**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](#_Toc192197002)

[**1.** **Разработка программных модулей** 4](#_Toc192197003)

[**1.1** **Импорт данных. Разработка базы данных** 4](#_Toc192197004)

[**1.2** **Avalonia** 6](#_Toc192197005)

[**1.3** **Тестирование** 12](#_Toc192197006)

[**1.4** **Диаграммы** 16](#_Toc192197007)

[**Заключение** 18](#_Toc192197008)

[**Приложение А** 19](#_Toc192197009)

# **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.

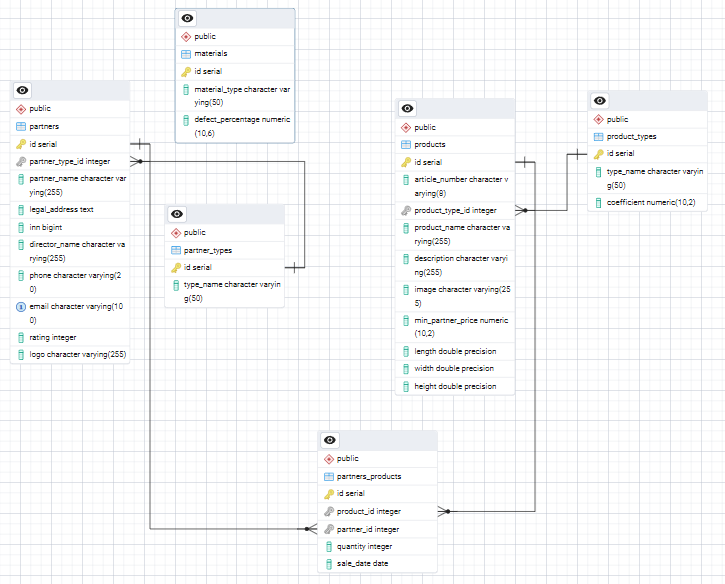


Рисунок №1 - ER Диаграмма

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.

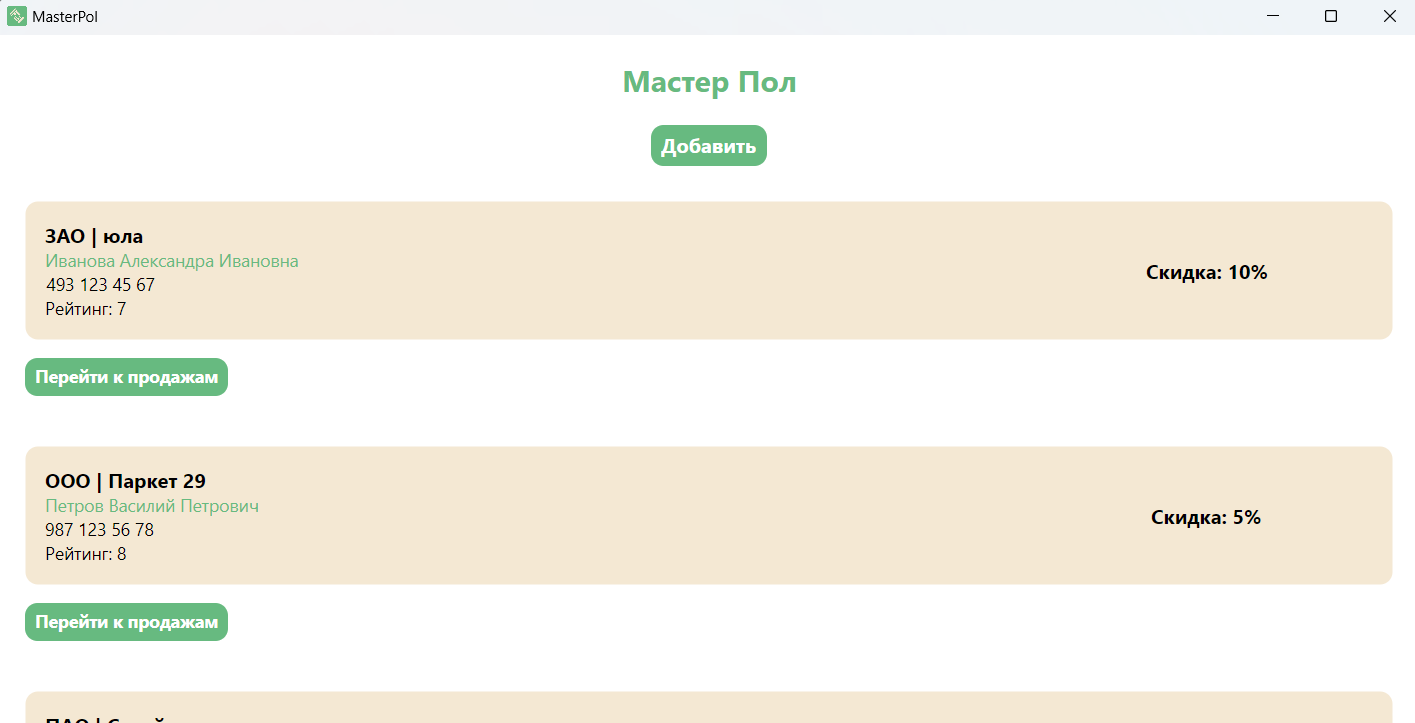


Рисунок №2- Show

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.

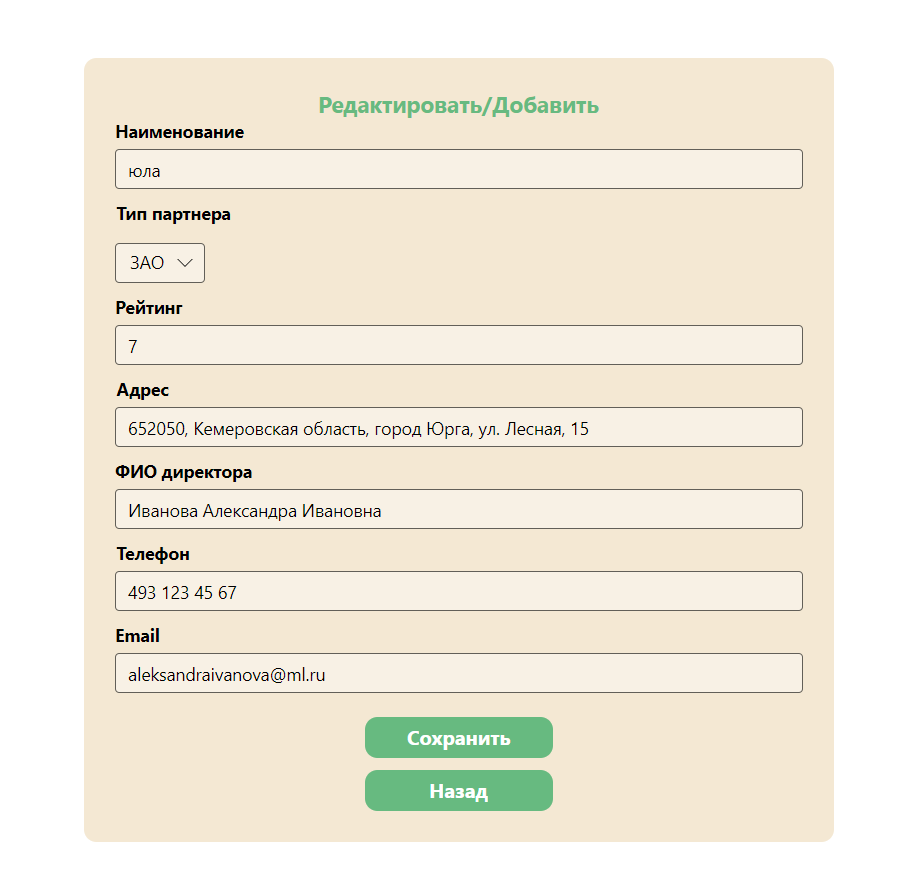


Рисунок №3 - Редактирование

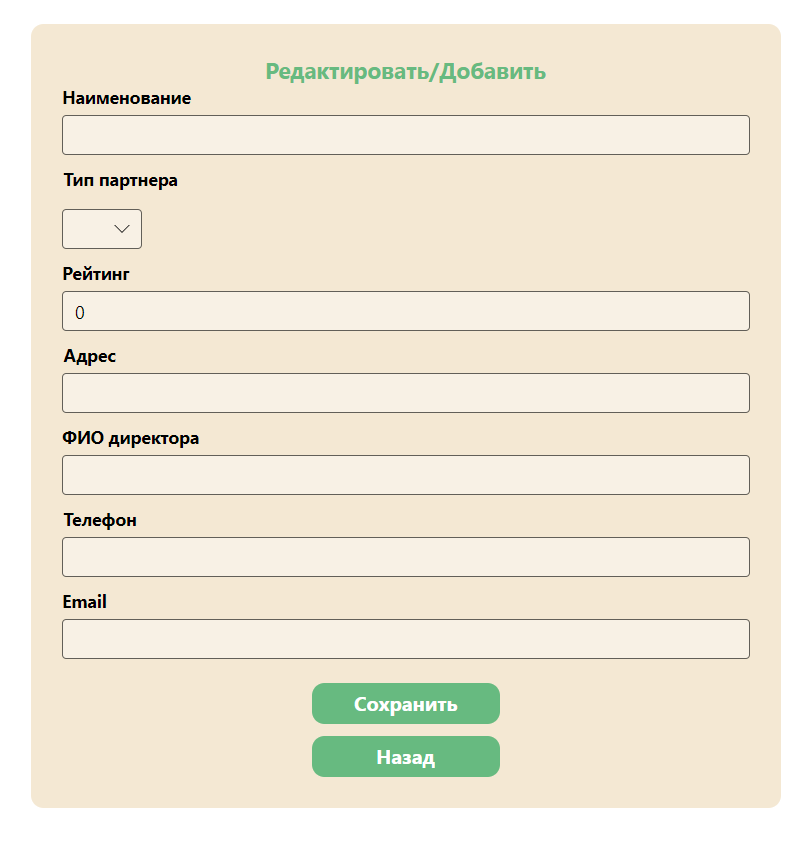


Рисунок №4 - Добавление

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.

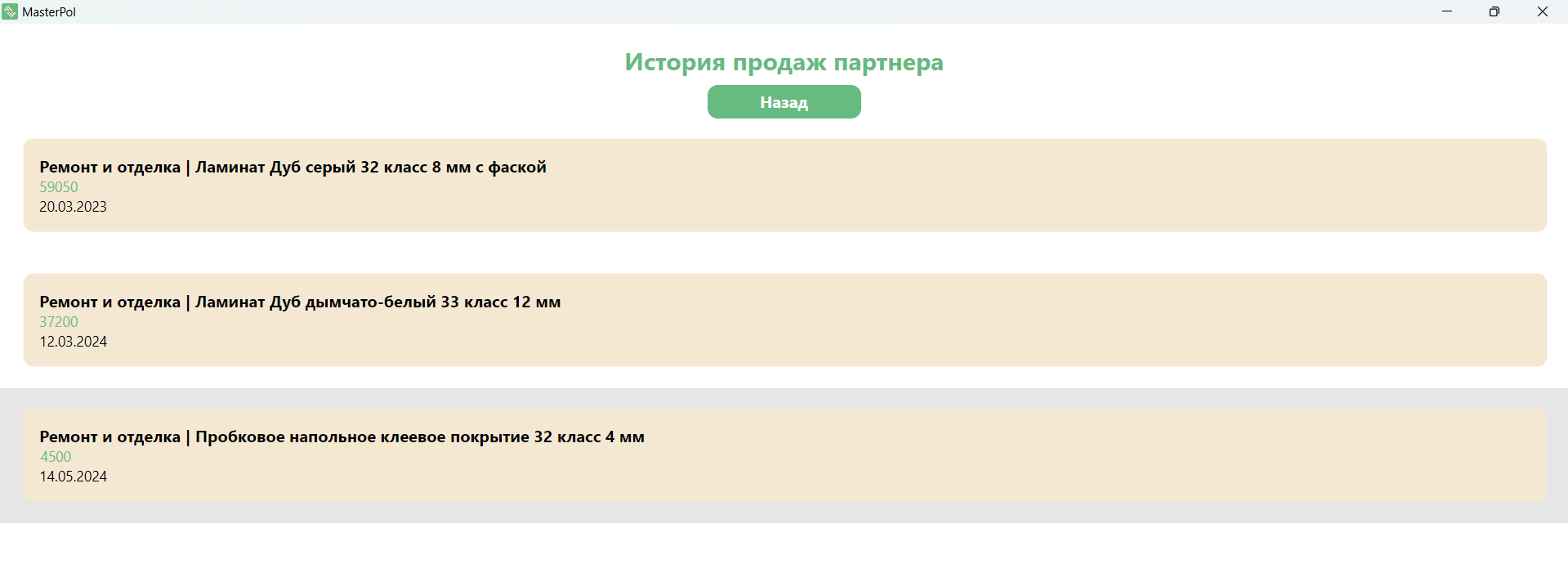


Рисунок №5 – История продаж

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.

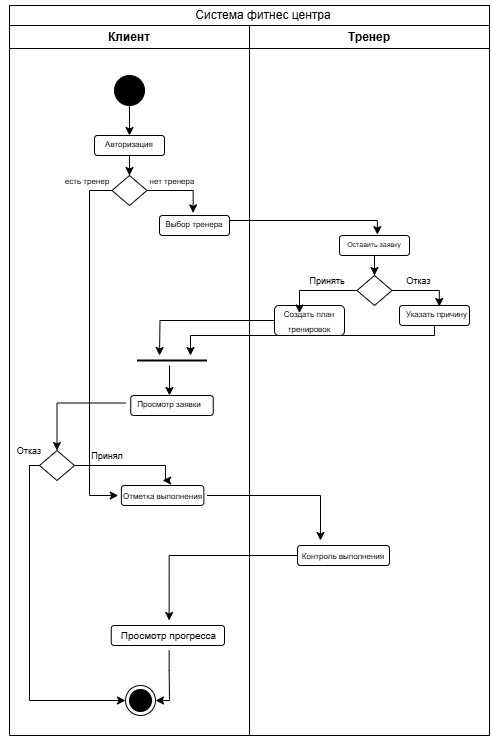


Рисунок №6 – Диаграмма деятельности

## **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. |