**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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колиниченко Е.С. 21.03.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

**Содержание:**

[Введение 3](#_Toc192202351)

[Основная часть 4](#_Toc192202352)

[Модуль 1. Разработка, администрирование и защита баз данных. 4](#_Toc192202353)

[Модуль 2. Разработка модулей программного обеспечения для компьютерных систем. 6](#_Toc192202354)

[Модуль 3. Сопровождение и обслуживание программного обеспечения компьютерных систем. 9](#_Toc192202355)

[Модуль 4. Осуществление интеграции программных модулей. 16](#_Toc192202356)

[Модуль 5. Разработка диаграммы использования и диаграммы деятельности. 18](#_Toc192202357)

[Заключение 21](#_Toc192202358)

[Приложение А 23](#_Toc192202359)

[Приложение Б 29](#_Toc192202360)

# **Введение**

Целью учебной практики является выполнение четырех сессий демонстрационного экзамена 2025 года, описанных в документе КОД 09.02.07-2-2025 Том 1. Эти сессии соответствуют следующим модулям:

1. Разработка, администрирование и защита баз данных;
2. Разработка модулей программного обеспечения для компьютерных систем;
3. Сопровождение и обслуживание программного обеспечения компьютерных систем;
4. Осуществление интеграции программных модулей.

Каждый модуль включал выполнение конкретных задач, таких как анализ требований, разработка программных решений, создание модульных тестов, описание тестовых сценариев и проектирование диаграмм.

В рамках третьей сессии были разработаны тестовые сценарии для процедуры редактирования партнера, демонстрирующие различные исходы работы алгоритма. Особое внимание было уделено четвертой сессии, где требовалось реализовать метод для расчета количества материалов в виде библиотеки и разработать проект модульных тестов.

Помимо выполнения этих сессий, также были разработаны диаграммы использования и диаграммы деятельности для информационной системы фитнес-центра.

# **Основная часть**

## **Модуль 1. Разработка, администрирование и защита баз данных.**

В рамках этого модуля были выполнены следующие задачи:

1. Создание ER-диаграммы: ER-диаграмма была сгенерирована с помощью системы управления базами данных (СУБД) и представлена в формате PDF. Она включает таблицы, связи между ними, атрибуты и ключи. На данном этапе типы данных не были приоритетными.

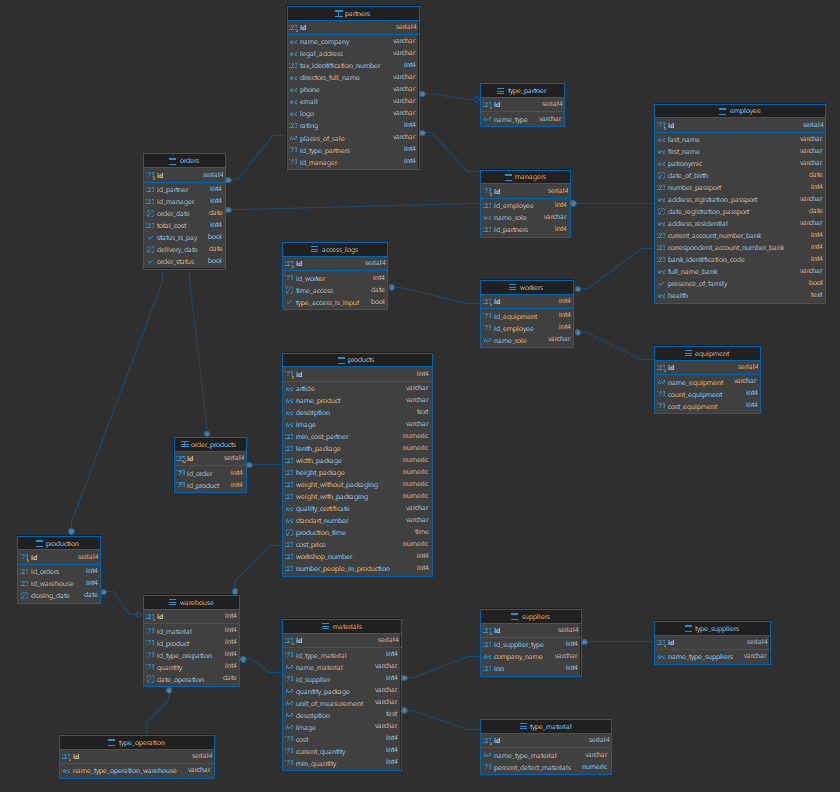


Рисунок № 1 - Разработанная ER-диаграмма для системы.

1. Разработка базы данных: на основе описания предметной области была создана база данных в СУБД Postgres для разрабатываемой системы. База данных была нормализована до третьей нормальной формы, что обеспечило ссылочную целостность данных. Скрипт созданной базы данных смотреть в [Приложении А](#_Приложение_А).
2. Импорт данных: Заказчик предоставил файлы с данными, помеченные как "import", для переноса в новую систему. Эти данные были подготовлены для импорта и успешно загружены в разработанную базу данных.

## **Модуль 2. Разработка модулей программного обеспечения для компьютерных систем.**

В рамках данного задания была выполнена разработка алгоритма подсистемы для работы с партнерами. Алгоритмы были реализованы в виде кода программного продукта с использованием среды разработки Visual Studio и языка программирования C#, с фреймворком Avalonia. Эта подсистема обеспечивает ряд важных функций:

1. Просмотр списка партнеров: на главной форме приложения отображается список партнеров, информация о которых хранится в базе данных. Отображение соответствует предоставленному макету, а цвета элементов определены в соответствии с руководством по стилю. Также, был разработан свойство, которое подсчитывала скидку для каждого партнера. Скидка отображается у каждого партнера в зависимости от общего количество реализованной продукции партнером.

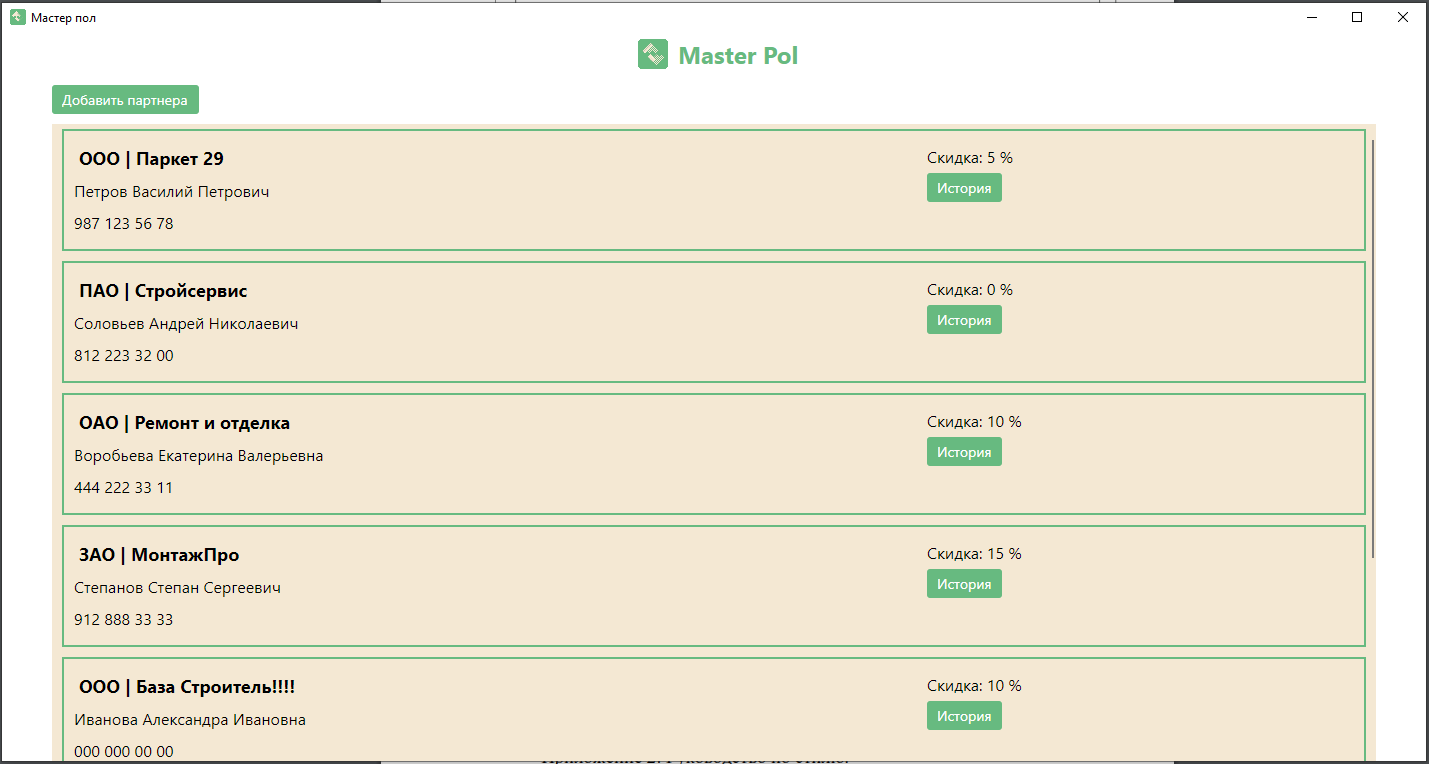


Рисунок № 2 - Список партнеров.

1. Добавление и редактирование данных о партнере: Реализованы удобные формы для добавления нового партнера и редактирования существующих данных. При нажатии на кнопку «Добавить партнера» открывается форма добавления, где создается новый объект модели. Сама форма представлена на рисунке №3. При нажатии на партнера открывается форма редактирования, где в конструктор класса передается выбранный объект. Форма редактирования представлена на рисунке №4. При добавлении или редактировании данных проверяется корректность введенных данных, включая обязательные поля и формат контактной информации.

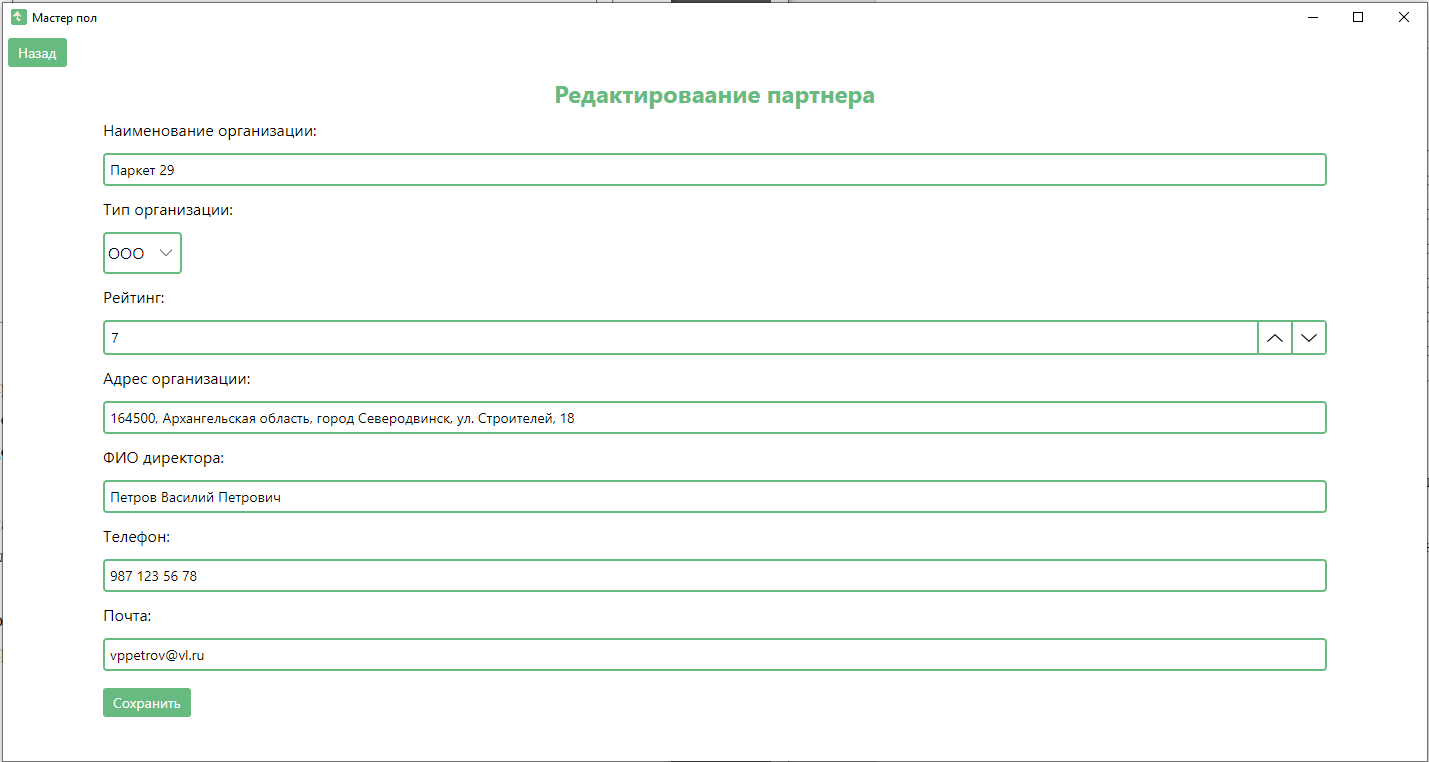


Рисунок № 3 - Форма редактирования партнера.

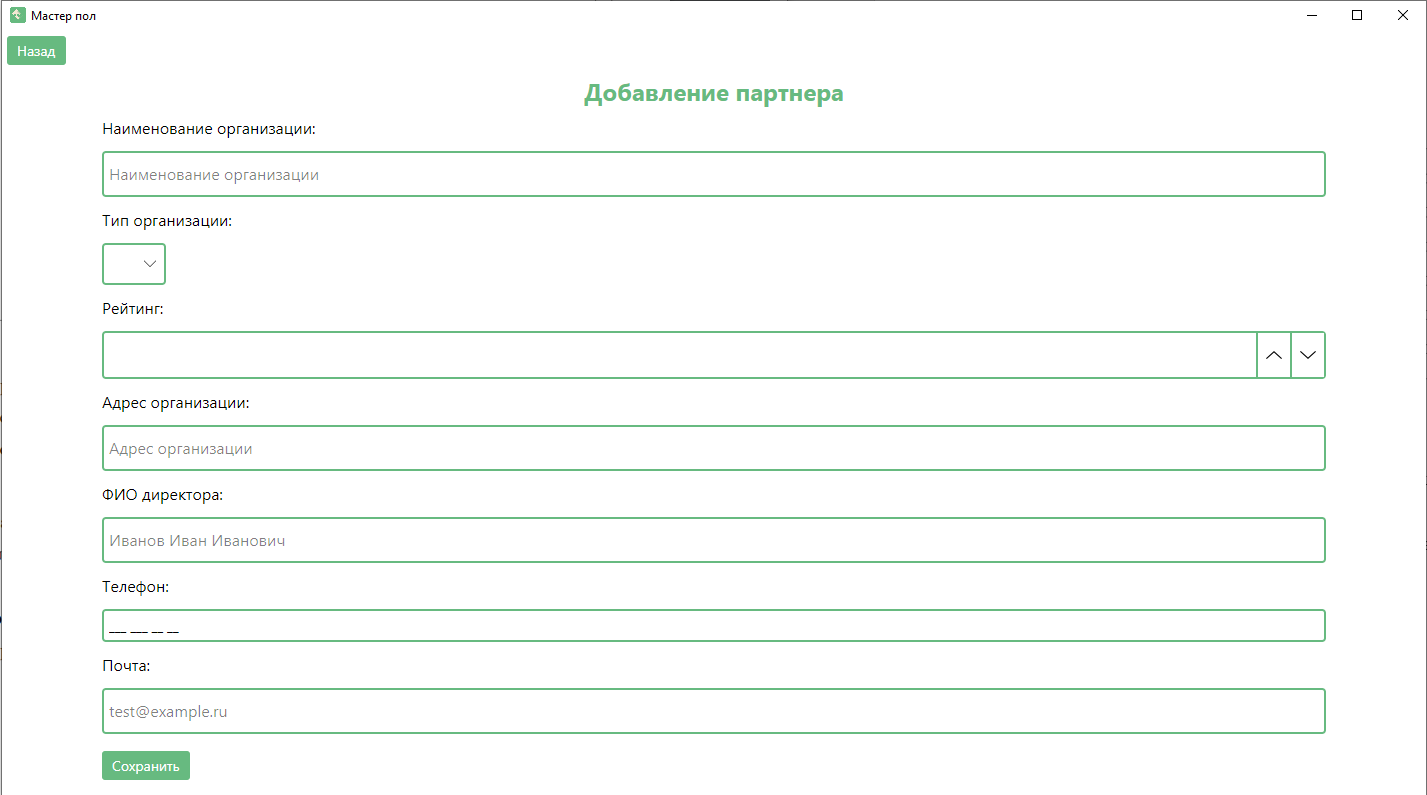


Рисунок № 4 - Форма добавления нового партнера.

1. Просмотр истории реализации продукции партнером: для каждого партнера доступна история реализации продукции, которую можно просмотреть в отдельной форме, нажав на кнопку «История реализации продукции». Форма представлена на рисунке №5.



Рисунок № 5 - История реализации продукции партнером.

Тестирование модуля: в ходе разработки был проведен комплексный тест модуля для проверки его функциональности. Приложение работает корректно и не завершается аварийно, что подтверждает его стабильность и надежность.

## **Модуль 3. Сопровождение и обслуживание программного обеспечения компьютерных систем.**

Разработан интуитивно понятный пользовательский интерфейс, обеспечивающий удобную навигацию между окнами (страницами) приложения. Переход на форму добавления или редактирования партнера осуществляется непосредственно с главной формы списка партнеров. Для возврата на предыдущую страницу предусмотрена кнопка «Назад».

Для обеспечения ясности и удобства использования, каждое окно имеет заголовок, четко отражающий его назначение, например: «Список партнеров», «Добавление партнера» или «Редактирование партнера».

В целях обеспечения стабильной работы приложения и предоставления пользователю информативной обратной связи реализована обработка исключительных ситуаций с использованием конструкции try-catch. Код обработки исключений представлен в приложении Б, Таблица 11. При возникновении ошибок или совершении запрещенных действий пользователь получает уведомление.

В приложении реализованы всплывающие окна с соответствующими заголовками, что позволяет пользователю быстро идентифицировать тип сообщения. Текст сообщений содержит подробную информацию об ошибках и рекомендации по их устранению. Всплывающие окна представлены на рисунках 6-8.

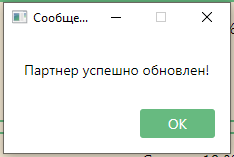


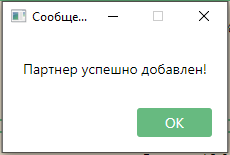
Рисунок № 6 - Сообщение об успешном добавлении партнера.

Рисунок № 7 - Сообщение об успешном добавлении партнера.

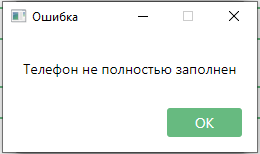


Рисунок № 8 - Пример неправильно заполненного поля с номером телефона.

Также, в рамках данной сессии были разработаны и описаны тестовые сценарии для процедуры добавления / редактирования формы добавления/редактирования партнера, демонстрирующие различные исходы работы алгоритма добавления/ редактирования партнёра.

Таблица № 1 - Аннотация теста.

|  |  |
| --- | --- |
| **Название проекта** | Мастер пол |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Екатерина Колиниченко |
| **Дата(ы) теста** | 05.03.2025 |

Таблица № 2 - Тестовый пример #1.

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Редактирование партнера |
| **Краткое изложение теста** | Редактирование поля «тип организации» ООО в форме редактирования для партнера «ООО | Тестовая организация» на ЗАО |
| **Этапы теста** | 1. Нажать на партнера из списка 2. В открывшемся окне нажать на выпадающий список «Тип организации» 3. Выбрать необходимый тип организации «ЗАО» 4. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Партнер «ООО|Тестовая организация» |
| **Ожидаемый результат** | Партнер «ЗАО|Тестовая организация» |
| **Фактический результат** | Партнер «ЗАО|Тестовая организация» |
| **Статус** | Зачет |
| **Предварительное условие** | Программа запущена, список партнеров отображается |
| **Постусловие** | Список партнеров отображается |
| **Примечания/комментарии** | - |

Таблица № 3 - Тестовый пример #2

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Редактирование поля «Почта» |
| **Краткое изложение теста** | Редактирование поля «Почта» в форме редактирования для партнера «ООО | Тестовая организация» с почтой «test@example.com» на не корректную «testexample.com» |
| **Этапы теста** | 1. Нажать на партнера из списка 2. В открывшемся окне нажать на поле редактирования почты 3. Изменить «test@example.com» на не корректную «testexample.com» 4. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Партнер «ООО | Тестовая организация» с почтой «test@example.com» |
| **Ожидаемый результат** | Выведется ошибка о неправильно указанном email |
| **Фактический результат** | Выводится ошибка о неправильно указанном email |
| **Статус** | Зачет |
| **Предварительное условие** | Программа запущена, список партнеров отображается |
| **Постусловие** | Выводится ошибка, и со страницы «редактирование партнера» никуда переход не осуществляется |
| **Примечания/комментарии** | - |

Таблица № 4- Тестовый пример #3

|  |  |
| --- | --- |
| Тестовый пример **#** | ТС\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Редактирование поля «ФИО директора» |
| **Краткое изложение теста** | Редактирование поля «ФИО директора»» в форме редактирования для партнера «ООО|Тестовая организация» с ФИО директора: «Иванов Иван Иванович» на не корректную «Иванов Иван иванович» |
| **Этапы теста** | 1. Нажать на партнера из списка 2. В открывшемся окне нажать на поле редактирования ФИО 3. Изменить «Иванов Иван Иванович» на не корректное значение «Иванов Иван иванович» 4. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Партнер «ООО|Тестовая организация» с ФИО директора: Иванов Иван Иванович |
| **Ожидаемый результат** | Выведется ошибка о том, что ФИО директора не верно заполнено, и подскажет, как исправить данную ошибку. |
| **Фактический результат** | Выводится ошибка о том, что ФИО директора не верно заполнено, и подсказывает, что ФИО должно быть написано русскими буквами и что имя, фамилия и отчество начинаются с заглавных букв. |
| **Статус** | Зачет |
| **Предварительное условие** | Запущенное приложение, с открытым списком партнеров. |
| **Постусловие** | Выводится ошибка, и со страницы «редактирование партнера» никуда переход не осуществляется |
| **Примечания/комментарии** | - |

Таблица № 5- Тестовый пример #4

|  |  |
| --- | --- |
| Тестовый пример **#** | ТС\_4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Редактирование поля «Телефон» |
| **Краткое изложение теста** | Редактирование поля «Телефон» в форме редактирования для партнера «ООО|Тестовая организация» с телефоном: «777 777 77 77» на телефон «777 777 77» |
| **Этапы теста** | 1. Нажать на партнера из списка 2. В открывшемся окне нажать на поле редактирования телефона 3. Изменить «777 777 77 77» на не корректное значение телефона « 777 777 77» 4. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Партнер «ООО|Тестовая организация» с телефоном: «777 777 77 77 |
| **Ожидаемый результат** | Выведется ошибка о том, что телефон не полностью заполнен |
| **Фактический результат** | Выводится ошибка о том, что телефон не полностью заполнен |
| **Статус** | Зачет |
| **Предварительное условие** | Запущенное приложение, с открытым списком партнеров. |
| **Постусловие** | Выводится ошибка, и со страницы «редактирование партнера» никуда переход не осуществляется |
| **Примечания/комментарии** | - |

Таблица № 6 - Тестовый пример #5

|  |  |
| --- | --- |
| **Test Case #** | ТС\_5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Редактирование поля «Рейтинг» |
| **Краткое изложение теста** | Редактирование поля «Рейтинг» в форме редактирования для партнера «ООО|Тестовая организация» рейтингом 3 на не корректный рейтинг «ффф» |
| **Этапы теста** | 1. Нажать на партнера из списка 2. В открывшемся окне нажать на поле редактирования рейтинга 3. Изменить «3» на не корректное значение «ффф» 4. Нажать на любое другое поле, что бы поле рейтинга очистилось. 5. Нажать кнопку «Сохранить» |
| **Тестовые данные** | Партнёр «ООО|Тестовая организация» с рейтингом «3» |
| **Ожидаемый результат** | Выведется ошибка о том, что рейтинг не может быть пустым |
| **Фактический результат** | Выводится ошибка о том, что рейтинг не может быть пустым |
| **Статус** | Зачет |
| **Предварительное условие** | Запущенное приложение, с открытым списком партнеров. |
| **Постусловие** | Выводится ошибка, и со страницы «редактирование партнера» никуда переход не осуществляется |
| **Примечания/комментарии** | - |

## **Модуль 4. Осуществление интеграции программных модулей.**

Разработка метода расчета количества материалов и модульных тестов

Был разработан метод MathCountMaterial, предназначенный для расчета количества необходимых материалов с учетом брака. Код метода представлен в [приложении Б](#_Приложение_Б), Таблица 8. Метод принимает следующие параметры:

* Идентификатор типа продукции (целое число)
* Идентификатор типа материала (целое число)
* Количество продукции (целое число)
* Параметры продукции (два вещественных положительных числа)

Метод возвращает целое число, представляющее количество необходимого материала. Если в качестве параметров метода указаны несуществующие типы продукции, материалов или другие неподходящие данные, метод возвращает значение -1.

Для обеспечения надежности метода был создан отдельный проект модульных тестов. В рамках этого проекта разработано 21 тест:

* 10 тестов низкой сложности: Код этих тестов представлен в [приложении Б](#_Приложение_Б), Таблица 9.
* 11 тестов высокой сложности: Код этих тестов представлен в [приложении Б](#_Приложение_Б), Таблица 10.

Эти тесты позволяют проверить корректность работы метода в различных сценариях, что гарантирует его стабильность и точность.

Результаты тестов представлены на рисунке № 9.

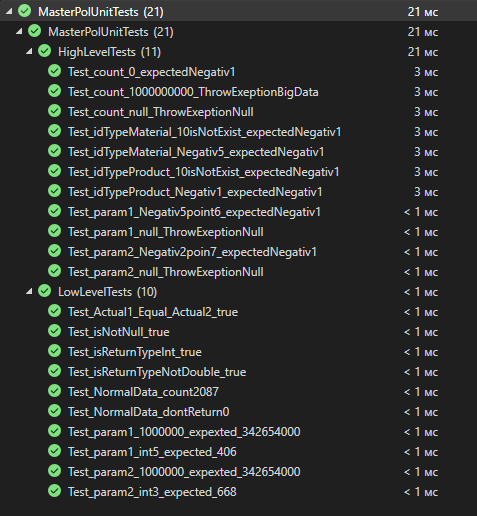


Рисунок № 9 - Результаты прохождения тестов низкого и высокого уровня сложности.

## **Модуль 5. Разработка диаграммы использования и диаграммы деятельности.**

В данном модуле необходимо было разработать 2 диаграммы (диаграмма использования и деятельности), которые представлены на рисунках № 10-11. Эти диаграммы предназначены для информационной системы фитнес центра. Которая автоматизирует процессы распределения фитнес-расписания, контроля его соблюдения и взаимодействия между клиентами, тренерами и администраторами. Система поддерживает три роли пользователей: клиенты, тренеры и администраторы. Авторизация в системе выполняется по телефону и паролю.

Можно выделить следующий основной функционал ИС:

* Для клиентов: регистрация в системе, выбор тренера, отправка заявок с указанием целей тренировок, просмотр плана тренировок, отметка выполнения упражнений с указанием пульса, отслеживание прогресса.
* Для тренеров: просмотр и обработка заявок от клиентов, составление индивидуальных планов тренировок, мониторинг прогресса клиентов, анализ среднего показателя пульса.
* Для администраторов: управление профилями тренеров, запись тренеров на курсы повышения квалификации, предоставление скидок постоянным клиентам.

Система обеспечивает удобное взаимодействие между участниками, автоматизирует процессы планирования и контроля тренировок, а также предоставляет инструменты для анализа эффективности занятий.

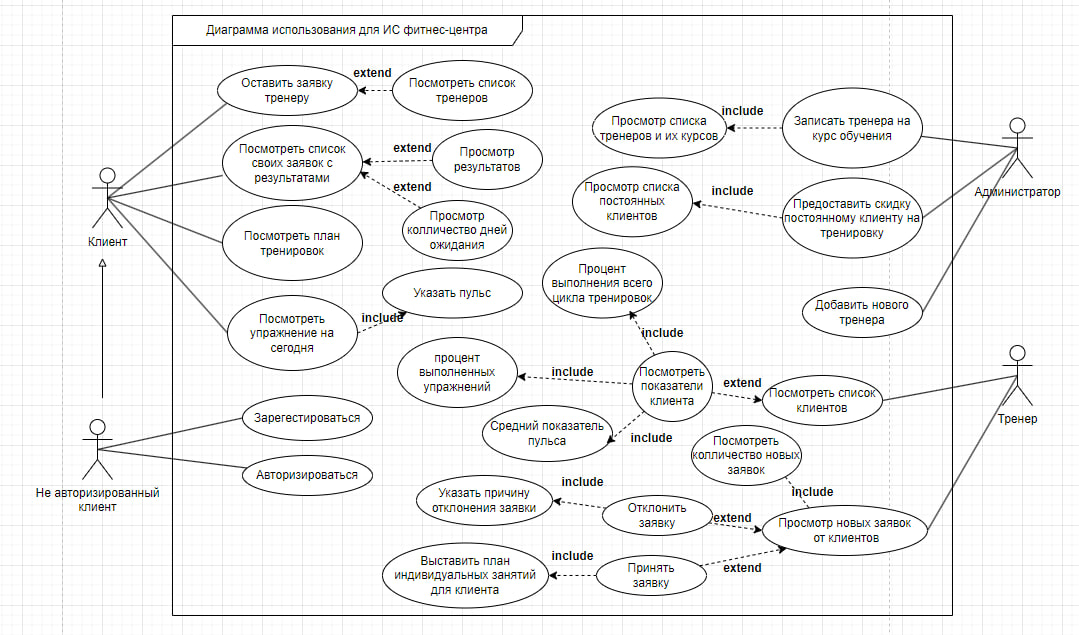


Рисунок № 10 - Диаграмма использования.

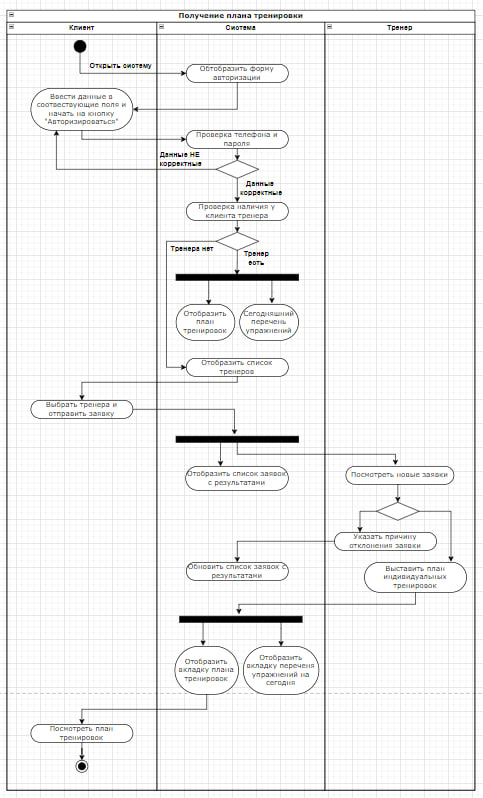


Рисунок № 11 - Диаграмма деятельности.

# **Заключение**

В ходе учебной практики были успешно выполнены задачи, соответствующие четырем сессиям демонстрационного экзамена 2025 года, описанным в документе КОД 09.02.07-2-2025 Том 1, а также составлены 3 диаграммы по представленной предметной области. Каждая сессия была направлена на отработку ключевых навыков в области разработки, администрирования, тестирования и интеграции программного обеспечения, что соответствует модулям:

1. Разработка, администрирование и защита баз данных;
2. Разработка модулей программного обеспечения для компьютерных систем;
3. Сопровождение и обслуживание программного обеспечения компьютерных систем;
4. Осуществление интеграции программных модулей.

В рамках практики были выполнены следующие основные задачи:

* Проведен анализ требований и разработаны программные решения для каждой сессии.
* Разработаны тестовые сценарии для процедуры редактирования партнера, охватывающие различные исходы работы алгоритма.
* Реализован метод для расчета количества материалов в виде библиотеки, а также создан проект модульных тестов, включающий 21 тестовых методов (10 низкой и 11 высокой сложности).
* Составлены диаграммы использования и диаграммы деятельности для информационной системы фитнес-центра.
* Все задачи выполнены в соответствии с требованиями, а результаты оформлены в виде репозитория с подробным описанием в файле README.md и текстового отчета.

Учебная практика позволила закрепить теоретические знания, развить практические навыки и подготовиться к дальнейшей профессиональной деятельности в области информационных технологий.

# **Приложение А**

Таблица № 7 - Скрипт создания БД.

|  |
| --- |
| -- public.employee определение  -- Drop table  -- DROP TABLE public.employee;  **CREATE** **TABLE** public.employee (  id serial4 **NOT** **NULL**,  last\_name **varchar** **NULL**,  first\_name **varchar** **NULL**,  patronymic **varchar** **NULL**,  date\_of\_birth **date** **NULL**,  number\_passport **int4** **NULL**,  address\_rigistration\_passport **varchar** **NULL**,  date\_registration\_passport **date** **NULL**,  address\_residential **varchar** **NULL**,  current\_account\_number\_bank **int4** **NULL**,  correspondent\_account\_number\_bank **int4** **NULL**,  bank\_identification\_code **int4** **NULL**,  full\_name\_bank **varchar** **NULL**,  presence\_of\_family **bool** **NULL**,  health **text** **NULL**,  **CONSTRAINT** employee\_pk **PRIMARY** **KEY** (id)  );  -- public.equipment определение  -- Drop table  -- DROP TABLE public.equipment;  **CREATE** **TABLE** public.equipment (  id serial4 **NOT** **NULL**,  name\_equipment **varchar** **NULL**,  count\_equipment **int4** **NULL**,  cost\_equipment **int4** **NULL**,  **CONSTRAINT** equipment\_pk **PRIMARY** **KEY** (id)  );  -- public.products определение  -- Drop table  -- DROP TABLE public.products;  **CREATE** **TABLE** public.products (  id **int4** **NOT** **NULL**,  article **varchar** **NOT** **NULL**,  name\_product **varchar** **NULL**,  description **text** **NULL**,  image **varchar** **NULL**,  min\_cost\_partner **numeric** **NULL**,  lenth\_package **numeric** **NULL**,  width\_package **numeric** **NULL**,  height\_package **numeric** **NULL**,  weight\_without\_packaging **numeric** **NULL**,  weight\_with\_packaging **numeric** **NULL**,  quality\_certificate **varchar** **NULL**,  standart\_number **varchar** **NULL**,  production\_time **time** **NULL**,  cost\_price **numeric** **NULL**,  workshop\_number **int4** **NULL**,  number\_people\_in\_production **int4** **NULL**,  **CONSTRAINT** products\_pk **PRIMARY** **KEY** (id)  );  -- public.type\_material определение  -- Drop table  -- DROP TABLE public.type\_material;  **CREATE** **TABLE** public.type\_material (  id serial4 **NOT** **NULL**,  name\_type\_material **varchar** **NULL**,  percent\_defect\_materials **numeric** **NULL**,  **CONSTRAINT** material\_history\_pk **PRIMARY** **KEY** (id)  );  -- public.type\_operation определение  -- Drop table  -- DROP TABLE public.type\_operation;  **CREATE** **TABLE** public.type\_operation (  id serial4 **NOT** **NULL**,  name\_type\_operation\_warehouse **varchar** **NULL**,  **CONSTRAINT** type\_operation\_pk **PRIMARY** **KEY** (id)  );  -- public.type\_partner определение  -- Drop table  -- DROP TABLE public.type\_partner;  **CREATE** **TABLE** public.type\_partner (  id serial4 **NOT** **NULL**,  name\_type **varchar** **NULL**,  **CONSTRAINT** type\_partner\_pk **PRIMARY** **KEY** (id)  );  -- public.type\_suppliers определение  -- Drop table  -- DROP TABLE public.type\_suppliers;  **CREATE** **TABLE** public.type\_suppliers (  id serial4 **NOT** **NULL**,  name\_type\_suppliers **varchar** **NULL**,  **CONSTRAINT** type\_suppliers\_pk **PRIMARY** **KEY** (id)  );  -- public.managers определение  -- Drop table  -- DROP TABLE public.managers;  **CREATE** **TABLE** public.managers (  id serial4 **NOT** **NULL**,  id\_employee **int4** **NOT** **NULL**,  name\_role **varchar** **NULL**,  id\_partners **int4** **NOT** **NULL**,  **CONSTRAINT** managers\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** managers\_employee\_fk **FOREIGN** **KEY** (id\_employee) **REFERENCES** public.employee(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.partners определение  -- Drop table  -- DROP TABLE public.partners;  **CREATE** **TABLE** public.partners (  id serial4 **NOT** **NULL**,  name\_company **varchar** **NULL**,  legal\_address **varchar** **NULL**,  tax\_identification\_number **int4** **NULL**,  directors\_full\_name **varchar** **NULL**,  phone **varchar** **NULL**,  email **varchar** **NULL**,  logo **varchar** **NULL**,  rating **int4** **NULL**,  places\_of\_sale **varchar** **NULL**,  id\_type\_partners **int4** **NULL**,  id\_manager **int4** **NOT** **NULL**,  **CONSTRAINT** partners\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** partners\_managers\_fk **FOREIGN** **KEY** (id\_manager) **REFERENCES** public.managers(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** partners\_type\_partner\_fk **FOREIGN** **KEY** (id\_type\_partners) **REFERENCES** public.type\_partner(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.suppliers определение  -- Drop table  -- DROP TABLE public.suppliers;  **CREATE** **TABLE** public.suppliers (  id serial4 **NOT** **NULL**,  id\_supplier\_type **int4** **NOT** **NULL**,  company\_name **varchar** **NULL**,  inn **int4** **NULL**,  **CONSTRAINT** suppliers\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** suppliers\_type\_suppliers\_fk **FOREIGN** **KEY** (id\_supplier\_type) **REFERENCES** public.type\_suppliers(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.workers определение  -- Drop table  -- DROP TABLE public.workers;  **CREATE** **TABLE** public.workers (  id **int4** **DEFAULT** **nextval**('newtable\_1\_id\_seq'::**regclass**) **NOT** **NULL**,  id\_equipment **int4** **NOT** **NULL**,  id\_employee **int4** **NOT** **NULL**,  name\_role **varchar** **NOT** **NULL**,  **CONSTRAINT** newtable\_1\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** newtable\_1\_employee\_fk **FOREIGN** **KEY** (id\_employee) **REFERENCES** public.employee(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** newtable\_1\_equipment\_fk **FOREIGN** **KEY** (id\_equipment) **REFERENCES** public.equipment(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.access\_logs определение  -- Drop table  -- DROP TABLE public.access\_logs;  **CREATE** **TABLE** public.access\_logs (  id serial4 **NOT** **NULL**,  id\_worker **int4** **NOT** **NULL**,  time\_access **date** **NOT** **NULL**,  type\_access\_is\_input **bool** **NOT** **NULL**,  **CONSTRAINT** access\_logs\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** access\_logs\_workers\_fk **FOREIGN** **KEY** (id\_worker) **REFERENCES** public.workers(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.materials определение  -- Drop table  -- DROP TABLE public.materials;  **CREATE** **TABLE** public.materials (  id serial4 **NOT** **NULL**,  id\_type\_material **int4** **NOT** **NULL**,  name\_material **varchar** **NULL**,  id\_supplier **int4** **NOT** **NULL**,  quantity\_package **varchar** **NULL**,  unit\_of\_measurement **varchar** **NULL**,  description **text** **NULL**,  image **varchar** **NULL**,  "cost" **int4** **NULL**,  current\_quantity **int4** **NULL**,  min\_quantity **int4** **NULL**,  **CONSTRAINT** materials\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** materials\_suppliers\_fk **FOREIGN** **KEY** (id\_supplier) **REFERENCES** public.suppliers(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** materials\_type\_material\_fk **FOREIGN** **KEY** (id\_type\_material) **REFERENCES** public.type\_material(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.orders определение  -- Drop table  -- DROP TABLE public.orders;  **CREATE** **TABLE** public.orders (  id serial4 **NOT** **NULL**,  id\_partner **int4** **NOT** **NULL**,  id\_manager **int4** **NOT** **NULL**,  order\_date **date** **NULL**,  total\_cost **int4** **NULL**,  status\_is\_pay **bool** **NULL**,  delivery\_date **date** **NULL**,  order\_status **bool** **NULL**,  **CONSTRAINT** manager\_partners\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** orders\_managers\_fk **FOREIGN** **KEY** (id\_manager) **REFERENCES** public.managers(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** orders\_partners\_fk **FOREIGN** **KEY** (id\_partner) **REFERENCES** public.partners(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.warehouse определение  -- Drop table  -- DROP TABLE public.warehouse;  **CREATE** **TABLE** public.warehouse (  id **int4** **NOT** **NULL**,  id\_material **int4** **NOT** **NULL**,  id\_product **int4** **NOT** **NULL**,  id\_type\_orepation **int4** **NOT** **NULL**,  quantity **int4** **NULL**,  date\_operation **date** **NULL**,  **CONSTRAINT** warehouse\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** warehouse\_materials\_fk **FOREIGN** **KEY** (id\_material) **REFERENCES** public.materials(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** warehouse\_products\_fk **FOREIGN** **KEY** (id\_product) **REFERENCES** public.products(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** warehouse\_type\_operation\_fk **FOREIGN** **KEY** (id\_type\_orepation) **REFERENCES** public.type\_operation(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.order\_products определение  -- Drop table  -- DROP TABLE public.order\_products;  **CREATE** **TABLE** public.order\_products (  id serial4 **NOT** **NULL**,  id\_order **int4** **NOT** **NULL**,  id\_product **int4** **NOT** **NULL**,  **CONSTRAINT** order\_products\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** order\_products\_orders\_fk **FOREIGN** **KEY** (id\_order) **REFERENCES** public.orders(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** order\_products\_products\_fk **FOREIGN** **KEY** (id\_product) **REFERENCES** public.products(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  );  -- public.production определение  -- Drop table  -- DROP TABLE public.production;  **CREATE** **TABLE** public.production (  id serial4 **NOT** **NULL**,  id\_orders **int4** **NOT** **NULL**,  id\_warehouse **int4** **NULL**,  closing\_date **date** **NULL**,  **CONSTRAINT** production\_pk **PRIMARY** **KEY** (id),  **CONSTRAINT** production\_orders\_fk **FOREIGN** **KEY** (id\_orders) **REFERENCES** public.orders(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,  **CONSTRAINT** production\_warehouse\_fk **FOREIGN** **KEY** (id\_warehouse) **REFERENCES** public.warehouse(id) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**  ); |

# **Приложение Б**

Таблица 8 - Код метода MathCountMaterial.

|  |
| --- |
| public class Class1  {  public static int MathCountMaterial(int idTypeProduct, int idTypeMaterial, int count, double param1, double param2 ) {  double ratioTypeProduct =0, percentDefectMaterials =0;  switch (idTypeProduct) {  case 1: ratioTypeProduct = 2.35; break;  case 2: ratioTypeProduct = 5.15; break;  case 3: ratioTypeProduct = 4.34; break;  case 4: ratioTypeProduct = 1.5; break;  default: ratioTypeProduct = -1; break;  }  switch (idTypeMaterial) {  case 1: percentDefectMaterials = 0.1; break;  case 2: percentDefectMaterials = 0.95; break;  case 3: percentDefectMaterials = 0.28; break;  case 4: percentDefectMaterials = 0.55; break;  case 5: percentDefectMaterials = 0.34; break;  default: percentDefectMaterials = -1; break;  }  if (percentDefectMaterials != -1 && ratioTypeProduct != -1 && param1 > 0 && param2 > 0 && count>0)  {  double result = param1 \* param2 \* ratioTypeProduct \* count;  result += result \* percentDefectMaterials;  return Convert.ToInt32(result);  }  else {  return -1;  }  }  } |

Таблица № 9 - Код тестов низкого уровня сложности.

|  |
| --- |
| using System.Diagnostics.Metrics;  using MasterPolLibrary;  namespace MasterPolUnitTests  {  [TestClass]  public sealed class LowLevelTests  {  [TestMethod]  public void Test\_NormalData\_count2087()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = 20;  int expected = 2087;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_NormalData\_dontReturn0()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = 20;  int expected = 0;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreNotEqual(expected, actual);  }  [TestMethod]  public void Test\_param1\_1000000\_expexted\_342654000()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 1000000;  double param2 = 2.7;  int count = 20;  int expected = 372654000;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_param2\_1000000\_expexted\_342654000()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 2.7;  double param2 = 1000000;  int count = 20;  int expected = 372654000;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_Actual1\_Equal\_Actual2\_true()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 2.7;  double param2 = 5.2;  int count = 20;    int actual1 = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  int actual2 = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param2, param1);  Assert.IsTrue(actual1 == actual2);  }  [TestMethod]  public void Test\_param1\_int5\_expected\_406()  {  int idTypeProduct = 1;  int idTypeMaterial = 3;  int param1 = 5;  double param2 = 2.7;  int count = 10;  int expected = 406;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_param2\_int3\_expected\_668()  {  int idTypeProduct = 1;  int idTypeMaterial = 3;  double param1 = 7.4;  int param2 = 3;  int count = 10;  int expected = 668;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_isNotNull\_true()  {  int idTypeProduct = 1;  int idTypeMaterial = 3;  double param1 = 7.4;  int param2 = 3;  int count = 10;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.IsNotNull(actual);  }  [TestMethod]  public void Test\_isReturnTypeInt\_true()  {  int idTypeProduct = 1;  int idTypeMaterial = 3;  double param1 = 7.4;  int param2 = 3;  int count = 10;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.IsInstanceOfType(actual,typeof(int));  }  [TestMethod]  public void Test\_isReturnTypeNotDouble\_true()  {  int idTypeProduct = 1;  int idTypeMaterial = 3;  double param1 = 7.4;  int param2 = 3;  int count = 10;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.IsNotInstanceOfType(actual, typeof(double));  }  }  } |

Таблица 10 - Код тестов высокого уровня сложности

|  |
| --- |
| using MasterPolLibrary;  namespace MasterPolUnitTests;  [TestClass]  public class HighLevelTests  {    [TestMethod]  public void Test\_count\_1000000000\_ThrowExeptionBigData()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = 1000000000;  Assert.ThrowsException<AssertFailedException>(() => Assert.ThrowsException<SystemException>(() => Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2)));  }  [TestMethod]  public void Test\_count\_null\_ThrowExeptionNull()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = Convert.ToInt32(null);  Assert.ThrowsException<AssertFailedException>(() => Assert.ThrowsException<SystemException>(() => Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2)));  }  [TestMethod]  public void Test\_param1\_null\_ThrowExeptionNull()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = Convert.ToDouble( null);  double param2 = 2.7;  int count = 10;  Assert.ThrowsException<AssertFailedException>(() => Assert.ThrowsException<SystemException>(() => Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2)));  }  [TestMethod]  public void Test\_param2\_null\_ThrowExeptionNull()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 2.7;  double param2 = Convert.ToDouble(null);  int count = 10;  Assert.ThrowsException<AssertFailedException>(() => Assert.ThrowsException<SystemException>(() => Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2)));  }  [TestMethod]  public void Test\_count\_0\_expectedNegativ1()  {  int idTypeProduct = 2;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = 0;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_idTypeProduct\_Negativ1\_expectedNegativ1()  {  int idTypeProduct = -1;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = 10;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_idTypeMaterial\_Negativ5\_expectedNegativ1()  {  int idTypeProduct = 1;  int idTypeMaterial = -5;  double param1 = 5.6;  double param2 = 2.7;  int count = 10;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_param1\_Negativ5point6\_expectedNegativ1()  {  int idTypeProduct = 1;  int idTypeMaterial = 5;  double param1 = -5.6;  double param2 = 2.7;  int count = 10;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_param2\_Negativ2poin7\_expectedNegativ1()  {  int idTypeProduct = 1;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = -2.7;  int count = 10;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_idTypeProduct\_10isNotExist\_expectedNegativ1()  {  int idTypeProduct = 10;  int idTypeMaterial = 5;  double param1 = 5.6;  double param2 = 2.7;  int count = 10;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  [TestMethod]  public void Test\_idTypeMaterial\_10isNotExist\_expectedNegativ1()  {  int idTypeProduct = 1;  int idTypeMaterial = 10;  double param1 = 5.6;  double param2 = 2.7;  int count = 10;  int expected = -1;  int actual = Class1.MathCountMaterial(idTypeProduct, idTypeMaterial, count, param1, param2);  Assert.AreEqual(expected, actual);  }  } |