**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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Проваленко Л. Н. 21.03.2025г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

Оглавление

[**Введение** 3](#_Toc192200860)

[**Предметная область и задание** 4](#_Toc192200861)

[**Разработка базы данных** 4](#_Toc192200862)

[**Разработка desktop приложения** 8](#_Toc192200863)

[**Разработка DLL библиотеки и её тестирование** 17](#_Toc192200864)

[**Ручное тестирование редактирования партнёра** 20](#_Toc192200865)

[**UML диаграммы** 24](#_Toc192200866)

[**Заключение** 25](#_Toc192200867)

[**Приложение А.** 26](#_Toc192200868)

[**Приложение Б.** 27](#_Toc192200869)

# **Введение**

Целью учебной практики являлось приобретение практических навыков и умений в области разработки программных модулей и тестирования. Актуальность в настоящее время очень высока, поскольку разработка программных модулей является одним из ключевых инструментов в решении сложных задач в различных областях науки, техники, экономики и бизнеса.

Разработка программных модулей позволяет эффективно создавать и интегрировать сложные системы и процессы, прогнозировать их поведение, оптимизировать параметры и принимать решения на основе полученных результатов. В настоящее время разработка программных модулей применяется в таких областях, как физика, биология, медицина, экономика, финансы, логистика, инженерия, экология, транспорт, образование и многие другие.

Для достижения цели поставлены следующие задачи:

1. Разработка базы данных по предметной области.
2. Преобразование данных в нужный вид с помощью excel.
3. Разработка приложения с помощью Avalonia MVVM.
4. Разработка DLL библиотеки и её тестирование.
5. Разработка ручных тестов.
6. Создание UML диаграмма по заданию на проектирование.

Практическим результатом прохождения практики является:

1. Заполненная база данных.
2. Рабочее desktop приложение.
3. Рабочая DLL библиотека и тесты, подходящие к ней.
4. Ручные тесты на редактирование.
5. Разработанные UML диаграммы.
6. Отчёт по пройденной части практики.

# **Предметная область и задание**

Компания занимается производством и реализует свою продукцию через партнеров, которые доставляют продукцию компании до конечных потребителей. Для эффективного взаимодействия с партнерами и контроля их работы требуется система, позволяющая обрабатывать всю информацию в цифровом формате. Разработать подсистему для работы с партнерами компании, обеспечивающую следующий функционал: просмотр списка партнеров; добавление/редактирование данных о партнере; просмотр истории реализации продукции партнером.

# **Разработка базы данных**

Для базы данных были предоставлены данные для импорта и документ с предметной областью, в которой описаны все сущности. Для разработки приложения не понадобятся все сущности, поэтому были реализованы только те, что фигурируют в задании и имеют данные в файлах импорта.

Таким образом в итоге выходит 6 таблиц:

* Партнёр
* Тип партнера
* Тип продукции
* Продукция
* Тип материала
* Продукты партнёров

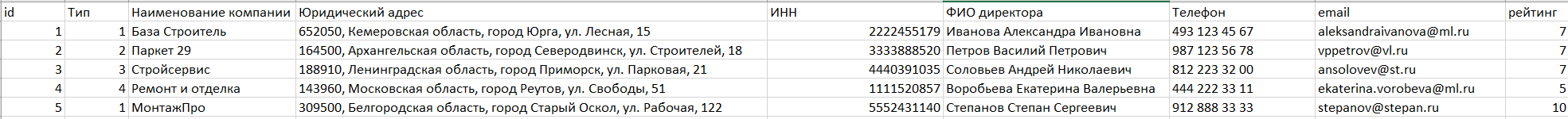


Рисунок №1 – Данные для импорта таблицы партнёров.

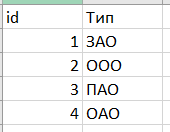


Рисунок №2 – Данные для импорта таблицы типов партнёров.

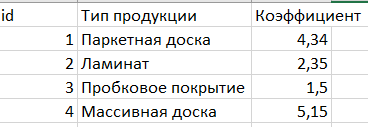


Рисунок №3 – Данные для импорта таблицы типов продукции.

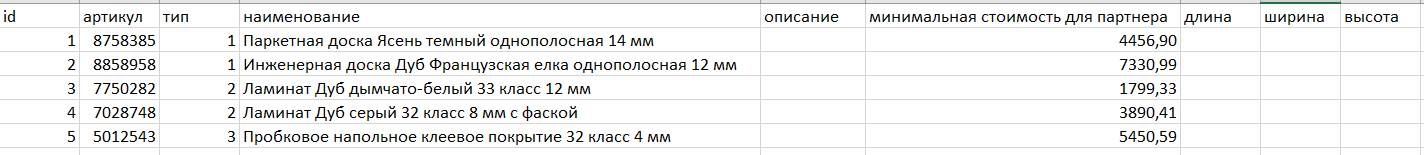


Рисунок №4 – Данные для импорта таблицы продукции.

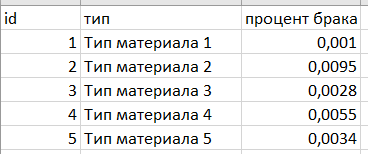


Рисунок №5 – Данные для импорта таблицы типов материала.



Рисунок №6 – Данные для импорта таблицы продукция партнеров.

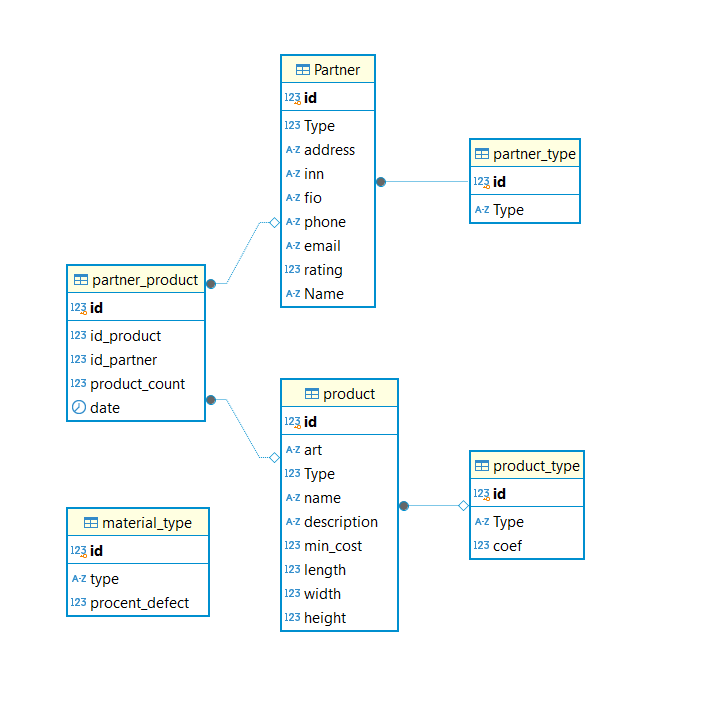


Рисунок №7 – Er-диаграмма базы данных.

# **Разработка desktop приложения**

Приложение разрабатывалось в Visual Studio с использованием Avalonia и архитектуры MVVM (Model-View-ViewModel), а также библиотеки ReactiveUI.

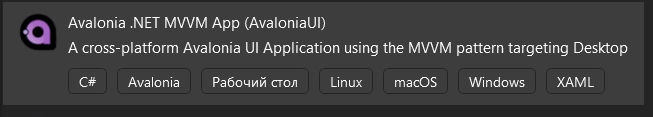


Рисунок №8 – Создание проекта.

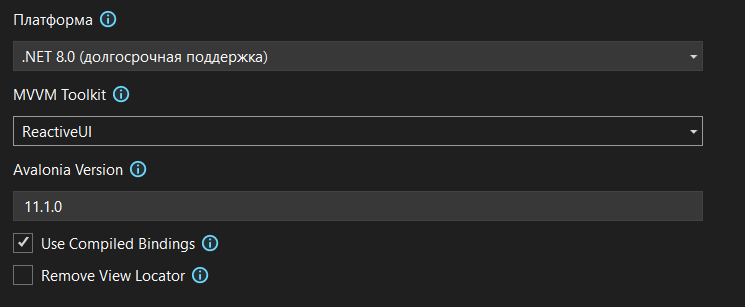


Рисунок №9 – Выбор библиотеки ReactiveUI.

Для начала в коде класса MainWindowViewModel была создана публичная статичная переменная Instance для того, чтобы с любой другой страницы можно было обращаться к свойству типа UserControl для перехода на другие страницы приложения.

Публичная переменная connection позволяет обращаться с любой страницы приложения к нашему соединению с базой данной.

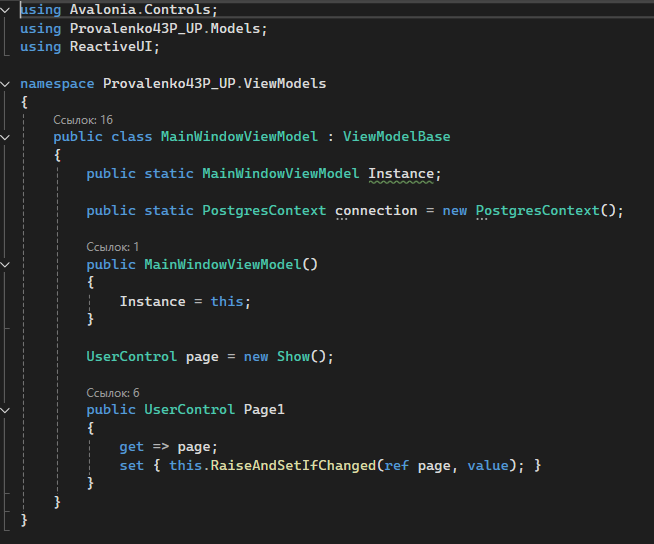


Рисунок №10 – код MainWindowViewModel.

На странице MainWindow был создан СontentСontrol для смены страниц в Avalonia.

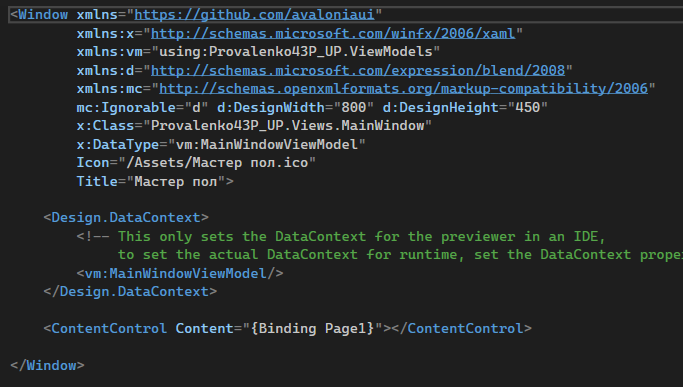


Рисунок №11 – код MainWindow.

Дальше была разработана страница, где можно просмотреть всех партнёров, чьи данные находятся в соответствующей таблице.

Создав объект ViewModel, в свойство типа таблицы партнеров была передана сама таблица, преобразованная в лист. Для того, чтобы на странице отображались партнёры используется DataContex.

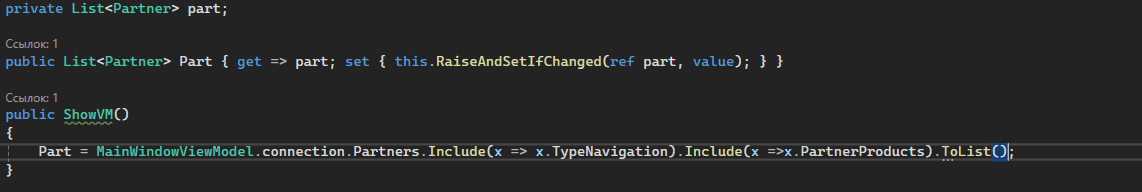


Рисунок №11 – код ShowViewModel.

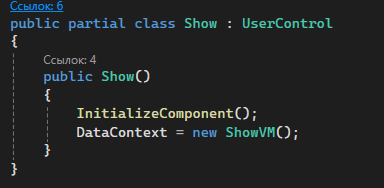


Рисунок №12 – привязка DataContex.

Также было написано 3 метода для переходов на страницы истории реализации продукции, редактирования и добавления партнёра. Сообщения нужны для подтверждения действий пользователя.

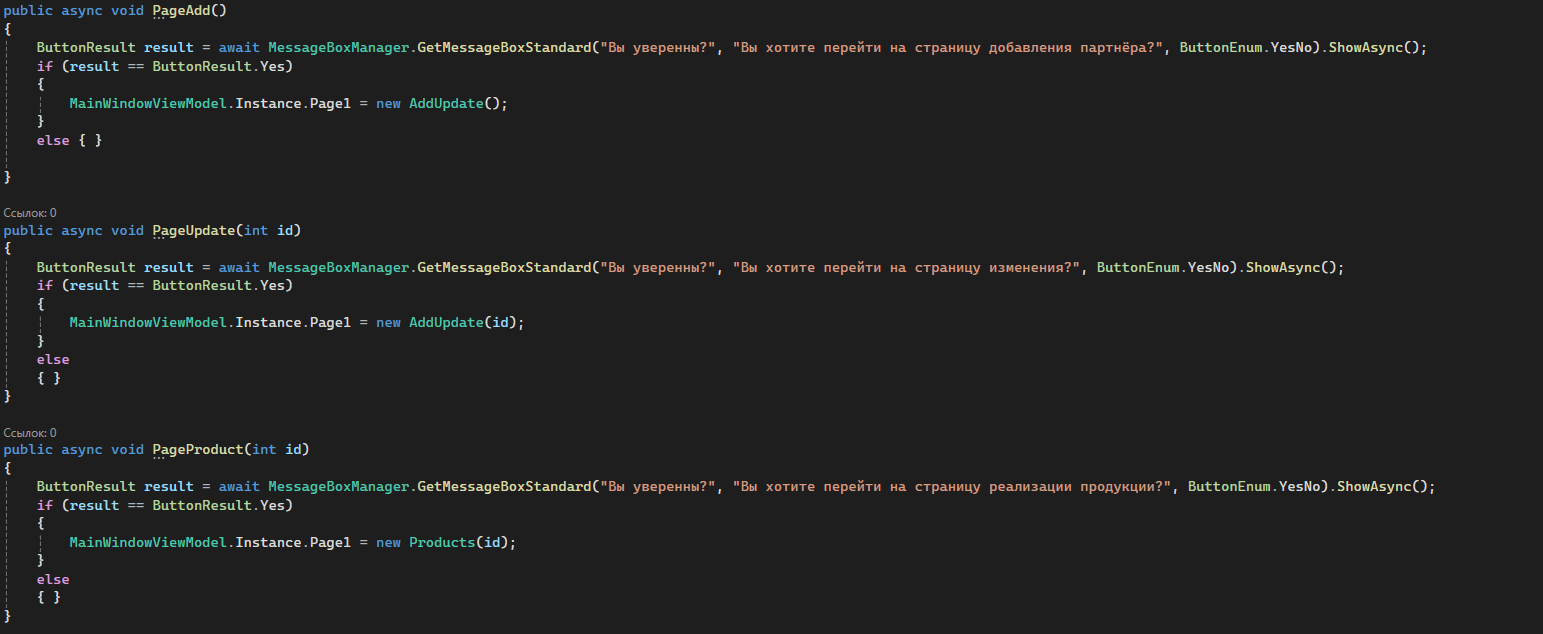


Рисунок №13 – методы для переходов на новые страницы.

Для подсчёта скидки партнёра было реализовано свойство с помощью которого считается количество реализованной продукции партнёра в таблица продукции партнёров.

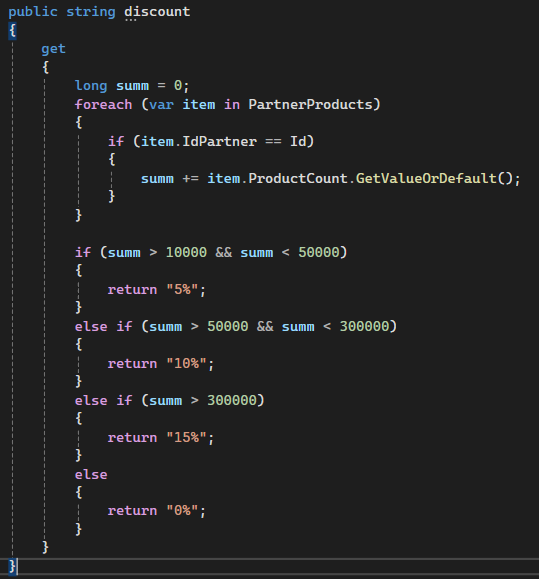


Рисунок №13 – свойство скидки.

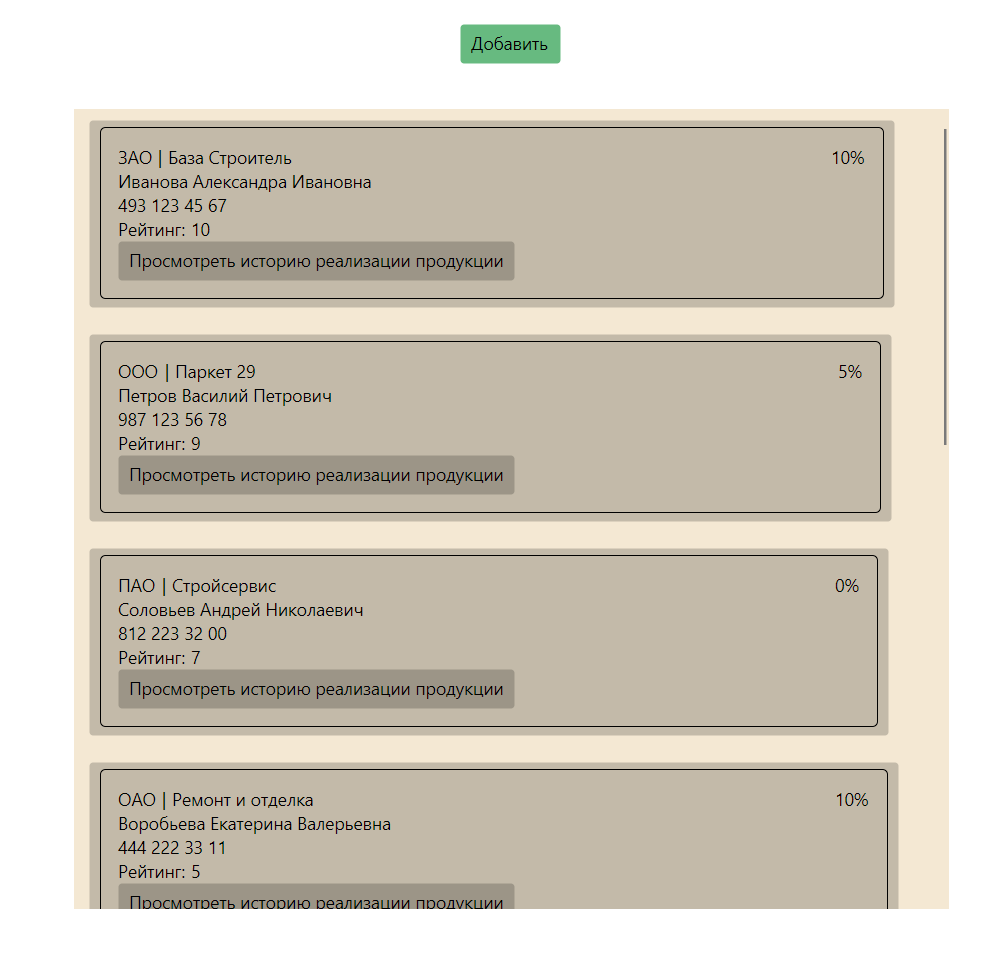


Рисунок №14 – Главная страница приложения.

Страница добавления и редактирования – это одна страница, но для редактирования передаётся id партнёра через командный параметр.



Рисунок №15 – Команда для передачи id партнёра.

В ViewModel добавления/редактирования основным является свойство типа партнёра, которое записывает данные введённые в TextBox и после преобразует их в одну общую запись для ввода в базу данных или же при редактировании возвращает те данные, которые находятся под id выбранного партнёра. Для выбора типа партнёра реализован ComboBox, поэтому за его наполнение отвечает свойство, преобразующее типы партнёров в лист.

Большое количество свойств отвечает за изменение названия страницы, сообщений в messageBox и названия кнопки.

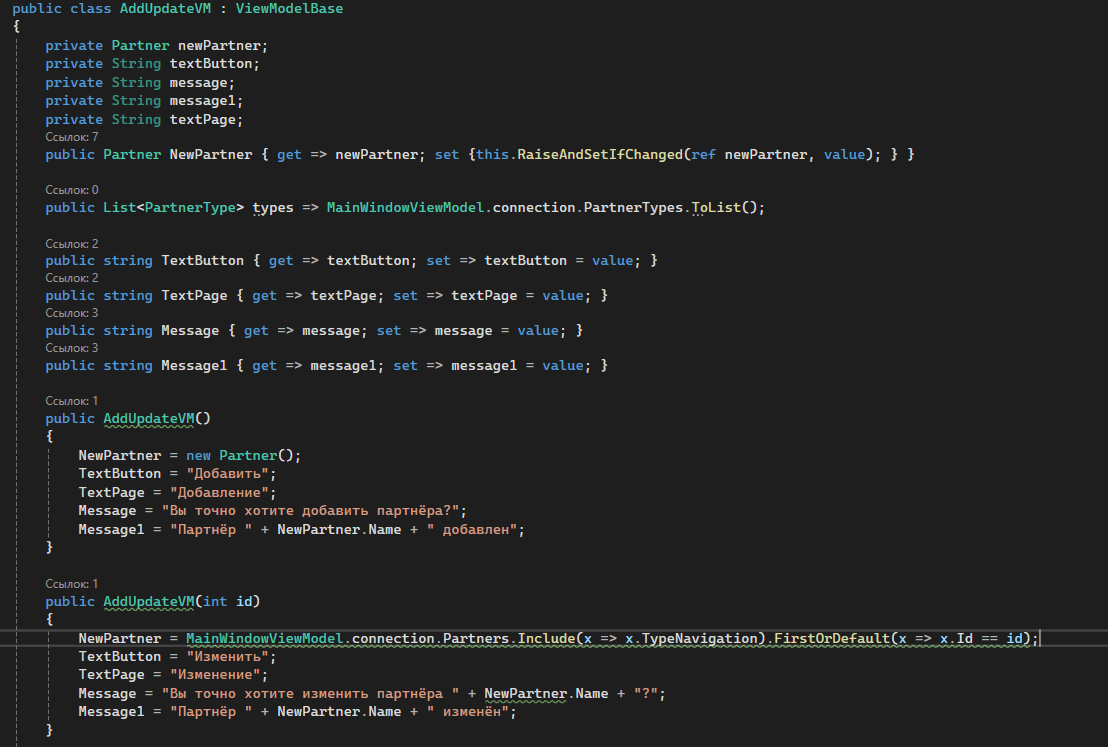


Рисунок №16 – Код страницы добавления/редактирования.

Также реализовано для метода. Первый метод отвечает за возврат на главную страницу по кнопке “Назад” при подтверждении от пользователя. Второй метод отвечает за добавление или редактирование в базу данных при условии, что рейтинг не отрицательный.

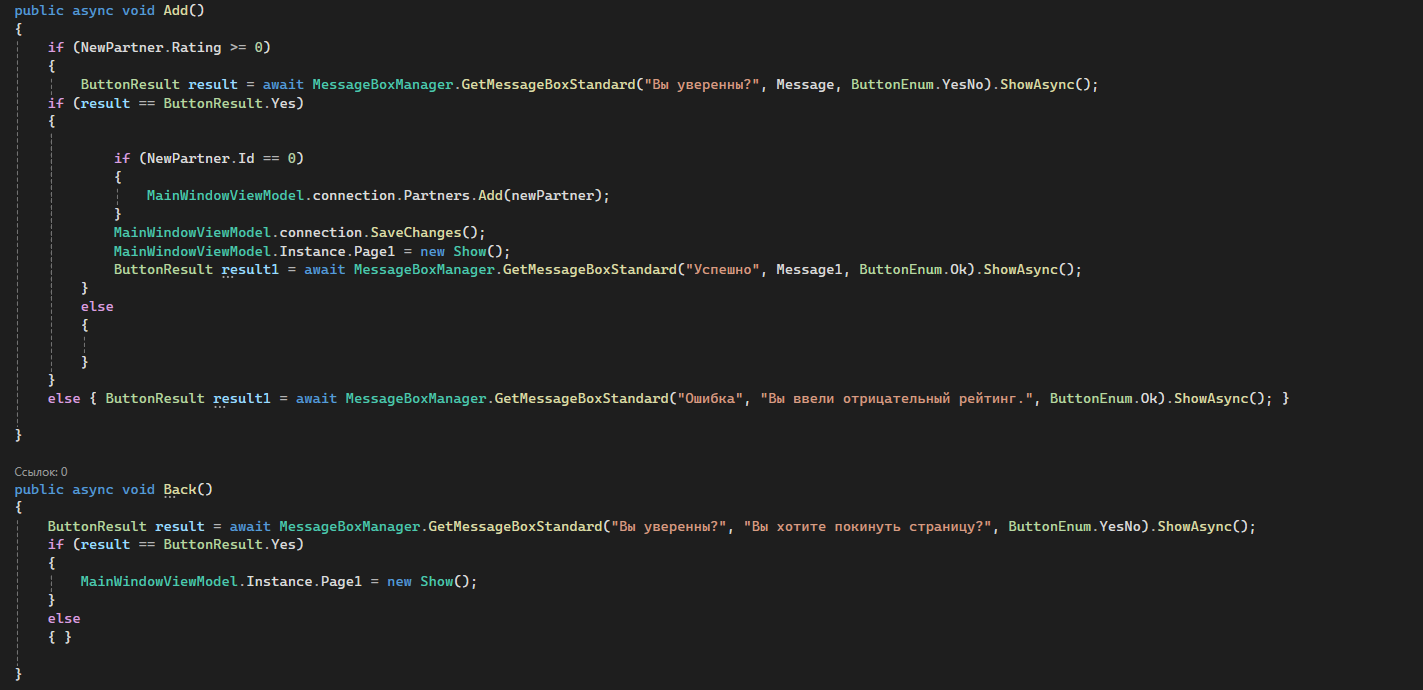


Рисунок №17 – Метод возврата на главную страницу и метод добавления/редактирования.

Для правильных отображений данных или же наоборот их записи есть для DataContext.

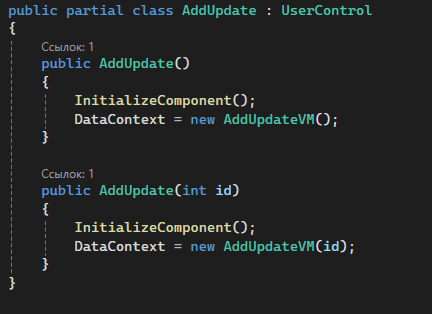


Рисунок №18 – DataContext страницы добавления/редактирования.

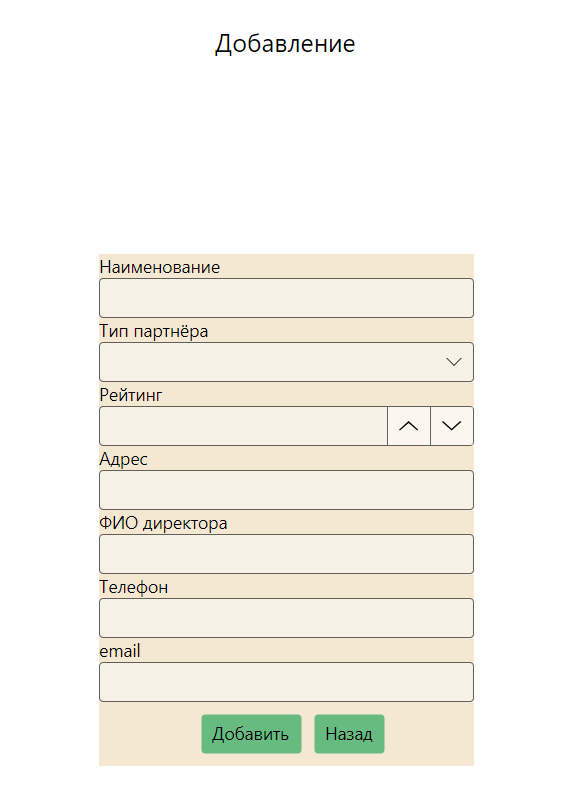


Рисунок №19 – Страница добавления партнёра.

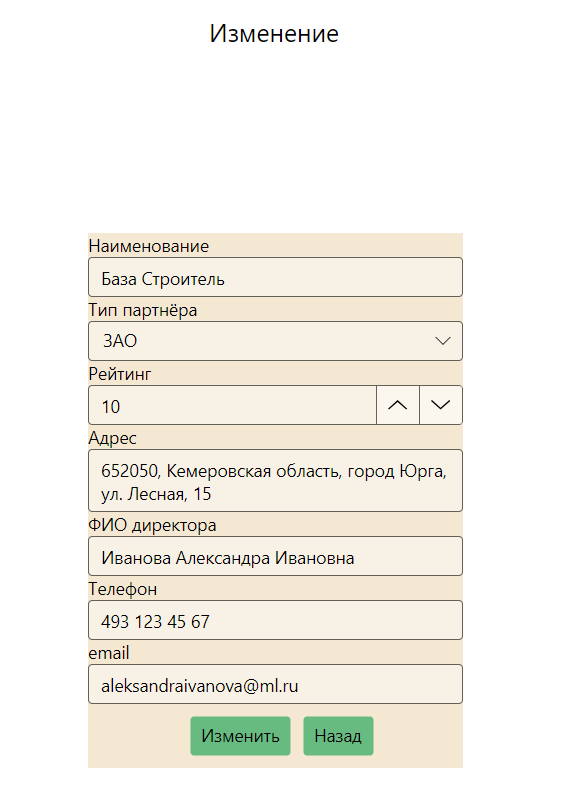


Рисунок №20 – Страница редактирования партнёра.

Последнее, что было реализовано в приложении была страница реализации продукции от конкретного партнёра.

Её создание не отличается от страницы, где были показаны все партнёры за исключением выбора показанных данных. Для этого в обеъекте ViewModel используется поиск по переданному id в таблице продукции от партнёров с помощью Where в запросе.

Было добавлено отображение наименования партнёра сверху, поэтому есть дополнительное свойство, в которое передаются данные партнёры по переданному id.

Также здесь реализован метод возврата на главную страницу.



Рисунок №21 –Код ViewModel реализации продукции партнёра.

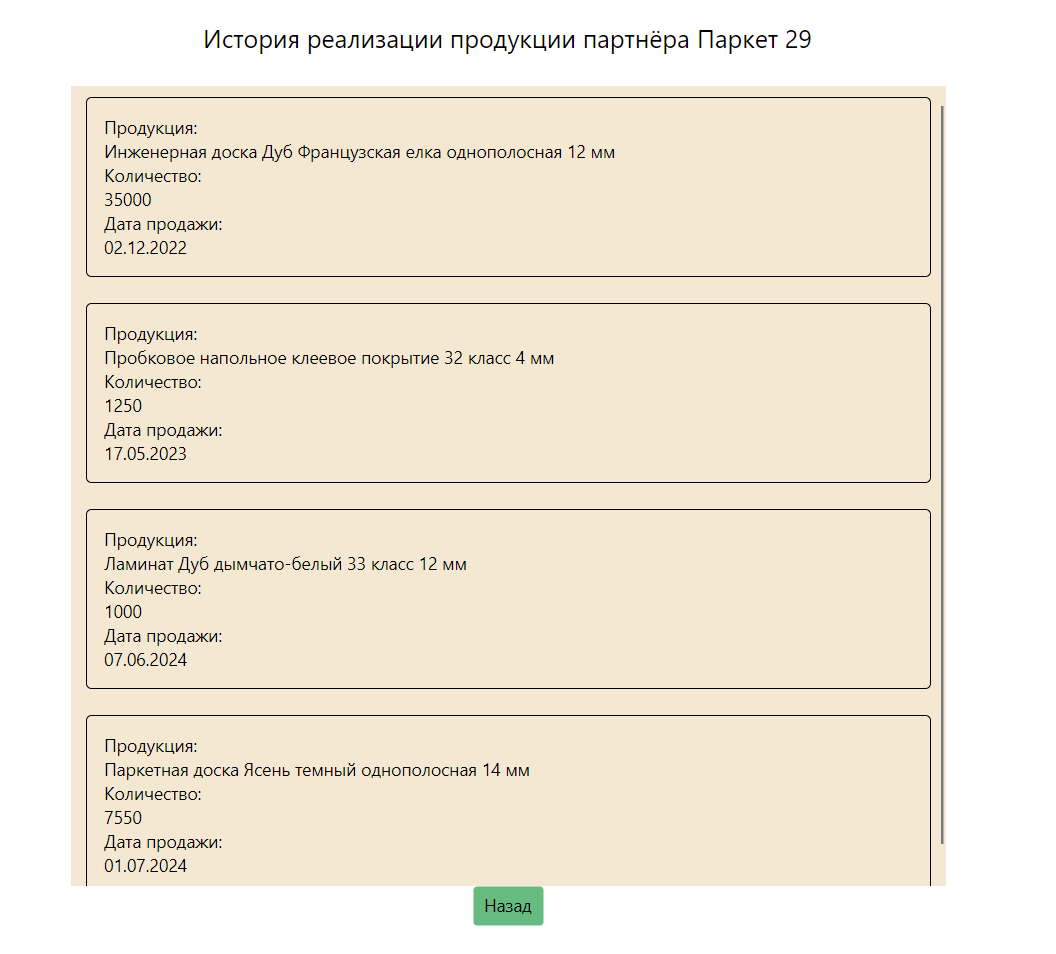


Рисунок №22 –страница реализации продукции партнёра.

Также в приложении была изменена иконка и само название.



Рисунок №22 –иконка и название приложения.

Был создан файл стилей для изменения шрифта во всём приложении.

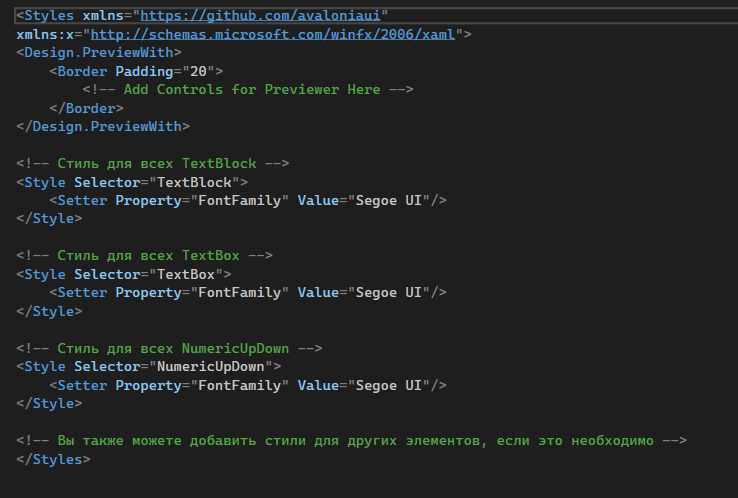


Рисунок №23 –файл стилей.

# **Разработка DLL библиотеки и её тестирование**

Для разработки библиотеки был создан новый проект.

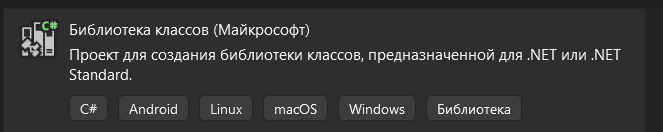


Рисунок №24 –создание проекта DLL библиотеки.

Метод, который находится в библиотеке рассчитывает количество необходимого материала для производства продукции, исходя из её количества, размера материала, коэффициента продукции и процента брака материала.

Данные были взяты из таблицы продукции и типа материалов. На рисунке 25 показан один из вариантов правильного ввода данных, где количество продукции, размер материала положительные, а тип продукции и тип материала существуют.

Для расчёта сначала находится количество материала без учёта брака, где число – коэффициент продукции, а после количество материала уже с учётом брака.

Возвращается округленное число в большую сторону.

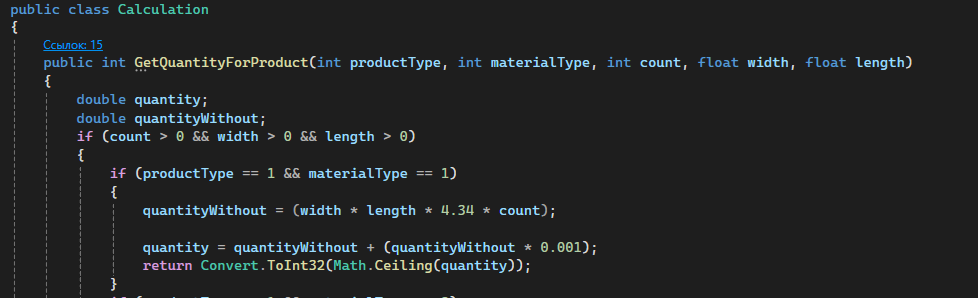


Рисунок №25 –часть DLL библиотеки.

Для тестирования был создан проект тестирования.

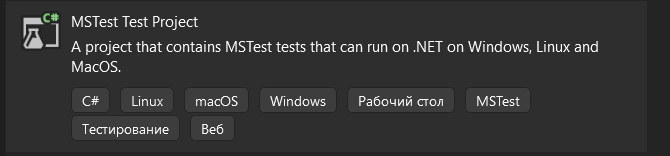


Рисунок №25 –создание проекта тестирования

Один из простых тестов проверяет, как поведёт себя метод при вводе несуществующего типа продукции.

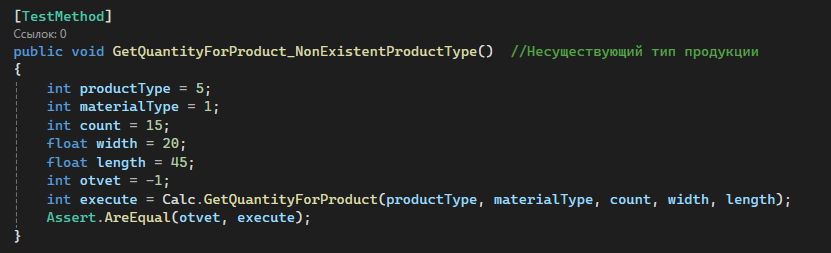


Рисунок №26 –Тест на несуществующий тип продукции.

Один из сложных тестов проверяет исключение при нулевой ширине. Данный тест не будет пройден, так как по условию задания при всех неверных данных методов возвращает -1.

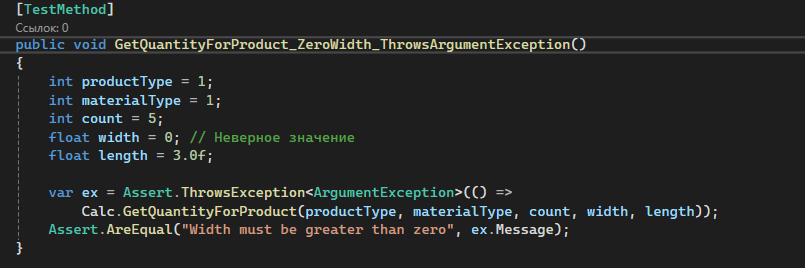


Рисунок №27 – Тест на исключение нулевой ширины.

Всего тестов 15. Каждый простой тест проверяет поведение метода при вводе различных данных, а каждый сложный проверяет возникшие исключения.

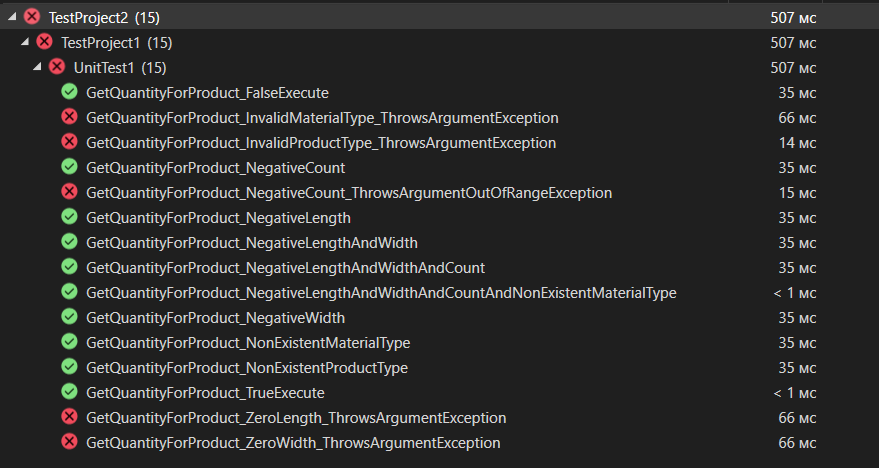


Рисунок №28 – Результаты тестирования библиотеки.

# **Ручное тестирование редактирования партнёра**

Таблица №1 – аннотация теста.

|  |  |
| --- | --- |
| **Название проекта** | Provalenko43P\_UP |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Проваленко Л.Н |
| **Дата(ы) теста** | 04.03.2025 |

Таблица №2 – тест правильного изменения.

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_Change\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Изменение партнёра с правильными данными |
| **Краткое изложение теста** | Проверка изменения партнёра с правильными данными |
| **Этапы теста** | 1. Заменить поле наименование партнёра на любое другое. 2. Нажать кнопку изменить 3. Подтвердить изменения |
| **Тестовые данные** | Наименование: "Партнёр 6"  Тип партнёра: ЗАО  Рейтинг: 5  Адрес: "Адрес 1"  ФИО директора: "Директор 1"  Телефон: "1234567890"  Email: "partner1@example.com" |
| **Ожидаемый результат** | Данные партнёра обновятся и изменятся на главной странице. |
| **Фактический результат** | Данные партнёра обновились и изменились на главной странице. |
| **Статус** | Пройден |
| **Предварительное условие** | Пользователь находится на странице изменения и имеет подключенную базу данных. |
| **Постусловие** | Данные партнёра обновляются в базе данных. |
| **Примечания/комментарии** | Проверить, что данные партнёра корректно сохранены. |

Таблица №2 – тест изменения с отрицательным рейтингом.

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_ Change \_2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Попытка обновления партнёра с отрицательным рейтингом |
| **Краткое изложение теста** | Проверка обработки ошибки при обновлении партнёра с отрицательным рейтингом. |
| **Этапы теста** | 1.Установить рейтинг партнёра в -1.  2.Нажать кнопку изменить. |
| **Тестовые данные** | Наименование: "Партнёр 6"  Тип партнёра: ЗАО  Рейтинг: -1  Адрес: "Адрес 1"  ФИО директора: "Директор 1"  Телефон: "1234567890"  Email: "partner2@example.com" |
| **Ожидаемый результат** | Появится сообщение с предупреждением. |
| **Фактический результат** |  |
| **Статус** | Пройден |
| **Предварительное условие** | Пользователь находится на странице изменения и имеет подключенную базу данных. |
| **Постусловие** | Партнёр не обновлен в базе данных. |
| **Примечания/комментарии** | Убедиться, что данные не сохраняются. |

Таблица №3 – тест отмены изменения партнёра.

|  |  |
| --- | --- |
| Тестовый пример **#** | TC\_Change\_3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Отмена изменения партнёра |
| **Краткое изложение теста** | Проверка, что изменение партнёра отменяется при выборе "No" в диалоговом окне. |
| **Этапы теста** | 1.Заполнить поля для ввода верными значениями  2.Нажать кнопку изменить  3.Отменить изменение |
| **Тестовые данные** | Наименование: "Партнёр 6"  Тип партнёра: ЗАО  Рейтинг: 1  Адрес: "Адрес 1"  ФИО директора: "Директор 4"  Телефон: "1234567890"  Email: "partner2@example.com" |
| **Ожидаемый результат** | Партнёр не обновлён. Пользователь остаётся на странице изменения. |
| **Фактический результат** | Партнёр не обновлён. Пользователь остаётся на странице изменения. |
| **Статус** | Пройден |
| **Предварительное условие** | Пользователь находится на странице изменения и имеет подключенную базу данных. |
| **Постусловие** | Партнёр не обновлен в базе данных. |
| **Примечания/комментарии** | Проверить, что данные не сохраняются. |

Таблица №4 – тест проверки совпадения данных.

|  |  |
| --- | --- |
| Тестовый пример **#** | TC\_Change\_4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка совпадения данных |
| **Краткое изложение теста** | Проверка, что данные на странице редактирования совпадают с данными в базе данных |
| **Этапы теста** | 1.Открыть в базе данных таблицу с партнёрами  2.Выбрать на странице всех партнёров конкретного партнёра для редактирования.  3.Подтвердить переход на страницу редактирования.  4.Сверить данные из таблицы с данными на странице. |
| **Тестовые данные** | Отсутствуют |
| **Ожидаемый результат** | Данные совпадают |
| **Фактический результат** | Данные совпадают |
| **Статус** | Пройден |
| **Предварительное условие** | Наличие доступа к базе данных и подключенная база данных в проект |
| **Постусловие** |  |
| **Примечания/комментарии** |  |

Таблица №5 – тест ухода со страницы изменения.

|  |  |
| --- | --- |
| **Test Case #** | TC\_Change\_5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Уход со страницы изменения |
| **Краткое изложение теста** | Проверка, что при уходе со страницы изменения данные не обновляются в базе данных. |
| **Этапы теста** | 1.Изменить любые данные партнёра  2.Нажать кнопку “Назад”.  3.Подтвердить переход. |
| **Тестовые данные** | Отсутствуют. |
| **Ожидаемый результат** | Пользователь вернётся на главную страницу, где ни один партнёр не изменился. |
| **Фактический результат** | Пользователь вернулся на главную страницу, где ни один партнёр не изменился. |
| **Статус** | Пройден. |
| **Предварительное условие** | Наличие доступа к базе данных и подключенная база данных в проект. |
| **Постусловие** | Партнёр не обновлен в базе данных. |
| **Примечания/комментарии** | Проверить, что данные не сохраняются. |

# **UML диаграммы**

Диаграмма вариантов использования наглядно показывает какие действия в системе фитнес центра могут делать различные пользователи: клиент, тренер и администратор. Диаграмма находится в приложении А.

Клиент:

* Регистрация и авторизация.
* Выбор тренера и отправка заявки.
* Просмотр плана тренировок и выполнение упражнений.
* Отметка выполненных упражнений и ввод показателя пульса.
* Просмотр прогресса.
* Просмотр своих заявок.

Тренер:

* Авторизация и просмотр заявок от клиентов.
* Принятие или отклонение заявок.
* Создание плана тренировок для клиентов.
* Просмотр прогресса клиентов.

Администратор:

* Авторизация.
* Управление тренерами и их обучением.
* Предоставление скидок клиентам.

Диаграмма деятельности наглядно показывает взаимодействия клиента и тренера в системе. Диаграмма находится в приложении Б.

1. Клиент отправляет заявку тренеру.
2. Тренер просматривает и обрабатывает заявку.
3. Тренер создает план тренировок.
4. Клиент выполняет упражнения и отмечает их.
5. Тренер просматривает прогресс клиента.

# **Заключение**

В ходе учебной практики были приобретены практические навыки и умения в области разработки программных модулей и тестирования. Актуальность данной практики подтверждается тем, что разработка программных модулей является ключевым инструментом для решения сложных задач в различных областях науки, техники, экономики и бизнеса.

В рамках практики были успешно выполнены следующие задачи:

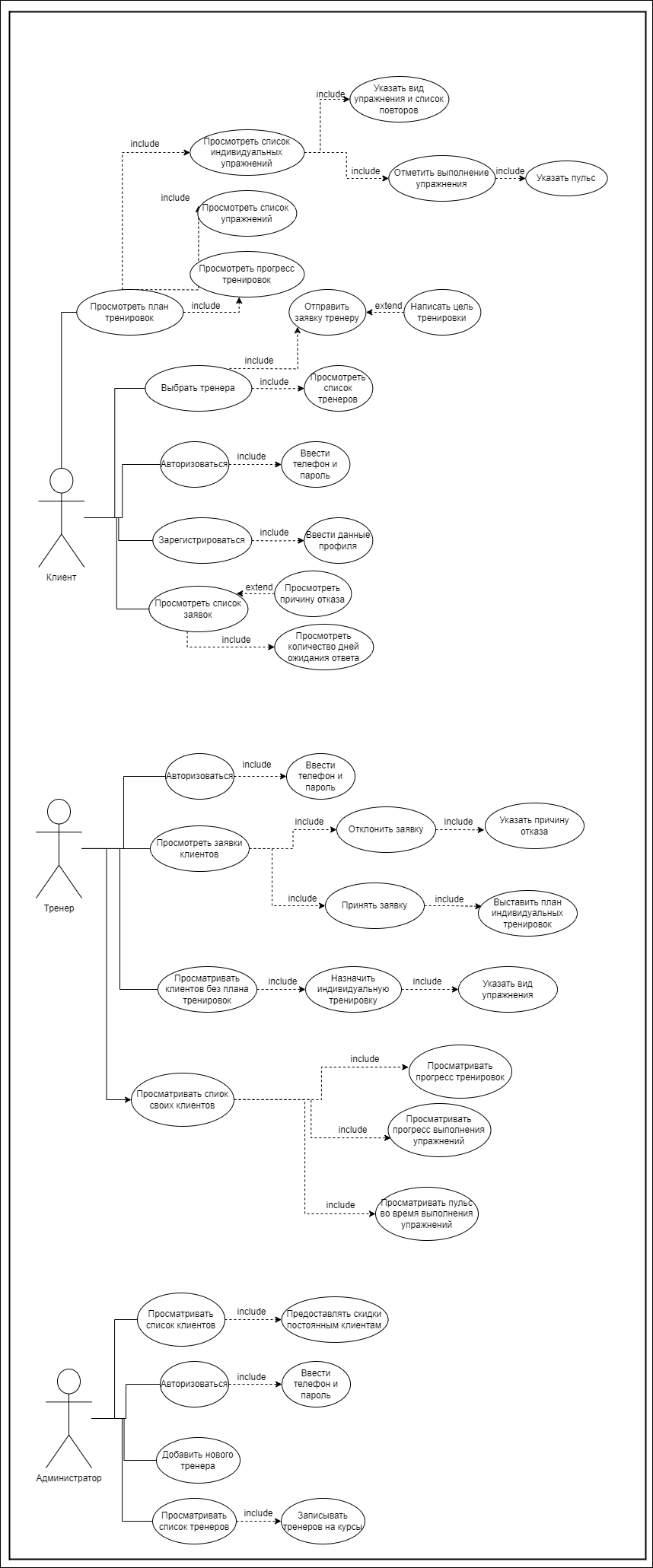
1. Разработана база данных по предметной области.
2. Произведено преобразование данных в нужный вид с помощью Excel.
3. Создано рабочее desktop-приложение с использованием Avalonia MVVM.
4. Разработана и протестирована DLL-библиотека.
5. Созданы ручные тесты для проверки функциональности.
6. Разработаны UML-диаграммы для проектирования системы.

Практическими результатами прохождения практики стали:

1. Заполненная база данных.
2. Рабочее desktop-приложение.
3. Рабочая DLL-библиотека и соответствующие тесты.
4. Ручные тесты для редактирования.
5. Разработанные UML-диаграммы.
6. Отчёт по пройденной части практики.

Таким образом, учебная практика способствовала углублению знаний и навыков в области разработки программных модулей, что является важным шагом на пути к профессиональному росту и успешной карьере в сфере информационных технологий.

# **Приложение А.**



# **Приложение Б.**

