**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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Любивая А. А. 21.03.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

**Содержание**

[Введение 3](#_Toc192232767)

[Разработка модулей ПО 5](#_Toc192232768)

[Заключение 27](#_Toc192232769)

[Приложения 29](#_Toc192232770)

Введение

В современном мире информационных технологий разработка программного обеспечения является одной из ключевых задач, стоящих перед специалистами в области компьютерных систем. В рамках учебной программы по предмету "Разработка модулей программного обеспечения для компьютерных систем" (код ПМ.01) была поставлена задача создать программное обеспечение, соответствующее требованиям демонстрационного экзамена 2025 года. Данный проект направлен на реализацию функционала, связанного с расчетом количества материалов, а также на разработку и тестирование алгоритмов редактирования данных партнеров.

Проект включает в себя несколько сессий, каждая из которых охватывает различные аспекты разработки, тестирования и документирования программного обеспечения. Важным элементом проекта является создание библиотеки для расчета количества материалов, а также модульных тестов, которые позволят проверить корректность работы этой библиотеки.

Цели:

1. Создание библиотеки для расчета количества материалов: Разработать метод, который будет выполнять расчеты на основе заданных параметров, обеспечивая точность и надежность результатов.
2. Разработка модульных тестов: Создать проект модульных тестов, который будет проверять функциональность разработанного метода, обеспечивая его покрытие тестами низкой и высокой сложности.
3. Описание сценариев редактирования данных партнера: Подготовить пять тестовых сценариев, которые будут демонстрировать различные исходы работы алгоритма редактирования, соответствующие требованиям предметной области.
4. Анализ проектной документации: Изучить файл "Задание на проектирование" и создать диаграммы использования и деятельности, которые визуализируют структуру и логику системы.
5. Подготовка репозитория и отчетной документации: Сформировать репозиторий с проектом, включающим все необходимые файлы, а также подготовить текстовый отчет с требованиями и титульным листом.

Задачи:

1. Анализ документации: Изучить документ "КОД 09.02.07-2-2025 Том 1", чтобы понять содержание демонстрационного экзамена и требования к проекту.
2. Реализация метода расчета: Разработать метод в виде библиотеки, который будет отвечать за расчет количества материалов, учитывая различные параметры.
3. Создание модульных тестов: Разработать 15 модульных тестов, разделенных на 10 тестов низкой сложности и 5 тестов высокой сложности, с понятными именами, отражающими их суть.
4. Подготовка тестовых сценариев: Описать пять различных сценариев редактирования данных партнера, учитывая возможные исходы и условия для выполнения или отклонения редактирования.
5. Создание диаграмм: Составить диаграммы использования и деятельности на основе анализа файла "Задание на проектирование".
6. Формирование репозитория: Подготовить репозиторий на платформе GitHub, включающий все файлы проекта, а также заполненный файл README.md и текстовый отчет.

Разработка модулей ПО

* 1. Создание базы данных

На основании предоставленной информации о производственной компании «Мастер пол», была разработана база данных, которая будет поддерживать все необходимые подсистемы. Давайте подробно рассмотрим каждую таблицу и её назначение в контексте системы. (см. Рис. 1, Приложение А)

1. Таблица partners\_type

Назначение: Эта таблица служит для хранения различных типов партнеров, что позволяет классифицировать их по категориям. Это может включать оптовых покупателей, розничные магазины, дистрибьюторов и т.д.

Поля:

id: Уникальный идентификатор типа партнера.

name: Название типа партнера (например, "Оптовый", "Розничный").

Связи:

Таблица partners ссылается на partners\_type, что позволяет каждому партнеру иметь определенный тип.

Таблица products\_type

Назначение: В этой таблице хранятся типы продукции, производимой компанией. Каждый тип может иметь свои характеристики и коэффициенты, которые влияют на ценообразование и скидки.

Поля:

id: Уникальный идентификатор типа продукта.

product\_type: Название типа продукта (например, "Ламинат", "Паркет").

koeff: Коэффициент, который может использоваться для расчета цен и скидок.

Связи:

Таблица products ссылается на products\_type, что позволяет каждому продукту принадлежать к определенному типу.

Таблица materials\_type

Назначение: Эта таблица содержит информацию о типах материалов, используемых в производственном процессе. Это важно для контроля качества сырья и управления запасами.

Поля:

id: Уникальный идентификатор типа материала.

material\_type: Название типа материала (например, "Дерево", "Ламинат").

defect: Параметр, указывающий на допустимый уровень дефектов для данного материала.

Связи:

Таблица materials может ссылаться на materials\_type для указания типа материала, который используется в производстве.

Таблица products

Назначение: Таблица для хранения информации о продукции, выпускаемой компанией. Включает все необходимые характеристики для управления продуктами и их продажами.

Поля:

id: Уникальный идентификатор продукта.

product\_type: Ссылка на тип продукта из таблицы products\_type.

product\_name: Название продукта (например, "Ламинат 32 класс").

article: Артикул продукта, используемый для идентификации.

min\_cost: Минимальная стоимость продукта для партнеров.

photo: Путь к изображению продукта.

length, width, height: Размеры упаковки продукта.

Связи:

Связь с таблицей partners\_products для отслеживания закупок и продаж.

Таблица partners

Назначение: Хранит информацию о партнерах, которые закупают продукцию. Это позволяет управлять отношениями с партнерами и отслеживать их активность.

Поля:

id: Уникальный идентификатор партнера.

partner\_type: Ссылка на тип партнера из таблицы partners\_type.

company\_name: Наименование компании-партнера.

FIO\_director: ФИО директора компании.

email: Контактный адрес электронной почты.

phone: Номер телефона.

ur\_address: Юридический адрес компании.

inn: ИНН компании.

rating: Рейтинг партнера, который может использоваться для расчета скидок.

logo: Путь к логотипу компании.

place\_sale: Места, где партнер реализует продукцию.

history: История реализации продукции.

Связи:

Связь с таблицей partners\_products для отслеживания закупок, а также с таблицей partners\_type.

Таблица partners\_products

Назначение: Связывает партнеров с продуктами, которые они закупают. Это позволяет отслеживать объемы закупок и даты.

Поля:

id: Уникальный идентификатор записи.

product\_id: Ссылка на продукт из таблицы products.

partner\_id: Ссылка на партнера из таблицы partners.

quantity: Количество закупленного продукта.

date: Дата закупки.

Связи:

Эта таблица связывает partners с products, что позволяет отслеживать, какие продукты закупаются каждым партнером.

Таблица materials

Назначение: Хранит информацию о материалах, используемых для производства продуктов. Это важно для управления запасами и контроля качества.

Поля:

id: Уникальный идентификатор материала.

material\_type: Ссылка на тип материала из таблицы materials\_type.

material\_name: Название материала (например, "Дуб", "Бук").

quantity: Количество материала на складе.

supplier: Поставщик материала.

cost: Стоимость материала.

Связи:

Связь с таблицей materials\_type для указания типа материала, а также возможные связи с таблицей products для отслеживания, какие материалы используются для производства каких продуктов.

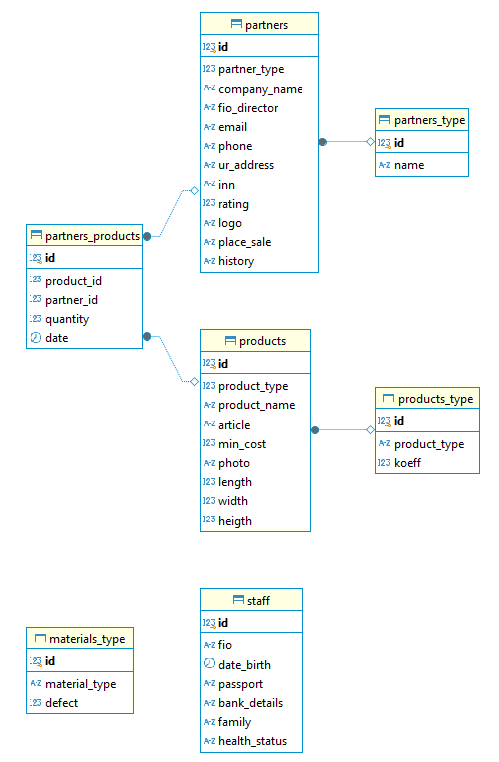


Рисунок №1 – Схема базы данных

Заполненные таблицы:

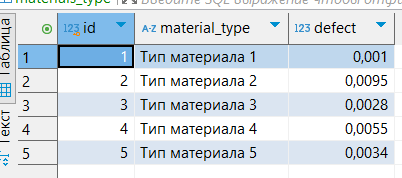


Рисунок №2 – Заполненная таблица 1

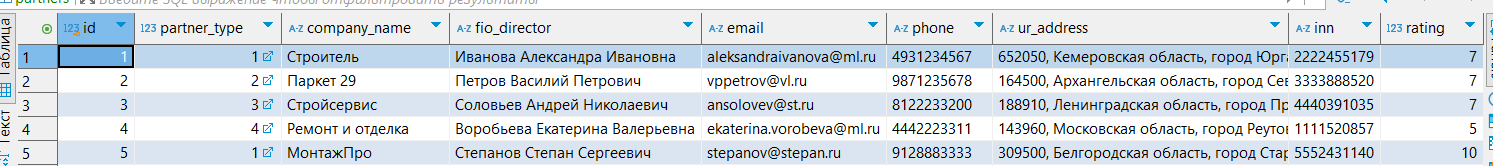


Рисунок №3 – Заполненная таблица 2

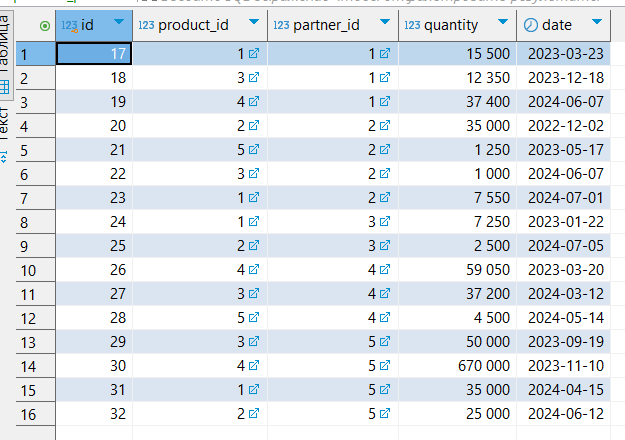


Рисунок №4 – Заполненная таблица 3

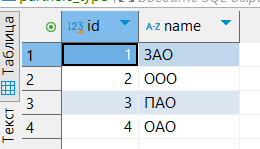


Рисунок №5 – Заполненная таблица 4

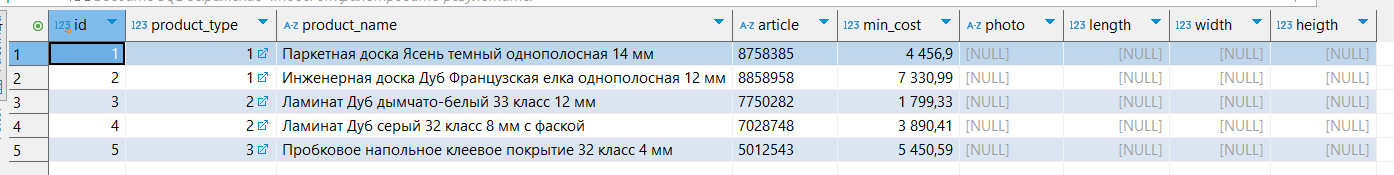


Рисунок №6 – Заполненная таблица 5

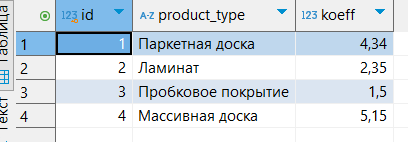


Рисунок №7 – Заполненная таблица 6

* 1. Разработка подсистемы для работы с партнерами компании
     1. Разработка страницы вывода партнеров (см. Рисунок 8)

Основной функционал страницы ShowPartner

Отображение заголовка:

В верхней части страницы находится заголовок "Партнёры", который выделен с помощью увеличенного шрифта и жирного начертания. Это помогает пользователю быстро понять, о чем идет речь на странице.

Кнопка "Добавить":

Кнопка, расположенная под заголовком, позволяет пользователю перейти к форме добавления нового партнера. При нажатии на кнопку срабатывает команда ToAddPartner, которая изменяет содержимое страницы на форму добавления партнера (AddPartner).

Список партнеров:

В основном разделе страницы находится прокручиваемый список (ScrollViewer), который содержит ListBox для отображения всех партнеров.

Список заполняется данными из PartnerList, который загружается в конструкторе ShowPartnerVM с использованием Entity Framework для получения информации о партнерах, включая их типы и связанные продукты.

Шаблон для отображения каждого партнера:

Каждый элемент списка отображается с использованием DataTemplate, который включает:

Название типа партнера и название компании.

ФИО директора, номер телефона и адрес электронной почты.

Кнопка "Редактировать", которая позволяет пользователю редактировать информацию о выбранном партнере. При нажатии на эту кнопку вызывается команда ToUpdatePartner, передавая идентификатор партнера как параметр.

Кнопка "Просмотр продуктов", которая открывает список продуктов, связанных с выбранным партнером, вызывая команду ToShowPartnerProduct с идентификатором партнера.

Отображение скидки:

В правой части каждого элемента списка отображается скидка, предоставляемая партнеру. Это может быть полезно для быстрого анализа условий сотрудничества.

Взаимодействие с данными

ViewModel:

ShowPartnerVM является моделью представления, которая управляет данными и логикой для этой страницы. Она загружает список партнеров и предоставляет команды для навигации (добавление, редактирование и просмотр продуктов).

Команды:

Использование команд в кнопках позволяет отделить логику взаимодействия с пользователем от представления, что соответствует принципам MVVM (Model-View-ViewModel). Это делает код более чистым и поддерживаемым.

Страница построена с использованием XAML, что позволяет легко определять визуальные компоненты и их взаимодействие.

Структура с использованием Grid, StackPanel и Border обеспечивает удобное размещение элементов и визуальную иерархию.

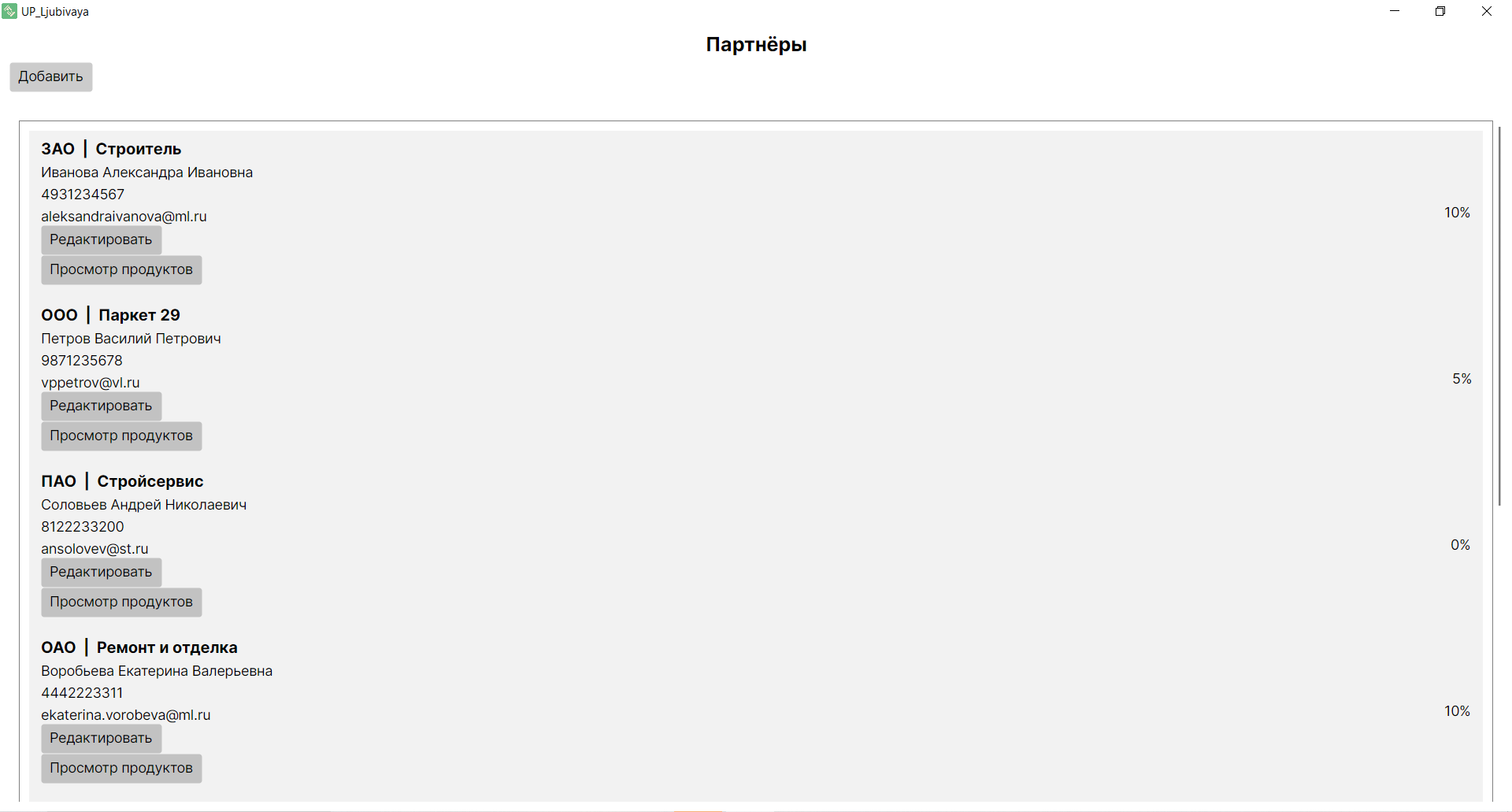


Рисунок №8 – Страница вывода партнеров

* 1. Страница редактирования/добавления партнера (Рисунок 9, 10)

Основной функционал страницы AddPartner

Заголовок страницы:

В верхней части страницы отображается заголовок, который изменяется в зависимости от режима: "Добавление партнера" или "Изменение партнера". Это позволяет пользователю понять, какую операцию он выполняет.

Кнопка "Назад":

Кнопка, расположенная рядом с заголовком, позволяет пользователю вернуться на страницу со списком партнеров (ShowPartner). При нажатии на кнопку вызывается команда ToShowPartner, что обеспечивает удобную навигацию.

Форма ввода данных:

Основная часть страницы содержит форму, где пользователь может ввести данные о новом партнере или изменить существующие. Форма включает следующие поля:

Наименование компании: текстовое поле для ввода названия компании партнера.

Тип партнера: выпадающий список (ComboBox), позволяющий выбрать тип партнера из загруженного списка PartnersTypes. Это обеспечивает удобный выбор и предотвращает ошибки ввода.

Рейтинг: текстовое поле для ввода рейтинга партнера. В коде предусмотрена проверка, чтобы рейтинг был целым неотрицательным числом.

Адрес: текстовое поле для ввода юридического адреса партнера.

ФИО директора: текстовое поле для ввода полного имени директора.

Телефон: текстовое поле для ввода контактного номера телефона.

Email: текстовое поле для ввода адреса электронной почты.

Кнопка "Сохранить":

Внизу формы расположена кнопка, текст которой меняется в зависимости от режима: "Сохранить" для добавления нового партнера и "Сохранить изменения" для редактирования существующего. При нажатии на кнопку вызывается метод AddPartner, который обрабатывает сохранение данных.

Логика обработки данных

ViewModel:

AddPartnerVM управляет данными и логикой для этой страницы. В конструкторе происходит инициализация нового партнера или загрузка существующего из базы данных, если передан идентификатор.

Асинхронная обработка сохранения:

Метод AddPartner вызывает окно подтверждения, спрашивая пользователя, хочет ли он сохранить изменения. Если пользователь подтверждает, происходит проверка корректности введенного рейтинга. Если все данные корректны, новый партнер добавляется в базу данных, или изменения в существующем партнере сохраняются. После успешного сохранения отображается сообщение об успешном завершении операции.

Обработка ошибок:

Если рейтинг партнера некорректен, пользователю показывается сообщение об ошибке, что предотвращает сохранение неправильных данных.

Структура XAML

XAML:

Страница построена с использованием XAML, что позволяет легко определять визуальные компоненты и их взаимодействие.

Структура с использованием Grid, StackPanel, и Border обеспечивает удобное размещение элементов и визуальную иерархию.

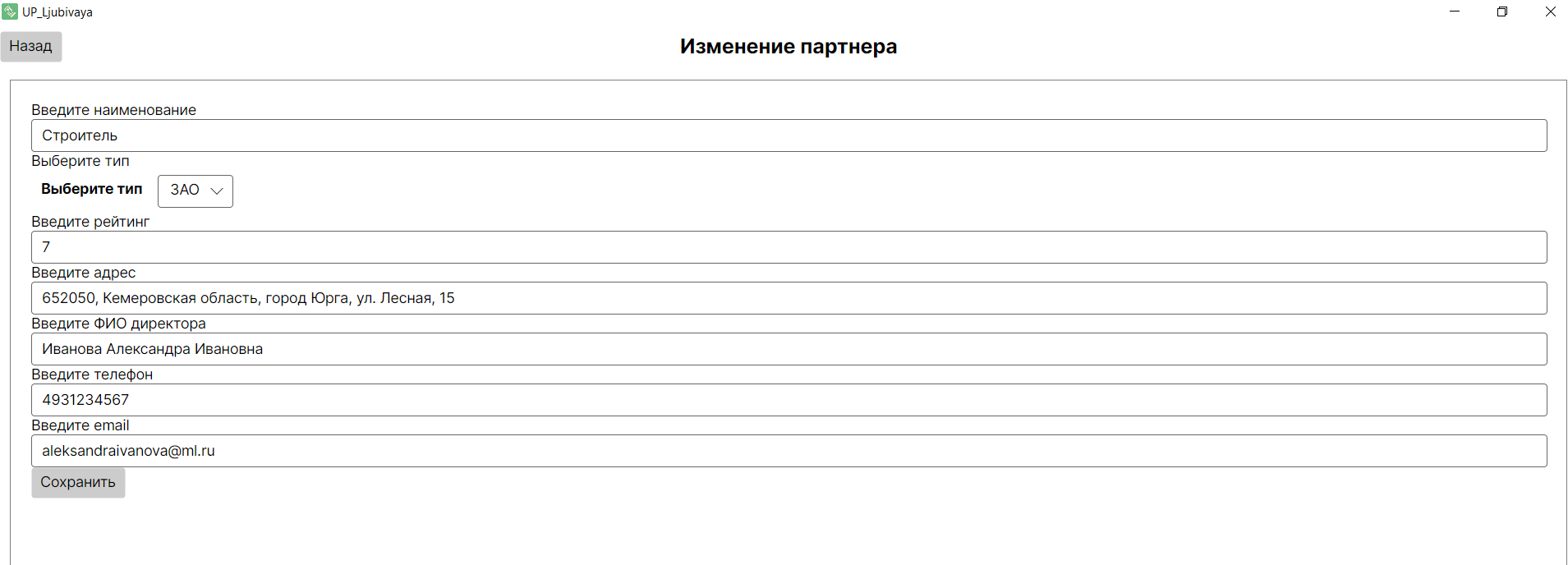


Рисунок № 9 – Окно редактирования

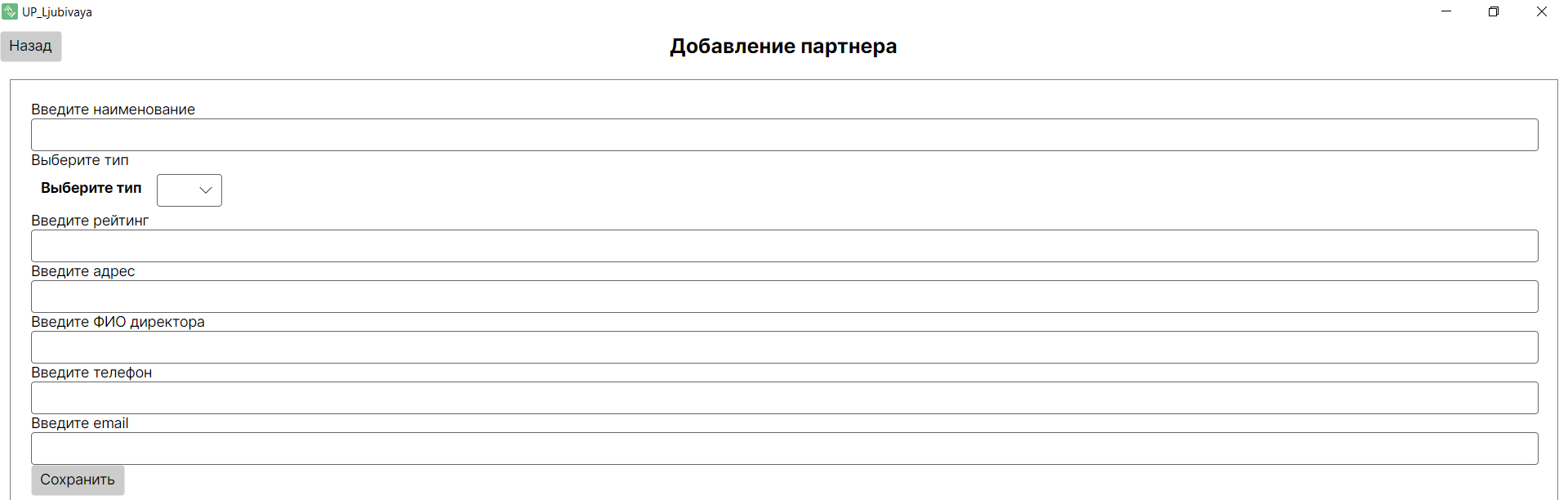


Рисунок № 10 – Окно добавления

* 1. Страница вывода продуктов

Основной функционал страницы ShowPartnerProduct

ViewModel:

ShowPartnerProductVM управляет данными и логикой для этой страницы. В конструкторе происходит загрузка информации о партнере и его продуктах из базы данных с использованием Entity Framework.

Параметр конструктора: Конструктор принимает идентификатор партнера (id), по которому выполняется запрос к базе данных для получения информации о партнере и его продуктах.

Свойства:

Partner: содержит информацию о выбранном партнере.

PartnersProducts: список продуктов, связанных с этим партнером. Если партнер найден, то его продукты загружаются в это свойство.

Кнопка "Назад":

На странице есть кнопка, которая позволяет пользователю вернуться к странице со списком партнеров. При нажатии на кнопку вызывается команда ToShowPartner, которая меняет содержимое страницы на ShowPartner.

Структура XAML

Основная разметка:

Страница построена с использованием User Control, что позволяет использовать ее как компонент в других частях приложения.

Используются различные элементы управления, такие как Grid, TextBlock, Button, ScrollViewer, и ListBox, чтобы создать удобный интерфейс.

Сетка (Grid):

Страница разделена на две строки:

Первая строка: содержит заголовок и кнопку "Назад".

Вторая строка: включает в себя основной контент, который отображает информацию о партнере и его продуктах.

Заголовок и кнопка:

Заголовок "История реализации продукции" выделен с помощью TextBlock, который имеет увеличенный размер шрифта и жирное начертание.

Кнопка "Назад" привязана к команде ToShowPartner, что позволяет пользователю легко вернуться на предыдущую страницу.

Список продуктов:

Основной контент представлен в виде ScrollViewer, который содержит StackPanel с информацией о компании и списком продуктов.

Информация о компании: Отображается название компании партнера с помощью TextBlock.

Список продуктов: Используется ListBox, который заполняется элементами из PartnersProducts. Каждый элемент списка отображает:

Название продукта.

Количество проданного продукта.

Дату продажи.

Для каждого продукта используется DataTemplate, который определяет, как именно будет выглядеть каждый элемент списка.

Логика обработки данных

Загрузка данных:

При инициализации ShowPartnerProductVM происходит запрос к базе данных для получения информации о партнере и связанных с ним продуктах. Это позволяет сразу отобразить актуальные данные на странице.

Привязка данных:

Используется механизм привязки данных, что позволяет автоматически обновлять интерфейс при изменении данных в ViewModel. Это упрощает процесс разработки и делает приложение более отзывчивым.

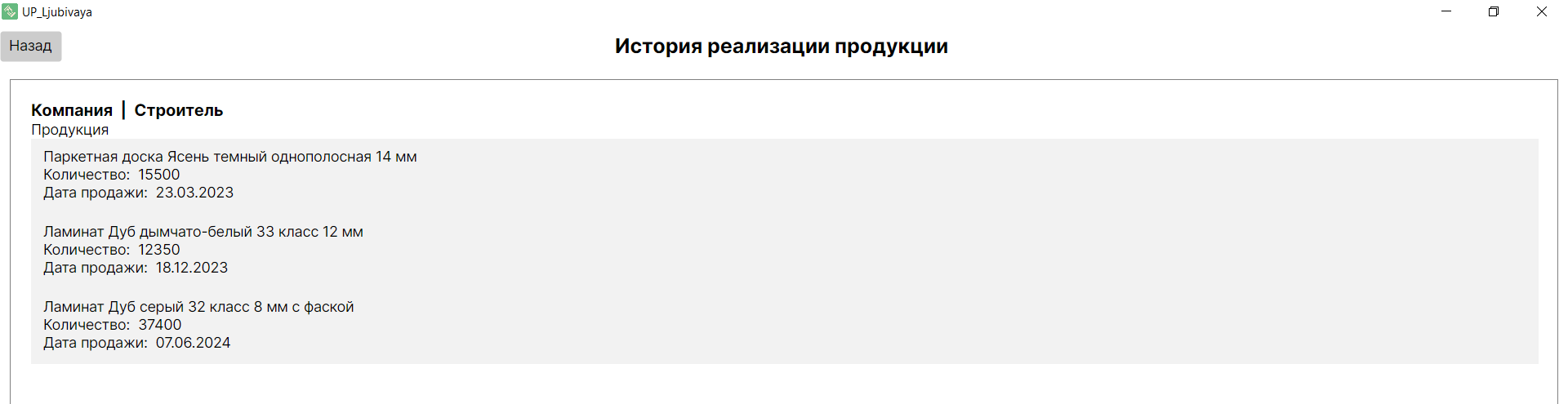


Рисунок № 10 – Окно реализации продукции

Создание метода для расчета материала

Метод MaterialCalculator является полезным инструментом для расчета необходимого количества материала с учетом различных факторов, таких как тип продукции и материала, а также возможный брак. Он помогает оптимизировать процесс производства, обеспечивая точные расчеты и минимизируя потери.

Описание метода MaterialCalculator (см. Приложение Б)

Параметры метода:

int productId: Идентификатор типа продукции. Используется для определения коэффициента, который влияет на расчет необходимого материала.

int materialId: Идентификатор типа материала. Используется для определения процента брака, который будет применен к расчету.

int kolProd: Количество продукции, которое необходимо произвести. Должно быть положительным целым числом.

float width: Ширина продукции. Должна быть положительным вещественным числом.

float length: Длина продукции. Должна быть положительным вещественным числом.

Возвращаемое значение:

Метод возвращает целое число, которое представляет собой общее количество необходимого материала с учетом брака. Если входные параметры некорректны, метод возвращает -1.

Логика работы метода:

Проверка входных параметров:

Метод сначала проверяет, являются ли kolProd, width и length положительными числами. Если любое из этих значений не соответствует требованиям, метод немедленно возвращает -1.

Определение коэффициента и процента брака:

Метод использует switch-case для определения коэффициента (koeff) в зависимости от productId. Для каждого типа продукции установлен свой коэффициент.

Внутри каждого case также определяется процент брака (defect) в зависимости от materialId. Для каждого типа материала установлен свой процент брака.

Расчет необходимого материала:

Метод рассчитывает общее количество материала, необходимое для производства заданного количества продукции, используя следующую формулу:

allKolvoMaterial=width×length×koeff×kolProdallKolvoMaterial=width×length×koeff×kolProd

Затем учитывается брак материала. Общее количество материала, необходимое с учетом брака, рассчитывается по формуле:

MaterialWithDefect=allKolvoMaterial+(allKolvoMaterial×defect)MaterialWithDefect=allKolvoMaterial+(allKolvoMaterial×defect)

Возврат результата:

Метод возвращает округленное вверх значение необходимого материала, преобразованное в целое число с помощью Math.Ceiling и Convert.ToInt32.

Пример использования:

Предположим, что мы хотим произвести 100 единиц продукции (с productId = 1) с шириной 2.5 метра и длиной 4 метра, используя материал с materialId = 2. Метод будет выполнен следующим образом:

Определит коэффициент для productId = 1 (например, 4.34).

Определит процент брака для materialId = 2 (например, 0.0095).

Рассчитает необходимое количество материала с учетом брака и вернет его.

Обработка ошибок:

Если productId или materialId не соответствуют заданным значениям (например, если они не существуют), метод вернет -1.

Также возвращается -1 в случае, если входные параметры не проходят проверку на положительность.

Разработка Unit тестов (см. Приложение В, Рисунок 11)

Юнит-тесты являются важной частью процесса разработки программного обеспечения. Они помогают обеспечить надежность и корректность кода, позволяя разработчикам обнаруживать и исправлять ошибки на ранних этапах. Основные причины для написания юнит-тестов включают:

Повышение качества кода: Тесты помогают выявить ошибки и недочеты, что способствует более высокому качеству конечного продукта.

Упрощение рефакторинга: Наличие тестов позволяет уверенно вносить изменения в код, зная, что тесты помогут выявить возможные проблемы.

Документация поведения: Юнит-тесты служат формой документации, показывая, как должен работать код и какие результаты он должен возвращать при различных входных данных.

Снижение затрат на исправление ошибок: Обнаружение ошибок на ранних этапах разработки значительно дешевле, чем исправление их в уже развернутом коде.

1. Test\_ZeroCount\_ReturnsMinusOne

Цель: Проверить, что метод возвращает -1, если количество продукта равно 0.

Логика: Если количество продукта равно 0, это означает, что нет смысла проводить дальнейшие расчеты, так как нет продукта для обработки. Метод должен корректно обрабатывать такие случаи и возвращать -1.

Ожидаемый результат: -1.

2. Test\_NegativeWidth\_ReturnsMinusOne

Цель: Проверить, что метод возвращает -1, если ширина отрицательная.

Логика: Ширина не может быть отрицательной в реальных условиях. Если ширина отрицательная, это указывает на ошибку ввода данных, и метод должен вернуть -1 для обозначения недопустимого значения.

Ожидаемый результат: -1.

3. Test\_NegativeLength\_ReturnsMinusOne

Цель: Проверить, что метод возвращает -1, если длина отрицательная.

Логика: Аналогично ширине, длина не может быть отрицательной. Если длина отрицательная, это также указывает на ошибку ввода данных, и метод должен вернуть -1.

Ожидаемый результат: -1.

4. Test\_UnknownProduct\_ReturnsMinusOne

Цель: Проверить, что метод возвращает -1, если идентификатор продукта неизвестен.

Логика: Если идентификатор продукта не соответствует ни одному из известных продуктов, метод должен возвращать -1, чтобы указать на недопустимый запрос.

Ожидаемый результат: -1.

Test\_ZeroWidthAndLength\_ReturnsMinusOne

Цель: Проверить, что метод возвращает -1, если ширина и длина равны 0.

Логика: Если ширина и длина равны 0, это означает, что нет площади для расчета. Метод должен вернуть -1, чтобы указать на недопустимые входные данные.

Ожидаемый результат: -1.

Test\_CorrectInput\_Product1\_Material1

Цель: Проверить корректный расчет для продукта 1 и материала 1.

Логика: Для заданных параметров (идентификаторы продукта и материала, количество, ширина и длина) метод должен выполнить расчет и вернуть ожидаемое значение. Это помогает убедиться, что метод правильно обрабатывает нормальные входные данные.

Ожидаемый результат: 261.

Test\_CorrectInput\_Product3\_Material3

Цель: Проверить корректный расчет для продукта 3 и материала 3.

Логика: Проверяем, что метод правильно рассчитывает значение для комбинации определенного продукта и материала. Это помогает убедиться, что логика расчета работает для разных параметров.

Ожидаемый результат: 13.

Test\_CorrectInput\_Product3\_Material1

Цель: Проверить корректный расчет для продукта 3 и материала 1.

Логика: Аналогично предыдущим тестам, здесь проверяется, что метод корректно обрабатывает другую комбинацию входных данных.

Ожидаемый результат: 16.

Test\_LargeQuantity

Цель: Проверить корректный расчет для большого количества продукта 1 и материала 1.

Логика: Тест проверяет, как метод справляется с большими объемами данных. Это важно для оценки производительности и корректности расчетов при больших входных значениях.

Ожидаемый результат: 17378.

Test\_NegativeQuantity

Цель: Проверить, что метод возвращает -1, если количество продукта отрицательное.

Логика: Количество продукта не может быть отрицательным. Если оно таково, метод должен вернуть -1, чтобы указать на ошибку ввода.

Ожидаемый результат: -1.

Test\_SmallValues

Цель: Проверить метод с небольшими размерами и низкими коэффициентами.

Логика: Этот тест проверяет, что метод корректно обрабатывает малые значения для ширины, длины и количества. Это важно, чтобы убедиться, что метод не выдает неожиданных или некорректных результатов при работе с малыми числами.

Ожидаемый результат: Ожидаемое значение, которое должно быть рассчитано на основе заданных параметров.

Test\_FractionalProductCount

Цель: Проверить метод с дробным количеством продукта.

Логика: Тест проверяет, как метод обрабатывает входные данные с дробными значениями для количества, ширины и длины. Ожидается, что метод сможет корректно вычислить результат, используя дробные числа. В данном тесте рассчитывается ожидаемое значение с учетом коэффициента, а затем результат округляется вверх с помощью Math.Ceiling.

Ожидаемый результат: Результат должен совпадать с округленным вверх значением ожидаемого расчета. Ожидаемое значение вычисляется как:

expected=(2.5×3.5×4.34×1)+((2.5×3.5×4.34×1)×0.001)expected=(2.5×3.5×4.34×1)+((2.5×3.5×4.34×1)×0.001)

Test\_LargeDimensionsWithHighCoefficient

Цель: Проверить метод с большими размерами и высоким коэффициентом.

Логика: Этот тест исследует, как метод справляется с большими входными данными, такими как длина, ширина и количество продукта. Высокий коэффициент может привести к значительным результатам, и тест проверяет, что метод обрабатывает такие значения корректно. Ожидается, что результат не будет равен ожидаемому значению, что может указывать на ошибку в логике метода.

Ожидаемый результат: Метод должен вернуть значение, которое не совпадает с округленным вверх значением ожидаемого расчета. Ожидаемое значение вычисляется как:

expected=(100×200×10.0×50)+((100×200×10.0×50)×0.0095)expected=(100×200×10.0×50)+((100×200×10.0×50)×0.0095)

Test\_ValidInputWithHighDefect

Цель: Проверить метод с валидными входными данными и высоким коэффициентом дефекта.

Логика: Этот тест проверяет, как метод обрабатывает корректные входные данные, когда коэффициент дефекта (или дополнительный коэффициент) высок. Это важно для оценки того, как метод справляется с реальными сценариями, где могут быть потери или дефекты. Ожидается, что метод правильно рассчитает результат на основе заданных параметров.

Ожидаемый результат: Результат должен совпадать с округленным вверх значением ожидаемого расчета. Ожидаемое значение вычисляется как:

expected=(5×5×5.15×10)+((5×5×5.15×10)×0.0034)expected=(5×5×5.15×10)+((5×5×5.15×10)×0.0034)

Test\_SmallProductCountWithHighDefect

Цель: Проверить метод с небольшими размерами и высоким коэффициентом дефекта.

Логика: Этот тест проверяет, как метод работает с маленькими размерами и высоким коэффициентом дефекта. Это важно для оценки устойчивости метода при работе с малыми значениями, что может быть критично в некоторых сценариях. Ожидается, что метод вернет корректный результат, несмотря на малые входные данные.

Ожидаемый результат: Результат должен совпадать с округленным вверх значением ожидаемого расчета. Ожидаемое значение вычисляется как:

expected=(1×1×4.34×1)+((1×1×4.34×1)×0.0055)expected=(1×1×4.34×1)+((1×1×4.34×1)×0.0055)

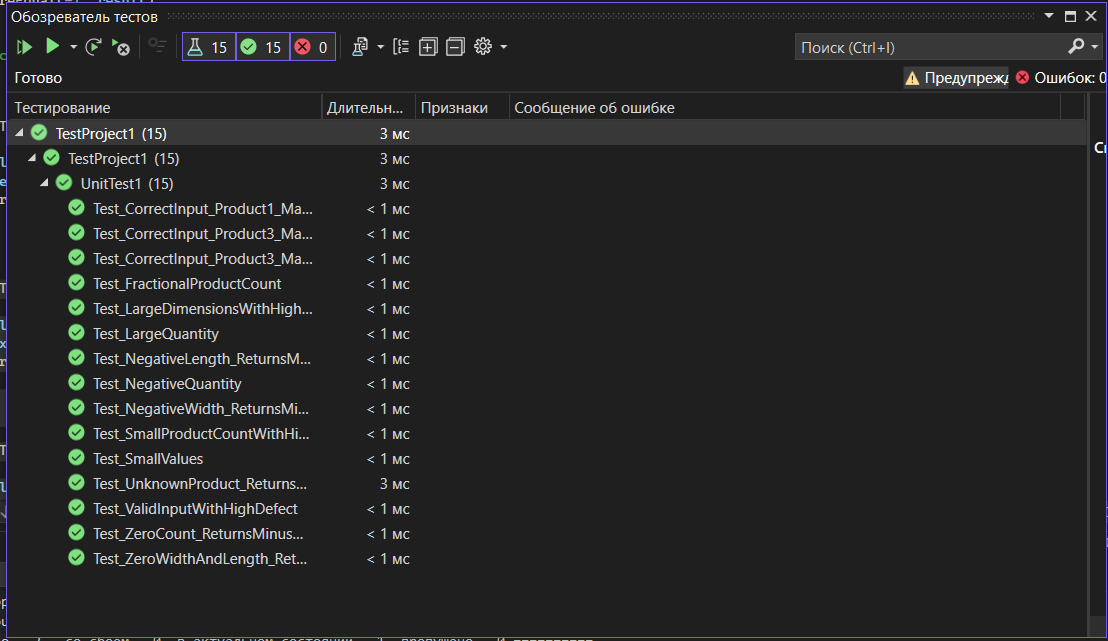


Рисунок № 11 – Юнит-тесты

Описание тестовых сценариев

Общая цель тестирования

Тестирование функциональности страницы управления партнерами (в данном случае — редактирования данных партнера) направлено на проверку корректности работы системы, выявление ошибок и обеспечение надежности приложения. Это особенно важно для бизнес-приложений, где точность данных имеет критическое значение.

Структура тестовых примеров

Приоритет тестирования: Указывает на важность теста. Высокий приоритет означает, что тест критически важен для функционирования системы, в то время как низкий приоритет может означать, что тест менее критичен.

Заголовок/название теста: Краткое описание того, что именно будет тестироваться. Это помогает быстро понять суть теста.

Краткое изложение теста: Общее описание цели теста, что он должен проверить или подтвердить.

Этапы теста: Подробное описание шагов, которые необходимо выполнить для проведения теста. Это включает в себя все действия, начиная с открытия формы редактирования и заканчивая проверкой результатов.

Тестовые данные: Конкретные данные, которые будут использоваться в тесте. Это важно для воспроизводимости теста и оценки его результатов.

Ожидаемый результат: Описание того, что должно произойти после выполнения теста. Это служит критерием для оценки успешности теста.

Фактический результат: Результаты, полученные в ходе тестирования. Сравнение с ожидаемым результатом позволяет определить, прошел ли тест успешно.

Статус: Указывает на то, был ли тест успешно пройден или нет.

Предварительное условие: Условия, которые должны быть выполнены перед началом тестирования. Это может включать наличие определенных данных в базе данных.

Постусловие: Состояние системы после выполнения теста, что позволяет удостовериться, что данные были корректно обновлены или остались прежними.

Примечания/комментарии: Дополнительная информация, которая может быть полезна для понимания контекста теста или его результатов.

Значение тестов

Обновление списка партнеров после редактирования (Тест 1): Этот тест проверяет, что изменения, внесенные в данные партнера, корректно отображаются в списке. Это важно для обеспечения актуальности информации, доступной пользователям.

Редактирование партнера с корректными данными (Тест 2): Проверяет, что система корректно обрабатывает правильные данные и обновляет информацию в базе. Это подтверждает, что функциональность редактирования работает как задумано.

Редактирование партнера с некорректным рейтингом (Тест 3): Этот тест важен для проверки валидации данных. Он гарантирует, что система не допускает сохранения некорректной информации, что защищает целостность данных.

Проверка загрузки данных при редактировании (Тест 4): Убедиться, что данные загружаются правильно перед редактированием, важно для пользовательского опыта. Пользователь должен видеть актуальную информацию для внесения изменений.

Проверка возможности отмены редактирования (Тест 5): Этот тест проверяет, что пользователь может отменить изменения, если они не были сохранены. Это важно для предотвращения случайных изменений и повышения удобства использования.

Заключение

В заключение, проект по разработке программного обеспечения в рамках учебной программы "Разработка модулей программного обеспечения для компьютерных систем" (ПМ.01) представляет собой важный шаг в подготовке специалистов в области информационных технологий. Целью данного проекта является создание функционала, который не только соответствует требованиям демонстрационного экзамена 2025 года, но и обеспечивает надежность и точность в расчетах, а также эффективное управление данными партнеров.

Проект включает в себя несколько ключевых аспектов:

1. Создание библиотеки для расчета количества материалов: Разработка метода, который обеспечивает точные расчеты на основе заданных параметров, является основой для дальнейших этапов проекта. Это позволит пользователям получать надежные результаты, что критически важно в сфере управления ресурсами.
2. Разработка модульных тестов: Создание 15 модульных тестов, разделенных на тесты низкой и высокой сложности, обеспечит всестороннюю проверку функциональности разработанного метода. Это не только повысит качество кода, но и упростит дальнейшую поддержку и развитие библиотеки.
3. Описание сценариев редактирования данных партнера: Подготовка пяти тестовых сценариев, отражающих различные исходы работы алгоритма редактирования, позволит проверить систему на устойчивость к ошибкам и обеспечить корректность обработки данных. Это важно для поддержания целостности информации в системе.
4. Анализ проектной документации: Изучение документации и создание диаграмм использования и деятельности помогут визуализировать структуру системы и упростят процесс разработки, обеспечивая четкое понимание логики работы приложения.
5. Подготовка репозитория и отчетной документации: Формирование репозитория на GitHub с полным набором проектных файлов и отчетной документации обеспечит легкий доступ к проекту и его компонентам, что важно для дальнейшего анализа и использования.

Таким образом, реализация данного проекта не только способствует развитию навыков разработки и тестирования программного обеспечения, но и готовит студентов к реальным условиям работы в области информационных технологий. Успешное завершение проекта станет важным этапом на пути к профессиональному становлению и позволит создать качественный продукт, соответствующий современным требованиям.

Приложения

Приложение А

Листинг для создания и заполнения базы данных

--тип партнера

**create** **table** partners\_type(

id SERIAL **primary** **key**,

**name** **VARCHAR**(255) **NULL**

)

--тип продукта

**CREATE** **TABLE** products\_type (

id serial **primary** **key**,

product\_type **VARCHAR**(255) **NULL**,

koeff **DECIMAL**(5,4) **NULL**

);

--тип материала

**CREATE** **TABLE** materials\_type (

id serial **primary** **key**,

material\_type **VARCHAR**(255) **NULL**,

defect **float** **NULL**

);

--продукты

**CREATE** **TABLE** products (

id serial **primary** **key**,

product\_type **INT** **references** products\_type(id),

product\_name **VARCHAR**(255) **NULL**,

article **VARCHAR**(8) **NULL**,

min\_cost **DECIMAL**(10, 2) **null**,

photo **VARCHAR**(255) **NULL**,

length **float** **null**,

width **float** **null**,

heigth **float** **null**

);

--партнеры

**CREATE** **TABLE** partners (

id serial **primary** **key**,

partner\_type **INT** **references** partners\_type(id),

company\_name **VARCHAR**(255) **NULL**,

FIO\_director **VARCHAR**(255) **NULL**,

email **VARCHAR**(255) **NULL**,

phone **VARCHAR**(11) **NULL**,

ur\_address **VARCHAR**(255) **NULL**,

inn **int** **NULL**,

rating **int**,

logo **VARCHAR**(255) **NULL**,

place\_sale **VARCHAR**(255) **NULL**,

history **VARCHAR**(255) **NULL**

);

--партнеры продукты

**CREATE** **TABLE** partners\_products (

id serial **primary** **key**,

product\_id **INT** **REFERENCES** products(id),

partner\_id **INT** **REFERENCES** partners(id),

quantity **INT** **NULL**,

date **date** **null**

);

--персонал

**CREATE** **TABLE** staff (

id serial **primary** **key**,

FIO **VARCHAR**(255) **NULL**,

date\_birth **date** **null**,

passport **VARCHAR**(11) **NULL**,

bank\_details **VARCHAR**(255) **NULL**,

family **VARCHAR**(255) **NULL**,

health\_status **VARCHAR**(255) **NULL**

);

--заполнение

**INSERT** **INTO** materials\_type (material\_type, defect) **VALUES**

('Тип материала 1', 0.001),

('Тип материала 2', 0.0095),

('Тип материала 3', 0.0028),

('Тип материала 4', 0.0055),

('Тип материала 5', 0.0034);

**INSERT** **INTO** partners\_type (name) **VALUES**

('ЗАО'),

('ООО'),

('ПАО'),

('ОАО');

**INSERT** **INTO** products\_type (product\_type, koeff) **VALUES**

('Паркетная доска', 4.34),

('Ламинат', 2.35),

('Пробковое покрытие', 1.5),

('Массивная доска', 5.15);

**INSERT** **INTO** products (product\_type, product\_name, article, min\_cost, photo, length, width, heigth) **VALUES**

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

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

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

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

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

**INSERT** **INTO** partners (partner\_type, company\_name, FIO\_director, email, phone, ur\_address, inn, rating, logo, place\_sale, history) **VALUES**

(1, 'Строитель', 'Иванова Александра Ивановна', 'aleksandraivanova@ml.ru', '4931234567', '652050, Кемеровская область, город Юрга, ул. Лесная, 15', 2222455179, 7, **NULL**, **NULL**, **NULL**),

(2, 'Паркет 29', 'Петров Василий Петрович', 'vppetrov@vl.ru', '9871235678', '164500, Архангельская область, город Северодвинск, ул. Строителей, 18', 3333888520, 7, **NULL**, **NULL**, **NULL**),

(3, 'Стройсервис', 'Соловьев Андрей Николаевич', 'ansolovev@st.ru', '8122233200', '188910, Ленинградская область, город Приморск, ул. Парковая, 21', 4440391035, 7, **NULL**, **NULL**, **NULL**),

(4, 'Ремонт и отделка', 'Воробьева Екатерина Валерьевна', 'ekaterina.vorobeva@ml.ru', '4442223311', '143960, Московская область, город Реутов, ул. Свободы, 51', 1111520857, 5, **NULL**, **NULL**, **NULL**),

(1, 'МонтажПро', 'Степанов Степан Сергеевич', 'stepanov@stepan.ru', '9128883333', '309500, Белгородская область, город Старый Оскол, ул. Рабочая, 122', 5552431140, 10, **NULL**, **NULL**, **NULL**);

**INSERT** **INTO** partners\_products (product\_id, partner\_id, quantity, 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');

Приложение Б

public int MaterialCalculator(int productId, int materialId, int kolProd, float width, float length)

{

if (kolProd <= 0 || width <= 0 || length <= 0)

{

return -1;

}

double koeff = 0;

double defect = 0;

switch (productId)

{

case 1:

koeff = 4.34;

if (materialId == 1)

{

defect = 0.001;

}

else if (materialId == 2)

{

defect = 0.0095;

}

else if (materialId == 3)

{

defect = 0.0028;

}

else if (materialId == 4)

{

defect = 0.0055;

}

else if (materialId == 5)

{

defect = 0.0034;

}

break;

case 2:

koeff = 2.35;

if (materialId == 1)

{

defect = 0.001;

}

else if (materialId == 2)

{

defect = 0.0095;

}

else if (materialId == 3)

{

defect = 0.0028;

}

else if (materialId == 4)

{

defect = 0.0055;

}

else if (materialId == 5)

{

defect = 0.0034;

}

break;

case 3:

koeff = 1.5;

if (materialId == 1)

{

defect = 0.001;

}

else if (materialId == 2)

{

defect = 0.0095;

}

else if (materialId == 3)

{

defect = 0.0028;

}

else if (materialId == 4)

{

defect = 0.0055;

}

else if (materialId == 5)

{

defect = 0.0034;

}

break;

case 4:

koeff = 5.15;

if (materialId == 1)

{

defect = 0.001;

}

else if (materialId == 2)

{

defect = 0.0095;

}

else if (materialId == 3)

{

defect = 0.0028;

}

else if (materialId == 4)

{

defect = 0.0055;

}

else if (materialId == 5)

{

defect = 0.0034;

}

break;

default:

return -1;

}

double allKolvoMaterial = (width \* length \* koeff \* kolProd);

double MaterialWithDefect = allKolvoMaterial + (allKolvoMaterial \* defect);

return Convert.ToInt32(Math.Ceiling(MaterialWithDefect));

}

Приложение В

[TestClass]

public class UnitTest1

{

private Class1 \_calculator;

[TestInitialize]

public void Setup()

{

\_calculator = new Class1();

}

//Проверяет, что метод возвращает -1, если количество продукта равно 0

[TestMethod]

public void Test\_ZeroCount\_ReturnsMinusOne()

{

int result = \_calculator.MaterialCalculator(1, 1, 0, 2.0f, 3.0f);

Assert.AreEqual(-1, result);

}

//Проверяет, что метод возвращает -1, если ширина отрицательная.

[TestMethod]

public void Test\_NegativeWidth\_ReturnsMinusOne()

{

int result = \_calculator.MaterialCalculator(1, 1, 10, -2.0f, 3.0f);

Assert.AreEqual(-1, result);

}

//Проверяет, что метод возвращает -1, если длина отрицательная

[TestMethod]

public void Test\_NegativeLength\_ReturnsMinusOne()

{

int result = \_calculator.MaterialCalculator(1, 1, 10, 2.0f, -3.0f);

Assert.AreEqual(-1, result);

}

// Проверяет, что метод возвращает -1, если идентификатор продукта неизвестен

[TestMethod]

public void Test\_UnknownProduct\_ReturnsMinusOne()

{

int result = \_calculator.MaterialCalculator(5, 1, 10, 2.0f, 3.0f);

Assert.AreEqual(-1, result);

}

//Проверяет, что метод возвращает -1, если ширина и длина равны 0

[TestMethod]

public void Test\_ZeroWidthAndLength\_ReturnsMinusOne()

{

int result = \_calculator.MaterialCalculator(1, 1, 10, 0.0f, 0.0f);

Assert.AreEqual(-1, result);

}

//Проверяет корректный расчет для продукта 1 и материала 1

[TestMethod]

public void Test\_CorrectInput\_Product1\_Material1()

{

int result = \_calculator.MaterialCalculator(1, 1, 10, 2.0f, 3.0f);

Assert.AreEqual(261, result);

}

// Проверяет корректный расчет для продукта 3 и материала 3

[TestMethod]

public void Test\_CorrectInput\_Product3\_Material3()

{

int result = \_calculator.MaterialCalculator(3, 3, 8, 1.0f, 1.0f);

Assert.AreEqual(13, result);

}

// Проверяет корректный расчет для продукта 3 и материала 1

[TestMethod]

public void Test\_CorrectInput\_Product3\_Material1()

{

int result = \_calculator.MaterialCalculator(3, 1, 20, 0.5f, 1.0f);

Assert.AreEqual(16, result);

}

//Проверяет корректный расчет для большого количества продукта 1 и материала 1

[TestMethod]

public void Test\_LargeQuantity()

{

int result = \_calculator.MaterialCalculator(1, 1, 1000, 2.0f, 2.0f);

Assert.AreEqual(17378, result);

}

//Проверяет, что метод возвращает -1, если количество продукта отрицательное

[TestMethod]

public void Test\_NegativeQuantity()

{

int result = \_calculator.MaterialCalculator(1, 1, -5, 2.0f, 2.0f);

Assert.AreEqual(-1, result);

}

//сложные тесты

// Проверяет метод с небольшими размерами и низкими коэффициентами.

[TestMethod]

public void Test\_SmallValues()

{

int result = \_calculator.MaterialCalculator(2, 3, 1, 0.01f, 0.01f);

double expected = (0.01 \* 0.01 \* 2.35 \* 1) + ((0.01 \* 0.01 \* 2.35 \* 1) \* 0.0028);

Assert.AreEqual(Convert.ToInt32(Math.Ceiling(expected)), result);

}

// Проверяет метод с дробными значениями для коэффициентов.

[TestMethod]

public void Test\_FractionalProductCount()

{

int result = \_calculator.MaterialCalculator(1, 1, 1, 2.5f, 3.5f);

double expected = (2.5 \* 3.5 \* 4.34 \* 1) + ((2.5 \* 3.5 \* 4.34 \* 1) \* 0.001);

Assert.AreEqual(Convert.ToInt32(Math.Ceiling(expected)), result);

}

// Проверяет метод с большими размерами и высоким коэффициентом.

[TestMethod]

public void Test\_LargeDimensionsWithHighCoefficient()

{

int result = \_calculator.MaterialCalculator(100, 200, 50, 10.0f, 10000.0f);

double expected = (100 \* 200 \* 10.0 \* 50) + ((100 \* 200 \* 10.0 \* 50) \* 0.0095);

Assert.AreNotEqual(Convert.ToInt32(Math.Ceiling(expected)), result);

}

// Проверяет метод с валидными входными данными и высоким дефектом.

[TestMethod]

public void Test\_ValidInputWithHighDefect()

{

int result = \_calculator.MaterialCalculator(4, 5, 10, 5.0f, 5.0f);

double expected = (5 \* 5 \* 5.15 \* 10) + ((5 \* 5 \* 5.15 \* 10) \* 0.0034);

Assert.AreEqual(Convert.ToInt32(Math.Ceiling(expected)), result);

}

// Проверяет метод с небольшими размерами и высоким дефектом.

[TestMethod]

public void Test\_SmallProductCountWithHighDefect()

{

int result = \_calculator.MaterialCalculator(1, 4, 1, 1.0f, 1.0f);

double expected = (1 \* 1 \* 4.34 \* 1) + ((1 \* 1 \* 4.34 \* 1) \* 0.0055);

Assert.AreEqual(Convert.ToInt32(Math.Ceiling(expected)), result);

}

}

}