Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора

по учебной работе

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

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021

РАЗРАБОТКА САЙТА ДЛЯ ОНЛАЙН-МАГАЗИНА

ЭЛЕКТРОНИКИ «I-BOZH»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.795009.401

Председатель цикловой комиссии ( Т.Г. Багласова )

Руководитель проекта ( Д.Ф. Клименко )

Консультант по экономической части ( М.А. Григораш )

Консультант по охране труда ( В.С. Кудласевич )

Учащейся ( Е.Ю. Божкова )

Рецензент ( )

2021

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ДП Т.795009.401 ПЗ

Разраб.

Божкова Е. Ю.

Провер.

*Клименко Д.Ф.*

Т. Контр.

*Баглагова Т.Г.*

Н. Контр.

*Басалыга Л.В.*

Утверд.

*Баглагова Т.Г.*

*Разработка сайта для онлайн-магазина электроники «i-Bozh»*

Лит.

Листов

КБП

109

у

[Введение 4](#_Toc74763015)

[1 Объектно-ориентированный анализ и проектирование приложения 6](#_Toc74763016)

[1.1 Назначение и цели создания Web-приложения 6](#_Toc74763017)

[1.2 Проектирование модели 7](#_Toc74763018)

[2 Проектирование Web-приложения 11](#_Toc74763019)

[2.1 Требования к Web-приложению 11](#_Toc74763020)

[2.2 Структура Web-приложения 11](#_Toc74763021)

[2.3 Проектирование макета Web-приложения 12](#_Toc74763022)

[2.4 Программно-технические средства, необходимые для разработки приложения 13](#_Toc74763023)

[2.5 Защита и сохранность данных 14](#_Toc74763024)

[2.6 Организация и ведение информационной базы 14](#_Toc74763025)

[3 Реализация Web-приложения 18](#_Toc74763026)

[3.1 Разработка административной части приложения 18](#_Toc74763027)

[3.2 Разработка клиентской части приложения 18](#_Toc74763028)

[3.3 Описание используемых функций и процедур 19](#_Toc74763029)

[4 Описание Web-приложения 21](#_Toc74763030)

[4.1 Общие сведения 21](#_Toc74763031)

[4.2 Функциональное назначение 21](#_Toc74763032)

[4.3 Описание разделов Web-приложения 22](#_Toc74763033)

[5 Методика испытаний 23](#_Toc74763034)

[5.1 Технические требования 23](#_Toc74763035)

[5.2 Функциональное тестирование 23](#_Toc74763036)

[6 Применение 37](#_Toc74763037)

[6.1 Назначение программы 37](#_Toc74763038)

[6.2 Программно-аппаратное обеспечение сервера и клиента 37](#_Toc74763039)

[6.3 Справочная система 37](#_Toc74763040)

[7 Охрана труда и окружающей среды на предприятии 39](#_Toc74763041)

[7.1 Правовые, нормативные, социально-экономические и организационные вопросы](#_Toc74763042)

[охраны труда 39](#_Toc74763042)

[7.2 Обеспечение оптимального газового и ионного воздуха в рабочей зоне помещения 41](#_Toc74763043)

[7.3 Пожарная безопасность 43](#_Toc74763044)

[7.4 Охрана окружающей среды 45](#_Toc74763045)

[8 Экономический раздел 47](#_Toc74763046)

[8.1 Технико-экономическое обоснование разработки программного средства 47](#_Toc74763047)

[8.2 Составление плана по разработке программного средства 47](#_Toc74763048)

[8.3 Определение цены программного средства 47](#_Toc74763049)

[8.4 Экономическая эффективность разработки 51](#_Toc74763050)

[Заключение 52](#_Toc74763051)

[Список информационных источников 53](#_Toc74763052)

[Приложение А](#_Toc74763053) – [Текст программы 55](#_Toc74763055)

[Приложение Б](#_Toc74763056) – [Формы выходных документов 108](#_Toc74763058)

# Введение

Интернет магазин электроники – это сайт, позволяющий осуществлять выбор, заказ и оплату электроники и аксессуаров онлайн. Иными словами, это магазин, который осуществляет торговлю товарами через интернет.

В интернете появились удобные и недорогие магазины, и их услугами пользуется все больше и больше людей. Интернет-магазины − это торговые сайты компаний малого и среднего бизнеса. В интернет-магазине электроники могут быть реализованы практически любые схемы: торговля со склада и на заказ, торговля с частными лицами и с организациями, торговля вещественными и цифровыми товарами, услугами, информацией и т.д.

На сайте магазина обычно представлен подробный каталог товаров с ценами, на основе которого пользователь формирует свой заказ. Заказывая товары в интернет-магазине, вы можете получать их по почте или с курьером, а оплачивать непосредственно при получении или также через Интернет.

Преимущества интернет-магазина электроники:

* удобство выбора товара из каталога;
* возможность оформления предзаказов;
* быстрая и легкая оплата;
* возможность оформления доставки товаров;
* круглосуточная работа без перерывов.

Целью разработки проекта на тему «Разработка сайта для онлайн-магазин электроники «i- Bozh»» является создание сайта, который будет осуществлять ведение базы данных, содержащей информацию об электронике, поставщиках, поставках, клиентах и их заказах. Для достижения цели дипломного проекта нужно решить следующие задачи:

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

В состав пояснительной записки будут входить восемь разделов, содержащие необходимую информацию по организации и эксплуатации Web-приложения.

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:

* лист 1 – диаграмма вариантов использования;
* лист 2 – диаграмма деятельности;
* лист 3 – диаграмма развёртывания;
* лист 4 – структура Web-приложения.

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

Предметной областью данного дипломного проекта является интернет-магазин электроники.

Интернет-магазин – сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ [21].

В услуги интернет-магазина включаются:

* взаимодействия с каталогом комплектующих;
* взаимодействие с заказом.

Каждое комплектующее обладает следующими характеристиками:

* конкретное название;
* год производства;
* бренд;
* тип;
* модель;
* срок гарантии;
* поставщик;
* количество на складе;
* цена.

В интернет-магазине будут присутствуют следующие должности: 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 [23].

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 [20].

Каталог – обозначает перечень товаров с их описанием [22].

Каталог включает в себя: список товаров и их категорий.

Заказ – поручение на изготовление чего-либо или оказание услуги. Заказ состоит из электроники и аксессуаров, заказанной заказчиком [19].

Учет заказов в интернет-магазине происходит следующим образом:

* заказ добавляет администратор;
* добавляется в базу заказов;
* исполняется или доставляется.

Склад – это территория, помещение, предназначенное для хранения материальных ценностей и оказания складских услуг [26].

На складе находится сырье, поступающее на предприятие розничных продаж, которое хранится в складских помещениях.

Складское хозяйство в интернет-магазине выполняет следующие функции:

* просмотр текущей насыщенности склада;
* добавление новых продуктов;
* редактирование комплектующих, лежащих на складе;
* удаление проданных комплектующих;

Существует две роли − администратор и клиент.

Администратор приложения может:

* редактировать, добавлять, удалять, просматривать каталог;
* редактировать, добавлять, удалять, просматривать информацию о товаре на складе;
* добавлять, удалять, просматривать список поставщиков и поставок;
* добавлять, удалять, просматривать список заказов;
* производить поиск по названию и цене товара.

Клиент приложения может:

* просматривать каталог;
* просматривать информацию об определённом товаре;
* редактировать, добавлять, удалять, просматривать список заказов;
* производить поиск по названию и цене товара.

На основе изученной предметной области приложение должно выполнять следующие функции:

* ведение единой базы данных, содержащей информацию о клиентах, товарах, заказах, поставках и поставщиках;
* поиск по названию товара;
* сортировка по цене товара;
* фильтрация по бренду и категории товара;
* формирование выходного документа Excel, который будет содержать данные обо всех оформленных заказах в разрезе дат − Report.xlsx;
* формирование выходного документа Excel, который будет содержать данные об оформленном заказе − CheckOnline.xlsx.

Данное приложение является небольшим аналогом сайта «5 элемент», которое также предназначено для розничных продаж электроники в интернете. Но из-за того, что все они выполняются по индивидуальному заказу, провести оценку и сравнение невозможно.

1. 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 программного средства можно выделить следующие сущности: «Заказ», «Клиент», «Продукт», «Поставка», «Поставщик», «Бренд», «Категория», «ПродуктЗаказИнфо», «Фото». Диаграмма «Сущность-связь» представлена на рисунке 1.1.

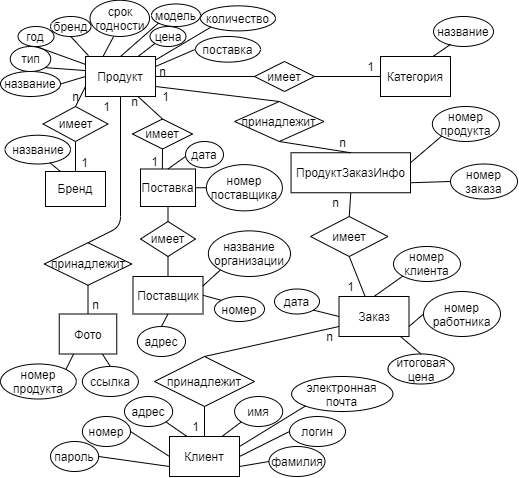


Рисунок 1.1 − Диаграмма «Сущность-связь»

Для сущности «Категория» атрибутом будет являться: название.

Для сущности «Поставщик» атрибутами будут являться:

* название организации;
* номер;
* адрес.

Для сущности «Поставка» атрибутами будут являться:

* номер поставщика;
* дата.

Для сущности «Бренд» атрибутом будет являться: название.

Для сущности «Продукт» атрибутами будут являться:

* название;
* год;
* бренд;
* тип;
* модель;
* срок годности;
* цена;
* количество;
* поставка.

Для сущности «Клиент» атрибутами будут являться:

* имя;
* фамилия;
* адрес;
* номер;
* логин;
* пароль;
* электронная почта.

Для сущности «Заказ» атрибутами будут являться:

* дата;
* номер работника;
* номер клиента;
* итоговая цена.

Для сущности «ПродуктЗаказИнфо» атрибутами будут являться:

* номер продукта;
* номер заказа.

Для сущности «Фото» атрибутами будут являться:

* ссылка;
* номер продукта.

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 . На данной диаграмме имеются два «актёр» − администратор и клиент [25].

Администратор и клиент имеют такие варианты использования как:

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

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

Данная диаграмма представлена 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 представлена 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 MSSQL;
* js 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 программное средство будет представлять собой Web-приложение, которое подразумевается использовать по сети Интернет.

Программное приложение будет находиться на сервере, и доступ к нему будут иметь все пользователи. Для каждой группы пользователей предусмотрено разграничение прав доступа к приложению. Администратор приложения может осуществлять все действия доступные в приложении, добавление, изменение и удаление любой информации в приложении. Пользователи приложения может просматривать и осуществлять заказы на своё имя.

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

Визуальное оформление Web-приложения 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.

Web-приложение 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 Web-приложения должна производиться в рамках идеологии разделения структуры и представления. JS-библиотека − React.js будет использоваться только для описания структуры документа, в то время как управление внешним видом Web-страниц должно осуществляться с помощью каскадных таблиц стилей CSS.

1. Структура Web-приложения

Web-приложение представляет собой набор React-компонентов, таблиц стилей, которые могут объединяться и выполняться в различных браузерах.

Общая структура сайта приведена на рисунке 2.1

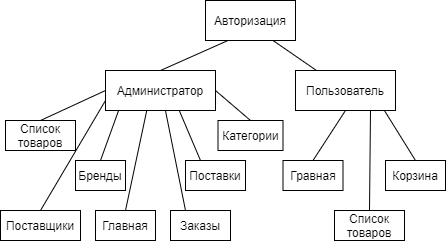


Рисунок 2.1 – Общая структура приложения

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 [12].

Дерево каталогов и файлов Web-приложения представлено на рисунке 2.2

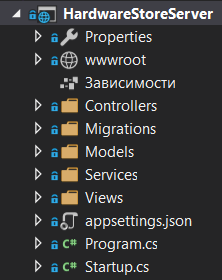


Рисунок 2.2 – Дерево каталогов

Описание структуры дерева каталогов:

* «HardwareStoreServer» − главная папка приложения;
* «Properties» и «wwwroot» − папки, содержащая различные настройки приложения;
* «Controllers» − папка, содержащая контроллеры;
* «Migrations» − папка, содержащая файлы необходимые для связи с базой данных;
* «Models» − папка, содержащая модели;
* «Services» − папка, содержащая логику контроллеров;
* «Views» − папка, содержащая представления.

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, и содержимое для контента.

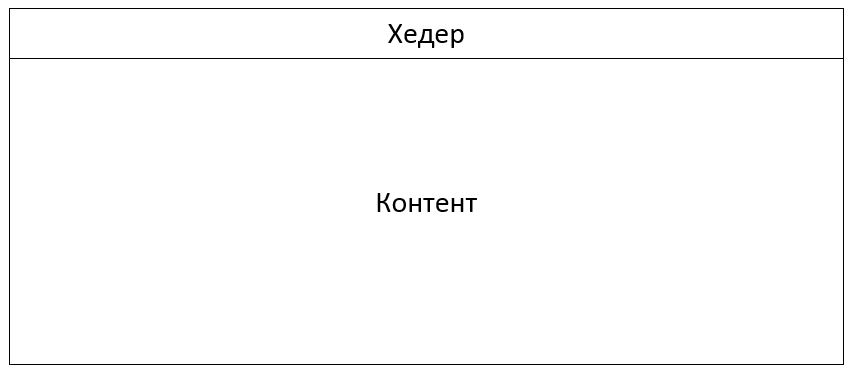


Рисунок 2.3 − Макет Web-приложения

1. Программно-технические средства, необходимые для разработки приложения

Верстка страниц веб-сайта будет производиться с помощью React-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 JavaScript. Они принимают произвольные входные данные (так называемые «пропсы») и возвращают React-элементы, описывающие, что мы хотим увидеть на экране.

При разработке приложения будут использоваться следующие технологии.

JavaScript − мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам [9].

React (иногда React.js или ReactJS) − JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций.

React может использоваться для разработки одностраничных и мобильных приложений. Его цель − предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL [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 [6].

jQuery − набор функций JavaScript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX [10].

Microsoft SQL Server − система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов − Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка [11].

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») − схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер − таким образом, что модификация каждого компонента может осуществляться независимо [12].

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

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 Web-приложение пользователю необходимо ввести логин и пароль.

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 [8].

Таблица «Types» хранит информацию о категориях товара, структура которой приведена в таблице 2.1.

Таблица 2.1 – Структура таблицы «Types»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер категории |
| name | NVARCHAR | 255 | Название категории |

Таблица «Brands» хранит информацию о бренде товара, структура которой приведена в таблице 2.2.

Таблица 2.2 – Структура таблицы «Brands»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер бренда |
| name | NVARCHAR | 255 | Название бренда |

Таблица «Suppliers» хранит информацию о поставщиках, структура которой приведена в таблице 2.3.

Таблица 2.3 – Структура таблицы «Suppliers»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер поставщика |
| nameOrganization | NVARCHAR | 255 | Название организации |
| number | NVARCHAR | 255 | Номер телефона |
| adress | NVARCHAR | 255 | Адрес организации |

Таблица «Supplies» хранит информацию о поставках, структура которой приведена в таблице 2.4.

Таблица 2.4 – Структура таблицы «Supplies»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер поставки |
| supplierId | INTEGER | 4 | Номер поставщика |
| Date | DATE | 3 | Дата поставки |

Таблица «Products» хранит информацию о товарах электроники, структура приведена в таблице 2.5.

Таблица 2.5 – Структура таблицы «Products»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер продукта |
| name | NVARCHAR | 255 | Название продукта |
| year | NVARCHAR | 4 | Год выпуска |
| brandId | INTEGER | 4 | Номер бренда |
| typeId | INTEGER | 4 | Номер категории |
| modal | NVARCHAR | 255 | Модель |
| warranty | INTEGER | 4 | Срок гарантии |
| price | DECIMAL | 12 | Цена |
| amount | INTEGER | 4 | Количество на складе |
| supplyId | INTEGER | 4 | Номер поставщика |

Таблица «Clients» хранит информацию о клиентах, структура которой приведена в таблице 2.6.

Таблица 2.6 – Структура таблицы «Clients»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер клиента |
| name | NVARCHAR | 255 | Имя клиента |
| surname | NVARCHAR | 255 | Фамилия клиента |
| adress | NVARCHAR | 255 | Адрес клиента |
| number | NVARCHAR | 255 | Номер телефона |
| email | NVARCHAR | 255 | Электронная почта клиента |
| login | NVARCHAR | 255 | Логин клиента |
| password | NVARCHAR | 255 | Пароль клиента |

Таблица «Orders» хранит информацию о заказах, структура которой приведена в таблице 2.7.

Таблица 2.7 – Структура таблицы «Orders»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер заказа |
| clientId | INTEGER | 4 | Номер клиента |
| date | DATE | 3 | Дата заказа |
| totalPrice | DECIMAL | 12 | Сумма заказа |

Таблица «ProductOrderInfos» хранит информацию о продуктах в заказе, структура которой приведена в таблице 2.8.

Таблица 2.8 – Структура таблицы «ProductOrderInfos»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер продукта в заказе |
| productId | INTEGER | 4 | Номер продукта |
| quantity | INTEGER | 4 | Количество продукта |
| orderId | INTEGER | 4 | Номер заказа |

Таблица «Images» хранит информацию о фото продукта, структура которой приведена в таблице 2.9.

Таблица 2.9 – Структура таблицы «Images»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| id | INTEGER | 4 | Номер фото продукта |
| url | NVARCHAR | 255 | Ссылка на фото |
| productId | INTEGER | 4 | Номер продукта |

Схема базы данных представлена на рисунке 2.4.

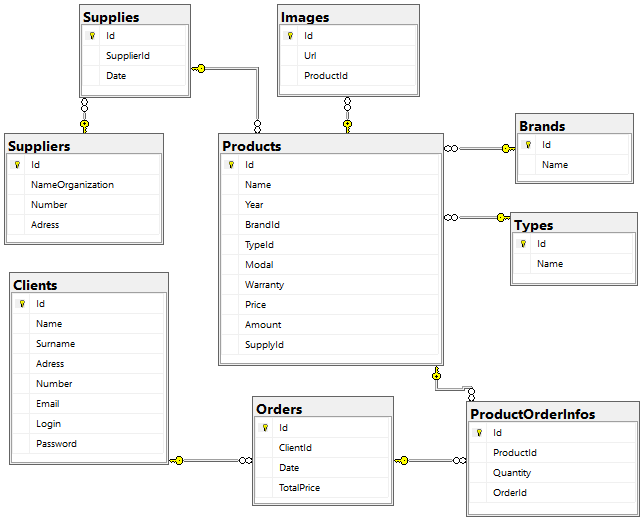


Рисунок 2.4 − Схема базы данных

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

При разработке административной части Web-приложения, его главная и все второстепенные страницы будут находиться в корневом каталоге «src/components», где файл «index.js» будет заглавным файлом (так определено требованиями хостинг-провайдера).

Описание физической структуры административной части Web-приложения:

* «HardwareStoreServer/Controllers/ApiControllers/BrandController.cs» − контроллер брендов;
* «HardwareStoreServer/Controllers/ApiControllers/ClientController.cs» − контроллер клиентов;
* «HardwareStoreServer/Controllers/ApiControllers/OrderController.cs» − контроллер заказов;
* «HardwareStoreServer/Controllers/ApiControllers/ProductController.cs» − контроллер продуктов (товаров);
* «HardwareStoreServer/Controllers/ApiControllers/ProductOrderInfoController.cs» − контроллер продуктов (товаров), находящиеся в заказе;
* «HardwareStoreServer/Controllers/ApiControllers/SupplierController.cs» − контроллер поставщиков;
* «HardwareStoreServer/Controllers/ApiControllers/SupplyController.cs» − контроллер поставок;
* «HardwareStoreServer/Controllers/ApiControllers/TypeController.cs» − контроллер категорий.

Описание логической структуры административной части Web-приложения:

«Главная» («<a href="/admin/home">») – относительная ссылка на выход из формы авторизация;

«Список товаров» («<a href="/admin /products">») – относительная ссылка на страницу со списком продуктов (товаров);

«Бренды» («<a href="/admin /brands">») − относительная ссылка на страницу с брендами;

«Категории» («<a href="/admin /types">») – относительная ссылка на страницу с категориями;

«Поставки» («<a href="/admin/supplies">») – относительная ссылка на страницу с поставками;

«Поставщики» («<a href="/admin/suppliers">») – относительная ссылка на страницу с поставщиками;

«Заказы» («<a href="/admin/orders">») – относительная ссылка на страницу с оформленными заказами;

«Помощь» («<a href="/admin /help">») – относительная ссылка на страницу с информацией о Web-приложении и его функциях;

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

1. Разработка клиентской части приложения

При разработке клиентской части Web-приложения, его главная и все второстепенные страницы будут находиться в корневом каталоге «src/components», где файл «index.js» будет заглавным файлом (так определено требованиями хостинг-провайдера).

Физическая структура клиенткой части Web-приложения 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 административной части Web-приложения:

«Главная» («<a href="/username/home">») – относительная ссылка на выход из формы авторизация;

«Список товаров» («<a href="/username/products">») – относительная ссылка на страницу со списком продуктов (товаров);

«Корзина» («<a href="/username/shoppingCart">») – относительная ссылка на страницу с продуктами (товарами) добавленные в корзинку;

«Помощь» («<a href="/username/help">») – относительная ссылка на страницу с информацией о Web-приложении и его функциях;

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

1. Описание используемых функций и процедур

Основными функциями программного средства будут являться добавление и изменение данных. Добавление и редактирование 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 программное средство − это ведение данных в базе данных, добавление новых данных, их редактирование, 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 handleSubmit, который обрабатывает заполненные поля формы и отправит Http запрос. Код реализации данной функции представлен ниже.

const handleSubmit = (event) => {

event.preventDefault();

axios.post(`https://localhost:5001/api/Type/create?${qs.stringify({

Name: event.target.name.value

})}`)

.then(res => { setSnackBaropen(true); setSnackBarMessage('Успешно добавлено'); dispatch(fetchTypes()); })

.catch(error => { setSnackBaropen(true); setSnackBarMessage('Ошибка добавления') });

}

Для редактирования элемента в любую сущность данного приложения, необходимо будет перейти на нужную модель сущности и нажать на кнопку «Редактировать». После нажатия на кнопку «Редактировать» вызовется обработчик события handleSubmit, который обрабатывает изменённые поля формы и отправляет Http запрос. Код реализации данной функции представлен ниже.

const handleSubmit = (event) => {

event.preventDefault();

axios.put(`https://localhost:5001/api/Type/edit?${qs.stringify({

Id: typeid,

Name: event.target.name.value

})}`)

.then(res => { setSnackBaropen(true); setSnackBarMessage('Успешно обновлён'); dispatch(fetchTypes()); })

.catch(error => { setSnackBaropen(true); setSnackBarMessage('Ошибка редактирования') });

}

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 отобразит диалоговое окно, после соглашения с которым, отправляет HTTP запрос. Код реализации данной функции представлен ниже.

const deleteType = (id) => {

if (window.confirm('Вы уверены?')) {

axios.delete(`https://localhost:5001/api/Type/delete/${id}`)

.then(res => dispatch(fetchTypes()))

.catch(error => console.log(error));

}

}

Полный текст программы представлен в приложении А.

1. Описание Web-приложения
2. Общие сведения

Программное средство «Онлайн-магазин электроники «i-Bozh»» выполнено в форме Web-приложения, и используется как администратором магазина, так и клиентами.

В Web-приложении существую разделение прав, и помощью чего реализовано разделений функций и действий для любого пользователя приложения, 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. Во время запуска приложения по умолчанию первым на сервере обрабатывается модуль «Startup.cs».

В разработанном программном средстве 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. Доступ к приложению возможен только после авторизации.

1. Функциональное назначение

Программное средство предназначено для автоматизации работы магазина электроники.

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

В разработанном программном средстве 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 Web-приложения представлена следующими группами пользователей:

* администратор;
* клиенты.

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

* просмотр, добавление, редактирование и удаление товара-электроники;
* просмотр, добавление, редактирование и удаление информации о брендах товаров;
* просмотр, добавление, редактирование и удаление информации о категориях товаров;
* просмотр, добавление, редактирование и удаление информации о поставщиках;
* просмотр, добавление, редактирование и удаление информации о поставках;
* способность сортировки списка товаров по цене;
* способность фильтрации списка товаров по брендам и категориям.

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

* просмотр товаров-элетроники;
* способность добавлять товар-электронику в корзину;
* просматривать личную корзину и итоговую стоимость предварительного заказа;
* способность сортировки списка товаров по цене;
* способность фильтрации списка товаров по брендам и категориям.

Разработанное программное средство 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: «Главная», «Список товаров», «Бренды», «Категории», «Поставки», «Поставщики», «?», «Выход».

Пункт меню «Главная» − выполняется переход на главную страницу приложения. Приветствует пользователя.

Пункт меню «Список товаров» − выполняется переход на страницу для просмотра и работы со всем списком товаров.

Пункт меню «Бренды» − выполняется переход на страницу для просмотра и работы со списком существующих брендов товаров.

Пункт меню «Категории» − выполняется переход на страницу для просмотра и работы со списком существующих категорий товаров.

Пункт меню «Поставки» − выполняется переход на страницу для просмотра и работы со списком принятых поставок.

Пункт меню «Поставщики» − выполняется переход на страницу для просмотра и ведения учёта информации о поставщиках.

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

Пункт меню «?» − выполняется переход на страницу справочной системы.

Пункт меню «Выход» − выполняется переход на страницу авторизации.

При входе в систему под пользователем, имеющем роль «Клиент», навигационное меню состоит из следующих пунктов: «Главная», «Список товаров», «Корзина», «?», «Выход».

Пункт меню «Главная» − выполняется переход на главную страницу приложения. Приветствует пользователя.

Пункт меню «Список товаров» − выполняется переход на страницу для просмотра всего списка товаров.

Пункт меню «Корзина» − выполняется переход на страницу для просмотра и работы со всем списком товаров добавленных в корзину, способность составлять чек заказа.

Пункт меню «?» − выполняется переход на страницу справочной системы.

Пункт меню «Выход» − выполняется переход на страницу авторизации.

1. Методика испытаний
2. Технические требования

Web-сайт корректно отображается в следующих браузерах:

* Mozilla Firefox (версия 2.0 и выше);
* Google Chrome (версия 5.0 и выше);
* Opera (версия 9.0 и выше).

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

К конфигурации Web-сервера предъявляются следующие технические требования:

* npm − менеджер пакетов, входящий в состав Node.js;
* сервер баз данных Microsoft SQL Server.

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 [27].

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.

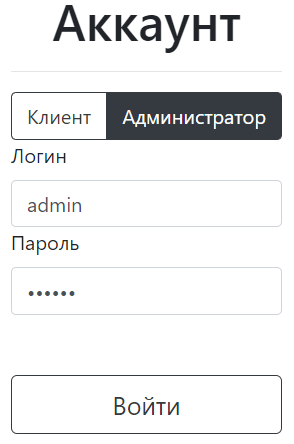


Рисунок 5.1 − Окно авторизации

Таблица 5.1 − Тест-кейс функции авторизации

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

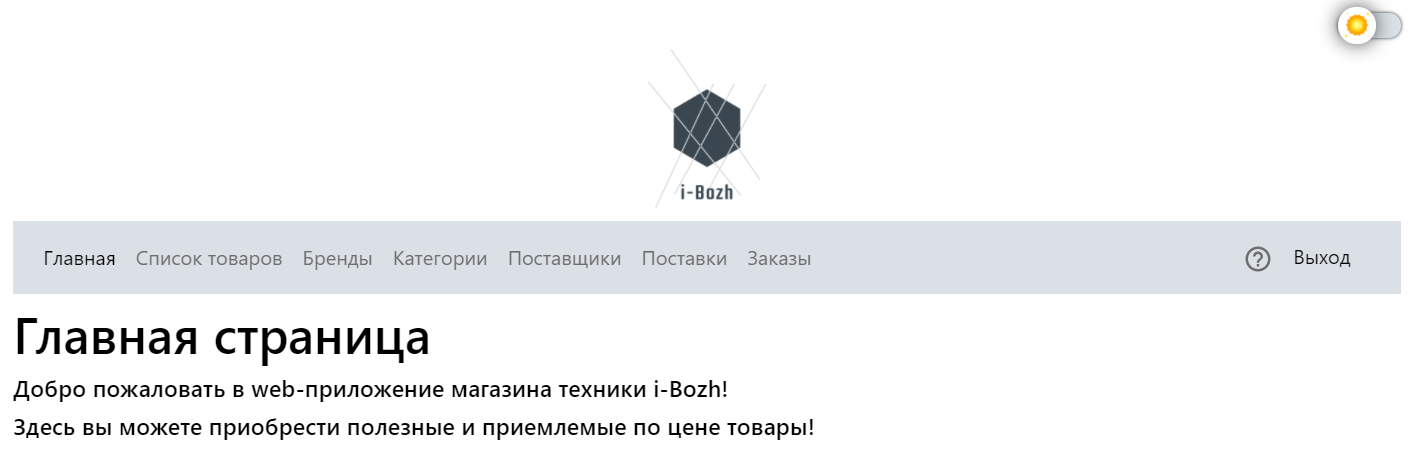


Рисунок 5.2 − Результат авторизации

Второй тест будет проведён на странице «Список товаров».

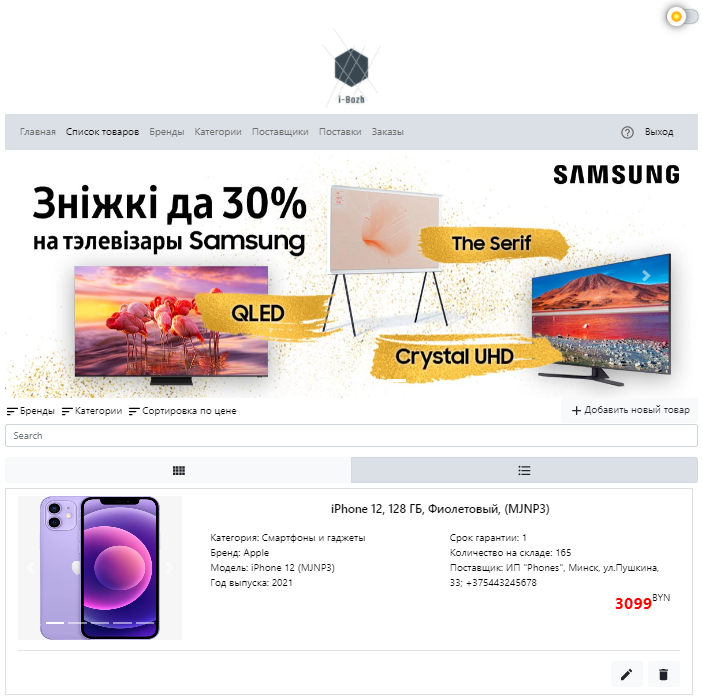


Рисунок 5.3 − Страница «Список товаров»

Таблица 5.2 − Тест-кейс функции добавления продукта (товара)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 2 | Добавление продукта в список товаров в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Нажать на кнопку «Добавить новый товар»;  3. В открывшемся окне заполнить поля:  − «Выбрать картинку для товара» − выбор картинок с проводника  − «Название » − iPhone 12 Pro Max, 256 ГБ, Золотой, (MGDE3)  − «Категория» − Смартфоны и гаджеты  − «Бренд» − Apple  − «Модель» − iPhone 12 Max, (MGDE3)  − «Год выпуска» − 2020  − «Срок гарантии» − 1  − «Количество на складе» − 53  − «Поставщик» − ИП «Phones»  − «Цена» − 4399  4. Нажать на кнопку «Добавить товар». | Ожидаемый результат:  Произойдёт добавление продукта в базу  Фактический результат:  Все поля заполнены, представлено на рисунке 5.4  Результат: появление нового продукта в списке и отображение сообщения «Успешно добавлено», представлен на рисунке 5.5 |

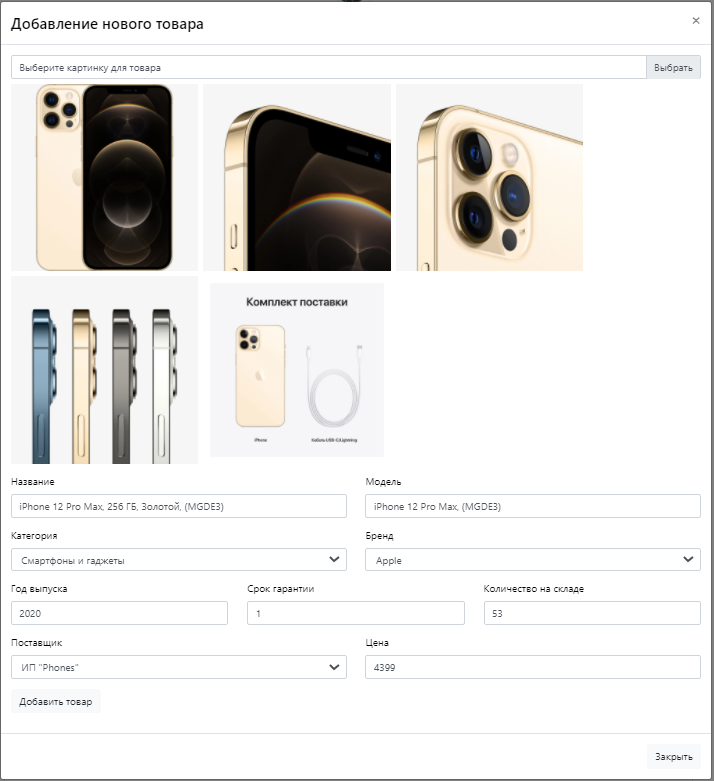


Рисунок 5.4 − Окно добавления продукта (товара)

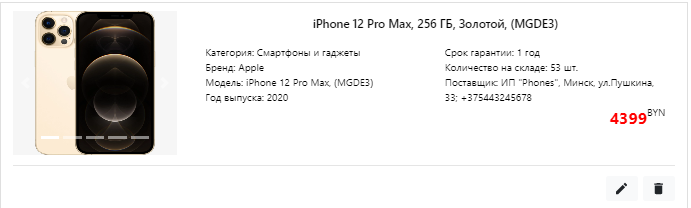


Рисунок 5.5 − Результат добавления продукта (товара)

Третий тест будет проведён также на странице «Список товаров».

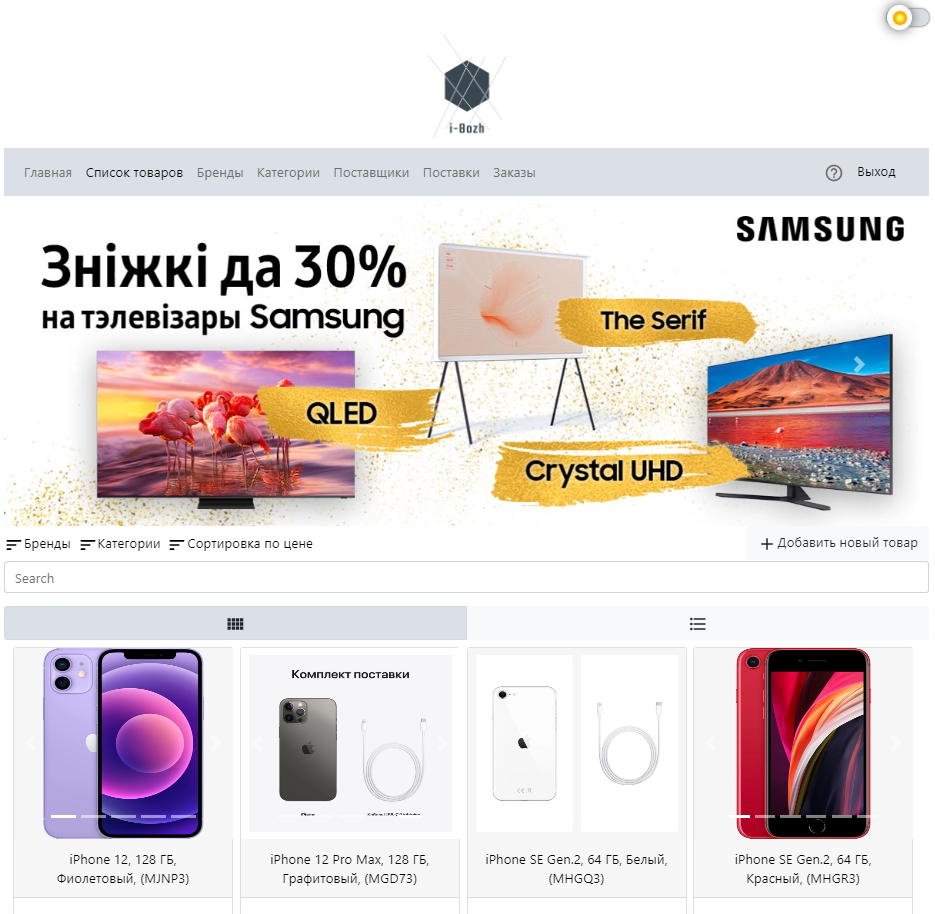


Рисунок 5.6 − Страница «Список товаров»

Таблица 5.3 − Тест-кейс функции изменения данных о продукте (товаре)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 3 | Редактирование продукта в списке товаров в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Нажать на кнопку соответствующего продукта « »;  3. В открывшемся окне заполнить поля:  − «Выбрать картинку для товара» − выбор картинок с проводника  − «Название » − Huawei P40 lite (розовая сакура)  − «Категория» − Смартфоны и гаджеты  − «Бренд» − Huawei  − «Модель» − P40 lite  − «Год выпуска» − 2020  − «Срок гарантии» − 1  − «Количество на складе» − 120  − «Поставщик» − ОАО «ТехНадо»  − «Цена» − 699  4. Нажать на кнопку «Изменить товар». | Ожидаемый результат:  Произойдёт изменение данных о продукте и обновление их в базе  Фактический результат:  Все поля заполнены, представлено на рисунке 5.7  Результат: появление изменённого продукта в списке и отображение сообщения «Updated successfully», представлен на рисунке 5.8 |

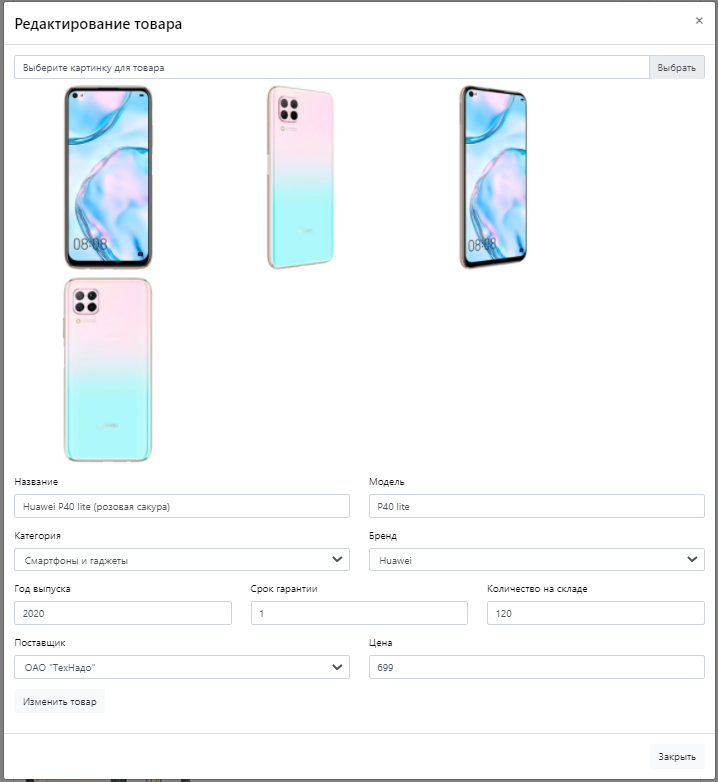


Рисунок 5.7 − Окно изменений данных о продукте (товаре)



Рисунок 5.8 − Результат изменения данных о продукте (товаре)

Четвёртый тест будет проведён также на странице «Список товаров».

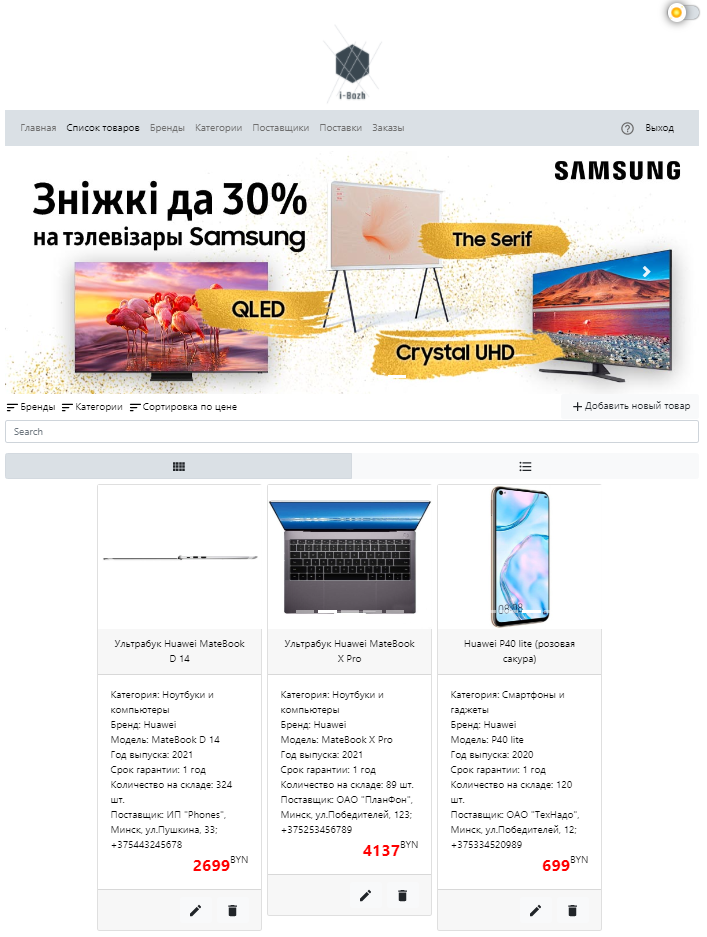


Рисунок 5.9 − Страница «Список товаров»

Таблица 5.4 − Тест-кейс функции удаления продукта (товара) со списка

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 4 | Удаление продукта со списка товаров в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Нажать на кнопку соответствующего товара «»;  3. В отобразившемся диалоговом окне нажать кнопку «ОК». | Ожидаемый результат:  Произойдёт удаление продукта со списка и удаление его из базы  Фактический результат:  Кнопка нажата, представлено на рисунке 5.10  Результат: удаление продукта со списка, представлено на рисунке 5.11 |

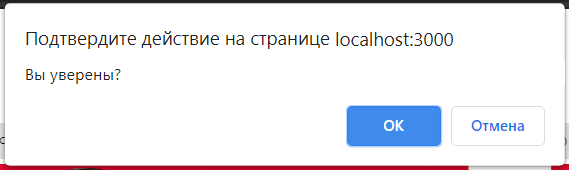


Рисунок 5.10 − Диалоговой окно для удаления продукта (товара)

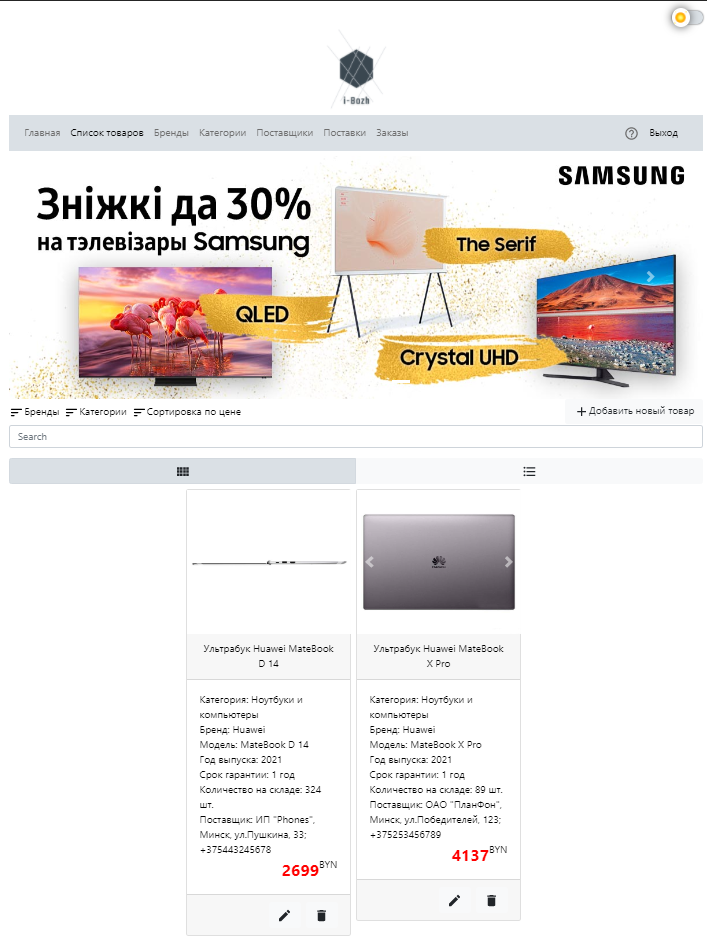


Рисунок 5.11 − Результат удаления продукта (товара)

Пятый тест будет проведён также на странице «Список товаров».

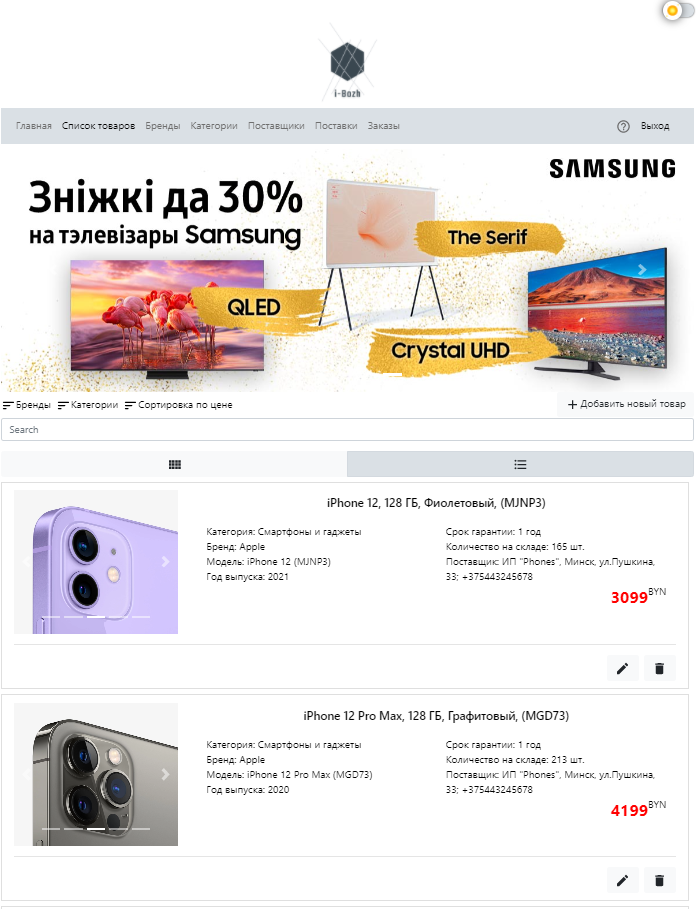


Рисунок 5.12 − Страница «Список товаров»

Таблица 5.5 − Тест-кейс функции поиска продуктов (товаров)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 5 | Поиск продуктов по названию в списке товаров в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Ввести название в строку поиска:  − «Search» − iphone se | Ожидаемый результат:  Произойдёт отображение нужных продуктов в списке  Фактический результат:  Название введено, представлено на рисунке 5.13  Результат: отображение продукта в списке, представлено на рисунке 5.13 |

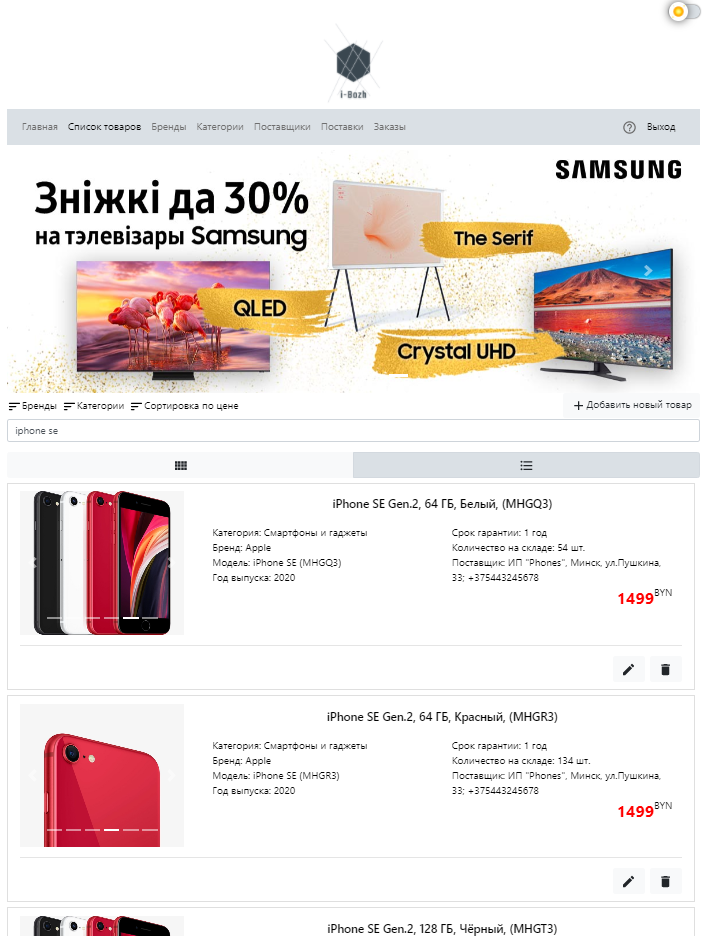


Рисунок 5.13 − Результат поиска продуктов (товаров)

Шестой тест будет проведён также на странице «Список товаров».

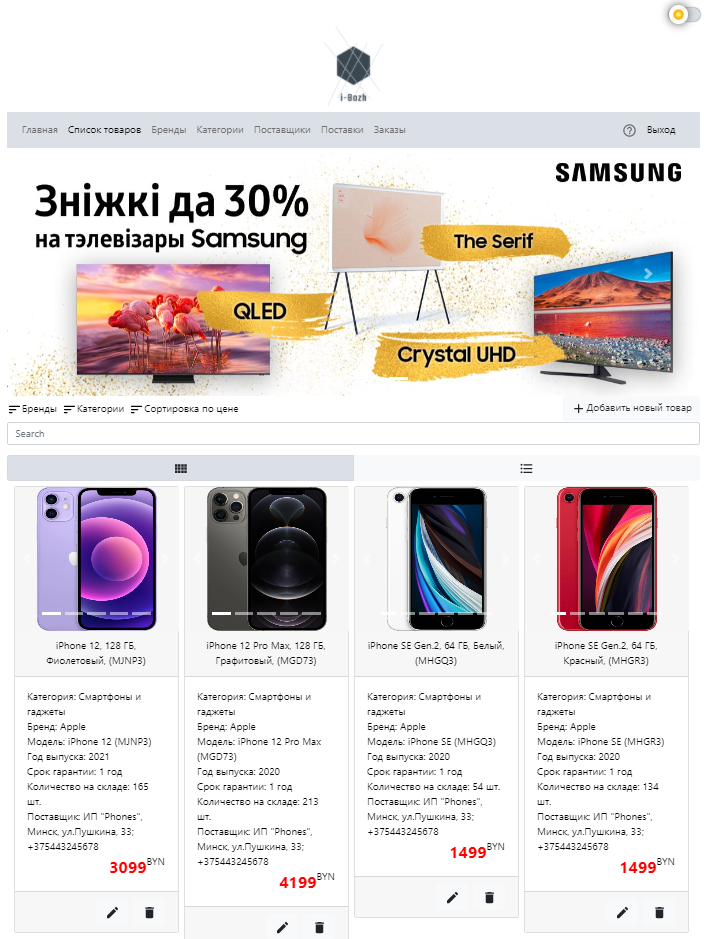


Рисунок 5.14 − Форма «Список товаров»

Таблица 5.6 − Тест-кейс функции фильтрации продуктов (товаров) по бренду и категории

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 6 | Фильтрация продуктов по брендам и категориям в списке товаров в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Выбрать бренды и категории в соответствующих чекбоксах:  − «Бренды» − Huawei, HP  − «Категории» − Ноутбуки и компьютеры | Ожидаемый результат:  Произойдёт отображение отфильтрованных продуктов в списке  Фактический результат:  Значения в чекбоксах выбраны, представлено на рисунках 5.15 и 5.16.  Результат: отображение отфильтрованных продуктов в списке, представлено на рисунке 5.17 |



Рисунок 5.15 − Выбранные значения в чекбоксе «Бренды»



Рисунок 5.16 − Выбранные значения в чекбоксе «Категории»

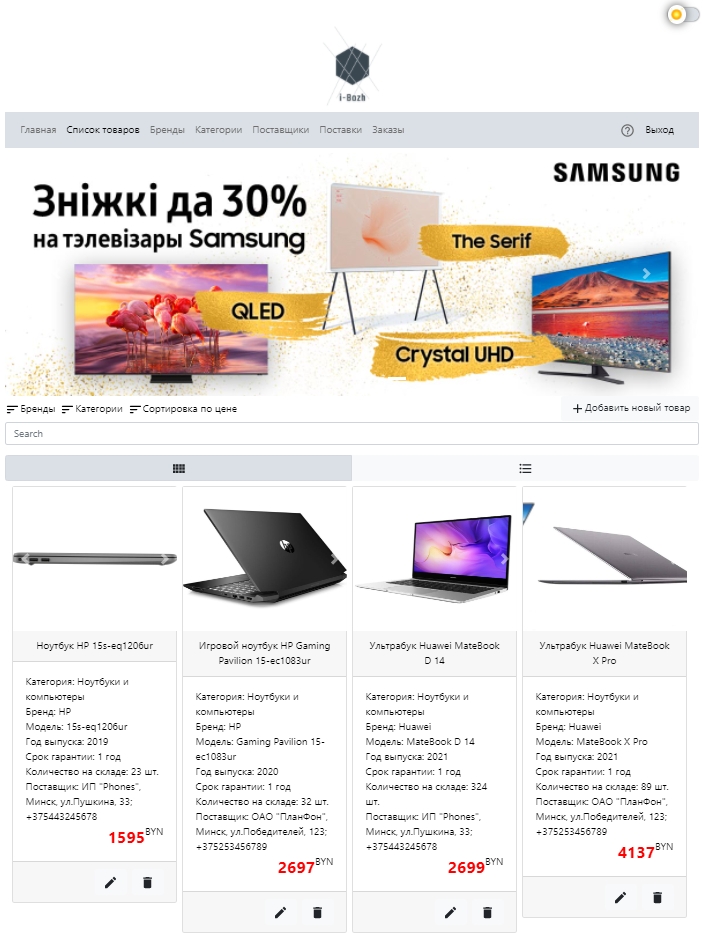


Рисунок 5.17 − Результат фильтрации продуктов (товаров)

Седьмой тест будет проведён также на странице «Список товаров».

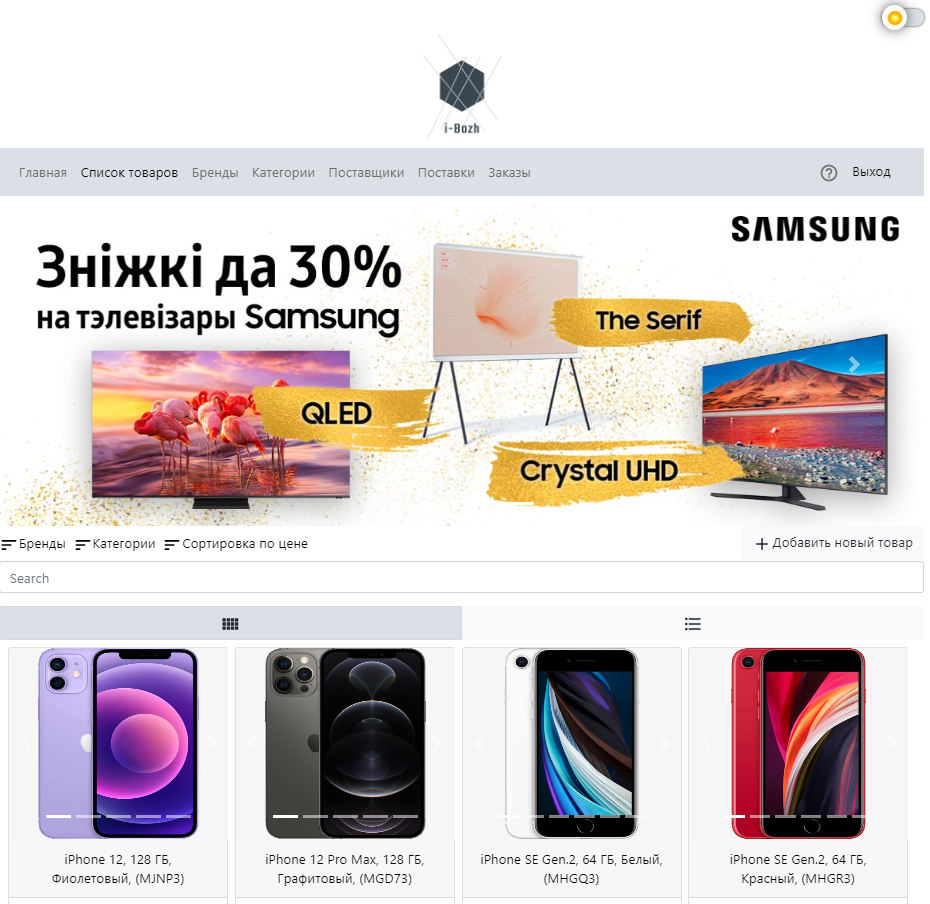


Рисунок 5.18 − Страница «Список товаров»

Таблица 5.7 − Тест-кейс функции сортировки продуктов (товаров) по цене

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 7 | Сортировка продуктов по цене в списке товаров в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Выбрать способ сортировки в соответствующем радиобоксе:  − «Сортировка по цене» − От большего к меньшему | Ожидаемый результат:  Произойдёт отображение отсортированных продуктов в списке  Фактический результат:  Значение в радиобоксе выбрано, представлено на рисунке 5.19  Результат: отображение отсортированных продуктов в списке, представлено на рисунке 5.20 |



Рисунок 5.19 − Выбранное значение в радиобоксе «Сортировка по цене»



Рисунок 5.20 − Результат сортировки продуктов (товаров)

Восьмой тест будет проведён также на странице «Список товаров» клиентской части.

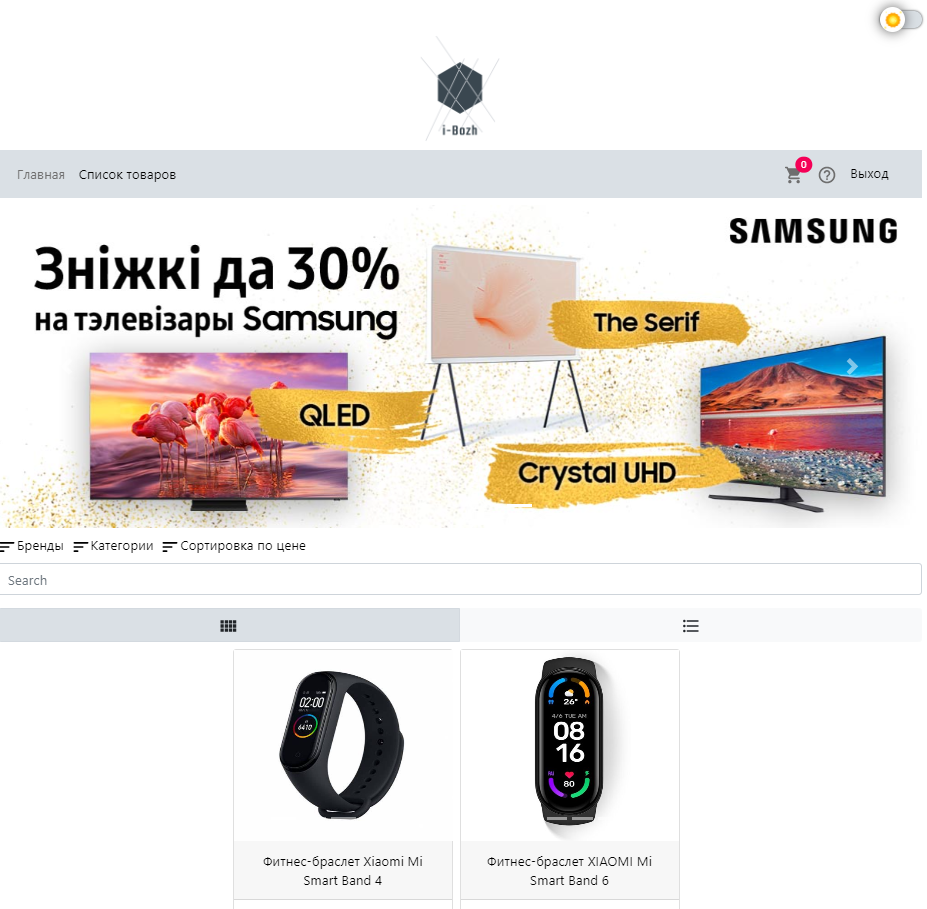


Рисунок 5.21 − Страница «Список товаров»

Таблица 5.8 − Тест-кейс функции оформления заказа

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 8 | Оформление заказа в личном кабинете клиента в приложении | 1. Нажать на пункт панели меню «Список товаров»;  2. Выбрать нужный товар;  3. Нажать на кнопку соответствующего товара «»;  4. Нажать на пункт меню «»;  5. Нажать на кнопку «Оформить заказ». | Ожидаемый результат:  Выбранный товар отобразиться в корзине, заказ добавиться в базу, появиться сообщение об оформленном заказе  Фактический результат: кнопка добавления товара нажата, товар отображается в корзине, кнопка оформления заказа нажата, представлено на рисунках 5.22, 5.23, 5.24 соответственно  Результат: заказ добавился в базу, появилось сообщение, представлено на рисунке 5.24 |



Рисунок 5.22 − Сообщение о добавленном товаре в корзину

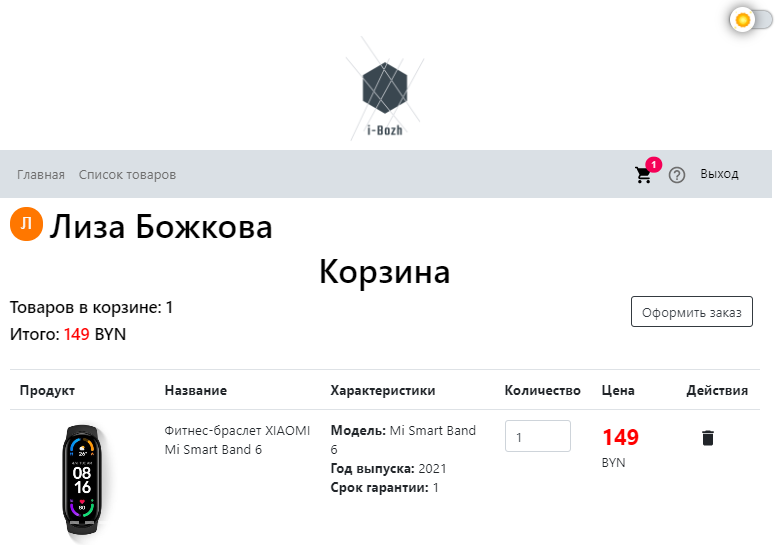


Рисунок 5.23 − Страница «Корзина»



Рисунок 5.24 − Сообщение об успешном оформлении заказа

1. Применение
2. Назначение программы

Программное средство «Онлайн-магазин электроники «i-Bozh»» предназначено для автоматизации работы магазина электроники.

Разработанное Web-приложение позволило автоматизировать ведение базы данных.

Разработанное программное средство имеет интуитивно понятный интерфейс, довольно прост в применении.

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

Приложение создано с использованием базы данных Microsoft SQL Server. Оно обладает рядом достоинств, повышающих его эффективность и полезность.

1. Программно-аппаратное обеспечение сервера и клиента

Для эксплуатации Web-сайта требуется Web-сервер с установленной операционной системой любого семейства. Размер дискового пространства Web-сервера должен быть достаточным для размещения файлов системы управления, базы данных, а также для всех текстовых и графических материалов Web-сайта (не менее 500 Мб).

Web-сайт корректно отображается в следующих браузерах:

* Mozilla Firefox (версия 2.0 и выше);
* Google Chrome (версия 5.0 и выше);
* Opera (версия 9.0 и выше).

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

К конфигурации Web-сервера предъявляются следующие технические требования:

* npm − менеджер пакетов, входящий в состав Node.js;
* сервер баз данных Microsoft SQL Server.

1. Справочная система

Для корректной работы с приложением требуется обеспечить пользователя справочной системой, в которой будут приведены действия работы с приложением, включающие данные о том, что произойдёт после нажатия на определенную кнопку или при выборе пункта меню. Существует страница справочной системы, на которой описаны основные цели данного приложения, описание кнопок и действий в данном приложении, а также список основных разделов приложения. Данная справочная система была разработана средствами JavaScript-библиотеки − React. Страница «Справочная система» для административной и пользовательской частей представлена на рисунках 6.1 и 6.2 соответственно.



Рисунок 6.1 − Справочная система административной части

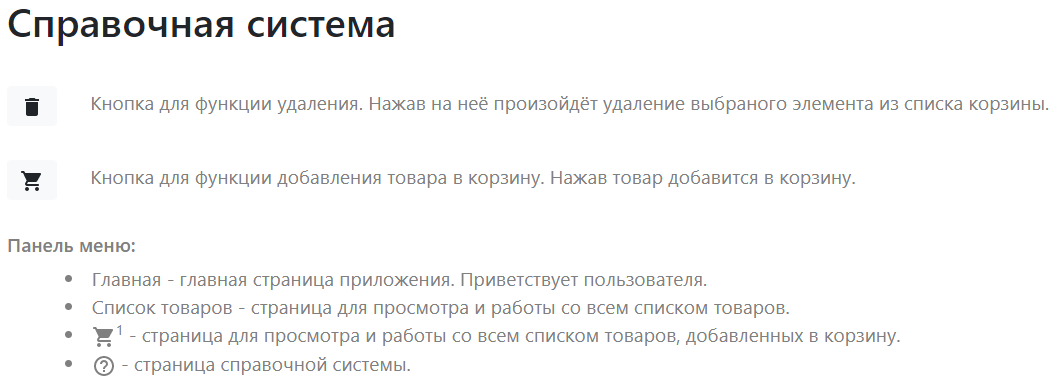


Рисунок 6.2 − Справочная система для пользовательской части

1. Охрана труда и окружающей среды на предприятии
   1. Правовые, нормативные, социально-экономические и организационные вопросы охраны труда

В соответствии с законодательством Республики Беларусь, на нанимателя возложена обязанность по организации безопасного труда работников.

В целях снижения производственного травматизма в соответствии с требованиями Закона Республики Беларусь «Об охране труда» работодатель обязан обеспечить идентификацию опасностей, оценить профессиональные риски, подготовить и реализовать мероприятия по снижению рисков с проведением анализа их эффективности.

Система управления охраной труда – это регламентированная законодательством совокупность методов и средств управления, направленных на организацию обеспечения безопасности, сохранения жизни, здоровья и работоспособности человека в процессе труда.

Основной целью СУОТ является создание условий, обеспечивающих сохранение жизни и здоровья работников в процессе трудовой деятельности.

Основными задачами СУОТ являются:

* предупреждение производственного травматизма и профессиональной заболеваемости;
* управление профессиональными рисками, устранение либо снижение их до допустимого уровня;
* непрерывное улучшение условий и охраны труда.

Создание здоровых и безопасных условий труда основывается на:

* ответственности руководителей и каждого работника за безопасность труда работающих, безаварийную работу производственных объектов;
* обучении, проведении инструктажа, стажировки и проверки знаний работников по вопросам охраны труда;
* соблюдении всех обязательных требований по охране труда, содержащихся в законодательных актах о труде и об охране труда, нормативных правовых актах в области условий и охраны труда (далее – НПА), в том числе обязательных для исполнения требований технических нормативных правовых актах, сфера действия которых распространяется на организацию (далее, если не определено иное, – ТНПА), локальных правовых актов по вопросам охраны труда (далее – ЛПА);
* вовлечении всех работников организации в обеспечение безопасных условий и охраны труда с целью применения современных методов управления охраной труда, направленных на предупреждение несчастных случаев на производстве и профессиональных заболеваний, непрерывное совершенствование деятельности по охране труда;
* организации эффективного контроля за соблюдением требований по охране труда, анализе состояния охраны труда, своевременном принятии управленческих решений по совершенствованию СУОТ;
* систематическом информировании работающих о состоянии условий и охраны труда на рабочем месте, существующем риске повреждения здоровья и полагающихся средствах индивидуальной защиты, компенсациях по условиям труда.

Представителем высшего руководства по СУОТ отделения приказом начальника генерала-майора милиции назначен начальник отделения информационного обеспечения. Он несет ответственность за разработку, внедрение, реализацию, поддержание и совершенствование СУОТ.

В функции руководства по СУОТ входит:

* организовать работу по охране труда;
* координировать деятельность структурных подразделений по обеспечению здоровых и безопасных условий труда, функционированию и совершенствованию Системы управления охраной труда;
* реализовать в пределах предоставленных прав и полномочий государственную политику в области охраны труда. Проводить анализ состояния условий и охраны труда, производственного травматизма, профессиональной и производственной обусловленной заболеваемости. Участвовать в разработке мероприятий по улучшению условий труда, предупреждению несчастных случаев и профессиональных заболеваний. Вносить предложения о внедрении передового опыта и научных разработок по безопасности и гигиене труда, разработке и внедрению более совершенных конструкций защитных, предохранительных и блокировочных устройств, других средств защиты работников от воздействия опасных и (или) вредных производственных факторов;
* разработке и пересмотре инструкций по охране труда, организационно-методических стандартов организации, содержащих требования по охране труда;
* обучении, инструктировании и проверке знаний работников по охране труда;
* составлении списков профессий и должностей, в соответствии с которыми работники должны проходить обязательные медицинские осмотры;
* проведении аттестации рабочих мест по условиям труда;
* проводить вводные инструктажи по охране труда со всеми вновь принимаемыми на работу, командированными, учащимися и студентами, прибывшими на производственное обучение или практику;
* участвовать в расследовании несчастных случаев на производстве и профессиональных заболеваний, разработке мероприятий по их профилактике, подготовке документов на выплату возмещения вреда, причиненного жизни и здоровью работников, связанного с исполнением ими трудовых обязанностей;
* подготавливать статистическую отчетность по охране и условиям труда по установленным формам и информацию по этим вопросам.

С целью реализации статьи 25 Закона Республики Беларусь от 23.06.2008 № 356-З «Об охране труда» и п. 39 Инструкции о порядке обучения, стажировки, инструктажа и проверки знаний работающих по вопросам охраны труда, утвержденной постановлением Министерства труда и социальной защиты Республики Беларусь от 28.11.2008 № 175 руководители и специалисты 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, проходят проверку знаний по вопросам охраны труда в соответствующих комиссиях для проверки знаний по вопросам охраны труда.

Руководители предприятий и организаций, не прошедшие проверку знаний по вопросам охраны труда, могут быть привлечены к административной ответственности в соответствии со статьёй 10.13 Кодекса Республики Беларусь от 06.01.2021 № 91-З «Кодекс Республики Беларусь об административных правонарушениях».

Руководители организаций, их заместители, ответственные за организацию охраны труда, главные специалисты, руководители (специалисты) служб охраны труда, члены комиссий организаций, которые не находятся в подчинении республиканского органа государственного управления, иной государственной организации, подчиненной Правительству Республики Беларусь, другой организации, проходят проверку знаний по вопросам охраны труда в комиссии Минского городского исполнительного комитета [24].

Все работающие проходят инструктаж по ОТ:

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

Порядок расследования НС на производстве, ПЗ, аварий и инцидентов на производственных объектах определены централизованно в соответствии с Правилами расследования и учета несчастных случаев на производстве и профессиональных заболеваний, утвержденных постановлением Совета Министров Республики Беларусь от 15.01.2004 № 30.

Расследование НС, ПЗ, на производстве проводится с целью:

* выявления вызвавших их причин и факторов способствующих несчастному случаю;
* установление виновных лиц либо лиц, бездействующих при организации, выполнении и контроле работ;
* разработка профилактических мероприятий, направленных на предупреждение аналогичных случаев травматизма;
* формирование организационных и технических рекомендаций по результатам расследования несчастного случая;
* оптимизация расходов на устранение последствий несчастного случая;
* принятия соответствующих мер для предотвращения повторения подобных ситуаций.

В организации ведется контроль за состоянием ОТ в соответствии с требованиями постановления Министерства труда и социальной защиты Республики Беларусь от 15.05.2020 № 51 «Об утверждении Инструкции о порядке осуществления контроля за соблюдением работниками требований по охране труда в организации и структурных подразделениях».

Контроль за соблюдением требований по охране труда проводится:

* ежедневно на рабочих местах работников их непосредственными руководителями, членами комиссии по охране труда и мастером, в подчинении которого находятся рабочие;
* ежемесячно в каждом структурном подразделении организации руководителями этих структурных подразделений и членами комиссии по охране труда;
* ежеквартально в организации членов комиссии по охране труда, руководителей структурных подразделений и непосредственных руководителей, иных работников организации.
  1. Обеспечение оптимального газового и ионного воздуха в рабочей зоне помещения

Воздушная среда, в которой осуществляется производственная деятельность человека, характеризуется химическим составом, физическими параметрами и другими показателями, оказывающими существенное влияние на здоровье работающих, их психофизиологическое состояние и работоспособность.

Атмосферный воздух, наиболее благоприятный дыхания, в своем составе содержит 78,08% азота, 20,95 % кислорода, 0,03% углекислого газа, 0,93 % инертных и 0,01 % прочих газов.

В процессе производственной деятельности и воздух рабочей зоны могут попадать вредные вещества различного происхождения (газы, пары, аэрозоли), которые способны вызывать заболевания или отклонения в состоянии здоровья работающих [18]. Ионы в воздухе (аэроионы) могут образовываться в результате естественной и искусственной ионизации. Естественное ионообразование обусловлено влиянием космических лучей, радиоактивным излучением природных материалов и является повсеместным и постоянным во времени процессом.

Искусственная аэроионизация возникает под действием ионизирующих факторов, сопровождающих некоторые технологические процессы (рентгеновские ультрафиолетовые излучения, термоэмиссия, фотоэффект), а также в специальных устройствах - ионизаторах, которые могут обеспечить в ограниченном объеме заданную концентрацию ионов определенной полярности.

Основной состав загрязнителей воздуха на многих производственных участках включает в себя оксиды углерода, серы, азота (CO, SO2, NO2 и др.), различные углеводороды (CN, HM), альдегиды (фенол формальдегид), пары минеральных кислот, аэрозоли красок и др.

В качестве регламентируемых показателей ионизации воздуха устанавливаются:

* минимально необходимый уровень;
* оптимальный уровень;
* максимально допустимый уровень;
* показатель полярности.

Минимально необходимый и максимально допустимый уровни определяют интервал концентраций во вдыхаемом воздухе названных помещений, отклонение от которого создает угрозу здоровью человека.

Оптимальным соотношением отрицательных положительных аэроионов в воздухе является П от -0,5 до 0 которое рассчитывается по формуле П = (n+ - n-)/(n+ + n-). Исходя из формулы положительно заряженных ионов в 1 см3 составляет от 1500 до 3000, а число отрицательных от 3000 до 5000.

Для обеспечения нормализации ионного состава воздуха применяются следующие методы и средства:

* удаление рабочего места неблагоприятным уровнем ионизации;
* обеспечение приточно-вытяжной вентиляции помещения;
* применение устройств для искусственной ионизации воздуха (групповые и индивидуальные ионизаторы, установки автоматического ионного режима и др.).

Компенсация ионной недостаточности может осуществляться с помощью искусственных ионизаторов чистого воздуха (аэроионизаторы, генерирующие ионы посредством коронного разряда, радиоактивные ионизаторы).

Регулируемые аэроионизаторы устанавливаются в любых воздухораспределительных устройствах приточной вентиляции, а также выполняются переносными и стационарными с автономной подачей воздуха.

Микроклимат производственных помещений – это физических факторов, оказывающих влияние на теплообмен человека и определяющих самочувствие, работоспособность, здоровье и производительность труда. Поддержание микроклимата рабочего места в пределах гигиенических норм – важнейшая задача охраны труда.

Показатели микроклимата:

* температура воздуха;
* относительная влажность воздуха;
* скорость движения воздуха;
* мощность теплового излучения.

Воздушная среда из всех элементов, составляющих среду обитания деятельности человека, является важнейшей. Природный воздух представляет собой сложную динамическую систему, образованную различными газами (и парами) и находящимися во взвешенном состоянии мельчайшими твердыми жидкими частицами – аэрозолями.

Под загрязнением воздуха понимается прямое или косвенное введение в него любого вещества в таком количестве, которое изменяет качество состав чистого атмосферного воздуха, нанося вред людям, живой и неживой природе.

Важнейшим газообразным веществом, определяющим качество воздуха, является водяной пар. Чем сильнее нагрет воздух, тем большее количество водяного пара он может содержать. Отношение содержащегося водяного пара к тому предельному количеству, которое может содержаться в воздухе при данной температуре, называется относительной влажностью.

Важнейшей характеристикой воздушной среды является барометрическое давление, поскольку разница барометрического давления и давления воздуха в альвеолах легких определяет величину газообмена. Барометрическое давление считается и называется нормальным на уровне моря (одна атмосфера) и экспоненциально убывает с высотой.

Помимо газового состава и барометрического давления, важнейшей характеристикой воздушной среды служит температура воздуха. В сочетании с подвижностью (скоростью) движения воздуха относительно тела человека температура воздуха определяет характер теплообмена – нагрев или охлаждение человека.

Жизнедеятельность человека может нормально протекать лишь при условии сохранения температурного гомеостаза организма, что достигается за счет системы терморегуляции и деятельности других функциональных систем: сердечно-сосудистой, выделительной, эндокринной и систем, обеспечивающих энергетический, водно-солевой и белковый обмен.

Для сохранения постоянной температуры тела организм должен находиться в термостабильном состоянии, которое оценивается по тепловому балансу. Тепловой баланс достигается координацией процессов теплопродукции и теплоотдачи.

Микроклимат по степени влияния на тепловой баланс человека подразделяется на:

* нейтральный;
* нагревающий;
* охлаждающий.

Нейтральный микроклимат – это такое сочетание составляющих, которое при воздействии человека течение рабочей смены обеспечивает тепловой баланс организма, разность между величиной теплопродукции и суммарной теплоотдачей находится в пределах ± 2 Вт, доля теплоотдачи испарением влаги не превышает 30%.

Охлаждающий микроклимат – это сочетание параметров, при котором имеет место превышение суммарной теплоотдачи в окружающую среду над величиной теплопродукции организма, приводящее к образованию общего и/или локального дефицита тепла в теле человека (> 2 Вт).

Нагревающий микроклимат – сочетание его параметров, при котором имеет место изменение теплообмена человека окружающей средой, проявляющееся в накоплении тепла в организме (> 2 Вт) и/или в увеличении доли потерь тепла испарением влаги (>30%).

Тепловое состояние человека – это функциональное состояние организма, обусловленное его теплообменом с окружающей средой, характеризующееся содержанием и распределением тепла в глубоких поверхностных тканях организма, а также степенью напряжения механизмов терморегуляции.

Теплового состояние человека классифицируется на:

* оптимальное;
* допустимое;
* предельно допустимое;
* недопустимое.

Разработан метод оценки теплового состояния в целях обоснования гигиенических требований к микроклимату рабочих мест, а также меры профилактики по защите работающих от возможного охлаждения и перегревания.

По степени влияния на самочувствие человека, его работоспособность микроклиматические условия подразделяются на:

* оптимальные;
* допустимые;
* вредные;
* опасные.

Нормативные гигиенические требования к отдельным показателям микроклимата, их сочетаниям, разработанные на основе изучения теплообмена и теплового состояния организма человека в микроклиматических камерах и в производственных условиях, а также на основе клинических и эпидемиологических исследований, изложены в "Требования к микроклимату рабочих мест в производственных и офисных помещениях" утвержденных постановлением Министерства здравоохранения Республики Беларусь от 30 апреля 2013 г. №33.

* 1. Пожарная безопасность

Здание УВД Администрации Фрунзенского района г. Минска относится к III степени огнестойкости, в соответствии с таблицей 7.1.

Таблица 7.1  Степень огнестойкости здания

**

Предел огнестойкости здания составляет:

* несущие элементы здания (R 90-KО) – предел огнестойкости 90 минут по потери несущей способности, не пожароопасное;
* самонесущие стены (RE 60-KO) – предел огнестойкости 60 минут по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее, не пожароопасное;
* наружные несущие стены (E 30-KO) – предел огнестойкости 30 минут по потере целостной способности, не пожароопасное;
* перекрытия междуэтажные (REI 60-KO) – предел огнестойкости 60 минут при потере несущей способности, потери целостности или теплоизолирующей способности независимо от того, какое из трех предельных состояний наступит ранее, не пожароопасное;
* настилы (RE 30-KO) – предел огнестойкости 30 минут при потере несущей способности или потери целостности независимо от того, какое из двух предельных состояний наступит ранее, не пожароопасное;
* фермы, балки, прогоны (R 30-KO) – предел огнестойкости 30 минут при потере несущей способности, не пожароопасное;
* внутренние стены (REI 105-KO) – предел огнестойкости 105 минут при потере несущей способности, потери целостности или теплоизолирующей способности независимо от того, какое из трёх предельных состояний наступит ранее, не пожароопасное;
* марши и лестничные площадки (R 45-KO) – предел огнестойкости 45 минут при потере несущей способности, не пожароопасное.

Согласно ПУЭ, помещение классифицируется на зоны. Помещение Фрунзенского РУВД г. Минска относится к категории пожароопасные В4, т.к. в помещении имеют горючие вещества и материалы хранящиеся или обращающиеся в помещении. Горючие материалы это бумага, дерево, полиэтиленовая упаковочная пленка, пластик, горючие жидкости − бензин, солярка, смазочные масла. Класс пожароопасных зон П-IIa, т.к. имеются зоны, расположенные в помещение, в которых обращаются твердые горючие вещества. Класс взрывоопасных зон В-IIа, т.к. зоны, расположенные в помещениях, в которых опасные состояния, не имеют места при нормальной эксплуатации, а возможны только в результате аварий или неисправностей.

Огнетушители переносные размещаются навеской на вертикальные конструкции на высоте 1,5 м от уровня пола до нижнего торца огнетушителя и на расстоянии от двери, достаточном для ее полного открывания, или в специальные тумбы или на пожарные щиты и стенды. Огнетушители располагаются так, чтобы основные надписи и пиктограммы, показывающие порядок приведения их в действие, были хорошо видны и обращены наружу или в сторону наиболее вероятного подхода к ним.

Все установленные на предприятии огнетушители в установленные сроки проверяются. Ответственные лица о проведенных проверках делают отметку в журнале учета огнетушителей.

Существует автоматическая система пожаротушения.

Сообщение об эвакуации может поступить в случае чрезвычайной ситуации природного или техногенного характера, при угрозе совершения акта терроризма или ликвидации его последствий, а также в иных ситуациях, требующих вывода людей в безопасную зону.

Получив сообщение от представителей властей, правоохранительных органов, специальных служб, администрации здания (сооружения, объекта) о начале эвакуации, следует соблюдать спокойствие и четко выполнять их команды.

Алгоритм действий:

* при получении сигнала об эвакуации возьмите личные документы, деньги, ценности;
* необходимо выйти из кабинета;
* не допускайте паники, истерики и спешки;
* если в здании, в котором Вы находитесь, произошел взрыв или пожар, никогда не пользуйтесь лифтом;
* необходимо идти к ближайшей лестнице, которая ведёт к основному или запасному выходу;
* возвращайтесь в покинутое помещение только после разрешения ответственных лиц.

В соответствии с п.6 ст.17 Закона Республики Беларусь от 15.06.1993 N 2403-XII (ред. от 30.11.2010) «О пожарной безопасности» и Положением для проведения профилактических мероприятий по предупреждению и тушению пожаров в организациях организуются добровольные пожарные дружины из числа рабочих и специалистов организаций в случае, если на предприятии работает более 15 человек. Во Фрунзенском УВД работают более 15 человек, следовательно, пожарной дружины существует и обязанности на случай возникновения пожара отведены на пожарную дружину.

Основные задачи ДПД:

* контроль за соблюдением противопожарного режима;
* проведение разъяснительных работ по соблюдению противопожарного режима на рабочем месте и правилам 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.

* 1. Охрана окружающей среды

Охрана окружающей среды является неотъемлемым условием обеспечения экологической безопасности, устойчивого экономического и социального развития общества. Настоящий Закон устанавливает правовые основы охраны окружающей среды, природопользования, сохранения и восстановления биологического разнообразия, природных ресурсов и объектов и направлен на обеспечение конституционных прав граждан на благоприятную для жизни и здоровья окружающую среду.

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 УВД Фрунзенского района г. Минска 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.

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 СООО «РЕМОНДИС Минск».

При утилизации и переработке твердых отходов применяют различные способы термической обработки исходных твердых материалов и полученных продуктов: пиролиз, переплав, обжиг и сжигание многих видов твердых отходов на органической основе.

Для переработки твердых отходов применяются такие процессы, как дробление и измельчение, классификация и сортировка, обогащение в тяжелых средах, отсадка, магнитная и электрическая сепарация, сушка и грануляция, термохимический обжиг, экстракция и др.

Воздух, должен быть очищен с учетом требований санитарных норм.

Мероприятия по защите воздуха:

* технологические. в ходе их проведения совершенствуются процессы изготовления и хранения топлива, производится очистка сырья от примесей, осуществляется правильная утилизация отходов. в качестве предупредительных мер создают замкнутые производственные процессы, вместо угля используют газ, широко применяют гидрообеспыливание, котельные установки заменяют на централизованное тепло;
* законодательные. к ним относятся федеральные законы. например, ограничение въезда автомобилей в места отдыха, строительство фабрик за пределами городов;
* планировочные. заключаются в организации санитарно-защитных зон, озеленении, правильной планировке городов, прокладывании автомобильных дорог в обход поселений, выборе мест для источников масштабных выбросов с учетом направления ветра;
* санитарно-технические. в их состав входит установка газоулавливающих приборов, электрофильтров, циклонов, герметизация транспортного оборудования.

Проводится следующий ряд мероприятии по очистке воздуха:

* установка пылевых фильтров на систему приточно-вытяжной вентиляции;
* регулярное проветривание помещения;
* влажная уборка помещении 2 раза в неделю.

Вид загрязнения влияет на выбор способа очистки сточных вод. Часто используют комбинированные методы для достижения наилучшего результата. Основные способы очистки сточных вод:

* механический – применяется для удаления нерастворимых примесей;
* биологический – очищение жидкости осуществляется без использования химикатов;
* биохимический – наряду с химическими реагентами применяются микроорганизмы, которые питаются загрязняющими веществами;
* химический – используется для удаления из стоков различных кислот и щелочей;
* физико-химический – включает в себя несколько способов удаления загрязнений.

Проводится следующий ряд мероприятии по очистке сточных вод:

* уменьшение количества сточных вод и содержанию в них загрязняющих веществ;
* контроль потенциально-опасных мест слива в канализацию веществ, слив которых недопустим;
* анализ воды по 20 контролируемым показателям.

1. Экономический раздел
2. Технико-экономическое обоснование разработки программного средства

Необходимо рассчитать экономическую эффективность разработки Web-приложения.

В технико-экономическом обосновании будут рассмотрены следующие вопросы:

* составление плана по разработке Web-приложения;
* расчет стоимости разработки;
* экономическая эффективность разработки.

В первой части технико-экономического обоснования необходимо рассчитать срок разработки сайта для онлайн-магазина электроники «i- Bozh».

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

В третьей части необходимо оценить экономическую эффективность Web-приложения.

1. 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 | 1 |
| Выбор методов и средств | техник-программист | 1 | 2 |
| Разработка алгоритмов и программ | техник-программист | 1 | 24 |
| Отладка программ и анализ результатов | техник-программист | 1 | 10 |
| Оформление документации и подготовка к сдаче разработки | техник-программист | 1 | 3 |

1. 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 | 7,95 | 1,59 | 9,54 |
| Ручка шариковая | штука | 1 | 0,50 | 0,10 | 0,60 |
| Папка-скоросшиватель | штука | 1 | 1,50 | 0,30 | 1,80 |
| Диск | штука | 1 | 0,80 | 0,16 | 0,96 |
| Всего расходов | | | | | 12,90 |
| Всего с транспортно-заготовительными расходами | | | | | 13,55 |

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.

Одноставочный тариф для непромышленного потребления 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 91 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 | 1,1 | 8 | 8,8 |
| Выбор методов и средств | техник-  программист | 7 | 2,03 | 1,1 | 16 | 17,6 |
| Разработка алгоритмов и программ | техник-  программист | 7 | 2,03 | 1,1 | 192 | 211,2 |
| Отладка программ и анализ результатов | техник-  программист | 7 | 2,03 | 1,1 | 80 | 88 |
| Оформление документации и подготовка к сдаче разработки | техник-  программист | 7 | 2,03 | 1,1 | 24 | 26,4 |
| Всего | | | | | | 352 |
| Всего с коэффициентом премий | | | | | | 387,2 |

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 = 387,2 × 10 / 100 = 38,72

Рассчитываются отчисления органам социальной защиты по формуле (8.4):

, (8.4)

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

Рос = (387,2 + 38,72) × 34 / 100 = 144,81

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 = (387,2 + 38,72) × 0,6 / 100 = 2,56

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 (8.6).

, (8.6)

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

Рпр = 387,2 × 20 / 100 = 77,44

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

, (8.7)

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

Ркос = 387,2 × 55 / 100 = 212,96

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

(8.8)

Сп = 13,55 + 55,76 + 387,2 + 38,72 + 144,81 + 2,56 + 77,44 + 212,96 = 933

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 = 933 × 10 / 100 = 93,3

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

. (8.10)

Цотп = 933 + 93,3 = 1026,3

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

, (8.11)

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

НДС = 1026,3 × 20 / 100 = 205,26

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

(8.12)

Цотп = 1036,3 + 205,26 = 1231,56

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. |
| 1 | 2 | 3 |
| Материалы, покупные полуфабрикаты и комплектующие изделия | Рм | 13,55 |
| Электроэнергия | Рэ | 55,76 |
| Основная заработная плата научно-производственного персонала | Роз | 387,2 |
| Дополнительная заработная плата научно-производственного персонала | Рдз | 38,72 |
| Отчисления на социальную защиту и на страхование от несчастных случаев | Рос  Остр | 144,81  2,56 |
| Прочие прямые расходы | Рпр | 77,44 |

Продолжение таблицы 8.4.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Полная себестоимость | Сп | 933 |
| Накладные расходы | Ркос | 212,96 |
| Плановые накопления (прибыль) | П | 93,3 |
| Отпускная цена (без НДС) | Ц | 1026,3 |
| Налог на добавленную стоимость | РНДС | 205,26 |
| Отпускная цена с НДС | Цотп | 1231,56 |

1. Экономическая эффективность разработки

В данном разделе представлено экономическое обоснование для дипломного проекта по теме «Разработка сайта для онлайн-магазин электроники «i- Bozh»».

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 1231,56 рублей.

Основное преимущество разработки Web-приложения в том, что будет возможность приобретать электронику в интернете, не прилагая сильных усилий. Также можно будет получать выходные документы: отчёт для администратора и чек оформленного заказа для клиента.

Заключение

В рамках дипломного проекта на тему «Разработка сайта для онлайн-магазина электроники «i-Bozh»» было разработано программное средство, автоматизирующее работу администратора и клиентов магазина.

Для достижения цели были решены следующие задачи:

* выполнен объектно-ориентированный анализ и проектирование системы;
* выбрано и описано проектирование к разработанному Web-приложению, на базе которой создавалось приложение;
* выполнено проектирование макетов для разработанного Web-приложения;
* описано созданное Web-приложение;
* описана методика проведения испытаний;
* описано применение созданного Web-приложения.

В Web-приложении были реализованы следующие функции:

* ведение единой базы данных, содержащей информацию о клиентах, товарах, заказах, поставках и поставщиках;
* поиск по названию товара;
* сортировку по цене товара;
* фильтрацию по бренду и категории товара;
* формирование выходного документа Excel, который будет содержать данные обо всех оформленных заказах в разрезе дат − Report.xlsx;
* формирование выходного документа Excel, который будет содержать данные об оформленном заказе − CheckOnline.xlsx.

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

Программное приложение реализовано в полном объеме и в соответствии с заданными требованиями, полностью отлажено и протестировано. Поставленные задачи выполнены.

Программное средство готово 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. Елецкий Т.В Экономика предприятия: учебное пособие / Елецкий Т.В – Под ред. Э.В. Крум, – Мн.: Выш.шк., 2005
3. Лобан Л.Д. Экономика предприятия: учебный комплекс/ Лобан Л.Д. – Мн.: Мисанта, 2006
4. Общие требования к тестовым документам: ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 1995.
5. 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.
6. CSS [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/CSS> − Дата доступа: 26.04.2021
7. Data Annotations - ForeignKey Attribute in EF 6 & EF Core [Электронный ресурс] – Entity Framework Tutorial – Режим доступа: <https://www.entityframeworktutorial.net/code-first/foreignkey-dataannotations-attribute-in-code-first.aspx> − Дата доступа: 22.04.2021
8. 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: 25.04.2021
9. JavaScript [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/JavaScript> − Дата доступа: 26.04.2021
10. jQuery [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/JQuery> − Дата доступа: 26.04.2021
11. Microsoft SQL Server [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_SQL_Server> − Дата доступа: 26.04.2021
12. Model-View-Controller [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller> − Дата доступа: 26.04.2021
13. React [Электронный ресурс] − Режим доступа: <https://ru.wikipedia.org/wiki/React> − Дата доступа: 26.04.2021
14. React JS. Подключение React в ASP.NET Core [Электронный ресурс] – Metanit – Режим доступа: <https://metanit.com/sharp/aspnet5/25.1.php> − Дата доступа: 22.04.2021
15. StarUML [Электронный ресурс] − Wikipedia − Режим доступа: <https://en.wikipedia.org/wiki/StarUML> − Дата доступа: 22.04.2021
16. Visual Studio [Электронный ресурс] − Microsoft − Режим доступа: <https://visualstudio.microsoft.com/ru/downloads/> − Дата доступа: 22.04.2021
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](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

//DBModels

//DBModels/Brand.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Brand //модель бренда

{

public int Id { get; set; } //номер бренда в базе

[Required]

public string Name { get; set; } //название бренда

[JsonIgnore]

public ICollection<Product> Products { get; set; } //FK на модель Product

}

}

//DBModels/Client.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Client //модель клиента

{

public int Id { get; set; } //номер клиента в базе

public string Name { get; set; } //имя клиента

public string Surname { get; set; } //фамилия клиента

public string Adress { get; set; } //адрес клиента

public string Number { get; set; } //номер телефона клиента

public string Login { get; set; } //логин клиента

public string Password { get; set; } //пароль клиента

[JsonIgnore]

public ICollection<Order> Orders { get; set; } //FK на модель Order

}

}

//DBModels/Order.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Order //модель заказа

{

public int Id { get; set; } //номер заказа в базе

[ForeignKey("Client")]

public int ClientId { get; set; } //номер клиента

public Client Client { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице Order

public DateTime Date { get; set; } //дата заказа

public double TotalPrice { get; set; } //итогова цена заказа

[JsonIgnore]

public ICollection<ProductOrderInfo> ProductOrderInfos { get; set; } //FK на модель ProductOrderInfo

}

}

//DBModels/Product.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Product //модель продукта

{

public int Id { get; set; } //номер товара в базе

public string Name { get; set; } //название товара

public string Year { get; set; } //год выпуска товара

[ForeignKey("Brand")]

public int BrandId { get; set; } //номер бренда

public Brand Brand { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице Product

[ForeignKey("Type")]

public int TypeId { get; set; } //номер категории

public Type Type { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице Product

public string Modal { get; set; } //модель товара

public int Warranty { get; set; } //срок гарантии

public double Price { get; set; } //цена товара

public int Amount { get; set; } //количество на складе

[ForeignKey("Supply")]

public int SupplyId { get; set; } //номер поставки

public Supply Supply { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице Product

public ICollection<Image> Images { get; set; } //коллекция фото товара

[JsonIgnore]

public ICollection<ProductOrderInfo> ProductOrderInfos { get; set; } //FK на модель ProductOrderInfo

public Product()

{

Images = new HashSet<Image>(); //создание коллекции фото товара

}

}

}

// DBModels/Images.cs

namespace HardwareStoreServer.Models.DBModels

{

public class Image //модель фото

{

public int Id { get; set; } //номер фото в базе

public string Url { get; set; } //ссылка на фото

public override bool Equals(object obj) //функция сравнения экземпляров коллекции

{

if (!(obj is Image))

{

return false;

}

return ((Image)obj).Url.Equals(Url);

}

}

}

//DBModels/ProductOrderInfo.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class ProductOrderInfo //модель продукт заказ инфо

{

public int Id { get; set; } //номер продукт заказ инфо в базе

[ForeignKey("Product")]

public int ProductId { get; set; } //номер продукта

public Product Product { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице ProductOrderInfo

public int Quantity { get; set; } //количество товара

[ForeignKey("Order")]

public int? OrderId { get; set; } //номер заказа

public Order Order { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице ProductOrderInfo

}

}

//DBModels/Supplier.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Supplier //модель поставщика

{

public int Id { get; set; } //номер поставщика в базе

public string NameOrganization { get; set; } //название организации

public string Number { get; set; } //номер телефона

public string Adress { get; set; } //адрес офиса поставщика

[JsonIgnore]

public ICollection<Supply> Supplies { get; set; } //FK на модель Supply

}

}

//DBModels/Supply.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Supply //модель поставки

{

public int Id { get; set; } //номер поставки в базе

[ForeignKey("Supplier")]

public int SupplierId { get; set; } //номер поставщика

public Supplier Supplier { get; set; } //навигационное свойство, сгенерирует столбец для внешнего ключа в таблице Supply

public DateTime Date { get; set; } //дата поставки

[JsonIgnore]

public ICollection<Product> Products { get; set; } //FK на модель Product

}

}

//DBModels/Type.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models.DBModels

{

public class Type //модель категории

{

public int Id { get; set; } //номер категории в базе

[Required]

public string Name { get; set; } //наименование категории

[JsonIgnore]

public ICollection<Product> Products { get; set; } //FK на модель Product

}

}

//ApplicationDbContext.cs

using HardwareStoreServer.Models.DBModels;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace HardwareStoreServer.Models

{

public class ApplicationDbContext: DbContext //определение контекста данных (подключение бд)

{

public ApplicationDbContext() //конструктор

{

this.Database.EnsureCreated(); //инициализация бд начальными данными

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("Data Source=DESKTOP-HO2E7N9;Initial Catalog=HardwareStoreDB;Trusted\_connection=True"); //строка подключения к бд

}

public virtual DbSet<Brand> Brands { get; set; }

public virtual DbSet<Client> Clients { get; set; }

public virtual DbSet<Order> Orders { get; set; }

public virtual DbSet<Product> Products { get; set; }

public virtual DbSet<ProductOrderInfo> ProductOrderInfos { get; set; }

public virtual DbSet<Supplier> Suppliers { get; set; }

public virtual DbSet<Supply> Supplies { get; set; }

public virtual DbSet<DBModels.Type> Types { get; set; }

public virtual DbSet<Image> Images { get; set; }

}

}

//Services

//IDBService.cs

using System.Collections.Generic;

namespace HardwareStoreServer.Services.DBServices

{

public interface IDBService<T> //интерфейс для определения функционала сервисор

{

bool Create(T entity); //метод добавления

IList<T> GetAll(); //метод выборки

T GetById(int id); //метод выборки экземпляра по id

bool Remove(int id); //метод удаления экземпляра по id

bool Update(T newEntity); //метод редактирования экземпляра

}

}

//DBProductOrderInfoService.cs

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace HardwareStoreServer.Services.DBServices

{

public class DBProductOrderInfoService: IDBService<ProductOrderInfo> //наследование от интерфейса IDBService

{

private readonly ApplicationDbContext context; //определение контекста

public DBProductOrderInfoService(ApplicationDbContext context)

{

this.context = context; //присваение контекста сущностей, которые включены в модель данных

}

public bool Create(ProductOrderInfo entity) //определение метода добавления

{

if (entity == null) //проверка по пустоту добавляемого экземпляра

{

return false;

}

var state = context.Add(entity).State; //определение состояния добавления экземпляра

if (state != EntityState.Added) //проверка на добавления

{

return false; //возрат значения при неудачном добалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат сообщения об успешном добавлении

}

public IList<ProductOrderInfo> GetAll() //определение метода выборки

{

return context.ProductOrderInfos.ToList(); //возрат листа записей в сущности ProductOrderInfos в бд

}

public ProductOrderInfo GetById(int id) //определение метода выборки по id

{

return context.ProductOrderInfos.FirstOrDefault(x => x.Id == id); //возрат экземпляра сущности ProductOrderInfos в бд

}

public bool Remove(int id) //определение метода удаления по id

{

var deleted = context.ProductOrderInfos.FirstOrDefault(x => x.Id == id); //выборка нужного для удаления экземпляра сущности ProductOrderInfos

if (deleted == null) //проверка на существование экземпляра

{

return false;

}

var result = context.ProductOrderInfos.Remove(deleted).State; //определение состояния удаления экземпляра

if (result != EntityState.Deleted) //проверка на удаление

{

return false; //возрат значения при неудачном удалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном удалении

}

public bool Update(ProductOrderInfo newEntity) //определение метода редактирования

{

if (newEntity == null) //проверка на пустоту экземпляра для редактирования

{

return false;

}

var prevEntity = context.ProductOrderInfos.FirstOrDefault(x => x.Id == newEntity.Id); //выборка нужного для изменения экземляра сущности ProductOrderInfos

if (prevEntity == null) //проверка на существование экземпляра

{

return false;

}

prevEntity.ProductId = newEntity.ProductId; //присваение значений нового экземпляра

prevEntity.Quantity = newEntity.Quantity;

prevEntity.OrderId = newEntity.OrderId;

try

{

context.SaveChanges(); //сохранение изменений в экземпляре

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном редактировании

}

}

}

//DBProductService.cs

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace HardwareStoreServer.Services.DBServices

{

public class DBProductService: IDBService<Product> //наследование от интерфейса IDBService

{

private readonly ApplicationDbContext context; //определение контекста

public DBProductService(ApplicationDbContext context)

{

this.context = context; //присваение контекста сущностей, которые включены в модель данных

}

public bool Create(Product entity) //определение метода добавления

{

if (entity == null) //проверка по пустоту добавляемого экземпляра

{

return false;

}

var state = context.Add(entity).State; //определение состояния добавления экземпляра

if (state != EntityState.Added) //проверка на добавления

{

return false; //возрат значения при неудачном добалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат сообщения об успешном добавлении

}

public IList<Product> GetAll() //определение метода выборки

{

return context.Products.ToList(); //возрат листа записей в сущности Products в бд

}

public Product GetById(int id) //определение метода выборки по id

{

return context.Products.FirstOrDefault(x => x.Id == id); //возрат экземпляра сущности ProductOrderInfos в бд

}

public bool Remove(int id) //определение метода удаления по id

{

var deleted = context.Products.FirstOrDefault(x => x.Id == id); //выборка нужного для удаления экземпляра сущности Products

if (deleted == null) //проверка на существование экземпляра

{

return false;

}

var result = context.Products.Remove(deleted).State; //определение состояния удаления экземпляра

if (result != EntityState.Deleted) //проверка на удаление

{

return false; //возрат значения при неудачном удалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном удалении

}

public bool Update(Product newEntity) //определение метода редактирования

{

if (newEntity == null) //проверка на пустоту экземпляра для редактирования

{

return false;

}

var prevEntity = context.Products.FirstOrDefault(x => x.Id == newEntity.Id); //выборка нужного для изменения экземляра сущности Products

if (prevEntity == null) //проверка на существование экземпляра

{

return false;

}

prevEntity.Name = newEntity.Name; //присваение значений нового экземпляра

prevEntity.Year = newEntity.Year;

prevEntity.BrandId = newEntity.BrandId;

prevEntity.TypeId = newEntity.TypeId;

prevEntity.Modal = newEntity.Modal;

prevEntity.Warranty = newEntity.Warranty;

prevEntity.Price = newEntity.Price;

prevEntity.Amount = newEntity.Amount;

prevEntity.SupplyId = newEntity.SupplyId;

prevEntity.Image = newEntity.Image;

try

{

context.SaveChanges(); //сохранение изменений в экземпляре

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном редактировании

}

}

}

//DBOrderService.cs

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace HardwareStoreServer.Services.DBServices

{

public class DBOrderService: IDBService<Order> //наследование от интерфейса IDBService

{

private readonly ApplicationDbContext context; //определение контекста

private readonly ExcelService excelService; //определение сервира

public DBOrderService(ApplicationDbContext context, ExcelService excelService)

{

this.context = context; //присваение контекста сущностей, которые включены в модель данных

this.excelService = excelService; //присваение сервиса

}

public bool Create(Order entity) //определение метода добавления

{

if (entity == null) //проверка по пустоту добавляемого экземпляра

{

return false;

}

var state = context.Add(entity).State; //определение состояния добавления экземпляра

if (state != EntityState.Added) //проверка на добавления

{

return false; //возрат значения при неудачном добалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат сообщения об успешном добавлении

}

public byte[] ExportOrdersToExcel(DateTime from, DateTime to) //определние метода экпорта заказов

{

var q = context.Orders.Where(o => o.Date > from && o.Date < to); //выборка нужных записей для экспорта

return excelService.ExportDataForOrders(q); //воздат содержание файла в памяти

}

public byte[] ExportCartToExcel() //определение метода экспорта данных в чек

{

return excelService.ExportDataForCart(); //воздат содержание файла в памяти

}

public IList<Order> GetAll() //определение метода выборки

{

return context.Orders.ToList(); //возрат листа записей в сущности Orders в бд

}

public Order GetById(int id) //определение метода выборки по id

{

return context.Orders.FirstOrDefault(x => x.Id == id); //возрат экземпляра сущности Orders в бд

}

public bool Remove(int id) //определение метода удаления по id

{

var deleted = context.Orders.FirstOrDefault(x => x.Id == id); //выборка нужного для удаления экземпляра сущности Orders

var prodDel = context.ProductOrderInfos.Where(x => x.OrderId == id); //выборка нужных для удаления экземпляров сущности ProductOrderInfo

if (deleted == null) //проверка на существование экземпляра

{

return false;

}

foreach (var prod in prodDel)

{

context.ProductOrderInfos.Remove(prod); //удаление продуктов из заказа

}

var result = context.Orders.Remove(deleted).State; //определение состояния удаления экземпляра

if (result != EntityState.Deleted) //проверка на удаление

{

return false; //возрат значения при неудачном удалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном удалении

}

public bool Update(Order newEntity) //определение метода редактирования

{

if (newEntity == null) //проверка на пустоту экземпляра для редактирования

{

return false;

}

var prevEntity = context.Orders.FirstOrDefault(x => x.Id == newEntity.Id); //выборка нужного для изменения экземляра сущности Orders

if (prevEntity == null) //проверка на существование экземпляра

{

return false;

}

prevEntity.ClientId = newEntity.ClientId; //присваение значений нового экземпляра

prevEntity.Date = newEntity.Date;

prevEntity.TotalPrice = newEntity.TotalPrice;

try

{

context.SaveChanges(); //сохранение изменений в экземпляре

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном редактировании

}

}

}

//DBClientService.cs

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace HardwareStoreServer.Services.DBServices

{

public class DBClientService: IDBService<Client> //наследование от интерфейса IDBService

{

private readonly ApplicationDbContext context; //определение контекста

public DBClientService(ApplicationDbContext context)

{

this.context = context; //присваение контекста сущностей, которые включены в модель данных

}

public bool Create(Client entity) //определение метода добавления

{

if (entity == null) //проверка по пустоту добавляемого экземпляра

{

return false;

}

var user = context.Clients.FirstOrDefault(c => c.Login.Equals(entity.Login)); //выборка экземпляра сущности Clients

if (user != null) //проверка на существование добавляемого клиента

{

throw new Exception("User already register");

}

var state = context.Add(entity).State; //определение состояния добавления экземпляра

if (state != EntityState.Added) //проверка на добавления

{

return false; //возрат значения при неудачном добалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат сообщения об успешном добавлении

}

public IList<Client> GetAll() //определение метода выборки

{

return context.Clients.ToList(); //возрат листа записей в сущности Clients в бд

}

public Client GetById(int id) //определение метода выборки по id

{

return context.Clients.FirstOrDefault(x => x.Id == id); //возрат экземпляра сущности Clients в бд

}

public Client GetByLogin(string login) //определение метода выборки по login

{

return context.Clients.FirstOrDefault(x => x.Login == login); //возрат экземпляра сущности Clients в бд

}

public bool Remove(int id) //определение метода удаления по id

{

var deleted = context.Clients.FirstOrDefault(x => x.Id == id); //выборка нужного для удаления экземпляра сущности Clients

if (deleted == null) //проверка на существование экземпляра

{

return false;

}

var result = context.Clients.Remove(deleted).State; //определение состояния удаления экземпляра

if (result != EntityState.Deleted) //проверка на удаление

{

return false; //возрат значения при неудачном удалении

}

try

{

context.SaveChanges(); //сохранение изменений в контексте

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном удалении

}

public bool Update(Client newEntity) //определение метода редактирования

{

if (newEntity == null) //проверка на пустоту экземпляра для редактирования

{

return false;

}

var prevEntity = context.Clients.FirstOrDefault(x => x.Id == newEntity.Id); //выборка нужного для изменения экземляра сущности Clients

if (prevEntity == null) //проверка на существование экземпляра

{

return false;

}

prevEntity.Name = newEntity.Name; //присваение значений нового экземпляра

prevEntity.Surname = newEntity.Surname;

prevEntity.Adress = newEntity.Adress;

prevEntity.Number = newEntity.Number;

prevEntity.Login = newEntity.Login;

prevEntity.Password = newEntity.Password;

try

{

context.SaveChanges(); //сохранение изменений в экземпляре

}

catch

{

return false; //возрат значения при неудачном измении

}

return true; //возрат значения при удачном редактировании

}

}

}

//ExcelService.cs

using ClosedXML.Excel;

using HardwareStoreServer.Models;

using HardwareStoreServer.Models.DBModels;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace HardwareStoreServer.Services.DBServices

{

public class ExcelService

{

public ExcelService(){}

private string clientFormate(IList<Client> clients, int id) //определение метода для формиррования данных о клиенте

{

foreach (var client in clients)

{

if (id == client.Id)

{

return string.Concat(client.Name + " " + client.Surname);

}

}

return " ";

}

public byte[] ExportDataForOrders(IEnumerable<Order> orders) //определение метода экпорта в Excel для заказов

{

using (var context = new ApplicationDbContext()) //определение контекста данных

{

var clients = context.Clients.ToList(); //выборка листа записей сущности Clients в бд

using (var workbook = new XLWorkbook()) //создание excel книги

{

IXLWorksheet workSheet = workbook.Worksheets.Add("Отчёт"); //добавление листа в книгу

var range = workSheet.Range("A1:D1"); //определение полей файла

range.Value = "Заказы";

range.Merge(); //объединение ячеек

range.Style.Font.Bold = true;

workSheet.Cell(2, 1).Value = "Номер заказа";

workSheet.Cell(2, 2).Value = "Клиент";

workSheet.Cell(2, 3).Value = "Дата заказа";

workSheet.Cell(2, 4).Value = "Сумма заказа";

int i = 3;

foreach (var item in orders)

{

workSheet.Cell(i, 1).Value = item.Id;

workSheet.Cell(i, 2).Value = clientFormate(clients, item.ClientId);

workSheet.Cell(i, 3).Value = item.Date.ToString("dd/MM/yyyy");

workSheet.Cell(i, 4).Value = item.TotalPrice;

++i;

}

workSheet.Columns().AdjustToContents(); //авторазмер ячеек

workSheet.Style.Alignment.Horizontal = XLAlignmentHorizontalValues.Center; //выравнивание по центру ячейки

var table = workSheet.Range(("A1:D" + (i-1)).ToString());

table.Style.Border.OutsideBorder = XLBorderStyleValues.Thin; //внешние границы ячеек

table.Style.Border.InsideBorder = XLBorderStyleValues.Thin; //внутрнении границы ячеек

using (MemoryStream stream = new MemoryStream()) //создание хранилища данных

{

workbook.SaveAs(stream); //сохранение книги в хранилице

return stream.ToArray(); //возрат массива данных

}

}

}

}

[Obsolete]

public byte[] ExportDataForCart()

{

using (var context = new ApplicationDbContext()) //определение контекста данных

{

var order = context.Orders.ToList().Last(); //выборка последнего экземпляра сущности Orders в бд

var productsOrder = context.ProductOrderInfos.ToList().Where(product => product.OrderId == order.Id); //выборка листа продуктов нужного заказа

var products = context.Products.ToList(); // выборка листа записей сущности Products в бд

using (var workbook = new XLWorkbook()) //создание excel книги

{

IXLWorksheet workSheet = workbook.Worksheets.Add("Чек"); //добавление листа в книгу

var range = workSheet.Range("A1:E1"); //определение полей файла

range.Value = "ООО «i-Bozh shop», ИНН: 77100678000";

range.Merge(); //объединение ячеек

range.Style.Font.Bold = true;

range = workSheet.Range("A2:E2");

range.Value = "г. Минск, пр. Пушкина, 13, оф. 43";

range.Merge();

range.Style.Font.Bold = true;

range = workSheet.Range("A4:E4");

range.Value = "Товарный чек №" + order.Id + " от " + order.Date.ToString("dd/MM/yyyy");

range.Merge();

range.Style.Font.Bold = true;

range.Style.Alignment.Horizontal = XLAlignmentHorizontalValues.Center; //выравнивание по центру ячейки

workSheet.Cell(5, 1).Value = "№";

workSheet.Cell(5, 2).Value = "Наименование товара";

workSheet.Cell(5, 3).Value = "Кол-во";

workSheet.Cell(5, 4).Value = "Цена";

workSheet.Cell(5, 5).Value = "Итого";

range = workSheet.Range("A5:E5");

range.Style.Fill.BackgroundColor = XLColor.LightGray;

range.Style.Alignment.Horizontal = XLAlignmentHorizontalValues.Center;

int i = 6;

int idProduct = 1; //переменная для определения номера продукта в таблице документа

foreach (var item in productsOrder)

{

var product = products.FirstOrDefault(product => item.ProductId == product.Id);

workSheet.Cell(i, 1).Value = idProduct;

workSheet.Cell(i, 2).Value = product.Name;

workSheet.Cell(i, 3).Value = item.Quantity;

workSheet.Cell(i, 4).Value = product.Price + " руб.";

workSheet.Cell(i, 5).Value = item.Quantity \* product.Price + " руб.";

++idProduct;

++i;

}

workSheet.Cell(i, 5).Value = order.TotalPrice + " руб.";

workSheet.Columns().AdjustToContents(); //авторазмер ячеек

var table = workSheet.Range("A5:E" + (i - 1));

table.Style.Border.OutsideBorder = XLBorderStyleValues.Thin;

table.Style.Border.InsideBorder = XLBorderStyleValues.Thin;

range = workSheet.Range(("A"+(i + 1) + ":E" + (i + 1)).ToString());

range.Value = "Всего наименований " + productsOrder.Count() + " на сумму " + order.TotalPrice + " руб.";

range.Merge(); //объединение ячеек

range.Style.Alignment.Horizontal = XLAlignmentHorizontalValues.Left;

using (MemoryStream stream = new MemoryStream()) //создание хранилища данных

{

workSheet.Protect("i-Bozh2021"); //защита листа

workbook.Protect(true, true); //защита книги

workbook.SaveAs(stream); //сохранение книги в хранилице

return stream.ToArray(); //возрат массива данных

}

}

}

}

//ApiControllers

//ClientController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Services.DBServices;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace HardwareStoreServer.Controllers

{

//создание http путей

[Route("api/[controller]")]

[EnableCors("AllowMyOrigin")] //подключение cors для доступа в контроллеру сервера

public class ClientController : Controller

{

private readonly DBClientService service; //определение сервиса

public ClientController(DBClientService service)

{

this.service = service; //присваение сервиса

}

[HttpPost("create")]

public bool Create([FromBody]Client client) //метод добавления

{

return service.Create(client);

}

[HttpGet("getAll")]

public IList<Client> GetAll() //метод выборки

{

return service.GetAll();

}

[HttpGet("getById/{id}")]

public Client GetById(int id) //метод выборки по id

{

return service.GetById(id);

}

[HttpGet("getByLogin/{login}")]

public Client GetByLogin(string login) //метод выборки по login

{

return service.GetByLogin(login);

}

[HttpDelete("delete/{id}")]

public bool Delete(int id) //метод удаления

{

return service.Remove(id);

}

[HttpPut("edit")]

public bool Update(Client client) //метод редактирования

{

return service.Update(client);

}

}

}

//OrderController.cs

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using HardwareStoreServer.Models;

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Services.DBServices;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

namespace HardwareStoreServer.Controllers

{

//создание http путей

[Route("api/[controller]")]

[EnableCors("AllowMyOrigin")] //подключение cors для доступа в контроллеру сервера

public class OrderController : Controller

{

private readonly DBOrderService service; //определение сервиса

private readonly ApplicationDbContext \_context; //определение контекста

public OrderController(ApplicationDbContext context, DBOrderService service)

{

this.\_context = context; //присваение контекста

this.service = service; //присваение сервиса

}

[HttpPost("create")]

public bool Create(Order order) //метод добавления

{

return service.Create(order);

}

[HttpPost("makeOrder")]

public async Task MakeOrder([FromQuery]int clientId, [FromQuery]int totalPrice, [FromBody]ProductOrderInfo[] items) //метод создания заказа

{

if (items.Length == 0) //проверка на пустоту массива товаров

{

return;

}

var newOrder = new Order() //создание нового заказа

{

ClientId = clientId, //присваение значений

Date = DateTime.Now,

TotalPrice = totalPrice

};

await \_context.AddAsync(newOrder); //добавление в заказ

await \_context.SaveChangesAsync(); //сохранение изменений в контексте

for (int i = 0; i < items.Length; i++)

{

//items[i].ProductId = items[i].Id;

items[i].OrderId = newOrder.Id; //присваение значений номера заказа в листе товаров

}

await \_context.AddRangeAsync(items); //добавление в лист товаров

await \_context.SaveChangesAsync(); //сохранение изменений в контексте

}

[HttpGet("getAll")]

public IList<Order> GetAll() //метод выборки

{

return service.GetAll();

}

[HttpGet("excelOrders")]

public FileResult GetExcelOrders([FromQuery]DateTime from, [FromQuery]DateTime to) //метод экпорта данных о заказах в Excel файл

{

var data = service.ExportOrdersToExcel(from, to);

return File(data, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", "Report.xlsx");

}

[HttpGet("excelCart")]

public FileResult GetExcelCart() //метод экпорта данных корзины в Excel файл

{

var data = service.ExportCartToExcel();

return File(data, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", "CheckOnline.xlsx");

}

[HttpGet("getById/{id}")]

public Order GetById(int id) //метод выборки по id

{

return service.GetById(id);

}

[HttpDelete("delete/{id}")]

public bool Delete(int id) //метод удаления

{

return service.Remove(id);

}

[HttpPut("edit")]

public bool Update(Order order) //метод редактирования

{

return service.Update(order);

}

}

}

//ProductOrderInfoController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Services.DBServices;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace HardwareStoreServer.Controllers

{

//создание http путей

[Route("api/[controller]")]

[EnableCors("AllowMyOrigin")] //подключение cors для доступа в контроллеру сервера

public class ProductOrderInfoController : Controller

{

private readonly DBProductOrderInfoService service; //определение сервиса

public ProductOrderInfoController(DBProductOrderInfoService service)

{

this.service = service; //присваение сервиса

}

[HttpPost("create")]

public bool Create(ProductOrderInfo productOrderInfo) //метод добавления

{

return service.Create(productOrderInfo);

}

[HttpGet("getAll")]

public IList<ProductOrderInfo> GetAll() //метод выборки

{

return service.GetAll();

}

[HttpGet("getById/{id}")]

public ProductOrderInfo GetById(int id) //метод выборки по id

{

return service.GetById(id);

}

[HttpDelete("delete/{id}")]

public bool Delete(int id) //метод удаления

{

return service.Remove(id);

}

[HttpPut("edit")]

public bool Update(ProductOrderInfo productOrderInfo) //метод редактирования

{

return service.Update(productOrderInfo);

}

}

}

//ProductController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using HardwareStoreServer.Models.DBModels;

using HardwareStoreServer.Services.DBServices;

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace HardwareStoreServer.Controllers

{

//создание http путей

[Route("api/[controller]")]

[EnableCors("AllowMyOrigin")] //подключение cors для доступа в контроллеру сервера

public class ProductController : Controller

{

private readonly DBProductService service; //определение сервиса

public ProductController(DBProductService service)

{

this.service = service; //присваение сервиса

}

[HttpPost("create")]

public bool Create(Product product) //метод добавления

{

return service.Create(product);

}

[HttpGet("getAll")]

public IList<Product> GetAll() //метод выборки

{

return service.GetAll();

}

[HttpGet("getById/{id}")]

public Product GetById(int id) //метод выборки по id

{

return service.GetById(id);

}

[HttpDelete("delete/{id}")]

public bool Delete(int id) //метод удаления

{

return service.Remove(id);

}

[HttpPut("edit")]

public bool Update(Product product) //метод редактирования

{

return service.Update(product);

}

}

}

//RegistrationForm.js

import React, { Component } from 'react';

import { Form, Button, ButtonGroup, Alert, Container, Row, Col } from 'react-bootstrap';

import axios from 'axios';

import { SHA256 } from 'crypto-js';

import { Redirect } from 'react-router';

export default class RegistrationForm extends Component {

constructor(props) {

super(props); //передача параметров, принадлежащие родителю объекта

this.state = { error: '', redirect: false, loading: false } //инициализация начальных состояний компонента

}

hashingPassword = (password) => { //функция хэширования пароля

return SHA256(password).toString();

}

handleSubmit = (event) => { //обработчик события

event.preventDefault();

this.setState({ loading: true });

if (event.target.password.value === event.target.password2.value) {

axios.post(`https://localhost:5001/api/Client/create`, {

//отправка данных введённых в форму

name: event.target.name.value,

surname: event.target.surname.value,

adress: event.target.city.value + ', ' + event.target.street.value + ', д. ' + event.target.house.value + ', кв. ' + event.target.flat.value,

number: event.target.number.value,

email: event.target.email.value,

login: event.target.login.value,

password: this.hashingPassword(event.target.password.value)

})

.then(res => this.setState({ error: '', redirect: true, loading: false })) //если статус ответа 200

.catch(error => this.setState({ error: 'Пользователь уже существует', loading: false })); //если статус ответа ошибка

} else {

this.setState({ error: 'Пароли не совпадают', loading: false }); //если пароли не совпадают

}

}

//код представления компонента

render() {

const { redirect, loading } = this.state; //деструктуризация состояний

return (

<div>

{redirect ? <Redirect to={`/`} /> : null}

<Container className='d-flex flex-column justify-content-center align-items-center'>

<Row>

<Col>

<h1 align='center'>Регистрация</h1>

<hr />

<Form onSubmit={this.handleSubmit}>

<Form.Label>Ваше имя</Form.Label>

<Form.Control type="text" name='name' required placeholder="Ваше имя" />

<Form.Label>Ваша фамилия</Form.Label>

<Form.Control type="text" name='surname' required placeholder="Ваше фамилия" />

<Row>

<Col>

<Form.Label>Город</Form.Label>

<Form.Control type="text" name='city' required placeholder="Город"></Form.Control>

</Col>

<Col>

<Form.Label>Улица</Form.Label>

<Form.Control type="text" name='street' required placeholder="Улица"></Form.Control>

</Col>

<Col>

<Form.Label>Номер дома</Form.Label>

<Form.Control type="number" name='house' required placeholder="Номер дома"></Form.Control>

</Col>

<Col>

<Form.Label>Номер квартиры</Form.Label>

<Form.Control type="number" name='flat' required placeholder="Номер квартиры"></Form.Control>

</Col>

</Row>

<Row>

<Col>

<Form.Label>Номер телефона</Form.Label>

<Form.Control type="tel" name="number" defaultValue='+375' required maxLength={13}></Form.Control>

</Col>

<Col>

<Form.Label>Электронная почта</Form.Label>

<Form.Control type="email" name="email" required placeholder="Электронная почта"></Form.Control>

</Col>

</Row>

<Form.Label>Логин</Form.Label>

<Form.Control type="text" name='login' required placeholder="Логин" />

<Form.Label>Пароль</Form.Label>

<Form.Control type="password" name="password" required placeholder="Пароль" />

<Form.Label>Повтор пароля</Form.Label>

<Form.Control type="password" name="password2" required placeholder="Повтор пароля" />

<br />

{(this.state.error !== '') ? (<Alert className='d-flex justify-content-center' variant='danger'>{this.state.error}</Alert>) : ''}

<br />

<ButtonGroup className='d-flex justify-content-center align-items-center' variant='horizontal'>

<Button className='active' variant='outline-dark' size="lg" type="submit">{loading ? 'Ожидание...' : 'Зарегистрироваться'}</Button>

<Button variant='outline-dark' size="lg" onClick={() => { this.setState({ redirect: true }) }}>

Отмена</Button>

</ButtonGroup>

</Form>

</Col>

</Row>

</Container>

</div>

)

}

}

//LoginForm.js

import React, { Component } from 'react';

import { Form, Button, Container, Row, Col, Alert } from 'react-bootstrap';

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

import axios from 'axios';

import { ToggleButtons } from '..';

import { connect } from 'react-redux';

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

import { updateRole } from '../../redux/actions/ActionsRole';

import { SHA256 } from 'crypto-js';

import { Redirect } from 'react-router-dom';

import { clearCart } from '../../redux/actions/ActionsCart';

class LoginForm extends Component {

constructor(props) {

super(props); //передача параметров, принадлежащие родителю объекта

this.state = { radioValue: 0, error: '', login: '', redirectClient: false, redirectAdmin: false, loading: false } //определение начальных состояний компонента

}

hashingPassword = (password) => { //функция хэширования пароля

return SHA256(password).toString();

}

handleSubmit = (event) => { //обработчик события

event.preventDefault();

this.setState({ loading: true });

axios.get(`https://localhost:5001/api/Client/getByLogin/${event.target.login.value}`)

.then(res => {

if (res.data !== '') {

const password = this.hashingPassword(event.target.password.value);

if (password === res.data.password) {

this.props.dispatch(updateRole('client'));

this.props.dispatch(clearCart());

this.setState({ error: '', login: event.target.login.value, redirectClient: true, loading: false }); //если статус ответа 200 и пароли совпадают

} else {

this.setState({ error: 'Неверный пароль', loading: false }); //если статус ответа 200 и пароли не совпадают

}

} else {

this.setState({ error: 'Пользователь не зарегистрирован', loading: false }) //если ответ пользователь не существует

}

})

.catch(err => {

this.setState({ error: 'Ошибка сервера', loading: false }) //если статус ответа ошибка

})

}

handleSubmitAdmin = (event) => { //обработчик события

event.preventDefault();

this.setState({ loading: true });

setTimeout(() => {

if ((event.target.loginAdmin.value === 'admin') && (event.target.passwordAdmin.value === 'qwerty')) {

this.props.dispatch(updateRole('admin'));

this.setState({ error: '', redirectAdmin: true, loading: false }) //если данные введены верно

} else {

this.setState({ error: 'Неверный логин или пароль', loading: false }) //если данные введены неверно

}

}, 1000)

}

//код представления компонента

render() {

const { radioValue, error, redirectClient, redirectAdmin, login, loading } = this.state; //деструктуризация состояний

return (

<div>

{redirectClient ? <Redirect to={`/client/${login}/home`} /> : null}

{redirectAdmin ? <Redirect to='/admin/home' /> : null}

<Container className='mt-5 d-flex flex-column justify-content-center align-items-center'>

<Row>

<Col>

<h1 align='center'>Аккаунт</h1>

<hr />

<ToggleButtons items={['Клиент', 'Администратор']} variant='outline-dark' value={radioValue} onChange={(radioValue) => this.setState({ radioValue })} />

{(radioValue === 0) ? (

<Form style={{ display: 'block' }} onSubmit={this.handleSubmit}>

<Form.Group>

<Form.Label>Логин</Form.Label>

<Form.Control required name='login' type="text" placeholder="Логин" defaultValue='' />

<Form.Label>Пароль</Form.Label>

<Form.Control required name='password' type="password" placeholder="Пароль" defaultValue='' />

<br />

{(error !== '') ? (<Alert className='d-flex justify-content-center' variant='danger'>{error}</Alert>) : ''}

<br />

<Button type='submit' variant='outline-dark' size="lg" block>{!loading ? 'Войти' : 'Ожидание...'}</Button>

<br />

<Link className='d-flex justify-content-center' to='/registration'>Регистрация</Link>

</Form.Group>

</Form>

) :

(

<Form onSubmit={this.handleSubmitAdmin}>

<Form.Group>

<Form.Label>Логин</Form.Label>

<Form.Control required name='loginAdmin' type="text" placeholder="Логин" defaultValue='' />

<Form.Label>Пароль</Form.Label>

<Form.Control required name='passwordAdmin' type="password" placeholder="Пароль" defaultValue='' />

<br />

{(error !== '') ? (<Alert className='d-flex justify-content-center' variant='danger'>{error}</Alert>) : ''}

<br />

<Button type='submit' variant='outline-dark' size="lg" block>{!loading ? 'Войти' : 'Ожидание...'}</Button>

</Form.Group>

</Form>

)

}

</Col>

</Row>

</Container>

</div>

)

}

}

const mapStateToProps = state => {

return { role: state.roleReducer.role, items: state.cartReducer.cartItems }; //деструкторизация состояний из redux

};

export default withRouter(connect(mapStateToProps)(LoginForm)); //подключение redux

//Cart.js

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

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

import axios from 'axios';

import qs from 'querystring';

import { ButtonToolbar, ButtonGroup, Button, Table, Alert, Form } from 'react-bootstrap';

import DeleteIcon from '@material-ui/icons/Delete';

import { useDispatch, useSelector } from 'react-redux';

import { deleteCartItem, updateQuantityCartItem } from '../../redux/actions/ActionsCart';

import SnackBar from '@material-ui/core/Snackbar';

import MuiAlert from '@material-ui/lab/Alert';

import ExportCSV from '../../ExcelCheck/Check';

import ScrollTop from '../ScrollTop';

import { Carousel, PaymentSystem, Spinner } from '..';

import { connect } from 'react-redux';

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

import emailjs from "emailjs-com";

import { init } from "emailjs-com";

import { fetchUser, setLoad } from '../../redux/actions/ActionFetchData';

init("user\_ZNDi6J5mnaAE4KFSr9mch");

const Cart = ({ match }) => { //передача параметров, принадлежащие родителю объекта

const dispatch = useDispatch(); //инициализация метода, для обновления глобального состояния в хранилище Redux

const { user, loading } = useSelector(({ fetchDataReducer }) => fetchDataReducer); //выборка глобальных состояний из хранилища Redux

const { cartItems } = useSelector(({ cartReducer }) => cartReducer);

const [open, setOpen] = useState(false); //инициализация начальных состояний компонента

const [message, setMessage] = useState('');

const [severity, setSeverity] = useState('');

const [buttonHiddenCheck, setButtonHiddenCheck] = useState(true);

const [buttonHiddenPayment, setButtonHiddenPayment] = useState(true);

const fileName = 'CheckOnline';

useEffect(() => { //определение метода жизненого цикла компонента

if (user.login !== match.params.login || user === {}) {

dispatch(setLoad(true));

dispatch(fetchUser(match.params.login));

}

}, [])

const requestOrder = () => { //отработчик события (создание заказа в бд)

if (cartItems && cartItems.length === 0) { //проверка на пустоту корзины

setOpen(true);

setMessage('Список пуст');

setSeverity('warning');

} else {

let arrOrder = [];

cartItems.map(item => {

arrOrder.push({ productId: item.id, quantity: item.quantity });

})

const totalPrice = cartItems.reduce((accumulator, product) => {

return accumulator + product.price \* product.quantity;

}, 0);

axios.post(`https://localhost:5001/api/Order/makeOrder/?${qs.stringify({

ClientId: user.id,

TotalPrice: totalPrice

})}`, arrOrder)

.then(res => { //если статус ответа 200

setOpen(true);

setMessage('Заказ оформлен');

setSeverity('success');

setButtonHiddenPayment(false)

})

.catch(err => console.log(err)) //если статус ответа ошибка

}

}

const date = (dateOrder) => { //метод форматирования даты

var date = new Date(dateOrder);

return date.toLocaleDateString();

}

const htmlMess = (user, order, products) => { //метод для формирования сообщения на почту

return (

`<div>

<p>Здравствуйте, ${user.name} ${user.surname}.</p>

<p>Ваш заказ №<b>${order.id}</b> от ${date(order.date)} на сумму <b>${order.totalPrice}</b>р.</p>

<h3>Продукты</h3>

<hr>

${products.map(prod => {

return (

`<div>

<br/>

<img src=${prod.images[0].url} alt='Error' style='float:left; margin: 7px 7px 7px 7px' height='150px' width='150px' />

<h3>${prod.name}</h3>

<h4>Кол-во: ${prod.quantity}</h4>

<h4><span style='color: red'>${prod.price}</span> BYN</h4><br/>

</div>`

)

})}

<br/><br/><hr>

<p>С наилучшими пожеланиями, интернет-магазин i-Bozh shop.</p>

</div>`

);

}

const createLetter = () => { //метод создания сообщения

const templateId = "template\_o16n64t";

let products = [];

cartItems.map((product, index) => {

products.push(` ${index + 1}) ${product.name} x ${product.quantity} шт.`)

});

axios.get(`https://localhost:5001/api/Order/getAll`)

.then(res => {

const order = res.data[res.data.length - 1];

sendFeedback(templateId, {

email: user.email,

html: htmlMess(user, order, cartItems)

});

})

}

const sendFeedback = (templateId, variables) => { //метод отправки сообщения на почту

emailjs

.send("service\_e2uee2x", templateId, variables)

.then((res) => {

setOpen(true);

setMessage(`Письмо отправлено на почту ${user.email}`);

setSeverity('success');

})

.catch((err) => console.error("Oh well, you failed. Here some thoughts on the error that occured:", err));

}

//код представления компонента

return (

<div className='container'>

<SnackBar open={open} autoHideDuration={3000} onClose={() => setOpen(false)}>

<MuiAlert onClose={() => setOpen(false)} severity={severity} variant="filled">

<b className='snackBar-label'>{message}</b>

</MuiAlert>

</SnackBar>

{(!loading) ? (

<h1 className='mt-2 d-flex justify-content-left align-items-center'>

<Avatar className='mr-2' style={{ backgroundColor: '#FF7700' }}>{(`${user.name}`).split('')[0].toLocaleUpperCase()}</Avatar>

{user.name + ' ' + user.surname}

</h1>

) : (null)}

<h1 className='mt-2 d-flex justify-content-center align-items-center'>Корзина</h1>

<ButtonToolbar className='mb-2 float-right'>

<ButtonGroup>

<ExportCSV hidden={buttonHiddenCheck} fileName={fileName} buttonHidden={hidden => setTimeout((hidden) => setButtonHiddenCheck(hidden), 10000, hidden)} />

</ButtonGroup>

<ButtonGroup>

<PaymentSystem totalPrice={cartItems.reduce((accumulator, product) => {

return accumulator + product.price \* product.quantity;

}, 0)} user={user} hidden={buttonHiddenPayment} onHidden={hidden => setButtonHiddenCheck(hidden)} createLetter={() => createLetter()} />

</ButtonGroup>

<ButtonGroup>

<Button className='mr-2' variant="outline-dark" onClick={() => requestOrder()}>Оформить заказ</Button>

</ButtonGroup>

</ButtonToolbar>

<h5>Товаров в корзине: {cartItems.reduce((accumulator, product) => {

return accumulator + product.quantity;

}, 0)}</h5>

<h5>Итого: <span className='price'>{cartItems.reduce((accumulator, product) => {

return accumulator + product.price \* product.quantity;

}, 0)}</span> BYN</h5>

{(cartItems && cartItems.length !== 0) ? (

<Table className='mt-4' responsive="xl">

<thead>

<tr>

<th>Продукт</th>

<th>Название</th>

<th>Характеристики</th>

<th>Количество</th>

<th>Цена</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{cartItems.map(product =>

<tr key={product.id}>

<td><Carousel images={product.images} height='150px' width='150px' /></td>

<td style={{ maxWidth: '200px' }}>{product.name}</td>

<td style={{ maxWidth: '210px' }}>

<div>

<b>Модель: </b>{product.modal}<br />

<b>Год выпуска: </b>{product.year}<br />

<b>Срок гарантии: </b>{product.warranty}

</div>

</td>

<td>

<Form.Control

type="number"

required

style={{ width: '80px' }}

value={product.quantity}

onChange={e => {

let quantity = parseInt(e.target.value, 10);

if (isNaN(quantity)) return 0;

if (quantity < 0) return;

dispatch(updateQuantityCartItem({ id: product.id, quantity }));

}} />

</td>

<td><b className='price product'>{product.price}</b> BYN</td>

<td>

<Button className="mr-2"

variant="ligth"

onClick={() => {

dispatch(deleteCartItem(product.id));

}}>

{<DeleteIcon />}

</Button>

</td>

</tr>

)}

</tbody>

</Table>

) : (<Alert className='mt-2 d-flex justify-content-center' variant='secondary'>Список пуст</Alert>)}

<ScrollTop />

</div>

)

}

export default Cart;

//ExcelCheck/Check.js

import React from 'react'

import Button from 'react-bootstrap/Button';

import axios from 'axios';

const ExportCSV = ({ hidden, fileName, buttonHidden }) => { //передача параметров, принадлежащие родителю объекта

const exportToCSV = (fileName) => { //отработчик события (создание excel документа)

axios({

url: `https://localhost:5001/api/Order/excelCart`,

method: 'GET',

responseType: 'blob', // important

}).then((response) => { //если статус ответа 200

const url = window.URL.createObjectURL(new Blob([response.data]));

const link = document.createElement('a'); //создане ссылки

link.href = url; //присваение адресса ссылки

link.setAttribute('download', `${fileName}.xlsx`); //добавление к ссылке

document.body.appendChild(link); //перемещение на страницу ссылки

link.click(); //событие click

buttonHidden(!hidden); // change state buttonHidden

});

}

//код представления компонента

return (

<Button className='mr-2' variant="outline-dark" hidden={hidden} onClick={(e) => exportToCSV(fileName)}>Чек</Button>

)

}

export default ExportCSV;

//Order.js

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

import axios from 'axios';

import qs from 'querystring';

import { Button, Table, Alert, Form, Row, Col, InputGroup } from 'react-bootstrap';

import SnackBar from '@material-ui/core/Snackbar';

import MuiAlert from '@material-ui/lab/Alert';

import ScrollTop from '../ScrollTop';

import { fetchClients, fetchOrders, setLoad } from '../../redux/actions/ActionFetchData';

import { useDispatch, useSelector } from 'react-redux';

import { Spinner } from '..';

const Order = () => {

const dispatch = useDispatch(); //инициализация метода, для обновления глобального состояния в хранилище Redux

const { orders, clients, loading } = useSelector(({ fetchDataReducer }) => fetchDataReducer); //выборка глобальных состояний из хранилища Redux

const [open, setOpen] = useState(false); //инициализация начальных состояний компонента

const [message, setMessage] = useState('');

const [severity, setSeverity] = useState('');

const [to, setTo] = useState('');

const [from, setFrom] = useState('');

const date = (dateSupply) => { //метод форматирования даты

var date = new Date(dateSupply);

return date.toLocaleDateString('en-GB');

}

const handleSubmit = (event) => { //обработчик события (создание excel документа)

event.preventDefault();

axios({

url: `https://localhost:5001/api/Order/excelOrders?${qs.stringify({

From: event.target.from.value,

To: event.target.to.value

})}`,

method: 'GET',

responseType: 'blob', // important

})

.then((response) => { ////если статус ответа 200

const url = window.URL.createObjectURL(new Blob([response.data]));

const link = document.createElement('a'); //создане ссылки

link.href = url; //присваение адресса ссылки

link.setAttribute('download', 'Report.xlsx'); //добавление к ссылке

document.body.appendChild(link); //перемещение на страницу ссылки

link.click(); //событие click

})

.catch(err => { //если статус ответа ошибка

setOpen(true);

setMessage('Ошибка');

setSeverity('warning');

});

}

const filterList = (rows) => { //метод фильтрации записей по дате оформления заказа

let newList = rows;

if (from !== '') {

newList = rows.filter(row => row.date >= from);

}

if (to !== '') {

newList = newList.filter(row => row.date < to);

}

return newList;

}

useEffect(() => { //метод жизенного цикла

if (orders.length === 0) {

dispatch(setLoad(true));

dispatch(fetchOrders()); //обращение к экшкн-функции в хранилище Redux

} if (clients.length === 0) {

dispatch(fetchClients());

}

}, [])

const filterOrder = filterList(orders); //отфильтрованный лист записей

//код представления компонента

return (

<div>

<SnackBar open={open} autoHideDuration={8000} onClose={() => setOpen(false)}>

<MuiAlert onClose={() => setOpen(false)} severity={severity} variant="filled">

<b className='snackBar-label'>{message}</b>

</MuiAlert>

</SnackBar>

<h1 className='mt-2 d-flex justify-content-center align-items-center'>Заказы</h1>

<Form className='mr-3' onSubmit={handleSubmit}>

<Row className='order-settings'>

<Col>

<InputGroup>

<InputGroup.Prepend>

<InputGroup.Text>С</InputGroup.Text>

</InputGroup.Prepend>

<Form.Control type="date" name='from' required onChange={(e) => setFrom(e.target.value)} />

</InputGroup>

</Col>

<Col>

<InputGroup>

<InputGroup.Prepend>

<InputGroup.Text>До</InputGroup.Text>

</InputGroup.Prepend>

<Form.Control type="date" name='to' required onChange={(e) => setTo(e.target.value)} />

</InputGroup>

</Col>

<Button variant='outline-dark' type='submit'>Сформировать отчёт</Button>

</Row>

</Form>

{(!loading) ? ((filterOrder && filterOrder.length !== 0) ? (

<Table className='mt-4' responsive="xl">

<thead>

<tr>

<th>Номер заказа</th>

<th>Клиент</th>

<th>Дата заказа</th>

<th>Сумма заказа</th>

</tr>

</thead>

<tbody>

{filterOrder.map(order =>

<tr key={order.id}>

<td>{order.id}</td>

<td>{clients.filter(client => client.id === order.clientId).map(client => { return client.name + ' ' + client.surname })}</td>

<td>{date(order.date)}</td>

<td>{order.totalPrice}</td>

</tr>

)}

</tbody>

</Table>

) : (<Alert className='mt-2 d-flex justify-content-center' variant='secondary'>Список пуст</Alert>))

: (

<div className='mt-2 d-flex justify-content-center'>

<Spinner />

</div>

)}

<ScrollTop />

</div>

)

}

export default Order;

//AddProduct.js

import React, { useState } from 'react';

import { Modal, Row, Col, Form } from 'react-bootstrap';

import { Button } from 'react-bootstrap';

import SnackBar from '@material-ui/core/Snackbar';

import IconButton from "@material-ui/core/IconButton";

import CloseIcon from '@material-ui/icons/Close';

import axios from 'axios';

import Tooltip from '@material-ui/core/Tooltip';

import Spinner from '../Spinner/Spinner';

import { useDispatch } from 'react-redux';

import { fetchProducts } from '../../redux/actions/ActionFetchData';

const AddProductModal = ({ show, onHide, brands, types, suppliers, images }) => { //передача параметров, принадлежащие родителю объекта

const dispatch = useDispatch(); //инициализация метода, для обновления глобального состояния в хранилище Redux

const [snackBaropen, setSnackBaropen] = useState(false); //инициализация начальных состояний компонента

const [snackBarMessage, setSnackBarMessage] = useState('');

const [loading, setLoading] = useState(false);

const [imagesProduct, setImages] = useState([]);

const handleSubmit = (event) => { //обработчик события (добавления товара в бд)

event.preventDefault();

axios.post(`https://localhost:5001/api/Product/create`, {

name: event.target.name.value,

year: event.target.year.value,

brandId: parseInt(event.target.brand.value),

typeId: parseInt(event.target.type.value),

modal: event.target.modal.value,

warranty: parseInt(event.target.warranty.value),

price: parseInt(event.target.price.value),

amount: parseInt(event.target.amount.value),

supplyId: parseInt(event.target.supplier.value),

images: imagesProduct

})

.then(res => { setSnackBaropen(true); setSnackBarMessage('Успешно добавлено'); dispatch(fetchProducts()); setImages([]) }) //если статус ответа 200

.catch(error => { setSnackBaropen(true); setSnackBarMessage('Ошибка добавления') }); //если статус ответа ошибка

}

const uploadImage = async event => { //метод для отправки фото товара на облако

const files = event.target.files;

const data = new FormData();

for (let i = 0; i !== files.length; ++i) {

data.append('file', files[i]);

data.append('upload\_preset', 'hardware-store');

setLoading(true)

const res = await fetch(`https://api.cloudinary.com/v1\_1/dzlhauo5h/image/upload`,

{

method: 'POST',

body: data

}

);

const file = await res.json();

setLoading(false);

setImages(imagesProduct => [...imagesProduct, { url: file.secure\_url }]);

}

}

const snackBarClose = () => setSnackBaropen(false); //метод смены состояния компонента

const imagesView =

imagesProduct && imagesProduct.map((image, index) =>

<img className='mt-2 mr-2' key={index} src={image.url} style={{ width: '300px' }} alt='Error, sorry...' />

); //представление листа фото в форме

//код представления компонента

return (

<div className='container'>

<SnackBar

anchorOrigin={{ vertical: 'bottom', horizontal: 'center' }}

open={snackBaropen}

autoHideDuration={1000}

onClose={snackBarClose}

message={<span id='message-id'>{snackBarMessage}</span>}

action={[

<IconButton color="inherit" size="small"

onClick={snackBarClose}

><CloseIcon /></IconButton>

]} />

<Modal size='xl'

show={show}

onHide={onHide}

aria-labelledby="contained-modal-title-vcenter"

centered

>

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-vcenter">

Добавление нового товара

</Modal.Title>

</Modal.Header>

<Modal.Body>

<Row key={1}>

<Col>

<Form onSubmit={handleSubmit}>

<Form.Group>

<div>

<Form.File multiple required onChange={uploadImage} label='Выберите картинку для товара' data-browse='Выбрать' custom />

{loading ? (

<div>

{imagesView}

<div className='mt-4' style={{ display: 'inline-flex' }}><Spinner /></div>

</div>

) : (imagesProduct && imagesProduct.map((image, index) =>

<img className='mt-2 mr-2' key={index} src={image.url} style={{ width: '300px' }} alt='' />

))

}

</div>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Название</Form.Label>

<Form.Control

type="text"

name="name"

required

placeholder="Название" />

</Col>

<Col>

<Form.Label>Модель</Form.Label>

<Form.Control

type="text"

name="modal"

required

placeholder="Модель" />

</Col>

</Row>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Категория</Form.Label>

<Form.Control as="select" name='type'>

{types.map(type =>

<Tooltip key={type.id} title={type.name}>

<option key={type.id} value={type.id}>{type.name}</option>

</Tooltip>

)}

</Form.Control>

</Col>

<Col>

<Form.Label>Бренд</Form.Label>

<Form.Control as="select" name='brand'>

{brands.map(brand =>

<Tooltip key={brand.id} title={brand.name}>

<option key={brand.id} value={brand.id}>{brand.name}</option>

</Tooltip>

)}

</Form.Control>

</Col>

</Row>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Год выпуска</Form.Label>

<Form.Control

type="text"

name="year"

required

placeholder="Год выпуска" />

</Col>

<Col>

<Form.Label>Срок гарантии</Form.Label>

<Form.Control

type="text"

name="warranty"

required

placeholder="Срок гарантии" />

</Col>

<Col>

<Form.Label>Количество на складе</Form.Label>

<Form.Control

type="text"

name="amount"

required

placeholder="Количество на складе" />

</Col>

</Row>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Поставщик</Form.Label>

<Form.Control as="select" name='supplier'>

{suppliers.map(supplier =>

<Tooltip key={supplier.id} title={supplier.nameOrganization}>

<option key={supplier.id} value={supplier.id}>{supplier.nameOrganization}</option>

</Tooltip>

)}

</Form.Control>

</Col>

<Col>

<Form.Label>Цена</Form.Label>

<Form.Control

type="text"

name="price"

required

placeholder="Цена" />

</Col>

</Row>

</Form.Group>

<Form.Group>

<Button variant="light" type="submit">

Добавить товар

</Button>

</Form.Group>

</Form>

</Col>

</Row>

</Modal.Body>

<Modal.Footer>

<Button variant="light" onClick={onHide}>

Закрыть

</Button>

</Modal.Footer>

</Modal>

</div>

)

}

export default AddProductModal;

//EditProduct.js

import React, { useState } from 'react';

import { Modal, Row, Col, Form } from 'react-bootstrap';

import { Button } from 'react-bootstrap';

import SnackBar from '@material-ui/core/Snackbar';

import IconButton from "@material-ui/core/IconButton";

import CloseIcon from '@material-ui/icons/Close';

import axios from 'axios';

import Tooltip from '@material-ui/core/Tooltip';

import { Spinner } from '..';

import { useDispatch } from 'react-redux';

import { fetchProducts } from '../../redux/actions/ActionFetchData';

const EditProductModal = ({ show, onHide, id, name, year, type, brand, modal, warranty, amount, supply, price, images, types, brands, suppliers }) => { //передача параметров, принадлежащие родителю объекта

const dispatch = useDispatch(); //инициализация метода, для обновления глобального состояния в хранилище Redux

const [snackBaropen, setSnackBaropen] = useState(false); //инициализация начальных состояний компонента

const [snackBarMessage, setSnackBarMessage] = useState('');

const [loading, setLoading] = useState(false);

const [imagesState, setImagesState] = useState([]);

const handleSubmit = (event) => { //обработчик события (изменение товара в бд)

event.preventDefault();

let imagesArray = [];

if (images.length !== 0) {

images.map(image =>

imagesArray.push({ url: image.url })

)

}

axios.put(`https://localhost:5001/api/Product/edit`, {

id: id,

name: event.target.name.value,

year: event.target.year.value,

brandId: parseInt(event.target.brand.value),

typeId: parseInt(event.target.type.value),

modal: event.target.modal.value,

warranty: parseInt(event.target.warranty.value),

price: parseInt(event.target.price.value),

amount: parseInt(event.target.amount.value),

supplyId: parseInt(event.target.supplier.value),

images: (imagesState.length !== 0) ? (imagesState) : (imagesArray)

})

.then(res => { setSnackBaropen(true); setSnackBarMessage('Успешно обновлён'); dispatch(fetchProducts()); }) //если статус ответа 200

.catch(error => { setSnackBaropen(true); setSnackBarMessage('Ошибка редактирования') }); //если статус ответа ошибка

}

const uploadImage = async event => { //метод для отправки фото товара на облако

const files = event.target.files;

const data = new FormData();

for (let i = 0; i !== files.length; ++i) {

data.append('file', files[i]);

data.append('upload\_preset', 'hardware-store');

setLoading(true)

const res = await fetch(`https://api.cloudinary.com/v1\_1/dzlhauo5h/image/upload`,

{

method: 'POST',

body: data

}

);

const file = await res.json();

setLoading(false);

setImagesState(imagesState => [...imagesState, { url: file.secure\_url }])

}

}

const snackBarClose = () => setSnackBaropen(false); //метод смены состояния компонента

const imagesPropsView =

images && images.map((image, index) =>

<img className='mt-2 mr-2' key={index} src={image.url} style={{ width: '300px' }} alt='Error, sorry...' />

); //представление передаваемого листа (из компонента родителя) фото в форме

const imagesStateView =

imagesState && imagesState.map((image, index) =>

<img className='mt-2 mr-2' key={index} src={image.url} style={{ width: '300px' }} alt='Error, sorry...' />

); //представление листа фото в форме

//код представления компонента

return (

<div className='container'>

<SnackBar

anchorOrigin={{ vertical: 'bottom', horizontal: 'center' }}

open={snackBaropen}

autoHideDuration={1000}

onClose={snackBarClose}

message={<span id='message-id'>{snackBarMessage}</span>}

action={[

<IconButton color="inherit" size="small"

onClick={snackBarClose}

><CloseIcon /></IconButton>

]} />

<Modal size='xl'

show={show}

onHide={onHide}

aria-labelledby="contained-modal-title-vcenter"

centered

>

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-vcenter">

Редактирование товара

</Modal.Title>

</Modal.Header>

<Modal.Body>

<Row>

<Col>

<Form onSubmit={handleSubmit}>

<Form.Group controlId="image">

<div>

<Form.File multiple name='file' onChange={uploadImage} label='Выберите картинку для товара' data-browse='Выбрать' custom />

{loading ? (

<div>

{imagesStateView}

<div className='mt-4' style={{ display: 'inline-flex' }}><Spinner /></div>

</div>

) : ((imagesState.length === 0) ? (imagesPropsView) : (imagesStateView))}

</div>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Название</Form.Label>

<Form.Control

type="text"

name="name"

required

defaultValue={name}

placeholder="Название" />

</Col>

<Col>

<Form.Label>Модель</Form.Label>

<Form.Control

type="text"

name="modal"

required

defaultValue={modal}

placeholder="Модель" />

</Col>

</Row>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Категория</Form.Label>

<Form.Control as="select" name='type'

defaultValue={type}>

{types.map(type =>

<Tooltip key={type.id} title={type.name}>

<option key={type.id} value={type.id}>{type.name}</option>

</Tooltip>

)}

</Form.Control>

</Col>

<Col>

<Form.Label>Бренд</Form.Label>

<Form.Control as="select" name='brand'

defaultValue={brand}>

{brands.map(brand =>

<Tooltip key={brand.id} title={brand.name}>

<option key={brand.id} value={brand.id}>{brand.name}</option>

</Tooltip>

)}

</Form.Control>

</Col>

</Row>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Год выпуска</Form.Label>

<Form.Control

type="text"

name="year"

required

defaultValue={year}

placeholder="Год выпуска" />

</Col>

<Col>

<Form.Label>Срок гарантии</Form.Label>

<Form.Control

type="text"

name="warranty"

required

defaultValue={warranty}

placeholder="Срок гарантии" />

</Col>

<Col>

<Form.Label>Количество на складе</Form.Label>

<Form.Control

type="text"

name="amount"

required

defaultValue={amount}

placeholder="Количество на складе" />

</Col>

</Row>

</Form.Group>

<Form.Group>

<Row>

<Col>

<Form.Label>Поставщик</Form.Label>

<Form.Control as="select" name='supplier'

defaultValue={supply}>

{suppliers.map(supplier =>

<Tooltip key={supplier.id} title={supplier.nameOrganization}>

<option key={supplier.id} value={supplier.id}>{supplier.nameOrganization}</option>

</Tooltip>

)}

</Form.Control>

</Col>

<Col>

<Form.Label>Цена</Form.Label>

<Form.Control

type="text"

name="price"

required

defaultValue={price}

placeholder="Цена" />

</Col>

</Row>

</Form.Group>

<Form.Group>

<Button variant="light" type="submit">

Изменить товар

</Button>

</Form.Group>

</Form>

</Col>

</Row>

</Modal.Body>

<Modal.Footer>

<Button variant="light" onClick={onHide}>

Закрыть

</Button>

</Modal.Footer>

</Modal>

</div>

)

}

export default EditProductModal;

//ProductOfList.js

import React, { useState } from 'react';

import { ListGroup, Row, Col, Container } from 'react-bootstrap';

import { ButtonToolbar, Button } from 'react-bootstrap';

import EditProductModal from './EditProduct';

import EditIcon from '@material-ui/icons/Edit';

import DeleteIcon from '@material-ui/icons/Delete';

import AddShoppingCartRoundedIcon from '@material-ui/icons/AddShoppingCartRounded';

import RemoveShoppingCartRoundedIcon from '@material-ui/icons/RemoveShoppingCartRounded';

import axios from 'axios';

import { useDispatch, useSelector } from 'react-redux';

import { addItemInCart } from "../../redux/actions/ActionsCart";

import SnackBar from '@material-ui/core/Snackbar';

import MuiAlert from '@material-ui/lab/Alert';

import { Carousel } from '..';

import { fetchProducts } from '../../redux/actions/ActionFetchData';

const ProductOfList = ({ product, role }) => { //передача параметров, принадлежащие родителю объекта

const dispatch = useDispatch(); //инициализация метода, для обновления

const { brands, types, suppliers } = useSelector(({ fetchDataReducer }) => fetchDataReducer) //выборка глобальных состояний из хранилища Redux

const [editModalShow, setEditModalShow] = useState(false); //инициализация начальных состояний компонента

const [open, setOpen] = useState(false);

const [Id, setId] = useState(null);

const [Name, setName] = useState('');

const [Year, setYear] = useState('');

const [Brand, setBrand] = useState(null);

const [Type, setType] = useState(null);

const [Modal, setModal] = useState('');

const [Warranty, setWarranty] = useState(null);

const [Amount, setAmount] = useState(0);

const [Supply, setSupply] = useState('');

const [Price, setPrice] = useState(0);

const [Images, setImages] = useState([]);

const editModalClose = () => setEditModalShow(false); //метод смены состояния компонента

const deleteProduct = (id) => { //обработчик события (удаление товара из бд)

if (window.confirm('Вы уверены?')) {

axios.delete(`https://localhost:5001/api/Product/delete/${id}`)

.then(res => dispatch(fetchProducts()))

.catch(error => console.log(error));

}

}

//код представления компонента

return (

<div>

<SnackBar open={open} autoHideDuration={400} onClose={() => setOpen(false)}>

<MuiAlert onClose={() => setOpen(false)} severity="success" variant="filled">

<b className='snackBar-label'>Товар добавлен</b>

</MuiAlert>

</SnackBar>

<ListGroup.Item className='mr-2 mt-2' key={product.id}>

<Row>

<Col md='auto' className='d-flex justify-content-center'>

<Carousel images={product.images} height='230px' width='16.4rem' />

</Col>

<Col>

<Container>

<h3 align='center' className='mt-2 mb-4'>{product.name}</h3>

<Row>

<Col>

Категория: {types.filter(type => type.id === product.typeId).map(type => { return type.name })}<br />

Бренд: {brands.filter(brand => brand.id === product.brandId).map(brand => { return brand.name })}<br />

Модель: {product.modal}<br />

Год выпуска: {product.year}<br />

</Col>

<Col>

Срок гарантии: {product.warranty}<br />

Количество на складе: {product.amount}<br />

Поставщик: {suppliers.filter(supplier => supplier.id === product.supplyId).map(supplier => { return supplier.nameOrganization + ', ' + supplier.adress + '; ' + supplier.number })}<br />

<div className='d-flex justify-content-end'><b className='price product'>{product.price}</b>BYN</div>

</Col>

</Row>

</Container>

</Col>

</Row>

<hr />

<ButtonToolbar className='d-flex justify-content-end'>

{(role === 'admin') ? (

<div>

<Button variant="light"

onClick={() => {

setEditModalShow(true);

setId(product.id);

setName(product.name);

setYear(product.year);

setBrand(product.brandId);

setType(product.typeId);

setModal(product.modal);

setWarranty(product.warranty);

setAmount(product.amount);

setSupply(product.supplyId);

setPrice(product.price);

setImages(product.images);

}}>

{<EditIcon />}

</Button>

<Button className='ml-2' variant="light"

onClick={() => deleteProduct(product.id)}>

{<DeleteIcon />}

</Button>

<EditProductModal

show={editModalShow}

onHide={editModalClose}

id={Id}

name={Name}

year={Year}

brand={Brand}

type={Type}

modal={Modal}

warranty={Warranty}

amount={Amount}

supply={Supply}

price={Price}

images={Images}

types={types}

brands={brands}

suppliers={suppliers} />

</div>

) : (

<div>

{product.amount > 0 ? <Button variant="light"

onClick={e => {

e.stopPropagation();

setOpen(!open);

dispatch(

addItemInCart({ ...product, quantity: 1 })

);

}}>

{<AddShoppingCartRoundedIcon />}</Button> :

<Button variant="light" disabled>{<RemoveShoppingCartRoundedIcon />}Sold out</Button>}

</div>

)}

</ButtonToolbar>

</ListGroup.Item>

</div>

)

}

export default ProductOfList;

//Products.js

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

import { ButtonToolbar, Button, FormControl, Form, FormGroup, Alert, ListGroup, Row } from 'react-bootstrap';

import AddProductModal from './AddProduct';

import ProductOfGrid from './ProductOfGrid';

import ProductOfList from './ProductOfList';

import AddIcon from '@material-ui/icons/Add';

import CheckBox from '../CheckBox';

import RadioBox from '../RadioBox';

import FormatListBulletedIcon from '@material-ui/icons/FormatListBulleted';

import ViewComfyIcon from '@material-ui/icons/ViewComfy';

import ScrollTop from '../ScrollTop';

import { Carousel, Spinner, ToggleButtons } from '..';

import { useDispatch, useSelector } from 'react-redux';

import { fetchBaners, fetchBrands, fetchProducts, fetchSuppliers, fetchTypes, setLoad } from '../../redux/actions/ActionFetchData';

const items = [

{ id: 1, label: 'Любой' },

{ id: 2, label: 'От большего к меньшему' },

{ id: 3, label: 'От меньшего к большему' }

]; //инициализация массива объектов сортировки

const Products = ({ role }) => { //передача параметров, принадлежащие родителю объекта

const dispatch = useDispatch(); //инициализация метода, для обновления глобального состояния в хранилище Redux

const { products, brands, types, baners, suppliers, loading } = useSelector(({ fetchDataReducer }) => fetchDataReducer); //выборка глобальных состояний из хранилища Redux

const [addModalShow, setAddModalShow] = useState(false); //инициализация начальных состояний компонента

const [search, setSearch] = useState('');

const [newFiltersBrands, setNewFiltersBrands] = useState([]);

const [newFiltersTypes, setNewFiltersTypes] = useState([]);

const [sortBy, setSortBy] = useState('');

const [grid, setGrid] = useState(true);

const addModalClose = () => setAddModalShow(false); //метод смены состояния компонента

useEffect(() => { //определение метода жизненого цикла компонента

if (products.length === 0) {

dispatch(setLoad(true));

dispatch(fetchProducts());//обращение к экшкн-функции в хранилище Redux

} if (brands.length === 0) {

dispatch(setLoad(true));

dispatch(fetchBrands());

} if (types.length === 0) {

dispatch(setLoad(true));

dispatch(fetchTypes());

} if (baners.length === 0) {

dispatch(setLoad(true));

dispatch(fetchBaners());

} if (suppliers.length === 0) {

dispatch(setLoad(true));

dispatch(fetchSuppliers());

}

}, [])

const searchPanel = (rows) => { //метод поиска товара по названию

return rows.filter((row) => row.name.toLowerCase().indexOf(search.toLocaleLowerCase()) > -1);

}

const filterList = (list) => { //метод фильтрации листа товаров по брендам и категориям

let newList = list;

if (newFiltersBrands && newFiltersBrands.length) {

newList = list.filter(a => newFiltersBrands.indexOf(a.brandId) > -1);

}

if (newFiltersTypes && newFiltersTypes.length) {

newList = newList.filter(a => newFiltersTypes.indexOf(a.typeId) > -1);

}

return newList;

}

const sortList = (list, sortType) => { //метод сортировки листа товаров по цене

let oldList = list;

if (sortType === 'От меньшего к большему') {

oldList = list.sort((a, b) => (a.price > b.price) ? 1 : -1);

}

if (sortType === 'От большего к меньшему') {

oldList = list.sort((a, b) => (a.price < b.price) ? 1 : -1);

}

return oldList;

}

const handleFiltersBrands = (filters) => { //метод смены состояния компонента

var newFilters = [...filters];

setNewFiltersBrands(newFilters);

}

const handleFiltersTypes = (filters) => { //метод смены состояния компонента

var newFilters = [...filters];

setNewFiltersTypes(newFilters);

}

const handleSortPrice = (sortType) => { //метод смены состояния компонента

setSortBy(sortType);

}

const productsSearch = searchPanel(filterList(sortList(products, sortBy))); //листа товаров

const list = (!loading) ? ((productsSearch && productsSearch.length !== 0) ? (

(grid === true) ? (<Row className='d-flex justify-content-center'>{productsSearch.map(product => <ProductOfGrid key={product.id} product={product} role={role} />)}</Row>)

: (<ListGroup>{productsSearch.map(product => <ProductOfList key={product.id} product={product} role={role} />)}</ListGroup>))

: (<Alert className='mt-2 d-flex justify-content-center' variant='secondary'>Список пуст</Alert>))

: (

<div className='mt-2 d-flex justify-content-center'>

<Spinner />

</div>

) //представление листа товаров ввиде grid или list

//код представления компонента

return (

<div>

{(baners.length !== 0) ? <div className="mt-2"><Carousel images={baners} /></div> : null}

<ButtonToolbar className='float-right'>

{(role === 'admin') ? (

<Button variant="light"

onClick={() => setAddModalShow(true)}>

{<AddIcon />}Добавить новый товар

</Button>

) : (null)}

</ButtonToolbar>

<div style={{ display: 'inline-flex' }}>

{/\*Filter by brands\*/}

<CheckBox items={brands} sortBy='Бренды' handleFilters={filters => handleFiltersBrands(filters)} />

<div className="mr-2"></div>

{/\*Filter by category(types)\*/}

<CheckBox items={types} sortBy='Категории' handleFilters={filters => handleFiltersTypes(filters)} />

<div className="mr-2"></div>

{/\*Sorting by price\*/}

<RadioBox list={items} handleSort={sort => handleSortPrice(sort)} />

</div>

<AddProductModal

show={addModalShow}

onHide={addModalClose}

types={types}

brands={brands}

suppliers={suppliers}>

</AddProductModal>

<div className="mt-2"></div>

{/\*Search Panel\*/}

<Form>

<FormGroup>

<FormControl type="text" value={search} placeholder="Search" className="mr-sm-2"

onChange={(e) => setSearch(e.target.value)} />

</FormGroup>

</Form>

<ToggleButtons items={[<ViewComfyIcon />, <FormatListBulletedIcon />]} variant='light' value={grid} onChange={(grid) => setGrid(grid)} />

{list}

<ScrollTop />

</div>

)

}

export default Products;

Приложение Б

(справочное)

Формы выходных документов

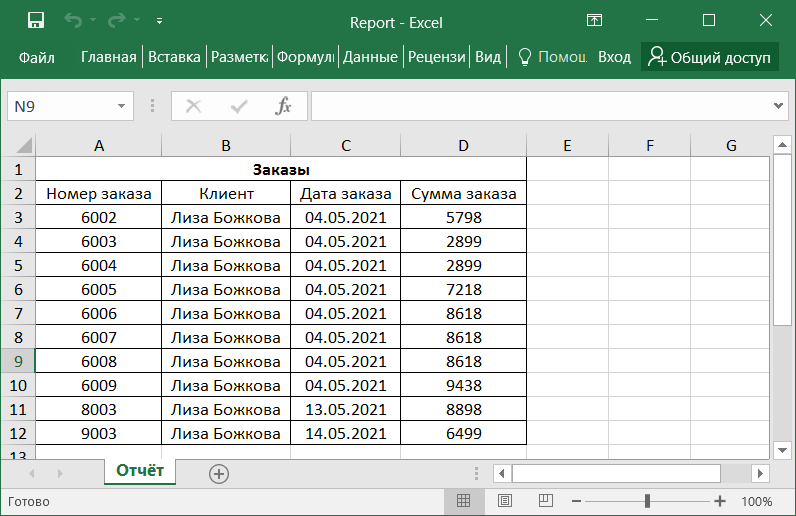


Рисунок Б.1 − Отчёт по заказам в разрезе дат

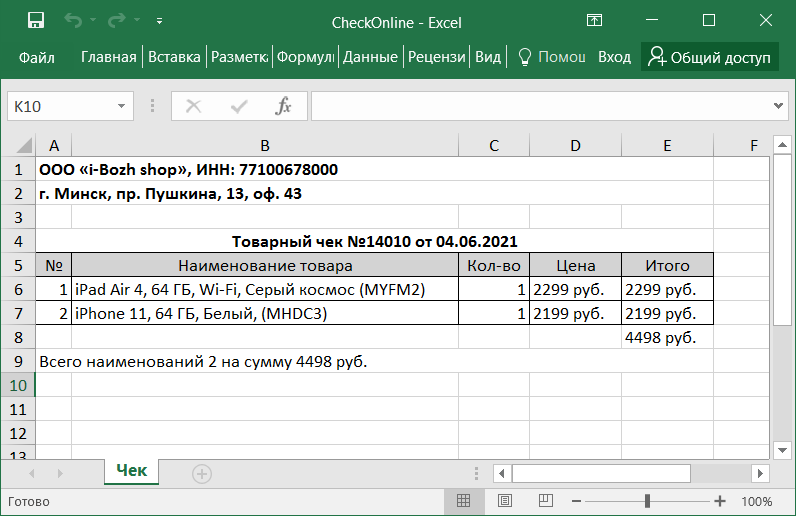


Рисунок Б.2 − Товарный чек оформленного заказа