**Государственное бюджетное профессиональное образовательное учреждение**

**«Нижегородский Губернский колледж»**

Методическая комиссия «Информатика и вычислительная техника»

Допущен к защите:

преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Мухина,

\_\_\_\_\_\_\_\_\_\_\_\_ Е.П. Голубева,

«21»\_\_марта\_\_2025 г.

**ОТЧЕТ** **ПО**

**УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Руководитель: \_\_\_\_\_\_\_\_\_\_ Мухина Л.В., Голубева Е.П. 21.03.2025 г.

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Касаткина А.Е. 21.03.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

**Содержание**

[**Введение** 3](#_Toc192876716)

[**1.** **Разработка программных модулей** 4](#_Toc192876717)

[**1.1** **Импорт данных. Разработка базы данных** 4](#_Toc192876718)

[**1.2** **Avalonia** 4](#_Toc192876719)

[**1.3** **Тестирование** 4](#_Toc192876720)

[**1.4** **Диаграммы** 4](#_Toc192876721)

[**2.** **Поддержка и тестирование программных модулей** 4](#_Toc192876722)

[**2.1.** **База данных и импорт** 4](#_Toc192876723)

[**2.2.** **Avalonia** 4](#_Toc192876724)

[**2.3.** **Тестирование** 4](#_Toc192876725)

[**Заключение** 9](#_Toc192876726)

[**Приложение А** 11](#_Toc192876727)

[**Приложение Б** 11](#_Toc192876728)

# **Введение**

Учебная практика по теме "ПМ.01 Разработка модулей программного обеспечения для компьютерных систем" направлена на закрепление знаний и получение практического опыта в программировании. В современном мире компьютерные программы используются повсюду, поэтому важно уметь их создавать, улучшать и тестировать.

Во время практики происходит освоение разработки отдельных частей программ, работа с кодом, поиск и исправление ошибок. Используются различные инструменты, такие как среды программирования, базы данных и системы контроля версий.

Практика помогает лучше понять процесс создания программ и подготовиться к будущей профессиональной деятельности.

Цели и задачи учебной практики

Цель практики:

Освоение разработки программных модулей, применение полученных знаний на практике и изучение основных инструментов программирования.

Задачи практики:

Закрепление знаний, полученных на занятиях.

Освоение процесса создания программных модулей.

Изучение среды программирования и вспомогательных инструментов.

Освоение тестирования кода и поиска ошибок.

Ознакомление с правилами написания понятного и удобного кода.

## **Разработка программных модулей**

## **Импорт данных. Разработка базы данных**

Была создана база данных 43P\_Kasatkina\_UP (Листинг 1, Приложение А) в PostgreSQL. Она настроена для хранения информации о партнёрах, материалах, продукции и продажах, с поддержкой русской локализации.

Структура базы данных

База данных включает следующие таблицы:

Таблица материалов – содержит данные о типах материалов и их проценте дефекта.

Таблица типов партнёров – хранит список организационно-правовых форм, таких как ООО, ЗАО и другие.

Таблица типов продукции – описывает категории продукции, а также коэффициенты, используемые в расчётах.

Таблица продукции – содержит информацию о товарах, включая артикулы, наименования, описания, минимальные цены и размеры.

Таблица партнёров – хранит данные о компаниях-партнёрах, их юридических адресах, руководителях, контактах и рейтинге.

Таблица продаж – фиксирует информацию о том, какие товары, в каком количестве и когда были проданы партнёрам.

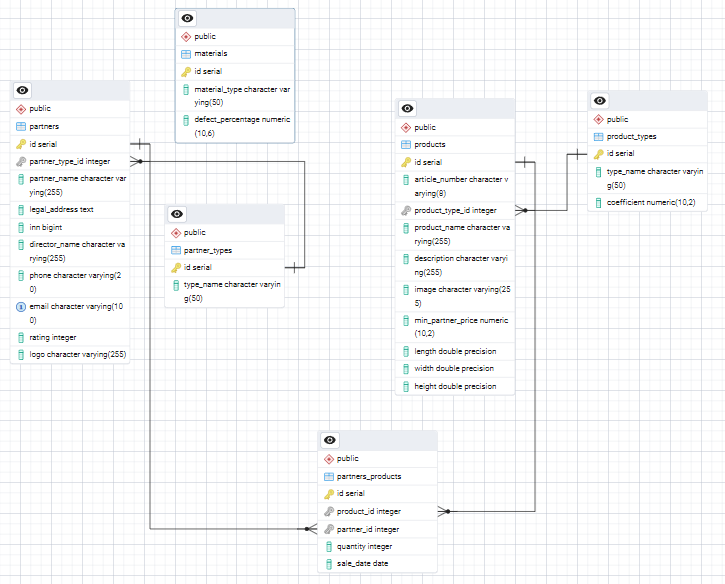


Рисунок №1 - ER Диаграмма

Импорт данных из Excel

Был выполнен импорт данных из Excel в PostgreSQL.

## **Avalonia**

Приложение "Мастер Пол" разработано для управления партнёрами и их продажами. Оно предоставляет удобный интерфейс для добавления, редактирования и просмотра информации о партнёрах, их скидках, рейтинге, контактных данных и истории продаж.

Основные функции приложения:

1. Просмотр списка партнёров – список партнёров с основными данными, такими как название, тип, директор, телефон, рейтинг и скидка.
2. Добавление и редактирование партнёров – возможность вносить новые записи или изменять существующие данные.
3. Просмотр истории продаж – отображение списка товаров, приобретённых каждым партнёром, с указанием количества и даты покупки.
4. Навигация между экранами – кнопки позволяют переключаться между главной страницей, формой редактирования и экраном истории продаж.

Приложение выполнено на AvaloniaUI, использует MVVM-подход, что обеспечивает чёткое разделение логики и интерфейса.

Просмотр списка партнёров

Страница Show (Листинг 2, Приложение А) предназначена для отображения списка партнеров компании, а также предоставления возможностей для добавления, редактирования и просмотра истории продаж партнера.

Описание структуры и элементов интерфейса

Общий макет страницы:

Используется Grid с тремя строками:

Заголовок страницы.

Кнопка "Добавить" для перехода к форме создания нового партнера.

Список партнеров, отображаемый в ListBox с прокруткой.

Элементы интерфейса:

Заголовок – текст "Мастер Пол" в центре экрана с зеленым цветом.

Кнопка "Добавить" – расположена под заголовком, позволяет перейти на страницу добавления нового партнера.

Список партнеров – представлен в виде ListBox, элементы которого содержат следующую информацию:

Тип партнера и название организации.

ФИО директора.

Контактный телефон.

Рейтинг партнера.

Размер скидки.

Кнопка "Перейти к продажам" – расположена внутри каждого элемента списка, ведет на страницу истории продаж партнера.

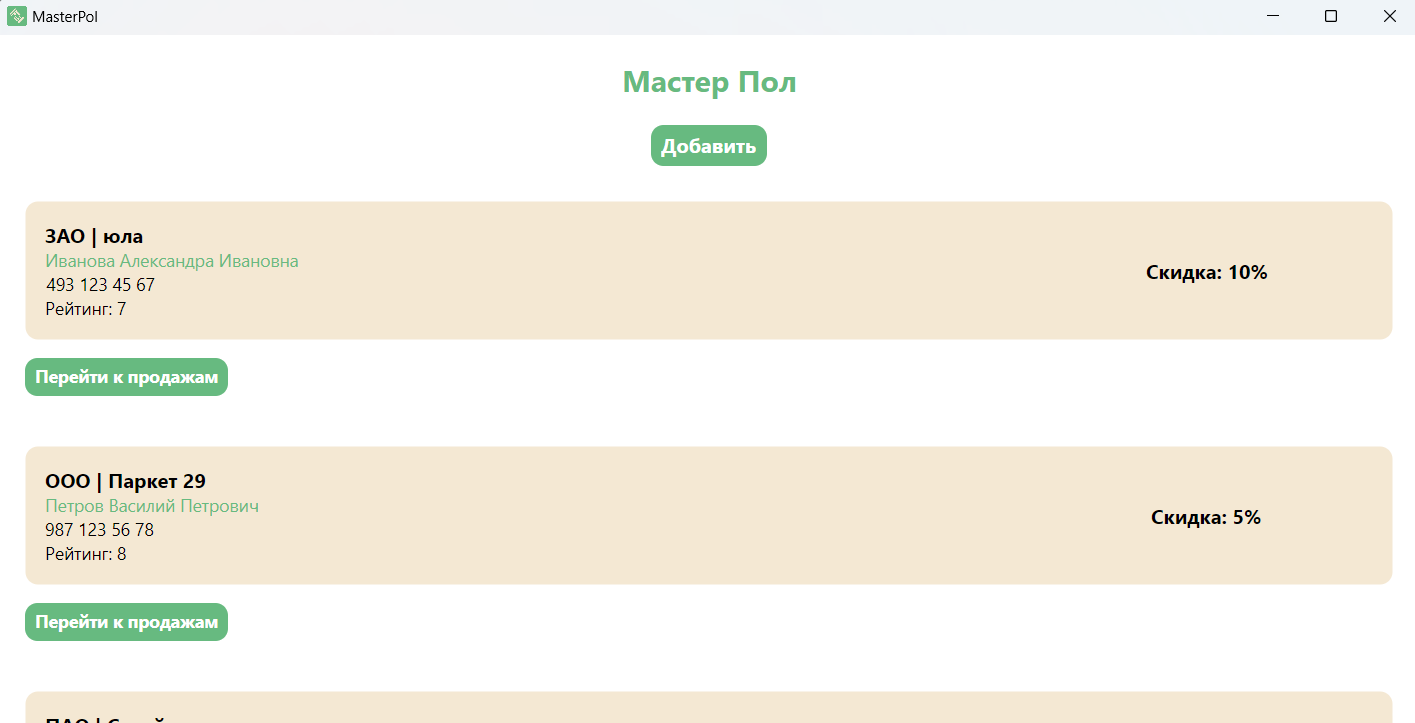


Рисунок №2- Show

Логика работы страницы (Листинг 3, Приложение А)

Загрузка данных:

Данные загружаются в ViewModel из базы данных при инициализации страницы.

Используется Entity Framework с Include, чтобы загрузить данные о партнерах, их типах и проданных товарах.

Функции, доступные пользователю:

Добавление партнера – кнопка "Добавить" открывает форму создания нового партнера.

Редактирование партнера – при нажатии на карточку партнера происходит переход к форме редактирования с передачей ID.

Просмотр истории продаж – кнопка "Перейти к продажам" загружает страницу со списком продаж конкретного партнера.

Добавление и редактирование партнёров

Страница Add (Листпнг 4, Приложение А) предназначена для добавления нового партнера или редактирования данных существующего партнера в системе.

Описание структуры и элементов интерфейса

Общий макет страницы:

Вся информация расположена внутри Border с округленными углами.

Внутри StackPanel содержатся текстовые поля, выпадающий список и кнопки управления.

Элементы интерфейса:

1. Заголовок "Редактировать/Добавить" – располагается в верхней части формы.
2. Текстовые поля для ввода данных:

Наименование – ввод названия организации.

Тип партнера – выбор из списка возможных типов партнеров (ComboBox).

Рейтинг – ввод числового значения рейтинга партнера.

Адрес – юридический адрес компании.

ФИО директора – имя руководителя компании.

Телефон – контактный номер.

Email – электронная почта.

1. Кнопки:

"Сохранить" – отправляет введенные данные в базу.

"Назад" – возвращает пользователя на страницу со списком партнеров.

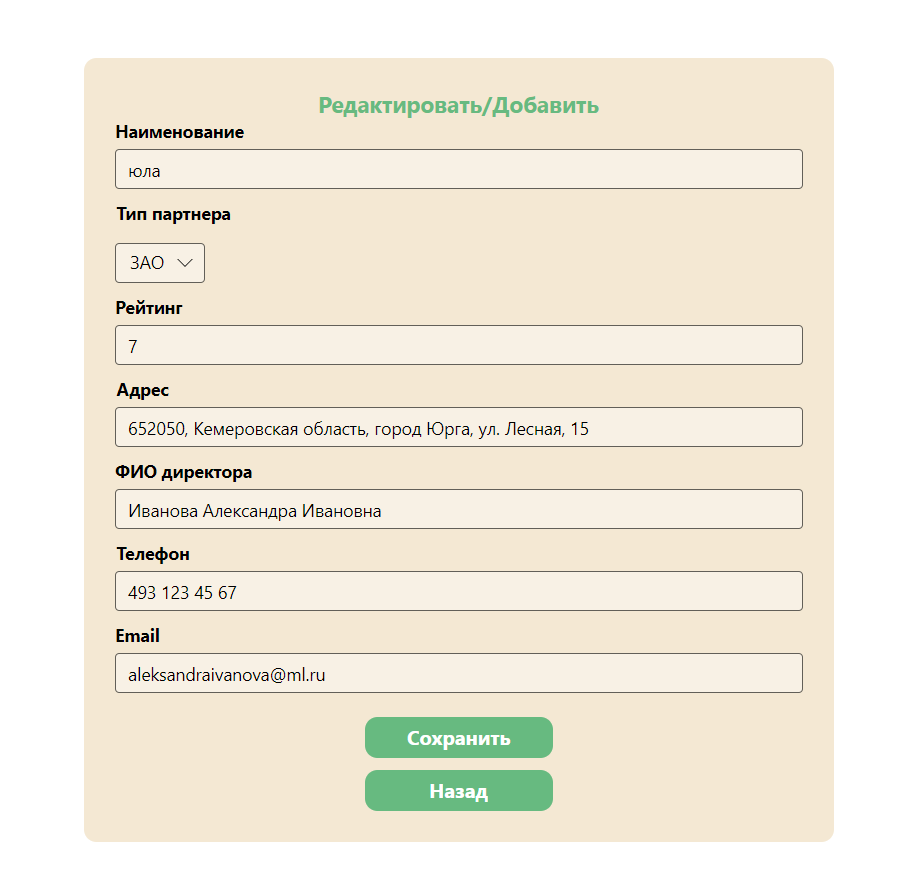


Рисунок №3 - Редактирование

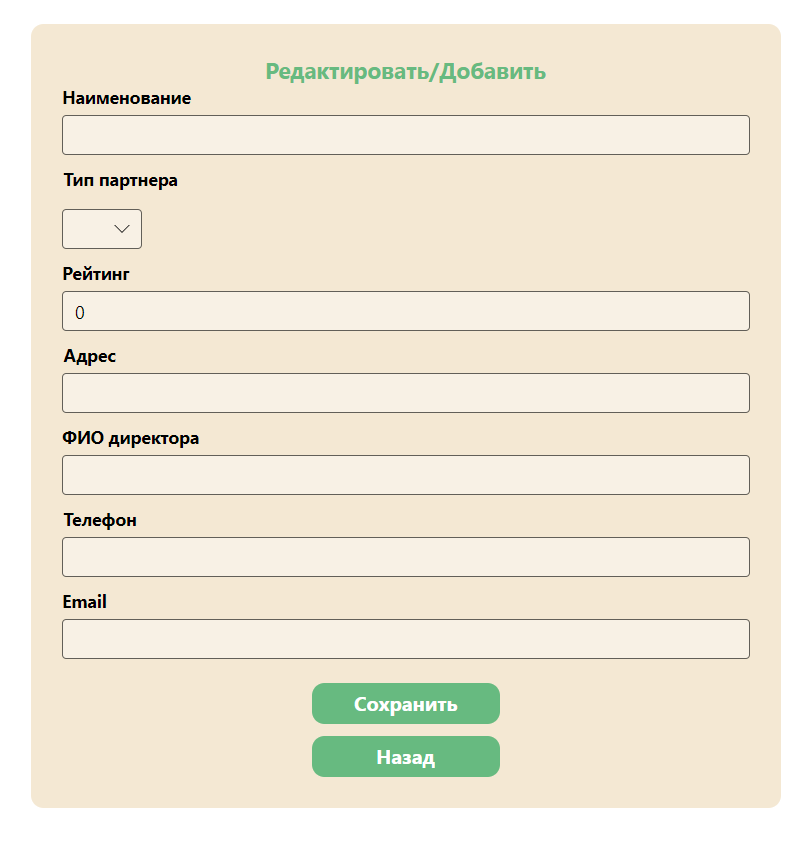


Рисунок №4 - Добавление

3. Логика работы страницы (Листинг 5, Приложение А)

Загрузка данных:

Если пользователь открывает страницу без параметра ID, создается новый пустой объект Partner.

Если передан ID, загружаются данные партнера из базы и подставляются в соответствующие поля.

Проверка введенных данных:

Перед сохранением выполняются проверки:

* Наименование не должно быть пустым.
* Рейтинг не может быть равен нулю.
* Адрес, ФИО директора, телефон и email должны быть заполнены.
* В случае ошибки выводится сообщение пользователю.

Сохранение данных:

* Если у партнера ID = 0, создается новая запись в базе.
* Если ID уже существует, данные обновляются.
* Перед сохранением появляется диалоговое окно подтверждения.
* В случае успешного сохранения выводится уведомление, и происходит переход на страницу списка партнеров.

Выход без сохранения:

Кнопка "Назад" возвращает пользователя к списку партнеров.

Просмотр истории продаж

Страница Sale\_history (Листинг 6, Приложение А) предназначена для отображения истории продаж для конкретного партнера. На этой странице пользователь может увидеть все товары, проданные партнером, а также информацию о количестве и дате продажи.

Описание структуры и элементов интерфейса

Общий макет страницы: Все элементы находятся внутри контейнера Grid, который организует элементы интерфейса по строкам. Страница делится на три строки:

Первая строка содержит заголовок.

Вторая строка — кнопку "Назад".

Третья строка содержит прокручиваемый список с данными.

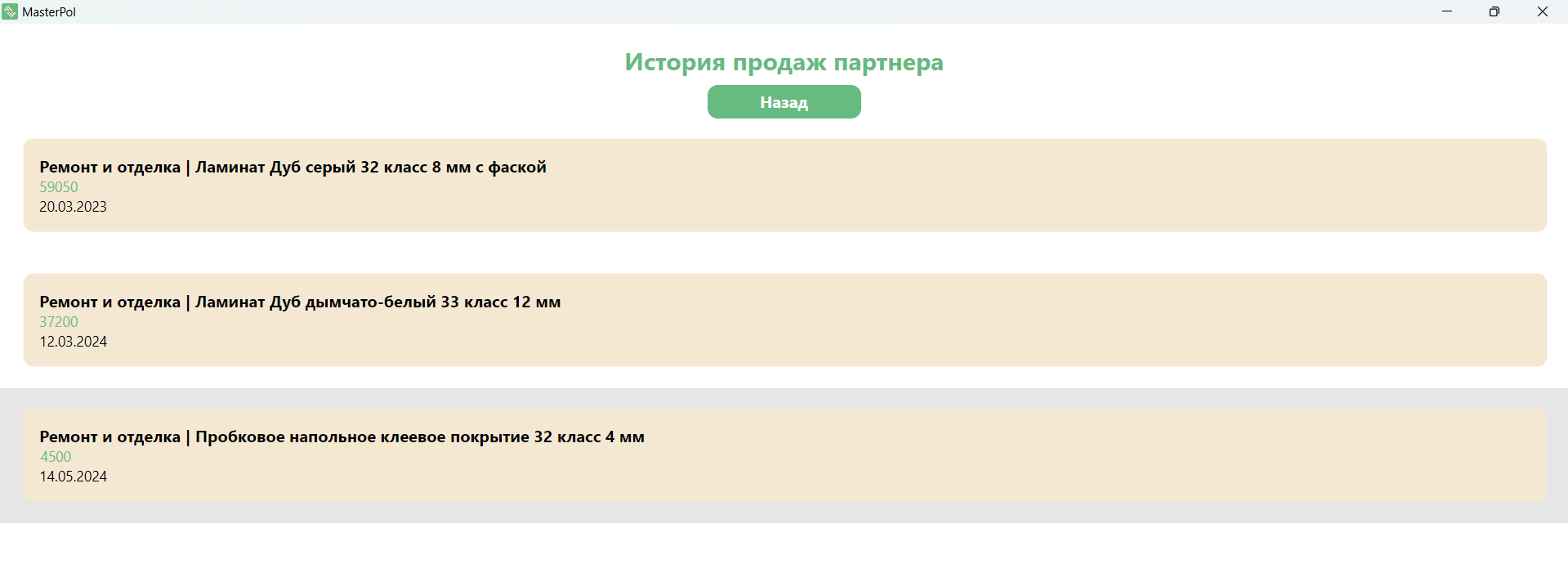


Рисунок №5 – История продаж

Логика работы страницы (Листинг 7, Приложение А)

Загрузка данных:

Конструктор Sale\_historyViewModel(int partnerId) загружает данные о продаже, если передан partnerId (ID партнера). В случае, если ID равен 0, создается пустой список.

Для каждого элемента из коллекции PartnersProducts загружаются связанные данные: информация о продукте и партнере.

Работа с данными:

Список продаж:

Из базы данных загружаются все продажи конкретного партнера. Для этого используется Entity Framework с методом Include(), чтобы извлечь данные о продукте и партнере вместе с продажами.

Данные отображаются в ListBox с привязкой к свойствам модели PartnersProduct, таким как имя партнера, название продукта, количество и дата продажи.

Навигация:

Кнопка "Назад":

При нажатии на кнопку выполняется команда Exit(), которая приводит к возвращению на страницу с списком партнеров (Show).

## **Тестирование**

Данная библиотека (DLL) (Листинг 8, Приложение А), содержащая класс MaterialCalculator, предназначена для расчета необходимого количества материалов для производства продукции с учетом типа продукции, типа материала и процента брака. Это позволяет автоматически учитывать различные параметры производства и дефекты материалов, что является полезным для расчета материалов в производственных процессах.

Описание функциональности и назначения

Библиотека предназначена для использования в производственных системах или расчетных приложениях, где важно учитывать следующие аспекты:

1. Тип продукции и коэффициент расхода материала:

Для каждого типа продукции (в данном случае два типа, с ID 1 и 2) существует свой коэффициент расхода материала (ProductType1Coefficient и ProductType2Coefficient). Эти коэффициенты позволяют корректно учитывать, сколько материала необходимо для производства одной единицы продукции.

1. Тип материала и процент брака:

Для каждого типа материала (материалы с ID 1 и 2) задан свой процент брака. Это позволяет учесть, что при производственном процессе часть материала может быть утеряна или повреждена (например, из-за дефектов материала) — для каждого типа материала коэффициент брака определен как MaterialType1DefectRate и MaterialType2DefectRate.

1. Метод расчета потребности в материале:

Метод CalculateMaterialRequired выполняет расчеты на основе следующих входных данных:

productId — идентификатор типа продукции.

materialId — идентификатор типа материала.

productCount — количество производимых единиц продукции.

productParam1 и productParam2 — параметры продукции, влияющие на расчет потребности в материале.

В методе сначала проверяется корректность входных данных (например, что типы продукции и материала находятся в пределах допустимых значений). Далее выбирается соответствующий коэффициент для типа продукции и процент брака для типа материала.

Рассчитывается необходимое количество материала для производства одной единицы продукции с учетом этих коэффициентов.

Учитывается процент брака, после чего вычисляется общее количество материала, необходимое для производства заданного количества продукции. Результат округляется в большую сторону с использованием Math.Ceiling, чтобы учесть все мелкие отклонения.

Тесты для библиотеки MaterialCalculator направлены на проверку корректности работы метода CalculateMaterialRequired для различных сценариев, охватывающих как простые, так и более сложные случаи. Ниже представлено краткое описание тестов:

Простые тесты:

1. Test\_CalculateMaterialRequired\_ZeroProductCount: Проверяет, что при количестве продукции равном 0 метод возвращает 0 единиц материала.
2. Test\_CalculateMaterialRequired\_Product1\_InvalidProductId: Проверяет, что при некорректном ID продукта метод возвращает -1.
3. Test\_CalculateMaterialRequired\_Product2\_InvalidMaterialId: Проверяет, что при некорректном ID материала метод возвращает -1.
4. Test\_CalculateMaterialRequired\_ValidInput\_ZeroProductParam: Проверяет, что при параметре продукции равном 0 возвращается 0 единиц материала.
5. Test\_CalculateMaterialRequired\_ValidInput\_LargeProductCount: Проверяет, что метод корректно рассчитывает требуемое количество материала при большом количестве продукции.
6. Test\_CalculateMaterialRequired\_ValidInput\_LargeParams: Проверяет работу метода с большими значениями параметров продукции и материала.
7. Test\_CalculateMaterialRequired\_ValidInput\_ExactValues: Проверяет точность расчета для стандартных значений.
8. Test\_CalculateMaterialRequired\_ZeroProductParam1: Проверяет, что при параметре продукции равном 0 материал не требуется.
9. Test\_CalculateMaterialRequired\_MinimumParam: Проверяет корректный расчет при минимальных значениях параметров.
10. Test\_CalculateMaterialRequired\_ZeroMaterialCount: Проверяет, что при количестве материала равном 0 результат также будет 0.

Сложные тесты:

1. Test\_CalculateMaterialRequired\_RoundingErrorCheck: Проверяет точность округления при расчете требуемого материала с учетом брака.
2. Test\_CalculateMaterialRequired\_VaryingProductCoefficients: Проверяет, что для второго типа продукции, с более высоким коэффициентом, результат будет больше, чем для первого типа.
3. Test\_CalculateMaterialRequired\_ProductType2\_MaterialType2\_LargeValue: Проверяет работу метода с большими значениями для второго типа продукции и материала, убедившись, что результат корректно превышает 10,000.
4. Test\_CalculateMaterialRequired\_InvalidProductType: Проверяет, что при несуществующем типе продукции метод возвращает -1.
5. Test\_CalculateMaterialRequired\_RoundingDown: Проверяет округление в меньшую сторону и гарантирует, что погрешность округления не превышает 5 единиц.

Также было разработано 5 сценариев

Сохранение данных партнера с валидными значениями: Проверяется, что при правильном вводе всех данных, партнер успешно сохраняется в базе данных.

Отмена редактирования партнера: Проверяется, что при отмене изменений партнер не сохраняет новые данные, а остается в исходном состоянии.

Сохранение партнера с некорректным email: Проверяется, что система позволяет сохранить партнера, даже если его email некорректен.

Сохранение данных партнера с пустыми обязательными полями: Проверяется, что если обязательные поля (например, наименование) пусты, партнер не сохраняется в базе данных.

Отображение всех типов партнеров в выпадающем списке: Проверяется, что все типы партнеров, загруженные из базы данных, отображаются в выпадающем списке при редактировании данных.

Более подробно можно ознакомиться в Листинг 10. Приложение А

## **Диаграммы**

Были разработаны диаграмма использования и диаграмма деятельности, которые помогают наглядно представить взаимодействие пользователей с системой и процессы внутри системы.

1. Диаграмма использования: Эта диаграмма демонстрирует основные сценарии взаимодействия различных ролей пользователей (клиенты, тренеры и администраторы) с системой. Включает такие функции, как авторизация, регистрация, выбор тренера, подача и принятие заявок, создание и просмотр планов тренировок, а также предоставление скидок постоянным клиентам. Диаграмма показывает, какие действия могут быть выполнены каждым типом пользователя, а также какие процессы требуют взаимодействия с другими пользователями системы.



Рисунок №6 – Диаграмма использования

1. Диаграмма деятельности: Диаграмма описывает поток действий в системе, связанных с планированием и выполнением тренировок. Она охватывает процессы, такие как создание тренировочного плана тренером, выполнение упражнений клиентом, а также контроль выполнения упражнений, с учетом параметров, таких как пульс клиента. Диаграмма показывает последовательность шагов в различных процессах, начиная от подачи заявки клиентом и заканчивая проверкой выполнения упражнений тренером.

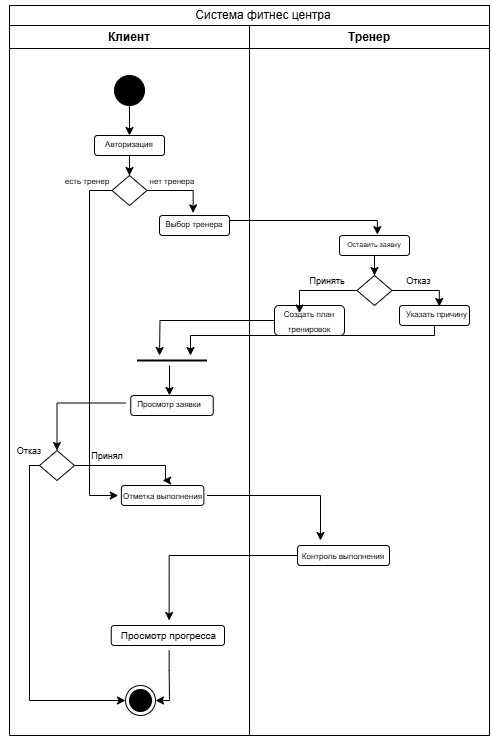


Рисунок №6 – Диаграмма деятельности

# **Поддержка и тестирование программных модулей**

# **База данных и импорт**

В рамках учебной практики была создана база данных (Листинг 1, Приложение Б), содержащая следующие таблицы:

categories (категории товаров)

manufacturers (производители)

pick\_up\_points (пункты самовывоза)

roles (роли пользователей)

status (статусы заказов)

suppliers (поставщики)

units (единицы измерения)

products (товары)

users (пользователи)

orders (заказы)

orders\_products (товары в заказах)

Все таблицы были созданы с соответствующими ограничениями целостности (первичные и внешние ключи, каскадные обновления/удаления, автоинкрементные идентификаторы).

После создания структуры базы был выполнен импорт данных из Excel.

Импорт данных позволил заполнить справочные таблицы и добавить начальный ассортимент товаров. Теперь база готова к использованию для ведения учета заказов и работы с клиентами.

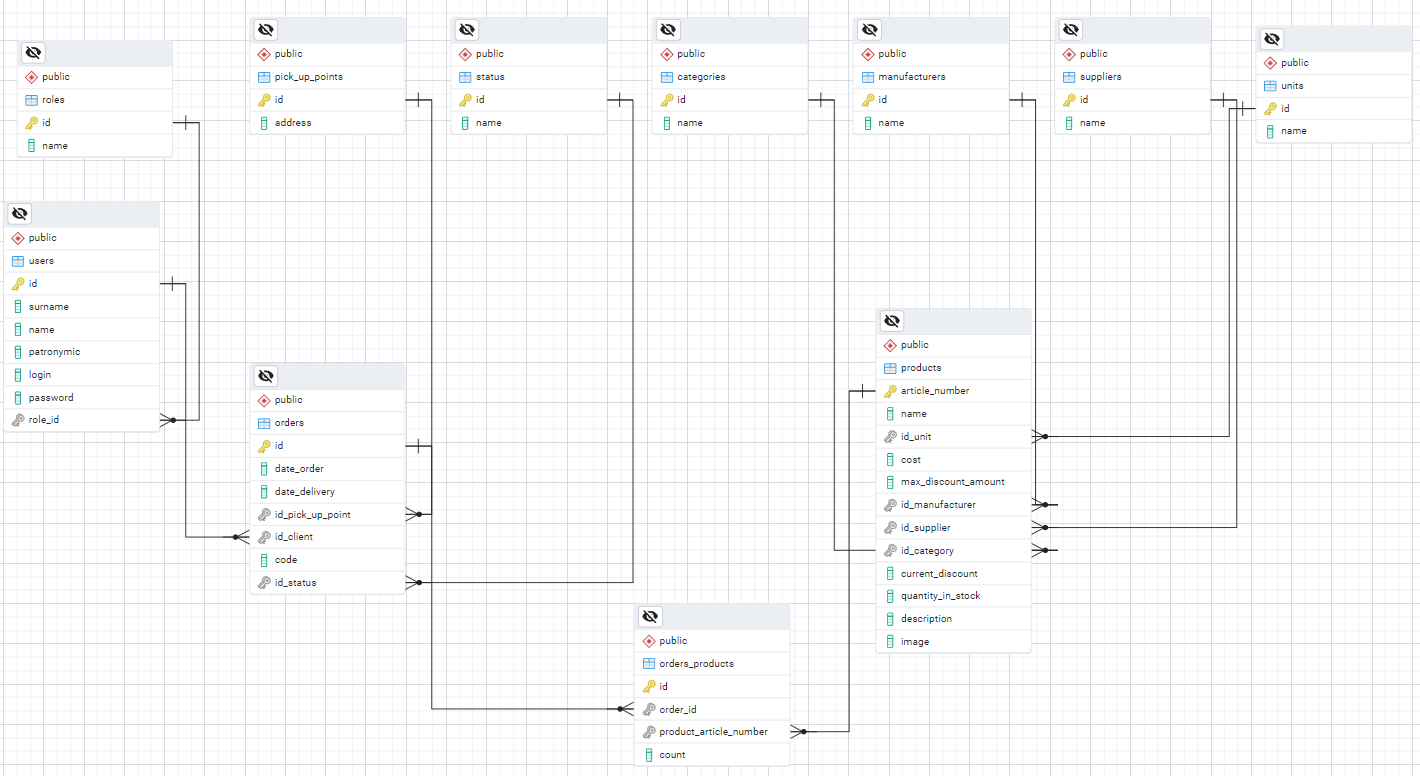


Рисунок 1 – ER Диаграмма

# **Avalonia**

Верстка (XAML)

Файл: Authorization.xaml (Листинг 2, Приложение Б)

Форма авторизации реализована на Avalonia UI, адаптирована для центрирования на экране и содержит следующие элементы:

Заголовок – TextBlock с надписью "Авторизация".

Поле ввода логина – TextBox, привязанный к Login.

Поле ввода пароля – TextBox, привязанный к Password, скрывающий вводимые символы (PasswordChar="\*").

CAPTCHA (появляется при неудачных попытках):

Изображение CAPTCHA – Image, привязанное к CaptchaImage.

Поле ввода CAPTCHA – TextBox, привязанное к CaptchaInput.

Кнопки:

"Войти" (Button), привязан к LoginCommand.

"Войти как гость" (Button), привязан к GuestCommand.

Стилизация:

Все TextBlock используют шрифт Comic Sans MS.

Кнопки имеют зеленый фон (rgb(73, 140, 81)), белый текст и размер FontSize=16.

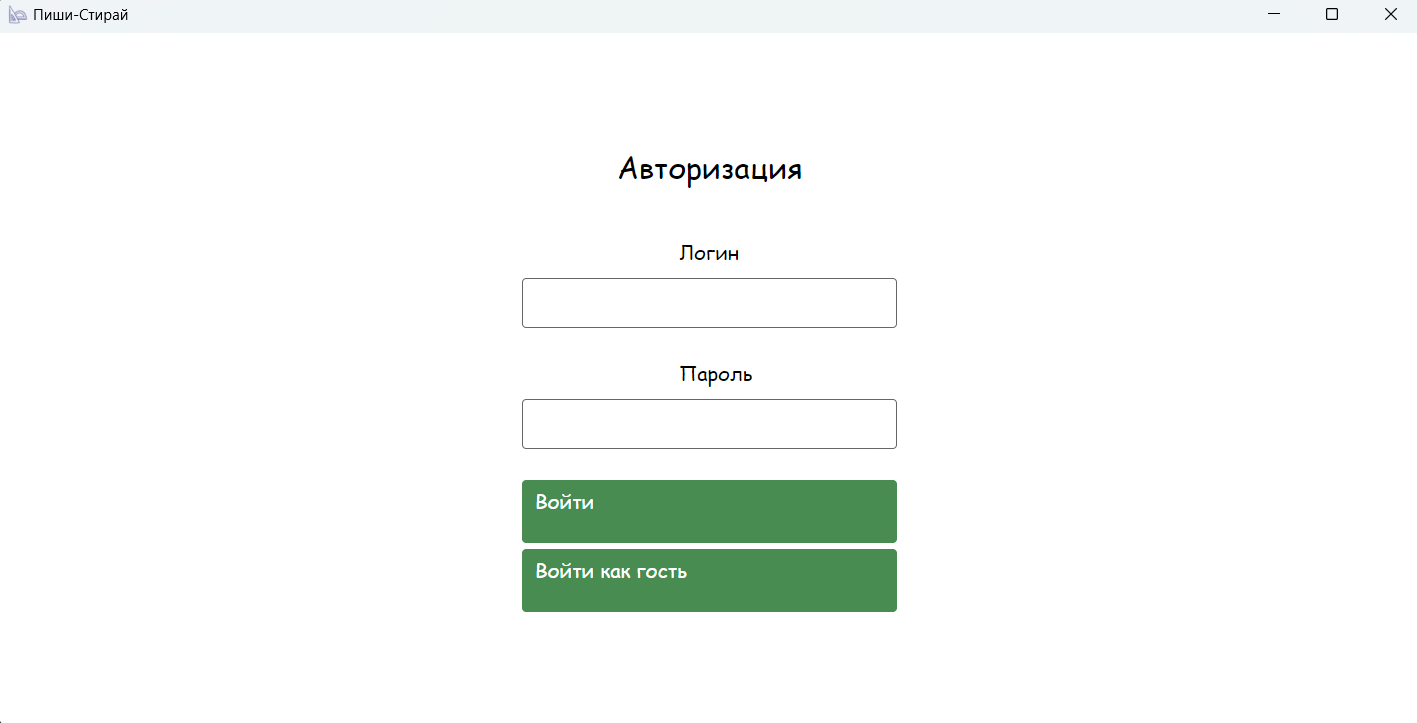


Рисунок 2 – Авторизация

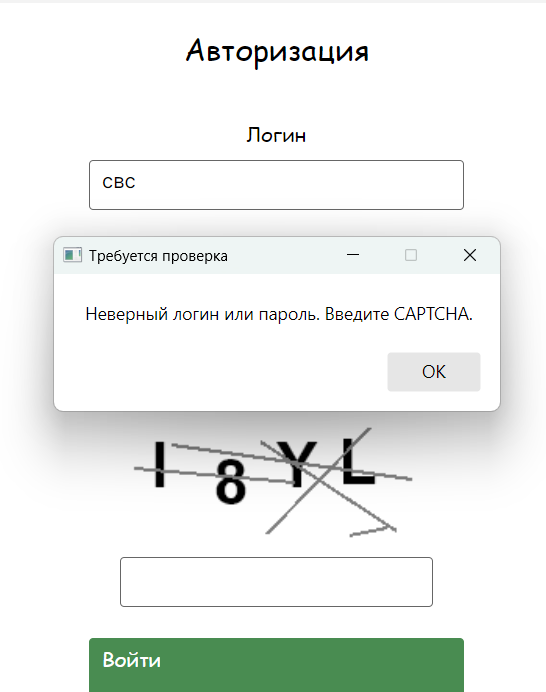


Рисунок 3 – CAPTCHA

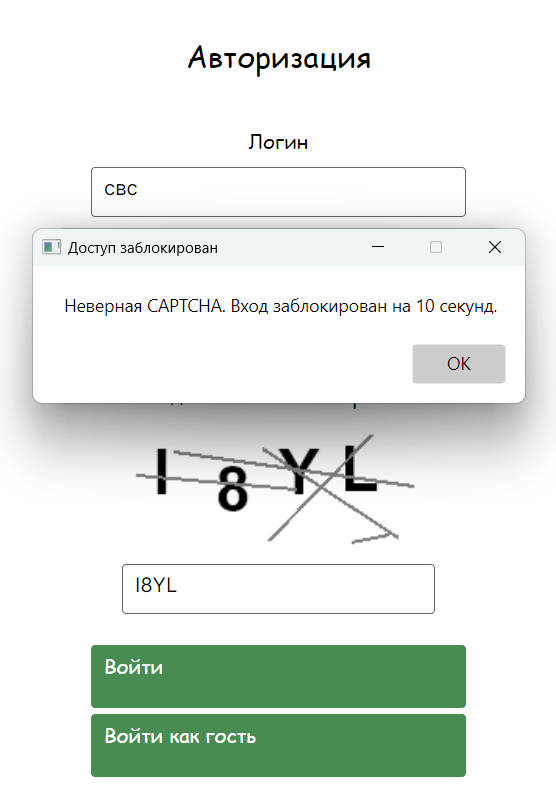


Рисунок 4 – Блокировка

Функционал (C#)

Файл: AuthorizationViewModel.cs (Листинг 3, Приложение Б)

Форма авторизации управляется AuthorizationViewModel, реализующим авторизацию пользователя с проверкой CAPTCHA и защитой от подбора пароля.

Основные свойства

Login и Password – хранят введенные пользователем данные.

CaptchaInput – введенный пользователем текст CAPTCHA.

CaptchaText – сгенерированный текст CAPTCHA.

IsCaptchaRequired – определяет, нужно ли отображать CAPTCHA.

CaptchaImage – изображение CAPTCHA.

\_failedAttempts – счетчик неудачных попыток.

\_lockTime – время блокировки при множественных ошибках.

Команды

LoginCommand – обработчик авторизации.

GuestCommand – вход в систему без авторизации.

Алгоритм авторизации (LoginUser)

Если время \_lockTime еще не истекло – выводится сообщение о блокировке.

Если CAPTCHA требуется, но введена неверно – обновляется изображение и выводится сообщение об ошибке.

Поиск пользователя в базе данных (\_43pKasatkina2upContext).

Если пользователь найден:

Открывается основное окно (OpenUserWindow).

Сбрасываются ошибки и CAPTCHA.

Если пользователь не найден:

Увеличивается \_failedAttempts.

После первой ошибки – показывается CAPTCHA.

После второй ошибки – блокировка на 10 секунд.

Show.axaml (Листинг 4, Приложение Б)

Этот XAML-файл представляет пользовательский элемент управления (UserControl) в AvaloniaUI. Он отображает список товаров, позволяя пользователю сортировать, фильтровать и выполнять действия с ними.

1. Заголовки и информация о пользователе

В верхней части интерфейса находятся:

Имя текущего пользователя (CurrentUserFullName)

Количество товаров в списке (ProductList.Count)

2. Панель поиска и фильтрации

Есть возможность:

Искать товары по названию (TextBox, привязанный к SearchQuery)

Фильтровать по диапазону скидок (ComboBox, привязанный к SelectedDiscountRange)

Сортировать по цене (две RadioButton-кнопки для сортировки по возрастанию/убыванию)

3. Кнопки управления

Посмотреть заказ (если пользователь имеет доступ к заказам)

Список заказов (для менеджера)

Добавить новый товар (для администратора)

4. Список товаров (ListBox)

Используется ListBox для отображения списка товаров (ProductList).

Каждый товар представлен в DataTemplate, включающем:

Изображение (Image с конвертером ImageConverter)

Название (TextBlock, привязанный к Name)

Описание (TextBlock, привязанный к Description)

Цена (отображается со скидкой или без нее)

Размер скидки (TextBlock, привязанный к CurrentDiscount)

Кнопки Редактировать и Удалить (только для администратора)

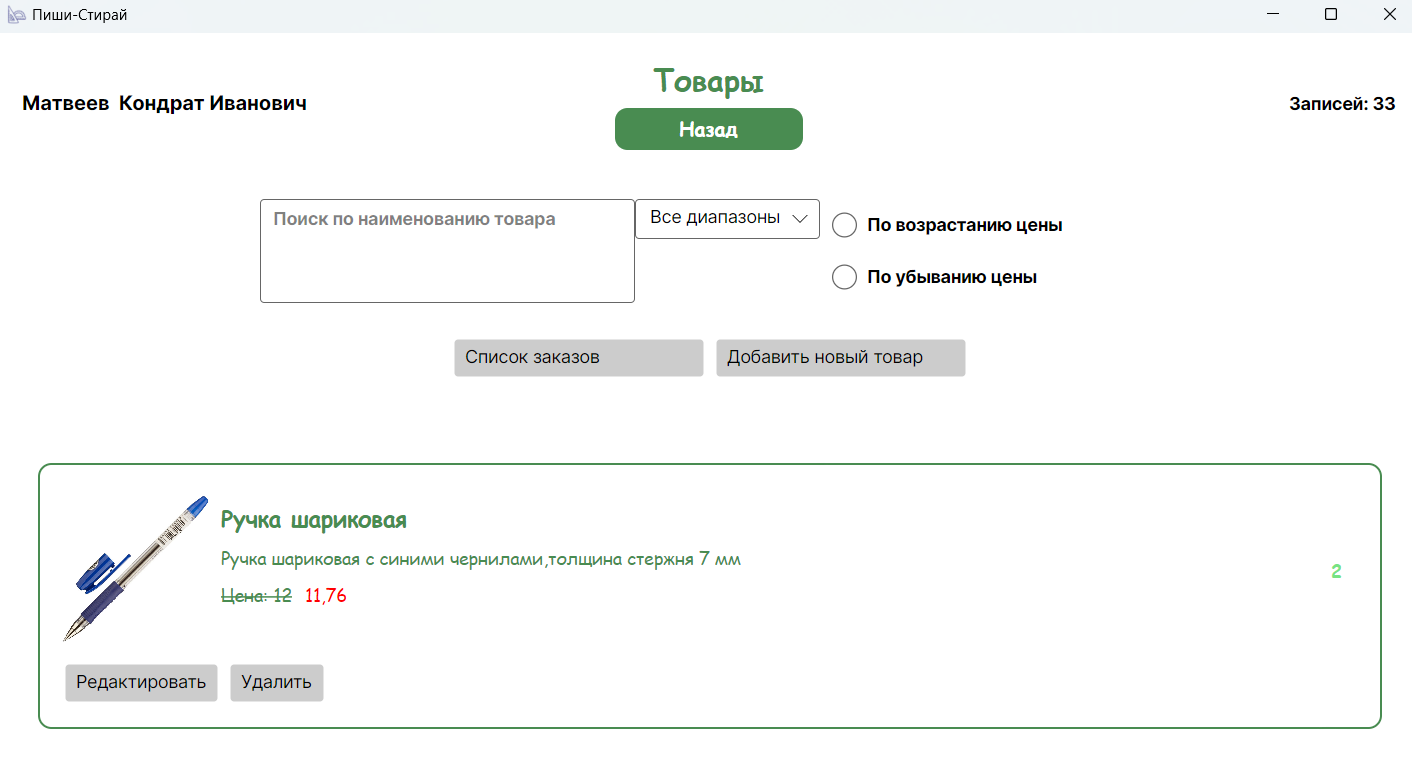


Рисунок 5 – Show

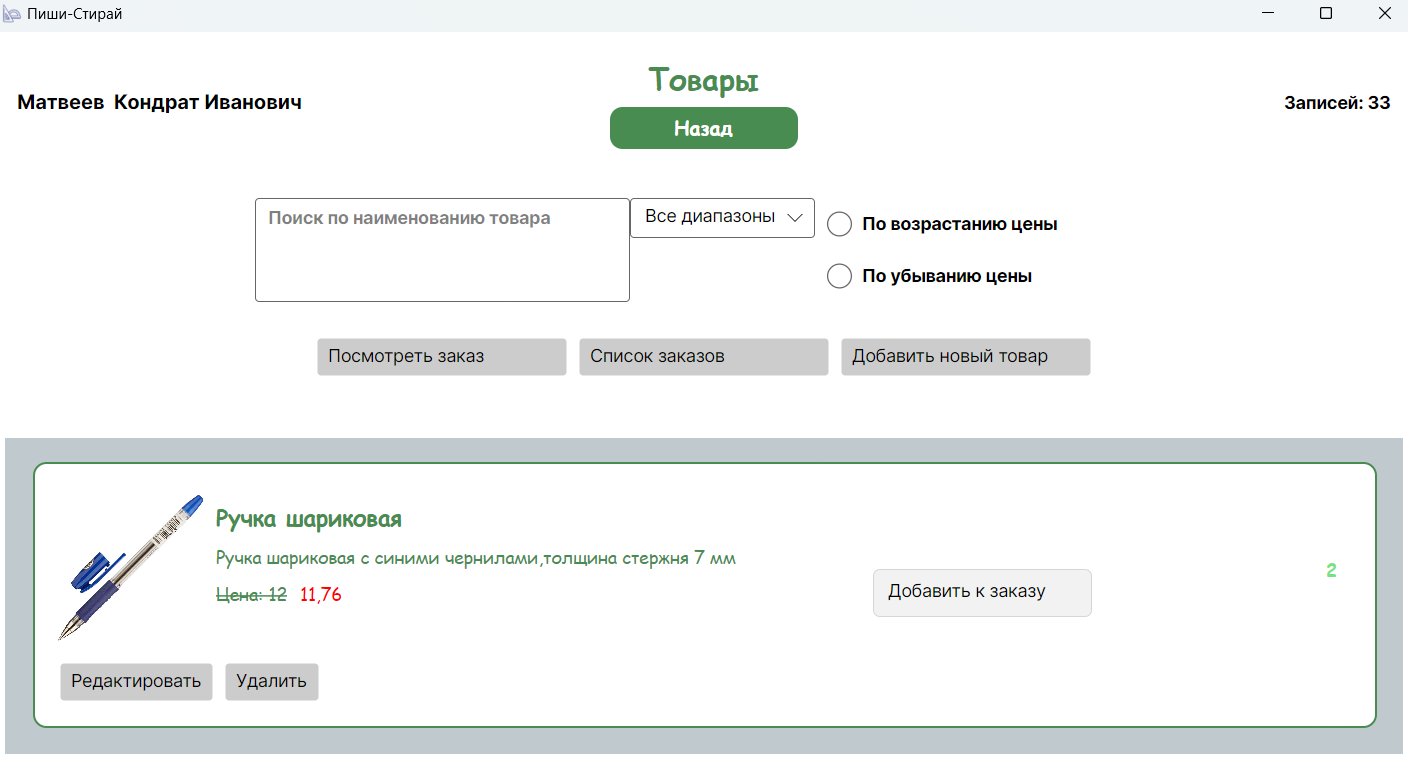


Рисунок 6 – Добавление в заказ

ShowViewModel (Листинг 5, Приложение Б)

Этот ViewModel управляет логикой взаимодействия с пользователем.

1. Список товаров (ProductList)

Загружается из базы данных через MainWindowViewModel.myConnection.Products.

2. Фильтрация и сортировка

Поиск (SearchQuery): фильтрует список товаров по названию.

Фильтрация по скидкам (SelectedDiscountRange): оставляет только товары, попадающие в определённый диапазон скидок.

Сортировка (IsPriceAscending, IsPriceDescending): меняет порядок отображения товаров по цене.

3. Управление заказами

AddProductToOrder() – добавляет товар в заказ.

ViewOrder() – открывает страницу текущего заказа.

DeleteProduct(Product product) – удаляет товар (если у пользователя есть права).

Edit(Product product) – открывает страницу редактирования товара.

AddProduct() – добавляет новый товар.

Разметка XAML OrderUser.xaml (Листинг 6, Приложение Б)

Этот файл определяет визуальный интерфейс окна управления заказами.

Основная структура

UserControl – корневой элемент, представляющий пользовательский интерфейс компонента.

Grid – используется для организации элементов на странице.

DockPanel – выводит имя текущего пользователя.

TextBlock – отображает заголовок "Заказ".

Button – кнопка "Назад", привязанная к команде Exit.

Список товаров

ListBox привязан к коллекции OrdersProducts и отображает элементы заказа.

Каждый элемент содержит:

Image – изображение товара (с конвертером ImageConverter).

TextBlock – название, описание, производитель, категория, поставщик, цена и скидка.

StackPanel с кнопками + и - для изменения количества.

Итоговые данные заказа

TextBlock показывает итоговую сумму и скидку.

ComboBox позволяет выбрать пункт выдачи (PickUpPoint).

Button для оформления заказа (SaveOrder).

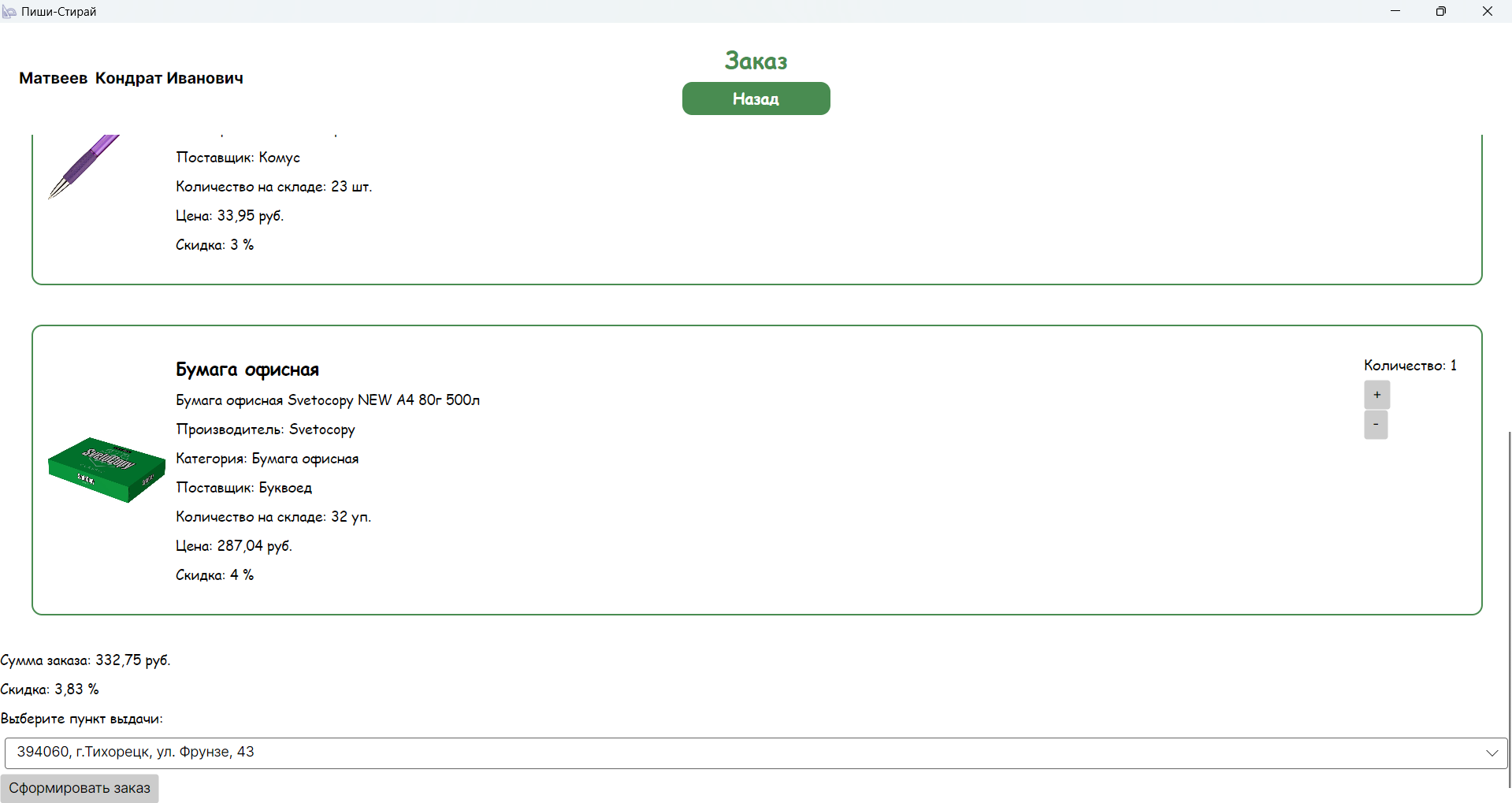


Рисунок 7 - Order

2. ViewModel OrderUserViewModel.cs (Листинг 7, Приложение Б)

Этот класс управляет данными и логикой работы представления.

Свойства

FullCost – полная стоимость заказа.

FullDiscount – скидка на заказ.

CurrentUserFullName – ФИО текущего пользователя (или "Гость").

Order – текущий заказ.

OrdersProducts – список товаров в заказе.

PickUpPoint – список пунктов выдачи.

Конструктор

Принимает User и OrdersProducts.

Загружает данные из БД через MainWindowViewModel.myConnection.

Вычисляет стоимость заказа (UpdateCost()).

Методы

UpdateCost() – пересчитывает стоимость и скидку.

AddUnit(product) – увеличивает количество товара в заказе.

DeleteUnit(product) – уменьшает количество, удаляет товар при Count == 1.

SaveOrder() – оформляет заказ:

Генерирует дату доставки (3 или 6 дней в зависимости от наличия товара).

Присваивает случайный код (100–999).

Обновляет БД.

Завершает оформление.

Exit() – возвращает пользователя в главное меню.

Разметка XAML ListOrders.xaml (Листинг 8, Приложение Б)

Этот XAML-файл описывает интерфейс страницы списка заказов.

Основная структура

Grid – используется для организации элементов.

Заголовок (TextBlock) – отображает "Заказы клиентов".

Кнопка "Назад" (Button) – позволяет вернуться на предыдущую страницу.

Панель фильтрации и сортировки (StackPanel):

ComboBox для выбора типа сортировки (ListSortedType).

ComboBox для выбора фильтра по скидкам (ListFilterType).

Список заказов (ListBox), привязанный к OrdersPreview:

Каждый заказ оформляется в Border с динамическим цветом (Color).

В StackPanel отображаются:

ID заказа (Id).

ФИО клиента (FIO).

Дата заказа (DateOrder).

Дата доставки (DateDelivery).

Адрес пункта выдачи (IdPickUpPointNavigation.Address).

Код получения (Code).

Статус заказа (IdStatusNavigation.Name).

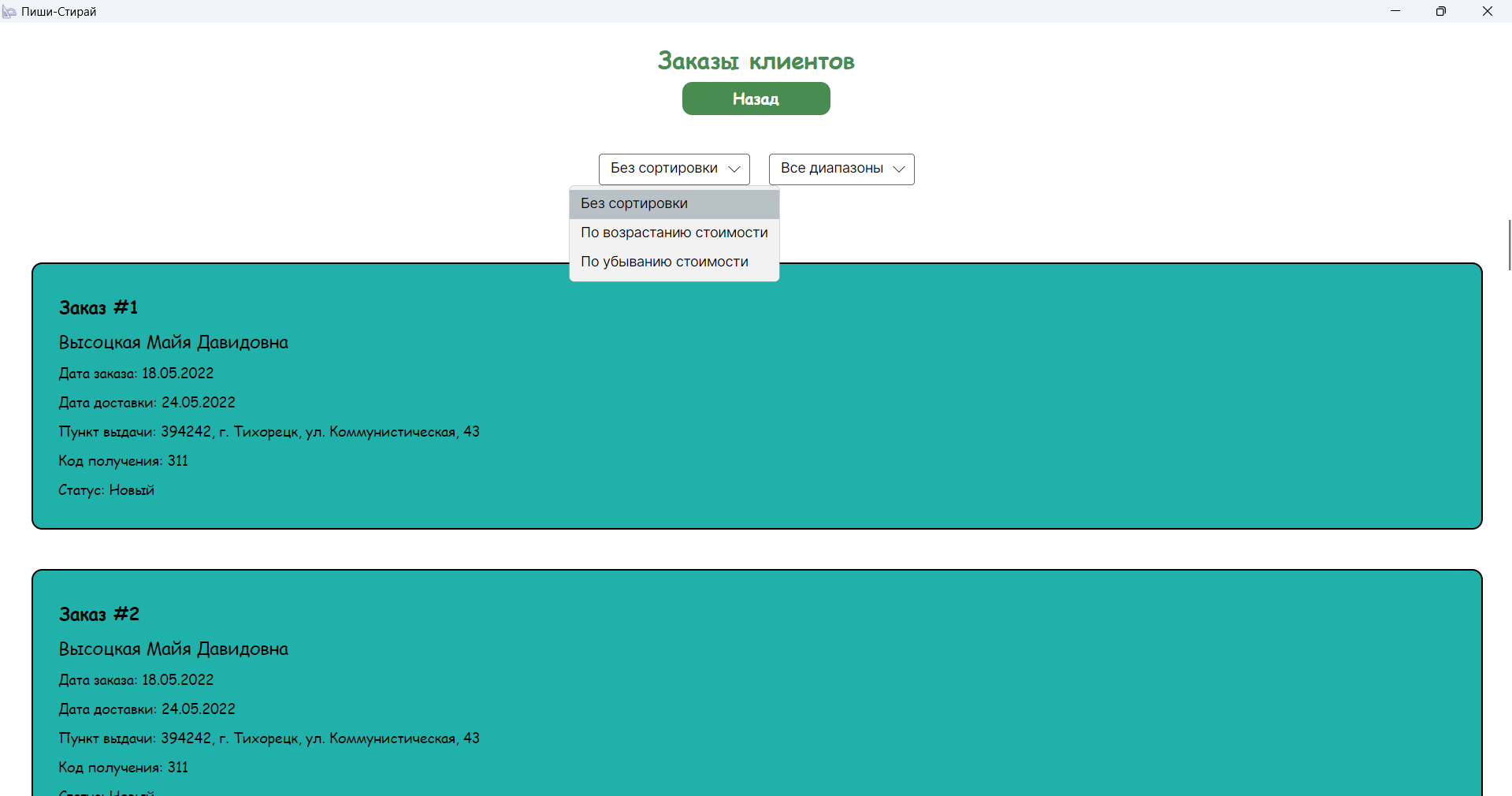


Рисунок 8 – Заказы клиентов

ViewModel ListOrdersViewModel.cs (Листинг 9, Приложение Б)

Основные свойства

\_currentUser – текущий пользователь.

Orders – полный список заказов.

OrdersPreview – отфильтрованный список заказов.

ListSortedType – список вариантов сортировки.

ListFilterType – список вариантов фильтрации.

SelectedSortedType – индекс выбранного типа сортировки.

SelectedFilterType – индекс выбранного фильтра.

SelectedOrder – выбранный заказ.

Функционал

Фильтрация и сортировка (Filter())

Если SelectedSortedType:

0 – без сортировки.

1 – сортировка по возрастанию стоимости (OrdersPreview.OrderBy(it => it.Cost)).

2 – сортировка по убыванию стоимости (OrdersPreview.OrderByDescending(it => it.Cost)).

Если SelectedFilterType:

0 – все заказы.

1 – скидка 0–10%.

2 – скидка 11–14%.

3 – скидка 15%+.

Загрузка заказов

Orders загружаются из базы данных (MainWindowViewModel.myConnection.Orders).

Добавляется информация о клиенте (FIO).

Определяется цвет (Color):

"#ffffff" – по умолчанию.

"#20b2aa" – все товары есть в наличии (более 3 штук).

"#ff8c00" – хотя бы один товар отсутствует.

Выбор заказа (SelectedOrder)

При выборе заказа (ListBox) открывается окно редактирования (EditOrder).

Выход (Exit())

Возвращает пользователя на основное окно (Show).

Разметка XAML EditProduct.xaml (Листинг 10, Приложение Б)

Основная структура

Grid – используется для организации элементов управления.

Заголовок (TextBlock) – динамически изменяется (Title), отображает "Добавление товара" или "Редактирование товара".

Кнопка "Назад" (Button) – привязана к команде Exit, возвращает на предыдущую страницу.

Форма ввода данных о товаре (ScrollViewer с Grid):

TextBox для ввода:

Артикул (ArticleNumber, только при добавлении товара).

Название (Name).

Описание (Description).

NumericUpDown для числовых значений:

Стоимость (Cost).

Текущая скидка (CurrentDiscount).

Максимальная скидка (MaxDiscountAmount).

Количество (QuantityInStock).

* + ComboBox для выбора:

Единицы измерения (Units).

Категории (Categories).

Производителя (Manufacturers).

Поставщика (Suppliers).

Кнопка "Сохранить" (Button) – привязана к команде Save.

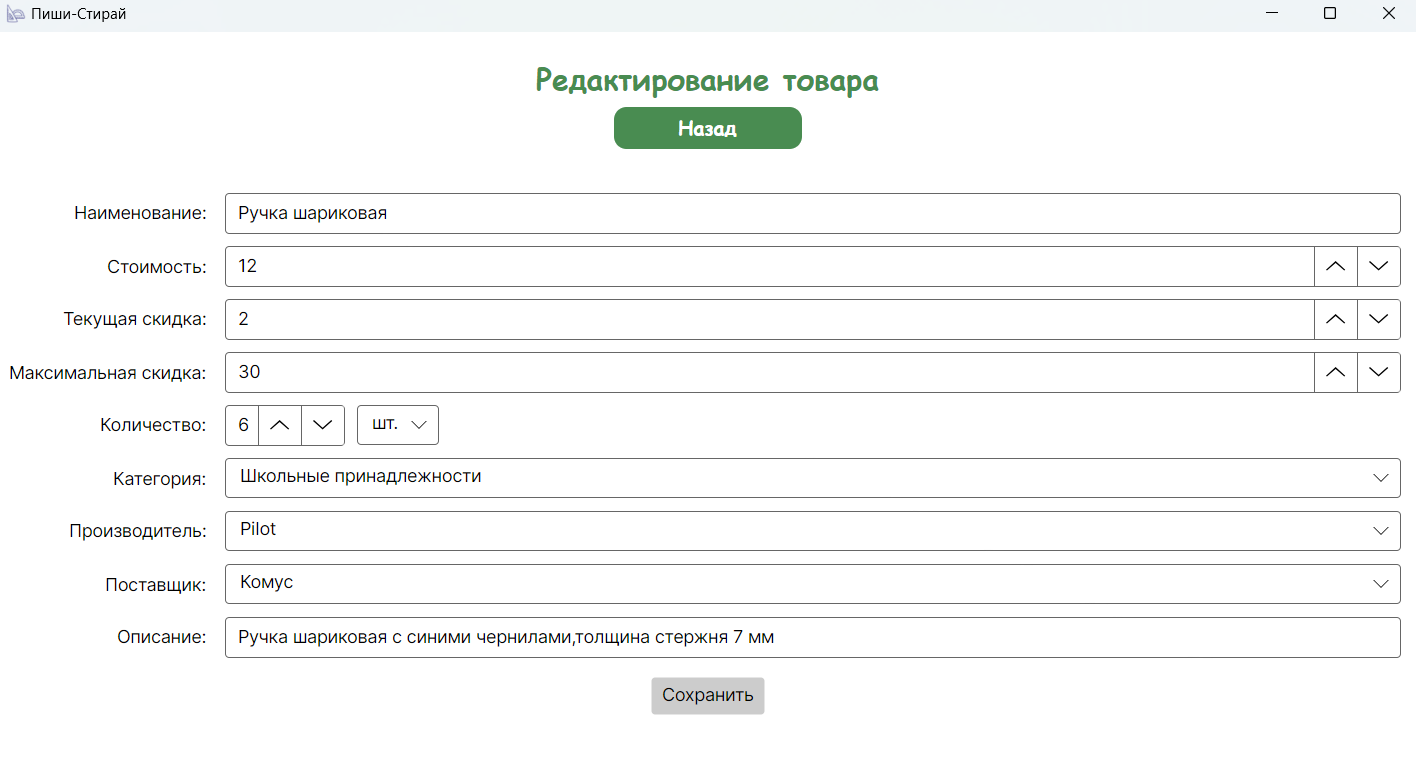


Рисунок 9 – Редактирование товара

EditProductViewModel.cs (Листинг 11, Приложение Б)

Основные свойства

\_currentUser – текущий пользователь.

Title – заголовок страницы (определяет, добавляется новый товар или редактируется существующий).

Product – объект, представляющий редактируемый товар.

IsNewProduct – определяет, создается новый товар (true) или редактируется существующий (false).

Categories, Manufacturers, Suppliers, Units – списки для заполнения ComboBox.

Функционал

Инициализация данных (Launch())

Загружает из базы доступные категории, производителей, поставщиков и единицы измерения.

Конструкторы

EditProductViewModel(User? user) – для создания нового товара.

EditProductViewModel(User? user, string article) – для редактирования существующего товара (загружает товар по article).

Сохранение (Save())

Если IsNewProduct == true, то товар добавляется в базу.

Если false, то товар обновляется.

Выводит сообщение об успехе или ошибке.

Разметка XAML EditOrder.xaml (Листинг 12, Приложение Б)

Основная структура

Grid – используется для организации элементов.

Заголовок (TextBlock) – статичный, отображает "Изменение заказа".

Кнопка "Назад" (Button) – привязана к команде Exit, возвращает пользователя на страницу списка заказов.

Информация о заказе (StackPanel):

TextBlock с ID заказа (Order.Id).

TextBlock с ФИО клиента (Order.FIO).

TextBlock с датой заказа (Order.DateOrder).

TextBox для редактирования даты доставки (Order.DateDelivery).

TextBlock с пунктом выдачи (Order.IdPickUpPointNavigation.Address).

TextBlock с кодом получения (Order.Code).

ComboBox для выбора статуса заказа (Statuses).

Button "Сохранить" – привязан к команде Save.

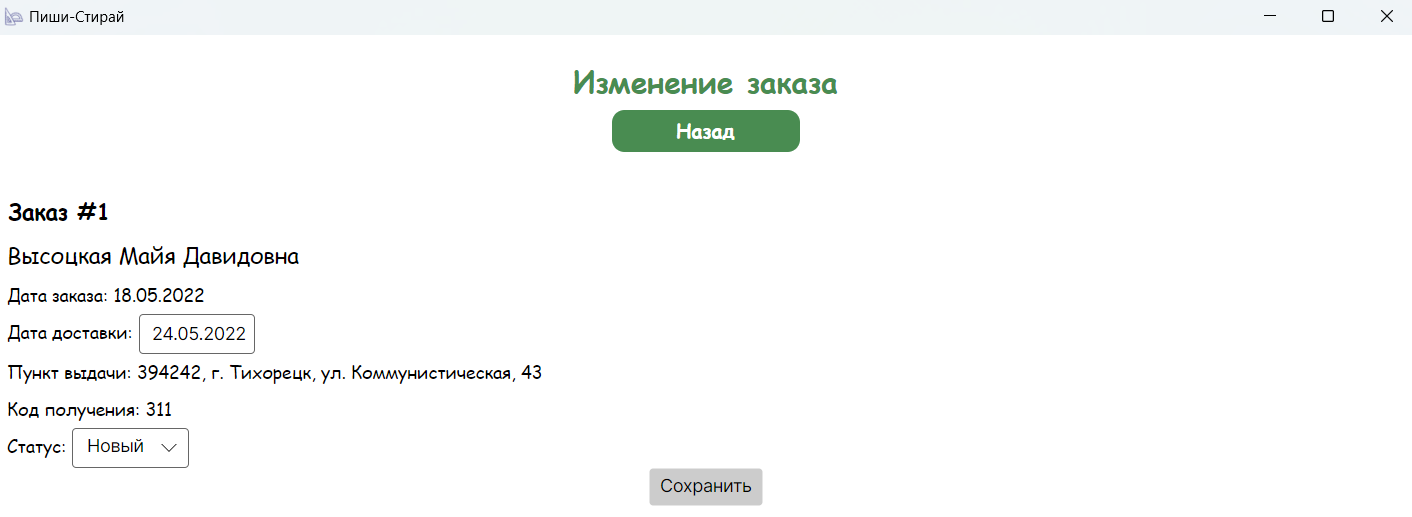


Рисунок 10 – Изменение заказа

EditOrderViewModel.cs (Листинг 13, Приложение Б)

Основные свойства

\_currentUser – текущий пользователь.

Order – редактируемый заказ.

Statuses – список доступных статусов заказа.

Функционал

Инициализация (EditOrderViewModel)

Загружает список статусов заказа (Statuses).

Получает переданный в конструктор заказ (Order).

Сохранение (Save())

Обновляет запись заказа в базе (Orders.Update(Order)).

Сохраняет изменения (SaveChanges()).

Показывает сообщение о сохранении (MessageBoxManager).

Возвращает пользователя на страницу списка заказов.

Выход (Exit())

Возвращает пользователя в ListOrders.

# **Тестирование**

Библиотека WriteEraseLibrary (Листинг 14, Приложение Б) была разработана в соответствии с заданием для вычисления списка свободных временных интервалов заданного размера в графике сотрудника. Она предназначена для формирования оптимального рабочего расписания сотрудников, предоставляя доступные временные промежутки для встреч, консультаций или других задач.

Функциональное назначение

Основная задача библиотеки – обработка информации о занятых временных слотах и определение свободных промежутков времени, доступных для записи.

Соответствие требованиям задания

Структура входных данных

Библиотека принимает:

Список занятых промежутков времени, представленный двумя массивами:

startTimes – массив начальных временных точек уже занятых записей.

durations – массив длительностей занятых слотов (в минутах).

Рабочий день сотрудника:

beginWorkingTime – время начала рабочего дня.

endWorkingTime – время окончания рабочего дня.

Минимальное необходимое время для работы менеджера (consultationTime).

Выходные данные

Метод возвращает массив строк, где каждая строка представляет собой свободный временной интервал в формате "HH:mm-HH:mm".

Реализация

Класс Calculations

Класс Calculations реализует метод AvailablePeriods, который:

1. Формирует список занятых временных промежутков.
2. Сортирует их по времени начала.
3. Определяет свободные интервалы между занятой зоной и границами рабочего времени.
4. Формирует список подходящих временных слотов.

Для библиотеки WriteEraseLibrary были разработаны 10 тестов (Листинг 15, Приложение Б), которые проверяют различные сценарии работы метода AvailablePeriods. Эти тесты охватывают множество случаев, включая ситуации с полными или частично занятыми временными интервалами, а также проверку правильности работы с пустыми или перекрывающимися интервалами. Ниже приведены краткие описания каждого теста:

1. Test\_NoFreeSlots  
   Проверяет случай, когда весь рабочий день занят (например, с 8:00 до 18:00), и не остается свободных интервалов для записи. Ожидаемый результат — пустой массив.
2. Test\_BusyPeriodsWithNoFreeTime\_ReturnsEmptyArray  
   Тестирует ситуацию, когда рабочий день полностью занят (например, с 8:00 до 10:00), и нет места для консультаций. Ожидаемый результат — пустой массив.
3. Test\_FullDayBusyWithGapsSmallerThanConsultation  
   Проверяет случай, когда между занятиями есть промежутки, но они меньше, чем минимальная длительность консультации. В этом случае метод должен вернуть пустой массив, поскольку нет подходящих временных интервалов для консультаций.
4. Test\_ComplexScheduleWithIrregularGaps  
   Проверяет более сложное расписание с нерегулярными промежутками времени. Тест проверяет правильность вычисления всех возможных свободных интервалов, даже если промежутки между занятиями различной длительности.
5. Test\_NoBusyPeriods\_ReturnsFullDaySlots  
   Проверяет случай, когда рабочий день полностью свободен (нет занятых интервалов). Ожидается, что метод вернет полный список всех возможных слотов на рабочий день.
6. Test\_EmptyBusyPeriods\_ReturnsFullDaySlots  
   Похож на предыдущий тест, но проверяет обработку пустых массивов для занятых интервалов. Метод должен вернуть список всех возможных временных интервалов, равных длине рабочего дня.
7. Test\_SingleSlot  
   Тестирует ситуацию, когда всего один временной интервал занят, и должен быть найден один свободный интервал для консультации.
8. Test\_LateEveningConsultationSlot  
   Проверяет случай, когда занятие начинается поздно в рабочий день (например, с 16:00) и проверяется, будут ли оставаться свободные интервалы после этого времени.
9. Test\_SingleFreeSlotAtStart  
   Проверяет ситуацию, когда в начале рабочего дня есть единственный свободный интервал (например, с 8:00 до 8:30), и остальная часть дня занята.
10. Test\_OverlappingBusyPeriods  
    Тестирует случай с перекрывающимися занятыми периодами, чтобы убедиться, что метод правильно обрабатывает ситуации, когда два или более события перекрываются. Ожидается, что свободные интервалы будут правильно рассчитаны.

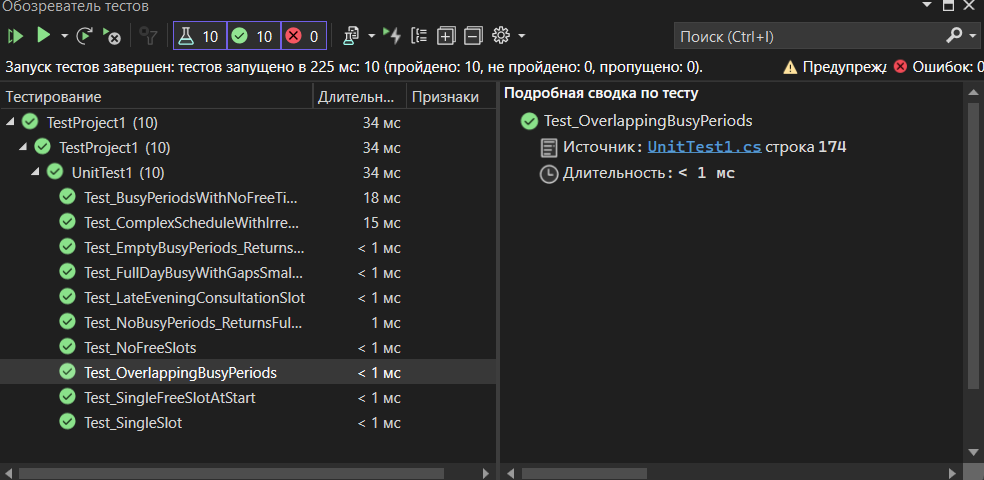


Рисунок 11 – Тесты

Были разработаны 5 тестовых кейсов (Листинг 16, Приложение Б) для проверки функционала добавления товара в систему. Каждый тест проверяет определенные ситуации и функциональность, связанную с заполнением обязательных полей и правильностью данных. Вот краткое описание каждого теста:

1. Тест 1: Успешное добавление товара с корректными данными  
   Проверяется, что при вводе всех корректных данных товар успешно добавляется в систему. Ожидаемый результат — товар добавляется в систему, и появляется сообщение об успешном добавлении.
2. Тест 2: Ошибка при добавлении товара без артикула  
   Тестируется ситуация, когда товар добавляется без артикула. Ожидаемый результат — система должна выдать сообщение об ошибке, информируя, что артикул является обязательным полем.
3. Тест 3: Добавление товара с отрицательной стоимостью  
   Проверяется, что система не позволяет добавить товар с отрицательной стоимостью. Ожидаемый результат — товар не добавляется в систему, и данные не сохраняются в базе.
4. Тест 4: Добавление товара с отсутствующими данными в полях категории, производителя или поставщика  
   Тестируется ситуация, когда обязательные поля (категория, производитель или поставщик) не заполнены. Ожидаемый результат — система должна вернуть сообщение об ошибке, что эти поля обязательны.
5. Тест 5: Проверка добавления товара с пустыми полями  
   Проверяется, что система не позволит добавить товар, если все обязательные поля остаются пустыми. Ожидаемый результат — появляется сообщение об ошибке, информирующее о том, что все обязательные поля должны быть заполнены.

## **Заключение**

Учебная практика по теме "ПМ.01 Разработка модулей программного обеспечения для компьютерных систем" позволила углубить знания и навыки в области разработки программного обеспечения. В процессе практики был освоен весь цикл разработки программных модулей, включая создание кода, его тестирование и отладку.

Целью практики было применение теоретических знаний на практике, что позволило познакомиться с процессами проектирования, разработки и внедрения программных решений. В ходе выполнения задач использовались различные инструменты, такие как среды программирования, базы данных, системы контроля версий, что обеспечило практический опыт в реальной среде разработки.

Кроме того, практика способствовала лучшему пониманию принципов написания понятного и структурированного кода. Были изучены методы и подходы, такие как использование вспомогательных инструментов, написание документации и взаимодействие с базами данных.

В результате прохождения практики был приобретен уверенный опыт работы в области разработки программного обеспечения, освоены новые инструменты и технологии, что существенно поможет в дальнейшей профессиональной деятельности.

## **Приложение А**

**Листинг 1. Создание и заполнение таблиц**

-- Database: 43P\_Kasatkina\_UP

-- DROP DATABASE IF EXISTS "43P\_Kasatkina\_UP";

CREATE DATABASE "43P\_Kasatkina\_UP"

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251'

LOCALE\_PROVIDER = 'libc'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

-- Создание таблицы материалов

CREATE TABLE materials (

id SERIAL PRIMARY KEY,

material\_type VARCHAR(50) NOT NULL,

defect\_percentage DECIMAL(10,6) NOT NULL

);

-- Заполнение таблицы materials

INSERT INTO materials (id, material\_type, defect\_percentage) VALUES

(1, 'Тип материала 1', 0.001),

(2, 'Тип материала 2', 0.0095),

(3, 'Тип материала 3', 0.0028),

(4, 'Тип материала 4', 0.0055),

(5, 'Тип материала 5', 0.0034);

-- Создание таблицы типов партнеров

CREATE TABLE partner\_types (

id SERIAL PRIMARY KEY,

type\_name VARCHAR(50) NOT NULL

);

-- Заполнение таблицы partner\_types

INSERT INTO partner\_types (id, type\_name) VALUES

(1, 'ЗАО'),

(2, 'ООО'),

(3, 'ПАО'),

(4, 'ОАО');

-- Создание таблицы типов продукции

CREATE TABLE product\_types (

id SERIAL PRIMARY KEY,

type\_name VARCHAR(50) NOT NULL,

coefficient DECIMAL(10,2) NOT NULL

);

-- Заполнение таблицы product\_types

INSERT INTO product\_types (id, type\_name, coefficient) VALUES

(1, 'Ламинат', 2.35),

(2, 'Массивная доска', 5.15),

(3, 'Паркетная доска', 4.34),

(4, 'Пробковое покрытие', 1.5);

-- Создание таблицы продукции

CREATE TABLE products (

id SERIAL PRIMARY KEY,

article\_number VARCHAR(8) NOT NULL,

product\_type\_id INT REFERENCES product\_types(id),

product\_name VARCHAR(255) NOT NULL,

description VARCHAR(255) NULL,

image VARCHAR(255) NULL,

min\_partner\_price DECIMAL(10,2) NOT NULL,

length float null,

width float null,

height float null

);

-- Заполнение таблицы products

INSERT INTO products (article\_number, product\_type\_id, product\_name,description, image, min\_partner\_price,length,width, height ) VALUES

(8758385, 3, 'Паркетная доска Ясень темный однополосная 14 мм', NULL,NULL,4456.90, NULL, NULL, NULL),

(8858958, 3, 'Инженерная доска Дуб Французская елка однополосная 12 мм',NULL,NULL, 7330.99,NULL,NULL,NULL),

(7750282, 1, 'Ламинат Дуб дымчато-белый 33 класс 12 мм',NULL,NULL, 1799.33,NULL,NULL,NULL),

(7028748, 1, 'Ламинат Дуб серый 32 класс 8 мм с фаской', NULL,NULL,3890.41,NULL,NULL,NULL),

(5012543, 4, 'Пробковое напольное клеевое покрытие 32 класс 4 мм',NULL,NULL, 5450.59,NULL,NULL,NULL);

-- Создание таблицы партнеров

CREATE TABLE partners (

id SERIAL PRIMARY KEY,

partner\_type\_id INT REFERENCES partner\_types(id),

partner\_name VARCHAR(255) NOT NULL,

legal\_address TEXT NOT NULL,

inn BIGINT NULL,

director\_name VARCHAR(255) NOT NULL,

phone VARCHAR(20) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

rating INT NOT NULL,

logo VARCHAR(255) NULL

);

-- Заполнение таблицы partners

INSERT INTO partners (partner\_type\_id, partner\_name, legal\_address, inn, director\_name, phone, email, rating, logo) VALUES

(1, 'База Строитель', '652050, Кемеровская область, город Юрга, ул. Лесная, 15', 2222455179, 'Иванова Александра Ивановна', '493 123 45 67', 'aleksandraivanova@ml.ru', 7, NULL),

(2, 'Паркет 29', '164500, Архангельская область, город Северодвинск, ул. Строителей, 18', 3333888520, 'Петров Василий Петрович', '987 123 56 78', 'vppetrov@vl.ru', 7, NULL),

(3, 'Стройсервис', '188910, Ленинградская область, город Приморск, ул. Парковая, 21', 4440391035, 'Соловьев Андрей Николаевич', '812 223 32 00', 'ansolovev@st.ru', 7, NULL),

(4, 'Ремонт и отделка', '143960, Московская область, город Реутов, ул. Свободы, 51', 1111520857, 'Воробьева Екатерина Валерьевна', '444 222 33 11', 'ekaterina.vorobeva@ml.ru', 5, NULL),

(1, 'МонтажПро', '309500, Белгородская область, город Старый Оскол, ул. Рабочая, 122', 5552431140, 'Степанов Степан Сергеевич', '912 888 33 33', 'stepanov@stepan.ru', 10, NULL);

-- Создание таблицы продаж

CREATE TABLE partners\_products (

id SERIAL PRIMARY KEY,

product\_id INT REFERENCES products(id),

partner\_id INT REFERENCES partners(id),

quantity INT NOT NULL,

sale\_date DATE NOT NULL

);

INSERT INTO partners\_products (product\_id, partner\_id, quantity, sale\_date) VALUES

(1, 1, 15500, '2023-03-23'),

(3, 1, 12350, '2023-12-18'),

(4, 1, 37400, '2024-06-07'),

(2, 2, 35000, '2022-12-02'),

(5, 2, 1250, '2023-05-17'),

(3, 2, 1000, '2024-06-07'),

(1, 2, 7550, '2024-07-01'),

(1, 3, 7250, '2023-01-22'),

(2, 3, 2500, '2024-07-05'),

(4, 4, 59050, '2023-03-20'),

(3, 4, 37200, '2024-03-12'),

(5, 4, 4500, '2024-05-14'),

(3, 5, 50000, '2023-09-19'),

(4, 5, 670000, '2023-11-10'),

(1, 5, 35000, '2024-04-15'),

(2, 5, 25000, '2024-06-12');

Листинг 2. Show

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:DataType="vm:ShowViewModel"

xmlns:vm="using:MasterPol.ViewModels"

x:Class="MasterPol.Show">

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Мастер Пол"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="#67BA80"

FontFamily="Segoe UI"

Margin="20"/>

<Button Grid.Row="1"

Command="{Binding ToAdd}"

Content="Добавить"

HorizontalAlignment="Center"

Background="#67BA80"

Foreground="White"

FontSize="16"

FontWeight="Bold"

FontFamily="Segoe UI"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"

Margin="0 0 0 10"/>

<ScrollViewer Grid.Row="2" VerticalScrollBarVisibility="Auto">

<ListBox ItemsSource="{Binding PartnerList}" Background="White" HorizontalAlignment="Stretch">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel>

<Button Command="{Binding $parent[UserControl].((vm:ShowViewModel)DataContext).ToUpdate}"

CommandParameter="{Binding Id}"

Background="Transparent"

BorderThickness="0"

Padding="0"

HorizontalAlignment="Stretch">

<Border BorderThickness="1" CornerRadius="10" Padding="15" Margin="10" Background="#F4E8D3">

<Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" HorizontalAlignment="Stretch">

<TextBlock FontFamily="Segoe UI" FontSize="16" FontWeight="Bold">

<TextBlock.Text>

<MultiBinding StringFormat="{}{0} | {1}">

<Binding Path="PartnerType.TypeName"/>

<Binding Path="PartnerName"/>

</MultiBinding>

</TextBlock.Text>

</TextBlock>

<TextBlock Text="{Binding DirectorName}" Foreground="#67BA80" FontFamily="Segoe UI" FontSize="14"/>

<TextBlock Text="{Binding Phone}" FontFamily="Segoe UI" FontSize="14"/>

<TextBlock Text="{Binding Rating, StringFormat='Рейтинг: {0}'}" FontFamily="Segoe UI" FontSize="14"/>

</StackPanel>

<TextBlock Grid.Column="1"

Text="{Binding Discount, StringFormat='Скидка: {0}%'}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontWeight="Bold"

FontSize="16"

FontFamily="Segoe UI"/>

</Grid>

</Border>

</Button>

<Button Command="{Binding $parent[UserControl].((vm:ShowViewModel)DataContext).ToSale}"

CommandParameter="{Binding Id}"

Content="Перейти к продажам"

Background="#67BA80"

Foreground="White"

FontSize="14"

FontWeight="Bold"

FontFamily="Segoe UI"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"

Margin="10,5,10,10"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 3. ShowViewModel

using System;

using System.Collections.Generic;

using System.Linq;

using MasterPol.Models;

using Microsoft.EntityFrameworkCore;

using ReactiveUI;

namespace MasterPol.ViewModels

{

public class ShowViewModel : ViewModelBase

{

public List<Partner> \_partnerList;

public List<Partner> PartnerList { get => \_partnerList; set => this.RaiseAndSetIfChanged(ref \_partnerList, value); }

public ShowViewModel()

{

Load();

}

private void Load()

{

PartnerList = MainWindowViewModel.myConnection.Partners.

Include(x => x.PartnerType).

Include(x => x.PartnersProducts).

ThenInclude(x => x.Product).ToList();

}

public void ToAdd()

{

MainWindowViewModel.Instance.PageContent = new Add();

}

public void ToUpdate(int Id)

{

MainWindowViewModel.Instance.PageContent = new Add(Id);

}

public void ToSale(int Id)

{

MainWindowViewModel.Instance.PageContent = new Sale\_history(Id);

}

public async void Delete(int Id)

{

var delete = MainWindowViewModel.myConnection.Partners.Include(x => x.PartnersProducts).FirstOrDefault(x => x.Id == Id);

if (delete != null)

{

MainWindowViewModel.myConnection.PartnersProducts.RemoveRange(delete.PartnersProducts);

MainWindowViewModel.myConnection.Partners.Remove(delete);

await MainWindowViewModel.myConnection.SaveChangesAsync();

Load();

}

}

}

}

Листинг 4. Add

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:DataType="vm:AddViewModel"

xmlns:vm="using:MasterPol.ViewModels"

x:Class="MasterPol.Add">

<Grid Background="White">

<Border Background="#F4E8D3" Padding="25" CornerRadius="10"

HorizontalAlignment="Center" VerticalAlignment="Center">

<StackPanel Width="550" VerticalAlignment="Top" HorizontalAlignment="Stretch">

<TextBlock Text="Редактировать/Добавить"

FontSize="18" FontWeight="Bold"

HorizontalAlignment="Center" Foreground="#67BA80"

FontFamily="Segoe UI"/>

<TextBlock Text="Наименование" FontWeight="Bold" FontFamily="Segoe UI"/>

<TextBox Text="{Binding NewPartner.PartnerName, Mode=TwoWay}" Margin="0,5,0,10" FontFamily="Segoe UI"/>

<TextBlock Text="Тип партнера" FontWeight="Bold" Margin="0,0,0,10" FontFamily="Segoe UI"/>

<ComboBox ItemsSource="{Binding PartnerTypes}"

SelectedItem="{Binding NewPartner.PartnerType, Mode=TwoWay}"

DisplayMemberBinding="{Binding TypeName}"

Margin="0,5,0,10" FontFamily="Segoe UI" />

<TextBlock Text="Рейтинг" FontWeight="Bold" FontFamily="Segoe UI"/>

<TextBox Text="{Binding NewPartner.Rating, Mode=TwoWay}" Margin="0,5,0,10" FontFamily="Segoe UI"/>

<TextBlock Text="Адрес" FontWeight="Bold" FontFamily="Segoe UI"/>

<TextBox Text="{Binding NewPartner.LegalAddress, Mode=TwoWay}" Margin="0,5,0,10" FontFamily="Segoe UI"/>

<TextBlock Text="ФИО директора" FontWeight="Bold" FontFamily="Segoe UI"/>

<TextBox Text="{Binding NewPartner.DirectorName, Mode=TwoWay}" Margin="0,5,0,10" FontFamily="Segoe UI"/>

<TextBlock Text="Телефон" FontWeight="Bold" FontFamily="Segoe UI"/>

<TextBox Text="{Binding NewPartner.Phone, Mode=TwoWay}" Margin="0,5,0,10" FontFamily="Segoe UI"/>

<TextBlock Text="Email" FontWeight="Bold" FontFamily="Segoe UI"/>

<TextBox Text="{Binding NewPartner.Email, Mode=TwoWay}" Margin="0,5,0,10" FontFamily="Segoe UI"/>

<Button Content="Сохранить"

Command="{Binding SavePartner}"

Background="#67BA80"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,10,0,0"

FontSize="16"

FontWeight="Bold"

FontFamily="Segoe UI"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="#67BA80"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,10,0,0"

FontSize="16"

FontWeight="Bold"

FontFamily="Segoe UI"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

</StackPanel>

</Border>

</Grid>

</UserControl>

Листинг 5. AddViewModel

using System;

using System.Collections.Generic;

using System.Linq;

using MasterPol.Models;

using Microsoft.EntityFrameworkCore;

using ReactiveUI;

using Avalonia.Controls;

using MsBox.Avalonia.Enums;

using MsBox.Avalonia;

namespace MasterPol.ViewModels

{

public class AddViewModel : ViewModelBase

{

private Partner? \_newPartner;

public Partner? NewPartner

{

get => \_newPartner;

set => this.RaiseAndSetIfChanged(ref \_newPartner, value);

}

private List<PartnerType> \_partnerTypes;

public List<PartnerType> PartnerTypes

{

get => \_partnerTypes;

set => this.RaiseAndSetIfChanged(ref \_partnerTypes, value);

}

public AddViewModel()

{

LoadPartnerTypes();

NewPartner = new Partner();

}

public AddViewModel(int Id)

{

LoadPartnerTypes();

if (Id != 0)

{

NewPartner = MainWindowViewModel.myConnection.Partners

.Include(x => x.PartnerType)

.FirstOrDefault(x => x.Id == Id) ?? new Partner();

}

else

{

NewPartner = new Partner();

}

}

private void LoadPartnerTypes()

{

PartnerTypes = MainWindowViewModel.myConnection.PartnerTypes.ToList();

}

public async void SavePartner()

{

if (NewPartner == null) return;

// Проверка обязательных полей

if (string.IsNullOrWhiteSpace(NewPartner.PartnerName))

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Поле 'Наименование' обязательно для заполнения.", ButtonEnum.Ok).ShowAsync();

return;

}

if (NewPartner.Rating == 0)

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Поле 'Рейтинг' обязательно для заполнения.", ButtonEnum.Ok).ShowAsync();

return;

}

if (string.IsNullOrWhiteSpace(NewPartner.LegalAddress))

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Поле 'Адрес' обязательно для заполнения.", ButtonEnum.Ok).ShowAsync();

return;

}

if (string.IsNullOrWhiteSpace(NewPartner.DirectorName))

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Поле 'ФИО директора' обязательно для заполнения.", ButtonEnum.Ok).ShowAsync();

return;

}

if (string.IsNullOrWhiteSpace(NewPartner.Phone))

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Поле 'Телефон' обязательно для заполнения.", ButtonEnum.Ok).ShowAsync();

return;

}

if (string.IsNullOrWhiteSpace(NewPartner.Email))

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Поле 'Email' обязательно для заполнения.", ButtonEnum.Ok).ShowAsync();

return;

}

// Сохраняем текущее состояние объекта

var originalPartner = new Partner

{

Id = NewPartner.Id,

PartnerName = NewPartner.PartnerName,

Rating = NewPartner.Rating,

LegalAddress = NewPartner.LegalAddress,

DirectorName = NewPartner.DirectorName,

Phone = NewPartner.Phone,

Email = NewPartner.Email,

PartnerType = NewPartner.PartnerType

};

ButtonResult result = await MessageBoxManager.GetMessageBoxStandard("Подтверждение", "Вы хотите сохранить изменения?", ButtonEnum.YesNo).ShowAsync();

if (result == ButtonResult.Yes)

{

if (NewPartner.Id == 0)

{

MainWindowViewModel.myConnection.Partners.Add(NewPartner);

}

MainWindowViewModel.myConnection.SaveChanges();

await MessageBoxManager.GetMessageBoxStandard("Успех", "Изменения успешно сохранены.", ButtonEnum.Ok).ShowAsync();

MainWindowViewModel.Instance.PageContent = new Show();

}

else if (result == ButtonResult.No)

{

var entry = MainWindowViewModel.myConnection.Entry(NewPartner);

entry.CurrentValues.SetValues(originalPartner);

MainWindowViewModel.Instance.PageContent = new Show();

}

}

public void Exit()

{

MainWindowViewModel.Instance.PageContent = new Show();

}

}

}

Листинг 6. Sale\_history

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:DataType="vm:Sale\_historyViewModel"

xmlns:vm="using:MasterPol.ViewModels"

x:Class="MasterPol.Sale\_history">

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="История продаж партнера"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="#67BA80"

FontFamily="Segoe UI"

Margin="20"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="#67BA80"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,60,0,0"

FontSize="16"

FontWeight="Bold"

FontFamily="Segoe UI"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<ScrollViewer Grid.Row="2" VerticalScrollBarVisibility="Auto">

<ListBox ItemsSource="{Binding PartnersProducts}" Background="White" HorizontalAlignment="Stretch">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderThickness="1" CornerRadius="10" Padding="15" Margin="10" Background="#F4E8D3">

<Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" HorizontalAlignment="Stretch">

<TextBlock FontFamily="Segoe UI" FontSize="16" FontWeight="Bold">

<TextBlock.Text>

<MultiBinding StringFormat="{}{0} | {1}">

<Binding Path="Partner.PartnerName"/>

<Binding Path="Product.ProductName"/>

</MultiBinding>

</TextBlock.Text>

</TextBlock>

<TextBlock Text="{Binding Quantity}" Foreground="#67BA80" FontFamily="Segoe UI" FontSize="14"/>

<TextBlock Text="{Binding SaleDate, StringFormat='{}{0:dd.MM.yyyy}'}" FontFamily="Segoe UI" FontSize="14"/>

</StackPanel>

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 7. Sale\_historyViewModel

using System;

using System.Collections.Generic;

using System.Linq;

using MasterPol.Models;

using Microsoft.EntityFrameworkCore;

using ReactiveUI;

namespace MasterPol.ViewModels

{

public class Sale\_historyViewModel : ViewModelBase

{

private List<PartnersProduct> \_partnersProducts;

public List<PartnersProduct> PartnersProducts

{

get => \_partnersProducts;

set => this.RaiseAndSetIfChanged(ref \_partnersProducts, value);

}

public Sale\_historyViewModel(int partnerId)

{

if (partnerId != 0)

{

PartnersProducts = MainWindowViewModel.myConnection.PartnersProducts

.Include(x => x.Product)

.Include(x => x.Partner)

.Where(x => x.Partner.Id == partnerId)

.ToList();

}

else

{

PartnersProducts = new List<PartnersProduct>();

}

}

public void Exit()

{

MainWindowViewModel.Instance.PageContent = new Show();

}

}

}

Листинг 8. Dll библиотека

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LibraryMasterPol

{

public class MaterialCalculator

{

// Коэффициенты типа продукции и процент брака для каждого типа продукции и материала

private const double ProductType1Coefficient = 1.2;

private const double ProductType2Coefficient = 1.5;

private const double MaterialType1DefectRate = 0.05;

private const double MaterialType2DefectRate = 0.1;

public static int CalculateMaterialRequired(int productId, int materialId, int productCount, double productParam1, double productParam2)

{

// проверка на корректн

if (productId < 1 || productId > 2 || materialId < 1 || materialId > 2)

{

return -1;

}

// выбор коэффициента в зависимости от типа продукции

double productCoefficient;

if (productId == 1)

{

productCoefficient = ProductType1Coefficient;

}

else

{

productCoefficient = ProductType2Coefficient;

}

double defectRate; // выбор процента брака в зависимости от типа материала

if (materialId == 1)

{

defectRate = MaterialType1DefectRate;

}

else

{

defectRate = MaterialType2DefectRate;

}

double materialPerProduct = productParam1 \* productParam2 \* productCoefficient; // материал на 1 единицу продукции

double materialWithDefect = materialPerProduct \* (1 + defectRate);// проуент брака

int totalMaterial = (int)Math.Ceiling(materialWithDefect \* productCount); // общее колво

return totalMaterial;

}

}

}

Листинг 9. Unit-test

using LibraryMasterPol;

namespace TestMasterPol

{

[TestClass]

public class UnitTest1

{

// Простые тесты

//1 тест: Проверяет, что при количестве продукции равном 0, метод возвращает 0 единиц материала

[TestMethod]

public void Test\_CalculateMaterialRequired\_ZeroProductCount()

{

int productId = 1;

int materialId = 1;

int productCount = 0;

double productParam1 = 1.0;

double productParam2 = 1.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(0, result);

}

//2 тест: Проверяет, что если передан некорректный ID продукта (например, 3 вместо существующих значений), метод возвращает -1

[TestMethod]

public void Test\_CalculateMaterialRequired\_Product1\_InvalidProductId()

{

int productId = 3;

int materialId = 1;

int productCount = 100;

double productParam1 = 1.5;

double productParam2 = 1.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(-1, result);

}

[TestMethod]

public void Test\_CalculateMaterialRequired\_Product2\_InvalidMaterialId()

{

int productId = 2;

int materialId = 3;

int productCount = 100;

double productParam1 = 2.0;

double productParam2 = 2.5;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(-1, result);

}

//4 тест: Проверяет, что если один из параметров продукции равен 0, то возвращается 0 единиц материала

[TestMethod]

public void Test\_CalculateMaterialRequired\_ValidInput\_ZeroProductParam()

{

int productId = 1;

int materialId = 1;

int productCount = 100;

double productParam1 = 0;

double productParam2 = 1.5;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(0, result);

}

//5 тест: Проверяет, что при очень большом количестве продукции метод корректно рассчитывает требуемое количество материала, но не выдает ошибку при большом объеме данных

[TestMethod]

public void Test\_CalculateMaterialRequired\_ValidInput\_LargeProductCount()

{

int productId = 1;

int materialId = 1;

int productCount = 10000;

double productParam1 = 1.5;

double productParam2 = 2.5;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreNotEqual(112500, result);

}

// 6 тест: Проверяет, что метод корректно работает при больших значениях параметров продукции и материала

[TestMethod]

public void Test\_CalculateMaterialRequired\_ValidInput\_LargeParams()

{

int productId = 2;

int materialId = 2;

int productCount = 100;

double productParam1 = 1000.0;

double productParam2 = 1000.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreNotEqual(150000000, result);

}

//7 тест: Проверяет точный расчет требуемого материала для обычного случая с заданными параметрами

[TestMethod]

public void Test\_CalculateMaterialRequired\_ValidInput\_ExactValues()

{

int productId = 1;

int materialId = 1;

int productCount = 100;

double productParam1 = 1.0;

double productParam2 = 1.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(126, result);

}

[TestMethod]

public void Test\_CalculateMaterialRequired\_ZeroProductParam1()

{

int productId = 1;

int materialId = 1;

int productCount = 10;

double productParam1 = 0.0;

double productParam2 = 2.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(0, result);

}

//9 тест: Проверяет, что при минимальных значениях параметров для продукции результат корректен

[TestMethod]

public void Test\_CalculateMaterialRequired\_MinimumParam()

{

int productId = 2;

int materialId = 1;

int productCount = 1;

double productParam1 = 0.1;

double productParam2 = 0.1;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(1, result);

}

// 10 тест: Проверка с материалом типа 1 и количеством 0

[TestMethod]

public void Test\_CalculateMaterialRequired\_ZeroMaterialCount()

{

int productId = 1;

int materialId = 1;

int productCount = 0;

double productParam1 = 1.0;

double productParam2 = 1.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(0, result);

}

//Сложные

// 11 тест: Проверяет точность округления. Метод должен корректно округлить результаты при вычислении требуемого материала с учетом брака

[TestMethod]

public void Test\_CalculateMaterialRequired\_RoundingErrorCheck()

{

int productId = 1;

int materialId = 1;

int productCount = 50;

double productParam1 = 1.987;

double productParam2 = 2.345;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

double expectedMaterialPerProduct = productParam1 \* productParam2 \* 1.2;

double expectedMaterialWithDefect = expectedMaterialPerProduct \* (1 + 0.05);

double expectedTotalMaterial = expectedMaterialWithDefect \* productCount;

Assert.IsTrue(Math.Abs(result - expectedTotalMaterial) <= 1, "Ожидаемое значение и результат отличаются слишком сильно");

}

// 12 тест: Проверяет, что для разных типов продукции с различными коэффициентами, результат для второго типа продукции будет больше, если коэффициент больше

[TestMethod]

public void Test\_CalculateMaterialRequired\_VaryingProductCoefficients()

{

int productId1 = 1;

int productId2 = 2;

int materialId = 1;

int productCount = 10;

double productParam1 = 2.0;

double productParam2 = 3.0;

int result1 = MaterialCalculator.CalculateMaterialRequired(productId1, materialId, productCount, productParam1, productParam2);

int result2 = MaterialCalculator.CalculateMaterialRequired(productId2, materialId, productCount, productParam1, productParam2);

Assert.IsTrue(result2 > result1, "Результат для второго типа продукции должен быть больше, так как коэффициент больше");

}

// 13 тест: Проверяет, что при больших значениях для второго типа продукции и материала, количество материала будет достаточно большим

[TestMethod]

public void Test\_CalculateMaterialRequired\_ProductType2\_MaterialType2\_LargeValue()

{

int productId = 2;

int materialId = 2;

int productCount = 500;

double productParam1 = 10.5;

double productParam2 = 8.7;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.IsTrue(result > 10000, "Результат должен быть больше 10000");

}

// 14 тест

[TestMethod]

public void Test\_CalculateMaterialRequired\_InvalidProductType()

{

int productId = 3;

int materialId = 1;

int productCount = 100;

double productParam1 = 3.0;

double productParam2 = 2.0;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

Assert.AreEqual(-1, result, "Метод должен вернуть -1 для несуществующего типа продукции");

}

// 15 тест

[TestMethod]

public void Test\_CalculateMaterialRequired\_RoundingDown()

{

int productId = 2;

int materialId = 1;

int productCount = 10;

double productParam1 = 2.3;

double productParam2 = 3.1;

int result = MaterialCalculator.CalculateMaterialRequired(productId, materialId, productCount, productParam1, productParam2);

double materialPerProduct = productParam1 \* productParam2 \* 1.5;

double defectRate = 0.05;

double expectedMaterial = materialPerProduct \* (1 + defectRate);

double expectedTotalMaterial = expectedMaterial \* productCount;

double expected = Math.Ceiling(expectedTotalMaterial);

Assert.IsTrue(Math.Abs(result - expected) < 5, "Ожидаемая погрешность округления должна быть не более 5");

}

}

}

Листинг 10. 5 сценариев

Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Сохранение данных партнера с валидными значениями |
| **Краткое изложение теста** | Проверка, что при вводе валидных данных партнер сохраняется в базу данных. |
| **Этапы теста** | 1. Открыть форму редактирования партнера.  2. Ввести валидные данные:   * Наименование: "Партнер 1" * Рейтинг: 5 * Адрес: "Москва, ул. Ленина, д. 1" * ФИО директора: "Иванов И.И." * Телефон: "+7 (123) 456-79-99" * Email: "partner1@mail.com" * Тип партнера: "ООО"   3. Нажать кнопку "Сохранить".  4. Подтвердить изменения в диалоговом окне. |
| **Тестовые данные** | * Наименование: "Партнер 1" * Рейтинг: 5 * Адрес: "Москва, ул. Ленина, д. 1" * ФИО директора: "Иванов И.И." * Телефон: "+7 (123) 456-79-99" * Email: "partner1@mail.com" * Тип партнера: "ООО" |
| **Ожидаемый результат** | Партнер успешно сохраняется в базе данных.  Сообщение "Изменения успешно сохранены" отображается в диалоговом окне.  В базе данных появляется новая запись с введенными данными. |
| **Фактический результат** | Партнер сохранен успешно.  Появилось сообщение об успешном сохранении. |
| **Статус** | Зачет |
| **Предварительное условие** | Партнер с данным ID существует или создается новый. |
| **Постусловие** | В базе данных сохраняются изменения для данного партнера. |
| **Примечания/комментарии** | Все данные были корректно сохранены и отображены. |

Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | 2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Отмена редактирования партнера |
| **Краткое изложение теста** | Проверка, что при отмене изменений партнер возвращается к исходным данным. |
| **Этапы теста** | 1. Открыть форму редактирования партнера. 2. Ввести новые данные:  * Наименование: "Партнер 2" * Рейтинг: 4  1. Нажать кнопку "Сохранить". 2. В диалоговом окне нажать "Нет", чтобы отменить изменения. |
| **Тестовые данные** | * Наименование: "Партнер 2" * Рейтинг: 4 |
| **Ожидаемый результат** | Партнер не сохраняет изменения.  Исходные данные партнера остаются без изменений.  Отображается сообщение о том, что изменения были отменены. |
| **Фактический результат** | Изменения отменены.  Данные партнера не изменились. |
| **Статус** | Зачет |
| **Предварительное условие** | Партнер с данным ID существует в базе данных. |
| **Постусловие** | Партнер остается в своем исходном состоянии. |
| **Примечания/комментарии** | Ожидаемое поведение подтвердилось, данные не были изменены. |

Тестовый пример #3:

|  |  |
| --- | --- |
| Тестовый пример **#** | 3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка сохранения партнера с некорректным email |
| **Краткое изложение теста** | Проверка, что система сохраняет данные, если email некорректен. |
| **Этапы теста** | 1. Открыть форму редактирования партнера. 2. Ввести следующие данные:  * Наименование: "Партнер 3" * Рейтинг: 5 * Адрес: "Москва, ул.Ленина, д. 2" * ФИО директора: "Иванов П.П." * Телефон: "+7 (123) 400-99-91" * Email: "vvvemail.com" * Тип партнера: "ЗАО"  1. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Некорректный email: "vvvemail.com" |
| **Ожидаемый результат** | Партнер сохраняется в базу данных. |
| **Фактический результат** | Партнер сохранен. |
| **Статус** | Зачет |
| **Предварительное условие** | Партнер с данным ID существует или создается новый |
| **Постусловие** | Партнер был сохранен в базе данных |
| **Примечания/комментарии** | Тест пройден успешно |

Тестовый пример #4:

|  |  |
| --- | --- |
| Тестовый пример **#** | 4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка сохранения данных партнера с пустыми обязательными полями |
| **Краткое изложение теста** | Проверка, что система не сохраняет партнера, если обязательные поля (например, наименование) пустые. |
| **Этапы теста** | * 1. Открыть форму редактирования партнера.   2. Оставить поле "Наименование" пустым, остальные поля заполнить корректно.   3. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Партнер 1, Рейтинг: 4, Адрес: "Москва", ФИО: "Иванов И.И.", Телефон: "+7 (123) 000-00-90", Email: "partner1@mail.com", Тип партнера: "ООО". |
| **Ожидаемый результат** | Система отобразит сообщение об ошибке, что наименование не может быть пустым.  Партнер не сохранится в базе данных. |
| **Фактический результат** | Сообщение об ошибке появилось, данные не были сохранены. |
| **Статус** | Зачет |
| **Предварительное условие** | Все редактируемые поля обязательны для заполнения. |
| **Постусловие** | Партнер не был сохранен в базе данных. |
| **Примечания/комментарии** | Поведение системы соответствует ожиданиям. |

Test case #5:

|  |  |
| --- | --- |
| **Test Case #** | 5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Отображение всех типов партнеров в выпадающем списке |
| **Краткое изложение теста** | Проверка, что все типы партнеров отображаются в выпадающем списке. |
| **Этапы теста** | 1. Открыть форму редактирования партнера. 2. Перейти к полю "Тип партнера". 3. Проверить, что все типы партнеров, загруженные из базы данных, отображаются в выпадающем списке. |
| **Тестовые данные** | Типы партнеров в базе данных: "ООО", "ЗАО", "ПАО", "ОАО". |
| **Ожидаемый результат** | В выпадающем списке должны отображаться все типы партнеров, загруженные из базы данных. |
| **Фактический результат** | Все типы партнеров отображаются в списке. |
| **Статус** | Зачет |
| **Предварительное условие** | База данных содержит несколько типов партнеров. |
| **Постусловие** | Отображаются все типы партнеров. |
| **Примечания/комментарии** | Тест пройден успешно, все типы партнеров видны в списке. |

## **Приложение Б**

Листинг 1. Создание и заполнение базы данных

CREATE TABLE public.categories (

id int4 NOT NULL DEFAULT nextval('categoty\_id\_seq'::regclass),

"name" varchar(100) NOT NULL,

CONSTRAINT categoty\_pkey PRIMARY KEY (id)

);

CREATE TABLE public.manufacturers (

id int4 NOT NULL DEFAULT nextval('manufacturer\_id\_seq'::regclass),

"name" varchar(100) NOT NULL,

CONSTRAINT manufacturer\_pkey PRIMARY KEY (id)

);

CREATE TABLE public.pick\_up\_points (

id int4 NOT NULL DEFAULT nextval('pick\_up\_point\_id\_seq'::regclass),

address varchar(200) NOT NULL,

CONSTRAINT pick\_up\_point\_pkey PRIMARY KEY (id)

);

CREATE TABLE public.roles (

id int4 NOT NULL DEFAULT nextval('role\_id\_seq'::regclass),

"name" varchar(100) NOT NULL,

CONSTRAINT role\_pkey PRIMARY KEY (id)

);

CREATE TABLE public.status (

id serial4 NOT NULL,

"name" varchar(100) NOT NULL,

CONSTRAINT status\_pkey PRIMARY KEY (id)

);

CREATE TABLE public.suppliers (

id int4 NOT NULL DEFAULT nextval('supplier\_id\_seq'::regclass),

"name" varchar(100) NOT NULL,

CONSTRAINT supplier\_pkey PRIMARY KEY (id)

);

CREATE TABLE public.units (

id serial4 NOT NULL,

"name" varchar(10) NOT NULL,

CONSTRAINT units\_pk PRIMARY KEY (id)

);

CREATE TABLE public.products (

article\_number varchar(100) NOT NULL,

"name" varchar(255) NOT NULL,

id\_unit int4 NULL,

"cost" int4 NOT NULL,

max\_discount\_amount int4 NULL,

id\_manufacturer int4 NULL,

id\_supplier int4 NULL,

id\_category int4 NULL,

current\_discount int4 NULL,

quantity\_in\_stock int4 NULL,

description text NULL,

image varchar(100) NULL,

CONSTRAINT product\_pkey PRIMARY KEY (article\_number),

CONSTRAINT product\_category\_fkey FOREIGN KEY (id\_category) REFERENCES public.categories(id),

CONSTRAINT product\_fk FOREIGN KEY (id\_unit) REFERENCES public.units(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT product\_manufacturer\_fkey FOREIGN KEY (id\_manufacturer) REFERENCES public.manufacturers(id),

CONSTRAINT product\_supplier\_fkey FOREIGN KEY (id\_supplier) REFERENCES public.suppliers(id)

);

CREATE TABLE public.users (

id serial4 NOT NULL,

surname varchar(100) NOT NULL,

"name" varchar(100) NOT NULL,

patronymic varchar(100) NULL,

login text NOT NULL,

"password" text NOT NULL,

role\_id int4 NOT NULL,

CONSTRAINT users\_pkey PRIMARY KEY (id),

CONSTRAINT users\_role\_id\_fkey FOREIGN KEY (role\_id) REFERENCES public.roles(id)

);

CREATE TABLE public.orders (

id serial4 NOT NULL,

date\_order date NULL,

date\_delivery date NULL,

id\_pick\_up\_point int4 NULL,

id\_client int4 NULL,

code int4 NULL,

id\_status int4 NOT NULL,

CONSTRAINT orders\_pkey PRIMARY KEY (id),

CONSTRAINT orders\_fk FOREIGN KEY (id\_client) REFERENCES public.users(id),

CONSTRAINT orders\_pick\_up\_fkey FOREIGN KEY (id\_pick\_up\_point) REFERENCES public.pick\_up\_points(id),

CONSTRAINT orders\_status\_fkey FOREIGN KEY (id\_status) REFERENCES public.status(id)

);

CREATE TABLE public.orders\_products (

id int4 NOT NULL DEFAULT nextval('orderproduct\_id\_seq'::regclass),

order\_id int4 NOT NULL,

product\_article\_number varchar(100) NOT NULL,

count int4 NOT NULL,

CONSTRAINT orderproduct\_pkey PRIMARY KEY (id),

CONSTRAINT orderproduct\_order\_id\_fkey FOREIGN KEY (order\_id) REFERENCES public.orders(id),

CONSTRAINT orderproduct\_product\_article\_number\_fkey FOREIGN KEY (product\_article\_number) REFERENCES public.products(article\_number)

);

INSERT INTO public.categories

(id, "name")

VALUES(1, 'Школьные принадлежности');

INSERT INTO public.categories

(id, "name")

VALUES(2, 'Для офиса');

INSERT INTO public.categories

(id, "name")

VALUES(3, 'Бумага офисная');

INSERT INTO public.categories

(id, "name")

VALUES(4, 'Тетради школьные');

INSERT INTO public.manufacturers

(id, "name")

VALUES(1, 'Pilot');

INSERT INTO public.manufacturers

(id, "name")

VALUES(2, 'Attache');

INSERT INTO public.manufacturers

(id, "name")

VALUES(3, 'Unimax');

INSERT INTO public.manufacturers

(id, "name")

VALUES(4, 'FLEXOFFICE CANDEE');

INSERT INTO public.manufacturers

(id, "name")

VALUES(5, 'Svetocopy');

INSERT INTO public.manufacturers

(id, "name")

VALUES(6, 'Hatber');

INSERT INTO public.manufacturers

(id, "name")

VALUES(7, 'Erich Krause');

INSERT INTO public.manufacturers

(id, "name")

VALUES(8, 'Комус');

INSERT INTO public.manufacturers

(id, "name")

VALUES(9, 'GoodMark');

INSERT INTO public.manufacturers

(id, "name")

VALUES(10, ' GoodMark');

INSERT INTO public.suppliers

(id, "name")

VALUES(1, 'Комус');

INSERT INTO public.suppliers

(id, "name")

VALUES(2, 'Буквоед');

INSERT INTO public.units

(id, "name")

VALUES(1, 'шт.');

INSERT INTO public.units

(id, "name")

VALUES(2, 'уп.');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('А112Т4', 'Ручка шариковая', 1, 12, 30, 1, 1, 1, 2, 6, 'Ручка шариковая с синими чернилами,толщина стержня 7 мм', 'А112Т4.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('A346R4', 'Ручка шариковая автоматическая', 1, 35, 20, 2, 1, 1, 3, 23, 'Ручка шариковая автоматическая с синими чернилами, диаметр шарика 0,9 мм', 'A346R4.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('T564P5', 'Набор шариковых ручек одноразовых', 2, 50, 15, 2, 1, 1, 9, 5, 'Набор шариковых ручек одноразовых Attache Economy Spinner 10 цветов (толщина линии 0.5 мм)', 'T564P5.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('B730E2', 'Ручка шариковая', 1, 41, 10, 3, 1, 1, 3, 45, 'Ручка шариковая одноразовая автоматическая Unimax Fab GP синяя (толщина линии 0.5 мм)', 'B730E2.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('G278R6', 'Ручка шариковая', 1, 15, 30, 4, 2, 2, 7, 23, 'Ручка шариковая FLEXOFFICE CANDEE 0,6 мм, синяя', 'G278R6.png');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('R259E6', 'Бумага офисная', 2, 299, 25, 5, 2, 3, 4, 32, 'Бумага офисная Svetocopy NEW A4 80г 500л', 'R259E6.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('H452A3', 'Тетрадь', 1, 10, 8, 6, 2, 4, 3, 25, 'Тетрадь, 24 листа, Зелёная обложка Hatber/Хатбер, офсет, клетка с полями', 'H452A3.png');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('A543T6', 'Ручка шариковая', 1, 13, 30, 7, 2, 2, 6, 12, 'Ручка шариковая Erich Krause, R-301 ORANGE 0.7 Stick, синий', 'A543T6.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('F719R5', 'Папка-скоросшиватель', 1, 18, 20, 6, 2, 2, 3, 8, 'Папка-скоросшиватель, А4 Hatber/Хатбер 140/180мкм АССОРТИ, пластиковая с перфорацией прозрачный верх', 'F719R5.jpg');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('D419T7', 'Клей-карандаш', 1, 61, 18, 7, 2, 2, 4, 26, 'Клей-карандаш Erich Krause 15 гр.', 'D419T7.png');

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('N592T4', 'Стикеры', 2, 34, 15, 2, 1, 2, 2, 17, 'Стикеры Attache 76x76 мм пастельные желтые (1 блок, 100 листов)', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('N459R6', 'Стикеры', 2, 194, 25, 2, 1, 2, 3, 9, 'Стикеры Attache Selection 51х51 мм неоновые 5 цветов (1 блок, 250 листов)', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('S276E6', 'Скрепки', 2, 46, 30, 8, 1, 2, 2, 14, 'Скрепки Комус металлические никелированные 33 мм (100 штук в упаковке)', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('S453G7', 'Скрепки', 2, 21, 15, 2, 1, 2, 4, 20, 'Скрепки 28 мм Attache металлические (100 штук в упаковке)', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('J539R3', 'Кнопки', 2, 96, 20, 8, 1, 2, 3, 24, 'Кнопки канцелярские Комус металлические цветные (50 штук в упаковке)', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('Z539E3', 'Лента клейкая', 1, 16, 15, 6, 2, 2, 2, 14, 'Лента клейкая 12мм\*33м прозрачная, Hatber/Хатбер', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('A297U6', 'Ручка гелевая', 1, 52, 20, 7, 2, 1, 4, 18, 'Ручка гелевая ErichKrause® G-Cube®, цвет чернил черный', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('A567R4', 'Ручка шариковая', 1, 64, 30, 1, 2, 1, 2, 32, 'Шариковая ручка PILOT SuperGrip 0,7 мм синяя BPGP-10R-F-L', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('K932R4', 'Корректирующая лента', 1, 70, 25, 9, 2, 2, 3, 16, 'Корректор лента 5мм\*4м, блистер, GoodMark', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('K345R5', 'Корректирующая лента', 1, 87, 20, 2, 1, 2, 3, 12, 'Корректирующая лента Attache Economy 5 мм x 5 м', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('K753R3', 'Корректирующая жидкость', 1, 50, 30, 2, 1, 1, 2, 5, 'Корректирующая жидкость (штрих) Attache быстросохнущая 20 мл', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('S563T6', 'Степлер', 1, 231, 25, 2, 1, 2, 4, 17, 'Степлер Attache 8215 до 25 листов черный', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('S425T6', 'Скобы', 2, 25, 20, 2, 1, 2, 2, 16, 'Скобы для степлера №24/6 Attache оцинкованные (1000 штук в упаковке)', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('A340R5', 'Маркер', 1, 66, 15, 9, 2, 2, 4, 27, 'Маркер перманентный GoodMark 2-х сторонний.для СD/DVD черный', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('A384T5', 'Тетрадь', 2, 87, 20, 6, 2, 1, 3, 23, 'Тетрадь, 18 листов, А5 линейка Hatber/Хатбер Серия Зеленая 10шт в блистере', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('K502T9', 'Карандаш-корректор', 1, 70, 25, 9, 2, 2, 2, 7, 'Карандаш-корректор GoodMark, морозостойкий, 8мл, металлический наконечник', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('S512T7', 'Скобы', 2, 25, 15, 10, 2, 2, 3, 32, 'Скобы №10 1000шт, к/к, GoodMark', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('D367R4', 'Клей ПВА', 1, 26, 20, 6, 2, 1, 4, 16, 'Клей ПВА 85г Hatber/Хатбер', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('M892R4', 'Ножницы', 1, 209, 15, 2, 1, 1, 5, 13, 'Ножницы 195 мм Attache с пластиковыми прорезиненными анатомическими ручками бирюзового/черного цвета', NULL);

INSERT INTO public.products

(article\_number, "name", id\_unit, "cost", max\_discount\_amount, id\_manufacturer, id\_supplier, id\_category, current\_discount, quantity\_in\_stock, description, image)

VALUES('Z390R4', 'Лента клейкая', 1, 195, 20, 8, 1, 2, 2, 9, 'Клейкая лента упаковочная Комус 50 мм x 100 м 50 мкм прозрачная', NULL);

INSERT INTO public.roles

(id, "name")

VALUES(1, 'Клиент');

INSERT INTO public.roles

(id, "name")

VALUES(2, 'Администратор');

INSERT INTO public.roles

(id, "name")

VALUES(3, 'Менеджер');

INSERT INTO public.status

(id, "name")

VALUES(1, 'Новый ');

INSERT INTO public.status

(id, "name")

VALUES(2, 'Завершен');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(1, '344288, г. Тихорецк, ул. Чехова, 1');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(2, '614164, г.Тихорецк, ул. Степная, 30');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(3, '394242, г. Тихорецк, ул. Коммунистическая, 43');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(4, '660540, г. Тихорецк, ул. Солнечная, 25');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(5, '125837, г. Тихорецк, ул. Шоссейная, 40');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(6, '125703, г. Тихорецк, ул. Партизанская, 49');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(7, '625283, г. Тихорецк, ул. Победы, 46');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(8, '614611, г. Тихорецк, ул. Молодежная, 50');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(9, '454311, г.Тихорецк, ул. Новая, 19');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(10, '660007, г.Тихорецк, ул. Октябрьская, 19');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(11, '603036, г. Тихорецк, ул. Садовая, 4');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(12, '450983, г.Тихорецк, ул. Комсомольская, 26');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(13, '394782, г. Тихорецк, ул. Чехова, 3');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(14, '603002, г. Тихорецк, ул. Дзержинского, 28');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(15, '450558, г. Тихорецк, ул. Набережная, 30');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(16, '394060, г.Тихорецк, ул. Фрунзе, 43');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(17, '410661, г. Тихорецк, ул. Школьная, 50');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(18, '625590, г. Тихорецк, ул. Коммунистическая, 20');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(19, '625683, г. Тихорецк, ул. 8 Марта');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(20, '400562, г. Тихорецк, ул. Зеленая, 32');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(21, '614510, г. Тихорецк, ул. Маяковского, 47');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(22, '410542, г. Тихорецк, ул. Светлая, 46');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(23, '620839, г. Тихорецк, ул. Цветочная, 8');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(24, '443890, г. Тихорецк, ул. Коммунистическая, 1');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(25, '603379, г. Тихорецк, ул. Спортивная, 46');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(26, '603721, г. Тихорецк, ул. Гоголя, 41');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(27, '410172, г. Тихорецк, ул. Северная, 13');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(28, '420151, г. Тихорецк, ул. Вишневая, 32');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(29, '125061, г. Тихорецк, ул. Подгорная, 8');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(30, '630370, г. Тихорецк, ул. Шоссейная, 24');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(31, '614753, г. Тихорецк, ул. Полевая, 35');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(32, '426030, г. Тихорецк, ул. Маяковского, 44');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(33, '450375, г. Тихорецк ул. Клубная, 44');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(34, '625560, г. Тихорецк, ул. Некрасова, 12');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(35, '630201, г. Тихорецк, ул. Комсомольская, 17');

INSERT INTO public.pick\_up\_points

(id, address)

VALUES(36, '190949, г. Тихорецк, ул. Мичурина, 26');

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(1, 'Никифоров ', 'Всеволод', 'Иванович', 'loginDEjrm2018', 'Cpb+Im', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(2, 'Воронов ', 'Донат', 'Никитевич', 'loginDEpxl2018', 'P6h4Jq', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(3, 'Игнатьева ', 'Евгения', 'Валентиновна', 'loginDEwgk2018', '&mfI9l', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(4, 'Буров ', 'Федот', 'Егорович', 'loginDEpou2018', 'gX3f5Z', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(5, 'Иван ', 'Семёновна', NULL, 'loginDEjwl2018', 'D4ZYHt', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(6, 'Денисов ', 'Дамир', 'Филатович', 'loginDEabf2018', '\*Tasm+', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(7, 'Ершов ', 'Максим', 'Геласьевич', 'loginDEwjm2018', 'k}DJKo', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(8, 'Копылов ', 'Куприян', 'Пётрович', 'loginDEjvz2018', '&|bGTy', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(9, 'Носов ', 'Валерьян', 'Дмитрьевич', 'loginDEuyv2018', '8hhrZ}', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(10, 'Силин ', 'Игорь', 'Авдеевич', 'loginDExdm2018', 'DH68L9', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(11, 'Дроздова ', 'Александра', 'Мартыновна', 'loginDEeiv2018', 'H\*BxlS', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(12, 'Дроздов ', 'Аркадий', 'Геласьевич', 'loginDEfuc2018', 'VuM+QT', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(13, 'Боброва ', 'Варвара', 'Евсеевна', 'loginDEoot2018', 'usi{aT', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(14, 'Чернова ', 'Агата', 'Данииловна', 'loginDElhk2018', 'Okk0jY', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(15, 'Лыткина ', 'Ульяна', 'Станиславовна', 'loginDEazg2018', 's3bb|V', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(16, 'Лаврентьев ', 'Леонид', 'Игнатьевич', 'loginDEaba2018', '#ИМЯ?', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(17, 'Кулаков ', 'Юрий', 'Владленович', 'loginDEtco2018', 'tTKDJB', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(18, 'Соловьёв ', 'Андрей', 'Александрович', 'loginDEsyq2018', '2QbpBN', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(19, 'Корнилова ', 'Марфа', 'Макаровна', 'loginDEpxi2018', '+5X&hy', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(20, 'Белоусова ', 'Любовь', 'Георгьевна', 'loginDEicr2018', '3+|Sn{', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(21, 'Анисимов ', 'Никита', 'Гордеевич', 'loginDEcui2018', 'Zi1Tth', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(22, 'Стрелкова ', 'Фаина', 'Федосеевна', 'loginDEpxc2018', 'G+nFsv', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(23, 'Осипов ', 'Евгений', 'Иванович', 'loginDEqrd2018', 'sApUbt', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(24, 'Владимирова ', 'Иванна', 'Павловна', 'loginDEsso2018', '#ИМЯ?', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(25, 'Кудрявцева ', 'Жанна', 'Демьяновна', 'loginDErsy2018', '{Aa6nS', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(26, 'Матвиенко ', 'Яков', 'Брониславович', 'loginDEvpz2018', 'mS0UxK', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(27, 'Селезнёв ', 'Егор', 'Артёмович', 'loginDEfog2018', 'glICay', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(28, 'Брагин ', 'Куприян', 'Митрофанович', 'loginDEpii2018', 'Ob}RZB', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(29, 'Гордеев ', 'Виктор', 'Эдуардович', 'loginDEhyk2018', '\*gN}Tc', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(30, 'Мартынов ', 'Онисим', 'Брониславович', 'loginDEdxi2018', 'ywLUbA', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(31, 'Никонова ', 'Евгения', 'Павловна', 'loginDEzro2018', 'B24s6o', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(32, 'Полякова ', 'Анна', 'Денисовна', 'loginDEuxg2018', 'K8jui7', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(33, 'Макарова ', 'Пелагея', 'Антониновна', 'loginDEllw2018', 'jNtNUr', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(34, 'Андреева ', 'Анна', 'Вячеславовна', 'loginDEddg2018', 'gGGhvD', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(35, 'Кудрявцева ', 'Кира', 'Ефимовна', 'loginDEpdz2018', '#ИМЯ?', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(36, 'Шилова ', 'Кира', 'Егоровна', 'loginDEyiw2018', 'cnj3QR', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(37, 'Ситников ', 'Игорь', 'Борисович', 'loginDEqup2018', '95AU|R', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(38, 'Русаков ', 'Борис', 'Христофорович', 'loginDExil2018', 'w+++Ht', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(39, 'Капустина ', 'Ульяна', 'Игоревна', 'loginDEkuv2018', 'Ade++|', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(40, 'Беляков ', 'Семён', 'Германнович', 'loginDEmox2018', 'Je}9e7', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(41, 'Гурьев ', 'Ириней', 'Игнатьевич', 'loginDEvug2018', 'lEa{Cn', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(42, 'Мишин ', 'Христофор', 'Леонидович', 'loginDEzre2018', 'N\*VX+G', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(43, 'Лазарева ', 'Антонина', 'Христофоровна', 'loginDEbes2018', 'NaVtyH', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(44, 'Маркова ', 'Ираида', 'Сергеевна', 'loginDEkfg2018', 'r1060q', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(45, 'Носкова ', 'Пелагея', 'Валерьевна', 'loginDEyek2018', 'KY2BL4', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(46, 'Баранов ', 'Станислав', 'Дмитрьевич', 'loginDEloq2018', 'NZV5WR', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(47, 'Ефремов ', 'Демьян', 'Артёмович', 'loginDEjfb2018', 'TNT+}h', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(48, 'Константинов ', 'Всеволод', 'Мэлсович', 'loginDEueq2018', 'GqAUZ6', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(49, 'Ситникова ', 'Ираида', 'Андреевна', 'loginDEpqz2018', 'F0Bp7F', 3);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(50, 'Матвеев ', 'Кондрат', 'Иванович', 'loginDEovk2018', 'JyJM{A', 2);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(51, 'Высоцкая', 'Майя', 'Давидовна', 'loginDEvmd2018', 'QwEr}3', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(52, 'Агеев', 'Дамир', 'Давидович', 'loginDEadd2018', 'XyZ{8tR', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(53, 'Терентьев', 'Филипп', 'Богданович', 'loginDEtfb2018', 'M\*PQ+J', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(54, 'Голубева', 'Лея', 'Петровна', 'loginDEglp2018', 'LmN4vB', 1);

INSERT INTO public.users

(id, surname, "name", patronymic, login, "password", role\_id)

VALUES(55, '1', '1', '1', '1', '1', 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(1, '2022-05-18', '2022-05-24', 3, 51, 311, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(2, '2022-05-19', '2022-05-25', 4, 52, 312, 2);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(3, '2022-05-20', '2022-05-26', 5, NULL, 313, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(4, '2022-05-21', '2022-05-27', 6, NULL, 314, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(5, '2022-05-22', '2022-05-28', 7, NULL, 315, 2);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(6, '2022-05-23', '2022-05-29', 10, NULL, 316, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(7, '2022-05-24', '2022-05-30', 11, NULL, 317, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(8, '2022-05-25', '2022-05-31', 20, 53, 318, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(9, '2022-05-26', '2022-06-01', 30, 54, 319, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(10, '2022-05-27', '2022-06-02', 33, NULL, 320, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(12, '2025-03-12', NULL, NULL, 55, NULL, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(13, '2025-03-12', NULL, NULL, 55, NULL, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(14, '2025-03-12', NULL, NULL, 55, NULL, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(15, '2025-03-12', NULL, NULL, 55, NULL, 1);

INSERT INTO public.orders

(id, date\_order, date\_delivery, id\_pick\_up\_point, id\_client, code, id\_status)

VALUES(16, '2025-03-12', NULL, NULL, 55, NULL, 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(1, 1, 'А112Т4', 10);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(2, 2, 'A346R4', 10);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(3, 3, 'B730E2', 10);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(4, 4, 'H452A3', 10);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(5, 5, 'F719R5', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(6, 6, 'N459R6', 2);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(7, 7, 'J539R3', 20);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(8, 8, 'A567R4', 5);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(9, 9, 'K753R3', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(10, 10, 'S425T6', 5);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(11, 1, 'R259E6', 2);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(12, 2, 'T564P5', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(13, 3, 'G278R6', 10);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(14, 4, 'A543T6', 10);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(15, 5, 'D419T7', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(16, 6, 'S276E6', 15);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(17, 7, 'Z539E3', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(18, 8, 'K932R4', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(19, 9, 'S563T6', 1);

INSERT INTO public.orders\_products

(id, order\_id, product\_article\_number, count)

VALUES(20, 10, 'A340R5', 3);

Листинг 2. Authorization

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:DataType="vm:AuthorizationViewModel"

xmlns:vm="using:WriteErasee.ViewModels"

x:Class="WriteErasee.Authorization">

<UserControl.Styles>

<Style Selector="TextBlock">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style Selector="Button">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

<Setter Property="Background" Value="rgb(73, 140, 81)"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Padding" Value="10,5"/>

<Setter Property="BorderBrush" Value="rgb(73, 140, 81)"/>

<Setter Property="BorderThickness" Value="1"/>

</Style>

</UserControl.Styles>

<Grid Background="White" HorizontalAlignment="Center" VerticalAlignment="Center">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Авторизация" FontSize="24" HorizontalAlignment="Center" Margin="0,20,0,20" />

<StackPanel Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="20">

<TextBlock Text="Логин" FontSize="16" Margin="0,0,0,5" HorizontalAlignment="Center"/>

<TextBox Text="{Binding Login, Mode=TwoWay}" Width="300" Height="40" Margin="0,5" FontSize="16" />

<TextBlock Text="Пароль" FontSize="16" Margin="10,20,0,5" HorizontalAlignment="Center"/>

<TextBox Text="{Binding Password, Mode=TwoWay}" Width="300" Height="40" Margin="0,5" FontSize="16" PasswordChar="\*"/>

<StackPanel IsVisible="{Binding IsCaptchaRequired}" Margin="0,20,0,0">

<TextBlock Text="Введите символы с картинки" FontSize="16" HorizontalAlignment="Center" Margin="0,10"/>

<Image Source="{Binding CaptchaImage}" Width="250" Height="100" Margin="0,5,0,5" HorizontalAlignment="Center"/>

<TextBox Text="{Binding CaptchaInput, Mode=TwoWay}" Width="250" Height="40" Margin="0,5" FontSize="16" HorizontalAlignment="Center"/>

</StackPanel>

<Button Content="Войти" Command="{Binding LoginCommand}" Width="300" Height="50" Margin="0,20,0,5" HorizontalAlignment="Center"/>

<Button Content="Войти как гость" Command="{Binding GuestCommand}" Width="300" Height="50" HorizontalAlignment="Center"/>

</StackPanel>

</Grid>

</UserControl>

Листинг 3. AuthorizationViewModel

using System;

using System.Collections.Generic;

using Avalonia.Media.Imaging;

using MsBox.Avalonia.Enums;

using MsBox.Avalonia;

using System.Threading.Tasks;

using ReactiveUI;

using WriteErasee.Converter;

using WriteErasee.Models;

using Microsoft.EntityFrameworkCore;

using System.Linq;

namespace WriteErasee.ViewModels

{

public class AuthorizationViewModel : ViewModelBase

{

private string \_login = string.Empty;

private string \_password = string.Empty;

private string \_captchaInput = string.Empty;

private string \_captchaText = string.Empty; // Сгенерированный текст CAPTCHA

private int \_failedAttempts = 0;

private DateTime \_lockTime;

private User? \_currentUser;

public string Login

{

get => \_login;

set => this.RaiseAndSetIfChanged(ref \_login, value);

}

public string Password

{

get => \_password;

set => this.RaiseAndSetIfChanged(ref \_password, value);

}

public string CaptchaInput

{

get => \_captchaInput;

set => this.RaiseAndSetIfChanged(ref \_captchaInput, value);

}

public string CaptchaText

{

get => \_captchaText;

set => this.RaiseAndSetIfChanged(ref \_captchaText, value);

}

private bool \_isCaptchaRequired;

public bool IsCaptchaRequired

{

get => \_isCaptchaRequired;

set => this.RaiseAndSetIfChanged(ref \_isCaptchaRequired, value);

}

public ReactiveCommand<System.Reactive.Unit, System.Reactive.Unit> LoginCommand { get; }

public ReactiveCommand<System.Reactive.Unit, System.Reactive.Unit> GuestCommand { get; }

private Bitmap? \_captchaImage;

public Bitmap? CaptchaImage

{

get => \_captchaImage;

set => this.RaiseAndSetIfChanged(ref \_captchaImage, value);

}

public AuthorizationViewModel()

{

LoginCommand = ReactiveCommand.Create(LoginUser);

GuestCommand = ReactiveCommand.Create(ToGuest);

// Генерация CAPTCHA при старте

GenerateCaptcha();

}

private void GenerateCaptcha()

{

CaptchaText = CaptchaGenerator.GenerateCaptcha();

CaptchaImage = CaptchaGenerator.GenerateCaptchaImage(CaptchaText);

}

private async void LoginUser()

{

// Проверяем, заблокирован ли вход

if (\_lockTime > DateTime.Now)

{

await MessageBoxManager.GetMessageBoxStandard(

"Доступ временно заблокирован",

$"Попробуйте снова через {(int)(\_lockTime - DateTime.Now).TotalSeconds} секунд.",

ButtonEnum.Ok

).ShowAsync();

return;

}

// Если CAPTCHA требуется, но введена неверно – обновляем CAPTCHA и выходим

if (IsCaptchaRequired && CaptchaInput != CaptchaText)

{

await MessageBoxManager.GetMessageBoxStandard(

"Ошибка CAPTCHA",

"Введенный код не совпадает с изображением. Попробуйте еще раз.",

ButtonEnum.Ok

).ShowAsync();

GenerateCaptcha(); // Обновляем CAPTCHA

return;

}

using var db = new \_43pKasatkina2upContext(); // Контекст базы данных

\_currentUser = db.Users.Include(u => u.Role).FirstOrDefault(u => u.Login == Login && u.Password == Password);

if (\_currentUser != null)

{

// Если авторизация успешна

OpenUserWindow(\_currentUser);

\_failedAttempts = 0; // Сбрасываем счетчик неудачных попыток

CaptchaInput = string.Empty; // Очищаем поле CAPTCHA

IsCaptchaRequired = false; // Скрываем CAPTCHA

}

else

{

// Увеличиваем счетчик неудачных попыток

\_failedAttempts++;

// Первая неудачная попытка - показываем CAPTCHA

if (\_failedAttempts == 1)

{

IsCaptchaRequired = true; // Показываем CAPTCHA

GenerateCaptcha();

await MessageBoxManager.GetMessageBoxStandard(

"Требуется проверка",

"Неверный логин или пароль. Введите CAPTCHA.",

ButtonEnum.Ok

).ShowAsync();

}

// Вторая неудачная попытка (с CAPTCHA)

else if (\_failedAttempts == 2)

{

// Блокируем кнопку входа на 10 секунд

\_lockTime = DateTime.Now.AddSeconds(10);

await MessageBoxManager.GetMessageBoxStandard(

"Доступ заблокирован",

"Неверная CAPTCHA. Вход заблокирован на 10 секунд.",

ButtonEnum.Ok

).ShowAsync();

// Ожидаем 10 секунд перед повторной попыткой

await Task.Delay(10000);

// После 10 секунд разблокируем кнопку входа

\_lockTime = DateTime.Now; // Сбрасываем время блокировки

}

}

}

private void OpenUserWindow(User user)

{

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel(user) };

}

private void ToGuest()

{

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel() };

}

}

}

Листинг 4. Show

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

xmlns:vm="using:WriteErasee.ViewModels"

xmlns:conv="using:WriteErasee.Converter"

x:DataType="vm:ShowViewModel"

x:Class="WriteErasee.Show">

<UserControl.Resources>

<conv:StrikeThroughConverter x:Key="StrikeThroughConverter"/>

<conv:DiscountToBackgroundConverter x:Key="DiscountConverter"/>

<conv:ImageConverter x:Key="myImageConverter"/>

</UserControl.Resources>

<Grid Background="White" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<DockPanel Grid.Row="0" HorizontalAlignment="Stretch" Margin="10">

<TextBlock Text="{Binding CurrentUserFullName}"

FontSize="16"

FontWeight="Bold"

Foreground="Black"

HorizontalAlignment="Left"

VerticalAlignment="Center"

Margin="10,0"/>

<TextBlock Text="{Binding ProductList.Count, StringFormat='Записей: {0}'}"

FontSize="14"

FontWeight="Bold"

Foreground="Black"

HorizontalAlignment="Right"

VerticalAlignment="Center"

Margin="0,0,10,0"/>

</DockPanel>

<TextBlock Text="Товары"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="rgb(73, 140, 81)"

FontFamily="Comic Sans MS"

Margin="20"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="rgb(73, 140, 81)"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,60,0,20"

FontSize="16"

FontWeight="Bold"

FontFamily="Comic Sans MS"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,10">

<TextBox Width="300" Margin="0,10"

Watermark="Поиск по наименованию товара"

Text="{Binding SearchQuery}" FontSize="14" FontWeight="Bold"/>

<ComboBox HorizontalAlignment="Center" Margin="0,10"

ItemsSource="{Binding DiscountRanges}"

SelectedItem="{Binding SelectedDiscountRange}">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding}" />

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<StackPanel Orientation="Vertical" HorizontalAlignment="Center" Margin="10">

<RadioButton Content="По возрастанию цены"

GroupName="SortGroup"

IsChecked="{Binding IsPriceAscending}"

Command="{Binding SortByPriceAscending}"

Width="250" Margin="0,5"

FontSize="14" FontWeight="Bold"/>

<RadioButton Content="По убыванию цены"

GroupName="SortGroup"

IsChecked="{Binding IsPriceDescending}"

Command="{Binding SortByPriceDescending}"

Width="250" Margin="0,5"

FontSize="14" FontWeight="Bold" />

</StackPanel>

</StackPanel>

<StackPanel Grid.Row="2" HorizontalAlignment="Center" Margin="10" Orientation="Horizontal">

<Button Content="Посмотреть заказ" IsVisible="{Binding IsVisibleCreateOrder}" Command="{Binding ViewOrder}" Width="200" Margin="0 0 10 0"/>

<Button Content="Список заказов" IsVisible="{Binding IsVisibleManager}" Command="{Binding ViewOrders}" Width="200" Margin="0 0 10 0"/>

<Button Content="Добавить новый товар" IsVisible="{Binding IsVisibleAdmin}" Command="{Binding AddProduct}" Width="200"/>

</StackPanel>

<ScrollViewer Grid.Row="3" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

<ListBox ItemsSource="{Binding ProductList}" SelectedItem="{Binding SelectedProduct}"

Background="White" HorizontalAlignment="Stretch" Padding="10" Margin="0,30,0,0">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderThickness="2" BorderBrush="rgb(73, 140, 81)" CornerRadius="10" Padding="15" Margin="10" Background="{Binding CurrentDiscount, Converter={StaticResource DiscountConverter}}">

<Grid RowDefinitions="Auto,Auto">

<Grid Grid.Row="0" HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="120"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Image Height="120" Width="120" Source="{Binding Image, Converter={StaticResource myImageConverter}, FallbackValue='avares://WriteErase/Assets/picture.png'}" HorizontalAlignment="Center" Margin="0,10"/>

<StackPanel Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10">

<TextBlock FontFamily="Comic Sans MS" FontSize="18" FontWeight="Bold" Text="{Binding Name}" Foreground="rgb(73, 140, 81)" Margin="0,5"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Text="{Binding Description}" Foreground="rgb(73, 140, 81)" Margin="0,5"/>

<StackPanel Orientation="Horizontal">

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Text="{Binding Cost, StringFormat='Цена: {0}'}" Margin="0,5" Foreground="rgb(73, 140, 81)" TextDecorations="{Binding HasDiscount, Converter={StaticResource StrikeThroughConverter}}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Text="{Binding FinalPrice, StringFormat='{}{0:F2}'}" Foreground="Red" Margin="10,5,0,5" IsVisible="{Binding HasDiscount}"/>

</StackPanel>

</StackPanel>

<TextBlock Grid.Column="2" FontFamily="Comic Sans MS" FontSize="14" Text="{Binding CurrentDiscount}" Foreground="rgb(118, 227, 131)" FontWeight="Bold" Background="White" Padding="5" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

</Grid>

<StackPanel Grid.Row="1" Orientation="Horizontal"

IsVisible="{Binding $parent[UserControl].((vm:ShowViewModel)DataContext).IsVisibleAdmin}">

<Button Command="{Binding $parent[UserControl].((vm:ShowViewModel)DataContext).Edit}"

Content="Редактировать" CommandParameter="{Binding}" Margin="5"/>

<Button Command="{Binding $parent[UserControl].((vm:ShowViewModel)DataContext).DeleteProduct}"

Content="Удалить" CommandParameter="{Binding}" Margin="5"/>

</StackPanel>

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

<ListBox.ContextMenu>

<ContextMenu>

<MenuItem Header="Добавить к заказу" Command="{Binding AddProductToOrder}"/>

</ContextMenu>

</ListBox.ContextMenu>

</ListBox>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 5. ShowViewModel

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

using System.Linq;

using ReactiveUI;

using WriteErasee.Models;

using Avalonia.Controls;

namespace WriteErasee.ViewModels

{

public class ShowViewModel : ViewModelBase

{

private List<Product> \_productList;

public List<Product> ProductList

{

get => \_productList;

set => this.RaiseAndSetIfChanged(ref \_productList, value);

}

private string \_searchQuery;

public string SearchQuery

{

get => \_searchQuery;

set

{

this.RaiseAndSetIfChanged(ref \_searchQuery, value);

ApplySearchFilter();

}

}

private int \_totalProductCount;

public int TotalProductCount

{

get => \_totalProductCount;

set => this.RaiseAndSetIfChanged(ref \_totalProductCount, value);

}

private int \_filteredProductCount;

public int FilteredProductCount

{

get => \_filteredProductCount;

set => this.RaiseAndSetIfChanged(ref \_filteredProductCount, value);

}

private User? \_currentUser;

public string CurrentUserFullName => \_currentUser != null

? $"{\_currentUser.Surname} {\_currentUser.Name} {\_currentUser.Patronymic}"

: "Гость";

private bool \_canViewOrder;

public bool CanViewOrder

{

get => \_canViewOrder;

set => this.RaiseAndSetIfChanged(ref \_canViewOrder, value);

}

public Product SelectedProduct

{

get => \_selectedProduct;

set => this.RaiseAndSetIfChanged(ref \_selectedProduct, value);

}

private Product \_selectedProduct;

public ReactiveCommand<Product, Unit> AddToOrderCommand { get; }

public ShowViewModel(User? user = null)

{

\_currentUser = user;

Load();

CanViewOrder = false;

}

private void Load()

{

ProductList = MainWindowViewModel.myConnection.Products.Include(x => x.OrdersProducts).ToList();

if (\_currentUser != null)

{

var roleName = \_currentUser.Role?.Name;

if (roleName == "Менеджер")

{

IsVisibleManager = true;

}

else if (roleName == "Администратор")

{

IsVisibleAdmin = true;

IsVisibleManager = true;

}

}

ApplyFilters();

TotalProductCount = ProductList.Count;

}

private bool \_isPriceAscending;

public bool IsPriceAscending

{

get => \_isPriceAscending;

set

{

if (this.RaiseAndSetIfChanged(ref \_isPriceAscending, value) && value)

{

SortByPriceAscending();

}

}

}

private bool \_isPriceDescending;

public bool IsPriceDescending

{

get => \_isPriceDescending;

set

{

if (this.RaiseAndSetIfChanged(ref \_isPriceDescending, value) && value)

{

SortByPriceDescending();

}

}

}

public void SortByPriceAscending()

{

ProductList = ProductList.OrderBy(p => p.Cost).ToList();

this.RaisePropertyChanged(nameof(ProductList));

}

public void SortByPriceDescending()

{

ProductList = ProductList.OrderByDescending(p => p.Cost).ToList();

this.RaisePropertyChanged(nameof(ProductList));

}

private string \_selectedDiscountRange = "Все диапазоны";

public string SelectedDiscountRange

{

get => \_selectedDiscountRange;

set

{

this.RaiseAndSetIfChanged(ref \_selectedDiscountRange, value);

ApplyDiscountFilter();

}

}

public List<string> DiscountRanges { get; } = new List<string>

{

"Все диапазоны",

"0-9,99%",

"10-14,99%",

"15% и более"

};

public void ApplyDiscountFilter()

{

if (SelectedDiscountRange == "Все диапазоны")

{

Load();

}

else

{

switch (SelectedDiscountRange)

{

case "0-9,99%":

ProductList = ProductList.Where(p => p.CurrentDiscount >= 0 && p.CurrentDiscount < 10).ToList();

break;

case "10-14,99%":

ProductList = ProductList.Where(p => p.CurrentDiscount >= 10 && p.CurrentDiscount < 15).ToList();

break;

case "15% и более":

ProductList = ProductList.Where(p => p.CurrentDiscount >= 15).ToList();

break;

}

}

this.RaisePropertyChanged(nameof(ProductList)); // Уведомляем об изменении

}

public void ApplyFilters()

{

// Загружаем все товары, фильтруем их по скидкам на уровне базы данных

var filteredProducts = MainWindowViewModel.myConnection.Products.AsQueryable();

// Фильтрация по диапазону скидок на сервере

if (SelectedDiscountRange != "Все диапазоны")

{

switch (SelectedDiscountRange)

{

case "0-9,99%":

filteredProducts = filteredProducts.Where(p => p.CurrentDiscount >= 0 && p.CurrentDiscount < 10);

break;

case "10-14,99%":

filteredProducts = filteredProducts.Where(p => p.CurrentDiscount >= 10 && p.CurrentDiscount < 15);

break;

case "15% и более":

filteredProducts = filteredProducts.Where(p => p.CurrentDiscount >= 15);

break;

}

}

// Теперь делаем выборку в памяти, если нужно

var productsListInMemory = filteredProducts.ToList(); // Загружаем в память

// Фильтрация по поисковому запросу на стороне клиента

if (!string.IsNullOrEmpty(SearchQuery))

{

productsListInMemory = productsListInMemory

.Where(p => p.Name.Contains(SearchQuery, StringComparison.OrdinalIgnoreCase))

.ToList();

}

if (IsPriceAscending)

{

productsListInMemory = productsListInMemory.OrderBy(p => p.Cost).ToList();

}

else if (IsPriceDescending)

{

productsListInMemory = productsListInMemory.OrderByDescending(p => p.Cost).ToList();

}

TotalProductCount = filteredProducts.Count(); // Общее количество товаров

FilteredProductCount = productsListInMemory.Count; // Отфильтрованные товары

// Применяем результат фильтрации и сортировки

ProductList = productsListInMemory;

this.RaisePropertyChanged(nameof(ProductList)); // Уведомляем об изменении

}

public void ApplySearchFilter()

{

if (string.IsNullOrEmpty(SearchQuery))

{

// Если поисковый запрос пустой, просто загружаем все товары

Load();

}

else

{

// Фильтруем товары по наименованию уже в памяти, после загрузки

ProductList = ProductList.Where(p => p.Name.Contains(SearchQuery, StringComparison.OrdinalIgnoreCase)).ToList();

this.RaisePropertyChanged(nameof(ProductList)); // Уведомляем об изменении

}

}

public void Exit()

{

MainWindowViewModel.Instance.PageContent = new Authorization();

}

private List<OrdersProduct> \_cartItems = new List<OrdersProduct>();

public List<OrdersProduct> CartItems

{

get => \_cartItems;

set => this.RaiseAndSetIfChanged(ref \_cartItems, value);

}

private Order? \_order;

private List<OrdersProduct> \_ordersProducts = new();

public bool IsVisibleCreateOrder { get => \_isVisibleCreateOrder; set => this.RaiseAndSetIfChanged(ref \_isVisibleCreateOrder, value); }

private bool \_isVisibleCreateOrder = false;

public bool IsVisibleAdmin { get => \_isVisibleAdmin; set => this.RaiseAndSetIfChanged(ref \_isVisibleAdmin, value); }

private bool \_isVisibleAdmin = false;

public bool IsVisibleManager { get => \_isVisibleManager; set => this.RaiseAndSetIfChanged(ref \_isVisibleManager, value); }

private bool \_isVisibleManager = false;

public void AddProductToOrder()

{

bool flagNoProduct = true;

if (\_order == null) CreateOrder();

foreach (var product in \_ordersProducts)

{

if (product.ProductArticleNumber == SelectedProduct.ArticleNumber)

{

product.Count++;

flagNoProduct = false;

}

}

if (flagNoProduct)

{

OrdersProduct ordersProduct = new OrdersProduct

{

OrderId = \_order!.Id,

ProductArticleNumber = SelectedProduct.ArticleNumber,

Count = 1,

Order = \_order!,

ProductArticleNumberNavigation =

MainWindowViewModel.myConnection.Products

.Include(it => it.IdCategoryNavigation)

.Include(it => it.IdManufacturerNavigation)

.Include(it => it.IdSupplierNavigation)

.Include(it => it.IdUnitNavigation)

.First(it => it.ArticleNumber == SelectedProduct.ArticleNumber),

};

\_ordersProducts.Add(ordersProduct);

}

if (\_ordersProducts.Count() > 0) IsVisibleCreateOrder = true;

}

private void CreateOrder()

{

\_order = new Order();

DateTime date = DateTime.Now;

\_order.DateOrder = new DateOnly(date.Year, date.Month, date.Day);

\_order.IdClient = (\_currentUser != null) ? \_currentUser.Id : null;

\_order.IdStatus = 1;

MainWindowViewModel.myConnection.Orders.Add(\_order);

MainWindowViewModel.myConnection.SaveChanges();

}

public void ViewOrder() => MainWindowViewModel.Instance.PageContent = new OrderUser(\_currentUser, \_ordersProducts);

public void DeleteProduct(Product product)

{

MainWindowViewModel.myConnection.Products.Remove(product);

MainWindowViewModel.myConnection.SaveChanges();

Load();

}

public void Edit(Product product) => MainWindowViewModel.Instance!.PageContent = new EditProduct(\_currentUser, product.ArticleNumber);

public void AddProduct() => MainWindowViewModel.Instance.PageContent = new EditProduct(\_currentUser);

public void ViewOrders() => MainWindowViewModel.Instance.PageContent = new ListOrders(\_currentUser);

}

}

Листинг 6. OrderUser

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="WriteErasee.OrderUser"

xmlns:conv="using:WriteErasee.Converter"

xmlns:vm="using:WriteErasee.ViewModels"

x:DataType="vm:OrderUserViewModel">

<UserControl.Resources>

<conv:ImageConverter x:Key="myImageConverter"/>

</UserControl.Resources>

<Grid Background="White" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<DockPanel Grid.Row="0" HorizontalAlignment="Stretch" Margin="10">

<TextBlock Text="{Binding CurrentUserFullName}"

FontSize="16"

FontWeight="Bold"

Foreground="Black"

HorizontalAlignment="Left"

VerticalAlignment="Center"

Margin="10,0"/>

</DockPanel>

<TextBlock Text="Заказ"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="rgb(73, 140, 81)"

FontFamily="Comic Sans MS"

Margin="20"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="rgb(73, 140, 81)"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,60,0,20"

FontSize="16"

FontWeight="Bold"

FontFamily="Comic Sans MS"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<ScrollViewer Grid.Row="1" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

<Grid RowDefinitions="\*, auto">

<ListBox Grid.Row="0" ItemsSource="{Binding OrdersProducts}" Background="White" HorizontalAlignment="Stretch" Padding="10" Margin="0,30,0,0">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderThickness="2" BorderBrush="rgb(73, 140, 81)" CornerRadius="10" Padding="15" Margin="10">

<Grid HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="120"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Image Height="120" Width="120" Source="{Binding ProductArticleNumberNavigation.Image, Converter={StaticResource myImageConverter}, FallbackValue='avares://WriteErase/Assets/picture.png'}"

HorizontalAlignment="Center" Margin="0,10"/>

<StackPanel Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10">

<TextBlock Text="{Binding ProductArticleNumberNavigation.Name}" FontFamily="Comic Sans MS" FontSize="18" FontWeight="Bold" Margin="0,5"/>

<TextBlock Text="{Binding ProductArticleNumberNavigation.Description}" FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock Text="{Binding ProductArticleNumberNavigation.IdManufacturerNavigation.Name, StringFormat=Производитель: {0}}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock Text="{Binding ProductArticleNumberNavigation.IdCategoryNavigation.Name, StringFormat=Категория: {0}}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock Text="{Binding ProductArticleNumberNavigation.IdSupplierNavigation.Name, StringFormat=Поставщик: {0}}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5">

<TextBlock.Text>

<MultiBinding StringFormat="Количество на складе: {0} {1}">

<Binding Path="ProductArticleNumberNavigation.QuantityInStock"/>

<Binding Path="ProductArticleNumberNavigation.IdUnitNavigation.Name"/>

</MultiBinding>

</TextBlock.Text>

</TextBlock>

<TextBlock Text="{Binding ProductArticleNumberNavigation.FinalPrice, StringFormat=Цена: {0} руб.}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock Text="{Binding ProductArticleNumberNavigation.CurrentDiscount, StringFormat=Скидка: {0} %}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

</StackPanel>

<StackPanel Grid.Column="2" Margin="10">

<TextBlock Text="{Binding Count, StringFormat=Количество: {0}}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<Button Command="{Binding $parent[UserControl].((vm:OrderUserViewModel)DataContext).AddUnit}"

Content="+" CommandParameter="{Binding}"/>

<Button Command="{Binding $parent[UserControl].((vm:OrderUserViewModel)DataContext).DeleteUnit}"

Content="-" CommandParameter="{Binding}"/>

</StackPanel>

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<StackPanel Grid.Row="1">

<TextBlock Text="{Binding FullCost, StringFormat=Сумма заказа: {0:F2} руб.}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock Text="{Binding FullDiscount, StringFormat=Скидка: {0:F2} %}"

FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<TextBlock Text="Выберите пункт выдачи: " FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"/>

<ComboBox Grid.Row="1" Grid.Column="1" ItemsSource="{Binding PickUpPoint}"

SelectedItem="{Binding Order.IdPickUpPointNavigation}" HorizontalAlignment="Stretch" Margin="5">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Address}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<Button Command="{Binding SaveOrder}" Content="Сформировать заказ"/>

</StackPanel>

</Grid>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 7. OrderUserViewModel

using DynamicData.Kernel;

using Microsoft.EntityFrameworkCore;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WriteErasee.Models;

namespace WriteErasee.ViewModels

{

public class OrderUserViewModel : ViewModelBase

{

public double FullCost { get => \_fullCost; set => this.RaiseAndSetIfChanged(ref \_fullCost, value); }

private double \_fullCost = 0;

public double FullDiscount { get => \_fullDiscount; set => this.RaiseAndSetIfChanged(ref \_fullDiscount, value); }

private double \_fullDiscount = 0;

public string CurrentUserFullName { get => \_currentUserFullName; set => this.RaiseAndSetIfChanged(ref \_currentUserFullName, value); }

private string \_currentUserFullName = "Гость";

User? \_currentUser;

public Order Order { get => \_order; set => this.RaiseAndSetIfChanged(ref \_order, value); }

Order \_order;

public List<OrdersProduct>? OrdersProducts { get => \_ordersProducts; set => this.RaiseAndSetIfChanged(ref \_ordersProducts, value); }

List<OrdersProduct>? \_ordersProducts;

public List<PickUpPoint> PickUpPoint { get => \_pickUpPoint; set => this.RaiseAndSetIfChanged(ref \_pickUpPoint, value); }

List<PickUpPoint> \_pickUpPoint;

public OrderUserViewModel(User? user, List<OrdersProduct> ordersProducts)

{

\_currentUser = user;

if (user != null) CurrentUserFullName = user.Surname + " " + user.Name + " " + user.Patronymic;

OrdersProducts = ordersProducts;

PickUpPoint = MainWindowViewModel.myConnection.PickUpPoints.ToList();

Order = MainWindowViewModel.myConnection.Orders.Include(it => it.IdPickUpPointNavigation)

.First(it => it.Id == ordersProducts[0].OrderId);

UpdateCost();

}

private void UpdateCost()

{

FullCost = 0;

FullDiscount = 0;

double cost = 0;

if (OrdersProducts != null)

{

OrdersProducts.ForEach(it =>

{

double costProd = it.ProductArticleNumberNavigation.Cost;

double currentDiscount = (double)it.ProductArticleNumberNavigation.CurrentDiscount;

cost += (it.Count \* costProd);

double costOneProd = (costProd / 100) \* (100 - currentDiscount);

FullCost += it.Count \* costOneProd;

});

if (cost > 0) FullDiscount = (1.0 - (FullCost / cost)) \* 100;

}

}

public void AddUnit(OrdersProduct product)

{

foreach (OrdersProduct item in OrdersProducts)

if (item.ProductArticleNumber == product.ProductArticleNumber)

item.Count++;

OrdersProducts = new List<OrdersProduct>(OrdersProducts);

UpdateCost();

}

public void DeleteUnit(OrdersProduct product)

{

if (product.Count == 1)

{

if (OrdersProducts.Count == 1) OrdersProducts = null;

else

{

OrdersProducts.Remove(product);

OrdersProducts = new List<OrdersProduct>(OrdersProducts);

}

}

else

{

foreach (OrdersProduct item in OrdersProducts)

if (item.ProductArticleNumber == product.ProductArticleNumber)

item.Count--;

OrdersProducts = new List<OrdersProduct>(OrdersProducts);

}

UpdateCost();

}

/// <summary>

/// Создание заказа

/// </summary>

public void SaveOrder()

{

try

{

int count = 0;

foreach (var item in OrdersProducts)

{

if (item.ProductArticleNumberNavigation.QuantityInStock > 3) count++;

}

DateOnly dateOrder = new DateOnly(Order.DateOrder!.Value.Year,

Order.DateOrder!.Value.Month, Order.DateOrder!.Value.Day);

//срок доставки – 3 дня

if (OrdersProducts.Count == count) Order.DateDelivery = dateOrder.AddDays(3);

//иначе 6 дней

else Order.DateDelivery = dateOrder.AddDays(6);

Random rnd = new Random();

//Код генерируется случайным образом (100-999)

Order.Code = rnd.Next(100, 1000);

MainWindowViewModel.myConnection.Orders.Update(Order);

MainWindowViewModel.myConnection.OrdersProducts.AddRange(OrdersProducts);

MainWindowViewModel.myConnection.SaveChanges();

Exit();

}

catch

{

Exit();

}

}

public void Exit()

{

if (\_currentUser != null)

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel(\_currentUser) };

else

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel() };

}

}

}

Листинг 8. ListOrders

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="WriteErasee.ListOrders"

xmlns:vm="using:WriteErasee.ViewModels"

x:DataType="vm:ListOrdersViewModel">

<Grid Background="White" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Заказы клиентов"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="rgb(73, 140, 81)"

FontFamily="Comic Sans MS"

Margin="20"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="rgb(73, 140, 81)"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,60,0,20"

FontSize="16"

FontWeight="Bold"

FontFamily="Comic Sans MS"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<StackPanel Grid.Row="1" Orientation="Horizontal"

HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,10">

<ComboBox HorizontalAlignment="Center" Margin="10"

ItemsSource="{Binding ListSortedType}"

SelectedIndex="{Binding SelectedSortedType}"/>

<ComboBox HorizontalAlignment="Center" Margin="10"

ItemsSource="{Binding ListFilterType}"

SelectedIndex="{Binding SelectedFilterType}"/>

</StackPanel>

<ScrollViewer Grid.Row="2" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

<ListBox ItemsSource="{Binding OrdersPreview}" SelectedItem="{Binding SelectedOrder}"

Background="White" HorizontalAlignment="Stretch" Padding="10" Margin="0,30,0,0">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderThickness="2" BorderBrush="Black" Background="{Binding Color}" CornerRadius="10" Padding="15" Margin="10">

<StackPanel Grid.Column="1" Margin="10">

<TextBlock FontFamily="Comic Sans MS" FontSize="18" FontWeight="Bold" Margin="0,5"

Text="{Binding Id, StringFormat='Заказ #{0}'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="18" Margin="0,5"

Text="{Binding FIO}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding DateOrder, StringFormat='Дата заказа: {0}'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding DateDelivery, StringFormat='Дата доставки: {0}', TargetNullValue='Дата доставки неизвестна'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding IdPickUpPointNavigation.Address, StringFormat='Пункт выдачи: {0}'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding Code, StringFormat='Код получения: {0}'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding IdStatusNavigation.Name, StringFormat='Статус: {0}'}"/>

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 9. ListOrdersViewModel

using Microsoft.EntityFrameworkCore;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using WriteErasee.Models;

namespace WriteErasee.ViewModels

{

public class ListOrdersViewModel : ViewModelBase

{

User? \_currentUser;

public int SelectedSortedType

{

get => \_selectedSortedType;

set

{

this.RaiseAndSetIfChanged(ref \_selectedSortedType, value);

Filter();

}

}

private int \_selectedSortedType = 0;

public int SelectedFilterType

{

get => \_selectedFilterType;

set

{

this.RaiseAndSetIfChanged(ref \_selectedFilterType, value);

Filter();

}

}

private int \_selectedFilterType = 0;

public List<string> ListSortedType { get => \_listSortedType; set => this.RaiseAndSetIfChanged(ref \_listSortedType, value); }

private List<string> \_listSortedType = new()

{

"Без сортировки",

"По возрастанию стоимости",

"По убыванию стоимости",

};

public List<string> ListFilterType { get => \_listFilterType; set => this.RaiseAndSetIfChanged(ref \_listFilterType, value); }

private List<string> \_listFilterType = new()

{

"Все диапазоны",

"от 0 до 10%",

"от 11 до 14%",

"15% и более",

};

public List<Order> Orders { get => \_orders; set => this.RaiseAndSetIfChanged(ref \_orders, value); }

private List<Order> \_orders = new();

public List<Order> OrdersPreview { get => \_ordersPreview; set => this.RaiseAndSetIfChanged(ref \_ordersPreview, value); }

private List<Order> \_ordersPreview = new();

private void Filter()

{

OrdersPreview = Orders;

if (\_selectedSortedType != 0)

{

if (\_selectedSortedType == 1) OrdersPreview = OrdersPreview.OrderBy(it => it.Cost).ToList();

if (\_selectedSortedType == 2) OrdersPreview = OrdersPreview.OrderByDescending(it => it.Cost).ToList();

}

if (\_selectedFilterType != 0)

{

if (\_selectedFilterType == 2) OrdersPreview = OrdersPreview

.Where(it => it.Discount >= 0 && it.Discount <= 10).ToList();

if (\_selectedFilterType == 2) OrdersPreview = OrdersPreview

.Where(it => it.Discount >= 11 && it.Discount <= 14).ToList();

if (\_selectedFilterType == 2) OrdersPreview = OrdersPreview.Where(it => it.Discount >= 15).ToList();

}

}

public ListOrdersViewModel(User? user)

{

\_currentUser = user;

Orders = MainWindowViewModel.myConnection.Orders

.Include(it => it.IdClientNavigation)

.Include(it => it.IdPickUpPointNavigation)

.Include(it => it.IdStatusNavigation)

.ToList();

Orders.ForEach(it => {

it.FIO = (it.IdClientNavigation == null) ? "Гость" :

(it.IdClientNavigation.Surname + " " + it.IdClientNavigation.Name + " " + it.IdClientNavigation.Patronymic );

// Цвет по умолчанию

it.Color = "#ffffff";

int count = 0;

double cost = 0;

List<OrdersProduct> ordersProducts = MainWindowViewModel.myConnection.OrdersProducts

.Where(op => op.OrderId == it.Id).Include(it => it.ProductArticleNumberNavigation).ToList();

foreach (OrdersProduct product in ordersProducts)

{

double costProd = product.ProductArticleNumberNavigation.Cost;

double currentDiscount = (double)product.ProductArticleNumberNavigation.CurrentDiscount;

cost += (product.Count \* costProd);

double costOneProd = (costProd / 100) \* (100 - currentDiscount);

it.Cost += product.Count \* costOneProd;

if (cost > 0)

it.Discount = (1.0 - (it.Cost / cost)) \* 100;

if (product.ProductArticleNumberNavigation.QuantityInStock > 3)

count++;

else if (product.ProductArticleNumberNavigation.QuantityInStock == 0)

it.Color = "#ff8c00"; // Xотя бы одного товара нет на складе

}

//Все товары в заказе есть на складе в наличии более 3 позиций

if (it.Color == "#ffffff" && ordersProducts.Count == count) it.Color = "#20b2aa";

});

Filter();

}

public void Exit()

{

if (\_currentUser != null)

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel(\_currentUser) };

else

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel() };

}

public Order SelectedOrder

{

get => \_selectedOrder;

set

{

this.RaiseAndSetIfChanged(ref \_selectedOrder, value);

MainWindowViewModel.Instance.PageContent = new EditOrder(\_currentUser, SelectedOrder);

}

}

private Order \_selectedOrder;

}

}

Листинг 10. EditProduct

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="WriteErasee.EditProduct"

xmlns:vm="using:WriteErasee.ViewModels"

xmlns:conv="using:WriteErasee.Converter"

x:DataType="vm:EditProductViewModel">

<Grid Background="White" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="{Binding Title}"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="rgb(73, 140, 81)"

FontFamily="Comic Sans MS"

Margin="20"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="rgb(73, 140, 81)"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,60,0,20"

FontSize="16"

FontWeight="Bold"

FontFamily="Comic Sans MS"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<ScrollViewer Grid.Row="1" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

<Grid ColumnDefinitions="auto, \*" RowDefinitions="auto, auto, auto, auto, auto, auto, auto, auto, auto, auto, auto" Margin="10">

<TextBlock Grid.Row="0" Grid.Column="0" Text="Артикл:" IsVisible="{Binding IsNewProduct}"

HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<TextBox Grid.Row="0" Grid.Column="1" Text="{Binding Product.ArticleNumber}" IsVisible="{Binding IsNewProduct}"

Watermark="Арктикл товара" Margin="5"/>

<TextBlock Grid.Row="1" Grid.Column="0" Text="Наименование:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<TextBox Grid.Row="1" Grid.Column="1" Text="{Binding Product.Name}" Watermark="Название товара" Margin="5"/>

<TextBlock Grid.Row="2" Grid.Column="0" Text="Стоимость:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<NumericUpDown Grid.Row="2" Grid.Column="1" Value="{Binding Product.Cost}" Increment="1" Minimum="0" Margin="5" Watermark="0 руб."/>

<TextBlock Grid.Row="3" Grid.Column="0" Text="Текущая скидка:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<NumericUpDown Grid.Row="3" Grid.Column="1" Value="{Binding Product.CurrentDiscount}" Increment="1" Minimum="0" Margin="5" Watermark="0 %"/>

<TextBlock Grid.Row="4" Grid.Column="0" Text="Максимальная скидка:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<NumericUpDown Grid.Row="4" Grid.Column="1" Value="{Binding Product.MaxDiscountAmount}" Increment="1" Minimum="0" Margin="5" Watermark="0 %"/>

<TextBlock Grid.Row="5" Grid.Column="0" Text="Количество:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<StackPanel Grid.Row="5" Grid.Column="1" Orientation="Horizontal">

<NumericUpDown Value="{Binding Product.QuantityInStock}" Increment="1" Minimum="0" Margin="5"/>

<ComboBox Grid.Row="1" Grid.Column="1" ItemsSource="{Binding Units}"

SelectedItem="{Binding Product.IdUnitNavigation}" HorizontalAlignment="Stretch" Margin="5">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

</StackPanel>

<TextBlock Grid.Row="6" Grid.Column="0" Text="Категория:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<ComboBox Grid.Row="6" Grid.Column="1" ItemsSource="{Binding Categories}"

SelectedItem="{Binding Product.IdCategoryNavigation}" HorizontalAlignment="Stretch" Margin="5">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Grid.Row="7" Grid.Column="0" Text="Производитель:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<ComboBox Grid.Row="7" Grid.Column="1" ItemsSource="{Binding Manufacturers}"

SelectedItem="{Binding Product.IdManufacturerNavigation}" HorizontalAlignment="Stretch" Margin="5">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Grid.Row="8" Grid.Column="0" Text="Поставщик:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<ComboBox Grid.Row="8" Grid.Column="1" ItemsSource="{Binding Suppliers}"

SelectedItem="{Binding Product.IdSupplierNavigation}" HorizontalAlignment="Stretch" Margin="5">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<TextBlock Grid.Row="9" Grid.Column="0" Text="Описание:" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 0 10 0"/>

<TextBox Grid.Row="9" Grid.Column="1" Text="{Binding Product.Description}" TextWrapping="Wrap" Margin="5"/>

<Button Grid.Row="10" Grid.ColumnSpan="2" Content="Сохранить" Command="{Binding Save}" HorizontalAlignment="Center" Margin="10"/>

</Grid>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 11. EditProductViewModel

using MsBox.Avalonia.Enums;

using MsBox.Avalonia;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WriteErasee.Models;

using ReactiveUI;

using Microsoft.EntityFrameworkCore;

namespace WriteErasee.ViewModels

{

public class EditProductViewModel : ViewModelBase

{

private User? \_currentUser;

public bool IsNewProduct { get => \_isNewProduct; set => this.RaiseAndSetIfChanged(ref \_isNewProduct, value); }

private bool \_isNewProduct = true;

public string Title { get => \_title; set => this.RaiseAndSetIfChanged(ref \_title, value); }

private string \_title = "Добавление товара";

public Product Product { get => \_product; set => this.RaiseAndSetIfChanged(ref \_product, value); }

private Product \_product = new();

public List<Category> Categories { get => \_categories; set => this.RaiseAndSetIfChanged(ref \_categories, value); }

private List<Category> \_categories = new();

public List<Manufacturer> Manufacturers { get => \_manufacturers; set => this.RaiseAndSetIfChanged(ref \_manufacturers, value); }

private List<Manufacturer> \_manufacturers = new();

public List<Supplier> Suppliers { get => \_suppliers; set => this.RaiseAndSetIfChanged(ref \_suppliers, value); }

private List<Supplier> \_suppliers = new();

public List<Unit> Units { get => \_units; set => this.RaiseAndSetIfChanged(ref \_units, value); }

private List<Unit> \_units = new();

private void Launch()

{

Categories = MainWindowViewModel.myConnection.Categories.ToList();

Manufacturers = MainWindowViewModel.myConnection.Manufacturers.ToList();

Suppliers = MainWindowViewModel.myConnection.Suppliers.ToList();

Units = MainWindowViewModel.myConnection.Units.ToList();

}

public EditProductViewModel(User? user)

{

\_currentUser = user;

Launch();

}

public EditProductViewModel(User? user, string article)

{

Title = "Редактирование товара";

IsNewProduct = false;

\_currentUser = user;

Product = MainWindowViewModel.myConnection.Products

.Include(it => it.IdCategoryNavigation)

.Include(it => it.IdManufacturerNavigation)

.Include(it => it.IdSupplierNavigation)

.Include(it => it.IdUnitNavigation)

.First(it => it.ArticleNumber == article);

Launch();

}

public async Task Save()

{

try

{

if (IsNewProduct)

{

MainWindowViewModel.myConnection.Products.Add(Product);

MainWindowViewModel.myConnection.SaveChanges();

await MessageBoxManager.GetMessageBoxStandard("Добавление", "Товар успешно добавлен", ButtonEnum.Ok).ShowAsync();

}

else

{

MainWindowViewModel.myConnection.Products.Update(Product);

MainWindowViewModel.myConnection.SaveChanges();

await MessageBoxManager.GetMessageBoxStandard("Редактирование", "Данные о товаре изменены", ButtonEnum.Ok).ShowAsync();

}

Exit();

}

catch

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Проверьте правильность заполнения полей и попробуйте еще раз", ButtonEnum.Ok).ShowAsync();

}

}

public void Exit()

{

if (\_currentUser != null)

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel(\_currentUser) };

else

MainWindowViewModel.Instance.PageContent = new Show { DataContext = new ShowViewModel() };

}

}

}

Листинг 12. EditOrder

<UserControl xmlns="https://github.com/avaloniaui"

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

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

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

mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="450"

x:Class="WriteErasee.EditOrder"

xmlns:vm="using:WriteErasee.ViewModels"

xmlns:conv="using:WriteErasee.Converter"

x:DataType="vm:EditOrderViewModel">

<Grid Background="White" VerticalAlignment="Stretch" HorizontalAlignment="Stretch">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Изменение заказа"

Grid.Row="0"

HorizontalAlignment="Center"

FontSize="24"

FontWeight="Bold"

Foreground="rgb(73, 140, 81)"

FontFamily="Comic Sans MS"

Margin="20"/>

<Button Content="Назад"

Command="{Binding Exit}"

Background="rgb(73, 140, 81)"

Foreground="White"

HorizontalAlignment="Center"

Width="150"

Margin="0,60,0,20"

FontSize="16"

FontWeight="Bold"

FontFamily="Comic Sans MS"

BorderThickness="0"

CornerRadius="10"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<ScrollViewer Grid.Row="1" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">

<StackPanel Grid.Column="1" Margin="10">

<TextBlock FontFamily="Comic Sans MS" FontSize="18" FontWeight="Bold" Margin="0,5"

Text="{Binding Order.Id, StringFormat='Заказ #{0}'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="18" Margin="0,5"

Text="{Binding Order.FIO}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding Order.DateOrder, StringFormat='Дата заказа: {0}'}"/>

<StackPanel Orientation="Horizontal">

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0 5 5 5" Text="Дата доставки:"/>

<TextBox Text="{Binding Order.DateDelivery}"/>

</StackPanel>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding Order.IdPickUpPointNavigation.Address, StringFormat='Пункт выдачи: {0}'}"/>

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5"

Text="{Binding Order.Code, StringFormat='Код получения: {0}'}"/>

<StackPanel Orientation="Horizontal">

<TextBlock FontFamily="Comic Sans MS" FontSize="14" Margin="0,5" Text="Статус: "/>

<ComboBox ItemsSource="{Binding Statuses}" SelectedItem="{Binding Order.IdStatusNavigation}">

<ComboBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Name}"/>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

</StackPanel>

<Button Content="Сохранить" Command="{Binding Save}" HorizontalAlignment="Center"/>

</StackPanel>

</ScrollViewer>

</Grid>

</UserControl>

Листинг 13. EditOrderViewModel

using Avalonia.Controls;

using MsBox.Avalonia.Enums;

using MsBox.Avalonia;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WriteErasee.Models;

using Microsoft.EntityFrameworkCore;

namespace WriteErasee.ViewModels

{

public class EditOrderViewModel : ViewModelBase

{

User? \_currentUser;

public List<Status> Statuses { get => \_statuses; set => this.RaiseAndSetIfChanged(ref \_statuses, value); }

List<Status> \_statuses;

public Order Order { get => \_order; set => this.RaiseAndSetIfChanged(ref \_order, value); }

Order \_order;

public EditOrderViewModel(User? user, Order order)

{

\_currentUser = user;

Statuses = MainWindowViewModel.myConnection.Statuses.ToList();

Order = order;

}

public void Exit() => MainWindowViewModel.Instance.PageContent = new ListOrders(\_currentUser);

/// <summary>

/// Изменения статуса заказа и даты доставки

/// </summary>

/// <returns></returns>

public async Task Save()

{

try

{

MainWindowViewModel.myConnection.Orders.Update(Order);

MainWindowViewModel.myConnection.SaveChanges();

await MessageBoxManager.GetMessageBoxStandard("Сохранено", "Данные заказа были изменены", ButtonEnum.Ok).ShowAsync();

Exit();

}

catch

{

await MessageBoxManager.GetMessageBoxStandard("Ошибка", "Проверьте правильность заполнения полей и попробуйте еще раз", ButtonEnum.Ok).ShowAsync();

}

}

}

}

Листинг 14. DLL библиотека

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WriteEraseLibrary

{

public class Calculations

{

public static string[] AvailablePeriods(TimeSpan[] startTimes, int[] durations,

TimeSpan beginWorkingTime, TimeSpan endWorkingTime,

int consultationTime)

{

List<(TimeSpan start, TimeSpan end)> busyPeriods = new List<(TimeSpan start, TimeSpan end)>();

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

{

busyPeriods.Add((startTimes[i], startTimes[i] + TimeSpan.FromMinutes(durations[i])));

}

busyPeriods = busyPeriods.OrderBy(p => p.start).ToList();

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

TimeSpan currentTime = beginWorkingTime;

foreach (var (start, end) in busyPeriods)

{

while (currentTime + TimeSpan.FromMinutes(consultationTime) <= start)

{

freeSlots.Add($"{currentTime:hh\\:mm}-{(currentTime + TimeSpan.FromMinutes(consultationTime)):hh\\:mm}");

currentTime += TimeSpan.FromMinutes(consultationTime);

}

currentTime = end > currentTime ? end : currentTime;

}

while (currentTime + TimeSpan.FromMinutes(consultationTime) <= endWorkingTime)

{

freeSlots.Add($"{currentTime:hh\\:mm}-{(currentTime + TimeSpan.FromMinutes(consultationTime)):hh\\:mm}");

currentTime += TimeSpan.FromMinutes(consultationTime);

}

return freeSlots.ToArray();

}

}

}

Листинг 15. Unit тесты

using WriteEraseLibrary;

namespace TestProject1

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void Test\_NoFreeSlots()

{

TimeSpan[] startTimes = { TimeSpan.FromHours(8) };

int[] durations = { 600 };

TimeSpan beginWorkingTime = TimeSpan.FromHours(8);

TimeSpan endWorkingTime = TimeSpan.FromHours(18);

int consultationTime = 30;

var result = Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

Assert.AreEqual(0, result.Length);

}

[TestMethod]

public void Test\_BusyPeriodsWithNoFreeTime\_ReturnsEmptyArray()

{

// Arrange

TimeSpan[] startTimes = { new TimeSpan(8, 0, 0), new TimeSpan(9, 0, 0) };

int[] durations = { 60, 60 };

TimeSpan beginWorkingTime = new TimeSpan(8, 0, 0);

TimeSpan endWorkingTime = new TimeSpan(10, 0, 0);

int consultationTime = 30;

// Act

var result = WriteEraseLibrary.Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

// Assert

Assert.AreEqual(0, result.Length);

}

[TestMethod]

public void Test\_FullDayBusyWithGapsSmallerThanConsultation()

{

TimeSpan[] startTimes = { TimeSpan.FromHours(8), TimeSpan.FromHours(9), TimeSpan.FromHours(10), TimeSpan.FromHours(11) };

int[] durations = { 50, 50, 50, 50 };

TimeSpan beginWorkingTime = TimeSpan.FromHours(8);

TimeSpan endWorkingTime = TimeSpan.FromHours(12);

int consultationTime = 60;

var result = Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

Assert.AreEqual(0, result.Length);

}

[TestMethod]

public void Test\_ComplexScheduleWithIrregularGaps()

{

// Arrange

TimeSpan[] startTimes =

{

TimeSpan.FromHours(8),

TimeSpan.FromHours(9.5),

TimeSpan.FromHours(12),

TimeSpan.FromHours(14.5),

TimeSpan.FromHours(17)

};

int[] durations = { 30, 45, 90, 60, 60 };

TimeSpan beginWorkingTime = TimeSpan.FromHours(8);

TimeSpan endWorkingTime = TimeSpan.FromHours(18);

int consultationTime = 30;

// Act

var result = WriteEraseLibrary.Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

// Assert

string[] expectedSlots = {

"08:30-09:00", "09:00-09:30",

"10:15-10:45", "10:45-11:15", "11:15-11:45",

"13:30-14:00", "14:00-14:30",

"15:30-16:00", "16:00-16:30", "16:30-17:00"

};

CollectionAssert.AreEqual(expectedSlots, result);

}

[TestMethod]

public void Test\_NoBusyPeriods\_ReturnsFullDaySlots()

{

// Arrange

TimeSpan[] startTimes = new TimeSpan[0];

int[] durations = new int[0];

TimeSpan beginWorkingTime = new TimeSpan(8, 0, 0);

TimeSpan endWorkingTime = new TimeSpan(18, 0, 0);

int consultationTime = 30;

// Act

var result = WriteEraseLibrary.Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

// Assert

Assert.IsTrue(result.Length == 20);

Assert.IsTrue(result.All(slot => TimeSpan.Parse(slot.Split('-')[0]) >= beginWorkingTime && TimeSpan.Parse(slot.Split('-')[1]) <= endWorkingTime));

}

[TestMethod]

public void Test\_EmptyBusyPeriods\_ReturnsFullDaySlots()

{

// Arrange

TimeSpan[] startTimes = new TimeSpan[0];

int[] durations = new int[0];

TimeSpan beginWorkingTime = new TimeSpan(8, 0, 0);

TimeSpan endWorkingTime = new TimeSpan(18, 0, 0);

int consultationTime = 30;

// Act

var result = WriteEraseLibrary.Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

// Assert

Assert.AreEqual(20, result.Length);

Assert.AreEqual("08:00-08:30", result[0]);

Assert.AreEqual("17:30-18:00", result[^1]);

}

[TestMethod]

public void Test\_SingleSlot()

{

TimeSpan[] startTimes = { TimeSpan.FromHours(8) };

int[] durations = { 30 };

TimeSpan beginWorkingTime = TimeSpan.FromHours(8);

TimeSpan endWorkingTime = TimeSpan.FromHours(18);

int consultationTime = 30;

var result = Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

string[] expectedSlots = { "08:30-09:00" };

CollectionAssert.AreNotEqual(expectedSlots, result);

}

[TestMethod]

public void Test\_LateEveningConsultationSlot()

{

TimeSpan[] startTimes = { TimeSpan.FromHours(16) };

int[] durations = { 120 };

TimeSpan beginWorkingTime = TimeSpan.FromHours(8);

TimeSpan endWorkingTime = TimeSpan.FromHours(18);

int consultationTime = 30;

var result = Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

string[] expectedSlots = { "08:00-08:30", "08:30-09:00", "09:00-09:30", "09:30-10:00", "10:00-10:30", "10:30-11:00", "11:00-11:30", "11:30-12:00", "12:00-12:30", "12:30-13:00", "13:00-13:30", "13:30-14:00", "14:00-14:30", "14:30-15:00", "15:00-15:30", "15:30-16:00" };

CollectionAssert.AreEqual(expectedSlots, result);

}

[TestMethod]

public void Test\_SingleFreeSlotAtStart()

{

TimeSpan[] startTimes = { new TimeSpan(9, 0, 0) };

int[] durations = { 60 };

TimeSpan beginWorkingTime = new TimeSpan(8, 0, 0);

TimeSpan endWorkingTime = new TimeSpan(18, 0, 0);

int consultationTime = 30;

var result = Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

Assert.AreEqual(18, result.Length);

Assert.IsTrue(result.Contains("08:00-08:30"));

}

[TestMethod]

public void Test\_OverlappingBusyPeriods()

{

TimeSpan[] startTimes = { new TimeSpan(9, 0, 0), new TimeSpan(9, 15, 0) };

int[] durations = { 30, 30 };

TimeSpan beginWorkingTime = new TimeSpan(8, 0, 0);

TimeSpan endWorkingTime = new TimeSpan(12, 0, 0);

int consultationTime = 30;

// Act

var result = Calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);

// Assert

Assert.AreEqual(6, result.Length);

Assert.IsTrue(result.Contains("08:00-08:30"));

Assert.IsTrue(result.Contains("08:30-09:00"));

}

}

}

Листинг 16. Тестовая документация

Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешное добавление товара с корректными данными |
| **Краткое изложение теста** | Проверить, что при введении всех корректных данных товар успешно добавляется в систему. |
| **Этапы теста** | 1. Открыть страницу добавления товара. 2. Заполнить все поля (артикул, наименование, стоимость, скидка, количество, категория, производитель, поставщик, описание). 3. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Артикул: "12345"  Наименование: "Ручка"  Стоимость: "100"  Текущая скидка: "10"  Максимальная скидка: "20"  Количество: "10"  Категория: "Школьные принадлежности"  Производитель: "Комус"  Поставщик: "Pilot"  Описание: "Описание товара" |
| **Ожидаемый результат** | Товар добавлен в систему, отображается сообщение об успешном добавлении. |
| **Фактический результат** | Товар успешно добавлен, отображается соответствующее сообщение. |
| **Статус** | *Зачет* |
| **Предварительное условие** | Администратор авторизован в системе |
| **Постусловие** | Товар добавлен в базу данных. |
| **Примечания/комментарии** | Нет |

Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | 2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Ошибка при добавлении товара без артикула |
| **Краткое изложение теста** | Проверить, что система не позволяет добавить товар без артикула |
| **Этапы теста** | 1. Открыть страницу добавления товара. 2. Заполнить все поля, кроме артикула. 3. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Артикул: ""  Наименование: "Ручка"  Стоимость: "100"  Текущая скидка: "10"  Максимальная скидка: "20"  Количество: "10"  Категория: "Школьные принадлежности"  Производитель: "Комус"  Поставщик: "Pilot"  Описание: "Описание товара" |
| **Ожидаемый результат** | Появляется сообщение об ошибке, информирующее о том, что артикул является обязательным полем |
| **Фактический результат** | Сообщение об ошибке |
| **Статус** | *Зачет* |
| **Предварительное условие** | Администратор авторизован в системе |
| **Постусловие** | Товар не добавлен в систему |
| **Примечания/комментарии** | Нет |

Тестовый пример #3:

|  |  |
| --- | --- |
| Тестовый пример **#** | 3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Добавление товара с отрицательной стоимостью |
| **Краткое изложение теста** | Проверить, что система не позволяет добавить товар с отрицательной стоимостью |
| **Этапы теста** | 1. Открыть страницу добавления товара. 2. Заполнить все поля с отрицательной стоимостью. 3. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Артикул: "1234"  Наименование: "Ручка"  Стоимость: "-200"  Текущая скидка: "10"  Максимальная скидка: "20"  Количество: "10"  Категория: "Школьные принадлежности"  Производитель: "Комус"  Поставщик: "Pilot"  Описание: "Описание товара" |
| **Ожидаемый результат** | Товар не добавляется в бд |
| **Фактический результат** | Товар не добавляется в бд |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизован в системе |
| **Постусловие** | Товар не добавлен в систему |
| **Примечания/комментарии** | Нет |

Тестовый пример #4:

|  |  |
| --- | --- |
| Тестовый пример **#** | 4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Добавление товара с отсутствующими данными в полях категории, производителя или поставщика |
| **Краткое изложение теста** | Проверить, что система не позволяет добавить товар без заполнения этих обязательных полей. |
| **Этапы теста** | Открыть страницу добавления товара.  Заполнить все поля, кроме категории, производителя или поставщика. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Артикул: "1234"  Наименование: "Ручка"  Стоимость: "-200"  Текущая скидка: "10"  Максимальная скидка: "20"  Количество: "10"  Категория: ""  Производитель: ""  Поставщик: ""  Описание: "Описание товара" |
| **Ожидаемый результат** | Появляется сообщение об ошибке, информирующее о том, что категория является обязательным полем. |
| **Фактический результат** | Сообщение об ошибке |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизован в системе |
| **Постусловие** | Товар не добавлен в систему |
| **Примечания/комментарии** | Нет |

Test case #5:

|  |  |
| --- | --- |
| **Test Case #** | 5 |
| **Приоритет тестирования** | Низкий |
| **Заголовок/название теста** | Проверка добавления товара с пустыми полями |
| **Краткое изложение теста** | Проверить, что система не позволит добавить товар с пустыми обязательными полями |
| **Этапы теста** | 1. Открыть страницу добавления товара. 2. Оставить все обязательные поля пустыми. 3. Нажать кнопку "Сохранить". |
| **Тестовые данные** | Артикул: ""  Наименование: ""  Стоимость: ""  Текущая скидка: ""  Максимальная скидка: ""  Количество: ""  Категория: ""  Производитель: ""  Поставщик: ""  Описание: "" |
| **Ожидаемый результат** | Появляется сообщение об ошибке, информирующее о том, что все обязательные поля должны быть заполнены |
| **Фактический результат** | Сообщение об ошибке |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизован в системе |
| **Постусловие** | Товар не добавлен в систему |
| **Примечания/комментарии** | Нет |