Частное учреждение образования

«Колледж бизнеса и права»

ДОПУЩЕН К ЗАЩИТЕ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_ И. В. Малафей

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023

СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ ЗНАНИЙ УЧАСТНИКОВ ОЛИМПИАДЫ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ДП Т.992018.401

Председатель цикловой комиссии (К. О. Якимович )

Руководитель проекта (К. О. Якимович )

Консультант по экономической части (Е. А. Андреева )

Консультант по охране труда (В. С. Кудласевич )

Учащийся (А. С. Парфянович )

Рецензент ( )

2023

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ДП Т.992018.401 ПЗ

Разраб.

Парфянович А.С.

Провер.

Якимович К.О.

Т.Контр.

*Якимович К.О.*

Н. контр.

Якимович К.О.

Утверд.

Багласова Т.Г.

*Создание программного средства для проведения тестирования знаний участников олимпиады*

Лит.

Листов

КБП

??

у

[Введение 4](#_Toc132559210)

[1 Описание задачи 6](#_Toc132559211)

[1.1 Анализ предметной области 6](#_Toc132559212)

[1.2 Постановка задачи 6](#_Toc132559213)

[2 Вычислительная система 8](#_Toc132559214)

[2.1 Требования к аппаратным и операционным ресурсам 8](#_Toc132559215)

[2.2 Инструменты разработки 8](#_Toc132559216)

[3 Проектирование задачи 11](#_Toc132559217)

[3.1 Требования к приложению 11](#_Toc132559218)

[3.2 Концептуальный прототип 11](#_Toc132559219)

[3.3 Организация данных 22](#_Toc132559220)

[3.4 Функции: логическая и физическая организация 28](#_Toc132559221)

[3.5 Проектирование справочной системы приложения 36](#_Toc132559222)

[4 Описание программного средства 38](#_Toc132559223)

[4.1 Общие сведения 38](#_Toc132559224)

[4.2 Функциональное назначение 38](#_Toc132559225)

[4.3 Входные и выходные данные 38](#_Toc132559226)

[5 Методика испытаний 40](#_Toc132559227)

[5.1 Технические требования 40](#_Toc132559228)

[5.2 Функциональное тестирование 40](#_Toc132559229)

[6 Применение 74](#_Toc132559230)

[6.1 Назначение программы 74](#_Toc132559231)

[6.2 Условие применения 74](#_Toc132559232)

[6.3 Справочная система 74](#_Toc132559233)

[7 Охрана труда и окружающей среды 79](#_Toc132559234)

[7.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда 79](#_Toc132559235)

[7.2 Разработка мер по нормализации инфра- и ультразвука в помещении 83](#_Toc132559236)

[7.3 Обеспечение пожарной безопасности 86](#_Toc132559237)

[7.4 Охрана окружающей среды 87](#_Toc132559238)

[8 Экономический раздел 90](#_Toc132559239)

[8.1 Технико-экономическое обоснование разработки программного средства 90](#_Toc132559240)

[8.2 Составление плана по разработке программного средства 90](#_Toc132559241)

[8.3 Определение цены программного средства 90](#_Toc132559242)

[8.4 Экономическая эффективность разработки 94](#_Toc132559243)

[Заключение 96](#_Toc132559244)

[Список информационных источников 97](#_Toc132559245)

[Приложение А 98](#_Toc132559246)

[Приложение Б 3](#_Toc132559247)

# Введение

Олимпиады – это академические состязания, проверяющие знания и навыки участников по различным предметам, таким как математика, физика, информатика и лингвистика. Процесс участия в олимпиаде обычно включает в себя регистрацию, подготовку и прохождение теста. Задача состоит в создании программного средства, которое позволит регистрировать участников олимпиады и создавать различные тестовые задания. Программное средство будет иметь два режима: режим организатора и режим участника. В режиме организатора организаторы олимпиады смогут создавать и редактировать тестовые задания, устанавливать критерии оценки и лимиты тестирования. В режиме участника участники олимпиады смогут войти в систему, просмотреть доступные тестовые задания и отправить свои ответы.

Создание данного программного инструмента очень актуально, так как значительно упростит процесс организации и участия в олимпиадных соревнованиях. В настоящее время большинство олимпиад проводится вручную, что отнимает много времени и чревато ошибками. Программный инструмент позволит автоматизировать процесс, сделав его более эффективным и точным. Это приведет к созданию более справедливой и конкурентной среды для всех участников.

Для достижения цели дипломного проектирования нужно решить следующие задачи:

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

Решение поставленных задач отражено в пояснительной записке, состоящей из шести разделов, и содержит информацию по организации и использованию данного программного средства [1].

В первом разделе «Описание задачи» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, проектирование модели, отображающей функциональную структуру объекта.

Второй раздел «Проектирование системы» включает требования к программному средству и представление модели пользовательского итерфейса.

Третий раздел «Описание реализации программного средства» отражает общие сведения о программе, предоставляет порядок авторизации пользователей, описывает инфраструктуру базы данных, а также логическую и физическую организацию данных реализованных функций. Предусматривает проведение испытаний разработанного программного средства, с описанием входным и выходных данных, справочной системы.

Четвёртый раздел «Применение» содержит информацию, необходимую в процессе эксплуатации программного средства: его назначение и условия применения.

Пятый раздел «Охрана труда и окружающей среды» содержит информацию о правовых, нормативных, социально-экономических и организационных вопросах по охране труда, о пожарной безопасности, об охране окружающей среды, а также о разработке мер по нормализации инфра- и ультразвука в помещении.

Шестой экономический раздел содержит сведения о технико-экономическом обосновании разработки программного средства, составление плана по разработке программного средства, определение цены программного средства, а также приводится экономическая эффективность разработки.

В заключении подводится общий итог о проделанной работе, анализируется степень реализации функций и извлекаются соответствующие выводы о качестве разработанного программного средства.

В приложении А представлен текст программы.

В приложении Б представлены формы выходных документов.

Графическая часть представлена диаграммами вариантов использования, классов, последовательности, деятельности и компонентов.

# Описание задачи

## Анализ предметной области

Предметная область данного проекта – организация олимпиадных соревнований. Олимпиады – это академические соревнования, которые проверяют знания участников в различных областях, включая математику, естественные науки и языковые искусства. Предметная область включает в себя ряд бизнес-процессов, в том числе регистрацию участников, создание тестовых заданий и подсчет баллов.

Регистрация участников – это процесс сбора информации об участниках, включая их имя, образовательное учреждение, курс, возраст, адрес электронной почты и номер контактного телефона. Эта информация используется для проверки личности участников и для передачи важной информации об олимпиадном соревновании.

Создание тестовых заданий – это процесс проектирования и разработки тестовых заданий, которые будут использоваться для оценки знаний участников. Тестовые задания могут иметь несколько различных форм, включая вопросы с множественным выбором, кратким ответом и эссе. Также устанавливаются критерии оценки, чтобы гарантировать, что участники оцениваются справедливо и точно.

Подсчет баллов – это процесс оценки работы участников и присвоения баллов на основе их ответов. Процесс подсчета баллов может варьироваться в зависимости от типа тестового задания и критериев оценки. В некоторых случаях подсчет баллов может быть автоматизирован, а в других случаях может потребоваться вмешательство человека.

Алгоритмы получения промежуточных и итоговых показателей зависят от типа тестового задания и критериев оценки. Для вопросов с несколькими вариантами ответов промежуточные и итоговые показатели могут быть получены путем подсчета количества правильных ответов. Для вопросов с краткими ответами промежуточные и итоговые показатели могут быть получены с помощью алгоритмов обработки естественного языка для оценки точности и полноты ответов. Для вопросов с эссе промежуточные и итоговые показатели могут быть получены с помощью алгоритмов машинного обучения для анализа содержания и качества ответов.

Предметная область может быть подробно описана путем определения задействованных сущностей и их взаимоотношений. Сущности, вовлеченные в предметную область, включают участников, образовательные учреждения, тестовые задания, критерии оценки, подсчет баллов и результаты. Участники связаны с образовательными учреждениями, а образовательные учреждения связаны с тестовыми заданиями. Тестовые задания связаны с критериями оценки, а критерии оценки связаны с подсчетом баллов. Подсчет баллов связан с результатами, которые используются для определения победителей олимпиады. Понимая взаимосвязи между этими сущностями, можно разработать программное средство, способное автоматизировать процесс организации и участия в олимпиадах.

## Постановка задачи

Бизнес-проблема, которую можно решить с помощью программного средства для тестирования знаний участников олимпиад, заключается в ручном и трудоемком процессе организации и участия в олимпиадах. В настоящее время этот процесс включает в себя ручную регистрацию участников, ручное создание и распределение тестовых заданий, ручное подсчет результатов тестирования и ручное определение победителей. Этот процесс подвержен ошибкам и неэффективен с точки зрения затрат времени и ресурсов.

Существующие решения для организации и участия в олимпиадных соревнованиях включают ручные процессы и некоторые коммерческие программные решения. Ручные процессы чреваты ошибками и отнимают много времени, а коммерческие программные решения могут быть дорогими и не могут быть адаптированы к конкретным потребностям каждого соревнования. Кроме того, многие из этих коммерческих решений могут потребовать значительных технических знаний для настройки и эффективного использования.

Программное средство для проверки знаний участников олимпиады может решить эти проблемы, предоставляя эффективное, настраиваемое и удобное решение для организации и участия в олимпиадах. Программное средство автоматизирует процесс регистрации участников, создание и распределение тестовых заданий, а также подсчет результатов тестирования. Оно также предоставляет критерии оценки, которые могут быть настроены в соответствии с конкретными потребностями каждого конкурса. Программный инструмент позволит организаторам управлять и контролировать соревнования в режиме реального времени и предоставит участникам удобный интерфейс для прохождения тестовых заданий. Кроме того, программное обеспечение может быть настроено в соответствии с конкретными потребностями каждого соревнования, что делает его экономически эффективным решением для небольших и масштабных олимпиад.

# Проектирование системы

## Требования к приложению

Чтобы обеспечить успех программного инструмента для тестирования знаний участников олимпиады, необходимо выполнить определенные требования. К ним относятся функции безопасности, ограничения и требования к интерфейсу.

Безопасность – важнейший аспект любого программного приложения, особенно того, которое работает с конфиденциальными данными, такими как личная информация и результаты тестирования. Поэтому программное средство должно быть разработано с надежными функциями безопасности, которые предотвращают несанкционированный доступ, обеспечивают конфиденциальность данных и защищают от киберугроз.

Ограничения также являются важным фактором для программного инструмента. Например, инструмент должен иметь ограничения по количеству попыток, которые участник может предпринять для данного теста, продолжительности теста и типам вопросов, которые могут быть заданы. Эти ограничения помогут обеспечить справедливый и последовательный процесс тестирования для всех участников.

Что касается требований к интерфейсу, то программное обеспечение должно быть удобным и интуитивно понятным, с четкими инструкциями и простой навигацией. Интерфейс также должен быть разработан таким образом, чтобы быть отзывчивым и совместимым с различными устройствами и операционными системами.

Следует также обратить внимание на требования к аппаратным и операционным ресурсам программного средства. Инструмент должен быть разработан таким образом, чтобы эффективно работать с имеющимися аппаратными и программными ресурсами, а любые системные требования должны быть четко указаны для обеспечения совместимости с устройствами пользователей.

Минимальные системные требования для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* 32 – разрядный (x86) или 64 – разрядный (x64) процессор с тактовой частотой 1 ГГц или выше;
* 1 ГБ (для 32 – разрядного процессора) или 2 ГБ (для 64 – разрядного процессора) ОЗУ;
* 16 ГБ (для 32 – разрядной системы) или 20 ГБ (для 64 – разрядной системы) свободного места на жестком диске;
* графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.

## Проектирование модели

## Проектирование структуры базы данных

## Концептуальный прототип

# Проектирование задачи

## Требования к приложению

Разрабатываемое программное средство должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействия между программой и пользователем было максимально упрощено. Для обучения пользователей необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения. Также, в программном средстве будут реализованы функции, связанные с ведением базы данных, авторизацией, поиском и фильтрацией списка изделий, оформлением заказов, просмотром статистики и формированием документов по оформленные заказам и статистике.

Кроме того, при разработке окон необходимо соблюдать определенные требования: окна не должны быть перегружены информацией, должны иметь стандартные элементы управления данными, должны быть понятны пользователю. При конструировании окон в необходимых случаях нужно предусмотреть возможность защиты данных, установить ограничения на корректный ввод данных. Примером защиты будет служить окно авторизации, где необходимо будет ввести логин и пароль.

## Концептуальный прототип

Концептуальный прототип состоит из описания внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

Основной интерфейс программного средства будут представлять окна (WPF). Все окна будут содержать стандартные пользовательские элементы управления.

В программе предполагается создание навигационных кнопок.

В приложении было разработано главное меню, обеспечивающее быстрый и удобный доступ к функциям приложения и структурирующее их в однородные группы.

После запуска приложения будет появляться окно «AutorizationWindow», представленное на рисунке 3.1.

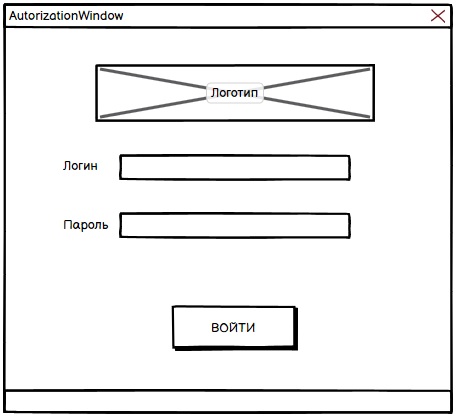


Рисунок 3.1 – Прототип окна авторизации

Кнопка «Войти» будет открывать главное меню, но только после успешного прохождения авторизации.

Прототип окна, служащего главным меню, «MainWindow» представлен на рисунке 3.2.

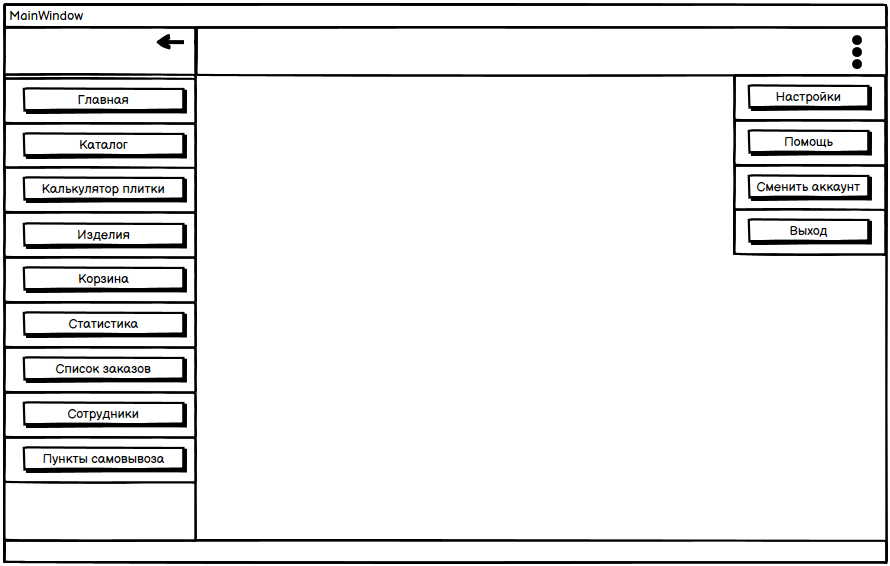


Рисунок 3.2 – Прототип окна главного меню

Кнопка «Главная» будет открывать окно, содержащее в себе информацию о розничной сети «Керамин».

Кнопка «Каталог» будет открывать окно, где представлен перечень изделий, доступных для покупки.

Кнопка «Калькулятор плитки» будет открывать окно, где по параметрам изделия и комнаты или пола можно будет рассчитать количество изделий для них.

Кнопка «Изделия» будет отображать меню, в котором можно будет выбрать между добавлением нового изделия или просмотром списка уже существующих изделий, с последующей возможностью изменения их информации или удаления.

Кнопка «Корзина» будет открывать окно, в котором можно будет ознакомиться с изделиями, готовыми для покупки, а также, при необходимости, удалять их, непосредственно, из корзины.

Кнопка «Статистика» будет отображать меню, в котором можно будет выбрать между просмотром статистики о продуктивности сотрудников и статистики по популярности изделий.

Кнопка «Список заказов» будет открывать окно, в котором будет доступна возможность просмотра информации о завершенных заказах, а также возможность создания документа по каждому заказу.

Кнопка «Сотрудники» будет отображать меню, в котором администратору можно будет выбрать между добавлением нового сотрудника или просмотром списка учетных записей сотрудников, с последующей возможностью изменения их информации или удаления.

Кнопка «Пункты самовывоза» будет отображать меню, в котором можно будет выбрать между добавлением нового пункта самовывоза или просмотром списка уже существующих, с последующей возможностью изменения их информации или удаления.

Кнопка «Настройки» будет открывать окно, в котором сотруднику можно будет изменить информацию о своей учетной записи.

Кнопка «Помощь» будет открывать справку.

Кнопка «Сменить аккаунт» будет открывать окно авторизации, то есть осуществлять выход из учетной записи.

Кнопка «Выход» будет закрывать программу.

Прототип окна, служащего начальным окном, «MainPage» представлен на рисунке 3.3.

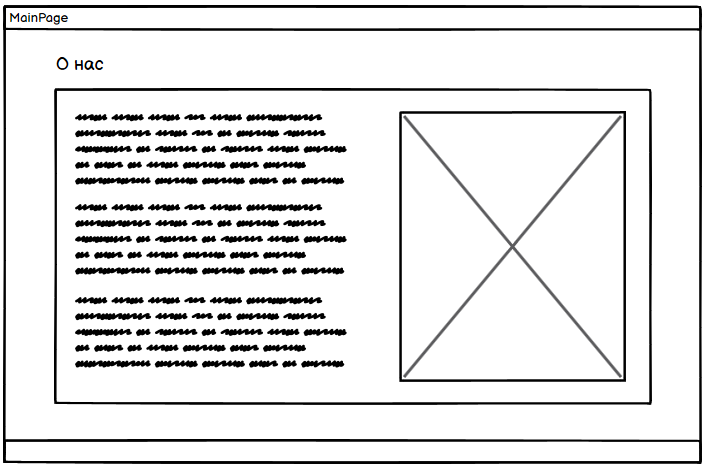


Рисунок 3.3 – Прототип начального окна

Прототип окна, служащего каталогом изделий, «CatalogWindow» представлен на рисунке 3.4.



Рисунок 3.4 – Прототип окна с каталогом изделий

Кнопка «Применить» будет применять условия фильтра и отображать в таблице все соответствия.

Кнопка «Очистить» будет очищать все критерии фильтра.

Кнопка «Добавить» будет добавлять выбранное изделие в корзину.

Прототип окна, где будет рассчитываться необходимое количество изделий, «TileCalculatorWindow» представлен на рисунке 3.5.

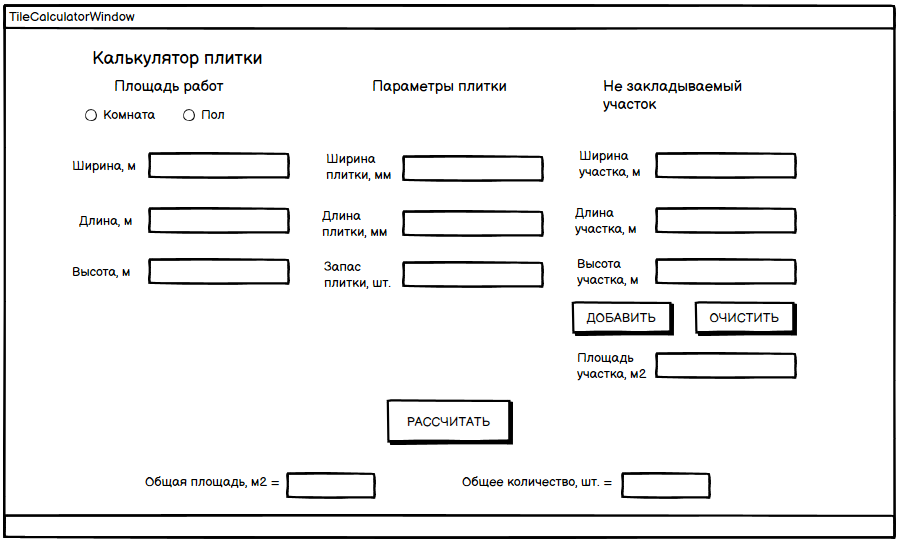


Рисунок 3.5 – Прототип окна, где рассчитывается необходимое количество изделий

Кнопка «Добавить» будет добавлять площадь не закладываемого участка в соответствующее поле.

Кнопка «Очистить» будет очищать поле с площадью не закладываемого участка.

Кнопка «Рассчитать» будет рассчитывать общую закладываемую площадь, а также и необходимое количество изделий для ее закладывания.

Прототип окна добавления нового изделия «AddNewProductWindow» представлен на рисунке 3.6.

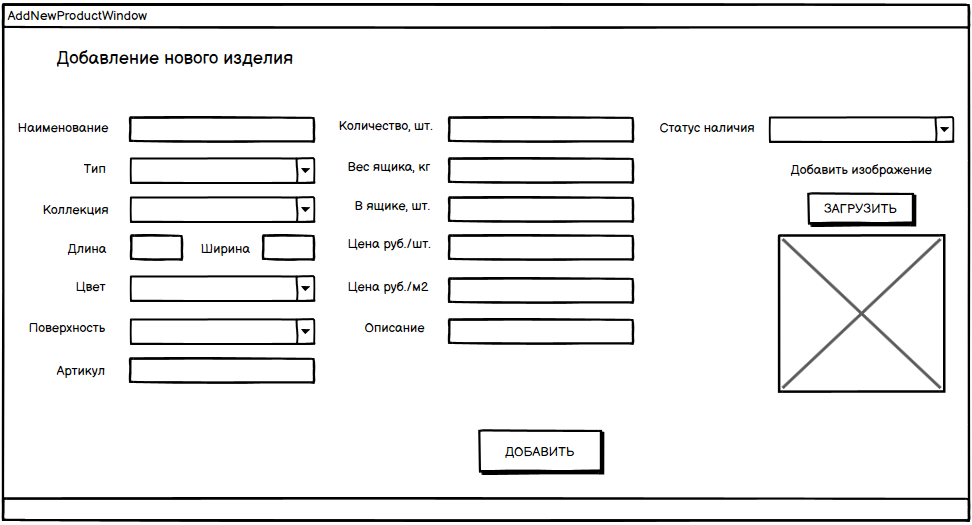


Рисунок 3.6 – Прототип окна добавления нового изделия

Кнопка «Загрузить» будет открывать диалоговое окно, в котором сотруднику необходимо будет выбрать изображение для изделия.

Кнопка «Добавить» будет добавлять изделие в базу, но только при заполнении всех обязательных полей валидными данными.

Прототип окна со списком изделий, предназначенного для работы с информацией об изделиях, «ProductsListWindow» представлен на рисунке 3.7.

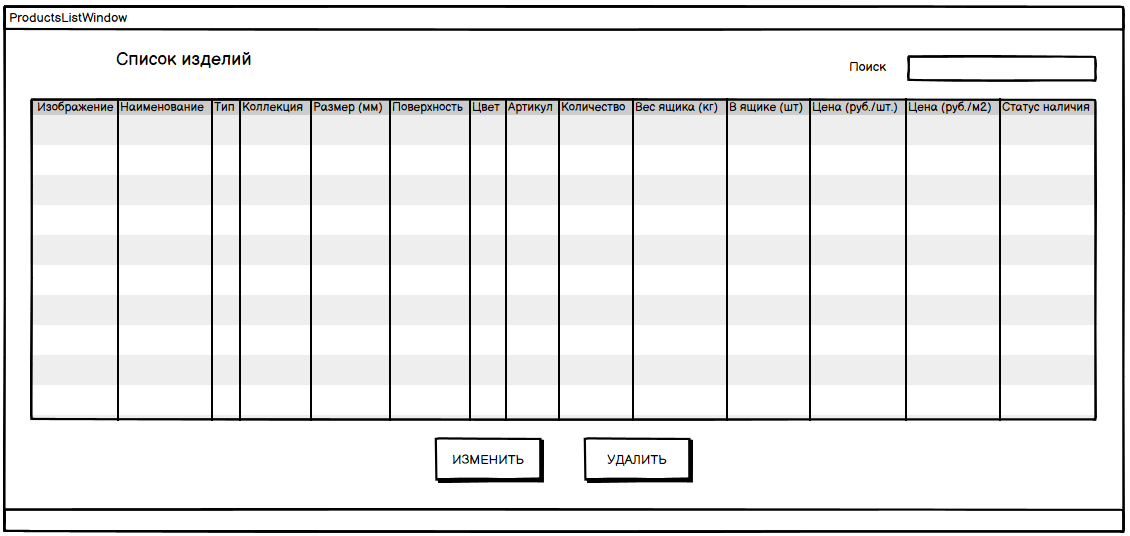


Рисунок 3.7 – Прототип окна со списком изделий

Кнопка «Изменить» будет открывать окно, в котором будет доступна возможность изменения информации о выбранном изделии.

Кнопка «Удалить» будет удалять выбранное изделие.

Прототип окна, предназначенного для изменения данных о выбранном изделии, «ChangeProductInfoWindow» представлен на рисунке 3.8.

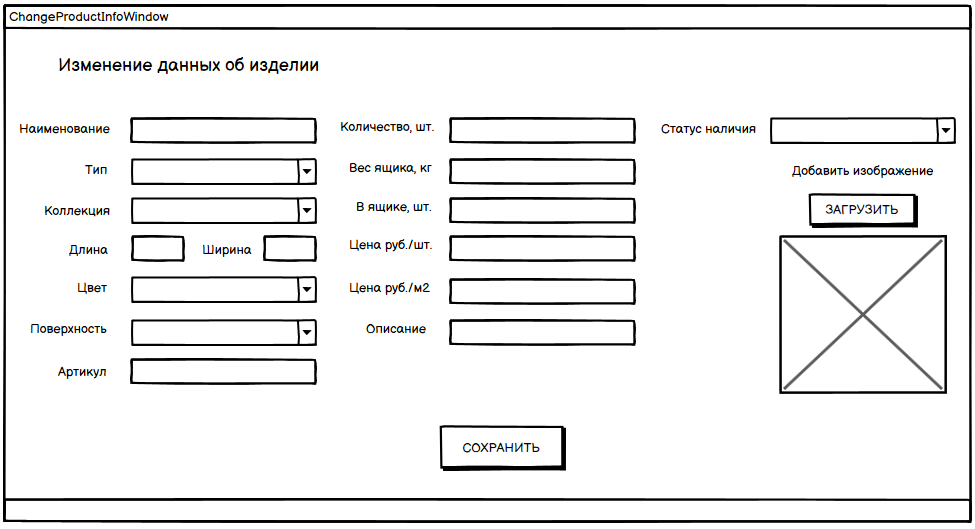


Рисунок 3.8 – Прототип окна изменения данных о выбранном изделии

Кнопка «Загрузить» будет открывать диалоговое окно, в котором сотруднику необходимо будет выбрать изображение для изделия.

Кнопка «Сохранить» будет применять внесенные изменения к выбранному изделию.

Прототип окна, предназначенного для оформления заказа, «CreateOrderWindow» представлен на рисунке 3.9.

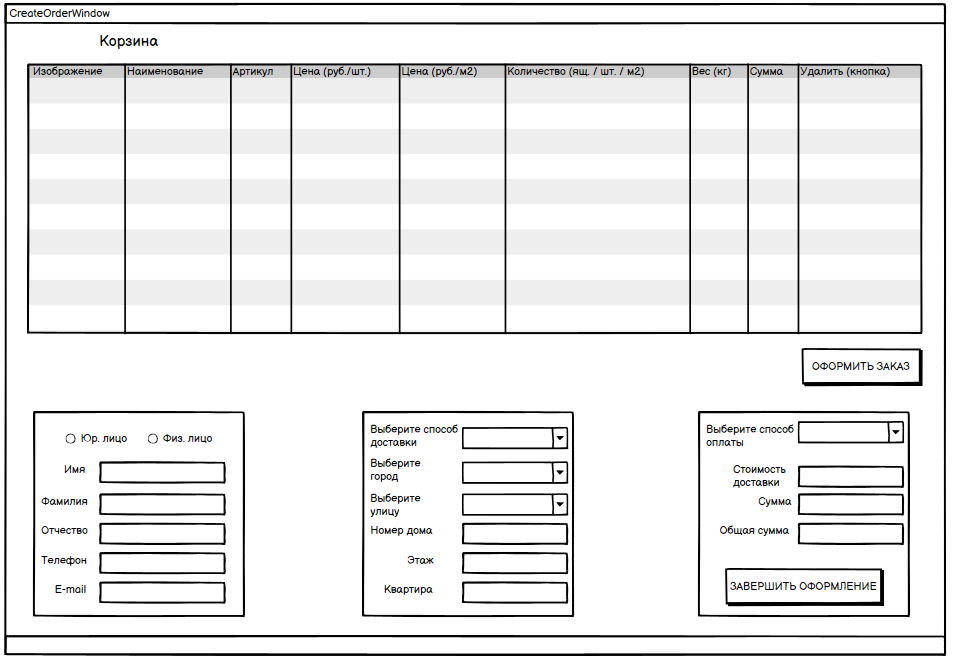


Рисунок 3.9 – Прототип окна, предназначенного для оформления заказа

Кнопка «Оформить заказ» будет открывать ту часть окна, которая находится под данной кнопкой, но только при условии, что корзина не пустая.

Кнопка «Завершить оформление» будет создавать новый заказ, но при условии, что все соответствующие поля заполнены.

Прототип окна, предназначенного для просмотра статистики о продуктивности сотрудников, «EmployeesStatistics» представлен на рисунке 3.10.

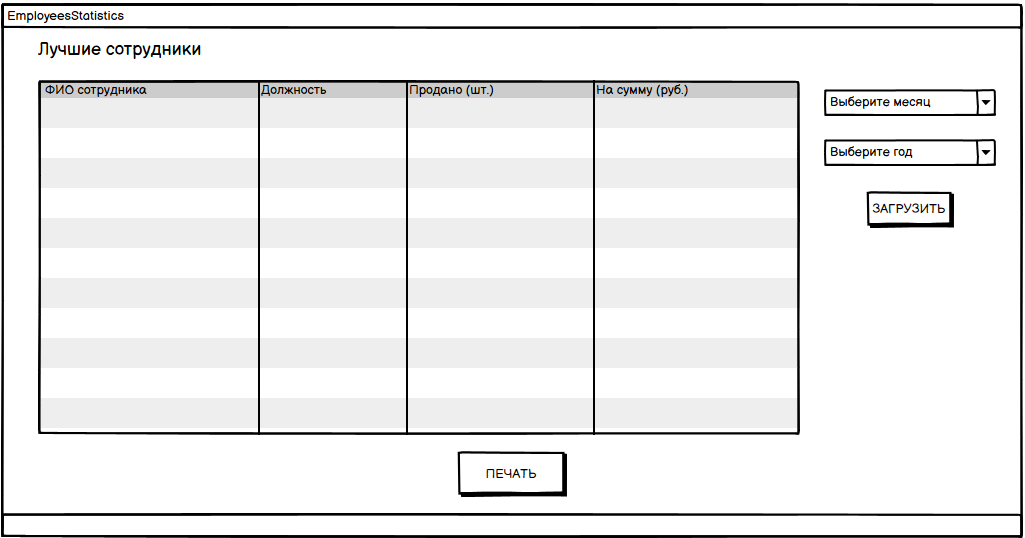


Рисунок 3.10 – Прототип окна со статистикой о продуктивности сотрудников

Кнопка «Загрузить» будет открывать отображать всю информацию о продуктивности сотрудников за период, указанный пользователем.

Кнопка «Печать» будет создавать документ со всей информацией о сотрудниках, представленной в таблице.

Прототип окна, предназначенного для просмотра статистики о популярности изделий, «ProductsStatistics» представлен на рисунке 3.11.

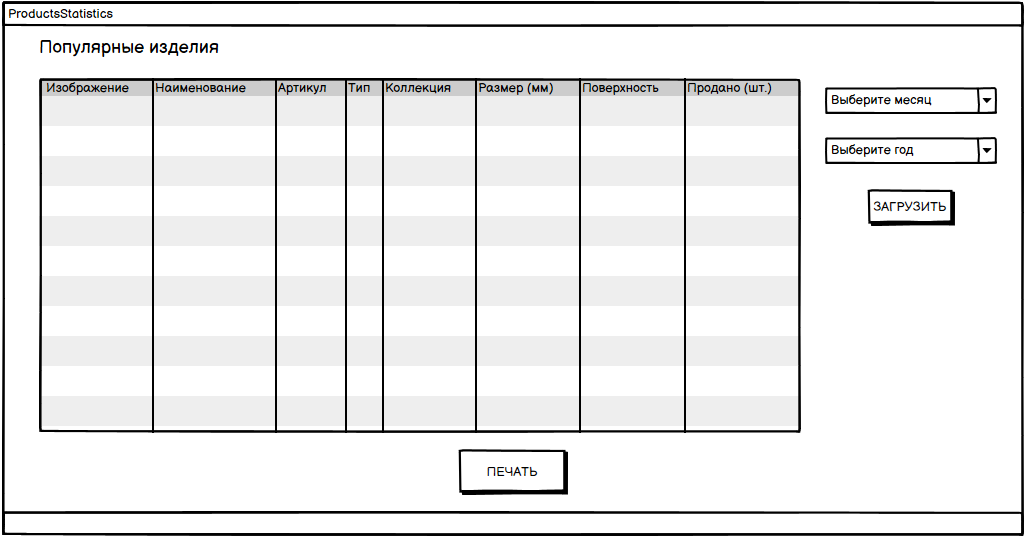


Рисунок 3.11 – Прототип окна со статистикой о популярности изделий

Кнопка «Загрузить» будет открывать отображать всю информацию о популярности изделий за период, указанный пользователем.

Кнопка «Печать» будет создавать документ со всей информацией об изделиях, представленной в таблице.

Прототип окна, предназначенного для просмотра списка завершенных заказов, «OrdersListWindow» представлен на рисунке 3.12.

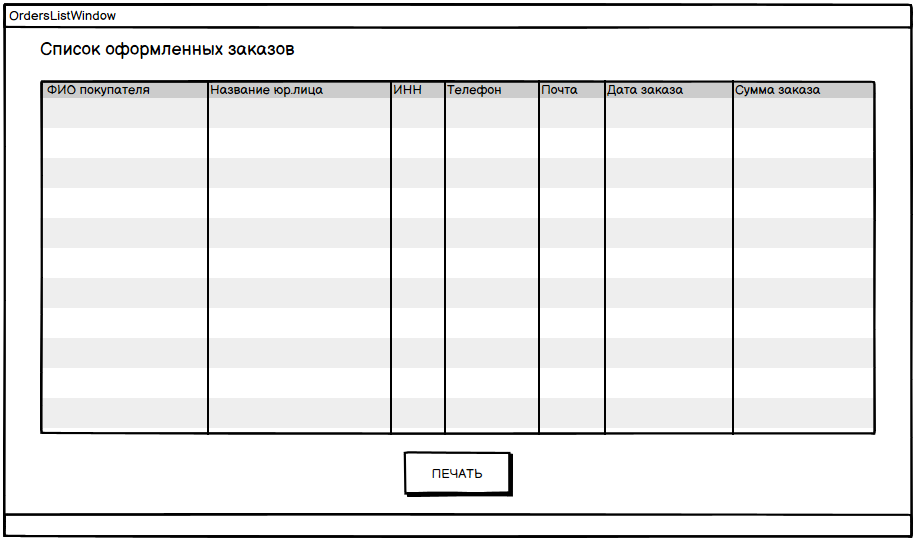


Рисунок 3.12 – Прототип окна со списком завершенных заказов

Кнопка «Печать» будет создавать документ со всей информацией о выбранном заказе.

Прототип окна добавления сотрудника «RegistrationWindow» представлен на рисунке 3.13.

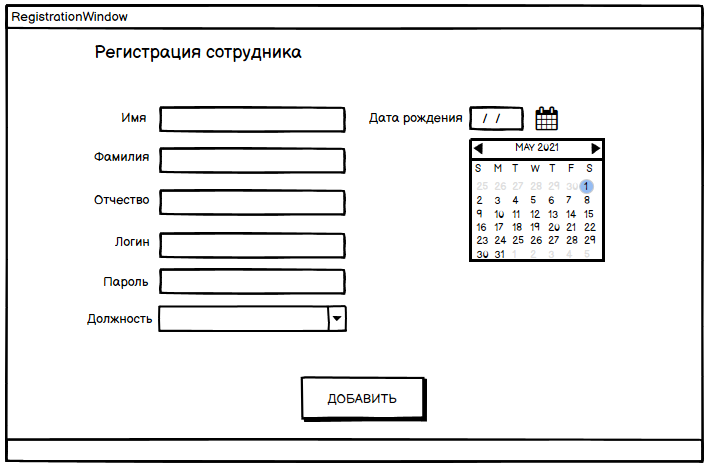


Рисунок 3.13 – Прототип окна добавления нового сотрудника

Кнопка «Добавить» будет добавлять сотрудника в базу, но только при заполнении всех обязательных полей валидными данными.

Прототип окна со списком сотрудников, предназначенного для работы с информацией о сотрудниках, «StaffWindow» представлен на рисунке 3.14.

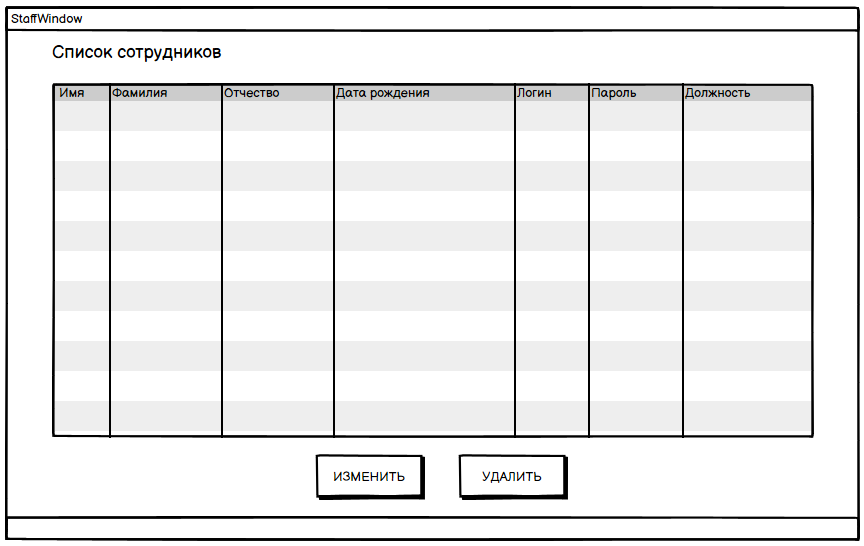


Рисунок 3.14 – Прототип окна со списком сотрудников

Кнопка «Изменить» будет открывать окно, в котором будет доступна возможность изменения информации о выбранном сотруднике.

Кнопка «Удалить» будет удалять выбранного сотрудника.

Прототип окна изменения данных о выбранном сотруднике «ChangeEmployeeInfoWindow» представлен на рисунке 3.15.



Рисунок 3.15 – Прототип окна изменения данных о выбранном сотруднике

Кнопка «Сохранить» будет применять внесенные изменения к выбранному сотруднику.

Прототип окна добавления нового пункта самовывоза «AddPickupAdressWindow» представлен на рисунке 3.16.

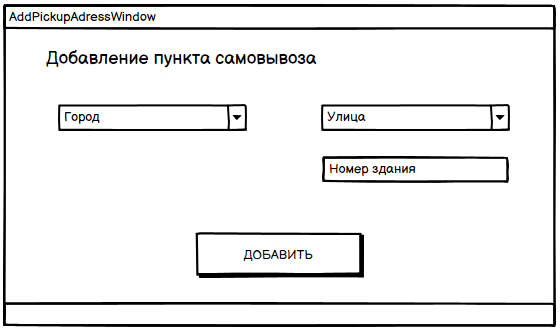


Рисунок 3.16 – Прототип окна добавления нового пункта самовывоза

Кнопка «Добавить» будет добавлять пункт самовывоза в базу, но только при указании всех данных.

Прототип окна со списком пунктов самовывоза, предназначенного для работы с информацией о них, «PickupsListWindow» представлен на рисунке 3.17.

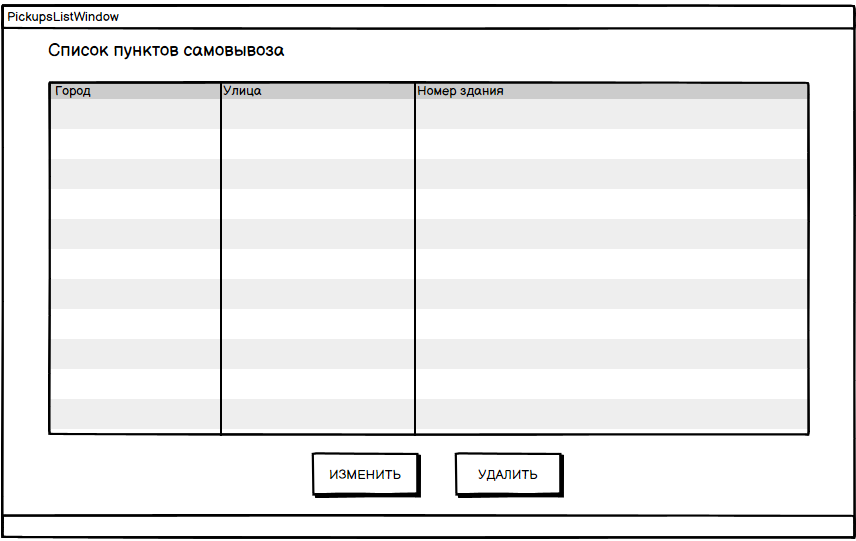


Рисунок 3.17 – Прототип окна со списком пунктов самовывоза

Кнопка «Изменить» будет открывать окно, в котором будет доступна возможность изменения информации о выбранном пункте самовывоза.

Кнопка «Удалить» будет удалять выбранный пункт самовывоза.

Прототип окна изменения данных о выбранном пункте самовывоза «ChangePickupAdressWindow» представлен на рисунке 3.18.

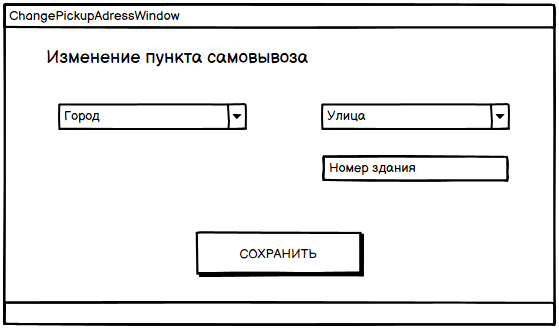


Рисунок 3.18 – Прототип окна изменения данных о выбранном пункте самовывоза

Кнопка «Сохранить» будет применять внесенные изменения к выбранному пункту самовывоза.

Прототип окна настроек учетной записи «AccountSettingsWindow» представлен на рисунке 3.19.



Рисунок 3.19 – Прототип окна настроек учетной записи

Кнопка «Сохранить» будет сохранять внесенные изменения. В случае, если будут указаны неверные данные, отобразится соответствующее сообщение.

## Организация данных

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 атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенной в ходе проектирования сущности соответствует таблица.

Структура базы данных разрабатываемого программного средства была создана с использованием СУБД SQL Server 2017 и включает 17 таблиц.

Таблица «AvailabilityStatus» хранит информацию о статусах наличия изделий. Структура приведена в таблице 3.1.

Таблица 3.1 – Структура таблицы «AvailabilityStatus»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| availabilityStatusCode | int | 4 | Идентификатор статуса наличия |
| availabilityStatusName | varchar | 14 | Название статуса наличия |

Таблица «Basket» хранит информацию об изделиях, которые были добавлены в корзину для оформления заказа. Структура приведена в таблице 3.2.

Таблица 3.2 – Структура таблицы «Basket»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| basketCode | int | 4 | Идентификатор записи в корзине |
| basketNumber | int | 4 | Номер заказа |
| productsCount | int | 4 | Количество изделий |
| boxesCount | int | 4 | Количество ящиков |
| productsArea | float | 8 | Площадь изделий |
| productsWeight | float | 8 | Вес изделий |
| generalSum | float | 8 | Сумма позиции в корзине |
| paymentStatus | bit | 1 | Статус оплаты |
| productCode | int | 4 | Код изделия |

Таблица «Color» хранит информацию о цветах изделий. Структура приведена в таблице 3.3.

Таблица 3.3 – Структура таблицы «Color»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| colorCode | int | 4 | Идентификатор цвета изделия |
| colorName | varchar | 100 | Название цвета |

Таблица «Customer» хранит информацию о покупателях, которыми могут быть как физические лица, так и юридические лица. Структура приведена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Customer»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| customerCode | int | 4 | Идентификатор покупателя |
| customerName | varchar | 50 | Имя покупателя |
| customerSurname | varchar | 50 | Фамилия покупателя |
| customerPatronymic | varchar | 50 | Отчество покупателя |
| phone | varchar | 17 | Контактный телефон покупателя |
| mail | varchar | 40 | Почта покупателя |
| building | varchar | 8 | Здание для доставки |
| floor\_ | int | 4 | Этаж для доставки |
| apartment | int | 4 | Квартира или офис для доставки |
| legalName | varchar | 100 | Наименование юридического лица |
| UTN | int | 4 | Уникальный номер плательщика (для юридического лица) |
| townCode | int | 4 | Код города для доставки |
| streetCode | int | 4 | Код улицы для доставки |
| orderNumber | int | 4 | Номер заказа |

Таблица «CustomerOrder» хранит информацию о заказах, которые были успешно оформлены. Структура приведена в таблице 3.5.

Таблица 3.5 – Структура таблицы «CustomerOrder»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| customerOrderCode | int | 4 | Идентификатор заказа |
| orderNumber | int | 4 | Номер заказа |
| issueDate | date | 3 | Дата заказа |
| generalSum | float | 8 | Общая сумма заказа |

Продолжение таблицы 3.5

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| deliveryCost | float | 8 | Стоимость доставки |
| deliveryCode | int | 4 | Код типа доставки |
| basketCode | int | 4 | Код записи в корзине |
| customerCode | int | 4 | Код покупателя |
| employeeCode | int | 4 | Код сотрудника |
| paymentCode | int | 4 | Код типа оплаты |
| pickupCode | int | 4 | Код пункта самовывоза |

Таблица «Delivery» хранит информацию о типах доставки изделий. Структура приведена в таблице 3.6.

Таблица 3.6 – Структура таблицы «Delivery»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| deliveryCode | int | 4 | Идентификатор типа доставки |
| deliveryName | varchar | 100 | Наименование типа доставки |

Таблица «Employee» хранит информацию о сотрудниках. Структура приведена в таблице 3.7.

Таблица 3.7 – Структура таблицы «Employee»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| employeeCode | int | 4 | Идентификатор сотрудника |
| employeeLogin | varchar | 30 | Логин сотрудника |
| employeeAdminStatus | bit | 1 | Статус администратора |
| employeePassword | varchar | 30 | Пароль сотрудника |
| employeeName | varchar | 50 | Имя сотрудника |
| employeeSurname | varchar | 50 | Фамилия сотрудника |
| employeePatronymic | varchar | 50 | Отчество сотрудника |
| employeeDateOfBirth | date | 3 | Дата рождения сотрудника |
| postCode | int | 4 | Код должности |

Таблица «Payment» хранит информацию о типах оплаты. Структура приведена в таблице 3.8.

Таблица 3.8 – Структура таблицы «Payment»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| paymentCode | int | 4 | Идентификатор типа оплаты |
| paymentType | varchar | 100 | Наименование типа оплаты |

Таблица «Pickup» хранит информацию о пунктах самовывоза. Структура приведена в таблице 3.9.

Таблица 3.9 – Структура таблицы «Pickup»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| pickupCode | int | 4 | Идентификатор пункта самовывоза |
| streetName | varchar | 100 | Наименование улицы для пункта самовывоза |
| building | varchar | 8 | Номер здания |
| pickupTownCode | int | 4 | Код города для пункта самовывоза |

Таблица «PickupTown» хранит информацию о городах для пунктов самовывоза. Структура приведена в таблице 3.10.

Таблица 3.10 – Структура таблицы «PickupTown»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| pickupTownCode | int | 4 | Идентификатор города для пункта самовывоза |
| pickupTownName | varchar | 100 | Наименование города для пункта самовывоза |

Таблица «Post» хранит информацию о должностях сотрудников. Структура приведена в таблице 3.11.

Таблица 3.11 – Структура таблицы «Post»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| postCode | int | 4 | Идентификатор должности |
| postName | varchar | 100 | Наименование должности |

Таблица «Product» хранит информацию обо всех изделиях. Структура приведена в таблице 3.12.

Таблица 3.12 – Структура таблицы «Product»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| productCode | int | 4 | Идентификатор изделия |
| productName | varchar | 100 | Наименование изделия |

Продолжение таблицы 3.12

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| productArticle | varchar | 11 | Артикул изделия |
| productWidth | float | 8 | Ширина изделия |
| productLenght | float | 8 | Длина изделия |
| productBoxWeight | float | 8 | Вес ящика с изделиями |
| productCountBox | int | 4 | Количество изделий в наличии |
| productCountInBox | int | 4 | Количество изделий в ящике |
| productCostCount | float | 8 | Стоимость за единицу изделия |
| productCostArea | float | 8 | Стоимость изделий за квадратный метр |
| productImage | varchar | 250 | Путь к изображению изделия |
| productDescription | varchar | 300 | Описание изделия |
| productCollectionCode | int | 4 | Код коллекции изделия |
| availabilityStatusCode | int | 4 | Код статуса наличия изделия |
| productTypeCode | int | 4 | Код типа изделия |
| surfaceCode | int | 4 | Код поверхности изделия |
| colorCode | int | 4 | Код цвета изделия |

Таблица «ProductCollection» хранит информацию о коллекциях изделий. Структура приведена в таблице 3.13.

Таблица 3.13 – Структура таблицы «ProductCollection»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| productCollectionCode | int | 4 | Идентификатор коллекции изделия |
| productCollectionName | varchar | 100 | Наименование коллекции изделия |

Таблица «ProductType» хранит информацию о типах изделий. Структура приведена в таблице 3.14.

Таблица 3.14 – Структура таблицы «ProductType»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| productTypeCode | int | 4 | Идентификатор типа изделия |
| productTypeName | varchar | 100 | Наименование типа изделия |

Таблица «Street» хранит информацию об улицах для доставки или самовывоза изделий. Структура приведена в таблице 3.15.

Таблица 3.15– Структура таблицы «Street»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| streetCode | int | 4 | Идентификатор улицы |
| streetName | varchar | 100 | Наименование улицы |

Таблица «Surface» хранит информацию о поверхностях изделий. Структура приведена в таблице 3.16.

Таблица 3.16 – Структура таблицы «Surface»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| surfaceCode | int | 4 | Идентификатор поверхности изделия |
| surfaceName | varchar | 100 | Наименование поверхности изделия |

Таблица «Town» хранит информацию о городах для доставки изделий. Структура приведена в таблице 3.17.

Таблица 3.17 – Структура таблицы «Town»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| townCode | int | 4 | Идентификатор города для доставки |
| townName | varchar | 100 | Наименование города для доставки |

Логическая структура базы данных представлена схемой данных на рисунке 3.20.

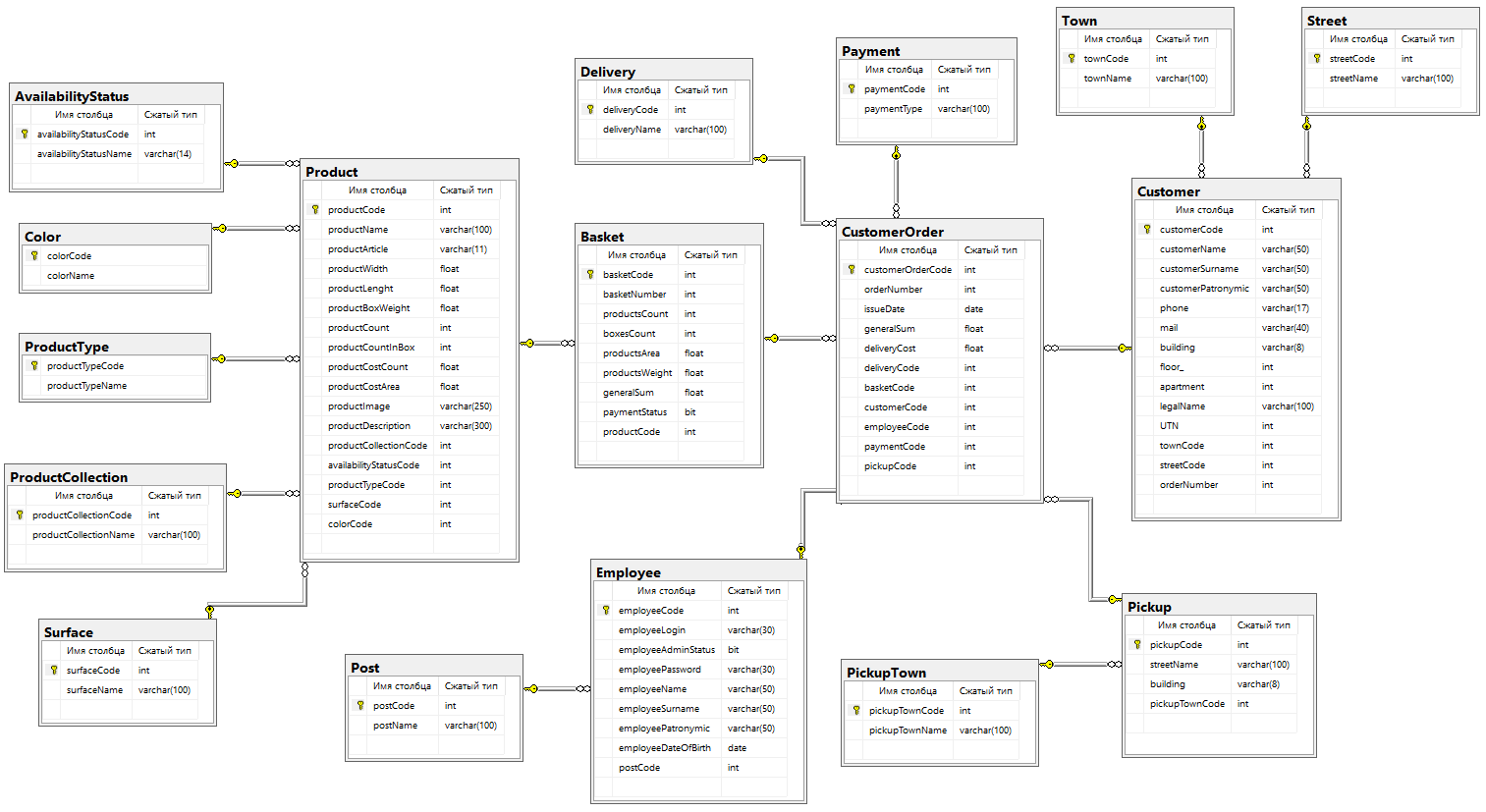


Рисунок 3.20– Структура базы данных

## Функции: логическая и физическая организация

Для реализации графического интерфейса программного средства была создана единая структура WPF-окон, которые содержат все необходимые функции программного средства.

За каждым пунктом навигационного меню закреплен модуль информационной части. При выборе требуемого пункта меню подключается закрепленный за этим пунктом модуль.

Обязательными функциями программного средства являются удаление, добавление и изменение данных.

Функции, которые выполняет программа – ведение данных в базе данных, поиск данных при помощи фильтра, формирование заказа.

Для успешного выполнения вышеперечисленных функций программное средство необходимо запустить на персональном компьютере (ПК). После запуска оно отобразит окно с авторизацией, где сотруднику потребуется ввести свои логин и пароль. В случае, если он не был зарегистрирован ранее, то это должен сделать администратор с целью создания учетной записи сотрудника. Затем, после успешной авторизации, перед ним откроется окно главного меню, где будет доступ ко всем функциям.

Данные о изделиях формируются и выводятся в таблицах для удобства пользования графическим интерфейсом программного средства.

Функция «Добавить новое изделие» для последующего добавления изделия обрабатывается событием AddButton\_Click, закрепленным за элементом управления AddButton «ДОБАВИТЬ». Данный компонент расположен в окне «AddNewProductWindow». Код функции представлен ниже:

private void AddButton\_Click(object sender, RoutedEventArgs e)

{

//Проверки вводимых данных на корректность (начало)

if (productNameField.Text != Product.CheckProductName(productNameField.Text, "Наименование изделия не может быть пустым.", "Наименование изделия содержит недопустимые символы.", "Длина наименования изделия может составлять 10-100 символов."))

{

MessageBox.Show(Product.CheckProductName(productNameField.Text, "Наименование изделия не может быть пустым.", "Наименование изделия содержит недопустимые символы.", "Длина наименования изделия может составлять 10-100 символов."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productTypeField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать тип изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productCollection.Text == string.Empty)

{

MessageBox.Show("Необходимо указать коллекцию, к которой относится изделие.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productLenght.Text != Product.CheckProductLenght(productLenght.Text, "Вы не указали длину изделия.", "Длина изделия может составлять от 98 до 900 мм.", "Вы указали неверную длину изделия."))

{

MessageBox.Show(Product.CheckProductLenght(productLenght.Text, "Вы не указали длину изделия.", "Длина изделия может составлять от 98 до 900 мм.", "Вы указали неверную длину изделия."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productWidth.Text != Product.CheckProductWidth(productWidth.Text, "Вы не указали ширину изделия.", "Ширина изделия может составлять от 9 до 400 мм.", "Вы указали неверную ширину изделия."))

{

MessageBox.Show(Product.CheckProductWidth(productWidth.Text, "Вы не указали ширину изделия.", "Ширина изделия может составлять от 9 до 400 мм.", "Вы указали неверную ширину изделия."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (colorField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать цвет изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productSurfaceField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать поверхность изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productArticleField.Text != Product.CheckProductArticle(productArticleField.Text, "Вы не указали артикул изделия.", "Артикул должен быть указан в формате YYY12345678."))

{

MessageBox.Show(Product.CheckProductArticle(productArticleField.Text, "Вы не указали артикул изделия.", "Артикул должен быть указан в формате YYY12345678."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (countField.Text != Product.CheckCountInBox(countField.Text, "Вы не указали количество изделий.", "Количество изделий не может быть отрицательным.", "Вы указали неверное количество изделий."))

{

MessageBox.Show(Product.CheckCountInBox(countField.Text, "Вы не указали количество изделий.", "Количество изделий не может быть отрицательным.", "Вы указали неверное количество изделий."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (boxWeightField.Text == string.Empty)

{

MessageBox.Show("Вы не указали вес ящика.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (boxWeightField.Text != Product.CheckProductCostOrWeight(boxWeightField.Text, "Вес ящика не может быть отрицательным.", "Вы указали недопустимые символы в весе ящика."))

{

MessageBox.Show(Product.CheckProductCostOrWeight(boxWeightField.Text, "Вес ящика не может быть отрицательным.", "Вы указали недопустимые символы в весе ящика."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productCountInBox.Text != Product.CheckCountInBox(productCountInBox.Text, "Вы не указали количество изделий в ящике.", "Количество изделий в ящике не может быть отрицательным.", "Вы указали неверное количество изделий."))

{

MessageBox.Show(Product.CheckCountInBox(productCountInBox.Text, "Вы не указали количество изделий в ящике.", "Количество изделий в ящике не может быть отрицательным.", "Вы указали неверное количество изделий."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productCostAreaField.Text == string.Empty && productCostCountField.Text == string.Empty)

{

MessageBox.Show("Вы должны указать стоимость изделия за штуку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (productCostCountField.Text != string.Empty)

{

if (productCostCountField.Text != Product.CheckProductCostOrWeight(productCostCountField.Text, "Стоимость изделия не может быть отрицательной.", "Вы указали недопустимые символы в стоимости изделия."))

{

MessageBox.Show(Product.CheckProductCostOrWeight(productCostCountField.Text, "Стоимость изделия не может быть отрицательной.", "Вы указали недопустимые символы в стоимости изделия."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

if (productStatusField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать статус наличия изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productDescriptionField.Text != Product.CheckProductDescription(productDescriptionField.Text, "Длина описания не может быть более 300 символов."))

{

MessageBox.Show(Product.CheckProductDescription(productDescriptionField.Text, "Длина описания не может быть более 300 символов."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productImage.Source == null)

{

MessageBox.Show("Необходимо загрузить изображение изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

//Проверки вводимых данных на корректность (конец)

int typeCode = 0;

int collectionCode = 0;

int surfaceCode = 0;

int statusCode = 0;

int colorCode = 0;

string selectCodesQuery = "SELECT productCollectionCode, availabilityStatusCode, productTypeCode, surfaceCode, colorCode FROM Surface, ProductType, AvailabilityStatus, ProductCollection, Color " +

"WHERE productCollectionName = '" + productCollection.Text + "' AND availabilityStatusName = '" + productStatusField.Text + "' AND productTypeName = '" + productTypeField.Text + "'" +

"AND surfaceName = '" + productSurfaceField.Text + "' AND colorName = '" + colorField.Text + "'";

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(selectCodesQuery, connectionString)) //Получение кодов коллекции, статуса наличия, типа и повехности изделия для последующей работы с ними

{

DataTable table = new DataTable();

dataAdapter.Fill(table);

if (table.Rows.Count > 0)

{

typeCode = int.Parse(table.Rows[0]["productTypeCode"].ToString());

collectionCode = int.Parse(table.Rows[0]["productCollectionCode"].ToString());

surfaceCode = int.Parse(table.Rows[0]["surfaceCode"].ToString());

statusCode = int.Parse(table.Rows[0]["availabilityStatusCode"].ToString());

colorCode = int.Parse(table.Rows[0]["colorCode"].ToString());

}

}

string selectUniqProductQuery = "SELECT \* FROM Product " +

"JOIN ProductCollection ON Product.productCollectionCode = ProductCollection.productCollectionCode " +

"JOIN ProductType ON Product.productTypeCode = ProductType.productTypeCode " +

"JOIN Color ON Product.colorCode = Color.colorCode " +

"WHERE productName= '" + productNameField.Text + "' AND productCollectionName = '" + productCollection.Text + "' AND productTypeName = '" + productTypeField.Text + "'";

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(selectUniqProductQuery, connectionString)) //Проверка уникальности товара, то есть его наличия в базе

{

DataTable table = new DataTable();

dataAdapter.Fill(table);

if (table.Rows.Count > 0) //В случае, если таковой имеется, отобразится соответсвующее предупреждение

{

MessageBox.Show("Это изделие уже имеется в базе.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (table.Rows.Count == 0)

{

string selectUniqArticleQuery = "SELECT \* FROM Product WHERE productArticle= '" + productArticleField.Text + "'";

using (SqlDataAdapter articleAdapter = new SqlDataAdapter(selectUniqArticleQuery, connectionString))

{

DataTable articleTable = new DataTable();

articleAdapter.Fill(articleTable);

if (articleTable.Rows.Count > 0) //Проверка уникальности артикула

{

MessageBox.Show("Этот артикул принаджет другому изделию. Проверьте, не допустили ли вы ошибку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

SqlCommand cmd = new SqlCommand(); //Добавление изделия в базу

cmd.CommandType = CommandType.Text;

cmd.CommandText = "INSERT Product (productName, productArticle, productWidth, productLenght, productCount, productBoxWeight, productCountInBox, productCostCount, productCostArea, productImage, productDescription, productCollectionCode, availabilityStatusCode, productTypeCode, surfaceCode, colorCode) " +

"VALUES (@name, @article, @width, @lenght, @count, @boxWeight, @countInBox, @countCost, @areaCost, @image, @description, @collectionCode, @statusCode, @typeCode, @surfaceCode, @colorCode)";

cmd.Parameters.Add("@name", SqlDbType.VarChar).Value = productNameField.Text;

cmd.Parameters.Add("@article", SqlDbType.VarChar).Value = productArticleField.Text;

cmd.Parameters.Add("@width", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productWidth.Text), 2);

cmd.Parameters.Add("@lenght", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productLenght.Text), 2);

cmd.Parameters.Add("@count", SqlDbType.Int).Value = Convert.ToInt32(countField.Text);

cmd.Parameters.Add("@boxWeight", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(boxWeightField.Text), 2);

cmd.Parameters.Add("@countInBox", SqlDbType.Int).Value = Convert.ToInt32(productCountInBox.Text);

cmd.Parameters.Add("@image", SqlDbType.VarChar).Value = productImage.Source.ToString();

cmd.Parameters.Add("@description", SqlDbType.VarChar).Value = productDescriptionField.Text;

cmd.Parameters.Add("@collectionCode", SqlDbType.Int).Value = collectionCode;

cmd.Parameters.Add("@statusCode", SqlDbType.Int).Value = statusCode;

cmd.Parameters.Add("@typeCode", SqlDbType.Int).Value = typeCode;

cmd.Parameters.Add("@surfaceCode", SqlDbType.Int).Value = surfaceCode;

cmd.Parameters.Add("@colorCode", SqlDbType.Int).Value = colorCode;

if (productCostAreaField.IsEnabled == false)

{

cmd.Parameters.Add("@countCost", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productCostCountField.Text), 2);

cmd.Parameters.Add("@areaCost", SqlDbType.Float).Value = DBNull.Value;

}

else

{

cmd.Parameters.Add("@countCost", SqlDbType.Float).Value = DBNull.Value;

cmd.Parameters.Add("@areaCost", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productCostAreaField.Text), 2);

}

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

connectionString.Close();

MessageBox.Show("Добавление изделия прошло успешно.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

ClearFields();

}

}

}

Функция «Удалить сотрудника» для последующего удаления сотрудника обрабатывается событием DeleteButton\_Click, закрепленным за элементом управления DeleteButton «УДАЛИТЬ». Данный компонент расположен в окне «StaffWindow». Код функции представлен ниже:

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

if (EmployeesInfoGrid.SelectedItem == null) //Проверка выбора сотрудника, который будет удален

{

MessageBox.Show("Невозможно удалить информацию о сотруднике, так как он не был выбран.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

DataRowView employeeInfo = (DataRowView)EmployeesInfoGrid.SelectedItems[0]; //Получание информации о сотруднике из таблицы

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "DELETE FROM Employee WHERE employeeLogin = @login"; //Запрос на удаление выбранного сотрудника

cmd.Parameters.Add("@login", SqlDbType.VarChar).Value = employeeInfo["employeeLogin"].ToString();

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

FillDataGrid();

connectionString.Close();

MessageBox.Show("Удаление успешно выполнено.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

Функция «Добавить в корзину» для последующего добавления выбранного изделия в корзину обрабатывается событием Button\_Click, закрепленным за элементом управления Button «ДОБАВИТЬ». Данный компонент расположен в окне «CatalogWindow». Код функции представлен ниже:

private void Button\_Click(object sender, RoutedEventArgs e)

{

StreamReader streamReader = new StreamReader("BasketNumber.txt"); //Считывание номера заказа, к которому будет относится выбранное изделие

string basketNumber = streamReader.ReadLine();

streamReader.Close();

DataRowView productInfo = (DataRowView)ProductsInfoGrid.SelectedItem;

SqlCommand command = new SqlCommand("SELECT productCode FROM Product WHERE productArticle = @article", connectionString);

command.Parameters.AddWithValue("@article", productInfo["productArticle"].ToString());

connectionString.Open();

using (SqlDataReader reader = command.ExecuteReader()) //Получение кода выбранного изделия

{

if (reader.Read())

{

StreamWriter productFile = new StreamWriter("ProductCode.txt");

productFile.Write(reader["productCode"].ToString());

productFile.Close();

}

}

connectionString.Close();

File.WriteAllText(@"ProductCount.txt", string.Empty);

ProductCountWindow productCountWindow = new ProductCountWindow(); //Открытие окна, где будет указано количество изделий для добавления в корзину

productCountWindow.ShowDialog();

File.WriteAllText(@"ProductCode.txt", string.Empty);

int count = 0;

StreamReader countFile = new StreamReader("ProductCount.txt");

if (countFile == null) //Проверка на пустоту файла с количеством изделий для добавления в корзину

{

countFile.Close();

return;

}

else

{

string checkContent = countFile.ReadLine();

if (!String.IsNullOrEmpty(checkContent)) //В случае, если файл не пустой

{

int.TryParse(checkContent, out count);

countFile.Close();

double lenght = 0;

double width = 0;

double weight = 0;

int countInBox = 0;

double costCount = 0;

double costArea = 0;

int code = 0;

string selectProductInfoQuery = "SELECT productCode, productLenght, productWidth, productBoxWeight, productCountInBox, productCostCount, productCostArea, productTypeName FROM Product " +

"JOIN ProductType ON Product.productTypeCode = ProductType.productTypeCode " +

"WHERE productArticle = '" + productInfo["productArticle"].ToString() + "'"; //Получение информации о выбранном изделии

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(selectProductInfoQuery, connectionString))

{

DataTable table = new DataTable();

dataAdapter.Fill(table);

if (table.Rows.Count > 0)

{

code = int.Parse(table.Rows[0]["productCode"].ToString());

lenght = double.Parse(table.Rows[0]["productLenght"].ToString());

width = double.Parse(table.Rows[0]["productWidth"].ToString());

weight = double.Parse(table.Rows[0]["productBoxWeight"].ToString());

countInBox = int.Parse(table.Rows[0]["productCountInBox"].ToString());

if (table.Rows[0]["productTypeName"].ToString() != "Настенная плитка" && table.Rows[0]["productTypeName"].ToString() != "Напольная плитка")

{

costCount = double.Parse(table.Rows[0]["productCostCount"].ToString());

string selectUniqsProductInfoQuery = "SELECT productsCount FROM Basket JOIN Product ON Basket.productCode = Product.productCode " +

"WHERE productArticle = '" + productInfo["productArticle"].ToString() + "' AND basketNumber = '" + basketNumber + "'"; //Проверка на наличие данного изделия

using (SqlDataAdapter adapter = new SqlDataAdapter(selectUniqsProductInfoQuery, connectionString))

{

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0) //В случае, если изделие имеется в корзине, его количество увеличится

{

int currentProductsCount = int.Parse(dataTable.Rows[0]["productsCount"].ToString());

currentProductsCount = currentProductsCount + count;

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "UPDATE Basket SET productCode = @productCode, basketNumber = @number, productsCount = @count, boxesCount = @boxes, productsArea = @area, productsWeight = @weight, generalSum = @sum, paymentStatus = @status " +

"WHERE productCode = @productCode AND basketNumber = @number";

cmd.Parameters.Add("@number", SqlDbType.VarChar).Value = basketNumber;

cmd.Parameters.Add("@count", SqlDbType.Int).Value = currentProductsCount;

cmd.Parameters.Add("@boxes", SqlDbType.Int).Value = currentProductsCount / countInBox + 1;

cmd.Parameters.Add("@area", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(currentProductsCount \* (lenght \* width / 1000000)), 2);

cmd.Parameters.Add("@weight", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(weight / countInBox \* currentProductsCount), 2);

cmd.Parameters.Add("@sum", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(currentProductsCount \* costCount), 2);

cmd.Parameters.Add("@productCode", SqlDbType.Float).Value = code;

cmd.Parameters.Add("@status", SqlDbType.Bit).Value = false;

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

connectionString.Close();

}

else //В случае отсутсвия изделия в корзине, случится его добавление

{

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "INSERT Basket (basketNumber, productsCount, boxesCount, productsArea, productsWeight, generalSum, productCode, paymentStatus) " +

"VALUES (@number, @count, @boxes, @area, @weight, @sum, @productCode, @status)";

cmd.Parameters.Add("@number", SqlDbType.VarChar).Value = basketNumber;

cmd.Parameters.Add("@count", SqlDbType.Int).Value = count;

cmd.Parameters.Add("@boxes", SqlDbType.Int).Value = count / countInBox + 1;

cmd.Parameters.Add("@area", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(count \* (lenght \* width / 1000000)), 2);

cmd.Parameters.Add("@weight", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(weight / countInBox \* count), 2);

cmd.Parameters.Add("@sum", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(count \* costCount), 2);

cmd.Parameters.Add("@productCode", SqlDbType.Float).Value = code;

cmd.Parameters.Add("@status", SqlDbType.Bit).Value = false;

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

connectionString.Close();

}

}

}

else

{

costArea = double.Parse(table.Rows[0]["productCostArea"].ToString());

string selectUniqsProductInfoQuery = "SELECT productsCount FROM Basket JOIN Product ON Basket.productCode = Product.productCode " +

"WHERE productArticle = '" + productInfo["productArticle"].ToString() + "' AND basketNumber = '" + basketNumber + "'"; //Проверка на наличие данного изделия

using (SqlDataAdapter adapter = new SqlDataAdapter(selectUniqsProductInfoQuery, connectionString))

{

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0) //В случае, если изделие имеется в корзине, его количество увеличится

{

int currentProductsCount = int.Parse(dataTable.Rows[0]["productsCount"].ToString());

currentProductsCount = currentProductsCount + count;

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "UPDATE Basket SET productCode = @productCode, basketNumber = @number, productsCount = @count, boxesCount = @boxes, productsArea = @area, productsWeight = @weight, generalSum = @sum, paymentStatus = @status " +

"WHERE productCode = @productCode AND basketNumber = @number";

cmd.Parameters.Add("@number", SqlDbType.VarChar).Value = basketNumber;

cmd.Parameters.Add("@count", SqlDbType.Int).Value = currentProductsCount;

cmd.Parameters.Add("@boxes", SqlDbType.Int).Value = currentProductsCount / countInBox + 1;

cmd.Parameters.Add("@area", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(currentProductsCount \* (lenght \* width / 1000000)), 2);

cmd.Parameters.Add("@weight", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(weight / countInBox \* currentProductsCount), 2);

cmd.Parameters.Add("@sum", SqlDbType.Float).Value = Math.Round(currentProductsCount \* (costArea \* (lenght \* width / 1000000)), 2);

cmd.Parameters.Add("@productCode", SqlDbType.Float).Value = code;

cmd.Parameters.Add("@status", SqlDbType.Bit).Value = false;

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

connectionString.Close();

}

else //В случае отсутсвия изделия в корзине, случится его добавление

{

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "INSERT Basket (basketNumber, productsCount, boxesCount, productsArea, productsWeight, generalSum, productCode, paymentStatus) " +

"VALUES (@number, @count, @boxes, @area, @weight, @sum, @productCode, @status)";

cmd.Parameters.Add("@number", SqlDbType.VarChar).Value = basketNumber;

cmd.Parameters.Add("@count", SqlDbType.Int).Value = count;

cmd.Parameters.Add("@boxes", SqlDbType.Int).Value = count / countInBox + 1;

cmd.Parameters.Add("@area", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(count \* (lenght \* width / 1000000)), 2);

cmd.Parameters.Add("@weight", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(weight / countInBox \* count), 2);

cmd.Parameters.Add("@sum", SqlDbType.Float).Value = Math.Round(count \* (costArea \* (lenght \* width / 1000000)), 2);

cmd.Parameters.Add("@productCode", SqlDbType.Float).Value = code;

cmd.Parameters.Add("@status", SqlDbType.Bit).Value = false;

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

connectionString.Close();

}

}

}

MessageBox.Show("Изделие добавлено в корзину.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

string updateProductCount = "SELECT productCount FROM Product WHERE productArticle = '" + productInfo["productArticle"].ToString() + "'";

using (SqlDataAdapter countAdapter = new SqlDataAdapter(updateProductCount, connectionString))

{

DataTable dataTable = new DataTable();

countAdapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

int productCurrentCount = int.Parse(dataTable.Rows[0]["productCount"].ToString());

SqlCommand updateProductInfoCmd = new SqlCommand();

updateProductInfoCmd.CommandType = CommandType.Text;

if (productCurrentCount - count == 0)

{

updateProductInfoCmd.CommandText = "UPDATE Product SET productCount = @count, availabilityStatusCode = @status WHERE productArticle = @article";

updateProductInfoCmd.Parameters.Add("@count", SqlDbType.Int).Value = productCurrentCount - count;

updateProductInfoCmd.Parameters.Add("@status", SqlDbType.Int).Value = 2;

updateProductInfoCmd.Parameters.Add("@article", SqlDbType.VarChar).Value = productInfo["productArticle"].ToString();

updateProductInfoCmd.Connection = connectionString;

connectionString.Open();

updateProductInfoCmd.ExecuteNonQuery();

connectionString.Close();

}

else

{

updateProductInfoCmd.CommandText = "UPDATE Product SET productCount = @count WHERE productArticle = @article";

updateProductInfoCmd.Parameters.Add("@count", SqlDbType.Int).Value = productCurrentCount - count;

updateProductInfoCmd.Parameters.Add("@article", SqlDbType.VarChar).Value = productInfo["productArticle"].ToString();

updateProductInfoCmd.Connection = connectionString;

connectionString.Open();

updateProductInfoCmd.ExecuteNonQuery();

connectionString.Close();

}

}

}

ProductsInfoGrid.ItemsSource = null;

ProductsInfoGrid.Items.Refresh();

connectionString.Open();

FillDataGrid();

connectionString.Close();

}

}

}

else

{

countFile.Close();

return;

}

}

}

Остальной код программы представлен в приложении А.

## Проектирование справочной системы приложения

Для работы с приложением начинающего пользователя необходимо обеспечить качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе определенного пункта меню; сведения о том, какую информацию и в каком виде следует вводить в соответствующие поля.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может пригодиться пользователю: о возможностях приложения и о разработчике программы.

Система справки данного программного средства будет содержать следующие разделы:

* «1. О программе»;
* «2. Начало работы»;
* «3. Руководство администратора»;
* «4. Руководство сотрудника»;
* «5. О разработчике».

Справочная система будет создана с помощью программного средства Dr.Explain.

В разделе «1. О программе» предоставляется информация о программном средстве, а именно: предназначение, цели создания, описание возможностей.

В разделе «2. Начало работы» содержится информация о том, как запустить приложение, об окне, которое появится после запуска программного средства, а также описание элементов, находящихся в нем.

В разделе «3. Руководство администратора» предоставлена информация обо всех возможностях администратора, в том числе и описание всех кнопок, полей и окон, с которыми взаимодействует администратор.

В разделе «4. Руководство сотрудника» содержится информация, а также описание всех кнопок, окон и полей, которые доступны для взаимодействия сотруднику.

В разделе «5. О разработчике» предоставлена информация о разработчике.

# Описание программного средства

## Общие сведения

Разрабатываемое программное средство предназначено для автоматизации учета реализации керамических изделий, поиска, фильтрации и работы с информацией, касающейся непосредственно этих керамических изделий, заказов, сотрудников и статистики.

Программное средство позволяет добавлять, удалять и изменять данные, то есть, иначе говоря, осуществляет ведение базы данных.

Данное программное средство было разработано при использовании персонального компьютера со следующей конфигурацией:

* процессор Intel Core i5 2,5 ГГц;
* видеокарта NVidia GeForce GTX 1050 (2 ГБ GDDR5);
* оперативная память 8 ГБ (DDR4);
* SSD диск на 128 ГБ + HDD диск на 1 ТБ;
* интегрированная видеокарта Intel HD Graphics 630.

Программное средство создано в среде разработки Visual Studio 2019 на языке программирования C# в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows, начиная с Windows 8. Программа не требовательна к системным ресурсам компьютера, также очень проста в использовании. Обладание специальными профессиональными навыками для работы с ней не требуется. Для работы данного программного средства необходима предварительная установка и настройка следующих программных продуктов:

* система управления базами данных Microsoft SQL Server 2017;
* платформа Microsoft .Net Framework 4.7.2.

Примерный размер программного средства составляет 470 Мбайт

Инсталляция программного средства не требуется. Достаточно будет скопировать готовый проект на персональный компьютер, подключить базу данных к серверу с именем «экземпляр\_сервера\SQLEXPRESS» и запустить файл «KeraminStore.sln».

## Функциональное назначение

Программное средство разрабатывалось для сотрудников с целью упрощения работы с информацией о керамических изделиях, сотрудниках, пунктах самовывоза и статистике, путём ведения базы данных, а также повышения их работоспособности.

Программа использует стандартные элементы управления, такие как кнопки, меню, поля ввода, таблицы, что обеспечивает единство интерфейса системы и программного средства.

В работе программного средства предусмотрены некоторые ситуации, которые должны предупреждать сотрудника, чтобы он выполнял все необходимые требования по эксплуатации программы. Для этого существуют сообщения системы. Например, если сотрудник заполнил не все поля при оформлении заказа, то на экран выведется сообщение, где будет указано об отсутствии информации в том или ином поле.

Таким образом, программа может применяться в реальных условиях, представляя собой достаточно удобный помощник.

## Входные и выходные данные

В данном программном средстве входными данными являются: информация об изделиях, о покупателях и сотрудниках.

Входными данными при добавлении изделия являются:

* наименование изделия;
* артикул изделия;
* ширина изделия;
* длина изделия;
* количество изделий в наличии;
* вес ящика с изделиями;
* количество изделий в ящике;
* стоимость единицы изделия;
* стоимость за квадратный метр изделия;
* изображение изделия;
* описание изделия;
* статус наличия изделия;
* коллекция изделия;
* тип изделия;
* цвет изделия;
* поверхность изделия.

Входными данными при добавлении покупателя являются:

* имя покупателя;
* фамилия покупателя;
* отчество покупателя;
* контактный телефон;
* почтовый адрес;
* номер здания;
* этаж;
* квартира или офис;
* наименование юридического лица;
* ИНН (для юридического лица);
* город;
* улица.

Входными данными при регистрации сотрудника являются:

* имя сотрудника;
* фамилия сотрудника;
* отчество сотрудника;
* логин сотрудника;
* пароль сотрудника;
* дата рождения сотрудника;
* должность сотрудника.

Входными данными при добавлении пункта самовывоза являются:

* город;
* улица;
* номер здания.

Выходными данными являются товарные чеки для юридических и физических лиц, которые представлены в приложении Б на рисунках Б.1 и Б.2, а также статистика за определенный период по продуктивности сотрудников и популярным изделиям, представленные на рисунках Б.3 и Б.4.

# Методика испытаний

## Технические требования

Минимальными требованиями для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* процессор Intel Core i5 2,5 ГГц;
* видеокарта NVidia GeForce GTX 1050 (2 ГБ GDDR5);
* оперативная память 8 ГБ (DDR4);
* SSD диск на 128 ГБ + HDD диск на 1 ТБ;
* интегрированная видеокарта Intel HD Graphics 630;
* программа для работы с таблицами Microsoft Excel 2016.

## Функциональное тестирование

Данное тестирование проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Функциональное тестирование предполагает проверку выполнения определенных на этапе проектирования функций.

Начальное окно, оно же окно авторизации, представлено на рисунке 5.1.

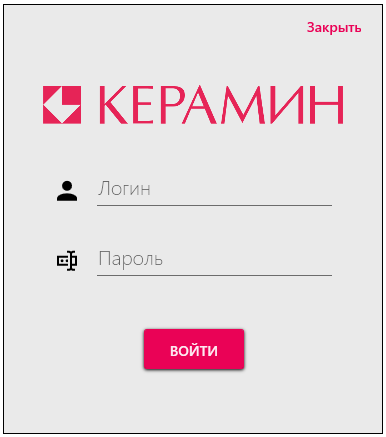


Рисунок 5.1 – Окно авторизации

Протестируем функцию авторизации. Для этого, находясь в данном окне, введем логин и пароль, после чего нажмем на кнопку «ВОЙТИ».

Тест кейсы для функции авторизации представлены в таблице 5.1.

Таблица 5.1 – Тест-кейсы для функции «Авторизация»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 1 | Авторизация | 1. Заполнить поля данными:   * Логин: не заполняем * Пароль: «Qwerty1»   2. Нажать на кнопку «ВОЙТИ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.2. |
| Фактический результат: соответствует ожидаемому |
| 2 | Авторизация | 1. Заполнить поля данными:   * Логин: «Admin» * Пароль: «fuieonbfwhi»   2. Нажать на кнопку «ВОЙТИ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.2. |
| Фактический результат: соответствует ожидаемому |

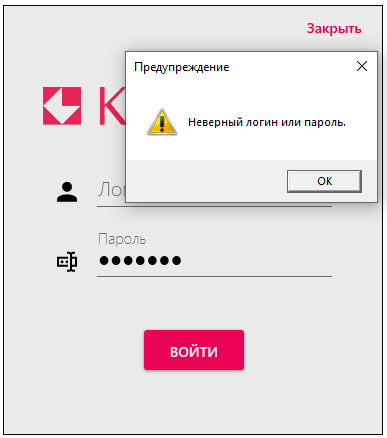


Рисунок 5.2 – Предупреждение, связанное с вводом неверных данных при авторизации

После успешной авторизации отобразится главное меню. В случае если авторизация осуществлялась от имени администратора, то главное меню будет выглядеть также, как на рисунке 5.3. В случае если авторизация осуществлялась от имени обычного сотрудника, то последних двух пунктов, представленных в левом меню, не будет. Это сделано с целью разделения функционала.



Рисунок 5.3 – Окно главное меню

Протестируем функцию изменения данных учетной записи. Для этого, находясь в окне главного меню, необходимо нажать на кнопку, расположенную в правом верхнем углу. Затем появится выпадающее меню, в котором необходимо нажать на кнопку «Настройки», после чего откроется окно изменения данных учетной записи, представленное на рисунке 5.4.

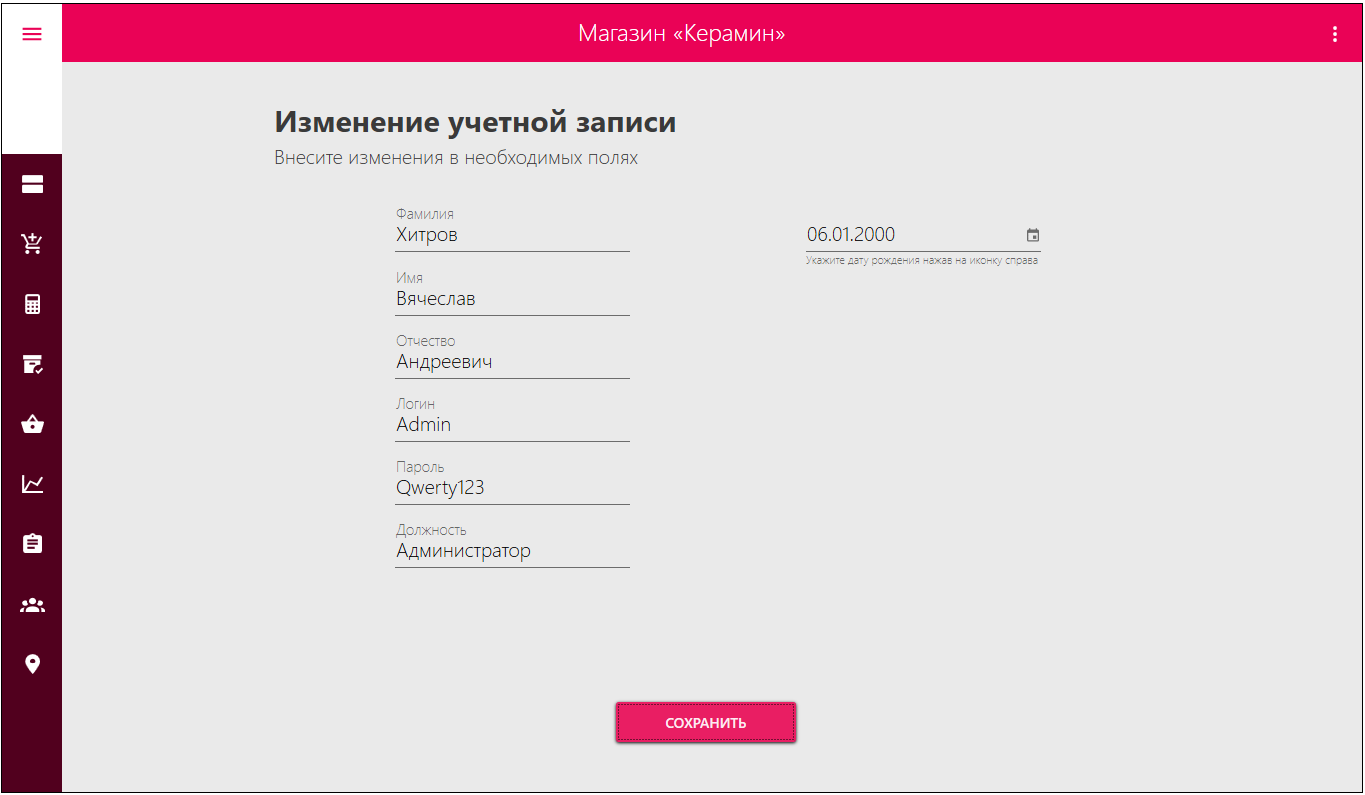


Рисунок 5.4 – Окно изменения данных учетной записи

Тест-кейсы для функции изменения данных учетной записи представлены в таблице 5.2.

Таблица 5.2 – Тест-кейсы для функции «Изменение данных учетной записи»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 3 | Изменение данных учетной записи | 1. Изменить данные полей:   * Фамилия: «Хитров» * Имя: «Вячеслав» * Отчество: «Андреевич» * Логин: «Admin» * Пароль: «Qwerty123» * Должность: «Администратор» * Дата рождения: «06.01.2000»   2. Нажать на кнопку «СОХРАНИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.5. |
| Фактический результат: соответствует ожидаемому |
| 4 | Изменение данных учетной записи | 1. Изменить данные полей:   * Фамилия: «!Хитров!» * Имя: «Вячеслав» * Отчество: «Андреевич» * Логин: «Admin» * Пароль: «Qwerty123» * Должность: «Администратор» * Дата рождения: «06.01.2000»   2. Нажать на кнопку «СОХРАНИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.6. |
| Фактический результат: соответствует ожидаемому |
| 5 | Изменение данных учетной записи | 1. Изменить данные полей:   * Фамилия: «Хитров» * Имя: «Вячеслав» * Отчество: «Андреевич» * Логин: «Admin» * Пароль: оставляем пустым * Должность: «Администратор» * Дата рождения: «06.01.2000»   2. Нажать на кнопку «СОХРАНИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.7. |
| Фактический результат: соответствует ожидаемому |

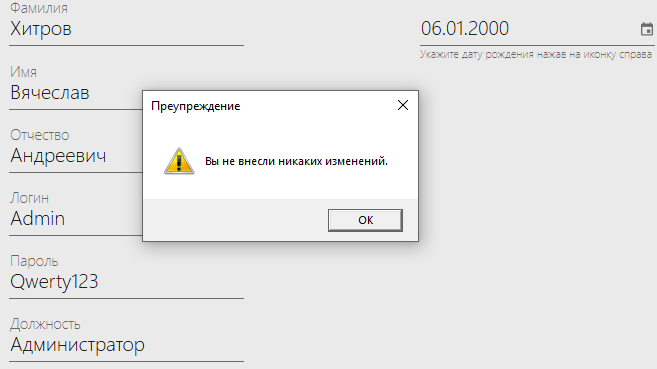


Рисунок 5.5 – Предупреждение, связанное с отсутствием изменений

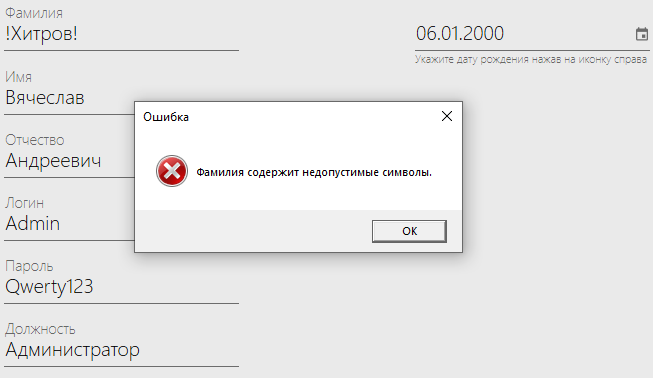


Рисунок 5.6 – Предупреждение, связанное с вводом недопустимых символов

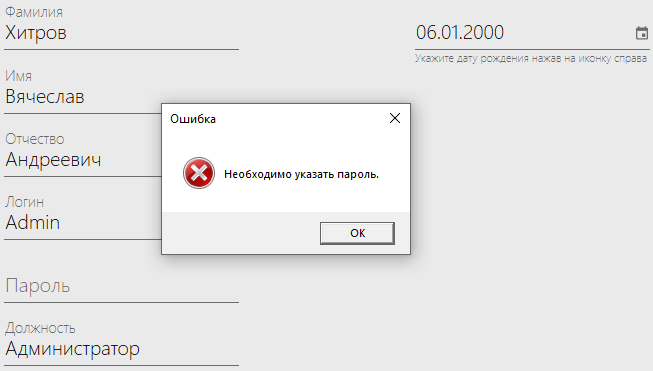


Рисунок 5.7 – Предупреждение, связанное с пустым полем

В случае, если сотрудник успешно изменит данные, то результатом будет уведомление, представленное на рисунке 5.8.

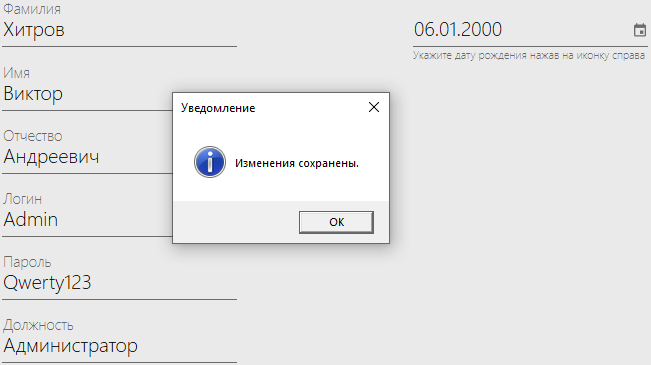


Рисунок 5.8 – Уведомление об успешном изменении данных учетной записи

Протестируем функцию регистрации нового сотрудника. Для этого, необходимо нажать на кнопку в левом верхнем углу, после чего отобразится боковое меню. В данном меню необходимо нажать на кнопку «Сотрудники». Откроется контекстное меню, котором потребуется нажать на пункт «Добавить нового сотрудника». После этого откроется окно добавления сотрудника, представленное на рисунке 5.9.

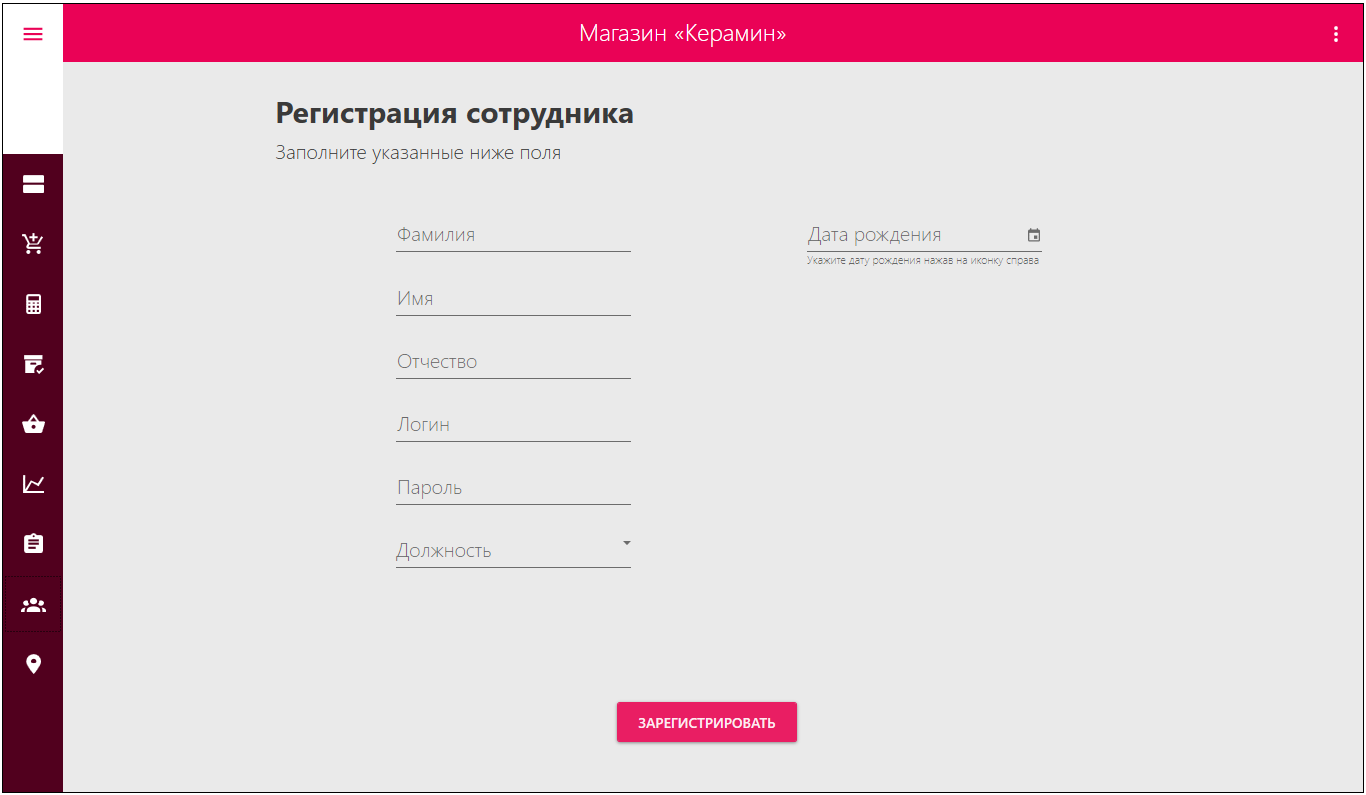


Рисунок 5.9 – Окно регистрации нового сотрудника

Тест-кейс для функции добавления сотрудника представлен в таблице 5.3.

Таблица 5.3 – Тест-кейс для функции «Регистрация сотрудника»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 6 | Регистрация сотрудника | 1. Заполнить поля данными:   * Фамилия: «Серов» * Имя: «Андрей» * Отчество: «Владимирович» * Логин: «Admin» * Пароль: «Qwerty123» * Должность: «Штатный сотрудник» * Дата рождения: «09.09.2001»   2. Нажать на кнопку «СОХРАНИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.10. |
| Фактический результат: соответствует ожидаемому |

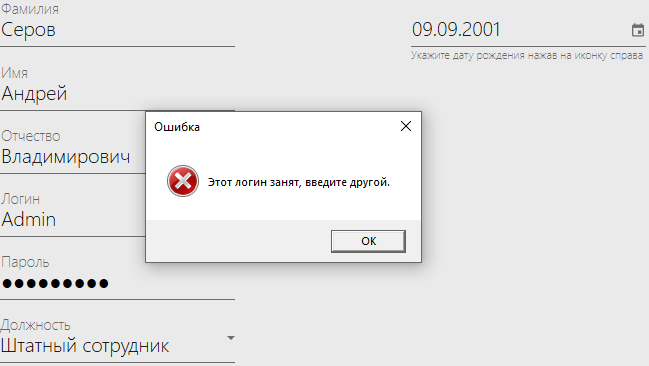


Рисунок 5.10 – Предупреждение, связанное с указанием занятого логина

Все проверки вводимых для регистрации данных идентичны тем, что используются при изменении данных учетной записи.

При успешном добавлении сотрудника отобразится уведомление, представленное на рисунке 5.11.

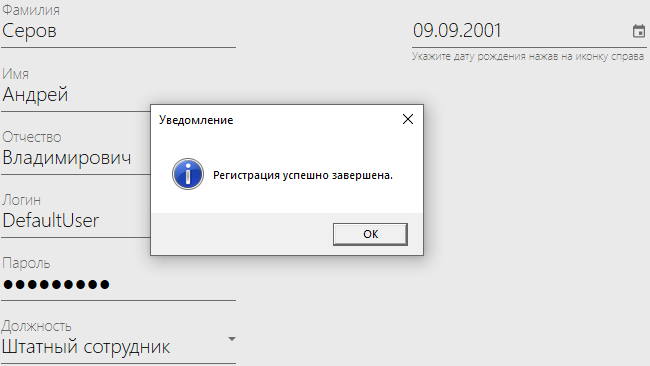


Рисунок 5.11 – Уведомление об успешном добавлении сотрудника

Теперь нажмем на иконку в левом верхнем углу, после чего, в открывшемся боковом меню, нажмем на кнопку «Сотрудники». Откроется контекстное меню, котором потребуется нажать на пункт «Список сотрудников». После чего откроется окно со списком сотрудников, представленное на рисунке 5.12. В целях предотвращения ошибок, данные об администраторе, который авторизовался в системе, не выводятся.

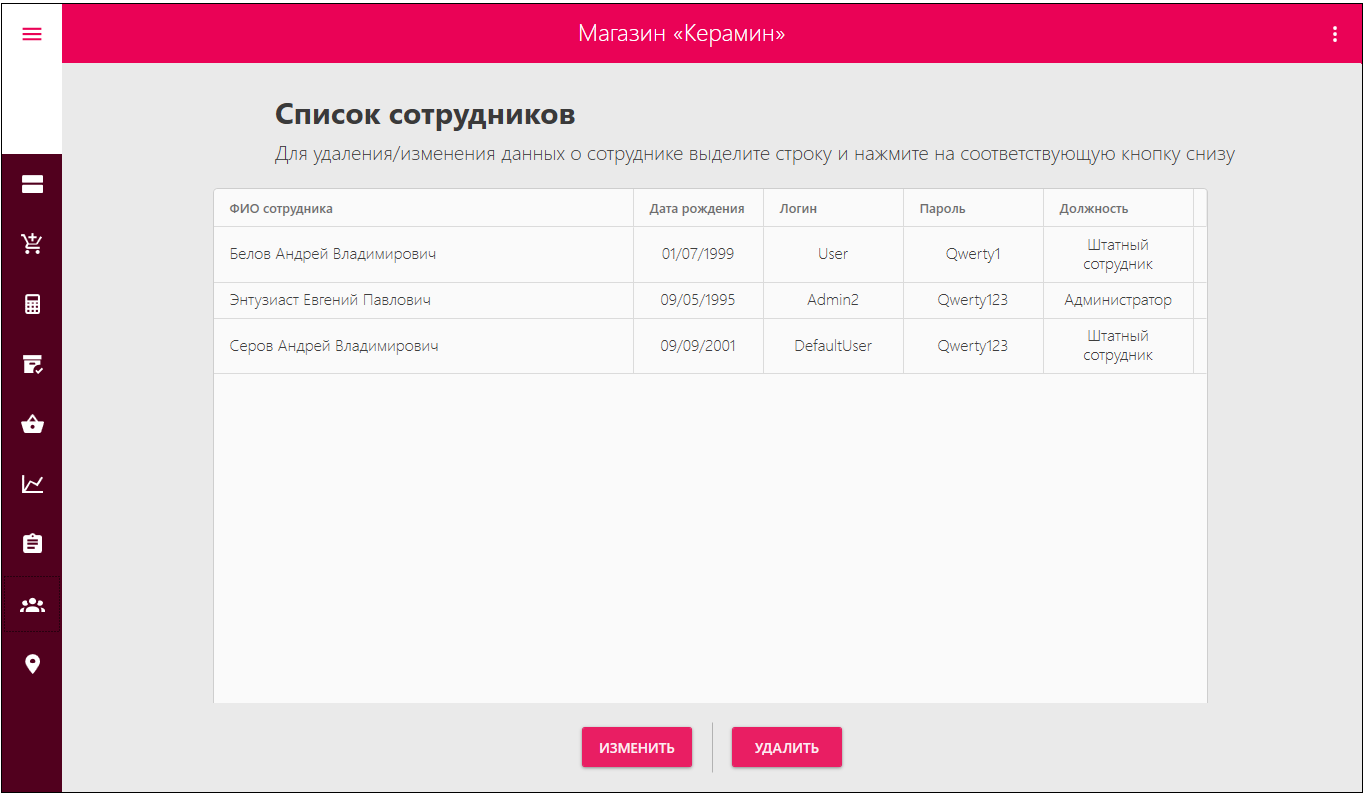


Рисунок 5.12 – Окно со списком сотрудников

Протестируем функцию изменения данных сотрудников. Для этого, выделив необходимую строку среди списка, нажмем на кнопку «ИЗМЕНИТЬ». В случае, если не выделили запись и нажали на кнопку, то отобразится предупреждение, представленное на рисунке 5.13. Также, при выборе нескольких записей, окно изменения данных сотрудника, представленное на рисунке 5.14, откроется только для самой первой записи среди выделенных.

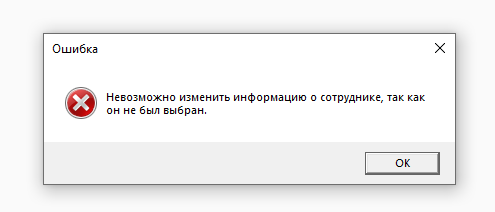


Рисунок 5.13 – Предупреждение, связанное с тем, что сотрудник из списка не был выбран



Рисунок 5.14 – Окно изменения данных выбранного сотрудника

Все проверки данных идентичны тем, что используются при изменении данных учетной записи и регистрации нового сотрудника.

Протестируем функцию удаления сотрудников. Для этого нажмем на кнопку «Закрыть», находясь в окне изменения данных сотрудника, после чего вновь откроется окно со списком всех сотрудников. Выберем в списке необходимого для удаления сотрудника и нажмем на кнопку «УДАЛИТЬ». После этого отобразится уведомление об успешном удалении сотрудника, представленное на рисунке 5.15. В случае, если сотрудник не был выбран, то отобразится предупреждение, представленное на рисунке 5.16.

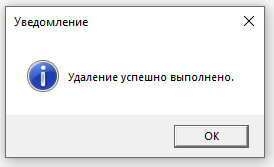


Рисунок 5.15 – Уведомление об успешном удалении сотрудника

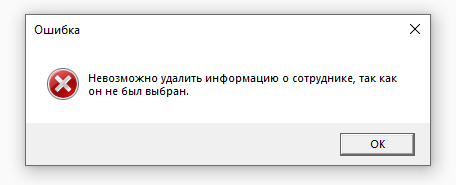


Рисунок 5.16 – Предупреждение, связанное с тем, что сотрудник из списка не был выбран

Протестируем функцию добавления нового изделия. Для этого нажмем на иконку в левом верхнем углу, а в боковом меню нажмем на кнопку «Изделия». Откроется контекстное меню, котором потребуется нажать на пункт «Добавить новое изделие». После откроется окно добавления изделия, представленное на рисунке 5.17.

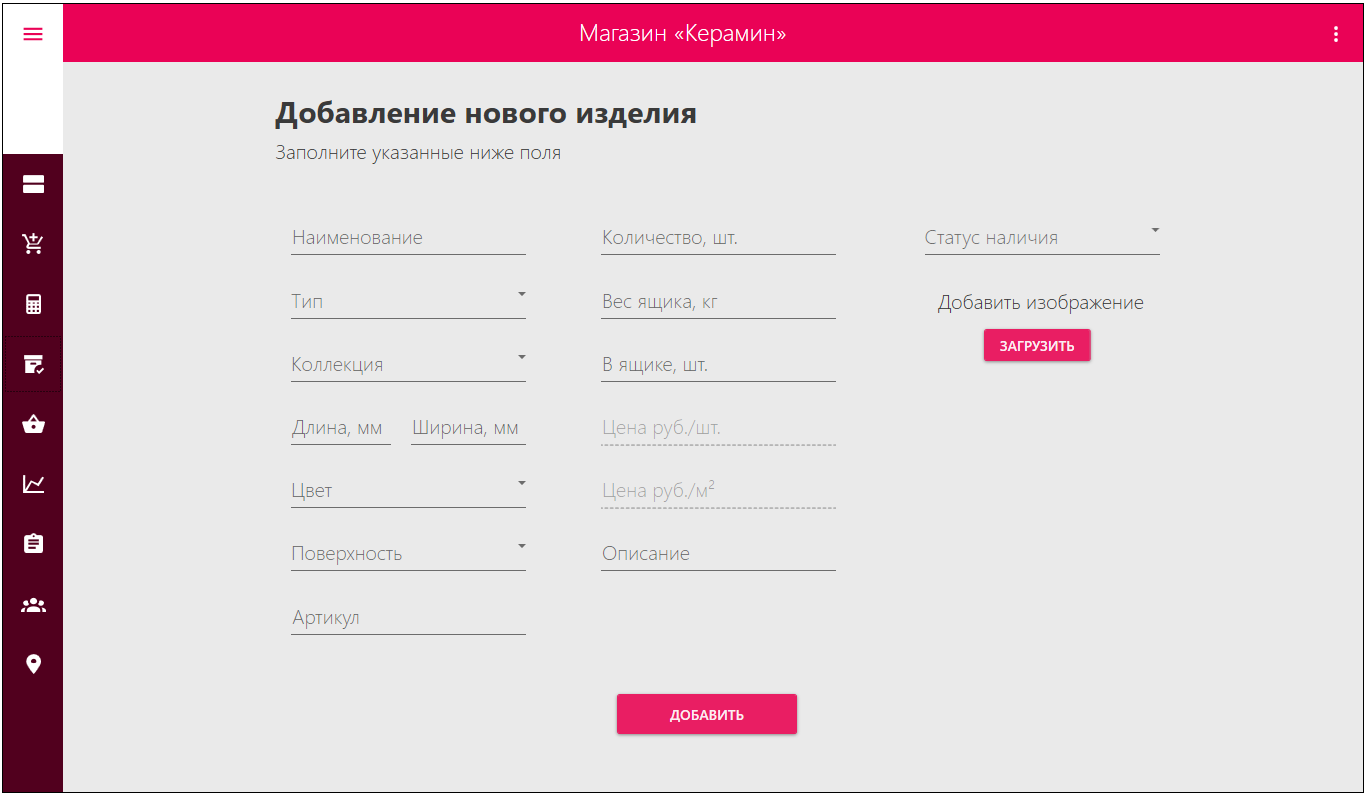


Рисунок 5.17 – Окно добавления нового изделия

Тест-кейсы для функции добавления нового изделия представлены в таблице 5.4.

Таблица 5.4 – Тест-кейсы для функции «Добавление нового изделия»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 7 | Добавление нового изделия | 1. Заполнить поля данными:   * Наименование: «Керамическая плитка Керамин Классик 5Т 300х93,8» * Тип: «Настенная плитка» * Коллекция: «Классик» * Длина, мм: «300» * Ширина, мм: «93,8» * Цвет: «Черный» * Поверхность: «Глянцевая» * Артикул: «CDB00028925» * Количество, шт.: «5000» * Вес ящика, кг: «7,37» * В ящике, шт.: «16» * Цена руб./шт.: недоступно * Цена руб./ м².: «35,81» * Описание: оставляем пустым * Статус наличия: «Есть в наличии»   2. Нажать на кнопку «ЗАГРУЗИТЬ» для добавления изображения  3. Нажать на кнопку «ДОБАВИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.18. |
| Фактический результат: соответствует ожидаемому |
| 8 | Добавление нового изделия | 1. Заполнить поля данными:   * Наименование: «Керамическая плитка Керамин Мари Эрми 7С 750х250» * Тип: «Настенная плитка» * Коллекция: «Мари Эрми» * Длина, мм: «750» * Ширина, мм: «250» * Цвет: «Разноцветный» * Поверхность: оставляем пустой * Артикул: «CDB00022604» * Количество, шт.: «800» * Вес ящика, кг: «28,88» * В ящике, шт.: «9» * Цена руб./шт.: недоступно * Цена руб./ м².: «42,89» * Описание: оставляем пустым * Статус наличия: «Есть в наличии»   2. Нажать на кнопку «ЗАГРУЗИТЬ» для добавления изображения  3. Нажать на кнопку «ДОБАВИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.19. |
| Фактический результат: соответствует ожидаемому |

Продолжение таблицы 5.4

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 9 | Добавление нового изделия | 1. Заполнить поля данными:   * Наименование: «Керамическая плитка Керамин Мари Эрми 7С 750х250» * Тип: «Настенная плитка» * Коллекция: «Мари Эрми» * Длина, мм: «750» * Ширина, мм: «250» * Цвет: «Разноцветный» * Поверхность: «Матовая» * Артикул: «CDB00028925» * Количество, шт.: «800» * Вес ящика, кг: «28,88» * В ящике, шт.: «9» * Цена руб./шт.: недоступно * Цена руб./ м².: «42,89» * Описание: оставляем пустым * Статус наличия: «Есть в наличии»   2. Нажать на кнопку «ЗАГРУЗИТЬ» для добавления изображения  3. Нажать на кнопку «ДОБАВИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.20. |
| Фактический результат: соответствует ожидаемому |
| 10 | Добавление нового изделия | 1. Заполнить поля данными:   * Наименование: «Керамическая плитка Керамин Мари Эрми 7С 750х250» * Тип: «Настенная плитка» * Коллекция: «Мари Эрми» * Длина, мм: «750» * Ширина, мм: «250» * Цвет: «Разноцветный» * Поверхность: «Матовая» * Артикул: «CDB00022604» * Количество, шт.: «800» * Вес ящика, кг: «28,88» * В ящике, шт.: «9» * Цена руб./шт.: недоступно * Цена руб./ м².: «42,89» * Описание: оставляем пустым * Статус наличия: «Есть в наличии»   2. Не нажимать на кнопку «ЗАГРУЗИТЬ» для добавления изображения  3. Нажать на кнопку «ДОБАВИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.21. |
| Фактический результат: соответствует ожидаемому |



Рисунок 5.18 – Предупреждение о попытке добавления изделия, которое уже имеется в базе

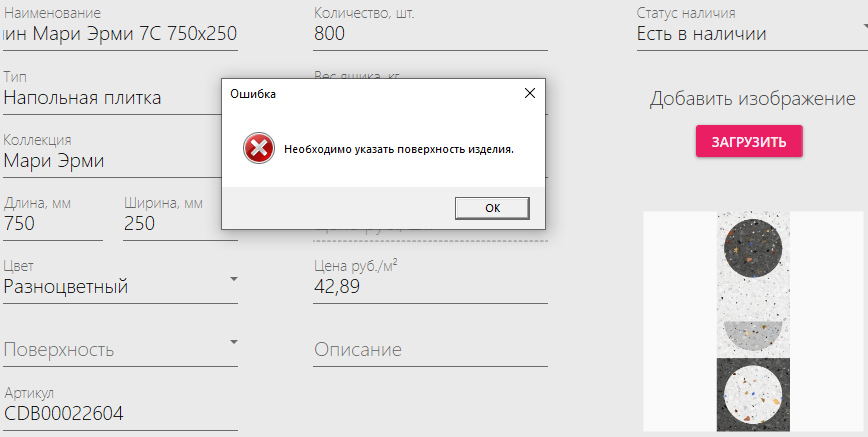


Рисунок 5.19 – Предупреждение, связанное с тем, что значение поля не было выбрано

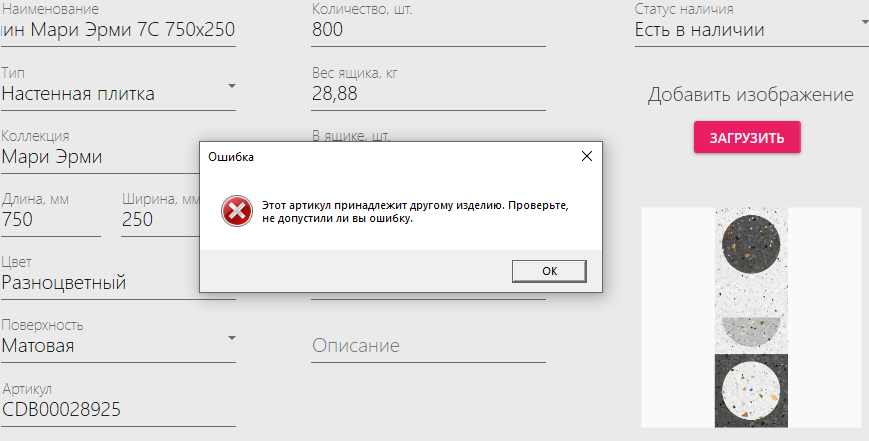


Рисунок 5.20 – Предупреждение, связанное с тем, что введенный артикул принадлежит другому изделию

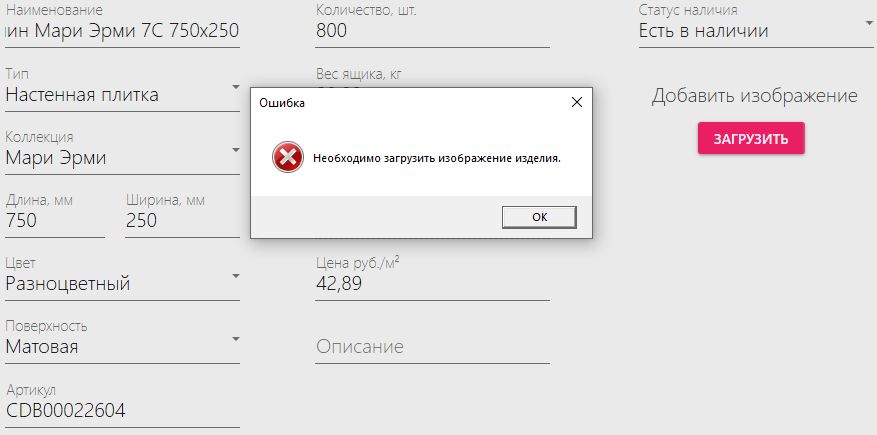


Рисунок 5.21 – Предупреждение, связанное с тем, что изображение не было загружено

В случае успешного заполнения всех полей отобразится уведомление, представленное на рисунке 5.22.

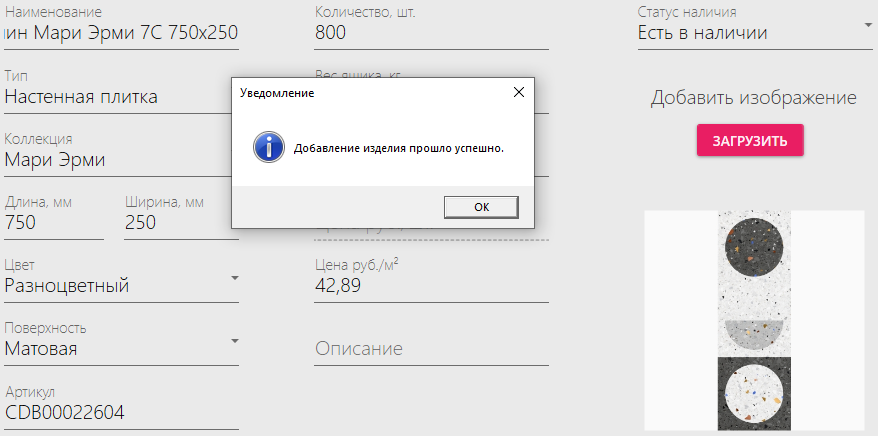


Рисунок 5.22 – Уведомление об успешном добавлении изделия

Нажмем на иконку в левом верхнем углу, после чего, в открывшемся боковом меню, нажмем на кнопку «Изделия». Откроется контекстное меню, котором потребуется нажать на пункт «Список изделий». После чего откроется окно со списком изделий, представленное на рисунке 5.23.

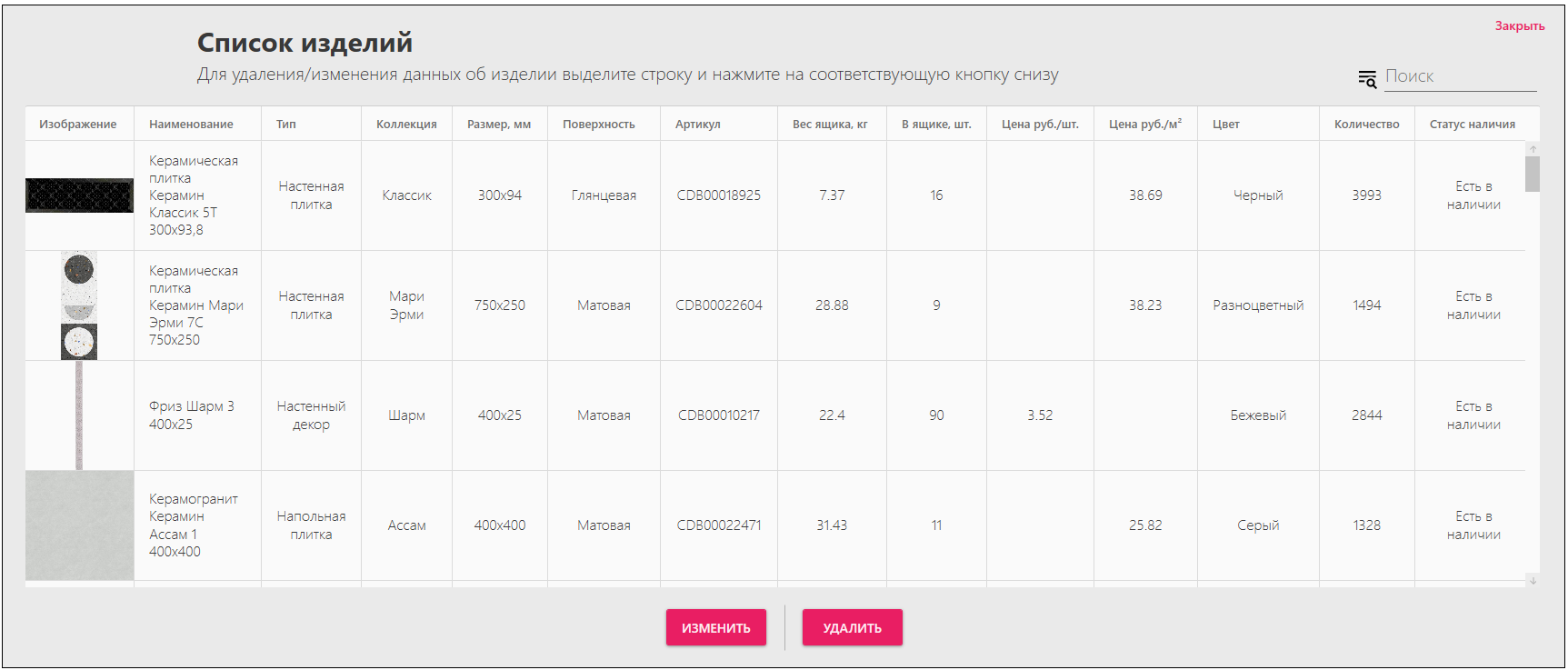


Рисунок 5.23 – Окно со списком изделий

Протестируем функцию изменения данных изделий. Для этого, выделив необходимую строку среди списка, нажмем на кнопку «ИЗМЕНИТЬ». В случае, если не выделили запись и нажали на кнопку, то отобразится предупреждение, представленное на рисунке 5.28, только для изделия. Также, при выборе нескольких записей, окно изменения данных изделия, представленное на рисунке 5.24, откроется только для самой первой записи среди выделенных.

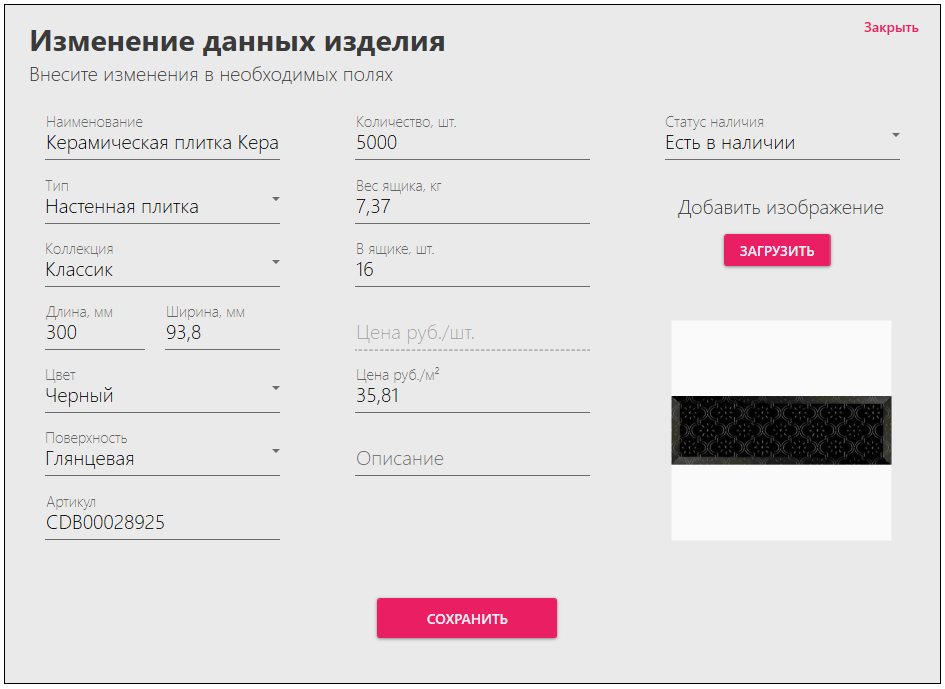


Рисунок 5.24 – Окно изменения данных выбранного изделия

Все проверки данных идентичны тем, что используются при добавлении нового изделия.

Протестируем функцию удаления изделий. Для этого нажмем на кнопку «Закрыть», находясь в окне изменения данных изделия, после чего вновь откроется окно со списком всех изделий. Выберем в списке необходимое для удаления изделие и нажмем на кнопку «УДАЛИТЬ». После этого отобразится уведомление об успешном удалении изделия, представленное на рисунке 5.15. В случае, если изделие не было выбрано, то отобразится предупреждение, представленное на рисунке 5.16, только для изделия.

Протестируем функцию расчета необходимого количества изделий для комнаты или какой-либо поверхности. Для этого, в левом верхнем углу, нажмем на иконку и после чего в боковом меню нажмем на кнопку «Калькулятор плитки». Далее откроется окно, представленное на рисунке 5.25.

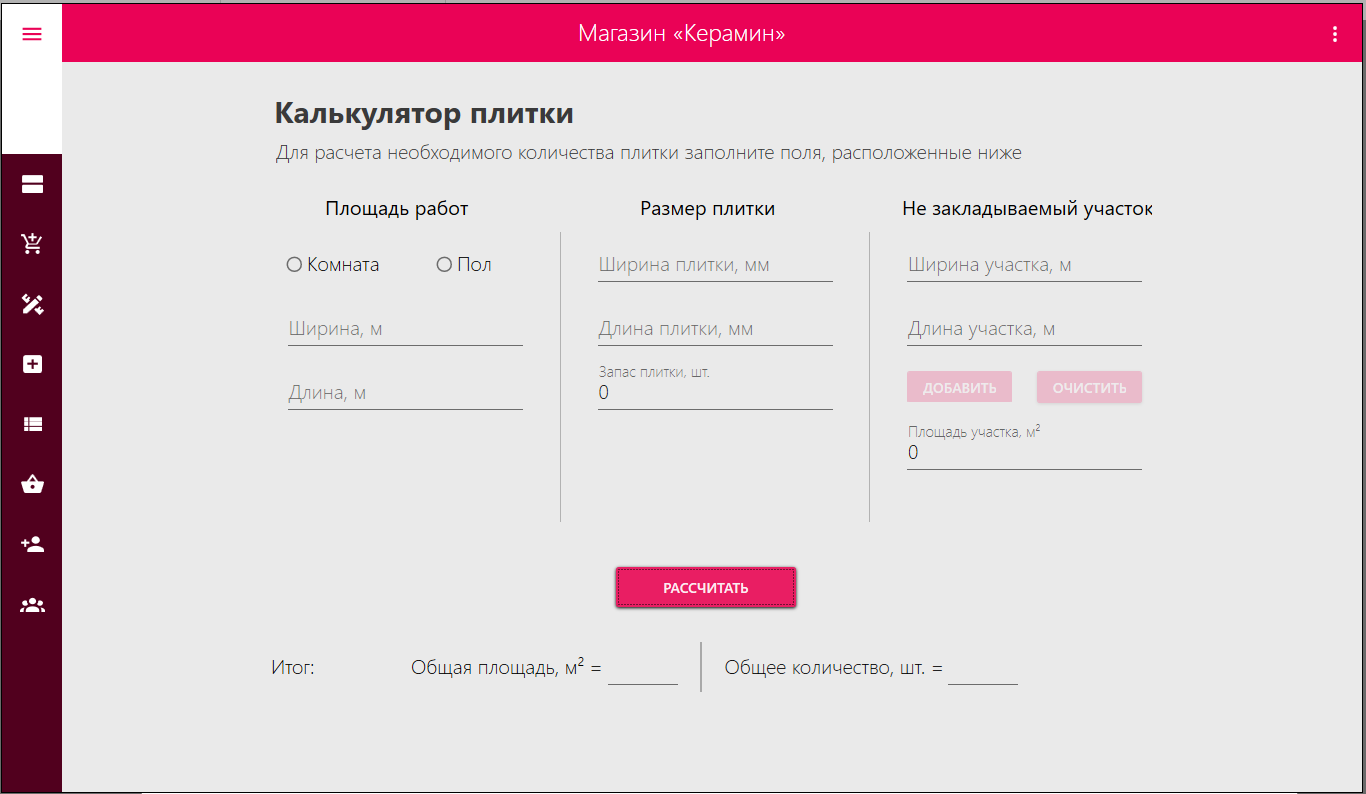


Рисунок 5.25 – Окно расчета необходимого количества изделий

Тест-кейсы для функции расчета необходимого количества изделий представлены в таблице 5.5. В связи с тем, что площадь работ не была выбрана, отобразится предупреждение, представленное на рисунке 5.26. До тех пор, пока площадь работ не выбрана, нельзя вносить никакие изменения в поля.

Таблица 5.5 – Тест-кейсы для функции «Расчет необходимого количества изделий»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 11 | Расчет необходимого количества изделий | 1. Заполнить поля данными:   * Площадь работы: «Пол» * Ширина, м: не указываем * Длина, м: «2» * Ширина плитки, мм: «93,8» * Длина плитки, мм: «300» * Запас плитки: «0» * Ширина участка, м: не заполняем * Длина участка, м: не заполняем   2. Нажать на кнопку «РАССЧИТАТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.27. |
| Фактический результат: соответствует ожидаемому |

Продолжение таблицы 5.5

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 12 | Расчет необходимого количества изделий | 1. Заполнить поля данными:   * Площадь работы: «Пол» * Ширина, м: «3» * Длина, м: «2» * Ширина плитки, мм: «93,8» * Длина плитки, мм: «300» * Запас плитки: «0» * Ширина участка, м: «0,30» * Длина участка, м: не заполняем   2. Нажать на кнопку «ДОБАВАТЬ» для расчета площади не закладываемого участка  3. Нажать на кнопку «РАССЧИТАТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.28. |
| Фактический результат: соответствует ожидаемому |
| 13 | Расчет необходимого количества изделий | 1. Заполнить поля данными:   * Площадь работы: «Пол» * Ширина, м: «3» * Длина, м: «2» * Ширина плитки, мм: «93,8» * Длина плитки, мм: «300» * Запас плитки: «0.0» * Ширина участка, м: «0,30» * Длина участка, м: «0,80»   2. Нажать на кнопку «ДОБАВАТЬ» для расчета площади не закладываемого участка  3. Нажать на кнопку «РАССЧИТАТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.29. |
| Фактический результат: соответствует ожидаемому |

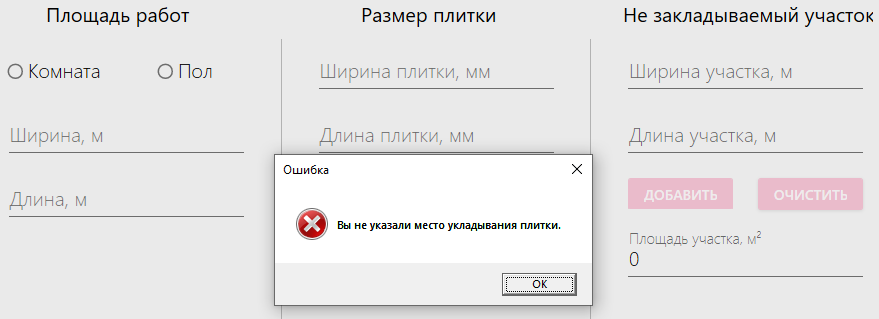


Рисунок 5.26 – Предупреждение о том, что площадь работ не была выбрана

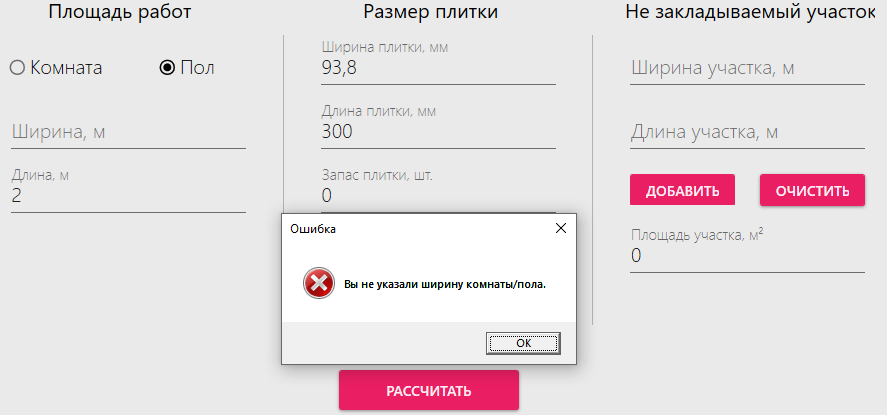


Рисунок 5.27 – Предупреждение о том, что параметр комнаты или пола не был указан

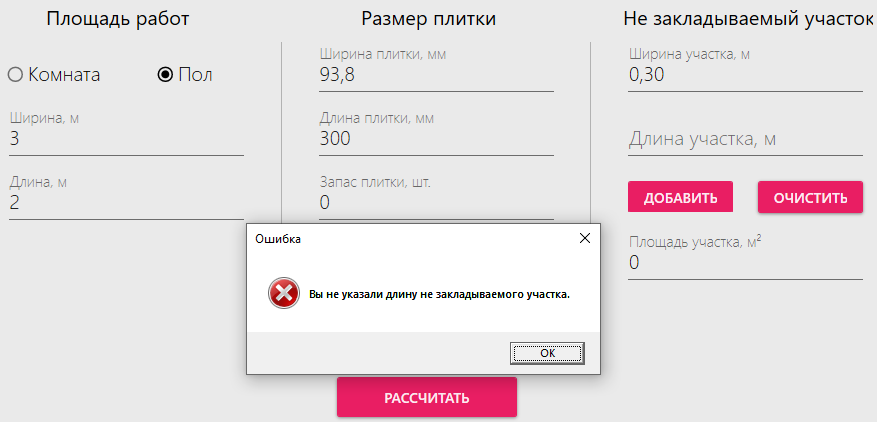


Рисунок 5.28 – Предупреждение о том, что параметр не закладываемого участка не был указан

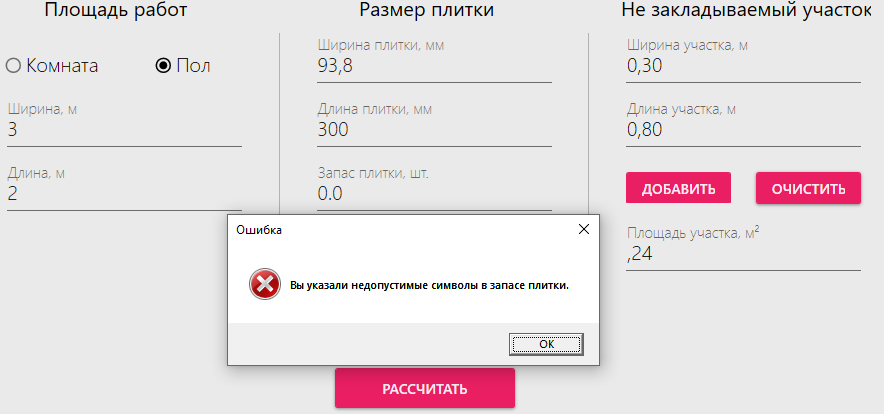


Рисунок 5.29– Предупреждение о том, что в поле указаны недопустимые символы

В случае успешного заполнения всех полей отобразится результат, представленный на рисунке 5.30.

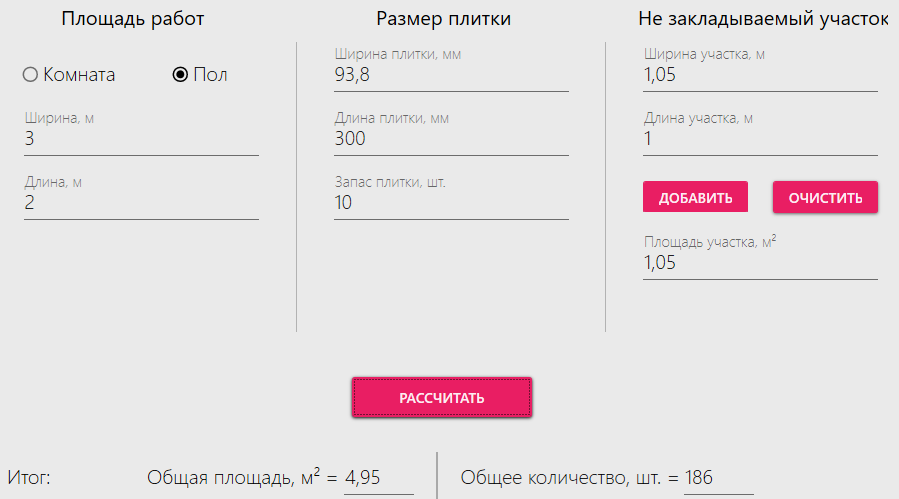


Рисунок 5.30 – Результат расчета необходимого количества изделий

Протестируем функцию добавления изделий в корзину. Для этого нажмем на иконку в левом верхнем углу, затем в боковом меню выберем кнопку «Каталог». После чего откроется окно каталога, представленное на рисунке 5.31.

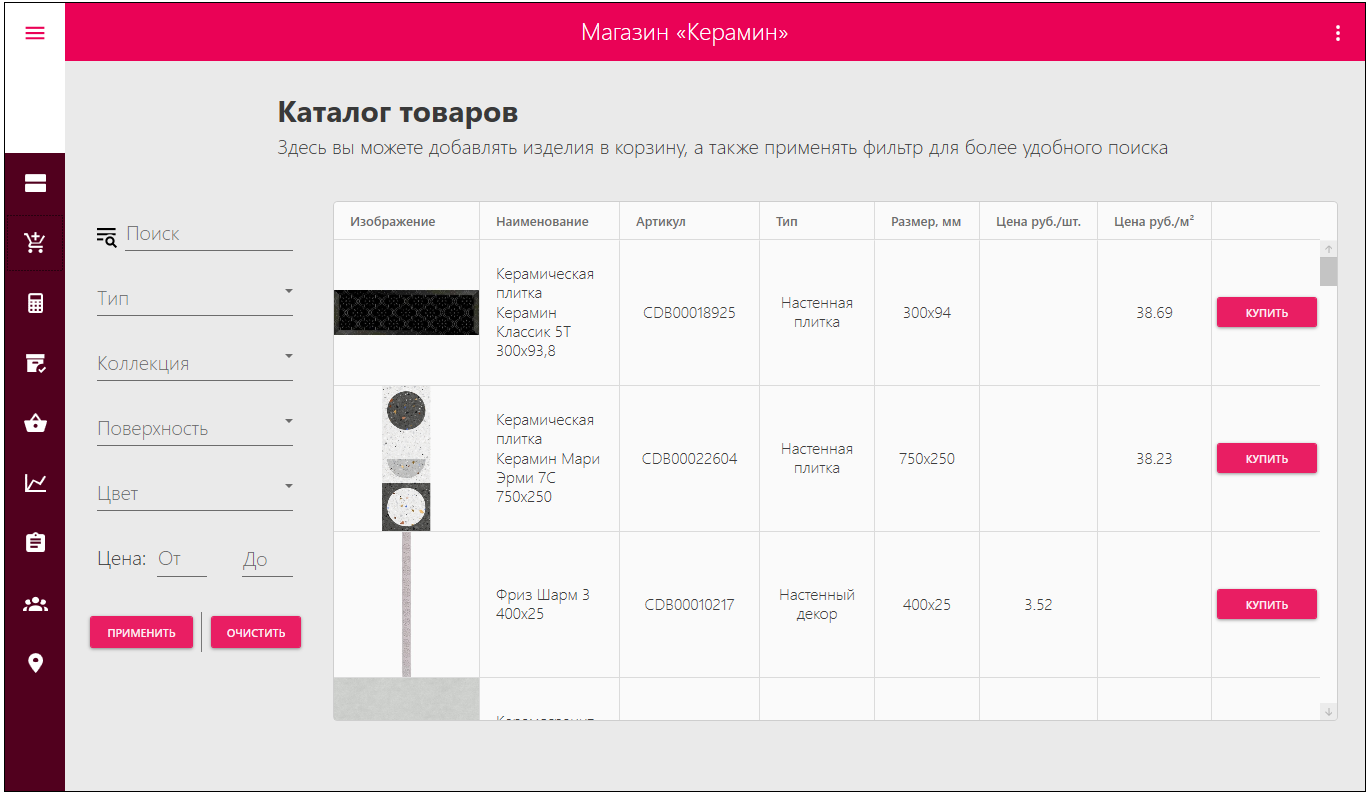


Рисунок 5.31 – Окно каталога изделий

Протестируем функцию добавления изделия в корзину, нажав на кнопку «ДОБАВИТЬ». После этого отобразится окно, где необходимо указать количество изделий для добавления. Окно представлено на рисунке 5.32.

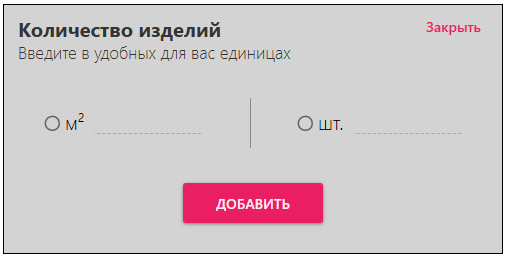


Рисунок 5.32 – Окно указания количества изделий для добавления в корзину

Тест-кейсы для функции указания количества изделий представлены в таблице 5.6.

Таблица 5.6 – Тест-кейсы для функции «Указание количества изделий»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 14 | Указание количества изделий | 1. Выбрать единицу измерения: не выбираем  2. Нажать на кнопку «ДОБАВИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.33. |
| Фактический результат: соответствует ожидаемому |

Продолжение таблицы 5.6

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 15 | Указание количества изделий | 1. Выбрать единицу измерения: «шт.»  2. Заполнить поле данными: не заполняем  3. Нажать на кнопку «ДОБАВИТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.34. |
| Фактический результат: соответствует ожидаемому |
| 16 | Указание количества изделий | 1. Выбрать единицу измерения: «шт.»  2. Заполнить поле данными: «1,5» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.35. |
| Фактический результат: соответствует ожидаемому |

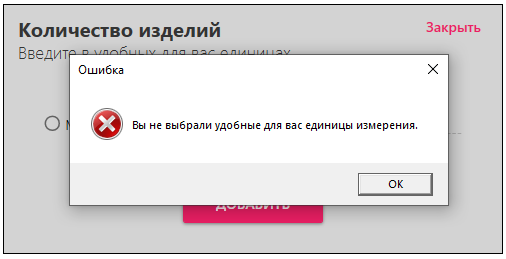


Рисунок 5.33 – Предупреждение о том, что единицы измерения не были выбраны

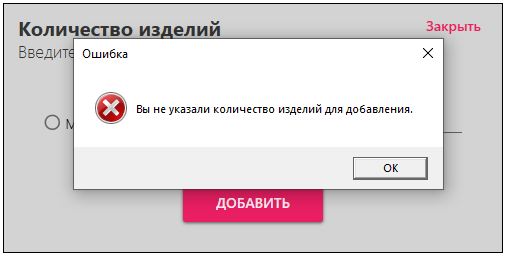


Рисунок 5.34 – Предупреждение о том, что значение поля не указано

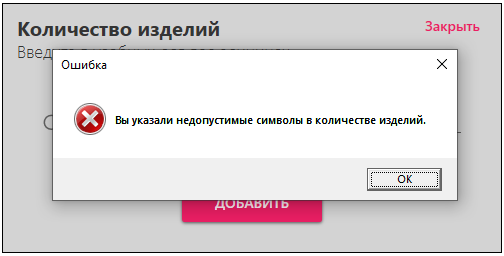


Рисунок 5.35 – Предупреждение о том, что в поле указаны недопустимые символы

После заполнения поля валидными данными, отобразится уведомление, представленное на рисунке 5.36.

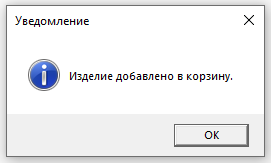


Рисунок 5.36 – Уведомление об успешном добавлении изделия в корзину

Протестируем функцию оформления заказа. Для этого нажмем на иконку в левом верхнем углу, а после чего на кнопку «Корзина». Отобразится окно, представленное на рисунке 5.37.

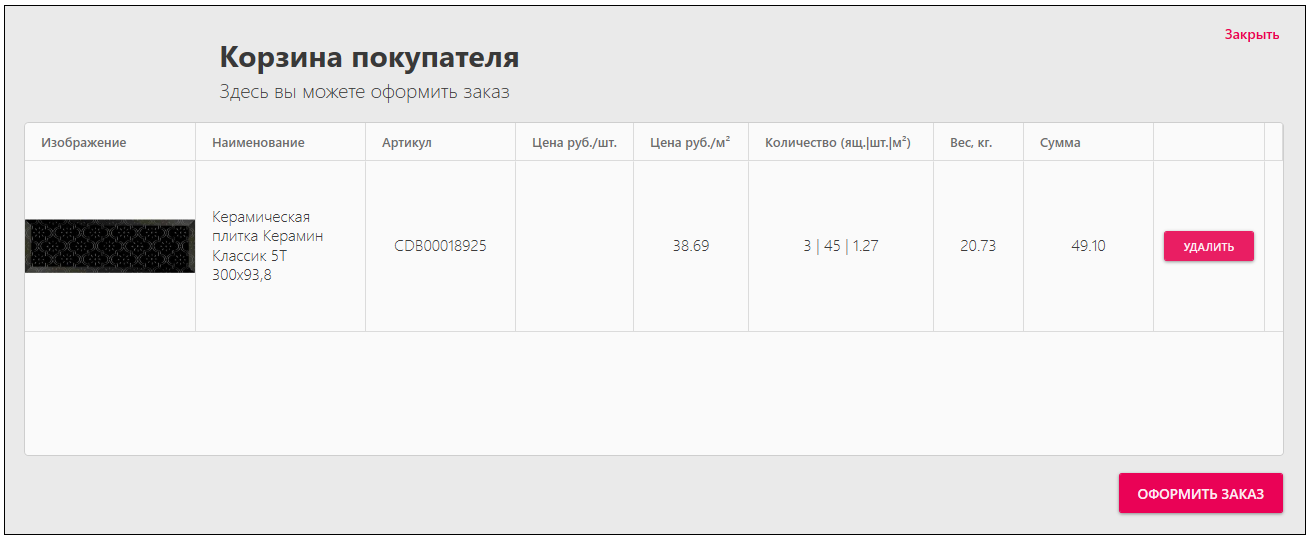


Рисунок 5.37 – Окно корзины покупателя

Для продолжения необходимо нажать на кнопку «ОФОРМИТЬ ЗАКАЗ». В случае, если в корзине не будет товаров, отобразится предупреждение, представленное на рисунке 5.38.

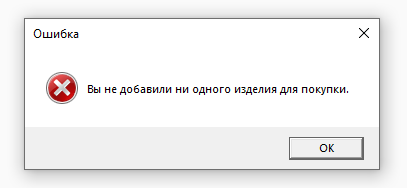


Рисунок 5.38 – Предупреждение о том, что в корзине нет изделий для оформления заказа

После нажатия на кнопку «ОФОРМИТЬ ЗАКАЗ» отобразится та часть окна, которая отвечает за информацию о заказе. Окно представлено на рисунке 5.39.

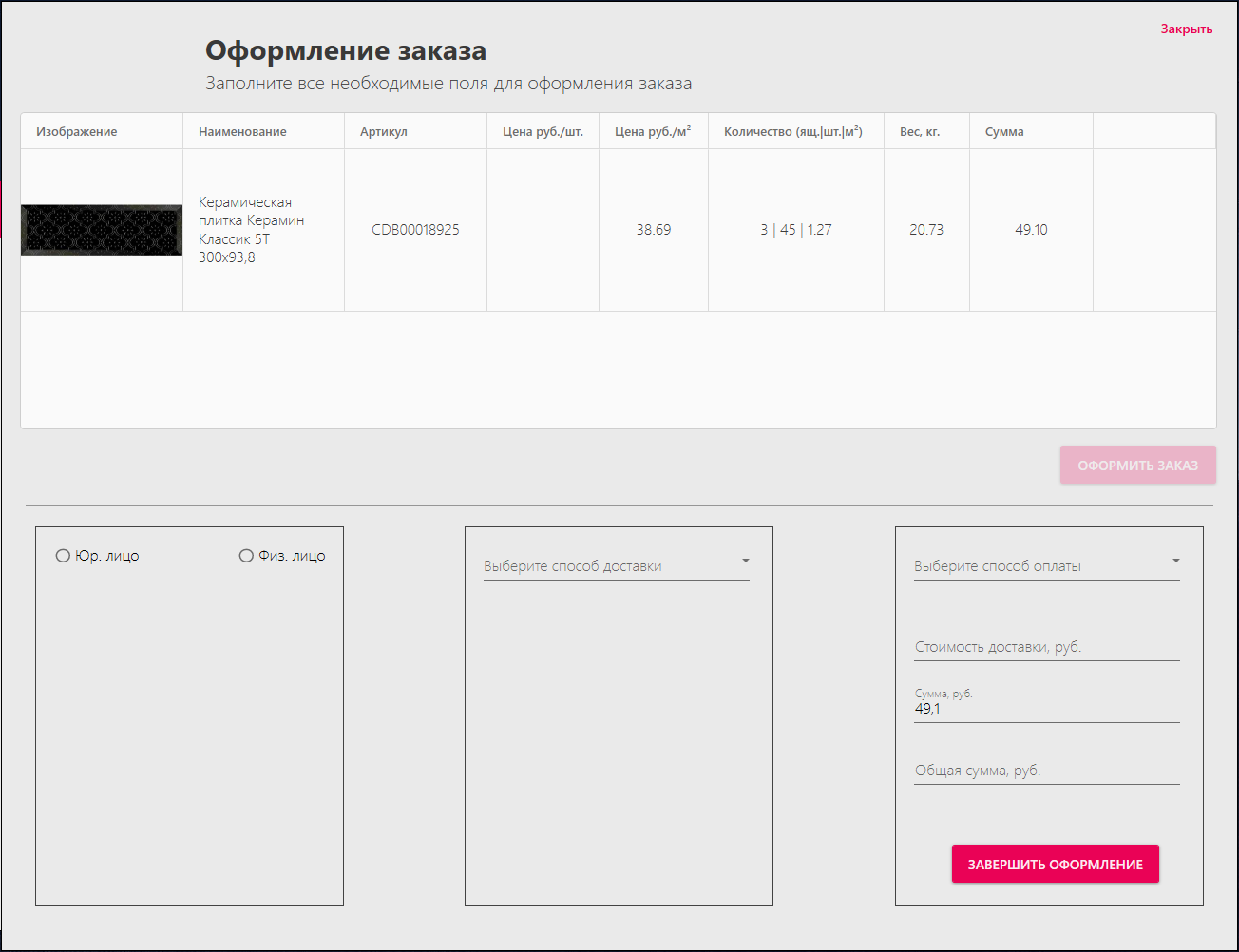


Рисунок 5.39 – Развернутое окно оформления заказа

Тест-кейсы для функции оформления заказа представлены в таблице 5.7.

Таблица 5.7 – Тест-кейсы для функции «Оформление заказа»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | | Результат | |
| 17 | Оформление заказа | 1. Выбрать тип покупателя: не выбираем  2. Выбрать способ доставки: «Самовывоз»  3. Выбрать город: «Минск»  4. Выбрать адрес: «ул. Авакяна, 14»  5. Выбрать способ оплаты: «Наличными»  6. Нажать на кнопку «ЗАВЕРШИТЬ ОФОРМЛЕНИЕ» | | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.40. | |
| Фактический результат: соответствует ожидаемому | |
| 18 | Оформление заказа | | 1. Выбрать тип покупателя: «Юр. лицо»  2. Заполняем поля данными:   * Название организации: не заполняем * ИНН: 123456789 * Контактный телефон: «80(29)444-44-44» * E-mail: «mail@mail.ru»   3. Выбрать способ доставки: «Самовывоз»  4. Выбрать город: «Минск»  5. Выбрать адрес: «ул. Авакяна, 14»  6. Выбрать способ оплаты: «Наличными»  7. Нажать на кнопку «ЗАВЕРШИТЬ ОФОРМЛЕНИЕ» | | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.41. |
| Фактический результат: соответствует ожидаемому |
| 19 | Оформление заказа | | 1. Выбрать тип покупателя: «Юр. лицо»  2. Заполняем поля данными:   * Название организации: «ОАО "Ваша почта"» * ИНН: 123456789 * Контактный телефон: «80(29)444-44-44» * E-mail: «mail@mail.ru!!»   3. Выбрать способ доставки: «Самовывоз»  4. Выбрать город: «Минск»  5. Выбрать адрес: «ул. Авакяна, 14»  6. Выбрать способ оплаты: «Наличными»  7. Нажать на кнопку «ЗАВЕРШИТЬ ОФОРМЛЕНИЕ» | | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.42. |
| Фактический результат: соответствует ожидаемому |

Продолжение таблицы 5.7

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 20 | Оформление заказа | 1. Выбрать тип покупателя: «Юр. лицо»  2. Заполняем поля данными:   * Название организации: «ОАО "Ваша почта"» * ИНН: 123456789 * Контактный телефон: «80(29)444-44-44» * E-mail: «mail@mail.ru»   3. Выбрать способ доставки: «Курьером по г.Минску (С подъемом)»  4. Выбрать город: не выбираем  5. Выбрать улицу: «Алибегова»  6. Заполнить поля данными:   * Номер здания: «3» * Этаж: «4» * Квартира/офис: «45»   7. Выбрать способ оплаты: «Наличными»  8. Нажать на кнопку «ЗАВЕРШИТЬ ОФОРМЛЕНИЕ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.43. |
| Фактический результат: соответствует ожидаемому |

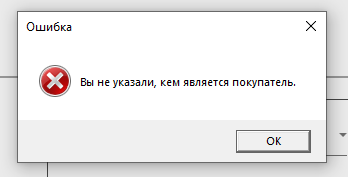


Рисунок 5.40 – Предупреждение о том, что не указан тип покупателя

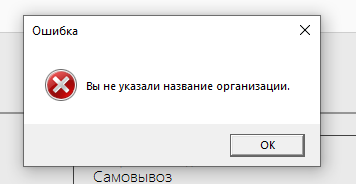


Рисунок 5.41 – Предупреждение о том, что поле не заполнено

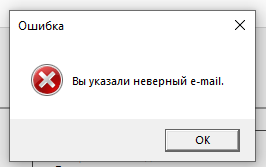


Рисунок 5.42 – Предупреждение о том, что поле с почтой указано неверно

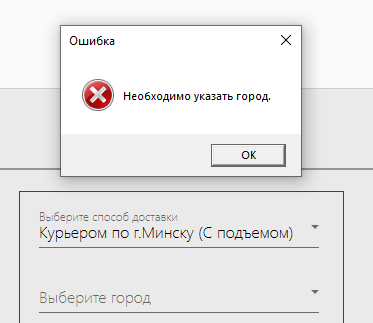


Рисунок 5.43 – Предупреждение о том, что необходимо указать город доставки

После заполнения всех полей валидными данными, отобразится уведомление об успешном оформлении заказа, представленное на рисунке 5.44, а также уведомление о создании чека, представленное на рисунке 5.45.

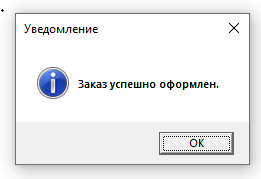


Рисунок 5.44 – Уведомление об успешном оформлении заказа

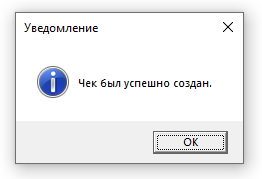


Рисунок 5.45 – Уведомление о создании чека

Протестируем функцию формирования документов по оформленным заказам. Для этого нажмем на иконку в левом верхнем углу, а после чего на кнопку «Список заказов». Отобразится окно, представленное на рисунке 5.46.

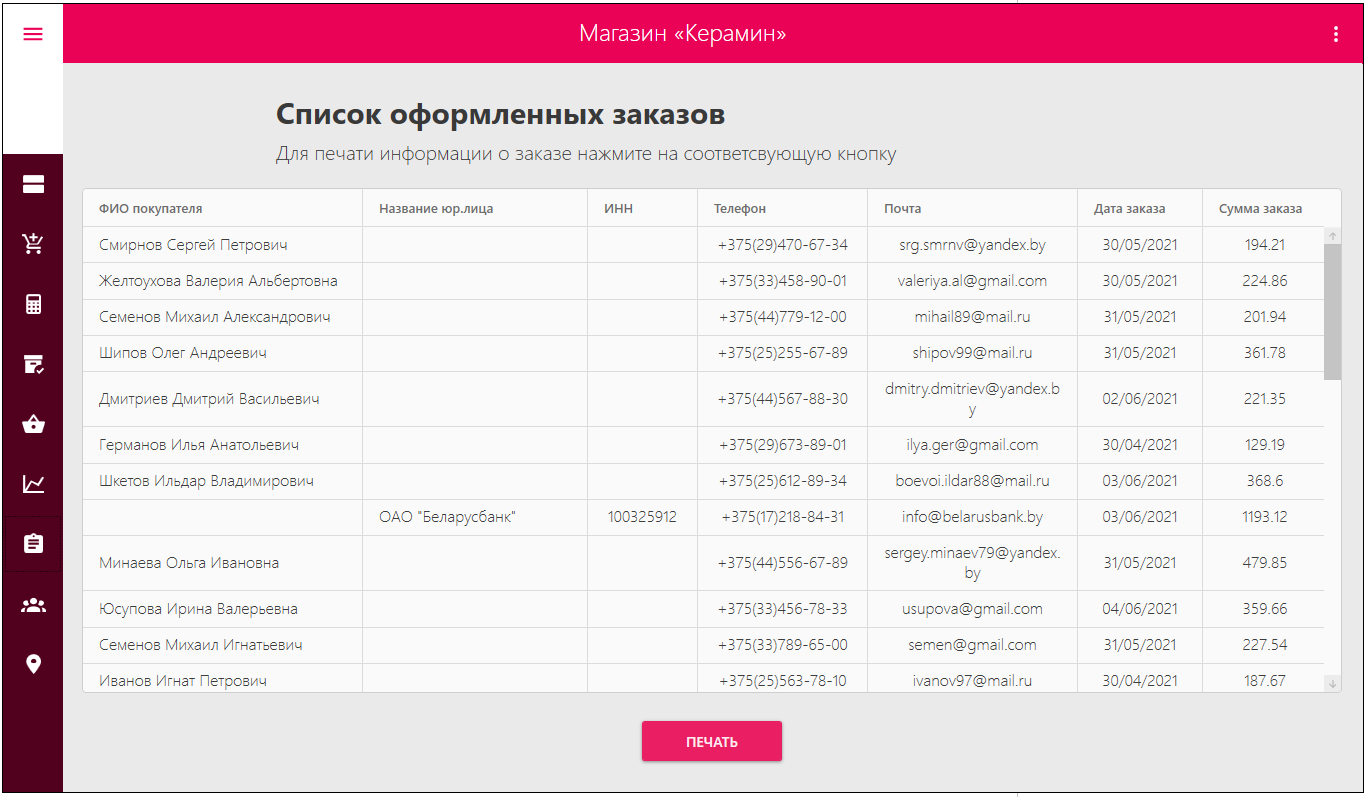


Рисунок 5.46 – Список оформленных заказов

Тест-кейс для функции формирования чеков по оформленным заказам представлен в таблице 5.8.

Таблица 5.8 – Тест-кейс для функции «Формирование чека по оформленному заказу»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 21 | Формирование чека по оформленному заказу | 1. Выбрать заказ: не выбираем  2. Нажать на кнопку «ПЕЧАТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.47. |
| Фактический результат: соответствует ожидаемому |

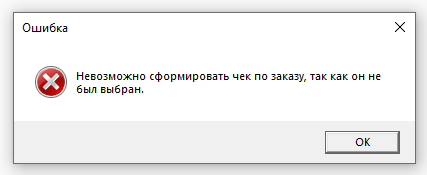


Рисунок 5.47 – Предупреждение, связанное с тем, что заказ не был выбран

После выбора заказа из таблицы, отобразится уведомление об успешном формировании чека, представленное на рисунке 5.45.

Протестируем функцию просмотра статистики о продуктивности сотрудников. Для этого нажмем на иконку в левом верхнем углу, а после чего на кнопку «Статистика». После отобразится контекстное меню, в котором необходимо нажать на пункт «Лучшие сотрудники». Затем откроется окно, представленное на рисунке 5.48.

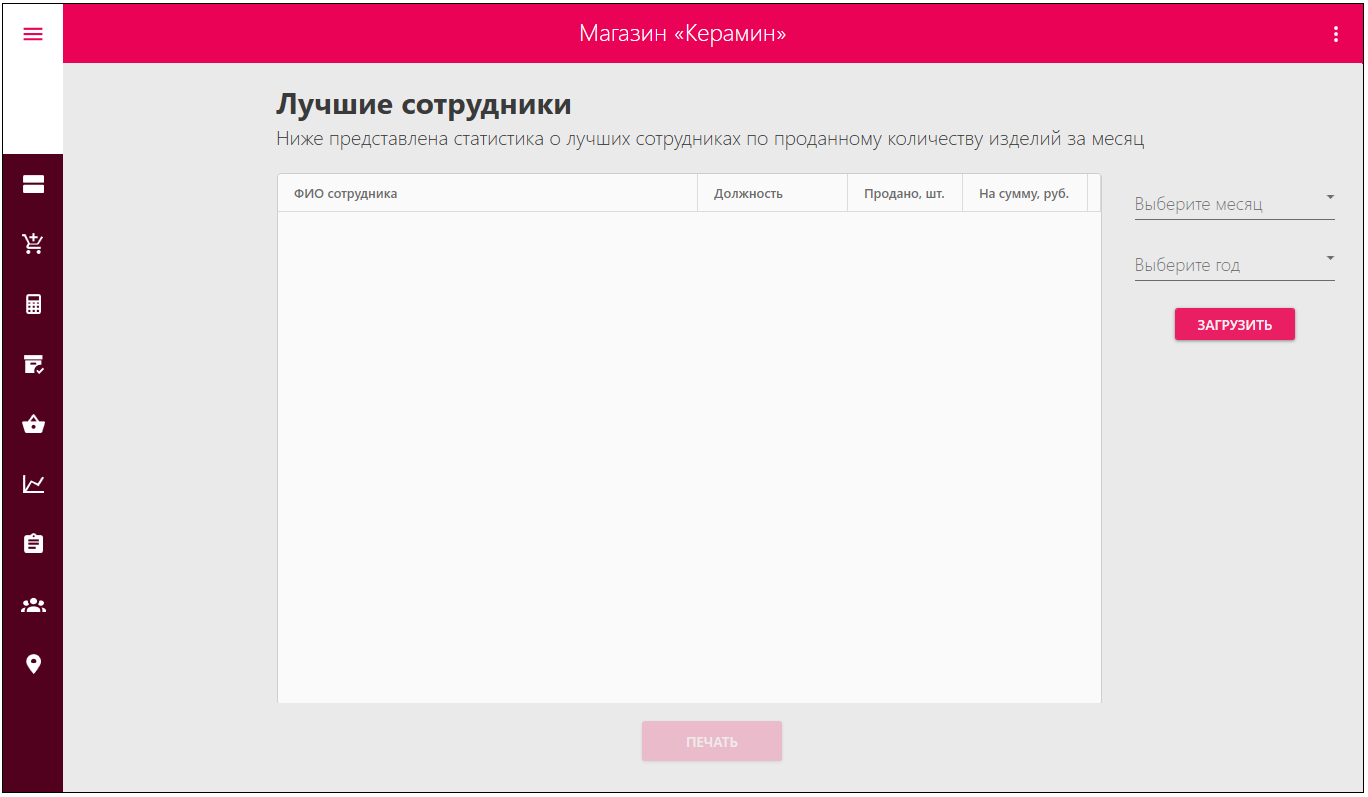


Рисунок 5.48 – Окно со статистикой о продуктивности сотрудников

Тест-кейсы для функции просмотра статистики о продуктивности сотрудников представлены в таблице 5.9.

Таблица 5.9 – Тест-кейсы для функции «Просмотр статистики о продуктивности сотрудников»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 22 | Просмотр статистики о продуктивности сотрудников | 1. Выбрать месяц: не выбираем  2. Выбрать год: «2021»  3. Нажать на кнопку: «Загрузить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.49. |
| Фактический результат: соответствует ожидаемому |

Продолжение таблицы 5.9

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 23 | Просмотр статистики о продуктивности сотрудников | 1. Выбрать месяц: «Май»  2. Выбрать год: не выбираем  3. Нажать на кнопку: «Загрузить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.50. |
| Фактический результат: соответствует ожидаемому |
| 24 | Просмотр статистики о продуктивности сотрудников | 1. Выбрать месяц: «Май»  2. Выбрать год: «2020»  3. Нажать на кнопку: «Загрузить»  4. Нажать на кнопку: «ПЕЧАТЬ» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.51. |
| Фактический результат: соответствует ожидаемому |

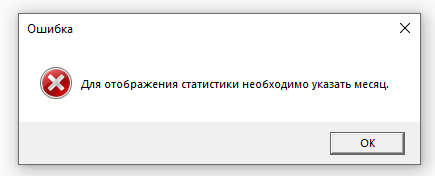


Рисунок 5.49 – Предупреждение о том, что месяц не был указан

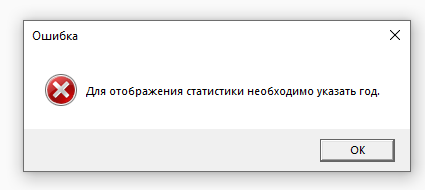


Рисунок 5.50 – Предупреждение о том, что год не был указан

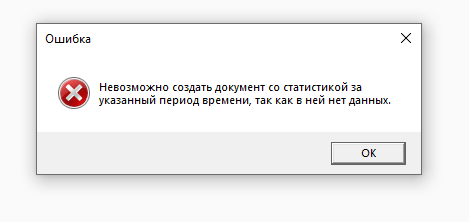


Рисунок 5.51 – Предупреждение о том, что в таблице нет данных

После отображаения информации в таблице, представленной на рисунке 5.52, и нажатия на кнопку «ПЕЧАТЬ», сформируется документ, представленный на рисунке Б.3.

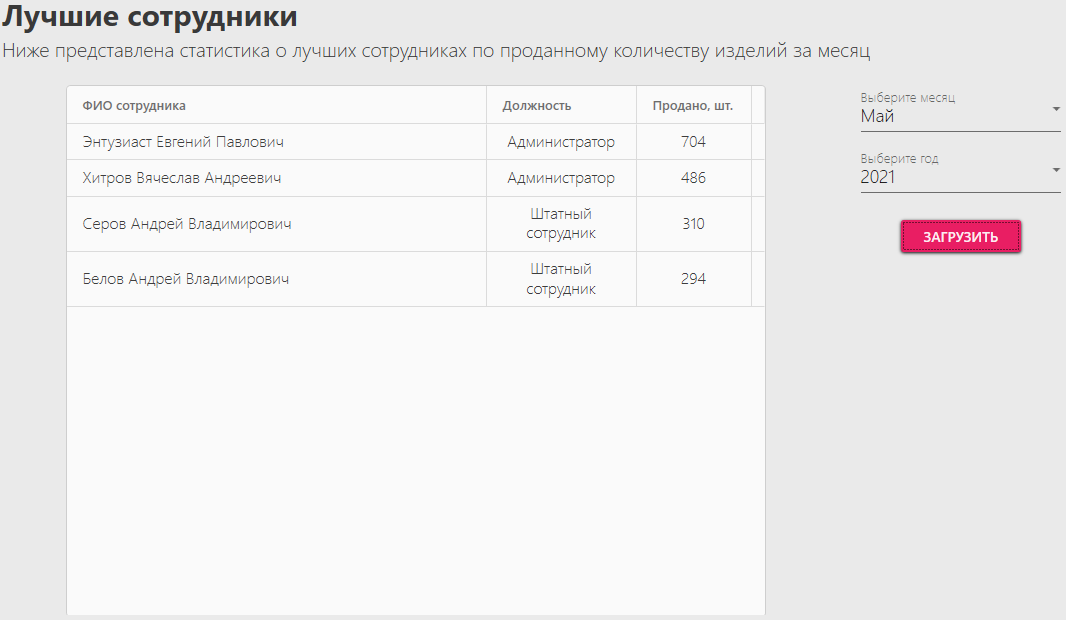


Рисунок 5.52 – Таблица с лучшими сотрудниками за указанный период

Функция просмотра статистики о популярности изделий имеет такие же проверки, что и для просмотра статистики о продуктивности сотрудников.

Протестируем функцию добавления нового пункта самовывоза. Для этого нажмем на иконку в левом верхнем углу, а после чего на кнопку «Пункты самовывоза». После отобразится контекстное меню, в котором необходимо нажать на пункт «Добавить новый пункт самовывоза». Затем откроется окно, представленное на рисунке 5.53.

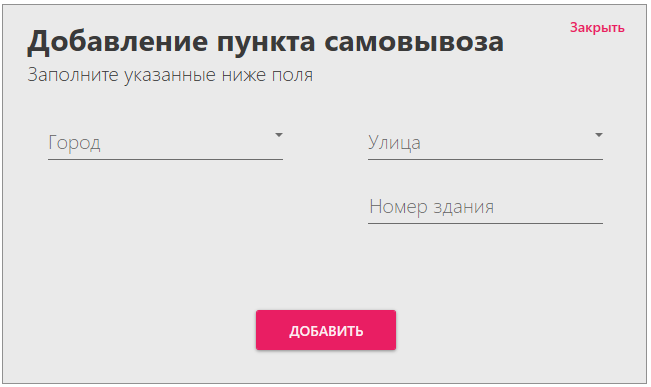


Рисунок 5.53 – Окно добавления нового пункта самовывоза

Тест-кейсы для функции добавления нового пункта самовывоза представлены в таблице 5.10.

Таблица 5.10 – Тест-кейсы для функции «Добавление нового пункта самовывоза»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Модуль/функция | Шаги воспроизведения | Результат |
| 25 | Добавление нового пункта самовывоза | 1. Выбрать город: не выбираем  2. Выбрать улицу: «Авакяна»  3. Ввести номер здания: «14»  4. Нажать на кнопку «Добавить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.54. |
| Фактический результат: соответствует ожидаемому |
| 26 | Добавление нового пункта самовывоза | 1. Выбрать город: «Минск»  2. Выбрать улицу: «Авакяна»  3. Ввести номер здания: «14!»  4. Нажать на кнопку «Добавить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.55. |
| Фактический результат: соответствует ожидаемому |
| 27 | Добавление нового пункта самовывоза | 1. Выбрать город: «Минск»  2. Выбрать улицу: «Авакяна»  3. Ввести номер здания: «14»  4. Нажать на кнопку «Добавить» | Ожидаемый результат: отображение предупреждения, представленного на рисунке 5.56. |
| Фактический результат: соответствует ожидаемому |

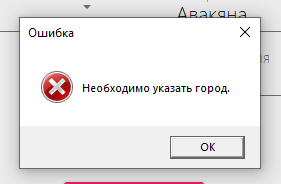


Рисунок 5.54 – Предупреждение о том, что значение в поле не указано

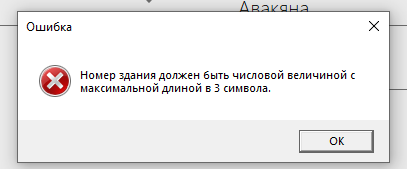


Рисунок 5.55 – Предупреждение о том, что значение в поле с номером здания указано неверно

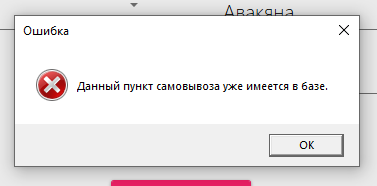


Рисунок 5.56 – Предупреждение о том, что пункт самовывоза с указанными данными уже имеется в базе

После заполнения всех полей и в случае, если данного пункта самовывоза не было в базе, отобразится уведомление, представленное на рисунке 5.57.

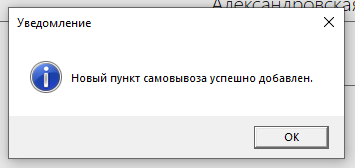


Рисунок 5.57 – Уведомление об успешном добавлении нового пункта самовывоза

Нажмем на иконку в левом верхнем углу, после чего, в открывшемся боковом меню, нажмем на кнопку «Список пунктов самовывоза». Отобразится контекстное меню, где необходимо нажать на пункт «Список пунктов самовывоза». Затем откроется окно со списком пунктов самовывоза, представленное на рисунке 5.58.

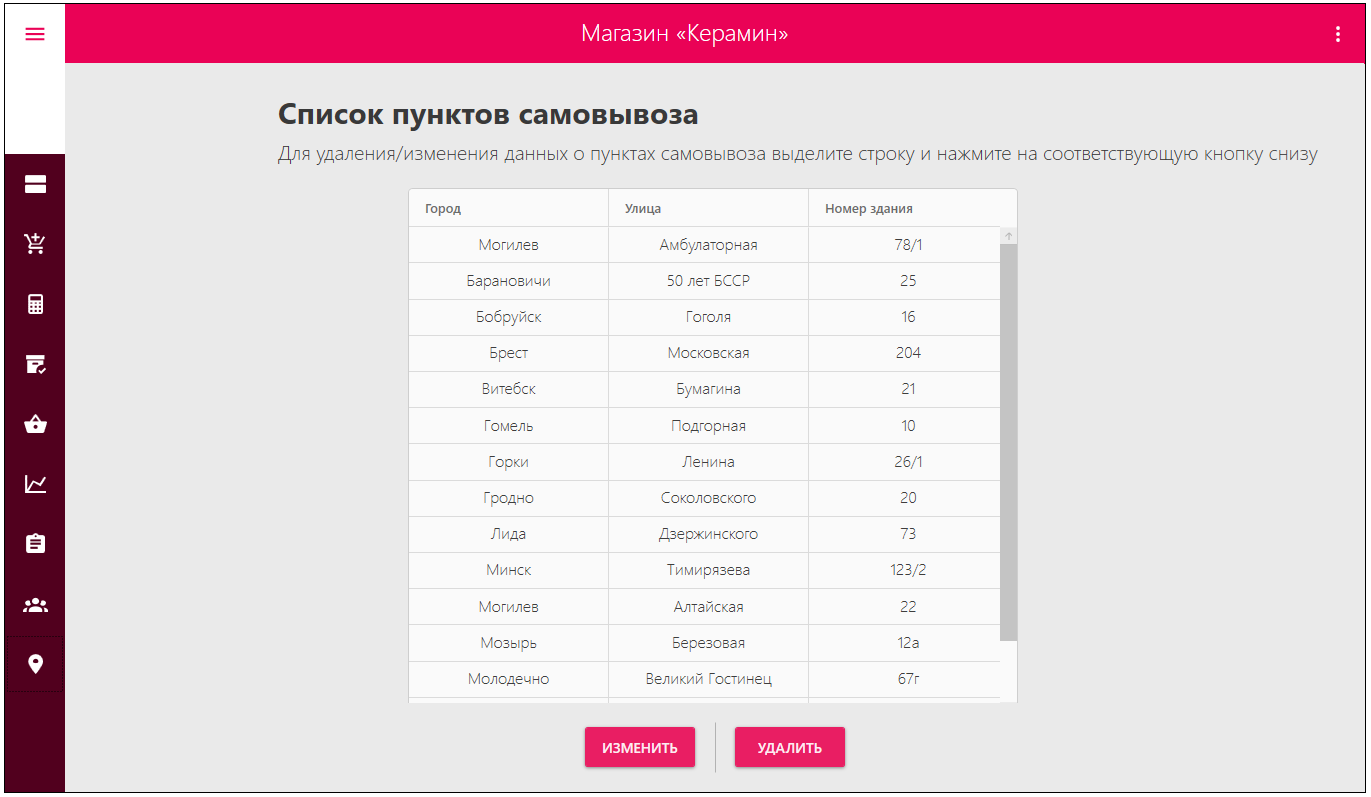


Рисунок 5.58 – Окно со списком пунктов самовывоза

Протестируем функцию изменения данных пунктов самовывоза. Для этого, выделив необходимую строку среди списка, нажмем на кнопку «ИЗМЕНИТЬ». В случае, если не выделили запись и нажали на кнопку, то отобразится предупреждение, представленное на рисунке 5.28, только для пункта самовывоза. Также, при выборе нескольких записей, окно изменения данных пункта самовывоза, представленное на рисунке 5.59, откроется только для самой первой записи среди выделенных.

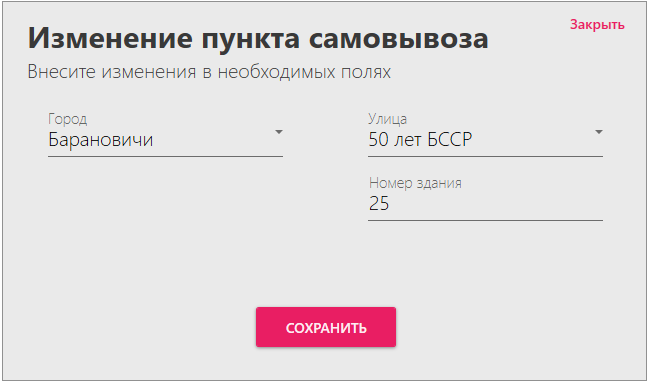


Рисунок 5.59 – Окно изменения данных выбранного пункта самовывоза

Все проверки данных идентичны тем, что используются при добавлении нового пункта самовывоза.

Протестируем функцию удаления изделий. Для этого нажмем на кнопку «Закрыть», находясь в окне изменения данных пункта самовывоза, после чего вновь откроется окно со списком всех изделий. Выберем в списке необходимый для удаления пункт самовывоза и нажмем на кнопку «УДАЛИТЬ». После этого отобразится уведомление об успешном удалении пункта самовывоза, представленное на рисунке 5.15. В случае, если пункт самовывоза не был выбран, то отобразится предупреждение, представленное на рисунке 5.16, только для пункта самовывоза.

# Применение

## Назначение программы

Данное программное средство разработано для сотрудников магазина ОАО «Керамин».

Основной целью программного средства является обеспечение удобной работы с данными об изделиях, то есть осуществление поиска по артикулу и фильтрации этих данных, оформление заказов, а также с данными о сотрудниках и пунктах самовывоза, просмотр списка оформленных заказов и статистики, а также создание документов на их основании. В том числе повышение работоспособности сотрудников, а также экономия времени на создание чеков по сформированным заказам.

База данных проекта содержит огромный объем необходимой информации для реализации поставленных задач.

Предоставлена возможность быстрого добавления новых изделий, сотрудников, пунктов самовывоза, изменения данных об этих изделиях, сотрудниках и пунктах самовывоза, подсчет необходимого количества керамических изделий для реализации, оформление заказов, формирование статистики по популярности изделий и продуктивности сотрудников, создание документов по данным статистикам и заказам.

## Условие применения

Для бесперебойной работы данного программного средства необходимы следующие технические требования:

* процессор Intel Core i5 и новее;
* оперативная память 8 ГБ и больше;
* жесткий диск с минимальным объёмом памяти – 128 ГБ;
* операционная система Windows 8 и новее;
* платформа Microsoft .Net Framework 4.7.2;
* клавиатура;
* мышь;
* программа для работы с таблицами Microsoft Excel 2016;
* система управления базами данных Microsoft SQL Server 2017.

## Справочная система

Справочная система программного средства представляет собой отдельный файл «help.chm» с полным описанием основных функций программы в формате \*.chm. В справочной системе даны ответы на вопросы, которые могут возникнуть при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справочная система содержит следующие разделы:

* «1. О программе»;
* «2. Начало работы»;
* «3. Руководство администратора»;
* «4. Руководство сотрудника»;
* «5. О разработчике».

Структура справочной системы представлена на рисунке 6.1.

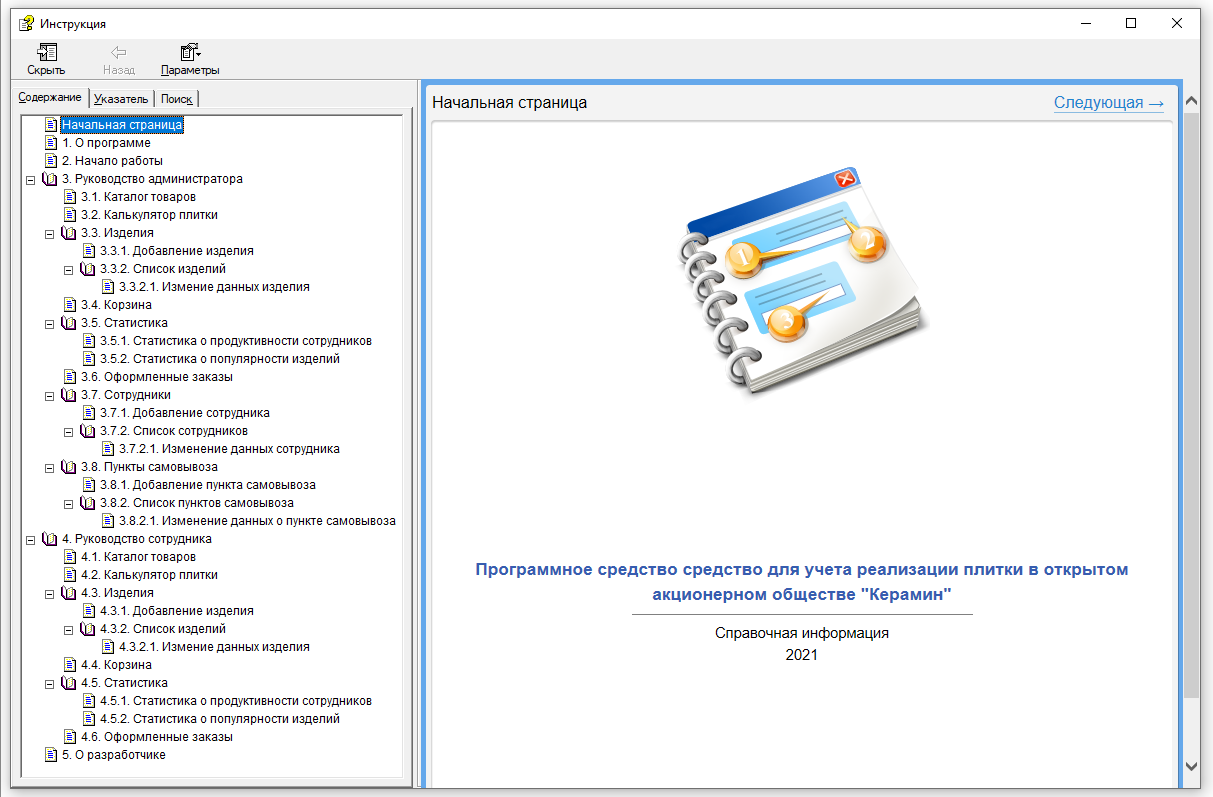


Рисунок 6.1 – Структура справочной системы

В разделе «1. О программе» представляется информация о программном средстве, а именно: его предназначение, цели его создания, описание его возможностей. Данный раздел представлен на рисунке 6.2.

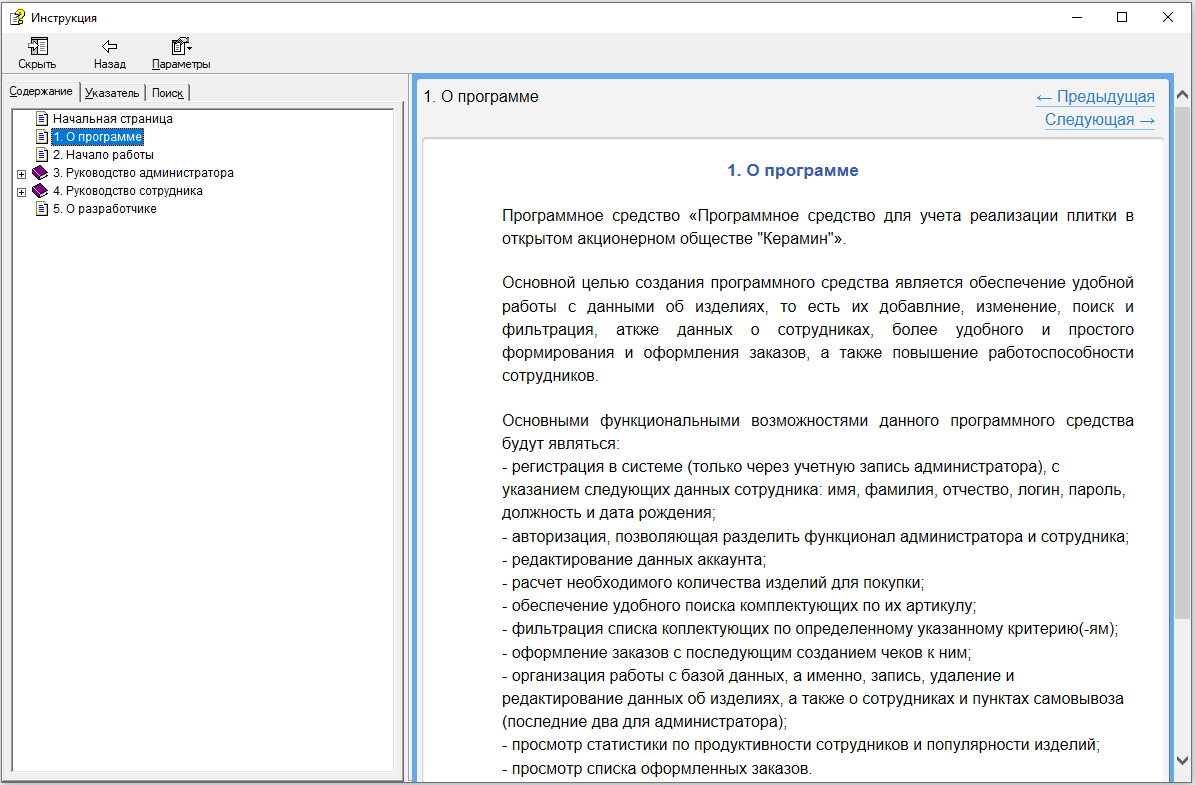


Рисунок 6.2 – Содержание раздела «1. О программе»

В разделе «2. Начало работы» содержится информация о том, как запустить приложение, об окне, которое появится после запуска программного средства, а также описание элементов, находящихся в нем. Данный раздел представлен на рисунке 6.3.

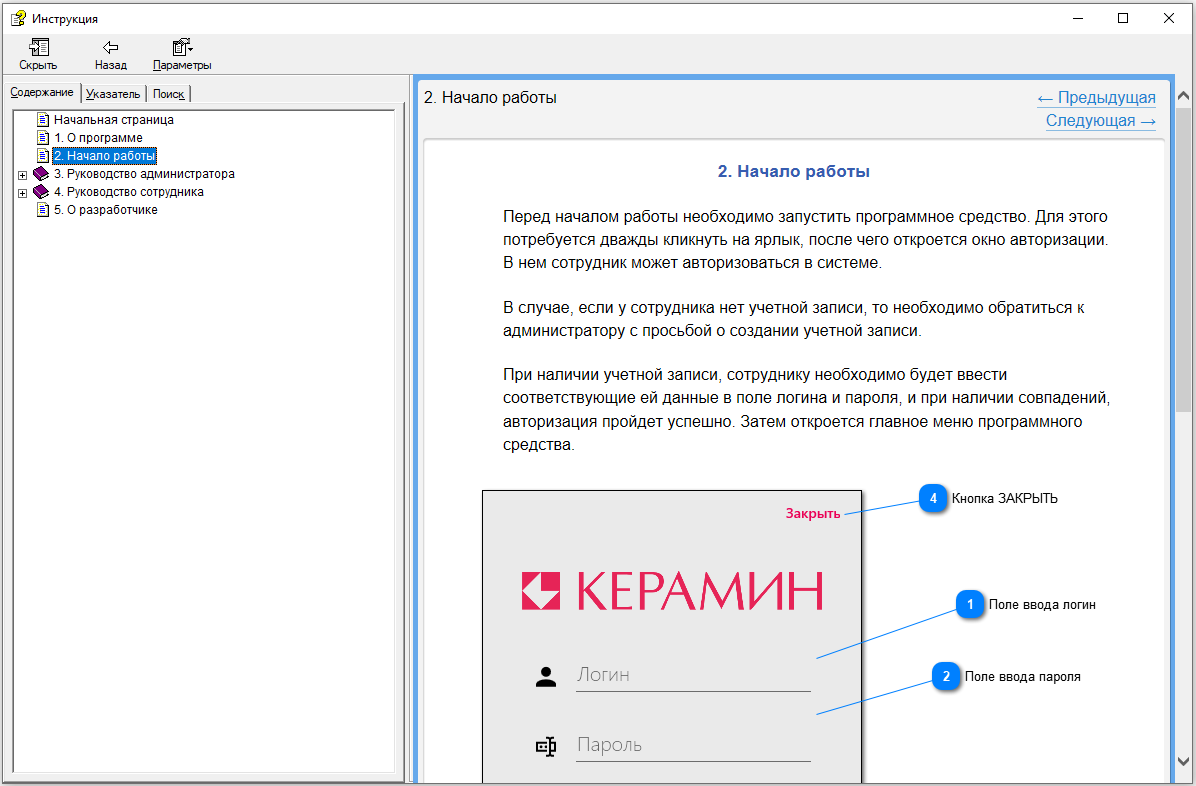


Рисунок 6.3 – Содержание раздела «2. Начало работы»

В разделе «3. Руководство администратора» представлена информация обо всех возможностях администратора, в том числе и описание всех кнопок, полей и окон, с которыми взаимодействует администратор. Данный раздел представлен на рисунке 6.4.

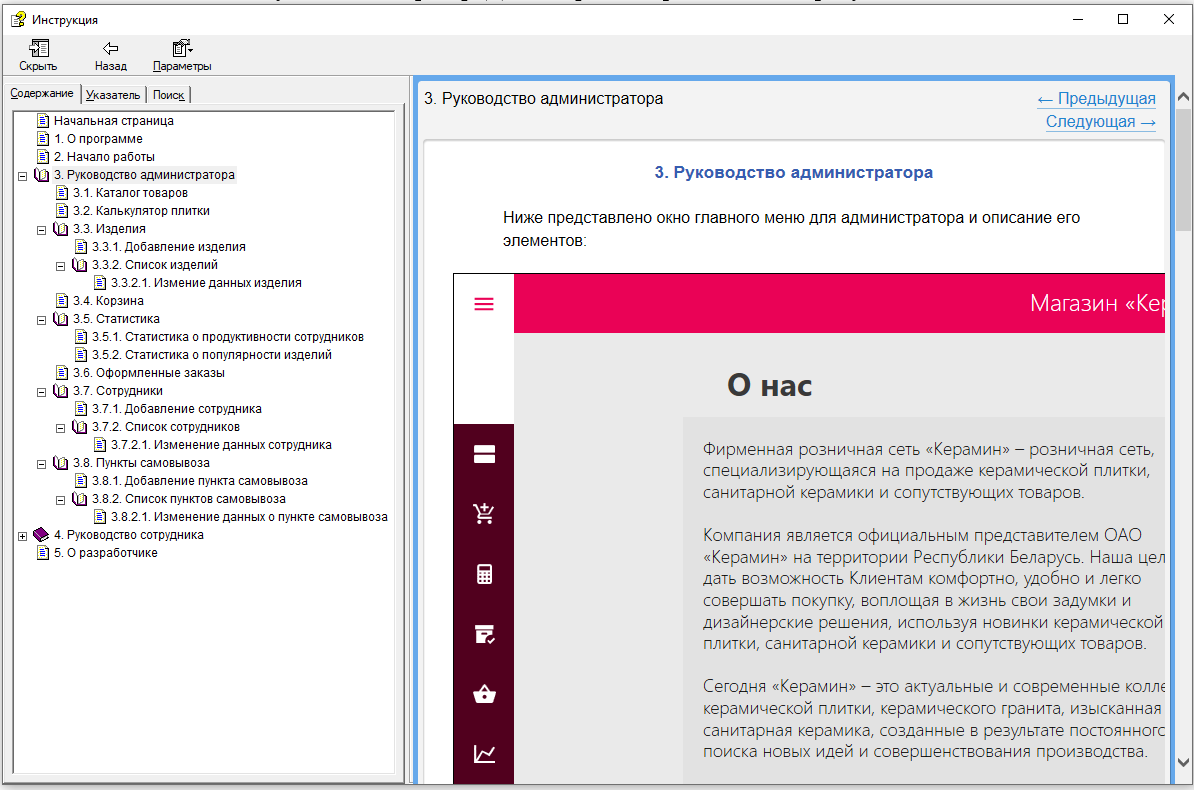


Рисунок 6.4 – Содержание раздела «3. Руководство администратора»

В разделе «4. Руководство сотрудника» содержится информация, а также описание всех кнопок, окон и полей, которые доступны для взаимодействия сотруднику. Данный раздел представлен на рисунке 6.5.

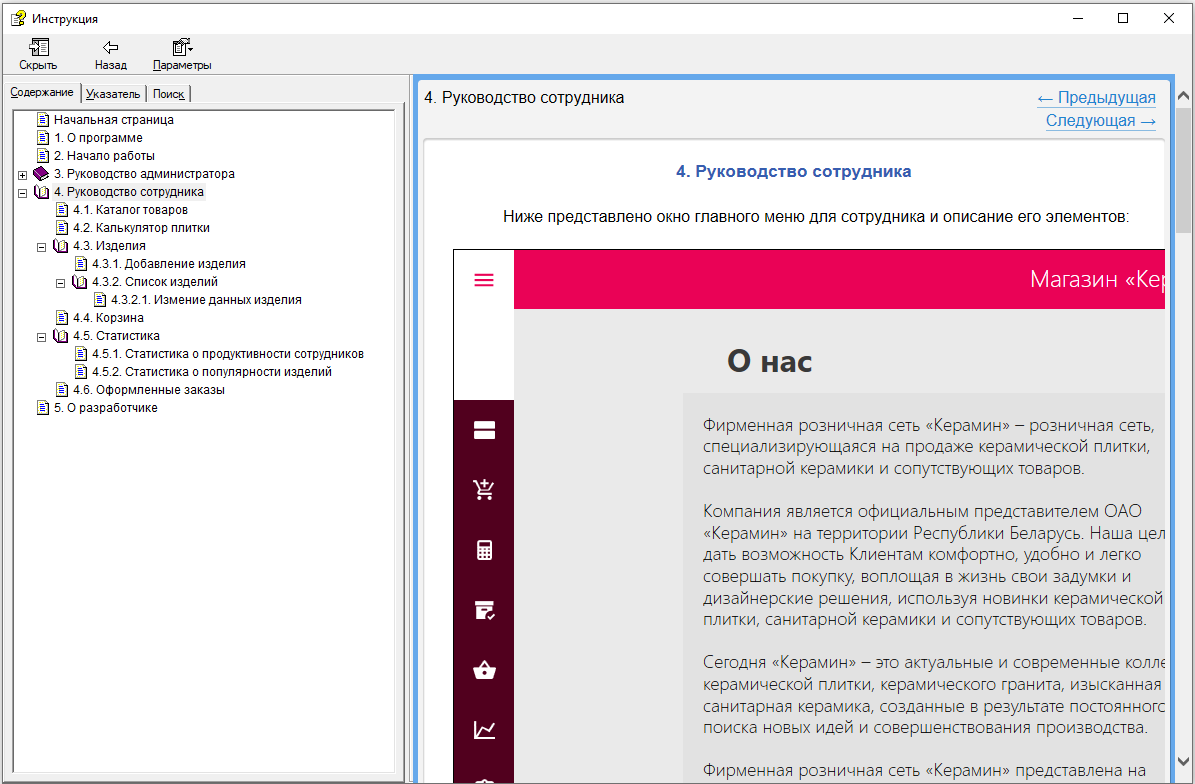


Рисунок 6.5 – Содержание раздела «4. Руководство сотрудника»

В разделе «5. О разработчике» представлена информация о разработчике. Данный раздел представлен на рисунке 6.6.

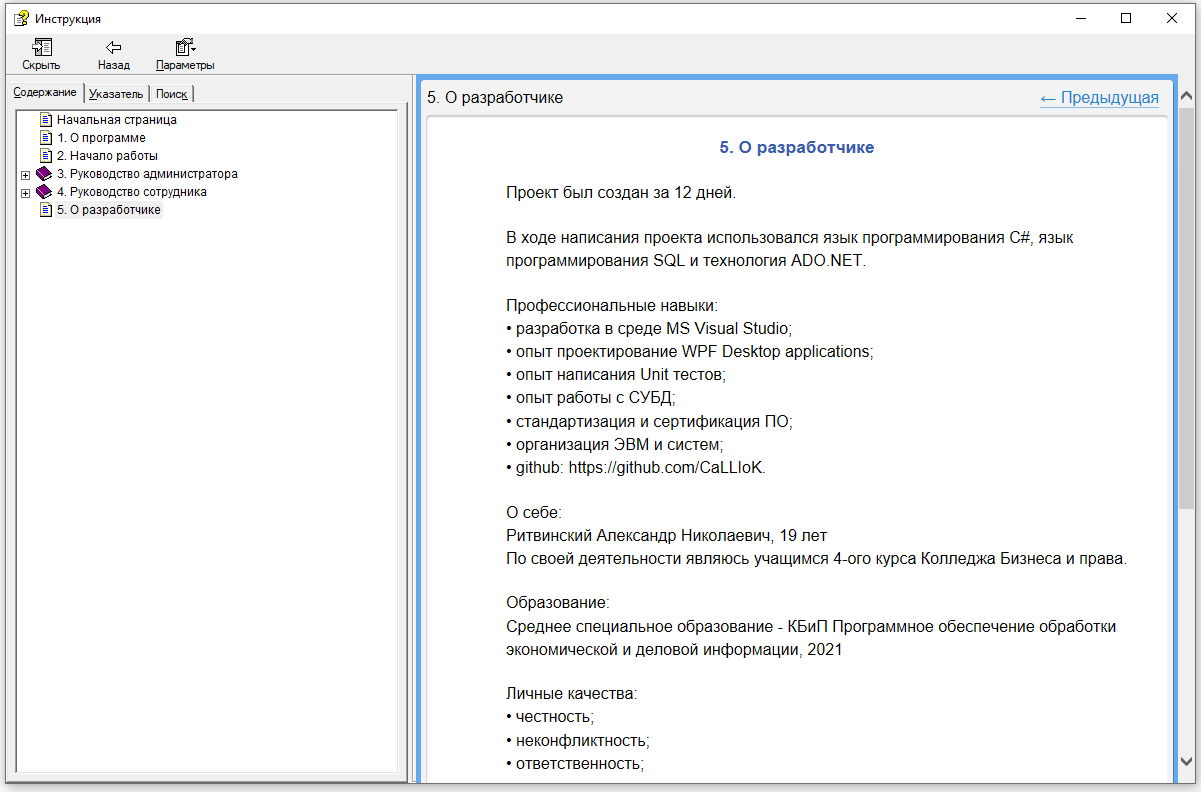


Рисунок 6.6 – Содержание раздела «5. О разработчике»

# Охрана труда и окружающей среды

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

Охрана труда – это система обеспечения безопасности жизни и здоровья работников в процессе трудовой деятельности, включающая правовые, социально-экономические, организационные, санитарногигиенические, психофизические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

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

Основными задачами службы по охране труда являются:

* координация деятельности по обеспечению здоровых и безопасных условий труда;
* совершенствование СУОТ;
* внедрение передового опыта и научных разработок по безопасности и гигиене труда, пропаганда охраны труда;
* осуществление контроля за соблюдением требований актов законодательства, технических нормативных правовых актов и локальных нормативных правовых актов, содержащих требования охраны труда.

Управление охраной труда является составной частью общей системы управления предприятием. Оно направлено на улучшение сложившихся условий труда, повышение их безопасности.

Для организации работы и осуществления контроля по охране труда ОАО «Керамин» в установленном законодательством порядке создает службу охраны труда.

Ответственность за организацию своевременного и качественного проведения обучения, инструктажа, стажировки и проверки знаний работников по вопросам охраны труда в предприятии возлагается на начальника отдела охраны труда и окружающей среды.

На охрану труда предприятием было выделено в 2015 году 283 тыс. руб., в том числе на одного работающего 0,92 тыс. руб.; в 2016 году – 260 тыс. руб., в том числе на одного работающего 0,77 тыс. руб.; в 2017 году – 30 тыс. руб., в том числе на одного работающего 0,08 тыс. руб. В 2018 году на охрану труда предприятие выделено 66 тыс. руб.

Цель государственной политики в области охраны труда – сохранение жизни и здоровья граждан в процессе трудовой деятельности.

Основными принципами государственной политики в области охраны труда являются:

– приоритет жизни и здоровья по отношению к результатам производственной деятельности;

– обеспечение гарантий права работников на охрану труда;

– установление обязанностей всех субъектов правоотношений в области охраны труда, полной ответственности нанимателей за обеспечение здоровых и безопасных условий труда.

– совершенствование правоотношений и управления в этой сфере, включая внедрение экономического механизма обеспечения охраны труда.

Охрана труда выявляет и изучает возможные причины производственных несчастных случаев, профессиональных заболеваний, аварий, взрывов, пожаров и разрабатывают систему мероприятий и требований с целью устранения этих причин и создания безопасных и благоприятных для человека условий труда.

При работе в неблагоприятных для здоровья условиях снижается производительность труда, происходят потери рабочего времени из-за травматизма, профессиональных и производственно-обусловленных заболеваний, текучести кадров, что отрицательно сказывается на объеме выпуска продукции, ее качестве, использовании оборудования. Кроме того, увеличивается численность работающих, имеющих право на получение льгот и компенсаций, растут затраты на возмещение последствий воздействия неудовлетворительных и опасных условий труда.

Нарушение инструкций, правил и норм техники безопасности, пожарной безопасности и производственной санитарии могут привести к травматизму и профессиональным заболеваниям.

Опасный производственный фактор – производственный фактор, воздействие которого на работающего в определенных условиях приводит к травме или другому внезапному резкому ухудшения здоровья.

Травма (несчастный случай) на производстве обычно бывает следствием внезапного воздействия на работника какого-либо опасного производственного фактора при выполнении им трудовых обязанностей или заданий руководителя работ. Травмы подразделяются на: механические (ушибы, переломы, раны и др.); тепловые (ожоги, обморожения, тепловые удары); химические (химические ожоги, острые отравления); электрические и др.

К несчастным случаям на производстве, подлежащим расследованию, оформлению и учету в соответствии с Положением, относятся несчастные случаи, независимо от причин, происшедшие в течение рабочего времени; во время установленных перерывов, в периоды времени до начала и после окончания работ, при выполнении работ в сверхурочное время в выходные и праздничные дни:

– на территории нанимателя или в ином месте работы, в том числе в командировке, а также в любом другом месте, где потерпевший находился в связи с работой или совершал действия в интересах нанимателя;

– на транспорте нанимателя, сторонней организации; предоставившей его нанимателю, согласно договору (заявки);

– на личном транспорте, используемом в интересах нанимателя;

– общественном или ином транспорте, а также при следовании пешком;

– с работником, чья деятельность связана с передвижением между объектами обслуживания;

– при выполнении работ по ликвидации чрезвычайных ситуаций природного и техногенного характера их последствий;

– при выполнении общественных работ, организуемых исполнительными и распорядительными органами совместно с хозяйствующими субъектами и службами

В целом анализ причин заболеваемости и травматизма показывает, что происходят они по нескольким причинам: из-за нарушения трудовой и производственной дисциплины и технологических процессов на производственных работах (несоблюдение правил техники безопасности, регулировки, наладки и чистки станков до полной остановки их вращающихся и движущихся частей) и по причине личной недисциплинированности отдельных работников.

В целях снижения производственного травматизма и дальнейшего улучшения необходимо:

* усиление административно-общественного контроля;
* проведение постоянной работы по подготовке и повышению квалификации рабочих;
* осуществление качественной подготовки рабочих мест;
* соблюдение санитарно-гигиенических требований в цехах;
* совершенствование безопасных приёмов работы и производственной санитарии;
* регулярное проведение анализа причин, вызывающих производственный травматизм и заболеваемость, разработку и осуществление конкретных мероприятий по их устранению;
* учёт состояния охраны труда при премировании;
* воспитание у работников дисциплинированности.

Показатели производственного травматизма носят не только социальный характер, но также и экономический, поскольку, в соответствии с законодательством о труде, наниматель несет ответственность за вред, причиненный работникам увечьем, профессиональным заболеванием либо иным повреждением здоровья, связанными с исполнением ими своих трудовых обязанностей и произошедшими как на территории нанимателя, так и за ее пределами, а также во время следования работника к месту работы или с работы на транспорте, предоставленном нанимателем.

Наниматель обязан возместить в полном объёме вред, причиненный работнику трудовым увечьем при исполнении им трудовых обязанностей. Однако в данном случае следует отметить, что травмы, полученные в результате опьянения, при хищении материальных ценностей, изготовлении каких-либо предметов для личных целей и без разрешения администрации, не считаются производственными и ущерб работнику в результате таких несчастных случаев не возмещается.

Охрана труда в ОАО «Керамин» проводится на основании перечня основных обязательных документов предприятия, ведение которых предписано действующим законодательством о безопасности и гигиене труда, ремонтной и эксплуатационной документации. Таким образом, ОАО «Керамин» производит следующие виды работ по охране труда и отчитывается о выполнении по указанным документам:

* обучение, инструктажи, проверка знаний по охране труда (приказ о создании комиссии о проверке знаний соответствующих работников по охране труда, программа подготовки рабочих по охране труда, перечень работ с повышенной опасностью, журнал регистрации вводного инструктажа по охране труда);
* обеспечение электробезопасности (приказ о назначении лица, ответственного за общее состояние электрохозяйства – лица, ответственного за электрохозяйство, журнал проверки знаний по технике безопасности у персонала с группой электробезопасности, техническая документация на электрохозяйства, паспорта на заземляющие устройства и др.);
* контроль необходимого уровня производственного освещения (проектная документация на устройство электрического освещения, схема сети освещения, картотека текущей эксплуатации, ремонтов и изменений в схемах, акт о приемке осветительной установки комиссией, документы о проведении осмотров и проверок осветительной сети в сроки);
* обеспечение пожарной безопасности (приказ о назначении ответственных за пожарную безопасность отдельных территорий, зданий и сооружений, а также технологического и инженерного оборудования, паспорт пожарной безопасности, приказ об установлении противопожарного режима на предприятии, система обеспечения пожарной безопасности на предприятии, график дежурств, техническая документация на установки пожарной автоматики);
* мониторинг радиационной безопасности (санитарный паспорт на право работы с источниками ионизирующих излучений, на специализированные автомашины для постоянной перевозки радионуклидных веществ и их отходов);
* обеспечение безопасности при работе с грузоподъемными машинами (лицензия на осуществление соответствующих видов деятельности по грузоподъемным машинам, приказ о назначении лиц ответственных за содержание грузоподъемных машин в исправном состоянии, журнал регистрации результатов осмотров, технических обслуживаний, сведений о ремонтах грузоподъемных машин);
* снабжение работников спецодеждой, спецобувью и другими средствами индивидуальной защиты (перечень профессий и должностей работников на бесплатную выдачу спецодежды, спецобувь и другие средства индивидуальной защиты);
* расследование и учёт несчастных случаев на производстве, профессиональных заболеваний (акты о несчастных случаях на производстве формы, акты о профессиональных заболеваниях формы ПЗ-1, журналы регистрации микротравм, сообщения о несчастных случаях на производстве с тяжелым, смертельным исходом, групповых несчастных случаях, документы расследований несчастных случаев);
* организация медицинских осмотров работников (утвержденный нанимателем перечень контингентов и лиц, подлежащих периодическим медицинским осмотрам, утвержденные нанимателем пофамильные списки лиц, подлежащих медицинскому осмотру, санитарно-гигиеническая характеристика условий труда работающих при подозрении у них профессиональных заболеваний);
* проведение паспортизации санитарно-технического состояния условий и охраны труда, аттестации рабочих мест по условиям труда (приказ о сроках проведения паспортизации и объектах, подлежащих паспортизации, порядке сбора и обобщения исходных данных, а также о лицах, осуществляющих эту работу, карты условий труда на рабочие места).

Все работники при поступлении на работу проходят следующие виды инструктажа по ОТ.

Вводный инструктаж (проводится со всеми вновь поступающими на работу, делается соответствующая запись в журнале регистрации вводного инструктажа и в удостоверении по технике безопасности).

Первичный инструктаж на рабочем месте (проводится после прослушивания вводного инструктажа мастером на рабочем месте, а также если рабочий переводится на другое место работы с записью в журнале регистрации инструктажа на рабочем месте).

Внеплановый инструктаж (проводится: при введении в действие новых или пересмотренных стандартов, правил, инструкций по охране труда, а также изменений и дополнений к ним; при изменении технологического процесса, замене или модернизации оборудования, приспособлений, исходного сырья, материалов и других факторов, влияющих на безопасность труда; при нарушении работающими требований безопасности труда, которые могут привести или привели к травмам, взрыву или пожару).

На ОАО «Керамин» имеется ряд недостатков: несвоевременное проведение инструктажей и нарушение в правилах их проведения; нет данных о проведении аттестации; нет материальной и моральной заинтересованности.

Для предупреждения заболеваний каждый работник должен соблюдать правила личной гигиены.

Перед едой тщательно вымыть руки с мылом. Пищу принимать только в отведенных для этого помещениях, столовых или буфетах.

Запрещается пить воду, предназначенную для технических нужд. Пользоваться следует только питьевыми фонтанчиками, сатураторными установками.

При каких-либо повреждениях кожного покрова нельзя накладывать повязки из бумаги или тряпок, так как этим можно загрязнить рану, что вызовет серьезное заболевание.

При каждом, даже незначительном несчастном случае (порезы, уколы, ожоги, засорение глаз и тому подобное) необходимо немедленно обратиться в медпункт. По окончании работы обязательно принять душ, переодеться, а спецодежду оставить в раздевалке.

Организация медицинских (профессиональных) осмотров и профилактического питания работающих и обоснование ее необходимости

В соответствии со статьей 228 ТК РБ для обеспечения безопасности труда и предупреждения профессиональных заболеваний наниматель обязан организовать проведение предварительных (при поступлении на работу) и периодических (в течение трудовой деятельности) медицинских осмотров работников, занятых на работах с вредными или опасными условиями труда или на работах, где есть необходимость в профессиональном отборе.

На время прохождения периодических медицинских осмотров за работником сохраняется его место работы и средний заработок.

Предварительные при поступлении на работу и периодические медицинские осмотры осуществляются с целью:

* определения соответствия (пригодности) рабочих и служащих поручаемой им работе;
* обеспечения безопасности труда и предотвращения распространения инфекционных и паразитарных заболеваний;
* выявление лиц с профессиональными заболеваниями или с подозрением на профессиональные заболевания;
* распознавание общих заболеваний, при которых дальнейшая работа в контакте с профессиональной вредностью может ухудшить их течение.

Медицинские осмотры работников осуществляются медико-санитарными частями (включая ведомственные учреждения) и территориальными лечебно-профилактическими учреждениями.

Медицинские осмотры работников осуществляются комиссией врачей, утверждаемой приказом главного врача лечебного учреждения. Ответственность за качество профилактических осмотров возлагается на заместителя главного врача по лечебным вопросам или по поликлинической службе.

Главные врачи обеспечивают мероприятия по проведению медицинских осмотров и несут ответственность за их качество, проведение всех лабораторных и функциональных исследований.

Предварительные осмотры проводятся только по направлению нанимателя с указанием производств, профессий и профессиональных вредностей согласно перечню вредных веществ и неблагоприятных производственных факторов, при работе с которыми обязательны предварительные (при поступлении на работу) и периодические медицинские осмотры в целях предупреждения заболеваний и обеспечения безопасности труда.

Результаты предварительного осмотра заносятся в форму 025/у с оформлением социально-клинического заключения. На руки пациенты выдается справка с указанием сведений «годен» или «негоден» к выполнению данной работы в контакте с указанными вредностями.

Календарный план проведения медицинских осмотров согласовывается с ЦГиЭ и нанимателем, утверждается главным врачом лечебно-профилактического учреждения, один экземпляр направляется в территориальный ЦГиЭ.

Наниматель на основании списка контингентов составляет на очередной год список лиц, подлежащих периодическому медицинскому осмотру по предприятию в установленной форме.

Списки лиц, подлежащих периодическому медицинскому осмотру, согласовываются с территориальными центрами гигиены и эпидемиологии, подписываются нанимателем и за 15-20 дней до начала осмотра предоставляются в лечебно-профилактическое учреждение.

Очередность и порядок осмотра оформляются приказом с указанием лиц, ответственных за явку работников на осмотр. Наниматель обеспечивает явку работников на осмотры и обследования и несет ответственность за допуск к работе лиц, не прошедших медицинский осмотр.

Все данные периодического медицинского осмотра заносятся в медицинскую карту, при этом врачи, принимающие участие в осмотре, дают свое заключение о профессиональной пригодности с учетом проведенных специальных исследований. На отдельный лист амбулаторной карты заносятся данные профессионального маршрута работника с указанием предприятия, цеха, участка, профессии, стажа работы, вредных и неблагоприятных условий труда. При последующем прохождении периодических осмотров данные профессионального маршрута уточняются и, при необходимости, вносятся соответствующие изменения.

Периодические медицинские осмотры заканчиваются до 1 декабря и завершаются составлением в двухмесячный срок заключительного акта по периодическому медицинскому осмотру по установленной форме.

Заключительный акт оформляется в четырех экземплярах и направляется: нанимателю, территориальному центру гигиены и эпидемиологии, цеховому врачу, один экземпляр остается в учреждении, проводившем осмотр.

## Разработка мер по нормализации инфра- и ультразвука в помещении

Производственные факторы, формирующие условия труда в процессе труда на организм работающих кратковременно или длительно могут воздействовать неблагоприятные факторы (пыль, газы, шум, избыточное тепло и так далее), вызывающие профессиональные заболевания: глухоту, дерматит, конъюнктивит и другие заболевания.

Шум вызывает у людей раздражительность, головные боли, головокружение, приводит к тугоухости и глухоте.

В результате угнетающего воздействия шума на нервную систему в организме могут возникать различные заболевания сердечно-сосудистой системы (гипертония) и органов пищеварения.

Защита от шума достигается разработкой шумобезопасной техники, применением средств и методов индивидуальной и коллективной защиты, строительно-акустическими методами. Средства коллективной зашиты делятся по отношению к источнику шума на снижающие шум в источнике возникновения (наиболее эффективно) и снижающие шум на путях его распространения. По способу реализации различают следующие методы защиты:

* акустические - основаны на акустическом расчете помещения и подборе по принципу действия средств звукоизоляции, звукопоглощения, виброизоляции, демпфирования, глушителей шума;
* строительно-акустические экраны, звукоизоляция, кабины наблюдения, дистанционное управление, кожухи, уплотнения и т.д. Наиболее эффективны такие звукоизолирующие материалы, как трипласт (композиционный материал) и пластобетоны с наполнителями из хлопка, опилок древесины, соломы и т.д. Звукопоглощающими материалами являются также мрамор, бетон, гранит, кирпич, ДВП, ДСП, войлок, минераловата, материалы со щелевой перфорацией;
* архитектурно-планировочные - рациональное размещение рабочих мест; рациональный режим труда и отдыха.

Инфразвук – это акустические колебания с частотой ниже 20 Гц, которые находятся в частотном диапазоне ниже порога слышимости. Производственный инфразвук возникает в тех процессах, что и шум слышимых частот.

Инфразвук как физическое явление подчиняется общим закономерностям, характерным для звуковых волн, однако обладает целым рядом особенностей, связанных с низкой частотой колебаний упругой среды:

* имеет во много раз большие амплитуды колебаний, чем акустические волны при равных мощностях источников звука;
* распространяется на большие расстояния от источника генерирования ввиду слабого поглощения его атмосферой.

Большая длина волны делает характерным для инфразвука явление дифракции – огибание волнами различных препятствий, если размеры препятствия около длины волны или больше. Инфразвук проникает в помещения и обходит преграды, задерживающие слышимые звуки. Инфразвуковые колебания способны вызвать вибрацию крупных объектов вследствие явлений резонанса. Указанные особенности инфразвука затрудняют борьбу с ним.

Воздействуя на организм человека, инфразвук вызывает неприятные субъективные ощущения и многочисленные реактивные измене­ния, к числу которых относятся астенизация, изменения в центральной нервной, сердечно-сосудистой и дыхательной системах, вестибулярном анализаторе.

Основными методами и средствами защиты от инфразвука являются:

* изменение режима работы технологического оборудования – увеличение его быстроходности с тем, чтобы основная частота следования силовых импульсов лежала за пределами инфразвукового диапазона;
* снижение интенсивности аэродинамических процессов: огра­ничение скоростей движения транспорта, снижение скоростей истечения жидкостей;
* глушители интерференционного типа;
* рациональный режим труда и отдыха;
* использование средств индивидуальной защиты (противошумы, специальные пояса и др.).

Ультразвук – это упругие волны с частотой колебаний от 20 кГц и до 1 ГГц, которые не слышимы человеческим ухом. Источниками ультразвука являются все виды ультразвукового технологического оборудования; ультразвуковые приборы и аппаратура промышленного, медицинского и бытового назначения, которые генерируют ультразвуковые колебания в диапазоне от 18 кГц до 100 МГц и выше.

Низкочастотные ультразвуковые колебания хорошо распространяются в воздухе. Биологический эффект воздействия их на организм зависит от интенсивности, длительности воздействия и размеров поверхности тела, подвергаемого действию ультразвука.

При систематическом воздействии интенсивного низкочастотного ультразвука, если его уровень превышает предельно допустимый, у работников могут наблюдаться функциональные изменения центральной и периферической нервной системы, сердечно-сосудистой и эндокринной систем, слухового и вестибулярного анализаторов, гуморальные нарушения. Данные о действии высокочастотного ультразвука на организм человека свидетельствуют о поли­морфных изменениях почти во всех тканях, органах и системах. Происходящие под воздействием ультразвука (воздушного и контактного) изменения подчиняются общей закономерности: малые интенсивности стимулируют, активируют. Средние и большие – угнетают, тормозят и могут полностью подавлять функции.

Контактное воздействие высокочастотного ультразвука на руки приводит к нарушению капиллярного кровообращения в кистях рук, снижению болевой чувствительности, т.е. развиваются периферические неврологические нарушения. Установлено, что ультразвуковые колебания могут вызывать изменения костной структуры с разрежением плотности костной ткани.

Контактное воздействие ультразвука исключается при автоматизации производственных процессов, применении дистанционного управления, технических средств фиксации источника ультразвука относительно обрабатываемой детали, нанесении виброизолирующих покрытий на ручки ультразвукового оборудования.

Защита работников от неблагоприятного воздействия ультразвука достигается путем:

* проведения предварительных и периодических медосмотров;
* физиопрофилактических процедур (тепловые воздушные с микромассажем и тепловые гидропроцедуры для рук, массаж верхних конечностей и др.);
* рефлексопрофилактики;
* гимнастических упражнений;
* психофизической разгрузки;
* витаминизации, сбалансированного питания;
* организации рационального режима труда и отдыха и др.

Источниками шума на ОАО «Керамин» является производственное оборудование: пресса для производства плитки, шаровые мельницы, дробилки, насосы, вентиляторы и другое оборудование.

Действие шума и ультразвука на работающих уменьшается путем устранения или ослабления шума в местах их возникновения, звукоизоляция и поглощение звука. Акустическая обработка помещений, архитектурно-планировочные мероприятия, а также применение индивидуальных средств защиты.

Чтобы уменьшить шум от машин, агрегатов, вентиляторов и обезопасить конструктивные элементы строений, механизмы устанавливают на специальные фундаменты. Чтобы не допускать распространения шума по вентиляционным каналам, вентиляторы соединяют с воздуховодами с помощью гибких вставок из брезента. Шум значительно уменьшает звукоизоляция. С этой целью машины покрывают кожухами или выносят в отдельные помещения.

Наиболее шумными в ОАО «Керамин» являются массозаготовительные цеха, в которых производится обработка сырья в различного рода дробилках и помол компонентов в шаровых мельницах. Шаровые мельницы с кремниевой футеровкой переводят на резиновую, что приводит к снижению шума. Индивидуальные средства защиты, применяемые рабочими – противошумы (антифоны) и беруши.

Защита от ультразвука (вибрация) состоит также и в устранении контакта с колеблющимися средами. Используют в этом случае специальные приспособления, не связанные жестко с колеблющейся поверхностью, либо защитные рукавицы или перчатки.

## Обеспечение пожарной безопасности

Согласно ТКП 474-2013 по взрывопожарной и пожарной опасности помещения подразделяются на категории А, Б, В1–В4, Г1, Г2, Д, а здания – на категории А, Б, В, Г и Д. По взрывопожарной и пожарной опасности наружные установки подразделяются на категории АН, БН, ВН, ГН, ДН.

Таблица 7.1 – Категория помещений

|  |  |  |
| --- | --- | --- |
| Категория помещения | Характеристика веществ и материалов, находящихся (обращающихся) в помещении | |
| А  (взрывопожароопасная) | Горючие газы, легковоспламеняющиеся жидкости с температурой вспышки не более 28 °С в таком количестве, что могут образовывать взрывоопасные парогазовоздушные смеси, при воспламенении которых развивается расчетное избыточное давление взрыва в помещении, превышающее 5 кПа. Вещества и материалы, способные взрываться и гореть при взаимодействии с водой, кислородом воздуха или друг с другом в таком количестве, что расчетное избыточное давление взрыва в помещении превышает 5 кПа | |
| Б  (взрывопожароопасная) | Горючие пыли или волокна, ЛВЖ с температурой вспышки более 28 °С, горючие жидкости в таком количестве, что могут образовывать взрывоопасные пыле- или паровоздушные смеси, при воспламенении которых развивается расчетное избыточное давление взрыва в помещении, превышающее 5 кПa | |
| B1–В4  (пожароопасные) | ГГ, ЛВЖ, ГЖ и трудногорючие жидкости, твердые горючие и трудногорючие вещества и материалы (в том числе пыли и волокна), вещества и материалы, способные при взаимодействии с водой, кислородом воздуха или друг с другом взрываться и гореть, при условии, что помещения, в которых они имеются в наличии или обращаются, не относятся к категории А или Б | |
| Г1 | ГГ, ЛВЖ, ГЖ, твердые горючие вещества и материалы, которые  сжигаются или утилизируются в процессе контролируемого горения в качестве топлива |
| Г2 | Негорючие вещества и материалы в горячем, раскаленном или расплавленном состоянии, процесс обработки которых сопровождается выделением лучистого тепла, искр и пламени |
| Д | Негорючие вещества и материалы в холодном состоянии, горючие и трудногорючие вещества и материалы в таком количестве, что удельная пожарная нагрузка на участке их размещения в помещении не превышает 100 МДж/м2, а пожарная нагрузка в пределах помещения – 1000 МДж |

Помещение относится к категории B1-B4 по причине того, что там обращаются твердые горючие вещества и материалы, способные, при взаимодействии с кислородом, гореть. Классом пожарной опасности для здания является П-Паб, так как в нем обращаются твердые горючие вещества.

Степень огнестойкости здания равна III, что, непосредственно, связано с ограждающими и несущими конструкциями из естественных или искусственных каменных материалов, железобетона с применением листовых и плитных негорючих материалов или бетона.

По пределу огнестойкости и классу пожарной опасности строительных конструкций здание характеризуется:

* наружные несущие стены (Е 30-КО): предел огнестойкости составляет 30 минут – по потере целостности и не пожароопасные;
* несущие элементы здания (R 90-КО): предел огнестойкости составляет 90 минут – по потере несущей способности и не пожароопасные;
* самонесущие стены (RE 60-КО): предел огнестойкости составляет 60 минут – по потере несущей способности и потере целостности независимо от того, какое из двух предельных состояний наступит ранее, а также не пожароопасные;
* перекрытия междуэтажные (REI 60-КО): предел огнестойкости составляет 60 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое их трех предельных состояний наступит ранее, а также не пожароопасные;
* внутренние стены (REI 105-КО): предел огнестойкости составляет 105 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое их трех предельных состояний наступит ранее, а также не пожароопасные;
* марши и площадки лестниц (R 45-КО): предел огнестойкости составляет 45 минут – по потере несущей способности, а также не пожароопасные;
* настилы, в том числе и с утеплителем (RE 30-КО): предел огнестойкости составляет 30 минут – по потере несущей способности, целостности и теплоизолирующей способности независимо от того, какое их трех предельных состояний наступит ранее, а также не пожароопасные.

В помещении используются ИПД-3 (извещатель пожарный дымовой оптико-электронный автономный) для обнаружения возгорания и оповещения в службу пожарной охраны. Его предназначением является обнаружение возгораний, которые сопровождаются появлением дыма, а после чего выдача светового и громкого звукового сигнала «Тревога».

Первичные средства пожаротушения должны размешаться в легкодоступных местах и не должны быть помехой и препятствием при эвакуации персонала из помещений.

В каждом помещении имеется по одному углекислотному огнетушителю ОУ-2, которые расположены возле выходной двери на высоте полтора метра от уровня пола.

Для проведения эвакуации людей и имущества разрабатывается план эвакуации. План эвакуации включает мероприятия, которые обеспечивают своевременное оповещение о пожаре, выход всех людей наружу по наиболее безопасным путям, спокойствие и порядок при движении, тушение пожара первичными средствами.

План эвакуации состоит из двух частей: текстовой (инструкции) и графической (плана). В инструкции излагаются обязанности лиц. Которые осуществляют эвакуацию, порядок исполнения обязанностей. В графической части приводят маршруты движения и приводят соответствующие пояснения к ним. Утверждает план эвакуации руководитель предприятия, а также и объявляет приказ по утверждению о вступлении его в действие, определяет сроки изучения и практические отработки этого плана сотрудниками. План эвакуации составляют в двух экземплярах: один вывешивают в помещении, а другой хранят в деле у руководителя. На графической части плана эвакуации направление движения эвакуационных потоков указывают стрелками.

Пожарные дружины создаются на всех объектах, независимо от наличия других видов пожарных служб, где численность работников превышает 15 человек.

Основные задачи пожарной дружины:

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

## Охрана окружающей среды

Проблема защиты окружающей среды ‒ одна из важнейших задач современности. Выбросы промышленных предприятий, энергетических систем и транспорта в атмосферу, водоемы и недра на современном этапе развития достигли таких размеров, что в большинстве районов земного шара уровни загрязнений существенно превышают допустимые санитарные нормы. Множество разработанных технологических процессов и появление новых видов продукции, особенно в химической промышленности, привели не только к увеличению количества загрязнений, но и к существенному увеличению числа токсичных примесей, поступающих в окружающую среду. Вредные выбросы промышленных предприятий и других источников загрязнения оказывают отрицательное воздействие не только на окружающую среду, но и в ряде случаев значительно влияют на процесс эксплуатации технических средств.

Решение проблемы защиты окружающей среды от выбросов промышленных предприятий состоит в создании замкнутых технологических циклов (безотходные системы), однако их разработка и внедрение требуют новых технологических и конструктивных решений, а также больших капиталовложений. В современных условиях часто используют способы защиты окружающей среды от примесей, основанные на максимальном их улавливании или обезвреживании в специальных аппаратах.

Проблема охраны окружающей среды − комплексная проблема. Планируя дальнейшее развитие индустриального производства, необходимо оценивать эффективность его развития. Комплексный характер проблемы охраны окружающей среды определяется сложностью системы, состоящей из природы и производства. Оптимальное развитие этой системы невозможно без комплексного учета социальных, экологических, технических, экономических, правовых и международных аспектов проблемы.

Вопросы экологии на ОАО «Керамин» играет не последнюю роль в его производственной деятельности.

На предприятии перерабатывается большое количество разнообразных твердых материалов, часть которых в процессе производства керамической плитки попадает в виде различных отходов в помещения цехов и окружающую среду.

Для уменьшения влияния отходов на окружающую среду на предприятии совершенствуются технологические процессы, а также внедряются эффективные методы очистки отходящих газов и сточных вод.

Основными факторами вредного воздействия на окружающую среду при керамической плитки являются:

– выбросы в атмосферу, загрязняющие воздух: пыль от технологического оборудования и вентиляционных систем; летучие вещества, выделяющиеся при термообработке и других процессах;

– сброс производственных, хозяйственно-бытовых и поверхностных вод в водотоки, водоемы и на почву;

– размещение отходов на организованных и несанкционированных свалках (полигонах);

– ущерб природе, связанный с потреблением (изъятием) различных видов сырья, материалов, энергоносителей и отчуждением земель;

– шум, вибрация, излучение.

Получение стекла для защитной оболочки связано с отвешиванием исходных сырьевых материалов, просеиванием, смешиванием, варкой и выработкой. В процессе составления шихты и варки сырья часть компонентов улетучивается в атмосферу в виде пыли и газов.

На предприятии ОАО «Керамин» оборудование для просеивания сырьевых материалов и приготовления сырья является источником выделения пыли и загрязняющих веществ в окружающую среду.

Выбросы вредных веществ в тигельной печи формируются из продуктов уноса. По агрегатному состоянию они делятся на твердые и газообразные. Их количество относительно невелико, однако из-за их токсичности и потенциальной опасности выбрасываемых газов и пыли такая печь относится к интенсивным загрязнителям окружающей среды. Твердыми выбросами печи является пыль, состоящая из легколетучих компонентов шихты и продуктов их химического взаимодействия.

При получении керамической плитки используется тигельная электрическая печь. Полная замена процесса сжигания топлива устраняет выделение продуктов горения, в частности оксидов азота, диоксида углерода, диоксида серы. При использовании электрической энергии снижаются потери сырьевых материалов. Однако, если учитывать эффективность процессов производства и передачи электрической энергии и связанное с ними выделение загрязняющих веществ, использование электрической энергии для изготовления плитки является менее эффективным с точки зрения минимизации воздействия на окружающую среду и, в частности, выделения парниковых газов.

Контроль за соблюдением нормативов ПДВ осуществляет предприятие по расчетным данным отчета по инвентаризации источников выбросов загрязняющих веществ и инструментальными методами лаборатории промсанитарии АКП для каждого вредного вещества в зависимости от мощности выброса вещества источниками, максимальной разовой предельно допустимой концентрации для безопасного ориентировочного уровня воздействия вредного вещества в атмосферном воздухе населенных мест.

Санитарными нормами проектирования промышленных предприятий СН-245-71 установлены ПДК вредных веществ в атмосферном воздухе населенных мест, а ГОСТ 12.1.005-88 нормирует ПДК вредных веществ в воздухе рабочей зоны производственных помещений. Величина ПДК для населенных мест в 10 и более раз ниже, чем для производственных помещений.

# Экономический раздел

## Технико-экономическое обоснование разработки программного средства

Необходимо рассчитать экономическую эффективность разработки программного средства.

В технико-экономическом обосновании будут рассмотрены следующие вопросы:

* составление плана по разработке программного средства;
* расчет стоимости разработки;
* экономическая эффективность разработки.

В первой части технико-экономического обоснования необходимо рассчитать срок разработки по созданию программного средства «Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин».

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

В третьей части необходимо оценить экономическую эффективность программного средства.

## 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 |
| Инициализация | техник-программист | 1 | 4 |
| Планирование | техник-программист | 1 | 2 |
| Разработка | техник-программист | 1 | 20 |
| Отладка программ и анализ результатов | техник-программист | 1 | 8 |
| Оформление документации и подготовка к сдаче разработки | техник-программист | 1 | 6 |

## 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 – коэффициент, учитывающий транспортно-заготовительные расходы, ≈от 1,05 до 1,10;

– норма расхода i-го вида материалов на макет или опытный образец (кг, м, и т.д.);

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

– возвратные отходы i-го вида материала (кг, м, и т.д.);

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

– 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 приобретения без НДС, руб. | НДС, руб. | Цена с НДС, руб. |
| Бумага | пачка | 1 | 9,03 | 1,81 | 10,84 |
| Ручка шариковая | штука | 1 | 2,04 | 0,41 | 2,45 |
| Папка-скоросшиватель | штука | 1 | 13,90 | 2,78 | 16,68 |
| Диск | штука | 1 | 1,29 | 0,26 | 1,55 |
| Всего расходов | | | | | 31,52 |
| Всего с транспортно-заготовительными расходами | | | | | 34,67 |

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)

где– стоимость одного кВт/ч, руб.;

– количество кВт/ч.

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

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

Рэ = 0.38994 × 143 = 55,76 руб.

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 (т.е. на 168 часов).

Тарифная ставка первого разряда на предприятии составляет 90 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. |
| Подготовительный | техник-программист | 7 | 2,03 | 1,09 | 32 | 34,88 |
| Выбор методов и средств | техник-  программист | 7 | 2,03 | 1,09 | 16 | 17,44 |
| Разработка алгоритмов и программ | техник-  программист | 7 | 2,03 | 1,09 | 160 | 174,40 |
| Отладка программ и анализ результатов | техник-  программист | 7 | 2,03 | 1,09 | 64 | 69,76 |
| Оформление документации и подготовка к сдаче разработки | техник-  программист | 7 | 2,03 | 1,09 | 48 | 52,32 |
| Всего | | | | | | 348,8 |
| Всего с коэффициентом премий | | | | | | 383,68 |

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=383,68\*=38,37 руб.

Рассчитываются отчисления органам социальной защиты по формуле (8.4)

, (8.4)

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

Рос=(383,68+38,37)\*=143,50 руб.

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=(383,68+38,37)\*=2,53 руб.

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=383,68\*=38,37 руб.

Исчисляются косвенные (накладные) расходы по формуле (8.7)

, (8.7)

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

Ркос=383,68\*=211,02 руб.

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

 (8.8)

Сп=34,67+55,76+383,68+38,37+143,50+2,53+38,37+211,02=907,90 руб.

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=907,90\*=90,79 руб.

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

 (8.10)

Цотп=907,90+90,79=998,69 руб.

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

, (8.11)

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

НДС=998,69\*=199,74 руб.

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

 (8.12)

Цотп=998,69+199,74=1198,43 руб.

Все приведенные выше расчеты целесообразно объединить в сводную таблицу 8.4.

Таблица 8.4 – Расчет ориентировочной цены научно-технической продукции

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Условные обозначения | Сумма, руб. |
| Материалы, покупные полуфабрикаты и комплектующие изделия | Рм | 34,67 |
| Электроэнергия | Рэ | 55,76 |
| Основная заработная плата научно-производственного персонала | Роз | 383,68 |
| Дополнительная заработная плата научно-производственного персонала | Рдз | 38,37 |
| Отчисления на социальную защиту | Рос | 143,50 |
| Отчисления на страхование от несчастных случаев | Остр | 2,53 |
| Прочие прямые расходы | Рпр | 38,37 |
| Накладные расходы | Ркос | 211,02 |
| Полная себестоимость | Сп | 907,90 |
| Плановые накопления (прибыль) | П | 90,79 |
| Отпускная цена (без НДС) | Ц | 998,69 |
| Налог на добавленную стоимость | РНДС | 199,74 |
| Отпускная цена с НДС | Цотп | 1198,43 |

## Экономическая эффективность разработки

В данном разделе представлено экономическое обоснование для дипломного проекта по теме «Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин».

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 1198,43 рублей.

Основное преимущество разработки программного средства в том, что оно имеет интуитивно понятный графический интерфейс, в связи с чем является простым в использовании и не требует наличия специальных навыков и знаний для его эксплуатации. Данное программное средство имеет ряд преимуществ перед своим аналогом в виде веб-сайта «Интернет-магазин Керамин», таких как: наличие калькулятора для расчета необходимого количества керамических изделий для комнаты и ее отдельных частей, возможность реализации изделий не только физическим лицам, но и юридическим, ведение комфортного учета, а также реализации изделий без использования большого количества сотрудников. У изделий учитываются такие параметры, как: наименование, тип, коллекция, размер, цвет, поверхность, артикул, количество, вес ящика, количество в ящике, цена, описание, статус наличия и изображение. Также, в программном средстве предусмотрена возможность формирования статистики по продуктивности сотрудников и по популярности изделий, с последующим формированием документов по ним. К преимуществам можно отнести быстродействие и малые системные требования программного средства, а также экономическую выгоду в его использовании, так как оно является абсолютно бесплатным и получает регулярные обновления.

# Заключение

В рамках дипломного проектирования была произведена работа над проектом «Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин».

Для достижения цели дипломного проектирования были решены следующие задачи:

* определена вычислительная система, необходимая для создания программного средства;
* разработана физическая и логическая модель данных;
* по модели выполнено проектирование задачи;
* разработано программное средство;
* описано созданное программное средство;
* выбрана методика испытаний;
* описан процесс тестирования;
* приведены примеры области применения.

Проект был разработан в среде Microsoft Visual Studio 2019 на языке C#. База данных была разработана в системе управления базами данных Microsoft SQL Server 2017 с использованием технологии ADO.NET.

В ходе разработки была исследована предметная область, после чего осуществлялась проектировка базы данных, разработка структура приложения, дизайн WPF-окон. Также был реализован ряд задач: просмотр списка изделий, сотрудников, пунктов самовывоза, завершенных заказов и статистики по изделиям и сотрудникам, разграничение прав доступа к данным, добавление и редактирование информации об изделиях, сотрудниках и пунктах самовывоза, добавление изделий в корзину, формирование заказов, поиск изделий по их артикулу, фильтрация списка изделий, просмотр статистики и списка оформленных заказов, изменение данных учетной записи, рассчет необходимого количества изделий для реализации, формирование документов по заказам и статистике. Для проверки функционирования были составлены тесты, которые были успешно пройдены.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий провести автоматизированную работу с данными о деталях.

В итоге, программное средство имеет свои преимущества:

* интуитивно понятный графический интерфейс;
* простота в использовании;
* отсутствие необходимости в обладании специальными навыками при работе с программой;
* наличие калькулятора плитки;
* возможность просмотра статистики по продуктивности сотрудников и популярности изделий за определенный период, а также создание документов на основании этой статистики.

К минусам можно отнести возможность релизации лишь керамической плитки, что в последующем будет доработано.

В рамках дипломного проектирования также были изучены правовые, нормативные, социально-экономические и организационные вопросы охраны труда, обеспечения пожарной безопасности и охраны окружающей среды, а также рассмотрен вопрос о разработке мер по нормализации инфра- и ультразвука в помещении.

В экономическом разделе был произведен рассчет стоимости программного средства с учетом затрат на его разработку, которая составила 1198,43 рублей.

Программное средство реализовано и полностью соответствует техническому заданию, отлажено и протестировано. Приложение готово к практическому использованию.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по дипломному проектированию / Т.Г. Багласова – Минск: КБП, 2020.
2. Категорирование помещений, зданий и наружных установок по взрывопожарной и пожарной безопасности : ТКП 474-2013 (02300). Введ. 29.01.13. – Минск : Научно-исследовательский ин-т пожарной безопасности и проблем чрезвычайных ситуаций, 2013.
3. Об охране окружающей среды : Закон Респ. Беларусь, 26 янв. 1992 г., № 1982-XII : в ред. Закона Респ. Беларусь от 29.01.1993 г.
4. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны : ГОСТ 12.1.005-88. – Введ. 17.12.92. – Минск : Госстандарт : Государственный комитет по стандартизации Республики Беларусь, 1992.
5. Dr.Explain – программа для быстрого создания help файлов, online справки, руководств пользователей и документации [Электронный ресурс]. – drexplain.ru, 2021. – Режим доступа : https://www.drexplain.ru/. – Дата доступа : 22.04.2021.
6. Microsoft Office Excel – что это? [Электронный ресурс]. – Режим доступа : https://office-guru.ru/excel/microsoft-office-excel-chto-eto-59.html. – Дата доступа : 18.04.2021
7. SQL Server 2017 в Windows и Linux [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : https://www.microsoft.com/ru-ru/sql-server/sql-server-2017/. – Дата доступа : 07.04.2021.
8. Visual Studio 2019 [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019. – Дата доступа : 03.04.2021.
9. Полное руководство по языку программирования С# [Электронный ресурс]. – Metanit, 2021. – Режим доступа : https://metanit.com/sharp/tutorial/. – Дата доступа : 03.04.2021.
10. Руководство по ADO.NET и работе с базами данных в C# [Электронный ресурс]. – Metanit, 2021. – Режим доступа : https://metanit.com/sharp/adonet/. – Дата доступа : 04.04.2021.
11. Руководство по WPF [Электронный ресурс]. – Metanit, 2021. – Режим доступа : https://metanit.com/sharp/wpf/. – Дата доступа : 03.02.2021.
12. Скачивание SQL Server Management Studio (SSMS) [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. – Дата доступа : 02.04.2021.

# Приложение А

(обязательное)

Текст программы

«KeraminStore.sql»

create database KeraminStore

use KeraminStore

--Создание таблицы с должностями

create table Post

(postCode int not null identity,

postName varchar(100) not null,

constraint postNameUniq unique (postName),

constraint postCodePK primary key (postCode))

--Заполнение таблицы должностями

insert into Post(postName) values ('Администратор'), ('Штатный сотрудник')

--Создание таблицы с сотрудниками

create table Employee

(employeeCode int not null identity,

employeeLogin varchar(30) not null,

constraint employeeLoginUniq unique (employeeLogin),

employeeAdminStatus bit not null,

employeePassword varchar(30) not null,

employeeName varchar(50) not null,

employeeSurname varchar(50) not null,

employeePatronymic varchar(50) not null,

employeeDateOfBirth date not null,

constraint employeeCodePK primary key (employeeCode),

postCode int not null,

constraint postCodeFK foreign key (postCode) references Post (postCode) on delete cascade)

--Заполнение таблицы сотрудниками

insert into Employee(employeeLogin, employeePassword, employeeName, employeeSurname, employeePatronymic, employeeDateOfBirth, postCode, employeeAdminStatus) values ('Admin', 'Qwerty123', 'Вячеслав', 'Хитров', 'Андреевич', '06.01.2000', 1, 1)

--Создание таблицы с коллекциями для изделий

create table ProductCollection

(productCollectionCode int not null identity,

productCollectionName varchar(100) not null,

constraint productCollectionNameUniq unique (productCollectionName),

constraint productCollectionCodePK primary key (productCollectionCode))

--Заполнение таблицы коллекциями

insert into ProductCollection (productCollectionName) values ('Классик'), ('Рио'), ('Керкира'), ('Комо'), ('Болонья'), ('Гранада'), ('Букингем'), ('Дюна'), ('Марсала'), ('Метро'),

('Котто'), ('Атрум'), ('Монако'), ('Сидней'), ('Сонора'), ('Телари'), ('Шиен'), ('Сагано'), ('Мари Эрми'), ('Ассам'), ('Шарм'), ('Аляска'), ('Вайоминг'), ('Сохо'), ('Намиб'),

('Бунгало'), ('Гавана'), ('Монте'), ('Эклипс'), ('Скаген'), ('Сабвэй'), ('Нидвуд'), ('Киото'), ('Фландрия'), ('Дубай'), ('Сиена'), ('Прованс'), ('Вилландри'), ('Синай'), ('Авейру'),

('Саванна'), ('Асуан'), ('Верона'), ('Ванкувер'), ('Мирабель'), ('Барселона'), ('Фреско'), ('Тоскана'), ('Треви'), ('Винтаж'), ('Венто'), ('Сорбонна'), ('Риальто'), ('Оксфорд'),

('Невада'), ('Фоскари'), ('Бирма'), ('Марокко'), ('Дамаск'), ('Миф'), ('Форум'), ('Элегия'), ('Троя'), ('Майорка'), ('Помпеи'), ('Аксель'), ('Дежавю'), ('Калипсо'), ('Тренд'),

('Лаура'), ('Акцент'), ('Фреш'), ('Соло'), ('Орлеан'), ('Нью-Йорк'), ('Концепт'), ('Ирис'), ('Эллада'), ('Сиерра'), ('Дария'), ('Флориан'), ('Сакура'), ('Пастораль'), ('Органза'),

('Пальмира'), ('Мишель'), ('Манхэттен'), ('Гламур'), ('Бисер')

--Создание таблицы со статусами наличия изделий

create table AvailabilityStatus

(availabilityStatusCode int not null identity,

availabilityStatusName varchar(14),

constraint availabilityStatusNameUniq unique (availabilityStatusName),

constraint availabilityStatusCodePK primary key (availabilityStatusCode))

--Заполнение таблицы со статусами наличия изделий

insert into AvailabilityStatus (availabilityStatusName) values ('Есть в наличии'), ('Нет в наличии')

--Создание таблицы с типами изделий

create table ProductType

(productTypeCode int not null identity,

productTypeName varchar(100) not null,

constraint productTypeNameUniq unique (productTypeName),

constraint productTypeCodePK primary key (productTypeCode))

--Заполнение таблицы типами изделий

insert into ProductType (productTypeName) values ('Настенная плитка'), ('Напольная плитка'), ('Настенный декор'), ('Напольный декор'), ('Мозаика'), ('Бордюр')

--Создание таблицы с поверхностями

create table Surface

(surfaceCode int not null identity,

surfaceName varchar(100) not null,

constraint surfaceNameUniq unique (surfaceName),

constraint surfaceCodePK primary key (surfaceCode))

--Заполнение таблицы с поверхностями

insert into Surface (surfaceName) values ('Глянцевая'), ('Матовая')

--Создание таблицы с цветами

create table Color

(colorCode int not null identity,

colorName varchar(100) not null,

constraint colorNameUniq unique (colorName),

constraint colorCodePK primary key (colorCode))

--Заполнение таблицы цветами

insert into Color (colorName) values ('Разноцветный'), ('Светло-бежевый'), ('Светло-серый'), ('Синий'), ('Темно-бежевый'), ('Темно-серый'), ('Серый'), ('Черный'), ('Бежевый'), ('Коричневый'), ('Зеленый'), ('Голубой'), ('Розовый'), ('Красный'), ('Фиолетовый'), ('Желтый'), ('Оранжевый'), ('Белый')

--Создание таблицы с изделиями

create table Product

(productCode int not null identity,

productName varchar(100) not null,

productArticle varchar(11) not null,

constraint productArticleUniq unique (productArticle),

productWidth float not null,

productLenght float not null,

productBoxWeight float not null,

productCount int not null,

productCountInBox int null,

productCostCount float null,

productCostArea float null,

productImage varchar(250) not null,

productDescription varchar(300) null,

constraint productCodePK primary key (productCode),

productCollectionCode int not null,

availabilityStatusCode int not null,

productTypeCode int not null,

surfaceCode int not null,

colorCode int not null,

constraint productCollectionCodeFK foreign key (productCollectionCode) references ProductCollection (productCollectionCode) on delete cascade,

constraint availabilityStatusCodeFK foreign key (availabilityStatusCode) references AvailabilityStatus (availabilityStatusCode) on delete cascade,

constraint productTypeCodeFK foreign key (productTypeCode) references ProductType (productTypeCode) on delete cascade,

constraint colorCodeFK foreign key (colorCode) references Color (colorCode) on delete cascade,

constraint surfaceCodeFK foreign key (surfaceCode) references Surface (surfaceCode) on delete cascade)

--Создание таблицы корзина

create table Basket

(basketCode int not null identity,

basketNumber int not null,

productsCount int not null,

boxesCount int not null,

productsArea float not null,

productsWeight float not null,

generalSum float not null,

paymentStatus bit not null,

constraint basketNumberPK primary key (basketCode),

productCode int not null,

constraint productCodeFK foreign key (productCode) references Product (productCode) on delete cascade)

--Создание таблицы с видами доставки

create table Delivery

(deliveryCode int not null identity,

deliveryName varchar(100) not null,

constraint deliveryNameUniq unique (deliveryName),

constraint deliveryCodePK primary key (deliveryCode))

--Заполнение таблицы видами доставки

insert into Delivery (deliveryName) values ('Самовывоз'), ('Курьером по г.Минску (Самостоятельно)'), ('Курьером по г.Минску (С подъемом)'), ('Курьером за пределы г.Минска')

--Создание таблицы с видами оплаты

create table Payment

(paymentCode int not null identity,

paymentType varchar(100) not null,

constraint paymentTypeUniq unique (paymentType),

constraint paymentCodePK primary key (paymentCode))

--Заполнение таблицы видами оплаты

insert into Payment (paymentType) values ('Наличными'), ('Банковской картой через платежный терминал'), ('Онлайн оплата картой Visa, Master Card или Белкарт'), ('Через систему "Расчет" (ЕРИП)')

--Создание таблицы с городами

create table Town

(townCode int not null identity,

townName varchar(100) not null,

constraint townNameUniq unique (townName),

constraint townCodePK primary key (townCode))

--Заполнение таблицы городами

insert into Town (townName) values ('Минск'), ('Витебск'), ('Брест'), ('Гродно'), ('Гомель'), ('Могилев'), ('Барановичи'), ('Бобруйск'), ('Борисов'), ('Горки'), ('Лида'), ('Мозырь'), ('Молодечно'), ('Пинск'), ('Солигорск')

--Создание таблицы с городами для пунктов самовывоза

create table PickupTown

(pickupTownCode int not null identity,

pickupTownName varchar(100) not null,

constraint pickupTownNameUniq unique (pickupTownName),

constraint pickupTownCodePK primary key (pickupTownCode))

--Заполнение таблицы городами для пунктов самовывоза

insert into pickupTown (pickupTownName) values ('Минск'), ('Витебск'), ('Брест'), ('Гродно'), ('Гомель'), ('Могилев'), ('Барановичи'), ('Бобруйск'), ('Горки'), ('Лида'), ('Мозырь'), ('Молодечно'), ('Пинск'), ('Солигорск')

--Создание таблицы с улицами

create table Street

(streetCode int not null identity,

streetName varchar(100) not null,

constraint streetNameUniq unique (streetName),

constraint streetCodePK primary key (streetCode))

--Заполнение таблицы улицами

insert into Street (streetName) values ('Абрикосовая'), ('Авакяна'), ('Авангардная'), ('Авиации'), ('Авроровская'), ('Автодоровская'), ('Автозаводская'), ('Автомобилистов'), ('Азгура'),

('Азизова'), ('Азовская'), ('Айвазовского'), ('Академическая'), ('Аладовых'), ('Александровская'), ('Алеся Гаруна'), ('Алибегова'), ('Аллейная'), ('Алтайская'), ('Амбулаторная'),

('Амураторская'), ('Амурская'), ('Ангарская'), ('Андреевская'), ('Аннаева'), ('Антоновская'), ('Аполинарьевская'), ('Аранская'), ('Арктическая'), ('Артёма'), ('Артиллеристов'),

('Асаналиева'), ('Ауэзова'), ('Аэродромная'), ('Аэрофлотская'), ('50 лет БССР'), ('Гоголя'), ('Московская'), ('Бумагина'), ('Подгорная'), ('Ленина'), ('Соколовского'), ('Дзержинского'),

('Тимирязева'), ('Великий Гостинец'), ('Березовая'), ('Рокоссовского'), ('Заслонова ')

--Создание таблицы с покупателями

create table Customer

(customerCode int not null identity,

customerName varchar(50) null,

customerSurname varchar(50) null,

customerPatronymic varchar(50) null,

phone varchar(17) not null,

mail varchar(40) not null,

building varchar(8) null,

floor\_ int null,

apartment int null,

legalName varchar(100) null,

UTN int null,

townCode int null,

streetCode int null,

orderNumber int null,

constraint townCodeFK foreign key (townCode) references Town (townCode) on delete cascade,

constraint streetCodeFK foreign key (streetCode) references Street (streetCode) on delete cascade,

constraint customerCodePK primary key (customerCode))

--Создание таблицы с пунктом самовывоза

create table Pickup

(pickupCode int not null identity,

streetName varchar(100) not null,

building varchar(8) null,

pickupTownCode int not null,

constraint townFK foreign key (pickupTownCode) references PickupTown (pickupTownCode) on delete cascade,

constraint pickupCodePK primary key (pickupCode))

--Создание таблицы с заказами

create table CustomerOrder

(customerOrderCode int not null identity,

orderNumber int not null,

issueDate date not null,

generalSum float not null,

deliveryCost float null,

constraint customerOrderCodePK primary key (customerOrderCode),

deliveryCode int not null,

constraint deliveryCodeFK foreign key (deliveryCode) references Delivery (deliveryCode) on delete cascade,

basketCode int not null,

constraint basketCodeFK foreign key (basketCode) references Basket (basketCode) on delete cascade,

customerCode int not null,

constraint customerCodeFK foreign key (customerCode) references Customer (customerCode) on delete cascade,

employeeCode int not null,

constraint employeeCodeFK foreign key (employeeCode) references Employee (employeeCode) on delete cascade,

paymentCode int not null,

constraint paymentCodeFK foreign key (paymentCode) references Payment (paymentCode) on delete cascade,

pickupCode int null,

constraint pickupCodeFK foreign key (pickupCode) references Pickup (pickupCode) on delete no action)

«AddNewProductWindow.xaml»

<UserControl x:Class="KeraminStore.UI.Windows.AddNewProductWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KeraminStore.UI.Windows"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

Height="730" Width="1302" Background="#FFEAEAEA">

<Grid>

<Label Content="Добавление нового изделия" FontSize="30" FontWeight="Bold" FontFamily="Helvetica" Margin="210 30 0 0"/>

<Label Content="Заполните указанные ниже поля" FontSize="20" FontWeight="Light" FontFamily="Helvetica" Margin="210 75 0 0"/>

<Grid Width="300" Height="450" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="210 133 0 0">

<StackPanel Orientation="Horizontal" Margin="0 20 0 0">

<TextBox x:Name="productNameField" materialDesign:HintAssist.Hint="Наименование" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 84 0 0">

<ComboBox x:Name="productTypeField" MaxDropDownHeight="120" materialDesign:HintAssist.Hint="Тип" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintComboBox}" SelectionChanged="productTypeField\_SelectionChanged"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 147 0 0">

<ComboBox x:Name="productCollection" MaxDropDownHeight="120" materialDesign:HintAssist.Hint="Коллекция" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintComboBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 210 0 0">

<TextBox x:Name="productLenght" materialDesign:HintAssist.Hint="Длина, мм" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="100" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

<TextBox x:Name="productWidth" materialDesign:HintAssist.Hint="Ширина, мм" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="115" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 273 0 0">

<ComboBox x:Name="colorField" MaxDropDownHeight="120" materialDesign:HintAssist.Hint="Цвет" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintComboBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 336 0 0">

<ComboBox x:Name="productSurfaceField" MaxDropDownHeight="120" materialDesign:HintAssist.Hint="Поверхность" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintComboBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 400 0 0">

<TextBox x:Name="productArticleField" materialDesign:HintAssist.Hint="Артикул" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

</Grid>

<Grid Width="300" Height="460" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="520 133 0 0">

<StackPanel Orientation="Horizontal" Margin="0 20 0 0">

<TextBox x:Name="countField" materialDesign:HintAssist.Hint="Количество, шт." FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 84 0 0">

<TextBox x:Name="boxWeightField" materialDesign:HintAssist.Hint="Вес ящика, кг" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 147 0 0">

<TextBox x:Name="productCountInBox" materialDesign:HintAssist.Hint="В ящике, шт." FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 210 0 0">

<TextBox x:Name="productCostCountField" materialDesign:HintAssist.Hint="Цена руб./шт." FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}" IsEnabled="False"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0 273 0 0">

<TextBox x:Name="productCostAreaField" materialDesign:HintAssist.Hint="Цена руб./м²" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}" IsEnabled="False"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0,336,0,20">

<TextBox x:Name="productDescriptionField" materialDesign:HintAssist.Hint="Описание" FontFamily="Helvetica" TextWrapping="Wrap" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

</StackPanel>

</Grid>

<Grid Width="250" Height="300" VerticalAlignment="Bottom">

<StackPanel Orientation="Horizontal">

<Button x:Name="AddButton" Width="180" Height="40" VerticalAlignment="Center" HorizontalAlignment="Left" Content="ДОБАВИТЬ" Click="AddButton\_Click" Margin="30 145 0 0"/>

</StackPanel>

</Grid>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="640 153 0 0">

<ComboBox x:Name="productStatusField" materialDesign:HintAssist.Hint="Статус наличия" MaxDropDownHeight="120" FontFamily="Helvetica" FontWeight="Light" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20 -10 0 0" Foreground="Black" Background="Transparent" Width="235" Style="{StaticResource MaterialDesignFloatingHintComboBox}"/>

</StackPanel>

<DockPanel Margin="659 120 0 0" Width="235" Height="400">

<WrapPanel HorizontalAlignment="Center" DockPanel.Dock="Top" Width="235">

<Label FontSize="20" FontWeight="Light" Margin="10 0 0 0" FontFamily="Helvetica" Content="Добавить изображение"/>

<Button x:Name="LoadImgButton" Margin="60 10 0 0" Content="ЗАГРУЗИТЬ" Click="btnOpenFile\_Click"/>

</WrapPanel>

<Image Width="220" Height="300" Name="productImage" Margin="0 3 0 0"/>

</DockPanel>

</Grid>

</UserControl>

«AddNewProductWindow.xaml.cs»

using KeraminStore.Data.Models;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media.Imaging;

using System.Linq;

using System.Data;

namespace KeraminStore.UI.Windows

{

public partial class AddNewProductWindow : UserControl

{

readonly SqlConnection connectionString = new SqlConnection(@"Data Source=(local)\SQLEXPRESS; Initial Catalog=KeraminStore; Integrated Security=True");

private string imagePath = string.Empty;

public AddNewProductWindow()

{

InitializeComponent();

GetSurfaceName();

GetProductType();

GetAvailabilityStatus();

GetCollectionName();

GetColorName();

}

private void GetColorName() //Заполнение комбобокса цветами изделий

{

List<string> colorNames = new List<string>();

connectionString.Open();

string query = @"SELECT colorName FROM Color";

SqlCommand sqlCommand = new SqlCommand(query, connectionString);

SqlDataReader dataReader = sqlCommand.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

colorNames.Add(dataReader["colorName"].ToString());

var newList = from i in colorNames orderby i select i;

colorField.ItemsSource = newList;

}

}

dataReader.Close();

connectionString.Close();

}

private void GetCollectionName() //Заполнение комбобокса коллекциями изделий

{

List<string> collectionNames = new List<string>();

connectionString.Open();

string query = @"SELECT productCollectionName FROM ProductCollection";

SqlCommand sqlCommand = new SqlCommand(query, connectionString);

SqlDataReader dataReader = sqlCommand.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

collectionNames.Add(dataReader["productCollectionName"].ToString());

var newList = from i in collectionNames orderby i select i;

productCollection.ItemsSource = newList;

}

}

dataReader.Close();

connectionString.Close();

}

private void GetAvailabilityStatus() //Заполнение комбобокса статусами наличия изделий

{

List<string> availabilityStatusNames = new List<string>();

connectionString.Open();

string query = @"SELECT availabilityStatusName FROM AvailabilityStatus";

SqlCommand sqlCommand = new SqlCommand(query, connectionString);

SqlDataReader dataReader = sqlCommand.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

availabilityStatusNames.Add(dataReader["availabilityStatusName"].ToString());

var newList = from i in availabilityStatusNames orderby i select i;

productStatusField.ItemsSource = newList;

}

}

dataReader.Close();

connectionString.Close();

}

private void GetProductType() //Заполнение комбобокса типами изделий

{

List<string> typeNames = new List<string>();

connectionString.Open();

string query = @"SELECT productTypeName FROM ProductType";

SqlCommand sqlCommand = new SqlCommand(query, connectionString);

SqlDataReader dataReader = sqlCommand.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

typeNames.Add(dataReader["productTypeName"].ToString());

var newList = from i in typeNames orderby i select i;

productTypeField.ItemsSource = newList;

}

}

dataReader.Close();

connectionString.Close();

}

private void GetSurfaceName() //Заполнение комбобокса поверхностями изделий

{

List<string> surfaceNames = new List<string>();

connectionString.Open();

string query = @"SELECT surfaceName FROM Surface";

SqlCommand sqlCommand = new SqlCommand(query, connectionString);

SqlDataReader dataReader = sqlCommand.ExecuteReader();

if (dataReader.HasRows)

{

while (dataReader.Read())

{

surfaceNames.Add(dataReader["surfaceName"].ToString());

var newList = from i in surfaceNames orderby i select i;

productSurfaceField.ItemsSource = newList;

}

}

dataReader.Close();

connectionString.Close();

}

private void AddButton\_Click(object sender, RoutedEventArgs e)

{

//Проверки вводимых данных на корректность (начало)

if (productNameField.Text != Product.CheckProductName(productNameField.Text, "Наименование изделия не может быть пустым.", "Наименование изделия содержит недопустимые символы.", "Длина наименования изделия может составлять 10-100 символов."))

{

MessageBox.Show(Product.CheckProductName(productNameField.Text, "Наименование изделия не может быть пустым.", "Наименование изделия содержит недопустимые символы.", "Длина наименования изделия может составлять 10-100 символов."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productTypeField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать тип изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productCollection.Text == string.Empty)

{

MessageBox.Show("Необходимо указать коллекцию, к которой относится изделие.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productLenght.Text != Product.CheckProductLenght(productLenght.Text, "Вы не указали длину изделия.", "Длина изделия может составлять от 98 до 900 мм.", "Вы указали неверную длину изделия."))

{

MessageBox.Show(Product.CheckProductLenght(productLenght.Text, "Вы не указали длину изделия.", "Длина изделия может составлять от 98 до 900 мм.", "Вы указали неверную длину изделия."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productWidth.Text != Product.CheckProductWidth(productWidth.Text, "Вы не указали ширину изделия.", "Ширина изделия может составлять от 9 до 400 мм.", "Вы указали неверную ширину изделия."))

{

MessageBox.Show(Product.CheckProductWidth(productWidth.Text, "Вы не указали ширину изделия.", "Ширина изделия может составлять от 9 до 400 мм.", "Вы указали неверную ширину изделия."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (colorField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать цвет изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productSurfaceField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать поверхность изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productArticleField.Text != Product.CheckProductArticle(productArticleField.Text, "Вы не указали артикул изделия.", "Артикул должен быть указан в формате YYY12345678."))

{

MessageBox.Show(Product.CheckProductArticle(productArticleField.Text, "Вы не указали артикул изделия.", "Артикул должен быть указан в формате YYY12345678."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (countField.Text != Product.CheckCountInBox(countField.Text, "Вы не указали количество изделий.", "Количество изделий не может быть отрицательным.", "Вы указали неверное количество изделий."))

{

MessageBox.Show(Product.CheckCountInBox(countField.Text, "Вы не указали количество изделий.", "Количество изделий не может быть отрицательным.", "Вы указали неверное количество изделий."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (boxWeightField.Text == string.Empty)

{

MessageBox.Show("Вы не указали вес ящика.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (boxWeightField.Text != Product.CheckProductCostOrWeight(boxWeightField.Text, "Вес ящика не может быть отрицательным.", "Вы указали недопустимые символы в весе ящика."))

{

MessageBox.Show(Product.CheckProductCostOrWeight(boxWeightField.Text, "Вес ящика не может быть отрицательным.", "Вы указали недопустимые символы в весе ящика."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productCountInBox.Text != Product.CheckCountInBox(productCountInBox.Text, "Вы не указали количество изделий в ящике.", "Количество изделий в ящике не может быть отрицательным.", "Вы указали неверное количество изделий."))

{

MessageBox.Show(Product.CheckCountInBox(productCountInBox.Text, "Вы не указали количество изделий в ящике.", "Количество изделий в ящике не может быть отрицательным.", "Вы указали неверное количество изделий."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productCostAreaField.Text == string.Empty && productCostCountField.Text == string.Empty)

{

MessageBox.Show("Вы должны указать стоимость изделия за штуку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (productCostCountField.Text != string.Empty)

{

if (productCostCountField.Text != Product.CheckProductCostOrWeight(productCostCountField.Text, "Стоимость изделия не может быть отрицательной.", "Вы указали недопустимые символы в стоимости изделия."))

{

MessageBox.Show(Product.CheckProductCostOrWeight(productCostCountField.Text, "Стоимость изделия не может быть отрицательной.", "Вы указали недопустимые символы в стоимости изделия."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

if (productStatusField.Text == string.Empty)

{

MessageBox.Show("Необходимо указать статус наличия изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productDescriptionField.Text != Product.CheckProductDescription(productDescriptionField.Text, "Длина описания не может быть более 300 символов."))

{

MessageBox.Show(Product.CheckProductDescription(productDescriptionField.Text, "Длина описания не может быть более 300 символов."), "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (productImage.Source == null)

{

MessageBox.Show("Необходимо загрузить изображение изделия.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

//Проверки вводимых данных на корректность(конец)

int typeCode = 0;

int collectionCode = 0;

int surfaceCode = 0;

int statusCode = 0;

int colorCode = 0;

string selectCodesQuery = "SELECT productCollectionCode, availabilityStatusCode, productTypeCode, surfaceCode, colorCode FROM Surface, ProductType, AvailabilityStatus, ProductCollection, Color " +

"WHERE productCollectionName = '" + productCollection.Text + "' AND availabilityStatusName = '" + productStatusField.Text + "' AND productTypeName = '" + productTypeField.Text + "'" +

"AND surfaceName = '" + productSurfaceField.Text + "' AND colorName = '" + colorField.Text + "'";

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(selectCodesQuery, connectionString)) //Получение кодов коллекции, статуса наличия, типа и повехности изделия для последующей работы с ними

{

DataTable table = new DataTable();

dataAdapter.Fill(table);

if (table.Rows.Count > 0)

{

typeCode = int.Parse(table.Rows[0]["productTypeCode"].ToString());

collectionCode = int.Parse(table.Rows[0]["productCollectionCode"].ToString());

surfaceCode = int.Parse(table.Rows[0]["surfaceCode"].ToString());

statusCode = int.Parse(table.Rows[0]["availabilityStatusCode"].ToString());

colorCode = int.Parse(table.Rows[0]["colorCode"].ToString());

}

}

string selectUniqProductQuery = "SELECT \* FROM Product " +

"JOIN ProductCollection ON Product.productCollectionCode = ProductCollection.productCollectionCode " +

"JOIN ProductType ON Product.productTypeCode = ProductType.productTypeCode " +

"JOIN Color ON Product.colorCode = Color.colorCode " +

"WHERE productName= '" + productNameField.Text + "' AND productCollectionName = '" + productCollection.Text + "' AND productTypeName = '" + productTypeField.Text + "'";

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(selectUniqProductQuery, connectionString)) //Проверка уникальности товара, то есть его наличия в базе

{

DataTable table = new DataTable();

dataAdapter.Fill(table);

if (table.Rows.Count > 0) //В случае, если таковой имеется, отобразится соответсвующее предупреждение

{

MessageBox.Show("Это изделие уже имеется в базе.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else if (table.Rows.Count == 0)

{

string selectUniqArticleQuery = "SELECT \* FROM Product WHERE productArticle= '" + productArticleField.Text + "'";

using (SqlDataAdapter articleAdapter = new SqlDataAdapter(selectUniqArticleQuery, connectionString))

{

DataTable articleTable = new DataTable();

articleAdapter.Fill(articleTable);

if (articleTable.Rows.Count > 0) //Проверка уникальности артикула

{

MessageBox.Show("Этот артикул принадлежит другому изделию. Проверьте, не допустили ли вы ошибку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

string newImagePath = imagePath.Replace("KeraminStore\\bin\\Debug\\", "\*");

string[] imgPathArray = newImagePath.Split('\*');

SqlCommand cmd = new SqlCommand(); //Добавление изделия в базу

cmd.CommandType = CommandType.Text;

cmd.CommandText = "INSERT Product (productName, productArticle, productWidth, productLenght, productCount, productBoxWeight, productCountInBox, productCostCount, productCostArea, productImage, productDescription, productCollectionCode, availabilityStatusCode, productTypeCode, surfaceCode, colorCode) " +

"VALUES (@name, @article, @width, @lenght, @count, @boxWeight, @countInBox, @countCost, @areaCost, @image, @description, @collectionCode, @statusCode, @typeCode, @surfaceCode, @colorCode)";

cmd.Parameters.Add("@name", SqlDbType.VarChar).Value = productNameField.Text;

cmd.Parameters.Add("@article", SqlDbType.VarChar).Value = productArticleField.Text;

cmd.Parameters.Add("@width", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productWidth.Text), 2);

cmd.Parameters.Add("@lenght", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productLenght.Text), 2);

cmd.Parameters.Add("@count", SqlDbType.Int).Value = Convert.ToInt32(countField.Text);

cmd.Parameters.Add("@boxWeight", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(boxWeightField.Text), 2);

cmd.Parameters.Add("@countInBox", SqlDbType.Int).Value = Convert.ToInt32(productCountInBox.Text);

cmd.Parameters.Add("@image", SqlDbType.VarChar).Value = imgPathArray[1].ToString();

cmd.Parameters.Add("@description", SqlDbType.VarChar).Value = productDescriptionField.Text;

cmd.Parameters.Add("@collectionCode", SqlDbType.Int).Value = collectionCode;

cmd.Parameters.Add("@statusCode", SqlDbType.Int).Value = statusCode;

cmd.Parameters.Add("@typeCode", SqlDbType.Int).Value = typeCode;

cmd.Parameters.Add("@surfaceCode", SqlDbType.Int).Value = surfaceCode;

cmd.Parameters.Add("@colorCode", SqlDbType.Int).Value = colorCode;

if (productCostAreaField.IsEnabled == false)

{

cmd.Parameters.Add("@countCost", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productCostCountField.Text), 2);

cmd.Parameters.Add("@areaCost", SqlDbType.Float).Value = DBNull.Value;

}

else

{

cmd.Parameters.Add("@countCost", SqlDbType.Float).Value = DBNull.Value;

cmd.Parameters.Add("@areaCost", SqlDbType.Float).Value = Math.Round(Convert.ToDouble(productCostAreaField.Text), 2);

}

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

connectionString.Close();

MessageBox.Show("Добавление изделия прошло успешно.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

ClearFields();

}

}

}

private void ClearFields() //Очистка содержимого всех полей

{

productNameField.Clear();

productTypeField.SelectedIndex = -1;

productCollection.SelectedIndex = -1;

productLenght.Clear();

productWidth.Clear();

colorField.SelectedIndex = -1;

productSurfaceField.SelectedIndex = -1;

productArticleField.Clear();

boxWeightField.Clear();

productCountInBox.Clear();

productCostCountField.Clear();

productCostAreaField.Clear();

countField.Clear();

productDescriptionField.Clear();

productStatusField.SelectedIndex = -1;

productImage.Source = null;

}

private void btnOpenFile\_Click(object sender, RoutedEventArgs e) //Открытие файла

{

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

string filePath = openFileDialog.FileName;

if ((filePath[filePath.Length - 1] == 'g' && filePath[filePath.Length - 2] == 'p' && filePath[filePath.Length - 3] == 'j') || (filePath[filePath.Length - 1] == 'g' && filePath[filePath.Length - 2] == 'n' && filePath[filePath.Length - 3] == 'p')) //Проверка типа файла

{

productImage.Source = new BitmapImage(new Uri(@"" + filePath + "")); //Загрузка пути изображения

imagePath = filePath;

}

else

{

MessageBox.Show("Вы выбрали неверный файл.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

}

private void productTypeField\_SelectionChanged(object sender, SelectionChangedEventArgs e) //Включение/отключение возможности взаимодействия с полями

{

if (productTypeField.SelectedIndex != -1)

{

if (productTypeField.SelectedItem.ToString() == "Настенная плитка" || productTypeField.SelectedItem.ToString() == "Напольная плитка" || productTypeField.SelectedItem.ToString() == "Бордюр") //Проверка типа изделия

{

productCostCountField.IsEnabled = false;

productCostCountField.Clear();

productCostAreaField.IsEnabled = true;

}

else

{

productCostCountField.IsEnabled = true;

productCostAreaField.Clear();

productCostAreaField.IsEnabled = false;

}

}

}

}

}

«EmployeesWindow.xaml»

<UserControl x:Class="KeraminStore.UI.Windows.EmployeesWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KeraminStore.UI.Windows"

mc:Ignorable="d"

Width="1302" Height="730" BorderThickness="1" Foreground="White" Background="#FFEAEAEA">

<UserControl.Resources>

<Style x:Key="ElementStyle" TargetType="TextBlock">

<Setter Property="TextWrapping" Value="Wrap"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="ElementsStyle" TargetType="TextBlock">

<Setter Property="TextWrapping" Value="Wrap"/>

<Setter Property="TextAlignment" Value="Center"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

</UserControl.Resources>

<Grid>

<Label Content="Список сотрудников" FontSize="30" FontWeight="Bold" FontFamily="Helvetica" Margin="210 30 0 0"/>

<Label x:Name="Description" Content="Для удаления/изменения данных о сотруднике выделите строку и нажмите на соответствующую кнопку снизу" FontSize="20" FontWeight="Light" FontFamily="Helvetica" Margin="210 75 0 0"/>

<Grid Width="1302" Height="100" HorizontalAlignment="Left" VerticalAlignment="Bottom">

<Grid.RowDefinitions>

<RowDefinition/>

</Grid.RowDefinitions>

<Button x:Name="ChangeButton" Width="110" Height="40" HorizontalAlignment="Center" VerticalAlignment="Center" Content="ИЗМЕНИТЬ" Margin="0 13 150 0" Click="ChangeButton\_Click"/>

<Rectangle Width="1" Height="50" Fill="#FFB2B2B2" Margin="1 13 0 0"/>

<Button x:Name="DeleteButton" Width="110" Height="40" HorizontalAlignment="Center" VerticalAlignment="Center" Content="УДАЛИТЬ" Margin="150 13 0 0" Click="DeleteButton\_Click"/>

</Grid>

<DataGrid x:Name="EmployeesInfoGrid" Width="995" Height="532" HorizontalAlignment="Center" Margin="25,125,27,88" BorderThickness="1" AutoGenerateColumns="False" ScrollViewer.HorizontalScrollBarVisibility="Hidden" CanUserAddRows="False" CanUserDeleteRows="False" IsReadOnly="True" GridLinesVisibility="All">

<DataGrid.Columns>

<DataGridTextColumn ElementStyle="{StaticResource ElementStyle}" Binding="{Binding employeeInfo}" Header="ФИО сотрудника" Width="420" FontFamily="Helvetica" FontWeight="Light" FontSize="16"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding employeeSurname}" Width="0" FontFamily="Helvetica" FontWeight="Light" FontSize="16" Visibility="Hidden"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding employeeName}" Width="0" FontFamily="Helvetica" FontWeight="Light" FontSize="16" Visibility="Hidden"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding employeePatronymic}" Width="0" FontFamily="Helvetica" FontWeight="Light" FontSize="16" Visibility="Hidden"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding employeeDateOfBirth, StringFormat=\{0:dd/MM/yyyy\}}" Header="Дата рождения" Width="130" FontFamily="Helvetica" FontWeight="Light" FontSize="16"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding employeeLogin}" Header="Логин" Width="140" FontFamily="Helvetica" FontWeight="Light" FontSize="16"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding employeePassword}" Header="Пароль" Width="140" FontFamily="Helvetica" FontWeight="Light" FontSize="16"/>

<DataGridTextColumn ElementStyle="{StaticResource ElementsStyle}" Binding="{Binding postName}" Header="Должность" Width="150" FontFamily="Helvetica" FontWeight="Light" FontSize="16"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</UserControl>

«EmployeesWindow.xaml.cs»

using System;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Windows;

using System.Windows.Controls;

namespace KeraminStore.UI.Windows

{

public partial class EmployeesWindow : UserControl

{

readonly SqlConnection connectionString = new SqlConnection(@"Data Source=(local)\SQLEXPRESS; Initial Catalog=KeraminStore; Integrated Security=True");

public EmployeesWindow()

{

InitializeComponent();

connectionString.Open(); //Открытие подключения к БД

FillDataGrid();

connectionString.Close(); //Закрытие подключения к БД

}

private void FillDataGrid() //Заполнение таблицы данными

{

StreamReader file = new StreamReader("UserCode.txt"); //Получение кода сотрудника

int userCode = Convert.ToInt32(file.ReadLine());

file.Close();

string employeesInfoQuery = "SELECT CONCAT(employeeSurname, ' ', employeeName, ' ', employeePatronymic) as 'employeeInfo', employeeSurname, employeeName, employeePatronymic, employeeLogin, employeePassword, employeeDateOfBirth, postName " +

"FROM Employee " +

"JOIN Post ON Employee.postCode = Post.postCode WHERE employeeCode != '" + userCode + "'";

DataTable table = new DataTable();

using (SqlCommand cmd = new SqlCommand(employeesInfoQuery, connectionString))

{

using (IDataReader rdr = cmd.ExecuteReader())

{

table.Load(rdr);

}

}

EmployeesInfoGrid.ItemsSource = table.DefaultView;

}

private void ChangeButton\_Click(object sender, RoutedEventArgs e)

{

if (EmployeesInfoGrid.SelectedItem == null)

{

MessageBox.Show("Невозможно изменить информацию о сотруднике, так как он не был выбран.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

DataRowView employeeInfo = (DataRowView)EmployeesInfoGrid.SelectedItems[0];

ChangeEmployeeInfoWindow changeEmployeeInfo = new ChangeEmployeeInfoWindow();

string selectEmployeeInfoQuery = "SELECT employeeCode FROM Employee WHERE employeeLogin = '" + employeeInfo["employeeLogin"].ToString() + "'";

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(selectEmployeeInfoQuery, connectionString))

{

DataTable table = new DataTable();

dataAdapter.Fill(table);

if (table.Rows.Count > 0)

{

StreamWriter employeeCode = new StreamWriter("ChangingEmployee.txt");

employeeCode.Write(table.Rows[0]["employeeCode"].ToString());

employeeCode.Close();

}

}

//Присваивание данных выбранной строки полям в окне для изменения данных

changeEmployeeInfo.loginField.Text = employeeInfo["employeeLogin"].ToString();

changeEmployeeInfo.passwordField.Text = employeeInfo["employeePassword"].ToString();

changeEmployeeInfo.nameField.Text = employeeInfo["employeeName"].ToString();

changeEmployeeInfo.surnameField.Text = employeeInfo["employeeSurname"].ToString();

changeEmployeeInfo.patronymicField.Text = employeeInfo["employeePatronymic"].ToString();

changeEmployeeInfo.birthdayDateField.Text = employeeInfo["employeeDateOfBirth"].ToString();

changeEmployeeInfo.postField.Text = employeeInfo["postName"].ToString();

changeEmployeeInfo.ShowDialog();

EmployeesInfoGrid.ItemsSource = null;

EmployeesInfoGrid.Items.Refresh();

connectionString.Open();

FillDataGrid();

connectionString.Close();

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

if (EmployeesInfoGrid.SelectedItem == null) //Проверка выбора сотрудника, который будет удален

{

MessageBox.Show("Невозможно удалить информацию о сотруднике, так как он не был выбран.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

DataRowView employeeInfo = (DataRowView)EmployeesInfoGrid.SelectedItems[0]; //Получание информации о сотруднике из таблицы

SqlCommand cmd = new SqlCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "DELETE FROM Employee WHERE employeeLogin = @login"; //Запрос на удаление выбранного сотрудника

cmd.Parameters.Add("@login", SqlDbType.VarChar).Value = employeeInfo["employeeLogin"].ToString();

cmd.Connection = connectionString;

connectionString.Open();

cmd.ExecuteNonQuery();

FillDataGrid();

connectionString.Close();

MessageBox.Show("Удаление успешно выполнено.", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

# Приложение Б

(Справочное)

Выходные документы

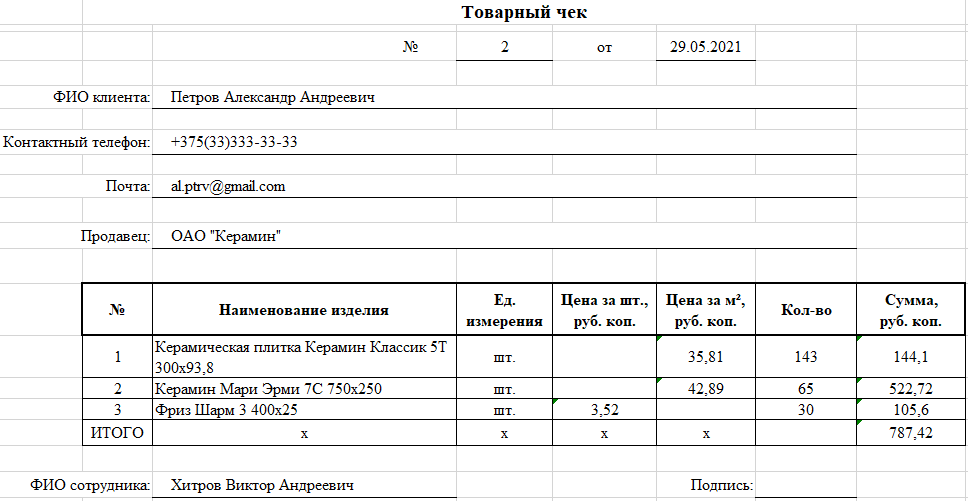


Рисунок Б.1 – Чек для физического лица

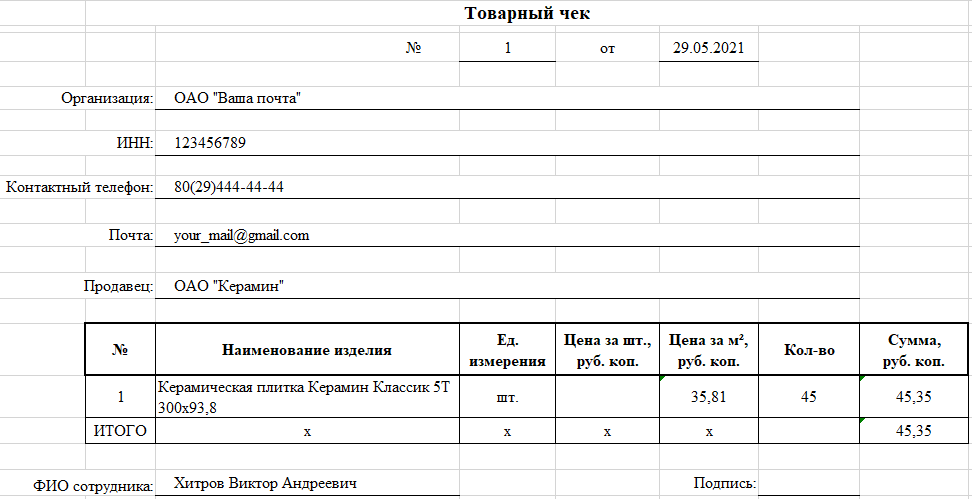


Рисунок Б.2 – Чек для юридического лица

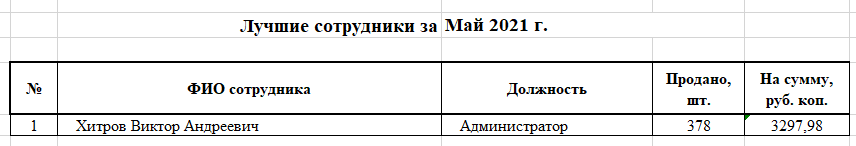


Рисунок Б.3 – Документ со статистикой о продуктивности сотрудников за май 2021 года

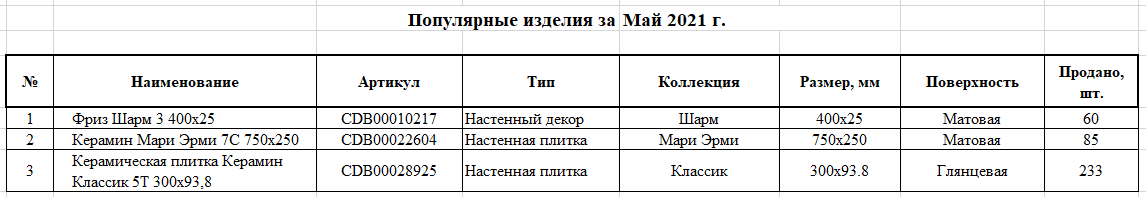


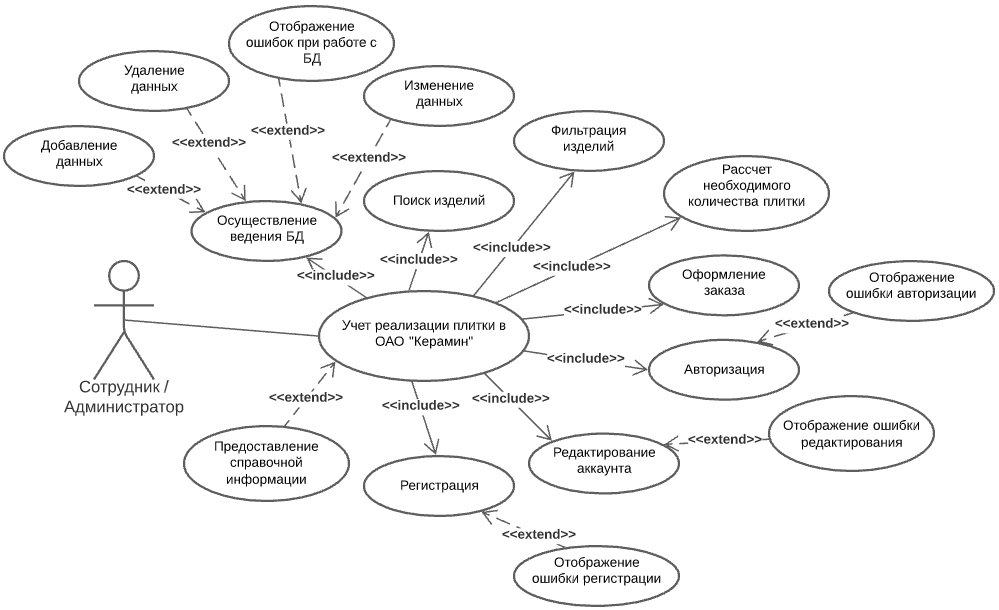
Рисунок Б.4 – Документ со статистикой о популярности изделий за май 2021 года



Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

ДП Т.791084.401 ПЗ

*Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин»*

Диаграмма вариантов использования

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Ритвинский* *А.Н.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 1

Н.Контр.

КБП

Реценз.

*Якимович К.О.*

Утверд.

Т.Контр.

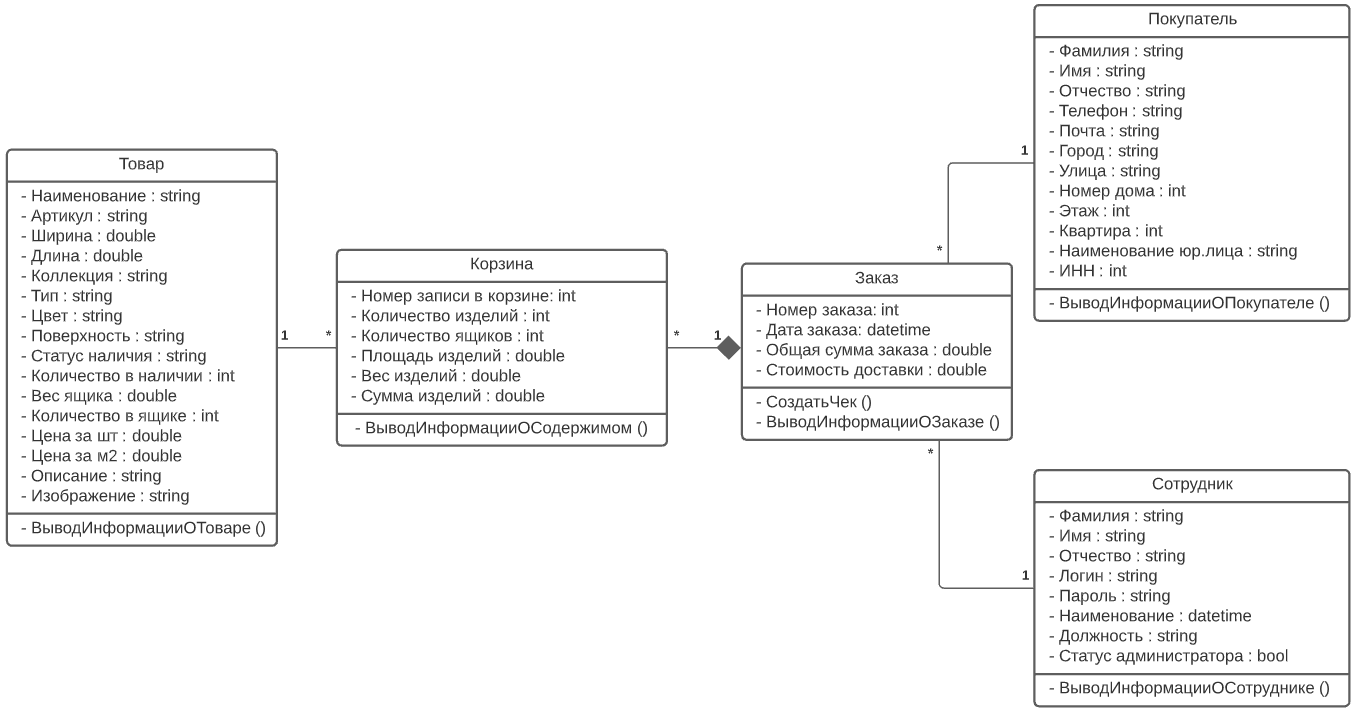
*Багласова Т.Г.*



Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

ДП Т.791084.401 ПЗ

*Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин»*

Диаграмма классов

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Ритвинский* *А.Н.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Лист 2

Н.Контр.

Листов 5

Багласова Е.В.

КБП

Реценз.

*Багласова Т.Г.*

*Якимович К.О.*

Утверд.

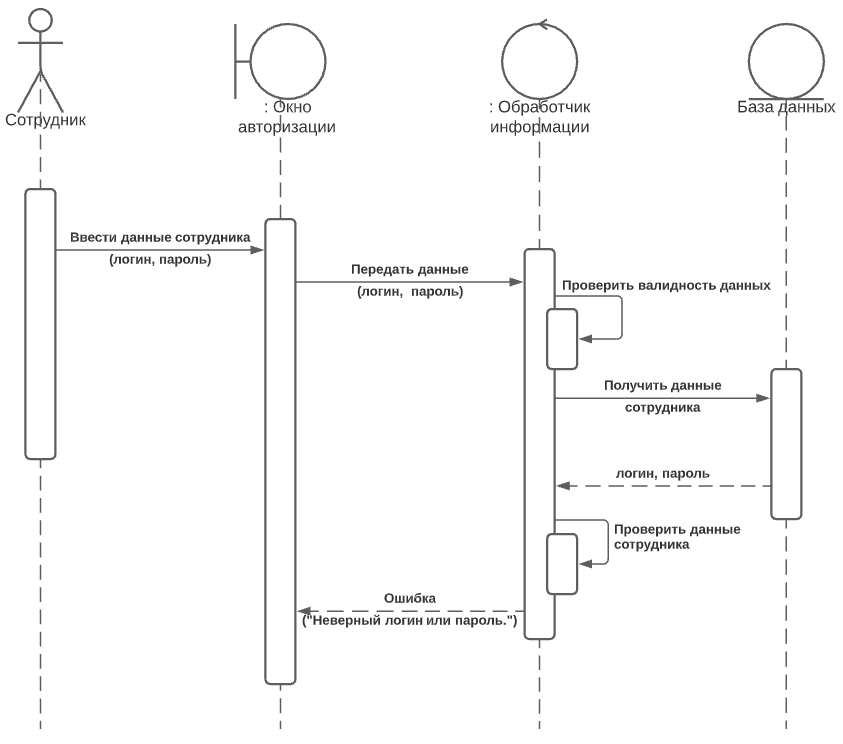
Т.Контр.



Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

ДП Т.791084.401 ПЗ

*Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин»*

Диаграмма последовательности для функции «Авторизация»

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Ритвинский* *А.Н.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Лист 3

Н.Контр.

Листов 5

КБП

Реценз.

*Багласова Т.Г.*

*Якимович К.О.*

Утверд.

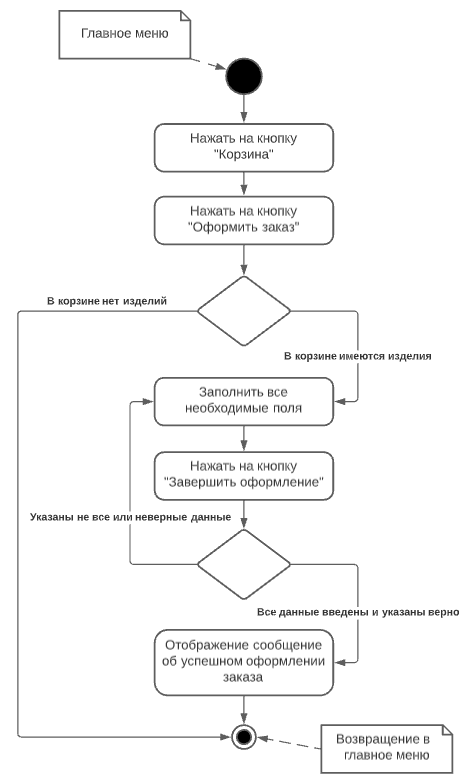
Т.Контр.



Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

ДП Т.791084.401 ПЗ

*Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин»*

Диаграмма деятельности для функции «Оформление заказа»

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Ритвинский* *А.Н.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 4

Н.Контр.

КБП

Реценз.

*Якимович К.О.*

*Багласова Т.Г.*

Утверд.

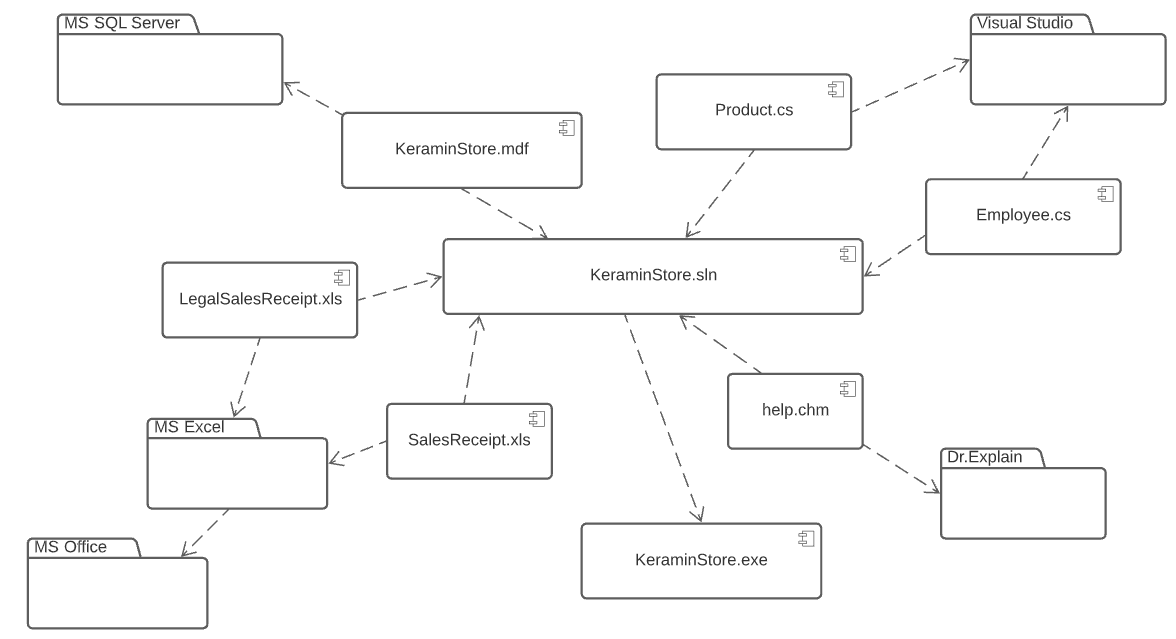
Т.Контр.



Подп. и дата

Инв.№ дубл.

Взам.инв.№



Подп. и дата

ДП Т.791084.401 ПЗ

*Разработка программного средства для учета реализации плитки в Открытом акционерном обществе «Керамин»*

Диаграмма компонентов

Масса

Лит.

Масштаб

№ докум.

Подпись

Дата

Изм.

Лист

*Ритвинский* *А.Н.*

Разраб.

У

Багласова Е.В.

Провер.

Инв.№ подл.

Багласова Е.В.

Листов 5

Лист 5

Н.Контр.

КБП

Реценз.

*Якимович К.О.*

*Багласова Т.Г.*

Утверд.

Т.Контр.