационной системы. Пользовательский интерфейс должен быть доступным и понятным для работы. Пользователь должен легко получать доступ к необходимой информации и не использовать затруднений в процессе навигации по разделам ресурса. Для реализации этих задач необходимо найти универсальное решение, позволяющее разместить элементы Web-страниц в наиболее понятном и удобном для конечного пользователя порядке.

Верстка страниц сайта должна производиться в рамках идеологии разделения структуры и представления. Ruby on Rails − фреймворк, написанный на языке программирования Ruby, реализует архитектурный шаблон Model-View-Controller для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером баз данных.

1. Структура Web-приложения

Сайт представляет собой набор компонентов, таблиц стилей, которые могут объединяться и выполняться в различных браузерах.

Общая структура сайта приведена на рисунке 2.1.



Рисунок 2.1 – Общая структура приложения

Разделы и страницы представляются в виде файлов и каталогов. Необходимо продумать их структуру, способы именования и полное дерево каталогов. Это важный момент, поскольку определяет способы связи между страницами. В приложении используется Model-view-controller -концепция.

Model-view-controller (MVC) − схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

* модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
* представление, вид − отвечает за отображение информации (визуализацию). Часто в качестве представления выступает [форма (окно)](http://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BD%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) с графическими элементами.
* контроллер. Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Помимо изолирования видов от логики приложения, концепция MVC существенно уменьшает сложность больших приложений. Код получается гораздо более структурированным, и, тем самым, облегчается поддержка, тестирование и повторное использование решений.

1. Проектирование макета web-приложения

Дизайн сайта будет представлять собой контейнер, так как большая часть информации будет представлена в виде форм с текстом, и для удобства восприятия был выбран именно этот тип. Также дизайн сайта выполнен блочной вёрсткой, так как она удобна и легка в редактировании.

Макет сайта представляет собой хедер, содержимое для контента, футер куда выносят полезную, но не первостепенную информацию. Макет сайта представлен на рисунке 2.2

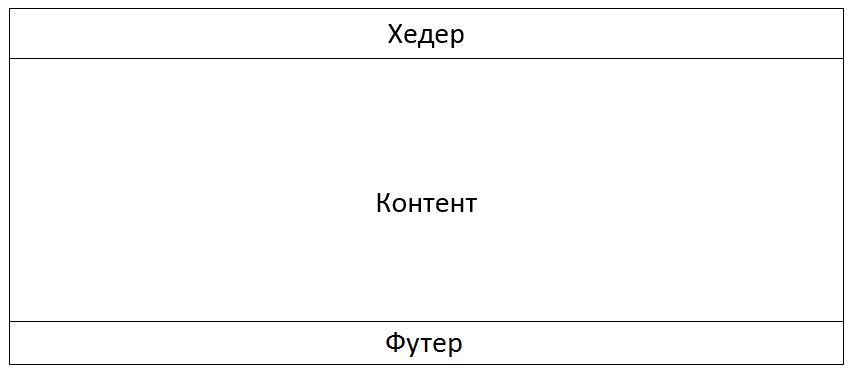


Рисунок 2.2 − Макет сайта

1. Программно-технические средства, необходимые для разработки приложения

Верстка страниц веб-сайта будет производиться с помощью компонентов. Компоненты позволяют разбить интерфейс на независимые части, про которые легко думать в отдельности. Их можно складывать вместе и использовать несколько раз.

Во многом компоненты ведут себя как обычные функции Ruby. Они принимают произвольные входные данные и возвращают элементы, описывающие, что мы хотим увидеть на экране.

При разработке приложения будут использоваться следующие технологии:

Ruby – динамический, рефлективный, интерпретируемый высокоуровневый язык программирования. Язык обладает независимой от операционной системы реализацией многопоточности, сильной динамической типизацией, сборщиком мусора и многими другими возможностями [12].

Ruby on Rails – фреймворк, написанный на языке программирования Ruby, реализует архитектурный шаблон Model-View-Controller для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером баз данных. [13].

CSS (англ. Cascading Style Sheets − каскадные таблицы стилей) − формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL [14].

PostgreSQL − свободная объектно-реляционная система управления базами данных (СУБД).

Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows. [15].

1. Защита и сохранность данных

Защита информации представляет собой деятельность по предотвращению утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию.

Для обеспечения защиты от несанкционированного доступа предусмотрено введение системы паролей (система паролей обеспечивает возможность входа в программе только для тех пользователей, которые знают логин и пароль), введение системы разграничения доступа пользователей (программа обеспечивает загрузку разного рода информации, но возможностью работать с данной информацией смогут только те, у кого есть соответствующий логин и пароль − администратор. В противном случае, пользователь не будет обладать возможностью доступа к данному действию).

Для того что бы войти в личный кабинет администратору необходимо ввести логин и пароль. Для пользователя личного кабинета не существует.

1. Организация и ведение информационной базы

Организация данных подразумевает создание модели данных, главными элементами которой являются сущности и их связи.

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами [10].

Таблица «Amenities» хранит информацию об услугах, структура приведена в таблице 2.1.

Таблица 2.1 – Структура таблицы «Amenities»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер услуги |
| name | VARCHAR | 255 | Название услуги |
| description | VARCHAR | 255 | Описание услуги |
| price | FLOAT | 4 | Цена без грузчика |
| price\_with\_workers | FLOAT | 4 | Цена с грузчиком |
| created\_at | DATETIME | 8 | Дата и время создания |
| updated\_at | DATETIME | 8 | Дата и время обновления |

Таблица «Tariffs» хранит информацию о тарифах доставки, структура приведена в таблице 2.2.

Таблица 2.2 – Структура таблицы «Tariffs»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер тарифа |
| name | VARCHAR | 255 | Название тарифа |
| created\_at | DATETIME | 8 | Дата и время создания |
| updated\_at | DATETIME | 8 | Дата и время обновления |

Таблица «Drivers» хранит информацию о водителях, структура приведена в таблице 2.3.

Таблица 2.3 – Структура таблицы «Drivers»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер водителя |
| name | VARCHAR | 255 | ФИО водителя |

Продолжение таблицы 2.3.

|  |  |  |  |
| --- | --- | --- | --- |
| phone | VARCHAR | 255 | Номер телефона |
| created\_at | DATETIME | 8 | Дата и время создания |
| updated\_at | DATETIME | 8 | Дата и время обновления |

Таблица «Cars» хранит информацию о машинах, структура приведена в таблице 2.4.

Таблица 2.4 – Структура таблицы «Cars»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер машины |
| name | VARCHAR | 255 | Название машины |
| height | FLOAT | 4 | Максимально допустимая высота вмещаемого груза |
| load\_capacity | FLOAT | 4 | Максимально допустимый вес вмещаемого груза |
| number | VARCHAR | 255 | Автомобильный номер |
| driver\_id | BIGINT | 8 | Номер водителя |
| created\_at | DATETIME | 8 | Дата и время создания |
| updated\_at | DATETIME | 8 | Дата и время обновления |

Таблица «Users» хранит информацию о работниках, структура приведена в таблице 2.5.

Таблица 2.5 – Структура таблицы «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер работника |
| name | VARCHAR | 255 | ФИО работника |
| email | VARCHAR | 255 | Электронная почта работника |
| password | VARCHAR | 255 | Пароль работника |
| phone | VARCHAR | 255 | Номер телефона |
| created\_at | DATETIME | 8 | Дата и время создания |
| updated\_at | DATETIME | 8 | Дата и время обновления |

Таблица «Orders» хранит информацию о заказах, структура приведена в таблице 2.6.

Таблица 2.6 – Структура таблицы «Orders»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер заказа |
| startPoint | VARCHAR | 255 | Начальный адрес заказа |
| finishPoint | VARCHAR | 255 | Конечный адрес заказа |
| height | FLOAT | 4 | Высота груза |
| weight | FLOAT | 4 | Вес груза |

Продолжение таблицы 2.6

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| note | TEXT | 255 | Дополнительная информация от клиента |
| customer\_phone | VARCHAR | 255 | Номер телефона клиента |
| customer\_email | VARCHAR | 255 | Электронная почта клиента |
| customer\_name | VARCHAR | 255 | ФИО клиента |
| car\_id | BIGINT | 8 | Номер машины |
| tariff\_id | BIGINT | 8 | Номер тарифа |
| amenity\_id | BIGINT | 8 | Номер услуги |
| created\_at | DATETIME | 8 | Дата и время создания |
| updated\_at | DATETIME | 8 | Дата и время обновления |

Схема базы данных представлена на рисунке 2.3.

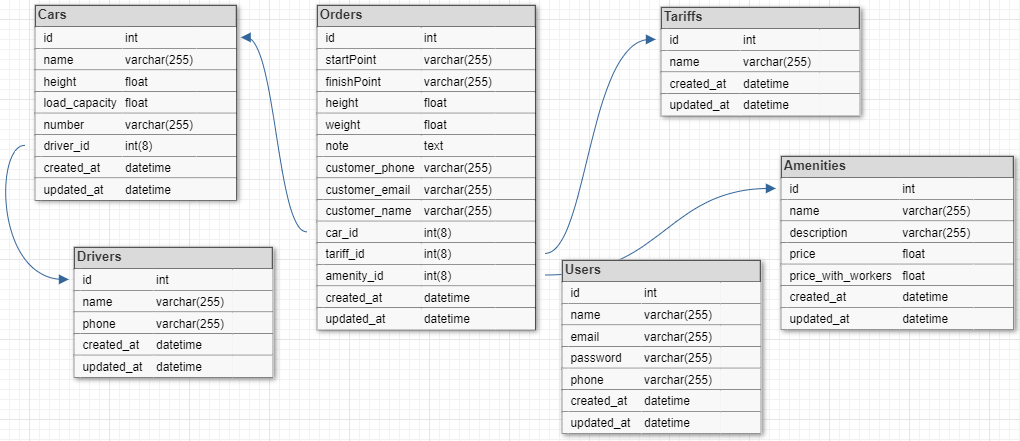


Рисунок 2.3 − Схема базы данных

1. Реализация Web-приложения
2. Разработка административной части приложения

При разработке административной части сайта, его главная и все второстепенные страницы будут находиться в корневом каталоге «Bel\_Gruz/app».

Описание физической структуры административной части сайта:

* «Bel\_Gruz/app/controllers/amenititesController.rb» − контроллер услуг;
* «Bel\_Gruz/app/controllers/usersController.rb» − контроллер клиентов;
* «Bel\_Gruz/app/controllers/driversController.rb» − контроллер водителей;
* «Bel\_Gruz/app/controllers/carsController.rb» − контроллер машин;
* «Bel\_Gruz/app/controllers/ordersController.rb» − контроллер заказов;
* «Bel\_Gruz/app/controllers/tariffsController.rb» − контроллер тарифов.

Описание логической структуры административной части сайта:

«Услуги» («<a href="/admin/amenity">») – относительная ссылка на страницу со списком услуг;

«Машины» («<a href="/admin/car">») – относительная ссылка на страницу со списком машин;

«Водители» («<a href="/admin/driver">») − относительная ссылка на страницу со списком водителей;

«Тарифы» («<a href="/admin/tariff">») – относительная ссылка на страницу со списком тарифов;

«Заказы» («<a href="/admin/order">») – относительная ссылка на страницу со списком оформленных заказов;

«Выход» («<a href="/">») – относительная ссылка на главную страницу сайта.

1. Разработка клиентской части приложения

При разработке клиентской части сайта, его главная и все второстепенные страницы будут находиться в корневом каталоге «Bel\_Gruz/app».

Физическая структура клиенткой части сайта будет состоять из таких же контроллеров, как и административная часть, но за счёт разделения ролей доступ будет уже не ко всем методам данного контроллера.

Описание логической структуры клиентской части сайта:

«О компании» («<a href="/">») – относительная ссылка на страницу, содержащую информацию о компании;

«Услуги» («<a href="/amenity">») – относительная ссылка на страницу со списком предоставляемых услуг;

«Оформить заказ» («<a href="/order">») – относительная ссылка на страницу с формой оформления заказа;

«Вход» («<a href="/">») – относительная ссылка на модель авторизации в административную часть.

1. Описание используемых функций и процедур

Основными функциями программного средства будут являться добавление и удаление данных. Добавление данных в базе производится нажатием по соответственной кнопке и заполнение формы. Удаление из базы производится, также, нажатием на кнопку соответствующего элемента.

Функции, которые будет выполнять сайт − это ведение данных в базе данных, добавление новых данных, удаление и просмотр данных.

Для добавления нового элемента в любую сущность данного приложения, необходимо будет перейти на нужную модель сущности и нажать на кнопку «Добавить». После нажатия на кнопку «Добавить» вызовется обработчик события create…, который обрабатывает заполненные поля формы и отправит Http запрос. Код реализации данной функции представлен ниже.

const createOrder = () => {

console.log(newOrder);

props.post(newOrder, "orders", indexOrder);

};

Для удаления элемента в любую сущность данного приложения, необходимо будет перейти на нужную модель сущности и нажать на кнопку «Удалить». После нажатия на кнопку «Удалить» вызовется обработчик события delete…, который отображает окно соглашения, после соглашения которого отправляет Http запрос. Код реализации данной функции представлен ниже.

const edits = {

onRowDelete: (oldData) => new Promise((resolve) => {

props.drop(oldData.id, 'orders', indexOrder)

resolve();

})

}

Полный текст программы представлен в приложении А.

1. Описание Web-приложения
2. Общие сведения

Программное средство «БелГруз» выполнено в форме сайта и используется как администратором организации, так и клиентами.

На сайте существую разделение прав, с помощью чего реализовано разделений функций и действий для любого пользователя сайта, как для администратора, так и для обычного пользователя-клиента.

Верстка осуществлена согласно разработанному макету и обеспечивает удобство пользователя во время работы с сайтом. В процессе верстки учтено, чтобы все элементы дизайна располагались в определенном и логичном порядке, что позволяет экономить время при работе с приложением.

Для правильного функционирования приложения на сервере соблюдена MVC-концепция.

На разработанном сайте защита приложения производится при помощи логина и пароля, и сверки о наличии в базе данных введенной информации пользователя и определение его роли. Защита доступа к данным разграничивает пользователей на группы: администратор, клиент.

1. Функциональное назначение

Сайт предназначен для автоматизации работы транспортно-логистической компании.

Целью разработки сайта является снижение бумажной работы, уменьшение количества ошибок при работе с заказами.

На сайте защита программного средства производиться при помощи ввода логина и пароля, а также сверки о наличии в базе данных введенной информации пользователя и определение его роли.

Целевая аудитория сайта представлена следующими группами пользователей:

* администратор;
* клиенты.

Пользователи, принадлежащие к группе «Администратор» имеют следующие функции:

* добавлять, удалять, просматривать каталог услуг;
* добавлять, удалять, просматривать каталог машин;
* добавлять, удалять, просматривать каталог тарифов;
* добавлять, удалять, просматривать каталог водителей;
* вести учёт заказов;
* производить поиск по названию в любой сущности.

Пользователи, принадлежащие к группе «Клиенты» имеют следующие функции:

* просматривать каталог услуг;
* просматривать информацию об определённой услуге;
* просматривать информацию о компании;
* оформлять заказы, с указанием подробной информации.

Разработанный сайт поддерживает кроссбраузерность, а значит будет правильно работать и отображать во всех известных браузерах.

Кроссбраузерность − это возможность приложения одинаково отображаться в разных браузерах.

1. Описание разделов Web-приложения

При входе в систему под пользователем, имеющем роль «Администратор», навигационное меню состоит из следующих пунктов: «Услуги», «Машины», «Водители», «Тарифы», «Заказы», «Выход».

Пункт меню «Услуги» обеспечивает переход на страницу для просмотра и работы со всем списком услуг, способность составлять отчёт.

Пункт меню «Машины» обеспечивает переход на страницу для просмотра и работы со списком имеющихся машин в компании, способность составлять отчёт.

Пункт меню «Водители» обеспечивает переход на страницу для просмотра и работы со списком существующих водителей, способность составлять отчёт.

Пункт меню «Тарифы» обеспечивает переход на страницу для просмотра и работы со списком существующих тарифов, способность составлять отчёт.

Пункт меню «Заказы» обеспечивает переход на страницу для просмотра и ведения учёта информации об оформленных заказах, способность составлять отчёт.

Пункт меню «Выход» обеспечивает переход на страницу на главную страницу сайта.

При входе в систему под пользователем, имеющем роль «Клиент», навигационное меню состоит из следующих пунктов: «О компании», «Услуги», «Оформить заказ».

Пункт меню «О компании» обеспечивает переход на страницу на главную страница сайта, содержащую информацию о компании и предоставляемых услугах.

Пункт меню «Услуги» обеспечивает переход на страницу для просмотра всего списка услуг.

Пункт меню «Оформить заказ» обеспечивает переход на страницу для оформления заказа с подробной информацией.

Пункт меню «Вход» обеспечивает переход на страницу на страницу авторизации.

1. Методика испытаний
2. Технические требования

Минимальные системные требования для проведения технических испытаний:

* 400 МБ свободного места на жестком диске;
* 1 ГБ оперативной памяти;
* процессор Intel Core 1,6 Ггц или совместимый аналог;
* операционная система семейства Microsoft Windows 7;
* интернет 2048/512 (Кбит/с).

Минимальная ширина окна браузера, при которой должно обеспечиваться полноценное отображение Web-страниц (без горизонтальной полосы прокрутки), составляет 1024 пикселей.

Компьютер должен работать под управлением операционной системы, начиная с Windows7 и выше. На компьютере должен быть установлен браузер. Браузер – это специальная программа, которая предназначена для просмотра Web-страниц. Рекомендуется использовать браузер Google Chrome.

В настоящее время существует много Web-браузеров, каждый из которых обладает своими преимуществами и недостатками. Одним из самых популярных и используемых на компьютере это браузеры – Internet Explorer, Mozilla Firefox, Opera, Safari, Google Chrome.

1. Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование − это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях [11].

Тестирование всего программного средства происходит по средствам проверки всех функций, представленных в программном средстве. Первый тест будет проведён в окне авторизации программы в роли администратор, которое представлено на рисунке 5.1.

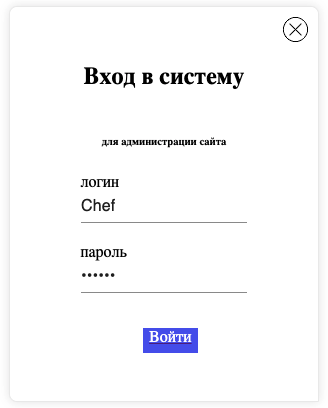


Рисунок 5.1 − Окно авторизации

Таблица 5.1 − Тест-кейс функции авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 1 | Авторизация в приложении | 1. Нажать на кнопку «Вход».  2. Заполнить требуемые поля данными:  - "Логин" − Chef  - "Пароль" – 123456  3. Нажать на кнопку «Войти». | Ожидаемый результат:  Произойдёт переход на страницу Услуги административной части.  Фактический результат:  Все поля заполнены.  Результат: успешный переход на страницу Услуги, представлено на рисунке 5.2 |

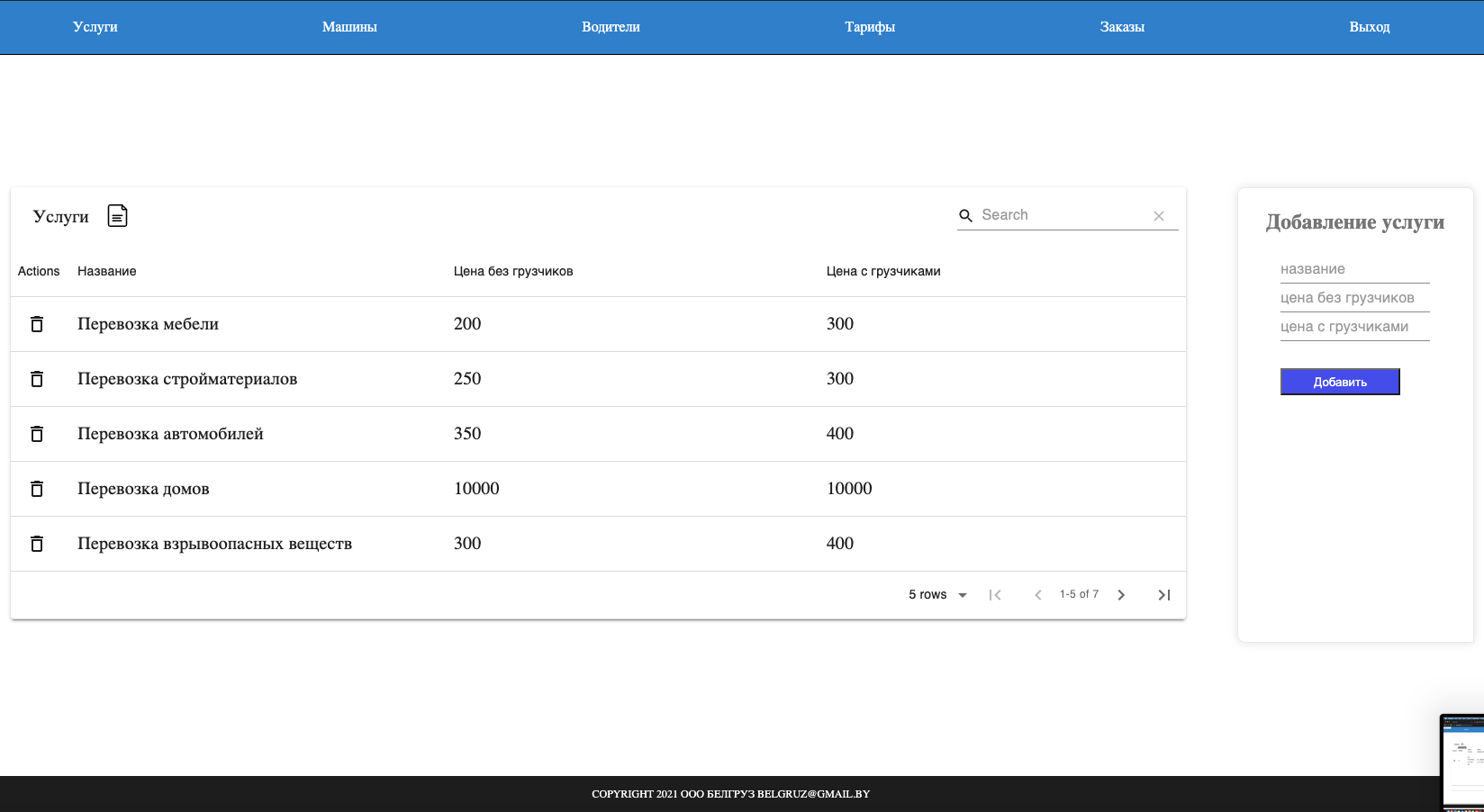


Рисунок 5.2 − Результат авторизации

Второй тест будет проведён в форме «Заказы», представленной на рисунке 5.3.

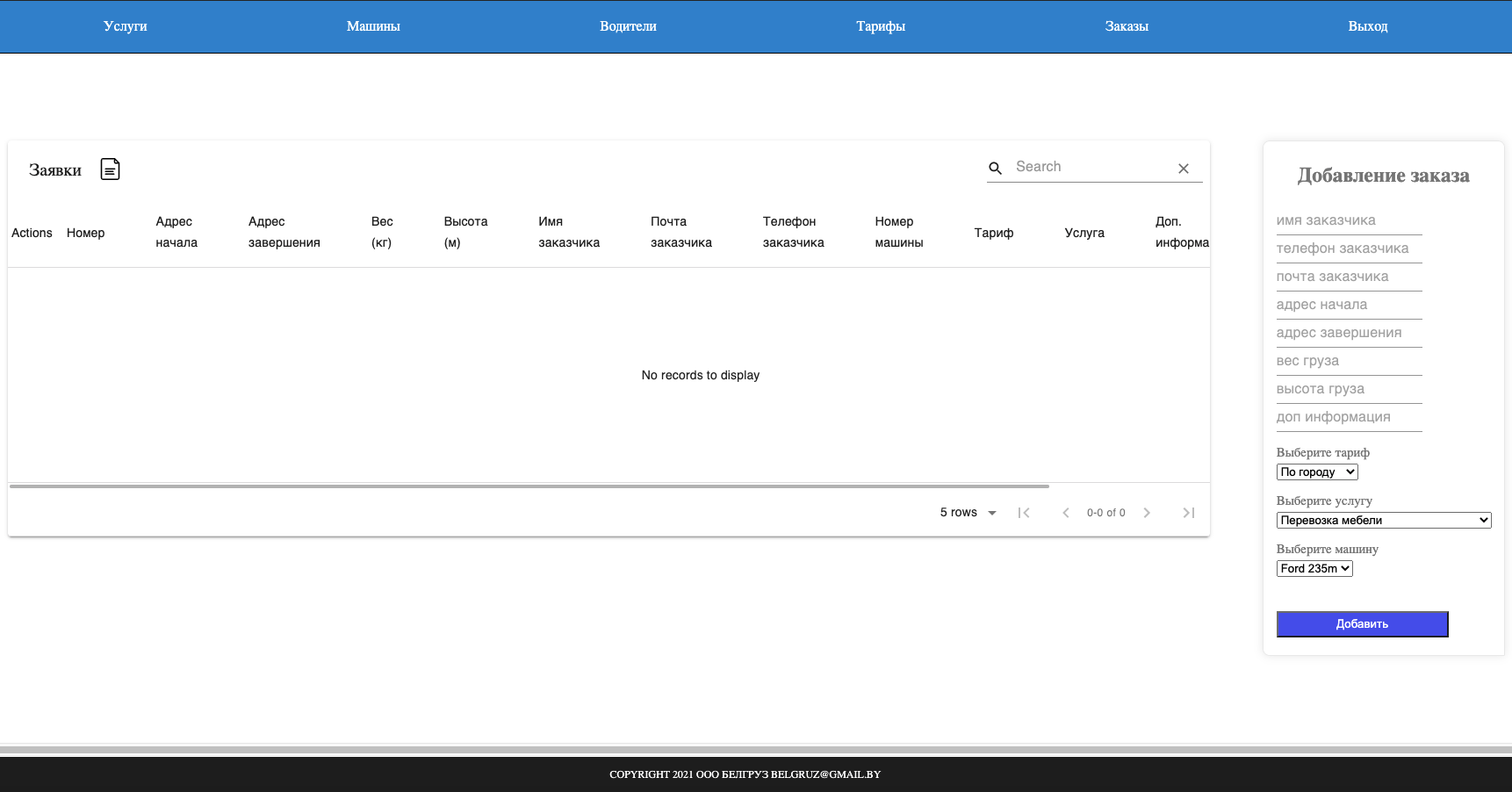


Рисунок 5.3 − Форма «Заказы»

Таблица 5.2 − Тест-кейс функции добавления заказа

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 2 | Добавление заказа в список заказов в приложении | 1. Нажать на кнопку «Добавить». | Ожидаемый результат:  Добавление заказа в базу не произойдёт.  Фактический результат:  Кнопка нажата.  Результат: появление подсветки полей формы, добавление не произошло, представлено на рисунке 5.4. |

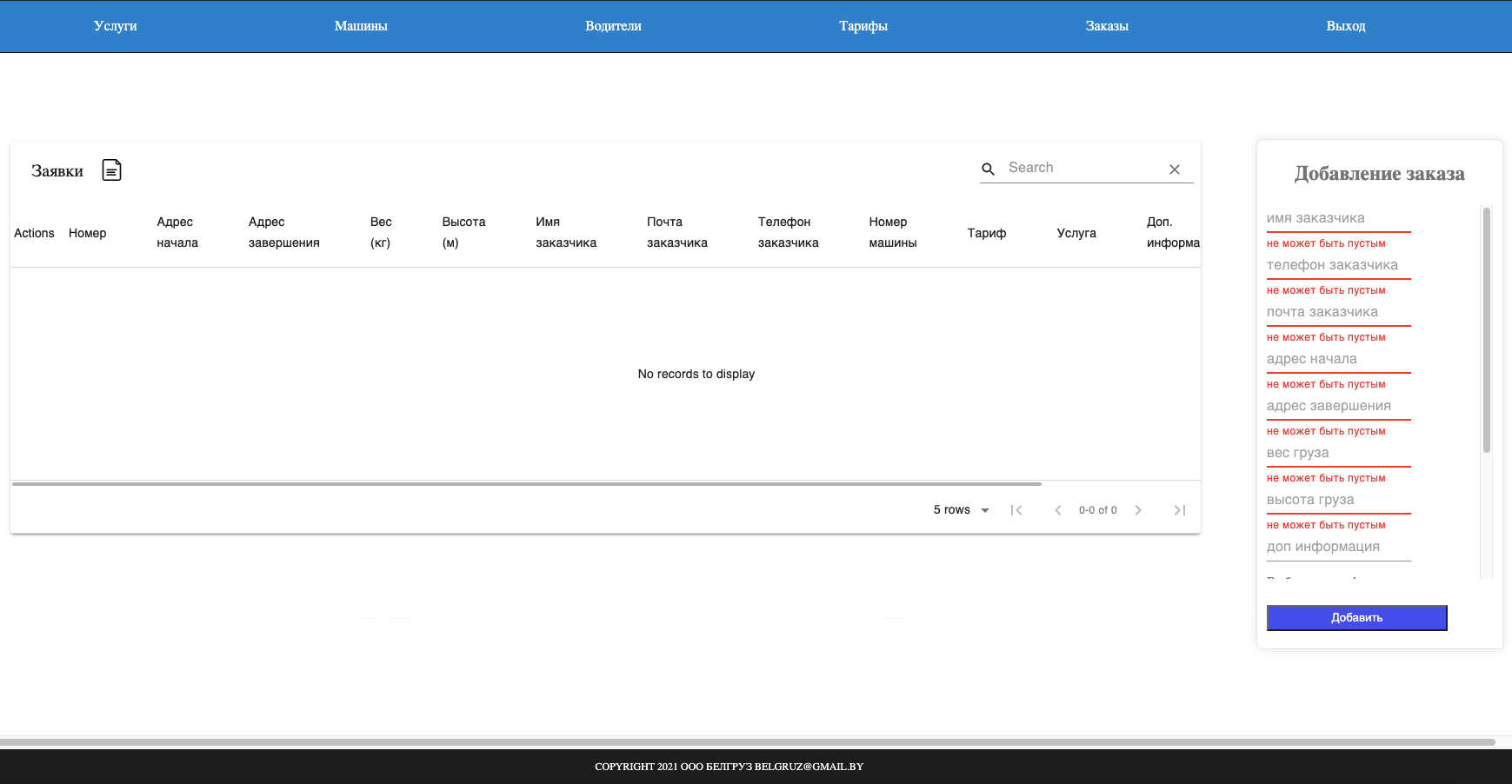


Рисунок 5.4 − Форма «Заказы»

Третий тест будет проведён также в форме «Заказы», представлено на рисунке 5.4.

Таблица 5.3 − Тест-кейс функции добавления заказа

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 3 | Добавление заказа в список заказов в приложении | 1. В открывшемся окне заполнить поля:  − «Имя заказчика» − Иванов Владислав Станиславович  − «Телефон заказчика » − +375443567784  − «Почта заказчика» − ivanov.stanis@gmail.com  − «Адрес начала» − ул. Голубка д.4 кв. 55  − «Адрес завершения» − ул. Звезды д. 17 кв. 164  − «Вес груза» − 389  − «Высота груза» − 2.5  − «Доп. информация»  − «Выберите тариф» − По городу  − «Выберите услугу» − Перевозка мебели  − «Выберите машину» − Ford 235m  2. Нажать на кнопку «Добавить». | Ожидаемый результат:  Произойдёт добавление заказа в базу  Фактический результат:  Все поля заполнены, кнопка нажата.  Результат: появление нового заказа в списке, представлен на рисунке 5.6 |

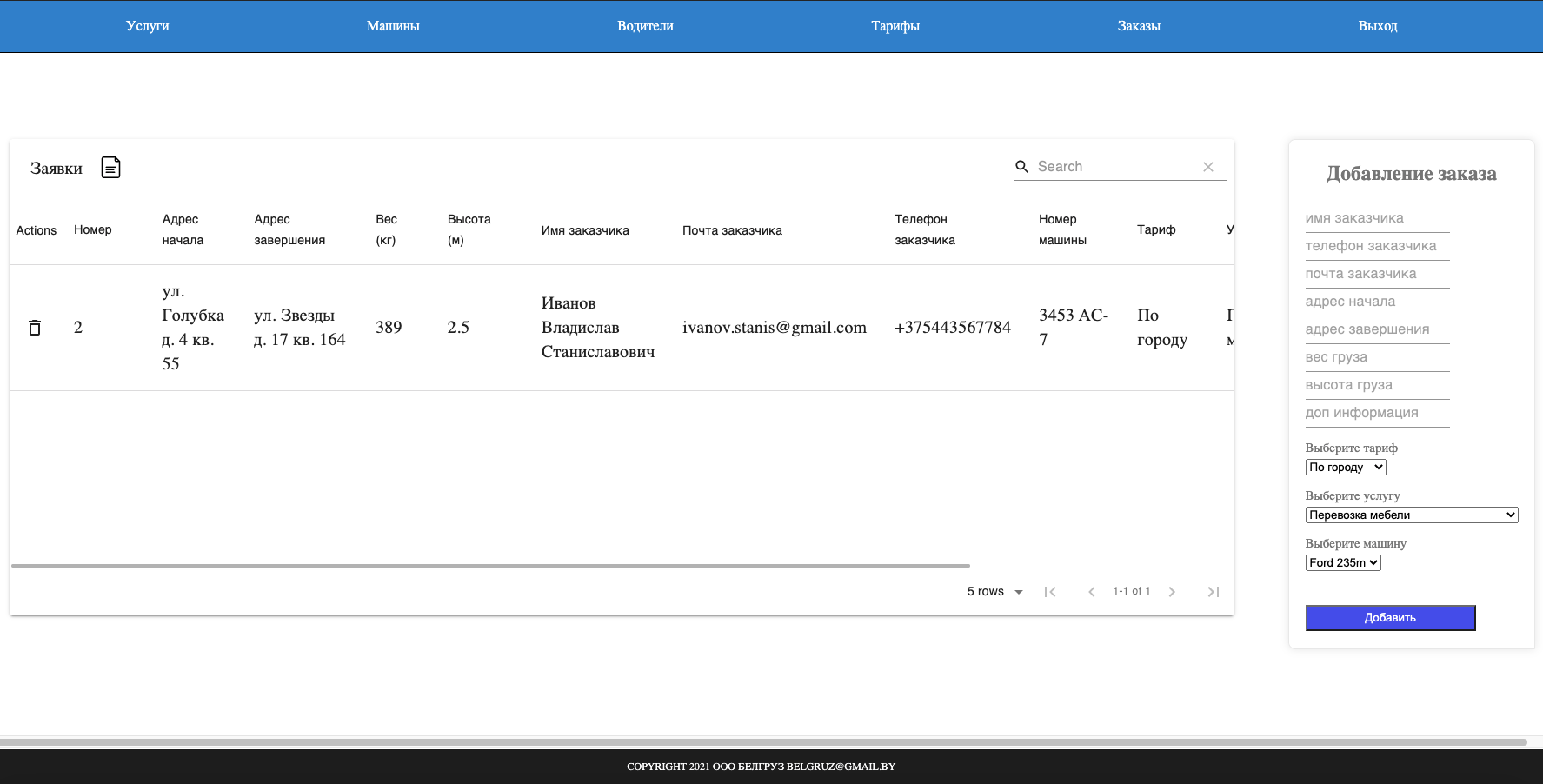


Рисунок 5.6 − Результат функции добавления

Четвёртый тест будет проведён также в форме «Заказы».

Таблица 5.4 − Тест-кейс функции удаления заказа со списка

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 4 | Удаление заказа со списка заказов в приложении | 1. Нажать на пункт панели меню «Заказы»;  2. Нажать на кнопку соответствующего заказа «». | Ожидаемый результат:  Произойдёт удаление заказа со списка и удаление его из базы.  Фактический результат:  Кнопка нажата.  Результат: удаление заказа со списка, представлено на рисунке 5.7. |

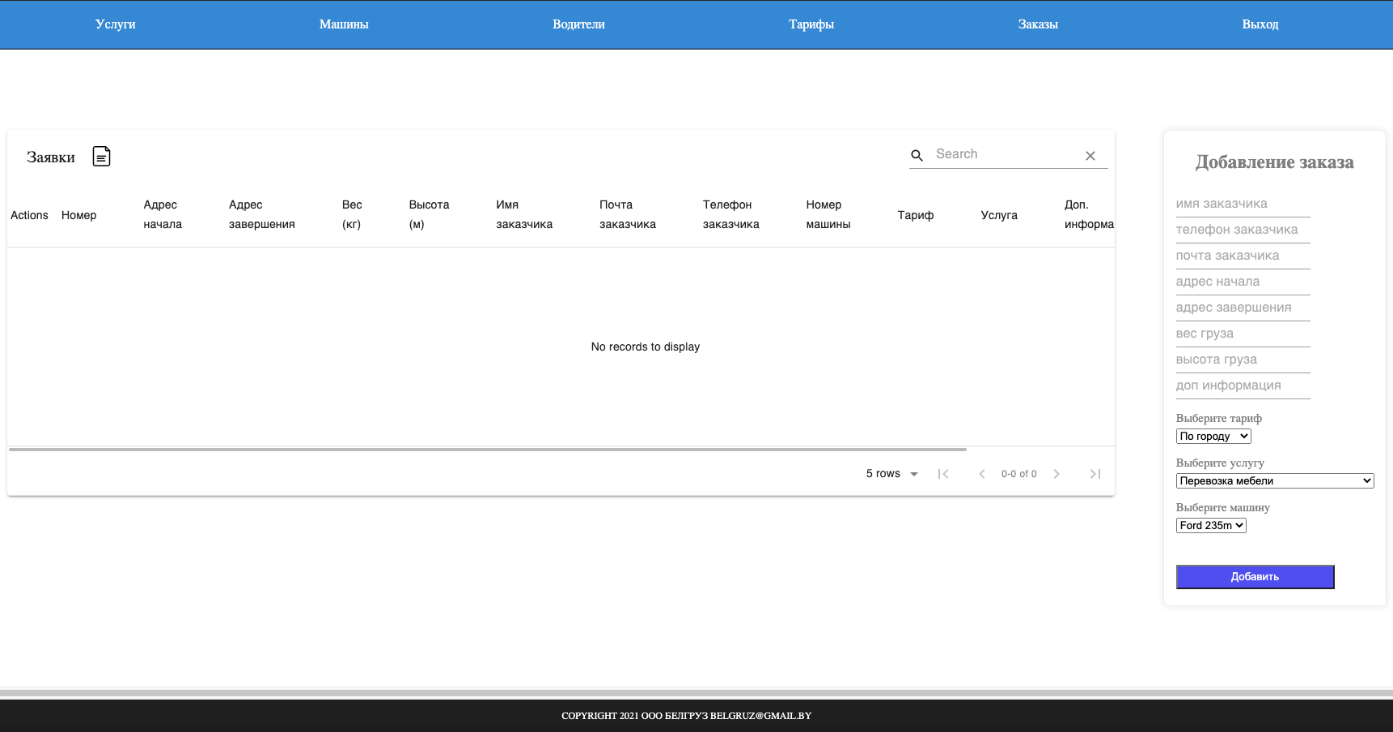


Рисунок 5.7 − Результат удаления заказа

Пятый тест будет проведён в форме «Услуги».

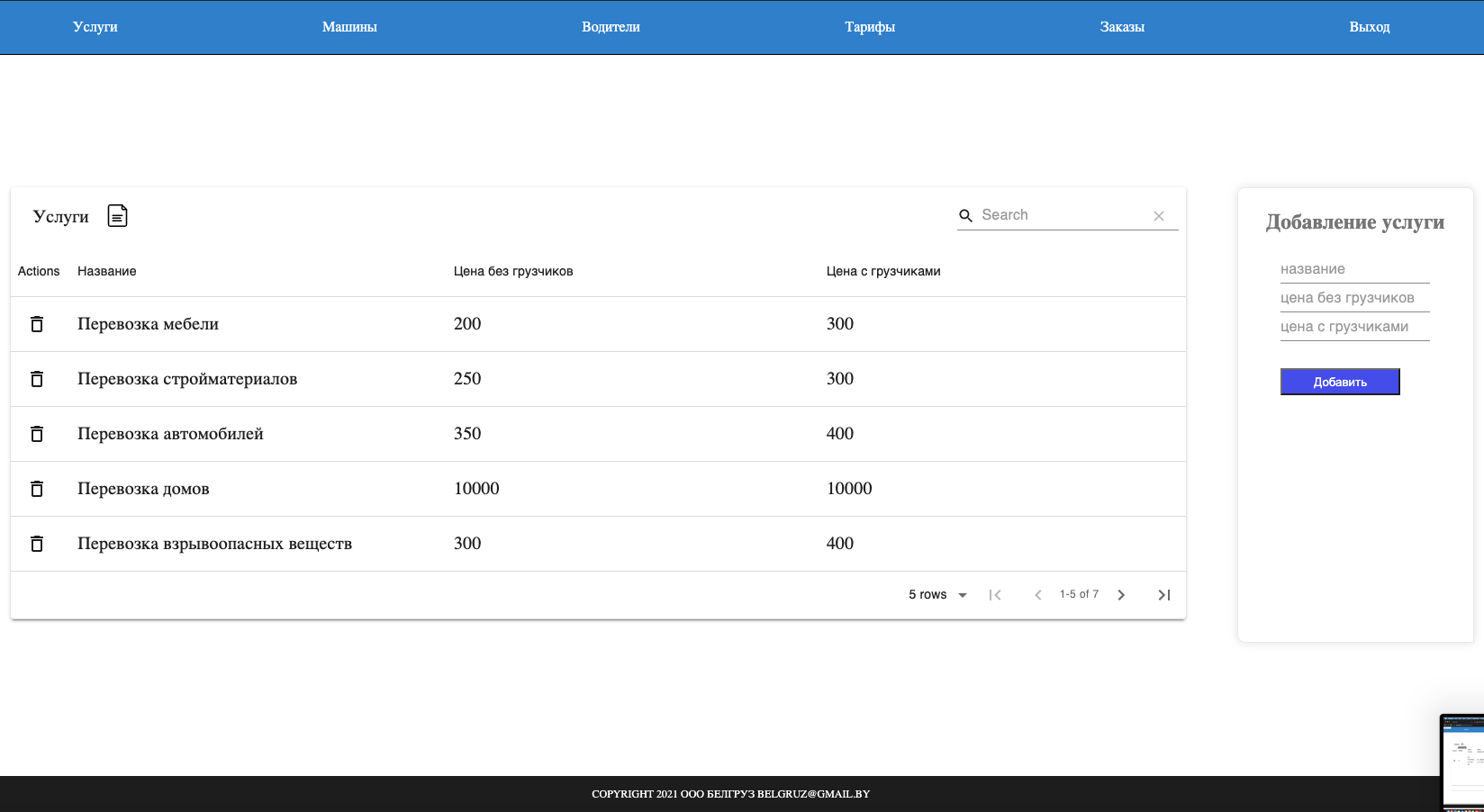


Рисунок 5.8 − Форма «Услуги»

Таблица 5.5 − Тест-кейс функции поиска услуги

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 5 | Поиск услуг по названию в списке услуг в приложении | 1. Нажать на пункт панели меню «Услуги»;  2. Ввести название в строку поиска:  − «Search» − мебели | Ожидаемый результат:  Произойдёт отображение нужных услуг в списке  Фактический результат:  Название введено, представлено на рисунке 5.9  Результат: отображение услуги в списке, представлено на рисунке 5.9 |

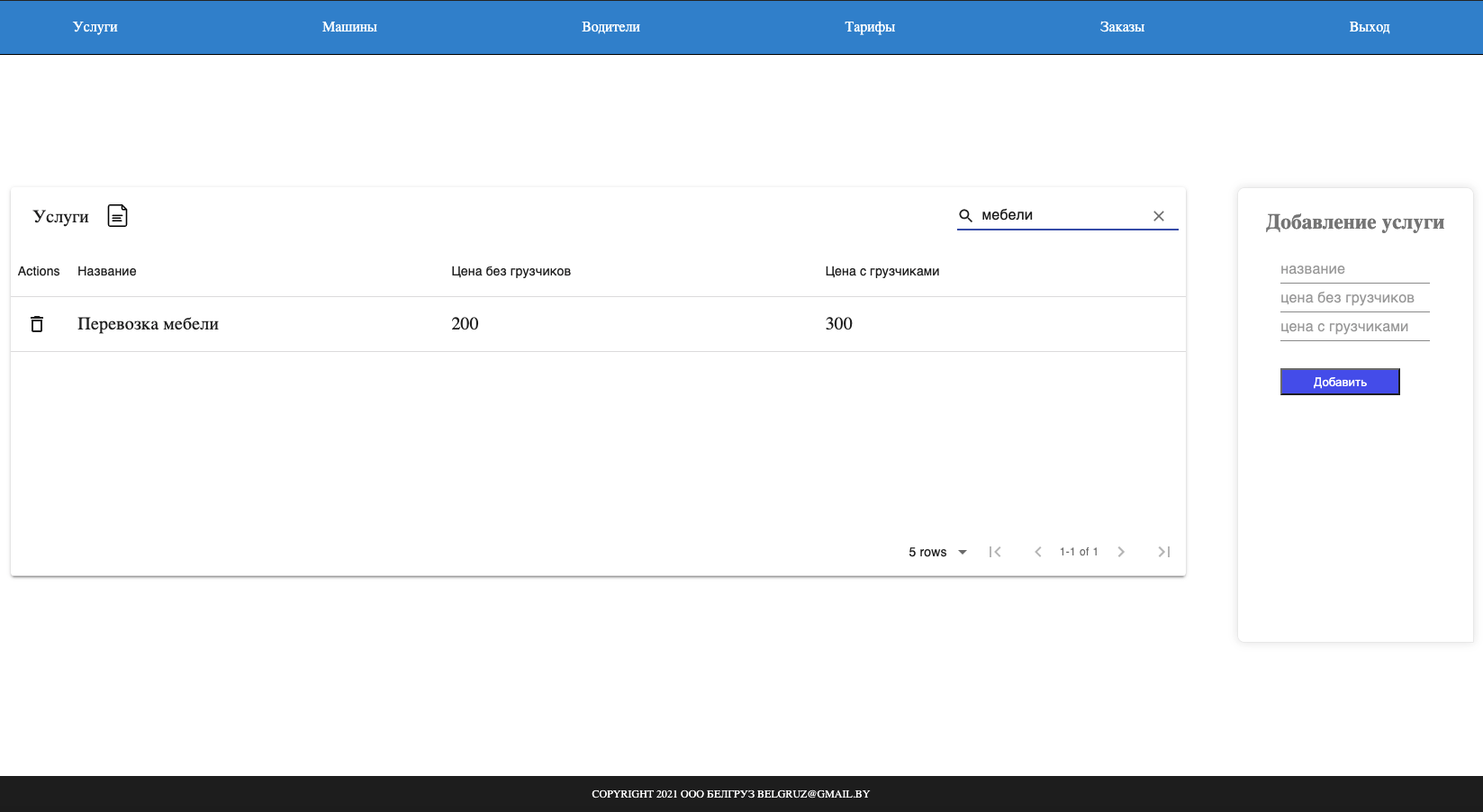


Рисунок 5.9 − Результат поиска услуги

1. Применение
2. Назначение программы

Сайт, реализованный в рамках дипломного проекта на тему «Реализация сайта Общества с ограниченной ответственностью «БелГруз», предназначен для автоматизации работы транспортно-логистической компании.

Разработанный сайт позволил автоматизировать ведение базы данных.

Разработанный сайт имеет интуитивно понятный интерфейс, довольно прост в применении.

Сайт объединяет в себе работу администратора и клиента. Для групп пользователей организованы уровни доступа, в зависимости от которых предоставлены различные возможности на сайте.

Сайт создан с использованием базы данных PostgreSQL. Оно обладает рядом достоинств, повышающих его эффективность и полезность.

1. Программно-аппаратное обеспечение сервера и клиента

Для корректной работы Web-сайта на компьютере пользователя должен быть установлен любой современный браузер.

Хостинг должен соответствовать следующим минимальным требованиям:

* предоставлять не менее 300Мб, дискового пространства;
* поддерживать язык программирования Ruby;
* предоставлять одну PostgreSQL базу данных;
* CPU не менее 100 МГерц;
* выделять не менее 60 Гбайт трафика.

Хостинг должен быстро и оперативно реагировать на любые обращения.

1. Справочная система

Для корректной работы с приложением требуется обеспечить пользователя справочной системой, в которой будут приведены действия работы на сайте. Существует страница справочной системы, на которой описаны основные цели данного сайта, а также список основных предоставляемых услуг компанией данного сайта. Данная справочная система была разработана средствами фреймворка – Ruby on Rails. Страница «О компании» представлена на рисунке 6.1.

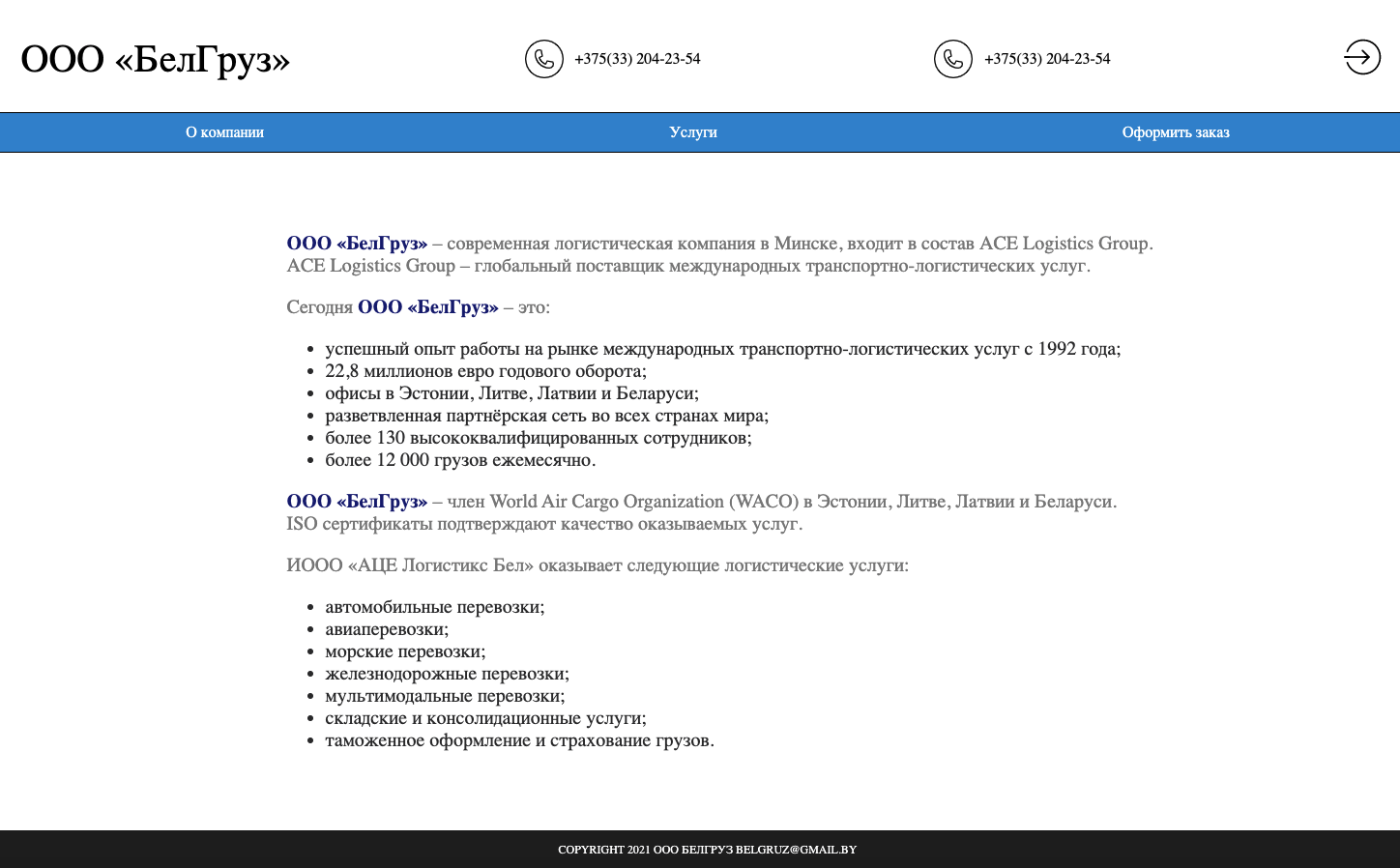


Рисунок 6.1 − Справочная система

# Охрана труда и окружающей среды на предприятии

## Правовые, нормативные, социально-экологические и организационные вопросы охраны труда

Деятельность по охране труда (ОТ) в ИП «Прокопович Н.В.» построена таким образом, что на всех уровнях руководства обеспечивается ответственность за выполнение требований системы управления охраной труда (СУОТ), а именно: разработка, документальное оформление, внедрение, поддержание в рабочем состоянии СУОТ, а также определение механизмов этих требований. На предприятии поддерживается сознательность и чувство собственной ответственности работающих.

Общее руководство работой по обеспечению здоровых и безопасных условий труда и обеспечение функционирования СУОТ в ИП «Прокопович Н.В.» возлагается на председателя товарищества.

Представителем высшего руководства по СУОТ приказом председателя товарищества назначается инженер по ОТ. Он несёт ответственность за разработку, внедрение, реализацию, поддержание и совершенствование СУОТ.

Основными задачами СУОТ являются:

* организация работы по охране труда;
* внедрение передового опыта и научных разработок по безопасности и гигиене труда;
* информирование и консультирование работников банка по вопросам охраны труда;
* осуществление контроля по охране труда;
* выполнение локальных нормативных правовых актов по вопросам охраны труда.

Начальник подразделения является ответственным за обеспечение здоровых и безопасных условий труда и обеспечение функционирования СУОТ в подразделении.

Распоряжением начальника подразделения назначается уполномоченный по СУОТ данного подразделения. Уполномоченный по СУОТ несёт ответственность за поддержание функционирования СУОТ в подразделении.

Каждый работник организации несёт ответственность за соблюдение требований по ОТ, установленных действующими нормативными правовыми актами (НПА) и техническими нормативными правовыми актами (ТНПА).

Непосредственное руководство организацией работы по охране труда в организации осуществляет заместитель председателя товарищества, осуществляющий руководства и контроль за деятельностью управления делами. Заместители председателя товарищества, руководители самостоятельных структурных подразделений осуществляют управление охраной труда по направлениям, относящимся к компетенции подчиненных подразделений, и обеспечивают соблюдение требований законодательства о труде и охране труда в непосредственно подчиненных им подразделениях. Управление охраной труда в областных дирекциях осуществляют их руководители. Организационное, методическое, информационное сопровождение деятельности СУОТ и контроль за её функционированием обеспечивают начальник управления по работе с персоналом и главный специалист по охране труда. Проверку знаний работающих по вопросам охраны труда осуществляет инженер по ОТ для проверки знаний работающих по вопросам охраны труда.

Функциями инженера по ОТ являются:

* анализ состояния условий труда, причин нарушений законодательства о труде и охране труда, производственного травматизма, профессиональной заболеваемости;
* разработка локальных нормативных правовых актов ИП «Прокопович Н.В.» по вопросам охраны труда;
* планирование и организация выполнения мероприятий по охране труда;
* привлечение работающих, профсоюзов к участию в формировании и реализации политики товарищества по улучшению условий и охраны труда;
* обеспечение безопасности при эксплуатации территорий, зданий, сооружений, электроустановок, транспортных средств, оборудования, приспособлений и инструмента;
* обеспечение условий труда на каждом рабочем месте, соответствующих требованиям техники безопасности и производственной санитарии;
* обеспечение установленного законодательством режима труда и отдыха работающих;
* обучение, стажировка, инструктаж и проверка знаний, работающих по вопросам охраны труда;
* проведение обязательных предварительных и периодических медицинских осмотров работающих;
* расследование и учет несчастных случаев на производстве, профессиональных заболеваний, аварий. Разработка и реализация мер по их профилактике;
* нормативное и информационное обеспечение работающих по вопросам охраны труда. Пропаганда знаний и передового опыта по охране труда;
* рассмотрение письменных и устных обращений работающих по вопросам условий и охраны труда;
* применение дисциплинарных и других мер воздействия к нарушителям требований по охране труда;
* подготовка и представление государственной статистической отчетности по охране и условиям труда.

Обучение охране труда производится в соответствии с Инструкцией о порядке подготовки, переподготовки, стажировки, инструктажа, повышения квалификации и проверки знаний, работающих по вопросам охраны труда, утверждённой постановлением Министерства труда и социальной защиты Республики Беларусь от 06.03.2018 № 28.

Для того чтобы требования охраны труда соблюдались всеми работниками, на инженера охраны труда товарищества возложена обязанность обучения работников вопросам охраны труда.

Обучение руководителей и специалистов осуществляется по учебным планам и программам, составленным на основании типового перечня вопросов для обучения и проверки знаний по вопросам охраны труда руководителей и специалистов в соответствующих учреждениях

Не позднее месяца со дня назначения на должность и периодически в соответствии с требованиями нормативных правовых актов, но не реже одного раза в три года, руководители и специалисты проходят проверку знаний по вопросам охраны труда в соответствующих комиссиях для проверки знаний по вопросам охраны труда.

По характеру и времени проведения предусматриваются следующие виды инструктажа:

* вводный инструктаж;
* первичный инструктаж на рабочем месте;
* повторный инструктаж;
* внеплановый инструктаж;
* целевой инструктаж.

Расследование несчастных случаев и профессиональных заболеваний на производстве осуществляется в соответствии с Правилами расследования и учета несчастных случаев на производстве и профессиональных заболеваний, утвержденными Постановлением Совета Министров Республики Беларусь от 10.01.2004 г. №30.

Целями расследования являются:

* установление причин, вызвавших несчастный случай;
* установление виновного в несчастном случае;
* разработка плана мероприятий, направленных на предотвращение подобных несчастных случаев.

Контроль осуществляется в форме проверок, обследований, осмотров, систематического учёта показателей, характеризующих состояние условий и охраны труда, затребования необходимой информации, рассмотрение обращений работников.

Основными видами контроля являются:

* контроль за соблюдением законодательства об охране труда, осуществляемый руководителями и специалистами организации в соответствии с их должностными обязанностями;
* контроль по охране труда, осуществляемый инженером по охране труда организации в соответствии с типовым положением о службе охраны труда организации;
* периодический контроль за соблюдением законодательства об охране труда, осуществляемый представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда;
* общественный контроль за соблюдением законодательства об охране труда, осуществляемый профсоюзами.

Периодический контроль функционирования СУОТ проводится нескольких видов, а именно:

* ежедневный – осуществляется в структурных подразделениях их руководителями и инженером по ОТ;
* ежемесячный – осуществляется в структурных подразделениях начальником отдела с участием общественного инспектора профсоюза по охране труда и инженером по ОТ;
* ежеквартальный – осуществляется в организации председателем товарищества с участием начальников отделов, общественного инспектора профсоюза по охране труда и инженером по ОТ.

Периодический контроль, осуществляется председателем с участием общественных инспекторов профсоюзов по охране труда (уполномоченных лиц по охране труда работников) проводится ежеквартально, так как в организации присутствуют начальники отделов, инженер по охране труда.

## Обеспечение светотехнических условий рабочего места

Практически всю информацию из внешнего мира человек получает с помощью зрения. Поэтому роль света и цвета для человеческой деятельности огромна. Восприятие света является важнейшим элементом нашей способности действовать, поскольку позволяет оценивать местонахождение, форму и цвет окружающих нас предметов.

Всякий светящийся предмет излучает энергию, которая в форме электромагнитных волн распространяется в разные стороны. Для оценки зрительного восприятия потока световой энергии используются понятия:

* световой поток, сила света, яркость, освещенность;
* световым потоком называют поток световой энергии, оцененный по его воздействию на глаз человека;
* силой света называют пространственную плотность светового потока, т.е. отношение светового потока точечного источника света к величине телесного угла, в котором этот поток распространяется;
* яркостью (или фотометрической яркостью) называют силу света в определенном направлении (в глаз наблюдателя), отнесенную к единице площади видимой светящейся поверхности, расположенной перпендикулярно к направлению распространения света;
* освещенностью называют поверхностную плотность светового потока, т.е. световой поток, отнесенный к единице площади освещаемой поверхности;
* контрастом называют разницу яркостей объекта наблюдения и его окружения (фона) или между различными частями объекта.

К функциям зрения, особенно необходимым для безопасности и результативности труда, относятся:

* контрастная чувствительность, острота зрения, быстрота различения деталей, устойчивость ясного видения, цветовая чувствительность.
* контрастная (различительная) чувствительность – это способность глаза различать минимальные значения разности яркости объекта (детали) и фона. Установлена зависимость контрастной чувствительности от условий освещения объекта и яркости, к которой глаз предельно адаптировался.
* острота зрения – это максимальная способность различать отдельные объекты. Большое влияние на остроту зрения оказывает освещенность. С ростом освещенности до определенного уровня растет и острота зрения.

Определенная роль при выполнении зрительной работы принадлежит такой зрительной функции, как цветоощущение. Значение этой функции возрастает при выполнении производственных операций, связанных с необходимостью цветоразличения. Наиболее благоприятные условия цветоощущения создаются при естественном (солнечном) освещении (поскольку оно достаточно велико), а также при искусственном освещении люминесцентными лампами с исправленной цветностью.

Для успешной зрительной работы в условиях изменчивости освещенности большое значение имеет так называемая зрительная адаптация, т.е. приспособление глаза к условиям освещения. Благодаря процессу адаптации зрительный анализатор обладает способностью работать в широком диапазоне освещенностей. Различают световую и темновую адаптации.

Световая адаптация – приспособление глаза к работе в условиях высокой яркости поля зрения. Световая адаптация при повышении яркостей в поле зрения происходит быстро – в течение 5–10 мин.

Темновая адаптация – приспособление глаза к более низким яркостям поля зрения. Темновая адаптация развивается медленнее – от 30 мин до 2 ч.

Процесс адаптации сопровождается фотохимическими и нервными процессами, перестройкой рецептивных полей в сетчатке глаза, изменением диаметра зрачка (зрачковый рефлекс). Частые изменения уровней яркости приводят к снижению зрительных функций, развитию утомления вследствие переадаптации глаза. Зрительное утомление, связанное с напряженной работой и частой переадаптацией, приводит к снижению зрительной и общей работоспособности.

Каждый вид деятельности, связанный с необходимостью различения того или иного объекта, требует определенного уровня освещенности на том участке, где эта деятельность осуществляется. Обычно чем сильнее затруднено зрительное восприятие, тем выше должен быть средний уровень освещенности.

Вместе с тем чрезмерная локальная яркость может вызывать ослепление. Когда в поле зрения попадает яркий источник света, глаз на какое-то время теряет способность различать предметы. Ослепление может быть прямым, когда оно вызвано нахождением ярких источников света в поле зрения, или отраженным, когда свет отражается от поверхностей с высоким коэффициентом отражения.

Человеческий глаз защищается от поражения слишком ярким светом с помощью мигательного рефлекса, поворота глаз и движения головы при воздействии яркого света.

При организации рационального освещения, выборе источников света и светильников учитываются назначение помещения, его размеры и ка­тегория по взрыво- пожароопасности, возможные загрязнения (пыль, газы, пары), характеристика и разряд выполняемой работы, нормированная освещенность и цветовая отделка.

Для создания нормальной световой среды применяют различные системы освещения.

Виды освещения производственных помещений и рабочих мест

Освещение как свет от какого-либо источника, создающее осве­щенность поверхностей предметов и обеспечивающее зрительное восприятие этих предметов, бывает:

* естественное;
* искусственное;
* совмещенное.

Естественное освещение – освещение помещений светом, исходящим от неба (прямым или отраженным), проникающим через световые проемы в наружных ограждающих конструкциях. Подразделяется на боковое, верхнее и комбинированное. Нормируемой характеристикой является коэффициент естественной освещенности.

* боковое естественное освещение – естественное освещение помещения через световые проемы в наружных стенах.
* верхнее естественное освещение – естественное освещение помещения через фонари, световые проемы в стенах (в местах перепада высот здания).
* комбинированное естественное освещение – сочетание верхнего и бокового естественного освещения.

Искусственное освещение – освещение помещений и других мест, где недостаточно естественного освещения. Подразделяется на:

* рабочее;
* аварийное;
* охранное;
* дежурное;
* общее;
* местное;
* комбинированное.

При необходимости часть светильников рабочего или аварийного освещения используется для дежурного освещения.

 Рабочее освещение обеспечивают во всех помещениях, а также на участках открытых пространств, предназначенных для работы, прохода людей и движения транспорта. Для помещений, имеющих зоны с разными условиями естественного освещения и с разными режимами работы, предусматривается раздельное управление рабочим освещением.

 Аварийное освещение – освещение объектов различного назначения, не прекращающееся или автоматически вводимое в действие при внезапном отключении рабочих (основных) источников света. Предназначено для обеспечения эвакуации людей или временного продолжения работы на объектах, где внезапное отключение освещения создает опасность травматизма или недопустимого нарушения технологического процесса. Аварийное освещение подразделяется на: освещение безопасности и эвакуационное освещение.

Освещение безопасности – освещение, предусматриваемое на случай аварийного отключения рабочего освещения, в результате чего возможны: длительное нарушение технологического процесса; нарушение работы таких объектов, как электрические станции, узлы радио и телевизионных передач и связи, диспетчерские пункты, насосные установки водоснабжения, канализации и теплофикации, установки вентиляции и кондиционирования воздуха в производственных помещениях, где недопустимо прекращение работ, и пр.

Эвакуационное освещение – освещение для эвакуации людей из помещений при аварийном отключении нормального освещения. Такое освещение (в помещениях или в местах производства работ вне зданий) следует предусматривать:

* в местах, опасных для прохода людей;
* в проходах и на лестницах, служащих для эвакуации людей, при числе эвакуирующихся более 50 человек;
* по основным проходам производственных помещений, в которых работают более 50 человек;
* на лестничных клетках жилых зданий высотой 6 этажей и более;
* в производственных помещениях с постоянно работающими в них людьми, где выход людей из помещения при аварийном отключении нормального освещения связан с опасностью травматизма из-за продолжения работы производственного оборудования;
* в помещениях общественных и вспомогательных зданий промышленных предприятий, если в помещениях могут одновременно находиться более 100 человек;
* в производственных помещениях без естественного света.

Охранное освещение (при отсутствии специальных технических средств охраны) предусматривается вдоль границ территорий, охраняемых в ночное время. Могут использоваться любые источники света, за исключением случаев, когда охранное освещение автоматически включается только при срабатывании охранной сигнализации или других технических средств. В таких случаях применяются лампы накаливания.

Дежурное освещение – освещение в нерабочее время. Область применения, величины освещенности, равномерность и требования к качеству не нормируются.

Общее освещение – освещение, при котором светильники размещаются в верхней зоне помещения равномерно (общее равномерное освещение) или применительно к расположению оборудования (общее локализованное освещение).

Местное освещение – освещение, дополнительное к общему, создаваемое светильниками, концентрирующими световой поток непосредственно на рабочих местах.

Комбинированное освещение – освещение, при котором к общему освещению добавляется местное.

Совмещенное освещение – освещение, при котором недостаточное по нормам естественное освещение дополняется искусственным.

Источниками искусственного освещения являются газоразрядные лампы и лампы накаливания. Газоразрядные лампы предпочтительнее для применения в системах искусственного освещения. Световой поток от газоразрядных ламп по спектральному составу близок к естественному освещению и поэтому более благоприятен для зрения.

Однако газоразрядные лампы имеют существенные недостатки, к числу которых относится пульсация светового потока. При рассмотрении быстро движущихся или вращающихся деталей в пульсирующем световом потоке возникает стробоскопический эффект, который проявляется в искажении зрительного восприятия объектов (вместо одного предмета видны изображения нескольких, искажаются направление и скорость движения).

В системах производственного освещения применяют люминесцентные газоразрядные лампы, имеющие форму цилиндрической стеклянной трубки. Внутренняя поверхность трубки покрыта тонким слоем люминофора, который преобразует ультрафиолетовое излучение газового электрического разряда в видимый свет. Люминесцентные газоразрядные лампы в зависимости от применяемого в них люминофора создают различный спектральный состав света. Различают несколько типов ламп: дневного света (ЛД), дневного света с улучшенной цветопередачей (ЛДЦ), холодного белого (ЛХБ), теплого белого (ЛТБ) и белого света (ЛБ).

Кроме люминесцентных газоразрядных ламп (низкого давления), в производственном освещении применяют газоразрядные лампы высокого давления:

* лампы ДРЛ (дуговые ртутные люминесцентные);
* галогенные лампы ДРИ (дуговые ртутные с йодидами);
* ксеноновые лампы ЛКсТ (дуговые ксеноновые трубчатые), которые в основном применяются для освещения территорий предприятия;
* натриевые лампы ДНаТ (дуговые натриевые трубчатые), используемые для освещения цехов с большой высотой (в частности, многих литейных цехов).

Применяются для освещения производственных помещений также лампы накаливания, в которых свечение возникает путем нагревания нити накала до высоких температур. Они просты и надежны в эксплуатации. Недостатками их являются низкая световая отдача (не более 20　лм/Вт), ограниченный срок службы (до 1000 ч), преобладание излучения в желто-красной части спектра, что искажает цветовое восприятие.

В осветительных системах используют лампы накаливания различных типов:

* вакуумные (НВ);
* газонаполненные биспиральные (НБ);
* биспиральные с криптоноксеноновым наполнением (НБК);
* зеркальные с диффузно отражающим слоем и др.

Все большее распространение получают лампы накаливания с йодным циклом – галоидные лампы, которые имеют лучший спектральный состав света и хорошие экономические характеристики.

Качественные показатели освещения в производственных помещениях во многом определяются правильным выбором светильников, представляющих собой совокупность источника света и осветительной арматуры. Основное назначение светильников заключается в перераспределении светового потока источников света в требуемых для освещения направлениях, механическом креплении источников света и подводе к ним электроэнергии, а также защите ламп, оптических и электрических элементов от воздействия окружающей среды.

## Пожарная безопасность

По классификации ПУЭ помещения ИП «Прокопович Н.В.» относятся к категории пожароопасные В2-В4, т.к. в них обращаются твердые горючие вещества, способные при взаимодействии с кислородом только гореть. Класс пожароопасных зон П-I., т.к. обращаются ГЖ с температурой вспышки выше 61°С.

Здание ИП «Прокопович Н.В.» согласно паспорту и конструктивным характеристикам, по степени огнестойкости относится к категории IV:

* несущие элементы здания – R 60-К0 – предел огнестойкости 60 минут – по потере несущей способности; по пожарной опасности, не пожароопасные;
* самонесущие стены – RE 45-К0 – предел огнестойкости 45 минут – по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее; по пожарной опасности, не пожароопасные;
* наружные ненесущие стены – Е30-К1 – предел огнестойкости 30 минут – по потере теплоизолирующей способности; по пожарной опасности, небольшая пожароопасность;
* перекрытия междуэтажные – REI 45-К0 – предел огнестойкости 45 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее; по пожарной опасности, не пожароопасные;
* настилы в том числе с утеплителями – RE 15-К1 – предел огнестойкости 15 минут – по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее; по пожарной опасности, мало пожароопасные;
* фермы, балки, прогоны – R 15-К1 – предел огнестойкости 15 минут – по потере несущей способности; по пожарной опасности, мало пожароопасные;
* внутренние стены – REI 90-К0 – предел огнестойкости 90 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее; по пожарной опасности, не пожароопасные;
* марши и площадки лестниц – R 45-К0 – предел огнестойкости 45 минут – по потере несущей способности; по пожарной опасности, не пожароопасные.

Средством сигнализации является извещатель пожарный СО-3. Предназначен для обнаружения возгорания, сопровождающегося появлением дыма, и выдачи громкого звукового и светового сигнала «Тревога», а также для работы в сети с другими автономными извещателями и внешними светозвуковыми оповещателями.

Первичными средствами пожаротушения являются углекислотные огнетушители ОУ-2, в каждом кабинете находится по одному углекислотному огнетушителю ОУ-2 размещенном в специальном шкафу, недалеко от входных дверей с внутренней стороны кабинета.

Во всех зданиях расположены дренчерные системы водяного пожаротушения. Они представляют собой напорный трубопровод и оросители, однако последние не имеют теплового элемента. Сама по себе система не реагирует на пожар, поэтому «пусковым механизмом» служит система пожарной сигнализации, которая также расположена по всему зданию, в непосредственной близости от каждого оросителя.

В соответствии с п.6 ст.17 Закона Республики Беларусь от 15.06.1993 «О пожарной безопасности» и Положением для проведения профилактических мероприятий по предупреждению и тушению пожаров в организациях, в которых численность сотрудников превышает 15 человек, организуются добровольные пожарные дружины из числа рабочих и специалистов организаций. На предприятии численность сотрудников составляет 26 человек из которых 15 состоят в добровольной пожарной дружине, основными задачами которой являются:

* контроль за соблюдением противопожарного режима;
* проведение разъяснительной работы среди рабочих, служащих, инженерно-технических работников по соблюдению противопожарного режима на рабочем месте и правил осторожного обращения с огнем в быту;
* надзор за исправностью средств пожаротушения и их укомплектованностью;
* вызов пожарной службы в случае возникновения пожара, принятие мер по его тушению имеющимися средствами пожаротушения;

Для безопасной эвакуации клиентов и работников в случае чрезвычайных ситуаций разработан эффективный план эвакуации, копии которого, расположены на каждом этаже, возле входа.

## Охрана окружающей среды

Вентиляция в любом помещении является обязательной. Естественная и искусственная вентиляция способствует нормальной циркуляции воздуха в помещении, выведению углекислого газа и других вредных веществ из воздуха. В ИП «Прокопович Н.В.» используется естественная и искусственная вентиляция во всех помещениях.

При естественной вентиляции свежий воздух проникает в помещение через различные отверстия, щели, открытые окна или двери и выводится через специальные вентиляционные каналы.

Для отчистки сточных вод в ИП «Прокопович Н.В.» используется сооружения типа ORL (ОДО «ВЭТО – сервис»). К установке приняты два блока очистных сооружений ORL 20-S с блоком тонкослойного отстаивания производительностью по 20 л/с, предназначенные для очистки ливневых и талых сточных вод с территорий населенных пунктов и промышленных площадок.

Утилизация твердых отходов на предприятие происходит следующим образом: отходы собираются в специально отведенном для этого месте, в нём располагаются специализированные контейнеры нескольких видов, при сборе отходов достаточных для утилизации их вывозит ОДО «Экология города».

ИП «Прокопович Н.В.» утилизирует лампы отдельно от остальных твердых отходов. Процесс утилизации предполагает передачу отработавших ламп предприятию переработчику – ООО «Поступ», которое с помощью специального оборудования перерабатывают их в безвредное сырье – сорбент, которое в последующем используют в качестве материала для производства.

# 8 Экономический раздел

## 8.1 Технико-экономическое обоснование разработки программного средства

Необходимо рассчитать экономическую эффективность разработки программного средства.

В технико-экономическом обосновании будут рассмотрены следующие вопросы:

- составление плана по разработке программного средства;

- расчет стоимости разработки;

- экономическая эффективность разработки.

В первой части технико-экономического обоснования необходимо рассчитать срок разработки по созданию программного средства «Разработка сайта Общества с ограниченной ответственностью «БелГруз».

Во второй части технико-экономического обоснования необходимо рассчитать стоимость разработки программного средства с момента получения первого варианта технического задания и заканчивая оформлением документации и сдачей разработки.

В третьей части необходимо оценить экономическую эффективность программного средства.

## 8.2 Составление плана по разработке программного средства

План разработки программного средства представлен в таблице 8.1.

Таблица 8.1 - План разработки программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов и видов работ | Исполнитель | Количество исполнителей | Трудоемкость, человеко-дни |
| Подготовительный | техник-программист | 1 | 2 |
| Выбор методов и средств | техник-программист | 1 | 5 |
| Разработка алгоритмов и программ | техник-программист | 1 | 20 |
| Отладка программ и анализ результатов | техник-программист | 1 | 10 |
| Оформление документации и подготовка к сдаче разработки | техник-программист | 1 | 3 |

## 8.3 Определение цены программного средства

В условиях рыночных отношений научно-техническая продукция также является товаром. Поэтому узловым вопросом технико-экономического обоснования выступает определение цены основного результата дипломного проекта.

Вместе с тем следует иметь в виду, что в отраслевых рекомендациях по установлению цены на научно-техническую продукцию фактически реализован вариант механической аналогии с материальным производством (издержек или сметной стоимости) и учет минимального уровня рентабельности.

Расчет цены основного результата дипломного проекта осуществляется в определенной последовательности.

Определяются материальные затраты на выполнение работ по теме, включая стоимость покупных комплектующих изделий и полуфабрикатов на изготовление макетов и опытных образцов.

Расчет осуществляется по формуле (8.1).

, (8.1)

где Ктзр - коэффициент, учитывающий транспортно-заготовительные расходы Ктзр≈от 1,05 до 1,10;

Hpi- норма расхода i-го вида материалов на макет или опытный образец (кг, м, и так далее);

Цi- действующая отпускная цена за единицу i-го вида материала, руб.;

Odi- возвратные отходы i-го вида материала (кг, м, и так далее);

Цdi- цена за единицу возвращенных отходов i-го вида материала, руб.;

n – количество применяемых видов материалов.

Расчет целесообразно представить в таблице 8.2.

Таблица 8.2 - Расчет затрат на материалы, покупные полуфабрикаты и комплектующие изделия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование материалов покупных полуфабрикатов и комплектующих изделий | Единица измерения | Количество | Цена при-обретения без НДС, руб. | НДС, руб. | Цена с НДС, руб. |
| Бумага | пачка | 1 | 10,00 | 1,00 | 12,46 |
| Ручка шариковая | штука | 1 | 3,00 | 0,60 | 3,60 |
| Папка-скоросшиватель | штука | 1 | 5,00 | 1,00 | 6,00 |
| Диск | штука | 1 | 1,00 | 0,20 | 1,20 |
| Всего расходов | | | | | 23,26 |
| Всего с транспортно-заготовительными расходами | | | | | 24,31 |

Затраты на электроэнергию находятся исходя из продолжительности периода разработки программного обеспечения, количества кВт/ч, затраченных на его проектирование и тарифа за 1 кВт/ч. по следующей формуле:

 (8.2)

гдеКэ– стоимость одного кВт/ч,руб;

Тр – количество кВт/ч.

Тарифы на электроэнергию применяются согласно приложения к Декларации *«*Об уровне тарифов на электроэнергию, отпускаемую РУП Электроэнергетики ГПО “Белэнерго” для юридических лиц и ИП» на соответствующий период времени, когда разрабатывается программное обеспечение.

Базовый тариф для прочих потребителей с 01.01.2021 г. составляет 0,38994 руб. за 1 кВт/ч. Время реализации проекта 40 дней, среднее потребление энергии в месяц составило 75 кВт/ч, то есть было потреблено 143 кВт/ч. Исходя из вышеизложенного получаем, что на электроэнергию было затрачено:

Рэ= 143 0,38994 = 55,76

Определяется основная заработная плата научно-технического персонала, непосредственно занятого выполнением работ.

К этой статье относятся основная заработная плата работников, а также премии, входящие в фонд заработной платы. Среднее количество рабочих дней в месяце равно 21, а средняя продолжительность рабочего дня составляет восемь часов. Следовательно, часовая заработная плата определяется делением размера оклада на количество рабочих часов в месяце (то есть на 168 часов).

Тарифная ставка первого разряда на предприятии составляет 85 руб. Трудоемкость определяется исходя из данных, представленных в таблице 8.1.

Произведение трудоемкости на сумму часовой заработной платы определяет затраты по зарплате для работника на все время разработки.

Расчет основной заработной платы по теме приведен в таблице 8.3.

Таблица 8.3 - Расчет затрат на основную заработную плату научно-производственного персонала

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование этапов работ | Исполнитель | Разряд | Тарифный  коэффициент | Часовая тарифная ставка, руб. | Трудоемкость,  (чел/час) | Затраты по заработной плате, руб. |
| Подготови-тельный | техник-программист | 7 | 2,03 | 1,04 | 16 | 16,64 |
| Выбор методов и средств | техник-  программист | 7 | 2,03 | 1,04 | 40 | 41,60 |
| Разработка алгоритмов и программ | техник-  программист | 7 | 2,03 | 1,04 | 160 | 166,40 |
| Отладка программ и анализ результатов | техник-  программист | 7 | 2,03 | 1,04 | 80 | 83,20 |
| Оформление документации и подготовка к сдаче разработки | техник-  программист | 7 | 2,03 | 1,04 | 24 | 24,96 |
| Всего | | | | | | 332,80 |
| Всего с коэффициентом премий | | | | | | 366,08 |

Определяется дополнительная заработная плата исполнителей, включающая разнообразные предусмотренные трудовым законодательством выплаты, по формуле (8.3).

, (8.3)

где Ндз - норматив дополнительной заработной платы, Ндз ≈от 10 до 25 %.

Рдз= 366,08 10/100 = 36,61

Рассчитываются отчисления органам социальной защиты по формуле (8.4):

, (8.4)

где Нос- норма отчислений на социальную защиту, Нос=34%.

Рос= (366,08 + 36,61) 34/100 = 136,92

Также рассчитываются отчисления на страхование от несчастных случаев на производстве и профессиональных заболеваний (Остр) по ставке действующего законодательства (Нбгс принимается равным от 0,3 до 0,9%). Для расчетов среднее значение Нбгс  принимается равным 0,6%. Остр рассчитывается по формуле (8.5).

. (8.5)

Остр= (366,08 + 36,61) 0,6/100 = 2,42

Определяются прочие прямые расходы, связанные с амортизационными отчислениями на полное восстановление основных производственных фондов, арендная плата и лизинговые платежи, компенсация за износ (амортизацию) использованного в процессе создания научно-технической продукции оборудования по договоренности. Рпр рассчитывается по формуле (865).

, (8.6)

где Нпр– норматив прямых расходов, Нпр≈от 10 до 20 %.

Рпр= 366,08 10/100 = 36,61

Исчисляются косвенные (накладные) расходы по формуле (8.7).

, (8.7)

где Нкос - норматив косвенных расходов, Нкос≈от 50 до 100%.

Ркос= 366,08 50/100 = 183,04

Определяется полная себестоимость научно-технической продукции как сумма всех групп затрат по формуле (8.8).

(8.8)

Сп= 17,64 + 55,76 + 366,08+ 36,61 + 136,92 + 2,42 + 36,61 + 183,04 = 835,08

По среднему уровню рентабельности в процентах от полной себестоимости определяется плановая прибыль единицы научно-технической продукции по формуле (8.9).

, (8.9)

где Ур - средний уровень рентабельности, Ур ≈от 10 до 30%.

П = 835,08 10/100 = 83,51

Определяется приближенная (ориентировочная) отпускная цена научно-технической продукции по формуле (8.10).

. (8.10)

Цотп= 835,08 + 83,51 = 918,59

Определяется налог на добавленную стоимость (НДС) по формуле (8.11).

, (8.11)

где ННДС - ставка налога НДС, ННДС=20 %.

НДС= 918,59 20/100 = 183,72

Определяется цена научно-технической продукции с учетом НДС по формуле (8.12).

( 8.12)

Цотп= 918,59 + 183,72= 1102,31

Все приведенные выше расчеты целесообразно объединить в сводную таблицу 8.4.

Таблица 8.4 - Расчет ориентировочной цены научно-технической продукции

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Условные обозначения | Сумма, руб. |
| Материалы, покупные полуфабрикаты и комплектующие изделия | Рм | 17,64 |
| Электроэнергия | Рэ | 55,76 |
| Основная заработная плата научно-производственного персонала | Роз | 366,08 |
| Дополнительная заработная плата научно-производственного персонала | Рдз | 36,61 |
| Отчисления на социальную защиту и на страхование от несчастных случаев | Рос  Остр | 136,92  2,42 |
| Прочие прямые расходы | Рпр | 36,61 |
| Накладные расходы | Ркос | 183,04 |
| Полная себестоимость | Сп | 835,08 |
| Плановые накопления (прибыль) | П | 83,51 |
| Отпускная цена (без НДС) | Ц | 918,59 |
| Налог на добавленную стоимость | РНДС | 183,72 |
| Отпускная цена с НДС | Цотп | 1102,31 |

## 8.4 Экономическая эффективность разработки

В данном разделе представлено экономическое обоснование для дипломного проекта по теме «Разработка сайта Общества с ограниченной ответственностью «БелГру».

В технико-экономическом обосновании были рассмотрены следующие вопросы:

- составление плана по разработке программы;

- расчет стоимости разработки.

В первой части технико-экономического обоснования был рассчитан срок разработки по созданию программного средства. Он составил 320 часов.

Во второй части технико-экономического обоснования была рассчитана стоимость разработки программного продукта, которая составила 1102,31 рубля.

Основное преимущество разработки программного средства в том, что данное программное средство является уникальным и подходит для обучения студентов Колледжа Бизнеса и Права основам работы с товарно-транспортными процессами. Данное программное средство разработано в соответствии со всеми нормами и правилами ведения данных о товарно-транспортных процессах, а также с выявлениями ошибок при заполнении данных.

Заключение

В рамках дипломного проекта на тему «Реализация сайта Общества с ограниченной ответственностью «БелГруз» был разработан сайт, автоматизирующий работу транспортно-логистической компании.

Для достижения цели были решены следующие задачи:

* выполнен объектно-ориентированный анализ и проектирование системы;
* выбрано и описано проектирование к разработанному Web-сайту, на базе которой создавался сайт;
* выполнено проектирование макетов для разработанного сайта;
* описан созданный сайт;
* описана методика проведения испытаний;
* описано применение созданного сайта.

На сайте были реализованы следующие функции:

* ведение единой базы данных, содержащая такие сущности как: «Cars», «Drivers», «Orders», «Tariffs», «Users», «Amenities»;
* поиск по названиям услуги;
* формирование выходных текстовых документов, которые будут содержать данные обо всех соответствующих сущностях.

В дипломном проекте были также рассмотрены вопросы охраны труда и окружающей среды, такие как основные правовые, разработка инженерно-технических решений по снижению шума в рабочей зоне вычислительных центров, вопросы пожарной безопасности, охраны окружающей среды.

В экономическом разделе были произведены расчеты экономического эффекта программного средства, расчеты заработной платы исполнителей, была проведена оценка сложности и трудоемкости разработки программного средства, были проведены расчеты всевозможных отчислений и расходов, был рассчитан срок разработки по созданию программного средства, который составил 320 часов, а также была рассчитана стоимость разработки программного продукта, которая составила 1102,31 рублей.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальными знаниями компьютера пользоваться сайтом.

Сайт реализован в полном объеме и в соответствии с заданными требованиями, полностью отлажено и протестировано. Поставленные задачи выполнены.

Сайт готов к практическому использованию.

Список информационных источников

1. Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2013
2. Общие требования к тестовым документам: ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 1995.
3. Описание и текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.402-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000.
4. Клиент [Электронный ресурс] – Wikipedia – Режим доступа: <https://ru.wikipedia.org/wiki/Клиент> − Дата доступа: 04.04.2021
5. Зачем нужен администратор интернет-магазину [Электронный ресурс] – Бизнесоголикс 2020 – Режим доступа: <https://biznesogoliks.ru/administrator-internet-magazina/> – Дата доступа: 04.04.2021
6. Каталог (торговля) [Электронный ресурс] – Wikipedia – Режим доступа: <https://ru.wikipedia.org/wiki/Каталог_(торговля)> − Дата доступа: 04.04.2021
7. Заказ [Электронный ресурс] – Wikipedia – Режим доступа: <https://ru.wikipedia.org/wiki/Заказ> − Дата доступа: 04.04.2021
8. Разница межу INCLUDE и EXTEND диаграмма вариантов использования (use case) [Электронный ресурс] − fkn+antitotal − Режим доступа: <http://fkn.ktu10.com/?q=node/2236> − Дата доступа: 11.04.2021
9. StarUML [Электронный ресурс] − Wikipedia − Режим доступа: <https://en.wikipedia.org/wiki/StarUML> − Дата доступа: 16.04.2021
10. Hормализация таблиц, ее цель. Первая нормальная форма. Вторая нормальная форма. Третья нормальная форма. [Электронный ресурс] − StudFiles − Режим доступа: <https://studfile.net/preview/7003805/page:17/> − Дата доступа: 16.04.2021
11. Тестирование клиентского приложения [Электронный ресурс] − Режим доступа:<http://lib.kstu.kz:8300/tb/books/2014/IS/Proektirovanie%20i%20razrabotka%20prilozheniy%20BD/lab/lr6.htm> − Дата доступа: 16.04.2021
12. Ruby [Электронный ресурс] − Wikipedia − Режим доступа: https://ru.wikipedia.org/wiki/Ruby − Дата доступа: 19.04.2021
13. Ruby on Rails [Электронный ресурс] − Wikipedia − Режим доступа: https://ru.wikipedia.org/wiki/Ruby\_on\_Rails − Дата доступа: 19.04.2021
14. CSS [Электронный ресурс] − Wikipedia − Режим доступа: <https://ru.wikipedia.org/wiki/CSS> − Дата доступа: 19.04.2021
15. PostgreSQL [Электронный ресурс] − Wikipedia − Режим доступа: https://ru.wikipedia.org/wiki/PostgreSQL − Дата доступа: 19.04.2021
16. Model-View-Controller [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller> − Дата доступа: 19.04.2021
17. Visual Studio Code [Электронный ресурс] − Wikipedia − Режим доступа: <https://ru.wikipedia.org/wiki/Visual_Studio_Code> − Дата доступа: 22.04.2021
18. Газовый состав воздушной среды и его изменение в результате производственных процессов [Электронный ресурс] – Всё о красках, 2021 – Режим доступа: <http://vseokraskah.net/rukovodstvo-po-bezopasnosti/gazovyj-sostav-vozdushnoj-sredy-i-ego-izmenenie-v-rezultate-proizvodstvennyx-processov.html> – Дата доступа: 14.03.2021
19. Заказ [Электронный ресурс] – Wikipedia 2020 – Режим доступа: <https://ru.wikipedia.org/wiki/Заказ> − Дата доступа: 20.04.2021
20. Зачем нужен администратор интернет-магазину [Электронный ресурс] – Бизнесоголикс 2020 – Режим доступа: <https://biznesogoliks.ru/administrator-internet-magazina/> – Дата доступа: 20.04.2021
21. Интернет-магазин [Электронный ресурс] – Wikipedia 2016 – Режим доступа: <https://ru.wikipedia.org/wiki/Интернет-магазин> – Дата доступа: 20.04.2021
22. Каталог (торговля) [Электронный ресурс] – Wikipedia 2018 – Режим доступа: <https://ru.wikipedia.org/wiki/Каталог_(торговля)> − Дата доступа: 20.04.2021
23. Клиент [Электронный ресурс] – Wikipedia 2016 – Режим доступа: <https://ru.wikipedia.org/wiki/Клиент> − Дата доступа: 20.04.2021
24. Охрана труда [Электронный ресурс] − Администрация Фрунзенского района города Минска, 2017 − Режим доступа: <https://www.fr.gov.by/main/labour-protection/> − Дата доступа: 10.02.2021
25. Разница межу INCLUDE и EXTEND диаграмма вариантов использования (use case) [Электронный ресурс] − fkn+antitotal − Режим доступа: <http://fkn.ktu10.com/?q=node/2236> − Дата доступа: 22.04.2021
26. Склад [Электронный ресурс] – Wikipedia 2020 – Режим доступа: <https://ru.wikipedia.org/wiki/Склад> − Дата доступа: 20.04.2021
27. Тестирование клиентского приложения [Электронный ресурс] − Режим доступа:<http://lib.kstu.kz:8300/tb/books/2014/IS/Proektirovanie%20i%20razrabotka%20prilozheniy%20BD/lab/lr6.htm> − Дата доступа: 25.04.2021
28. Учебник. Создание веб-API с помощью ASP.NET Core [Электронный ресурс] – Microsoft, 2019 – Режим доступа: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio> − Дата доступа: 22.04.2021
29. Учимся быть фуллстек разработчиками. Пишем приложение на React/Redux/Webpack/ASP.NET Core 2.0/EF Core [Электронный ресурс] – Харб 2018 − Режим доступа: 20.04.2021

Приложение А

(обязательное)

Текст программы

class UsersController < ApplicationController

def create

new\_user = User.new(user\_params) //Создание клиента

render\_created\_data(new\_user, users)

end

def destroy

user.destroy

render json: users

end

def index

render json: users

end

private

def user

@user ||= User.find(params[:id])

end

def users

@users ||= User.all

end

def user\_params

params.require(:user).permit(

:name,

:phone,

:password,

:email

)

end

end

class TariffsController < ApplicationController

def create

new\_tariff = Tariff.new(tariff\_params) //Создание тарифов

render\_created\_data(new\_tariff, tariffs) // Отрисовка тарифов

end

def destroy

tariff.destroy

render json: tariffs

end

def index

render json: tariffs

end

private

def tariff

@tariff ||= Tariff.find(params[:id])

end

def tariffs

@tariffs ||= Tariff.all

end

def tariff\_params

params.require(:tariff).permit(

:name

)

end

end

class StaticController < ApplicationController

def index; end

end

class ReportsController < ApplicationController

def amenities\_report

datetime = Time.now.strftime('%d/%m/%Y %H:%M')

file\_name = File.join(File.dirname(\_\_FILE\_\_), "../../reports/услуги #{datetime.tr('/', '.')}.txt")

File.open(file\_name, 'a') do |file|

file.puts "дата отчета: #{datetime}\n\n"

file.puts "услуги:\n"

Amenity.all.map { |item| file.puts "- #{item.name} | #{item.price} BYN | #{item.price\_with\_workers} BYN" }

end

end

def drivers\_report //Создание отчёта водители

datetime = Time.now.strftime('%d/%m/%Y %H:%M')

file\_name = File.join(File.dirname(\_\_FILE\_\_), "../../reports/водители #{datetime.tr('/', '.')}.txt")

File.open(file\_name, 'a') do |file|

file.puts "дата отчета: #{datetime}\n\n"

file.puts "водители:\n"

Driver.all.map { |item| file.puts "- #{item.name} | #{item.phone} | #{item.car.number} - #{item.car.name}" }

end

end

def cars\_report

datetime = Time.now.strftime('%d/%m/%Y %H:%M')

file\_name = File.join(File.dirname(\_\_FILE\_\_), "../../reports/машины #{datetime.tr('/', '.')}.txt")

File.open(file\_name, 'a') do |file|

file.puts "дата отчета: #{datetime}\n\n"

file.puts "машины:\n"

Car.all.map { |item| file.puts "- #{item.name} | #{item.number} | #{item.load\_capacity} кг. | #{item.height} м." }

end

end

def tariffs\_report //Создание отчёта тарифы

datetime = Time.now.strftime('%d/%m/%Y %H:%M')

file\_name = File.join(File.dirname(\_\_FILE\_\_), "../../reports/тарифы #{datetime.tr('/', '.')}.txt")

File.open(file\_name, 'a') do |file|

file.puts "дата отчета: #{datetime}\n\n"

file.puts "тарифы:\n"

Tariff.all.map { |item| file.puts "- #{item.name}" }

end

end

def orders\_report //Создание отчёта заказы

datetime = Time.now.strftime('%d/%m/%Y %H:%M')

file\_name = File.join(File.dirname(\_\_FILE\_\_), "../../reports/заказы #{datetime.tr('/', '.')}.txt")

File.open(file\_name, 'a') do |file|

file.puts "дата отчета: #{datetime}\n\n"

file.puts "заказы:\n"

Order.all.map do |item|

file.puts "##{item.id}\n"

file.puts "начало заказа: #{item.startPoint}\n"

file.puts "конец заказа: #{item.finishPoint}\n"

file.puts "вес груза: #{item.weight}\n"

file.puts "высота груза: #{item.height}\n"

file.puts "доп. информация: #{item.note}\n"

file.puts "имя заказчика: #{item.customer\_name}\n"

file.puts "телефон заказчика: #{item.customer\_phone}\n"

file.puts "почта заказчика: #{item.customer\_email}\n"

file.puts "машина: #{item.car.name} | #{item.car.number}\n"

file.puts "водитель: #{item.car.driver.name} | #{item.car.driver.phone}\n"

file.puts "тариф: #{item.tariff.name}\n"

file.puts "создано: #{item.created\_at}\n\n"

end

end

end

end

class OrdersController < ApplicationController

def create

new\_order = Order.new(order\_params) //Создание заказа

render\_created\_data(new\_order, orders)

end

def destroy

order.destroy

render json: orders

end

def index

render json: orders

end

private

def order

@order ||= Order.find(params[:id])

end

def orders

@orders ||= Order.all

end

def order\_params

params.require(:order).permit(

:startPoint,

:finishPoint,

:height,

:weight,

:note,

:customer\_phone,

:customer\_name,

:customer\_email,

:car\_id,

:amenity\_id,

:tariff\_id

)

end

end

class DriversController < ApplicationController

def create

new\_driver = Driver.new(driver\_params) // Создание водителя

render\_created\_data(new\_driver, drivers)

end

def destroy

driver.destroy

render json: drivers

end

def index

render json: drivers

end

private

def driver

@driver ||= Driver.find(params[:id])

end

def drivers

@drivers ||= Driver.all

end

def driver\_params // Параметры Водителя

params.require(:driver).permit(

:name,

:phone

)

end

end

class CarsController < ApplicationController

def create

new\_car = Car.new(car\_params)

render\_created\_data(new\_car, cars)

end

def destroy

car.destroy

render json: cars

end

def index

render json: cars

end

private

def car

@car ||= Car.find(params[:id])

end

def cars

@cars ||= Car.all

end

def car\_params

params.require(:car).permit(

:height,

:name,

:number,

:load\_capacity,

:driver\_id

)

end

end

class ApplicationController < ActionController::Base

protect\_from\_forgery with: :null\_session

def render\_created\_data(check, data)

if check.save

render json: data

else

render json: { errors: check.errors }, status: :unprocessable\_entity

end

end

end

class AmenitiesController < ApplicationController

def create

new\_amenity = Amenity.new(amenity\_params)

render\_created\_data(new\_amenity, amenities)

end

def destroy

amenity.destroy

render json: amenities

end

def index

render json: amenities

end

private

def amenity

@amenity ||= Amenity.find(params[:id])

end

def amenities

@amenities ||= Amenity.all

end

def amenity\_params

params.require(:amenity).permit(

:name,

:description,

:price\_with\_workers,

:price

)

end

end

import {

INDEX\_AMENITY,

INDEX\_CAR,

INDEX\_DRIVER,

INDEX\_ORDER,

INDEX\_TARIFF,

INDEX\_USER,

SET\_ERRORS,

CALL\_ADMIN\_ENTER,

SET\_CURRENT\_USER,

UNSET\_CURRENT\_USER

} from './constants';

export const indexAmenity = (items) => ({

type: INDEX\_AMENITY,

value: items,

});

export const indexCar = (items) => ({

type: INDEX\_CAR,

value: items,

});

export const indexDriver = (items) => ({

type: INDEX\_DRIVER,

value: items,

});

export const indexOrder = (items) => ({

type: INDEX\_ORDER,

value: items,

});

export const indexTariff = (items) => ({

type: INDEX\_TARIFF,

value: items,

});

export const indexUser = (items) => ({

type: INDEX\_USER,

value: items,

});

export const setErrors = (items) => ({

type: SET\_ERRORS,

value: items,

});

export const callAdminEnter = () => ({

type: CALL\_ADMIN\_ENTER,

});

export const setCurrentUser = (user) => ({

type: SET\_CURRENT\_USER,

value: user

});

export const unsetCurrentUser = () => ({

type: UNSET\_CURRENT\_USER,

});

import {

INDEX\_AMENITY,

INDEX\_CAR,

INDEX\_DRIVER,

INDEX\_ORDER,

INDEX\_TARIFF,

INDEX\_USER,

SET\_ERRORS,

CALL\_ADMIN\_ENTER,

SET\_CURRENT\_USER,

UNSET\_CURRENT\_USER

} from './constants'

let initialState = {

amenities: [],

cars: [],

drivers: [],

orders: [],

tariffs: [],

users: [],

errors: [],

isAdminEnterCalled: false,

currentUser: { },

};

const MainReducer = (state = initialState, action) => {

switch (action.type) {

case INDEX\_AMENITY: {

return { ...state, amenities: action.value };

}

case INDEX\_CAR: {

return { ...state, cars: action.value };

}

case INDEX\_DRIVER: {

return { ...state, drivers: action.value };

}

case INDEX\_ORDER: {

return { ...state, orders: action.value };

}

case INDEX\_TARIFF: {

return { ...state, tariffs: action.value };

}

case INDEX\_USER: {

return { ...state, users: action.value };

}

case SET\_ERRORS: {

return { ...state, errors: action.value };

}

case CALL\_ADMIN\_ENTER: {

return { ...state, isAdminEnterCalled: !state.isAdminEnterCalled };

}

case SET\_CURRENT\_USER: {

console.log(action.value)

return { ...state, currentUser: action.value };

}

case UNSET\_CURRENT\_USER: {

return { ...state, currentUser: null };

}

default: {

return state;

}

}

}

export default MainReducer;

import React, { useEffect } from 'react'

import { Link } from 'react-router-dom'

import phoneCall from '../icons/phone-call.svg'

import { callAdminEnter, indexUser, unsetCurrentUser } from '../main\_redux/actions'

import { connect } from "react-redux";

import './style.css'

import { get } from '../server\_connections';

import { TextField } from '@material-ui/core';

const AdminTopPanel = (props) => {

useEffect(() => {

props.set("users", indexUser);

}, []);

return (

<div>

{

props.currentUser ? <div>

<div className={props.isAdminEnterCalled && 'blured'}>

<div className='nav-block' style={{marginTop: '-10px', height: '60px'}}>

<Link to={`/admin/amenity`} className='nav-link'>Услуги</Link>

<Link to={`/admin/car`} className='nav-link'>Машины</Link>

<Link to={`/admin/driver`} className='nav-link'>Водители</Link>

<Link to={`/admin/tariff`} className='nav-link'>Тарифы</Link>

<Link to={`/admin/order`} className='nav-link'>Заказы</Link>

<Link to={`/`} onClick={props.unsetCurrentUser} className='nav-link'>Выход</Link>

</div>

<div className='page-content' style={{height: '800px'}}>

{props.children}

</div>

<div className='footer'>

<p>COPYRIGHT 2021 ООО БЕЛГРУЗ BELGRUZ@GMAIL.BY</p>

</div>

</div>

</div> :

<h1>

Данные введены неверно

</h1>

}

</div>

)

}

export default connect(

(state) => ({

users: state.users,

currentUser: state.currentUser,

isAdminEnterCalled: state.isAdminEnterCalled,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

callAdminEnter: () => dispatch(callAdminEnter()),

unsetCurrentUser: () => dispatch(unsetCurrentUser()),

})

)(AdminTopPanel);

import React, { useEffect } from "react";

import { connect } from "react-redux";

import { indexAmenity } from "../main\_redux/actions.js";

import { get } from "../server\_connections.js";

import "./style.css";

const AmenitiesList = (props) => {

useEffect(() => {

props.set("amenities", indexAmenity);

}, []);

return (

<div>

<div className='ametities-price-note'>

Все цены указаны исходя из среднего значения стоимости заказов, для уточнения цены необходимо оставить заявку и с вами свяжется наш специалист.

</div>

<ul className='list-amenities'>

<div className='list-amenities-item' style={{color: 'black'}}>

<div className='list-amenities-item-name'>

Название услуги

</div>

<div className='list-amenities-item-price'>

стоимость

</div>

<div className='list-amenities-item-price'>

стоимость с грузчиками

</div>

</div>

{

props.amenities.map(el =>

<li className='list-amenities-item' key={el.id}>

<div className='list-amenities-item-name'>

{el.name}

</div>

<div className='list-amenities-item-price'>

~{el.price} BYN

</div>

<div className='list-amenities-item-price'>

~{el.price\_with\_workers} BYN

</div>

</li>)

}

</ul>

</div>

);

};

export default connect(

(state) => ({

amenities: state.amenities,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

})

)(AmenitiesList);

import React, { useEffect } from 'react'

import MaterialTable from "material-table";

import AdminTopPanel from './AdminTopPanel';

import { connect } from "react-redux";

import { indexAmenity } from '../main\_redux/actions';

import { get, drop, post, createReport } from '../server\_connections';

import { tableIcons } from '../table\_icons';

import './style.css'

import { TextField, Tooltip } from '@material-ui/core';

import { useInputText } from './OrderCreation';

import exportFile from '../icons/file.svg'

const AmenityAdminPage = (props) => {

const columns = [

{ title: "Название", field: "name" },

{ title: "Цена без грузчиков", field: "price" },

{ title: "Цена с грузчиками", field: "price\_with\_workers" },

]

useEffect(() => {

props.set("amenities", indexAmenity);

}, []);

const edits = {

onRowDelete: (oldData) => new Promise((resolve) => {

props.drop(oldData.id, 'amenities', indexAmenity)

resolve();

})

}

const name = useInputText("");

const description = useInputText("");

const price = useInputText("");

const price\_with\_workers = useInputText("");

const newOrder = {

amenity: {

name: name.value,

description: description.value,

price: price.value,

price\_with\_workers: price\_with\_workers.value,

},

};

const createOrder = () => {

props.post(newOrder, "amenities", indexAmenity);

};

const createReport = () => {

props.createReport('amenities\_report')

}

return(

<div>

<AdminTopPanel>

<div className={'work-space'}>

<div className='admin-manage-table'>

<MaterialTable

icons={tableIcons}

title={

<div>

Услуги

<Tooltip title="Экспорт в файл">

<img style={{width: '25px', marginLeft: '20px', marginBottom: '-5px'}}

src={exportFile}

onClick={createReport}

alt={'экспортировать в файл'}/>

</Tooltip>

</div>

}

columns={columns}

data={props.amenities}

editable={edits}

/>

</div>

<div className='admin-manage-panel'>

<h3>Добавление услуги</h3>

<div>

<div className="admin-add-panel">

<div>

<TextField

error={props.errors.name != undefined}

helperText={

props.errors.name != undefined ? props.errors.name[0] : null

}

{...name} placeholder='название'/>

</div>

<div>

<TextField

error={props.errors.price != undefined}

helperText={

props.errors.price != undefined ? props.errors.price[0] : null

}

{...price} placeholder='цена без грузчиков'/>

</div>

<div>

<TextField

error={props.errors.price\_with\_workers != undefined}

helperText={

props.errors.price\_with\_workers != undefined ? props.errors.price\_with\_workers[0] : null

}

{...price\_with\_workers} placeholder='цена с грузчиками'/>

</div>

<div>

<button onClick={createOrder} style={{width: '80%', marginBottom: '20px'}} className="order-button">

Добавить

</button>

</div>

</div>

</div>

</div>

</div>

</AdminTopPanel>

</div>

)

}

export default connect(

(state) => ({

amenities: state.amenities,

errors: state.errors,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

drop: (id, path, setter) => dispatch(drop(id, path, setter)),

post: (obj, path, setter) => dispatch(post(obj, path, setter)),

createReport: (path) => dispatch(createReport(path))

})

)(AmenityAdminPage);

import React, { useEffect, useState } from 'react'

import MaterialTable from "material-table";

import AdminTopPanel from './AdminTopPanel';

import { connect } from "react-redux";

import { indexAmenity, indexCar, indexDriver } from '../main\_redux/actions';

import { get, drop, post, createReport } from '../server\_connections';

import { tableIcons } from '../table\_icons';

import './style.css'

import { FormHelperText, TextField, Tooltip } from '@material-ui/core';

import { useInputText } from './OrderCreation';

import exportFile from '../icons/file.svg'

const CarAdminPage = (props) => {

const columns = [

{ title: "Номер", field: "number" },

{ title: "Название", field: "name" },

{ title: "Высота (м)", field: "height" },

{ title: "Допустимая загруженность (кг)", field: "load\_capacity" },

{ title: "Водитель", field: "driver.name" },

]

useEffect(() => {

props.set("cars", indexCar);

props.set("drivers", indexDriver);

}, []);

const edits = {

onRowDelete: (oldData) => new Promise((resolve) => {

props.drop(oldData.id, 'cars', indexCar)

resolve();

})

}

const number = useInputText("");

const name = useInputText("");

const height = useInputText("");

const load\_capacity = useInputText("");

const [driver, setDriver] = useState(null);

const newOrder = {

car: {

number: number.value,

name: name.value,

height: height.value,

load\_capacity: load\_capacity.value,

driver\_id: +driver,

},

};

console.log(props.errors)

const createOrder = () => {

props.post(newOrder, "cars", indexCar);

};

const createReport = () => {

props.createReport('cars\_report')

}

return(

<div>

<AdminTopPanel>

<div className={'work-space'}>

<div className='admin-manage-table'>

<MaterialTable

icons={tableIcons}

title={

<div>

Машины

<Tooltip title="Экспорт в файл">

<img style={{width: '25px', marginLeft: '20px', marginBottom: '-5px'}}

src={exportFile}

onClick={createReport}

alt={'экспортировать в файл'}/>

</Tooltip>

</div>

}

columns={columns}

data={props.cars}

editable={edits}

/>

</div>

<div className='admin-manage-panel'>

<h3>Добавление машины</h3>

<div>

<div className="admin-add-panel">

<div>

<TextField

error={props.errors.name != undefined}

helperText={

props.errors.name != undefined ? props.errors.name[0] : null

}

{...name} placeholder='название'/>

</div>

<div>

<TextField

error={props.errors.number != undefined}

helperText={

props.errors.number != undefined ? props.errors.number[0] : null

}

{...number} placeholder='номер'/>

</div>

<div>

<TextField

error={props.errors.height != undefined}

helperText={

props.errors.height != undefined ? props.errors.height[0] : null

}

{...height} placeholder='высота'/>

</div>

<div>

<TextField

error={props.errors.load\_capacity != undefined}

helperText={

props.errors.load\_capacity != undefined ? props.errors.load\_capacity[0] : null

}

{...load\_capacity} placeholder='загруженность'/>

</div>

<div>

<p style={{fontSize: '15px', marginBottom: '0px'}}>Выберите водителя</p>

<select

onChange={(e) => {

setDriver(e.target.value);

}}

>

{props.drivers.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.driver != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.driver[1]}

</FormHelperText>

) : null}

</div>

<div>

<button onClick={createOrder} style={{width: '80%', marginBottom: '20px'}} className="order-button">

Добавить

</button>

</div>

</div>

</div>

</div>

</div>

</AdminTopPanel>

</div>

)

}

export default connect(

(state) => ({

cars: state.cars,

drivers: state.drivers,

errors: state.errors,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

drop: (id, path, setter) => dispatch(drop(id, path, setter)),

post: (obj, path, setter) => dispatch(post(obj, path, setter)),

createReport: (path) => dispatch(createReport(path))

})

)(CarAdminPage);

import React, { useEffect } from 'react'

import MaterialTable from "material-table";

import AdminTopPanel from './AdminTopPanel';

import { connect } from "react-redux";

import { indexAmenity, indexCar, indexDriver } from '../main\_redux/actions';

import { get, drop, post, createReport } from '../server\_connections';

import { tableIcons } from '../table\_icons';

import './style.css'

import { TextField, Tooltip } from '@material-ui/core';

import { useInputText } from './OrderCreation';

import exportFile from '../icons/file.svg'

const DriverAdminPage = (props) => {

const columns = [

{ title: "Имя", field: "name" },

{ title: "Телефон", field: "phone" },

{ title: "Номер машины", field: "car.number" },

]

useEffect(() => {

props.set("drivers", indexDriver);

}, []);

const edits = {

onRowDelete: (oldData) => new Promise((resolve) => {

props.drop(oldData.id, 'drivers', indexDriver)

resolve();

})

}

const name = useInputText("");

const phone = useInputText("");

const newOrder = {

driver: {

name: name.value,

phone: phone.value

},

};

const createOrder = () => {

props.post(newOrder, "drivers", indexDriver);

};

const createReport = () => {

props.createReport('drivers\_report')

}

return(

<div>

<AdminTopPanel>

<div className={'work-space'}>

<div className='admin-manage-table'>

<MaterialTable

icons={tableIcons}

title={

<div>

Водители

<Tooltip title="Экспорт в файл">

<img style={{width: '25px', marginLeft: '20px', marginBottom: '-5px'}}

src={exportFile}

onClick={createReport}

alt={'экспортировать в файл'}/>

</Tooltip>

</div>

}

columns={columns}

data={props.drivers}

editable={edits}

/>

</div>

<div className='admin-manage-panel'>

<h3>Добавление услуги</h3>

<div>

<div className="admin-add-panel">

<div>

<TextField

error={props.errors.name != undefined}

helperText={

props.errors.name != undefined ? props.errors.name[0] : null

}

{...name} placeholder='имя'/>

</div>

<div>

<TextField

error={props.errors.phone != undefined}

helperText={

props.errors.phone != undefined ? props.errors.phone[0] : null

}

{...phone} placeholder='номер телефона'/>

</div>

<div>

<button onClick={createOrder} style={{width: '80%', marginBottom: '20px'}} className="order-button">

Добавить

</button>

</div>

</div>

</div>

</div>

</div>

</AdminTopPanel>

</div>

)

}

export default connect(

(state) => ({

drivers: state.drivers,

errors: state.errors,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

drop: (id, path, setter) => dispatch(drop(id, path, setter)),

post: (obj, path, setter) => dispatch(post(obj, path, setter)),

createReport: (path) => dispatch(createReport(path))

})

)(DriverAdminPage);

import React from 'react'

const GeneralPage = () => {

return(

<div>

<p>

<span className='company-label'>ООО «БелГруз»</span> – современная логистическая компания в Минске, входит в состав ACE Logistics Group.<br/>

ACE Logistics Group – глобальный поставщик международных транспортно-логистических услуг.</p>

<p>

Сегодня <span className='company-label'>ООО «БелГруз»</span> – это:

</p>

<ul className='info-list'>

<li>успешный опыт работы на рынке международных транспортно-логистических услуг с 1992 года;</li>

<li>22,8 миллионов евро годового оборота;</li>

<li>офисы в Эстонии, Литве, Латвии и Беларуси;</li>

<li>разветвленная партнёрская сеть во всех странах мира;</li>

<li>более 130 высококвалифицированных сотрудников;</li>

<li>более 12 000 грузов ежемесячно.</li>

</ul>

<p>

<span className='company-label'>ООО «БелГруз»</span> – член World Air Cargo Organization (WACO) в Эстонии, Литве, Латвии и Беларуси.<br/>

ISO сертификаты подтверждают качество оказываемых услуг.</p>

<p>

ИООО «АЦЕ Логистикс Бел» оказывает следующие логистические услуги:

</p>

<ul className='info-list'>

<li>автомобильные перевозки;</li>

<li>авиаперевозки;</li>

<li>морские перевозки;</li>

<li>железнодорожные перевозки;</li>

<li>мультимодальные перевозки;</li>

<li>складские и консолидационные услуги;</li>

<li>таможенное оформление и страхование грузов.</li>

</ul>

</div>

)

}

export default GeneralPage

import React, { useEffect, useState } from 'react'

import MaterialTable from "material-table";

import AdminTopPanel from './AdminTopPanel';

import { connect } from "react-redux";

import { indexAmenity, indexCar, indexOrder, indexTariff } from '../main\_redux/actions';

import { get, drop, post, createReport } from '../server\_connections';

import { tableIcons } from '../table\_icons';

import './style.css'

import { FormHelperText, TextField, Tooltip } from '@material-ui/core';

import { useInputText } from './OrderCreation';

import exportFile from '../icons/file.svg'

export const datetimeFormat = (date) =>`${date.split(/(T|\.)/)[2]} ${date.split(/(T|\.)/)[0].split("-").reverse().join("-")}`;

const dataFormatter = (data) => data.map(el => (

{

...el,

created\_at: datetimeFormat(el.created\_at),

note: el.note ? el.note : '-'

})

)

const OrderAdminPage = (props) => {

const columns = [

{ title: "Номер", field: "id" },

{ title: "Адрес начала", field: "startPoint" },

{ title: "Адрес завершения", field: "finishPoint" },

{ title: "Вес (кг)", field: "weight" },

{ title: "Высота (м)", field: "height" },

{ title: "Имя заказчика", field: "customer\_name" },

{ title: "Почта заказчика", field: "customer\_email" },

{ title: "Телефон заказчика", field: "customer\_phone" },

{ title: "Номер машины", field: "car.number" },

{ title: "Тариф", field: "tariff.name" },

{ title: "Услуга", field: "amenity.name" },

{ title: "Доп. информация", field: "note" },

{ title: "Дата оформления", field: "created\_at" },

]

useEffect(() => {

props.set("orders", indexOrder);

props.set("amenities", indexAmenity);

props.set("cars", indexCar);

props.set("tariffs", indexTariff);

}, []);

const edits = {

onRowDelete: (oldData) => new Promise((resolve) => {

props.drop(oldData.id, 'orders', indexOrder)

resolve();

})

}

const customer\_name = useInputText("");

const customer\_email = useInputText("");

const customer\_phone = useInputText("");

const startPoint = useInputText("");

const finishPoint = useInputText("");

const weight = useInputText("");

const height = useInputText("");

const note = useInputText("");

const [tariff, setTariff] = useState(null);

const [amenity, setAmenity] = useState(null);

const [car, setCar] = useState(null);

const newOrder = {

order: {

customer\_name: customer\_name.value,

customer\_phone: customer\_phone.value,

customer\_email: customer\_email.value,

startPoint: startPoint.value,

finishPoint: finishPoint.value,

weight: weight.value,

height: height.value,

note: note.value,

car\_id: +car,

amenity\_id: +amenity,

tariff\_id: +tariff,

},

};

const createOrder = () => {

props.post(newOrder, "orders", indexOrder);

};

const createReport = () => {

props.createReport('orders\_report')

}

return(

<div>

<AdminTopPanel>

<div className={'work-space'}>

<div className='admin-manage-table'>

<MaterialTable

icons={tableIcons}

title={

<div>

Заявки

<Tooltip title="Экспорт в файл">

<img style={{width: '25px', marginLeft: '20px', marginBottom: '-5px'}}

src={exportFile}

onClick={createReport}

alt={'экспортировать в файл'}/>

</Tooltip>

</div>

}

columns={columns}

data={dataFormatter(props.orders)}

editable={edits}

/>

</div>

<div className='admin-manage-panel'>

<h3>Добавление заказа</h3>

<div>

<div className="admin-add-panel">

<div>

<TextField

error={props.errors.customer\_name != undefined}

helperText={

props.errors.customer\_name != undefined ? props.errors.customer\_name[0] : null

}

{...customer\_name} placeholder='имя заказчика'/>

</div>

<div>

<TextField

error={props.errors.customer\_phone != undefined}

helperText={

props.errors.customer\_phone != undefined ? props.errors.customer\_phone[0] : null

}

{...customer\_phone} placeholder='телефон заказчика'/>

</div>

<div>

<TextField

error={props.errors.customer\_email != undefined}

helperText={

props.errors.customer\_email != undefined ? props.errors.customer\_email[0] : null

}

{...customer\_email} placeholder='почта заказчика'/>

</div>

<div>

<TextField

error={props.errors.startPoint != undefined}

helperText={

props.errors.startPoint != undefined ? props.errors.startPoint[0] : null

}

{...startPoint} placeholder='адрес начала'/>

</div>

<div>

<TextField

error={props.errors.finishPoint != undefined}

helperText={

props.errors.finishPoint != undefined ? props.errors.finishPoint[0] : null

}

{...finishPoint} placeholder='адрес завершения'/>

</div>

<div>

<TextField

error={props.errors.weight != undefined}

helperText={

props.errors.weight != undefined ? props.errors.weight[0] : null

}

{...weight} placeholder='вес груза'/>

</div>

<div>

<TextField

error={props.errors.height != undefined}

helperText={

props.errors.height != undefined ? props.errors.height[0] : null

}

{...height} placeholder='высота груза'/>

</div>

<div>

<TextField

error={props.errors.note != undefined}

helperText={

props.errors.note != undefined ? props.errors.note[0] : null

}

{...note} placeholder='доп информация'/>

</div>

<div>

<p style={{fontSize: '15px', marginBottom: '0px'}}>Выберите тариф</p>

<select

onChange={(e) => {

setTariff(e.target.value);

}}

>

{props.tariffs.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.tariff != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.tariff[1]}

</FormHelperText>

) : null}

</div>

<div>

<p style={{fontSize: '15px', marginBottom: '0px'}}>Выберите услугу</p>

<select

onChange={(e) => {

setAmenity(e.target.value);

}}

>

{props.amenities.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.amenity != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.amenity[1]}

</FormHelperText>

) : null}

</div>

<div>

<p style={{fontSize: '15px', marginBottom: '0px'}}>Выберите машину</p>

<select

onChange={(e) => {

setCar(e.target.value);

}}

>

{props.cars.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.car != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.car[1]}

</FormHelperText>

) : null}

</div>

</div>

<div>

<button onClick={createOrder} style={{width: '80%', marginBottom: '20px'}} className="order-button">

Добавить

</button>

</div>

</div>

</div>

</div>

</AdminTopPanel>

</div>

)

}

export default connect(

(state) => ({

orders: state.orders,

amenities: state.amenities,

tariffs: state.tariffs,

cars: state.cars,

errors: state.errors,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

drop: (id, path, setter) => dispatch(drop(id, path, setter)),

post: (obj, path, setter) => dispatch(post(obj, path, setter)),

createReport: (path) => dispatch(createReport(path))

})

)(OrderAdminPage);

import { FormHelperText, TextField } from "@material-ui/core";

import React, { useEffect, useState } from "react";

import { connect } from "react-redux";

import {

indexAmenity,

indexCar,

indexOrder,

indexTariff,

} from "../main\_redux/actions.js";

import { get, post } from "../server\_connections.js";

import "./style.css";

const OrderCreation = (props) => {

useEffect(() => {

props.set("amenities", indexAmenity);

props.set("tariffs", indexTariff);

props.set("cars", indexCar);

}, []);

const customer\_name = useInputText("");

const customer\_email = useInputText("");

const customer\_phone = useInputText("");

const startPoint = useInputText("");

const finishPoint = useInputText("");

const weight = useInputText("");

const height = useInputText("");

const note = useInputText("");

const [tariff, setTariff] = useState(null);

const [amenity, setAmenity] = useState(null);

const [car, setCar] = useState(null);

const newOrder = {

order: {

customer\_name: customer\_name.value,

customer\_phone: customer\_phone.value,

customer\_email: customer\_email.value,

startPoint: startPoint.value,

finishPoint: finishPoint.value,

weight: weight.value,

height: height.value,

note: note.value,

car\_id: +car,

amenity\_id: +amenity,

tariff\_id: +tariff,

},

};

const createOrder = () => {

console.log(newOrder);

props.post(newOrder, "orders", indexOrder);

};

console.log(props.errors);

return (

<div className={"order-creation-layout"}>

<div>

<div>

<h4 style={{ textDecoration: "underLine" }}>

Укажите информацию о вас

</h4>

</div>

<div className={"input-info-block"}>

<div className={"text-block-input"}>

<p>Ф.И.О.</p>

<TextField

error={props.errors.customer\_name != undefined}

helperText={

props.errors.customer\_name != undefined

? props.errors.customer\_name[0]

: null

}

{...customer\_name}

style={{ width: "200px" }}

placeholder="введите полное имя..."

></TextField>

</div>

<div className={"text-block-input"}>

<p>Email</p>

<TextField

error={props.errors.customer\_email != undefined}

helperText={

props.errors.customer\_email != undefined

? props.errors.customer\_email[0]

: null

}

{...customer\_email}

style={{ width: "200px" }}

placeholder="введите почту..."

></TextField>

</div>

<div className={"text-block-input"}>

<p>Телефон</p>

<TextField

error={props.errors.customer\_phone != undefined}

helperText={

props.errors.customer\_phone != undefined

? props.errors.customer\_phone[0]

: null

}

{...customer\_phone}

style={{ width: "250px" }}

placeholder="введите номер телефона..."

></TextField>

</div>

</div>

</div>

<div>

<div>

<h4 style={{ textDecoration: "underLine" }}>

Укажите информацию о заказе

</h4>

</div>

<div className={"input-info-block"}>

<div className={"text-block-input"}>

<p>Выберите услугу</p>

<select

onChange={(e) => {

setAmenity(e.target.value);

}}

>

{props.amenities.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.amenity != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.amenity[1]}

</FormHelperText>

) : null}

</div>

<div className={"text-block-input"}>

<p>Выберите тариф</p>

<select

onChange={(e) => {

setTariff(e.target.value);

}}

>

{props.tariffs.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.tariff != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.tariff[1]}

</FormHelperText>

) : null}

</div>

</div>

<div className={"input-info-block"}>

<div className={"text-block-input"}>

<p>Адрес начала заказа</p>

<TextField

error={props.errors.startPoint != undefined}

helperText={

props.errors.startPoint != undefined

? props.errors.startPoint[0]

: null

}

{...startPoint}

style={{ width: "200px" }}

placeholder="введите адрес начала..."

></TextField>

</div>

<div className={"text-block-input"}>

<p>Адрес завершения заказа</p>

<TextField

error={props.errors.finishPoint != undefined}

helperText={

props.errors.finishPoint != undefined

? props.errors.finishPoint[0]

: null

}

{...finishPoint}

style={{ width: "240px" }}

placeholder="введите адрес завершения..."

></TextField>

</div>

</div>

<div className={"input-info-block"}>

<div className={"text-block-input"}>

<p>Предполагаемый вес груза</p>

<TextField

error={props.errors.weight != undefined}

helperText={

props.errors.weight != undefined ? props.errors.weight[0] : null

}

{...weight}

style={{ width: "200px" }}

placeholder="введите вес груза..."

></TextField>

</div>

<div className={"text-block-input"}>

<p>Предполагаемая высота груза</p>

<TextField

error={props.errors.height != undefined}

helperText={

props.errors.height != undefined ? props.errors.height[0] : null

}

{...height}

style={{ width: "270px" }}

placeholder="введите максимальную высоту..."

></TextField>

</div>

</div>

<div className={"input-info-block"}>

<div className={"text-block-input"}>

<p>Выберите машину</p>

<select

onChange={(e) => {

setCar(e.target.value);

}}

>

{props.cars.map((el) => (

<option key={el.id} value={el.id}>

{el.name}

</option>

))}

</select>

{props.errors.car != undefined ? (

<FormHelperText style={{ color: "red" }}>

{props.errors.car[1]}

</FormHelperText>

) : null}

</div>

<div className={"text-block-input"}>

<p>Дополнительная информация</p>

<TextField

error={props.errors.note != undefined}

helperText={

props.errors.note != undefined ? props.errors.note[0] : null

}

{...note}

style={{ width: "300px" }}

placeholder="укажите уточняющую информацию..."

></TextField>

</div>

</div>

</div>

<button onClick={createOrder} className="order-button">

оформить заявку

</button>

</div>

);

};

export const useInputText = (initialState) => {

const [value, setValue] = useState(initialState);

const handleChange = (e) => setValue(e.target.value);

return {

value,

onChange: handleChange,

};

};

export default connect(

(state) => ({

amenities: state.amenities,

tariffs: state.tariffs,

cars: state.cars,

errors: state.errors,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

post: (obj, path, setter) => dispatch(post(obj, path, setter)),

})

)(OrderCreation);

import React from "react";

import ReactDOM from "react-dom";

import \* as serviceWorker from "../serviceWorker";

import { BrowserRouter } from "react-router-dom";

import { createStore, applyMiddleware } from "redux";

import { Provider } from "react-redux";

import reducer from "../main\_redux";

import thunk from "redux-thunk";

import App from "../App";

let store = createStore(reducer, applyMiddleware(thunk));

class Root extends React.Component {

render() {

return (

<Provider store={store}>

<BrowserRouter>

<App/>

</BrowserRouter>

</Provider>

);

}

}

serviceWorker.unregister();

export default Root;

import React, { useEffect } from 'react'

import MaterialTable from "material-table";

import AdminTopPanel from './AdminTopPanel';

import { connect } from "react-redux";

import { indexAmenity, indexCar, indexTariff } from '../main\_redux/actions';

import { get, drop, post, createReport } from '../server\_connections';

import { tableIcons } from '../table\_icons';

import './style.css'

import { useInputText } from './OrderCreation';

import { TextField, Tooltip } from '@material-ui/core';

import exportFile from '../icons/file.svg'

const TariffAdminPage = (props) => {

const columns = [

{ title: "Название", field: "name" },

]

useEffect(() => {

props.set("tariffs", indexTariff);

}, []);

const edits = {

onRowDelete: (oldData) => new Promise((resolve) => {

props.drop(oldData.id, 'tariffs', indexTariff)

resolve();

})

}

const name = useInputText("");

const newOrder = {

tariff: {

name: name.value

},

};

const createOrder = () => {

props.post(newOrder, "tariffs", indexTariff);

};

const createReport = () => {

props.createReport('tariffs\_report')

}

return(

<div>

<AdminTopPanel>

<div className={'work-space'}>

<div className='admin-manage-table'>

<MaterialTable

icons={tableIcons}

title={

<div>

Тарифы

<Tooltip title="Экспорт в файл">

<img style={{width: '25px', marginLeft: '20px', marginBottom: '-5px'}}

src={exportFile}

onClick={createReport}

alt={'экспортировать в файл'}/>

</Tooltip>

</div>

}

columns={columns}

data={props.tariffs}

editable={edits}

/>

</div>

<div className='admin-manage-panel'>

<h3>Добавление тарифа</h3>

<div>

<div className="admin-add-panel">

<div>

<TextField

error={props.errors.name != undefined}

helperText={

props.errors.name != undefined ? props.errors.name[0] : null

}

{...name} placeholder='название'/>

</div>

<div>

<button onClick={createOrder} style={{width: '80%', marginBottom: '20px'}} className="order-button">

Добавить

</button>

</div>

</div>

</div>

</div>

</div>

</AdminTopPanel>

</div>

)

}

export default connect(

(state) => ({

tariffs: state.tariffs,

errors: state.errors,

}),

(dispatch) => ({

set: (path, setter) => dispatch(get(path, setter)),

drop: (id, path, setter) => dispatch(drop(id, path, setter)),

post: (obj, path, setter) => dispatch(post(obj, path, setter)),

createReport: (path) => dispatch(createReport(path))

})

)(TariffAdminPage);

Приложение Б

(справочное)

Формы выходных документов

Данный проект создает выходные текстовые документы, содержащий данные обо всех соответствующих сущностях. На рисунке Б.1 представлен текстовый документ сущности базы данных «Заказы».

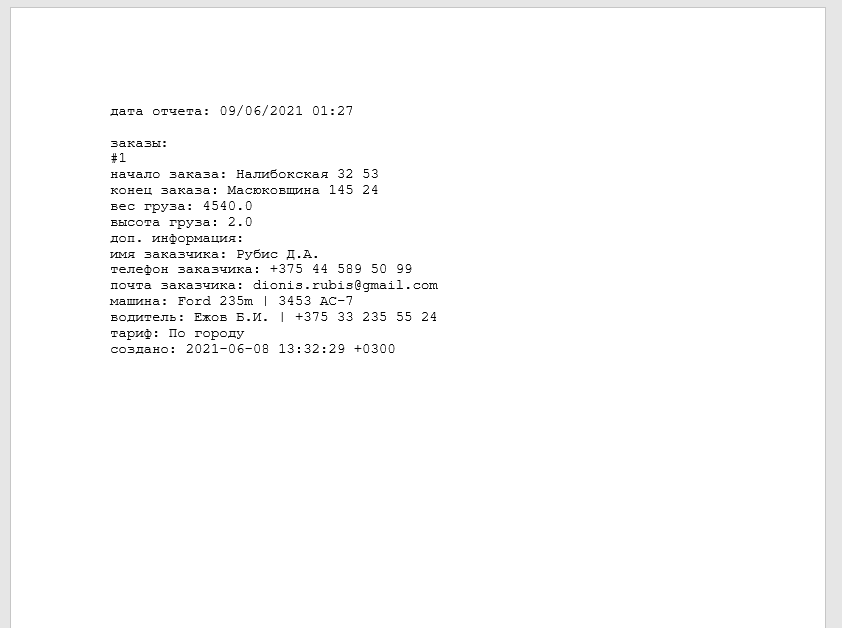


Рисунок Б.1 − Текстовый документ сущности «Заказы»